UM2052
User manual

Getting started with STM32 MCU Discovery Kits software development tools

Introduction

The STM32 Discovery boards are low-cost and easy-to-use development platforms used to quickly evaluate and start a development with an STM32 device.

This document provides guidelines for beginners on STM32 MCU Discovery Kits on how to build and run a sample application and allows them to build and debug their application. It has the following structure:

- The first chapter presents software and hardware requirements (some toolchains supporting the STM32 families, ST-LINK/V2 installation and firmware package presentation).
- The second chapter provides a step by step guideline on how to build and debug an application using some toolchains:
  - IAR Embedded Workbench® for Arm® (EWARM) by IAR systems®
  - Microcontroller Development Kit for Arm® (MDK-ARM) by Keil®
  - TrueSTUDIO® by Atollic®
  - System Workbench for STM32 (SW4STM32) by AC6

This manual does not cover all the topics relevant to software development environments, but it describes the first basic steps necessary to get started with the compilers and debuggers. It also offers links to the documents needed to fully understand every single step.

Table 1. Applicable products

Type	Part numbers
STM32 MCU Discovery Kits	STM32F7308-DK, 32F0308DISCOVERY, 32F072BDISCOVERY, 32F3348DISCOVERY, 32F411EDISCOVERY, 32F412GDISCOVERY, 32F413HDISCOVERY, 32F429IDISCOVERY, 32F469IDISCOVERY, 32F723EDISCOVERY, 32F746GDISCOVERY, 32F769IDISCOVERY, 32L0538DISCOVERY, 32L100CDISCOVERY, 32L152CDISCOVERY, 32L476GDISCOVERY, 32L496GDISCOVERY, 32L4R9IDISCOVERY, B-L072Z-LRWAN1, B-L475E-IOT01A, P-L496G-CELL01, P-L496G-CELL02, STM32F0DISCOVERY, STM32F3DISCOVERY, STM32F4DISCOVERY, STM32F7508-DK, STM32VLDISCOVERY, STM32G071B-DISCO

Contents

List of tables

List of figures

1 General information

The STM32 Discovery boards embed STM32 32-bit microcontrollers based on the Arm®[a] Cortex®-M processor.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and or elsewhere.

2 System requirements

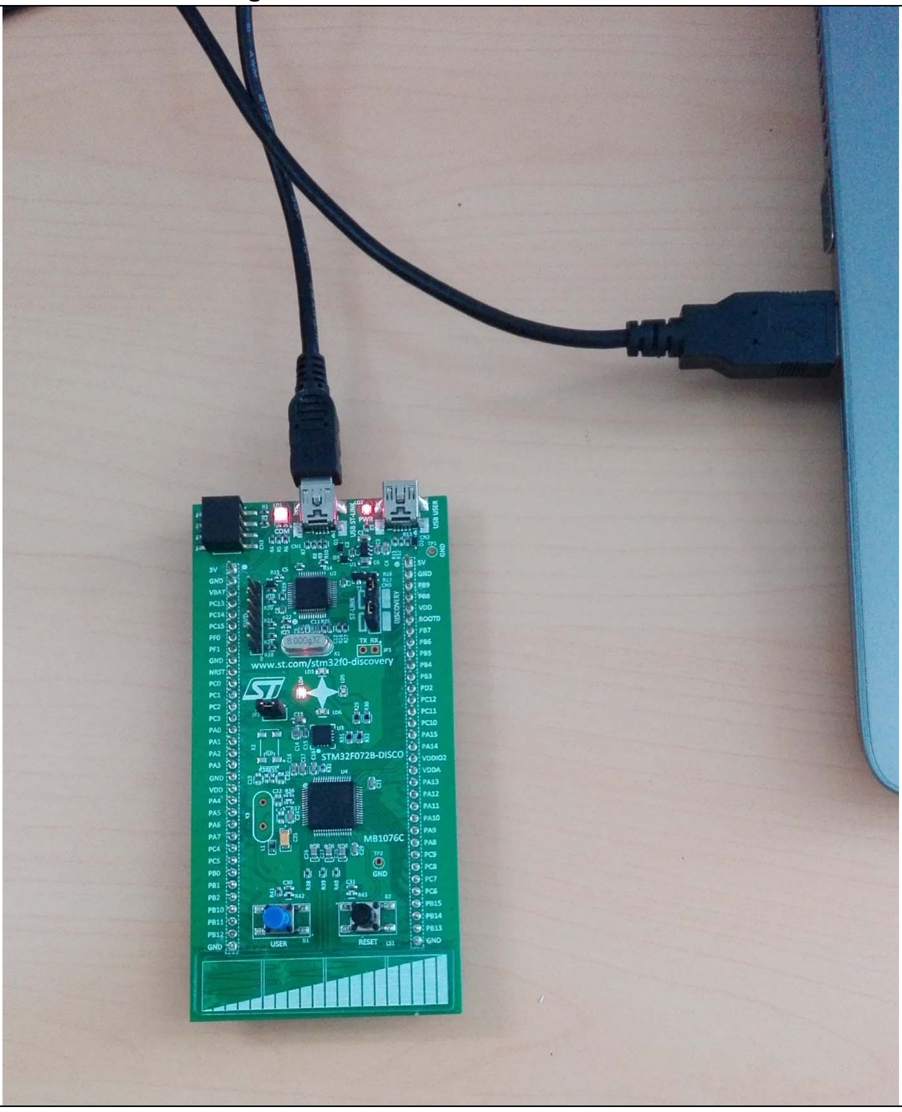
Before running the application, the user must:

1. Install the preferred Integrated Development Environment (IDE)
2. The ST-LINK/V2 driver is installed automatically. In case of problem, the user can install it manually from the toolchains install directory (further details are available in *Section 2.2: ST-LINK/V2 installation*).

Note: *This step is not needed for STM32VLDISCOVERY since it embeds an ST-LINK (not an ST-LINK/V2) which does not require a driver installation.*

3. Download the STM32 Discovery firmware from the STMicroelectronics website at *www.st.com*.
4. Establish the connection with the STM32 Discovery board as shown hereafter. In *Figure 1*, the STM32F072 Discovery board is chosen as an example.

