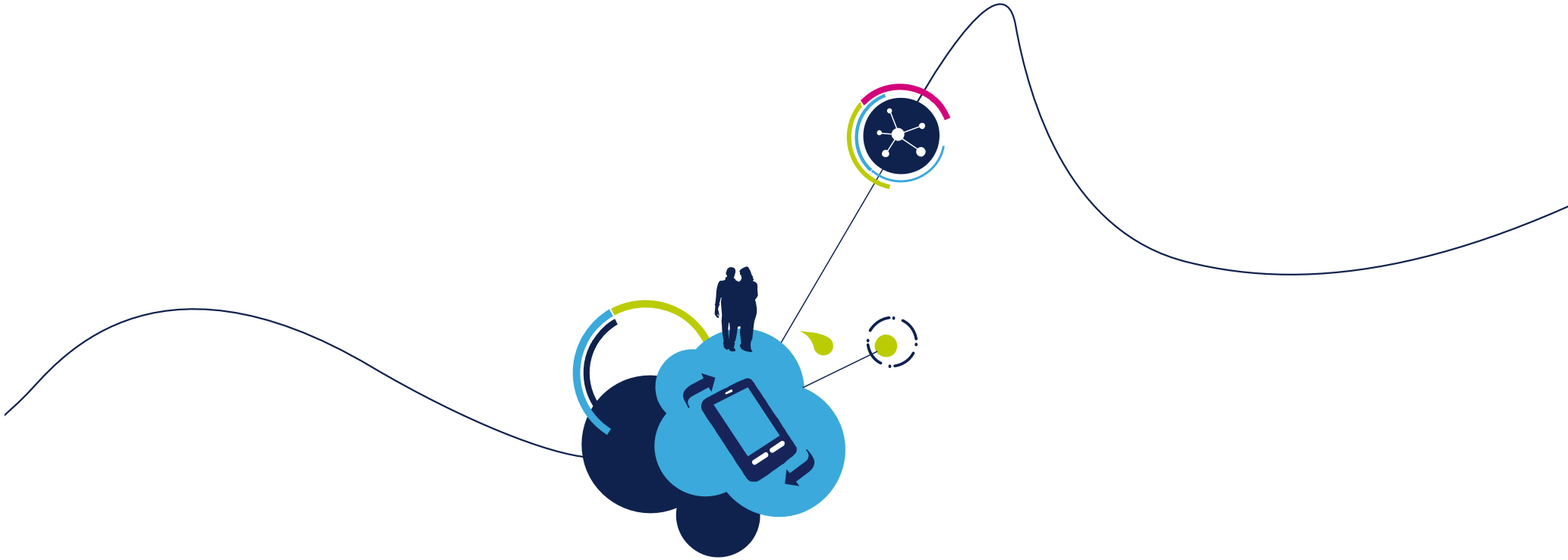
# STM32F3
# Hands-On Workshop

life.augmented

# Welcome – Hands-On



- Ensure you picked-up
  - USB Flash Drive with STM32F3 Discovery Kit Contents
  - USB Cable
  - STM32F3-Discovery Kit – will be provided after software is loaded



# Keil uVision IDE Installation

life.augmented

# Systems Check



- Everyone should have
  - A Windows ® Laptop (XP, Vista, or Windows 7)
  - USB Cable
  - USB Flash Drive
  - STM32F3-DISCOVERY kit:  provided during the software installation.

- Ready to begin?

Note: please do not attempt to plug in the STM32F3-Discovery Kit into your laptop until instructed to do so.

# Step #1 - File Installation

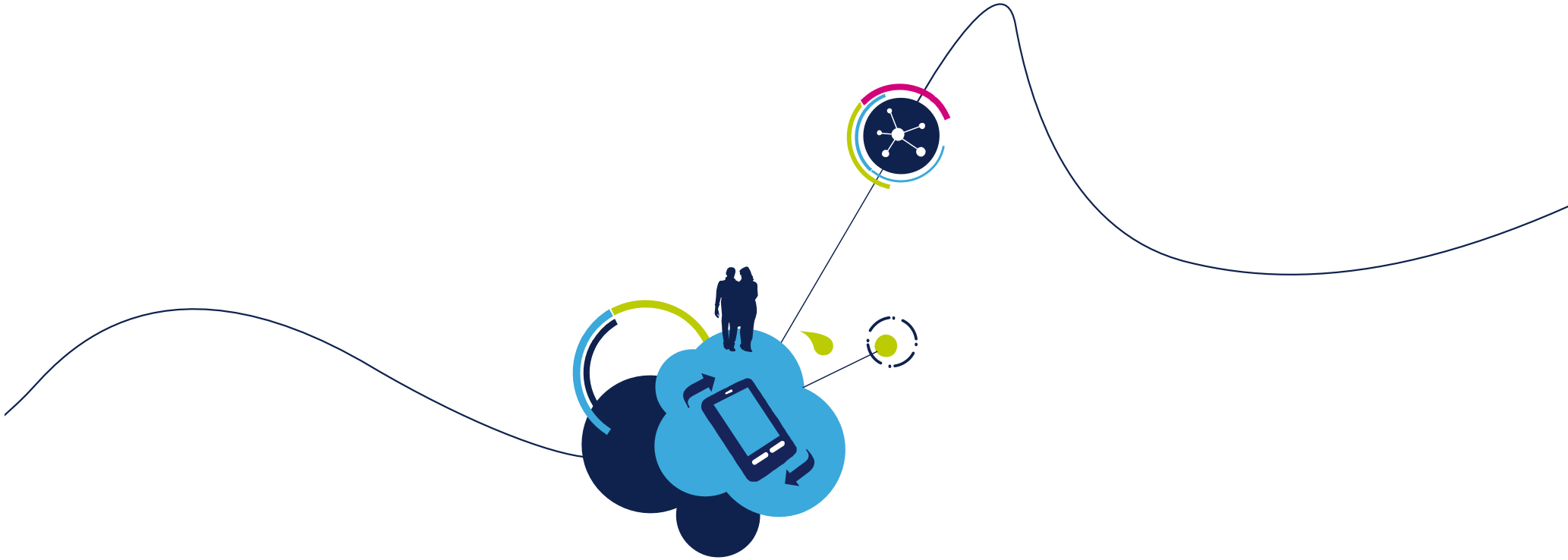


- Insert the USB Flash Drive into your Laptop
- Copy the folder **"…\STM32F3DISCOVERY_Kit"** on the USB flash drive to your root **"c:\"** folder
  - **C:\STM32F3DISCOVERY_Kit**
    - **Edit folder properties and remove 'Read-only' attribute for all sub-folders.**
- Open this directory and you will find the following:
  - Keil µVision v4.71 → IDE tool installation application and license file.
  - Docs →STM32F3 Datasheets, Programming Manual, Reference Manuals, Data Briefs, and The STM32F3 Discovery Board Manuals.
  - Library → STM32F3Discovery Firmware Library folder.
  - Utility → STM32F3 Clock Utility and ST-LINK Utility Application

# Step #2 - Install Keil µVision



- For this workshop, we will be using the evaluation version of the Microcontroller Development Kit from ARM. Some restrictions apply:
  - Program and debug up to 32 Kbytes of code
  - No disassembly listing
  - Some restriction on linkage usage
  - Limited base address usage
- Double-click on the file ***mdk.exe*** to begin installation. Please click-through the default options and accept the license agreement
- Ask for assistance if you have an issue



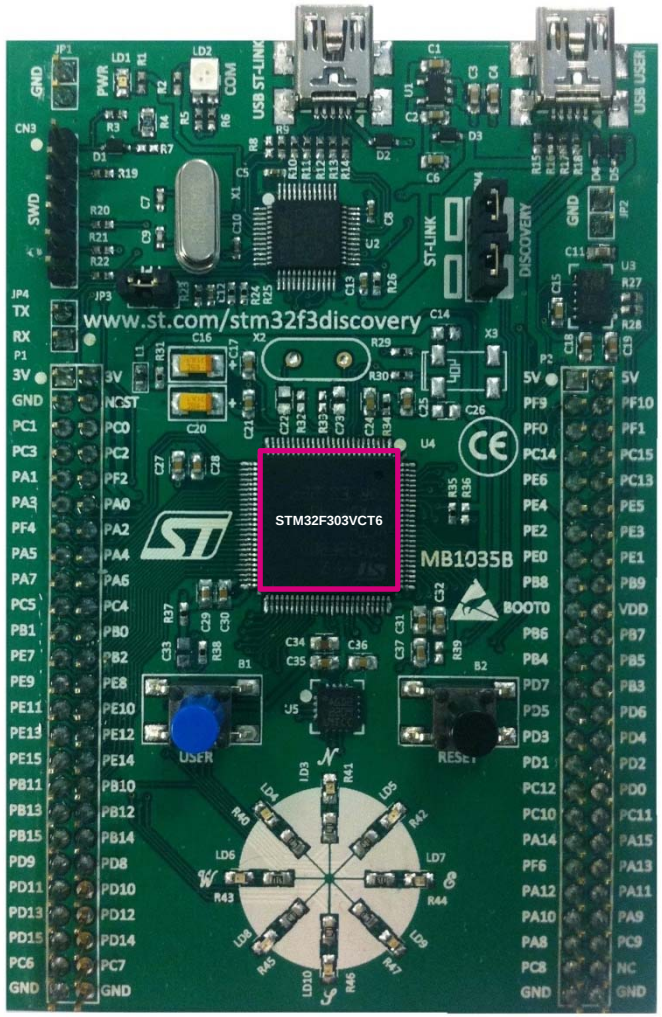
# Introducing the STM32F3Discovery Kit

life.augmented

# STM32F303VCT6



- 72 MHz Cortex-M4
- 100-pin LQFP
- 256 Kbytes Flash
- 40 Kbytes SRAM
- 8 Kbytes of CCM-SRAM




life.augmented

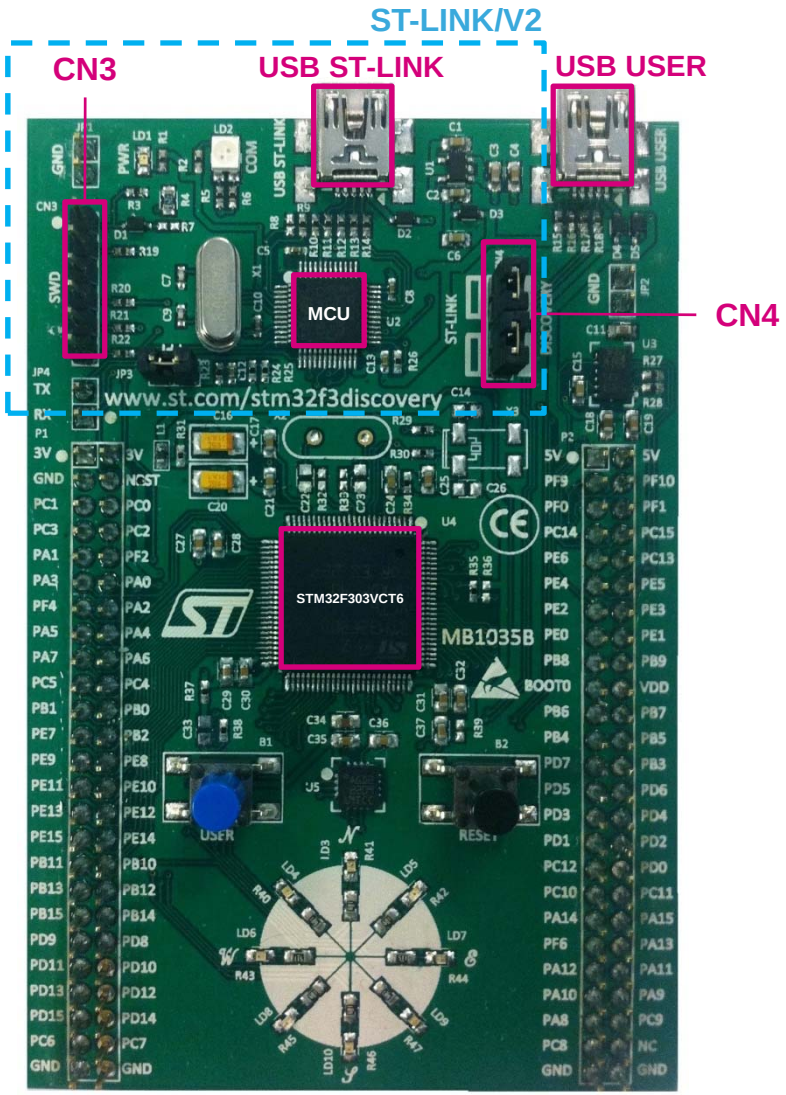
# Embedded ST-LINK/V2



- ST-LINK/V2 programming and debugging tool integrated on-board the kit (STM32F103C8T6)

- Can be used two different ways
  - Program and debug the MCU on the board
  - Program an MCU on another application board
  - Note: JTAG versus SWD configuration.

- Features
  - USB ST-LINK – USB Micro Type B
  - USB USER – USB Micro Type B (USB FS,2.0)
  - ST-LINK/V2 MCU (STM32F103)
  - 5V to 3V Regulator (USB power)
  - CN4 – MCU Program Jumper
  - CN3 – Application SWD connector

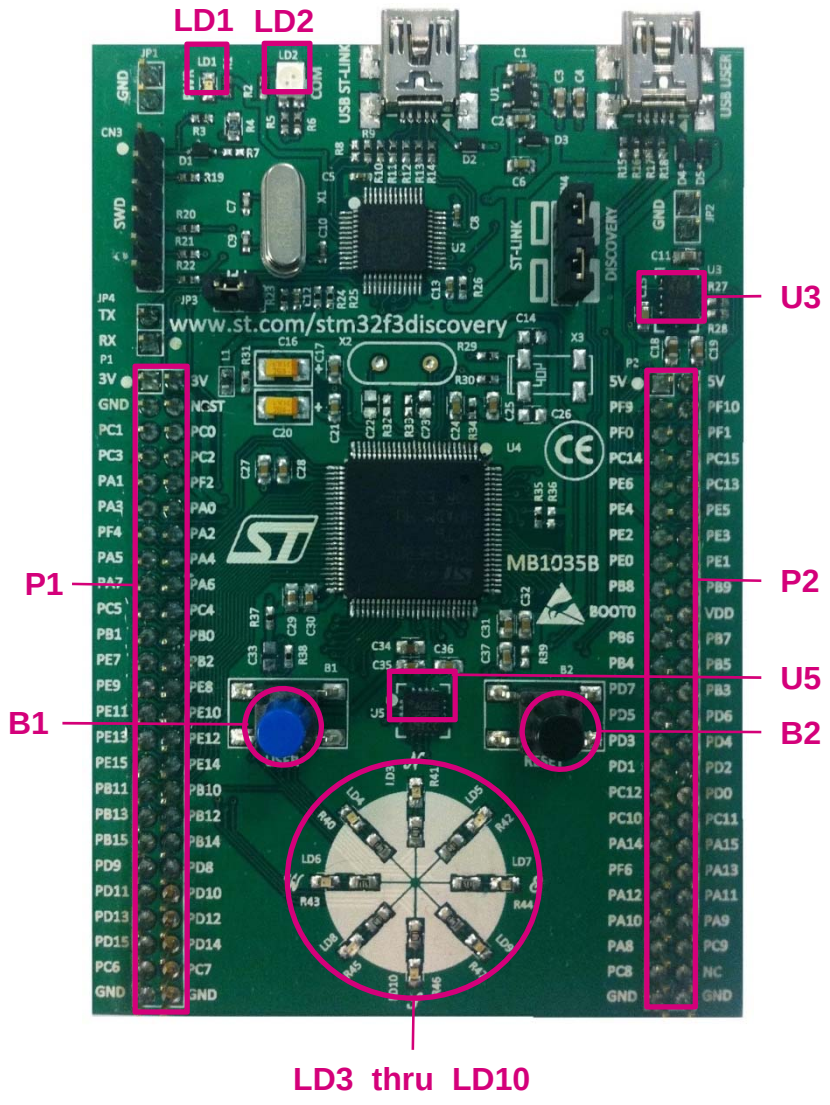



life.augmented

# LEDs/Push-Buttons/MEMs/ Extension Connector



- LEDS
  - LD1: Power indicator
  - LD2: ST-LINK Communication indicator
  - LD3 thru LD10: (PE8 thru PE15)
- Push-Buttons
  - B1: USER/Wake-up (PA0)
  - B2: RESET (NRST)
- Extension Connector
  - P1 and P2
  - All GPIOs are available for prototype
  - Includes 5V, 3V and GND pins
- MEMs Devices
  - U3: LSM303DLHC
  - U5: L3GD20

