

UM1467 User manual

Getting started with software and firmware environments for the STM32F4DISCOVERY Kit

1 Introduction

This document describes the software, firmware environment and development recommendations required to build an application around the STM32F4DISCOVERY board.

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 2: Getting started*.
- *Section 3* describes the firmware applications package.
- Section 5 presents development toolchain installation and overview of ST-LINK/V2 interface.
- Section 6, Section 7, Section 8, and Section 9 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
 - TASKING VX-toolset for ARM Cortex by Altium

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

- STM32F4DISCOVERY high-performance discovery board data brief
- STM32F4DISCOVERY peripherals firmware examples (AN3983)
- STM32F40x reference manual (RM0090)
- STM32F405xx STM32F407xx datasheet

The above documents are available at www.st.com/stm32f4-discovery.

Contents

1	Intro	Introduction							
2	Gett	g started							
	2.1	System requirements							
	2.2	Running the built-in demonstration							
3	Dese	iption of the firmware package							
	3.1	Libraries folder							
		3.1.1 CMSIS subfolder							
		3.1.2 STM32_USB_Device_Library	v subfolder7						
		3.1.3 STM32_USB_HOST_Library	subfolder7						
			ıbfolder						
			er subfolder						
	3.2	,							
		— ·							
		• – •	ler						
	3.3	Utilities folder							
4	Bina	y images for reprogramming fi	rmware applications						
5	ST-L	NK/V2 installation and develop	ment						
6	Usin	IAR Embedded Workbench® f	or ARM11						
	6.1	Building an existing EWARM project	.t						
	6.2	Debugging and running your EWAR	RM project 12						
	6.3	Creating your first application using	the EWARM toolchain 14						
		6.3.1 Managing source files							
		6.3.2 Configuring project options							
7	Usin	MDK-ARM Microcontroller De	velopment Kit by Keil™ 20						
	7.1	Building an existing MDK-ARM pro	ect 20						
	7.2	Debugging and running your MDK-	ARM project						
	7.3	Creating your first application using	the MDK-ARM toolchain 23						
57		Doc ID 022172 Re	/ 1 2/46						

		7.3.1	Managing source files	. 23
		7.3.2	Configuring project options	. 25
8	Using	Atollic	TrueSTUDIO®	28
	8.1	Building	an existing TrueSTUDIO project	28
	8.2	Debuggi	ng and running your TrueSTUDIO project	31
	8.3	Creating	your first application using TrueSTUDIO toolchain	32
9	Using	TASKIN	IG	36
	9.1	Building	an existing TASKING project	36
	9.2	Debuggi	ng and running your TASKING project	40
	9.3	Creating	your first application using TASKING toolchain	41
10	Revis	ion hist	ory	45



2 Getting started

2.1 System requirements

Before running your application, you should establish the connection with the STM32F4DISCOVERY board as following.



Figure 1. Hardware environment

To run and develop any firmware applications on your STM32F4DISCOVERY 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

Additional hardware accessories will be needed to run some applications:

- 'USB type A to Micro-B' cable, used to connect the board (through USB connector CN5) as USB Device to host PC.
- Headphone with male jack connector.



2.2 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 jumper position on the board, JP1 on, CN3 on (Discovery selected).
- Connect the STM32F4DISCOVERY board to a PC with a 'USB type A to Mini-B' cable through USB connector CN1 to power the board. Red LED LD2 (PWR) then lights up.
- Four LEDs between B1 and B2 are blinking.
- Press User Button B1 then MEMS sensor is enabled, move the board and observe the four LEDs blinking according to the motion direction and speed.
- If you connect a second 'USB type A to Micro-B' cable between PC and CN5 connector then the board is recognized as standard mouse and its motion will also control the PC cursor.





3 Description of the firmware package

The STM32F4DISCOVERY firmware applications are provided in one single package and supplied in one single zip file. The extraction of the zip file generates one folder, *STM32F4-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

3.1 Libraries folder

This folder contains the Hardware Abstraction Layer (HAL) for STM32F4xx Devices.

3.1.1 CMSIS subfolder

This subfolder contains the STM32F4xx and Cortex-M4F CMSIS files.

Cortex-M4F CMSIS files consist of:

- Core Peripheral Access Layer: contains name definitions, address definitions and helper functions to access Cortex-M4F core registers and peripherals. It defines also a device independent interface for RTOS Kernels that includes debug channel definitions.
- CMSIS DSP Software Library: features a suite of common signal processing functions for use on Cortex-M processor based devices. The library is completely written in C and is fully CMSIS compliant. High performance is achieved through maximum use of Cortex-M4F intrinsics.

STM32F4xx CMSIS files consist of:

- stm32f4xx.h: this file contains the definitions of all peripheral registers, bits, and memory mapping for STM32F4xx devices. The file is the unique include file used in the application programmer C source code, usually in the main.c.
- system_stm32f4xx.c/.h: This file contains the system clock configuration for STM32F4xx 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_stm32f4xx.s* file.

 startup_stm32f4xx.s: Provides the Cortex-M4F startup code and interrupt vectors for all STM32F4xx device interrupt handlers.

3.1.2 STM32_USB_Device_Library subfolder

This subfolder contains USB Device Library Core and the class drivers.

The Core folder contains the USB Device library machines as defined by the revision 2.0 Universal Serial Bus Specification.

The Class folder contains all the files relative to the Device class implementation. It is compliant with the specification of the protocol built in these classes.

3.1.3 STM32_USB_HOST_Library subfolder

This subfolder contains USB Host Library Core and the class drivers.

The Core folder contains the USB Host library machines as defined by the revision 2.0 Universal Serial Bus Specification.

The Class folder contains all the files relative to the Host class implementation. It is compliant with the specification of the protocol built in these classes.

3.1.4 STM32_USB_OTG_Driver subfolder

This subfolder contains the low level drivers for STM32F4xx USB HS and FS cores. It provides an hardware abstraction layer, USB communication operations and interfaces used by the high level Host and Device Libraries to access the core.

