

---

**STM8 SWIM communication protocol and debug module**

---

**Introduction**

This manual is addressed to developers who build programming, testing or debugging tools for the STM8 8-bit MCUs family. This document explains the debug architecture of the STM8 core.

The STM8 8-bit MCUs debug system includes two modules:

- DM: debug module
- SWIM: single wire interface module

Related documentation:

*How to program STM8S and STM8A Flash program memory and data EEPROM (PM0051)*

*How to program STM8L and STM8AL Flash program memory and data EEPROM (PM0054).*

# Contents

<b>1</b>	<b>Debug system overview</b>	<b>6</b>
<b>2</b>	<b>Communication layer</b>	<b>7</b>
<b>3</b>	<b>Single wire interface module (SWIM)</b>	<b>8</b>
3.1	Operating modes	8
3.2	SWIM entry sequence	9
3.3	Bit format	11
3.3.1	High speed bit format	11
3.3.2	Low speed bit format	12
3.4	SWIM communication protocol	13
3.5	SWIM commands	14
3.5.1	SRST: system reset	14
3.5.2	ROTF: read on-the-fly	14
3.5.3	WOTF: write on-the-fly	15
3.6	SWIM communication reset	15
3.7	CPU register access	16
3.8	SWIM communication in Halt mode	16
3.9	Physical layer	17
3.10	STM8 MCUs SWIM registers	18
3.10.1	SWIM control status register (SWIM_CSR)	18
3.10.2	SWIM clock control register (CLK_SWIMCCR)	19
<b>4</b>	<b>Debug module (DM)</b>	<b>20</b>
4.1	Introduction	20
4.2	Main features	20
4.3	Debug	22
4.3.1	Reset	22
4.3.2	Breakpoints	22
4.3.3	Abort	22
4.3.4	Watchdog control	22
4.3.5	Interaction with SWIM	22
4.4	Breakpoint decoding table	23

4.5	Software breakpoint mode .....	24
4.6	Timing description .....	24
4.7	Abort .....	24
4.8	Data breakpoint .....	25
4.9	Instruction breakpoint .....	25
4.10	Step mode .....	25
4.11	Application notes .....	27
4.11.1	Illegal memory access .....	27
4.11.2	Forbidden stack access .....	27
4.11.3	DM break .....	27
4.12	DM registers .....	28
4.12.1	DM breakpoint register 1 extended byte (DM_BKR1E) .....	28
4.12.2	DM breakpoint register 1 high byte (DM_BKR1H) .....	28
4.12.3	DM breakpoint register 1 low byte (DM_BKR1L) .....	28
4.12.4	DM breakpoint register 2 extended byte (DM_BKR2E) .....	29
4.12.5	DM breakpoint register 2 high byte (DM_BKR2H) .....	29
4.12.6	DM breakpoint register 2 low byte (DM_BKR2L) .....	29
4.12.7	DM control register 1 (DM_CR1) .....	30
4.12.8	DM control register 2 (DM_CR2) .....	31
4.12.9	DM control/status register 1 (DM_CSR1) .....	32
4.12.10	DM control/status register 2 (DM_CSR2) .....	33
4.12.11	DM enable function register (DM_ENFCTR) .....	34
4.12.12	Summary of SWIM, DM and core register maps .....	35
<b>Appendix A Description of the DM_ENFCTR register for each STM8 product .....</b>		<b>37</b>
<b>Revision history .....</b>		<b>38</b>

## List of tables

Table 1.	SWIM command summary . . . . .	14
Table 2.	CPU register memory mapping in STM8 MCUs . . . . .	16
Table 3.	SWIM pin characteristics . . . . .	17
Table 4.	Decoding table for breakpoint interrupt generation. . . . .	23
Table 5.	STM8 MCU registers . . . . .	35
Table 6.	Peripherals which are frozen by the bits of the DM_ENFCTR register for each STM8 product . . . . .	37
Table 7.	Document revision history . . . . .	38

## List of figures

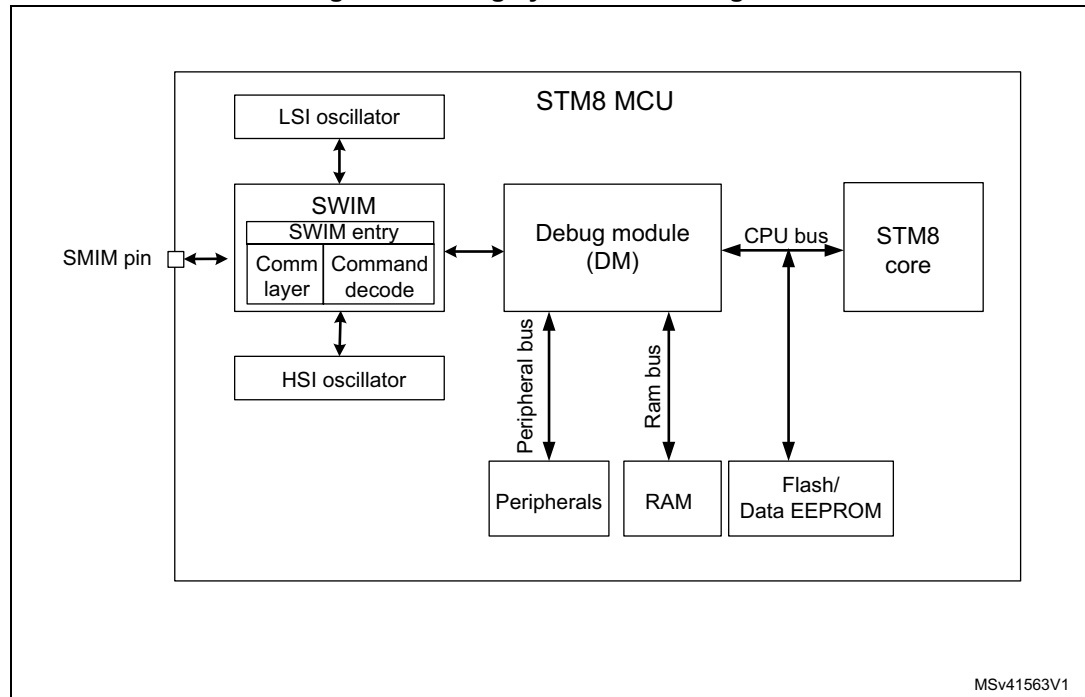
Figure 1.	Debug system block diagram . . . . .	6
Figure 2.	SWIM pin external connections . . . . .	7
Figure 3.	SWIM activation sequence . . . . .	8
Figure 4.	SWIM activation timing diagram . . . . .	9
Figure 5.	SWIM entry sequence . . . . .	10
Figure 6.	High speed bit format . . . . .	11
Figure 7.	Low speed bit format . . . . .	12
Figure 8.	Command format (host -> target) . . . . .	13
Figure 9.	Data format (target -> host) . . . . .	13
Figure 10.	Timings on the SWIM pin . . . . .	17
Figure 11.	Debug module block diagram . . . . .	21
Figure 12.	STM8 MCU instruction model . . . . .	24
Figure 13.	STM8 Debug module stall timing . . . . .	24
Figure 14.	STM8 DM data breaktiming . . . . .	25
Figure 15.	STM8 DM instruction break timing . . . . .	25
Figure 16.	STM8 DM step timing . . . . .	26

# 1 Debug system overview

The STM8 MCUs debug system interface allows a debugging or programming tool to be connected to the MCU through a single wire. This connection results in a bidirectional communication based on an open-drain line and provides a non-intrusive read/write access to RAM and peripherals during the program execution.

The block diagram is shown in [Figure 1](#).

**Figure 1. Debug system block diagram**



The debug module uses the two internal clock sources present in the device:

- the low speed internal clock (LSI clock): usually in the range of 30 kHz to 200 kHz depending on the product
- the high speed internal clock (HSI clock): usually in the range of 10 MHz to 25 MHz depending on the device.

The clocks are automatically started when necessary.

## 2 Communication layer

The SWIM is a single wire interface based on asynchronous, high sink (8 mA), open-drain, bidirectional communication. While the CPU is running, the SWIM allows a non-intrusive read/write accesses to be performed on-the-fly to the RAM and peripheral registers, for debug purposes.

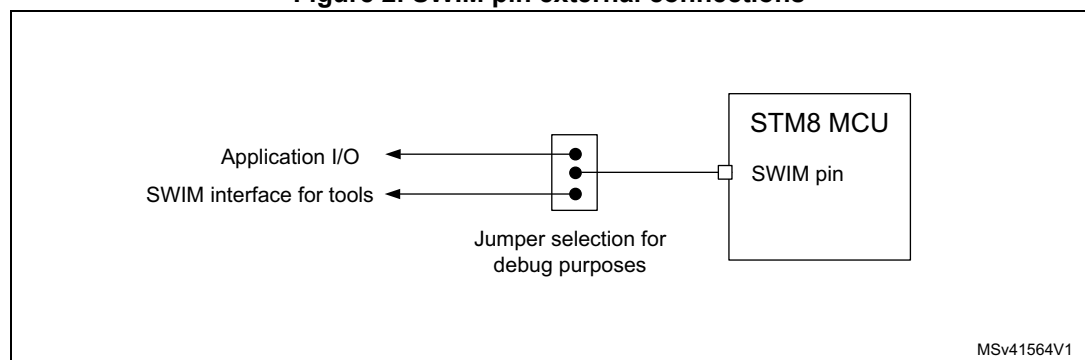
In addition, while the CPU is stalled, the SWIM allows read/write accesses to be performed to any other part of the MCU's memory space (data EEPROM and program memory).

