

---

**Getting started with the STM8S001J3 microcontroller**

---

**Introduction**

This application note complements the information available in the STM8S001J3 datasheet and in the application note *Getting started with the STM8S and STM8A* (AN2752). It describes the minimum hardware and software environment required to build an application around the STM8S001J3 8-bit microcontroller.

This application note is divided into the following sections:

- power supply
- clock management
- reset control
- analog-to-digital converter (ADC)
- debugging tool support
- recommendations for STM8S001J3 firmware
- pinout functionality overview
- peripherals specific usage and limitations
- STM8 software toolchain

This document also contains some hardware and software recommendations specific to STM8S001J3 devices.

# Contents

- 1      Hardware requirements summary ..... 5**
- 2      Power supply ..... 5**
  - 2.1    Main supply pair ..... 5
  - 2.2    Capacitor on VCAP pin ..... 5
  - 2.3    Power-on / power-down reset (POR/PDR) ..... 5
- 3      Clock management ..... 6**
  - 3.1    Clock management overview ..... 6
  - 3.2    Internal clocks ..... 6
  - 3.3    External clock ..... 6
- 4      Reset management overview ..... 6**
- 5      Analog-to-digital converter (ADC) ..... 7**
  - 5.1    Analog power ..... 7
  - 5.2    Analog input ..... 7
- 6      Debugging support ..... 8**
  - 6.1    SWIM (single wire interface module) overview ..... 8
  - 6.2    Recommendations for reliable SWIM connection ..... 8
    - 6.2.1    Program memory should contain always a valid program loop ..... 8
    - 6.2.2    Delay before configuration of pin with SWIM functionality ..... 9
- 7      Recommendation for STM8S001J3 firmware ..... 10**
  - 7.1    Set unbonded GPIOs to safe state ..... 10
  - 7.2    Delay before SWIM pin reconfiguration ..... 10
  - 7.3    Proper GPIOs modes programming ..... 10
  - 7.4    Example of recommended startup code ..... 10
- 8      Pinout functionality overview ..... 12**
- 9      Peripherals specific usage ..... 13**

---

9.1	UART1 .....	13
9.2	I2C .....	13
9.3	SPI .....	14
9.4	Oscillator (OSCIN input) .....	14
9.5	ADC .....	14
9.6	Timers (TIM1, TIM2) .....	14
9.7	SWIM .....	14
<b>10</b>	<b>STM8 software toolchain .....</b>	<b>15</b>
10.1	Integrated development environment .....	15
10.1.1	ST toolset: STVD, STVP .....	15
10.1.2	IAR embedded workbench for STM8 .....	15
10.1.3	RIDE-STM8 software development environment .....	15
10.2	Compiler .....	16
10.3	Firmware library .....	16
<b>11</b>	<b>Documentation and online support .....</b>	<b>17</b>
<b>12</b>	<b>Revision history .....</b>	<b>19</b>

## List of figures

Figure 1.	Default while (1) code program-memory content .....	9
Figure 2.	STM8S001J3 pinout .....	13
Figure 3.	STM8 firmware library examples .....	17

# 1 Hardware requirements summary

In order to build an application around the STM8S001J3, the application board should provide at least the following features:

- Power supply
- Clock management
- Reset management
- Debugging tool support, single wire interface module (SWIM) connector.

## 2 Power supply

STM8S001J3 devices can be supplied through a 3.0 V to 5.5 V external source.

An on-chip power management system provides the 1.8 V digital supply to the core logic, both in normal and low-power modes. This power management system is also capable of detecting voltage drops, on both main external (3.3 V/5 V) and internal (1.8 V) supplies.

### 2.1 Main supply pair

The STM8S001J3 includes one pair of supply pins, VDD/VSS (3.3 V  $\pm$  0.3 V to 5 V  $\pm$  0.5 V) dedicated to the core regulator supply, the power of I/O pins and the supply of the analog functions.

The power supply pair VDD/VSS should be decoupled by a filtering ceramic capacitor (100 nF) with a chemical capacitor (1 to 2  $\mu$ F) in parallel. The ceramic capacitor should be placed as close as possible to the supply pins.

