UM1451
User manual

Getting started with software development toolchains for the STM32L-DISCOVERY board

Introduction

This document provides an introduction on how to use the following software development toolchains with the **STM32L-DISCOVERY board**.

- IAR Embedded Workbench® for ARM (EWARM) by IAR Systems
- Microcontroller Development Kit for ARM (MDK-ARM) by Keil™
- TrueSTUDIO® by Atollic

It provides guidelines to novice users on how to build and run a sample program provided with this document and allows them to create and build their own application.

When running the sample program supplied with this application note, the Red LED LD2 (PWR) lights up. Then the user will be able to run a series of functions (VDD voltage measurement, STM32L current consumption...) by pressing the user button B1 to switch from a function to an other (please refer to AN3413).

Although this application note 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

- STM32L-DISCOVERY evaluation board user manual (UM1079)
- STM32L-DISCOVERY: current consumption measurement and touch sensing demonstration (AN3413)

The above documents are available at www.st.com/stm32l.

Contents

1 Overview of STLINK V2 interface

The STM32L-DISCOVERY board includes an ST-LINK/V2 embedded debug tool interface that is supported by the following software toolchain versions:

- **EWARM Version 6.20.4 and later available from www.iar.com**

  Installing the **EWARM** toolchain (using the default settings) results in the toolchain being installed in the *C:\Program Files\IAR Systems\Embedded Workbench 6.2* directory on the PC's local hard disk.

  After installing EWARM 6.20.4, 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*
- **MDK-ARM Version 4.21 and later available from www.keil.com**

  Installing the **MDK-ARM** toolchain (using the default settings) results in the toolchain being installed in the *C:\Keil* directory on the PC's local hard disk; the installer creates a start menu uVision4 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.
- **TrueSTUDIO Version 2.1.0 and later available from www.atollic.com**

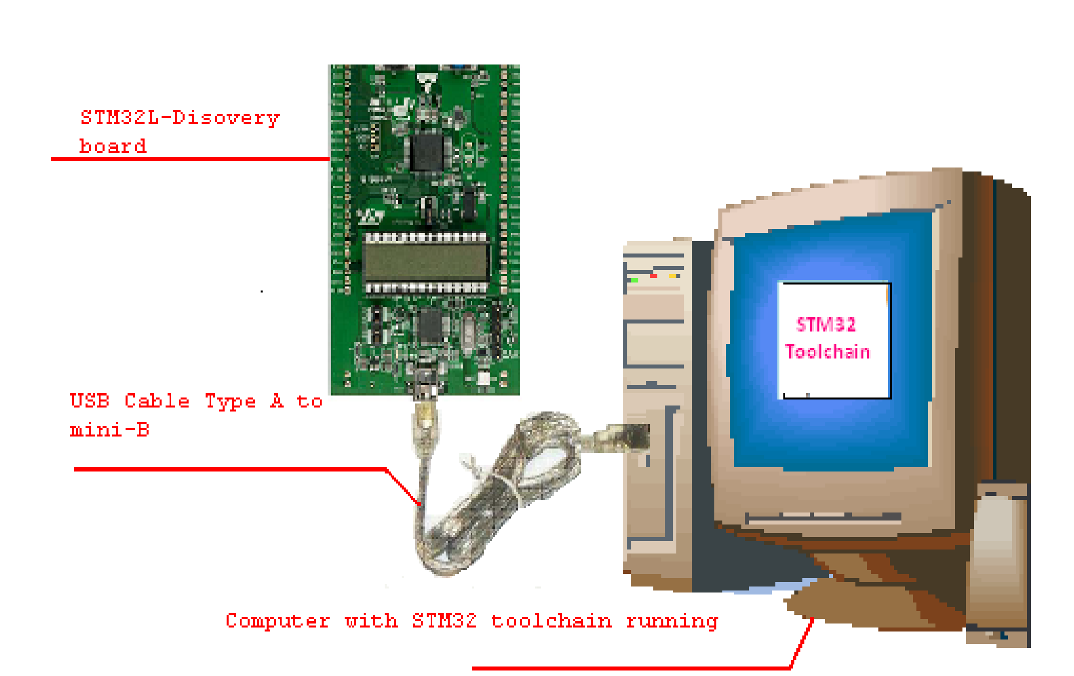
  Installing the **TrueSTUDIO** toolchain (using the default settings) results in the toolchain being installed in the "*C:\Program Files\Atollic* directory on the PC's local hard disk.

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

2 Hardware environment setup

Before running your application, you should establish the connection with the STM32L-DISCOVERY board as following.

Figure 1. Hardware environment


STM32L-Disovery board
USB Cable Type A to mini-B
STM32 Toolchain
Computer with STM32 toolchain running

For more details on how to establish your hardware environment you can refer to the **UM1079 User Manual:** STM32L-DISCOVERY available at www.st.com/internet/evalboard/product/250990.jsp

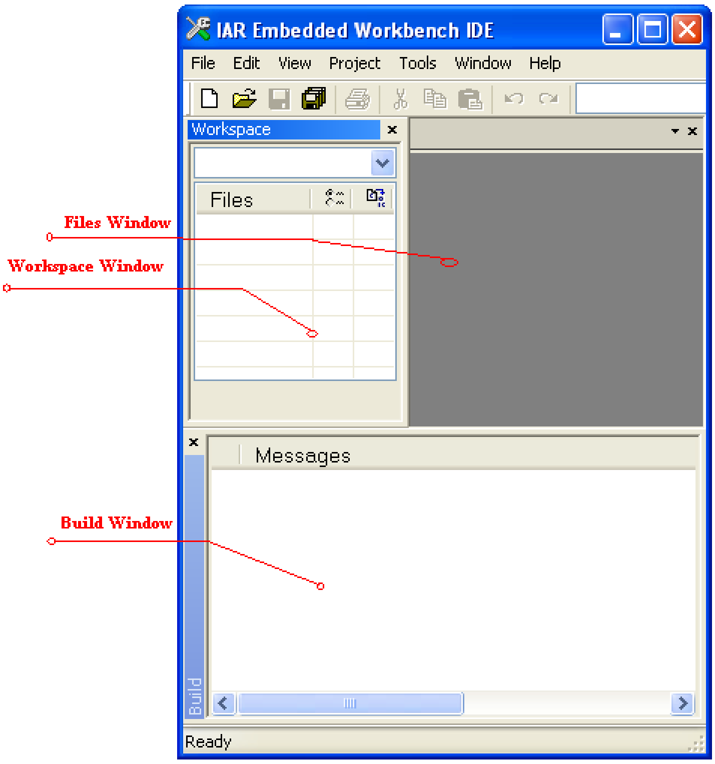
3 Using the IAR Embedded Workbench® for ARM

3.1 Building an existing EWARM project

1. Open the IAR Embedded Workbench® for ARM (EWARM).

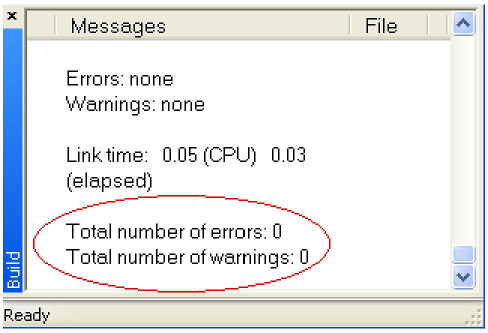
   Figure 2 shows the basic names of the windows referred to in this document.

Figure 2. IAR Embedded Workbench IDE environment



2. In the **File** menu, select **Open** and click **Workspace** to display the Open Workspace dialog box. Browse to select the *STM32L-Discovery.eww* 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 is displayed.

Figure 3. EWARM project successfully compiled



Errors: none
Warnings: none

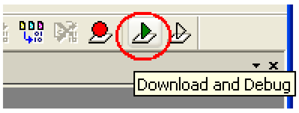
Link time: 0.05 (CPU) 0.03
(elapsed)

Total number of errors: 0
Total number of warnings: 0

3.2 Debugging and running your EWARM project

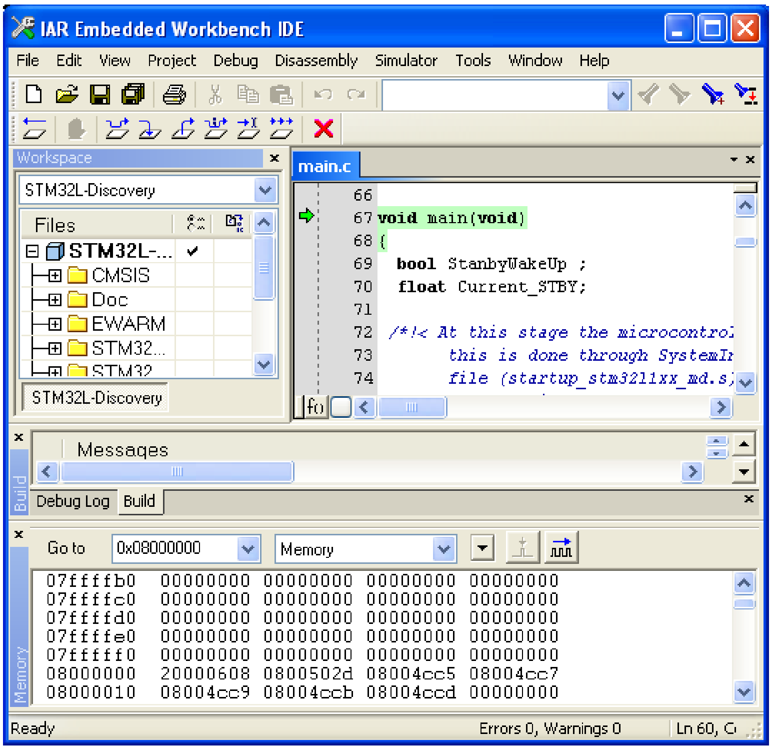
In the IAR Embedded Workbench IDE, from the **Project menu**, select **Download and Debug** or, alternatively, click the **Download and debug** button the in toolbar, to program the Flash memory and begin debugging.

