UM1969
User manual

Getting started with STM32F746G discovery software development tools

Introduction

This document describes the software environment recommendations, required to build an application using the STM32F746 discovery kit (32F746GDISCOVERY).

The document provides guidelines to user about how to build and run a simple example and how to create and build his own application. It has the following structure:

- The first chapter presents software and hardware requirements (some toolchains supporting the STM32 families, ST-LINK/V2-1 installation and firmware package presentation).
- The second chapter provides step by step guideline on how to execute and debug an application example using the following toolchains:
  - IAR Embedded Workbench® for ARM® (EWARM) by IAR Systems®
  - Microcontroller development kit for ARM® (MDK-ARM) by Keil®
  - System Workbench for STM32
  - SW4STM32® by AC6

Although this user manual does not cover all the topics relevant to software development environment, it demonstrates the first basic steps necessary to get started with the compilers/debuggers and includes references for complementary information.

Contents

List of tables

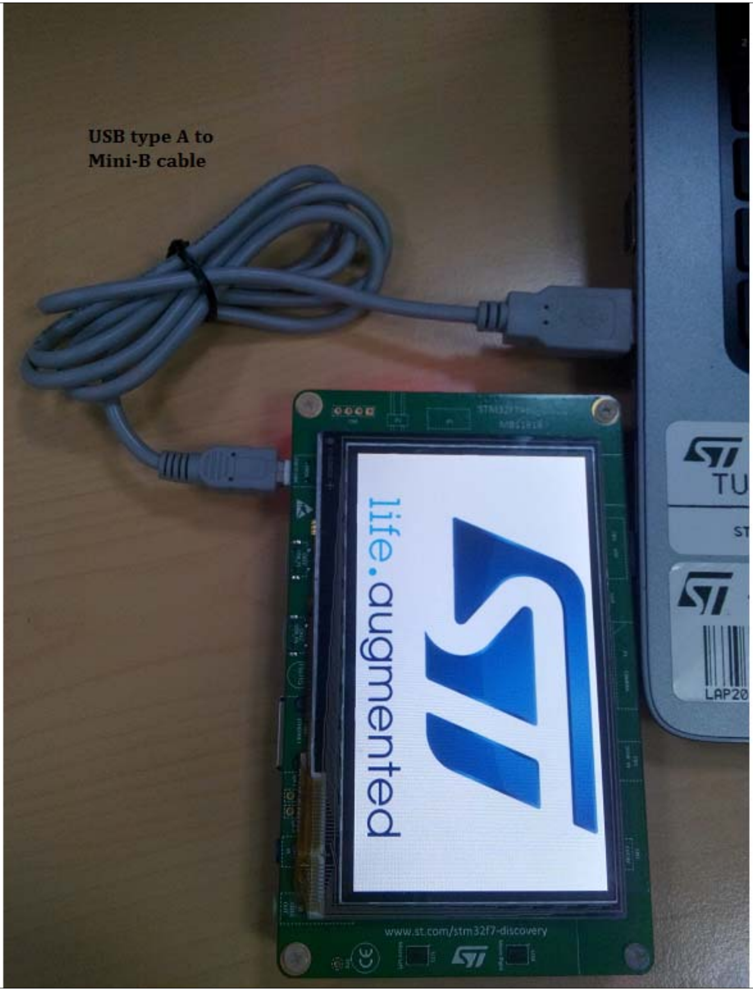
List of figures

1 System requirements

Before running an application, the user should:

1. Install his preferred Integrated Development Environment (IDE).
2. Install the ST-LINK V2-1 driver from the *www.st.com* web page.
3. Download the STM32F746G discovery firmware from the *www.st.com* web page.
4. Establish the USB connection with the STM32F746G discovery board as shown in *Figure 1*.

Figure 1. Hardware environment



The above steps will be detailed in the following sections.

To run and develop any firmware application on your STM32F746G discovery board, the minimum requirements are as follows:

- Windows OS (XP, 7, 8)
- "USB type A to Mini-B" cable, used to power the board (through USB connector CN14) from host PC and connect to the embedded ST-LINK/V2-1 for debugging and programming

2 IDEs supporting STM32 families

STM32 families of 32-bit ARM® Cortex® -M core-based microcontrollers are supported by a complete range of software tools. It encompasses traditional integrated development environments IDEs with C/C++ compilers and debuggers from major 3rd-parties (free versions up to 64 KBytes of code, depending on 3rd-parties), completed with innovative tools from STMicroelectronics.

The following table regroups general information about the three integrated development environments, as well as the version supporting the STM32F746G product.