### 2.2 Capacitor on VCAP pin

Stabilization of the main regulator is achieved by connecting an external capacitor, to the VCAP pin. Please refer to the STM8S001J3 device datasheet for more information about C<sub>EXT</sub> characteristics.

This external capacitor is charged during the device power-on-reset (POR) and the external power supply must provide current peak without V<sub>DD</sub> voltage decrease below POR level (it is recommended to use 10  $\mu$ F chemical capacitor on VDD/VSS for weaker power supply).

### 2.3 Power-on / power-down reset (POR/PDR)

The input supply is monitored by a power-on / power-down reset circuit. The monitoring voltage range is 0.7 V to 2.7 V.

During power-on, the POR/PDR keeps the device under reset until the supply voltage reaches their specified working area. The upper threshold for a reset release is defined in the electrical characteristics section of the product datasheet. An hysteresis is implemented (POR > PDR) to ensure clean detection of voltage rise and fall.

The POR/PDR also generates a reset when the supply voltage drops below the  $V_{\text{POR/PDR}}$  threshold.

## 3 Clock management

### 3.1 Clock management overview

STM8S001J3 devices offer a flexible way of selecting the core and peripheral clocks (ADC, memory, digital peripherals). These devices have internal and external clock source inputs and one output clock (CCO).

### 3.2 Internal clocks

The embedded RC oscillator has an internal capacitor (C) and an internal resistor ladder (R).

STM8S001J3 has two kinds of internal clock:

- a high speed internal clock (HSI) running at 16 MHz
- a low speed internal clock (LSI) running at 128 kHz.

After reset, the CPU starts with the internal RC (HSI clock signal) divided by 8, which means 2 MHz.

### 3.3 External clock

STM8S001J3 devices support connection to an external oscillator only. Connection to an external crystal/resonator is not available.

To use the external oscillator input, the option bit EXTCLK must be set (factory default).

*Note:* while no external clock is used, OSCIN pin can be used as general purpose I/O pin.

## 4 Reset management overview

STM8S001J3 microcontrollers have no external reset signal (NRST pin). The initial reset comes from power-on reset (POR).

The internal reset sources are:

- power-on reset (POR) and brown-out reset (BOR)  
During power-on, the POR keeps the device under reset until the supply voltage (VDD) reaches the voltage level at which the BOR starts to function.
- independent watchdog reset (IWDG)
- window watchdog reset (WWDG)
- software reset  
The application software can trigger reset.
- SWIM reset  
An external device connected to the SWIM interface can request the SWIM block to generate a microcontroller reset.
- illegal opcode reset  
If a code to be executed does not correspond to any opcode or pre-byte value, a reset is generated.
- electromagnetic susceptibility (EMS) reset  
Generated if critical registers are corrupted or badly loaded.

## 5 Analog-to-digital converter (ADC)

### 5.1 Analog power

The ADC unit uses common analog supply and reference voltage with digital power supply  $V_{DD}$ .

### 5.2 Analog input

The STM8S001J3 has three analog input channels (AIN2, AIN4/AIN5, AIN6), which are converted by the ADC one at a time. Each analog input is multiplexed with an I/O (analog inputs AIN4 and AIN5 share the same pin).

To activate the analog input AIN2, it is necessary to set the alternate function bit AFR2.

Please refer to the STM8S001J3 device datasheet and reference manual for more details.

## 6 Debugging support

### 6.1 SWIM (single wire interface module) overview

The in-circuit debugging and programming modes, are managed through a single-wire hardware interface based on an open-drain line, featuring ultra-fast memory programming. Coupled with an in-circuit debugging module, the SWIM offers also a non-intrusive read/write to RAM and peripherals. This makes the in-circuit debugger extremely powerful and close in performance to a full-featured emulator.

The SWIM pin can be used as a standard I/O, with some restrictions if the user wants to use it for debugging.

Please refer to the user manual, *STM8 SWIM communication protocol and debug module* (UM0470), for more details on the SWIM protocol.

### 6.2 Recommendations for reliable SWIM connection

#### 6.2.1 Program memory should contain always a valid program loop

For reliable connection of an STM8S001J3 device through the SWIM interface, it is recommended to avoid to erase completely the program-memory (for example by mass erase: set and remove RDP protection).

