Introduction

This document describes the software, firmware environment and development recommendations required to build an application around the STM32F3DISCOVERY 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 1: Getting started.
- Section 2 describes the firmware application 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™
  - TASKING VX-toolset for ARM Cortex by Altium
  - 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

- STM32F3DISCOVERY high-performance discovery board data brief
- STM32F3DISCOVERY peripheral firmware examples (AN4062)
- STM32F30x reference manual (RM0313)
- STM32F30xx datasheet

The above documents are available at www.st.com/stm32f3-discovery.

Table 1. Applicable tools

<table>
<thead>
<tr>
<th>Type</th>
<th>Part number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation tools</td>
<td>STM32F3DISCOVERY</td>
</tr>
</tbody>
</table>
Contents

1 Getting started ......................................................... 7
  1.1 System requirements .............................................. 7
  1.2 Running the built-in demonstration .......................... 8

2 Description of the firmware package ............................ 9
  2.1 Libraries folder .................................................. 9
    2.1.1 CMSIS subfolder ............................................ 9
    2.1.2 STM32F30x_StdPeriph_Driver subfolder .................. 10
    2.1.3 STM32_USB-FS-Device_Driver .............................. 10
  2.2 Project folder .................................................. 10
    2.2.1 Demonstration subfolder .................................. 10
    2.2.2 Master_Workspace subfolder ............................... 10
    2.2.3 Peripheral_Examples subfolder ............................ 10
  2.3 Utilities folder ................................................ 11

3 Binary images for reprogramming firmware applications ....... 12

4 ST-LINK/V2 installation and development ....................... 13

5 Using IAR Embedded Workbench® for ARM ....................... 14
  5.1 Building an existing EWARM project ......................... 14
  5.2 Debugging and running your EWARM project ................. 15
  5.3 Creating your first application using the EWARM toolchain .... 17
    5.3.1 Managing source files .................................... 17
    5.3.2 Configuring project options ............................. 19

6 Using MDK-ARM Microcontroller Development Kit by Keil™ ..... 23
  6.1 Building an existing MDK-ARM project ....................... 23
  6.2 Debugging and running your MDK-ARM project ............... 24
  6.3 Creating your first application using the MDK-ARM toolchain ... 26
    6.3.1 Managing source files .................................... 26
    6.3.2 Configuring project options ............................. 28

7 Using TASKING .......................................................... 31
## Contents

7.1 Building an existing TASKING project .............................................. 31
7.2 Debugging and running your TASKING project ............................... 34
7.3 Creating your first application using TASKING toolchain ..................... 35

### 8 Using Atollic TrueSTUDIO® ............................................................. 39
8.1 Building an existing TrueSTUDIO project ...................................... 39
8.2 Debugging and running your TrueSTUDIO project ............................ 42
8.3 Creating your first application using TrueSTUDIO toolchain ................. 43

### 9 Revision history ................................................................. 47
List of tables

Table 1. Applicable tools. ................................................................. 1
Table 2. Document revision history ................................................. 47
List of figures

