

Getting started with STM32F030 Value Line Discovery development tools

Introduction

This document describes the software, firmware environment and development recommendations required to build an application around the STM32F0308-DISCOVERY board (32F0308DISCOVERY) with demonstration firmware (STSW-STM32140).

It presents the firmware applications package provided within this board with details on its architecture and contents. It provides guidelines to novice users on how to build and run a sample application and allows them to create and build their own application.

This document is structured as follows:

- System requirements to use this board and how to run the built-in demonstration are provided in *Section 1: Getting started*.
- Section 2 describes the firmware applications package.
- Section 4 presents development toolchain installation and overview of ST-LINK/V2 interface.
- Section 5, Section 6, Section 7, and Section 8 introduce how to use the following software development toolchains:
 - IAR Embedded Workbench[®] for ARM (EWARM) by IAR Systems
 - Microcontroller Development Kit for ARM (MDK-ARM) by Keil™
 - TrueSTUDIO[®] by Atollic

Although this user manual cannot cover all the topics relevant to software development environments, it demonstrates the first basic steps necessary to get started with the compilers/debuggers.

Reference documents:

- STM32F0308-DISCOVERY high-performance discovery board data brief
- STM32F0308-DISCOVERY peripherals firmware examples (AN4062)
- STM32F0xx reference manual (RM0360)
- STM32F030x4 STM32F030x6 STM32F030x8 datasheet

The above documents are available at www.st.com/stm32f0-discovery.

Contents

1	Gett	Getting started 4						
	1.1	System requirements	-					
	1.2	Running the built-in demonstration	ŕ					
2	Desc	cription of the firmware package5	;					
	2.1	Libraries folder	,					
		2.1.1 CMSIS subfolder	,					
		2.1.2 STM32F0xx_StdPeriph_Driver subfolder6	;					
	2.2	Project folder	ĵ					
		2.2.1 Demonstration subfolder	;					
		2.2.2 Master_Workspace subfolder6	;					
		2.2.3 Peripheral_Examples subfolder	j					
	2.3	Utilities folder	;					
3	Bina	ry images for reprogramming firmware applications	,					
4	ST-L	INK/V2 installation and development8						
5	Usin	ig IAR Embedded Workbench [®] for ARM9)					
	5.1	Building an existing EWARM project)					
6	MDK	-ARM toolchain						
7	True	STUDIO toolchain						
8	Revi	sion history						



List of figures

Hardware environment	4
Hardware environment	5
IAR Embedded Workbench IDE (Integrated Design Environment)	9
EWARM project successfully compiled.	9
Download and Debug button1	0
IAR Embedded Workbench debugger screen1	0
Go button	1
uVision4 IDE	2
MDK-ARM project successfully compiled1	2
Start/Stop Debug Session button	3
MDK-ARM debuger screen	3
Run button	3
TrueSTUDIO [®] workspace launcher dialog box	4
Atollic TrueSTUDIO [®] import source select dialog box	4
Atollic TrueSTUDIO [®] import projects dialog box	5
TrueSTUDIO [®] project successfully compiled	6
TrueSTUDIO [®] debug window	6
	Hardware environment Hardware environment IAR Embedded Workbench IDE (Integrated Design Environment) EWARM project successfully compiled Download and Debug button 1 IAR Embedded Workbench debugger screen 1 Go button 1 uVision4 IDE 1 MDK-ARM project successfully compiled 1 Start/Stop Debug Session button 1 MDK-ARM debuger screen 1 MDK-ARM debuger screen 1 MDK-ARM debuger screen 1 Atollic TrueSTUDIO [®] workspace launcher dialog box 1 Atollic TrueSTUDIO [®] import source select dialog box 1 Atollic TrueSTUDIO [®] project successfully compiled 1 TrueSTUDIO [®] debug window 1



1 Getting started

1.1 System requirements

Before running your application, you should establish the connection with the STM32F0308-DISCOVERY board as following.



Figure 1. Hardware environment

To run and develop any firmware applications on your STM32F0308-DISCOVERY board, the minimum requirements are as follows:

- Windows PC (2000, XP, Vista, 7)
- 'USB type A to Mini-B' cable, used to power the board (through USB connector CN1) from host PC and connect to the embedded ST-LINK/V2 for debugging and programming.

1.2 Running the built-in demonstration

The board comes with the demonstration firmware preloaded in the Flash memory. Follow the steps below to run it:

- Check the jumper position on the board, JP2 on, CN2 on (Discovery selected).
- Connect the STM32F0308-DISCOVERY board to a PC with a 'USB type A to Mini-B' cable through USB connector CN1 to power the board. Then red LEDs LED1 (PWR) and LED2 (COM) light up and green LED3 blinks.
- Press user button B1 (Button left corner of the board). The blinking of green LED3 changes according to clicks on user button B1.
- Each click on the USER push-button is confirmed by blue LED4.