Another recommendation is that the new loaded code to execute, is always a valid program loop, such as while (1) code. Factory default program-memory content for STM8S001J3 devices is while (1) code (see [Figure 1](#)).

Explanation:

If the program-memory is empty (0x00 content), the device behavior is described below:

1. After the power-on, the empty code is executed (0x0000 opcodes = instructions NEG (0x00, SP)) until the PC counter reaches the end of the 8-Kbyte program-memory (end-address 0x9FFF). It takes around 4 ms to reach the end of the 8-Kbyte memory space, with HSI clock at 2 MHz.
2. Once the end of the 8-Kbyte program memory is reached, the program continues and code from a non-existing memory is fetched and executed. The reading of non-existing memory is a random content, which can lead to the execution of invalid instructions. The execution of invalid instructions generates a software reset and the program starts again. In the worst case, a reset is generated every 4 ms.
3. Only the connect-on-the-fly method can be used to program the device through the SWIM interface. The connect-under-reset method cannot be used because the reset pin (NRST) is not available on this device.
4. The connect-on-the-fly mode can be used while the device is executing code, but if there is a device reset (internal reset) during the SWIM connection, this connection is aborted and must be performed again from the debug tool. Note that the software reset occurrence can be of every 4 ms, making difficult to successfully connect to the debug tool (usually only one successful connection trial for every 10 attempts).
5. Once a successful connection is reached, the device can be programmed with a valid firmware without problems.



Figure 1 provides an example of while (1) code loaded into the device (factory default memory content) - The reset vector is pointing to reset vector.

Figure 1. Default while (1) code program-memory content

Address	Hex Value	OpCode	Comment
0x8000	0x82008000	INT	0x008000 INT 0x008000
0x8004	0x0000	NEG	(0x00, SP)
0x8006	0x0000	NEG	(0x00, SP)
0x8008	0x0000	NEG	(0x00, SP)
0x800a	0x0000	NEG	(0x00, SP)
0x800c	0x0000	NEG	(0x00, SP)

### 6.2.2 Delay before configuration of pin with SWIM functionality

If the application uses another function on the SWIM pin 8 (for example GPIO output), it is recommended to add a delay of around 5 s in the firmware, before changing the functionality of the pin. This action allows the user to set the device into SWIM mode, after the device powers on (during this delay), and to reprogram the device.

As the reset pin (NRST) is not available on STM8S001J3 devices, the connect-under-reset method is not available. If the SWIM pin 8 is set to I/O mode immediately after the device reset, the device is unable to connect through the SWIM interface and it gets locked forever.

This initial delay can be removed in the final/locked code (early I/O pin functionality is usually required).

## 7 Recommendation for STM8S001J3 firmware

The startup-code for STM8S001J3 devices has to be configured following the recommendations listed below.

### 7.1 Set unbonded GPIOs to safe state

GPIOs PA2, PB0, PB1, PB2, PB3, PB6, PB7, PC1, PC2, PC7, PD0, PD2, PD4, PD7, PE5 and PF4 are present on device only internally: they are not bonded to pins.

After the device reset, those GPIOs are configured by hardware into input-floating mode. It is recommended to set them into output-mode/low-output-level state. EMC immunity is then improved (GPIOs are grounded) and device power consumption is lower (no I/O toggling on floating inputs).

### 7.2 Delay before SWIM pin reconfiguration

To be able to debug the device through the SWIM interface, it is recommended to add a delay of around 5 s in the firmware startup code. This delay can be removed in final/locked code (see an example of delay code in [Section 7.4](#)).

The SWIM pin functionality is kept during this delay. The user can connect to the device through the SWIM interface and reprogram the device. See [Section 6.2.2: Delay before configuration of pin with SWIM functionality on page 9](#).

If the firmware does not implement this delay, and immediately reconfigures some GPIOs on the pin 8 into output mode, it is then not possible to connect to the device through the SWIM interface (because this pin is now an output). The device is locked permanently (because firmware cannot be changed).

### 7.3 Proper GPIOs modes programming

STM8S001J3 devices have several pins providing a connection to multiple GPIOs. The mode selected for any of these GPIOs impacts all the other GPIOs connected to the same pin. The proper setting of the GPIO modes is important to avoid conflicts between GPIOs bonded to the same pin (including their alternate functions).