Figure 4. 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.

Figure 5. IAR Embedded Workbench debugger screen

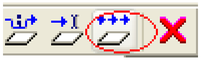


```
void main(void)
{
  bool StanbyWakeUp ;
  float Current_STBY;

/*!< At this stage the microcontrol
     this is done through SystemIr
     file (startup_stm32l1xx_md.s)
```

```
07ffffb0  00000000 00000000 00000000 00000000
07ffffc0  00000000 00000000 00000000 00000000
07ffffd0  00000000 00000000 00000000 00000000
07ffffe0  00000000 00000000 00000000 00000000
07fffff0  00000000 00000000 00000000 00000000
08000000  20000608 0800502d 08004cc5 08004cc7
08000010  08004cc9 08004ccb 08004ccd 00000000
```

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

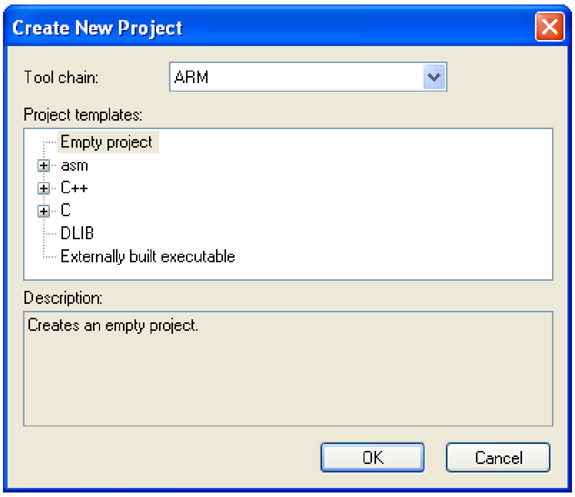


3.3 Creating your first application using the EWARM toolchain

3.3.1 Managing source files

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

Figure 6. Create new project dialog box

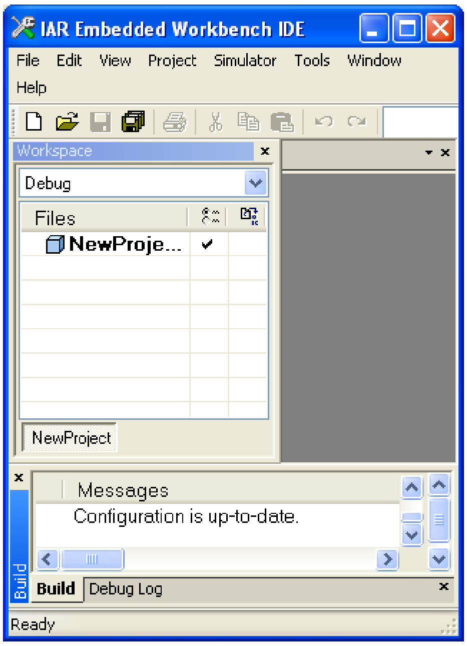


Tool chain: ARM

Description: Creates an empty project.

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

Figure 7. IDE interface



Workspace: Debug

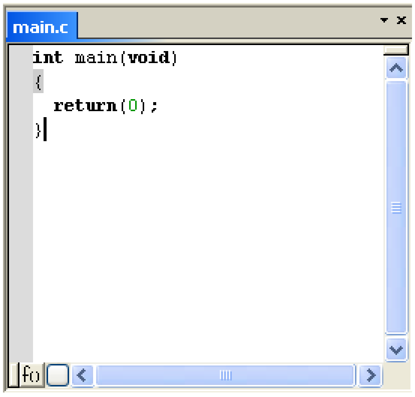
Configuration is up-to-date.

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 this example, the file is saved as **main.c**.

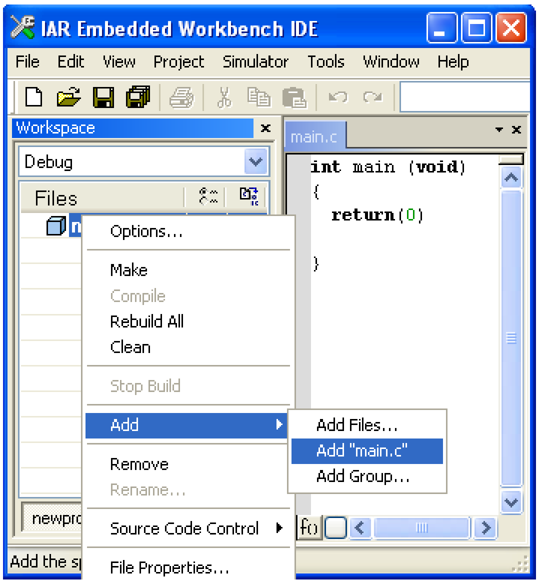
Figure 8. main.c example file



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

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.

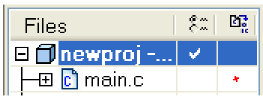
Figure 9. Adding files to a project



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

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

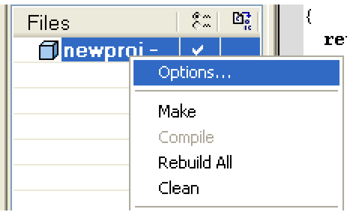
Figure 10. New project file tree structure



3.3.2 Configuring project options

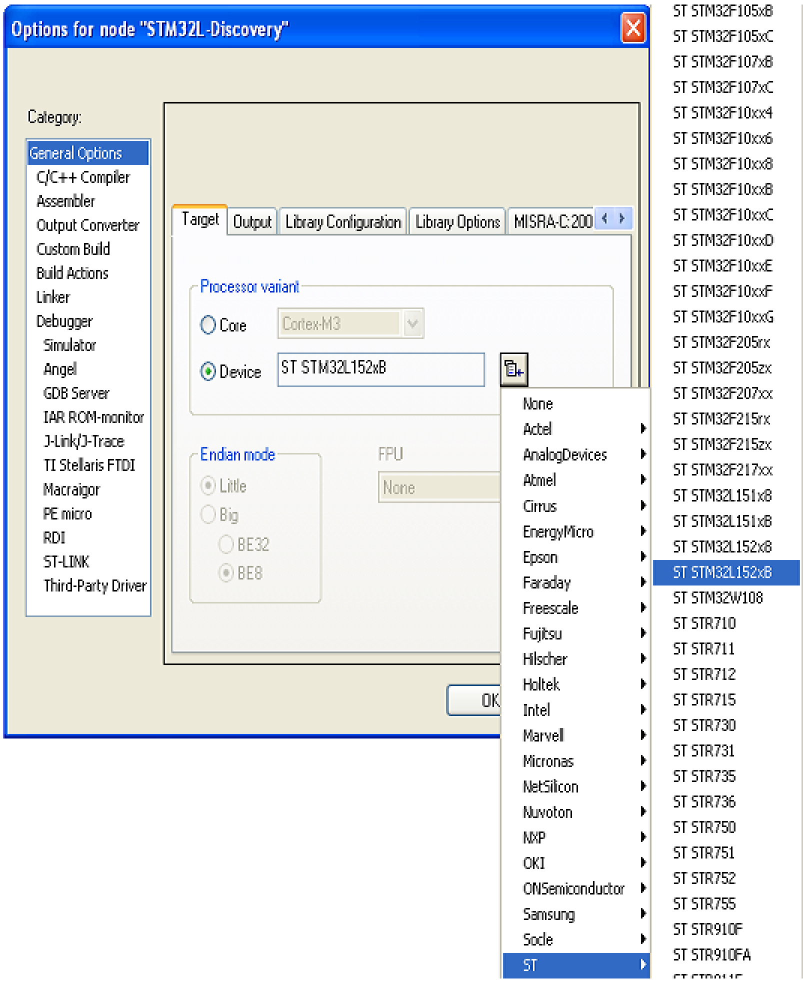
1. In the Project Editor, right-click on the project name and select **Options** to display the Options dialog box

Figure 11. Configuring project options



2. In the Options dialog box, select the **General Options** category, open the **Target** tab and select **Device - ST -STM32L152xB.**

Figure 12. General options > Target tab


Options for node "STM32L-Discovery"

Category: General Options

Target

Processor variant

○ Core Cortex-M3

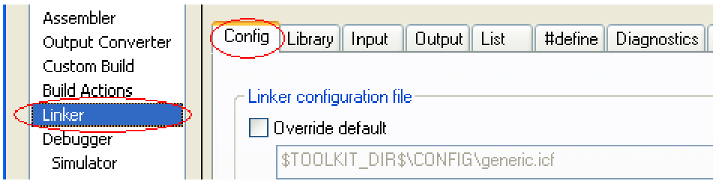
● Device ST STM32L152xB

Endian mode: ● Little ○ Big ○ BE32 ● BE8

FPU: None

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.

