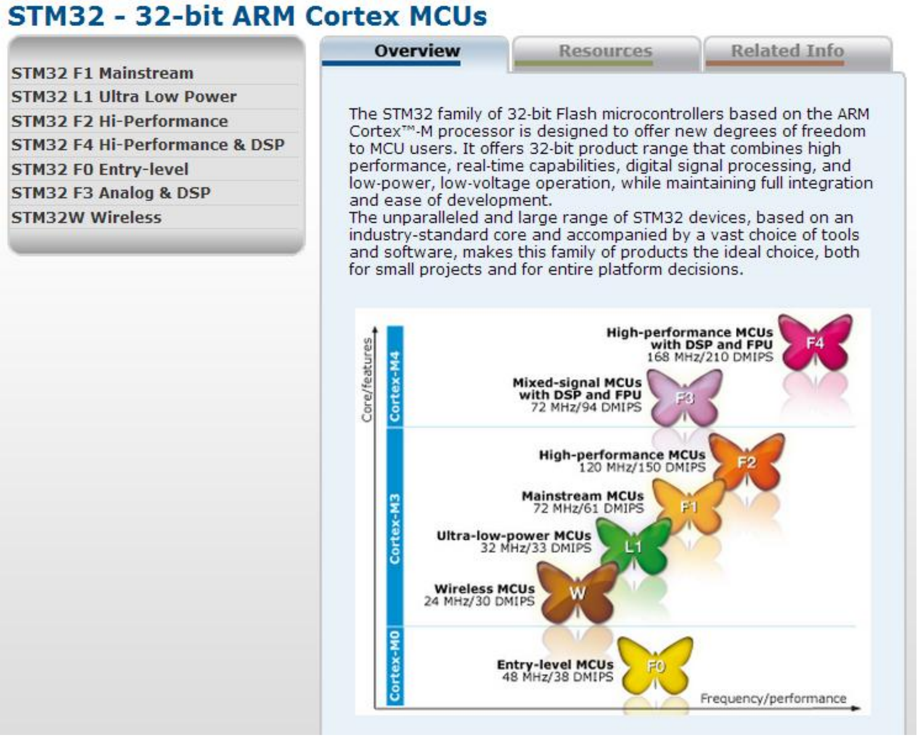
STM32xxx-Library

Oct. 2012

WWW.EMCU.IT

STM32xxx Library

STM for all STM32 family release the relative library that support all the peripherals that are inside the MCU.



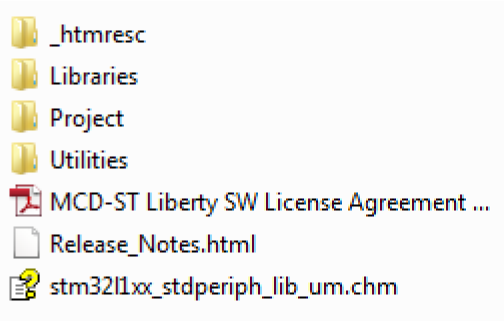
STM32 - 32-bit ARM Cortex MCUs

STM32 F1 Mainstream
STM32 L1 Ultra Low Power
STM32 F2 Hi-Performance
STM32 F4 Hi-Performance & DSP
STM32 F0 Entry-level
STM32 F3 Analog & DSP
STM32W Wireless

The STM32 family of 32-bit Flash microcontrollers based on the ARM Cortex™-M processor is designed to offer new degrees of freedom to MCU users. It offers 32-bit product range that combines high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.
The unparalleled and large range of STM32 devices, based on an industry-standard core and accompanied by a vast choice of tools and software, makes this family of products the ideal choice, both for small projects and for entire platform decisions.

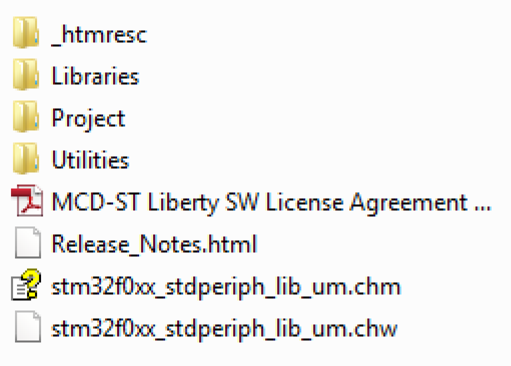
Core/features
Cortex-M4
Cortex-M3
Cortex-M0
High-performance MCUs with DSP and FPU 168 MHz/210 DMIPS
F4
Mixed-signal MCUs with DSP and FPU 72 MHz/94 DMIPS
F3
High-performance MCUs 120 MHz/150 DMIPS
F2
Mainstream MCUs 72 MHz/61 DMIPS
F1
Ultra-low-power MCUs 32 MHz/33 DMIPS
L1
Wireless MCUs 24 MHz/30 DMIPS
W
Entry-level MCUs 48 MHz/38 DMIPS
F0
Frequency/performance

STM32L1



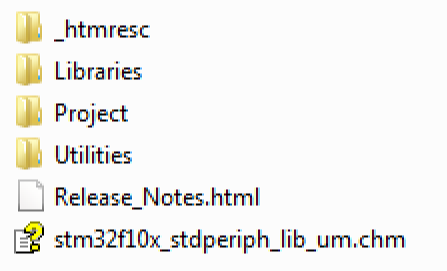
_htmresc
Libraries
Project
Utilities
MCD-ST Liberty SW License Agreement ...
Release_Notes.html
stm32l1xx_stdperiph_lib_um.chm

STM32F0



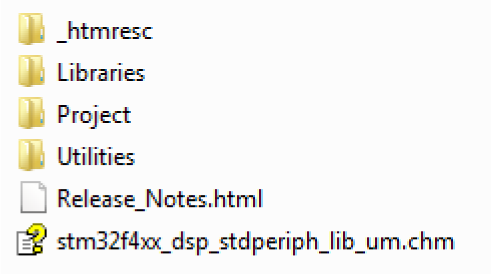
_htmresc
Libraries
Project
Utilities
MCD-ST Liberty SW License Agreement ...
Release_Notes.html
stm32f0xx_stdperiph_lib_um.chm
stm32f0xx_stdperiph_lib_um.chw

STM32F1



_htmresc
Libraries
Project
Utilities
Release_Notes.html
stm32f10x_stdperiph_lib_um.chm

STM32F4



_htmresc
Libraries
Project
Utilities
Release_Notes.html
stm32f4xx_dsp_stdperiph_lib_um.chm

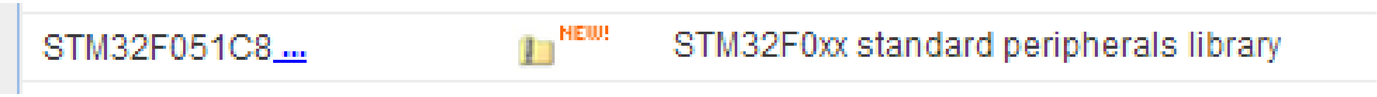
life.augmented

SILICA An Avnet Company

WWW.EMCU.IT

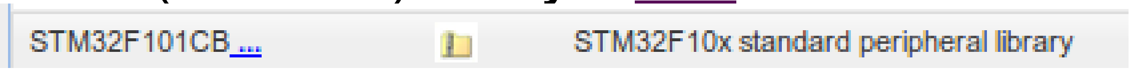
STM32xxx Library

The STM32F0xx (Cortex M0) **library is here**



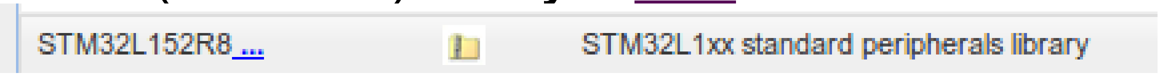
STM32F051C8 ... NEW! STM32F0xx standard peripherals library

The STM32F1xx (Cortex M3) **library is here**



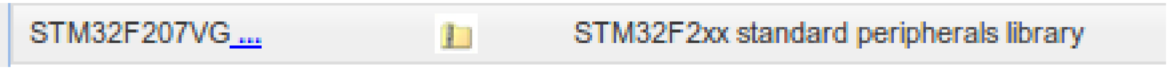
STM32F101CB ... STM32F10x standard peripheral library

The STM32L1xx (Cortex M3) **library is here**



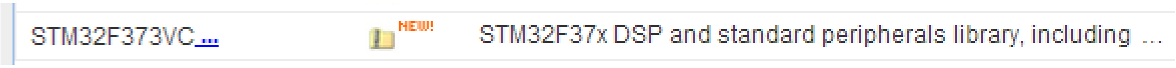
STM32L152R8 ... STM32L1xx standard peripherals library

The STM32F2xx (Cortex M3) **library is here**



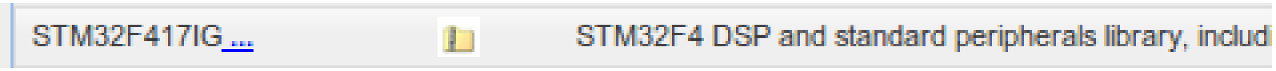
STM32F207VG ... STM32F2xx standard peripherals library