3.1.5 STM32F4xx_StdPeriph_Driver subfolder

This subfolder contains sources of STM32F4xx peripheral drivers (excluding USB and Ethernet).

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, *stm32f4xx_ppp.c*, and a header file, *stm32f4xx_ppp.h*. The *stm32f4xx_ppp.c* file contains all the firmware functions required to use the PPP peripheral.

3.2 Project folder

This folder contains the source files of the STM32F4DISCOVERY firmware applications.

3.2.1 Demonstration subfolder

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

Doc ID 022172 Rev 1



A binary images (*.hex and *.dfu) of this demonstration is provided under Binary subfolder. You can use the STM32F4xx's embedded Bootloader or any in-system programming tool to reprogram the demonstration using this binary image.

3.2.2 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.

3.2.3 Peripheral_Examples subfolder

This subfolder contains a set of examples for some peripherals with preconfigured projects for EWARM, MDK-ARM, TrueSTUDIO and TASKING toolchains. See *Section 5* and *STM32F4DISCOVERY peripheral firmware examples,* AN3983, for further details.

3.3 Utilities folder

This folder contains the abstraction layer for the STM32F4DISCOVERY hardware. It provides the following drivers:

- stm32f4_discovery.c: provides functions to manage the user push button and 4 LEDs (LD3.LD6)
- stm32f4_discovery_audio_codec.c/.h: provides functions to manage the audio DAC (CS43L22)
- stm32f4_discovery_lis302dl.c/.h: provides functions to manage the MEMS accelerometer (LIS302DL).



4 Binary images for reprogramming firmware applications

This section describes how to use the provided binary images to reprogram the firmware applications. The STM32F4DISCOVERY firmware package contains binary images (*.hex and *.dfu) of the provided applications which allow to use the STM32F4xx's embedded Bootloader or any in-system programming tool to reprogram these applications easily.

Below are the steps to follow:

- Using "in-system programming tool"
 - Connect the STM32F4DISCOVERY board to a PC with a 'USB type A to Mini-B' cable through USB connector CN1 to power the board.
 - 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\STM32F4-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).
- Using "Bootloader (USB FS Device in DFU mode)"
 - Configure the STM32F4DISCOVERY board to boot from "System Memory" (boot pins BOOT0:1 / BOOT1:0)
 - Set BOOT0 pin to high level: on the male header P2 place a jumper between BOOT0 pin and VDD pin
 - Set BOOT1(PB2) pin to low level: on the male header P1 place a jumper between PB2 pin and GND pin
 - Connect a 'USB type A to Mini-B' cable between PC and USB connector CN1 to power the board.
 - Connect a 'USB type A to Micro-B' cable between PC and USB connector CN5, the board will be detected as USB device.
 - Use *.dfu binary (for example, \Project\Demonstration\Binary\STM32F4-Discovery_Demonstration_V1.0.0.dfu) with "DFUse\DFUse Demonstration" tool (available for download from www.st.com) to reprogram the demonstration firmware.



5 ST-LINK/V2 installation and development

STM32F4DISCOVERY 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 6.2* 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 6.2\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.

- Altium[™] TASKING VX-toolset for ARM® Cortex-M available from www.tasking.com The toolchain is installed by default in the "C:\Program Files\TASKING 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.



6 Using IAR Embedded Workbench® for ARM

6.1 Building an existing EWARM project

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

- 1. 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 the *demonstration* 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

× Messages	File	
Errors: none Warnings: none		
Link time: 0.05 (CP (elapsed) Total number of error		
Total number of war		-
Ready		

6.2 Debugging and running your EWARM project

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



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.



The Edit View Project Debug	aamm		* 12 12 10 1	₽ # (b) D]	199 PR 👂 🗴	e eb
Contraction of the second s	and the second se	a Myr	N. 12 100 10 1	en des late i mus	40 91 20 9	17 (E)/
5 • 52653						
ortspace	× man <	*,	Cisssendly.	×	Register	×
Demo	67	* Sparan None	Goto		Current CPU	Registers 💌 💌
Files 🐉		* Gretval None	Disasser	nbly	R0 = 0x	00000000
E STM32F4-Dis V		-/		0023c2: 0	R1 = 0x	
HE CEWARM	7018	t nain(woid)	0.02270.0	0023c6: 0	$\mathbf{R2} = 0\mathbf{x}$ $\mathbf{R3} = 0\mathbf{x}$	
HE STM32_USB		RCC_ClocksTypeDef RCC_C		ck->VAL	R4 = 0x	00000000
B STM32_USB	73			0023ca: 0 0023ce: 0	R5 = 0x	
STM32F4-Discovery Demo		/* Initialize LEDs and ?	(UXO	002368. 0	K6 = UX	00000000
			1		1	
Configuration is up-to-d	ote					
Configuration is up-to-d	ate					
Configuration is up-to-d	ate					2
Configuration is up-to-di						2
C Debug Log Build						<u>k</u> x
Coto	Memory	. <u>.</u> .) x
CebugLog Build	Memory 9800572c	1 0800572f				<u>)</u> ×
Coto Colorodo Colorod	Memory 2005 8005 8005 8005733 0800573 00000000 00000000	1 0800572f 5 0000000 0 08005737				
Coto Concernation	Memory 3005 and 08005721 18005 and 08005723 18005733 08005733 1000000 0800573	1 0800572f 5 00000000 0 08005737 5 08005734				<u>></u>
Coto Contraction C	Memory 28005-ecd. 08005720 8005-ecd. 08005723 10000000 0000000 10000000 08005731 800561f9 080055ft 8005609 080055ft	1 0800572f 5 00000000 0 08005737 5 08005734 1 08006001 1 08006001				×
Go to Wild 000000000 20000c68 00000000 000000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000005739 00000000 00000000 080005739 00000000 00000000 08000515 000000000 00000000 080006015 000000601	Memory 18005723 18005scd 08005723 1800500 08005731 1000000 08005731 18005ff 08005ff	1 0800572f 5 0000000 0 08005737 5 08005734 1 08006001 4 08006011 1 08002abb				×
Cobug Log Build Go to 00000000 20000c68 0 0000000 0000000 0000000 0000000 00000	Memory 2000 18005-ccd 08005723 18005733 08005733 18005759 08005ff 18006009 08005ff 18006009 080056ff 18006019 08005616 18006029 08006026	1 0800572f 5 0000000 0 08005737 5 08005734 1 08006001 1 08006001 1 08006011 1 08006011 1 08006011 1 08006011 1 08006011				×
CobugLog Build Go to 00000000 20000068 (00000010 08005731 (00000020 0000000 (00000030 08005739 (00000030 08005739 (00000050 08006055 (00000000 08006015 (00000070 08006015 (00000090 08006015 (00000000 08006015 (00000000 08000605 (00000000 00000 (00000000000000000 (0000000000	Memory 28005-ccd 08005723 18005-ccd 08005723 18005000 08005733 18005000 08005731 18005019 080055ft 18006019 080056014 18006029 08006032 18006039 08006043 18006049 08006044	1 0800572f 5 0000000 0 08005737 5 08005734 1 08006001 1 08006001 1 08006001 1 08006031 1 08006031 1 08006051 1 08006051 1 08006051 1 08006051				(8) ×
Correction Build Go to 00000000 20000c68 00000000 00000010 08005731 00000000 00000000 000000030 08005739 00000000 00000000 000000050 080005739 00000000 00000000 000000050 080006055 000000070 080060155 00000070 080060155 000000070 080060355 000000090 080060455 000000090 080060555 000000055 000000055 000000055 000000055 000000055 000000005 000000005 00000000 000000055 000000000 000000055 000000000 0000000055 000000000 0000000055 000000000 000000000 0000000000 0000000055 000000000 000000000 0000000000 000000000	Memory 20005ecd. 08005720 8005ecd. 08005732 8005733 08005733 0000000 08005731 8005619 0800518 80066019 080058 80066019 08006012 80066019 08006012 8006019 08006012 8006019 08006012 8006019 08006012 8006019 08006012 8006019 08006012 8006019 08006012 8006019 08006012 80058000000000000000000000000000000000	1 0800572f 5 0000000 0 08005737 5 08005734 1 08006001 1 08006011 1 08006011 1 08006031 1 08006031 1 08006051 1 08006051 1 08006051				()

