STM32SnippetsL0

STM32L0xx Snippets firmware package

Data brief

Features

- Complete free C source code firmware examples for STM32L0xx microcontrollers
- Basic examples using direct-access registers as defined in CMSIS Cortex®-M0+ Device Peripheral Access Layer header file (sm32l0xx.h)
- Self-documented code
- Compliant with MISRA
- Compliant with EWARM, MDK-ARM™ and SW4STM32

Description

STM32L0xx Snippets provide a free source-code for the STM32L0 microcontroller series. The package includes a set of examples to help the designers when creating their own fully optimized C-code application on standard microcontrollers.

For further information contact your local STMicroelectronics sales office. www.st.com

Contents

List of tables

List of figures

1 General overview

1.1 Coding rules and conventions

This section describes the conventions used in the present document and in firmware.

1.1.1 Acronyms

Table 1 presents the acronyms used in this document.

Table 1. Glossary

Acronym	Definition
ADC	Analog-to-Digital Converter
COMP	Analog COMparators
CRC	CRC calculation unit
DAC	Digital to Analog Converter
DBGMCU	DeBuG MCU
DMA	DMA controller
EXTI	External interrupt/event controller
FLASH	FLASH memory
GPIO	General Purpose I/O
I2C	Inter-Integrated Circuit
I2S	Inter-Integrated Sound
IWDG	Independent WatchDoG
LPTIM	Low Power Timer
LPUART	Low Power Universal Asynchronous Receiver Transmitter
NVIC	Nested Vectored Interrupt Controller
PWR	PoWeR control
RCC	Reset and Clock Controller
RTC	Real-Time Clock
SPI	Serial Peripheral Interface
SysTick	System TICK timer
TIM	Advanced-control, general-purpose or basic timer
USART	Universal Synchronous Asynchronous Receiver Transmitter
WWDG	Window WatchDoG

1.1.2 Naming conventions

The following naming conventions are used in the drivers:

- PPP refers to any peripheral acronym, for example ADC. For more information see *Section 1.1.1: Acronyms*.
- Constants used in one file are defined within this file. A constant used in more than one file is defined in a header file. All constants are written in upper case, except for the peripheral driver function parameters.
- Registers are considered as constants. In most cases, their name is in upper case and uses the same acronyms as in the STM32L0xx Reference Manual.

1.1.3 Coding rules

This section describes the coding rules used in the Snippets.

General

- All codes should comply with ANSI C standard and should compile without warning under at least its main compiler. Any warnings that cannot be eliminated should be commented in the code.
- The Snippets use ANSI standard data types defined in the ANSI C header file <stdint.h>.
- The application code shall not have any blocking code. The designers define their own policy to avoid endless loops. Each time a polling is performed in a Snippet, a comment reminds the designers to add a time-out management as shown hereafter:

```
while ((ADC1->CR & ADC_CR_ADEN) != 0)
{
   /* For robust implementation, add here time-out management */
}
```

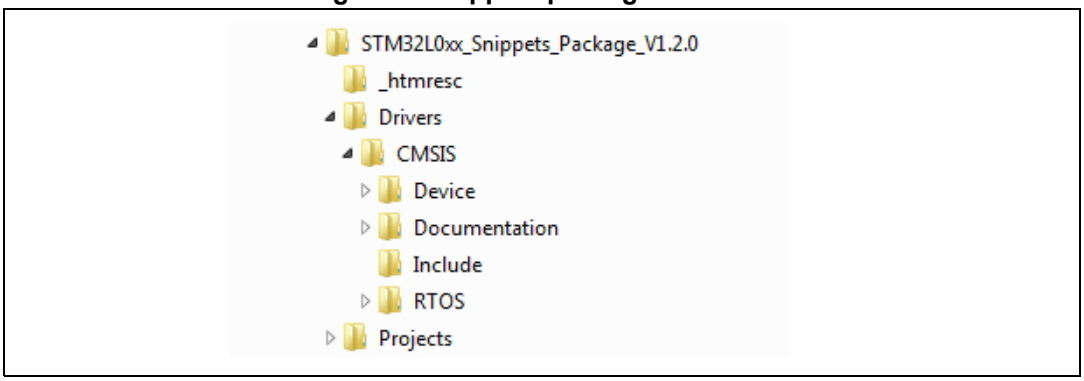
1.2 Architecture

To fully optimize the application code, the Snippets access directly the CMSIS layers (Core Peripheral Access Layer + STM32L0xx Device Peripheral Access Layer).