Example: PD1, PC6, PD3 and PD5 GPIOs are on pin 8. Pull-up enabled on PD1 is also seen on PC6, PD3 and PD5.

### 7.4 Example of recommended startup code

```
/* MAIN.C file */

#include "stm8s.h"
#include "stm8s_gpio.h"
#include "stm8s_clk.h"

#ifdef _COSMIC_
```

```

#define ASM _asm
#endif
#ifdef _IAR_
#define ASM asm
#endif
/* This delay should be added just after reset to have access to SWIM pin
and to be able to reprogram the device after power on (otherwise the device
will be locked) */
#define STARTUP_SWIM_DELAY_5S \
{ \
ASM("    PUSHW X    \n" \
"    PUSH  A    \n" \
"    LDW X, #0xFFFF \n" \
"loop1: LD  A, #50  \n" \
\
"loop2: DEC A    \n" \
"    JRNE loop2  \n" \
\
"    DECW X    \n" \
"    JRNE loop1  \n" \
\
"    POP  A    \n" \
"    POPW X    " ); \
}
/* not connected pins as output low state (the best EMC immunity)
(PA2, PB0, PB1, PB2, PB3, PB6, PB7, PC1, PC2, PC7, PD0, PD2, PD4, PD7, PE5,
PF4) */
#define CONFIG_UNUSED_PINS_STM8S001 \
{ \
GPIOA->DDR |= GPIO_PIN_2; \
GPIOB->DDR |= GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 |
GPIO_PIN_6 | GPIO_PIN_7; \
GPIOC->DDR |= GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_7; \
GIOD->DDR |= GPIO_PIN_0 | GPIO_PIN_2 | GPIO_PIN_4 | GPIO_PIN_7; \
GPIOE->DDR |= GPIO_PIN_5; \
GPIOF->DDR |= GPIO_PIN_4; \
}
/* pin for testing */
#define TEST_PORT GPIOA
#define TEST_PIN  GPIO_PIN_3
// #define HSE_TEST

/* Example of firmware for STM8S001: recommended startup + test of pins
functionality */
main()
{
/* -----STM8S001 startup----- */
/* configure unbonded pins */
CONFIG_UNUSED_PINS_STM8S001;
/* delay for SWIM connection: ~5seconds */
STARTUP_SWIM_DELAY_5S;

```

```

/* ----- */

/* configure all STM8S001 pins as input with pull up */
GPIO_Init(GPIOA, GPIO_PIN_1, GPIO_MODE_IN_PU_NO_IT); // pin 1
GPIO_Init(GPIOA, GPIO_PIN_3, GPIO_MODE_IN_PU_NO_IT); // pin 5
GPIO_Init(GPIOB, GPIO_PIN_4, GPIO_MODE_OUT_PP_LOW_FAST); // pin 6 (PB4
has no pull-up - configure it as output low)
GPIO_Init(GPIOC, GPIO_PIN_3, GPIO_MODE_IN_PU_NO_IT); // pin 7
GPIO_Init(GPIOC, GPIO_PIN_6, GPIO_MODE_IN_PU_NO_IT); // pin 8

/* disable peripherals clocks to decrease consumption */
CLK->PCKENR1 = 0x00;
CLK->PCKENR2 = 0x00;

/* for testing external HSE clock */
/* make sure that option bit EXTCLK=1 */
#ifdef HSE_TEST
/* test HSE (external clock) - apply input clock on PA1 (pin 1) */
GPIO_Init(GPIOC, GPIO_PIN_4, GPIO_MODE_OUT_PP_LOW_FAST); // CCO is on PC4
(pin 7)
CLK_CCOClockConfig(CLK_OUTPUT_CPU); // clock out on PC4/CCO (pin 7)
CLK_ClockSwitchConfig(CLK_SWITCHMODE_AUTO, CLK_SOURCE_HSE, DISABLE,
CLK_CURRENTCLOCKSTATE_DISABLE); //set HSE as clock
#endif //HSE_TEST

/* initialize tested pin */
GPIO_Init(TEST_PORT, TEST_PIN, GPIO_MODE_OUT_PP_LOW_FAST);
while (1)
{
    /* toggle with tested pin */
    GPIO_WriteReverse(TEST_PORT, TEST_PIN);
}
}

```

## 8 Pinout functionality overview

[Figure 2](#) and [Table 1](#) show an overview of STM8S001J3 devices pins and their peripherals functionality for each pin (SO8N package).