The STM32F3xx (Cortex M4) **library is here**



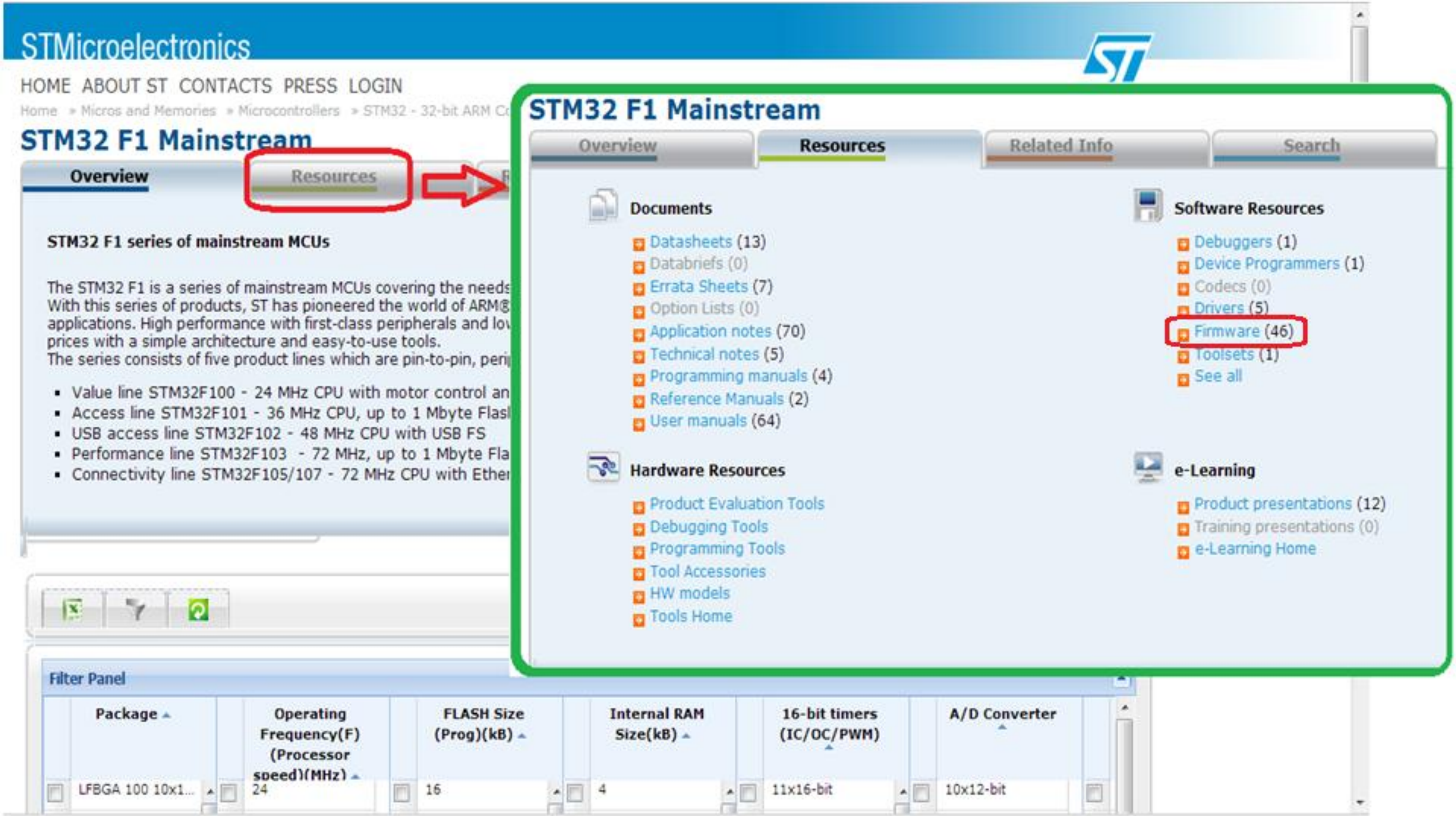
STM32F373VC ... NEW! STM32F37x DSP and standard peripherals library, including ...

The STM32F4xx (Cortex M4) **library is here**



STM32F417IG ... STM32F4 DSP and standard peripherals library, includ

STM32xxx Library



STMicroelectronics

HOME ABOUT ST CONTACTS PRESS LOGIN

Home » Micros and Memories » Microcontrollers » STM32 - 32-bit ARM Co

STM32 F1 Mainstream

STM32 F1 series of mainstream MCUs

The STM32 F1 is a series of mainstream MCUs covering the needs
With this series of products, ST has pioneered the world of ARM®
applications. High performance with first-class peripherals and lo
prices with a simple architecture and easy-to-use tools.
The series consists of five product lines which are pin-to-pin, peri

- Value line STM32F100 - 24 MHz CPU with motor control an
- Access line STM32F101 - 36 MHz CPU, up to 1 Mbyte Flas
- USB access line STM32F102 - 48 MHz CPU with USB FS
- Performance line STM32F103 - 72 MHz, up to 1 Mbyte Fla
- Connectivity line STM32F105/107 - 72 MHz CPU with Ether

Filter Panel

	Package		Operating Frequency(F) (Processor speed)(MHz)		FLASH Size (Prog)(kB)		Internal RAM Size(kB)		16-bit timers (IC/OC/PWM)		A/D Converter	
□	LFBGA 100 10x1...	□	24	□	16	□	4	□	11x16-bit	□	10x12-bit	□

STM32 F1 Mainstream

Documents

- Datasheets (13)
- Databriefs (0)
- Errata Sheets (7)
- Option Lists (0)
- Application notes (70)
- Technical notes (5)
- Programming manuals (4)
- Reference Manuals (2)
- User manuals (64)

Software Resources

- Debuggers (1)
- Device Programmers (1)
- Codecs (0)
- Drivers (5)
- Firmware (46)
- Toolsets (1)
- See all

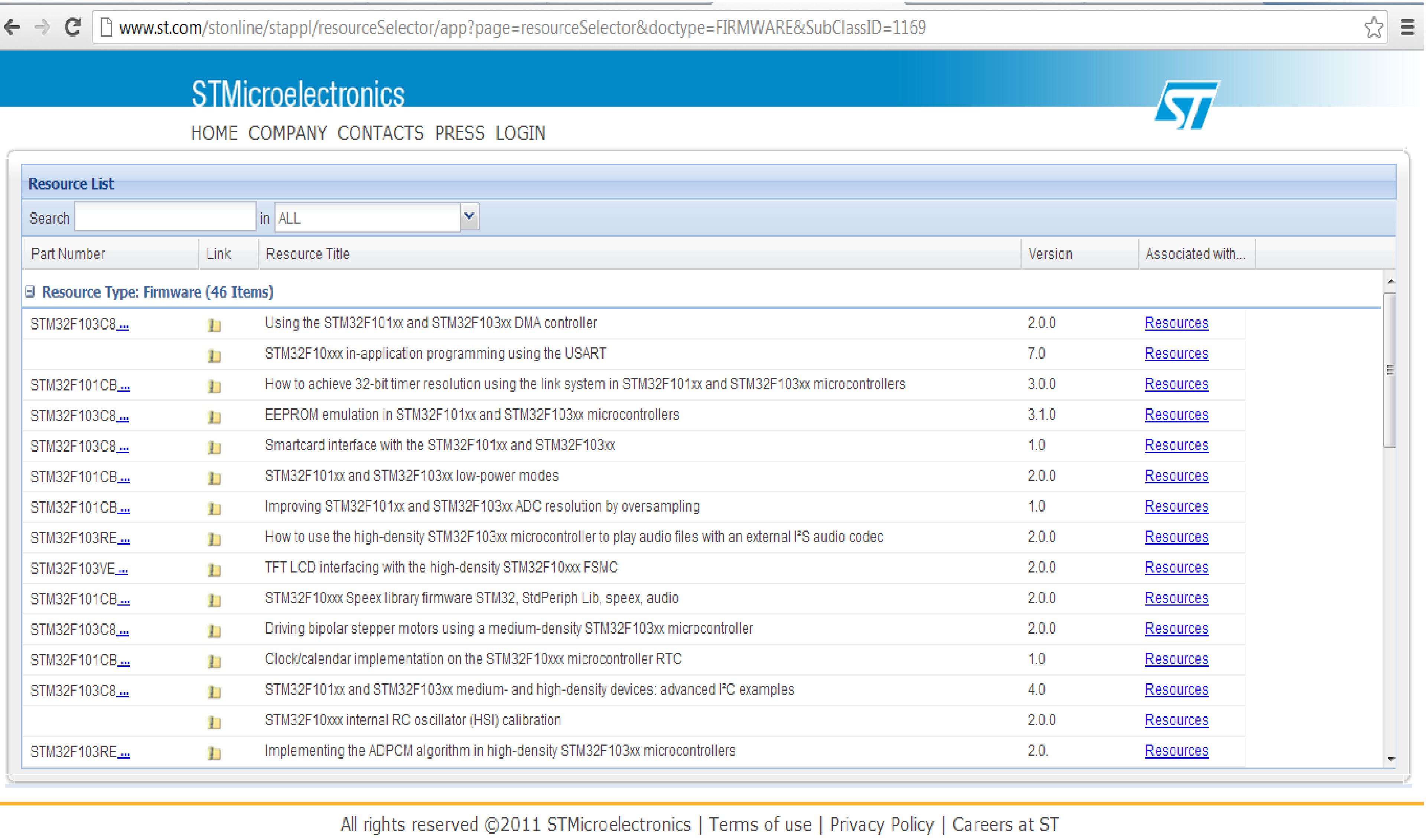
Hardware Resources

- Product Evaluation Tools
- Debugging Tools
- Programming Tools
- Tool Accessories
- HW models
- Tools Home

e-Learning

- Product presentations (12)
- Training presentations (0)
- e-Learning Home

STM32xxx Library



www.st.com/stonline/stappl/resourceSelector/app?page=resourceSelector&doctype=FIRMWARE&SubClassID=1169