2 Description of the firmware package

The STM32F0308-DISCOVERY firmware applications are provided in one single package and supplied in one single zip file. The extraction of the zip file generates one folder, *STM32F0308-Discovery_FW_VX.Y.Z*, which contains the following subfolders:



Figure 2. Hardware environment

1. VX.Y.Z refer to the package version, ex. V1.0.0

2.1 Libraries folder

This folder contains the Hardware Abstraction Layer (HAL) for STM32F0xx devices.

2.1.1 CMSIS subfolder

This subfolder contains the STM32F0xx and Cortex-M0 CMSIS files.

Cortex-M0 CMSIS files consist of:

 Core Peripheral Access Layer: contains name definitions, address definitions and helper functions to access Cortex-M0 core registers and peripherals. It defines also a device independent interface for RTOS Kernels that includes debug channel definitions.

STM32F0xx CMSIS files consist of:

- stm32f0xx.h: contains the definitions of all peripheral registers, bits, and memory mapping for STM32F0xx devices. The file is the unique include file used in the application programmer C source code, usually in main.c.
- system_stm32f0xx.c/.h: 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/APBx 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_stm32f030x8.s file.
- startup_stm32f030x8.s: provides the Cortex-M0 startup code and interrupt vectors for all STM32F0xx device interrupt handlers.



2.1.2 STM32F0xx_StdPeriph_Driver subfolder

This subfolder contains sources of STM32F0xx peripheral drivers.

Each driver consists of a set of routines and data structures covering all peripheral functionalities. The development of each driver is driven by a common API (application programming interface) which standardizes the driver structure, the functions and the parameter names.

Each peripheral has a source code file, *stm32f0xx_ppp.c*, and a header file, *stm32f0xx_ppp.h*. The *stm32f0xx_ppp.c* file contains all the firmware functions required to use the PPP peripheral.

2.2 Project folder

This folder contains the source files of the STM32F030 Discovery firmware applications.

2.2.1 Demonstration subfolder

This subfolder contains the demonstration source files with preconfigured project for EWARM, MDK-ARM and TrueSTUDIO[®] toolchains.

A binary image (*.hex) of this demonstration is provided under Binary subfolder. You can use any in-system programming tool to reprogram the demonstration using this binary image.

2.2.2 Master_Workspace subfolder

This subfolder contains, for some toolchains, a multi-project workspace allowing you to manage all the available projects (provided under the subfolders listed below) from a single workspace window.

2.2.3 Peripheral_Examples subfolder

This subfolder contains a set of examples for some peripherals with preconfigured projects for EWARM, MDK-ARM and TrueSTUDIO toolchains. See *Section 4* and *STM32F0308-DISCOVERY peripheral firmware examples,* AN4062, for further details.

2.3 Utilities folder

This folder contains the abstraction layer for the STM32F030 Discovery hardware. It provides the following drivers:

• *stm32f0308_discovery.c*: provides functions to manage the user push-button and 2 LEDs (LED3 and LED4).



3 Binary images for reprogramming firmware applications

This section describes how to use the provided binary images to reprogram the firmware applications. The STM32F030 Discovery firmware package contains binary images (*.hex) of the provided applications under Binary subfolder. You can use any in-system programming tool to reprogram the demonstration using this binary image.

to reprogram the firmware applications, use the "in-system programming tool" and:

- 1. Connect the STM32F030 Discovery board to a PC with a 'USB type A to Mini-B' cable through USB connector CN1 to power the board.
- 2. Make sure that the embedded ST-LINK/V2 is configured for in-system programming (both CN3 jumpers ON).
- Use *.hex binary (for example, \Project\Demonstration\Binary\STM32F0308-Discovery_Demonstration_V1.0.0.hex) with your preferred in-system programming tool to reprogram the demonstration firmware (ex. STM32 ST-LINK Utility, available for download from www.st.com).



4 ST-LINK/V2 installation and development

STM32F030 Discovery board includes an ST-LINK/V2 embedded debug tool interface that is supported by the following software toolchains:

IAR™ Embedded Workbench for ARM (EWARM) available from www.iar.com

The toolchain is installed by default in the C:\Program Files\IAR Systems\Embedded Workbench 6.5 directory on the PC's local hard disk.