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



6.3 Creating your first application using the EWARM toolchain

6.3.1 Managing source files

Follow these steps to manage source files.

1. In the **Project** menu, select **Create New Project** and click **OK** to save your settings.

Figure 8. Create New Project dialog box

Tool chain:	ARM	~	/
Project templates: Empty project asm C++ C++ DLIB Externally built e	executable		
Description: Creates an empty pro	oject.	ОК	Cancel

2. Name the project (for example, *NewProject.ewp*) and click **Save** to display the IDE interface.

Figure 9.	IDE interface

🔀 IAR Embedded Workbench IDE 💦 🔲 🔀
File Edit View Project Simulator Tools Window Help
Workspace Debug Files NewProje
NewProject * Messages Configuration is up-to-date. * Build Debug Log Ready

To create a new source file, in the **File menu**, open **New** and select **File** to open an empty editor window where you can enter your source code.



The IAR Embedded Workbench enables C color syntax highlighting when you save your file using the dialog **File** > **Save As...** under a filename with the *.c extension. In *Figure 10: main.c example file*, the file is saved as **main.c**.



main.c	+ x
int main(void)	~
return(0);	
1	
	~
lfo 🔍 🔍 💷	>

Once you have created your source file you can add this file to your project, by opening the **Project** menu, selecting **Add** and adding the selected file as in *Figure 11: Adding files to a project*.



🄀 IAR EI	nbedded Workbench	IDE	
File Edit	View Project Simulato	r Tools Window	Help
🗅 🖻	🛛 🕼 🕼 🖓 🖓		
Workspace	×	main.c	• x
Debug	~	int main (v	oid) 🗖
Files	8: B	{ return(0)	
	Options		
	Make	}	
	Compile Rebuild All		
	Clean		
	Stop Build		
	Add 🔶	Add Files	
	Remove	Add "main.c"	
	Rename	Add Group	~
newpro	Source Code Control 🔸	fo	>
Add the sj	File Properties		.:i

If the file is added successfully, *Figure 12: New project file tree structure* is displayed. **Figure 12. New project file tree structure**

Files	82	2:
🗆 🗊 newproj	¥ .	
🛛 🛏 🖸 main.c		*



57

6.3.2 Configuring project options

Follow these steps to configure project options.

1. In the Project Editor, right-click on the project name and select **Options...** to display the Options dialog box as in *Figure 13*.

Figure 13. Configuring project options



2. In the Options dialog box, select the **General Options** category, open the **Target** tab and select **Device - ST -STM32F4xx.**

						ST STM32F107×E
						ST STM32F107xC
Category:						ST STM32F10xx4
100 C						ST STM32F10xx6
General Options						ST STM32F10xx8
C/C++ Compiler						ST STM32F10xx8
Assembler	Target Output	Contractor Para	For arabica Liberrari Or	tions MISRA-C.200		ST STM32F10xxC
Output Converter Custom Build	raiger Durpu	Library Con	iguration Library up	ODES MISEARC.200	-	ST STM32F10xxD
Build Actions						ST STM32F10xxE
Linker	Processor v	sriant				ST STM32F10xxF
Debugger	O Core	Contex-M4				ST STM32F10xx0
Simulator	Cole	CONTRA-24(4				ST STM32F205rx
Angel	Device	ST STM32F	4888	12+		ST STM32F205vx
GDB Server	Opevice					ST STM32F205zx
IAR ROM-monitor				None		ST STM32F207xo
J-Link/J-Trace			2002	Actel	•	ST STM32F215rx
TI Stellaris FTDI	Endian mod	•	FPU	AnalogDevices	•	ST STM32F215zx
Macraigor	(e) Little		None	Atmel	1	ST STM32F217xx
PE micro	C) Big.			Cirrus		ST-STM32F4xxx
RDI	CIBE32			EnergyMicro		ST 5TM32L151x8
ST-LINK	(=) BE8			Epson		ST STM32L151x8
Third-Party Driver				Faraday	1	ST STM32L152x8
TI XDS100				Freescale		ST STM32L152x8
				Fujitsu		ST STM32W108
				Hilscher		ST STR710
				OK Holtek	1	ST STR711
			-	incel	1	ST STR712
				Marvell		ST STR715
				Micronas	1	ST STR730
				NetSilicon	•	ST STR731
				Nuvoton	•	ST STR735
				NOP	•	ST STR736
				OKI		ST STR750
				ONSemiconducto	r 🕨	ST STR751
				Samsung	•	ST STR752
				Sode		ST STR755