STMicroelectronics

HOME COMPANY CONTACTS PRESS LOGIN

Resource List

Search [ ] in ALL

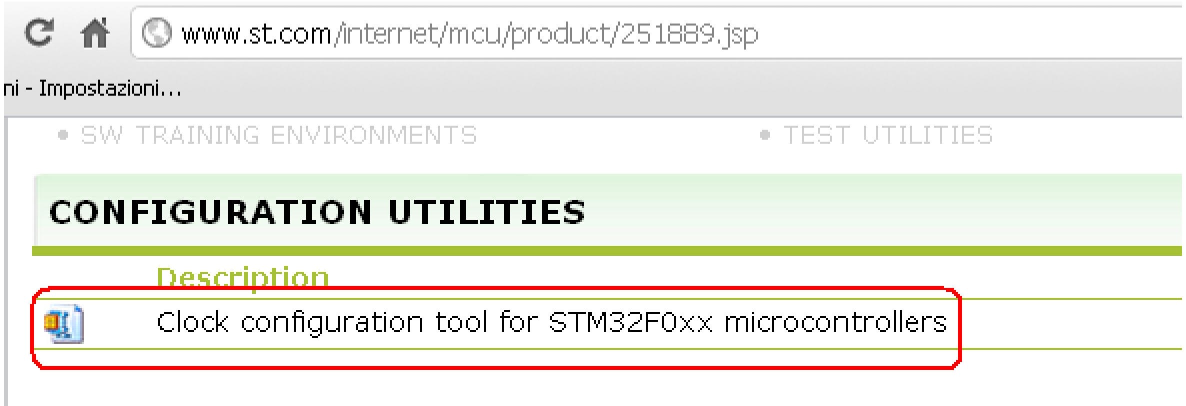
Resource Type: Firmware (46 Items)

Part Number	Link	Resource Title	Version	Associated with...
STM32F103C8...	📁	Using the STM32F101xx and STM32F103xx DMA controller	2.0.0	Resources
	📁	STM32F10xxx in-application programming using the USART	7.0	Resources
STM32F101CB...	📁	How to achieve 32-bit timer resolution using the link system in STM32F101xx and STM32F103xx microcontrollers	3.0.0	Resources
STM32F103C8...	📁	EEPROM emulation in STM32F101xx and STM32F103xx microcontrollers	3.1.0	Resources
STM32F103C8...	📁	Smartcard interface with the STM32F101xx and STM32F103xx	1.0	Resources
STM32F101CB...	📁	STM32F101xx and STM32F103xx low-power modes	2.0.0	Resources
STM32F101CB...	📁	Improving STM32F101xx and STM32F103xx ADC resolution by oversampling	1.0	Resources
STM32F103RE...	📁	How to use the high-density STM32F103xx microcontroller to play audio files with an external I²S audio codec	2.0.0	Resources
STM32F103VE...	📁	TFT LCD interfacing with the high-density STM32F10xxx FSMC	2.0.0	Resources
STM32F101CB...	📁	STM32F10xxx Speex library firmware STM32, StdPeriph Lib, speex, audio	2.0.0	Resources
STM32F103C8...	📁	Driving bipolar stepper motors using a medium-density STM32F103xx microcontroller	2.0.0	Resources
STM32F101CB...	📁	Clock/calendar implementation on the STM32F10xxx microcontroller RTC	1.0	Resources
STM32F103C8...	📁	STM32F101xx and STM32F103xx medium- and high-density devices: advanced I²C examples	4.0	Resources
	📁	STM32F10xxx internal RC oscillator (HSI) calibration	2.0.0	Resources
STM32F103RE...	📁	Implementing the ADPCM algorithm in high-density STM32F103xx microcontrollers	2.0.	Resources

STM32xxx Library
EXCEL worksheet for configuring the system clocks

Also for the STM32F0 family was released an EXCEL worksheet to guide us to set the system clocks; this tool generates the configuration file named:
system_stm32f0xx.c
which is then included in our project.

You find the EXCEL worksheet here:
http://www.st.com/internet/mcu/product/251889.jsp
When you are on the page mentioned above you must click on:
DESIGN SUPPORT
and then you have to scroll down for find:
Clock configuration tool for STM32F0xx microcontrollers (see below).


www.st.com/internet/mcu/product/251889.jsp

ni - Impostazioni...

- SW TRAINING ENVIRONMENTS
- TEST UTILITIES

CONFIGURATION UTILITIES

Description

Clock configuration tool for STM32F0xx microcontrollers

STM32xxx Library
What do you need to develop on F0

- **Data Sheet** - The manual of the MCU that you want to use
- **Reference Manual** of the STM32F0
- Also see, if there is, the **Errata Sheet**
- **Library**

Software Tools that to be immediately compatible with the STMF0 libraries must be selected from:

- **ATOLLIC**
- **KEIL**
- **IAR**
- **TASKING**

these software tools are also available for free, some up to 8K and some up to 32K (see KEIL)

Emulator JTAG e/o SWD, I suggest the **ST-LINK-v2**

To start using the STM32F0 I suggest the **STM32F0-Discovery** that also contains the ST-LINK-v2 and its cost is less than 10€.

STM32xxx Library
Basic things to remember

All the STM32 at startup have:
- **Internal Oscillator active** (HSI)
- **All the Peripherals are OFF**

Following this it is essential to remember:
Select the oscillator you want to use.
Enable the peripherals which are obtained by bringing the clock to the peripheral and/or peripherals that you want to use. To do this you use the command:
RCC_xxxxxx, (see example below).
For each peripheral you use is a must declare a **data structure** like this:

PeriphName_**InitTypeDef**   PeriphName_**InitStructure;**

This structure will be filled with data to the device configuration (see example below).
To access to the STM libraries you must include the file: **#include "stm32f0xx.h"**

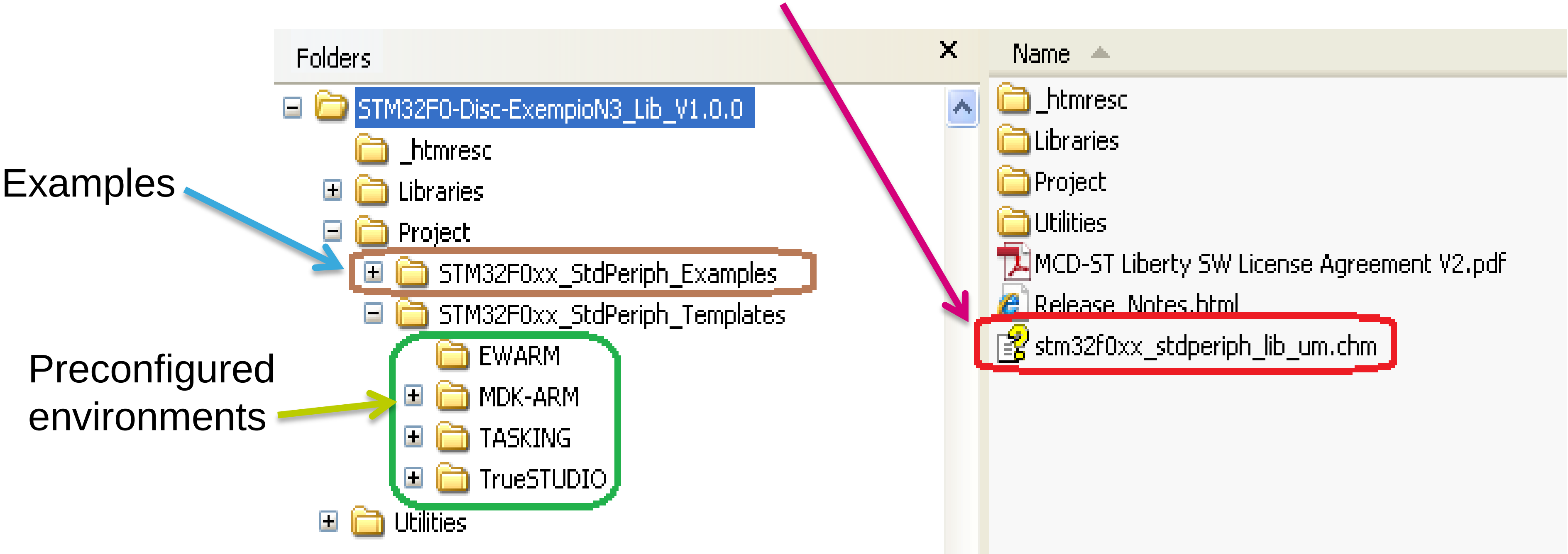
To quickly learn the use of STM32F0, I suggest to use the STM libraries which are: **CMSIS**, **ANSI C**, **Class B** and **MISRA C** compliant.
These libraries, that contain all the elements needed to manage all the peripherals, also contain numerous examples of applications ready for use and for each example there is a file named: **readme.txt**
that explains what does the example and explains the setup to use.

STM32xxx Library
Folder

The documentation of STM library is created using **DOXIGEN** that gives us the advantage to have the manual updated just a new version of the library is released, but in contrast is not present a manual in standard pdf format.