Table 1. Toolchains supporting STM32F746G discovery

Toolchain	Company	Compiler	Version	Download link[1]
EWARM	IAR Systems®	IAR C/C++	7.40 and later	www.iar.com 30-day evaluation edition KickStart edition(16-KByte code size limitation for Cortex M0)
MDK-ARM	Keil®	ARMCC	5.xx and later with *Keil.STM32F7xx_DFP. 1.0.0.pack*	www.keil.com MDK-Lite (16-KByte code size limitation)
SW4STM32	AC6	GNU C	2.1.0 and later	http://test.openstm32.org 32Ko Limitation (8Ko on Cortex-M0 and Cortex-M1 30-day Professional version (Trial)

1. Registration before downloading is required.

3 ST-LINK/V2-1 installation

The STM32F746G discovery board includes an embedded ST-LINK/V2-1 debug tool interface. The interface needs an ST-LINK/V2 dedicated USB driver to be installed. This driver is available from the *www.st.com* web page and it is supported by the 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 hard disk of the local PC.

  After installing EWARM, 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.
- Keil Microcontroller Development Kit (MDK-ARM) toolchain.

  The toolchain is installed by default in the C:\Keil directory on the hard disk of the local PC; the installer creates a ARM® Keil® µVision®4 shortcut in the start menu.

  When connecting the ST-LINK/V2 tool, the PC detects new hardware and prompts user to install the ST-Link_V2_USB driver. The “Found New Hardware” wizard displays and guides the user through the steps, required to install the driver from the recommended location.
- AC6 System Workbench for STM32 (SW4STM32).

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

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

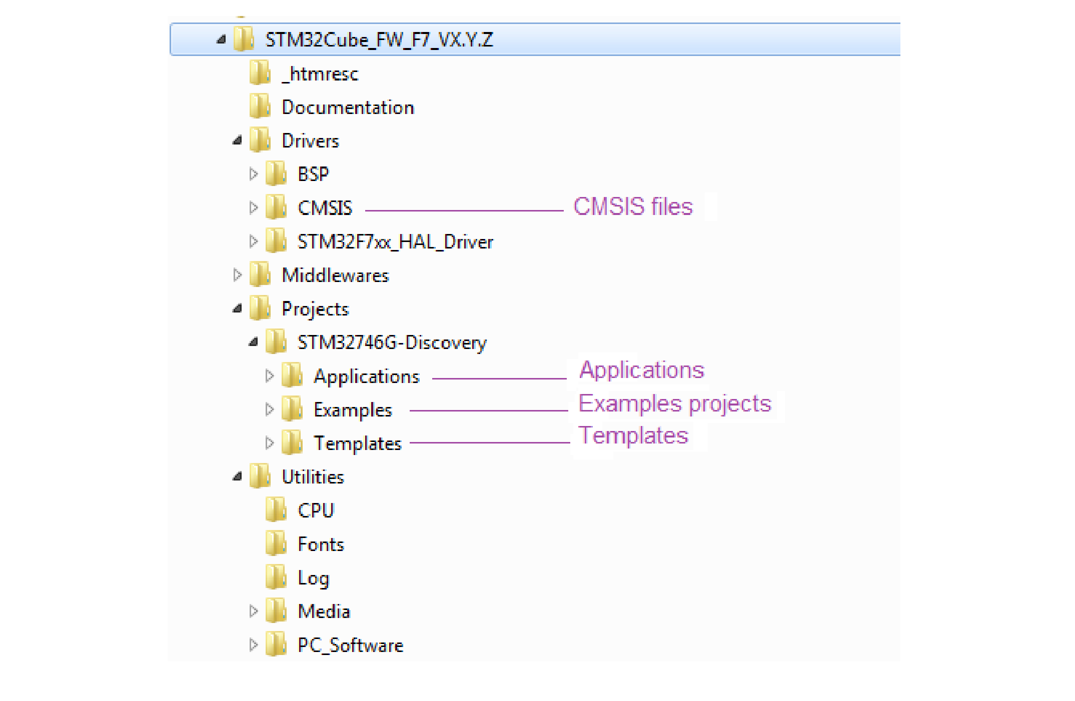
For complementary information on the firmware package content and the STM32F746G discovery requirements, refer to: *STM32CubeF7 demonstration platform* (UM1906 user manual).

Note: The embedded ST-LINK/V2-1 supports only the SWD interface for STM32 devices.

4 Firmware package

The STM32F746G discovery firmware applications, demonstration and examples are provided in one single.zip file. The extraction of the zip file generates a folder named “STM32Cube_FW_F4_VX.Y.Z”, which contains the subfolders shown in *Figure 2*.

Figure 2. Package contents



Template: pre-configured project with empty main function to be customized. This is helpful to start creating your own application.

Peripheral examples: including set of examples for each STM32F746 peripheral ready to be run.

Applications: including set of applications for each STM32F746 peripheral ready to be run.

Demonstrations: including demonstration firmware for STM32F746 boards ready to be run.