One pin is usually shared for multiple GPIOs (multiple GPIOs internally connected to the same pin). Only one functionality should be programmed at one time for a given pin. See device datasheet for more information.

Figure 2. STM8S001J3 pinout

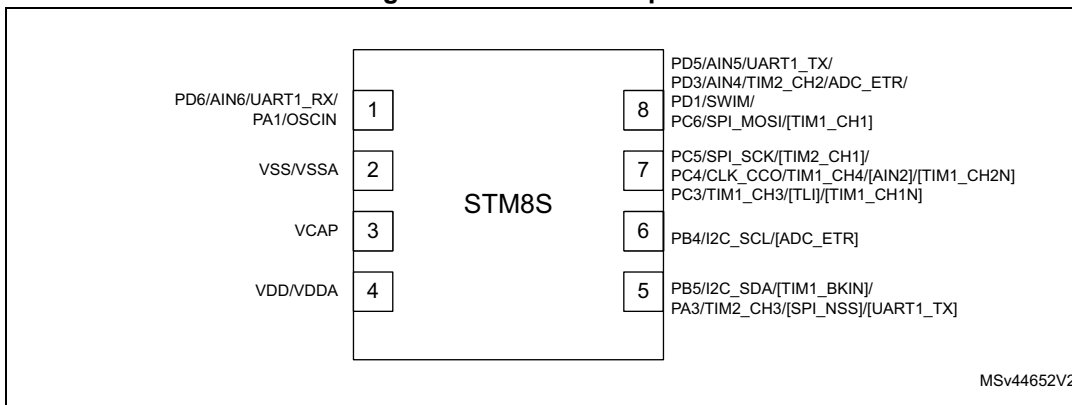


Table 1. STM8S001J3 alternate functions overview

Peripheral	Pin number							
	1	2	3	4	5	6	7	8
UART	RxD	-	-	-	TxD	-	-	TxD
I2C	-	-	-	-	SDA	SCL	-	-
ADC	AIN6	-	-	-	-	ETR	AIN2	AIN5/ETR
TIM1	-	-	-	-	BKIN	-	CH1N/CH2N/CH3/CH4	CH1
TIM2	-	-	-	-	CH3	-	CH1	CH2
OSC	OSCIN	-	-	-	-	-	CCO	-
Supply	-	VSS	VCAP	VDD	-	-	-	-
Debug	-	-	-	-	-	-	-	SWIM
SPI	-	-	-	-	-	-	SCK	MOSI

## 9 Peripherals specific usage

STM8S001J3 devices are packed in 8-pin SO8N package. Due to multiple GPIOs connected to the same pin, there are several specific peripherals usage described below.

### 9.1 UART1

RxD is on pin 1. TxD is on pin 8 (can be remapped on pin 5).

To remap TxD on pin 5, both alternate function bits AFR0 and AFR1 have to be set to one.

### 9.2 I2C

SCL is on pin 6. SDA is on pin 5.

SDA pin is not a true open-drain pin but only pseudo open-drain, because pin 7 is shared with another GPIO (on which is implemented protection diode to  $V_{DD}$ ).

### 9.3 SPI

SCK is on pin 7. MOSI is on pin 8.

Due to the missing MISO signal, SPI must be used only in one from the following operation modes:

- unidirectional transfer (master mode = transmit/receive only, slave mode = receive only)
- simplex master synchronous transfers on two lines with a possible bidirectional data.

### 9.4 Oscillator (OSCIN input)

An external oscillator can be connected on pin 1 (OSCIN) for precise HSE clock input.

HSE specific usage:

- It is impossible to connect an external crystal.
- The option bit EXTCLK must be set to one, for external oscillator input (factory default for STM8S001J3 devices).

### 9.5 ADC

ADC input channels (AIN6, AIN2 and AIN4/AIN5) are on pins 1, 7 and 8. The external ADC trigger (ADC\_ETR) is on pin 6.

ADC specific usage:

- The scan mode can be used but dummy conversions are performed for missing ADC channels.
- To enable AIN2 channel input, the alternate function remapping bit AFR2 must be set to one.