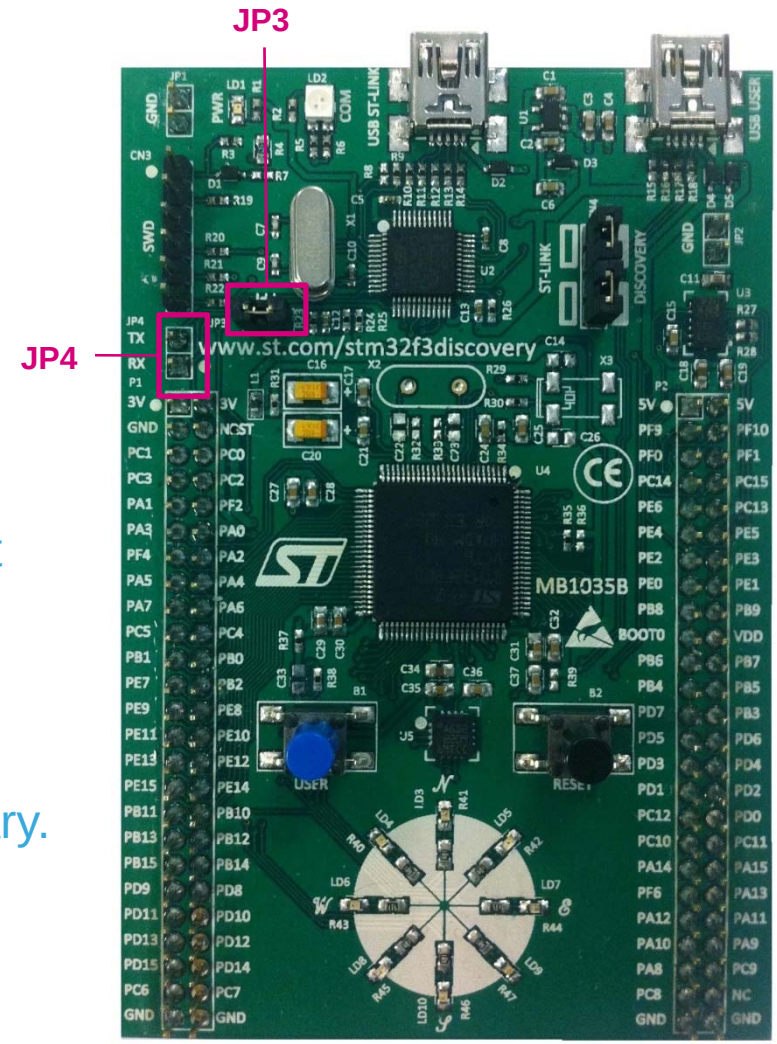



life.augmented

# Jumpers/User Manual/Firmware Library



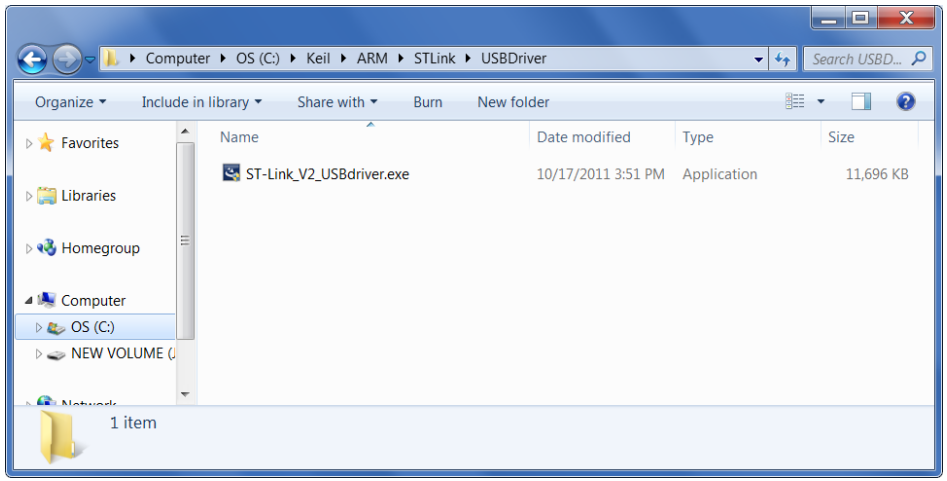
- Jumpers
  - JP3: USART1 TX and RX (not fitted, reserved function)
  - JP4: $I_{DD}$ for MCU current measurement (fitted by default)
- Documentation
  - UM1570 STM32F3DISCOVERY Kit
- Firmware Library
  - Contains STM32F3 Standard Firmware Library & ARM DSP Library.
  - Contains example code
    - UM1562
    - AN4157



# Step #3 - Install ST-Link Driver



- The STM32F3DISCOVERY board includes and ST-LINK/V2 embedded programming and debug tool
- The driver for ST-Link is contained in the Keil uVision toolchain and located in this directory:
  - C:\Keil\ARM\STLink\USBDriver
- Double-click on the file: ST-Link_V2_USBDriver.exe to install
- Click through the installation menu until the driver installation is complete



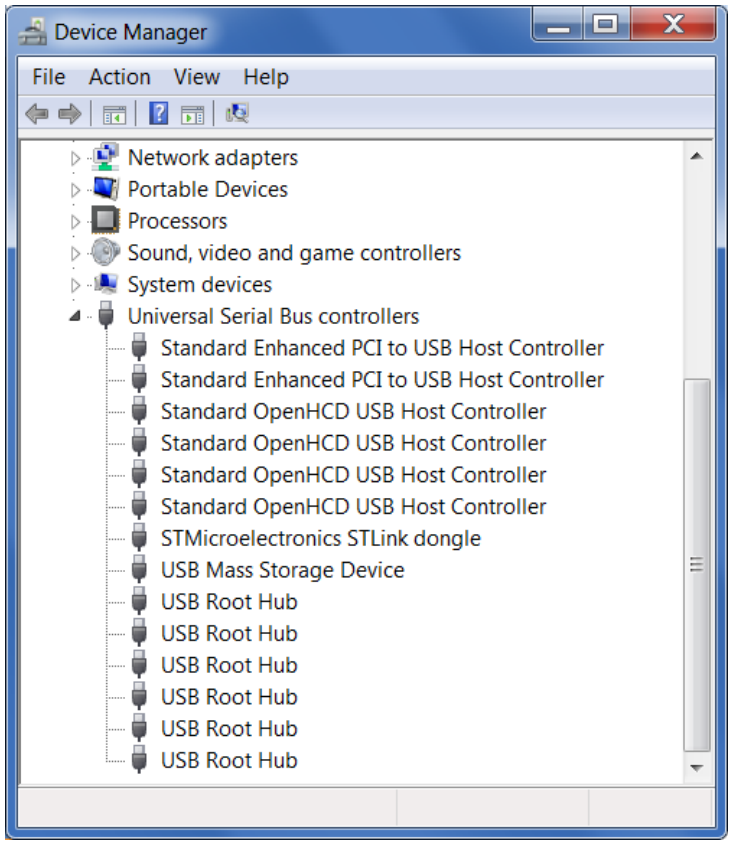
# Step #4: Connect the Discovery Kit/Enable ST-Link



- Using the USB cable, connect the mini-B male connector into the STM32F3DISCOVERY USB port and connect the A male connector into your Laptop

- Wait for Windows to recognize the ST-Link device and follow any step required to install the driver

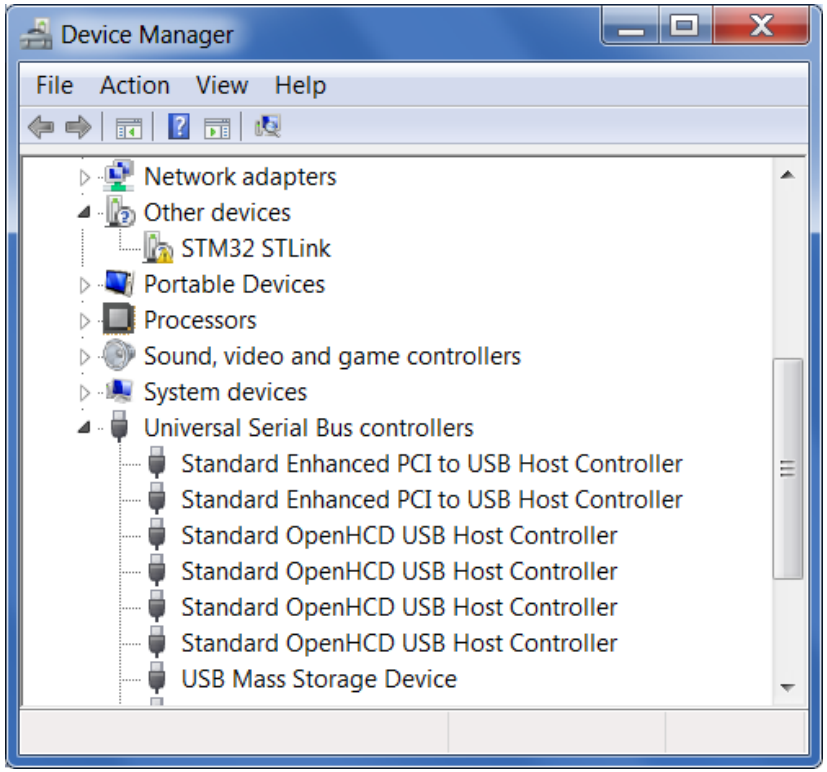
- Upon successful driver recognition, the ST-Link device should be fully enumerated in Windows Device Manager as show:



# Step #4
# ST-Link Driver Trouble Shooting

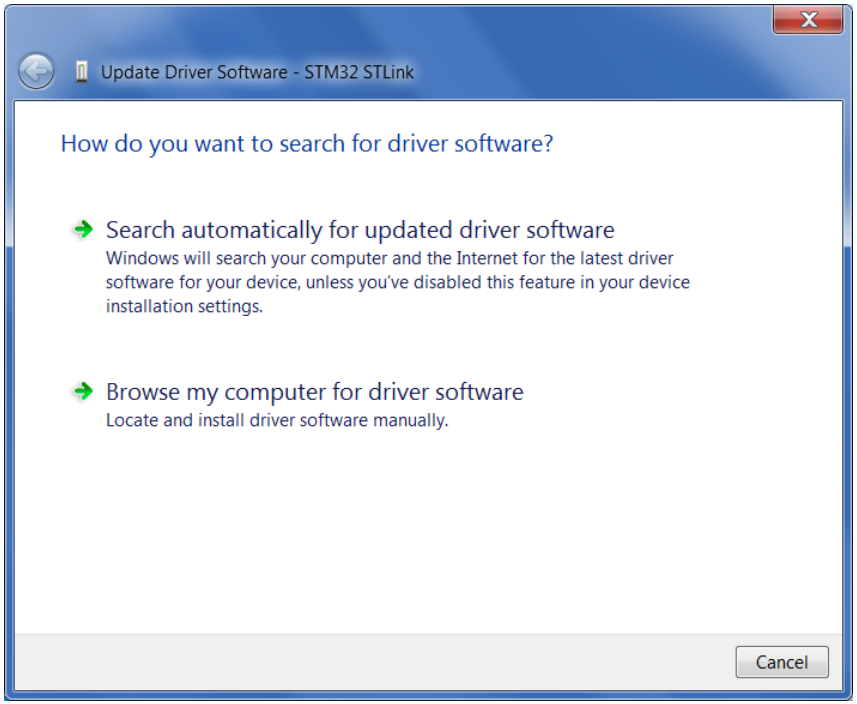


1. Open Device Manager
2. Right-click on the STM32 ST-Link Driver icon
3. Select “Update Driver Software”



# Step #4
# ST-Link Driver Trouble Shooting

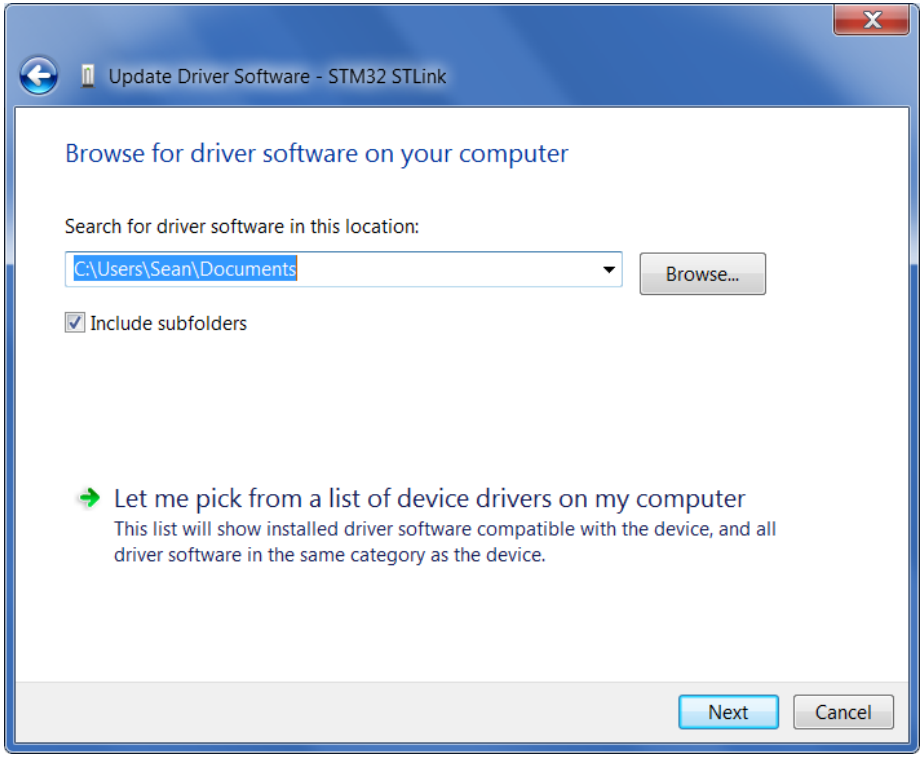





5. Select “Let me pick from a list of device drivers of my computer”

6. Click “Next”

4. Select “Browse my computer for driver software”



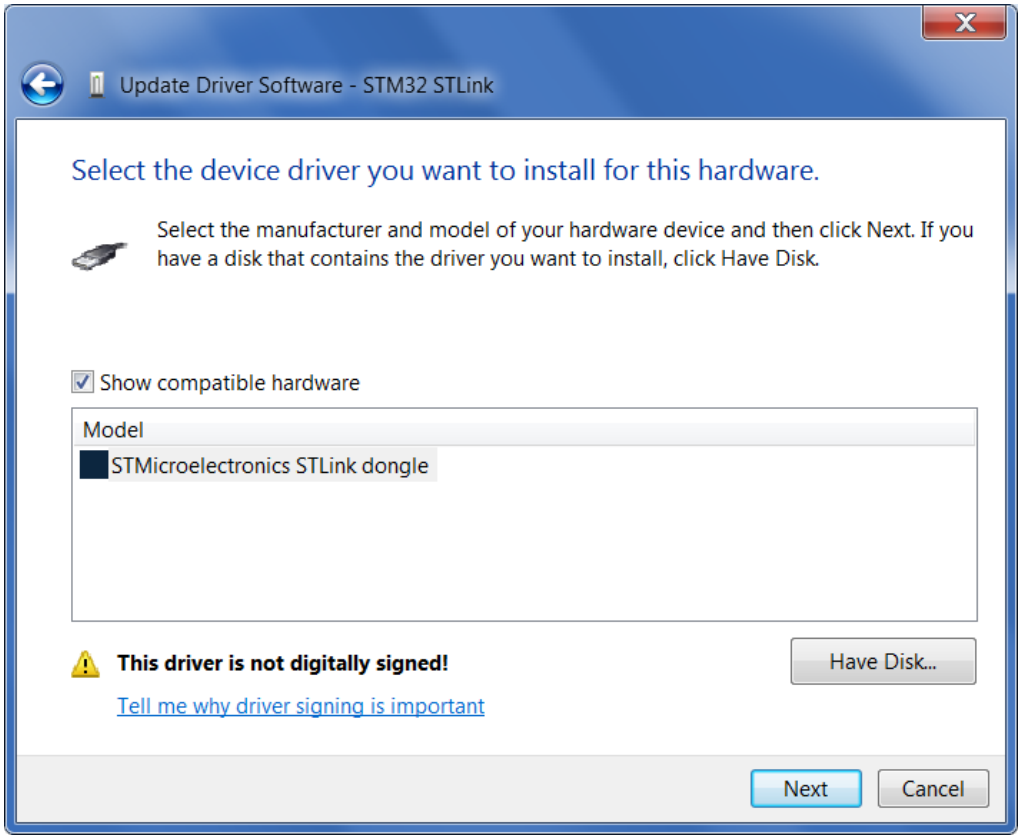

life.augmented

# Step #4
# ST-Link Driver Trouble Shooting



- The “STMicroelectronics ST-Link dongle” should listed

7. Click “Next”

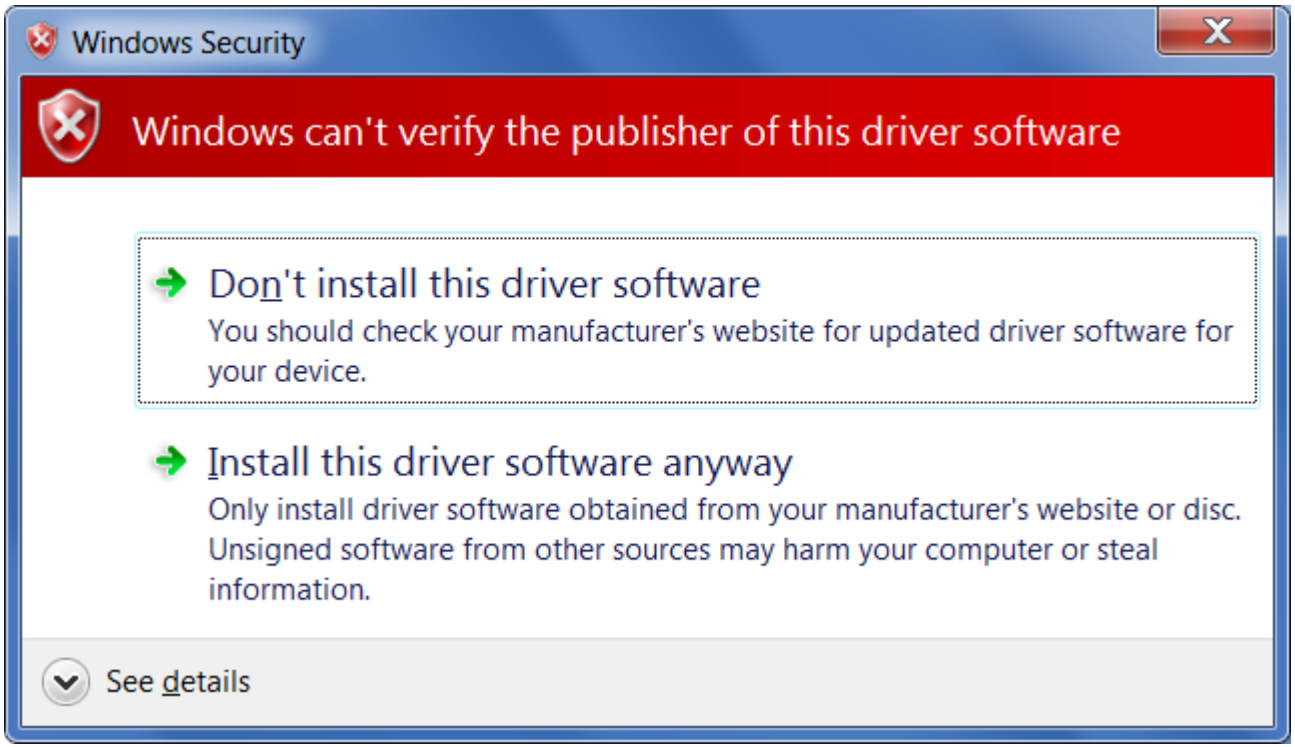


# Step #4
# ST-Link Driver Trouble Shooting



- A warning message may appear

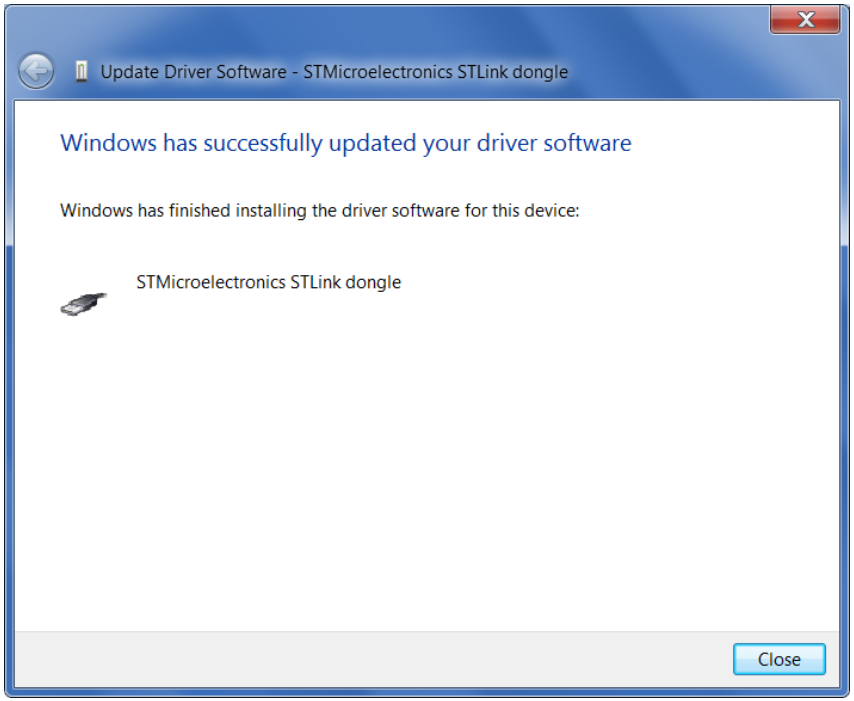
8. Select “Install this driver software anyway”

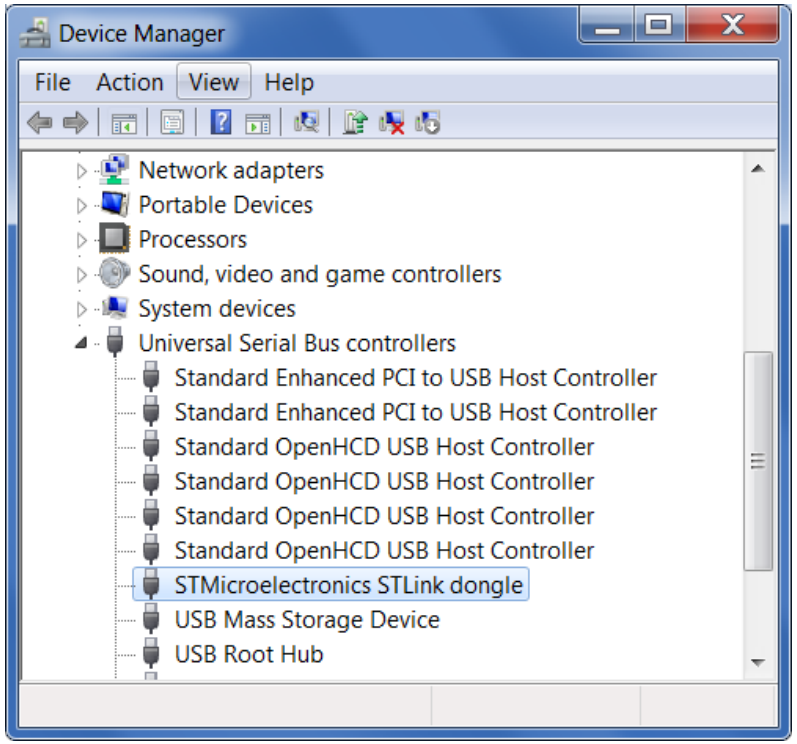


# Step #4
# ST-Link Driver Trouble Shooting



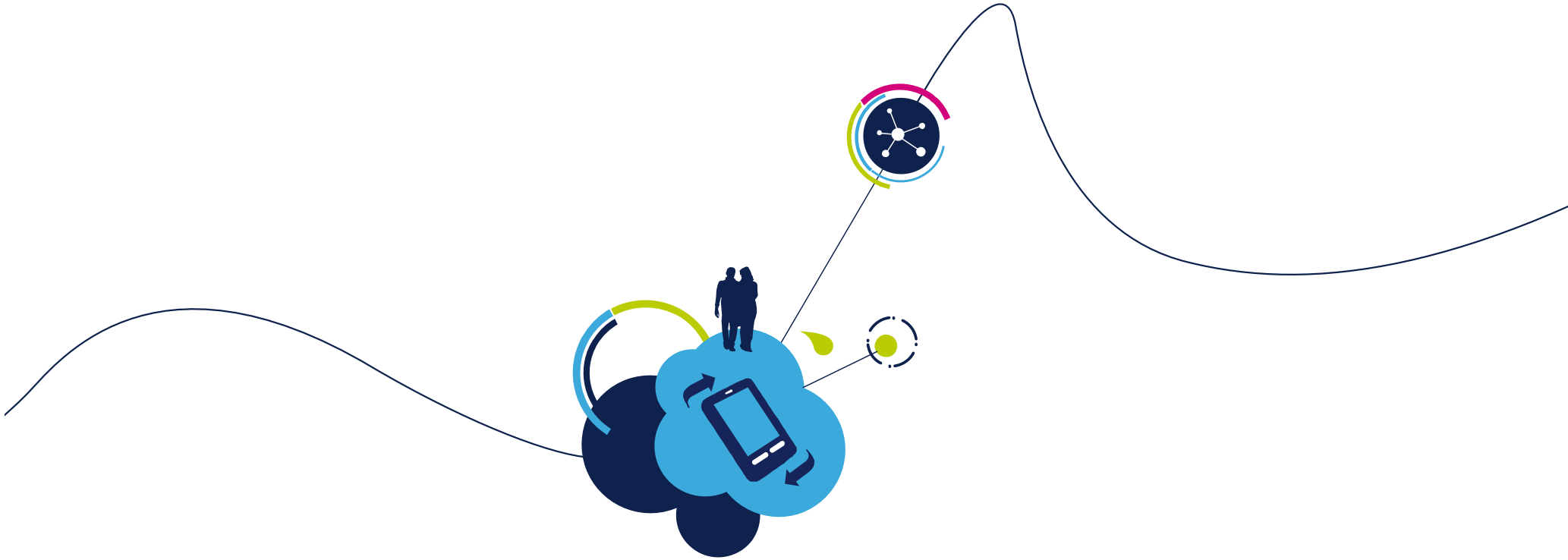
- You should receive a message: “Windows has successfully updated your driver software”




- Re-check device manager to ensure STMicroelectronics ST-Link dongle is functioning normally

life.augmented



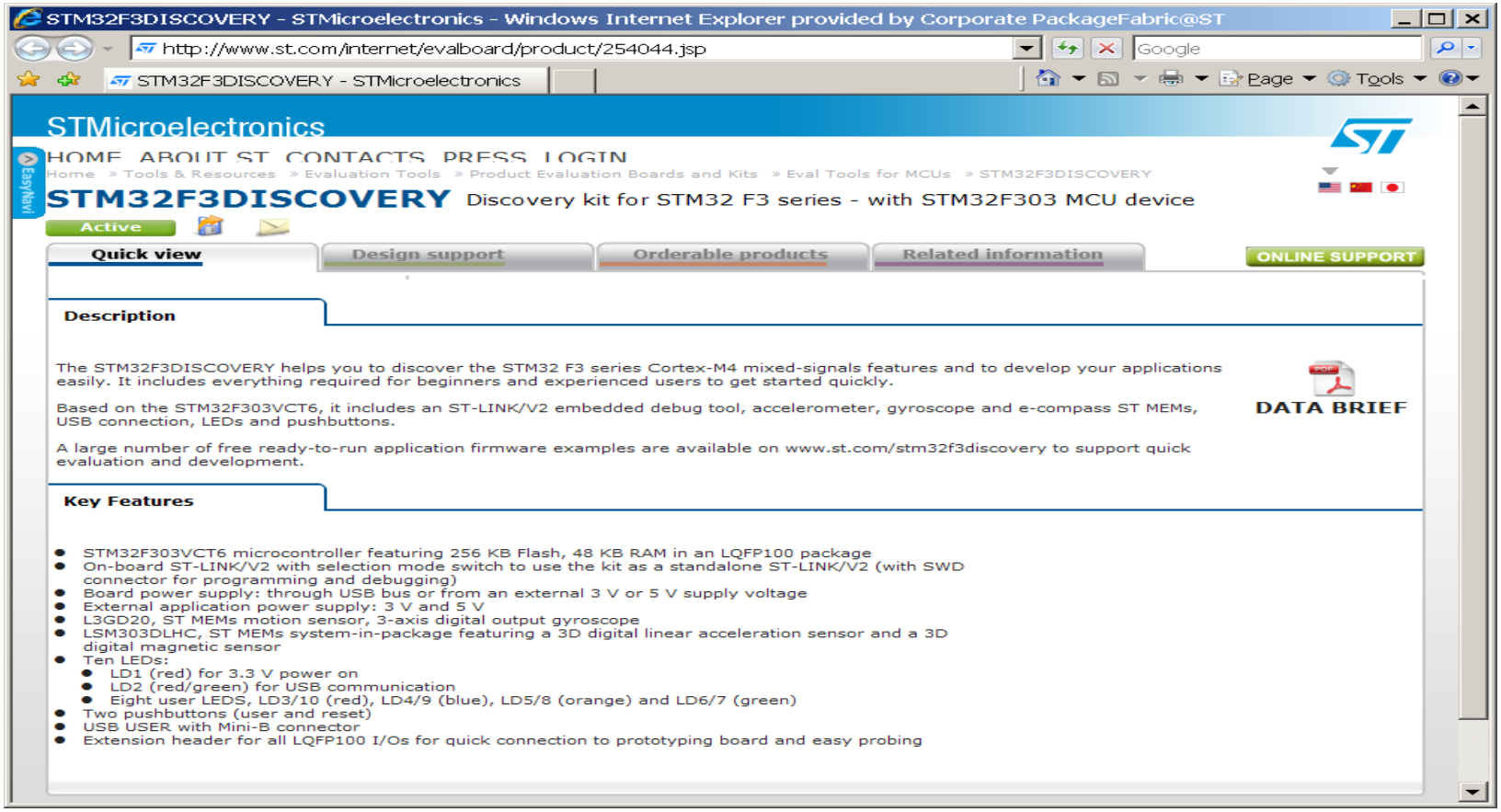
# STM32 F3 RESOURCES

life.augmented

# Documentation resources



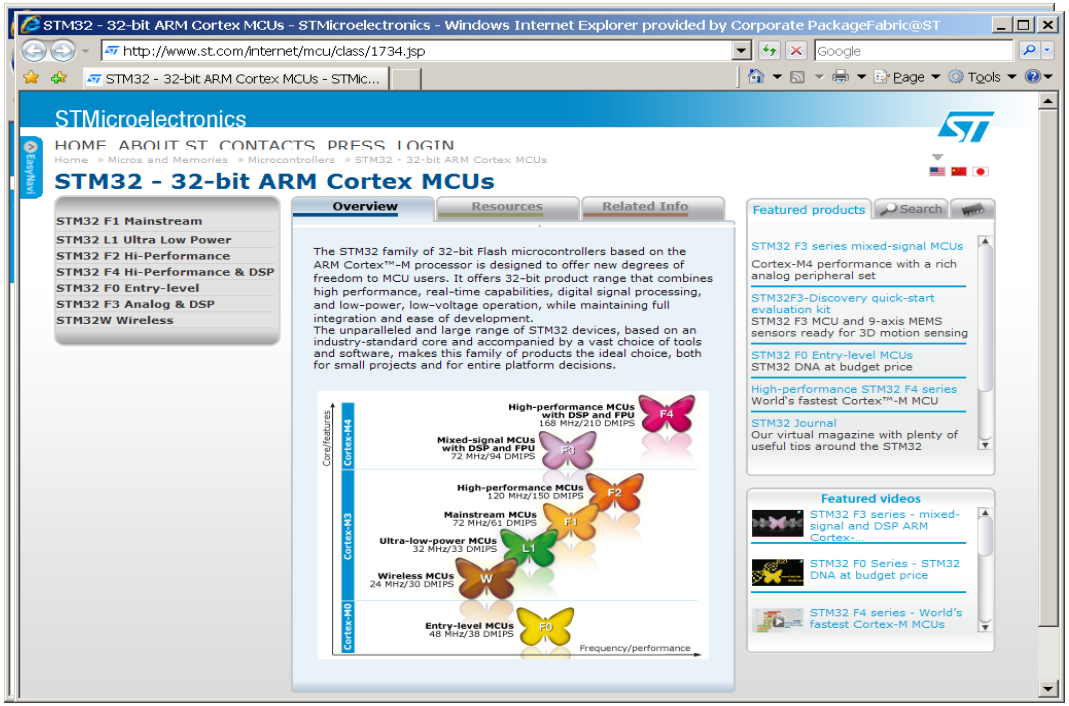
- All documentation can be found at www.st.com/stm32f3discovery under the “Design support” tab and….



# Documentation resources



- Main website page for the STM32 family
  - www.st.com/stm32
- For STM32F3
  - www.st.com/stm32f3
- You can find
  - Datasheets
  - Applications Notes
  - Errata
  - Technical Notes
  - Programming Manuals
  - Reference Manual
  - User Manuals
  - Firmware

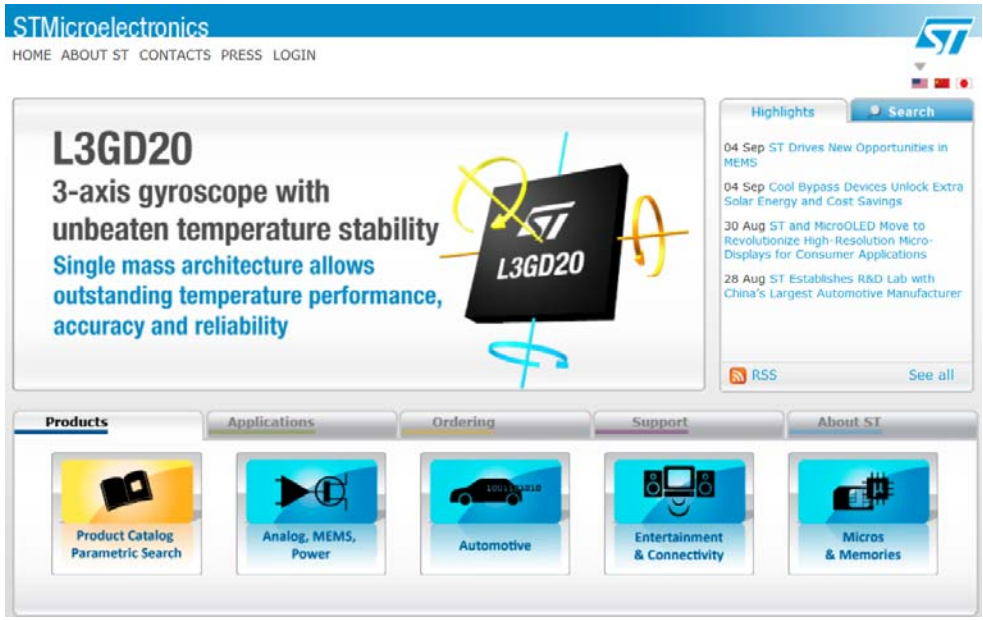



life.augmented

# Support resources



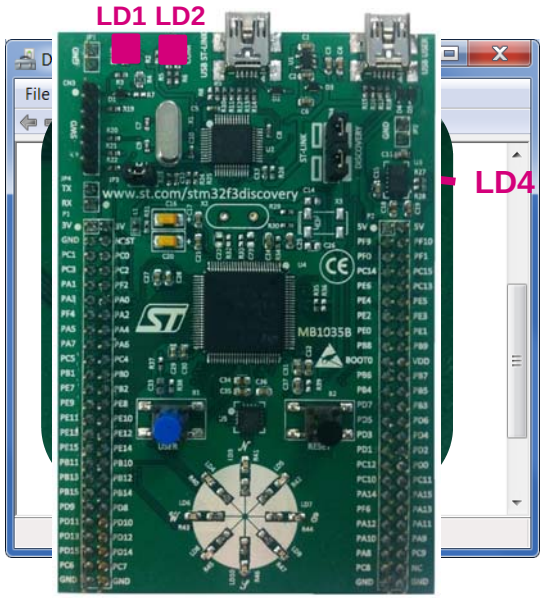
- Technically trained distributors
  - Distributors listed on CONTACTS page, www.st.com/contactus
- ST Public Forums:
  - Located on main www.st.com page under Support tab – ST e2e Communities
- Submit technical questions to ST Online Support:
  - Located on main www.st.com page under the Support tab – Online Support



