AN5065
Application note

Getting started with STM8L001J3 and STM8L050J3 microcontrollers

Introduction

This application note complements the information available in the STM8L001J3 and STM8L050J3 datasheets and in the application note *Getting started with STM8L and STM8AL* (AN3029). It describes the minimum hardware and software environment required to build an application around the STM8L001J3 and STM8L050J3 8-bit microcontrollers.

This application note is divided into the following sections:

- power supply
- clock management
- reset management overview
- analog-to-digital converter (ADC)
- comparators
- debugging support
- recommendations for STM8L001J3 and STM8L050J3 firmware
- pinout functionality overview
- STM8 software toolchain

This document also contains some hardware and software recommendations specific to STM8L001J3 and STM8L050J3 devices.

Contents

List of tables

List of figures

1 Hardware requirements summary

In order to build an application around STM8L001J3 and STM8L050J3 devices, the application board should provide at least the following features:

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

2 Power supply

STM8L001J3 and STM8L050J3 devices can be supplied through a 1.8 V to 3.6 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 power supply.

2.1 Main supply pair

STM8L001J3 and STM8L050J3 devices include one pair of supply pins: VDD and VSS (1.8 V to 3.6 V). These supply pins are dedicated to the core regulator supply, to the I/O pins supply and to the supply of the analog functions.

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

2.2 Power-on reset / power-down reset / brown-out reset (POR/PDR/BOR)

The input supply is monitored by a power-on / power-down / brown-out reset circuit. The monitoring voltage range is 1.35 V to 2.90 V.

During power-on, the POR/PDR/BOR 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/BOR also generates a reset when the supply voltage drops below the $V_{POR/PDR/BOR}$ threshold.

3 Clock management

STM8L001J3 and STM8L050J3 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).

3.1 Internal clocks

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

The STM8L001J3 and STM8L050J3 have two kinds of internal clock:

- a high speed internal clock (HSI) running at 16 MHz
- a low speed internal clock (LSI) running at ~38 kHz.

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

3.2 External clock

STM8L050J3 devices support two kinds of external clock:

- a high speed external clock (HSE) up to 16 MHz
- a low speed external clock (LSE) running typically at 32.768 kHz.

Both HSE and LSE clock support a connection to an external oscillator or an external crystal/resonator.

STM8L001J3 devices do not support connection to an external oscillator or a crystal/resonator.

Note: While LSE or HSE external clock is not used, given oscillator pins can be used as general purpose I/O pins.
Simultaneous usage of HSE and LSE clocks is possible only in case of LSE bypass mode (external LSE oscillator connected to OSC32_IN pin) because OSCIN and OSC32_OUT signals are connected to the same pin.

4 Reset management overview

STM8L001J3 and STM8L050J3 microcontrollers do not have an 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)

  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.

5 Analog-to-digital converter (ADC)

The STM8L050J3 offers one 12-bit ADC (analog-to-digital converter) with multiplexed analog input channels. The STM8L001J3 does not offer an 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

STM8L050J3 has four analog input channels (ADC1_IN4, ADC1_IN11, ADC1_IN12, ADC1_IN13/IN15/IN22), which are converted by the ADC one at a time. Each analog input is multiplexed with digital I/O signals. ADC1_IN13, ADC1_IN15 and ADC1_IN22 analog inputs share the same pin.

Refer to the STM8L050J3 device datasheet and reference manual for more details.

6 Comparators

STM8L001J3 and STM8L050J3 devices support two comparators.

STM8L001J3 devices have two comparators with shared inputs (the same comparing levels for both comparators). Comparators outputs can be connected to TIM2 (input capture / break) or TIM3 (input capture) inputs. Threshold reference level can be selected for both comparators either from GPIO pin or from internal ground reference.

STM8L050J3 devices have two independent comparators (COMP1, COMP2).

COMP1 properties are:

- fixed threshold (internal reference voltage)
- no rail-to-rail inputs
- 4 input channels
- interrupt with wakeup from Halt capability

COMP2 properties are:

- selectable threshold (internal reference voltage submultiple or GPIO pin)
- rail-to-rail inputs
- 1 input channel
- interrupt with wakeup from Halt capability
- output connection to timers

Refer to STM8L001J3 and STM8L050J3 devices datasheets and reference manual for more details.

7 Debugging support

The STM8L001J3 and STM8L050J3 can be debugged and programmed by SWIM hardware interface.