1.3 Package description

The Snippets are supplied in one single zip file. The zip file extraction results in the creation of the folder STM32L0xx_Snippets_Package_VX.Y.Z, where VX.Y.Z refers to the Snippets version, for example V1.0.0. *Figure 1* shows the package structure of the Snippets.

Figure 1. Snippets package structure

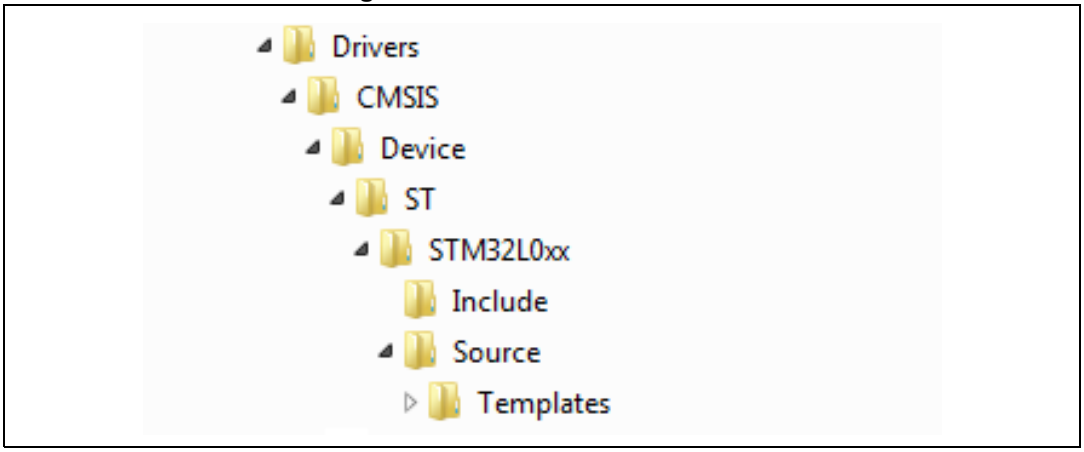


The package of the Snippets consists of three main folders, described in *Section 1.3.1: Drivers folder structure*.

1.3.1 Drivers folder structure

This folder contains all the CMSIS files. The folder structure for the drivers is shown in *Figure 2*.

Figure 2. Drivers folder structure



CMSIS sub folder

The CMSIS sub folder contains the STM32L0xx and Cortex®-M0+ CMSIS files:

- Cortex®-M CMSIS files include the name definitions, address definitions and helper functions to access Cortex®-M0+ core registers and peripherals. They also includes the setup to use to define a device independent interface for RTOS kernels that includes debug channel definitions.
- STM32L0xx CMSIS files consist of:
  - *stm32l0xx.h*: this file contains the definitions of all peripheral registers, bits, and memory mapping for STM32L0xx devices. It is the unique include file used in the application programmer C source code, usually in the main.c.
  - *stm32l053xx.h*: this file contains the definitions of all peripheral registers, bits and memory mapping for STM32L053XX devices.
  - *system_stm32l0xx.c/.h*: this file contains the system clock configuration for STM32L0xx devices. It exports SystemInit() function which resets the system clock source, PLL multiplier and divider factors, AHB/APB prescalers and Flash settings. This function is called at startup just after reset and before connecting to the main program. The call is made inside the *startup_stm32l0xx.s* file.
  - *startup_stm32l0xx.s*: this file contains the Cortex®-M0+ startup code and interrupt vectors for all STM32L0xx device interrupt handlers.

1.3.2 Project folder

For each peripheral this folder contains the minimum set of files needed to run a typical example on a given peripheral.

Aside the IDE folders, one source file is provided:

- *main.c*: this file contains the Snippets code itself.

1.4 Supported devices and development tools

1.4.1 Supported devices

The Snippets support the whole STM32L0 series of microcontrollers and peripherals.

Some examples in the Snippets firmware may use a feature that is not present in all STM32L0xx devices. The target board of the Snippets is the STM32L053 Discovery board. The STM32L053 device supports the full set of peripherals. When using a derivative of STM32L0, refer to the device datasheet to check the targeted feature availability.

1.4.2 Supported development tools and compilers

Lead suppliers support the STM32L0xx devices with a full range of development solutions. They deliver start-to-finish control of application development from a single integrated development environment.

A project is available for some development tools:

- AC6 System Workbench for STM32 (SW4STM32) development tool
  - Compiler: GCC C/C++ compile
- IAR™ Embedded Workbench for ARM® (EWARM) development tool
  - Compiler: IAR™ C/C++ compiler
- Keil® Microcontroller Development Kit (MDK-ARM™) development tool
  - Compiler: ARM® C/C++ compiler

2 Using and configuring an example

The following sections provide user with the steps required to configure, customize, and run his first example. They also provide the steps to follow to develop an application from the snippet examples.