Figure 13. Linker > Config tab



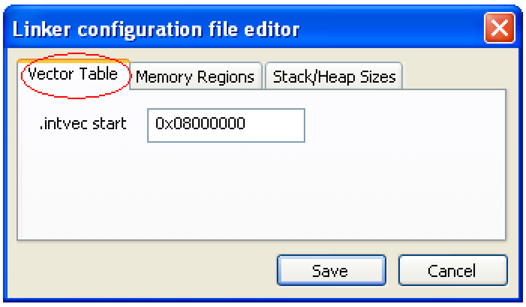
Linker configuration file

□ Override default

$TOOLKIT_DIR$\CONFIG\generic.icf

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

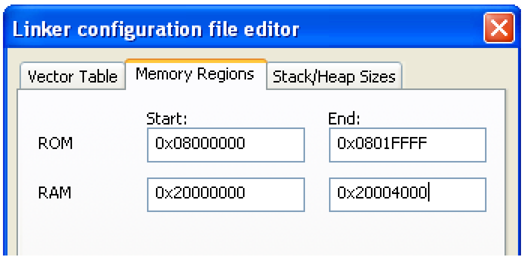
Figure 14. Linker configuration file editor dialog box > Vector Table tab



.intvec start 0x08000000

5. Open the **Memory Regions** tab, and enter the variables as shown in *Figure 15*.

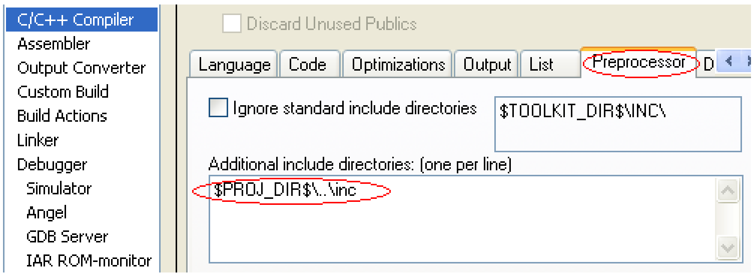
Figure 15. Linker configuration file editor dialog box > Memory Regions tab



	Start:	End:
ROM	0x08000000	0x0801FFFF
RAM	0x20000000	0x20004000

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 16*. The path of the include directory is a relative path, and always starts with the project directory location referenced by **`$PROJ_DIR$`**

Figure 16. C/C++ Compiler > Preprocessor tab



□ Discard Unused Publics

□ Ignore standard include directories

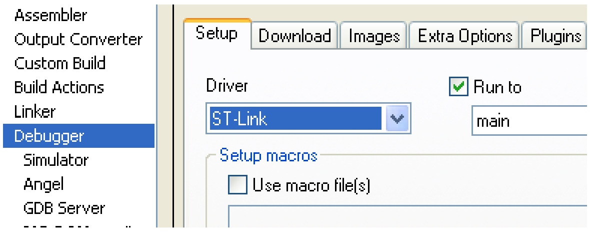
$TOOLKIT_DIR$\INC\

Additional include directories: (one per line)

$PROJ_DIR$\..\inc

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

Figure 17. Debugger > Setup tab



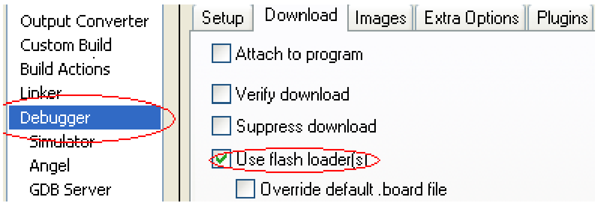
Driver: ST-Link

☑ Run to: main

□ Use macro file(s)

9. Open the **Download** tab and select **Use Flash loader(s)**.

Figure 18. Select Flash loaders



□ Attach to program

□ Verify download

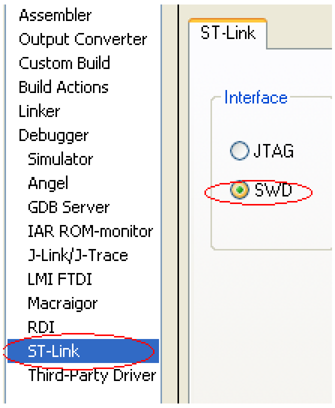
□ Suppress download

☑ Use flash loader(s)

□ Override default .board file

10. Select the **ST-Link** category, open the ST6Link tab and select **SWD** as the connection protocol.

Figure 19. ST-Link communication protocol



11. Click **OK** to save the project settings.
12. To build your project, follow the instructions given in *Section 3.1: Building an existing EWARM project on page 5*.
13. Before running your application, establish the connection with the STM32L-DISCOVERY board as described in *Section 2: Hardware environment setup on page 4*.
14. To program the Flash memory and begin debugging, follow the instructions given in *Section 3.2: Debugging and running your EWARM project on page 6*.

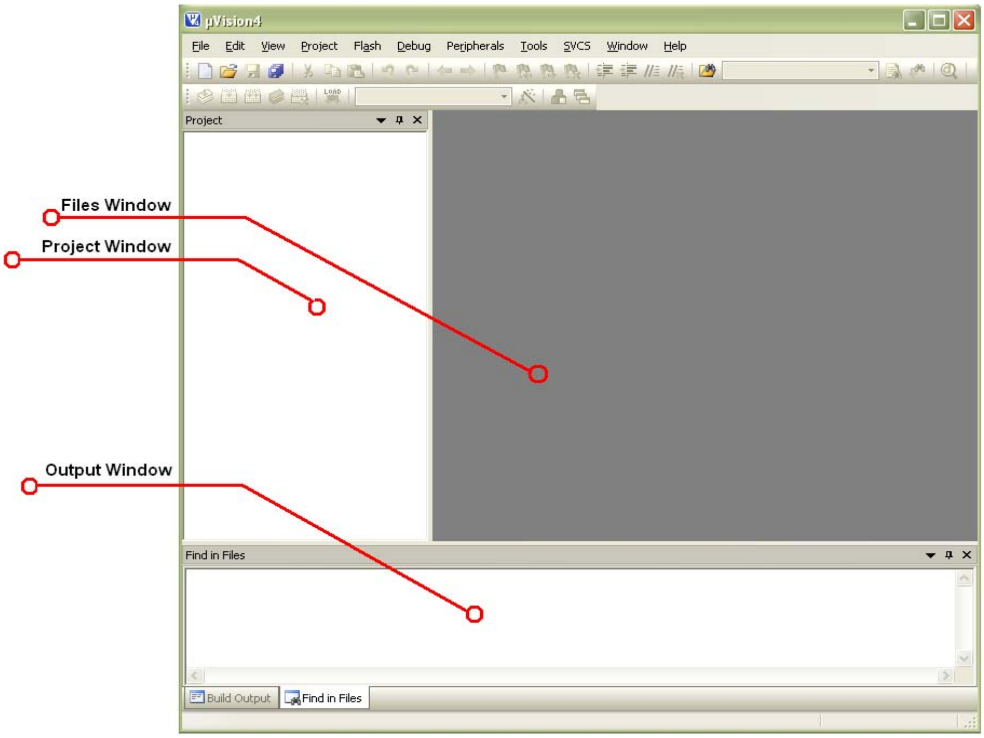
4 Using the MDK-ARM Microcontroller Development Kit by Keil™

4.1 Building an existing MDK-ARM project

1. Open the MDK-ARM µVision4 IDE, debugger, and simulation environment.

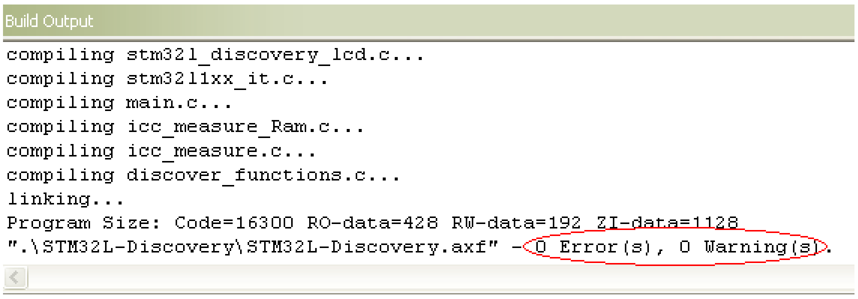
   Figure 2 shows the basic names of the windows referred to in this document.

Figure 20. 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 *STM32L-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 window is displayed.

Figure 21. MDK-ARM µVision4 project successfully compiled

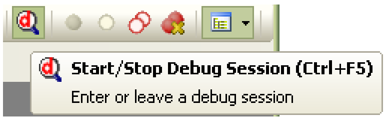


```
compiling stm32l_discovery_lcd.c...
compiling stm32l1xx_it.c...
compiling main.c...
compiling icc_measure_Ram.c...
compiling icc_measure.c...
compiling discover_functions.c...
linking...
Program Size: Code=16300 RO-data=428 RW-data=192 ZI-data=1128
".\STM32L-Discovery\STM32L-Discovery.axf" - 0 Error(s), 0 Warning(s).
```

4.2 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.