7.1 Single wire interface module (SWIM) overview

The in-circuit debugging and programming modes are managed through a single-wire hardware interface, This interface is based on an open-drain line and features an ultra-fast memory programming. The SWIM also offers a non intrusive read/write to RAM and peripherals when coupled with an in-circuit debugging module. These features make 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.

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

7.2 Recommendations for a reliable SWIM connection

STM8L001J3 and STM8L050J3 devices do not have an NRST pin and the SWIM function is shared with multiple GPIOs functionalities on the same pin. Therefore there are some recommendations to follow in order to have a reliable SWIM connection.

For a reliable SWIM connection the microcontroller must always execute a valid program loop and there must be a delay before the configuration of a pin with the SWIM functionality. This section explain in detail these two recommendations.

7.2.1 Valid program loop for the program memory

On STM8L001J3: for a reliable connection through the SWIM interface it is recommended not to completely erase the program-memory (for example by mass erase: set and remove RDP protection).

On STM8L050J3: the program-memory can be totally erased without an impact to connection reliability through the SWIM interface because STM8L050J3 device has an embedded bootloader which is executed in case of an empty memory detection.

For both devices, the loaded and executed code should always be a valid program loop. This means that the code never continues through the end of the memory. Factory default program-memory content for STM8L001J3 devices is while (1) code (see *Figure 1*).

Explanation:

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

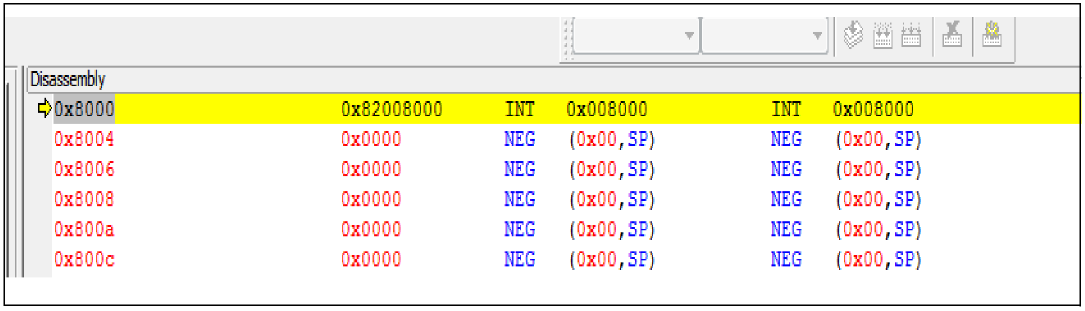
1. After the power-on, the empty code is executed (0x0000 opcodes = instructions NEG (0x00, SP)) until the PC counter reaches the end of the 8-Kbyte program-memory (end-address 0x9FFF). It takes around 4 milliseconds to reach the end of the 8-Kbyte memory space with HSI clock at 2 MHz.
2. Once the end 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



```
0x8000     0x82008000   INT   0x008000     INT   0x008000
0x8004     0x0000       NEG   (0x00,SP)    NEG   (0x00,SP)
0x8006     0x0000       NEG   (0x00,SP)    NEG   (0x00,SP)
0x8008     0x0000       NEG   (0x00,SP)    NEG   (0x00,SP)
0x800a     0x0000       NEG   (0x00,SP)    NEG   (0x00,SP)
0x800c     0x0000       NEG   (0x00,SP)    NEG   (0x00,SP)
```

7.2.2 Apply a delay before configuring a pin with the SWIM functionality

If the application uses another function on the SWIM pin 1 (for example GPIO output), it is recommended to add a delay of around 5 seconds 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 neither on STM8L001J3 nor on STM8L050J3 devices, the connect-under-reset method is not available. If the SWIM pin 1 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 (an early I/O pin functionality is usually required).

8 Recommendation for STM8L001J3 and STM8L050J3 firmware

In order to successfully configure the startup code for STM8L001J3 and STM8L050J3 devices the user must:

- Set unbonded GPIOS to safe state
- Apply a delay before reconfiguring the SWIM pin
- Set the proper modes for GPIOs

This chapter describes in detail the recommendations listed above and presents an startup code example for reference.

8.1 Setting unbonded GPIOS to safe state

On STM8L001J3: GPIOs PA1, PA3, PA5, PB0, PB1, PB2, PB4, PC5, PC6, PD1, PD2, PD3, PD4, PD5, PD6 and PD7 are only present internally; they are not bonded to pins.

On STM8L050J3: GPIOs PA1, PB0, PB1, PB2 and PB4 are only present 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 in order to improve the EMC immunity (as GPIOs are grounded) and to lower the device's power consumption (as there are no input GPIOs with undefined logic level).

