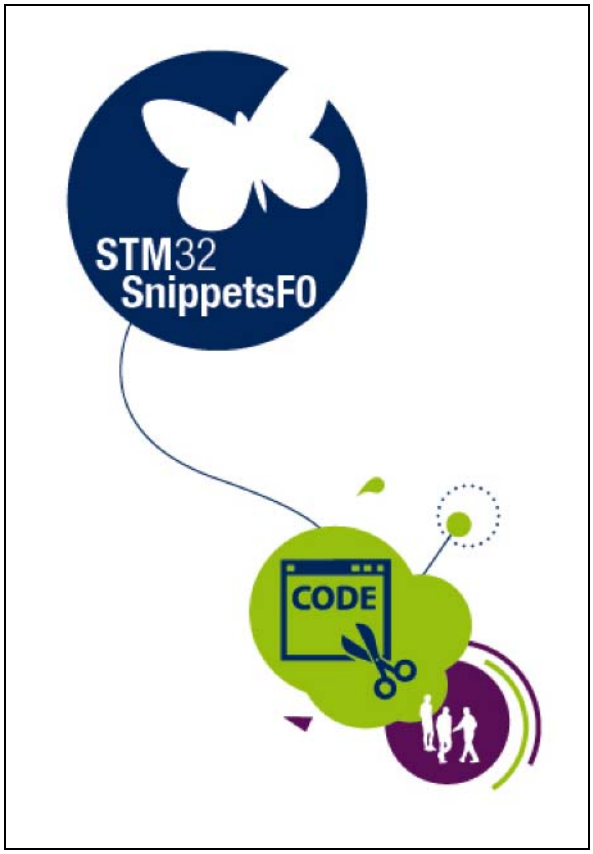
STM32SnippetsF0

STM32F0xx Snippets firmware package

Data brief

Features

- Complete free C source code firmware examples for STM32F0xx microcontrollers
- Basic examples using register direct-accesses as defined in CMSIS Cortex-M0 Device Peripheral Access Layer Header File (sm32f0xx.h)
- Self-documented code
- Compliant with MISRA
- Compliant with SW4STM32, EWARM and MDK-ARM



Description

STM32F0xx snippets provide a free source-code for the STM32F0xx microcontroller family. The package includes a set of examples to help the designers when creating their own fully optimized C-code application on standard microcontrollers.

Contents

1 General overview

1.1 Coding rules and conventions

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

1.1.1 Acronyms

Table 1 presents the acronyms used in this document.

Table 1. Glossary

Acronym	Definition
ADC	Analog-to-Digital Converter
CAN	Controller Area Network
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
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 library:

- PPP refers to any peripheral acronym, for example ADC. See *Section 1.1.1: Acronyms* for more information.
- 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 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 STM32F0xx 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 */
}
```

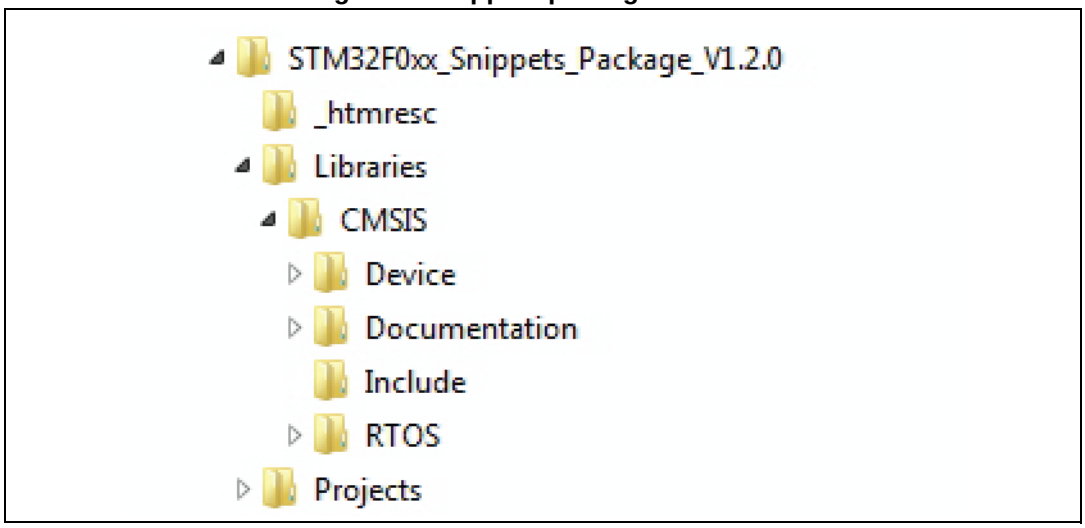
1.2 Architecture

To fully optimize the application code, the snippets access directly the CMSIS layers (Core Peripheral Access Layer + STM32F0xx 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 STM32F0xx_Snippets_Package_VX.Y.Z, where VX.Y.Z refers to the snippets version, for example V1.2.0. *Figure 1* shows the Snippets package structure.

Figure 1. Snippets package structure



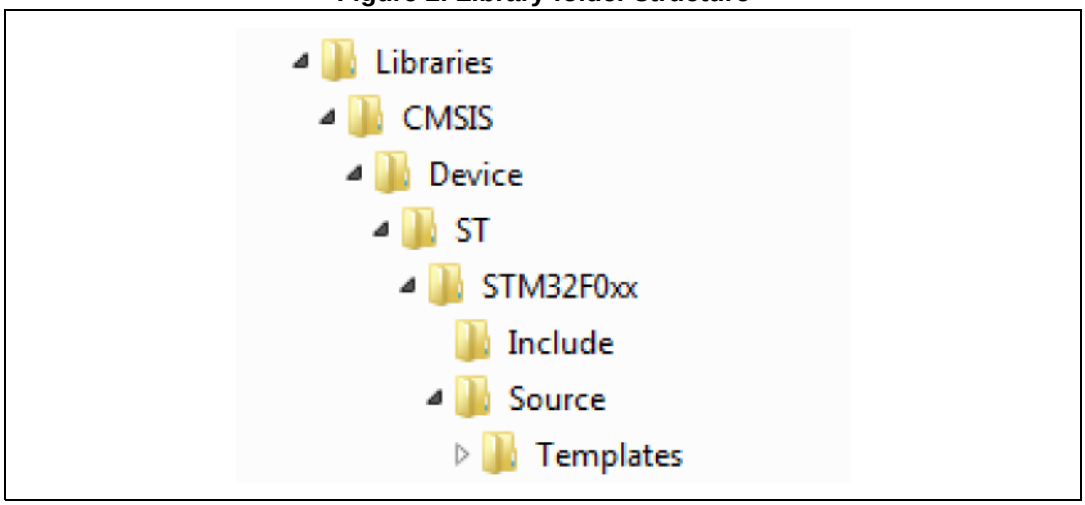
The snippets package consists of three main folders, described in *Section 1.3.1: Library folder structure*.

1.3.1 Library folder structure

This folder contains all the CMSIS files.

The library folder structure is shown in *Figure 2*.

Figure 2. Library folder structure



CMSIS sub folder

The CMSIS sub folder contains the STM32F0xx 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.
- STM32F0xx CMSIS files consist of:
  - *stm32f0xx.h*: this file contains the configuration section that allows to select the STM32F0xx device used in the target application and to use or not the peripheral™ drivers in application code (i.e. the code is based on direct access to peripheral™ registers rather than the driver APIs). It is the unique include file used in the application programmer C source code, usually in the main.c.
  - *stm32f072xb.h*: this file contains the definitions of all peripheral registers, bits and memory mapping for STM32F072XB devices.
  - *system_stm32f0xx.c/.h*: this file contains the system clock configuration for STM32F0xx devices. It exports SystemInit() function which sets up 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_stm32f0xx.s* file.
  - *startup_stm32f072xb.s*: this file contains the Cortex-M0 startup code and interrupt vectors for all STM32F072xB device interrupt handlers.

1.3.2 Project folder

This folder contains for each peripheral the minimum set of files needed to run a typical example on a peripheral.

Aside the IDE folders, two source files are provided:

- *main.c*: this file contains the snippets code itself.
- *system_stm32f0xx.c*: this file is the same for all examples, the main clock parameters are PLL x12 and HSI divided by 2, so the system clock is 48 MHz.

1.4 Supported devices and development tools

1.4.1 Supported devices

The snippets support the whole STM32F0 series of microcontrollers and peripherals.

Some examples in the Snippets firmware may use a feature that is not present in all STM32F0xx devices. The target board of the snippets is the STM32F072 Discovery board. The STM32F072 device supports the full set of peripherals. When using a derivative of STM32F0, refer to the device datasheet to check the targeted feature availability.

1.4.2 Supported development tools and compilers

Lead suppliers support the STM32F0xx 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++ compiler
- IAR Embedded Workbench for ARM (EWARM) development tool
  - Compiler: IAR C/C++ compiler
- RealView Microcontroller Development Kit (MDK-ARM) development tool
  - Compiler: ARM C/C++ compiler

2 Using and configuring an example

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

2.1 Using an example

Each example folder contains the workspace/project files corresponding to a supported toolchain. Once you have installed one of these toolchains:

- for SW4STM32: create a workspace and import this project file:
  `\Projects\PPP\nn_ExampleName\SW4STM32\Project\.project`
- for EWARM, double-click on this workspace file to launch the IDE:
  `\Projects\PPP\nn_ExampleName\EWARM\project.eww`
- for MDK-ARM, double-click on this workspace file to launch the IDE:
  `\Projects\PPP\nn_ExampleName\MDK-ARM\project.uvprojx`

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

- a brief description,
- the used MCU resources,
- 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 19-June-2015
  * @brief   This code example shows how to output the system clock on MCO
  *
 ==============================================================================
                     #####       MCU Resources     #####
 ==============================================================================
   - RCC
   - GPIO PA8,PC8,9

 ==============================================================================
                     ##### How to use this example #####
 ==============================================================================
    - this file must be inserted in a project containing  the following
      files :
      o system_stm32f0xx.c, startup_stm32f0xx.s
      o stm32f0xx.h to get the register definitions
      o CMSIS files
```

```
================================================================================
                    ##### How to test this example #####
================================================================================
   - This examples configure the PA8 to output the system clock.
   - The signal on PA8 @4MHz can be monitored with an oscilloscope
```