Figure 22. Starting a MDK-ARM µVision4 debugging session

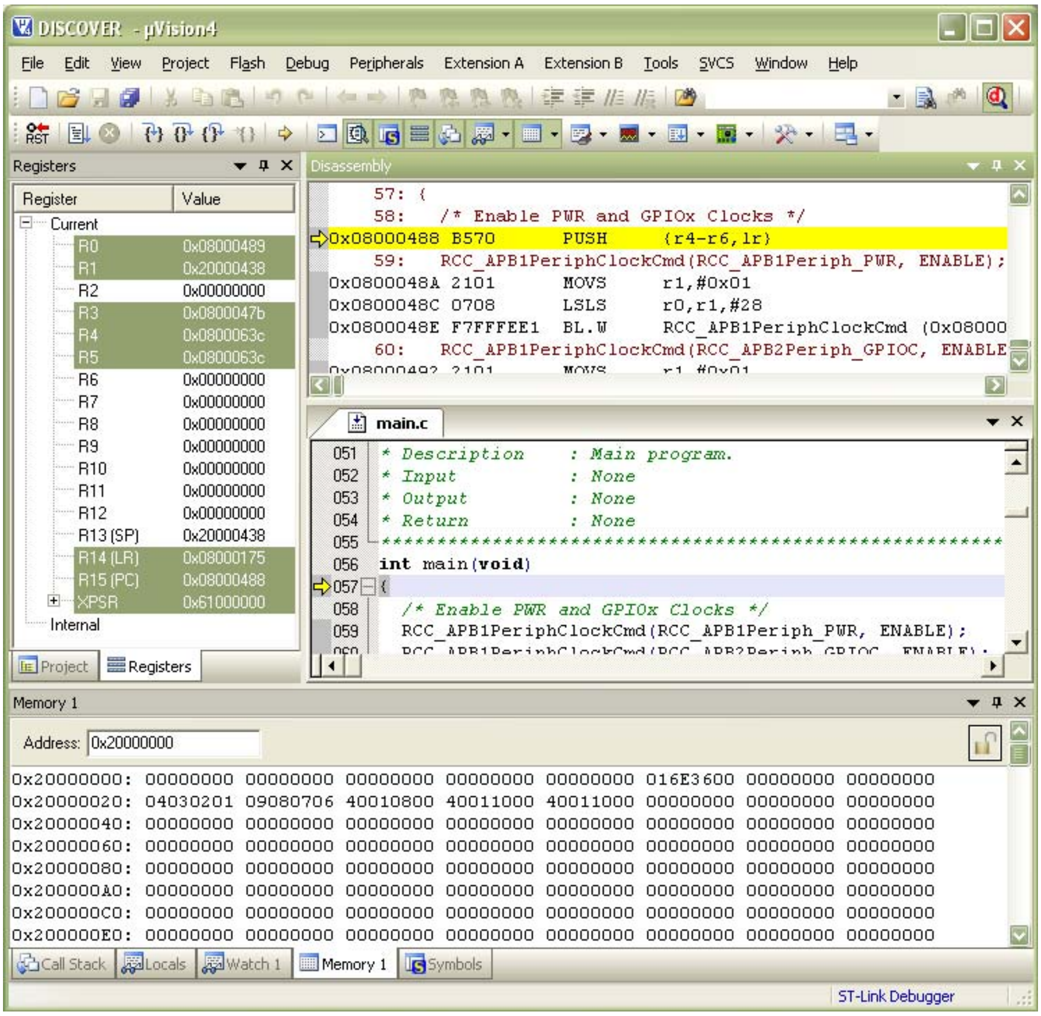


Start/Stop Debug Session (Ctrl+F5)

Enter or leave a debug 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.

Figure 23. MDK-ARM IDE workspace



DISCOVER - µVision4
File Edit View Project Flash Debug Peripherals Extension A Extension B Tools SVCS Window Help
RST
Registers
Disassembly
main.c
Project Registers
Memory 1
Call Stack Locals Watch 1 Memory 1 Symbols
ST-Link Debugger

Register	Value
Current	
R0	0x08000489
R1	0x20000438
R2	0x00000000
R3	0x0800047b
R4	0x0800063c
R5	0x0800063c
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000438
R14 (LR)	0x08000175
R15 (PC)	0x08000488
XPSR	0x61000000
Internal	

```
    57: {
    58:    /* Enable PWR and GPIOx Clocks */
0x08000488 B570      PUSH     {r4-r6,lr}
    59:    RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);
0x0800048A 2101      MOVS     r1,#0x01
0x0800048C 0708      LSLS     r0,r1,#28
0x0800048E F7FFFEE1  BL.W     RCC_APB1PeriphClockCmd (0x08000
    60:    RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE
0x08000492 2101      MOVS     r1,#0x01
```

```
* Description    : Main program.
* Input          : None
* Output         : None
* Return         : None
*******************************************************
int main(void)
{
  /* Enable PWR and GPIOx Clocks */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
```

Address: 0x20000000

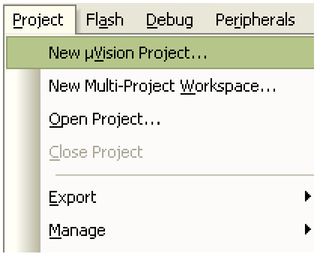
```
0x20000000: 00000000 00000000 00000000 00000000 00000000 016E3600 00000000 00000000
0x20000020: 04030201 09080706 40010800 40011000 40011000 00000000 00000000 00000000
0x20000040: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x20000060: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x20000080: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x200000A0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x200000C0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x200000E0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

4.3 Creating your first application using the MDK-ARM toolchain

4.3.1 Managing source files

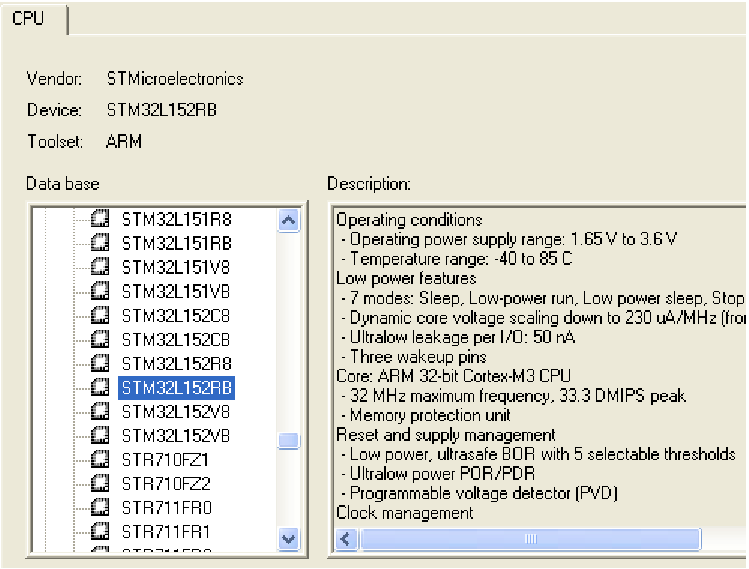
1. In the **Project** menu, select **New µvision Project...** to display the Create Project File dialog box. Name the new project and click **Save**.

Figure 24. Creating a new project



2. When a new project is saved, the IDE displays the device dialog box. Select the device used for testing. In this example, we will use the STMicroelectronics device mounted on the STM32L-DISCOVERY board. In this case, double-click on **STMicroelectronics**, select the **STM32L152RB** device and click **OK** to save your settings.

Figure 25. Device selection dialog box



Vendor: STMicroelectronics
Device: STM32L152RB
Toolset: ARM

Data base

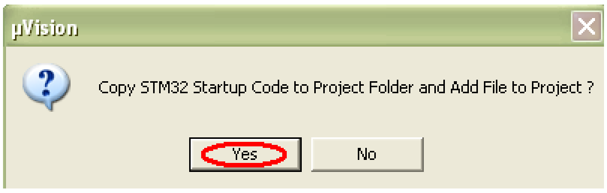
- STM32L151R8
- STM32L151RB
- STM32L151V8
- STM32L151VB
- STM32L152C8
- STM32L152CB
- STM32L152R8
- STM32L152RB
- STM32L152V8
- STM32L152VB
- STR710FZ1
- STR710FZ2
- STR711FR0
- STR711FR1

Description:

Operating conditions
- Operating power supply range: 1.65 V to 3.6 V
- Temperature range: -40 to 85 C
Low power features
- 7 modes: Sleep, Low-power run, Low power sleep, Stop
- Dynamic core voltage scaling down to 230 uA/MHz (fro
- Ultralow leakage per I/O: 50 nA
- Three wakeup pins
Core: ARM 32-bit Cortex-M3 CPU
- 32 MHz maximum frequency, 33.3 DMIPS peak
- Memory protection unit
Reset and supply management
- Low power, ultrasafe BOR with 5 selectable thresholds
- Ultralow power POR/PDR
- Programmable voltage detector (PVD)
Clock management

3. Click Yes to copy the STM32 Startup Code to the project folder and add the file to the project.

Figure 26. Copy the STM32 Startup Code dialog box



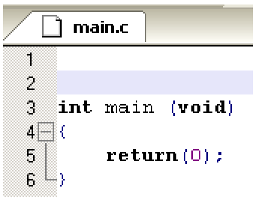
Copy STM32 Startup Code to Project Folder and Add File to Project ?

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_stm32l1xx.c file from the STM32l1xx 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, the file is saved as **main.c**.

Figure 27. main.c example file

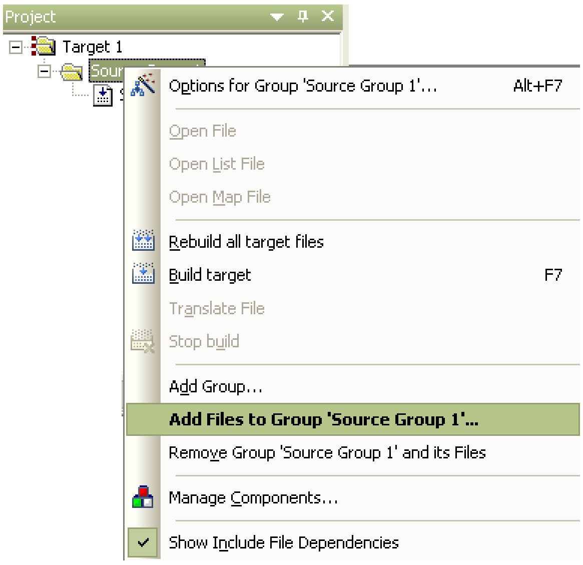


```

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.