Figure 1. Hardware environment ......................................................... 7
Figure 2. Firmware applications subfolders ........................................ 9
Figure 3. IAR Embedded Workbench IDE (Integrated Design Environment) ........ 14
Figure 4. EWARM project successfully compiled ..................................... 15
Figure 5. Download and Debug button ................................................ 15
Figure 6. IAR Embedded Workbench debugger screen ............................. 16
Figure 7. Go button ........................................................................... 16
Figure 8. Create New Project dialog box .............................................. 17
Figure 9. IDE interface ..................................................................... 17
Figure 10. IAR main.c example file ..................................................... 18
Figure 11. Adding files to a project ....................................................... 18
Figure 12. New project file tree structure ............................................. 18
Figure 13. Configuring project options ................................................ 19
Figure 14. General options > Target tab ............................................... 19
Figure 15. Linker > Config tab .............................................................. 20
Figure 16. Linker configuration file editor dialog box > Vector Table tab .... 20
Figure 17. Linker configuration file editor dialog box > Memory Regions tab 20
Figure 18. C/C++ Compiler > Preprocessor tab ..................................... 21
Figure 19. Debugger > Setup tab ......................................................... 21
Figure 20. Select Flash loaders ............................................................ 21
Figure 21. ST-Link communication protocol ........................................ 22
Figure 22. MDK-ARM µVision4 IDE environment ................................... 23
Figure 23. Build Output - MDK-ARM µVision4 project successfully compiled 24
Figure 24. Starting an MDK-ARM µVision4 debugging session ............... 24
Figure 25. MDK-ARM IDE workspace ................................................ 25
Figure 26. Creating a new project ....................................................... 26
Figure 27. Device selection dialog box ................................................ 26
Figure 28. Copy the STM32 Startup Code dialog box .............................. 26
Figure 29. MDK-ARM main.c example file .......................................... 27
Figure 30. Adding source files ........................................................... 27
Figure 31. New project file tree structure .............................................. 27
Figure 32. Target Options dialog box - Target tab ................................. 28
Figure 33. Target Options dialog box - Debug tab ................................ 29
Figure 34. Target Options dialog box - Utilities tab ............................... 29
Figure 35. MDK-ARM µVision4 project successfully built ..................... 30
Figure 36. TASKING workspace launcher dialog box ............................ 31
Figure 37. TASKING VX-Toolset for ARM Cortex welcome screen .......... 31
Figure 38. TASKING import source select dialog box ............................ 32
Figure 39. TASKING import projects dialog box .................................... 33
Figure 40. TASKING project successfully compiled .................................. 33
Figure 41. TASKING debug window .................................................... 34
Figure 42. TASKING Workspace Launcher dialog box ............................ 35
Figure 43. TASKING New C/C++ Project dialog box ............................. 35
Figure 44. Processor selection ............................................................ 36
Figure 45. Debug configuration .......................................................... 36
Figure 46. TASKING Import dialog box ............................................... 37
Figure 47. Adding a new source file window ....................................... 38
Figure 48. Tasking project successfully built ........................................ 38
List of figures

Figure 49. TrueSTUDIO workspace launcher dialog box .......................................................... 39
Figure 50. Atollic TrueSTUDIO®/STM32 Lite welcome screen .............................................. 39
Figure 51. Atollic TrueSTUDIO®/STM32 Lite import source select dialog box .......................... 40
Figure 52. Atollic TrueSTUDIO®/STM32 Lite import projects dialog box ................................. 41
Figure 53. TrueSTUDIO® project successfully compiled. .......................................................... 41
Figure 54. TrueSTUDIO Debug Configuration dialog box ......................................................... 42
Figure 55. TrueSTUDIO Debug window ................................................................................. 43
Figure 56. TrueSTUDIO workspace launcher dialog box .......................................................... 44
Figure 57. TrueSTUDIO® C Project dialog box ..................................................................... 44
Figure 58. TrueSTUDIO® Build Settings dialog box ............................................................... 45
Figure 59. TrueSTUDIO® Misc Settings dialog box ................................................................. 46
Figure 60. TrueSTUDIO® project successfully built ................................................................. 46
1 Getting started

1.1 System requirements

Before running your application, you should establish the connection with the STM32F3DISCOVERY board as follows.

To run and develop any firmware applications on your STM32F3DISCOVERY 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.
- ‘USB type A to Mini-B’ cable, used to power the board (through USB connector CN2) from host PC and connect USB to host PC.
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 jumper position on the board, JP4 on, CN4 on (Discovery selected).
- Connect the STM32F3-DISCOVERY board to a PC with a 'USB type A to Mini-B' cable through USB connector CN2 or CN1 to power the board. Then red LED LD1 (PWR) and LD2 (COM) light up.
- All 8 LEDs between B1 and B2 are blinking in a chase sequence.
- Press User Button B1 (Bottom left corner of the board) then gyroscope MEMS sensor is enabled, move the board and observe the four LEDs blinking according to the motion direction.
- Press User Button B1 (Bottom left corner of the board) then Compass MEMS sensor is enabled, move the board horizontally and observe the north direction. If you take the board and lean it then all LEDs are blinking.
2 Description of the firmware package