8.2 Applying a delay before the SWIM pin reconfiguration

It is recommended to add a delay of around 5 seconds in the firmware startup code in order to be able to debug the device through the SWIM interface. This delay can be removed in final/locked code (see an example of delay code in *Section 8.4*).

The SWIM pin functionality is kept during this delay. The user can connect to the device though the SWIM interface and reprogram the device. See *Section 7.2.2: Apply a delay before configuring a pin with the SWIM functionality on page 11*.

If the firmware does not implement this delay and immediately reconfigures some GPIOs on the pin 1 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).

If the initial delay is not acceptable for the application there is the option that the firmware reenables the SWIM pin functionality under specific conditions.

Some examples on how to manage the reenabling on the SWIM pin are listed below:

- The firmware keeps the SWIM pin functionality during startup if a dedicated pin is grounded during startup (or some pins combinations are in a specified state).
- If some communication interface is used in the application, a specific command can be implemented, after which the SWIM pin functionality is enabled and the debugger can connect to device.
- The application can implement a simple bootloader and update the firmware through a communication interface (instead of via the SWIM interface).

8.3 Setting the proper modes for GPIOs

STM8L001J3 and STM8L050J3 devices have several pins providing a connection to multiple GPIOs. The selected mode 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).

Example1 (STM8L001J3): PA0, PC3 and PC4 GPIOs are on pin 1. Pull-up enabled on PA0 is also seen on PC3 and PC4.
Example2 (STM8L050J3): PA0, PA2 and PC6 GPIOs are on pin 1. Pull-up enabled on PA0 is also seen on PA2 and PC6.

8.4 Example of recommended startup code

This section presents examples of recommended startup codes for STM8L001J3 and STM8L050J3.

Example of recommended startup code for STM8L001J3

```
/* MAIN.C file */

#include "stm8l10x.h"
#include "stm8l10x_gpio.h"
#include "stm8l10x_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)
(PA1, PA3, PA5, PB0, PB1, PB2, PB4, PC5, PC6, PD1, PD2, PD3, PD4, PD5,
 PD6, PD7)*/
#define CONFIG_UNUSED_PINS_STM8L001                                      \
{                                                                        \
  GPIOA->DDR |= GPIO_Pin_1 | GPIO_Pin_3 | GPIO_Pin_5;                    \
  GPIOB->DDR |= GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_4;       \
  GPIOC->DDR |= GPIO_Pin_5 | GPIO_Pin_6;                                 \
  GPIOD->DDR |= GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 |      \
                GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7;                    \
}
/* pin for testing */
#define TEST_PORT GPIOA
#define TEST_PIN  GPIO_Pin_2

/* Example of firmware for STM8L001: recommended startup + test of pins
   functionality */
main()
{
  uint16_t i;
  /* -------------STM8L001 startup-------------- */
  /* configure unbonded pins */
  CONFIG_UNUSED_PINS_STM8L001;
  /* delay for SWIM connection: ~5seconds */
  STARTUP_SWIM_DELAY_5S;
  /* ------------------------------------------- */

  /* configure all STM8L001 pins as input with pull up */
  GPIO_Init(GPIOA, GPIO_Pin_0, GPIO_Mode_In_PU_No_IT);      // pin 1
  GPIO_Init(GPIOA, GPIO_Pin_2, GPIO_Mode_In_PU_No_IT);      // pin 2
  GPIO_Init(GPIOD, GPIO_Pin_0, GPIO_Mode_In_PU_No_IT);      // pin 5
  GPIO_Init(GPIOB, GPIO_Pin_6, GPIO_Mode_In_PU_No_IT);      // pin 6
  GPIO_Init(GPIOB, GPIO_Pin_7, GPIO_Mode_In_PU_No_IT);      // pin 7
  GPIO_Init(GPIOC, GPIO_Pin_2, GPIO_Mode_In_PU_No_IT);      // pin 8

  /* initialize tested pin */
  GPIO_Init(TEST_PORT, TEST_PIN, GPIO_Mode_Out_PP_Low_Fast);
  /* initialize UART */
  GPIO_Init(GPIOC, GPIO_Pin_3, GPIO_Mode_Out_PP_High_Fast); // TxD: output
  CLK_PeripheralClockConfig(CLK_Peripheral_USART, ENABLE);
  USART_Init(9600, USART_WordLength_8D, USART_StopBits_1, USART_Parity_No,
             (USART_Mode_TypeDef)(USART_Mode_Rx | USART_Mode_Tx));
  USART_Cmd(ENABLE);
  /* initialize TIM3 */
  GPIO_Init(GPIOD, GPIO_Pin_0, GPIO_Mode_Out_PP_Low_Fast); // CH2: output
  CLK_PeripheralClockConfig(CLK_Peripheral_TIM3, ENABLE);
  TIM3_OC2Init(TIM3_OCMode_Toggle,
               TIM3_OutputState_Enable,
               9000,
               TIM3_OCPolarity_High,
               TIM3_OCIdleState_Reset);
  TIM3_Cmd(ENABLE);
  TIM3_CtrlPWMOutputs(ENABLE);

  while (1)
  {
    /* toggle with tested pin */
    GPIO_ToggleBits(TEST_PORT, TEST_PIN);
    /* send data to UART */
    USART_SendData8(0x55);
    /* delay */
    for(i=0; i<10000; i++);
  }
}
```