The CPU registers (A, X, Y, CC, SP) can also be accessed. These registers are mapped in the memory and can be accessed in the same way as any other memory addresses. It is important to note that:

- Register, peripherals and memory can be accessed only when the SWIM\_DM bit is set.
- When the system is in HALT, WFI or readout protection mode, the NO\_ACCESS flag in the SWIM\_CSR register is set. In this case, it is forbidden to perform any accesses because parts of the device may not be clocked and a read access could return garbage or a write access might not succeed.

The SWIM can perform a MCU device software reset. The SWIM pin can also be used by the MCU target application as a standard I/O port with some restrictions if the user also want to use it for debug. The safest way is to provide a strap option on the application PCB.

**Figure 2. SWIM pin external connections**



### 3 Single wire interface module (SWIM)

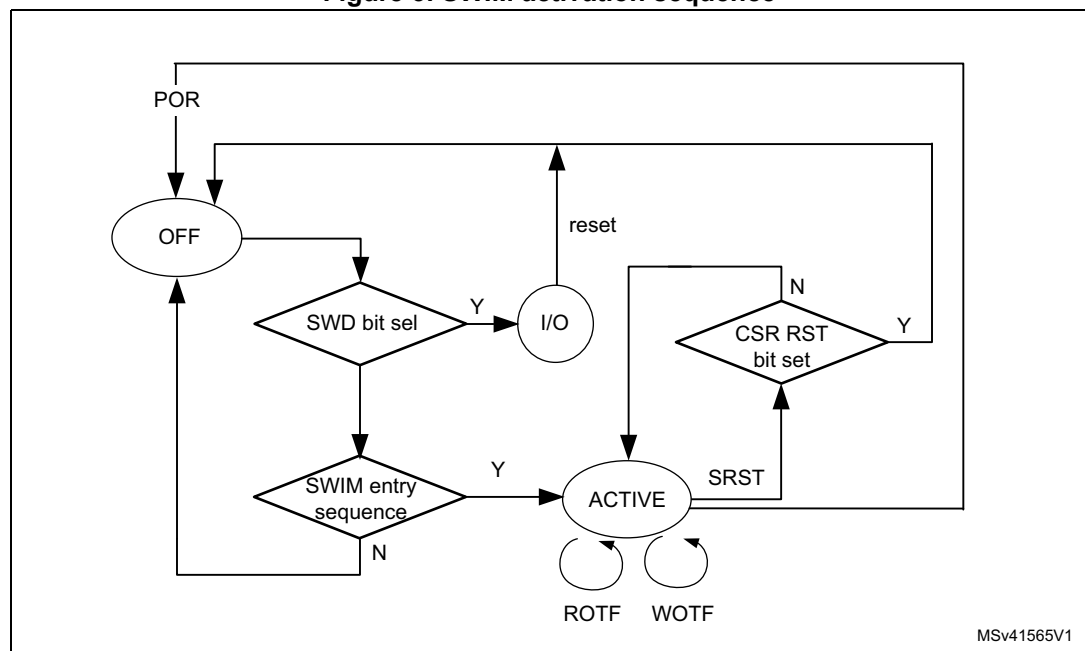
#### 3.1 Operating modes

After a power-on reset (powering of the device) the SWIM is reset and enters in its OFF mode.

1. **OFF**: in this mode the SWIM pin must not be used as an I/O by the application. It is waiting for the SWIM entry sequence or to be switched to I/O mode by the software application.
2. **I/O**: this state is entered by the software application by setting the SWIM disable bit (SWD) in the core configuration register (CFG\_GCR). In this state, the user application can use the SWIM pin as a standard I/O pin, the only drawback is that there is no way to debug the functionality of this pin with the built-in debug capabilities. In case of a reset, the SWIM goes back to OFF mode.
3. **ACTIVE**: this mode is entered when a specific sequence is detected on the SWIM pin while it is in the OFF state. In this state, the SWIM pin is used by the host tool to control the STM8 device with three commands: SRST (system reset), ROTF (read on-the-fly) and WOTF (write on-the-fly).

*Note:* Please note that the SWIM can be set as ACTIVE and communicate while the device is in RESET state (NRST pin forced low).

**Figure 3. SWIM activation sequence**





### 3.2 SWIM entry sequence

After a POR (power on reset), and as long as the SWIM is in OFF mode, the SWIM pin is sampled for entry sequence detection. In order to do this, the internal LSI (low speed RC - resistor/capacitor) clock is automatically turned ON after the POR and remains forced ON as long as the SWIM is in OFF mode.

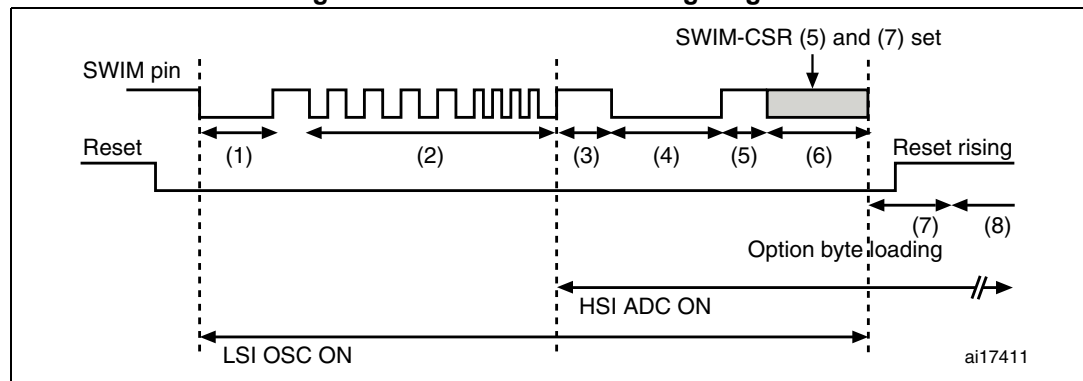
If the register that forces the SWIM in I/O mode is written before the entry sequence is finalized, the SWIM enters in I/O mode. Once the SWIM is ACTIVE, writing this bit has no influence on communication and the SWIM interface remains in ACTIVE mode.

If an application uses the SWIM pin as standard I/O, it puts the SWIM interface in I/O mode in the initialization section of the software code (typically, this is performed just after the reset). However, even in this case, it is still possible to put the SWIM interface in ACTIVE mode by forcing the RESET pin to 0 and keep it low for the duration of the SWIM entry sequence.

As long as the SWIM is in OFF mode, the SWIM entry sequence is detected at any moment, during reset or when the application is running.

If both the SWIM pin and the reset pin are multiplexed with I/Os, the way to enter the SWIM in ACTIVE state is to power down the MCU device, power up and to maintain the reset until the end of the SWIM entry sequence.

**Figure 4. SWIM activation timing diagram**

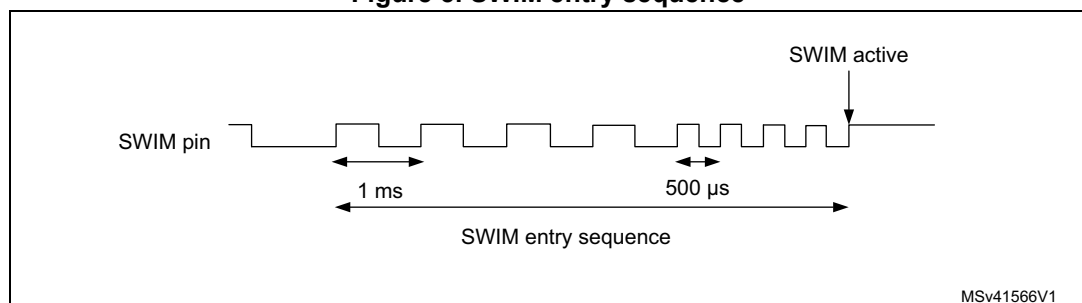


The SWIM activation is shown in [Figure 4](#) and each segment of the diagram is described below:

1. To make the SWIM active, the SWIM pin must be forced low during a period of 16  $\mu$ s.
2. After this first pulse at 0, the SWIM detects a **specific sequence** to guarantee the robustness in the SWIM active state entry. The SWIM entry sequence is: four pulses at 1 kHz followed by four pulses at 2 kHz. The frequency ratio is detected and allows the SWIM entry. The ratio can be easily detected regardless of the internal LSI frequency value. The waveform of the entry sequence is shown in [Figure 5](#). Note that the sequence starts and ends with the SWIM pin at 1.
3. After the entry sequence, the SWIM enters in SWIM active state, and the HSI oscillator is automatically turned ON.

4. After this delay, the SWIM sends a **synchronization** frame to the host.  
**Synchronization frame description:** a synchronization frame of 128 x SWIM clocks periods with the SWIM line at 0 is sent out by the MCU device to allow the measurement of the HSI by the debug host. An advanced debug host can re-calibrate its clock to adapt to the frequency of the internal HSI RC oscillator.
5. Before starting a SWIM communication, the SWIM line must be released at 1 to guarantee that the SWIM is ready for communication (at least 300 ns).
6. Write 0A0h in the SWIM\_CSR:
  - setting the bit 5 allows the whole memory range and the SRST command to be accessed.
  - setting the bit 7 masks the internal reset sources
7. Release the reset which starts the option byte loading sequence. Wait 1 ms for stabilization.
8. Once the option byte loading has occurred and that the stabilization time is reached, the CPU is in phase 8:
  - STM8 is stalled and HSI = 16 Mhz (see STM8 datasheets for HSI clock accuracy)
  - SWIM clock is at HSI/2 = 8 Mhz
  - SWIM is active in low speed bit format (see [Section 3.3.2](#))
9. After the HSI is calibrated internally, a copy of the factory calibration value is uploaded from the option bytes and stored into the HSI calibration register at the RAM, then a SWIM communication reset command can be generated to get the synchronization frame again but with greater reliability than in step 4. Depending on the target context since power on, the HSI clock could be not well calibrated in step 4, because at that moment the HSI calibration register in the RAM is not yet initialized with the proper value.