Figure 1. Hardware environment



The steps numbered above are detailed in the coming sections.

The minimum requirements to run and develop any firmware application on the STM32 Discovery board are:

- Windows® OS (XP, 7, 8) or Linux 64-bit or Mac OS® X
- *USB type A to Mini-B* cable, used to power the STM32 Discovery board (through USB connector CN1) from the host PC and connect to the embedded ST-LINK/V2 for debugging and programming.

2.1 IDEs supporting STM32 families

The STM32 microcontrollers of 32-bit Arm Cortex-M core-based microcontrollers are supported by a complete range of software tools.

It encompasses traditional integrated-development environments such as IDEs with C/C++ compilers and debuggers from major third-party companies, free versions of up to 64 KB of code (depending on partner) and completed with innovative tools from STMicroelectronics.

The following table includes general information about some integrated development environments as well as the versions supporting the STM32 microcontrollers.

Table 2. Most used integrated development environments

Toolchain	Company	Compiler	Version	Information(1)
EWARM(2)	IAR Systems	IAR C/C++	7.60.1 and later	www.iar.com – 30-day evaluation edition KickStart edition – 16 Ko limitation for Cortex M0
MDK-ARM(2)	Keil	ARMCC	5.18a and later	www.keil.com – MDK-Lite – 32 Ko code size limitation
TrueSTUDIO(2)	Atollic	GNUC	5.5.1 and later	www.atollic.com – 30 days professional version trial – 32 Ko limitation – 8 Ko on Cortex-M0 and Cortex-M1
SW4STM32(3)	AC6	GNUC	1.8 and later	www.openstm32.org No limitation

1. Register prior to downloading the toolchain.
2. On Windows only.
3. The SW4STM32 does not support ST-LINK. Consequently, STM32VLDISCOVERY can not be used with SW4STM32 software toolchain.

Information on the toolchain version supporting the STM32 devices is available on the toolchain release note at the third-party website.

2.2 ST-LINK/V2 installation

All the STM32 Discovery boards include the ST-LINK/V2 embedded debug tool interface. This interface requires to install the ST-LINK/V2 dedicated USB driver. The STM32 Discovery boards include an ST-LINK/V2 embedded debug tool interface that is supported by the following software toolchains:

- IAR Embedded Workbench for Arm (EWARM)

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

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

- RealView Microcontroller Development Kit for Arm (MDK-ARM) by Keil toolchain

The toolchain is installed by default in the C:\Keil directory on the PC local hard disk; the installer creates a start menu µVision5 shortcut.

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

- Atollic TrueSTUDIO STM32

The toolchain is installed by default in the C:\Program Files\Atollic directory on the PC local hard disk.

- AC6 SW4STM32 STM32

The toolchain is installed by default in the C:\Program Files\AC6 directory on the PC local hard disk.

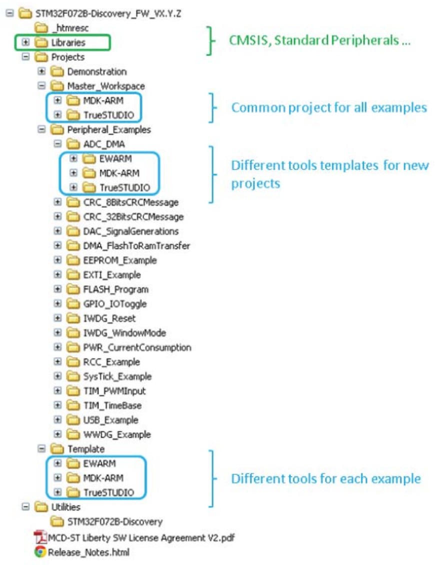
The ST-Link_V2_USB.exe is installed automatically when installing the software toolchain.

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

2.3 Firmware package

The STM32 Discovery firmware applications, demonstration and IPs examples are provided in one single package and supplied in one single .zip file. The extraction of the .zip file generates the folder STM32-Discovery_FW_VX.Y.Z, which contains the subfolders shown in *Figure 2* where the STM32 F072B-Discovery_FW_VX.Y.Z is chosen as an example.

Figure 2. Firmware package content



Template project is a pre-configured project with empty main function to be customized by the user. This is helpful to start creating an application based on the peripherals drivers.

Example project includes the toolchain projects for each peripheral example ready to be run.

Applications includes set of applications ready to be run.

Demonstration includes the demonstration firmware ready to be run.

3 Executing and debugging firmware using the software toolchains

The user can follow the procedure described below to compile/link and execute an existing EWARM project. The following steps can be applied to an already existing example, demonstration or template project available at STM32 Discovery firmware package available on STMicroelectronics website. In this manual the STM32 F072B-Discovery_FW_VX.Y.Z is used as an example.

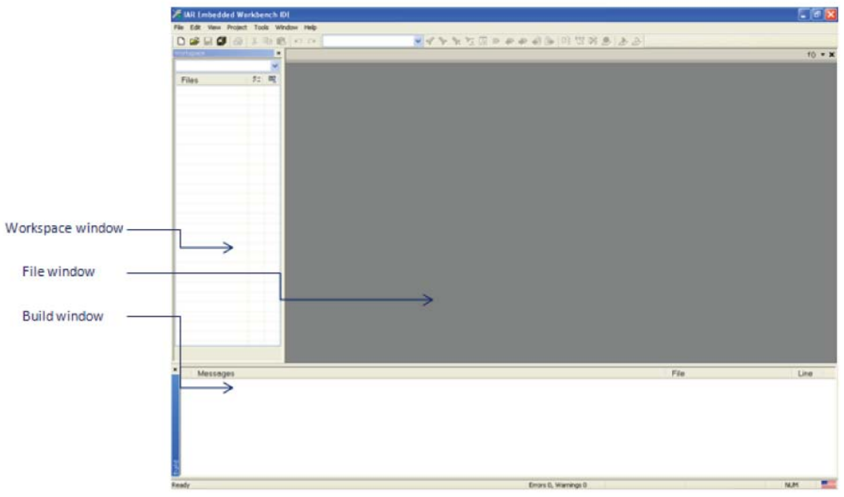
The user should go through the firmware/readme.txt file which contains the firmware description and hardware/software requirements.