The manual is called:

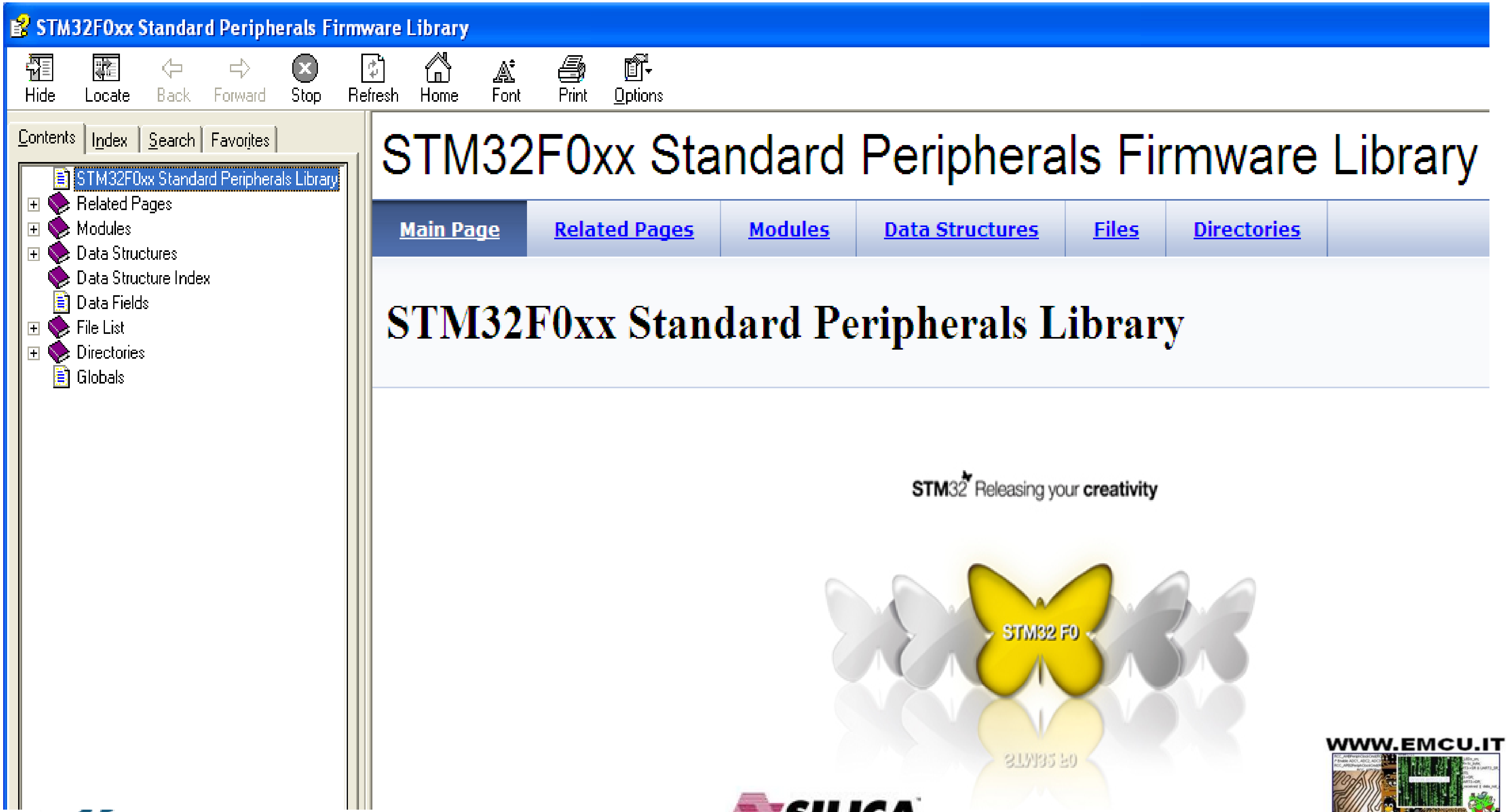
stm32f0xx_stdperiph_lib_um.chm



Examples
Preconfigured environments
Folders
STM32F0-Disc-ExempioN3_Lib_V1.0.0
_htmresc
Libraries
Project
STM32F0xx_StdPeriph_Examples
STM32F0xx_StdPeriph_Templates
EWARM
MDK-ARM
TASKING
TrueSTUDIO
Utilities
Name
_htmresc
Libraries
Project
Utilities
MCD-ST Liberty SW License Agreement V2.pdf
Release_Notes.html
stm32f0xx_stdperiph_lib_um.chm

STM32xxx Library
Manual

Opening the library manual appears the figure below.



STM32F0xx Standard Peripherals Firmware Library

- STM32F0xx Standard Peripherals Library
- ⊞ Related Pages
- ⊞ Modules
- ⊞ Data Structures
- Data Structure Index
- Data Fields
- ⊞ File List
- ⊞ Directories
- Globals

STM32F0xx Standard Peripherals Firmware Library

STM32F0xx Standard Peripherals Library

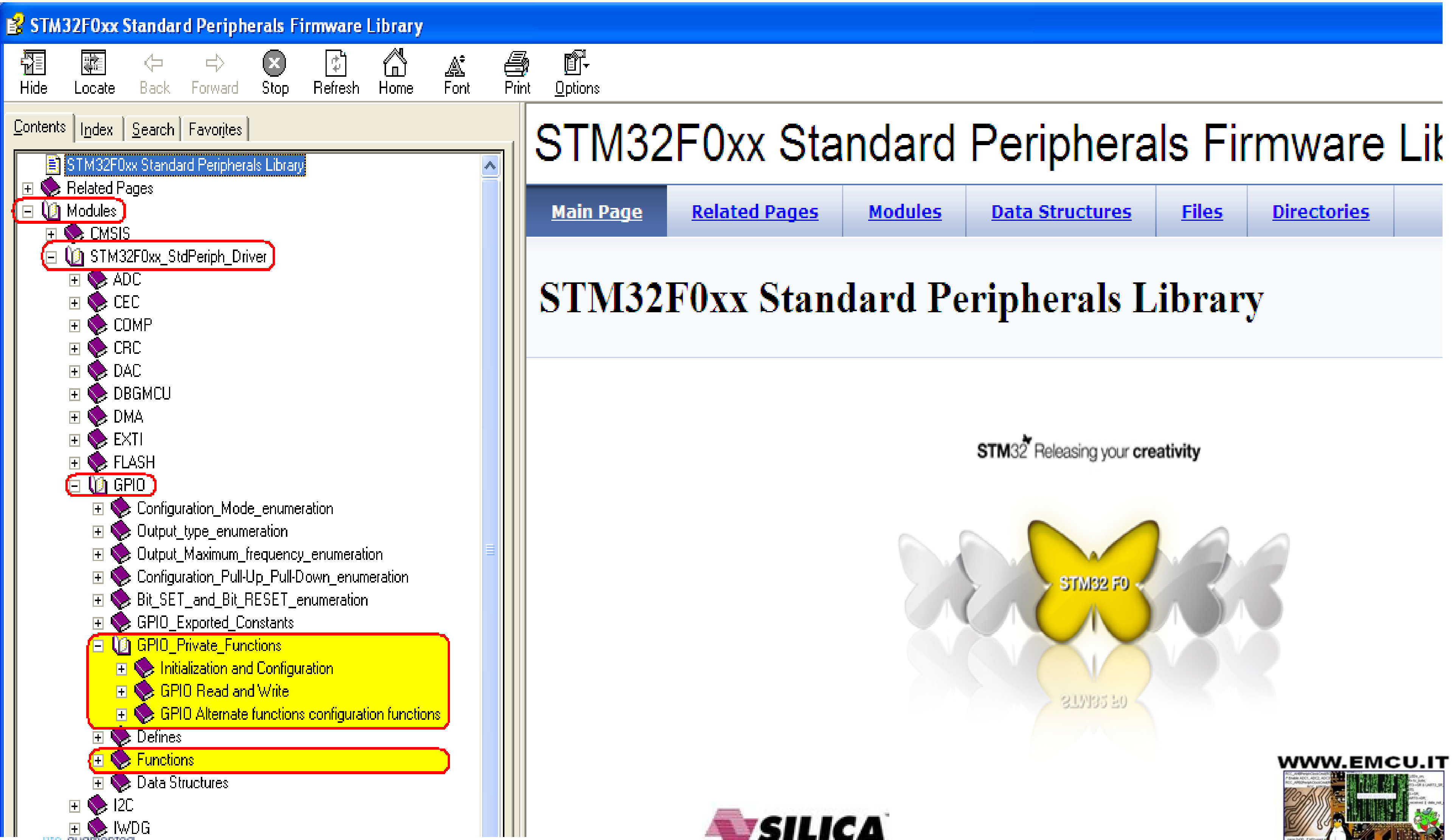
life.augmented

SILICA An Avnet Company

WWW.EMCU.IT

STM32xxx Library
Manual

What we need is to know the functions that we have at disposal to use the peripherals of STM32F0.


