

# CESE\_2019\_PROY-FINAL

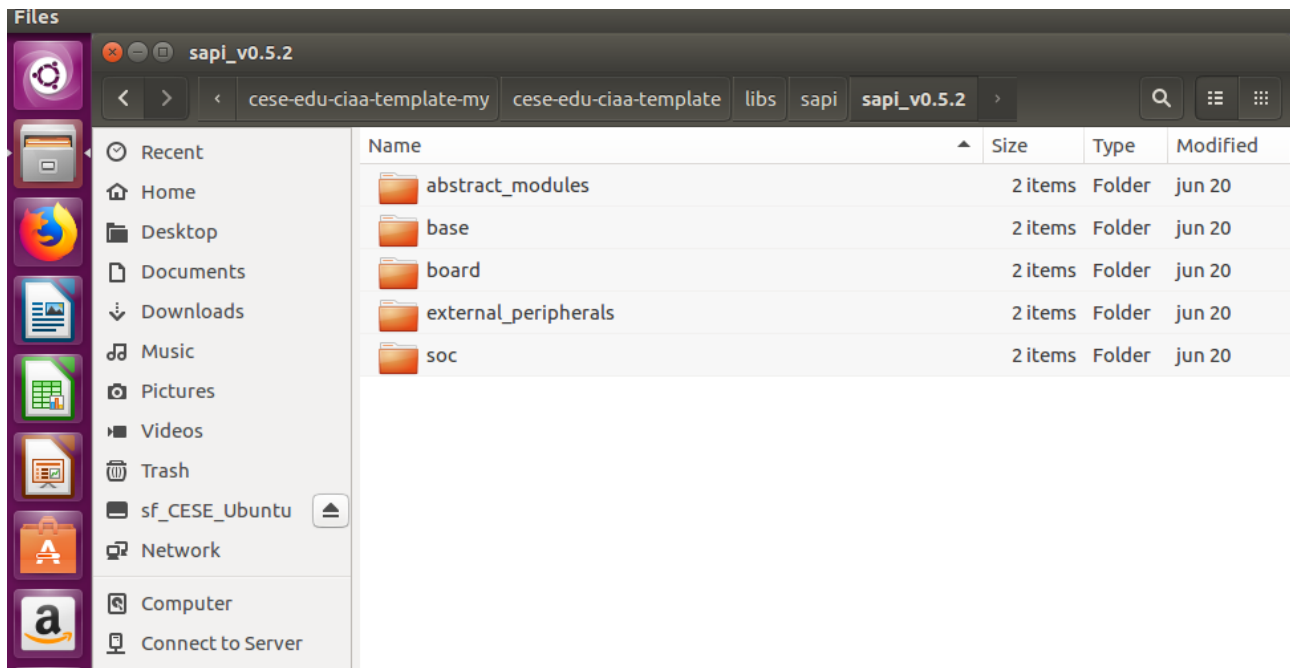
R. Oliva

Tareas de Avance con sAPI (de E.Pernia) a sapi\_3c – Upd (f) desde D.1.8) UART Test

(✓ Indica Realizado)

## A) Estructura

V0.5.2 – dividida como sigue (/sapi está dentro de /libs):



A.1) Abstract Modules:

A.2) Base:

A.3) Board:

A.3) External Peripherals:

A.4.1) /SOC/Core

A.4.2) /SOC/peripherals/usb

A.4.3) /soc/peripherals

## B) Tomado de sapi\_datatypes.h

// From: <https://es.coursera.org/lecture/embedded-software-hardware/9-register-definition-files-6pqVq>

```
// __I Defines 'read only' permissions: volatile const
// __O Defines 'write only' permissions: volatile
// __IO Defines 'read / write' permissions: volatile
```

```
#define HW_REG_8_R(x)      (*((__I uint8_t *) (x)))
#define HW_REG_16_R(x)     (*((__I uint16_t *) (x)))
#define HW_REG_32_R(x)     (*((__I uint32_t *) (x)))
```

```
#define HW_REG_8_W(x)      (*((__O uint8_t *) (x)))
#define HW_REG_16_W(x)     (*((__O uint16_t *) (x)))
#define HW_REG_32_W(x)     (*((__O uint32_t *) (x)))
```

```
#define HW_REG_8_RW(x)     (*((__IO uint8_t *) (x)))
#define HW_REG_16_RW(x)    (*((__IO uint16_t *) (x)))
#define HW_REG_32_RW(x)    (*((__IO uint32_t *) (x)))
```