5 Compiling/linking and executing firmware using software toolchain

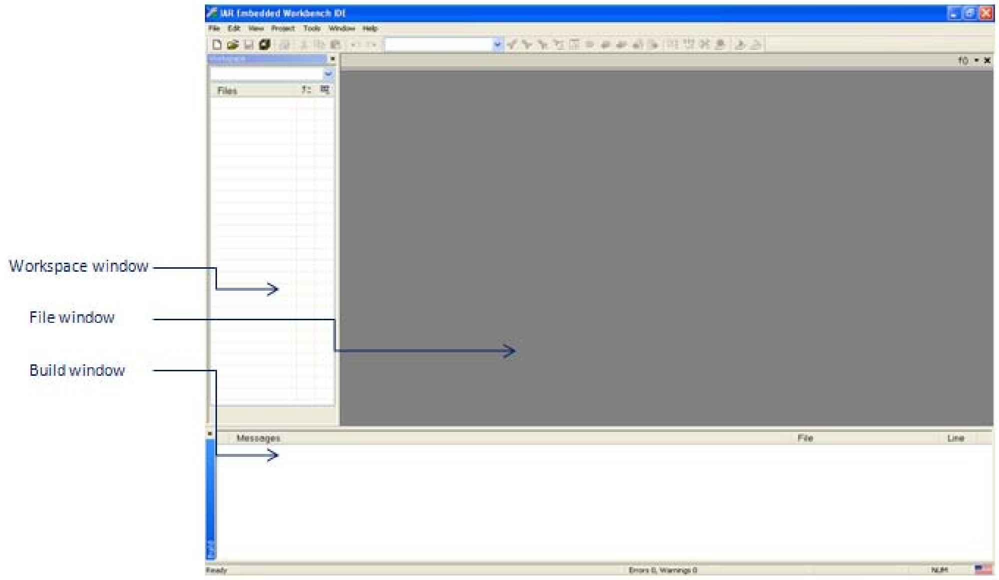
Steps below can be applied to an already existing example, demonstration or template project available at STM32F746G discovery package firmware available at *www.st.com* web site.

First of all, the user must read the readme.txt file, which contains a description of the firmware and hardware/software requirements.

5.1 EWARM toolchain

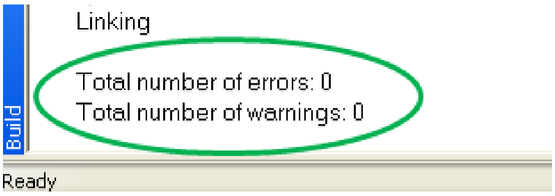
1. Open the IAR Embedded Workbench® for ARM® (EWARM). *Figure 3* shows the names of the windows, to which this document refers.

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), simply go through the project options:

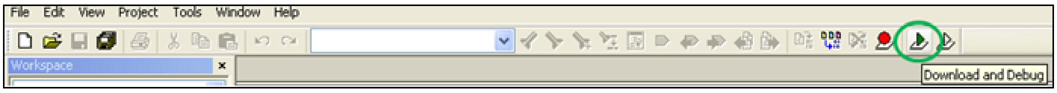
- For Include directories

  Project>Options…>C/C++ compiler>
- For pre-processor defines

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

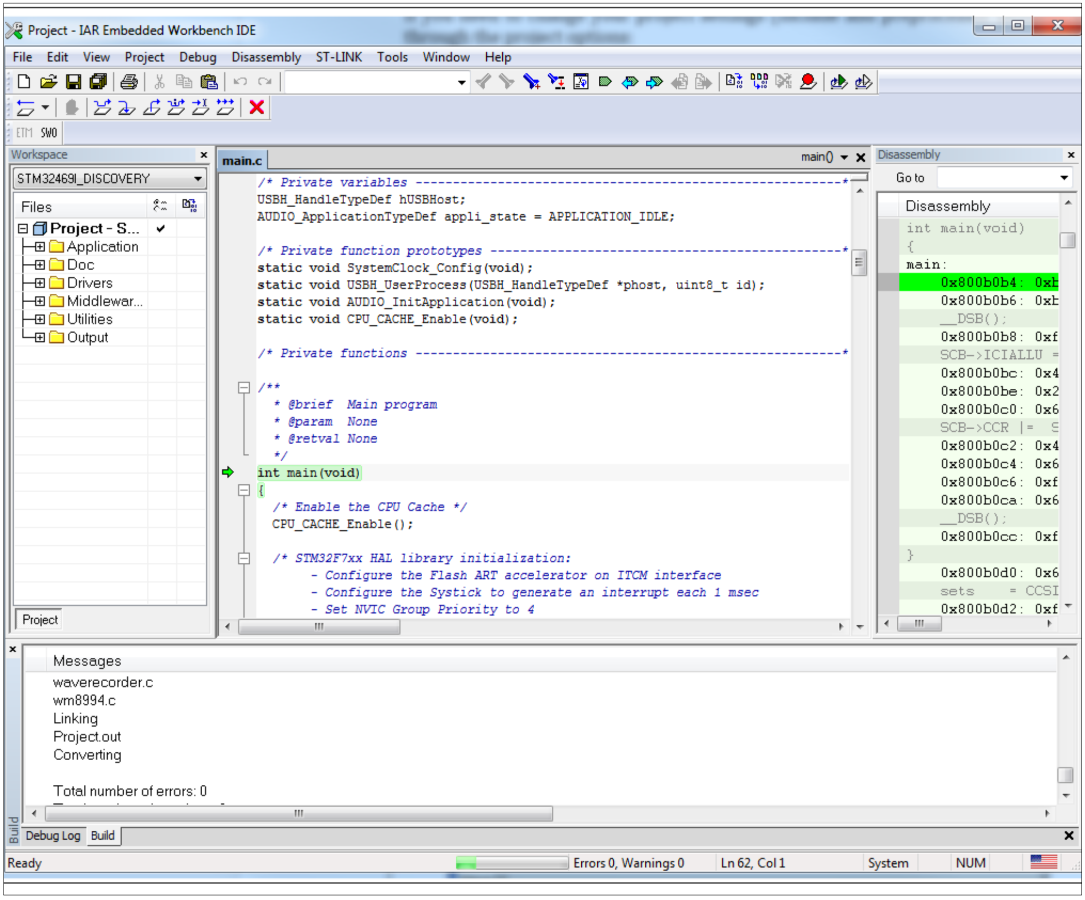
5. In the IAR Embedded Workbench® IDE, from the Project menu, select Download and Debug or, alternatively, click the Download and Debug button in the toolbar, to program the Flash memory and begin debugging (see *Figure 5*).