3.1 EWARM toolchain

1. Open the IAR Embedded Workbench for Arm (EWARM).

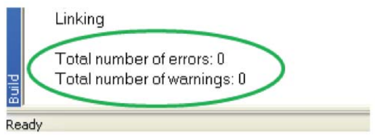
Figure 3 shows the basic names of the windows referred to in this document.

Figure 3. IAR embedded workbench IDE



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

Figure 4. EWARM project successfully compiled



Linking

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

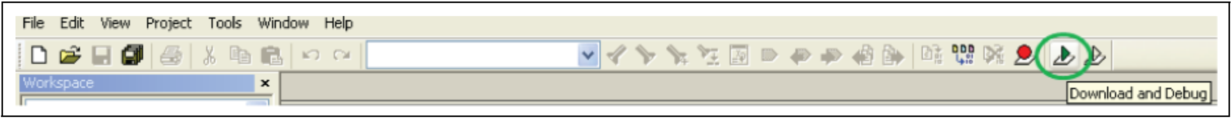
Ready

To change the project settings (Include and preprocessor defines), the user should go through project options:

- For 'Include directories', select **Project>Options…>C/C++ compiler>**
- For pre-processor, define **Project>Options…C/C++ compiler>pre-processor>**

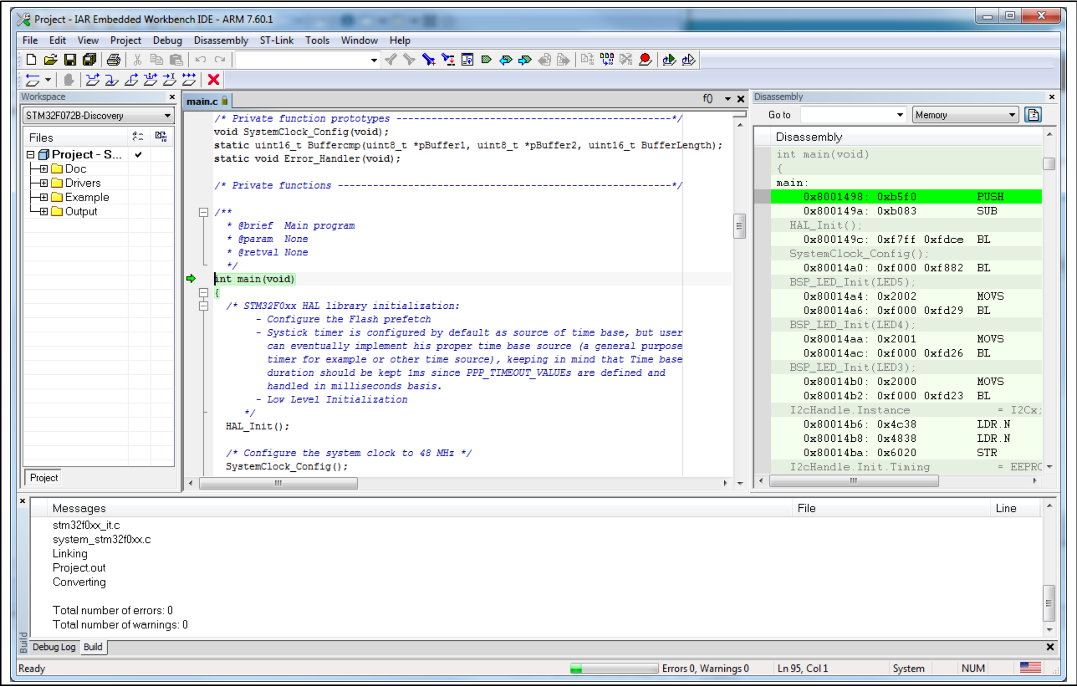
5. In the IAR Embedded Workbench IDE, from the **Project** menu, select **Download and Debug** or alternatively, click on the **Download and Debug** button the in tool bar, to program the Flash memory and start debugging.

