

Embedded Controller
Programming with Embedded
C

Lesson 2 - LL and Embedded C

Norman McEntire
norman.mcentre@gmail.com

Contents

- Reference Info
- Concepts
- STM32CubeIDE and LL Code Generation
 - Blink LED using LL APIs
- File Tours

Reference Info

https://www.st.com/content/ccc/resource/technical/document/user_manual/63/a8/8f/e3/ca/a1/4c/84/DM00173145.pdf/files/DM00173145.pdf/jcr:content/translations/en.DM00173145.pdf



UM1884 User manual

Description of STM32L4/L4+ HAL and low-layer drivers

Introduction

STMCube™ is STMicroelectronics's original initiative to ease developers' life by reducing development efforts, time and cost. STM32Cube covers the STM32 portfolio.

STM32Cube Version 1.x includes:

- The STM32CubeMX, a graphical software configuration tool that allows generating C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per series (such as STM32CubeL4 for STM32L4 series and STM32L4+ series)
 - The STM32Cube Hardware Abstraction Layer (HAL), an STM32 abstraction layer embedded software ensuring maximized portability across the STM32 portfolio. The HAL is available for all peripherals.
 - The low-layer APIs (LL) offering a fast light-weight expert-oriented layer which is closer to the hardware than the HAL. The LL APIs are available only for a set of peripherals.

Key Sections From Manual

UM1884

Contents

3	Overview of low-layer drivers.....	73
3.1	Low-layer files	73
3.2	Overview of low-layer APIs and naming rules	75
3.2.1	Peripheral initialization functions	75
3.2.2	Peripheral register-level configuration functions	79
4	Cohabiting of HAL and LL	81
4.1	Low-layer driver used in standalone mode.....	81
4.2	Mixed use of low-layer APIs and HAL drivers	81

Concepts

Introduction to LL and Embedded C

- **LL** = Low Layer Interface to Hardware
 - Fast and Lightweight
 - Better optimization, **Less Portability**
 - Expert -Oriented
 - **Require deep knowledge of MPU and peripherals**
 - Closer to hardware than HAL
 - Only for a subset of peripherals
 - Not provide for “complex” peripherals such as USB or SD/MMC
- **Strict ANSI C**
 - Checked with CodeSonar Static Analysis tool
 - Fully Documented and MIRSA-C 2004 Compliant

Features of LL Drivers

Function to:

- Initialize peripherals
 - Based on parameters specified in data structures
- Fill data structures with initial reset values
- De-Initialize Peripherals
- Direct and atomic register access
- Independent from HAL
 - Can be used with or without HAL

Features of LL Drivers

Function to:

- LL Drivers do not implement any processing
- LL Drivers do not need additional memory
- LL Drivers are all about **direct access to peripheral registers**

Low Layer Files

- No configuration file for LL
 - (Unlike HAL which uses configuration file to enable features)
- Include single include file
 - #include “stm32yyxx.h” (the STM32 CMSIS device file)
- stm32l4xx_ll_...
 - All filenames begin with the `stm32l4xx_ll_...` prefix
- Next several slides list specific LL files

stm32l4xx_ll_bus.h

- Core Bus Control
- Peripheral Clock Activation/Deactivation
- Example
 - LL_AHB2_GRP1_EnableClock

stm32l4xx_ll_cortex.h

- Cortex-M related register operations related to:
 - SysTick
 - `LL_SYSTICK_xxxx`
 - Low Power Mode
 - `LL_LPM_xxxx`

stm32l4xx_ll_system.h

- System related operations
 - LL_SYSCFG_xxx
 - LL_DBGMCU_xxx
 - LL_FLASH_xxx
 - LL_VREFBUF_xxx

stm32l4xx_ll_PPP.h/.c

- Peripheral Initialization Functions
 - LL_PPP_Init()
 - LL_PPP_StructInit()
 - LL_PPP_DelInit()

Note: *PPP* is replaced with specific peripheral, e.g. GPIO, SPI, etc.

stm32l4xx_ll_utils.h/.c

- Generic APIs
 - Read of device unique ID and electronic signature
 - Timebase and delay management
 - System clock configuration

Common Initialization Functions

- **LL_PPP_Init()**
- **LL_PPP_StructInit()**
- **LL_PPP_DelInit()**

PPP is
generic
placeholder

More info
on the
following
slides

LL_PPP_Init()

Example

- initialize peripheral main features
 - Example
 - ErrorStatus
- LL_USART_Init(**
- USART_TypeDef *USARTx,
 LL_USART_InitTypeDef *USART_InitStruct)

LL_PPP_StructInit()

Example

- Fill structures with init values
- Example
 - void
LL_USART_StructInit(
 LL_USART_InitTypeDef *USART_InitStruct)

LL_PPP_DelInit()

Example

- DelInit, i.e. restore to default reset values
 - Example
 - ErrorStatus
- LL_USART_DelInit(**
 USART_TypeDef *USARTx)

Additional LL Functions for Selected Peripherals

- **LL_PPP_{Category}_Init()**
 - **LL_ADC_INJ_Init(...)**
 - **LL_RTC_TIME_Init(...)**
 - **LL_RTC_DATE_Init(...)**
 - **LL_TIM_IC_Init(...)**
 - **LL_TIM_ENCODER_Init(...)**

LL Register Level Config Functions - Flags

- Get
 - **LL_PPP_IsActiveFlag_BITNAME**
- Clear
 - **LL_PPP_ClearFlag_BITNAME**

LL Register Level Config Functions - Interrupts

- Enable
 - **LL_PPP_EnableIT_BITNAME**
- Disable
 - **LL_PPP_DisableIT_BITNAME**
- Get
 - **LL_PPP_IsEnabled_BITNAME**

LL Register Level Config Functions - DMA

- Enable
 - **LL_PPP_EnableDMAReq_BITNAME**
- Disable
 - **LL_PPP_DisableDMAReq_BITNAME**
- Get
 - **LL_PPP_IsEnabledDMAReq_BITNAME**

DMA =
Direct
Memory
Access

Cohabiting of LL and HAL

- **LL** can be used in:
 - Stand alone mode (no HAL)
 - HAL = Hardware Abstraction Layer
 - Combined with HAL (but not controlling same peripherals as HAL)
- NOTE: STM provides some examples of using LL and HAL - see **Examples_MIX** projects

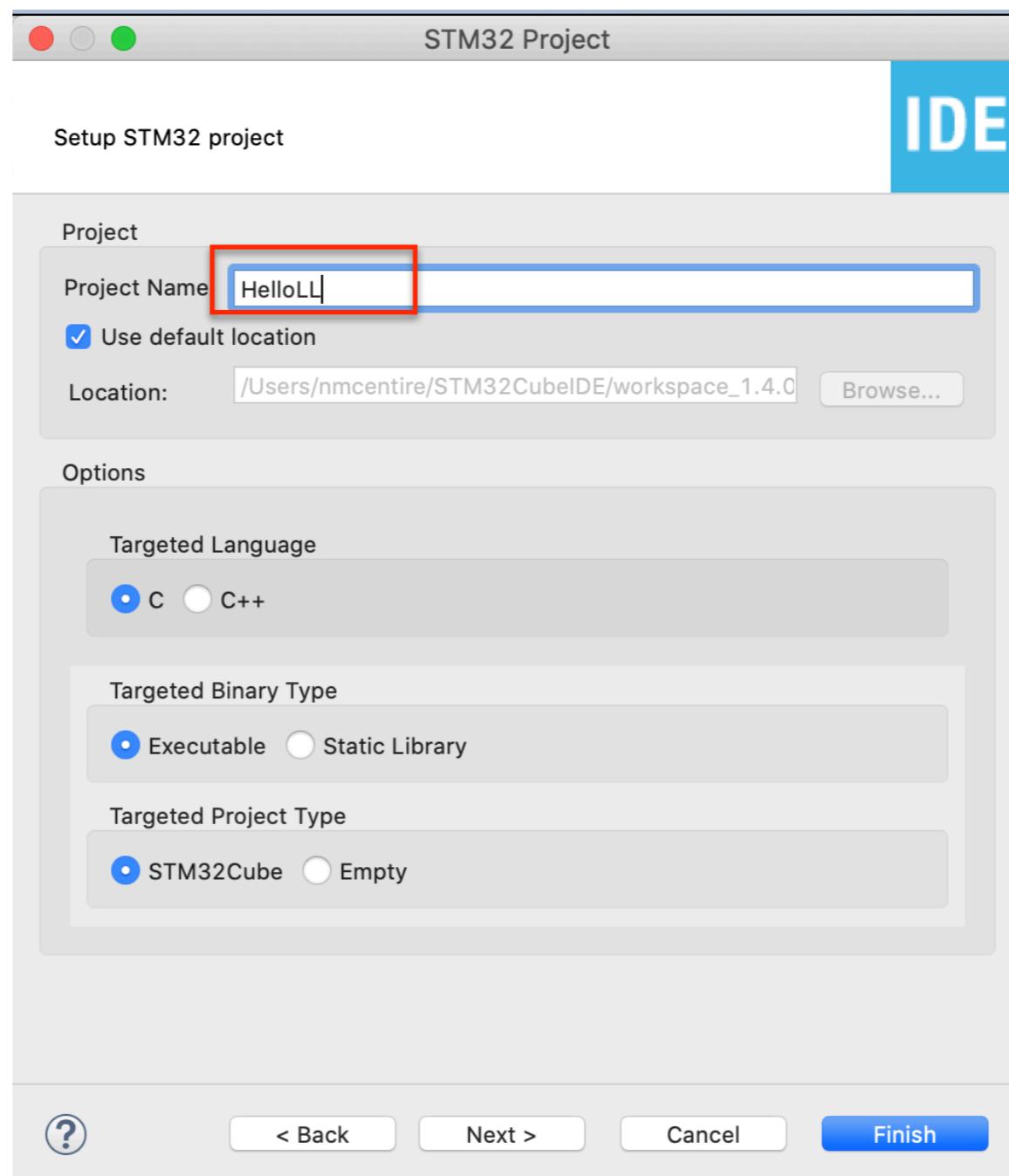
STM32CubelDE and Generation of LL Code

Step 1. Target Board Selector - B-L475-IOT01A1

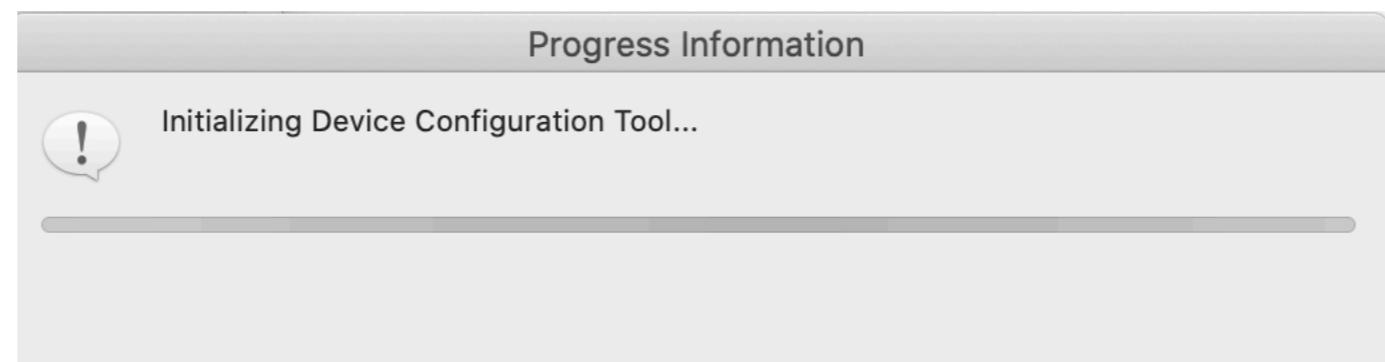
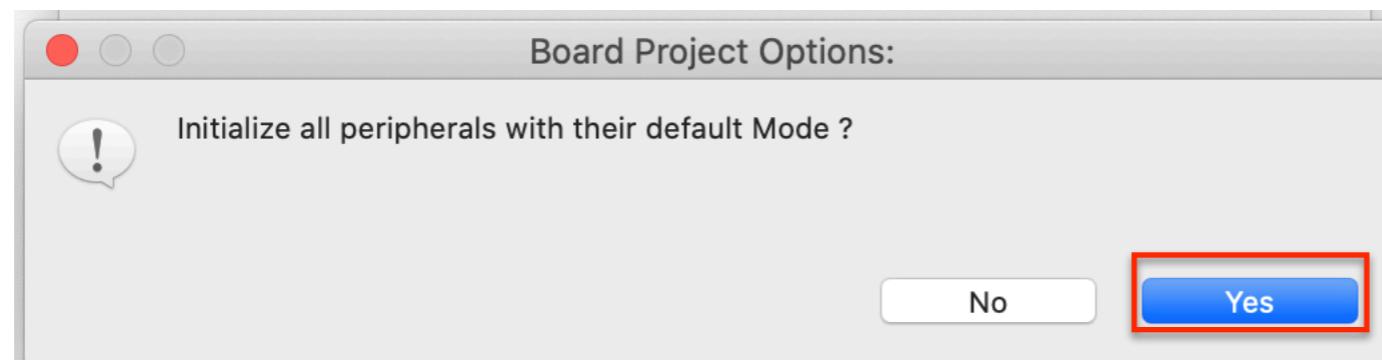
The screenshot shows the STM32 Project IDE interface for target selection. The 'Board Selector' tab is active, indicated by a red box. The main area displays the 'STM32L4 Series' with a 'Boards List: 153 items'. The board 'B-L475E-IOT01A1' is highlighted with a red box.

*	Overview	Commercial Part ...	Type	Marketing Status	Unit Price (US\$)	Mounted Device
★		B-G474E-DPOW1	Discovery Kit	Active	59.0	STM32G474RETx
★		B-L072Z-LRWAN1	Discovery Kit	Active	46.5	STM32L072CZYx
★		B-L462E-CELL1	Discovery Kit	NA	NA	STM32L462REYx
★		B-L475E-IOT01A1	Discovery Kit	Active	53.0	STM32L475VGTx
★		R-L475E-IOT01A2	Discovery Kit	NA	NA	STM32L475VGTx

Step 2. HelloLL



Step 3. Init all peripherals with their default mode



Step 4. Project Manager, Advanced Settings - Defaults

The screenshot shows the Project Manager interface for a project named "HelloLL.ioc". The interface is divided into three main sections: Pinout & Configuration, Clock Configuration, and Project Manager. The Project Manager section is active.