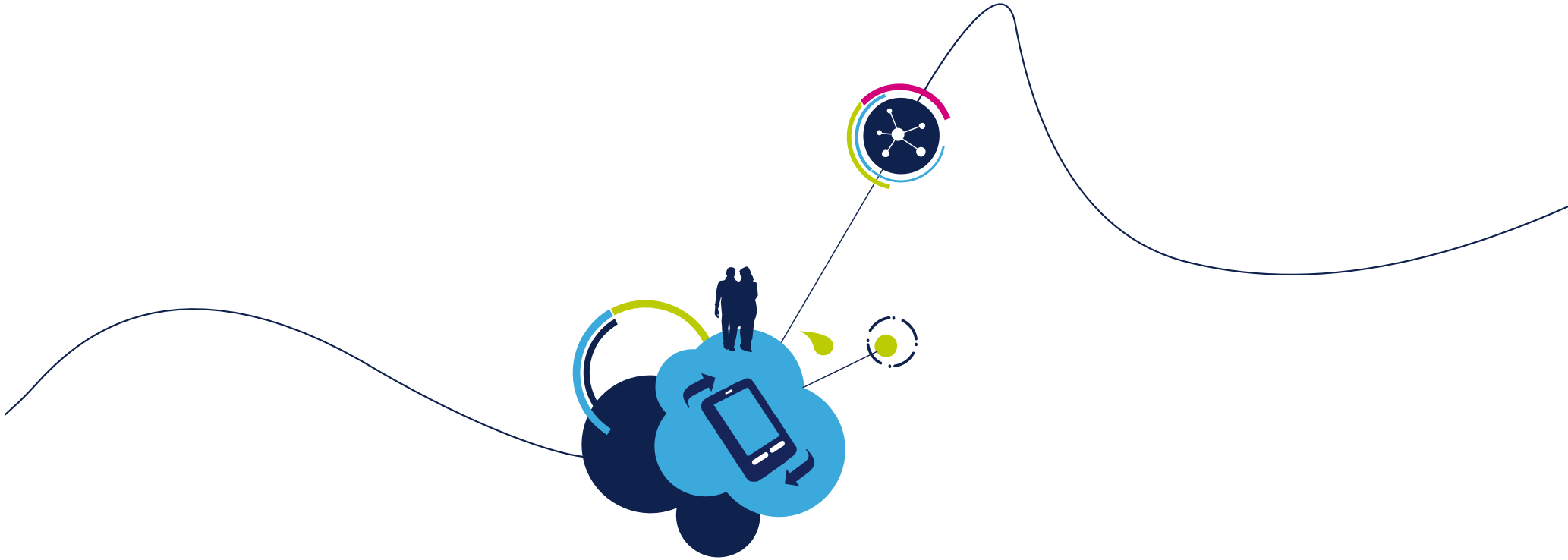

life.augmented

# Process check






- At this point the ST-Link V2 should be recognized by your system.
- LD1 and LD2 should be on ON (indicating the board is powered and ST-Link is functional).
- LD3 to LD10 will be flashing in a rotating pattern.
- Board Test:
  - Press the USER Button Once to Select Gyro Function
    - LD6 & LD9 (Green) will light when the Discovery board is rotated along the Roll access.
    - LD4 & LD10 (Blue) will light when the Discovery board is rotated along the Pitch access.
  - Press the USER Button a 2nd time to Select the Digital Compass Function.
    - LD3 thru LD10 will Flash randomly until the Discovery is rotated.
    - Rotate the Discovery board around the Yaw axis until LD4 (Blue) lights. LD4 will be pointing to magnetic North. (The STLINK USB connector will be pointing to the South.)
    - Rotate the Discovery Board around the Pitch or Roll axis.



# Hands-On Part I:

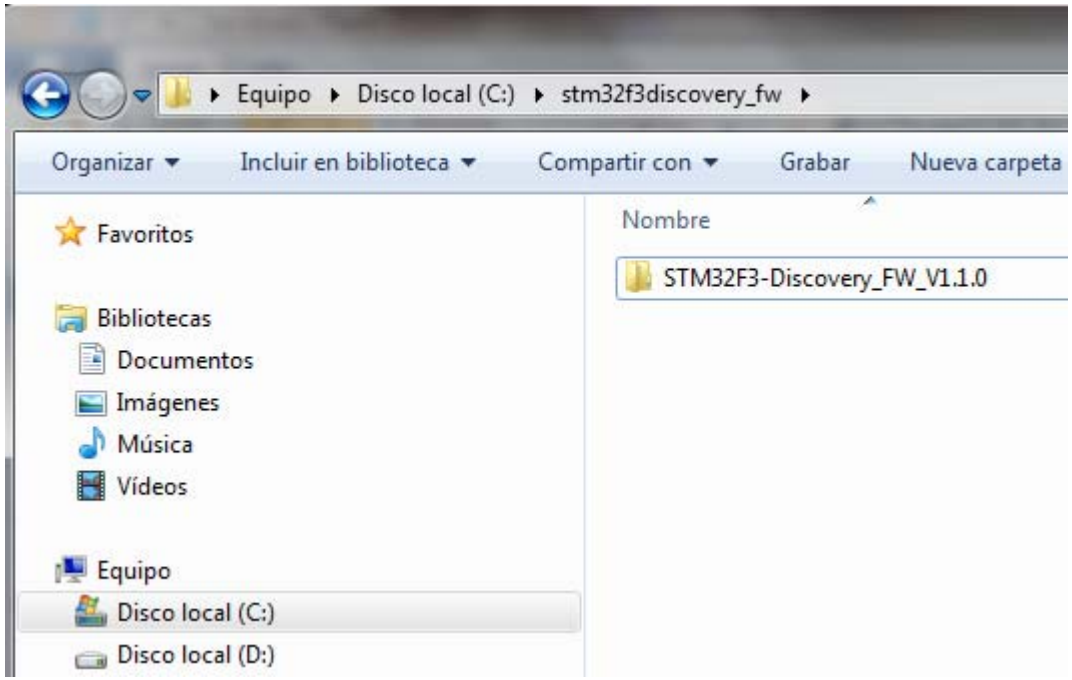
## Edit, Compile, Download, Debug, and Run

life.augmented

# Step #4b
# Change the project folder attributes



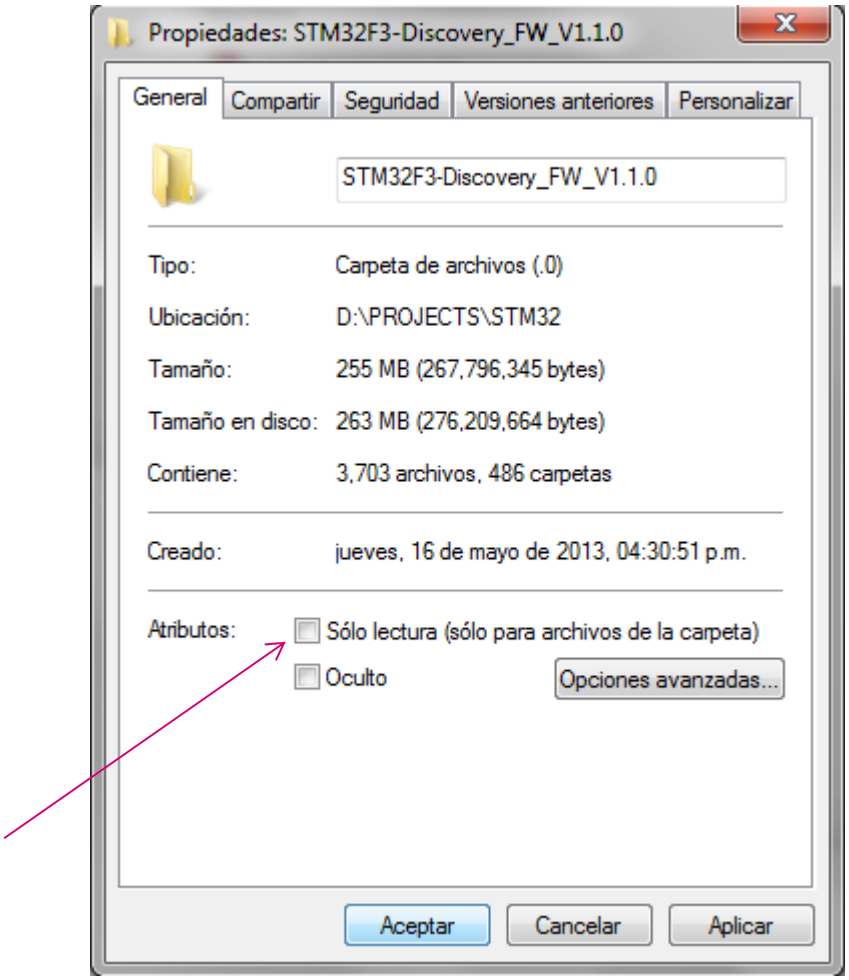
- Right-click on the STM32F3-Discovery_FW_V1.1.0 folder and select Properties…



# Step #4b



- Unselect the attribute: Only lecture and then Accept

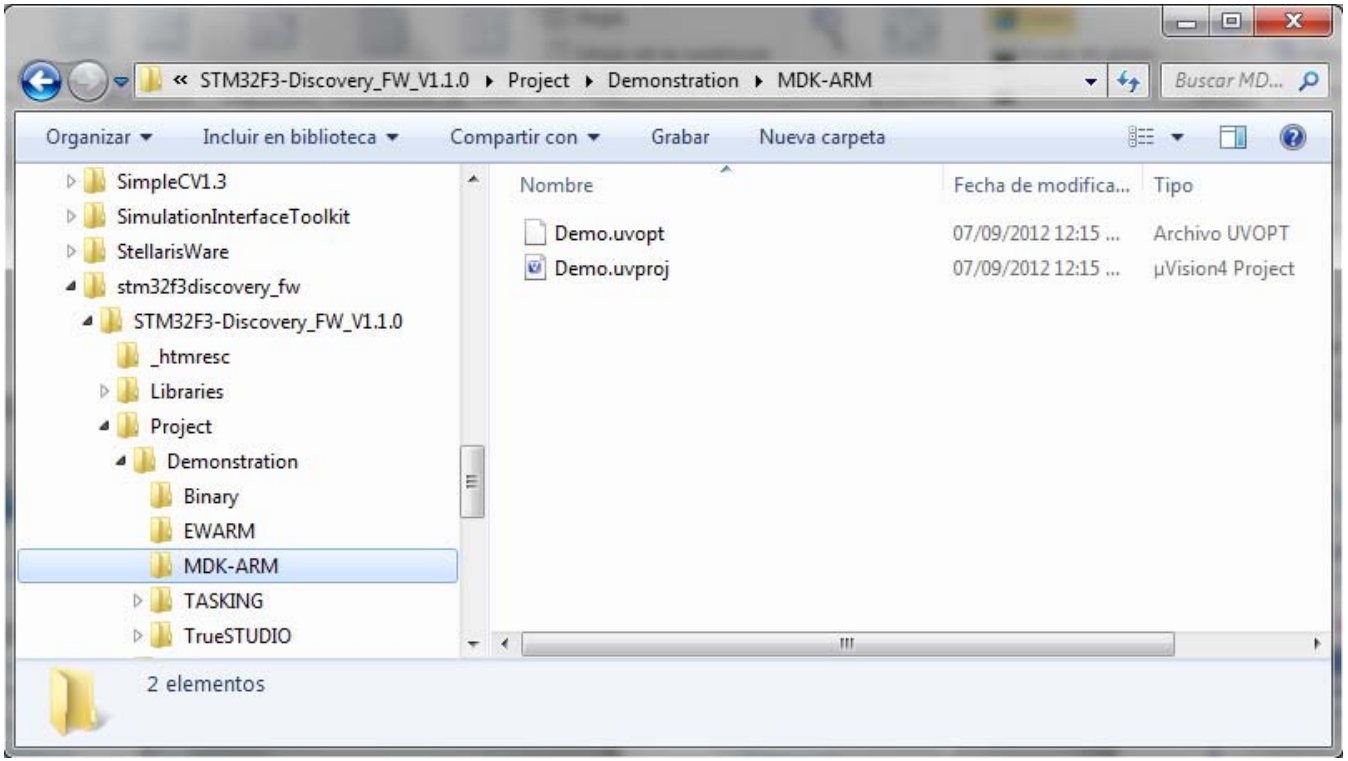


# Step #5
# Open FW demo project with Keil uVision

- Using explorer, go to the directory:

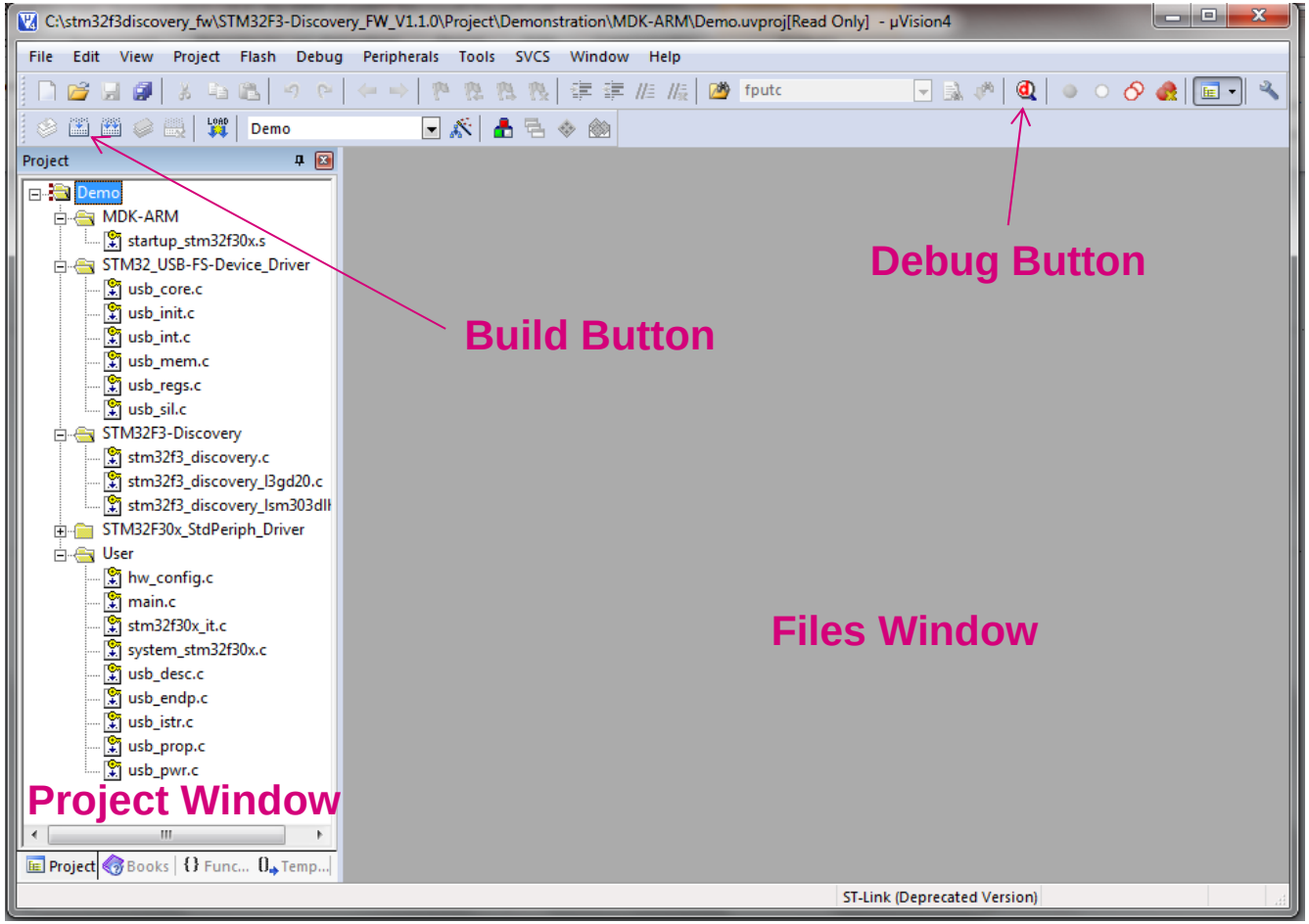
  C:\stm32f3discovery_fw\STM32F3-Discovery_FW_V1.0.0\Project\Demonstration\MDK-ARM