Example of recommended startup code for STM8L050J3

```
/* MAIN.C file */

#include "stm8l15x.h"
#include "stm8l15x_gpio.h"
#include "stm8l15x_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)
(PA1, PB0, PB1, PB2, PB4)*/
#define CONFIG_UNUSED_PINS_STM8L050                                      \
{                                                                        \
  GPIOA->DDR |= GPIO_Pin_1;                                              \
  GPIOB->DDR |= GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_4;       \
}
/* pin for testing */
#define TEST_PORT GPIOA
#define TEST_PIN  GPIO_Pin_3

uint16_t ADCvalue = 0;

/* Example of firmware for STM8L050: recommended startup + test of pins
   functionality */
main()
{
  uint16_t i;
  /* -------------STM8L050 startup-------------- */
  /* configure unbonded pins */
  CONFIG_UNUSED_PINS_STM8L050;
  /* delay for SWIM connection: ~5seconds */
  STARTUP_SWIM_DELAY_5S;
  /* ------------------------------------------- */

  /* configure all STM8L050 pins as input with pull up */
  GPIO_Init(GPIOC, GPIO_Pin_6, GPIO_Mode_In_PU_No_IT);     // pin 1
  GPIO_Init(GPIOA, GPIO_Pin_3, GPIO_Mode_In_PU_No_IT);     // pin 2
  GPIO_Init(GPIOD, GPIO_Pin_0, GPIO_Mode_In_PU_No_IT);     // pin 5
  GPIO_Init(GPIOB, GPIO_Pin_6, GPIO_Mode_In_PU_No_IT);     // pin 6
  GPIO_Init(GPIOB, GPIO_Pin_7, GPIO_Mode_In_PU_No_IT);     // pin 7
  GPIO_Init(GPIOC, GPIO_Pin_5, GPIO_Mode_In_PU_No_IT);     // pin 8

  /* initialize tested pin */
```

```
    GPIO_Init(TEST_PORT, TEST_PIN, GPIO_Mode_Out_PP_Low_Fast);
    /* initialize UART */
    GPIO_Init(GPIOC, GPIO_Pin_5, GPIO_Mode_Out_PP_High_Fast); // TxD: output
    CLK_PeripheralClockConfig(CLK_Peripheral_USART1, ENABLE);
    USART_Init(USART1, 9600, USART_WordLength_8b, USART_StopBits_1,
               USART_Parity_No,(USART_Mode_TypeDef)(USART_Mode_Rx |
               USART_Mode_Tx));

    USART_Cmd(USART1, ENABLE);
    /* initialize TIM3 */
    GPIO_Init(GPIOD, GPIO_Pin_0, GPIO_Mode_Out_PP_Low_Fast); // CH2: output
    CLK_PeripheralClockConfig(CLK_Peripheral_TIM3, ENABLE);
    TIM3_OC2Init(TIM3_OCMode_Toggle,
                 TIM3_OutputState_Enable,
                 9000,
                 TIM3_OCPolarity_High,
                 TIM3_OCIdleState_Reset);
    TIM3_Cmd(ENABLE);
    TIM3_CtrlPWMOutputs(ENABLE);
    /* initialize ADC */
    GPIO_Init(GPIOB, GPIO_Pin_6, GPIO_Mode_In_FL_No_IT);  // pin6: floating
    CLK_PeripheralClockConfig(CLK_Peripheral_ADC1, ENABLE);
    ADC_Init(ADC1, ADC_ConversionMode_Single, ADC_Resolution_12Bit,
                 ADC_Prescaler_1);
    ADC_ChannelCmd(ADC1, ADC_Channel_12, ENABLE); // ADC_IN12 on pin 6
    ADC_SamplingTimeConfig(ADC1,
                 ADC_Group_SlowChannels,ADC_SamplingTime_4Cycles);
    ADC_SchmittTriggerConfig(ADC1, ADC_Channel_12, DISABLE);
    ADC_Cmd(ADC1, ENABLE);

    while (1)
    {
      /* toggle with tested pin */
      GPIO_ToggleBits(TEST_PORT, TEST_PIN);
      /* send data to UART */
      USART_SendData8(USART1, 0x55);
      /* delay */
      for(i=0; i<10000; i++)
      {
        ADC_SoftwareStartConv(ADC1);
        while (!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));
        ADCvalue = ADC_GetConversionValue(ADC1);
      }
    }
  }
```