Figure 5. SWIM entry sequence



### 3.3 Bit format

The bit format is a return-to-zero format, which allows a synchronization of every bit. Two communication speeds are available. At SWIM activation, the low speed is selected, while the high speed is selected by setting the HS bit in the SWIM\_CSR register with the SWIM protocol.

When entering the SWIM mode during the RESET phase, it is possible that the option bytes have not yet been loaded from non volatile memory to their respective registers. The option byte loading is triggered by any internal or external reset.

In order to ensure proper system behavior, the HS bit should not be set until the option byte loading is finished. At the end of the option byte loading, the HSIT bit in the SWIM\_CSR is set by hardware.

#### 3.3.1 High speed bit format

One bit is generated with ten SWIM clock pulses.

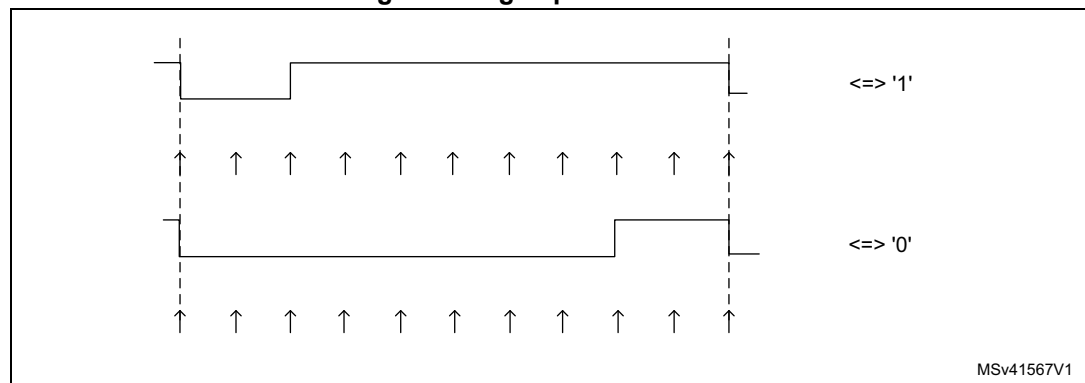
The bit format is:

- 2 pulses at '0' followed by 8 pulses at '1' for '1' value.
- 8 pulses at '0' followed by 2 pulses at '1' for '0' value.

When the SWIM receives a data packet, it will decode:

- '1' when the number of consecutive samples at '0' is less or equal to 4.
- '0' when the number of consecutive samples at '0' is greater or equal to 5.

**Figure 6. High speed bit format**



### 3.3.2 Low speed bit format

1 bit is generated with twenty-two SWIM clock pulses.

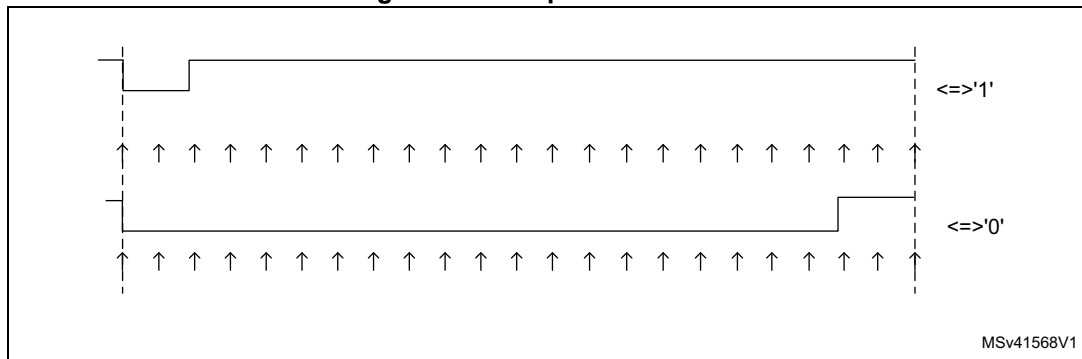
The bit format is:

- 2 pulses at '0' followed by 20 pulses at '1' for '1' value.
- 20 pulses at '0' followed by 2 pulses at '1' for '0' value.

When the SWIM receives a data packet, it will decode:

- '1' when the number of consecutive samples at '0' is less or equal to 8.
- '0' when the number of consecutive samples at '0' is greater or equal to 9.

Figure 7. Low speed bit format

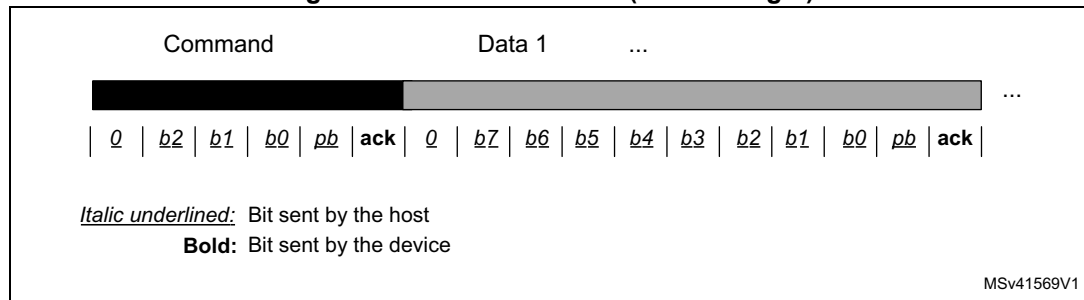


### 3.4 SWIM communication protocol

When in SWIM is in ACTIVE mode, the communication can be initiated either by the host or by the device. Each byte or command is preceded by a 1-bit header in order to arbitrate if both host and device initiate the communication at the same time.

The host header is '0' in order to have the priority over the device in case of arbitration, due to open-drain capability. The host can start the transfer only if there is no transfer ongoing.

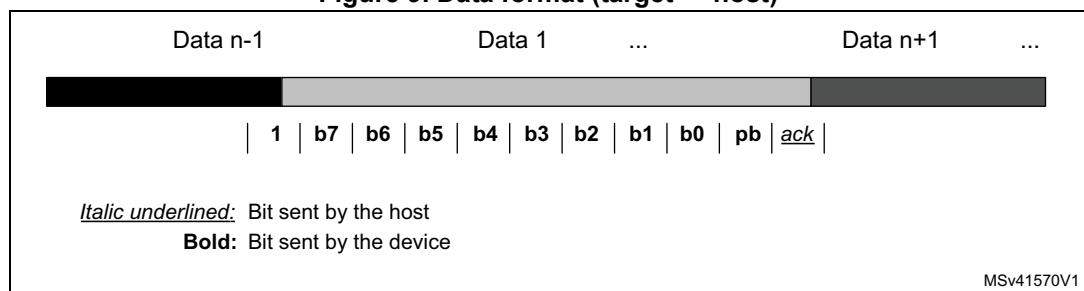
**Figure 8. Command format (host -> target)**



Each command sent by the host is made of:

- 1 command (ROTF, WOTF or SWRST) made of:
  - Header: 1 bit at '0'
  - b2-b0: 3-bit command
  - pb: parity bit: XOR between all b(i)
  - ack: acknowledge (1 bit at '1'). The receiver must send the not-acknowledge value if it has detected a parity error (NACK: not acknowledge = 1 bit at '0'), or it is not yet ready.
- optionally several data packets (in case of WOTF) made of:
  - Header: 1 bit at '0'
  - b7-b0: 8-bit data
  - pb: parity bit sent after data. XOR between all b(i)
  - ack: acknowledge

**Figure 9. Data format (target -> host)**



Each data frame is made of:

- Header: 1 bit at '1'
- b7-b0: 8-bit data
- pb: parity bit sent after data
- ack: acknowledge

### 3.5 SWIM commands

The host can send a command when the line is idle or after each data byte from the device. After sending the command, the host releases the line. When the SWIM is ready to answer to the command, it initiates the transfer. If a new command from the host occurs while a command is pending in SWIM, the pending command is canceled and the new command is decoded, except in the case of WOTF.

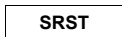
Three commands are available. They are listed in [Table 1](#).

**Table 1. SWIM command summary**

Command	Binary code
SRST	000
ROTF	001
WOTF	010
Reserved for future use	011
	1xx

#### 3.5.1 SRST: system reset

**Format:** 1 command from host to target



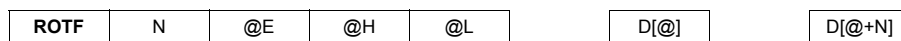
**Parameters:**

None.

The SRST command generates a system reset only if the SWIM\_CSR/SWIM\_DM bit is set.

#### 3.5.2 ROTF: read on-the-fly

**Format:** 1 command followed by the number of bytes to be read followed by the address on three bytes.



**Parameters:**

N            The 8 bits are the number of bytes to read (from 1 to 255)

@E/H/L:    This is the 24-bit address to be accessed.