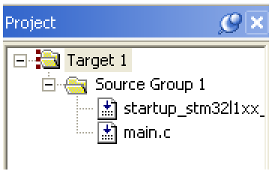
Figure 28. Adding source files



Project
Target 1
Sour
Options for Group 'Source Group 1'... Alt+F7
Open File
Open List File
Open Map File
Rebuild all target files
Build target F7
Translate File
Stop build
Add Group...
Add Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Manage Components...
Show Include File Dependencies

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

Figure 29. New project file tree structure

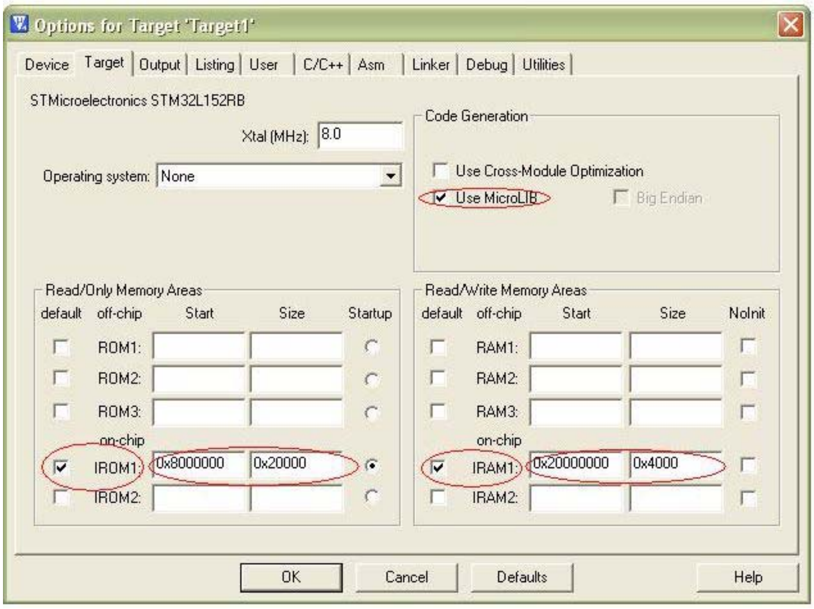


```
Project
⊟ Target 1
   ⊟ Source Group 1
        startup_stm32l1xx_
        main.c
```

4.3.2 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 30*.

Figure 30. Target Options dialog box - Target tab



Options for Target 'Target1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

STMicroelectronics STM32L152RB

Xtal (MHz): 8.0

Operating system: None

Code Generation

□ Use Cross-Module Optimization

☑ Use MicroLIB □ Big Endian

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
□	ROM1:			○
□	ROM2:			○
□	ROM3:			○
	on-chip			
☑	IROM1:	0x8000000	0x20000	●
□	IROM2:			○

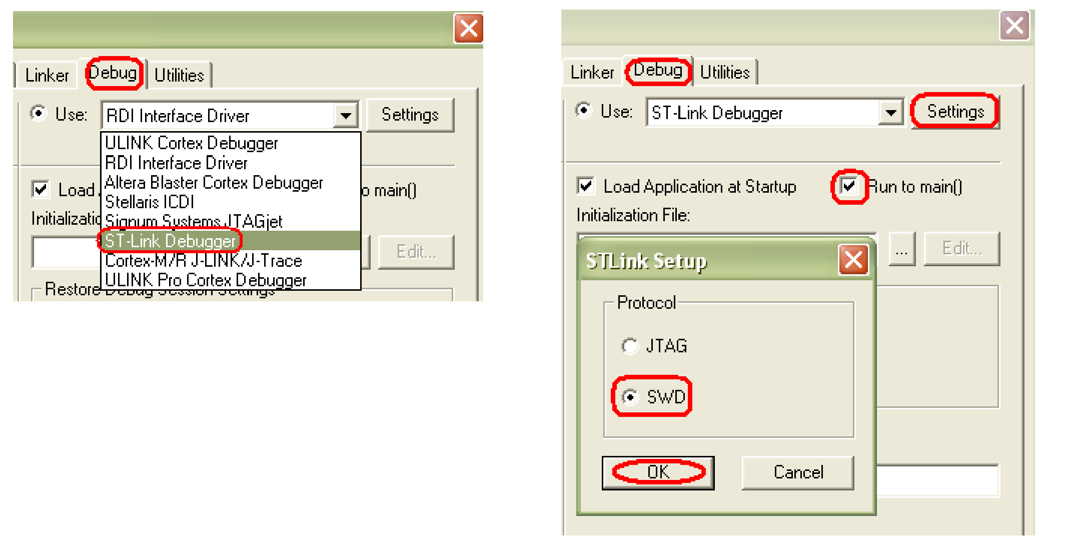
Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
□	RAM1:			□
□	RAM2:			□
□	RAM3:			□
	on-chip			
☑	IRAM1:	0x20000000	0x4000	□
□	IRAM2:			□

OK | Cancel | Defaults | Help

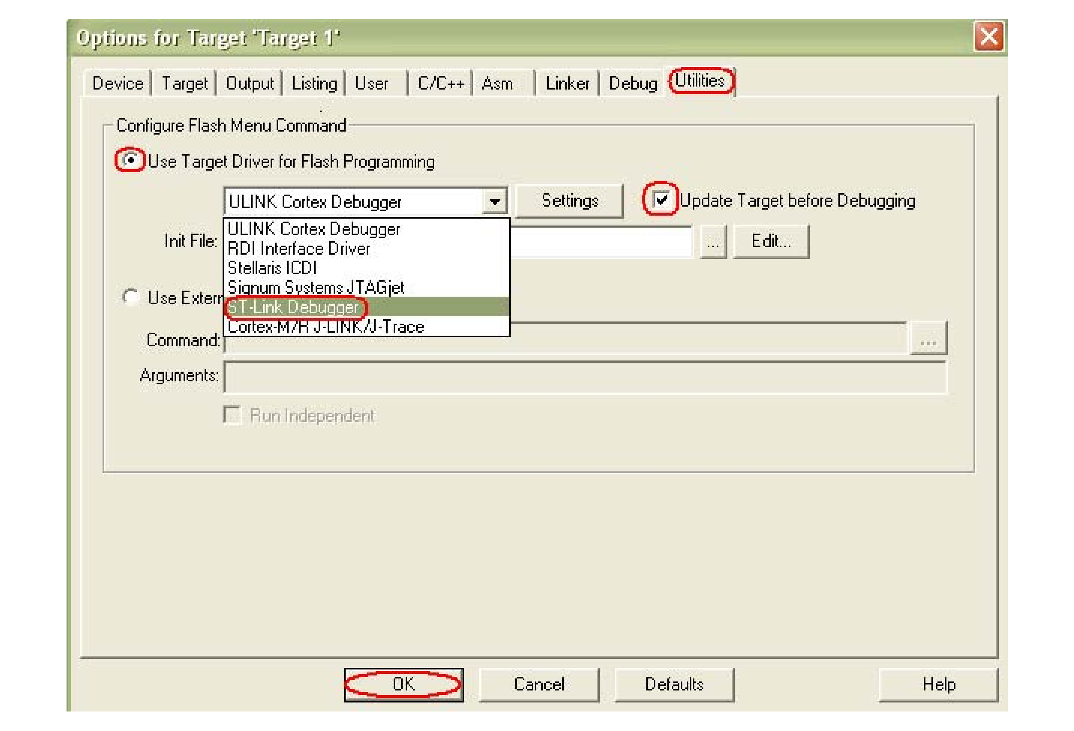
3. Open the **Debug** tab, click **Use** and select the **ST-Link Debugger**. Then, click **Settings** and select the **SWD** protocol. Click **OK** to save the ST-Link setup settings.
4. Select **Run to main()**.

Figure 31. 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.

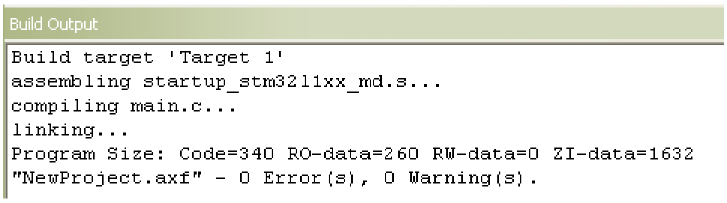
Figure 32. Target Options dialog box - Utilities tab



8. In the **Project** menu, select **Build Target**.

9. If your project is successfully built, the following window is displayed.

Figure 33. MDK-ARM µVision4 project successfully built



```
Build target 'Target 1'
assembling startup_stm32l1xx_md.s...
compiling main.c...
linking...
Program Size: Code=340 RO-data=260 RW-data=0 ZI-data=1632
"NewProject.axf" - 0 Error(s), 0 Warning(s).
```

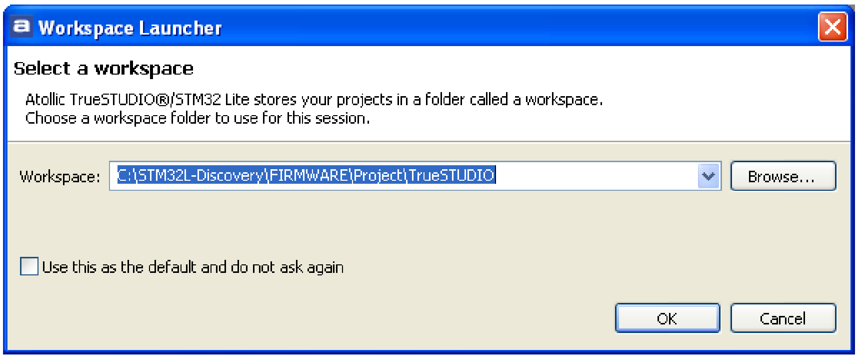
10. Before running your application, establish the connection with the STM32L-DISCOVERY board as described in *Section 2: Hardware environment setup on page 4*.
11. To program the Flash memory and begin debugging, follow the instructions given in *Section 4.2: Debugging and running your MDK-ARM project on page 14*.