- Double-click on the Demo.uvproj file



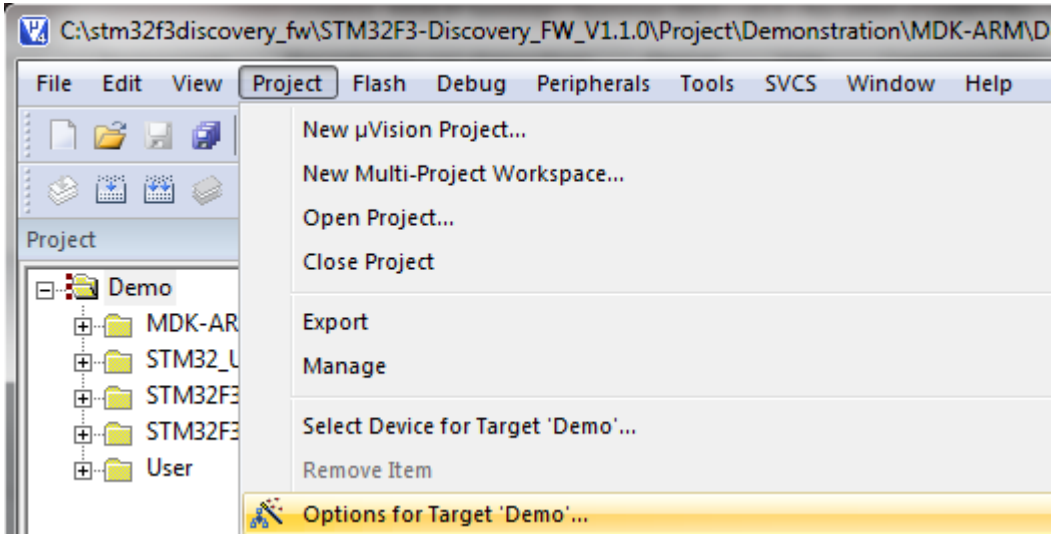
# Step #5 - Inside Keil uVision





# Step #5b
# Change the Options for Target ‘Demo’

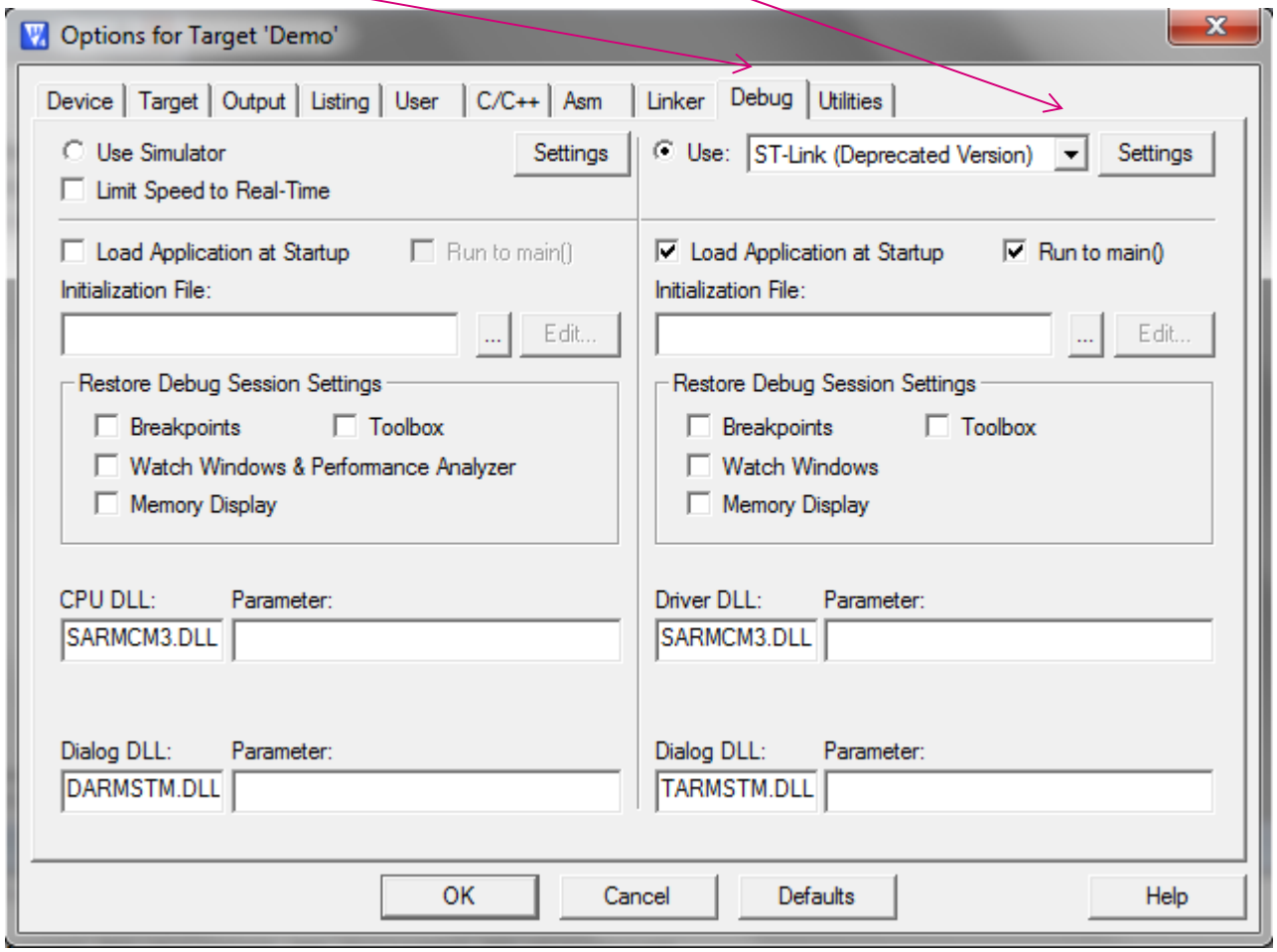
- Select Project::Options for Target ‘Demo’