Figure 14. General options > Target tab



3. Select the Linker category, open the **Config** tab, in the Linker configuration file pane select **Override default** and click **Edit** to display the Linker configuration file editor.



Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator	Config Library Input Output List #define Diagnostics
---	--

4. In the Linker configuration file editor dialog box, open the Vector Table tab and set the .intvec.start variable to 0x08000000.

Figure 16. Linker configuration file editor dialog box > Vector Table tab

Linker configuration file editor
Vector Table Memory Regions Stack/Heap Sizes
Save Cancel

5. Open the Memory Regions tab, and enter the variables as shown in *Figure 17*.

Figure 17. Linker configuration file editor dialog box > Memory Regions tab

Vector Table	Memory Regions	Stack/Heap Sizes	_
ROM	Start:	End: 0x080FFFFF	
RAM	0x20000000	0×20020000	

6. Click **Save** to save the linker settings automatically in the Project directory.



7. If your source files include header files, select the C/C++ Compiler category, open the Preprocessor tab, and specify their paths as shown in *Figure 18*. The path of the *include* directory is a relative path, and always starts with the project directory location referenced by \$PROJ_DIR\$

Figure 18. C/C++ Compiler > Preprocessor tab

Debugger Additional include directories: (one per line) Simulator \$PR0J_DIR\$\\inc Angel GD8 Server IAR ROM-monitor Image: Content of the second seco

8. To set up the ST-Link embedded debug tool interface, select the **Debugger** category, open the **Setup tab** and from the drop-down **Driver** menu, select **ST-Link** as shown in *Figure 19*.

Figure 19. Debugger > Setup tab

Build Actions Driver ✓ Run to Linker ST-Link main Debugger Setup macros Simulator Use macro file(s) GDB Server Use macro file(s)	Assembler Output Converter Custom Build	Setup Download Images Extra 0;	ptions Plugins
Debugger Simulator Angel Use macro file(s)	Build Actions		
	Simulator Angel	Setup macros	

9. Open the **Debugger** tab and select **Use flash loader(s)** as shown in *Figure 20*.

Figure 20. Select Flash loaders

Output Converter	Setup Download Images Extra Options Plugins
Custom Build Build Actions	Attach to program
Linker	Verify download
Debugger	Suppress download
Angel	Ise flash loader[s]>
GDB Server	Override default .board file



10. Select the **ST-Link** category, open the **ST-Link** tab and select **SWD** as the connection protocol as shown in *Figure 21*.



Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace LMI FTDI Macraigor RDI ST-Link Third-Party Drive	
---	--

- 11. Click **OK** to save the project settings.
- 12. To build your project, follow the instructions given in *Section 6.1: Building an existing EWARM project on page 11.*
- 13. Before running your application, establish the connection with the STM32F4DISCOVERY board as described in *Section 2: Getting started*.
- 14. To program the Flash memory and begin debugging, follow the instructions given in *Section 6.2: Debugging and running your EWARM project on page 12.*



7 Using MDK-ARM Microcontroller Development Kit by Keil[™]

7.1 Building an existing MDK-ARM project

Follow these steps to build an existing MDK-ARM project.

 Open the MDK-ARM μVision4 IDE, debugger, and simulation environment. *Figure 22: MDK-ARM μVision4 IDE environment* shows the basic names of the windows referred to in this section.



Figure 22. MDK-ARM µVision4 IDE environment

- 2. In the **Project** menu, select **Open Project...** to display the Select Project File dialog box. Browse to select the *STM32F4-Discovery.uvproj* 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 **Build Output** window (*Figure 23: Build Output - MDK-ARM μVision4 project successfully compiled*) is displayed.



compiling stm32f4x_rcc.c compiling misc.c compiling stm32f4xx_gpio.c compiling stm32f4xx_syscfg.c compiling stm32f4xx_exti.c linking Program Size: Code=1388 RO-data=460 RW-data=36 ZI-data=1028	compiling	stm32f4_discovery.c
compiling stm32f4xx_gpio.c compiling stm32f4xx_syscfg.c compiling stm32f4xx_exti.c linking	compiling	stm32f4xx_rcc.c
compiling stm32f4xx_syscfg.c compiling stm32f4xx_exti.c linking	compiling	misc.c
compiling stm32f4xx_exti.c linking	compiling	stm32f4xx_gpio.c
linking	compiling	stm32f4xx_syscfg.c
지수는 것이 가지 않는 것이 같이 하는 것이 가지 않는 것이 있는 것이 있는 것이 있는 것이 같이 있는 것이 없는 것이 없 것이 없	compiling	stm32f4xx_ext1.c
Program Size: Code=1388 RO-data=460 RW-data=36 ZI-data=1028	linking	
	Program Si	ze: Code=1388 RO-data=460 RW-data=36 ZI-data=1028

7.2 Debugging and running your MDK-ARM project

In the MDK-ARM μ Vision4 IDE, click the magnifying glass to program the Flash memory and begin debugging as shown below in *Figure 24*.

Figure 24. Starting a MDK-ARM μ Vision4 debugging session





The debugger in the MDK-ARM IDE can be used to debug source code at C and assembly levels, set breakpoints, monitor individual variables and watch events during the code execution as shown below in *Figure 25*.