5 Using the Atollic TrueSTUDIO®

5.1 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 34. TrueSTUDIO workspace launcher dialog box



Workspace Launcher

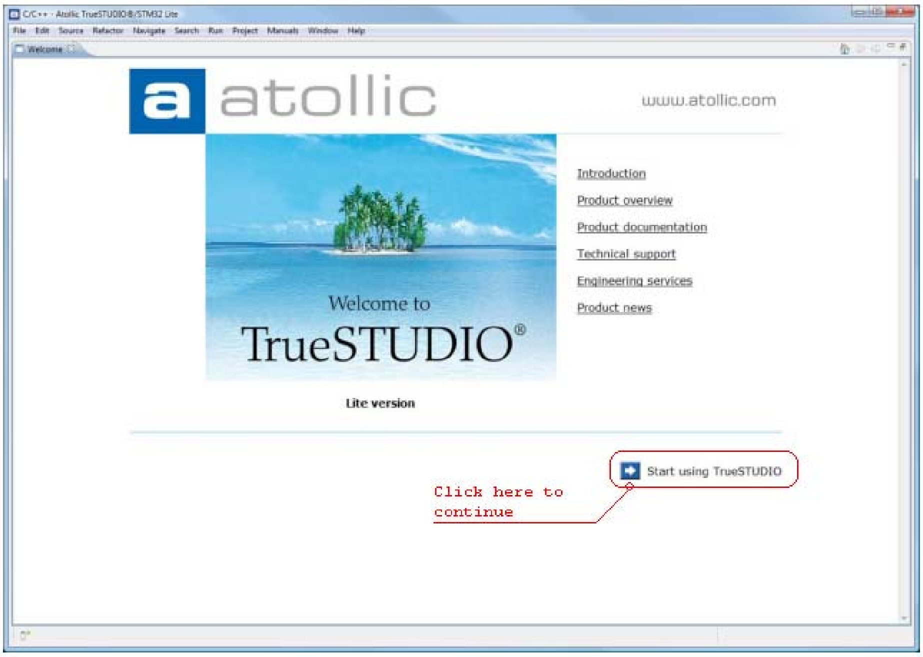
Select a workspace

Atollic TrueSTUDIO®/STM32 Lite stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\STM32L-Discovery\FIRMWARE\Project\TrueSTUDIO

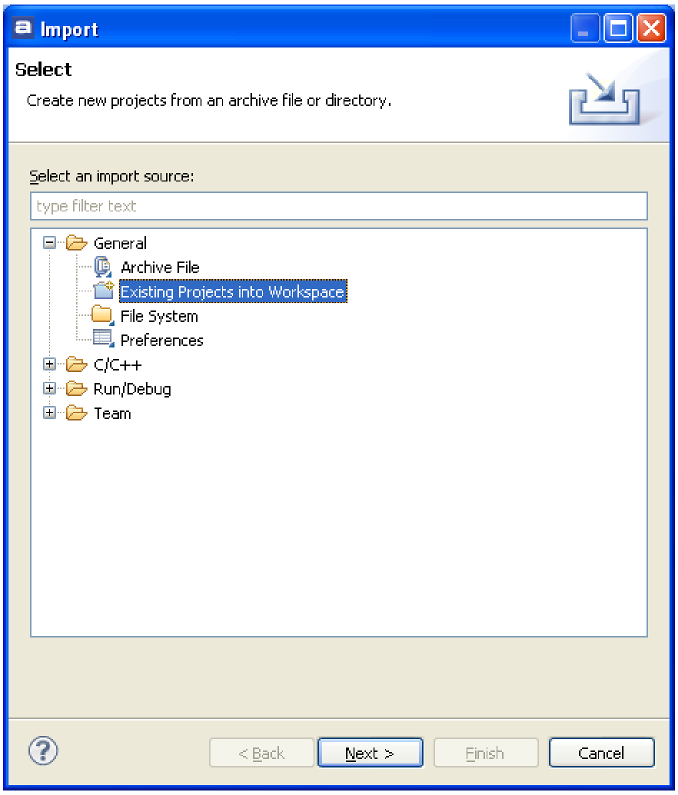
2. Browse to select the STM32L-DISCOVERY Demonstration TrueSTUDIO workspace and click **OK** to save your settings and to display the Welcome screen. To start using Atollic TrueSTUDIO®, click **Start using TrueSTUDIO**.

Figure 35. Atollic TrueSTUDIO®/STM32 Lite welcome screen



3. The TrueSTUDIO Discovery workspace contains a demo project for the STM32L-DISCOVERY 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 36. Atollic TrueSTUDIO®/STM32 Lite import source select dialog box



Import

Select

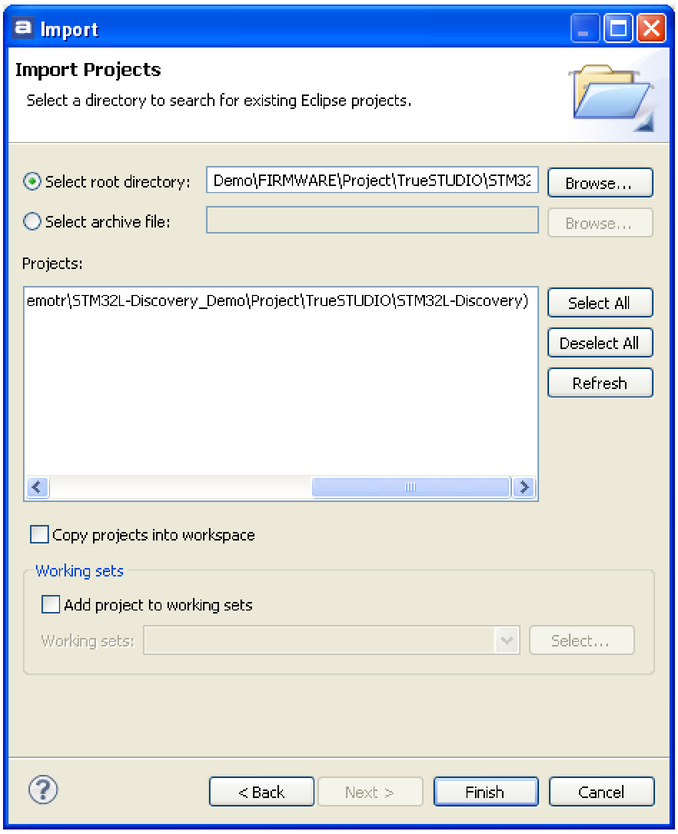
Create new projects from an archive file or directory.

Select an import source:

type filter text

5. Click **Select root directory**, browse to the TrueSTUDIO workspace folder and select the **STM32L-DISCOVERY** project.

Figure 37. Atollic TrueSTUDIO®/STM32 Lite import projects dialog box



Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: Demo\FIRMWARE\Project\TrueSTUDIO\STM32

○ Select archive file:

Projects:

emotr\STM32L-Discovery_Demo\Project\TrueSTUDIO\STM32L-Discovery)

□ Copy projects into workspace

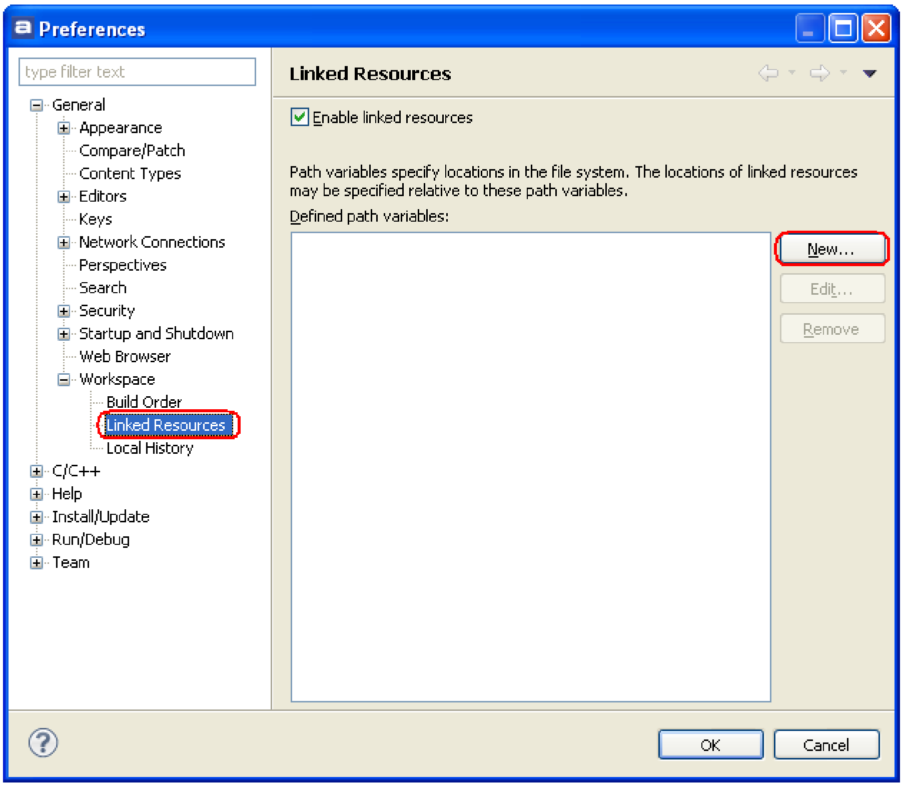
Working sets

□ Add project to working sets

Working sets:

6. In the **Projects** pane, select the **STM32L-DISCOVERY** and click **Finish**.
7. When the project is loaded, in the **Window** menu, open **Preferences**, select **General**, **Workspace**, **Linked Resources** and click **New** to add a path variable.