After installing EWARM, install the ST-LINK/V2 driver by running the *ST-Link_V2_USB.exe* from *[IAR_INSTALL_DIRECTORY]**Embedded Workbench* 6.5\arm\drivers\ST-Link \ST-Link_V2_USBdriver.exe

RealView Microcontroller Development Kit (MDK-ARM) toolchain available from
 www.keil.com

The toolchain is installed by default in the C:\Keil directory on the PC's local hard disk; the installer creates a start menu μ Vision4 shortcut.

When connecting the ST-LINK/V2 tool, the PC detects new hardware and asks to install the ST-LINK_V2_USB driver. The "Found New Hardware wizard" appears and guides you through the steps needed to install the driver from the recommended location.

Atollic TrueSTUDIO[®] STM32 available from www.atollic.com
 The toolchain is installed by default in the C:\Program Files\Atollic directory on the PC's local hard disk.

The *ST-Link_V2_USB.exe* is installed automatically when installing the software toolchain.

Note: The embedded ST-LINK/V2 supports only SWD interface for STM32 devices.

Refer to the firmware package release notes for the version of the supporting development toolchains.



5 Using IAR Embedded Workbench[®] for ARM

5.1 Building an existing EWARM project

The following is the procedure for building an existing EWARM project.

Open the IAR Embedded Workbench[®] for ARM (EWARM).
 Figure 3 shows the basic names of the windows referred to in this document.



Figure 3. IAR Embedded Workbench IDE (Integrated Design Environment)

- 2. In the **File** menu, select **Open** and click **Workspace** to display the Open Workspace dialog box. Browse to select an *example* or *demonstration* or *template* workspace file and click **Open** to launch it in the Project window.
- 3. In the **Project** menu, select **Rebuild All** to compile your project.
- 4. If your project is successfully compiled, the following window in *Figure 4* is displayed.

Figure 4. EWARM project successfully compiled





If you need to change your project settings (Include and preprocessor defines), you need just to go through project options:

- For Include directories
- Project>Options...>C/C++ compiler>
- For pre-processor defines

Project>Options...C/C++ compiler>pre-processor>

5. In the IAR Embedded Workbench IDE, from the **Project menu**, select **Download and Debug** or, alternatively, click the **Download and Debug** button the in toolbar, to program the Flash memory and begin debugging.

Figure 5. Download and Debug button



6. The debugger in the IAR Embedded Workbench can be used to debug source code at C and assembly levels, set breakpoints, monitor individual variables and watch events during the code execution.