### 9.6 Timers (TIM1, TIM2)

The timer channels inputs/outputs are on pins 5, 7 and 8. The timer break input (TIM1\_BKIN) is on pin 5.

The timer channels must be carefully programmed, to avoid collision with another timer channel output (only one output timer channel on a given pin).

### 9.7 SWIM

SWIM is on pin 8. The PD1/SWIM pin has an internal pull-up active, after device reset.

The pin 8 can be used by another output function, that disables the SWIM connection to the device by the debugger (see [Figure 2](#) and [Table 1](#)). During code development, it is recommended to implement a delay of around 5 s delay in the firmware, after the device reset (power-on) and afterwards to set pin 8 to the required output function. The SWIM

function has to be disabled by software (SWD bit in CFG\_GCR register) if another functionality is used on pin 8 (see [Section 7.2: Delay before SWIM pin reconfiguration on page 10](#) for more details).

## 10 STM8 software toolchain

In order to write, compile and run the first software on STM8S001J3 devices, the following components of the software toolchain are required:

- integrated development environment
- compiler
- firmware library (optional, used to ease the start-up).

For more detailed information please refer to application note *Getting started with the STM8S and STM8A* (AN2752).

### 10.1 Integrated development environment

#### 10.1.1 ST toolset: STVD, STVP

The integrated development environment ST visual develop (STVD) provides an easy-to-use and efficient environment for start-to-finish control of the application development, from building and debugging the application code to programming the microcontroller. STVD is delivered as part of the free ST toolset, which also includes the ST visual programmer (STVP) programming interface and the ST assembler linker.

To build the application, STVD provides seamless integration of C and assembly toolchains, including Cosmic and Raisonance C compilers, and the ST assembler linker. When debugging, STVD provides an integrated simulator (software) and supports a complete range of hardware tools including the low-cost ST-LINK in-circuit debugger/programmer.

For applications based on STM8S001J3 device, STVD also provides an interface for reading, writing to and verifying the microcontroller memories. This interface is based on STVP and supports STM8S001J3 device and programming tools.

The free ST toolset for STM8 is available from STMicroelectronics homepage [www.st.com](http://www.st.com).

#### 10.1.2 IAR embedded workbench for STM8

The IAR embedded workbench IAR-EWSTM8 is a software development tool, providing full support for devices in STM8AF, STM8AL, STM8L, and STM8S Series.

This product is supplied by a third party not affiliated to STMicroelectronics. For complete and latest information on the specification and the purchased parts package, refer to the IAR Systems website.

#### 10.1.3 RIDE-STM8 software development environment

RIDE-STM8 integrates the Raisonance C Compiler for STM8 (RKit-STM8 installation).

This product is supplied by a third party not affiliated to STMicroelectronics. For complete and latest information on the specification and the purchased parts package, refer to the Raisonance website.

## 10.2 Compiler

STM8S001J3 devices can be programmed by a free assembler toolchain included in the ST toolset. As the core is designed for optimized high-level-language support, use of a C compiler is recommended.

C compilers for STM8 are offered by the third party companies IAR Systems, Cosmic and Raisonance.

A free version of these C compilers, with limited size of generated code, is available at third party websites.