D[...]:     These are the data bytes read from the memory space

If the host sends a NACK to a data byte, the device will send the same byte again.

If the SWIM\_DM bit is cleared, the ROTF can only be done on the SWIM internal registers.

### 3.5.3 WOTF: write on-the-fly

1 command followed by the number of bytes to be written followed by the address on three bytes.

WOTF	N	@E	@H	@L	D[@]	D[@+N]
------	---	----	----	----	------	--------

**Parameters:**

- N            The 8 bits are the number of bytes to write (from 1 to 255)
- @E/H/L:    This is the 24-bit address to be accessed.
- D[...]:     These are the data bytes to write in the memory space

If a byte D [i] has not been written when the following byte D [i+1] arrives, D [i+1] will be followed by a NACK. In this case the host must send D [i+1] again until it is acknowledged.

For the last byte, if it is not yet written when a new command occurs, the new command will receive a NACK and will not be taken into account.

If the SWIM\_DM bit is cleared, the WOTF can only be done on the SWIM internal registers.

## 3.6 SWIM communication reset

In case of a problem during the communication, the host can reset the communication and the ongoing command by sending 128 x SWIM clocks periods low on the SWIM pin. If the SWIM logic detects that the SWIM pin is low for more than 64 x SWIM clocks periods, it will reset the communication state machine and will switch the SWIM to low-speed mode (SWIM\_CSR.HS <- 0). This is done to allow a variation in the frequency of the internal RC oscillator.

In response to this communication reset, the SWIM sends the synchronization frame which is 128 x SWIM clock periods low on the DBG pin.

### 3.7 CPU register access

The CPU registers are mapped in the STM8 memory, and they can be read or written directly using the ROTF and the WOTF SWIM commands. The write operations to the CPU registers are committed only when the CPU is stalled.

To flush the instruction decode phase, the user must set the FLUSH bit in the DM control/status register 2 (named DM\_CSR2) after writing a new value in the program counter (PCE, PCH, PCL) Refer to [Section 4.12.10 on page 33](#) for more details.

**Table 2. CPU register memory mapping in STM8 MCUs**

CPU register	Memory location
A	7F00h
PCE	7F01h
PCH	7F02h
PCL	7F03h
XH	7F04h
XL	7F05h
YH	7F06h
YL	7F07h
SPH	7F08h
SPL	7F09h
CC	7F0Ah

### 3.8 SWIM communication in Halt mode

To maintain the communication link with the debug host, the HSI oscillator remains ON when the MCU enters the Halt mode. This means that Halt mode power-consumption measurements have no meaning when the SWIM is active.

In Halt mode, the user can access the SWIM module but not the DEBUG module. The NO\_ACCESS bit in the SWIM\_CSR register is set when the system is in HALT, WFI or readout protection mode. It means that in this case, no connection is accessible between the SWIM module and the DEBUG module nor between the rest of the STM8 systems.

The OSCOFF bit in the SWIM\_CSR register is used to switch off the HSI oscillator. In this case, any access to the SWIM module is lost as long as the device is in Halt mode and that the SWIM pin is high. The only way to recover the debug control is to induce a falling edge on the SWIM pin: this will re-enable the HSI oscillator.



### 3.9 Physical layer

During the communication, the SWIM pin will be in pseudo-open drain configuration. The SWIM pin in the device is capable of sinking 8 mA when it drives the line to 0. The external pull-up on the SWIM line should be sized in a way that the maximum rise time  $t_r$  of the SWIM line is less than 1 sampling period of the bit (which is 100 ns +/- 4 %).

Figure 10. Timings on the SWIM pin

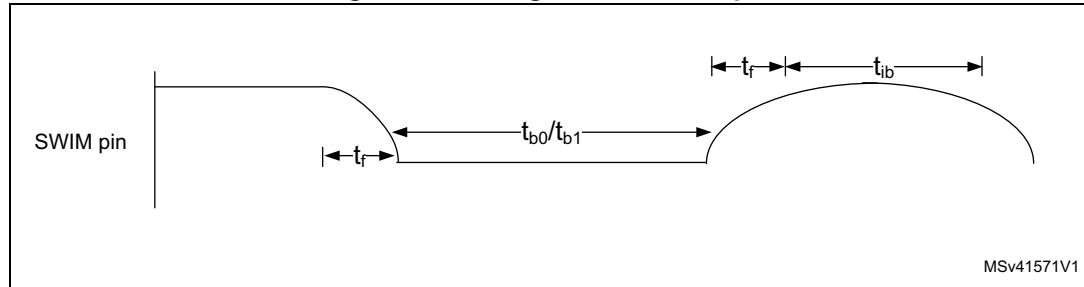


Table 3. SWIM pin characteristics

Parameter	Symbol	Generic formula	Timings for HSI = 10 MHz LSI = 32 to 64 kHz	
			Min	Max
Fall time on SWIM pin	$t_f$	TBD	-	50 ns
Rise time on SWIM pin	$t_r$	TBD	-	96 ns
Inter-bit time (The time which SWIM pin stays high between 2 bits)	$t_{ib}$	TBD	>0	-
Inter-frame time (Time between end of a frame and the next one)	$t_{if}$	TBD	0	-
Low time for a bit at 0	High speed: $t_{b0}$	TBD	768 ns	832 ns
	Low speed: $t_{b0}$	TBD	1.6 $\mu$ s	2.4 $\mu$ s
Low time for a bit at 1 (high speed)	High speed: $T_{b1}$	TBD	192 ns	208 ns
	Low speed: $T_{b1}$	TBD	150 ns	250 ns
Injected current on SWIM pin	-	TBD	-	8 mA

### 3.10 STM8 MCUs SWIM registers

#### 3.10.1 SWIM control status register (SWIM\_CSR)

Address: 7F80h

Reset value: 00h

This register is reset only by a power on reset or by a SWIM SRST command if the RST bit =1 in the SWIM\_CSR register.

7	6	5	4	3	2	1	0
SAFE_MASK	NO_ACCESS	SWIM_DM	HS	OSCOFF	RST	HSIT	PRI
rw	r	rw	rw	rw	rw	r	rw

**SAFE\_MASK:** *Mask internal RESET sources*

Bit 7 This bit can be read or written through the SWIM only. It cannot be accessed through the STM8 bus. It includes the watchdog reset.

- 0: Internal reset sources are not masked
- 1: Internal reset sources are masked.

**NO\_ACCESS:** *Bus not accessible*

Bit 6 This bit can be read through the SWIM only, to determine if the bus is accessible or not. It is set automatically if the device is in HALT, WFI or readout protection mode.

- 0: Bus is accessible
- 1: Bus is not accessible

**Caution:** Depending on the SWIM revision, in some devices, the NO\_ACCESS bit indicates only that the device is in HALT mode.

**SWIM\_DM:** *SWIM for debug module*

Bit 5 This bit can be read or written to 1 through the SWIM only. It cannot be accessed through the STM8 bus.

- 0: The SWIM can access only the SWIM\_CSR register. The SWIM reset command has no effect
- 1: The whole memory range can be accessed with the ROTF and WOTF commands. The SRST command generates a reset.

**HS:** *High speed*

Bit 4 This bit can be read or written through the SWIM only. It cannot be accessed through the STM8 bus.

- 0: Low speed bit format
- 1: High speed bit format

The speed change occurs when the communication is IDLE. It is reset by the SWIM communication reset condition as described in [Section 3.6](#).

**OSCOFF:** *Oscillators off control bit*

Bit 3 This bit can be read or written through the SWIM only. It cannot be accessed through the STM8 bus.

- 0: HSI oscillator remains ON in Halt mode
- 1: HSI oscillator is not requested ON in Halt mode

- RST: SWIM reset control bit**  
 This bit can be read or written through the SWIM only. It cannot be accessed through the STM8 bus.  
 0: SWIM is not reset when a SRST command occurs.  
 1: SWIM is reset when a SRST command occurs. The SWIM will re-enter OFF mode.
- HSIT: High speed internal clock is trimmed**  
 This bit is read only through SWIM only. It cannot be accessed through STM8 bus. It is set when the HSIT bit is set in the core configuration register and reset by an external reset.  
 0: High speed internal clock is not trimmed, the SWIM must remain in low speed mode.  
 1: High speed internal clock is trimmed, the SWIM high speed mode is allowed.
- PRI: SWIM access priority**  
 This bit can be read or written through the SWIM only. Usually the SWIM accesses to system resources are non-intrusive, the SWIM having the lowest priority. This can be overridden by setting this bit.  
 0: Non-intrusive access by the SWIM to system resources (low priority)  
 1: Intrusive access by the SWIM to system resources (SWIM has the priority, the CPU is stalled).  
**Note:** The SWD bit is located in the STM8 core configuration register. Refer to the corresponding datasheet for information on this register

### 3.10.2 SWIM clock control register (CLK\_SWIMCCR)

Address Offset: 50CDh (product dependent)

Reset value: xxxx 0000 (x0h)

	7	6	5	4	3	2	1	0
Reserved								SWIMCLK
-								rw

- Bits 7:1      Reserved, must be kept cleared.
- SWIMCLK SWIM clock divider**  
 This bit is set and cleared by software.  
 0: SWIM clock is divided by 2 (recommended)  
 1: SWIM clock is not divided by 2 (not recommended as communication is less reliable)  
**Note:** this register is not present in some STM8 devices.