Driver Selector: A dropdown menu labeled "Driver Selector" is open, showing options: GPIO (selected), RCC, DFSDM, I2C, QUADSPI, SPI, USART, and USB_OTG_FS. The "GPIO" option is highlighted with a red box.

Generated Function Calls: This section lists the generated function calls with their ranks, names, instance names, and visibility settings. The table has columns for Rank, Function Name, IP Instance Name, "Do Not Generate Function Call" (checkbox), and "Visibility (Static)" (checkbox).

Rank	Function Name	IP Instance Name	<input type="checkbox"/> Do Not Generate Function Call	<input checked="" type="checkbox"/> Visibility (Static)
1	MX_GPIO_Init	GPIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	SystemClock_Config	RCC	<input type="checkbox"/>	<input type="checkbox"/>
3	MX_DFSDM1_Init	DFSDM1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4	MX_I2C2_Init	I2C2	<input type="checkbox"/>	<input checked="" type="checkbox"/>
5	MX_QUADSPI_Init	QUADSPI	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6	MX_SPI3_Init	SPI3	<input type="checkbox"/>	<input checked="" type="checkbox"/>
7	MX_USART1_UART_Init	USART1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
8	MX_USART3_UART_Init	USART3	<input type="checkbox"/>	<input checked="" type="checkbox"/>
9	MX_USB_OTG_FS_PCDriverInit	USB_OTG_FS	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Step 4. Project Manager, Advanced Settings - Change to LL

The screenshot shows the Project Manager interface for a project named "HelloLL.ioc". The interface is divided into three main tabs: Pinout & Configuration, Clock Configuration, and Project Manager. The Project Manager tab is active.

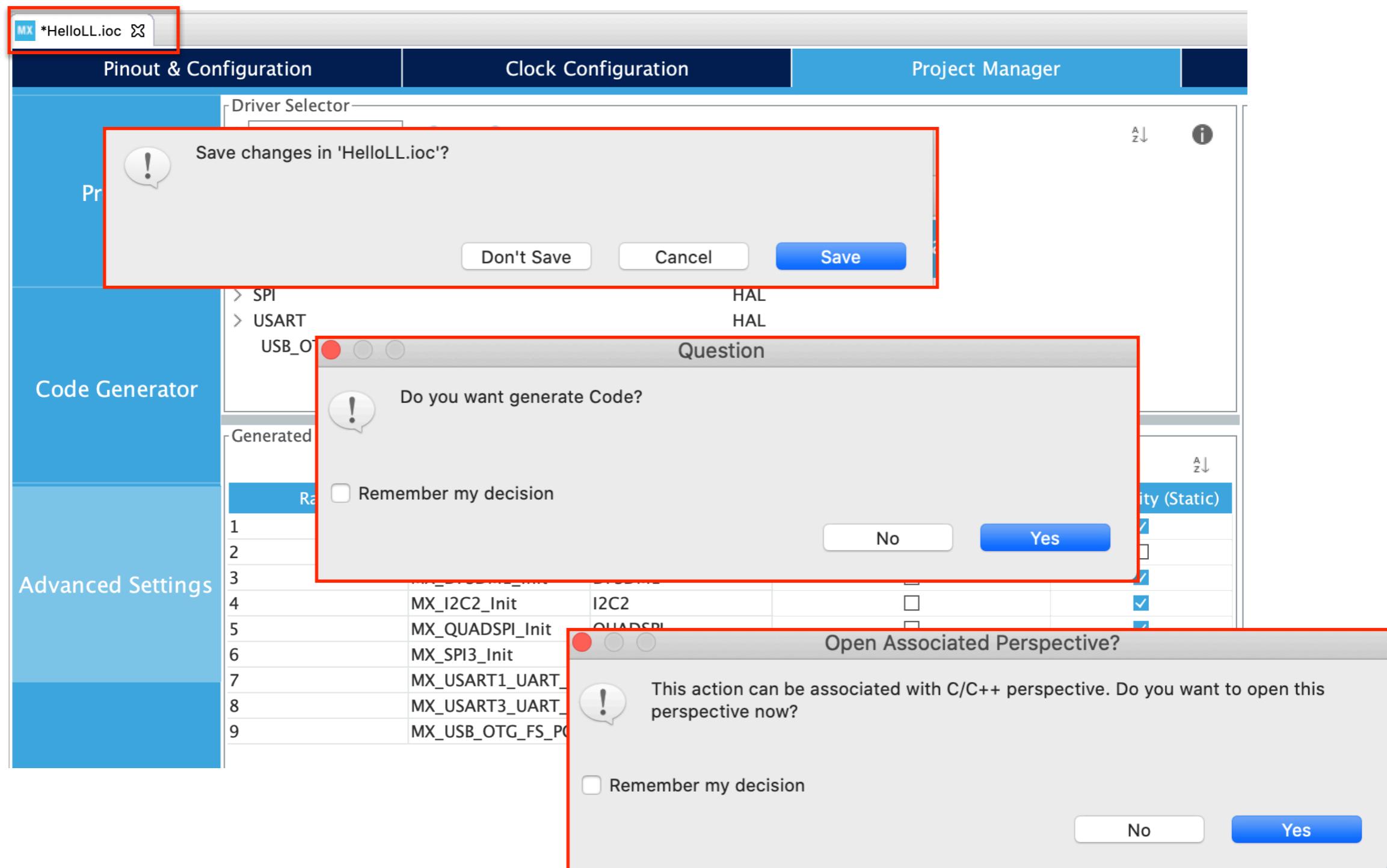
Driver Selector: A table showing driver configurations. The "Project" column lists drivers: GPIO, RCC, DFSDM, I2C, QUADSPI, SPI, USART, and USB_OTG_FS. The "Code Generator" column lists their corresponding HAL or LL implementations. The "LL" implementation for all drivers is highlighted with a red box.

Project	Code Generator
GPIO	HAL
RCC	HAL
> DFSDM	HAL
> I2C	HAL
QUADSPI	HAL
> SPI	HAL
> USART	HAL
USB_OTG_FS	HAL

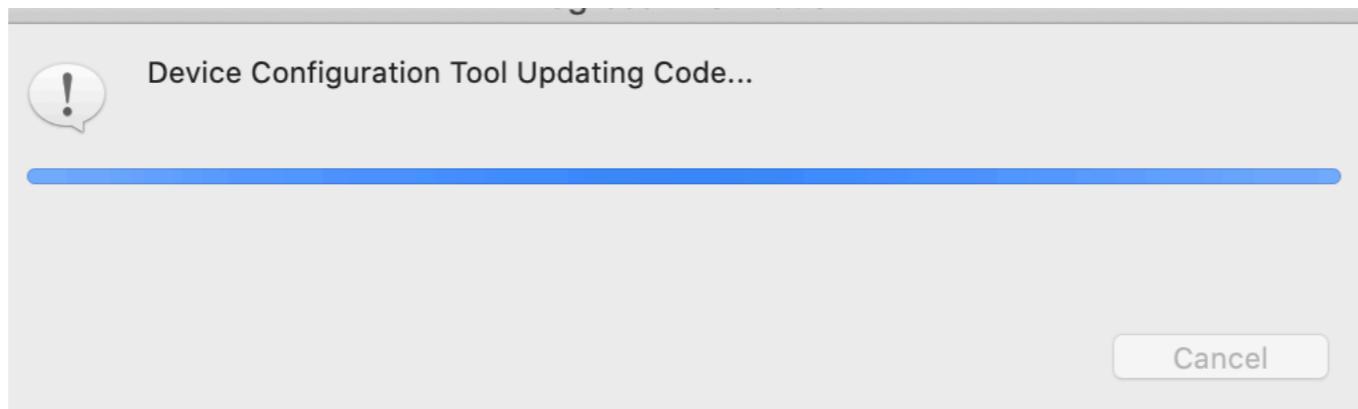
Generated Function Calls: A table listing generated function calls with their ranks, names, instance names, and visibility settings.

Rank	Function Name	IP Instance Name	<input type="checkbox"/> Do Not Generate Function Call	<input type="checkbox"/> Visibility (Static)
1	MX_GPIO_Init	GPIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	SystemClock_Config	RCC	<input type="checkbox"/>	<input type="checkbox"/>
3	MX_DFSDM1_Init	DFSDM1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4	MX_I2C2_Init	I2C2	<input type="checkbox"/>	<input checked="" type="checkbox"/>
5	MX_QUADSPI_Init	QUADSPI	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6	MX_SPI3_Init	SPI3	<input type="checkbox"/>	<input checked="" type="checkbox"/>
7	MX_USART1_UART_Init	USART1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
8	MX_USART3_UART_Init	USART3	<input type="checkbox"/>	<input checked="" type="checkbox"/>
9	MX_USB_OTG_FS_PCDriverInit	USB_OTG_FS	<input type="checkbox"/>	<input checked="" type="checkbox"/>

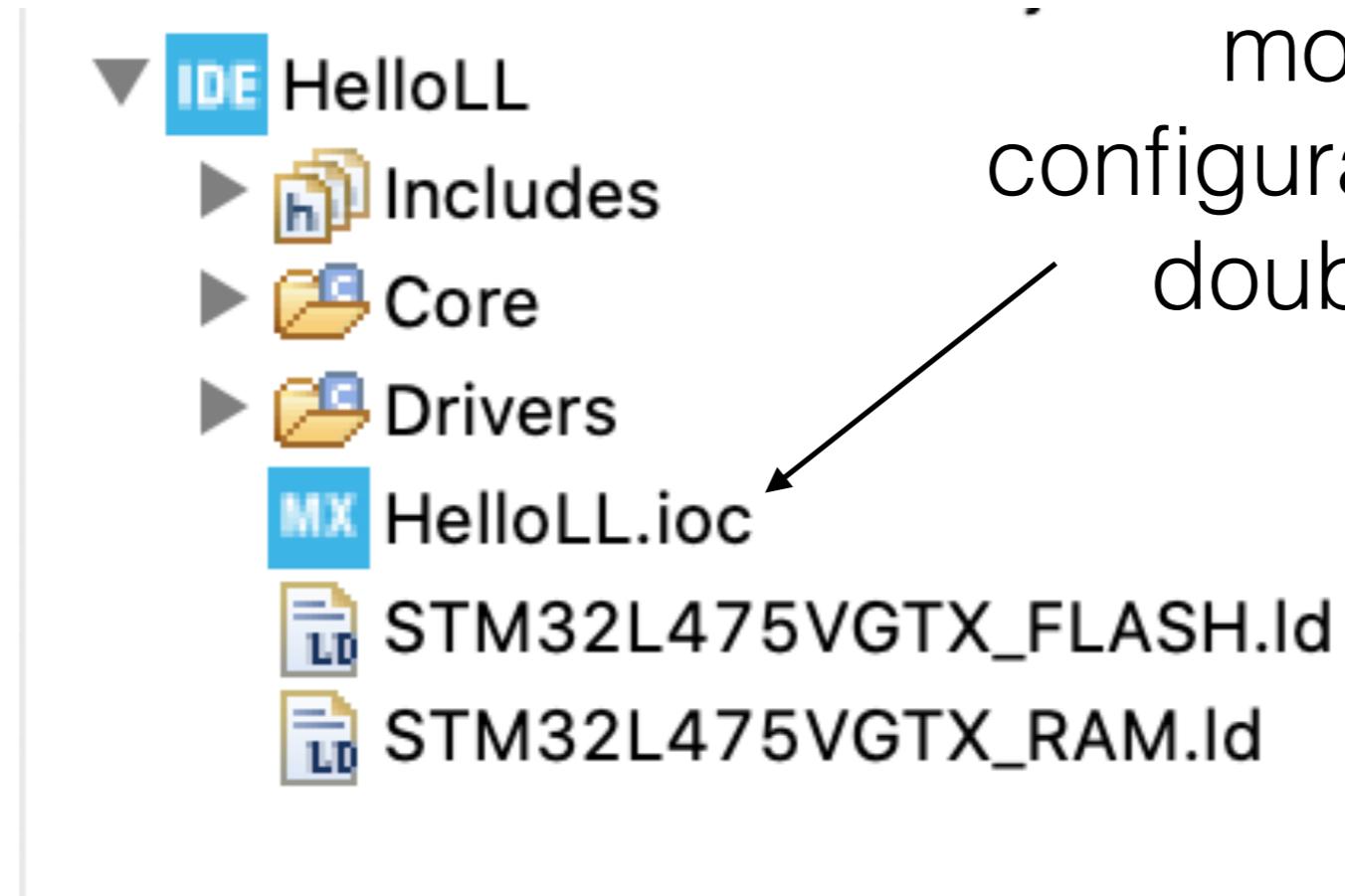
Step 5. Save Settings



Step 6. Wait for code generation



Step 7. Results



If you need to
modify the I/O
configuration we just did,
double-click here

Step 8. main.c

You can safely edit code in the range
USER CODE BEGIN and USER CODE END
(will not be replaced if code regenerated)

```
1  /* USER CODE BEGIN Header */
2  /**
3   ****
4   * @file          : main.c
5   * @brief         : Main program body
6   ****
7   * @attention
8   *
9   * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
10  * All rights reserved.</center></h2>
11  *
12  * This software component is licensed by ST under BSD 3-Clause license,
13  * the "License"; You may not use this file except in compliance with the
14  * License. You may obtain a copy of the License at:
15  *           opensource.org/licenses/BSD-3-Clause
16  *
17  ****
18  */
19  /* USER CODE END Header */
```

Step 9. main.c

Be sure to place your additional Header files (e.g. #include <string.h>) Between the USER CODE BEGIN and USER_CODE_END sections

```
-- /* Includes -----  
20  /* Includes -----  
21  #include "main.h"  
22  
23 ⊕/* Private includes -----  
24  /* USER CODE BEGIN Includes */  
25  
26  /* USER CODE END Includes */  
27
```