# Step #5b



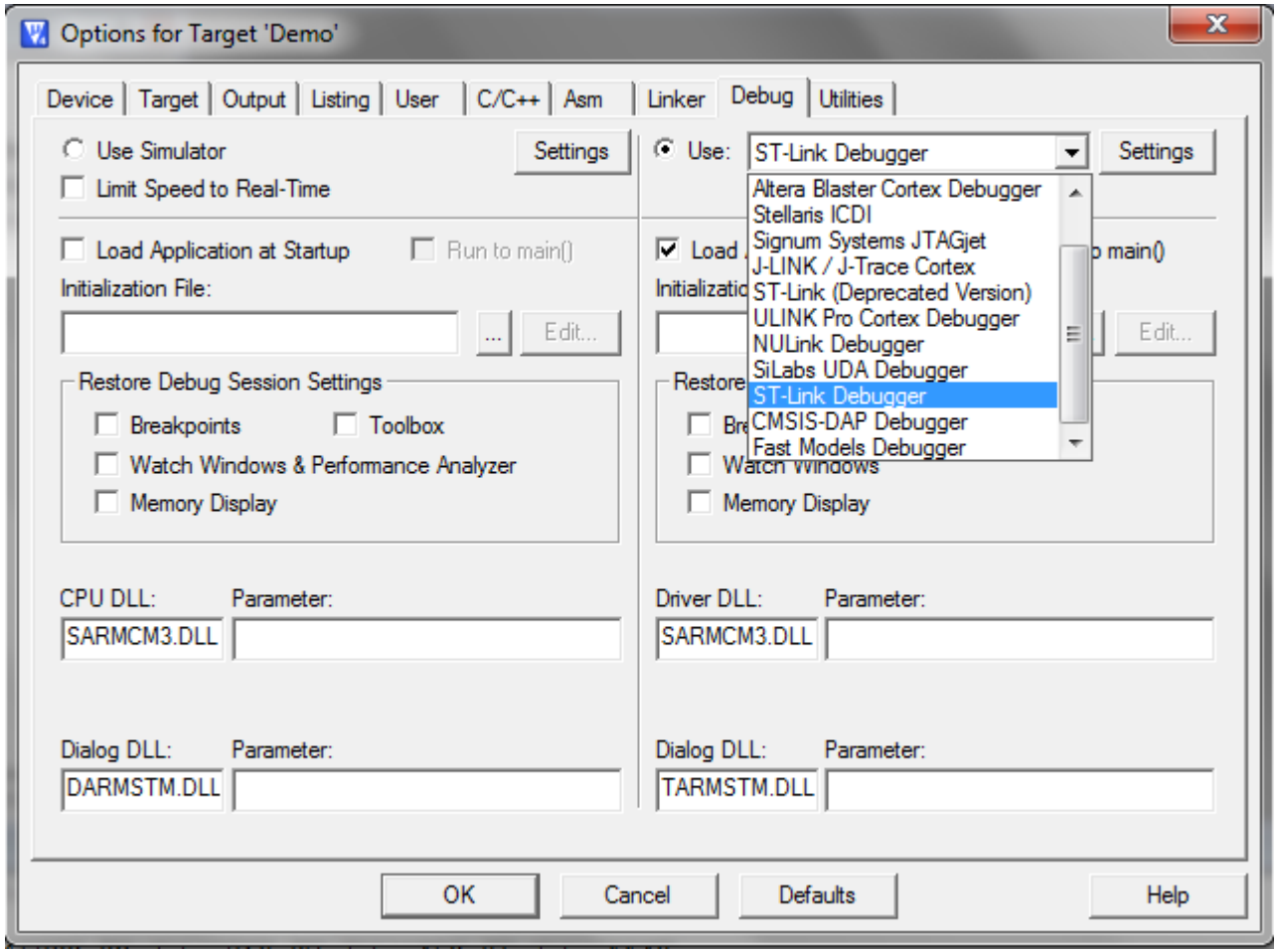
- Select Debug. Click on the ▾ symbol



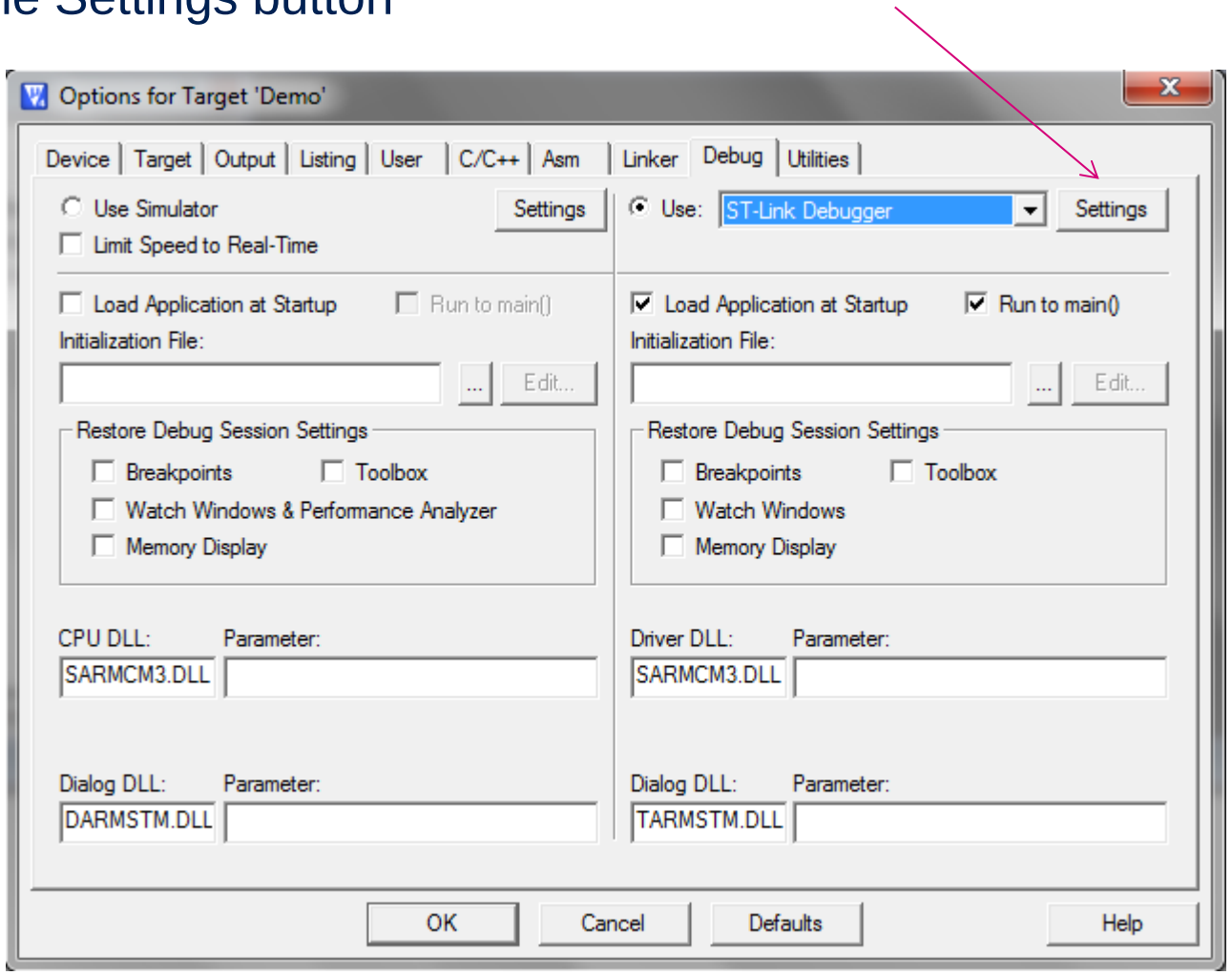
# Step #5b



- Select ST-Link Debugger



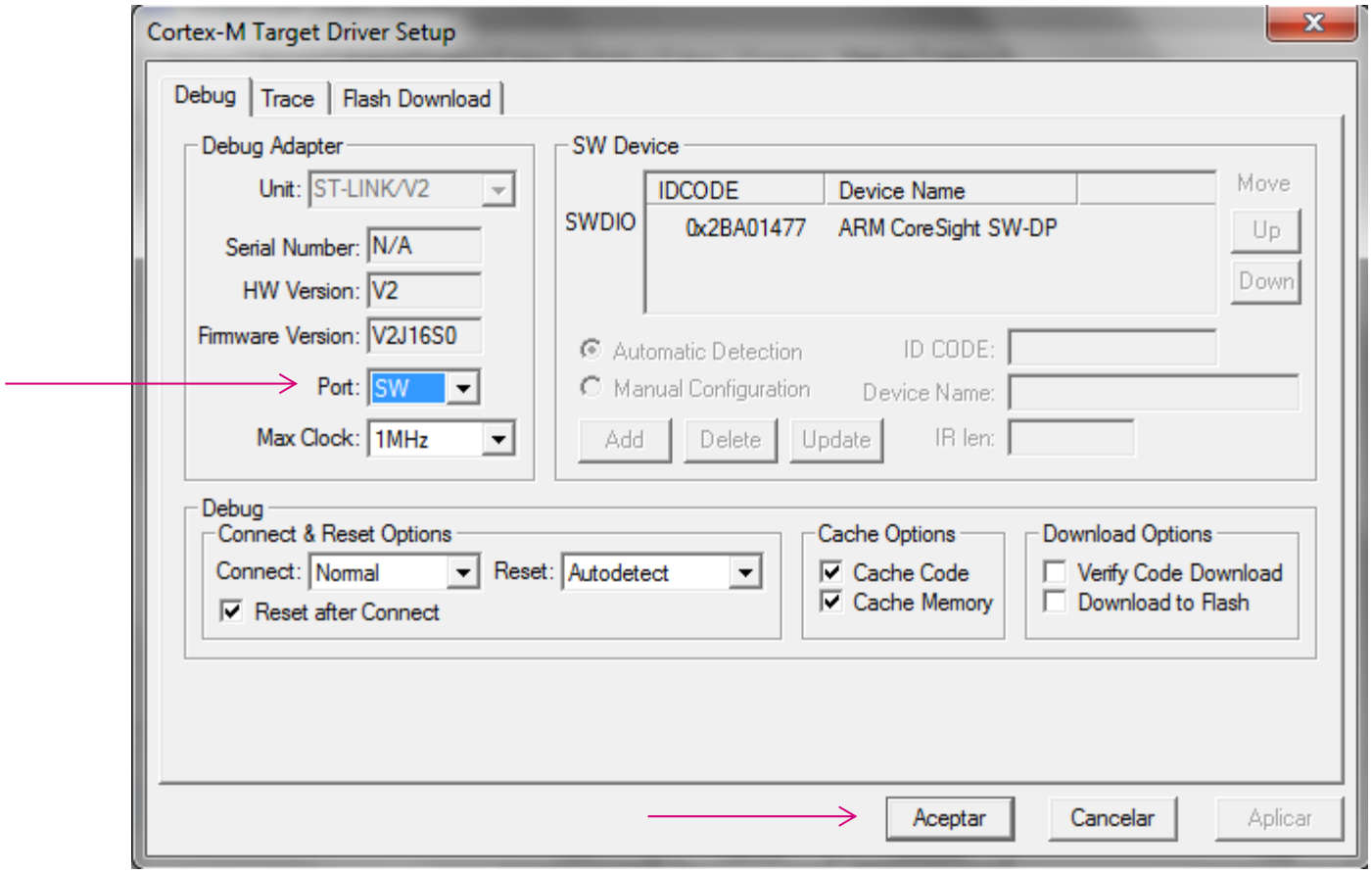
- Click the Settings button



# Step #5b



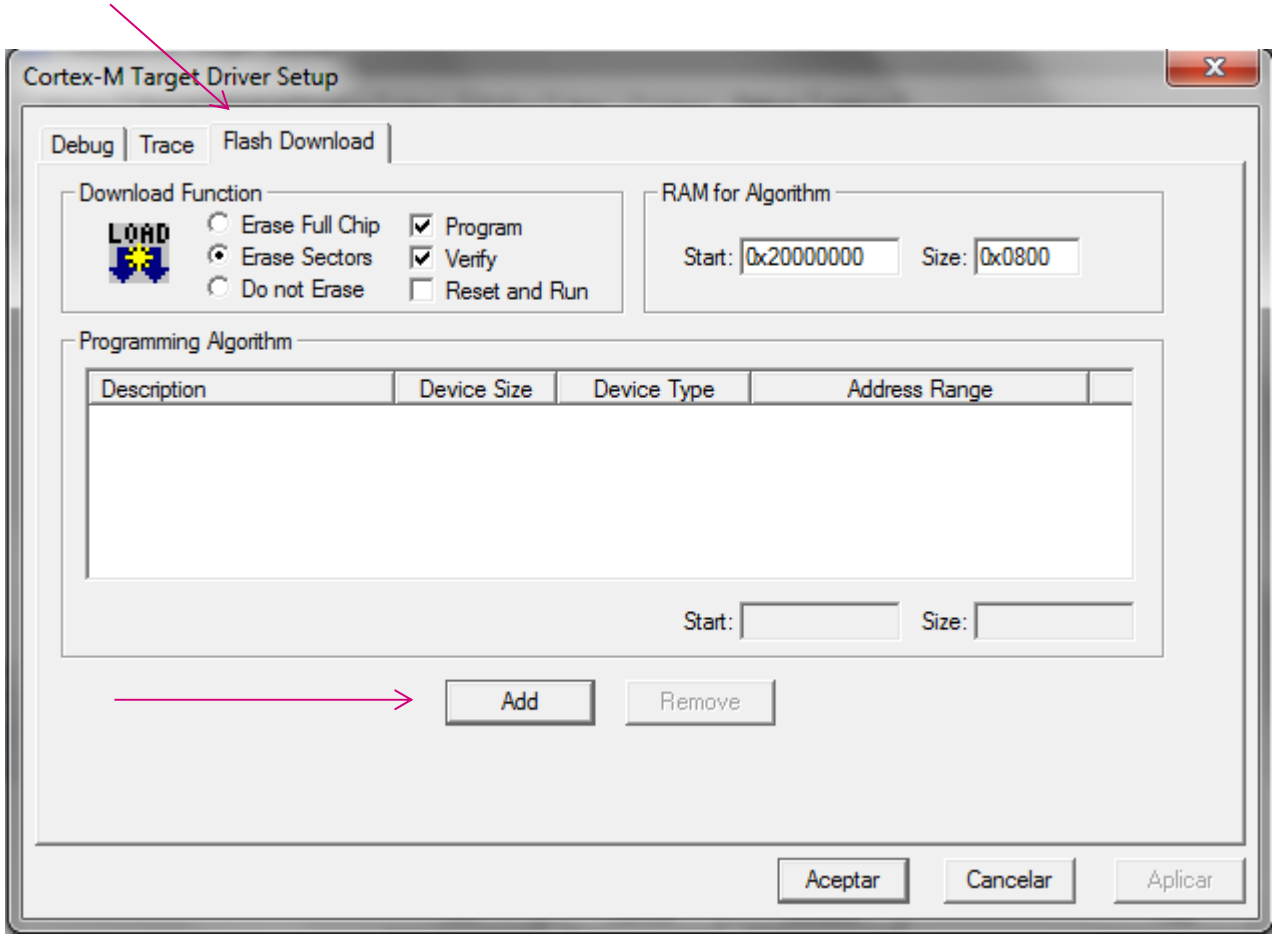
- Change Port to SW



# Step #5b



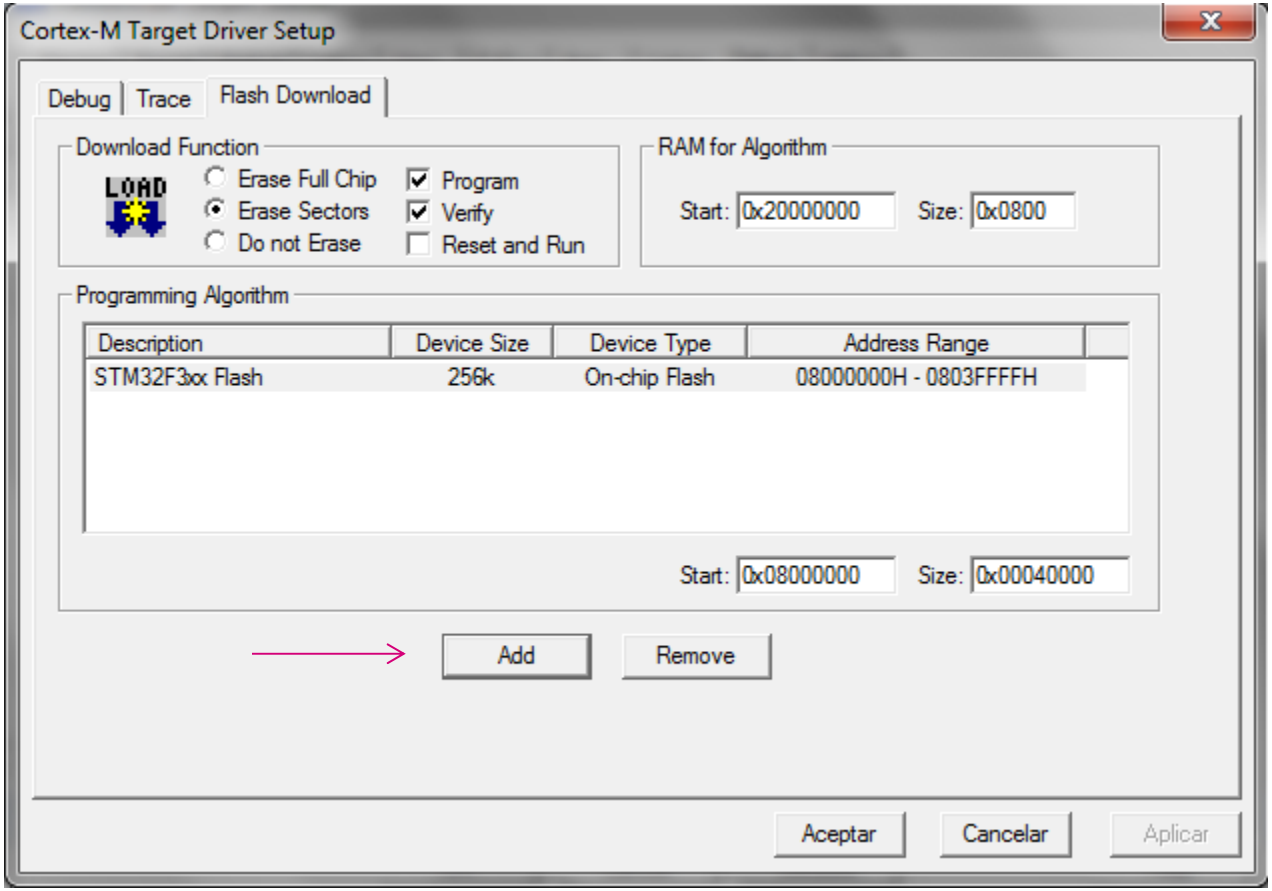
- Click on Flash Download and then the Add button



# Step #5b



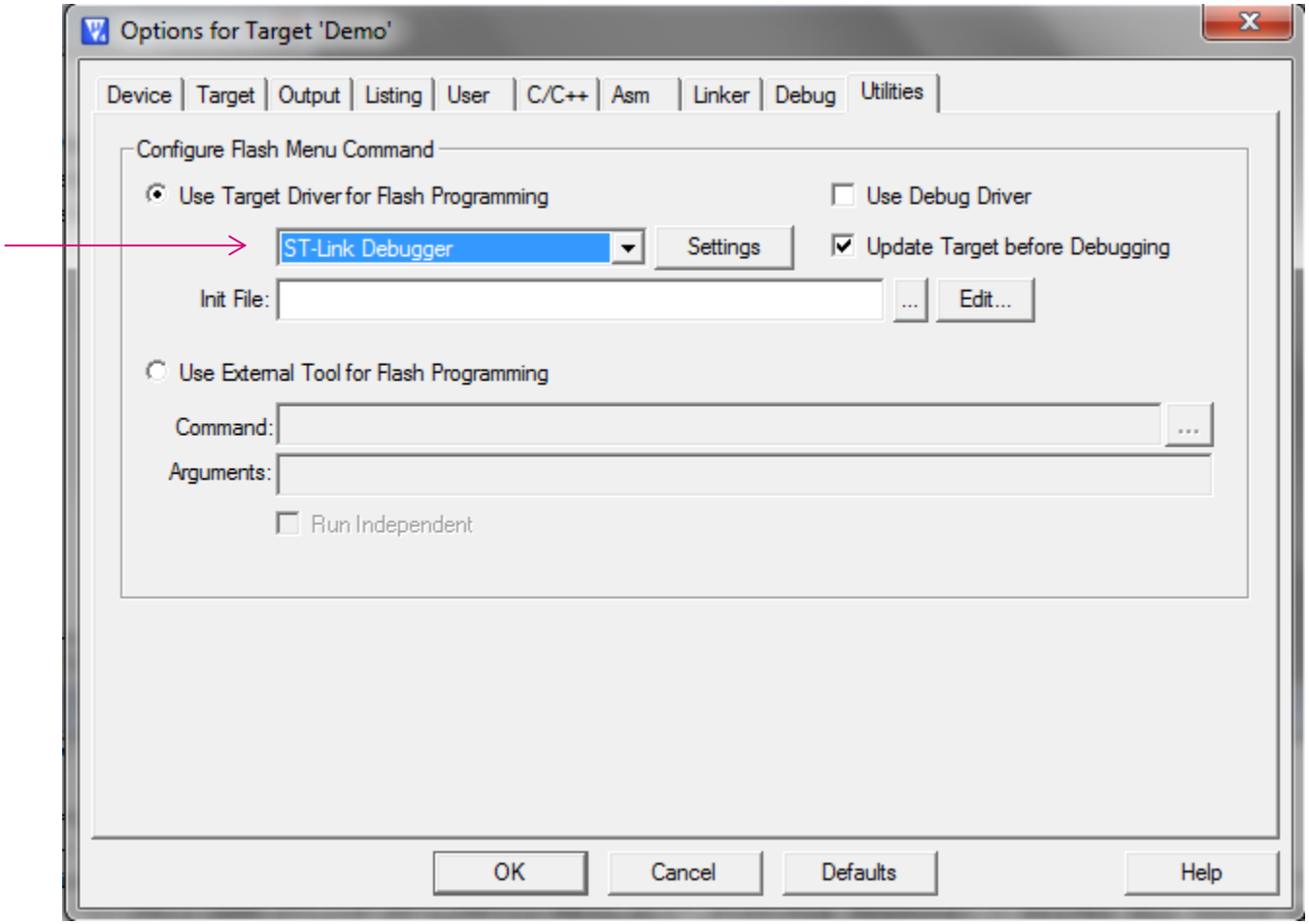
- Select STM32F3xx Flash and then click on the Add button



# Step #5b



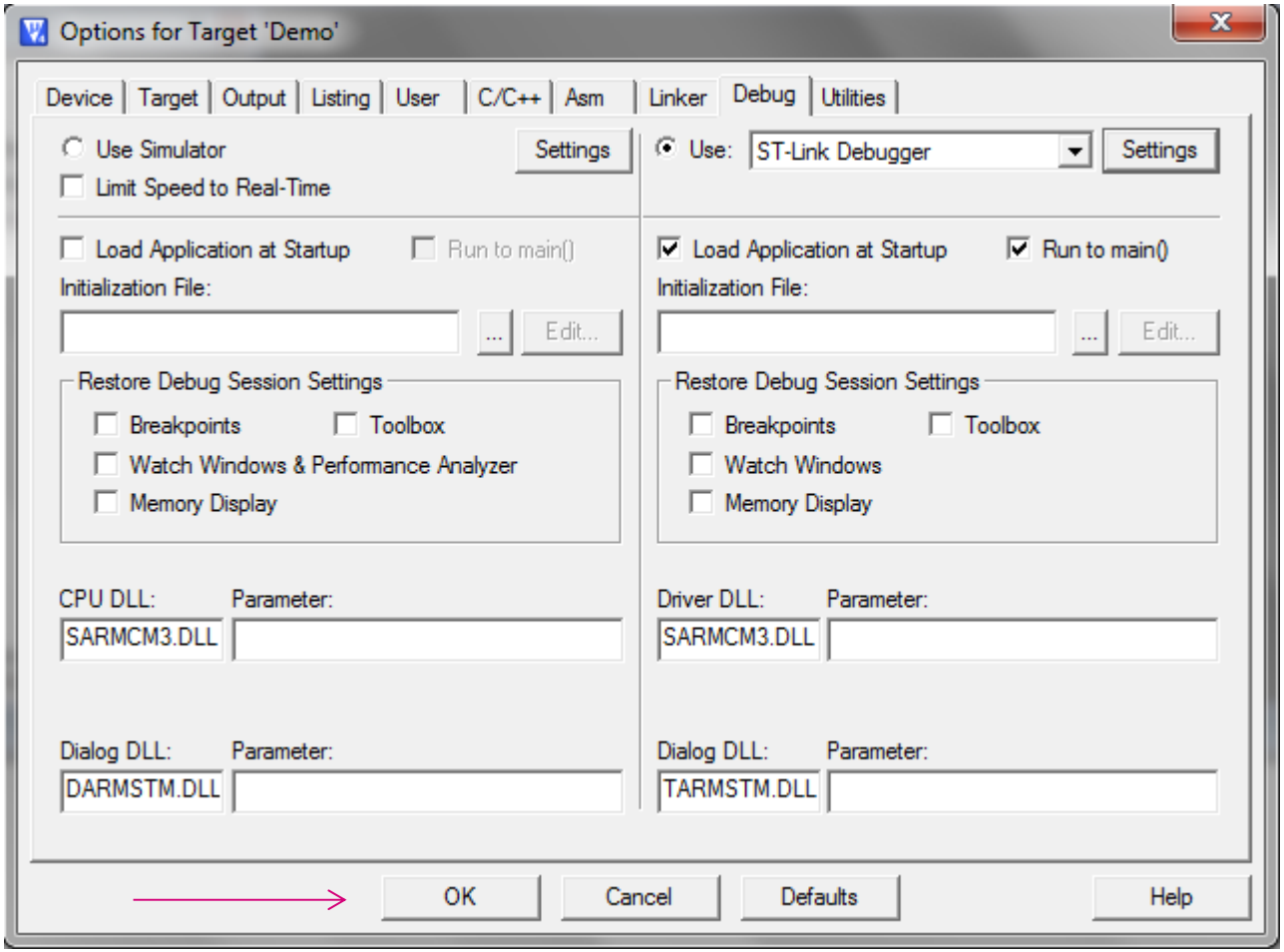
- Click on Utilities. Click the Settings button and select ST-Link Debugger



# Step #5b



- Finally, click on OK

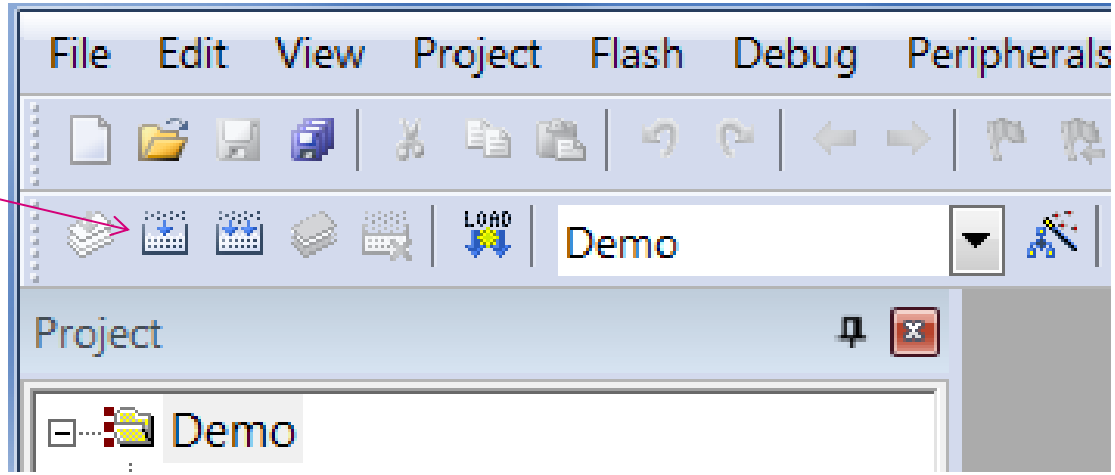


# Step #6 - Compile



- Click on the Build button or Menu::Project::Build Target

**Build Button**



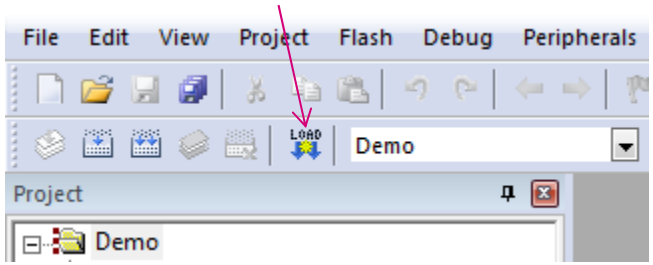

- The project should compile without errors

```
Build Output
compiling usb_istr.c...
compiling usb_prop.c...
compiling usb_pwr.c...
linking...
Program Size: Code=15792 RO-data=768 RW-data=452 ZI-data=1148
".\Demo\Demo.axf" - 0 Error(s), 2 Warning(s).
```

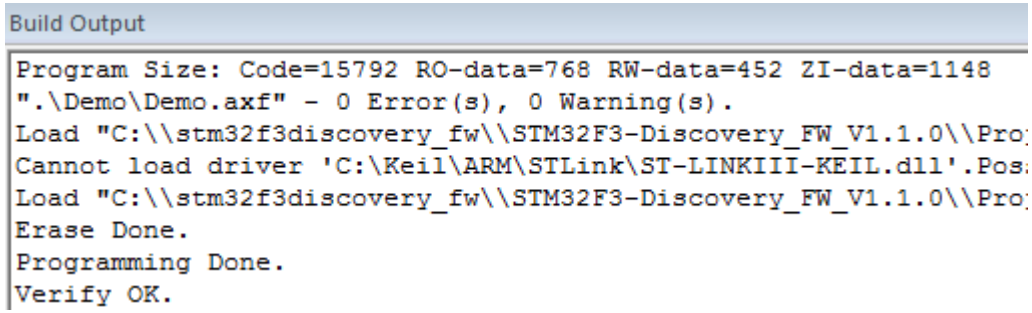
# Step #6b - Download



- Click on the Download Button




- The program is downloaded to the device's flash memory.