## 4 Debug module (DM)

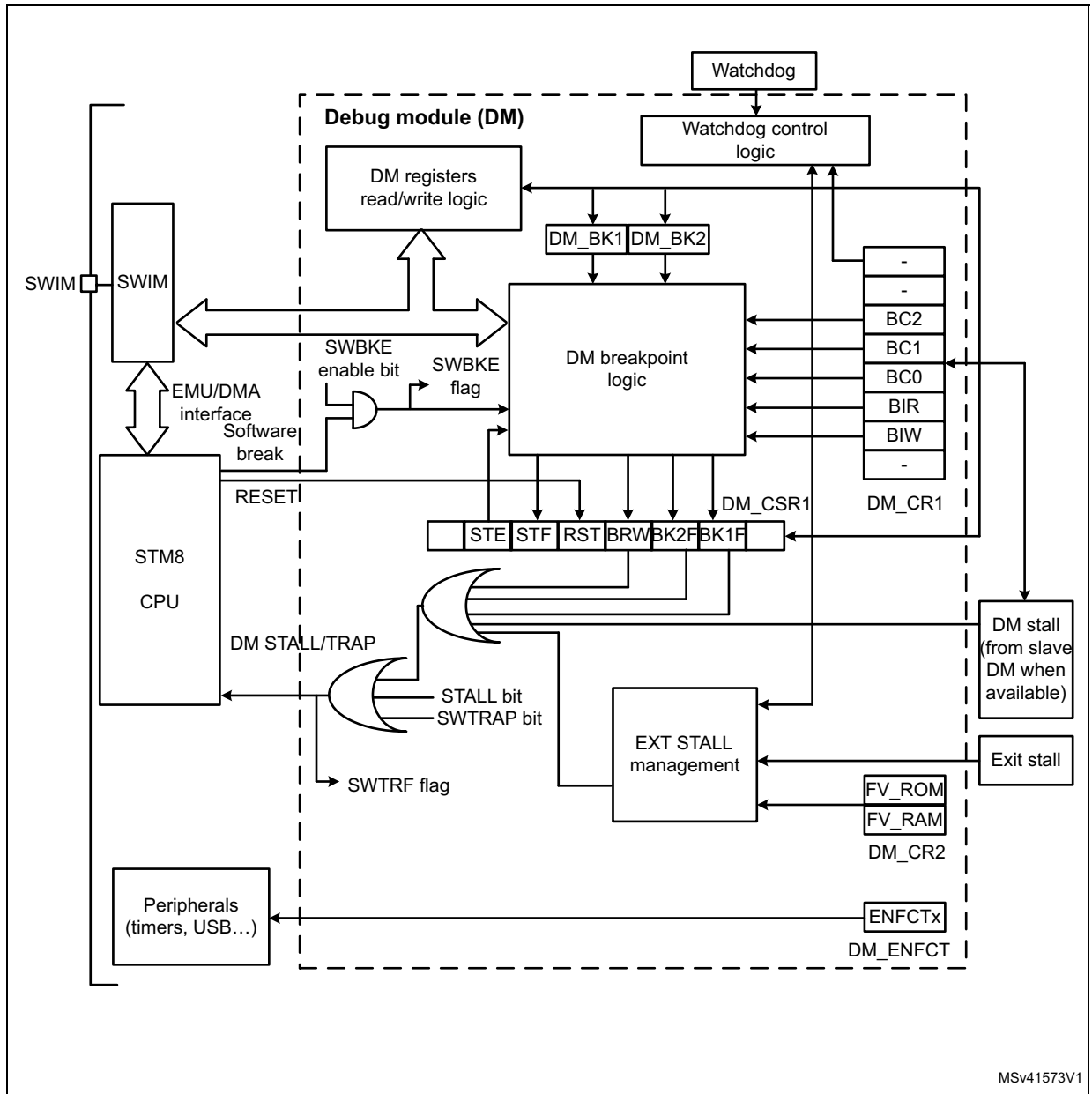
### 4.1 Introduction

The debug module (DM) allows the developer to perform certain debugging tasks without using an emulator. For example, the DM can interrupt the MCU to break infinite loops or to output the core context (stack) at a given point. The DM is mainly used for in-circuit debugging.

### 4.2 Main features

- Two conditional breakpoints (such as break on instruction fetch, data read or write, stack access)
- Software breakpoint control
- Step mode
- External stall capability on the WOTF command in the SWIM mode
- Watchdog and peripherals control
- DM version identification capability
- Interrupt vector table selection

Figure 11. Debug module block diagram



## 4.3 Debug

The DM registers can be read and written only through the SWIM interface. The STM8 core has no access to these registers.

### 4.3.1 Reset

Once the SWIM is active and that the SWIM\_DM bit is set in the SWIM\_CSR register, a 'data read' breakpoint at the reset vector address is automatically set, due to the reset values of the debug module registers. This breakpoint can be used to initialize the debug session.

### 4.3.2 Breakpoints

The DM generates a stall to the core when a breakpoint is reached. When the processor is stalled, the host can read or modify any address in memory. Access to the processor registers is explained in [Table 3.7: CPU register access on page 16](#).

To restart the program execution, the STALL bit in the DM\_CSR2 must be cleared using the WOTF command of the SWIM protocol.

### 4.3.3 Abort

To use the abort function, the host must write the STALL bit in the DM\_CSR2 using the SWIM WOTF command.

No interrupt is generated. The core is stalled in the current state. Using the SWIM commands, the host can read and modify the status of the MCU. Use the procedure described in [Section 3.7: CPU register access](#) if the CPU registers must be modified.

The host can restart the program execution by resetting the STALL bit using the SWIM commands.

### 4.3.4 Watchdog control

Using the WDGOFF bit in the DM control register 1 (DM\_CR1) (see [Section 4.12.7 on page 30](#)) the user can configure the window watchdog and the independent watchdog counters to be stopped while the CPU is stalled by the debug module. This bit must be set before the watchdogs are activated. If a watchdog is enabled by the hardware watchdog option bit, the WDGOFF bit has no effect on it.

### 4.3.5 Interaction with SWIM

The SWIM sends the status bit which indicates if the SWIM is active or not. When the SWIM is not active, the DM does not generate any break/stall request to the CPU.

## 4.4 Breakpoint decoding table

Table 4. Decoding table for breakpoint interrupt generation

DM_CR1					BREAK CONDITIONS	DM_CSR1		
BC2	BC1	BC0	BIR	BIW		BK1F	BK2F	BRW
0	0	0	0	0	Disabled (RESET state)	0	0	x
0	0	0	0	1	Data Write on @=BK1 and Data=BK2L	1	0	0
0	0	0	1	0	Data Read on @=BK1 and Data=BK2L	1	0	1
0	0	0	1	1	Data R/W on @=BK1 and Data=BK2L	1	0	0/1
0	0	1	0	0	Instruction fetch BK1<=@<=BK2	1	0	x
0	0	1	0	1	Data Write on BK1<=@<=BK2	1	0	0
0	0	1	1	0	Data Read on BK1<=@<=BK2	1	0	1
0	0	1	1	1	Data R/W on BK1<=@<=BK2	1	0	0/1
0	1	0	0	0	Instruction fetch on @<= BK1 or BK2<=@	1	0	x
0	1	0	0	1	Data Write on @<= BK1 or BK2<=@	1	0	0
0	1	0	1	0	Data Read on @<= BK1 or BK2<=@	1	0	1
0	1	0	1	1	Data R/W on @<= BK1 or BK2<=@	1	0	0/1
0	1	1	X	X	Disabled	0	0	x
1	0	0	0	0	Instruction fetch on @=BK1 then on @=BK2	0	1	x
1	0	0	0	1	Data Write on @=BK1 or @=BK2	10 or 01 or 11		0
1	0	0	1	0	Data Read on @=BK1 or @=BK2	10 or 01 or 11		1
1	0	0	1	1	Data R/W on @=BK1 or @=BK2	10 or 01 or 11		0/1
1	0	1	0	0	Instruction fetch on @=BK1 or @=BK2	10 or 01 or 11		x
1	0	1	0	1	Instruction fetch on @=BK1 / Data Write on @=BK2	10 or 01		x-0
1	0	1	1	0	Instruction fetch on @=BK1 / Data Read on @=BK2	10 or 01		x-1
1	0	1	1	1	Instruction fetch on @=BK1 / Data R/W on @=BK2	10 or 01		x-0/1
1	1	0	X	X	Disabled	0	0	x
1	1	1	0	0	Data Write in Stack on @<=BK1 / Instruction fetch on @=BK2	10 or 01		0-x
1	1	1	0	1	Data Write in Stack on @<=BK1 / Data Write on @=BK2	10 or 01 or 11		0
1	1	1	1	0	Data Write in Stack on @<=BK1 / Data Read on @=BK2	10 or 01		0-1
1	1	1	1	1	Data Write in Stack on @<=BK1 / Data R/W on @=BK2	10 or 01 or 11		0-0/1

### 4.5 Software breakpoint mode

The software breakpoint mode is reserved for the debugging tools to insert breakpoints into the user code by substituting a user instruction with a software break (reserved BKPT instruction #8b).

The software breakpoint mode is enabled using the SWBKPE bit in the DM control/status register 2 (DM\_CSR2) (see [Section 4.12.10 on page 33](#)).

When a BKPT instruction is decoded, the CPU is stalled and the STALL and SWBKF bits are set by hardware to indicate that a software breakpoint has occurred. To resume the execution, the debugger must restore the user's instruction, then set the FLUSH bit and clear the STALL bit.