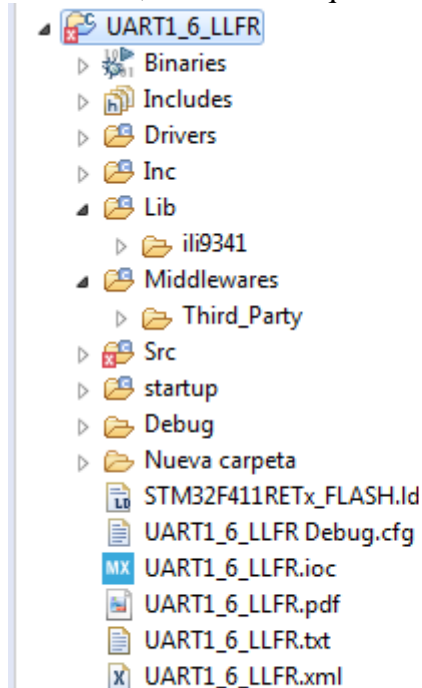
```
// Example:
// #define REG_NAME      (HW_REG_32_RW(0x4544555))
```

B.1) De- referenciando Registros de Hardware, sin consumo de RAM:

<https://es.coursera.org/lecture/embedded-software-hardware/9-register-definition-files-6pqVq>

## C) Estructura de sAPI\_3C

C.1) Usando como base el traslado de un proyecto que ya está configurado para operar en AC6 para STM32F4, la estructura que funciona actualmente para el proyecto de pruebas es la siguiente:



Se podría mantener el /lib y dentro de el generar un directorio /sAPI3C

Este tendría los directorios

- /board
- /base
- /soc
- /external\_peripherals

A imitación del / board de sAPI,

El sAPI3C podría tener (es una única placa)