2.2 Configuring an example

1. Create a project and setup all your toolchain startup files.
2. Select the startup file corresponding to the used device:
   – SW4STM32: *startup_stm32f072xb.s* under `Libraries\CMSIS\Device\ST\STM32F0xx\Source\Templates\gcc`
   – EWARM: *startup_stm32f072xb.s*, under *`Libraries\CMSIS\Device\ST\STM32F0xx\Source\Templates\iar`*
   – MDK-ARM: *startup_stm32f072xb.s*, under *`Libraries\CMSIS\Device\ST\STM32F0xx\Source\Templates\arm`*
3. The device entry point is *stm32f0xx.h* (under *`Libraries\CMSIS\Device\ST\STM32F0xx\Include`*), the user must include *stm32f0xx.h* in the application main.

Note: You can select the device in the project settings of the IDE. All the examples have the STM32F072xB device selected.

4. Add the *system_stm32f0xx.c* (under *`Libraries\CMSIS\Device\ST\STM32F0xx\Source\Templates`*) file in your application. This file provides the functions to setup the STM32 system, that is to configure the PLL, the system clock and initialize the Embedded Flash Interface. This file provides multiple choice for the system clock frequency.

   The *system_stm32f0xx* file configures the system clock as shown in *Table 2*:

Table 2. System clock configuration

Parameter	Value
System clock source	PLL(HSI/2)
SYSCLK	48000000 Hz
HCLK	48000000 Hz
AHB prescaler	1
APB prescaler	1
HSE frequency	8000000 Hz
PLL source	HSI/2
PLL MUL	12
VDD	3.3 V
Flash latency	1 WS

2.3 Programming model

Direct register access

This model is based on direct register access using the CMSIS layer. This layer provides the definition of all STM32F02xB peripheral registers and bits, as well as memory mapping as defined in *stm32f072xb.h* file.

The advantage of this approach if that the code produced is compact and efficient. The developer shall refer to the STM32F0 Reference Manual to know in details the peripheral operation, the registers and bits meaning, and the configuration procedure.

2.4 Running your 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 STM32F072 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. You can download the latest version of STM32F0xx related documents and snippets from STMicroelectronics website: *www.st.com/stm32*
2. Hardware: the snippets have been designed to run on the STM32F072 Discovery board from STMicroelectronics.
3. To use your own hardware, simply adapt the example hardware configuration to your platform, that is the GPIO configuration and alternate function.
4. Development tools:
   Use your preferred development tool: SW4STM32 (AC6), MDK-ARM (Keil) and EWARM (IAR). Check that the version you are using supports STM32F0xx 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 *STM32F0xx_Snippets_Package_VX.Y.Z.zip* in the folder of your choice.
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 STM32F072 Discovery board
4. The STM32F072 Discovery features a built-in ST-Link/V2 debugger and programmer which prevent from having to use external hardware debuggers to load and debug your program.
5. Select ST-Link/V2 as your debugger in the Development Tool configuration menu and connect the «USB ST-LINK» mini USB connector to your host PC through an USB cable.
6. Run the snippet example of your choice:
   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 your favorite IDE. Alternatively, use the free Notepad++ editor.

- In SW4STM32, perform Edit>Find and Replace>Find in Files
- In EWARM, 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. AN4055 Clock configuration tool for STM32F0xx microcontroller, application note
2. AN4235 I2C timing configuration tool for STM32F3xxxx and STM32F0xxxx microcontrollers, application note
3. DS9826 ARM-based 32-bit MCU, up to 128 KB Flash, crystal-less USB FS 2.0, CAN, 12 timers, ADC, DAC & comm. interfaces, 2.0 - 3.6 V, Datasheet
4. RM0091 STM32F0x1/STM32F0x2/STM32F0x8 advanced ARM-based 32-bit MCUs, reference manual
5. UM1690 Discovery kit for STM32 F0 series - with STM32F072RB MCU, user manual
6. UM1715 Getting started with STM32F072B Discovery software development tools, user manual

4 Revision history

Table 3. Document revision history

Date	Revision	Changes
24-Feb-2014	1	Initial release.
01-Jul-2015	2	Added the compliance to SW4STM32. Updated: – the package version to V1.2.0, – *Figure 1: Snippets package structure*, – *Section : CMSIS sub folder*, – *Section 1.4.2: Supported development tools and compilers*, – *Section 2.1: Using an example*, – *Section 2.2: Configuring an example*, – *Section 2.3: Programming model*, – *Section 2.4.1: Prerequisites*, – *Section 2.4.2: Running an example*, – *Section 2.4.3: Looking for specific feature*.
28-Jul-2015	3	Changed the visual for STM32SnippetsF0 on the cover page. No change in the document content. Removed the table providing the list of applicable products on the cover page.