The STM32F3DISCOVERY firmware applications are provided in one single package and supplied in one single zip file. The extraction of the zip file generates one folder, \textit{STM32F3-Discovery_FW_VX.Y.Z}, which contains the following subfolders:

\begin{itemize}
  \item \textbf{Figure 2. Firmware applications subfolders}
  \begin{itemize}
    \item \textit{STM32F3-Discovery_FW_VX.Y.Z}\_hбресц
    \item \textbf{Libraries}
    \begin{itemize}
      \item \textit{CMSIS5}
      \item \textit{STM32\_USB\_F5\_Device\_Driver}
      \item \textit{STM32F30x\_SxPeripheral\_Driver}
      \begin{itemize}
        \item \textit{inc}
        \item \textit{src}
      \end{itemize}
    \end{itemize}
    \item \textbf{Project}
    \begin{itemize}
      \item \textit{Demonstration}
      \item \textit{Master\_Workspace}
      \item \textit{Peripheral\_Examples}
    \end{itemize}
    \item \textbf{Utilities}
    \begin{itemize}
      \item \textit{STM32F3\_Discovery}
    \end{itemize}
  \end{itemize}
\end{itemize}

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 STM32F30x devices.

2.1.1 CMSIS subfolder

This subfolder contains the STM32F30x and Cortex-M4 CMSIS files.

\begin{itemize}
  \item \textbf{Cortex-M4 CMSIS files consist of:}
    \begin{itemize}
      \item 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.
      \item 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.
    \end{itemize}
  \item \textbf{STM32F30x CMSIS files consist of:}
    \begin{itemize}
      \item \textit{stm32f30x.h}: contains the definitions of all peripheral registers, bits, and memory mapping for STM32F30x devices. The file is the unique include file used in the application programmer C source code, usually in main.c.
      \item \textit{system_stm32f30x.c/h}: contains the system clock configuration for STM32F30x devices. It exports \texttt{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_stm32f30x.s` file.

- `startup_stm32f30x.s`: provides the Cortex-M4 startup code and interrupt vectors for all STM32F30x device interrupt handlers.

### 2.1.2 STM32F30x StdPeriph_Driver subfolder

This subfolder contains sources of STM32F30x 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, `stm32f30x_ppp.c`, and a header file, `stm32f30x_ppp.h`. The `stm32f30x_ppp.c` file contains all the firmware functions required to use the PPP peripheral.

### 2.1.3 STM32_USB-FS-Device_Driver

This subfolder contains USB Full Speed Library Core and the class drivers. The Core folder contains the USB 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.

### 2.2 Project folder

This folder contains the source files of the STM32F3DISCOVERY firmware applications.

#### 2.2.1 Demonstration subfolder

This subfolder contains the demonstration source files with preconfigured project for EWARM, MDK-ARM, TASKING 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, TASKING and TrueSTUDIO toolchains. See Section 4 and STM32F3DISCOVERY peripheral firmware examples, AN4062, for further details.
2.3 Utilities folder

This folder contains the abstraction layer for the STM32F3DISCOVERY hardware. It provides the following drivers:

- `stm32f3_discovery.c`: provides functions to manage the user push-button and 8 LEDs (from LD3 to LD10).
- `stm32f3_discovery_lsm303dlhc.c/.h`: provides functions to manage the MEMS (LSM303DLHC).
- `stm32f3_discovery_l3gd20.c/.h`: provides functions to manage the MEMS (L3GD20).
3 Binary images for reprogramming firmware applications

This section describes how to use the provided binary images to reprogram the firmware applications. The STM32F3DISCOVERY 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 STM32F3DISCOVERY 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 CN4 jumpers ON).
3. Use *.hex binary (for example, \Project\Demonstration\Binary\STM32F3-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

STM32F3DISCOVERY 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 6.30 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 6.30\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 µVision4 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.
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.

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.

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

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

**Figure 6.** IAR Embedded Workbench debugger screen

**Figure 7.** Go button
5.3 Creating your first application using the EWARM toolchain

5.3.1 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

   ![Create New Project dialog box]

   Follow these steps to manage source files.

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

   Figure 9. IDE interface

   ![IDE interface]

   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: IAR main.c example file, the file is saved as main.c.

Figure 10. IAR main.c example file

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.

Figure 11. Adding files to a project

If the file is added successfully, Figure 12 is displayed.

Figure 12. New project file tree structure
5.3.2 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 - STM32F303.