- /inc/sapi3c\_board.h
- /src/sapi3c\_board.c

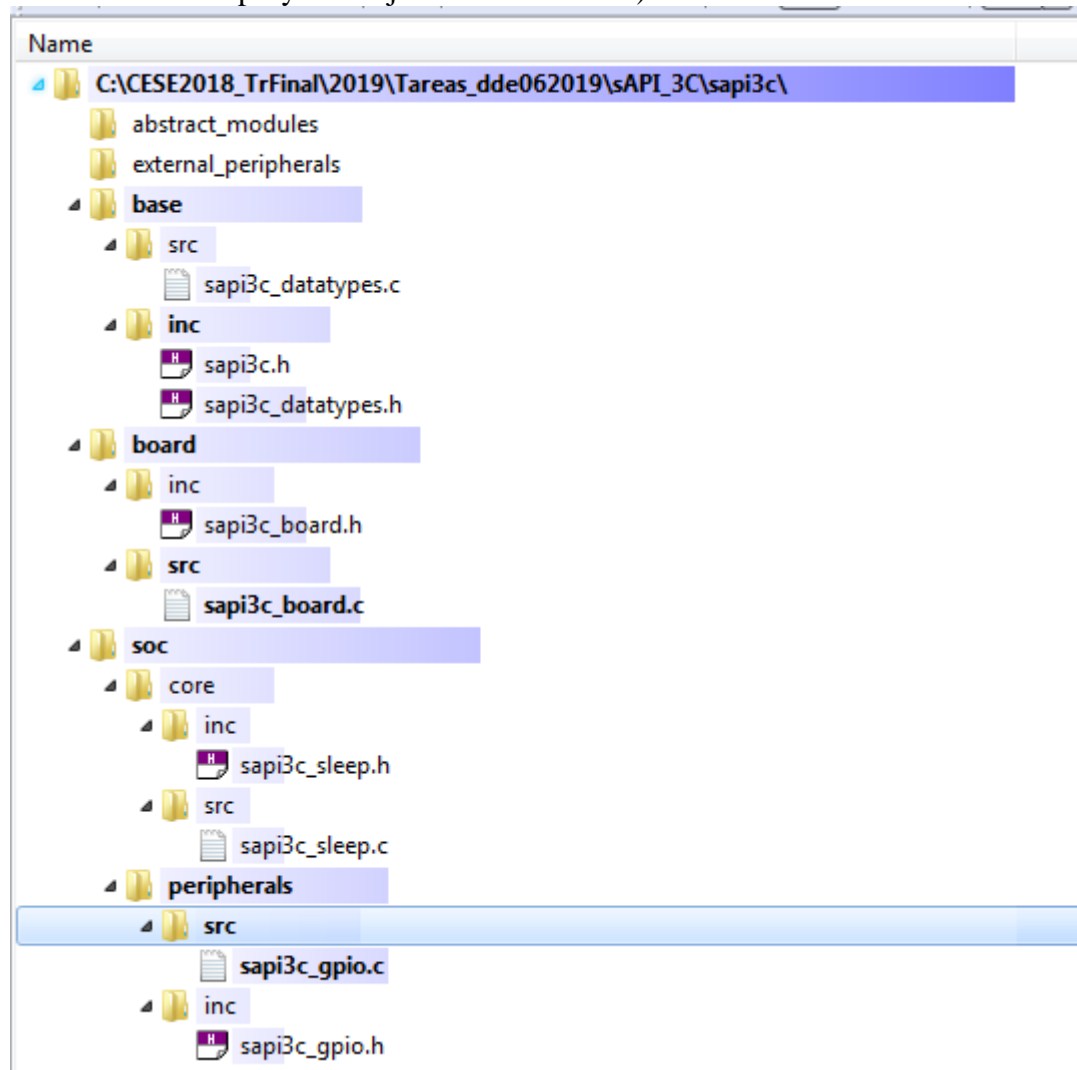
A imitación del / base de sAPI,

El sAPI3C podría tener

- /inc/sapi3c.h
- /inc/sapi3c\_datatypes.h

(no tiene mucho)      /src/sapi3c\_datatypes.c

**C) Ensayo elemental** Al 8.19 lo que tenemos para ensayar es nuestro **DIRECTORIO BASE 01** (esto estaría dentro del proyecto bajo un directorio /lib):



Un tema a tener en cuenta es lo que se mencionaba en el curso Udemey/Fastbit en el uso de AC6: los directorios se copian cerrando el AC6 previamente, o se no desde “pegar” que sí admite Eclipse pero después genera problemas. Ver procedimiento en:

C:\Users\Rafael\Desktop\Udemey\Admin\FreeRTOS\Printouts\CreacionProjectNvoResumido(16-03-2019).pdf

### C.1) Procedimiento: Allí se hace el procedimiento:

1. generando el proyecto en AC6,
2. cerrándolo, y..
3. luego copiando en el directorio del proyecto los directorios /Config (que guarda FreeRTOSConfig.h) y los directorios del FreeRTOS, como /ThirdParty.
4. Una vez que está todo copiado, al abrir AC6 se da “refresh” para el proyecto,
5. Se va para cada directorio “nuevo” eliminando el “exclude from build” (right click en cada directorio, C/C++Build, destildar el “exclude resource from build”).
6. Finalmente, parado en el directorio del proyecto, en properties del Proyecto, C/C++ Build-> settings->MCU GCC-> Includes --- presionar el “+” para agregar los .h que se incorporan de los nuevos directorios.