## 10.3 Firmware library

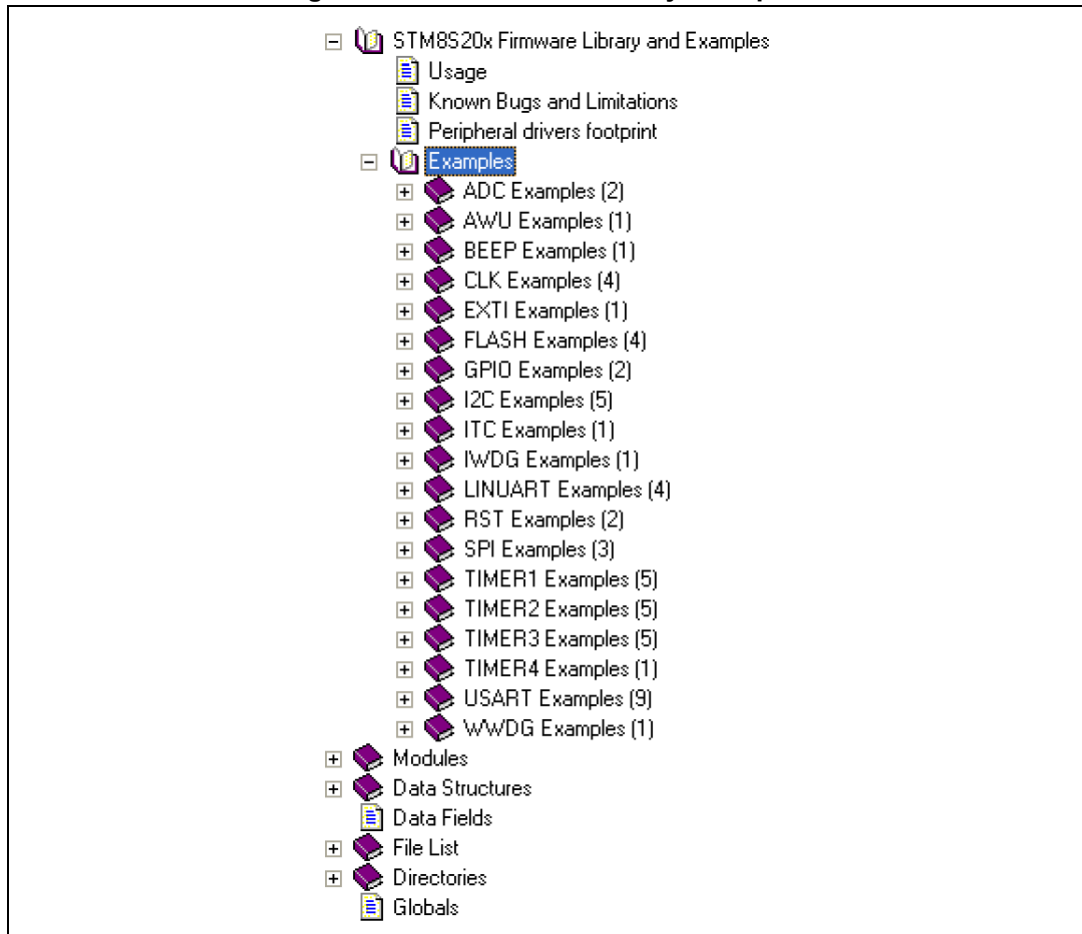
The STM8 firmware library is a complete set of source code examples for each STM8 peripheral, written in strict ANSI-C and fully MISRA C 2004 compliant (see [Figure 3](#)).

All examples are delivered with workspace and project definition files for STVD and Cosmic C compiler, enabling the user to load and compile them easily into the development environment. The examples run on STMicroelectronics STM8 evaluation boards and can be tailored easily to other types of hardware.

For additional information on STM8 firmware library, refer to STMicroelectronics homepage [www.st.com](http://www.st.com).



Figure 3. STM8 firmware library examples



## 11 Documentation and online support

Documentation resources related to STM8S001J3 device are listed below:

### Application

- STM8S001J3 datasheet
- *STM8S001J3, STM8S003xx, STM8S103xx and STM8S903xx device limitations* (ES0102)
- reference manual *STM8S series and STM8AF series 8-bit microcontrollers* (RM0016)
- programming manual *How to program STM8S and STM8A Flash program memory and data EEPROM* (PM0051)
- *STM8 CPU programming manual* (PM0044)

**Tools**

- STM8 firmware library and release note (detailed descriptions of the library are included as help files)
- ST visual develop tutorial (included as help files in the ST-toolchain)
- user manual *ST visual develop (STVD)* (UM0036)
- user manual *STM8 SWIM communication protocol and debug module* (UM0470)
- Cosmic C compiler user manual
- IAR embedded workbench development guide.

The microcontroller forum on [community.st.com](http://community.st.com) allows developers to exchange ideas. It is the best place to find different applications. In addition, the website has a knowledge base of FAQs for microcontrollers, providing answers to many queries and solutions to many problems.

## 12 Revision history

Table 2. Document revision history

Date	Revision	Changes
29-Jun-2017	1	Initial release.

**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.

© 2017 STMicroelectronics – All rights reserved