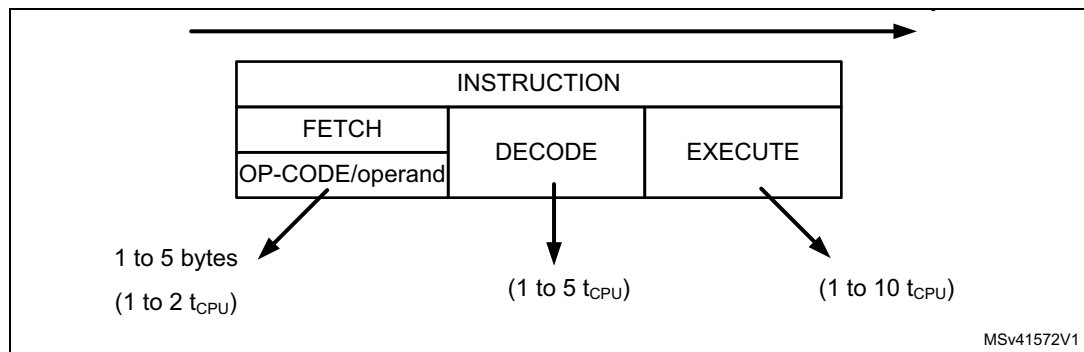
### 4.6 Timing description

This paragraph defines when the debug module stalls the CPU when using the different breakpoint sources.

- The STM8 MCU instruction can be modeled in time with an op-code/operand with a FETCH, DECODE and EXECUTION phases as shown in [Figure 12](#).

The timing information is based on these models.

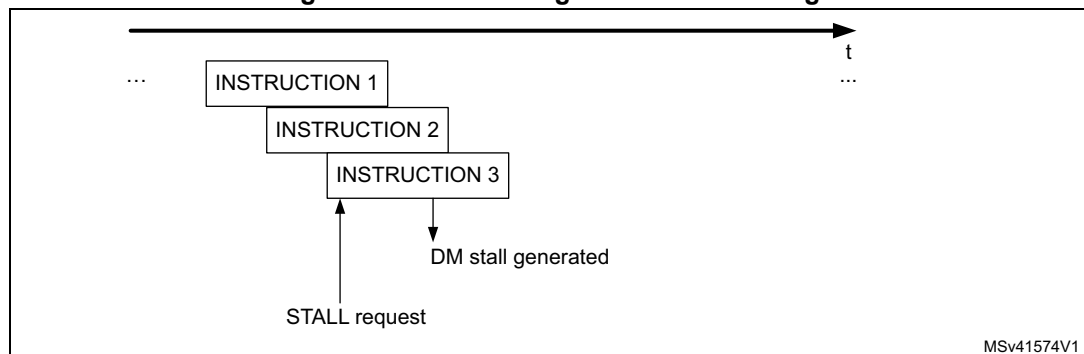
**Figure 12. STM8 MCU instruction model**



### 4.7 Abort

The stall is generated immediately when writing the STALL bit in the DM\_CSR2 register.

**Figure 13. STM8 Debug module stall timing**

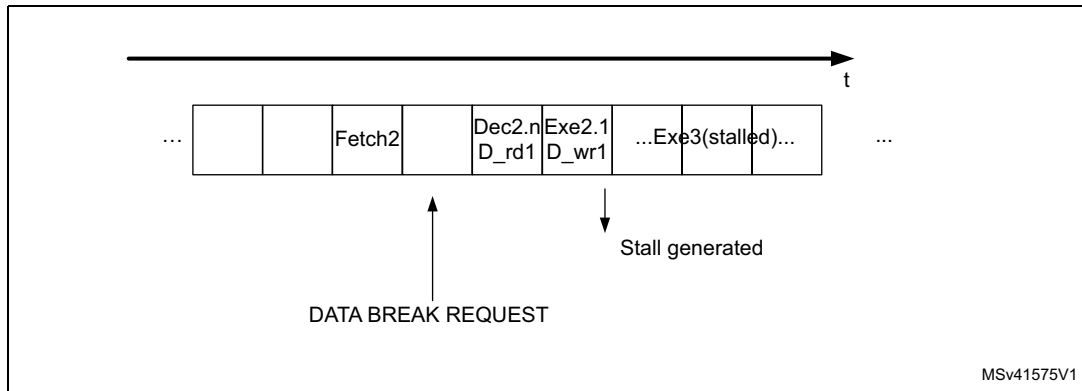




### 4.8 Data breakpoint

A stall is generated when the SWIM is active, and after the end of the current instruction execution.

Figure 14. STM8 DM data break timing



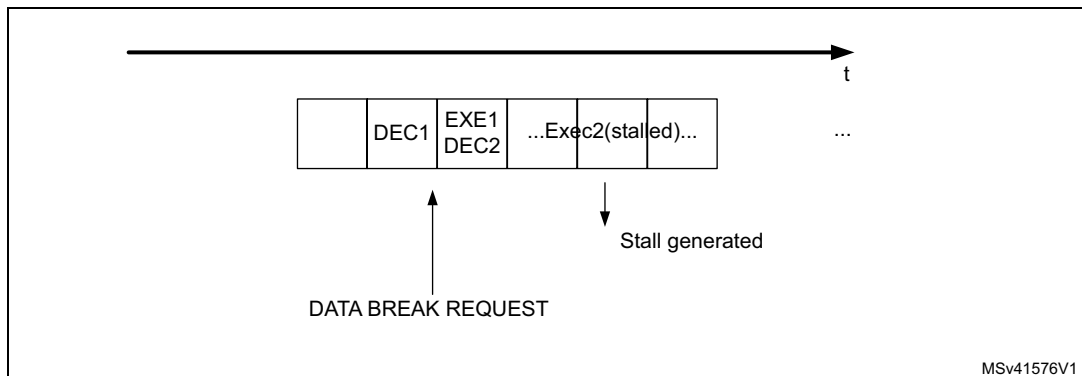
MSv41575V1

### 4.9 Instruction breakpoint

In the STM8 devices, with an instruction break, DM stalls the CPU before the selected instruction execution (while the instruction is in the decode stage). See [Figure 15](#).

*Note:* When the specified address does not correspond to a valid instruction address, no stall is generated.

Figure 15. STM8 DM instruction break timing

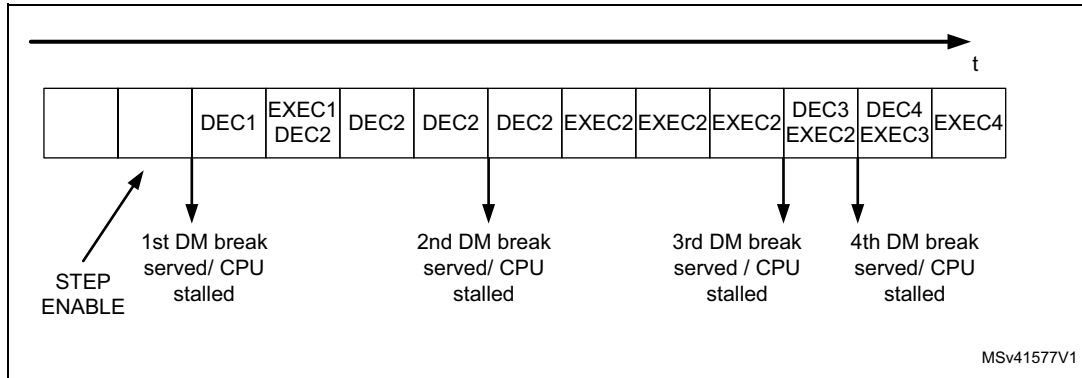


MSv41576V1

### 4.10 Step mode

The STM8 CPU stall is activated before the instruction execution, in the first decode cycle of the instruction. See [Figure 16](#).

Figure 16. STM8 DM step timing



*Note:* When the Step mode and the instruction break on the next instruction mode are both enabled, both the STF and the BKxF flags are set. When the user clears the STALL bit, the step function continues its normal operation.

## 4.11 Application notes

### 4.11.1 Illegal memory access

To verify if the program attempts to write or read in an illegal part of the memory (a reserved area), select the “Data R/W on BK1<=@<=BK2” condition, where BK1 and BK2 are the lower and upper addresses of the reserved memory.

### 4.11.2 Forbidden stack access

If part of the stack contains specific data or instructions that should not be overwritten, the DM can be used to prevent the access to these locations. Select one of the “Data Write in Stack on @<=BK1” conditions and set BK1 to the upper value where the specific data is located in the stack. If the STM8 device tries to overwrite these values (after an interrupt or a CALL), the DM will generate a break. The four possible associated conditions allow to manage another breakpoint capability at the same time.

### 4.11.3 DM break

After a DM break, the CPU is stalled (through the EMU\_Stall signal). While the CPU is stalled, the SWIM can read/write any memory location or any memory mapped register. The program can be continued from the breakpoint, by resetting the stall bit.

If a change of PC is needed, the SWIM must write the new PC value using the method described in [Section 3.7: CPU register access on page 16](#). In order to fetch the code from the new PC address, the SWIM must set the FLUSH bit in the DM control/status register 2 (DM\_CSR2) (refer to [Section 4.12.10 on page 33](#)) before resetting the STALL bit.

## 4.12 DM registers

These registers are read/write only through the SWIM interface.