Figure 5. Download and Debug button



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

Figure 6. IAR embedded workbench debugger screen



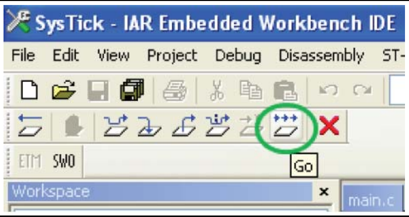
```
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static uint16_t Buffercmp(uint8_t *pBuffer1, uint8_t *pBuffer2, uint16_t BufferLength);
static void Error_Handler(void);

/* Private functions ---------------------------------------------------------*/

/**
  * @brief  Main program
  * @param  None
  * @retval None
  */
int main(void)
{
  /* STM32F0xx HAL library initialization:
       - Configure the Flash prefetch
       - Systick timer is configured by default as source of time base, but user
         can eventually implement his proper time base source (a general purpose
         timer for example or other time source), keeping in mind that Time base
         duration should be kept 1ms since PPP_TIMEOUT_VALUEs are defined and
         handled in milliseconds basis.
       - Low Level Initialization
     */
  HAL_Init();

  /* Configure the system clock to 48 MHz */
  SystemClock_Config();
```

```
Disassembly
int main(void)
{
main:
    0x8001498:  0xb5f0           PUSH
    0x800149a:  0xb083           SUB
  HAL_Init();
    0x800149c:  0xf7ff 0xfdce    BL
  SystemClock_Config();
    0x80014a0:  0xf000 0xf882    BL
  BSP_LED_Init(LED5);
    0x80014a4:  0x2002           MOVS
    0x80014a6:  0xf000 0xfd29    BL
  BSP_LED_Init(LED4);
    0x80014aa:  0x2001           MOVS
    0x80014ac:  0xf000 0xfd26    BL
  BSP_LED_Init(LED3);
    0x80014b0:  0x2000           MOVS
    0x80014b2:  0xf000 0xfd23    BL
  I2cHandle.Instance              = I2Cx;
    0x80014b6:  0x4c38           LDR.N
    0x80014b8:  0x4838           LDR.N
    0x80014ba:  0x6020           STR
  I2cHandle.Init.Timing           = EEPRC
```

```
Messages                                        File            Line
stm32f0xx_it.c
system_stm32f0xx.c
Linking
Project.out
Converting

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

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

Figure 7. Go button

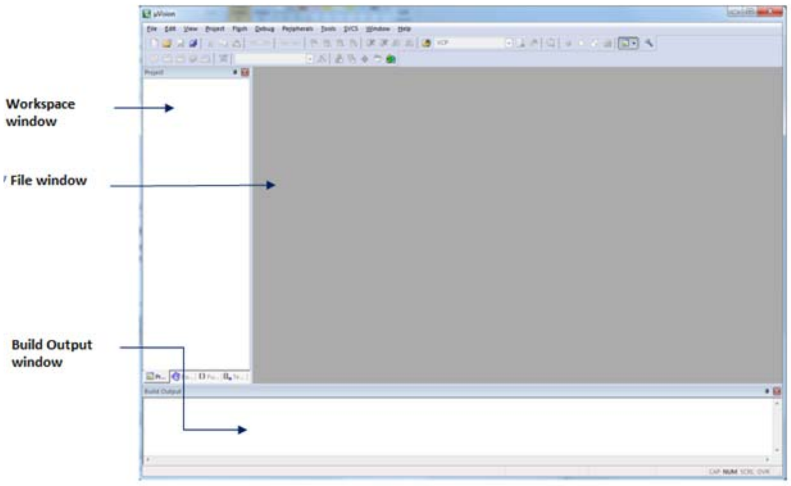


3.2 MDK-ARM toolchain

1. Open Keil MDK-ARM Microcontroller Kit,

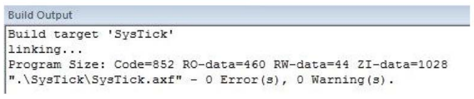
Figure 8 shows the basic names of the "Keil uVision5" windows referred to in this document.

Figure 8. uVision5 IDE



2. In the **Project** menu, select **Open Project**. Browse to select either an example or demonstration or template project file and click **Open** to launch it in the Project window.
3. In the **Project** menu, select **Rebuild All** target files to compile the project
4. If the project is successfully compiled, the window shown in *Figure 9* is displayed.

Figure 9. MDK-ARM project successfully compiled



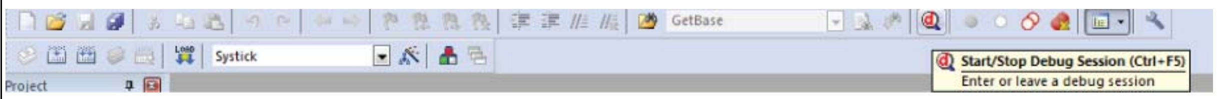
```
Build Output
Build target 'SysTick'
linking...
Program Size: Code=852 RO-data=460 RW-data=44 ZI-data=1028
".\SysTick\SysTick.axf" - 0 Error(s), 0 Warning(s).
```

If the user needs to change the project settings (Include and preprocessor defines), the user should go through the project options:

- For 'Include directories', select **Project>Options for Target > C/C++ > Include Paths**
- For pre-processor definition **Project>Options for Target > C/C++ > Preprocessor symbols > Define**

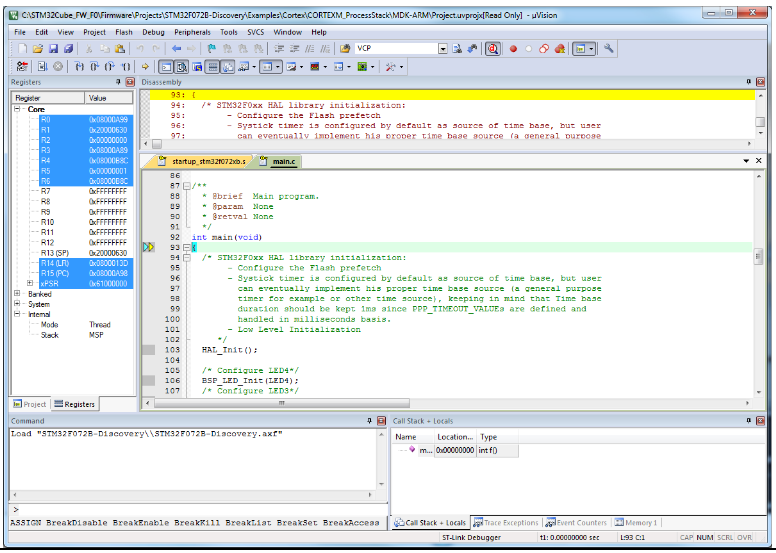
5. In the MDK-ARM IDE, from the **Debug** menu, select **Start/Stop Debug Session** or, alternatively, click the **Start/Stop Debug Session** button in the tool bar to program the Flash memory and begin debugging.

Figure 10. Start/Stop debug session button


GetBase
Systick
Project
Start/Stop Debug Session (Ctrl+F5)
Enter or leave a debug session

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

Figure 11. MDK-ARM debugger screen



C:\STM32Cube_FW_F0\Firmware\Projects\STM32F072B-Discovery\Examples\Cortex\CORTEXM_ProcessStack\MDK-ARM\Project.uvprojx[Read Only] - µVision

Register	Value
Core	
R0	0x08000A99
R1	0x20000630
R2	0x00000000
R3	0x08000A89
R4	0x08000B8C
R5	0x00000001
R6	0x08000B8C
R7	0xFFFFFFFF
R8	0xFFFFFFFF
R9	0xFFFFFFFF
R10	0xFFFFFFFF
R11	0xFFFFFFFF
R12	0xFFFFFFFF
R13 (SP)	0x20000630
R14 (LR)	0x0800013D
R15 (PC)	0x08000A98
xPSR	0x61000000
Banked	
System	
Internal	
Mode	Thread
Stack	MSP