Figure 14. General options > Target tab
3. Select the **Linker** category and open the **Config** tab; in the **Linker configuration file pane**, select **Override default** and click **Edit** to display the **Linker configuration file editor**.

**Figure 15. Linker > Config tab**

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**

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**

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

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

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

Figure 20. Select Flash loaders
10. Select the **ST-Link** category, open the **ST-Link** tab and select **SWD** as the connection protocol as shown in **Figure 21**.

**Figure 21. ST-Link communication protocol**

11. Click **OK** to save the project settings.

12. To build your project, follow the instructions given in **Section 5.1: Building an existing EWARM project**.

13. Before running your application, establish the connection with the STM32F3DISCOVERY board as described in **Section 1: Getting started**.

14. To program the Flash memory and begin debugging, follow the instructions given in **Section 5.2: Debugging and running your EWARM project**.
6 Using MDK-ARM Microcontroller Development Kit by Keil™

6.1 Building an existing MDK-ARM project

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

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

   **Figure 22.** MDK-ARM µVision4 IDE environment

   ![MDK-ARM µVision4 IDE environment](image)

2. In the *Project* menu, select *Open Project...* to display the Select Project File dialog box. Browse to select the *STM32F3-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) is displayed.

**Figure 23.** Build Output - MDK-ARM µVision4 project successfully compiled

6.2 Debugging and running your MDK-ARM project

In the MDK-ARM µVision4 IDE, click the magnifying glass to program the Flash memory and begin debugging as shown below in Figure 24.

**Figure 24.** Starting an MDK-ARM µVision4 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
6.3 Creating your first application using the MDK-ARM toolchain

6.3.1 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

2. When a new project is saved, the IDE displays the Device selection dialog box. Select the device used for testing. In this example, we use the STMicroelectronics device mounted on the STM32F3DISCOVERY board: double-click on STMicroelectronics, select the STM32F303VCT6 device and click OK to save your settings.

   Figure 27. Device selection dialog box

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

Note: The default STM32 startup file includes the SystemInit function. You can either comment out this file not to use it, or add the system_stm32f30x.c file from the STM32f30x 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 File > Save As… dialog under a filename with the *.
extension. In this example (Figure 29), the file is saved as main.c.

**Figure 29. MDK-ARM main.c example file**

```c
int main (void)
{
    return(0);
}
```

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.

**Figure 30. Adding source files**

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

**Figure 31. New project file tree structure**

```c
main.c
```
6.3.2 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

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()**.
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.

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 STM32F3DISCOVERY board as described in Section 1: Getting started.

11. To program the Flash memory and begin debugging, follow the instructions given in Section 5.2: Debugging and running your EWARM project.
7 Using TASKING

7.1 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.

Figure 36. TASKING workspace launcher dialog box

2. Browse to select the STM32F3DISCOVERY 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 37. TASKING VX-Toolset for ARM Cortex welcome screen
3. The TASKING Discovery workspace contains a demo project for the STM32F3DISCOVERY kit. To load this project, select **Import...** in the File menu to display the **Import** dialog box.

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

**Figure 38. TASKING import source select dialog box**

![TASKING import source select dialog box](image-url)
5. Click **Select root directory**, browse to the TASKING workspace folder and select the **STM32F3-Discovery** project.

**Figure 39. TASKING import projects dialog box**

![TASKING import projects dialog box](image1)

6. In the **Projects** window, select the **STM32F3_Discovery_Kit** and click **Finish**.

7. In the **Project Explorer**, select the **STM32F3-Discovery** project. Open the **Project** menu, and click **Build Project**.

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

**Figure 40. TASKING project successfully compiled**

![TASKING project successfully compiled](image2)
7.2 Debugging and running your TASKING project

Figure 41 shows the first step for debugging and running your TASKING project. From the project toolbar menu, select Debug > Debug STM32F3-Discovery_Demo.

Figure 41. TASKING debug window

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.
7.3 Creating your first application using TASKING toolchain

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 42. TASKING Workspace Launcher dialog box

2. 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 43. TASKING New C/C++ Project dialog box
4. From the list of supported devices, select STMicroelectronics > STM32F303> STM32F303VCT6 as shown below in *Figure 44*.