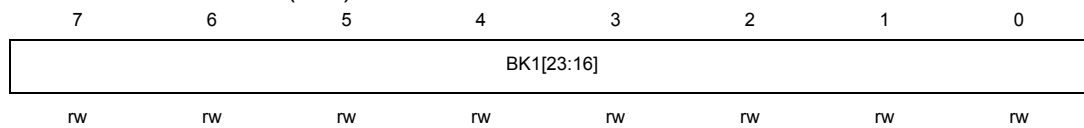
In this section, the following abbreviations are used:

read/write (rw)	The SWIM can read and write to these bits via the ROTF/WOTF commands.
read-only (r)	The SWIM can only read these bits via the ROTF command.

### 4.12.1 DM breakpoint register 1 extended byte (DM\_BKR1E)

STM8 Address: 7F90h

Reset value: 1111 1111 (FFh)

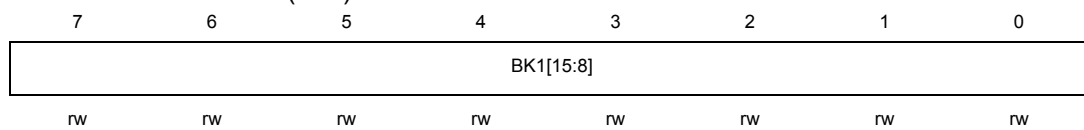


Bits 7:0 **BK1[23:16]: Breakpoint 1 extended byte value**  
 This register is written by software to define the extended address bits of breakpoint 1.

### 4.12.2 DM breakpoint register 1 high byte (DM\_BKR1H)

Address: 7F91h

Reset value: 1111 1111 (FFh)

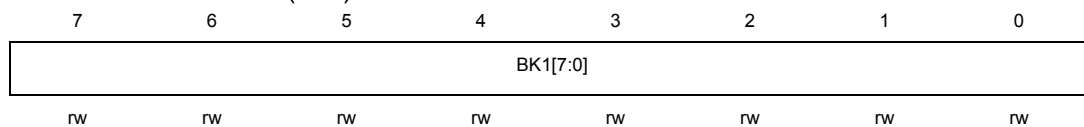


Bits 7:0 **BK1[15:8]: Breakpoint 1 high byte value**  
 This register is written by software to define the higher 8 address bits of breakpoint 1.

### 4.12.3 DM breakpoint register 1 low byte (DM\_BKR1L)

Address: 7F92h

Reset value: 1111 1111 (FFh)

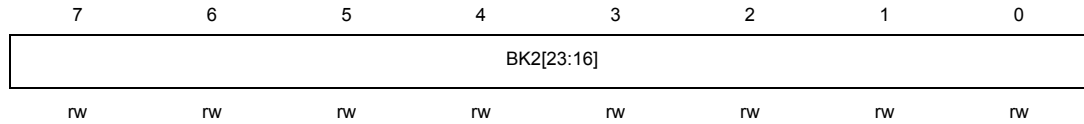


Bits 7:0 **BK1[7:0]: Breakpoint 1 low byte value**  
 This register is written by software to define the lower 8 address bits of breakpoint 1.

### 4.12.4 DM breakpoint register 2 extended byte (DM\_BKR2E)

Address: 7F93h

Reset value: 1111 1111 (FFh)



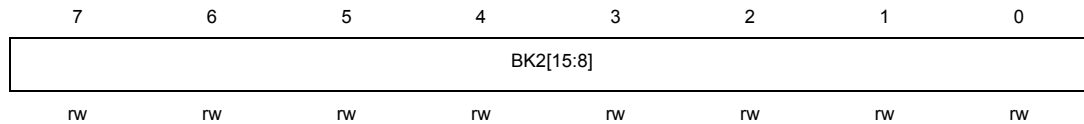
**BK2[23:16]: Breakpoint 2 extended byte value**

Bits 7:0 This register is written by software to define the extended 8 address bits of breakpoint 2.

### 4.12.5 DM breakpoint register 2 high byte (DM\_BKR2H)

Address: 7F94h

Reset value: 1111 1111 (FFh)



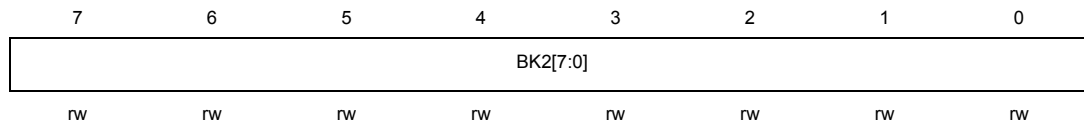
**BK2[15:8]: Breakpoint 2 high byte value**

Bits 7:0 This register is written by software to define the higher 8 address bits of Breakpoint 2.

### 4.12.6 DM breakpoint register 2 low byte (DM\_BKR2L)

Address: 7F95h

Reset value: 1111 1111 (FFh)



**BK2[7:0]: Breakpoint 2 low byte value**

Bits 7:0 This register is written by software to define the lower 8 address bits of breakpoint 2.

### 4.12.7 DM control register 1 (DM\_CR1)

Address: 7F96h

Reset value: 0000 0000 (00h)

7	6	5	4	3	2	1	0
WDGOFF	Reserved	BC[2:0]			BIR	BIW	Reserved
rw	-	rw	rw	rw	rw	rw	-

**WDGOFF** *Watchdog control enable.*

Bit 7 This bit must be set or cleared by software before the watchdogs (WWDG and/or IWDG) are activated. This bit has no effect if the hardware watchdog option is selected.  
 0: Watchdog counters are not stopped while the CPU is stalled by DM  
 1: Watchdog counters are stopped while the CPU is stalled by DM

Bit 6 Reserved.

**BC[2:0]** *Breakpoint control*

Bits 5:3 These bits are set and cleared by software, they are used to configure the breakpoints as shown in [Table 4](#).

**BIR** *Break on read control*

Bit 2 This bit enables a breakpoint on a data read operation. It is set and cleared by software.  
 0: No break on data read  
 1: Break on data read

**BIW** *Break on write control*

Bit 1 This bit enables a breakpoint on a data write operation. It is set and cleared by software.  
 0: No break on data write  
 1: Break on data write

Bit 0 Reserved.

### 4.12.8 DM control register 2 (DM\_CR2)

Address: 7F97h

Reset value: 0000 0000 (00h)

7	6	5	4	3	2	1	0
Reserved					FV_ROM	Reserved	FV_RAM
					rw		
							rw

Bit 7:3 These bits are reserved and must be kept at 0.

**FV\_ROM Remap vector table in ROM.**

Bit 2 This bit is set or cleared by software. It remaps the vector table to a ROM location (product dependent) instead of program memory (usually 8000h).  
 0: Vector table is in the program memory area (8000h)  
 1: Vector table is in the ROM memory area (depends on the product).

Bit 1 Reserved, must be kept at 0.

**FV\_RAM Remap vector table in RAM**

Bit 0 This bit is set or cleared by software. It remaps the interrupt vector table to a RAM location instead of program memory (usually 8000h).  
 0: Vector table is in Program Memory area (8000h)  
 1: Vector table is in RAM memory area (address depends on the product).

### 4.12.9 DM control/status register 1 (DM\_CSR1)

Address: 7F98h

Reset value: 0001 0000 (10h)

7	6	5	4	3	2	1	0
Reserved	STE	STF	RST	BRW	BK2F	BK1F	Reserved
	rw	rw	r	r	r	r	

Bit 7 Reserved.

**STE Step mode enable** (read / write)

Bit 6 This bit is set and cleared by software. It enables the Step mode.  
 0: Step mode disabled  
 1: Step mode enabled

**STF Step flag** (read only)

Bit 5 This bit indicates that the stall was generated by the Step mode. It is set and cleared by hardware. Writing to this bit does not change the bit value.  
 0: Step mode stall did not occur  
 1: Step mode stall occurred

**RST Reset flag** (read only)

Bit 4 This bit is set by hardware when the CPU was stalled by the debug module (DM), just after reset. It is cleared by hardware when the STALL bit is cleared. Writing to this bit does not change the bit value.  
 0: No reset occurred  
 1: A reset occurred

**BRW Break on read/write flag** (read only).

Bit 3 This bit gives the value of the read/write signal when a break occurs. Its value is not significant for the instruction fetch breaks. It is set by hardware depending on the breakpoint conditions (see [Table 4: Decoding table for breakpoint interrupt generation on page 23](#)) and is cleared by hardware depending on the next breakpoint conditions. Writing to this bit does not change the bit value.  
 0: Breakpoint on write  
 1: Breakpoint on read

**BK2F Breakpoint 2 flag** (read only).

Bit 2 This bit indicates that the DM stall was generated by breakpoint 2. It is set by hardware depending on the control conditions (see [Table 4: Decoding table for breakpoint interrupt generation on page 23](#)) and it is cleared by hardware when the STALL bit is cleared. Writing to this bit does not change the bit value.  
 0: Breakpoint 2 did not occur  
 1: Breakpoint 2 occurred

**BK1F Breakpoint 1 flag** (read only).

Bit 1 This bit indicates that the DM interrupt was generated by breakpoint 1. It is set by hardware depending on the control conditions (see [Table 4: Decoding table for breakpoint interrupt generation on page 23](#)) and it is cleared by hardware when the STALL bit is cleared. Writing to this bit does not change the bit value.  
 0: Breakpoint 1 did not occur  
 1: Breakpoint 1 occurred

Bit 0 Reserved



### 4.12.10 DM control/status register 2 (DM\_CSR2)

Address: 7F99h

Reset value: 0000 0000 (00h)

7	6	5	4	3	2	1	0
Reserved		SWBRK	SWBKF	STALL	Reserved		FLUSH
		rw	r	r			rw

Bits 7:6 Reserved. It must be kept at 0

**SWBKE** *Software breakpoint control bit (read/write)*

This bit is used to enable/disable the software breakpoint capability with NOP instruction

Bit 5 0: DM does not generate any event when NOP(SW BRK) instruction is fetched by the CPU

1: DM generates an event (CPU stalled in SWIM mode) when a software break instruction is fetched by the CPU.