```
    93: {
    94:   /* STM32F0xx HAL library initialization:
    95:        - Configure the Flash prefetch
    96:        - Systick timer is configured by default as source of time base, but user
    97:          can eventually implement his proper time base source (a general purpose
```

startup_stm32f072xb.s main.c

```

/**
  * @brief  Main program.
  * @param  None
  * @retval None
  */
int main(void)
{
  /* STM32F0xx HAL library initialization:
       - Configure the Flash prefetch
       - Systick timer is configured by default as source of time base, but user
         can eventually implement his proper time base source (a general purpose
         timer for example or other time source), keeping in mind that Time base
         duration should be kept 1ms since PPP_TIMEOUT_VALUEs are defined and
         handled in milliseconds basis.
       - Low Level Initialization
     */
  HAL_Init();

  /* Configure LED4*/
  BSP_LED_Init(LED4);
  /* Configure LED3*/
```

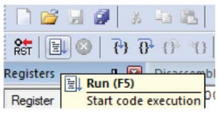
```
Load "STM32F072B-Discovery\\STM32F072B-Discovery.axf"
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Name	Location...	Type
m...	0x00000000	int f()

ST-Link Debugger t1: 0.00000000 sec L:93 C:1 CAP NUM SCRL OVR

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

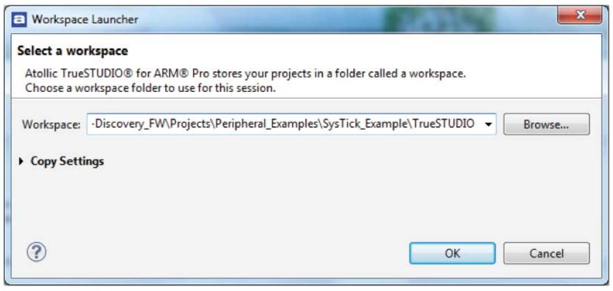
Figure 12. Run button



3.3 TrueSTUDIO toolchain

1. Open Atollic TrueSTUDIO for Arm microcontrollers. The program launches and asks for the Workspace location.

Figure 13. TrueSTUDIO workspace launcher dialog box



Workspace Launcher

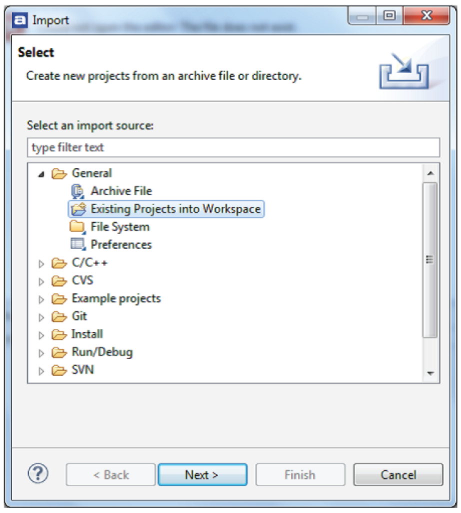
Select a workspace

Atollic TrueSTUDIO® for ARM® Pro stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: -Discovery_FW\Projects\Peripheral_Examples\SysTick_Example\TrueSTUDIO

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

Figure 14. Atollic TrueSTUDIO import source select dialog box



Import

Select

Create new projects from an archive file or directory.

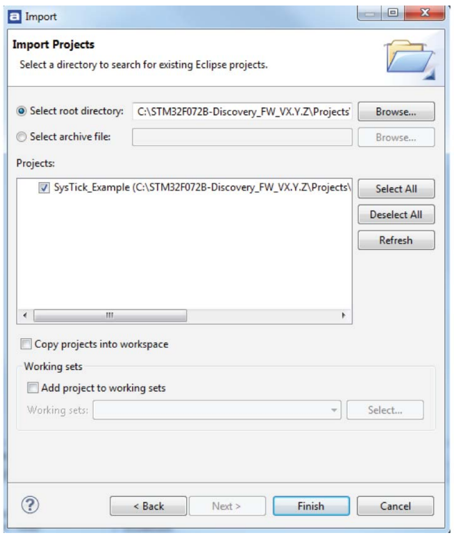
Select an import source:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- C/C++
- CVS
- Example projects
- Git
- Install
- Run/Debug
- SVN

5. Click **Select root directory**, browse to the TrueSTUDIO workspace folder.

Figure 15. Atollic TrueSTUDIO import projects dialog box



Import

Import Projects

Select a directory to search for existing Eclipse projects.

[✓ Select root directory:] C:\STM32F072B-Discovery_FW_VX.Y.Z\Projects\

Select archive file:

Projects:

[✓] SysTick_Example (C:\STM32F072B-Discovery_FW_VX.Y.Z\Projects\

[ ] Copy projects into workspace

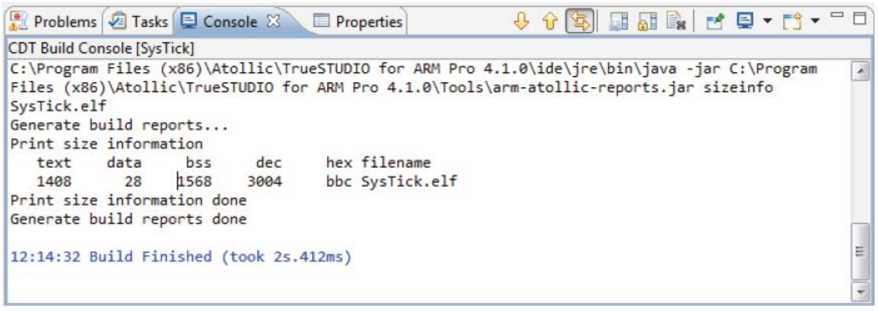
Working sets

[ ] Add project to working sets

Working sets:

6. In the **Projects** panel, select the project and click **Finish**.
7. In the **Project Explorer**, select the project, open the **Project** menu, and click **Build Project.**
8. If the project is successfully compiled, the following messages will be displayed on the Console window.