9 Pinout functionality overview

Figure 2 / *Table 1* and *Figure 3* / *Table 2* show an overview of STM8L001J3 and STM8L050J3 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. STM8L001J3 pinout

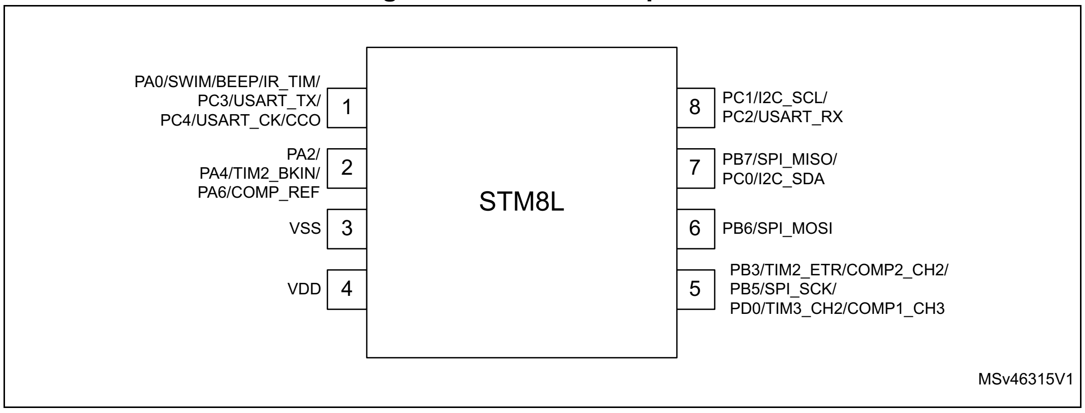


Table 1. STM8L001J3 alternate functions overview

Peripheral	Pin number							
	1	2	3	4	5	6	7	8
UART	TxD/CK	-	-	-	-	-	-	RxD
I2C	-	-	-	-	-	-	SDA	SCL
ADC	-	-	-	-	-	-	-	-
TIM2	IRTIM	BKIN	-	-	ETR	-	-	-
TIM3	IRTIM	-	-	-	CH2	-	-	-
OSC	CCO	-	-	-	-	-	-	-
Supply	-	-	VSS	VDD	-	-	-	-
Debug	SWIM	-	-	-	-	-	-	-
SPI	-	-	-	-	SCK	MOSI	MISO	-
COMP1	-	REF	-	-	CH3	-	-	-
COMP2	-	REF	-	-	CH2	-	-	-