Figure 5. Download and debug button



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

Figure 6. IAR embedded workbench debugger screen



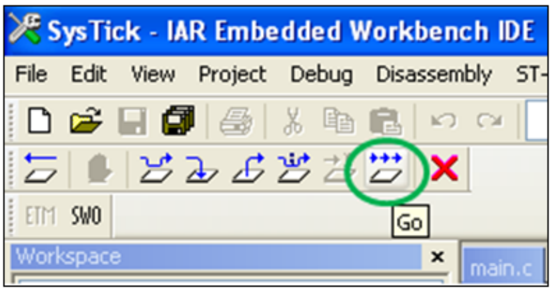
```
/* Private variables ---------------------------------------------------------*
USBH_HandleTypeDef hUSBHost;
AUDIO_ApplicationTypeDef appli_state = APPLICATION_IDLE;

/* Private function prototypes -----------------------------------------------*
static void SystemClock_Config(void);
static void USBH_UserProcess(USBH_HandleTypeDef *phost, uint8_t id);
static void AUDIO_InitApplication(void);
static void CPU_CACHE_Enable(void);

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

/**
  * @brief  Main program
  * @param  None
  * @retval None
  */
int main(void)
{
  /* Enable the CPU Cache */
  CPU_CACHE_Enable();

  /* STM32F7xx HAL library initialization:
       - Configure the Flash ART accelerator on ITCM interface
       - Configure the Systick to generate an interrupt each 1 msec
       - Set NVIC Group Priority to 4
```

```
Disassembly
int main(void)
{
main:
     0x800b0b4:  0xb
     0x800b0b6:  0xb
     __DSB();
     0x800b0b8:  0xf
     SCB->ICIALLU =
     0x800b0bc:  0x4
     0x800b0be:  0x2
     0x800b0c0:  0x6
     SCB->CCR |=  S
     0x800b0c2:  0x4
     0x800b0c4:  0x6
     0x800b0c6:  0xf
     0x800b0ca:  0x6
     __DSB();
     0x800b0cc:  0xf
}
     0x800b0d0:  0x6
     sets      = CCSI
     0x800b0d2:  0xf
```

```
Messages
waverecorder.c
wm8994.c
Linking
Project.out
Converting

Total number of errors: 0
```

To run the application, from the Debug menu, select Go. Alternatively, click the Go button in the toolbar to run the application (see *Figure 7*).

Figure 7. IAR Go button

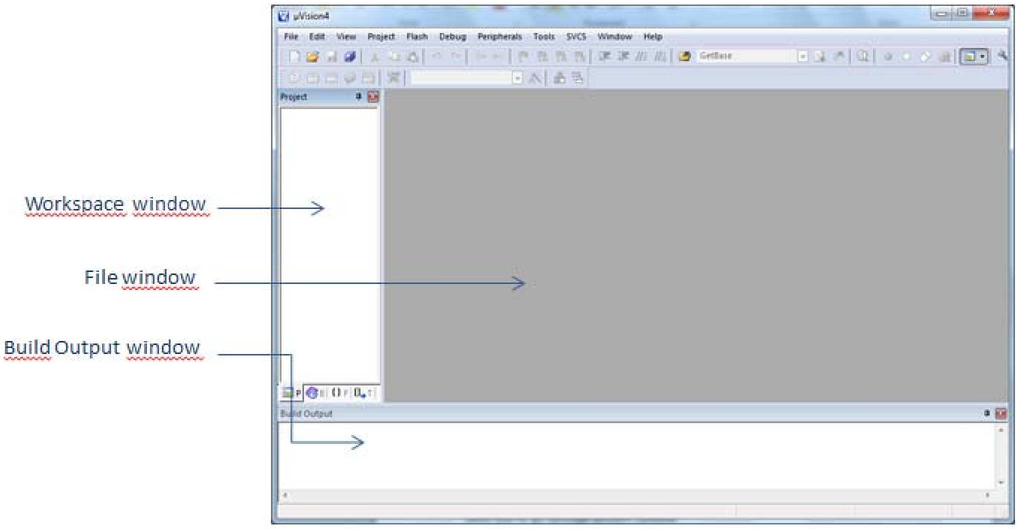

SysTick - IAR Embedded Workbench IDE
File Edit View Project Debug Disassembly ST-
ETM SWO
Go
Workspace
main.c