2.1 Using an example

Each example folder contains the workspace/project files corresponding to a supported toolchain. Once the user has installed one of these toolchains, he must double-click on the workspace file in the corresponding folder to launch the IDE:

- for SW4STM32: create a workspace and import this file: *\Projects\PPP\nn_ExampleName\SW4STM32\Project\.project*
- for IAR: *\Projects\PPP\nn_ExampleName\EWARM\project.eww*
- for MDK-ARM: *\Projects\PPP\nn_ExampleName\MDK-ARM\project.uvproj*

Only the *main.c* differs from one example to another. The code is self-documented with:

- a brief description
- the used MCU resources
- the RCC specific features
- the files needed to build the example
- the expected behavior
- a way to test if the code is executed correctly

```
/**

******************************************************************************
  * @file    03_MCO_GPIO_Configuration/main.c
  * @author  MCD Application Team
  * @version V1.2.0
  * @date    05-February-2016
  * @brief   This code example shows how to output the system clock on MCO
  *
==============================================================================
                     ##### RCC specific features #####

==============================================================================
    [..] After reset the device is running from MSI (2 MHz) with Flash 0 WS,
         and voltage scaling range is 2 (1.5V)
         all peripherals are off except internal SRAM, Flash and SW-DP.
         (+) There is no prescaler on High speed (AHB) and Low speed (APB)
             busses;
             all peripherals mapped on these busses are running at MSI
             speed.
         (+) The clock for all peripherals is switched off, except the SRAM
             and FLASH.
```

```
          (+) All GPIOs are in analog state, except the SW-DP pins which
              are assigned to be used for debug purpose.
     [..] Once the device started from reset, the user application has to:
          (+) Configure the clock source to be used to drive the System clock
              (if the application needs higher frequency/performance)
          (+) Configure the System clock frequency and Flash settings
          (+) Configure the AHB and APB busses prescalers
          (+) Enable the clock for the peripheral(s) to be used
          (+) Configure the clock source(s) for peripherals whose clocks
              are not derived from the System clock (ADC, RTC/LCD, RNG and
              IWDG)
==============================================================================
                    #####       MCU Resources     #####
==============================================================================

   - RCC
   - GPIO PA8,PA5 and PB4
   - SYSTICK (to manage led blinking)

==============================================================================

                    ##### How to use this example #####

==============================================================================

    - this file must be inserted in a project containing  the following
      files :
      o system_stm32l0xx.c, startup_stm32l053xx.s
      o stm32l0xx.h to get the register definitions
      o CMSIS files
==============================================================================

                    ##### How to test this example #####

==============================================================================

    - This example configures the PA8 to output the system clock.
    - The signal on PA8 @16MHz can be monitored with an oscilloscope.
```

2.2 Configuring an example

1. Create a project and setup all the toolchain startup files
2. Select the startup file corresponding to the used device:
   - SW4STM32: *startup_stm32l053xx.s* under:
     `Drivers\CMSIS\Device\ST\STM32L0xx\Source\Templates\gcc`
   - EWARM: *startup_stm32l0xx.s* under:
     `Drivers\CMSIS\Device\ST\STM32L0xx\Source\Templates\iar`
   - MDK-ARM: *startup_stm32l0xx.s* under:
     `Drivers\CMSIS\Device\ST\STM32L0xx\Source\Templates\arm`
3. The device entry point is *stm32l0xx.h* (under `Drivers\CMSIS\Device\ST\STM32L0xx\Include`), the user must include *stm32l0xx.h* in the application main

Note: The device can be selected in the project settings of the IDE. All the examples have the STM32L053 device selected.

4. Add the *system_stm32l0xx.c* (under `Drivers\CMSIS\Device\ST\STM32L0xx\Source\Templates`) file in the application. This file provides the functions to setup the STM32 system. The system clock is configured in the main.c, as shown in *Table 2*:

Table 2. System clock configuration

Parameter	Value
System clock source	PLL MUL8 DIV2
SYSCLK	16000000 Hz
HCLK	16000000 Hz
AHB prescaler	1
APB prescaler	1
HSE frequency	8000000 Hz
PLL source	HSI/4
PLL MUL	8
PLL DIV	2
VDD	3.3 V
Flash latency	0 WS

2.3 Programming model

Direct-access registers

This model is based on direct-access registers using the CMSIS layer. This layer provides the definition of all STM32L0xx peripheral registers and bits, as well as memory mapping as defined in *stm32l053xx.h* file.