**SWBKF** *Software breakpoint status bit (read only)*

This flag is set when the CPU executes the software break instruction.

Bit 4 0: No software break instruction detected.

1: Software break instruction detected. This bit is cleared when the STALL bit is cleared.

**STALL** *CPU stall control bit (read/write only in SWIM mode)*

This bit is used to stall the CPU. This bit is kept cleared if the device is not in SWIM mode.

Bit 3 This bit is set by WOTF command to generate an ABORT equivalent command. It is also set by an DM trap interrupt event.

This bit is cleared by WOTF command to re-start the CPU.

0: CPU runs normally

1: CPU is stalled

Bit: 2:1 Reserved. it must be kept at 0

**FLUSH** *Flush decode*

This bit is set by software to flush the instruction decode phase after a PC modification. It is cleared by hardware when the flush is completed.

Bit: 10

0: Default status

1: Flush decode

#### 4.12.11 DM enable function register (DM\_ENFCTR)

Address: 7F9Ah

Reset Value: 1111 1111 (FFh)

7							0
ENFCT7	ENFCT6	ENFCT5	ENFCT4	ENFCT3	ENFCT2	ENFCT1	ENFCT0
rw	rw	rw	rw	rw	rw	rw	rw

##### **ENFCTx** *Enable function*

Bits 7:0 This bit is set and cleared by software. it allows to freeze a particular function of a peripheral when the core is stalled. The ENFCTx bit definitions are product dependent.

0: Function is frozen when the CPU is stalled by a DM  
1: Function is active

See [Appendix A](#) for a full description of the DM\_ENFCTR register.

## 4.12.12 Summary of SWIM, DM and core register maps

Table 5. STM8 MCU registers

STM8 address	Register name	7	6	5	4	3	2	1	0
7F00h	A Reset value	A7 0	A6 0	A5 0	A4 0	A3 0	A2 0	A1 0	A0 0
7F01h	PCE <sup>(1)</sup>	PC23	PC22	PC21	PC20	PC19	PC18	PC17	PC16
7F02h	PCH <sup>(1)</sup>	PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8
7F03h	PCL <sup>(1)</sup>	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
7F04h	XH Reset value	X15 0	X14 0	X13 0	X12 0	X11 0	X10 0	X9 0	X8 0
7F05h	XL Reset value	X7 0	X6 0	X5 0	X4 0	X3 0	X2 0	X1 0	X0 0
7F06h	YH Reset value	Y15 0	Y14 0	Y13 0	Y12 0	Y11 0	Y10 0	Y9 0	Y8 0
7F07h	YL Reset value	Y7 0	Y6 0	Y5 0	Y4 0	Y3 0	Y2 0	Y1 0	Y0 0
7F08h	SPH <sup>(1)</sup>	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
7F09h	SPL <sup>(1)</sup>	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
7F0Ah	CC Reset value	V 0	- 0	I1 1	H 0	I0 1	N 0	Z 0	C 0
7F80h	SWIM_CSR Reset value	SAFE_MASK 0	NO_ACCESS 0	SWIM_DM 0	HS 0	OSCOFF 0	RST 0	HSIT 0	PRI 0
7F90h	DM_BK1RE Reset value	BK1R23 1	BK1R22 1	BK1R21 1	BK1R20 1	BK1R19 1	BK1R18 1	BK1R17 1	BK1R16 1
7F91h	DM_BK1RH Reset value	BK1R15 1	BK1R14 1	BK1R13 1	BK1R12 1	BK1R11 1	BK1R10 1	BK1R9 1	BK1R8 1
7F92h	DM_BK1RL Reset value	BK1R7 1	BK1R6 1	BK1R5 1	BK1R4 1	BK1R3 1	BK1R2 1	BK1R1 1	BK1R0 1
7F93h	DM_BK2RE Reset value	BK2R23 1	BK2R22 1	BK2R21 1	BK2R20 1	BK2R19 1	BK2R18 1	BK2R17 1	BK2R16 1
7F94h	DM_BK2RH Reset value	BK2R15 1	BK2R14 1	BK2R13 1	BK2R12 1	BK2R11 1	BK2R10 1	BK2R9 1	BK2R8 1
7F95h	DM_BK2RL Reset value	BK2R7 1	BK2R6 1	BK2R5 1	BK2R4 1	BK2R3 1	BK2R2 1	BK2R1 1	BK2R0 1
7F96h	DM_CR1 Reset value	WDGOFF 0	Reserved 0	BC2 0	BC1 0	BC0 0	BIR 0	BIW 0	Reserved 0
7F97h	DM_CR2 Reset value	Reserved					FV_ROM 0	Reserved 0	FV_RAM 0
7F98h	DM_CSR1 Reset value	Reserved 0	STE 0	STF 0	RST 0	BRW 0	BK2F 0	BK1F 0	Reserved 0

Table 5. STM8 MCU registers (continued)

STM8 address	Register name	7	6	5	4	3	2	1	0
7F99h	DM_CSR2 Reset value	Reserved 0	Reserved 0	SWBKE 0	SWBKF 0	STALL 0	Res	Reserved 0	FLUSH 0
7F9Ah	DM_ENFCTR Reset value	ENFCT7 1	ENFCT6 1	ENFCT5 1	ENFCT4 1	ENFCT3 1	ENFCT2 1	ENFCT1 1	ENFCT0 1

1. The reset value for the SP and PC registers is product dependent. Refer to the device datasheet for more details

## Appendix A Description of the DM\_ENFCTR register for each STM8 product

Some peripherals can be frozen through the debug module during the debug and while using the DM\_ENFCTR register (address: 7F9Ah). [Table 6](#) shows the peripherals which are frozen by the bits (ENFCT0 to ENFCT7) of the DM\_ENFCTR register.

**Table 6. Peripherals which are frozen by the bits of the DM\_ENFCTR register for each STM8 product**

DM_ENFCTR register	STM8AF51A STM8S207/208 (128 Kbyte die)	STM8AF616 STM8S105 (32 Kbyte die)	STM8AF51B (256 Kbyte die)	STM8S103/903 (8 Kbyte die)	STM8L101 (8 Kbyte die)	STM8L15x (32 Kbyte die)
Bit	Peripheral					
ENFCT0	Timer4/ System timer	Timer4/ System timer	Timer4/ System timer	Timer4/ System timer	I <sup>2</sup> C	I <sup>2</sup> C
ENFCT1	Timer2	Timer2	Tmer2	Tmer2	Timer2	Timer2
ENFCT2	Timer3	Timer3	Tmer3	Not used	Timer3	Timer3
ENFCT3	Timer1	Timer1	Tmer1	Timer1	- bit not used	Timer1
ENFCT4	- bit not used	- bit not used	- bit not used	- bit not used	Timer4	Timer4
ENFCT5	- bit not used	- bit not used	Tmer5	- bit not used	- bit not used	DMA
ENFCT6	- bit not used	- bit not used	I <sup>2</sup> C2	- bit not used	- bit not used	RTC
ENFCT7	I <sup>2</sup> C	I <sup>2</sup> C	I <sup>2</sup> C	I <sup>2</sup> C	- bit not used	- bit not used

## Revision history

**Table 7. Document revision history**

Date	Revision	Changes
15-Jan-2008	1	Initial release.
10-Dec-2009	2	Updated documentation references in <i>Introduction</i> . <i>Section 3.2: SWIM entry sequence</i> : updated <i>Figure 4</i> and explanation. <i>Section 3.3: Bit format</i> : replaced OBL bit with HSIT bit. Added <i>Appendix A</i> .
06-Jun-2011	3	<i>Introduction</i> : updated titles of reference documents. Updated name of 'IOM' bit to SWIM disable bit (SWD) in <i>Section 3.1: Operating modes, Figure 3</i> , and <i>SWIM control status register (SWIM_CSR)</i> . Updated name of 'MCR' register to 'CFG_GCR' register in <i>Section 3.1: Operating modes</i> .
24-Aug-2016	4	Updated sections: <i>Section 3.2: SWIM entry sequence</i> <i>Section 3.3: Bit format</i> <i>Section 3.6: SWIM communication reset</i> <i>Section 3.8: SWIM communication in Halt mode</i> Updated figures format: <i>Figure 1: Debug system block diagram</i> <i>Figure 2: SWIM pin external connections</i> <i>Figure 3: SWIM activation sequence</i> <i>Figure 5: SWIM entry sequence</i> <i>Figure 6: High speed bit format</i> <i>Figure 7: Low speed bit format</i> <i>Figure 8: Command format (host -&gt; target)</i> <i>Figure 9: Data format (target -&gt; host)</i> <i>Figure 11: Debug module block diagram</i> <i>Figure 12: STM8 MCU instruction model</i> <i>Figure 13: STM8 Debug module stall timing</i> <i>Figure 14: STM8 DM data break timing</i> <i>Figure 15: STM8 DM instruction break timing</i> <i>Figure 16: STM8 DM step timing</i>

**IMPORTANT NOTICE – PLEASE 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 acknowledgement.

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. 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.

© 2016 STMicroelectronics – All rights reserved