Figure 3. STM8L050J3 pinout

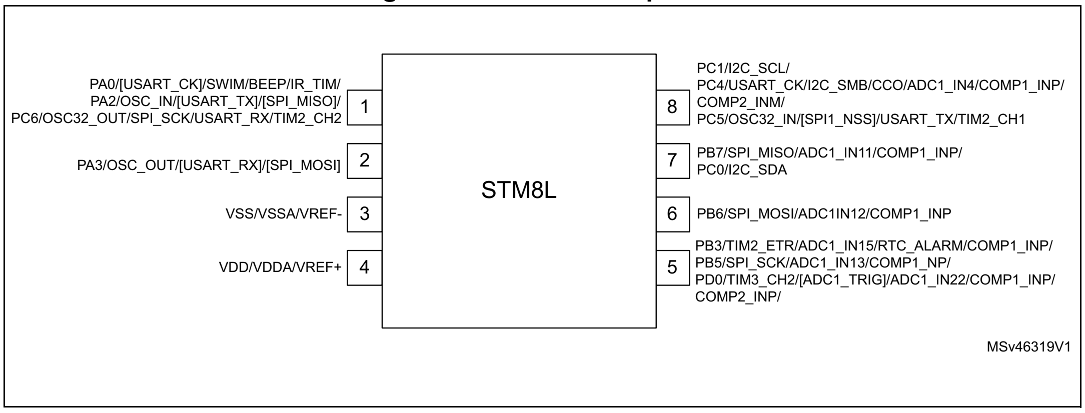

PA0/[USART_CK]/SWIM/BEEP/IR_TIM/
PA2/OSC_IN/[USART_TX]/[SPI_MISO]/
PC6/OSC32_OUT/SPI_SCK/USART_RX/TIM2_CH2
1
PA3/OSC_OUT/[USART_RX]/[SPI_MOSI]
2
VSS/VSSA/VREF-
3
VDD/VDDA/VREF+
4
STM8L
PC1/I2C_SCL/
PC4/USART_CK/I2C_SMB/CCO/ADC1_IN4/COMP1_INP/
COMP2_INM/
PC5/OSC32_IN/[SPI1_NSS]/USART_TX/TIM2_CH1
8
PB7/SPI_MISO/ADC1_IN11/COMP1_INP/
PC0/I2C_SDA
7
PB6/SPI_MOSI/ADC1IN12/COMP1_INP
6
PB3/TIM2_ETR/ADC1_IN15/RTC_ALARM/COMP1_INP/
PB5/SPI_SCK/ADC1_IN13/COMP1_NP/
PD0/TIM3_CH2/[ADC1_TRIG]/ADC1_IN22/COMP1_INP/
COMP2_INP/
5
MSv46319V1

Table 2. STM8L050J3 alternate functions overview

Peripheral	Pin number							
	1	2	3	4	5	6	7	8
UART	RxD/TxD /CK	RxD	-	-	-	-	-	TxD/CK
I2C	-	-	-	-	-	-	SDA	SCL
ADC	-	-	-	-	IN13/ TRIG	IN12	IN11	IN4
TIM2	CH2/ IRTIM	-	-	-	CH2	-	-	-
TIM3	IRTIM	-	-	-	CH2	-	-	-
OSC	32_OUT/ OSCIN	OSCOUT	-	-	RTC_ALARM	-	-	32_IN/ CCO
Supply	-	-	VSS	VDD	-	-	-	-
Debug	SWIM	-	-	-	-	-	-	-
SPI	SCK/ ~~MISO~~	MOSI	-	-	SCK	MOSI	MISO	NSS
COMP1	-	-	-	-	INP	INP	INP	INP
COMP2	-	-	-	-	INP	-	-	INM

9.1 Universal synchronous / asynchronous receiver transmitter (USART)

On STM8L001J3: RxD is on pin 8. TxD (and CK) is on pin 1. The synchronous mode cannot be used on this device because CK signal is shared with TxD signal on the same pin.

On STM8L050J3 the mapping options are:

- RxD is on pin 2. TxD is on pin 1. CK is pin 8.
- RxD is on pin 1. TxD is on pin 8. CK is pin 1. CK signal is shared with RxD signal on the same pin but the synchronous mode and smartcard mode can be used as those modes do not use RxD signal.

9.2 Inter-integrated circuit (I2C)

On STM8L001J3: SCL is on pin 8. SDA is on pin 7.

SCL and SDA pins are not true open-drain, because they are shared with another GPIO (on which the protection diode to VDD is implemented).

On STM8L050J3: SCL is on pin 8. SDA is on pin 7.

SCL and SDA pins are not true open-drain pins but only pseudo open-drain. This is because they are shared with another GPIO (on which the protection diode to VDD is implemented).

9.3 Serial peripheral interface (SPI)

On STM8L001J3: SCK is on pin 5. MOSI is on pin 6. MISO is on pin 7.

NSS signal is not available (slave detected management must be driven by software).

On STM8L050J3: the mapping options are:

- SCK is on pin 5. MOSI is on pin 6. MISO is on pin 7. NSS signal is not available (slave select management must be driven by software).
- SCK is on pin 1. MOSI is on pin 2. MISO is on pin 1. NSS signal is on pin 8.
  MISO signal is shared with SCK signal on the same pin, therefore only below operation modes are available:
  - unidirectional transfer (master mode = transmit only; slave mode = receive only)
  - simplex master synchronous transfers on two lines with a possible bidirectional data.

9.4 Oscillator

On STM8L001J3: external clock source input is not available. CCO output can be driven by HSI.