5.2 MDK-ARM toolchain

1. Open the Keil MDK-ARM Microcontroller Kit.

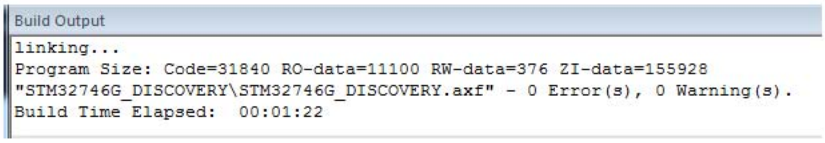
Figure 8 shows the basic names of the “Keil µVision4” windows to which this document refers.

Figure 8. µVision4 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 following window in *Figure 9* is displayed.

Figure 9. MDK-ARM project successfully compiled



```
linking...
Program Size: Code=31840 RO-data=11100 RW-data=376 ZI-data=155928
"STM32746G_DISCOVERY\STM32746G_DISCOVERY.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:01:22
```

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

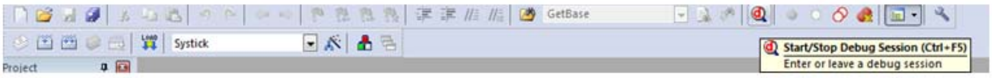
- For Include directories

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

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

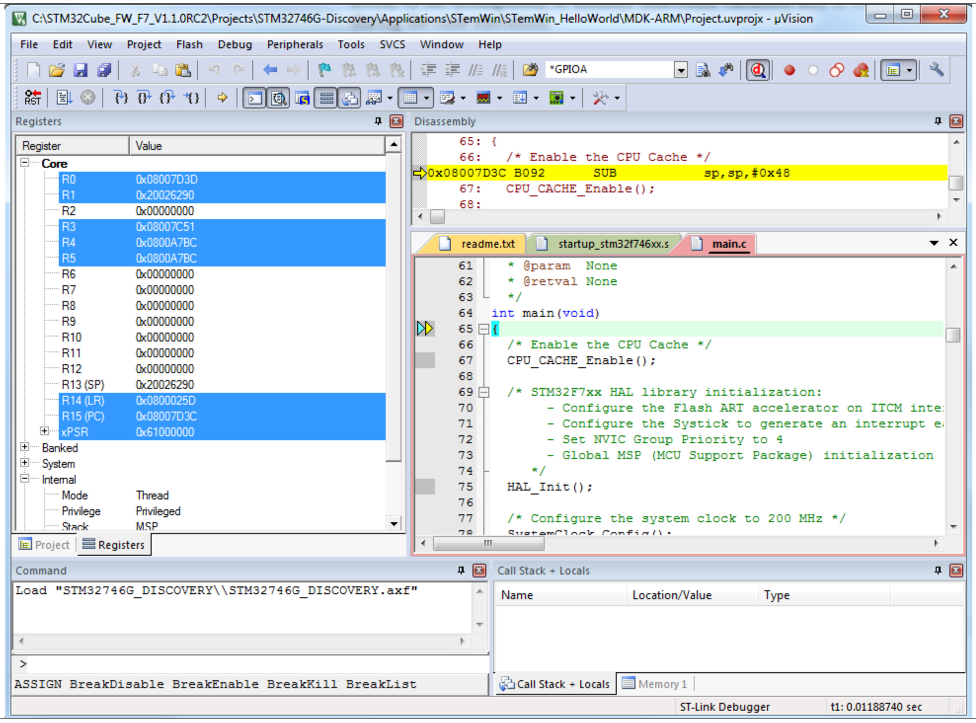
5. In the MDK-ARM IDE, from the Debug menu, select Start/Stop Debug Session or, alternatively, click the Start/Stop Debug Session button in the toolbar, to program the Flash memory and begin debugging (see *Figure 10*).

Figure 10. Start/Stop debug session button


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

6. The MDK-ARM debugger can be used to debug source code at C and assembly levels, to set breakpoints, to monitor individual variables and to watch events during the code execution (see *Figure 11*).

Figure 11. MDK-ARM debugger screen



C:\STM32Cube_FW_F7_V1.1.0RC2\Projects\STM32746G-Discovery\Applications\STemWin\STemWin_HelloWorld\MDK-ARM\Project.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
*GPIOA
Registers
Disassembly
readme.txt
startup_stm32f746xx.s
main.c
Project
Registers
Command
Call Stack + Locals
Memory 1
ST-Link Debugger
t1: 0.01188740 sec

Register	Value
Core	
R0	0x08007D3D
R1	0x20026290
R2	0x00000000
R3	0x08007C51
R4	0x0800A7BC
R5	0x0800A7BC
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20026290
R14 (LR)	0x0800025D
R15 (PC)	0x08007D3C
xPSR	0x61000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP

```
    65: {
    66:   /* Enable the CPU Cache */
0x08007D3C B092      SUB         sp,sp,#0x48
    67:   CPU_CACHE_Enable();
    68:
```