Step 10. main.c

Your typedefs , #defines, macros

```
--  
28④/* Private typedef -----  
29 /* USER CODE BEGIN PTD */  
30  
31 /* USER CODE END PTD */  
32  
33④/* Private define -----  
34 /* USER CODE BEGIN PD */  
35 /* USER CODE END PD */  
36  
37④/* Private macro -----  
38 /* USER CODE BEGIN PM */  
39  
40 /* USER CODE END PM */  
41
```

main.c

```
2:  */
3:  int main(void)
4:  {
5:    /* USER CODE BEGIN 1 */
6:
7:    /* USER CODE END 1 */
8:
9:    /* MCU Configuration-----*/
10:
11:   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
12:   HAL_Init();
13:
14:   /* USER CODE BEGIN Init */
15:
16:   /* USER CODE END Init */
17:
18:   /* Configure the system clock */
19:   SystemClock_Config();
20:
21:   /* USER CODE BEGIN SysInit */
22:
23:   /* USER CODE END SysInit */
24:
25:   /* Initialize all configured peripherals */
26:   MX_GPIO_Init();
27:   MX_DFSDM1_Init();
28:   MX_I2C2_Init();
29:   MX_QUADSPI_Init();
30:   MX_SPI3_Init();
31:   MX_USART1_UART_Init();
32:   MX_USART3_UART_Init();
33:   MX_USB_OTG_FS_PCD_Init();
34:   /* USER CODE BEGIN 2 */
35:
36:   /* USER CODE END 2 */
37:   /* Infinite loop */
38:   /* USER CODE BEGIN WHILE */
39:   while (1)
40:   {
41:     /* USER CODE END WHILE */
42:
43:     /* USER CODE BEGIN 3 */
44:   }
45:   /* USER CODE END 3 */
```

Pre-written initialization code

Put your code here

main.c

```
/*
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    LL EXTI_InitTypeDef EXTI_InitStruct = {0};
    LL GPIO_InitTypeDef GPIO_InitStruct = {0};

    /* GPIO Ports Clock Enable */
    LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOE);
    LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOC);
    LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOA);
    LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOB);
    LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOD);

    /**
     * @brief Reset output pins
     */
    LL_GPIO_ResetOutputPin(GPIOE, M24SR64_Y_RF_DISABLE_Pin|M24SR64_Y_GPO_Pin|ISM43362_RST_Pin);

    /**
     * @brief Reset output pins
     */
    LL_GPIO_ResetOutputPin(GPIOA, ARD_D10_Pin|SPBTLE_RF_RST_Pin|ARD_D9_Pin);

    /**
     * @brief Reset output pins
     */
    LL_GPIO_ResetOutputPin(GPIOB, ARD_D8_Pin|ISM43362_BOOT0_Pin|ISM43362_WAKEUP_Pin|LED2_Pin
                           |SPSGRF_915_SDN_Pin|ARD_D5_Pin);
}
```

Notice use of LL_ APIs

Step 11. main.c

Use LL APIs to toggle LED

```
118     /* Infinite loop */
119     /* USER CODE BEGIN WHILE */
120     while (1)
121     {
122         LL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
123
124         LL_mDelay(1000);
125
126     /* USER CODE END WHILE */
127
128     /* USER CODE BEGIN 3 */
129 }
130 /* USER CODE END 3 */
131 }
```

Step 12. main.c

Exploring implementation of
LL_GPIO_TogglePin

```
1014 */  
1015 _STATIC_INLINE void LL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint32_t PinMask)  
1016 {  
1017     uint32_t odr = READ_REG(GPIOx->ODR);  
1018     WRITE_REG(GPIOx->BSRR, ((odr & PinMask) << 16u) | (~odr & PinMask));  
1019 }  
1020
```

ODR = Output Direction Register

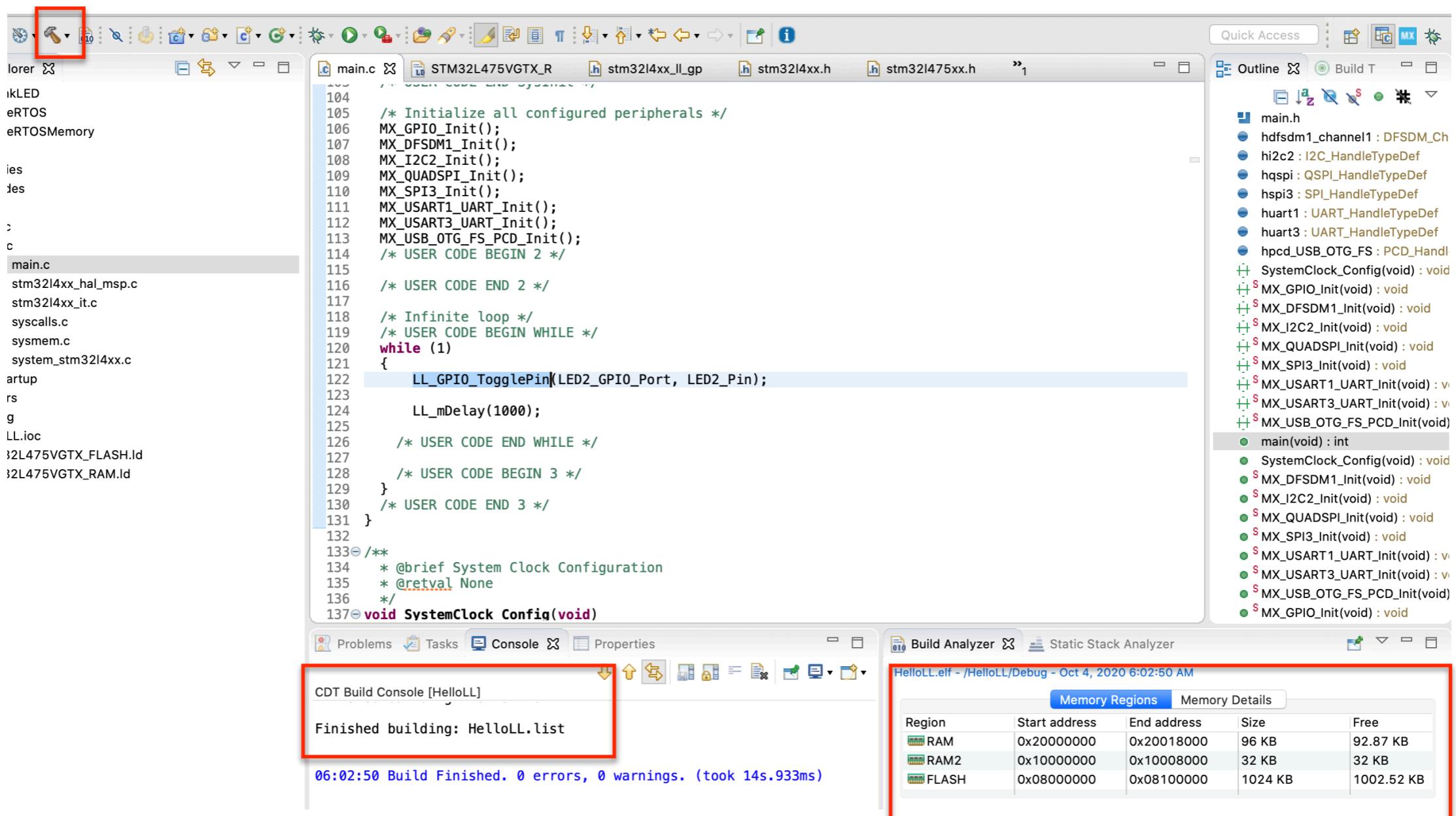
BSSR = Bit Set Reset Register

Step 13. main.c

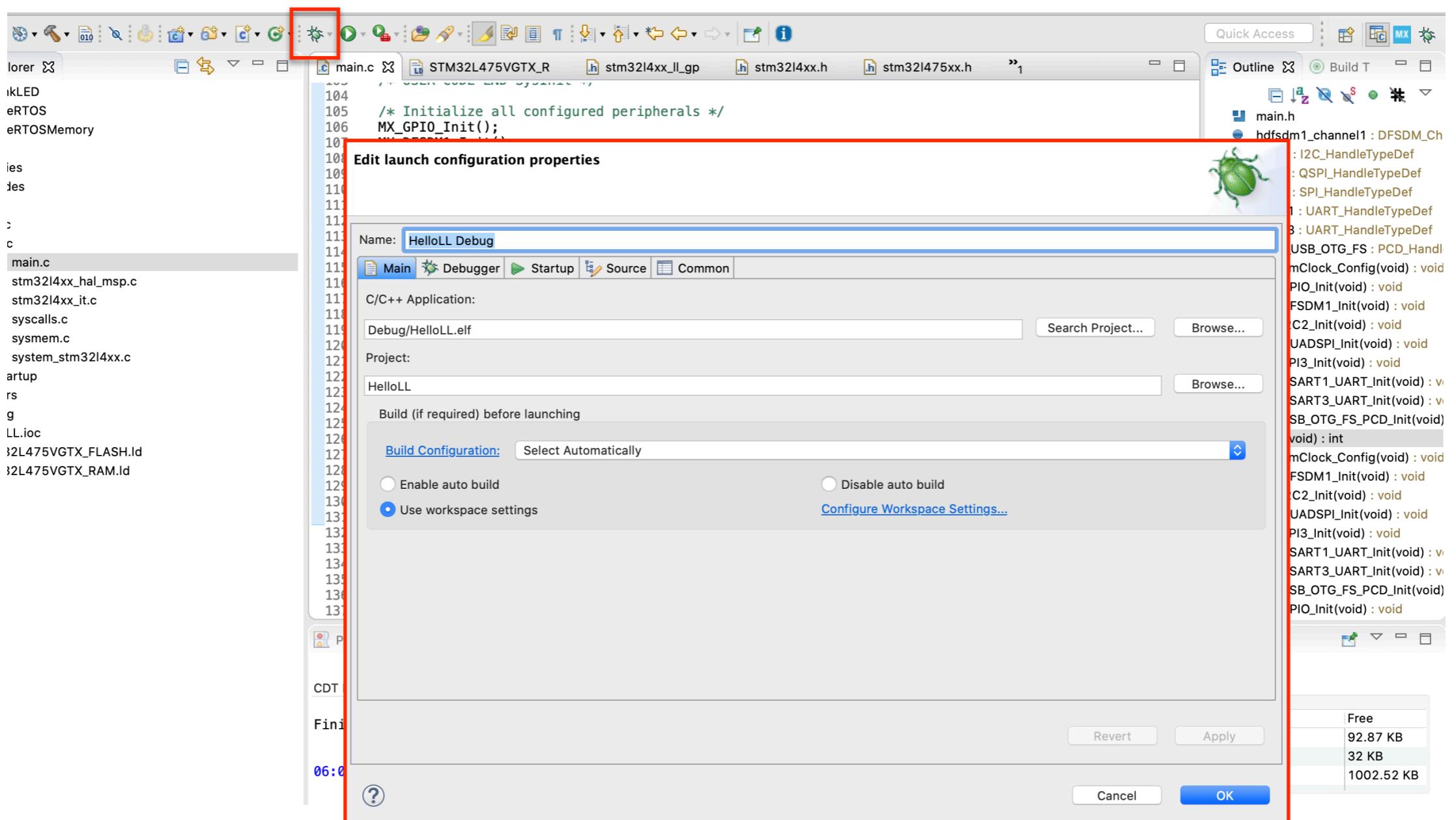
GPIO_TypeDef

```
527 /**
528  * @brief General Purpose I/O
529  */
530
531 typedef struct
532 {
533     __IO uint32_t MODER;      /*!< GPIO port mode register,          Address offset: 0x00      */
534     __IO uint32_t OTYPER;    /*!< GPIO port output type register,  Address offset: 0x04      */
535     __IO uint32_t OSPEEDR;   /*!< GPIO port output speed register, Address offset: 0x08      */
536     __IO uint32_t PUPDR;    /*!< GPIO port pull-up/pull-down register, Address offset: 0x0C      */
537     __IO uint32_t IDR;      /*!< GPIO port input data register,   Address offset: 0x10      */
538     __IO uint32_t ODR;      /*!< GPIO port output data register,  Address offset: 0x14      */
539     __IO uint32_t BSRR;     /*!< GPIO port bit set/reset register, Address offset: 0x18      */
540     __IO uint32_t LCKR;     /*!< GPIO port configuration lock register, Address offset: 0x1C      */
541     __IO uint32_t AFR[2];   /*!< GPIO alternate function registers, Address offset: 0x20-0x24 */
542     __IO uint32_t BRR;      /*!< GPIO Bit Reset register,        Address offset: 0x28      */
543     __IO uint32_t ASCR;    /*!< GPIO analog switch control register, Address offset: 0x2C      */
544
545 } GPIO_TypeDef;
```