STM32F0xx Standard Peripherals Firmware Library
Hide
Locate
Back
Forward
Stop
Refresh
Home
Font
Print
Options
Contents
Index
Search
Favorites
STM32F0xx Standard Peripherals Library
Related Pages
Modules
CMSIS
STM32F0xx_StdPeriph_Driver
ADC
CEC
COMP
CRC
DAC
DBGMCU
DMA
EXTI
FLASH
GPIO
Configuration_Mode_enumeration
Output_type_enumeration
Output_Maximum_frequency_enumeration
Configuration_Pull-Up_Pull-Down_enumeration
Bit_SET_and_Bit_RESET_enumeration
GPIO_Exported_Constants
GPIO_Private_Functions
Initialization and Configuration
GPIO Read and Write
GPIO Alternate functions configuration functions
Defines
Functions
Data Structures
I2C
IWDG
STM32F0xx Standard Peripherals Firmware Lib
Main Page
Related Pages
Modules
Data Structures
Files
Directories
STM32F0xx Standard Peripherals Library
STM32 Releasing your creativity
STM32 F0

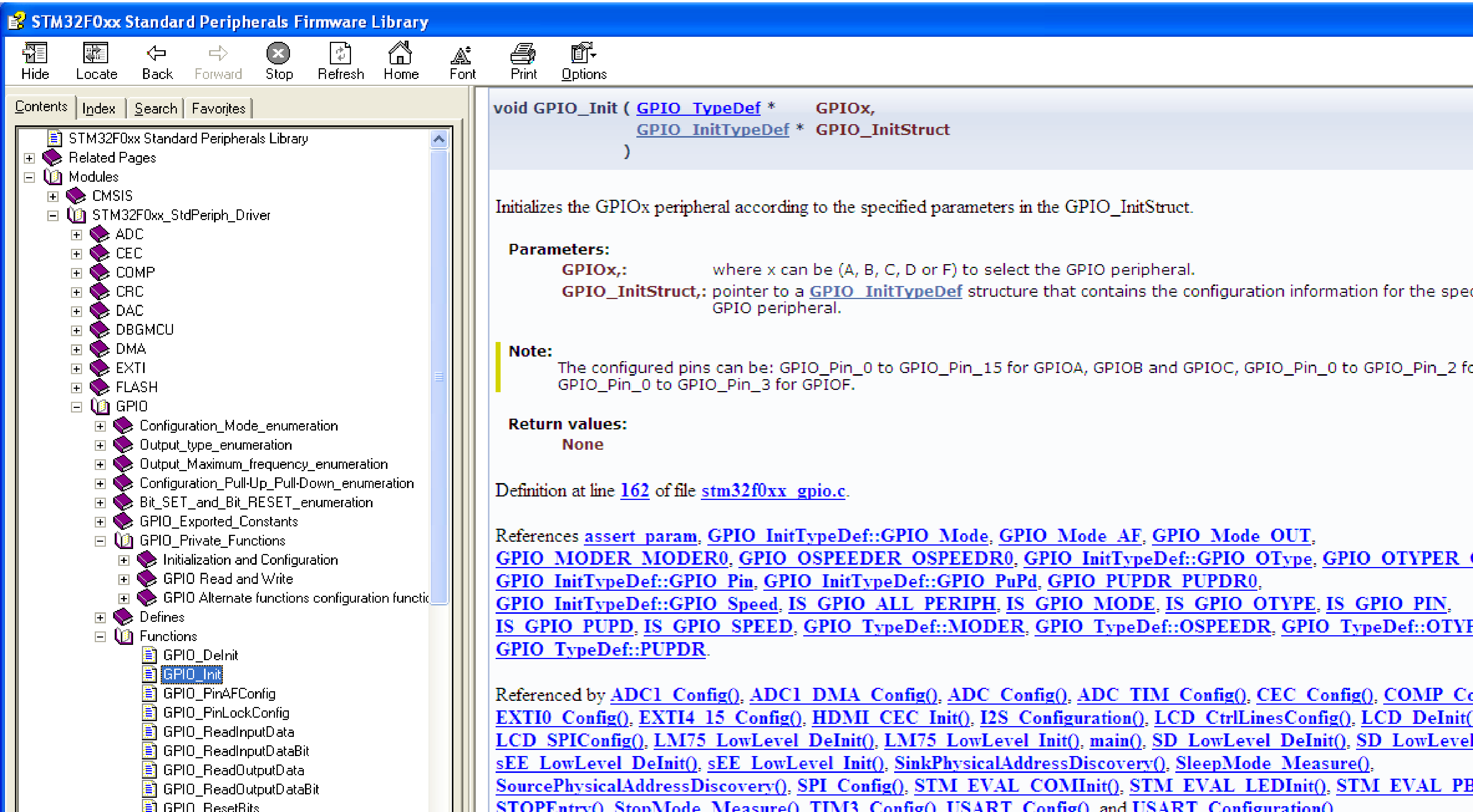
life.augmented

SILICA
An Avnet Company

WWW.EMCU.IT

STM32xxx Library **Manual**

Assuming that we want to see how to **initialize GPIO**, first select: **Modules→STM32F0xx_StdPeriph_Driver→GPIO→Functions→GPIO_Init** and will open the page with explanations, see below.



- STM32F0xx Standard Peripherals Library
- ⊞ Related Pages
- ⊟ Modules
  - ⊞ CMSIS
  - ⊟ STM32F0xx_StdPeriph_Driver
    - ⊞ ADC
    - ⊞ CEC
    - ⊞ COMP
    - ⊞ CRC
    - ⊞ DAC
    - ⊞ DBGMCU
    - ⊞ DMA
    - ⊞ EXTI
    - ⊞ FLASH
    - ⊟ GPIO
      - ⊞ Configuration_Mode_enumeration
      - ⊞ Output_type_enumeration
      - ⊞ Output_Maximum_frequency_enumeration
      - ⊞ Configuration_Pull-Up_Pull-Down_enumeration
      - ⊞ Bit_SET_and_Bit_RESET_enumeration
      - ⊞ GPIO_Exported_Constants
      - ⊟ GPIO_Private_Functions
        - ⊞ Initialization and Configuration
        - ⊞ GPIO Read and Write
        - ⊞ GPIO Alternate functions configuration functic
      - ⊞ Defines
      - ⊟ Functions
        - GPIO_DeInit
        - GPIO_Init
        - GPIO_PinAFConfig
        - GPIO_PinLockConfig
        - GPIO_ReadInputData
        - GPIO_ReadInputDataBit
        - GPIO_ReadOutputData
        - GPIO_ReadOutputDataBit
        - GPIO_ResetBits

```
void GPIO_Init ( GPIO_TypeDef *      GPIOx,
                 GPIO_InitTypeDef *  GPIO_InitStruct
               )
```

Initializes the GPIOx peripheral according to the specified parameters in the GPIO_InitStruct.

Parameters:

GPIOx,: where x can be (A, B, C, D or F) to select the GPIO peripheral.

GPIO_InitStruct,: pointer to a GPIO_InitTypeDef structure that contains the configuration information for the spec GPIO peripheral.

Note:

The configured pins can be: GPIO_Pin_0 to GPIO_Pin_15 for GPIOA, GPIOB and GPIOC, GPIO_Pin_0 to GPIO_Pin_2 fo GPIO_Pin_0 to GPIO_Pin_3 for GPIOF.

Return values:

None

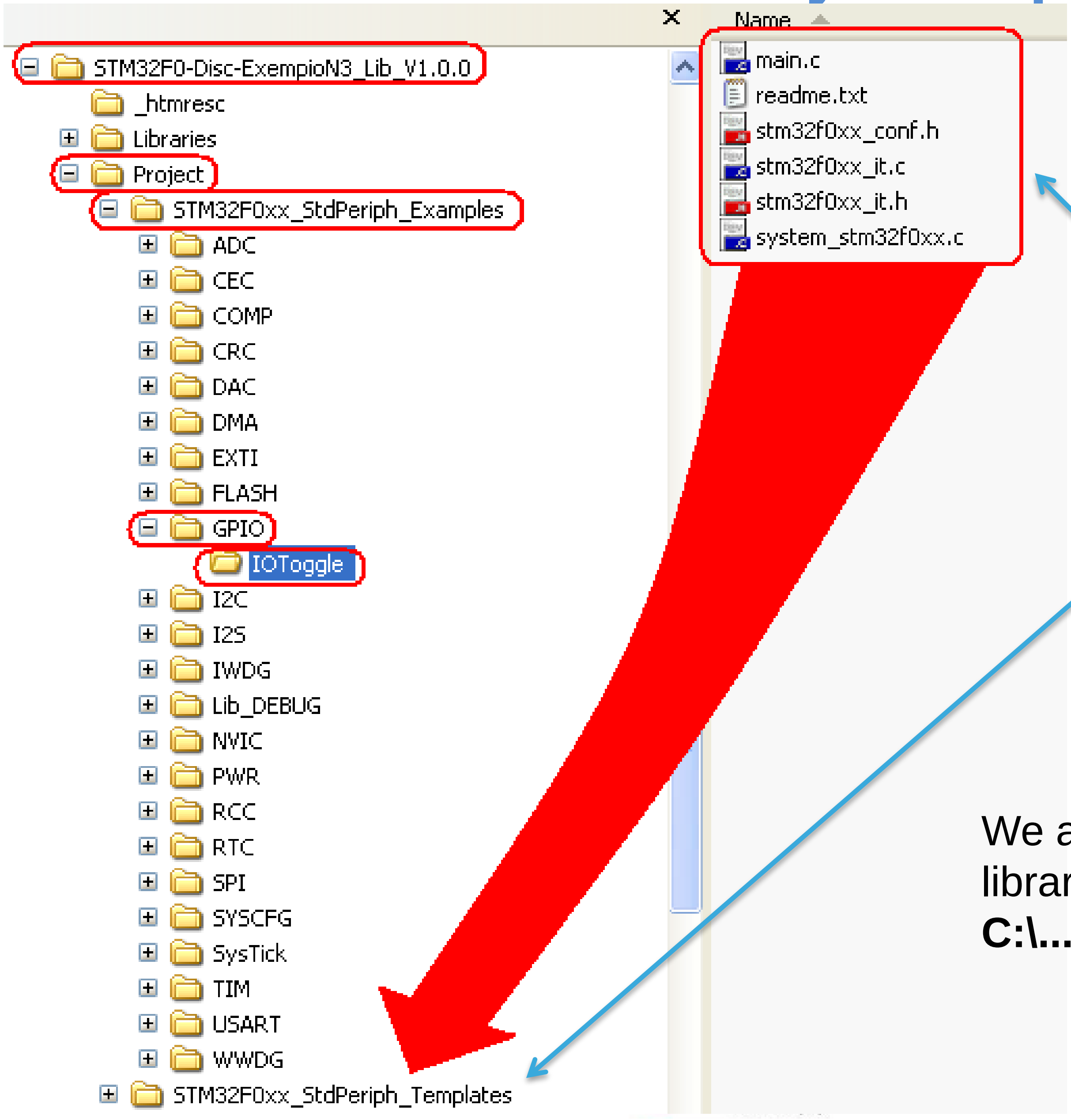
Definition at line 162 of file stm32f0xx_gpio.c.