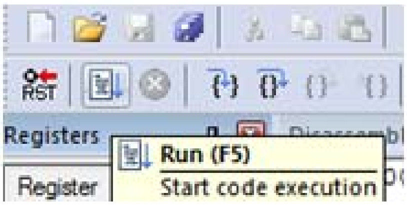
```
  * @param  None
  * @retval None
  */
int main(void)
{
  /* Enable the CPU Cache */
  CPU_CACHE_Enable();

  /* STM32F7xx HAL library initialization:
       - Configure the Flash ART accelerator on ITCM inte
       - Configure the Systick to generate an interrupt e
       - Set NVIC Group Priority to 4
       - Global MSP (MCU Support Package) initialization
     */
  HAL_Init();

  /* Configure the system clock to 200 MHz */
  SystemClock_Config();
```

```
Load "STM32746G_DISCOVERY\\STM32746G_DISCOVERY.axf"
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Location/Value	Type

To run the application, from the Debug menu, select Run. Alternatively, click the Run button in the toolbar. (see *Figure 12*).

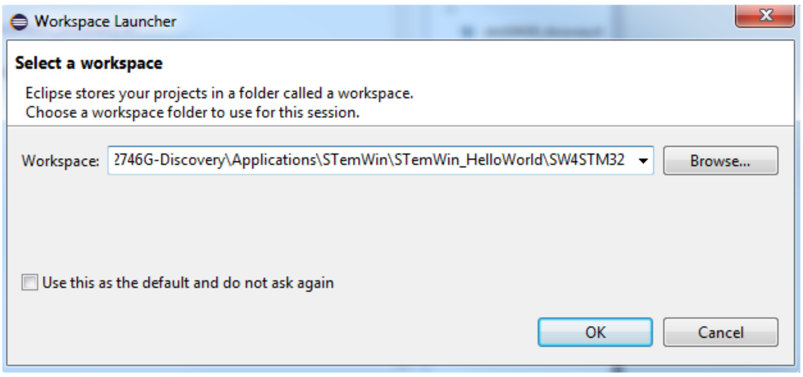
Figure 12. Run button



5.3 SW4STM32 toolchain

1. Open the AC6 SW4STM32® for ARM® products. The program launches and prompts for the workspace location (see *Figure 13*).

Figure 13. SW4STM32 workspace launcher dialog box



Select a workspace

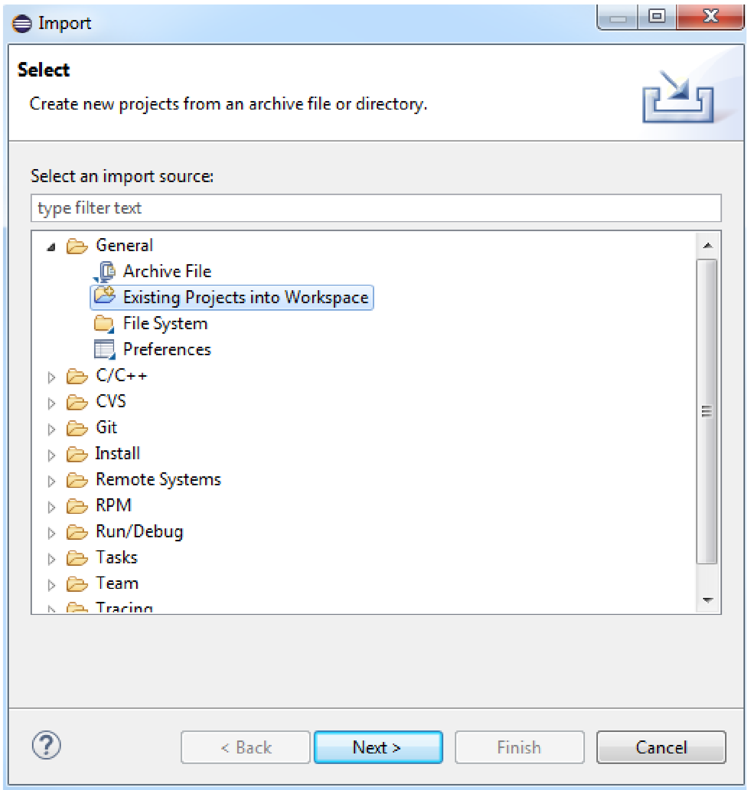
Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: 2746G-Discovery\Applications\STemWin\STemWin_HelloWorld\SW4STM32