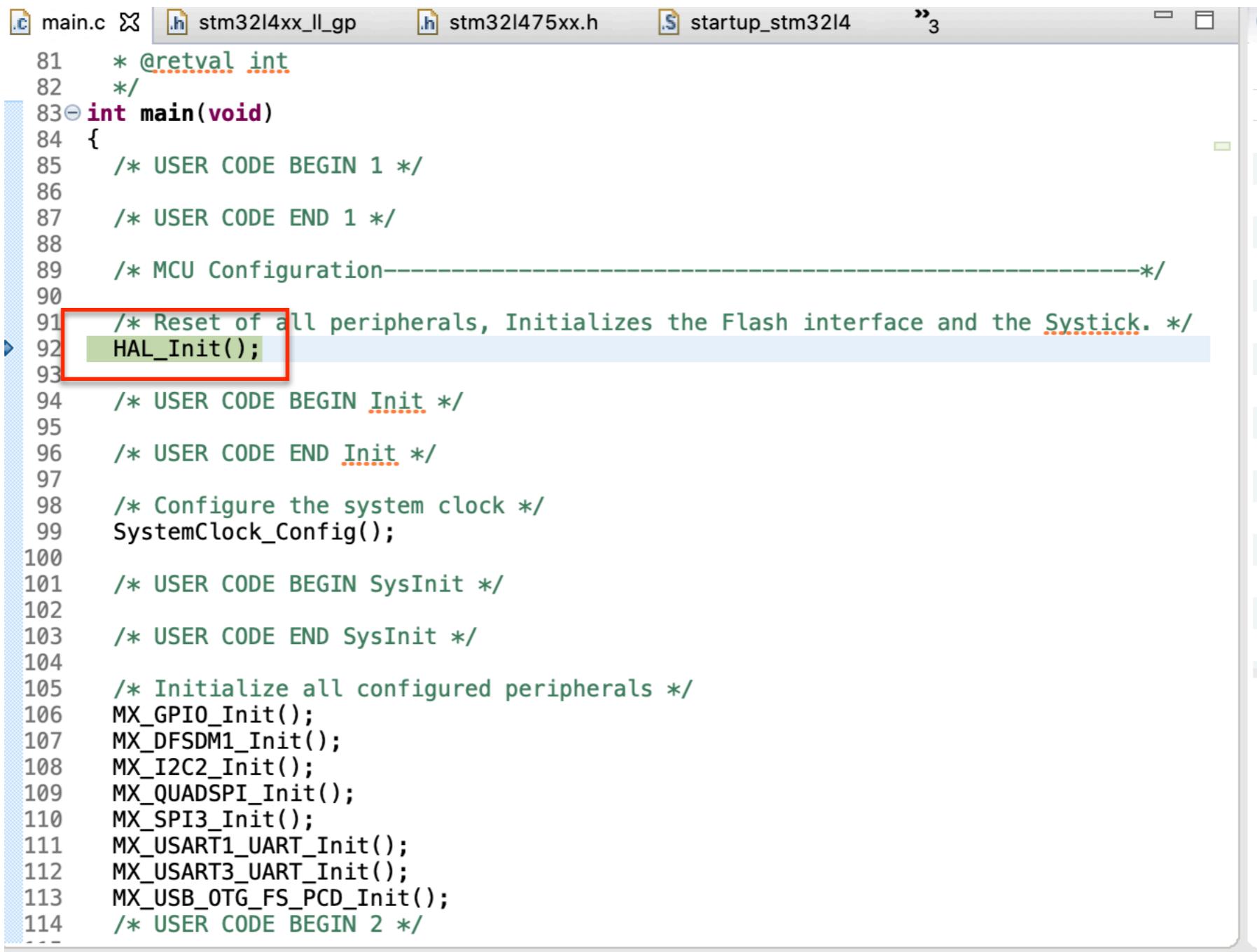
Step 14. Build Code



Step 15. Run Code



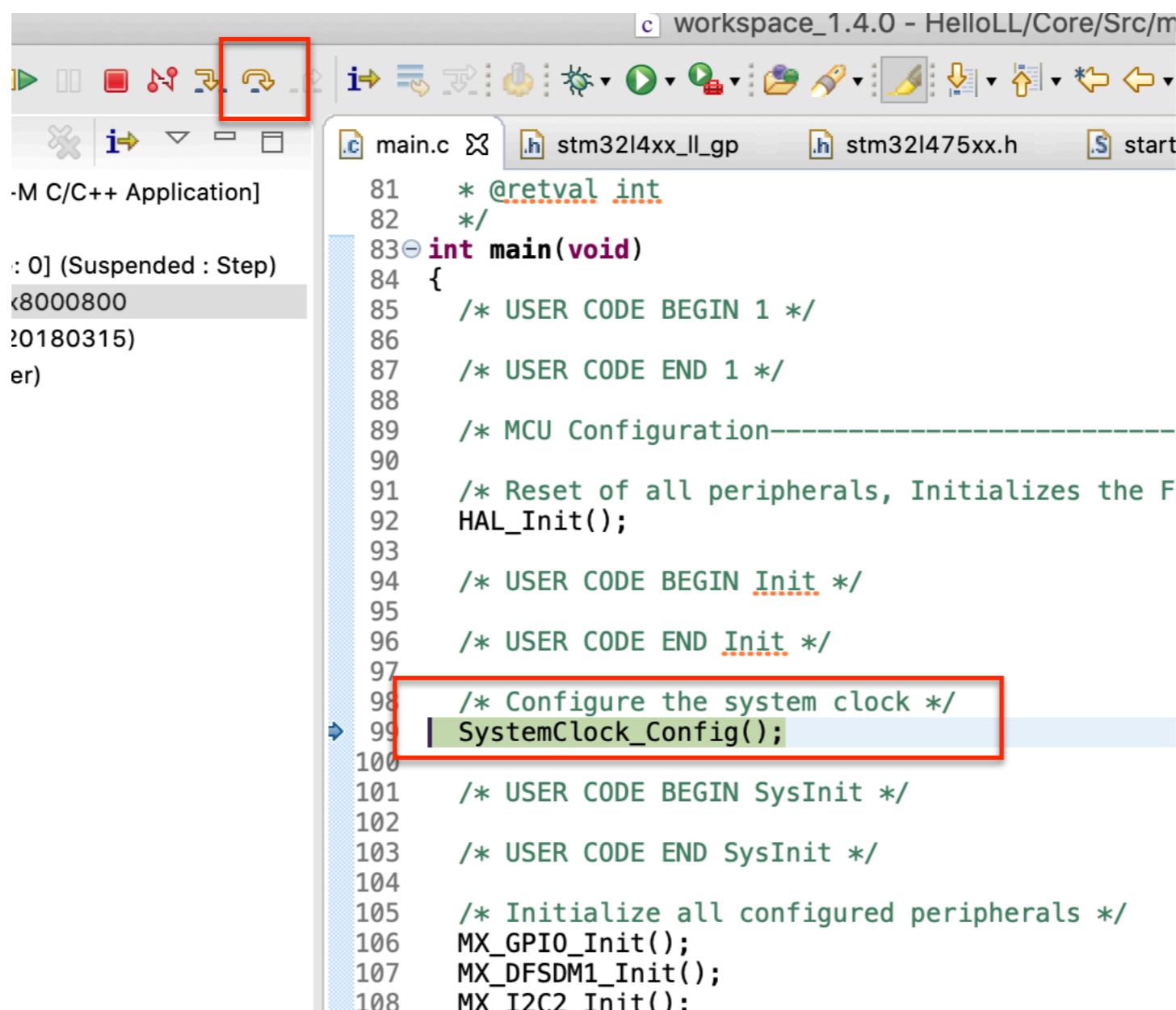
Step 16. Hit Breakpoint



```
main.c | stm32l4xx_ll_gp | stm32l475xx.h | startup_stm32l4 | »3
```

```
81     * @retval int
82     */
83 ⊞ int main(void)
84 {
85     /* USER CODE BEGIN 1 */
86
87     /* USER CODE END 1 */
88
89     /* MCU Configuration-----*/
90
91     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
92     HAL_Init(); // Line 92 is highlighted with a red rectangle
93
94     /* USER CODE BEGIN Init */
95
96     /* USER CODE END Init */
97
98     /* Configure the system clock */
99     SystemClock_Config();
100
101    /* USER CODE BEGIN SysInit */
102
103    /* USER CODE END SysInit */
104
105   /* Initialize all configured peripherals */
106   MX_GPIO_Init();
107   MX_DFSDM1_Init();
108   MX_I2C2_Init();
109   MX_QUADSPI_Init();
110   MX_SPI3_Init();
111   MX_USART1_UART_Init();
112   MX_USART3_UART_Init();
113   MX_USB_OTG_FS_PCD_Init();
114   /* USER CODE BEGIN 2 */
...
```

Step 17. Step Over

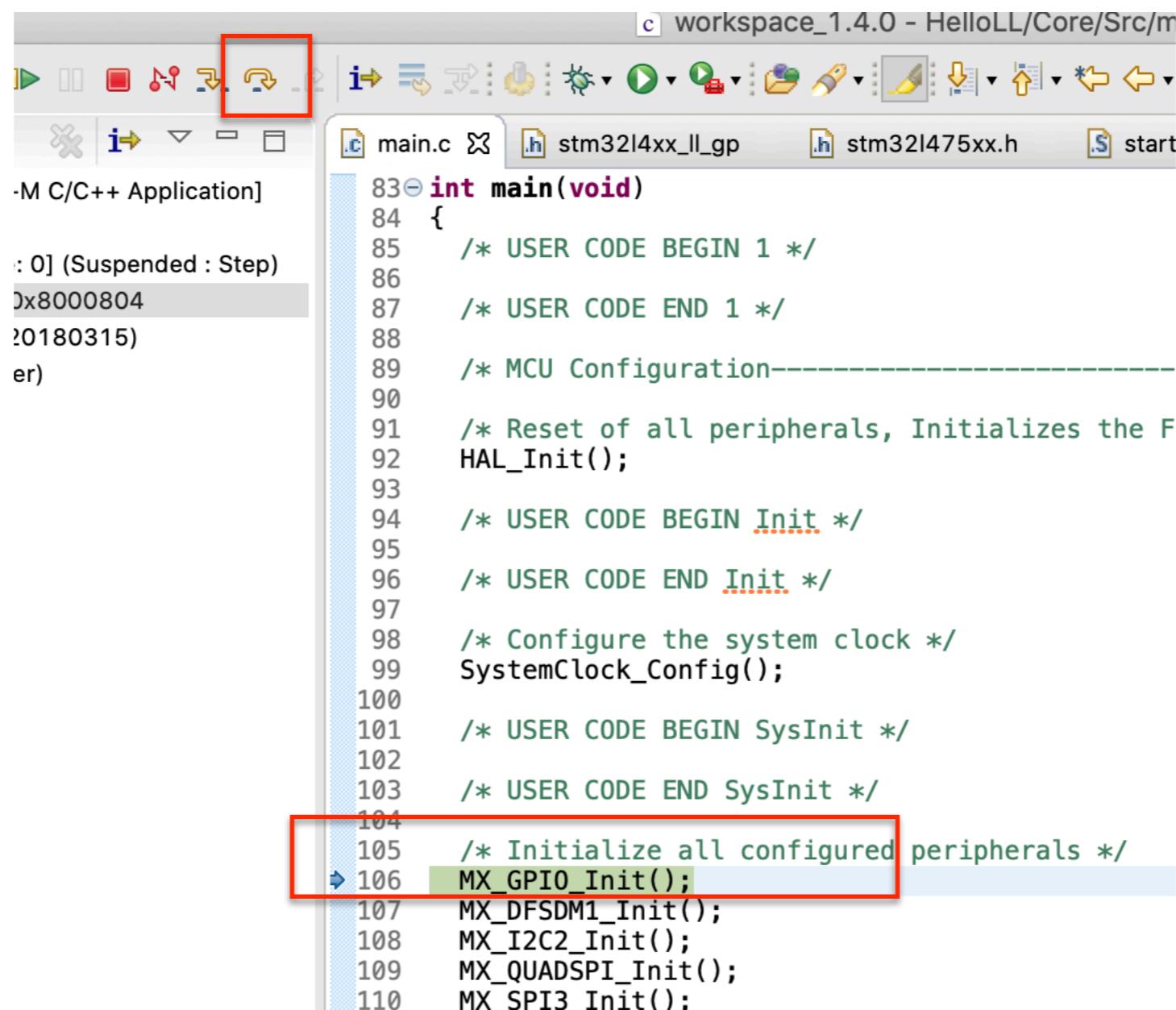


The screenshot shows the Keil MDK-ARM IDE interface. The title bar reads "workspace_1.4.0 - HelloLL/Core/Src/m". The toolbar has various icons for file operations, build, and debug. The main window displays the "main.c" source code. The code is as follows:

```
81     * @retval int
82     */
83     int main(void)
84     {
85         /* USER CODE BEGIN 1 */
86
87         /* USER CODE END 1 */
88
89         /* MCU Configuration-----*/
90
91         /* Reset of all peripherals, Initializes the F
92         HAL_Init();
93
94         /* USER CODE BEGIN Init */
95
96         /* USER CODE END Init */
97
98         /* Configure the system clock */
99         SystemClock_Config();
100
101        /* USER CODE BEGIN SysInit */
102
103        /* USER CODE END SysInit */
104
105        /* Initialize all configured peripherals */
106        MX_GPIO_Init();
107        MX_DFSDM1_Init();
108        MX_I2C2_Init();
```

The line "SystemClock_Config();" is highlighted with a red box, indicating it is the current instruction being stepped over. The status bar at the bottom left shows the memory address "0x8000800" and the time "20180315".