Figure 38. Adding a new path variable



Linked Resources

☑ Enable linked resources

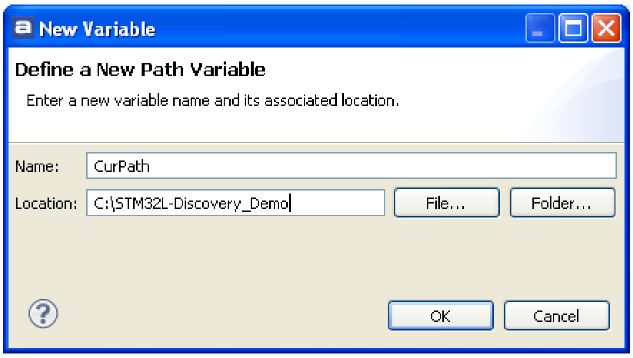
Path variables specify locations in the file system. The locations of linked resources may be specified relative to these path variables.

Defined path variables:

8. Add a path variable named `CurPath` which points to the *STM32L-Discovery_Demo* folder containing the *Libraries*, *Project* and *Utilities* folders.

Note: All files in the STM32L-DISCOVERY project are linked using a path variable called "CurPath" to allow users to copy and run this project under any path location, just by updating this variable.

Figure 39. Defining a new path variable



Define a New Path Variable

Enter a new variable name and its associated location.

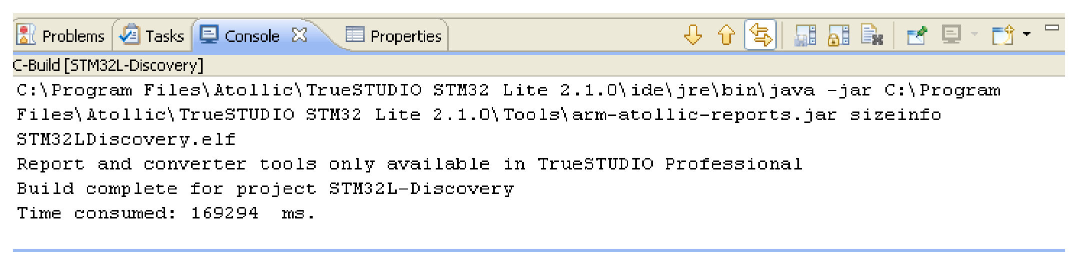
Name: CurPath

Location: C:\STM32L-Discovery_Demo

9. In the **Project Explorer**, select the STM32L-DISCOVERY project. Open the **Project** menu, and click **Build Project**.

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

Figure 40. TrueSTUDIO® project successfully compiled



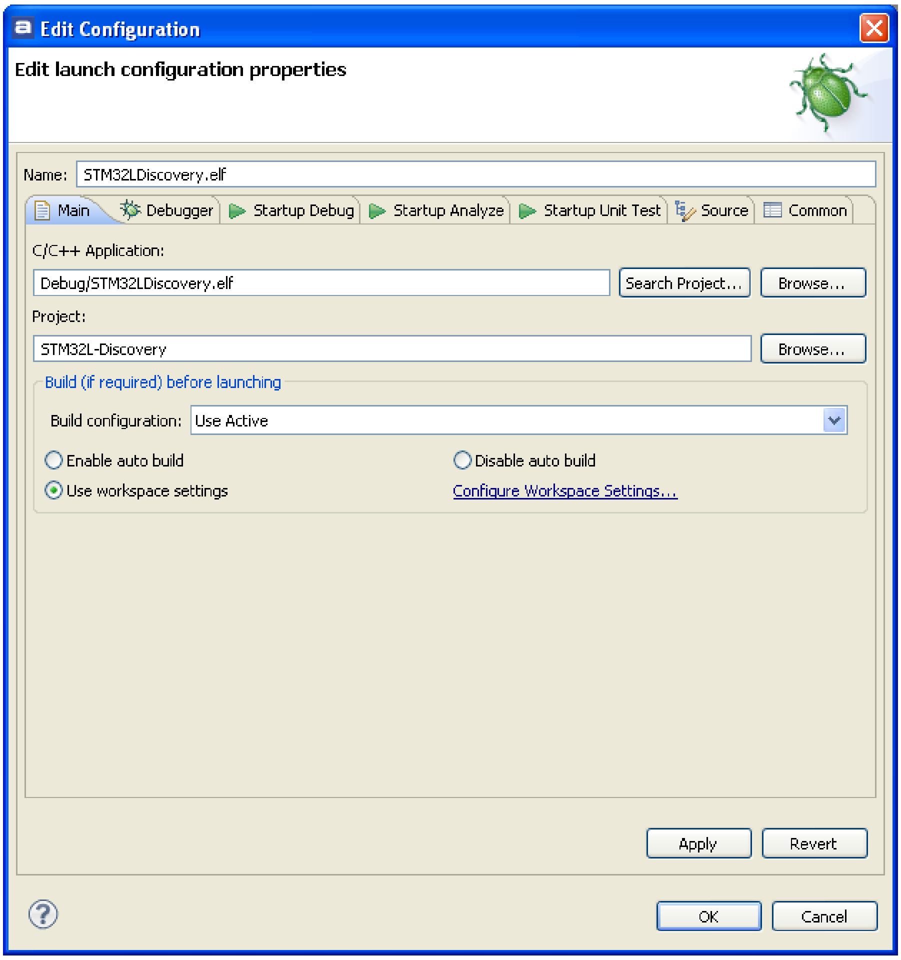
C-Build [STM32L-Discovery]

```
C:\Program Files\Atollic\TrueSTUDIO STM32 Lite 2.1.0\ide\jre\bin\java -jar C:\Program
Files\Atollic\TrueSTUDIO STM32 Lite 2.1.0\Tools\arm-atollic-reports.jar sizeinfo
STM32LDiscovery.elf
Report and converter tools only available in TrueSTUDIO Professional
Build complete for project STM32L-Discovery
Time consumed: 169294  ms.
```

5.2 Debugging and running your TrueSTUDIO project

In the **Project Explorer**, select the STM32L-DISCOVERY project and press **F11** to display the Edit Configuration dialog box.

Figure 41. TrueSTUDIO Edit Configuration dialog box



Name: STM32LDiscovery.elf

C/C++ Application: Debug/STM32LDiscovery.elf

Project: STM32L-Discovery

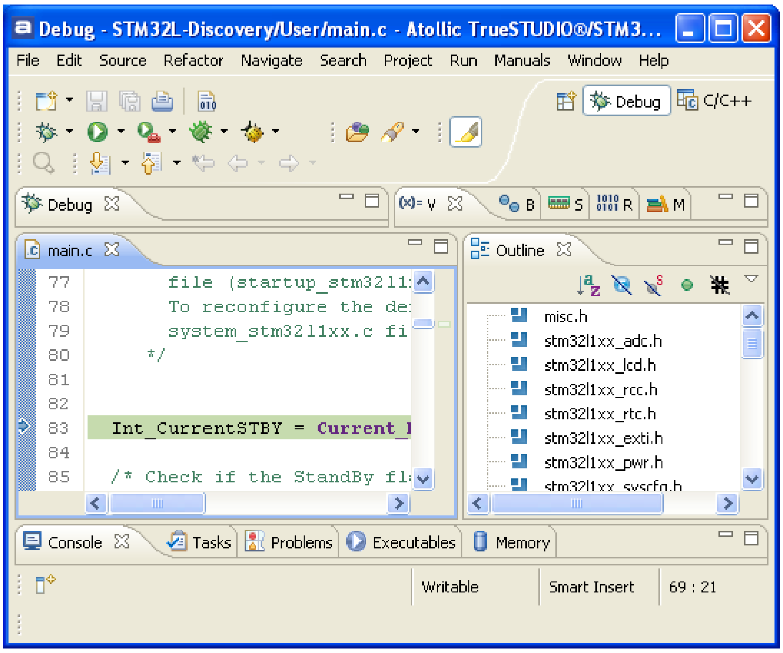
Build configuration: Use Active

○ Enable auto build ○ Disable auto build

● Use workspace settings

11. 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



```
        file (startup_stm32l1
        To reconfigure the de
        system_stm32l1xx.c fi
      */


  Int_CurrentSTBY = Current_I

/* Check if the StandBy fl
```

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.

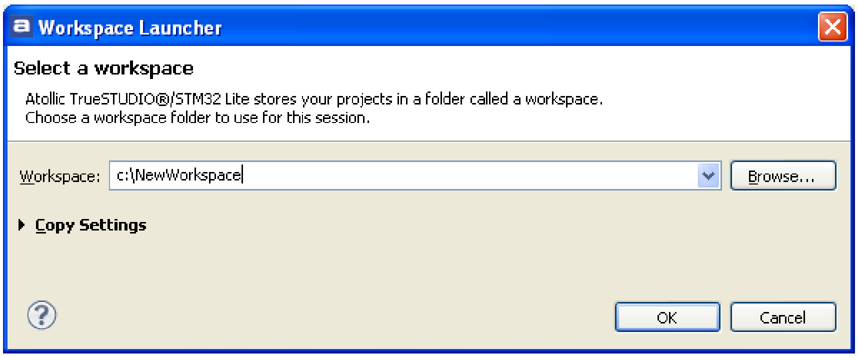
To run your application, from the **Run** menu, select **Resume**, or alternatively click the **Resume** button in the toolbar.