Owner North SysTick 38 SysTick 38 SysTick 38 SysTick 40 SysTick 40 SysTick 40 SysTick 40 SysTick 40 SysTick 40 SysTick 41 SysTick 41 SysTick 41 SysTick 42 SysTick 43 SysTick 44 SysTick 44 SysTick 45 SysTick 45 SysTick 45 SysTick 46 SysTick 47 SysTick 47 SysTick 47 SysTick 48 SysTick 48 <		Descendy Go to Disassembly Descendy 0x80002 int main(vo: (main: 0x80002	Memoy 36: 0xd002 38: 0x464d 38: 0x166d 38: 0x166d 38: 0x166d 38: 0x166d 38: 0x164d 42: 0x16b 42: 0x16b 44: 0xd000 45: 0xe7ta 48: 0x4770 VDC IROHanler dler: 1d) 50: 0x6560
y=Tick 38 Files 7.2 13 SysTick - SysTick		Goto Disessembly 0x80002; (max0002; main: * *	Memoy 36: 0xd002 30: 0x464d 33: 0x166d 33: 0x166d 33: 0x166d 33: 0x164d 40: 0x1624 40: 0x1624 44: 0x40f0 44: 0x40f0 44: 0x4770 40: 0x4770 40: 0x4770 40: 0x67fe id) 50: 0x6560
Files 12 B2 39 /* Firste typeld IG SyrEick - V 41 /* Firste sacro 41 IG DEVAPA 42 /* Firste sacro 41 IG DEVAPA 42 /* Firste sacro 42 IG STATEK-SQUPend 44 /* Firste sacro 43 6210 InitStructure: IG STATEK-SQUPend 44 /* Firste sacro 43 6210 InitStructure: IG STATEK-SQUPend 44 /* Firste sacro 44 45 IG Duput 45 /* Firste function prototypes 45 IG Duput 46 /* Firste functions 46 IG Duput 46 /* Firste functions 47 IG Duput 46 /* Firste functions 47 IG Duput 47 * Gorief Main program 48 IG Duput 47 * Gorief Main program 49 IG Duput 48 * Gorief Main program 49 IG Duput 49 /* (A this stage the microcontroller clock setting is already configured IG Duput 49 /* (A this is done through SystemInt() function main. IG IG Duput </td <td></td> <td>Disassembly 0x80002; int main(vo; (main: 0x80002;</td> <td>36: 0xd002 38: 0x464d 36: 0x1964 36: 0x1964 36: 0x1964 40: 0x1244 42: 0x111b 44: 0x676a 44: 0x4700 v0: 44: 0x4770 v0: 44: 0x470 v0: 44: 0x470 v0:</td>		Disassembly 0x80002; int main(vo; (main: 0x80002;	36: 0xd002 38: 0x464d 36: 0x1964 36: 0x1964 36: 0x1964 40: 0x1244 42: 0x111b 44: 0x676a 44: 0x4700 v0: 44: 0x4770 v0: 44: 0x470 v0: 44: 0x470 v0:
SysTick SysTick / 40 /* Pirvate define DEVARAM 41 /* Pirvate Macro DEVARAM 42 /* Pirvate Macro DESTM3PT0000-Discovery 43 /* Pirvate Macro District SysTick 54 /* Pirvate Macro District SysTick 54 /* Pirvate Macro District SysTick 44 /* Pirvate Macro District SysTick 54 /* Pirvate Macro District SysTick 44 /* Pirvate Macro District SysTick /* Pirvate Macro 44 District SysTick /* Pirvate Macro 45 District SysTick /* Pirvate Macro 45 District SysTick /* Pirvate Macro 46 District SysTick /* Pirvate Macro 47 District SysTick /* Pirvate Macro 47 District SysTick /* Pirvate Macro 47 Distr		0+80002 0+8	36: 0x4002 38: 0x46d 3a: 0x166d 3a: 0x166d 3a: 0x106d 3a: 0x102d 40: 0x102d 40: 0x102d 44: 0x40f0 44: 0x40f0 44: 0x40f0 44: 0x4770 v0: 0x02f70 40: 0x4770 40: 0x470 40: 0x47040 40: 0x
<pre> TVAXPAN</pre>	*/ */ */ tup	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	38: 0x464d 3a: 0x166d 3c: 0x1964 3e: 0x6022 40: 0x124 42: 0x1flb 44: 0x0f00 44: 0xe7fa 48: 0x4524 44: 0x4070 v0: 44: 0x4770 v0: 46: 0x710 v0: 46: 0x710v0: 46: 0x710 v0: 46: 0x710 v0:
Image: Static Constraint of the state o		0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 1nt main(vo: (main: 0x80002: ()	3a: 0x1=6d 3a: 0x1=64 3a: 0x6022 40: 0x1d24 42: 0x1d24 42: 0x1d24 44: 0x1d24 44: 0x1d24 44: 0x1d24 44: 0x1d24 45: 0x1=71 45: 0x1=71 45: 0x1=64 10 50: 0x1=64 10 10 10 10 10 10 10 10 10 10 10 10 10
<pre>a State10 units2 t Taindpelay:</pre>	*/ */ tup o	0 x80002: 0 x80	3c: 0x1964 3e: 0x6022 40: 0x1244 42: 0x1flb 44: 0x0f0 45: 0xx7fa 48: 0xbc32 48: 0xbc32 48: 0x4770 v0: 4c: 0x4770 v0c_IR0Handler 4e: 0xe7fe id) 50: 0xb580
Weet state00 unc2_c inlinguelay; 44 state function prototypes 45 * Private function prototypes 46 /* Private function prototypes 47 waid belay(_T0 unc3_c nTae); 48 * Brivate functions 49 /* 49 /* 40 * Brivate functions 51 /* 52 * Brivate functions 53 * Brivate functions 54 * Brivate functions 55 * Brivate functions 56 * Arrows 57 /* 58 * Gretal None 59 (* Int main program 50 (*/* At this stage the microcontroller clock setting is already configured 59 (*/* At this stage the microcontroller clock setting is already configured 50 (*/* At this stage the default setting of SystemInit() function, refer to system state system		0x80002; 0x80002; 0x80002; 0x80002; 0x80002; 0x80002; 0x80002; 0x80002; 3x80002; 3x80002; 0x80002; int main(vo; (xsin: \$\$0x80002]	3e: 0x6022 40: 0x1d24 42: 0x1d24 42: 0x1d1b 44: 0x0d00 46: 0xe7fa 48: 0xb-32 4a: 0x4770 v0: 4c: 0x4770 v0: 4c: 0x4770 v0: 4e: 0xe7fe id) 50: 0xb580