Step 18. Step Over Again

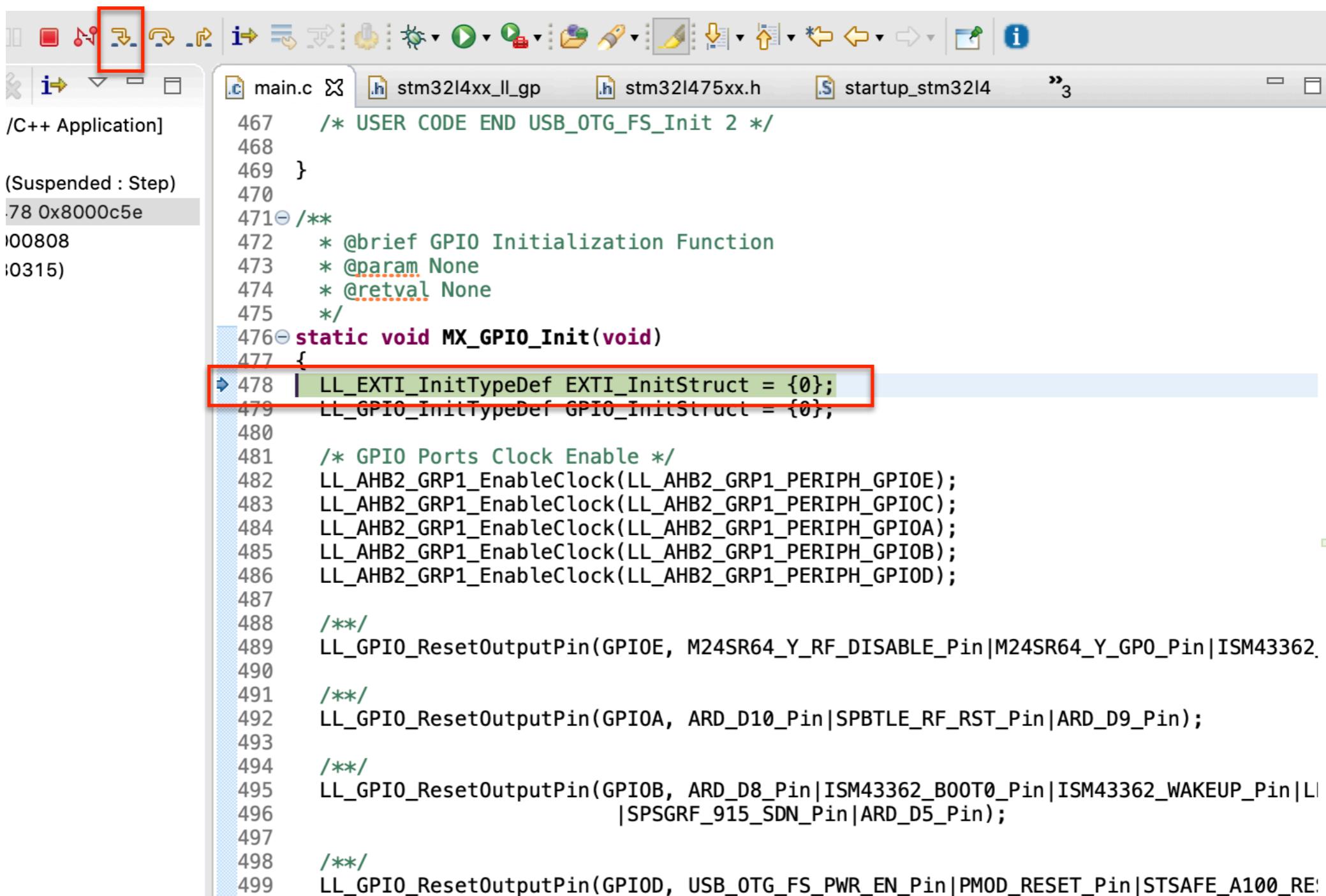


The screenshot shows a debugger interface with the title bar "workspace_1.4.0 - HelloLL/Core/Src/main.c". The toolbar has several icons, and the "Step Over" icon (a blue arrow) is highlighted with a red box. The code editor shows the main.c file with the following content:

```
83 int main(void)
84 {
85     /* USER CODE BEGIN 1 */
86
87     /* USER CODE END 1 */
88
89     /* MCU Configuration-----*/
90
91     /* Reset of all peripherals, Initializes the F
92     HAL_Init();
93
94     /* USER CODE BEGIN Init */
95
96     /* USER CODE END Init */
97
98     /* Configure the system clock */
99     SystemClock_Config();
100
101    /* USER CODE BEGIN SysInit */
102
103    /* USER CODE END SysInit */
104
105    /* Initialize all configured peripherals */
106    MX_GPIO_Init();
107    MX_DFSDM1_Init();
108    MX_I2C2_Init();
109    MX_QUADSPI_Init();
110    MX_SPI3_Init();
```

The line "MX_GPIO_Init();" is highlighted with a red box, indicating it is the current instruction being stepped over.

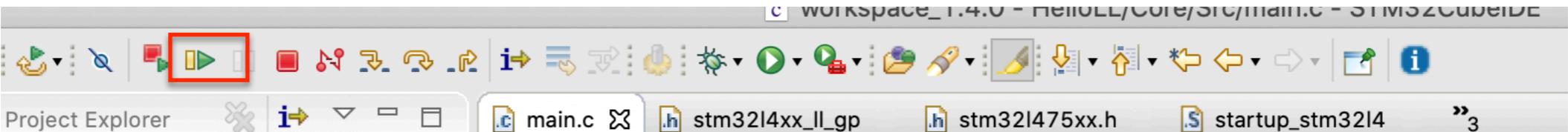
Step 19. Step Into



The screenshot shows a debugger interface with the 'Step Into' button highlighted by a red box. The code editor displays a C++ application with the file 'main.c' open. The cursor is positioned at line 478, which contains the code 'LL EXTI_InitTypeDef EXTI_InitStruct = {0};'. This line is highlighted with a green rectangular selection. The code block is part of the 'MX_GPIO_Init(void)' function, which is also highlighted with a blue rectangular selection. The stack trace on the left shows the current state as '(Suspended : Step)' at address '0x8000c5e'. The assembly code window shows the instruction at address 008080315.

```
467     /* USER CODE END USB_OTG_FS_Init 2 */
468
469 }
470
471 /**
472  * @brief GPIO Initialization Function
473  * @param None
474  * @retval None
475 */
476 static void MX_GPIO_Init(void)
477 {
478     LL EXTI_InitTypeDef EXTI_InitStruct = {0};
479     LL GPIO_InitTypeDef GPIO_InitStruct = {0};
480
481     /* GPIO Ports Clock Enable */
482     LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOE);
483     LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOC);
484     LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOA);
485     LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOB);
486     LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOD);
487
488     /**
489     LL_GPIO_ResetOutputPin(GPIOE, M24SR64_Y_RF_DISABLE_Pin|M24SR64_Y_GPO_Pin|ISM43362_
490
491     /**
492     LL_GPIO_ResetOutputPin(GPIOA, ARD_D10_Pin|SPBTLE_RF_RST_Pin|ARD_D9_Pin);
493
494     /**
495     LL_GPIO_ResetOutputPin(GPIOB, ARD_D8_Pin|ISM43362_BOOT0_Pin|ISM43362_WAKEUP_Pin|L_
496             |SPSGRF_915_SDN_Pin|ARD_D5_Pin);
497
498     /**
499     LL_GPIO_ResetOutputPin(GPIOD, USB_OTG_FS_PWR_EN_Pin|PMOD_RESET_Pin|STSAFE_A100_RE
```

Step 20. Resume

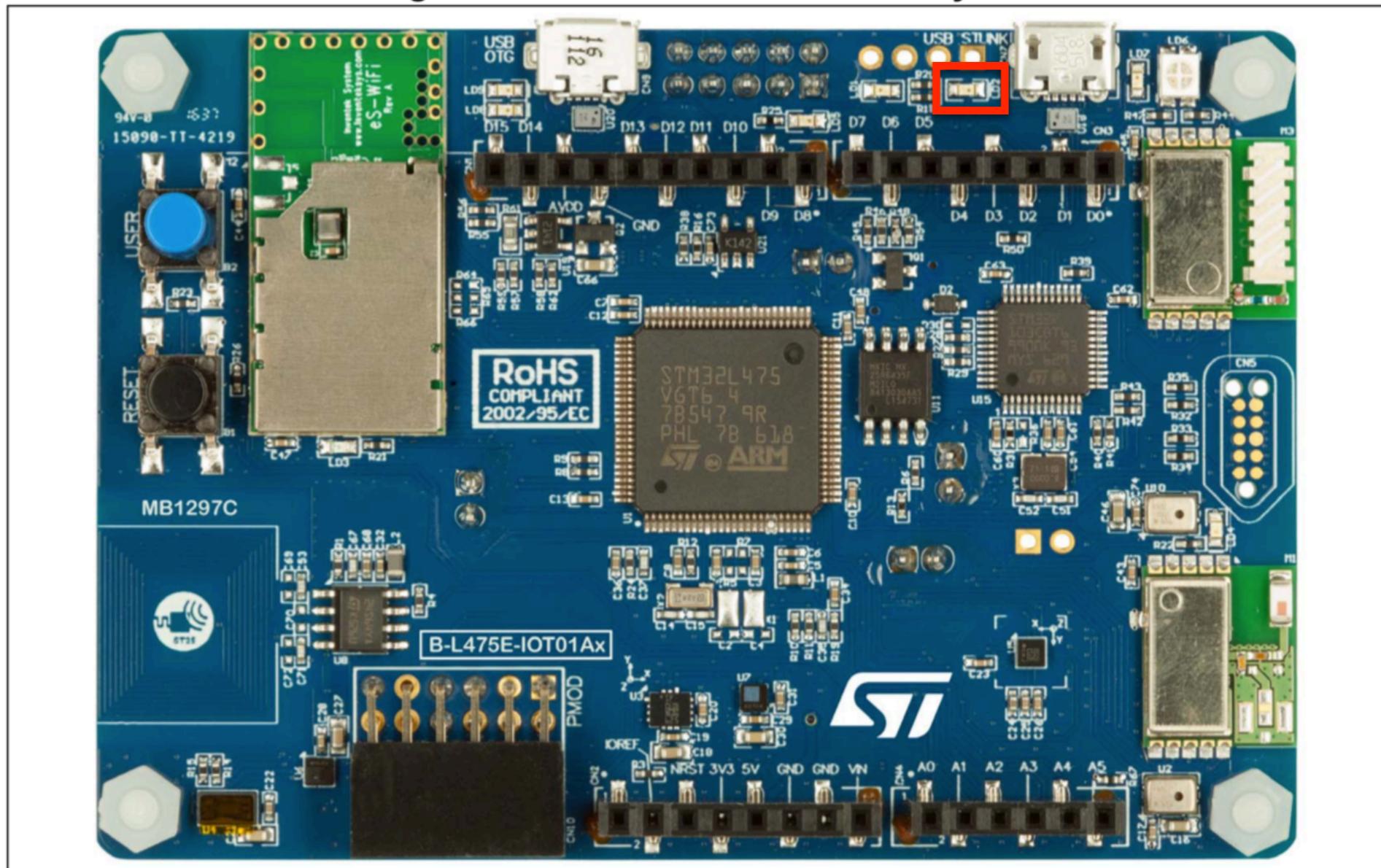


The screenshot shows the STM32CubeIDE interface. The top bar displays "WORKSPACE_1.4.0 - FILE\ROLL\CORE\SRC\main.c - STM32CubeIDE". The toolbar has various icons for file operations, build, and debug. The "Run" icon (play button) is highlighted with a red box. The "Project Explorer" view shows a file named "rg [STM32 Cortex-M C/C++ Application]" with a core count of 0, and a stack trace for a suspended step. The code editor shows the "main.c" file with several initialization functions. A cursor is positioned at the start of the "MX_GPIO_Init(void)" function. The code includes comments for GPIO initialization and clock enable.

```
467     /* USER CODE END USB_OTG_FS_Init 2 */
468
469 }
470
471 /**
472  * @brief GPIO Initialization Function
473  * @param None
474  * @retval None
475 */
476 static void MX_GPIO_Init(void)
477 {
478     LL EXTI_InitTypeDef EXTI_InitStruct = {0};
479     LL GPIO_InitTypeDef GPIO_InitStruct = {0};
480
481     /* GPIO Ports Clock Enable */
482     LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOE);
483     LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOC);
484     LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOA);
485     LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOB);
486     LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOD);
487
488     /**
489     LL_GPIO_ResetOutputPin(GPIOE, M24SR64_Y_RF_DISABLE_Pin|M24SR64_Y_GP(
490
491     /**
492     LL_GPIO_ResetOutputPin(GPIOA, ARD_D10_Pin|SPBTLE_RF_RST_Pin|ARD_D9_I
493
494     /**
495     LL_GPIO_ResetOutputPin(GPIOB, ARD_D8_Pin|ISM43362_B00T0_Pin|ISM43361
496     ISM43361_S01_S02_S03_S04_S05_S06_S07_S08_S09_S0A_S0B_S0C_S0D_S0E_S0F_S0
497 }
```

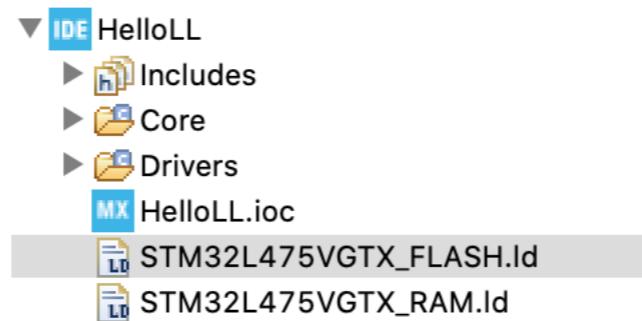
Results - Code Blinking LED use LL APIs

Figure 1. B-L475E-IOT01A Discovery kit

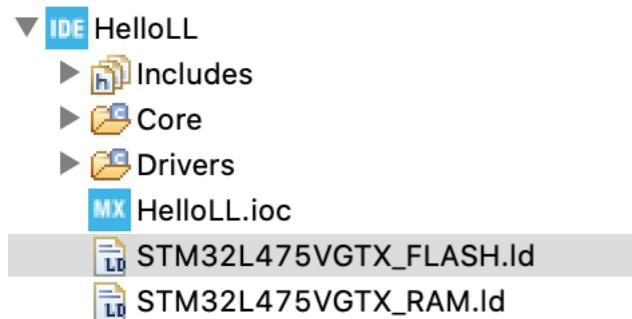


File Tour

Linker Scripts



Step 1. Linker Script - Flash



The screenshot shows the STM32CubeIDE interface with the 'LinkerScript.ld' tab selected. The code in the editor is as follows:

```
1 /**
2 ****
3 * @file      LinkerScript.ld
4 * @author    Auto generated by STM32CubeIDE
5 * Abstract   Linker script for B-L475E-IOT01A1 Board embedding STM32L475VGTx Device from stm32l4
6 *             1024Kbytes FLASH
7 *             96Kbytes RAM
8 *             32Kbytes RAM2
9 *
10 *           Set heap size, stack size and stack location according
11 *           to application requirements.
12 *
13 *           Set memory bank area and size if external memory is used
14 ****
15 * @attention
```

Two sections of the code are highlighted with red boxes:

- The first section, lines 5-8, describes the board and memory resources.
- The second section, lines 10-12, provides instructions for setting up the heap, stack, and stack location.