□ Use this as the default and do not ask again

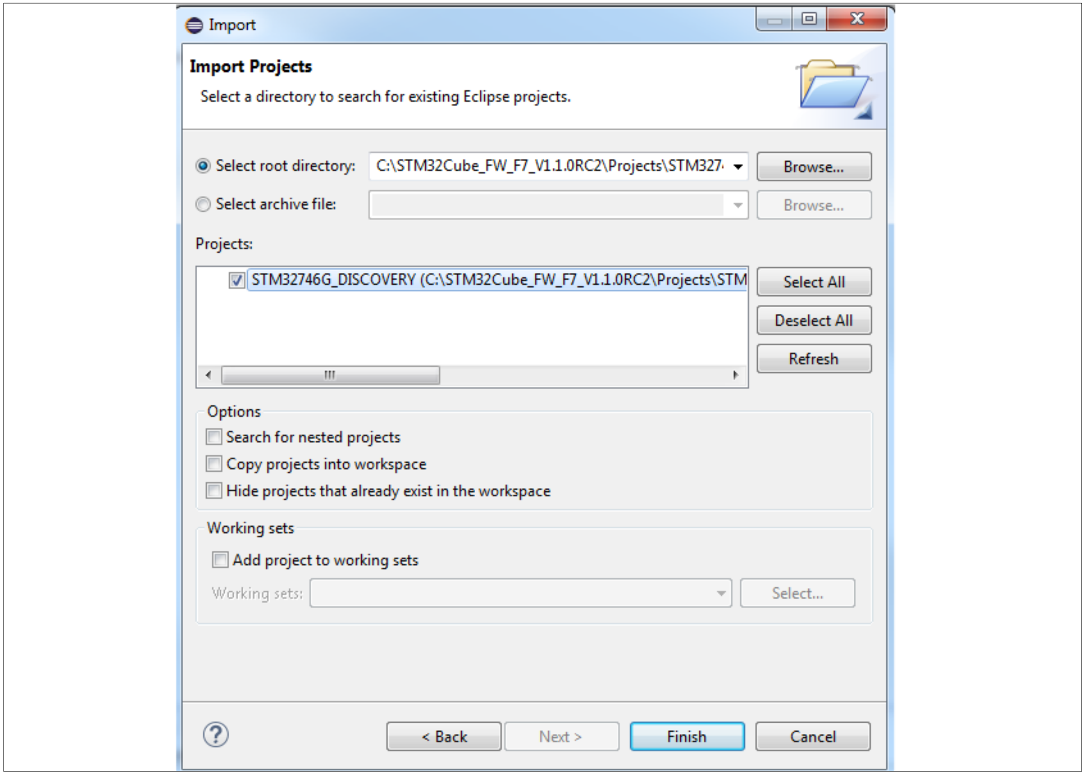
2. Browse to select a SW4STM32 workspace of either an example or demonstration or template workspace file and click OK to load it.
3. To load an existing project in the selected workspace, select Import from the File menu to display the Import dialog box.
4. In the Import window, open General, select Existing Projects into Workspace and click Next (see *Figure 14*).

Figure 14. SW4STM32 import source select dialog box



5. Click Select root directory, browse to the SW4STM32 workspace folder (see *Figure 15*).

Figure 15. SW4STM32 import projects dialog box



Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\STM32Cube_FW_F7_V1.1.0RC2\Projects\STM327·

○ Select archive file:

Projects:

☑ STM32746G_DISCOVERY (C:\STM32Cube_FW_F7_V1.1.0RC2\Projects\STM

Options

□ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

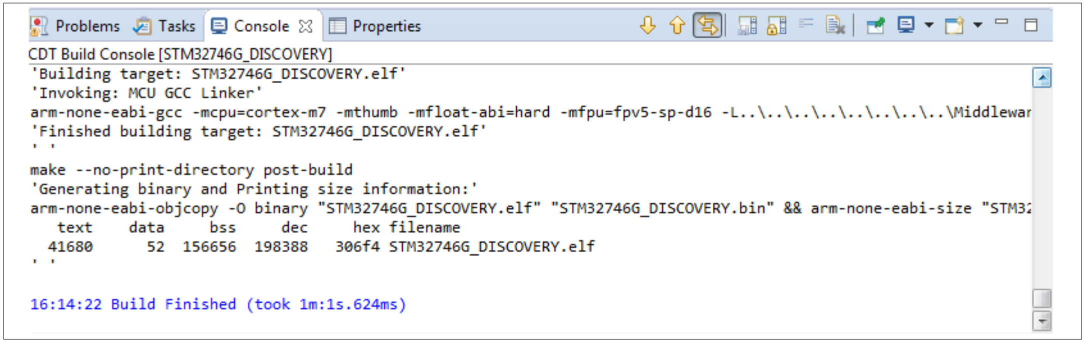
Working sets

□ 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 is displayed on the Console window (see *Figure 16*).

Figure 16. SW4STM32 project successfully compiled