References assert_param, GPIO_InitTypeDef::GPIO_Mode, GPIO_Mode_AF, GPIO_Mode_OUT, GPIO_MODER_MODER0, GPIO_OSPEEDER_OSPEEDR0, GPIO_InitTypeDef::GPIO_OType, GPIO_OTYPER_(GPIO_InitTypeDef::GPIO_Pin, GPIO_InitTypeDef::GPIO_PuPd, GPIO_PUPDR_PUPDR0, GPIO_InitTypeDef::GPIO_Speed, IS_GPIO_ALL_PERIPH, IS_GPIO_MODE, IS_GPIO_OTYPE, IS_GPIO_PIN, IS_GPIO_PUPD, IS_GPIO_SPEED, GPIO_TypeDef::MODER, GPIO_TypeDef::OSPEEDR, GPIO_TypeDef::OTYP GPIO_TypeDef::PUPDR.

Referenced by ADC1_Config(), ADC1_DMA_Config(), ADC_Config(), ADC_TIM_Config(), CEC_Config(), COMP_Co EXTI0_Config(), EXTI4_15_Config(), HDMI_CEC_Init(), I2S_Configuration(), LCD_CtrlLinesConfig(), LCD_DeInit(LCD_SPIConfig(), LM75_LowLevel_DeInit(), LM75_LowLevel_Init(), main(), SD_LowLevel_DeInit(), SD_LowLevel sEE_LowLevel_DeInit(), sEE_LowLevel_Init(), SinkPhysicalAddressDiscovery(), SleepMode_Measure(), SourcePhysicalAddressDiscovery(), SPI_Config(), STM_EVAL_COMInit(), STM_EVAL_LEDInit(), STM_EVAL_PE STOPEntry(), StopMode_Measure(), TIM3_Config(), USART_Config(), and USART_Configuration().

STM32xxx Library

How to use the library example



STM32F0-Disc-ExempioN3_Lib_V1.0.0
- _htmresc
- Libraries
- Project
  - STM32F0xx_StdPeriph_Examples
    - ADC
    - CEC
    - COMP
    - CRC
    - DAC
    - DMA
    - EXTI
    - FLASH
    - GPIO
      - IOToggle
    - I2C
    - I2S
    - IWDG
    - Lib_DEBUG
    - NVIC
    - PWR
    - RCC
    - RTC
    - SPI
    - SYSCFG
    - SysTick
    - TIM
    - USART
    - WWDG
  - STM32F0xx_StdPeriph_Templates

Name
- main.c
- readme.txt
- stm32f0xx_conf.h
- stm32f0xx_it.c
- stm32f0xx_it.h
- system_stm32f0xx.c

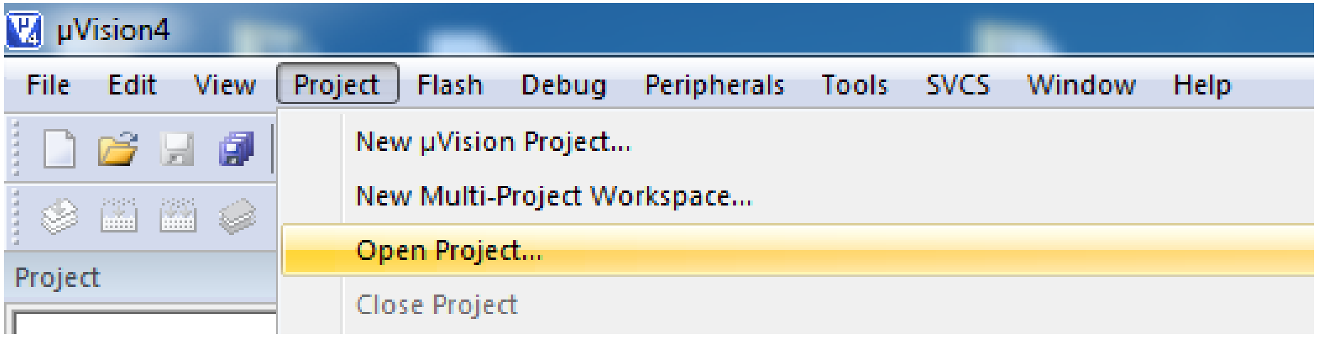
Copy and **Paste**

Do this

We assume that the library are in the folder: **C:\.....\HOn1**

STM32xxx Library
How to use the library example

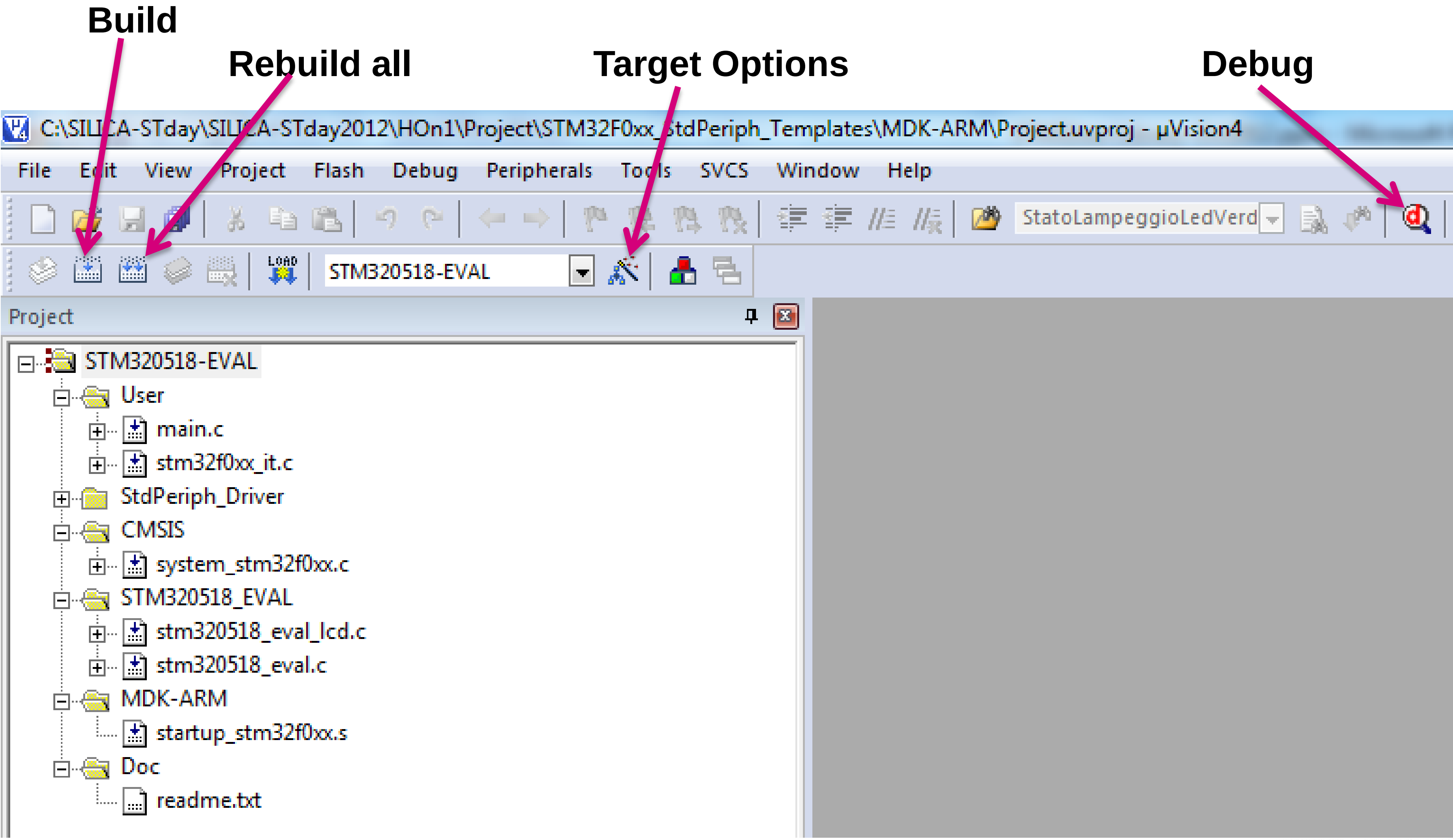
We assume that the library is in the folder:
C:\.....\HOn1
Now open your C Compiler, we assume that we use **KEIL**.
Choose: **Project -> Open Project**



And open the: **Project.uvproj**
that is in the folder:
C:\.....\HOn1\Project\STM32F0xx_StdPeriph_Templates\MDK-ARM