Step 2. Linker Script - Flash



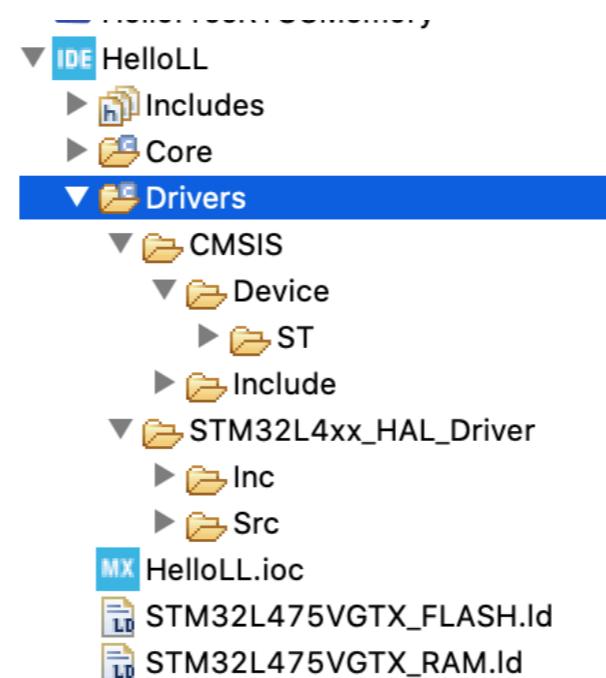
estack = end of stack

```
27
28 * Entry Point */
29 NTRY(Reset_Handler)
30
31 * Highest address of the user mode stack */
32 estack = ORIGIN(RAM) + LENGTH(RAM);      /* end of "RAM" Ram type memory */
33
34 Min_Heap_Size = 0x200 ;      /* required amount of heap */
35 Min_Stack_Size = 0x400 ;      /* required amount of stack */
36
37 * Memories definition */
38 EMORY
39
40 RAM      (xrw)    : ORIGIN = 0x20000000,    LENGTH = 96K
41 RAM2     (xrw)    : ORIGIN = 0x10000000,    LENGTH = 32K
42 FLASH    (rx)     : ORIGIN = 0x80000000,    LENGTH = 1024K
43
44
```

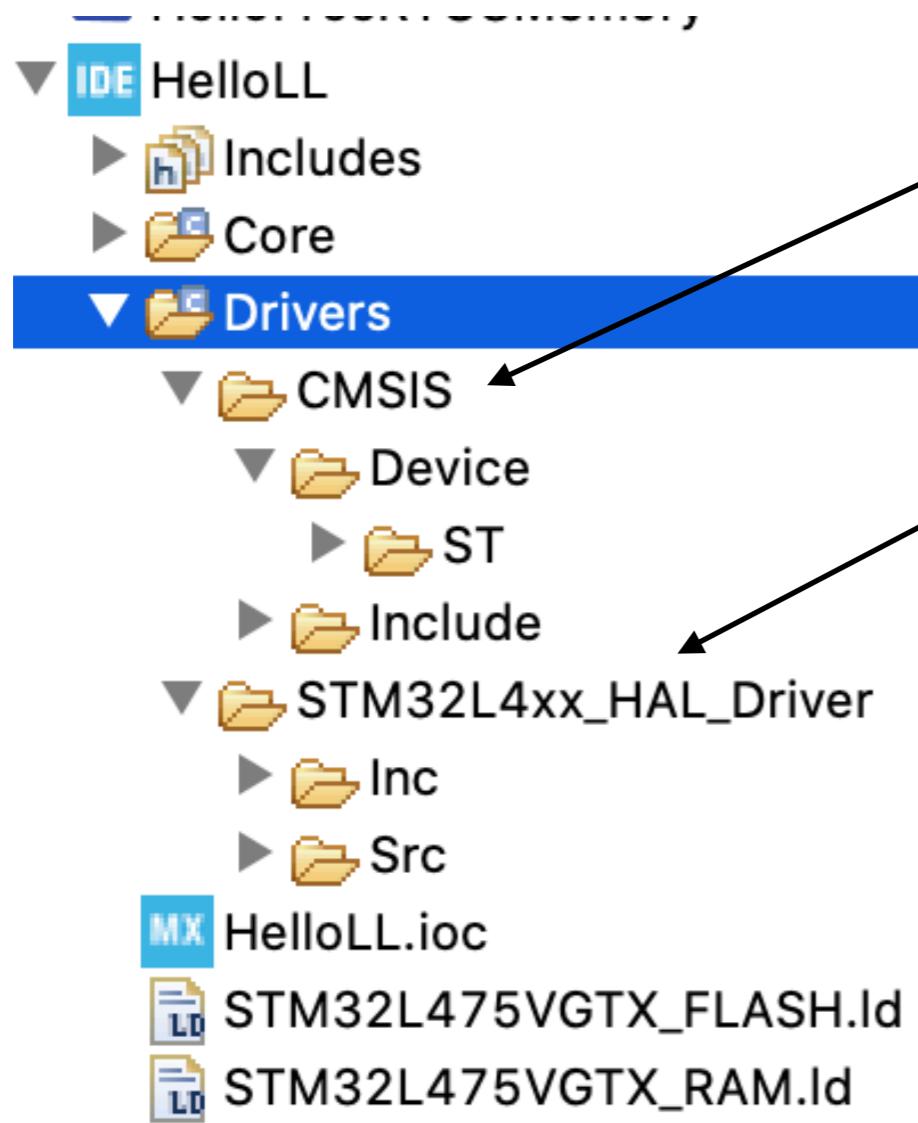
Stack - push/pop as you make function calls
Heap - Dynamic Memory Allocation

File Tour

Driver Files



Step 1. Drivers



Context M
Software Interface Standard

HAL
Hardware Abstraction Layer

Inc
Include Files (.h)

Src
Source Files (.c)

<https://developer.arm.com/tools-and-software/embedded/cmsis>

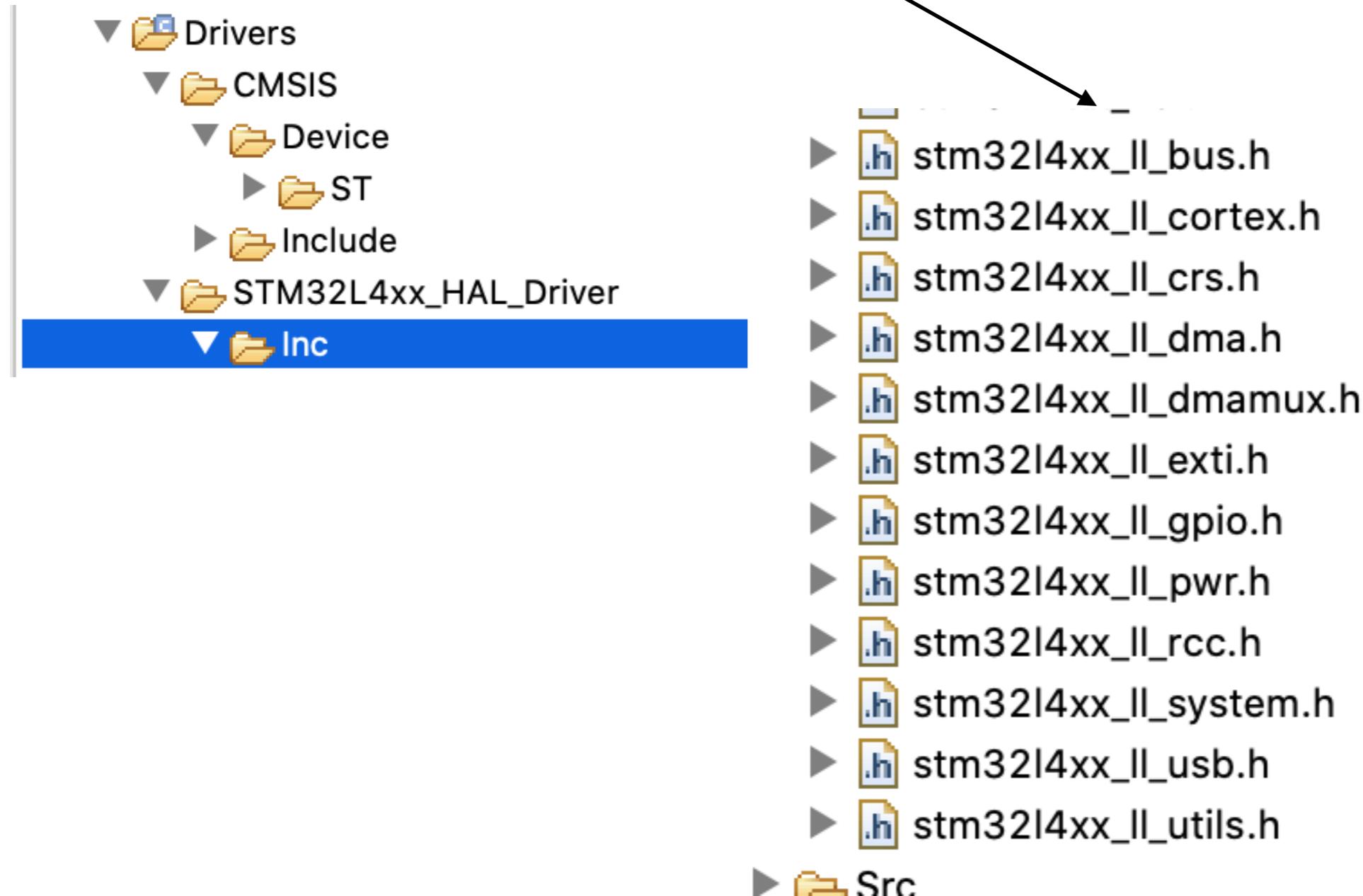
The Cortex Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for microcontrollers that are based on Arm® Cortex® processors. It defines generic tool interfaces and enables consistent device support. Its software interfaces simplify software re-use, reduce the learning curve for microcontroller developers, and improve time to market for new devices.

CMSIS provides interfaces to processor and peripherals, real-time operating systems, and middleware components. It includes a delivery mechanism for devices, boards, and software and enables the combination of software components from multiple vendors.



[List of software packs »](#) [Parametric search for devices »](#)

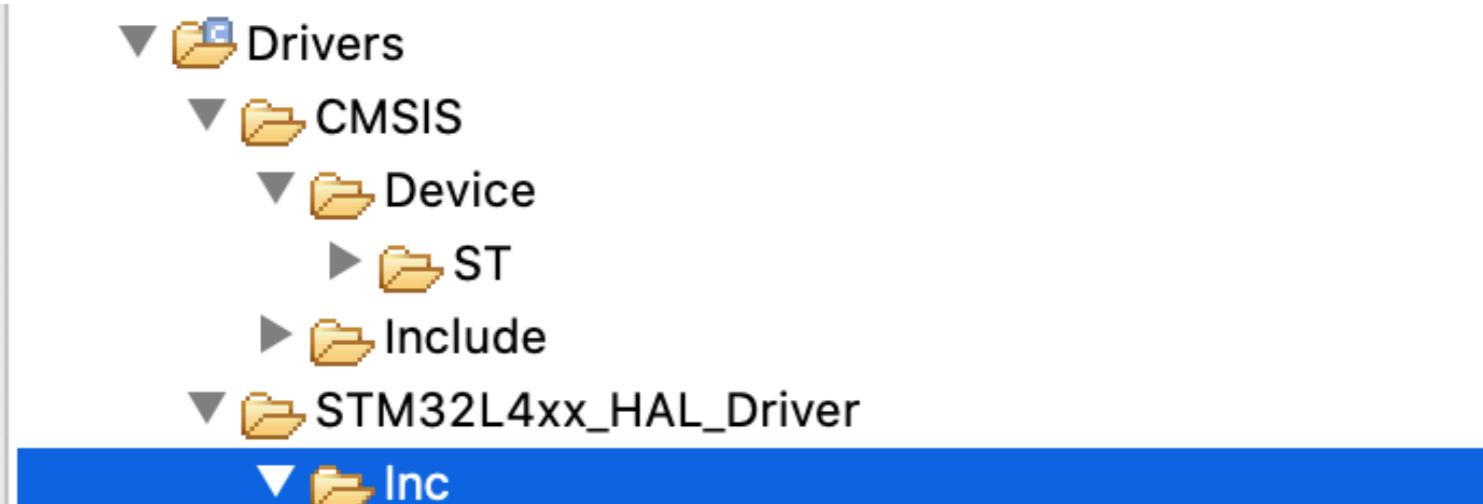
Step 3. LL Header Files



LL
Lower Level

File Tour

Part 1



- `.h stm32l4xx_ll_bus.h`
- `.h stm32l4xx_ll_cortex.h`
- `.h stm32l4xx_ll_crs.h`
- `.h stm32l4xx_ll_dma.h`
- `.h stm32l4xx_ll_dmamux.h`
- `.h stm32l4xx_ll_exti.h`
- `.h stm32l4xx_ll_gpio.h`
- `.h stm32l4xx_ll_pwr.h`
- `.h stm32l4xx_ll_rcc.h`
- `.h stm32l4xx_ll_system.h`
- `.h stm32l4xx_ll_usb.h`
- `.h stm32l4xx_ll_utils.h`
- `Src`

Step 1. stm32l4xx_ll_gpio.h

Automatic documentation generation
With **doxygen** tool

```
1① /**
2 ****
3 * @file    stm32l4xx_ll_gpio.h
4 * @author  MCD Application Team
5 * @brief   Header file of GPIO LL module.
6 ****
7 * @attention
8 *
9 * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics.
10 * All rights reserved.</center></h2>
11 *
12 * This software component is licensed by ST under BSD 3-Clause license,
13 * the "License"; You may not use this file except in compliance with the
14 * License. You may obtain a copy of the License at:
15 *               opensource.org/licenses/BSD-3-Clause
16 *
17 ****
18 */
```

[https://www.doxygen.nl/
index.html](https://www.doxygen.nl/index.html)

The screenshot shows a web browser displaying the official Doxygen website at <https://www.doxygen.nl/>. The page has a blue header with the Doxygen logo. Below the header is a navigation menu with links for Home, Downloads, Documentation, Extensions, and Support. A sidebar on the left contains links for Doxygen (About, Downloads, Changelog, Documentation, Get Involved, Wish list, Examples, Links, Extensions), Downloads, Changelog, Documentation, Get Involved, Wish list, Examples, and Links. The main content area features a large title "Doxygen" and a subtitle "Generate documentation from source code". It describes Doxygen as the de facto standard tool for generating documentation from annotated C++ source code in various languages. It also mentions that Doxygen can help you in three ways.