On STM8L050J3:

- OSCIN is on pin 1. OSCOUT is on pin 2. The external HSE crystal/resonator can be connected on those two pins. An external oscillator (HSE bypass) can be connected to OSCIN.
- OSC32_IN is on pin 8. OSC32_OUT is on pin 1. An external LSE crystal can be connected on those two pins. An external oscillator (LSE bypass) can be connected to OSC32_IN.
- OSCIN signal is shared with OSC32_OUT signal on the same pin, therefore if HSE clock is used (crystal/resonator or HSE bypass) then LSE must be only driven by external oscillator (LSE bypass).
- CCO output can be driven by internal or external clock source (HSI, LSI, HSE, LSE).

9.5 Analog-to-digital converter (ADC)

On STM8L001J3: ADC is not available.

On STM8L050J3: ADC input channels (ADC1_IN4, ADC1_IN11, ADC1_IN12, ADC1_IN13/IN15/IN22) are on pins 8, 7, 6 and 5. The external ADC trigger (ADC1_TRIG) is on pin 5 (shared with ADC input channel).

9.6 Comparators (COMP1, COMP2)

On STM8L001J3: COMP1 positive input is on pin 5. COMP1 negative (reference) input is on pin 2. COMP2 positive input is on pin 5. COMP2 negative (reference) input is on pin 2. In both comparators (COMP1 and COMP2) the inputs share the same pins and the outputs can internally drive different timers inputs.

On STM8L050J3: COMP1 positive inputs are on pins 5, 6, 7 and 8. COMP1 negative (reference) input is on the internal reference voltage. COMP2 positive input is on pin 5. COMP2 negative (reference) input is on pin 8 (or connected to the internal reference voltage submultiple).

9.7 Timers (TIM2, TIM3)

On STM8L001J3: the timer channels inputs/outputs are on pins 1 and 5 (including IRTIM high sink output). The timer break input (TIM2_BKIN) is on pin 2. The timer external trigger input (TIM2_ETR) is on pin 5.

On STM8L050J3: the timer channels inputs/outputs are on pins 1, 5 and 8 (including IRTIM high sink output). The timer external trigger input (TIM2_ETR) is on pin 5.

9.8 Single wire interface module (SWIM)

SWIM is on pin 1. The PA0/SWIM pin has an internal pull-up active after device reset.

The pin 1 can be used by another output function that disables the SWIM connection to the device by the debugger (see *Figure 2* / *Table 1* and *Figure 3* / *Table 2*). It is recommended to implement a ~5 seconds delay in the firmware during code development. This delay should be set after device reset (power on) and then set pin 1 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 1 (see *Section 8.2: Applying a delay before the SWIM pin reconfiguration* for more details).

10 STM8 software toolchain

In order to write, compile and run the first software on STM8L001J3 and STM8L050J3 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 refer to application note *Getting started with STM8L and STM8AL* (AN3029).

10.1 Integrated development environment (IDE)

The integrated development environment (IDE) provides an easy-to-use and efficient environment for start-to-finish control of the application code to program the microcontrollers.

10.1.1 ST toolset: ST visual develop (STVD) and ST visual programmer (STVP)

The STVD is delivered as part of the free ST toolset, which also includes the STVP programming interface and the ST assembler linker.

The STVD tool provides a seamless integration of C and assembly toolchains (such as Cosmic compiler, Raisonance C compiler and ST assembler linker) to build an application. 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.

The STVD also provides an interface for reading, writing and verifying the microcontroller memories for the applications based on STM8L001J3 and STM8L050J3. This interface is based on the STVP and supports STM8L001J3 and STM8L050J3 devices and programming tools.

The free ST toolset for STM8 is available from STMicroelectronics homepage *www.st.com.*

10.1.2 IAR embedded workbench for STM8

The IAR embedded workbench IAR-EWSTM8 is a software development tool, which provides 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

The STM8L001J3 and STM8L050J3 can be programmed by a free assembler toolchain included in the ST toolset. The use of a C compiler is recommended as the core is designed for optimized high-level-language support.

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.