STM32xxx Library

How to use the library example

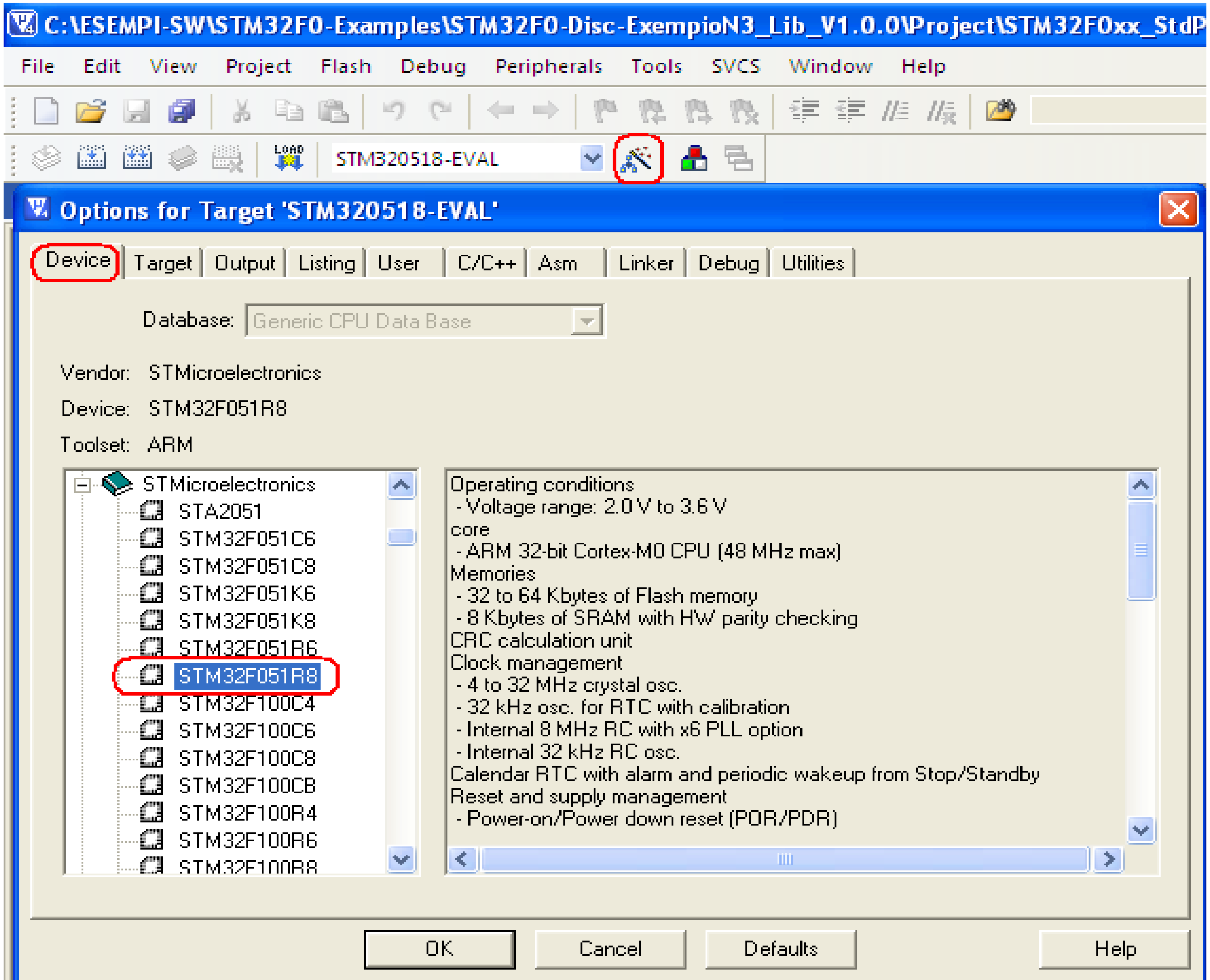


C:\SILICA-STday\SILICA-STday2012\HOn1\Project\STM32F0xx_StdPeriph_Templates\MDK-ARM\Project.uvproj - µVision4

```
STM320518-EVAL
  User
    main.c
    stm32f0xx_it.c
  StdPeriph_Driver
  CMSIS
    system_stm32f0xx.c
  STM320518_EVAL
    stm320518_eval_lcd.c
    stm320518_eval.c
  MDK-ARM
    startup_stm32f0xx.s
  Doc
    readme.txt
```

STM32xxx Library
How to use the library example

Now check that the environment configurations of the KEIL are correct.
For doing this, click on the icon named Target Options and check to have the same configuration displayed below.



C:\ESEMPI-SW\STM32F0-Examples\STM32F0-Disc-EsempioN3_Lib_V1.0.0\Project\STM32F0xx_StdP

Options for Target 'STM320518-EVAL'

Database: Generic CPU Data Base

Vendor: STMicroelectronics

Device: STM32F051R8

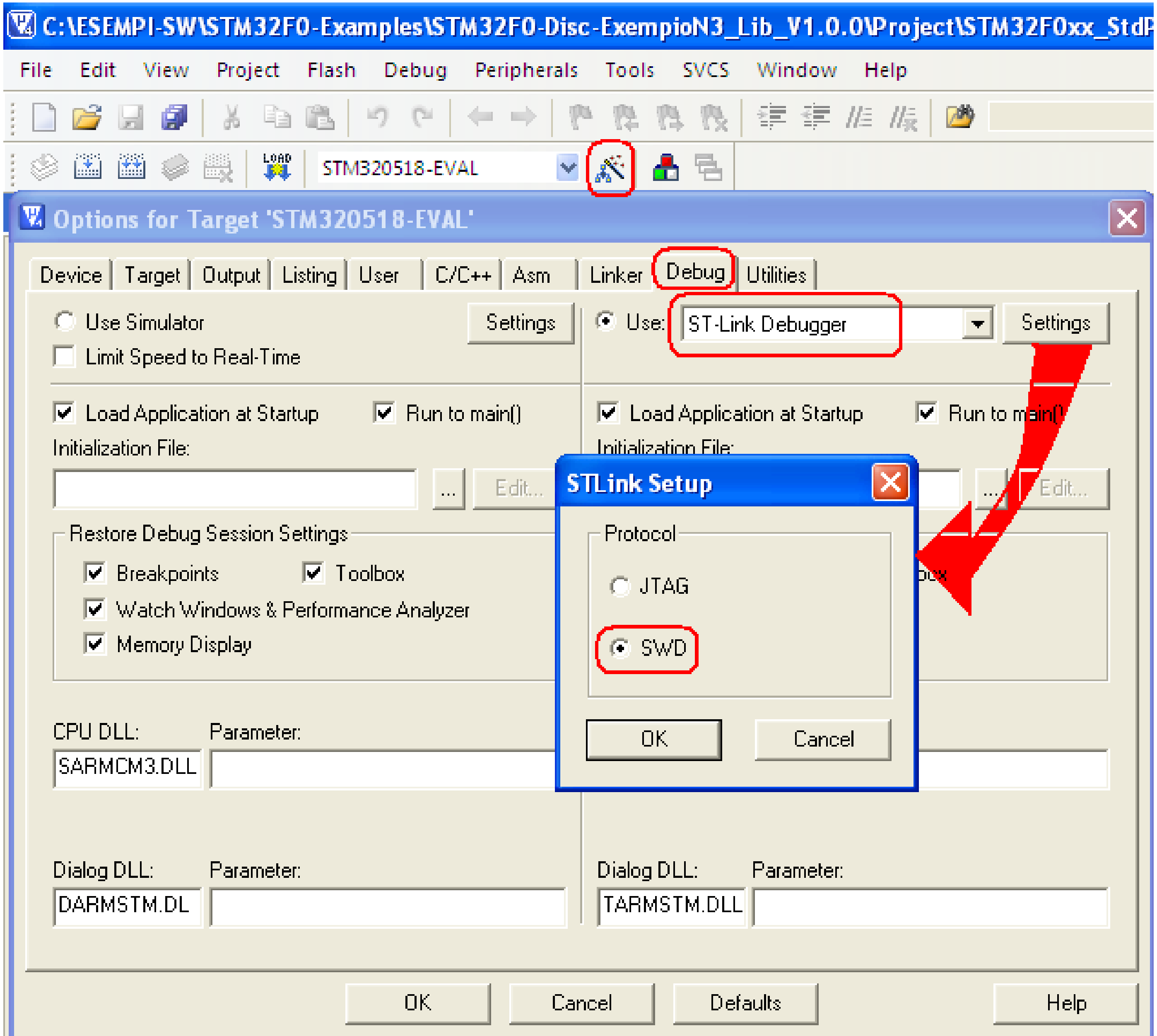
Toolset: ARM

Operating conditions
- Voltage range: 2.0 V to 3.6 V
core
- ARM 32-bit Cortex-M0 CPU (48 MHz max)
Memories
- 32 to 64 Kbytes of Flash memory
- 8 Kbytes of SRAM with HW parity checking
CRC calculation unit
Clock management
- 4 to 32 MHz crystal osc.
- 32 kHz osc. for RTC with calibration
- Internal 8 MHz RC with x6 PLL option
- Internal 32 kHz RC osc.
Calendar RTC with alarm and periodic wakeup from Stop/Standby
Reset and supply management
- Power-on/Power down reset (POR/PDR)

life.augmented

WWW.EMCU.IT

STM32xxx Library
How to use the library example



C:\ESEMPI-SW\STM32F0-Examples\STM32F0-Disc-ExempioN3_Lib_V1.0.0\Project\STM32F0xx_StdP

STM320518-EVAL

Options for Target 'STM320518-EVAL'

○ Use Simulator

□ Limit Speed to Real-Time

☑ Load Application at Startup ☑ Run to main()