B Output	*/ */ Eup 20	0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 0x80002: 0x80002: B W WUDG_IRQHan: 0x80002: int main: 0x80002: (main: 0x80002!	40: 0x1d24 42: 0x1f1b 44: 0xd0f0 46: 0xe7fa 48: 0xbc32 4a: 0x4770 v0: 4c: 0x4770 WDG_IRQHandler 4ler: 4e: 0xe7fe id)
<pre></pre>	*/ */ 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.	0x80002 0x8000 0x8000 0x80002 0x80002 0x80002 0x80002 0x80002 0x800	42: 0x1f1b 44: 0xd0f0 46: 0xe7fa 48: 0xbc32 48: 0x4770 v0: 4c: 0x4770 WDG_IRQHandler dler: 4e: 0xe7fe id) 50: 0xb580
<pre>/* First functions</pre>	tup	0x80002 0x80002 0x80002 0x80002 0x80002 0x80002 0x80002 0x80002 int main(vc; (main; 0x80002	44: 0xd0f0 46: 0xe7fa 48: 0xbc32 4a: 0x4770 v0: 4c: 0x4770 WDG_IRQHandler dler: 4e: 0xe7fe id) 50: 0xb580
<pre></pre>	*/ L up o	0x80002 0x80002 0x80002 aeabi_idiv 0x80002 B W WDG_IRQHann 0x80002 (main: 0x80002	<pre>%</pre>
<pre></pre>	j L, up	0x80002 0x80002 acabi_idiv 0x80002 B W WDG_IRQHW 0x80002 int main(vo: (main: \$\$\$0x80002!	<pre>No. 100-11 48: 0xbc32 4a: 0x4770 v0: 4c: 0x4770 WDG_IRQHandler dler: 4e: 0xe7fe id) 50: 0xb580</pre>
<pre></pre>	i, up	0x80002 0x80002 	A: 0x4770 v0: 4c: 0x4770 WDG_IRQHandler dler: 4e: 0xe7fe id) 50 0xb580
<pre></pre>	t, 1000		40: 0x4770 V0: 4c: 0x4770 WDG_IRQHandler dler: 4e: 0xe7fe id) 50: 0xb580
<pre>53 * @perram Nome 54 * @perram Nome 55 * @perram Nome 56 * @perram Nome 57 * @perram Nome 58 * @perram Nome 59 * @perram Nome 59 * @perram Nome 59 * @perram Nome 50 * @perram Nome 50 * @perram Nome 50 * @perram Nome 51 * @perram Nome 52 * @perram Nome 53 * @perram Nome 54 * @perram Nome 54 * @perram Nome 55 * @perram Nome 56 * @perram Nome 57 * STM_SVAL_BEDInit(LED3); 57 * STM_SVAL_BEDInit(LED4); 58 * @perram Nome 59 * @perram Nome 50 * @perram Nome 50 * @perram Nome 50 * @perram Nome 50 * @perram Nome 51 * @perram Nome 52 * @perram Nome 53 * @perram Nome 54 * @perram Nome 55 * @perram Nome 56 * @perram Nome 57 * @perram Nome 58 * @perram Nome 59 * @perram Nome 59 * @perram Nome 50 * @perram Nome 51 * @perram Nome 52 * @perram Nome 53 * @perram Nome 54 * @perram Nome 54 * @perram Nome 55 * @perram Nome 56 * @perram Nome 57 * @perram Nome 58 * @perram Nome 59 * @perram Nome 50 * @perram Nome 51 * @perram Nome 52 * @perram Nome 52 * @perram Nome 53 * @perram Nome 54 * @perram Nome 54 * @perram Nome 54 * @perram Nome 55 * @perram Nome 56 * @perram Nome 57 * @perram Nome 58 * @perram Nome 59 * @perram Nome 50 * @per</pre>	t, tup		V0. 4c: 0x4770 WDG_IRQHandler dler: 4e: 0xe7fe id)
<pre></pre>	ł, .up	B WWDG_IRQHand 0x800024 int main(vo: { main: \$ 0x800025	WDG_IRQHandler dler: 4e: 0xe7fe id)
<pre>55 */**********************************</pre>	1, 1.100 2	B 00 WWDG_IRQHand 0x800024 int main(vo: { main: \$ 0x800023	dler: 4e: 0xe7fe id)
56 ist main(void) 57 ist main(void) 58 (*/* At this stays the microcontroller clock setting is already configures 58 (*/* At this stays the microcontroller clock setting is already configures 58 (*/* At this stays the default setting of SystemIni() function which is called from stary 50 (*/* Clatting stallOxr.s) before to branch to application main. 51 To reconfigure the default setting of SystemIni() function, refer to 52 (*/* Initialize Leds mounted on SIN32F0-discovery */ 53 STH_EVAL_LEDInit(LED3); 54 (*/* Turn on LED3 and LED4 */ 55 (* Turn on LED3 and LED4 */ 56 (* Turn on LED3 and LED4 */ 57 STH_EVAL_LEDOn(LED3); 57 STH_EVAL_LEDOn(LED4); 68 (* Turn on LED3 and LED4 */ 57 STH_EVAL_LEDOn(LED4);	t, sup	0x800024 int main(vo: { main: \$ 0x800025	dler: 4e: 0xe7fe id) 50: 0xb580
<pre></pre>	t, cup	0x800024 int main(vo: { nain:	4e: Uxe/fe id)