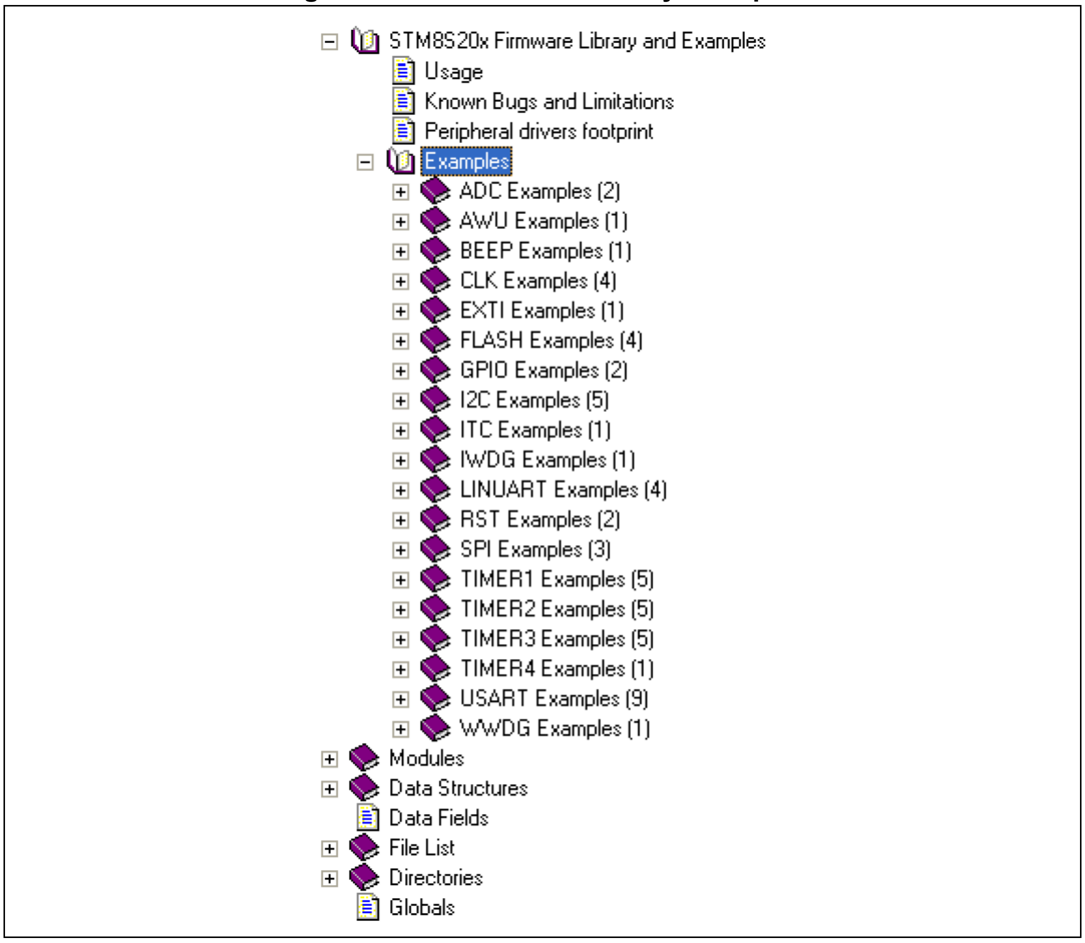
10.3 Firmware library

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 4*).

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

For additional information on STM8 firmware library, refer to STMicroelectronics homepage *www.st.com.*

Figure 4. STM8 firmware library examples



STM8S20x Firmware Library and Examples
- Usage
- Known Bugs and Limitations
- Peripheral drivers footprint
- Examples
  - ADC Examples (2)
  - AWU Examples (1)
  - BEEP Examples (1)
  - CLK Examples (4)
  - EXTI Examples (1)
  - FLASH Examples (4)
  - GPIO Examples (2)
  - I2C Examples (5)
  - ITC Examples (1)
  - IWDG Examples (1)
  - LINUART Examples (4)
  - RST Examples (2)
  - SPI Examples (3)
  - TIMER1 Examples (5)
  - TIMER2 Examples (5)
  - TIMER3 Examples (5)
  - TIMER4 Examples (1)
  - USART Examples (9)
  - WWDG Examples (1)
- Modules
- Data Structures
- Data Fields
- File List
- Directories
- Globals

11 Documentation and online support

Documentation resources related to STM8L001J3 and STM8L050J3 devices are listed below:

Application

- STM8L001J3 datasheet (DS12153)
- STM8L050J3 datasheet (DS12167)
- STM8L001xx and STM8L101xx device limitations errata sheet (ES0121)
- STM8L050J3, STM8L051F3, STM8L151x2 and STM8L151x3 device limitations errata sheet (ES0002)
- *STM8L001xx and STM8L101xx microcontroller families* reference manual (RM0013)
- *STM8L050/L051/L052 Value Line, STM8L151/L152, STM8L162, STM8AL31, STM8AL3L MCU lines* reference manual (RM0031)
- *How to program STM8L and STM8AL Flash program memory and data EEPROM* programming manual (PM0054)
- *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 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 3. Document revision history

Date	Revision	Changes
02-Oct-2017	1	Initial release.
04-Jul-2018	2	Updated: – *Section 8.2: Applying a delay before the SWIM pin reconfiguration*