Initialization File:

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows & Performance Analyzer

☑ Memory Display

CPU DLL: SARMCM3.DLL Parameter:

Dialog DLL: DARMSTM.DL Parameter:

● Use: ST-Link Debugger

☑ Load Application at Startup ☑ Run to main()

Initialization File:

Dialog DLL: TARMSTM.DLL Parameter:

STLink Setup

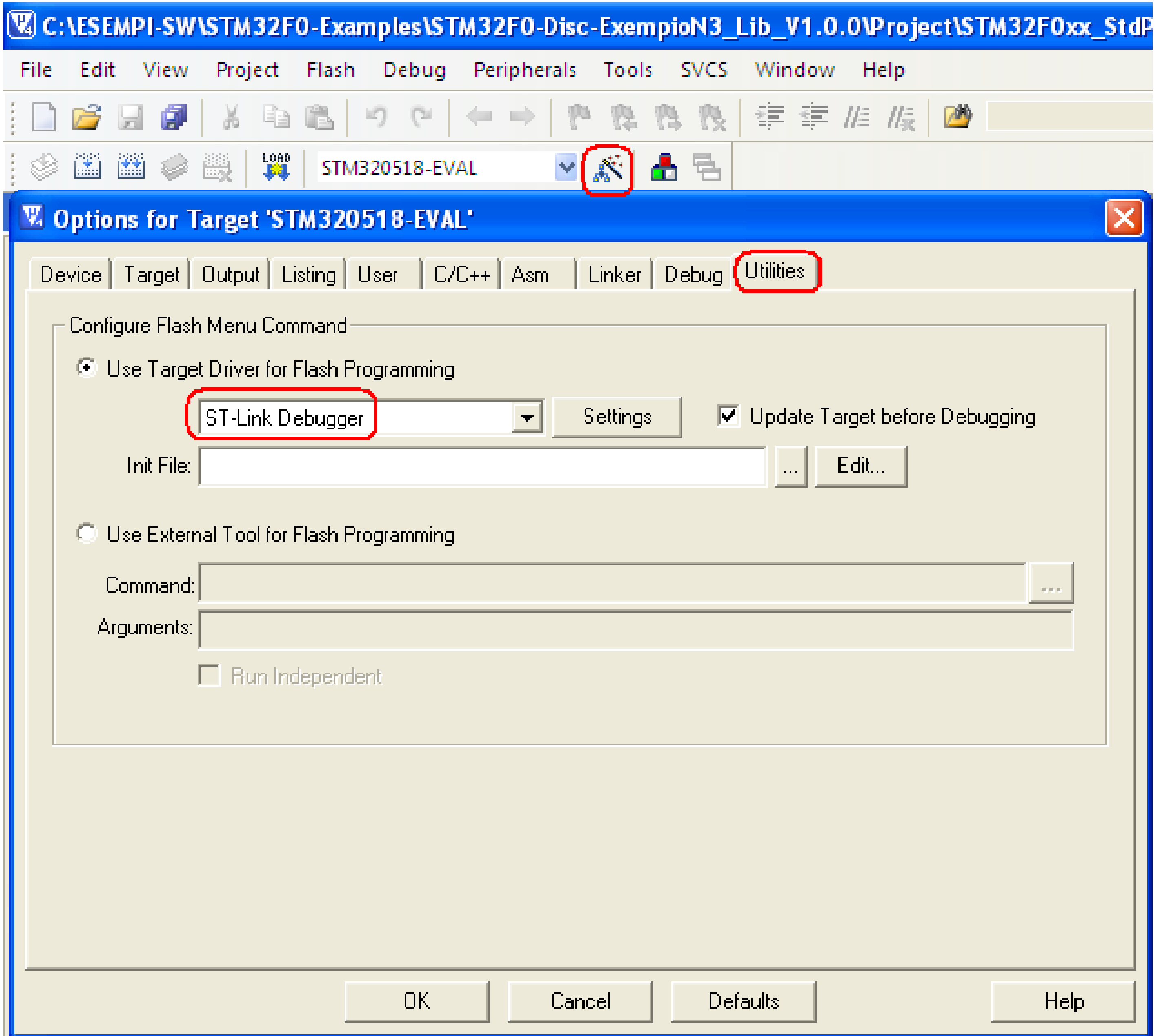
Protocol

○ JTAG

● SWD

STM32xxx Library

How to use the library example



C:\ESEMPI-SW\STM32F0-Examples\STM32F0-Disc-ExempioN3_Lib_V1.0.0\Project\STM32F0xx_StdP

Options for Target 'STM320518-EVAL'

Configure Flash Menu Command

● Use Target Driver for Flash Programming

ST-Link Debugger

☑ Update Target before Debugging

Init File:

○ Use External Tool for Flash Programming

Command:

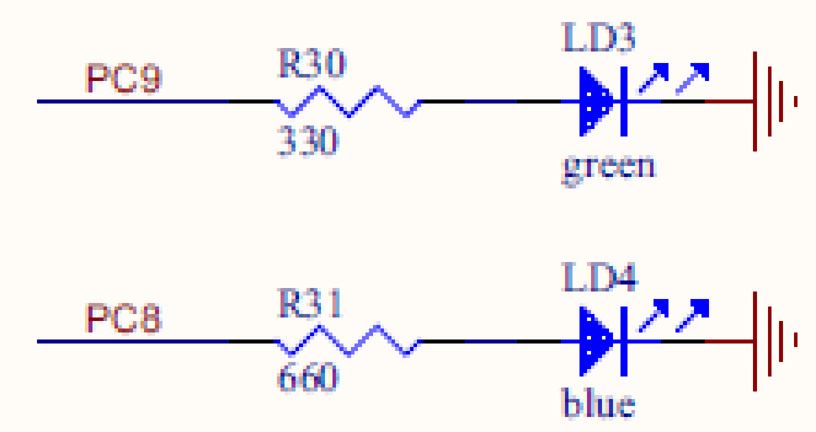
Arguments:

□ Run Independent

STM32xxx Library
How to use the library example

The examples inside the STM32F0 Library are ready to use with the eva-board: **STM320518-EVAL**

For using the STM32F0 Library examples on **STM32F0-Discovery** is necessary remap some I/O, LEDs, Push Buttons, etc.



STM32xxx Library
How to use the library example

```
/* Configure PC8 and PC9 in output pushpull mode */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_9;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_Init(GPIOC, &GPIO_InitStructure);
```

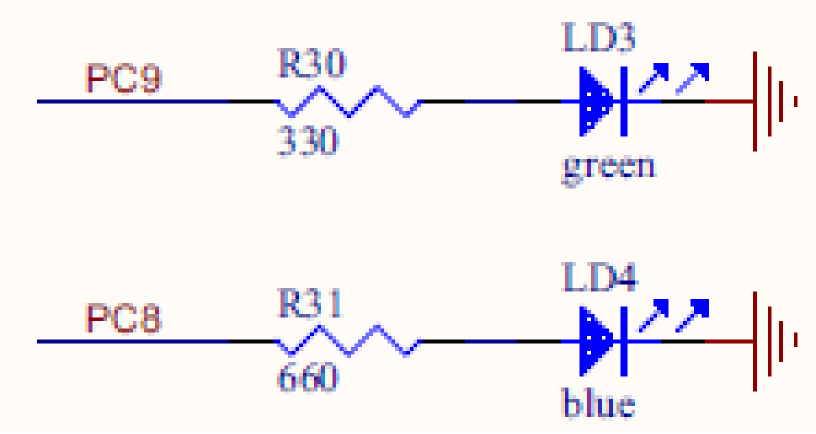
From the file: **readme.txt**
we note that:
In this example, HCLK is configured at 48 MHz so PC10 and PC11 toggles at 12MHz.
We are not interested in going to these frequencies, but we are interested to see visually the ON/OFF of the LEDs.
To do this, erase all the contents of:

```
while (1)
{
…
…
}
```

STM32xxx Library

How to use the library example

```
while (1)
 {
		// Set BITs
		GPIO_? ? ? ? (? ? ? ?, GPIO_Pin_? | GPIO_Pin_?);
		Delay(0xFFFF);
		// Reset BITs
		GPIO_? ? ? ? (? ? ? ?, GPIO_Pin_? | GPIO_Pin_?);
		Delay(0xFFFF);
 }
```



STM32xxx Library
How to use the library example

```
/* Private functions ------------------------------------------------------------*/
void Delay(long nCount);


// Function Delay
void Delay(long nCount)
{
        long n=0;

        for (n=0; n<1000000; n++)
        {
                while (nCount != 0)
                {
                        nCount--;
                }
        }
}
```

STM32xxx Library

```
/* Includes ------------------------------------------------------------------*/
#include "stm32f0xx.h“

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
#define BSRR_VAL 0x0C00

/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
GPIO_InitTypeDef        GPIO_InitStructure;

/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/
void Delay(long nCount);

int main(void)
{…………
```

STM32xxx Library

```
int main(void)
{
  /* GPIOC Periph clock enable */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);

  /* Configure PC8 and PC9 in output pushpull mode */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_9;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOC, &GPIO_InitStructure);

  while (1)
  {
    GPIO_SetBits(GPIOC, GPIO_Pin_8 | GPIO_Pin_9);
    Delay(0xFFFF);
    GPIO_ResetBits(GPIOC, GPIO_Pin_8 | GPIO_Pin_9);
    Delay(0xFFFF);
  }
}
```

STM32xxx Library

For more examples see here:
http://www.emcu.it/STM32F0xx/STM32F0xx.html#Tutorial