<pre>59 ///. At this stays the microcontroller clock setting is already configures this is done through Systemini() function which is allold from star file (startup stairform.s) before to hranch to application main. To reconfigure the default setting of SystemIni() function, refer to system_stairform.c file */ 64 */ 65 // Initialize Leds mounted on SIM32F0-discovery */ 66 STH_EVAL_EDInit(LED3); 67 STH_EVAL_EDInit(LED3); 68 */ 70 STH_EVAL_EDInit(LED3); 71 STH_EVAL_EDInit(LED3); 71 STH_EVAL_EDInit(LED3); 71 STH_EVAL_EDInit(LED3); 71 STH_EVAL_EDIn(LED3); 71 STH_EVAL_EDIN(LED3); 71 STH_EVAL_EDIN(LED3); 71 STH_EVAL_EDIN(LED3); 71 STH_EVAL_EDIN(LED3); 72 */ 72 */ 73 */ 74 */ 74 */ 75 */</pre>	t, tup	int main(vo: { main:	id)
<pre>59</pre>	, tup	{ nain: ♥ 0x800025	50: 0xb580
60 file (startup stabifOx.s) before to branch to application main. 61 To recomfigure the default setting of SystemInit() function, refer to system_stabifOx.c file 62 system_stabifOx.c file 64 */ 65 /* Initialize Leds nounted on SDM32F0-discovery */ 66 STH_FVAL_LEDInit(LED0); 67 STH_FVAL_LEDInit(LED0); 68 /* Turno n.EED and LED4 */ 70 STH_FVAL_LEDon(LED0); 71 STH_FVAL_LEDon(LED4); 72 C	>	nain:	50 · 0xb580
<pre>61 To reconfigure the default setting of SystemInit() function, refer to 62 system_sts220xx.c file 63 + */ 64 65 /* Initialize Leds mounted on SDM32F0-discovery */ 66 STH_EVAL_BEDInit(LED0); 67 STH_EVAL_BEDInit(LED0); 68 69 /* Turn on LEDS and LEDA */ 70 STH_EVAL_BEDInit(LED4); 71 STH_EVAL_BEDIn(LED4); 72 STH_EVAL_BEDIn(LED4); 72 STH_EVAL_BEDIn(LED4); 73 STH_EVAL_BEDIn(LED4); 74 STH_EVAL_BEDIn(LED4); 75 STH_EVAL_</pre>	>	♥ 0x800029	50 · 0xb580
62 system_stability.c file 63 */ 64 */ 65 /* Initialize Leds nounted on SIM32F0-discovery */ 66 STM_FVAL_LEDInit(LED0); 67 STM_FVAL_LEDInit(LED4); 68 /* Initialize Leds nounted on SIM32F0-discovery */ 69 STM_FVAL_LEDInit(LED4); 69 STM_FVAL_LEDIN(LED4); 60 STM_FVAL_LEDON(LED3); 71 STM_FVAL_LEDON(LED4); 72 STM_FVAL_LEDON(LED4);			
<pre>63 #/ " 64 65 64 65 74 Initialize Leds mounted on SIM32F0-discovery */ 66 65 STE[VAL_EEDInit(LED0); 67 57 STE[VAL_EEDInit(LED4); 68 69 70 STE[VAL_EEDon(LED3) 71 STE[VAL_EEDon(LED4); 72 71 57E[VAL_EEDon(LED4); 72 72 73 74 75 75E[VAL_EEDon(LED4); 75 75 75 75 75 75 75 75 75 75 75 75 75</pre>		STM_EVAL_I	LEDInit(LED3);
64 /* Initialize Leds mounted on SIM32F0-discovery */ 66 STM_TVAL_EDInit(LED3); 67 STM_EVAL_EDInit(LED4); 68 /* Turn on LED3 and LED4 */ 70 STM_EVAL_EDInit(LED3); 71 STM_EVAL_EDIn(LED3); 71 STM_EVAL_EDIn(LED4); 72 C		0x800025	52: 0x2000
<pre>65 // Initialize Leds nounded on SIM32F0-discovery */ 66 STM_EVAL_EDInit(LED3); 67 STM_EVAL_EDInit(LED4); 68 69 // Turn on LED3 and LED4 */ 70 STM_EVAL_ED0n(LED0); 71 STM_EVAL_ED0n(LED0); 72 STM_EVAL_ED0n(LED4); 73 STM_EVAL_ED0n(LED4); 74 STM_EVAL_ED0n(LED5); 75 STM_EVAL_ED0(LED5); 75 STM_EVAL_ED0(</pre>		0x800025	54: 0xf000 0xf
66 STH_EVAL_LEDInit(LED3); 67 STH_EVAL_LEDInit(LED4); 68 (* Turr on LED3 and LED4 */ 70 STH_EVAL_LEDOn(LED3); 71 STH_EVAL_LEDOn(LED4); 72 Control = Con		STM_EVAL_1	LEDInit(LED4);
67 STM_EVAL_LED Init(LED4); 68 69 /# Turn on LED3 and LED4 #/ 70 STM_EVAL_LED0n(LED3); 71 STM_EVAL_LED0n(LED3); 72 C (for the Complex Theorem for a lower intervention		0x800025	58: 0x2001
68 // Turn on LED3 and LED4 */ 70 STH EVAL LED0n(LED3); 71 STH EVAL_LED0n(LED4); 72 C		0x800025	5a: 0xf000 0xf
69 /* Turn on LED3 and LED4 */ 70 STH_EVAL_LED0n(LED3); 71 STH_EVAL_LED0n(LED4); 72 STH_EVAL_LED0n(LED4);		STM EVAL 1	LEDOn(LED3):
70 STM_FVAL_LEDOn(LEDO); 71 STM_FVAL_LEDOn(LEDO); 72 STM_FVAL_LEDON(LEDO); 72 CALLEDON(LEDO);		0x800025	5e: 0x2000
71 STH_EVAL_LEDOn(LED4); 72 72 72 72 72 72 72 72 72 72 72 72 72 7		0x800026	60: 0xf000 0xf
		STM EVAL 1	LEDOn (LED4)
To the second contract the first first first first second se		0x80002	64 · 0x2001
73 E 7* Secup Systick filler for I used interrupts.		0x80002	66: 0wf000 0wf
74		if (SpoTic	ck Config(Sect
75 1. The SysTick_Config() function is a CMSIS function which configure:		0x80002	6a: 0x481c
Tick Control C	> ~	<	
Massagas	File		Lino
messages Building configuration: SusTick - SusTick	File		Line
Durdening Comiguration. Systick - Systick			
Opdating build tee			
Configuration is up to which			
Conliguration is up-to-table.			
bug tog Build			