```
CDT Build Console [STM32746G_DISCOVERY]
'Building target: STM32746G_DISCOVERY.elf'
'Invoking: MCU GCC Linker'
arm-none-eabi-gcc -mcpu=cortex-m7 -mthumb -mfloat-abi=hard -mfpu=fpv5-sp-d16 -L..\..\..\..\..\..\..\Middlewar
'Finished building target: STM32746G_DISCOVERY.elf'
' '
make --no-print-directory post-build
'Generating binary and Printing size information:'
arm-none-eabi-objcopy -O binary "STM32746G_DISCOVERY.elf" "STM32746G_DISCOVERY.bin" && arm-none-eabi-size "STM32
   text    data     bss     dec     hex filename
  41680      52  156656  198388   306f4 STM32746G_DISCOVERY.elf
' '

16:14:22 Build Finished (took 1m:1s.624ms)
```

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’

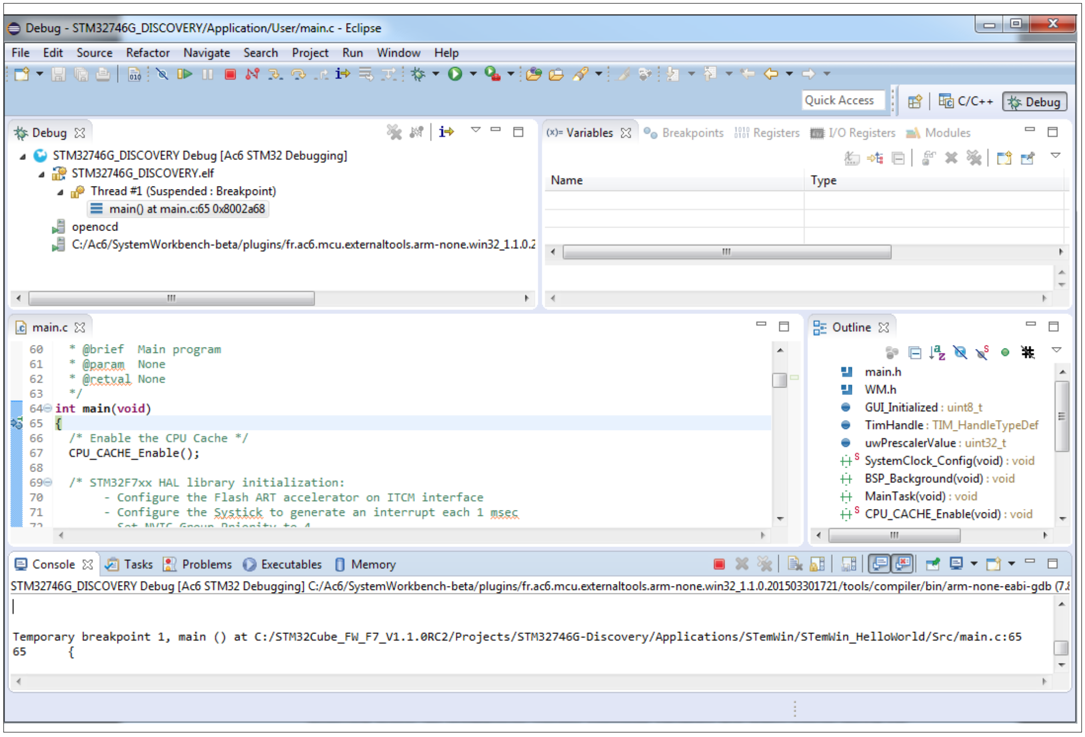
  C Compiler>Directories>Include path
- For pre-processor defines

  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.

In the Project Explorer, select the project and press F11 to start a debug session (see *Figure 17*).

Figure 17. SW4STM32 debug window



STM32746G_DISCOVERY Debug [Ac6 STM32 Debugging]
STM32746G_DISCOVERY.elf
Thread #1 (Suspended : Breakpoint)
main() at main.c:65 0x8002a68
openocd
C:/Ac6/SystemWorkbench-beta/plugins/fr.ac6.mcu.externaltools.arm-none.win32_1.1.0.2

```
  * @brief  Main program
  * @param  None
  * @retval None
  */
int main(void)
{
  /* Enable the CPU Cache */
  CPU_CACHE_Enable();

  /* STM32F7xx HAL library initialization:
       - Configure the Flash ART accelerator on ITCM interface
       - Configure the Systick to generate an interrupt each 1 msec
       - Set NVIC Group Priority to 4
```

main.h
WM.h
GUI_Initialized : uint8_t
TimHandle : TIM_HandleTypeDef
uwPrescalerValue : uint32_t
SystemClock_Config(void) : void
BSP_Background(void) : void
MainTask(void) : void
CPU_CACHE_Enable(void) : void

STM32746G_DISCOVERY Debug [Ac6 STM32 Debugging] C:/Ac6/SystemWorkbench-beta/plugins/fr.ac6.mcu.externaltools.arm-none.win32_1.1.0.201503301721/tools/compiler/bin/arm-none-eabi-gdb (7.8

```
Temporary breakpoint 1, main () at C:/STM32Cube_FW_F7_V1.1.0RC2/Projects/STM32746G-Discovery/Applications/STemWin/STemWin_HelloWorld/Src/main.c:65
65      {
```

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.

6 SW toolchains helpful references and links

The following *Table 2* provides useful references about integrated development environments described in this document:

Table 2. Links to software toolchains

Toolchain	Download link
EWARM	www.iar.com
MDK-ARM	www.keil.com
SW4STM32	www.ac6-tools.com

7 Revision history

Table 3. Document revision history

Date	Revision	Changes
24-Nov-2015	1	Initial release.