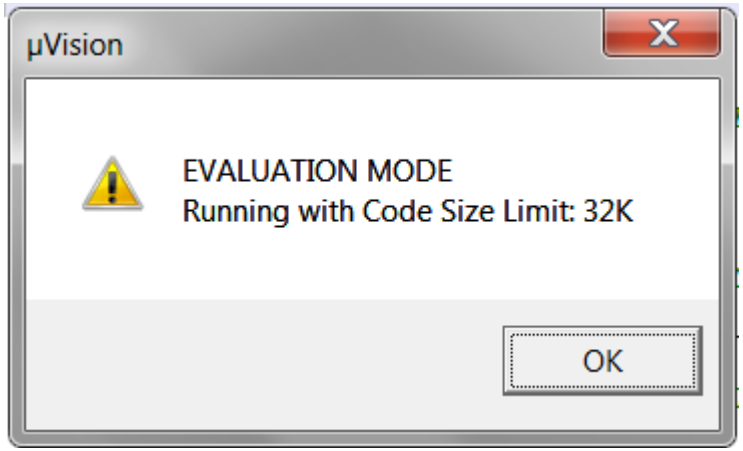


# Step #7 - Debug



- Click on the Start/Stop Debug Session button or Menu: Start/Stop Debug Session

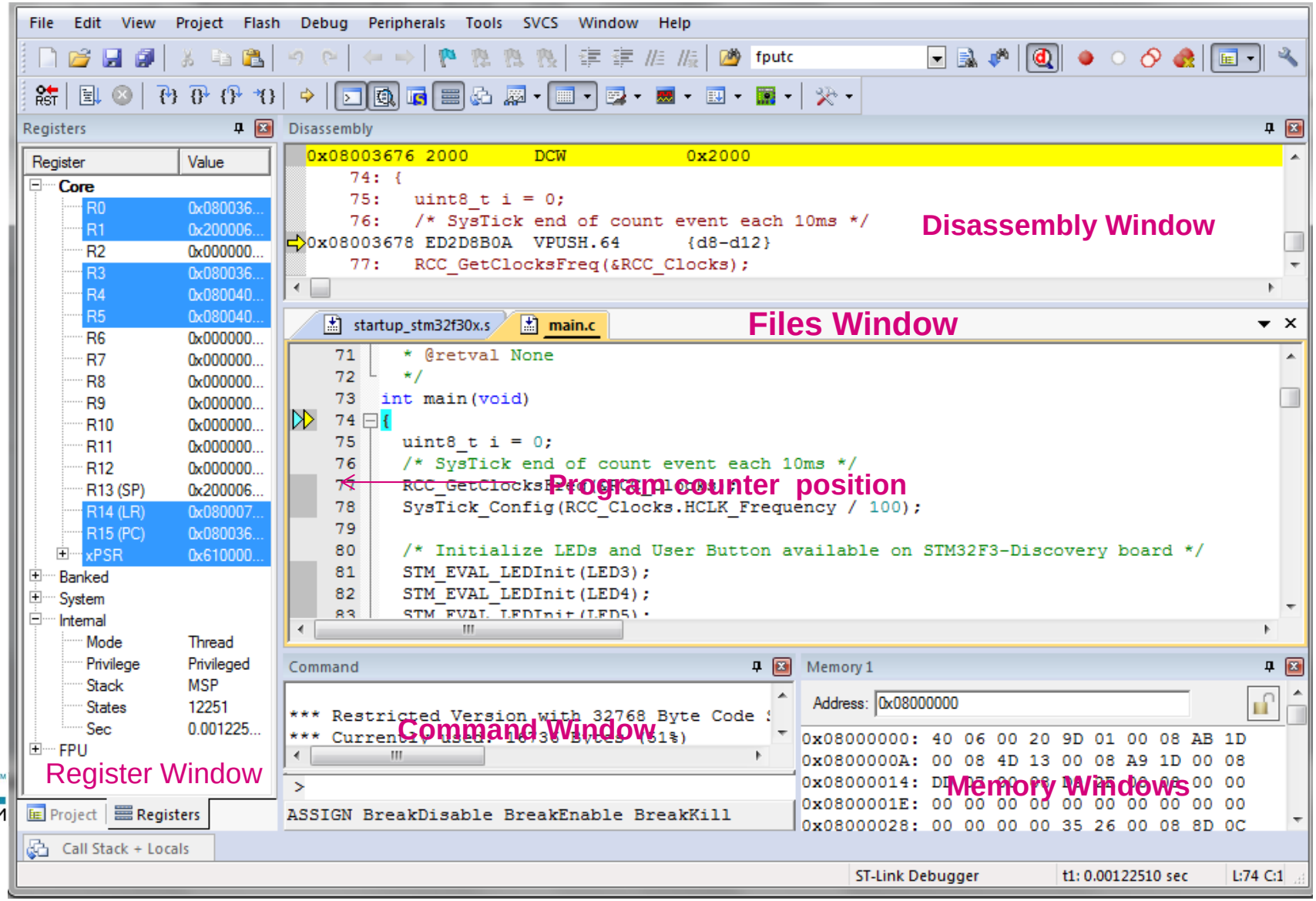
**Debug Button**




- You should receive a warning message. Click “OK”

# Step #7:
# The MDK-ARM IDE Debugger





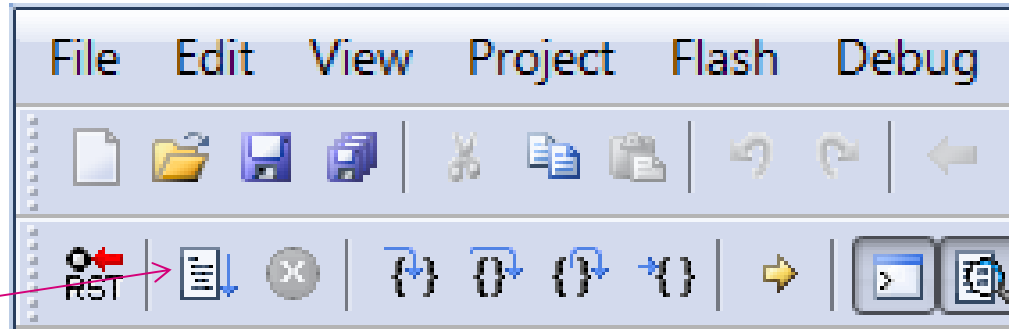

KEIL Tools by ARM

life.augmented

# Step #8 - Run



- Click on the Run button to start the program




Run Button

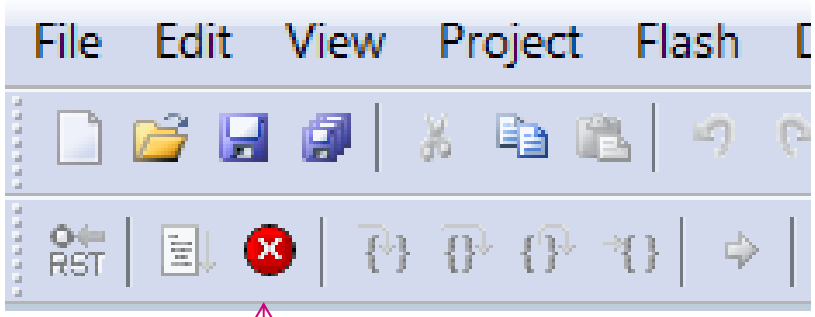
- Your STM32F3DISCOVERY board LD3 thru LD10 will be flashing in a rotating pattern.

- Note: LD2 (ST-Link Status) will be flashing because of the communication occurring between the STLINK/V2 and EWARM.

# Step #8 - Run



- Mission Accomplished
- Please click on the Stop button.
- You code will stop anywhere within the program flow
- Click on the Debug button to exit from the debugger




Stop Button

Debug Button

# Let's make a change



- Double-click to open the main.c file.
- Scroll down to line 117.
- Using MDK-ARM, What physical pin of the STM32F303 is LED3 connected to?
- Enter a number from 10 to 500 and place in the Delay(xxx) statement.
- Do the same thing with lines 121, 125, 129, 133, 137, 141, and 145.
- Compile, Debug, and Run
- Validate! Did it work?
- Stop debug and exit the debugger

```
113     {
114       /* Toggle LD3 */
115       STM_EVAL_LEDToggle(LED3);
116       /* Insert 50 ms delay */
117       Delay(5);
118       /* Toggle LD5 */
119       STM_EVAL_LEDToggle(LED5);
120       /* Insert 50 ms delay */
121       Delay(5);
122       /* Toggle LD7 */
123       STM_EVAL_LEDToggle(LED7);
124       /* Insert 50 ms delay */
125       Delay(5);
126       /* Toggle LD9 */
127       STM_EVAL_LEDToggle(LED9);
128       /* Insert 50 ms delay */
129       Delay(5);
130       /* Toggle LD10 */
131       STM_EVAL_LEDToggle(LED10);
132       /* Insert 50 ms delay */
133       Delay(5);
134       /* Toggle LD8 */
135       STM_EVAL_LEDToggle(LED8);
136       /* Insert 50 ms delay */
137       Delay(5);
138       /* Toggle LD6 */
139       STM_EVAL_LEDToggle(LED6);
140       /* Insert 50 ms delay */
141       Delay(5);
142       /* Toggle LD4 */
143       STM_EVAL_LEDToggle(LED4);
144       /* Insert 50 ms delay */
145       Delay(5);
146     }
```

# Step #10 Let's take a look and make a change



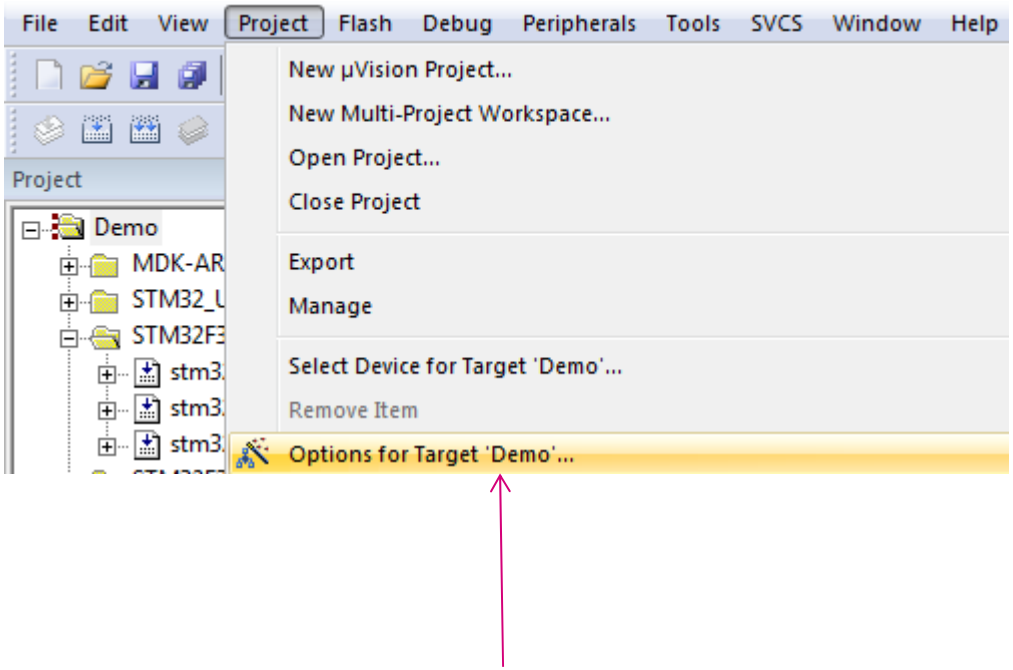
startup_stm32f30x.s | main.c | stm32f3_discovery.h | stm32f3_discovery.c

```
112  #define LED5_GPIO_CLK                    RCC_AHBPeriph_GPIOE
113
114  #define LED3_PIN                         GPIO_Pin_9
115  #define LED3_GPIO_PORT                   GPIOE
116  #define LED3_GPIO_CLK                    RCC_AHBPeriph_GPIOE
117
118  #define LED4_PIN                         GPIO_Pin_8
119  #define LED4_GPIO_PORT                   GPIOE
120  #define LED4_GPIO_CLK                    RCC_AHBPeriph_GPIOE
```

# Step #11 How Large Is The STM32F3Discovery Demo Code?



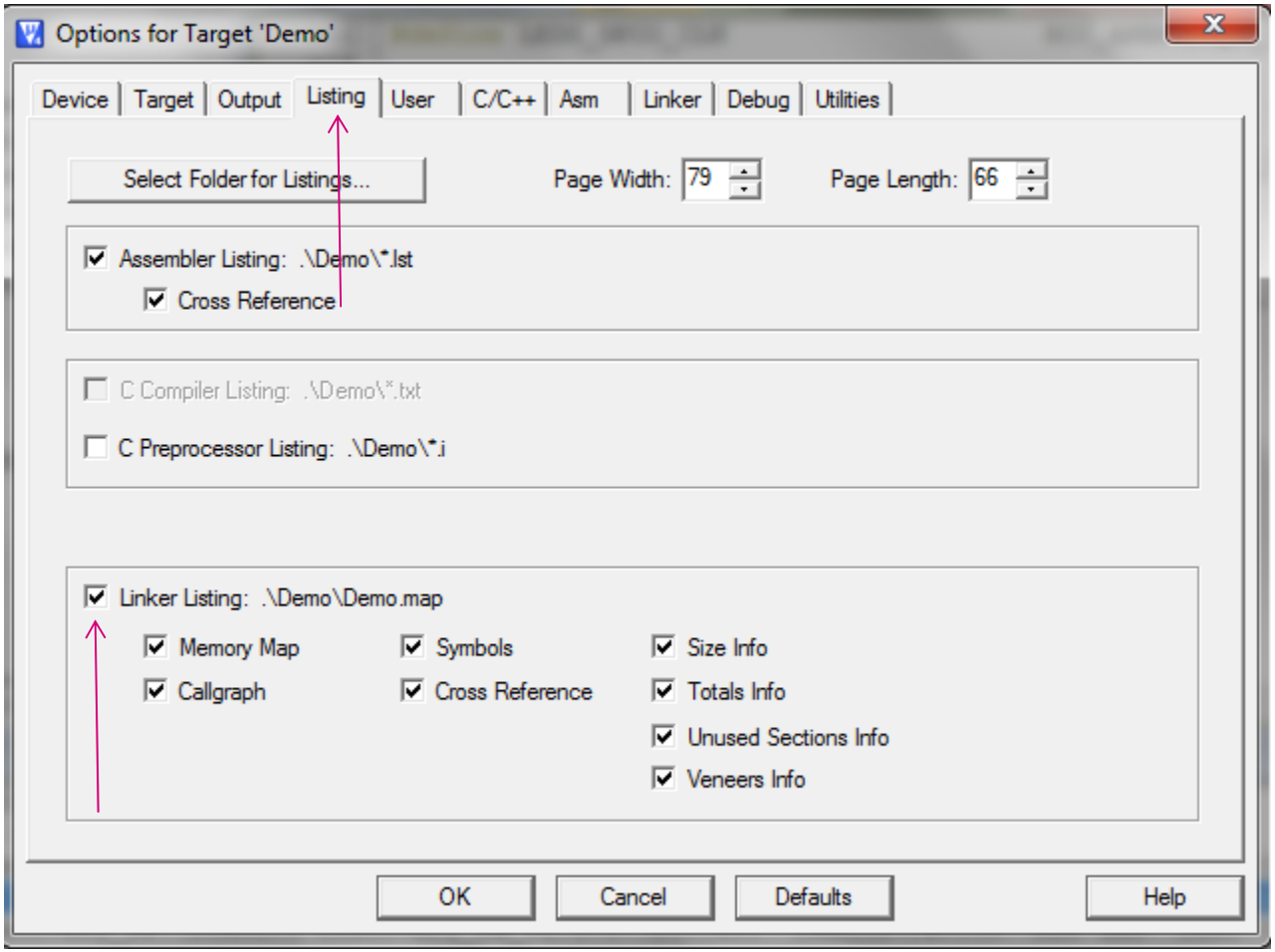
- Select Project::Options for Target ‘Demo’…



# Step #11 How Large Is The STM32F3Discovery Demo Code?



- Select Listing. Then select Linker Listing. Finally click on the OK button.

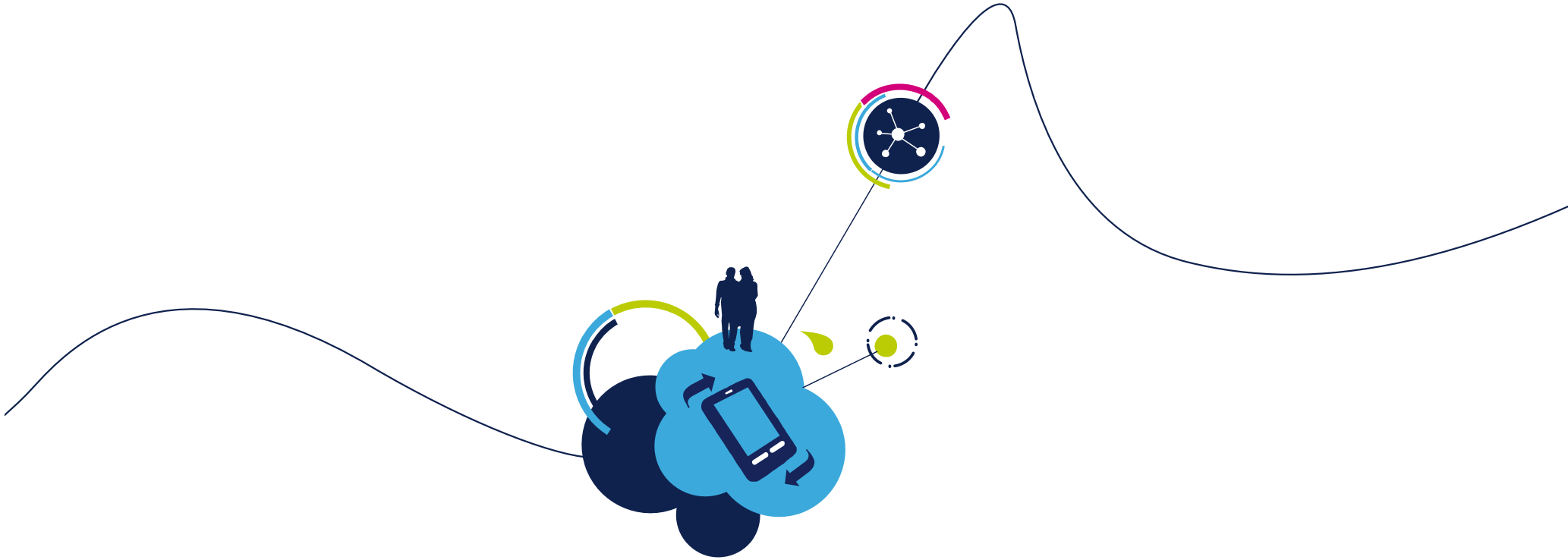


# Step #11 How Large Is The STM32F3Discovery Demo Code?



- Click on → Project → Build, to re-link the project and generate the 'Demo.map'.