Figure 6. IAR Embedded Workbench debugger screen



To run your application, from the **Debug** menu, select **Go**. Alternatively, click the **Go** button in the toolbar to run your application.

Figure 7. Go button





6 MDK-ARM toolchain

1. Open Keil MDK-ARM Microcontroller Kit,

Figure 8 shows the basic names of the "Keil uVision4" windows referred to in this document.



Figure 8. uVision4 IDE

- 2. In the **Project** menu, select **Open Project**... Browse to select either an example or demonstration or template project file and click **Open** to launch it in the Project window.
- 3. In the Project menu, select Rebuild All target files to compile your project
- 4. If your project is successfully compiled, the following window in Figure 3 is displayed

Figure 9. MDK-ARM project successfully compiled





If you need to change your project settings (Include and preprocessor defines), you need just to go through project options:

- For Include directories'
- Project>Options for Target > C/C++ > Include Paths
- For pre-processor defines

Project>Options for Target > C/C++ > Preprocessor symbols > Define

5. In the MDK-ARM IDE, from the **Debug** menu, select **Start/Stop Debug Session** or, alternatively, click the **Start/Stop Debug Session** button the in toolbar, to program the Flash memory and begin debugging.





6. The debugger in the MDK-ARM can be used to debug source code at C and assembly levels, set breakpoints, monitor individual variables and watch events during the code execution.



Figure 11. MDK-ARM debuger screen

To run your application, from the Debug menu, select Run. Alternatively, click the Run button in the toolbar to run your application.