**Figure 44. Processor selection**

![Processor selection](image)

5. To configure the project for STM32F3 DISCOVERY board, select **Debug > Debug configurations** and choose **STMicroelectronics STM32F3 Discovery Kit**. Choosing **STMicroelectronics STM32F3 Discovery Kit** as the evaluation board, will add automatically the needed linker file and will configure the project as follows:
   - Microcontroller: STM32F303VCT6
   - Debug probe: ST-LINK
   - Connection: Serial Wire Debugging (SWD)

**Figure 45. Debug configuration**

![Debug configuration](image)
6. To add a source file to your project, right-click on the project from the C/C++ project window and select **Import**.
7. From the **Import** dialog box, select **General** and the desired file as shown in **Figure 46**.

**Figure 46.** TASKING Import dialog box
8. Click Next. Fill the displayed window as follows and then browse to your source file.

**Figure 47. Adding a new source file window**

9. Select main.c file and click Finish.

10. To build your project, click on Project > Build Project from the toolbar menu.

11. Your project is compiled successfully.

**Figure 48. Tasking project successfully built**

12. Before running your application, establish the connection with the STM32F3DISCOVERY board as described in Section 1: Getting started.
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.

   ![Workspace launcher dialog box](image)

   (Figure 49. TrueSTUDIO workspace launcher dialog box)

2. Browse to select the STM32F3DISCOVERY 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**.

   ![Welcome screen](image)

   (Figure 50. Atollic TrueSTUDIO®/STM32 Lite welcome screen)
3. The TrueSTUDIO Discovery workspace contains a demo project for the STM32F3DISCOVERY kit. To load this project, select **Import**... in the File menu to display the **Import** dialog box.

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

**Figure 51. Atollic TrueSTUDIO®/STM32 Lite import source select dialog box**
5. Click **Select root directory**, browse to the TrueSTUDIO workspace folder and select the **STM32F3-Discovery** project.

**Figure 52. Atollic TrueSTUDIO®/STM32 Lite import projects dialog box**

6. In the **Projects** pane, select the **STM32F3_Discovery_Kit** and click **Finish**.

7. In the **Project Explorer**, select the **STM32F3-Discovery project**. Open the **Project** menu, and click **Build Project**.

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

**Figure 53. TrueSTUDIO® project successfully compiled**
8.2 Debugging and running your TrueSTUDIO project

In the Project Explorer, select the STM32F3-Discovery project and press F11 to display the Debug Configuration dialog box.

Figure 54. TrueSTUDIO Debug Configuration dialog box
9. In the **Main** tab, configure the project as shown in Figure 54 and click **OK** to save your settings and to program the Flash memory and begin debugging.

**Figure 55. 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 STM32F3DISCOVERY 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 the development. The debug settings are automatically configured by selecting STM32F3DISCOVERY as the evaluation board.

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

1. 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.
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, select **STM32 C Project** in the **Project type** pane, then click **Next**.

4. In the **TrueSTUDIO® Build Settings** dialog box, select **STM32F3_Discovery** as the **Evaluation board**, configure the other settings as shown in **Figure 58** and click **Next**.

**Figure 58. TrueSTUDIO® Build Settings dialog box**

![TrueSTUDIO® Build Settings dialog box](image)

**Note:** Choosing **STM32F3DISCOVERY** as the evaluation board will configure the project as follows:

**Microcontroller:** STM32F303VCT6  
**Debug probe:** ST-LINK  
**Connection:** Serial Wire Debug (SWD)
5. Verify that the **JTAG Probe** is **ST-LINK** and click **Finish** to confirm your settings.

![Figure 59. TrueSTUDIO® Misc Settings dialog box](image)

6. Your project has been created successfully. Atollic TrueSTUDIO® generates target specific sample files (main.c, stm32f30x_it.c...) in the Project folder to simplify the development. You can tailor this project to your needs by modifying these sample files.

7. To build your project, click **Build Project** in the **Project** menu.

8. Your project is compiled successfully.

![Figure 60. TrueSTUDIO® project successfully built](image)

9. Before running your application, establish the connection with the STM32F3DISCOVERY board as described in **Section 1: Getting started**. To program the Flash memory and begin debugging, follow the instructions given in **Section 8.2: Debugging and running your TrueSTUDIO project**.
## 9 Revision history

Table 2. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>04-Sep-2012</td>
<td>1</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>