- Edit the 'Demo.map' file.
  - How much STM32F3 FLASH is required?
  - How much STM32F3 SRAM is required?



# STM32F3-Discovery Demo

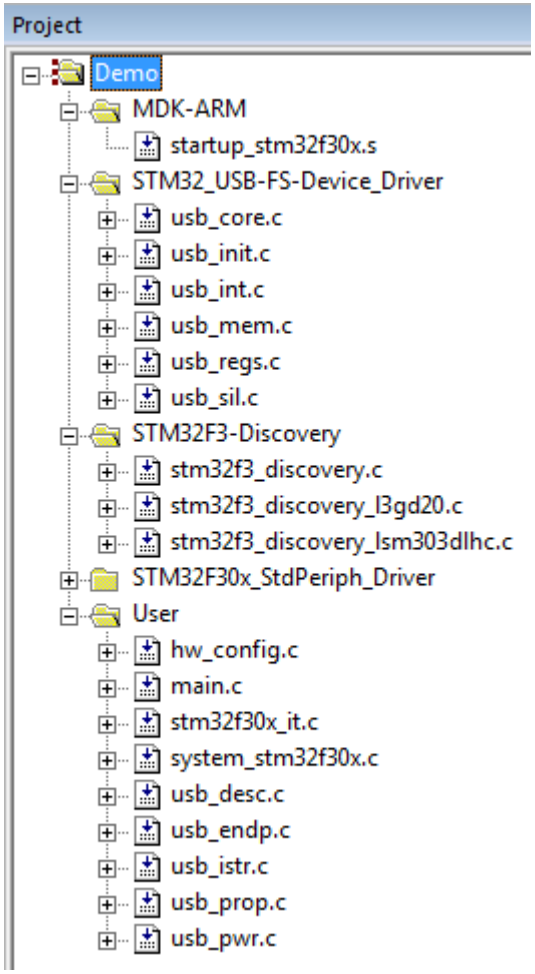
## Firmware Project Overview

life.augmented

# Project Files



- MDK-ARM
  - startup_stm32f30x.s
    - System initialization, vector table, reset and branch to main() (Unique for each 3rd party tool chain)
- STM32_USB-FS-Device_Driver
  - Contains ST FS USB library functions.
- ST,3F3-Discovery
  - Board specific functions
- STM32F30x_StdPeriph_Driver
  - Contains peripheral library functions
- User files
  - main.c (program entry point)
  - system_stm32f3xx.c (initial system configuration)
  - stm32f0xx_it.c (ISR's)
  - usb_xxxx.c (USB interface, not used)




life.augmented

# startup_stm32f30x.s



- Main Characteristics
  - Initializes stack pointer

```
;* Description       : STM32F30x devices vector table for EWARM toolchain.
;*                     This module performs:
;*                     - Set the initial SP
;*                     - Set the initial PC == _iar_program_start,
;*                     - Set the vector table entries with the exceptions ISR
;*                       address.
;*                     - Branches to main in the C library (which eventually
;*                       calls main()).
;*                     After Reset the Cortex-M4 processor is in Thread mode,
;*                     priority is Privileged, and the Stack is set to Main.
;*******************************************************************************
```

  - Contains the vector table for the part

```
        DATA
__vector_table
        DCD     sfe(CSTACK)
        DCD     Reset_Handler             ; Reset Handler

        DCD     NMI_Handler               ; NMI Handler
        DCD     HardFault_Handler         ; Hard Fault Handler
        DCD     MemManage_Handler         ; MPU Fault Handler
        DCD     BusFault_Handler          ; Bus Fault Handler
        DCD     UsageFault_Handler        ; Usage Fault Handler
```

  - Contains Reset handler – called on system reset
    - Calls SystemInit() function
    - Branches to main()

```
;;
;; Default interrupt handlers.
;;
        THUMB
        PUBWEAK Reset_Handler
        SECTION .text:CODE:REORDER(2)
Reset_Handler

        LDR     R0, =SystemInit
        BLX     R0
        LDR     R0, =__iar_program_start
        BX      R0
```

life.augmented

# system_stm32f30x.c

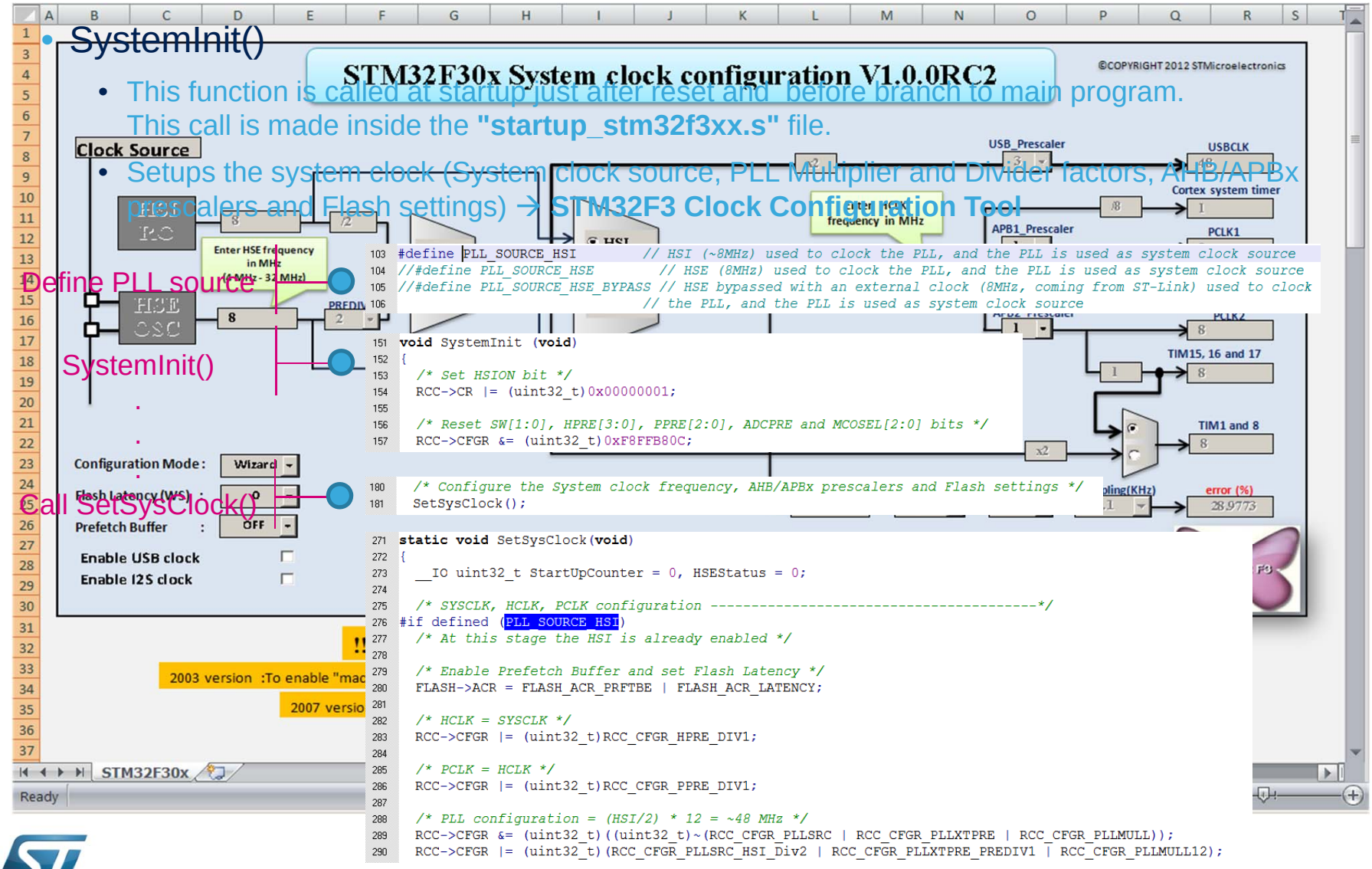


- SystemInit()
  - This function is called at startup just after reset and before branch to main program. This call is made inside the **"startup_stm32f3xx.s"** file.
  - Setups the system clock (System clock source, PLL Multiplier and Divider factors, AHB/APBx prescalers and Flash settings) → **STM32F3 Clock Configuration Tool**

Define PLL source

SystemInit()

Call SetSysClock()




```
103 #define PLL_SOURCE_HSI        // HSI (~8MHz) used to clock the PLL, and the PLL is used as system clock source
104 //#define PLL_SOURCE_HSE       // HSE (8MHz) used to clock the PLL, and the PLL is used as system clock source
105 //#define PLL_SOURCE_HSE_BYPASS // HSE bypassed with an external clock (8MHz, coming from ST-Link) used to clock
106                                 // the PLL, and the PLL is used as system clock source
```

```
151 void SystemInit (void)
152 {
153   /* Set HSION bit */
154   RCC->CR |= (uint32_t)0x00000001;
155
156   /* Reset SW[1:0], HPRE[3:0], PPRE[2:0], ADCPRE and MCOSEL[2:0] bits */
157   RCC->CFGR &= (uint32_t)0xF8FFB80C;
```

```
180   /* Configure the System clock frequency, AHB/APBx prescalers and Flash settings */
181   SetSysClock();
```

```
271 static void SetSysClock(void)
272 {
273   __IO uint32_t StartUpCounter = 0, HSEStatus = 0;
274
275   /* SYSCLK, HCLK, PCLK configuration ----------------------------------------*/
276 #if defined (PLL_SOURCE_HSI)
277   /* At this stage the HSI is already enabled */
278
279   /* Enable Prefetch Buffer and set Flash Latency */
280   FLASH->ACR = FLASH_ACR_PRFTBE | FLASH_ACR_LATENCY;
281
282   /* HCLK = SYSCLK */
283   RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;
284
285   /* PCLK = HCLK */
286   RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE_DIV1;
287
288   /* PLL configuration = (HSI/2) * 12 = ~48 MHz */
289   RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_PLLSRC | RCC_CFGR_PLLXTPRE | RCC_CFGR_PLLMULL));
290   RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_HSI_Div2 | RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLMULL12);
```

life.augmented

# main.c



- Example main()
  - Standard C main() function entry
  - Start of application program
  - What happens each time the USER Button is pushed?
    - Goto Line 99,  while(1)
    - Goto Line 112, LD3-LD10→ Pattern
    - Goto Line 164, LD3-LD10→ Gyro
    - Goto Line 232, LD3-LD10→ Compass

```
/**
  * @brief  Main program.
  * @param  None
  * @retval None
  */
int main(void)
{
  /* SysTick end of count event each 10ms */
  RCC_GetClocksFreq(&RCC_Clocks);
  SysTick_Config(RCC_Clocks.HCLK_Frequency / 100);

  /* Initialize LEDs and User Button available on STM32F3-Discovery board */
  STM_EVAL_LEDInit(LED3);
  STM_EVAL_LEDInit(LED4);
  STM_EVAL_LEDInit(LED5);
  STM_EVAL_LEDInit(LED6);
  STM_EVAL_LEDInit(LED7);
  STM_EVAL_LEDInit(LED8);
  STM_EVAL_LEDInit(LED9);
  STM_EVAL_LEDInit(LED10);
```

# stm32f30x_it.c



- Contains Cortex-M4  Processor Exception Handlers (ISRs)
  - void NMI_Handler(void);
  - void HardFault_Handler(void);
  - void SVC_Handler(void);
  - void PendSV_Handler(void);
  - void SysTick_Handler(void);
- Contains the STM32F30x Peripherals Interrupt Handlers (default is empty)
  - Add the Interrupt Handler for the used peripheral(s) (PPP), for the  available peripheral interrupt handler's name please refer to the startup file: startup_stm32f30x.s
  - Go to Line 148:  SysTick_Handler
    - What is SysTick ISR being used for?
  - Go to Line 166:  EXTI0_IRQHandler
    - What is the ISR use?
    - What physical pin of the STM32F3  is the ‘User Button’ connected to?

# STM32F3-discovery.c



- Contains board specific function and definition
- Defines Push-button and LED GPIO definitions
- Contains board specific functions
  - void STM_EVAL_LEDInit(Led_TypeDef Led);
  - void STM_EVAL_LEDOn(Led_TypeDef Led);
  - void STM_EVAL_LEDOff(Led_TypeDef Led);
  - void STM_EVAL_LEDToggle(Led_TypeDef Led);
  - void STM_EVAL_PBInit(Button_TypeDef Button, ButtonMode_TypeDef Button_Mode);
  - uint32_t STM_EVAL_PBGetState(Button_TypeDef Button);

# stm32f30-discovery_l3gd20.c
# stm32f3-discovery_lsm303dlhc.c



- Each contains the driver information for the MEMs devices on STM32F3-Discovery Board.
- Each contains board specific functions

```
/** @defgroup STM32F3_DISCOVERY_L3GD20_Exported_Functions
  * @{
  */
/* Sensor Configuration Functions */
void L3GD20_Init(L3GD20_InitTypeDef *L3GD20_InitStruct);
void L3GD20_RebootCmd(void);

/*INT1 Interrupt Configuration Functions */
void L3GD20_INT1InterruptCmd(uint8_t InterruptState);
void L3GD20_INT2InterruptCmd(uint8_t InterruptState);
void L3GD20_INT1InterruptConfig(L3GD20_InterruptConfigTypeDef *L3GD20_IntConfigStruct);
uint8_t L3GD20_GetDataStatus(void);

/* High Pass Filter Configuration Functions */
void L3GD20_FilterConfig(L3GD20_FilterConfigTypeDef *L3GD20_FilterStruct);
void L3GD20_FilterCmd(uint8_t HighPassFilterState);
void L3GD20_Write(uint8_t* pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite);
void L3GD20_Read(uint8_t* pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead);
```

# STM32F30x_StdPeriph_Driver



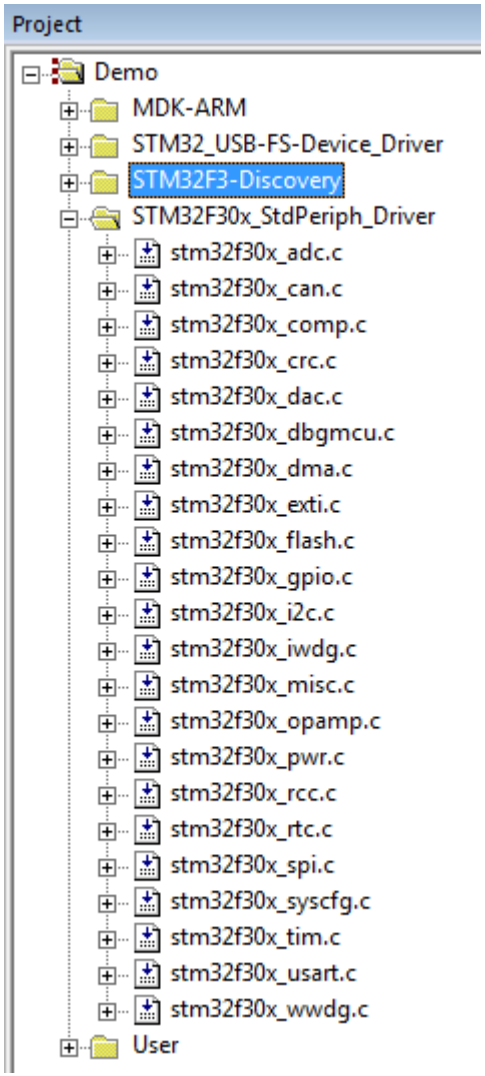
- Each file contains library functions that can be used for each peripheral and gives a standard API for access to peripheral functions.
  - Browse to → main.c, line 83, STM_EVAL_LEDInit(LED5), to investigate GPIO config.
  - Browse to →main.c, line 161, Demo_GyroConfig(), to investigate the Gryo config and the I2C config.

```
void Demo_GyroConfig(void)
{
  L3GD20_InitTypeDef L3GD20_InitStructure;
  L3GD20_FilterConfigTypeDef L3GD20_FilterStructure;

  /* Configure Mems L3GD20 */
  L3GD20_InitStructure.Power_Mode = L3GD20_MODE_ACTIVE;
  L3GD20_InitStructure.Output_DataRate = L3GD20_OUTPUT_DATARATE_1;
  L3GD20_InitStructure.Axes_Enable = L3GD20_AXES_ENABLE;
  L3GD20_InitStructure.Band_Width = L3GD20_BANDWIDTH_4;
  L3GD20_InitStructure.BlockData_Update = L3GD20_BlockDataUpdate_Continous;
  L3GD20_InitStructure.Endianness = L3GD20_BLE_LSB;
  L3GD20_InitStructure.Full_Scale = L3GD20_FULLSCALE_500;
  L3GD20_Init(&L3GD20_InitStructure);

  L3GD20_FilterStructure.HighPassFilter_Mode_Selection =L3GD20_HPM_NORMAL_MODE_RES;
  L3GD20_FilterStructure.HighPassFilter_CutOff_Frequency = L3GD20_HPFCF_0;
  L3GD20_FilterConfig(&L3GD20_FilterStructure) ;

  L3GD20_FilterCmd(L3GD20_HIGHPASSFILTER_ENABLE);
}
```




# Thank you

www.st.com/stm32f3discovery

life.augmented