Figure 25. MDK-ARM IDE workspace



7.3 Creating your first application using the MDK-ARM toolchain

7.3.1 Managing source files

Follow these steps to manage source files.

1. In the **Project** menu, select **New μVision Project...** to display the Create Project File dialog box. Name the new project and click **Save**.

Figure 26. Creating a new project

Pro	ject Fl <u>a</u> sh	Debug	Peripherals
	New µ⊻ision	Project	
	New Multi-Pr	roject <u>W</u> o	rkspace
	Open Projec	:t	
	<u>C</u> lose Projec	t	
	<u>E</u> xport		
	<u>M</u> anage		

 When a new project is saved, the IDE displays the *Device selection dialog box*. Select the device used for testing. In this example, we will use the STMicroelectronics device mounted on the STM32F4DISCOVERY board. In this case, double-click on STMicroelectronics, select the STM32F407VGT6 device and click OK to save your settings.

Figure 27. Device selection dialog box

Vendor: STMicroelectronics	
Device: STM32F407VGT6	
Toolset: ARM	
STMicroelectronics	* Core: ARM 32-bit CortexM4 CPU with Adaptive real-time (ART) accelerator allowing

3. Click **Yes** to copy the STM32 Startup Code to the project folder and add the file to the project as shown in *Figure 28*.

Figure 28. Copy the STM32 Startup Code dialog box

μVision	\mathbf{X}	
2	Copy STM32 Startup Code to Project Folder and Add File to Project ?	
	Ves No	

Note: The default STM32 startup file includes the SystemInit function. You can either comment out this file to not use it or add the system_stm32f4xx.c file from the STM32f4xx firmware library.



To create a new source file, in the **File menu**, select **New** to open an empty editor window where you can enter your source code.

The MDK-ARM toolchain enables C color syntax highlighting when you save your file using the dialog **File > Save As...** under a filename with the ***.c** extension. In this example (*Figure 29*), the file is saved as **main.c**.



i main.c
1 2
3 int main (void) 4 — (
5 return (0); 6 }
· · · ·

MDK-ARM offers several ways to add source files to a project. For example, you can select the file group in the **Project Window** > **Files** page and right-click to open a contextual menu. Select the **Add Files...** option, and browse to select the *main.c* file previously created.



Project		▲ ů ×
🖃 🚵 Target 1		
in the second s	8	Options for Group 'Source Group 1' Alt+F7
		Open File
		Open List File
		Open <u>M</u> ap File
Ĺ		Rebuild all target files
ů.		Build target F7
		Tr <u>a</u> nslate File
		Stop b <u>u</u> ild
		Add Group
1		Add Files to Group 'Source Group 1'
		Remove Group 'Source Group 1' and its Files
	÷	Manage <u>C</u> omponents
	~	Show Include File Dependencies
IL.	¥	Snow I <u>n</u> clude File Dependencies

If the file is added successfully, the following window is displayed.

Figure 31. New pro	ject file tree structure
--------------------	--------------------------

Project 🕑 🔀
Target 1 Source Group 1 Startup_stm32f4xx main.c



7.3.2 Configuring project options

- 1. In the **Project** menu, select **Options for Target 1** to display the Target Options dialog box.
- 2. Open the Target tab and enter IROM1 and IARM1 start and size settings as shown in *Figure 32*.

Figure 32. Target Options dialog box - Target tab

		Xal (MHz):	3.0	Code (Generation			
Operating a	ustem: None		-	ΓU	se Cross-Mo	dule Optimiza	tion	
	Contraction of the local			TU	se MicroLIB	> r	Big Endian	
				1.08 - 10				
	Memory Areas		-	A DECK STORE	Write Memo		~	
default off	-	st Size	Startup	default	off-chip	Start	Size	Nolnit
E BC	DM1:	1	0	E.	BAM1:			
□ R	DM2		C	F	RAM2			Г
E B	0M3		- c	Г	RAM3			-
-00	chip			1	on-chip			
1 100	068000	000 0x20000	50	(F	IRAM1	\$20000000	0x4000	Sr
		and the second s	C		IRAM2		1	-

- 3. Open the **Debug** tab, click **Use** and select the **ST-Link Debugger**. Then, click **Settings** and select the **SWD** protocol. Click **OK** to save the ST-Link setup settings.
- 4. Select Run to main().

57

	×
Linker [Debug] Utilities	Linker (Debug) Utilities
 Use: RDI Interface Driver ▼ Settings ULINK Cortex Debugger RDI Interface Driver RDI Interface Driver RDI Interface Driver RDI Interface Driver Stellaris ICDI Initializatid Signum Sustems ITAGjet S1-Link Debugger Cortex-M/R J-LINK/J-Trace ULINK Pro J-LINK/J-Trace ULINK Pro J-LINK/J-Trace ULINK Pro J-LINK/J-Trace ULINK Pro J-LINK/J-Trace 	 Use: ST-Link Debugger ✓ Load Application at Startup ✓ Tunk Setup ✓ Funk Setup ✓ Edit

Figure 33. Target Options dialog box - Debug tab

- 5. Open the **Utilities** tab, select **Use Target Driver for Flash Programming** and select the **ST-Link Debugger** from the drop-down menu.
- 6. Verify that the **Update Target before Debugging option** is selected.
- 7. Click **OK** to save your settings.

Figure 34. Target Options dialog box - Utilities tab

Use la	get Driver for Flash Programming	
Init Fil	ULINK Cortex Debugger	Settings Update Target before Debugging
	e: RDI Interface Driver Stellaris ICDI	
C Use Ext	Signum Systems JTAGjet ST-Link Debugger Cortex-M/R J-LINK/J-Trace	
Commar	d:	
Argumen	s:	
	F Run Independent	

- 8. In the Project menu, select Build Target.
- 9. If your project is successfully built, the following window is displayed.