Figure 12. Run button



Г

7 TrueSTUDIO toolchain

Follow these steps to use the $\mathsf{TrueSTUDIO}^{\texttt{R}}$ toolchain:

1. Open Atollic TrueSTUDIO[®] for ARM product. The program launches and asks for the Workspace location.

Figure 13. TrueSTUDIO[®] workspace launcher dialog box

Workspace	ce Launcher		Sec. Sec.	×
Select a wo Atollic True Choose a v	orkspace eSTUDIO® for ARM® Pr vorkspace folder to use fi	o stores your projects in a or this session.	older called a workspace.	
Workspace:	· Discovery_FW\Project	s\Peripheral_Examples\Sy	Tick_Example\TrueSTUDIO ▼	Browse
?			ОК	Cancel

- 2. Browse to select a TrueSTUDIO[®] workspace of either an *example* or *demonstration* or *template* workspace file and click OK to load it.
- 3. To load an existing project in the selected workspace, select **Import** from the **File** menu to display the **Import** dialog box.
- 4. In the **Import** window, open **General**, select **Existing Projects** into Workspace and click **Next**.

		â		
Figure 14.	Atollic TrueSTUDI	O [®] import sou	rce select dia	log box

Select Create new projects from an archive file or directory.
Select an import source: type filter text General Archive File Existing Projects into Workspace File System Preferences Preferences <

DocID025326 Rev 1

Figure 15. Atolli	c TrueSTUDIO [®] i	mport pro	iects dialog box
-------------------	-----------------------------	-----------	------------------

Select a directory to sear	ch for existing Eclipse projects.	
 Select root directory: Select archive file: 	ects\Peripheral_Examples\Systick\TrueSTUDIO	Browse
SysTick (C:\STM	2F0308-Discovery_FW_V1.0.0\Projects\Periphera	Select All Deselect All Refresh
Copy projects into we Working sets Add project to work Working sets:	ing sets	Select
?	< Back Next > Finish	Cancel

- 6. In the Projects panel, select the project and click Finish.
- 7. In the Project Explorer, select the project, open the Project menu, and click Build Project.

8. If your project is successfully compiled, the following messages will be displayed on the **Console** window.

If you need to change the project settings (Include directories and preprocessor defines), you need just to go through Project>Properties, select C/C++ Build>Settings from the left panel:

- For Include directories
- C Compiler>Directories>Include path
 - For pre-processor defines
- C Compiler>Symbols> Defined symbols
- 9. To debug and run the application, select the project In the **Project Explorer** and press **F11** to start a debug session. (See *Figure 17*.)

Debug - SysTick/User/main	n.c - Atollic Tr	ueSTUDIO® for ARM® Pr	D	COMP + THE	100	
File Edit View Run Wi	ndow Help					
🍓 🚳 🖆 🕼	🔌 🔶 🕶	⇒ • 🚺 🗖 • 🖽 •	8 D			😰 🏇 Debug) 🔤 C/C+ »
🏇 Debug 🛿		× *	! 🕹 🕪 🗉 🔳 💺	🥆 🔿 🖈 🔤		🗱 Varia 💁 Brea 👭 Regis 🕰 Expr 🛛 🗆
▲ 10° SysTick.elf ▲ 10° Thread [1] < main> (Suspended : Step) ■ main() at main.c:99 0x80002be ■ adb						
<pre>STM32_FLASH.Id @ main.c % P4 Reload Value = SysTick Counter Clock (Hz) x Desired Time base (s) P5 Reload Value is the parameter to be passed for SysTick_Config() function P7 Reload Value is the parameter to be passed for SysTick_Config() function P7 Reload Value is the parameter to be passed for SysTick_Config() function P7 P8 P8</pre>						
📮 Console 🧟 Tasks 🔝 P	roblems 🚺	Executables 🛛		🤣 🍪 🌾 🗉		🔚 SWV Tra 📮 SWV Co 🚺 Memory 🛛 🖓 🖓
Executable Name	Project	Location	Source File Name	Location		🤐 🗋 🖬 🖤 🖤 🐼 🏭 🔛 🔻 🔨
SysTick.elf	SysTick	C:\STM32F0308-Discove				★ 0x20000000: 0x20000 S3 **1 Address 0 - 3 4 - 7 8 - ^ 20000000 000CDC02 00080048 0€ 200000010 00000000 00000000 00000000 (1) (1) (1) (1)

The debugger in the Atollic TrueSTUDIO[®] can be used to debug source code at C and assembly levels, set breakpoints, monitor individual variables and watch events during the code execution.

To run your application, from the Run menu, select **Resume**, or alternatively, click the **Resume** button in the toolbar.

DocID025326 Rev 1

8 **Revision history**

Table 1. Document revision history							
Date	Revision	Changes					
03-Oct-2013	1	Initial release.					

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER'S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR "AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL" INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries. Information in this document supersedes and replaces all information previously supplied. The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2013 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan -Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com

DocID025326 Rev 1