The advantage of this approach is that the code produced is compact and efficient. The developer shall refer to the STM32L0 Reference Manual to know in details the peripheral operation, the registers and bits meaning, and the configuration procedure.

2.4 Running user's examples

The snippet package provides a rich set of examples covering the main features of each peripheral.

All the examples are independent from the development tools. The examples run on STMicroelectronics STM32L053 Discovery board. They can be easily tailored to any other supported device and development board. Source files and projects are provided for each example for a limited set of development tools.

2.4.1 Prerequisites

1. Latest release of documents and snippet code. Users can download the latest version of STM32L0xx related documents and Snippets from STMicroelectronics website: *www.st.com/stm32.*
2. Hardware: the Snippets have been designed to run on the STM32L053 Discovery board from STMicroelectronics.
3. The user can use his hardware, simply adapting the hardware configuration example to his platform, that is the GPIO configuration and alternate functions.
4. Development tools:
   Select the preferred development tool: SW4STM32 (AC6), MDK-ARM (Keil) and EWARM (IAR). Check that the used version supports the STM32L0xx devices.

2.4.2 Running an example

This section describes how to load and execute an example provided within the Snippets package.

Proceed as described below to load and execute an example:

1. Download and unzip the *STM32L0xx_Snippets_Package_VX.Y.Z.zip* in the selected folder
2. Check information in «How to use this example» in *main.c* to know if some solder bridges must be shunted
3. Power-up the STM32L053 Discovery board
4. The STM32L053 Discovery features a built-in ST-LINK/V2 debugger and programmer, which prevent from having to use external hardware debuggers to load and debug the user's program
5. Select ST-LINK/V2 as debugger in the Development Tool configuration menu and connect the «USB ST-LINK» mini USB connector to the host PC through an USB cable
6. Run the selected Snippets:
   a) SW4STM32
   – Launch SW4STM32
   – Create a workspace
   – Import the project file: File>Import...>General>Existing Projects Into Workspace
   – Build all files: Project>Build All
   – Run the program: Run>Debug As...>Resume (F8)
   b) EWARM
   – Open the `EWARM\Project.eww` workspace
   – Rebuild all files: Project->Rebuild all
   – Load the project image: Project->Debug
   – Run the program: Debug->Go(F5)
   c) MDK-ARM
   – Open the `MDK-ARM\Project.uvproj` project
   – Rebuild all files: Project->Rebuild all target files
   – Load the project image: Debug->Start/Stop Debug Session
   – Run the program: Debug->Run (F5)

Test the behavior of the application with the expected behavior that is described in the «How to test this example» section in *main.c*.

2.4.3 Looking for specific feature

To avoid the duplication of the examples, some features or configurations are only present in some examples.

All the configurations related to GPIO or RCC can be found in all the examples as a signal must always be output and the clock must always be configured.

The examples are sorted by peripherals. For example all the ADC examples are in the *..\projects\ADC* folder.

The best way to find a specific feature is to perform a «Find in Files» in the favorite IDE. Alternatively, use the free Notepad++ editor.

- In SW4STM32, perform Edit>Find and Replace>Find in Files
- In IAR, perform Edit->Find and Replace->Find in Files
- In MDK-ARM, perform Edit->Find in Files
- Notepad++, perform Search->Find in Files

3 References

The following documents have been used as reference to develop the Snippets code:

1. AN4235 I2C timing configuration tool for STM32F3xxxx and STM32F0xxxx microcontrollers, Application Note.
2. DS10152 Ultra-low-power 32-bit MCU ARM -based Cortex-M0+, up to 64KB Flash, 8KB SRAM, 2KB EEPROM, LCD, USB, ADC, DAC Datasheet.
3. RM0367 Ultra-low-power STM32L0x3 advanced ARM -based 32-bit MCUs Reference Manual.
4. UM1775 Discovery kit for STM32 L0 series - with STM32L053C8 MCU, User manual
5. UM1790 Getting started with STM32L053 Discovery software development tools, User manual.

4 Revision history

Table 3. Document revision history

Date	Revision	Changes
20-Jun-2014	1	Initial release.
16-Feb-2016	2	Updated *Section : CMSIS sub folderSection 1.4.2: Supported development tools and compilers*, *Section 2.1: Using an example*, *Section 2.2: Configuring an example*, *Section 2.4: Running user's examples*, *Section 2.4.3: Looking for specific feature*.

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