- 10. Before running your application, establish the connection with the STM32F4DISCOVERY board as described in *Section 2: Getting started*.
- 11. To program the Flash memory and begin debugging, follow the instructions given in *Section 6.2: Debugging and running your EWARM project on page 12.*



8 Using Atollic TrueSTUDIO®

8.1 Building an existing TrueSTUDIO project

1. Open the TrueSTUDIO®/STM32 product folder and select the Atollic TrueSTUDIO® STM32 product name. The program launches and asks for the Workspace location.

Figure 36. TrueSTUDIO workspace launcher dialog box

a Workspa	ice Launcher		
	TUDIO®/STM32 Lite stores your projects in a folder called a workspace.		
Choose a w	rkspace folder to use for this session.		
<u>W</u> orkspace:	c:\NewWorkspace	~	Browse
▶ <u>C</u> opy Set	tings		
-			
?		ОК	Cancel

2. Browse to select the STM32F4DISCOVERY Demonstration TrueSTUDIO workspace and click **OK** to save your settings and to display the Welcome screen. To start using Atollic TrueSTUDIO®, click **Start using TrueSTUDIO**.

Figure 37.	Atollic TrueSTUDIO®/STM32 Lite welcome screen

File Edit Source Refactor Navigate Search			600-
а	atollic	www.atollic.com	
	Welcome to TrueSTUDIO	Introduction Product overview Product documentation Technical support Engineering services Product news	
	Lite version		
	Click he continue		

 The TrueSTUDIO Discovery workspace contains a demo project for the STM32F4DISCOVERY kit. To load this project, in the File menu, select Import... to display the Import dialog box.



4. In the **Import** window, open **General**, select **Existing Projects into Workspace** and click **Next**.

Figure 38. Atollic TrueSTUDIO®/STM32 Lite import source select dialog box

Select an import source: type filter text

5. Click **Select root directory**, browse to the TrueSTUDIO workspace folder and select the **STM32F4-Discovery** project.



Import Projects Select a directory to sear	ch for existing Eclipse projects.	
Select root directory: Select archive file: Projects:	C:\PWA_2007\Manta_Discovery_Kit\FIRMWAF	Browse
STM32F4-Discove	rry_Demo (C:\PWA_2007(Manta_Discovery_Kit)F	Select All Deselect All Refresh
Copy projects into wo Working sets		Select

Figure 39. Atollic TrueSTUDIO®/STM32 Lite import projects dialog box

- 6. In the Projects pane, select the STM32F4-Discovery and click Finish.
- 7. In the **Project Explorer**, select the **STM32F4-Discovery project**. Open the **Project** menu, and click **Build Project**.
- 8. If your project is successfully compiled, the following window is displayed.

Figure 40. TrueSTUDIO® project successfully compiled





8.2 Debugging and running your TrueSTUDIO project

In the **Project Explorer**, select the **STM32F4-Discovery project** and press **F11** to display the **Debug Configuration** dialog box.

Figure 41. TrueSTUDIO Debug Configuration dialog box

Debug Configurations		
reate, manage, and run Program does not exist	configurations	Ś
type filter text	Name: NewProject.simulator Main 04: Arguments Source Common Execution Environment Communication Setup Initialization Miscellaneous Target: ARM Simulator Memory configuration: ARM Simulator Abum General Ethinkedistromics STM32F4 Marks Discovery Kit Perpheral simulation	
Filter matched 2 of 8 items		ily Revert

9. In the **Main** tab, configure the project as shown in *Figure 41* and click **OK** to save your settings and to program the Flash memory and begin debugging.





Figure 42. TrueSTUDIO Debug window

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.

8.3 Creating your first application using TrueSTUDIO toolchain

TrueSTUDIO includes a dedicated connection to the STM32F4DISCOVERY board. When choosing this connection, all required files (startup file, firmware library, etc.) are added to the workspace and sample files are generated in the project folder to simplify development. The debug settings are automatically configured by selecting STM32F4DISCOVERY as the evaluation board.

Follow these steps to create your first application using TrueSTUDIO toolchain.

 Open the TrueSTUDIO®/STM32 product folder and select the Atollic TrueSTUDIO® STM32 product name. The program launches and asks for the Workspace location. Browse to select an existing workspace, or enter a new workspace location and click OK to confirm.



Figure 43. TrueSTUDIO workspace launcher dialog box

Workspace Launcher	
Select a workspace Atollic TrueSTUDIO®/STM32 Lite stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.	
Workspace: c:\NewWorkspace	Browse
?	OK Cancel

- 2. When the Atollic TrueSTUDIO® displays its Welcome window, click **Start using TrueSTUDIO** to open the main window. In the **File** menu, select **New** and click **C Project**.
- 3. Name the new project, in the **Project** type pane select **STM32 C Project** and click **Next**.

Figure 44. TrueSTUDIO® C Project dialog box

a C Project C Project			
Create C project of selected type			
Project name: NewProject			
Use <u>d</u> efault location			
Location: C:\NewWorkspace\New	Project	Browse	
Project type:	Toolchains:		
Executable Empty Project STM32 C Project Akefile project	Atollic ARM Tools		
Show project types and toolcha	ns only if they are supported on	the platform	
?	<u>N</u> ext > Einish	Cancel	



Note:

4. In the **TrueSTUDIO® Build Settings** dialog box, select **STM32F4-Discovery** as **Evaluation board**, configure the other settings as shown in *Figure 45* and click **Next**.

Figure 45.	TrueSTUDIO® Build Settings dialog	box
i iguio ioi	haddi obioc bana coango alalog	201

ST-LINK

?