Figure 16. TrueSTUDIO project successfully compiled



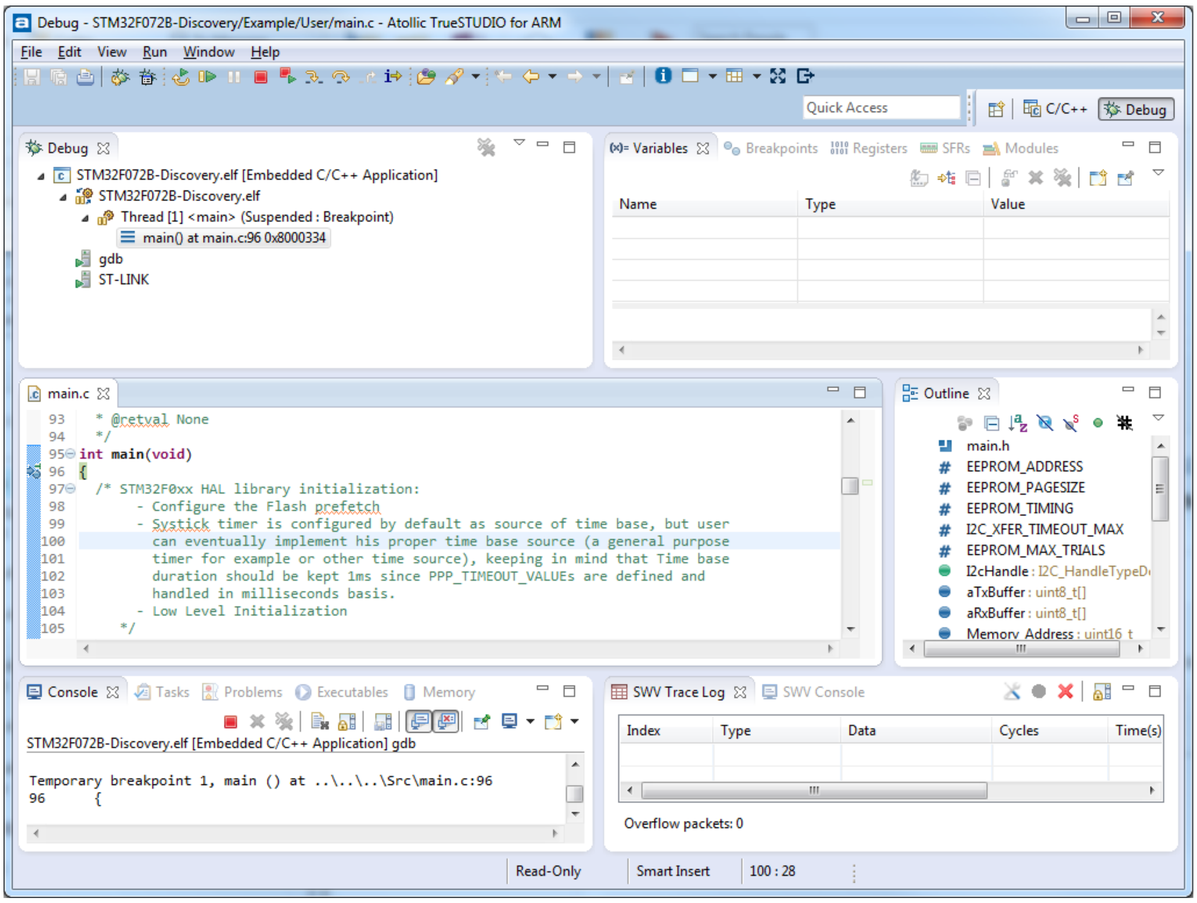
```
CDT Build Console [SysTick]
C:\Program Files (x86)\Atollic\TrueSTUDIO for ARM Pro 4.1.0\ide\jre\bin\java -jar C:\Program
Files (x86)\Atollic\TrueSTUDIO for ARM Pro 4.1.0\Tools\arm-atollic-reports.jar sizeinfo
SysTick.elf
Generate build reports...
Print size information
   text    data     bss     dec     hex filename
   1408      28    1568    3004     bbc SysTick.elf
Print size information done
Generate build reports done

12:14:32 Build Finished (took 2s.412ms)
```

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

- For 'Include directories' **C Compiler>Directories>Include path**
- For pre-processor defines **C Compiler>Symbols> Defined symbols**

9. To debug and run the application, the user should select the project In the Project Explorer and press F11 to start a debug session (see *Figure 17*).

Figure 17. TrueSTUDIO debug window



```
STM32F072B-Discovery.elf [Embedded C/C++ Application]
  STM32F072B-Discovery.elf
    Thread [1] <main> (Suspended : Breakpoint)
      main() at main.c:96 0x8000334
  gdb
  ST-LINK
```

```
  * @retval None
  */
int main(void)
{
  /* STM32F0xx HAL library initialization:
       - Configure the Flash prefetch
       - Systick timer is configured by default as source of time base, but user
         can eventually implement his proper time base source (a general purpose
         timer for example or other time source), keeping in mind that Time base
         duration should be kept 1ms since PPP_TIMEOUT_VALUEs are defined and
         handled in milliseconds basis.
       - Low Level Initialization
     */
```

```
main.h
EEPROM_ADDRESS
EEPROM_PAGESIZE
EEPROM_TIMING
I2C_XFER_TIMEOUT_MAX
EEPROM_MAX_TRIALS
I2cHandle : I2C_HandleTypeD
aTxBuffer : uint8_t[]
aRxBuffer : uint8_t[]
Memory_Address : uint16_t
```

```
STM32F072B-Discovery.elf [Embedded C/C++ Application] gdb

Temporary breakpoint 1, main () at ..\..\..\Src\main.c:96
96	{
```

```
Overflow packets: 0
```

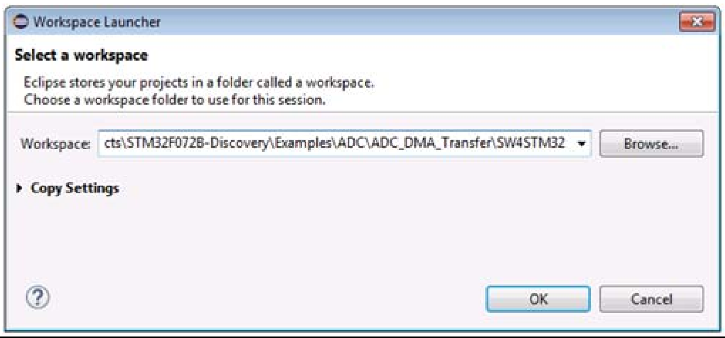
The debugger in the Atollic TrueSTUDIO can be used to debug source code at C and assembly levels, set breakpoints, monitor individual variables and watch events during the code execution.