5.3 Creating your first application using TrueSTUDIO toolchain

TrueSTUDIO includes a dedicated connection to the STM32L-DISCOVERY 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 **STM32L_DISCOVERY** as the evaluation board.

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.

Figure 43. TrueSTUDIO workspace launcher dialog box



Workspace Launcher

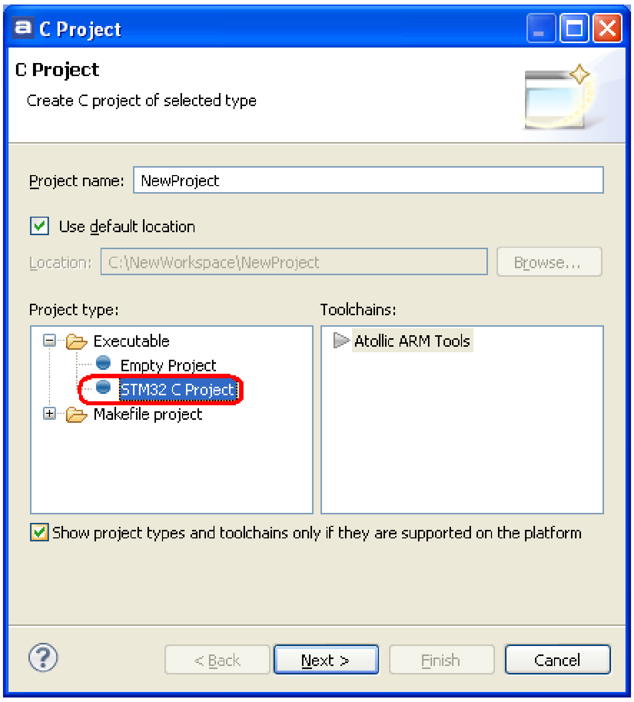
Select a workspace

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

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



C Project

C Project

Create C project of selected type

Project name: NewProject

☑ Use default location

Location: C:\NewWorkspace\NewProject

Project type:

- Executable
  - Empty Project
  - STM32 C Project
- Makefile project

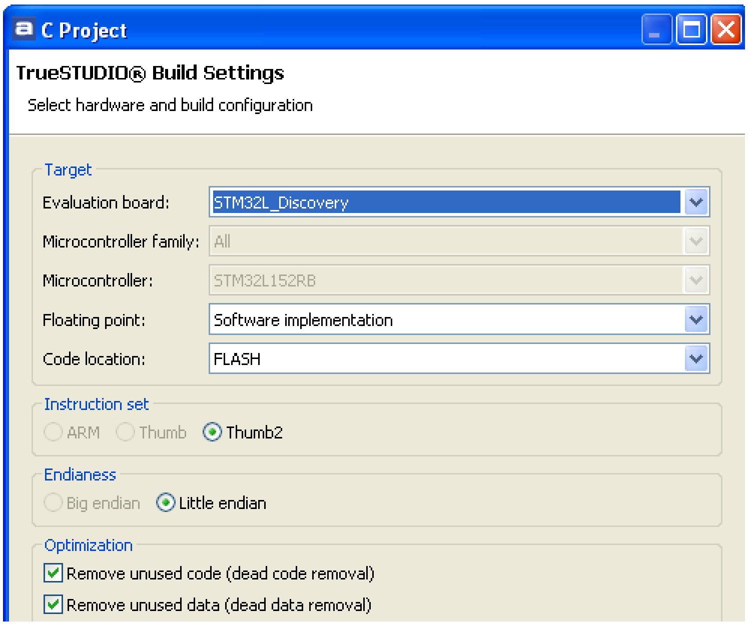
Toolchains:

- Atollic ARM Tools

☑ Show project types and toolchains only if they are supported on the platform

4. In the TrueSTUDIO® Build Settings dialog box, select **STM32L_DISCOVERY** as **Evaluation board**, configure the other settings as shown in *Figure 45* and click **Next**.

Figure 45. TrueSTUDIO® Build Settings dialog box



TrueSTUDIO® Build Settings

Select hardware and build configuration

Target

Evaluation board: STM32L_Discovery

Microcontroller family: All

Microcontroller: STM32L152RB

Floating point: Software implementation

Code location: FLASH

Instruction set

○ ARM ○ Thumb ● Thumb2

Endianess

○ Big endian ● Little endian

Optimization

☑ Remove unused code (dead code removal)

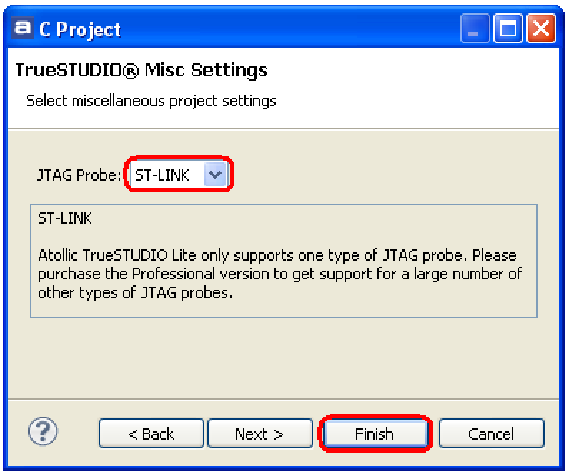
☑ Remove unused data (dead data removal)

Note: *Choosing STM32L-DISCOVERY as the evaluation board, will configure the project as follows:*

- *Microcontroller: STM32L152RB*
- *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 46. TrueSTUDIO® Misc Settings dialog box



TrueSTUDIO® Misc Settings

Select miscellaneous project settings

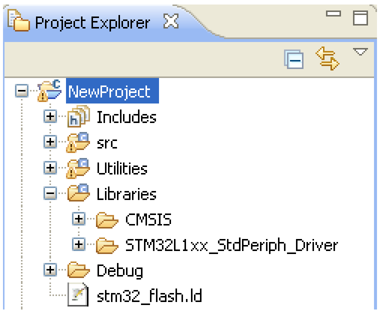
JTAG Probe: ST-LINK

ST-LINK

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.

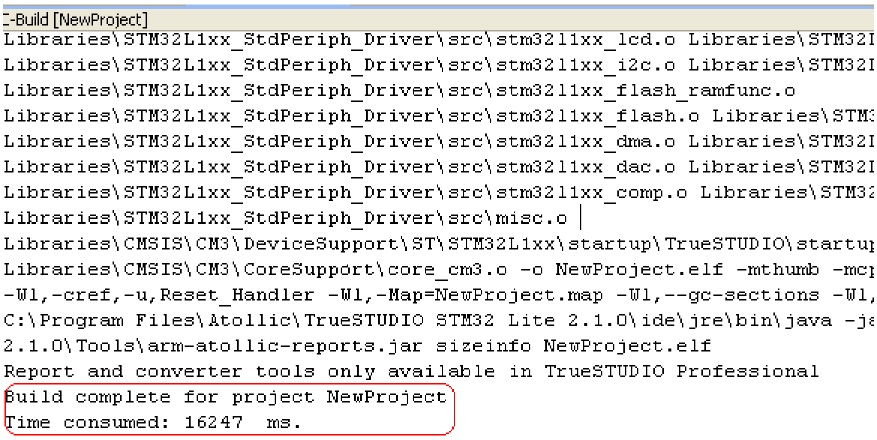
6. Your project is successfully created. Atollic TrueSTUDIO® generates target specific sample files (main.c, stm32l1xx_it.c...) in the Project folder to simplify development. You can tailor this project to your needs by modifying these sample files.

Figure 47. TrueSTUDIO® project folder example



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

Figure 48. TrueSTUDIO® project successfully built



```
-Build [NewProject]
Libraries\STM32L1xx_StdPeriph_Driver\src\stm32l1xx_lcd.o Libraries\STM32
Libraries\STM32L1xx_StdPeriph_Driver\src\stm32l1xx_i2c.o Libraries\STM32
Libraries\STM32L1xx_StdPeriph_Driver\src\stm32l1xx_flash_ramfunc.o
Libraries\STM32L1xx_StdPeriph_Driver\src\stm32l1xx_flash.o Libraries\STM3
Libraries\STM32L1xx_StdPeriph_Driver\src\stm32l1xx_dma.o Libraries\STM32
Libraries\STM32L1xx_StdPeriph_Driver\src\stm32l1xx_dac.o Libraries\STM32
Libraries\STM32L1xx_StdPeriph_Driver\src\stm32l1xx_comp.o Libraries\STM3
Libraries\STM32L1xx_StdPeriph_Driver\src\misc.o
Libraries\CMSIS\CM3\DeviceSupport\ST\STM32L1xx\startup\TrueSTUDIO\startu
Libraries\CMSIS\CM3\CoreSupport\core_cm3.o -o NewProject.elf -mthumb -mc
-Wl,-cref,-u,Reset_Handler -Wl,-Map=NewProject.map -Wl,--gc-sections -Wl,
C:\Program Files\Atollic\TrueSTUDIO STM32 Lite 2.1.0\ide\jre\bin\java -j
2.1.0\Tools\arm-atollic-reports.jar sizeinfo NewProject.elf
Report and converter tools only available in TrueSTUDIO Professional
Build complete for project NewProject
Time consumed: 16247  ms.
```

9. Before running your application, establish the connection with the STM32L-DISCOVERY board as described in *Section 2: Hardware environment setup on page 4*.
10. To program the Flash memory and begin debugging, follow the instructions given in *Section 5.2: Debugging and running your TrueSTUDIO project on page 24*.

6 Revision history

Table 1. Document revision history

Date	Revision	Changes
27-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.

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