< Back

	C Project			
	TrueSTUDIO® Build Select hardware and build	10		
	Target			
	Evaluation board:	STN32F4-Discovery	 Image: A set of the set of the	
	Microcontroller family: Microcontroller: Floating point:	Al	-	
		STN32F407VG	-	
		Software implementation	~	
	Code location:	FLASH	~	
	Instruction set	Thumb?		
	Endianess Big endian ③ Little endian			
	Optimization			
	Remove unused data (dead data removal)			
	Remove unused da	ta (dead data removal)		
ollows: Microcor	M32F4DISCOVER	Y as the evaluation board, will	l configure the project a	
ollows: Microcor	M32F4DISCOVER ntroller: STM32F robe: ST-LINK	Y as the evaluation board, will -407VGT6	configure the project a	
ollows: Microcor Debug p Connect	M32F4DISCOVER ntroller: STM32F robe: ST-LINK ion: Serial W	Y as the evaluation board, will 407VGT6 //ire Debug (SWD).		
ollows: Microcor Debug p Connect Verify tha	M32F4DISCOVER ntroller: STM32F robe: ST-LINK ion: Serial W at the JTAG Probe	Y as the evaluation board, will 407VGT6 //ire Debug (SWD). e is ST-LINK and click Finish t		
ollows: Microcor Debug p Connect . Verify tha	M32F4DISCOVER ntroller: STM32F robe: ST-LINK ion: Serial W at the JTAG Probe TrueSTUDIO® Mis	TY as the evaluation board, will 407VGT6 <i>(ire Debug (SWD).</i> is ST-LINK and click Finish t Sc Settings dialog box	o confirm your settings.	
ollows: Microcor Debug p Connect . Verify tha	M32F4DISCOVER ntroller: STM32F probe: ST-LINK ion: Serial W at the JTAG Probe FrueSTUDIO® Mis	RY as the evaluation board, will F407VGT6 (<i>ire Debug (SWD).</i> e is ST-LINK and click Finish t sc Settings dialog box		
ollows: Microcor Debug p Connect Verify tha	M32F4DISCOVER ntroller: STM32F probe: ST-LINK ion: Serial W at the JTAG Probe TrueSTUDIO® Mis	AY as the evaluation board, will F407VGT6 (<i>ire Debug (SWD).</i> e is ST-LINK and click Finish t sc Settings dialog box	o confirm your settings.	
ollows: Microcor Debug p Connect . Verify tha	M32F4DISCOVER ntroller: STM32F probe: ST-LINK ion: Serial W at the JTAG Probe TrueSTUDIO® Mis	RY as the evaluation board, will F407VGT6 (<i>ire Debug (SWD).</i> e is ST-LINK and click Finish t sc Settings dialog box	o confirm your settings.	

6. Your project is successfully created. Atollic TrueSTUDIO® generates target specific sample files (main.c, stm32f4xx_it.c...) in the Project folder to simplify development. You can tailor this project to your needs by modifying these sample files.

Next >

Atollic TrueSTUDIO Lite only supports one type of JTAG probe. Please purchase the Professional version to get support for a large number of other types of JTAG probes.

Finish

Cancel

Doc ID 022172 Rev 1



- 7. To build your project, in the **Project** menu, click **Build Project**.
- 8. Your project is successfully compiled.

Figure 47. TrueSTUDIO® project successfully built



9. Before running your application, establish the connection with the STM32F4DISCOVERY board as described in *Section 2: Getting started*. To program the Flash memory and begin debugging, follow the instructions given in *Section 8.2: Debugging and running your TrueSTUDIO project on page 31*.



9 Using TASKING

9.1 Building an existing TASKING project

Follow these steps to build an existing TASKING project.

1. Open the **TASKING VX-toolset for ARM Cortex** IDE. The program launches and asks for the Workspace location.



TASKING VX-toolset for ARM Cortex v4.0r1 stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.	
Workspace: C:\/Manta_Discovery_Kit\FIRMWARE\Project\Demonstration\TASKING	Browse
Use this as the default and do not ask again	

2. Browse to select the STM32F4DISCOVERY Demonstration TASKING workspace and click **OK** to save your settings and to display the Welcome screen. To start using TASKING, click **Go to the workbench**.



Figure 49. TASKING VX-Toolset for ARM Cortex welcome screen



- The TASKING Discovery workspace contains a demo project for the STM32F4DISCOVERY kit. To load this project, in the File menu, select Import... to display the Import dialog box.
- 4. In the **Import** window, open **General**, select **Existing Projects into Workspace** and click **Next**.

Select	
Create new projects from an archive file or directory.	
Select an import source:	
type filter text	
Cancel	

Figure 50. TASKING import source select dialog box



Figure 51.	TASKING import projects	dialog box

Select a directory to sear	ch for existing Eclipse projects.	
 Select root directory: Select archive file: Projects: 	C:\pWA_2007\Manka_Discovery_Kit\FIRMWAF	Browse
Copy projects into we	ry_Demo (C:\PWA_2007(Manta_Discovery_Kit)F	Select All Deselect All Refresh
Working sets Add project to work Working sets	ing sets	Select

- 6. In the **Projects** window, select the **STM32F4-Discovery** and click **Finish**.
- 7. In the **Project Explorer**, select the **STM32F4-Discovery** project. Open the **Project** menu, and click **Build Project**.



8. If your project is successfully compiled, the following window is displayed.

Figure 52. TASKING project successfully compiled

Import Projects Select a directory to sear	ch for existing Eclipse projects.	
Select root directory: Select archive file: Projects:	C:\PWA_2007\Manta_Discovery_Kit\FIRMWAF	Browse
STM32F4-Discove	ry_Demo (C:\PWA_2007\Manta_Discovery_Kit)F	Select All Deselect All Refresh
Copy projects into wo Working sets		
Worlung sets:	(M)	Select



9.2 Debugging and running your TASKING project

Figure 53 shows the first step for debugging and running your TASKING project. From the project toolbar menu select **Debug > Debug STM32F4-Discovery_Demo.**