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

3.4 SW4STM32 toolchain

1. Open the AC6 SW4STM32 for Arm microcontrollers. The program launches and prompts for the Workspace location.

Figure 18. SW4STM32 workspace launcher dialog box



Workspace Launcher

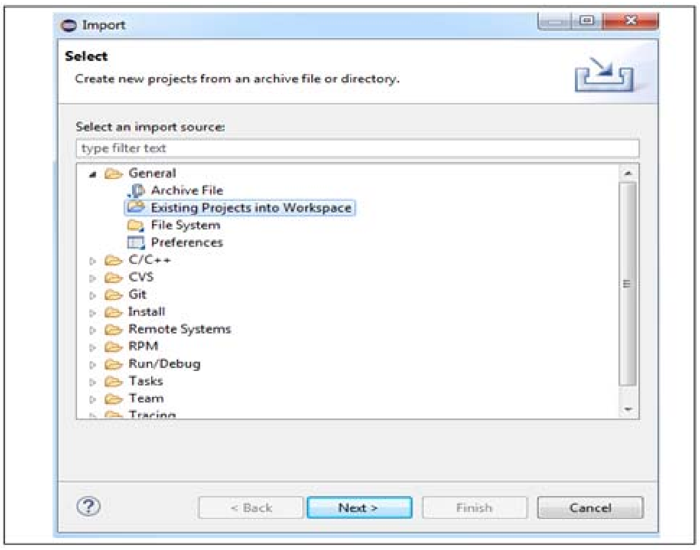
Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: cts\STM32F072B-Discovery\Examples\ADC\ADC_DMA_Transfer\SW4STM32

2. Browse to select a SW4STM32 workspace of either an example or demonstration or template workspace file and click **OK** to load it.
3. To load an existing project in the selected workspace, select **Import** from the **File** menu to display the **Import** dialog box.
4. In the **Import** window, open **General**, select **Existing Projects** into Workspace and click **Next**.

Figure 19. SW4STM32 import source select dialog box



Import

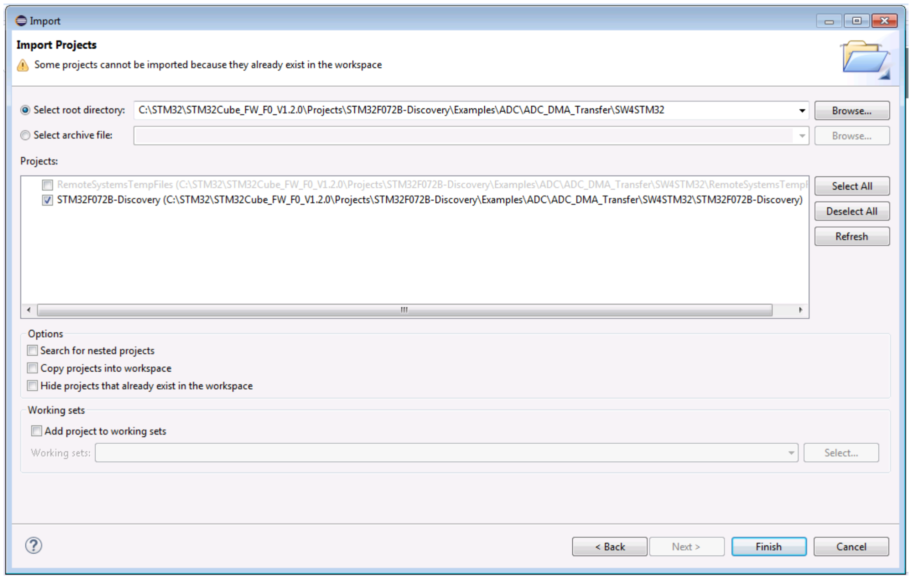
Select

Create new projects from an archive file or directory.

Select an import source:

5. Click **Select root directory**, browse to the SW4STM32 workspace folder.

Figure 20. SW4STM32 import projects dialog box



Import

Import Projects

Some projects cannot be imported because they already exist in the workspace

[✓ Select root directory:] C:\STM32\STM32Cube_FW_F0_V1.2.0\Projects\STM32F072B-Discovery\Examples\ADC\ADC_DMA_Transfer\SW4STM32

Select archive file:

Projects:

- [ ] RemoteSystemsTempFiles (C:\STM32\STM32Cube_FW_F0_V1.2.0\Projects\STM32F072B-Discovery\Examples\ADC\ADC_DMA_Transfer\SW4STM32\RemoteSystemsTempl
- [✓] STM32F072B-Discovery (C:\STM32\STM32Cube_FW_F0_V1.2.0\Projects\STM32F072B-Discovery\Examples\ADC\ADC_DMA_Transfer\SW4STM32\STM32F072B-Discovery)

Options

- [ ] Search for nested projects
- [ ] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

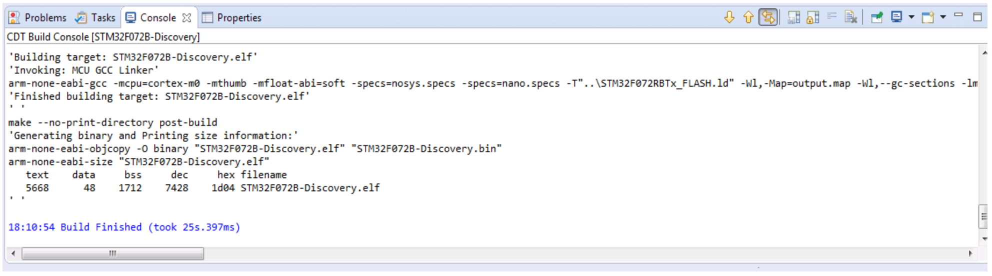
Working sets

- [ ] Add project to working sets

Working sets:

6. In the **Projects** panel, select the project and click **Finish**.
7. In the **Project Explorer**, select the project, open the **Project** menu, and click **Build Project**.
8. If the project is successfully compiled, the following messages display on the Console window.