Step 2. stm32l4xx_ll_gpio.h

```
15  
20 /* Define to prevent recursive inclusion -  
21 #ifndef STM32L4xx_LL_GPIO_H  
22 #define STM32L4xx_LL_GPIO_H  
23  
24 #ifdef __cplusplus  
25 extern "C" {  
26 #endif  
27  
28 /* Includes -----  
29 #include "stm32l4xx.h"  
30
```

NOTE: Grayed out
Means we are not using
C++ compiler for this
build

Include this header
file a single time

If compiling with
C++ compiler, use
C naming instead
of C++ “mangled names”
For this code

Step 3. stm32l4xx_ll_gpio.h

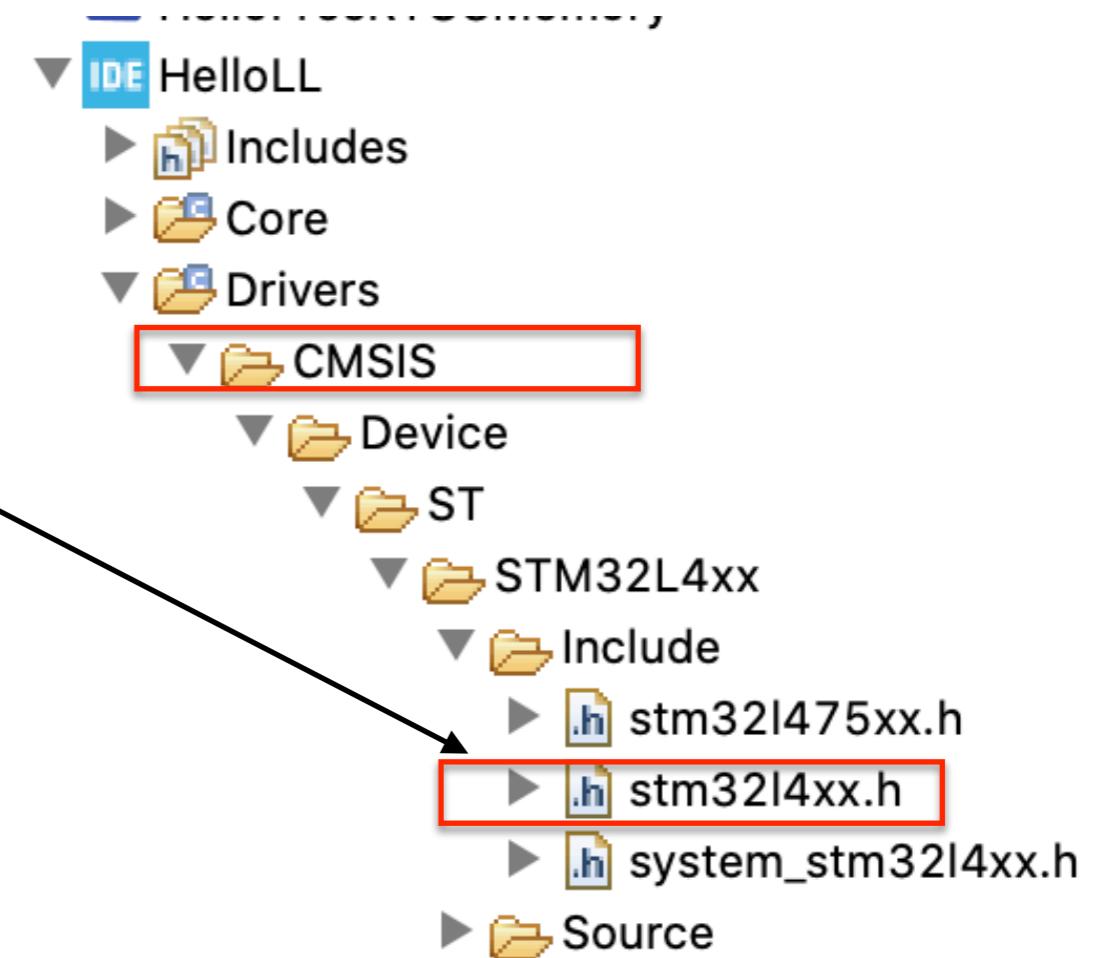
Include the remainder of this code only
if using GPIOA or GPIOB or ...

Step 4. stm32l4xx_ll_gpio.h

Include file for CMSIS code common to all STM32L4 devices

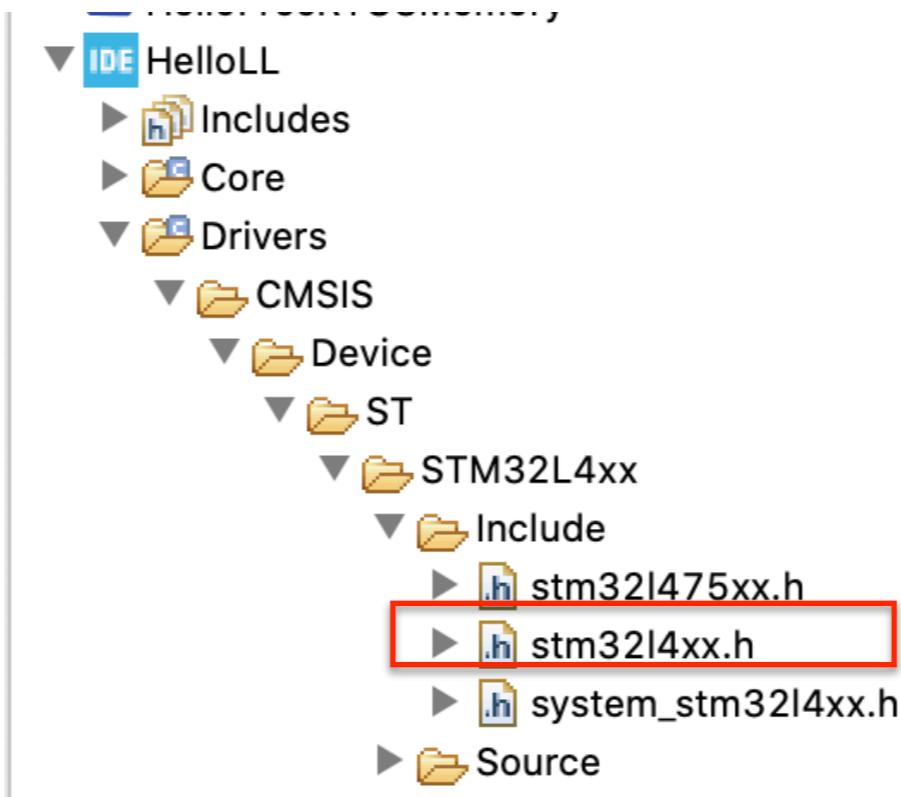
```
27  
28 /* Includes -----  
29 #include "stm32l4xx.h"  
30
```

CMSIS
Cortex M
Software Interface Standard



File Tour

stm32l4xx.h



stm32l4xx.h

Part 1

```
1 /*
2 ****
3 * @file    stm32l4xx.h
4 * @author  MCD Application Team
5 * @brief   CMSIS STM32L4xx Device Peripheral Access Layer Header File.
6 *
7 *          The file is the unique include file that the application programmer
8 *          is using in the C source code, usually in main.c. This file contains:
9 *          - Configuration section that allows to select:
10 *              - The STM32L4xx device used in the target application
11 *              - To use or not the peripheral's drivers in application code(i.e.
12 *                  code will be based on direct access to peripheral's registers
13 *                  rather than drivers API), this option is controlled by
14 *                  "#define USE_HAL_DRIVER"
15 *
```

stm32l4xx.h

Part 2

```
37  
38 #ifndef __STM32L4xx_H  
39 #define __STM32L4xx_H  
40  
41 #ifdef __cplusplus  
42     extern "C" {  
43 #endif /* __cplusplus */  
44
```

NOTE: Grayed out
Means we are not using
C++ compiler for this
build

Include this header
file a single time

If compiling with
C++ compiler, use
C naming instead
of C++ “mangled names”
For this code

stm32l4xx.h

Part 3

```
49 /*
50   * @brief STM32 Family
51   */
52 #if !defined (STM32L4)
53 #define STM32L4
54 #endif /* STM32L4 */
55
```

If not defined,
Then define it.
Says we are using
The STM32L4 family

stm32l4xx.h

Part 4

```
55
56④ /* Uncomment the line below according to the target STM32L4 device used in your
57    application
58 */
59
60 #if !defined (STM32L412xx) && !defined (STM32L422xx) && \
61   !defined (STM32L431xx) && !defined (STM32L432xx) && !defined (STM32L433xx) && !defined (ST
62   !defined (STM32L451xx) && !defined (STM32L452xx) && !defined (STM32L462xx) && \
63   !defined (STM32L471xx) && !defined (STM32L475xx) && !defined (STM32L476xx) && !defined (ST
64   !defined (STM32L496xx) && !defined (STM32L4A6xx) && \
65   !defined (STM32L4P5xx) && !defined (STM32L4Q5xx) && \
66   !defined (STM32L4R5xx) && !defined (STM32L4R7xx) && !defined (STM32L4R9xx) && !defined (ST
67   /* #define STM32L412xx */ /*!< STM32L412xx Devices */
68   /* #define STM32L422xx */ /*!< STM32L422xx Devices */
69   /* #define STM32L431xx */ /*!< STM32L431xx Devices */
70   /* #define STM32L432xx */ /*!< STM32L432xx Devices */
71   /* #define STM32L433xx */ /*!< STM32L433xx Devices */
72   /* #define STM32L442xx */ /*!< STM32L442xx Devices */
73   /* #define STM32L443xx */ /*!< STM32L443xx Devices */
```

We use STM32L475xx on Discovery Kit

stm32l4xx.h

Part 5

CMSIS Version Number is 01.07.00

```
--.  
105④ /**  
106   * @brief CMSIS Device version number  
107   */  
108 #define __STM32L4_CMSIS_VERSION_MAIN    (0x01) /*!< [31:24] main version */  
109 #define __STM32L4_CMSIS_VERSION_SUB1     (0x07) /*!< [23:16] sub1 version */  
110 #define __STM32L4_CMSIS_VERSION_SUB2     (0x00) /*!< [15:8]  sub2 version */  
111 #define __STM32L4_CMSIS_VERSION_RC      (0x00) /*!< [7:0]   release candidate */  
112④ #define __STM32L4_CMSIS_VERSION        ((__STM32L4_CMSIS_VERSION_MAIN << 24) \  
113                                         |(__STM32L4_CMSIS_VERSION_SUB1 << 16) \  
114                                         |(__STM32L4_CMSIS_VERSION_SUB2 << 8 ) \  
115                                         |(__STM32L4_CMSIS_VERSION_RC))  
116
```

stm32l4xx.h

Part 6

Pull in info specific to the MCU we are using

```
142 #include "stm32l452xx.h"
143 #elif defined(STM32L462xx)
144     #include "stm32l462xx.h"
145 #elif defined(STM32L471xx)
146     #include "stm32l471xx.h"
147 #elif defined(STM32L475xx)
148     #include "stm32l475xx.h"
149 #elif defined(STM32L476xx)
150     #include "stm32l476xx.h"
151 #elif defined(STM32L485xx)
152     #include "stm32l485xx.h"
```

stm32l4xx.h

Part 7

Types used throughout the code

```
185 //  
186 typedef enum  
187 {  
188     RESET = 0,  
189     SET = !RESET  
190 } FlagStatus, ITStatus;  
191  
192 typedef enum  
193 {  
194     DISABLE = 0,  
195     ENABLE = !DISABLE  
196 } FunctionalState;  
197 #define IS_FUNCTIONAL_STATE(STATE) (((STATE) == DISABLE) || ((STATE) == ENABLE))  
198  
199 typedef enum  
200 {  
201     SUCCESS = 0,  
202     ERROR = !SUCCESS  
203 } ErrorStatus;  
204
```

FlagStatus

FunctionalState

ErrorStatus

stm32l4xx.h

Part 8

Macros used throughout the code

```
!09
!10 /* @addtogroup Exported_macros
!11   * @{
!12   */
!13 #define SET_BIT(REG, BIT)      ((REG) |= (BIT))
!14
!15 #define CLEAR_BIT(REG, BIT)    ((REG) &= ~(BIT))
!16
!17 #define READ_BIT(REG, BIT)     ((REG) & (BIT))
!18
!19 #define CLEAR_REG(REG)        ((REG) = (0x0))
!20
!21 #define WRITE_REG(REG, VAL)   ((REG) = (VAL))
!22
!23 #define READ_REG(REG)        ((REG))
!24
!25 #define MODIFY_REG(REG, CLEARMASK, SETMASK)  WRITE_REG((REG), (((READ_REG(REG)) & ( ~(CLEARMASK))) | (SETMASK) & (~SETMASK)))
!26
!27 #define POSITION_VAL(VAL)    (__CLZ(__RBIT(VAL)))
!28
```

stm32l4xx.h

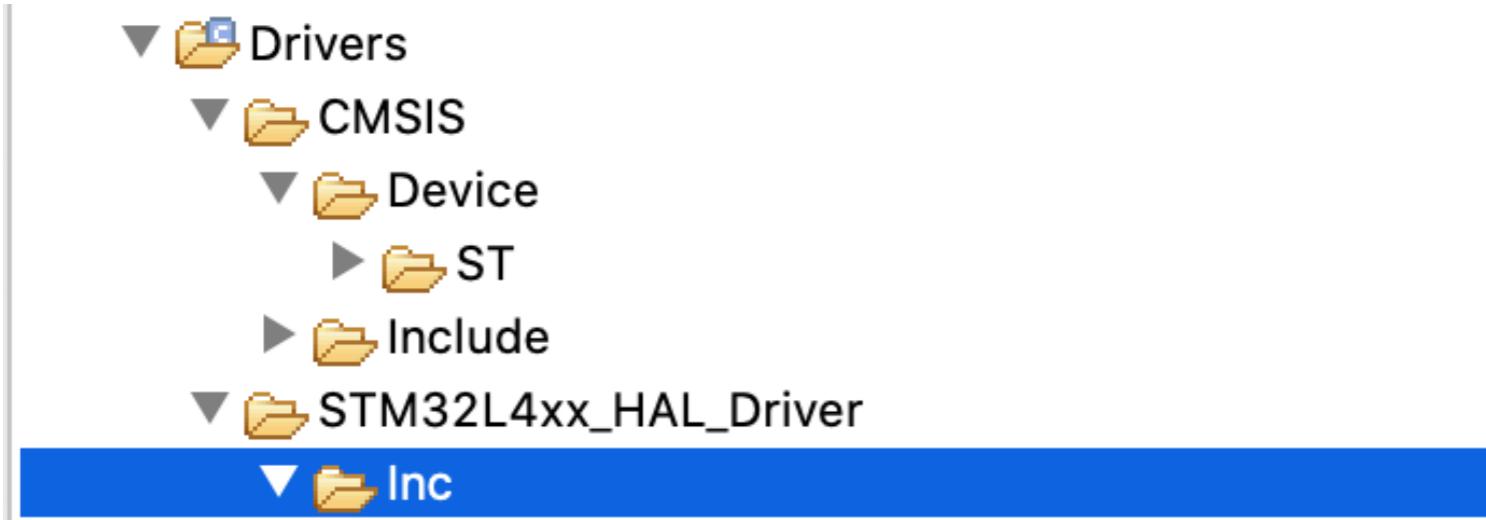
Part 9

If using HAL drivers

```
.32    "
.33
.34 #if defined (USE_HAL_DRIVER)
.35 #include "stm32l4xx_hal.h"
.36 #endif /* USE_HAL_DRIVER */
```