Figure 53. TASKING debug window

* 34. 1		<u>▶ 回・回 朝</u>	10	TASKING Del	bug
Debug 🕅 👘 🗆	04= Variable 😫 🗣	Breakpo 📟 🗖	IIII TASKING	Registers 23	- 0
0	20 44 E	ST X N V			4
🔆 14 os 🌛 🕪 10 🔳 14 🗍	Name	Value	Group: Co	re 🗸	
入 の 北 司 お 戸 武 昌	H 🟉 RCC_Clocks	{SYSCLK			10
B 5TM32F4-Discovery_Demo [TA5I A			RO	0×0	~
B # TASKING Debugger (9/15/11			R1	0×0	10
🗟 🧬 Thread [1:1:ARM] (Susp	4	>	R2	0×40023600	
= 2 main() main.c:70 0		0	R3	0x80051e8	
t hevt rstart() rst	20	3	R4	0x0	4
R main.c 2 0 0 0 0 TASK	ING Disassembly	Outine	ine .	mm	-
In marke 23	ING Disassembly 15				375
< > Address	0x08002678				
	- 0	Memory 23			- 0
Console Tasks		_	COME BAC DO	44 11 45	
	1 FA E F T	1. T PRO 100	1	[
Debug [STM32F4-Discovery_Demo]	718 T				
Console 23 2 Tasks Debug [STM32F4-Discovery_Demo] 28 Copyright 2006-2011 Altium	EV A	10 ⊕ ≍ 🙀 _			-
Debug[STM32F4-Discovery_Demo] Copyright 2006-2011 Altium ARM debug instrument is rur	EV A				
Debug[STM32F4-Discovery_Demo]	EV A				

The debugger in TASKING 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.



The debug session is launched as follows:

1. Open TASKING VX-Toolset for ARM Cortex. The program launches and asks for the Workspace location. Browse to select an existing workspace, or enter a new workspace location and click **OK** to confirm.

```
Figure 54. TASKING workspace launcher dialog box
```

Select a w	orkspace		
	-toolset for ARM Cortex v3.2r1 stores your proje orkspace folder to use for this session.	cts in a folder called a workspace.	
Workspace:	c:NewWorkspace	Browse	∍
🔲 Use this a	is the default and do not ask again		
		OK Cano	

- When TASKING displays its Welcome window, click Go to workbench to open the main window. In the File menu, select New > TASKING VX-toolset for ARM C/C++ Project.
- 3. In the New C/C++ Project dialog box, enter the new Project name; then in the Project type box, select TASKING ARM Application and click Next.

Figure 55. TASKING New C/C++ Project dialog box

C/C++ Proje Create a new C		ASKING VX-toolset for A	RM	
Project name: Use defaul Location: C:/	location	icarm v3.2r1/eclipse/Nev	wWorkspace/Nei	Browse
● H ● H ● H	NG ARM Application hpty Project slo World C Project allo World C++ Project NG ARM Library NG ARM MIL Library			
?	<	Back Next >	Finish	Cancel



The list of the supported devices is shown, select STMicroelectronics > STM32F407VG > STM32F407VGT6 as shown below in *Figure 56*.

Figure 56. Processor selection

	Expand Al
(Timercelectronics)	
G = 0 5TM22F100	Expand Selected
III 511432F101	
II 517432F102	Collapse Al
🗟 🛄 5TM32F103	
🖼 🛄 5TM32F105	
🖼 🛄 STM32F107	
🗄 🔲 STM32L151	
🖼 🔲 STM32L152	
😥 🔄 571432F205	
😥 📃 51M32F207	
🕑 📃 STM32F215	
🔟 🔲 5TM32F217	
CH V 5TM32F402>	
5 TM32F 407JE	
5TM32F4071G	
5TM32E407VE	
5TH32F407VG	
5 114327 40721	
5T/432F4072/5	

To configure the project for Manta DISCOVERY board, select **Debug > Debug configurations** and choose **STMicroelectronics STM32F4 Manta Discovery Kit**. Choosing **STMicroelectronics STM32F4 Manta Discovery Kit** as the evaluation board, will add automatically the needed linker file and will configure the project as follows:

- Microcontroller: STM32F407VGT6
- Debug probe: ST-LINK
- Connection: Serial Wire Debugging (SWD).

Figure 57. Debug configuration

Project	Debug	Window Help
• 🞯 •	(7)	Debug STM32F4-Discovery_Demo



- 4. To add source file to your project right click on the project from the C/C++ project window and select **Import**.
- 5. The **Import** dialog box is displayed, select **General** and the desired file as shown in *Figure 58: TASKING Import dialog box*

Figure 58. TASKING Import dialog box

	type filter text	Felect	
type filter text General Gen	type filter text		Ľ
General Archive File Existing Projects into Workspace File System Preferences CV5 CV5 CV5 CV5 CV5 CV5 CV5 CV5 CV5	General G	Select an import source:	
Archive File Existing Projects into Workspace File System Preferences C/C++ CV5 CV5 CV5 CV5 CV5 CV5	Archive File Existing Projects Into Workspace File System Preferences C/C++ CV5 CV5 CV5 Ann/Debug Ann/Debug	type filter text	

6. Click **Next**.Fill the displayed window as following and then browse to your source file.



ile system Import resources from the local file syst	tem.		
From directory: C:WewWorkspace			Browse
NewWorkspace	Ø	R main.c	
	Deselect Al		
Filter Types Select Al [
Filter Types Select All	ASSIGLT HE		Browse
Into folder: NewProject Options Overwrite existing resources witho Ocreate complete folder structure			Browse
Into folder: NewProject Options Overwrite existing resources witho	ut warning		Browse

Figure 59. Adding a new source file window.

- 7. Select **main.c** file and click **Finish**.
- 8. To build your project click on **Project** > **Build Project** from the toolbar menu.
- 9. Your Project is successfully compiled.

Figure 60. Tasking project successfully built

items		
Description	Resource	Path
C/C++ build completed successfully	NewProject	

10. Before running your application, establish the connection with the STM32F4DISCOVERY board as described in *Section 2: Getting started*.



10 Revision history

Table 1.Document revision history

Date	Revision	Changes
23-Sep-2011	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.

UNLESS EXPRESSLY APPROVED IN WRITING BY TWO AUTHORIZED ST REPRESENTATIVES, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.

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.

© 2011 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



Doc ID 022172 Rev 1