Figure 21. SW4STM32 project successfully compiled



CDT Build Console [STM32F072B-Discovery]

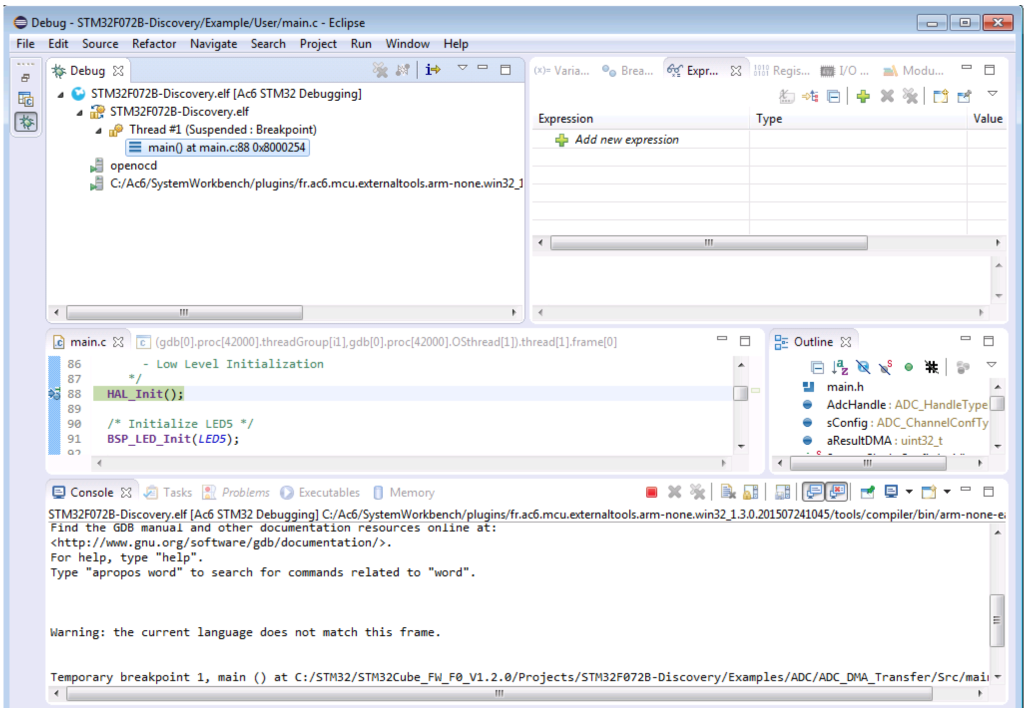
```
'Building target: STM32F072B-Discovery.elf'
'Invoking: MCU GCC Linker'
arm-none-eabi-gcc -mcpu=cortex-m0 -mthumb -mfloat-abi=soft -specs=nosys.specs -specs=nano.specs -T"..\STM32F072RBTx_FLASH.ld" -Wl,-Map=output.map -Wl,--gc-sections -lm
'Finished building target: STM32F072B-Discovery.elf'
' '
make --no-print-directory post-build
'Generating binary and Printing size information:'
arm-none-eabi-objcopy -O binary "STM32F072B-Discovery.elf" "STM32F072B-Discovery.bin"
arm-none-eabi-size "STM32F072B-Discovery.elf"
   text    data     bss     dec     hex filename
   5668      48    1712    7428    1d04 STM32F072B-Discovery.elf
' '

18:10:54 Build Finished (took 25s.397ms)
```

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

- For Include directories **C Compiler>Directories>Include path**
- For pre-processor defines **C Compiler>Symbols> Defined symbols**

9. To debug and run the application, select the project In the **Project Explorer** and press F11 to start a debug session. See *Figure 22*.

Figure 22. SW4STM32 debug window


Debug - STM32F072B-Discovery/Example/User/main.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
(x)= Varia... Brea... Expr... Regis... I/O ... Modu...
Expression Type Value
Add new expression
main.c
Outline
Console Tasks Problems Executables Memory

```
STM32F072B-Discovery.elf [Ac6 STM32 Debugging]
  STM32F072B-Discovery.elf
    Thread #1 (Suspended : Breakpoint)
      main() at main.c:88 0x8000254
  openocd
  C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.arm-none.win32_1
```

```
(gdb[0].proc[42000].threadGroup[i1],gdb[0].proc[42000].OSthread[1]).thread[1].frame[0]
```

```c
      - Low Level Initialization
    */
  HAL_Init();

  /* Initialize LED5 */
  BSP_LED_Init(LED5);
```

```
main.h
AdcHandle : ADC_HandleType
sConfig : ADC_ChannelConfTy
aResultDMA : uint32_t
```

```
STM32F072B-Discovery.elf [Ac6 STM32 Debugging] C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.arm-none.win32_1.3.0.201507241045/tools/compiler/bin/arm-none-e
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".


Warning: the current language does not match this frame.


Temporary breakpoint 1, main () at C:/STM32/STM32Cube_FW_F0_V1.2.0/Projects/STM32F072B-Discovery/Examples/ADC/ADC_DMA_Transfer/Src/mai
```

The debugger in the AC6 SW4STM32 can be used to debug source code at the C and assembly levels, to set breakpoints, to monitor individual variables and to watch events during the code execution.

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

4 Revision history

Table 3. Document revision history

Date	Revision	Changes
20-May-2016	1	Initial release.
25-Feb-2019	2	Added *Table 1: Applicable products*. Added *Section 1: General information*.