File Tour

Part 2



- `.h stm32l4xx_ll_bus.h`
- `.h stm32l4xx_ll_cortex.h`
- `.h stm32l4xx_ll_crs.h`
- `.h stm32l4xx_ll_dma.h`
- `.h stm32l4xx_ll_dmamux.h`
- `.h stm32l4xx_ll_exti.h`
- `.h stm32l4xx_ll_gpio.h`
- `.h stm32l4xx_ll_pwr.h`
- `.h stm32l4xx_ll_rcc.h`
- `.h stm32l4xx_ll_system.h`
- `.h stm32l4xx_ll_usb.h`
- `.h stm32l4xx_ll_utils.h`
- `Src`

Step: View “Drivers/STM32L4xx_HAL_Driver/Inc/stm32l4xx_ll_gpio.h” - Part 1

```
*****  
* @file    stm32l4xx_ll_gpio.h  
* @author  MCD Application Team  
* @brief   Header file of GPIO LL module.  
*****
```

```
/* Define to prevent recursive inclusion -----  
#ifndef __STM32L4xx_LL_GPIO_H  
#define __STM32L4xx_LL_GPIO_H  
  
#ifdef __cplusplus  
extern "C" {  
#endif  
  
/* Includes -----  
#include "stm32l4xx.h"
```

Step: View “Drivers/ STM32L4xx_HAL_Driver/Inc/ stm32l4xx_ll_gpio.h” - Part 2

```
#if defined (GPIOA) || defined (GPIOB) || defined (GPIOC) || defined (GPIOD) || defined (GPIOE) ||  
defined (GPIOF) || defined (GPIOG) || defined (GPIOH) || defined (GPIOI)  
  
/*  
/** MISRA C:2012 deviation rule has been granted for following rules:  
 * Rule-18.1_d - Medium: Array pointer `GPIOx' is accessed with index [...,...]  
 * which may be out of array bounds [...,UNKNOWN] in following APIs:  
 * LL_GPIO_SetAFPin_0_7  
 * LL_GPIO_SetAFPin_0_7  
 * LL_GPIO_SetAFPin_8_15  
 * LL_GPIO_SetAFPin_8_15  
 */
```

Step: View “Drivers/STM32L4xx_HAL_Driver/Inc/stm32l4xx_ll_gpio.h” - Part 3

```
/*
typedef struct
{
    uint32_t Pin;          /*!< Specifies the GPIO pins to be configured.  
This parameter can be any value of @ref GPIO_LL_EC_PIN */

    uint32_t Mode;         /*!< Specifies the operating mode for the selected pins.  
This parameter can be a value of @ref GPIO_LL_EC_MODE.  
  
GPIO HW configuration can be modified afterwards using unitary function @ref LL_GPIO_SetPinMode().*/

    uint32_t Speed;        /*!< Specifies the speed for the selected pins.  
This parameter can be a value of @ref GPIO_LL_EC_SPEED.  
  
GPIO HW configuration can be modified afterwards using unitary function @ref LL_GPIO_SetPinSpeed().*/

    uint32_t OutputType;   /*!< Specifies the operating output type for the selected pins.  
This parameter can be a value of @ref GPIO_LL_EC_OUTPUT.  
  
GPIO HW configuration can be modified afterwards using unitary function @ref LL_GPIO_SetPinOutputType().*/

    uint32_t Pull;          /*!< Specifies the operating Pull-up/Pull down for the selected pins.  
This parameter can be a value of @ref GPIO_LL_EC_PULL.  
  
GPIO HW configuration can be modified afterwards using unitary function @ref LL_GPIO_SetPinPull().*/

    uint32_t Alternate;    /*!< Specifies the Peripheral to be connected to the selected pins.  
This parameter can be a value of @ref GPIO_LL_EC_AF.  
  
GPIO HW configuration can be modified afterwards using unitary function @ref LL_GPIO_SetAFPin_0_7() and LL_GPIO_SetAFPin_8_15().*/
} LL_GPIO_InitTypeDef;
```

LL_GPIO_InitTypeDef

Step: View “Drivers/STM32L4xx_HAL_Driver/Inc/stm32l4xx_ll_gpio.h” - Part 4

```
/** @defgroup GPIO_LL_EC_PIN PIN
 * @{
 */
#define LL_GPIO_PIN_0           GPIO_BSRR_BS0 /*!< Select pin 0 */
#define LL_GPIO_PIN_1           GPIO_BSRR_BS1 /*!< Select pin 1 */
#define LL_GPIO_PIN_2           GPIO_BSRR_BS2 /*!< Select pin 2 */
#define LL_GPIO_PIN_3           GPIO_BSRR_BS3 /*!< Select pin 3 */
#define LL_GPIO_PIN_4           GPIO_BSRR_BS4 /*!< Select pin 4 */
#define LL_GPIO_PIN_5           GPIO_BSRR_BS5 /*!< Select pin 5 */
#define LL_GPIO_PIN_6           GPIO_BSRR_BS6 /*!< Select pin 6 */
#define LL_GPIO_PIN_7           GPIO_BSRR_BS7 /*!< Select pin 7 */
#define LL_GPIO_PIN_8           GPIO_BSRR_BS8 /*!< Select pin 8 */
#define LL_GPIO_PIN_9           GPIO_BSRR_BS9 /*!< Select pin 9 */
#define LL_GPIO_PIN_10          GPIO_BSRR_BS10 /*!< Select pin 10 */
#define LL_GPIO_PIN_11          GPIO_BSRR_BS11 /*!< Select pin 11 */
#define LL_GPIO_PIN_12          GPIO_BSRR_BS12 /*!< Select pin 12 */
#define LL_GPIO_PIN_13          GPIO_BSRR_BS13 /*!< Select pin 13 */
#define LL_GPIO_PIN_14          GPIO_BSRR_BS14 /*!< Select pin 14 */
#define LL_GPIO_PIN_15          GPIO_BSRR_BS15 /*!< Select pin 15 */
#define LL_GPIO_PIN_ALL         (GPIO_BSRR_BS0 | GPIO_BSRR_BS1 | GPIO_BSRR_BS2 | \
                                GPIO_BSRR_BS3 | GPIO_BSRR_BS4 | GPIO_BSRR_BS5 | \
                                GPIO_BSRR_BS6 | GPIO_BSRR_BS7 | GPIO_BSRR_BS8 | \
                                GPIO_BSRR_BS9 | GPIO_BSRR_BS10 | GPIO_BSRR_BS11 | \
                                GPIO_BSRR_BS12 | GPIO_BSRR_BS13 | GPIO_BSRR_BS14 | \
                                GPIO_BSRR_BS15) /*!< Select all pins */

/**
```

Step: View “Drivers/STM32L4xx_HAL_Driver/Inc/stm32l4xx_ll_gpio.h” - Part 5

```
/** @defgroup GPIO_LL_EC_MODE Mode  
 * @{
 */
#define LL_GPIO_MODE_INPUT (0x00000000U) /*!< Select input mode */
#define LL_GPIO_MODE_OUTPUT GPIO_MODER_MODE0_0 /*!< Select output mode */
#define LL_GPIO_MODE_ALTERNATE GPIO_MODER_MODE0_1 /*!< Select alternate function mode */
#define LL_GPIO_MODE_ANALOG GPIO_MODER_MODE0 /*!< Select analog mode */
/** @}
 */

/** @defgroup GPIO_LL_EC_OUTPUT Output Type  
 * @{
 */
#define LL_GPIO_OUTPUT_PUSHPULL (0x00000000U) /*!< Select push-pull as output type */
#define LL_GPIO_OUTPUT_OPENDRAIN GPIO_OTYPER_OT0 /*!< Select open-drain as output type */
/** @}
 */

/** @defgroup GPIO_LL_EC_SPEED Output Speed  
 * @{
 */
#define LL_GPIO_SPEED_FREQ_LOW (0x00000000U) /*!< Select I/O low output speed */
#define LL_GPIO_SPEED_FREQ_MEDIUM GPIO_OSPEEDR_OSPEED0_0 /*!< Select I/O medium output speed */
#define LL_GPIO_SPEED_FREQ_HIGH GPIO_OSPEEDR_OSPEED0_1 /*!< Select I/O fast output speed */
#define LL_GPIO_SPEED_FREQ_VERY_HIGH GPIO_OSPEEDR_OSPEED0 /*!< Select I/O high output speed */
/** @}
 */

#define LL_GPIO_SPEED_FREQ_LOW LL_GPIO_SPEED_FREQ_LOW
#define LL_GPIO_SPEED_FREQ_MEDIUM LL_GPIO_SPEED_FREQ_MEDIUM
#define LL_GPIO_SPEED_FREQ_HIGH LL_GPIO_SPEED_FREQ_HIGH
#define LL_GPIO_SPEED_FREQ_VERY_HIGH LL_GPIO_SPEED_FREQ_VERY_HIGH
```

Step: View “Drivers/STM32L4xx_HAL_Driver/Inc/stm32l4xx_ll_gpio.h” - Part 6

```
/** @defgroup GPIO_LL_EC_PULL Pull Up Pull Down
 * @{
 */
#define LL_GPIO_PULL_NO          (0x00000000U) /*!< Select I/O no pull */
#define LL_GPIO_PULL_UP          GPIO_PUPDR_PUPD0_0 /*!< Select I/O pull up */
#define LL_GPIO_PULL_DOWN         GPIO_PUPDR_PUPD0_1 /*!< Select I/O pull down */
/**@
 * @}
 */

/** @defgroup GPIO_LL_EC_AF Alternate Function
 * @{
 */
#define LL_GPIO_AF_0              (0x00000000U) /*!< Select alternate function 0 */
#define LL_GPIO_AF_1              (0x0000001U) /*!< Select alternate function 1 */
#define LL_GPIO_AF_2              (0x0000002U) /*!< Select alternate function 2 */
#define LL_GPIO_AF_3              (0x0000003U) /*!< Select alternate function 3 */
#define LL_GPIO_AF_4              (0x0000004U) /*!< Select alternate function 4 */
#define LL_GPIO_AF_5              (0x0000005U) /*!< Select alternate function 5 */
#define LL_GPIO_AF_6              (0x0000006U) /*!< Select alternate function 6 */
#define LL_GPIO_AF_7              (0x0000007U) /*!< Select alternate function 7 */
#define LL_GPIO_AF_8              (0x0000008U) /*!< Select alternate function 8 */
#define LL_GPIO_AF_9              (0x0000009U) /*!< Select alternate function 9 */
#define LL_GPIO_AF_10             (0x000000AU) /*!< Select alternate function 10 */
#define LL_GPIO_AF_11             (0x000000BU) /*!< Select alternate function 11 */
#define LL_GPIO_AF_12             (0x000000CU) /*!< Select alternate function 12 */
#define LL_GPIO_AF_13             (0x000000DU) /*!< Select alternate function 13 */
#define LL_GPIO_AF_14             (0x000000EU) /*!< Select alternate function 14 */
#define LL_GPIO_AF_15             (0x000000FU) /*!< Select alternate function 15 */
/**@}
```

Step: View “Drivers/STM32L4xx_HAL_Driver/Inc/stm32l4xx_ll_gpio.h” - Part 7

```
/* Exported macro -----*/
/** @defgroup GPIO_LL_Exported_Macros GPIO Exported Macros
 * @{
 */

/** @defgroup GPIO_LL_EM_WRITE_READ Common Write and read registers Macros
 * @{
 */

/** @brief Write a value in GPIO register
 * @param __INSTANCE__ GPIO Instance
 * @param __REG__ Register to be written
 * @param __VALUE__ Value to be written in the register
 * @retval None
 */
#define LL_GPIO_WriteReg(__INSTANCE__, __REG__, __VALUE__) WRITE_REG(__INSTANCE__->__REG__, (__VALUE__))

/** @brief Read a value in GPIO register
 * @param __INSTANCE__ GPIO Instance
 * @param __REG__ Register to be read
 * @retval Register value
 */
#define LL_GPIO_ReadReg(__INSTANCE__, __REG__) READ_REG(__INSTANCE__->__REG__)

/**
 * @}
 */

```

Step: View “Drivers/STM32L4xx_HAL_Driver/Inc/stm32l4xx_ll_gpio.h” - Part 8

```
/**  
 * @brief Write output data register for the port.  
 * @rmtoll ODR      ODy      LL_GPIO_WriteOutputPort  
 * @param  GPIOx  GPIO Port  
 * @param  PortValue Level value for each pin of the port  
 * @retval None  
 */  
_STATIC_INLINE void LL_GPIO_WriteOutputPort(GPIO_TypeDef *GPIOx, uint32_t PortValue)  
{  
    WRITE_REG(GPIOx->ODR, PortValue);  
}  
  
/**  
 * @brief Return full output data register value for a dedicated port.  
 * @rmtoll ODR      ODy      LL_GPIO_ReadOutputPort  
 * @param  GPIOx  GPIO Port  
 * @retval Output data register value of port  
 */  
_STATIC_INLINE uint32_t LL_GPIO_ReadOutputPort(GPIO_TypeDef *GPIOx)  
{  
    return (uint32_t)(READ_REG(GPIOx->ODR));  
}
```

Step: View “Drivers/STM32L4xx_HAL_Driver/Inc/stm32l4xx_ll_gpio.h” - Part 9

```
#if defined(USE_FULL_LL_DRIVER)
/** @defgroup GPIO_LL_EF_Init Initialization and de-initialization functions
 * @{
 */
ErrorStatus LL_GPIO_DeInit(GPIO_TypeDef *GPIOx);
ErrorStatus LL_GPIO_Init(GPIO_TypeDef *GPIOx, LL_GPIO_InitTypeDef *GPIO_InitStruct);
void      LL_GPIO_StructInit(LL_GPIO_InitTypeDef *GPIO_InitStruct);
```

Summary

- Reference Info
- Concepts
- STM32CubeIDE and LL Code Generation
 - Blink LED using LL APIs
- File Tours