### C.2) Lo que queremos emular en AC6:

C.2.1) blinky.c (C:\CESE2019\EDU\_CIAA\_Repo\cese-edu-ciaa-template\examples\c\sapi\gpio\blinky\src\blinky.c)

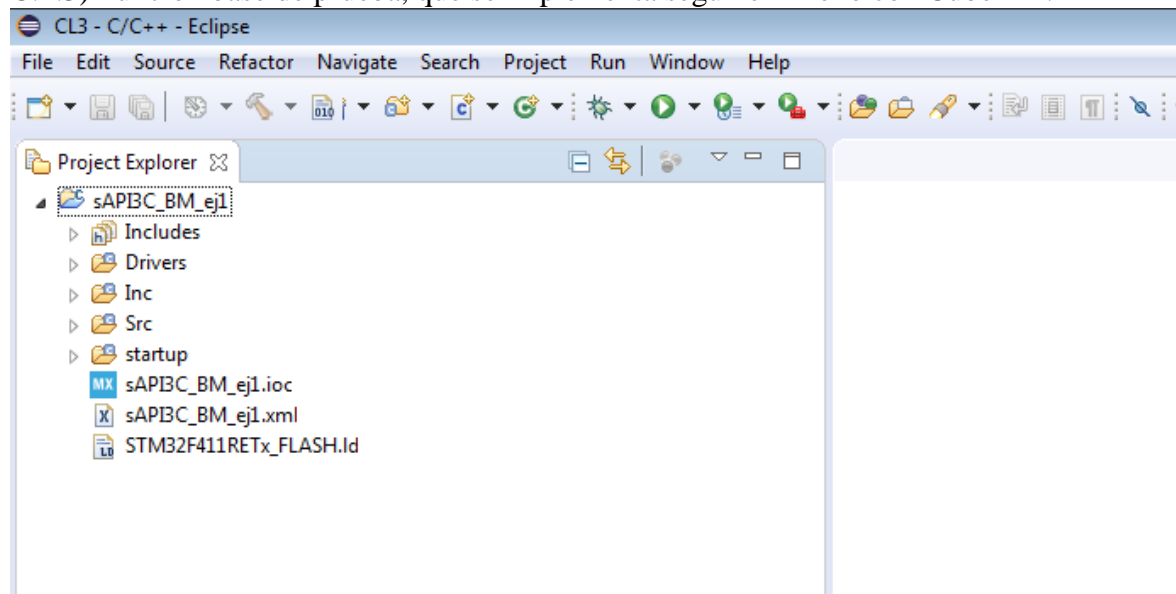
```
#include "sapi.h"          // <= sAPI header
int main(void){
    /* Inicializar la placa */
    boardConfig();
    /* ----- REPETIR POR SIEMPRE ----- */
    while(1) {
        /* Prendo el led azul */
        gpioWrite( LEDB, ON );
        delay(500);
        /* Apago el led azul */
        gpioWrite( LEDB, OFF );
        delay(500);
    }
    return 0 ;
}
```

C.2.2) O el “switches\_leds.c”

```
#include "sapi.h"          // <= sAPI header
int main(void){
    /* Inicializar la placa */
    boardConfig();
    gpioConfig( GPIO0, GPIO_INPUT );
    gpioConfig( GPIO1, GPIO_OUTPUT );
    /* Variable para almacenar el valor de tecla leído */
    bool_t valor;
    /* ----- REPETIR POR SIEMPRE ----- */
    while(1) {
        valor = !gpioRead( TEC1 );
        gpioWrite( LEDB, valor );
        valor = !gpioRead( TEC2 );
        gpioWrite( LED1, valor );
        valor = !gpioRead( TEC3 );
        gpioWrite( LED2, valor );
        valor = !gpioRead( TEC4 );
        gpioWrite( LED3, valor );
        valor = !gpioRead( GPIO0 );
        gpioWrite( GPIO1, valor );
    }
    return 0 ;
}
```

Partiendo de algo como esto:

### C.2.3) Funcion base de prueba, que se implementa según el Anexo con CubeMX:



Aquí, dentro de /src está main.c que tiene:

```
/* USER CODE END Header */

/* Includes -----*/
#include "main.h"
#include "adc.h"
#include "i2c.h"
#include "rtc.h"
#include "sdio.h"
#include "spi.h"
#include "usart.h"
// #include "sapi3gpio.h"

void SystemClock_Config(void);

int main(void)
{
    HAL_Init();

    SystemClock_Config();

    MX_GPIO_Init();
    MX_I2C1_Init();
    MX_USART6_UART_Init();
    MX_USART1_UART_Init();
    MX_ADC1_Init();
    MX_SPI1_Init();
    MX_USART2_UART_Init();
    MX_SDIO_SD_Init();
    MX_I2C3_Init();
    MX_RTC_Init();
    while (1)
    {
        HAL_Delay(1000);
        LL_GPIO_TogglePin(OLED_GPIO_Port,OLED_Pin);
    }
}
```

### C.3) Implementación en AC6:

#### C.3.1) Pasos: (Editamos todavía en nuestro DIRECTORIO BASE 01)

**C.3.1a) Ver funciones a utilizar, para lograr a partir de C.2.3) una estructura “limpia” similar a la de los ejemplos /sapi, sin archivos adicionales en el directorio /src. Los archivos de /stm32xx que están en el /src principal podrían ir en el lib/sapi3c/board (/inc o /src, según corresponda)**

**C.3.1b) Modificar el par: /src/main.c y /inc/main.h a utilizar en DB01, para un primer ensayo, como se indica aquí:**

##### C.3.1b.1) main.c ✓

C:\CESE2018\_TrFinal\2019\Tareas\_dde062019\sAPI\_3C\Tests\Test01sAPI3C\_BM\_ej1\main.c

```
/* Includes -----*/
#include "main.h"
#include "sapi3c.h"

int main(void)
{
    boardInitCL3();

    while (1)
    {
        HAL_Delay(500);
        gpioWrite(OLED_PB2, ON);
        HAL_Delay(500);
        gpioWrite(OLED_PB2, OFF);
    }
}
```

##### C.3.1b.2) main.h ✓

C:\CESE2018\_TrFinal\2019\Tareas\_dde062019\sAPI\_3C\Tests\Test01sAPI3C\_BM\_ej1\main.h

```
/**
*****
* @file    main.h
* @author  E.Pernia / adapted by R.Oliva
* @brief   This file Tests basic operations for CL3
*
*
**/

/* Define to prevent recursive inclusion -----*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----*/
#include "stm32f4xx_hal.h"
#include "stm32f4xx_hal.h"
#include "stm32f4xx_ll_usart.h"
#include "stm32f4xx_ll_rcc.h"
#include "stm32f4xx.h"
#include "stm32f4xx_ll_system.h"
#include "stm32f4xx_ll_gpio.h"
#include "stm32f4xx_ll_exti.h"
#include "stm32f4xx_ll_bus.h"
#include "stm32f4xx_ll_cortex.h"
#include "stm32f4xx_ll_utils.h"
#include "stm32f4xx_ll_pwr.h"
#include "stm32f4xx_ll_dma.h"
```

```

/* Private includes -----*/

/* Exported functions prototypes -----*/
// moved to sapi3c_board.h / .c
// void Error_Handler(void);

/* Private defines -----*/
// moved to sapi3c_board.h
// #define SPI_RES_Pin LL_GPIO_PIN_13
// .. etc.

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */

```

**C.3.1c) Editar sapi3c.h para que solo incluya \_board y \_gpio (esto lo hacemos con un #ifdef SECOND\_TEST, por ahora). Esto evita problemas porque no tenemos todavía las demás rutinas resaltadas en celeste.**

**Editamos:** ✓

C:\CESE2018\_TrFinal\2019\Tareas\_dde062019\sAPI\_3C\sAPI3C\base\inc\sapi3c.h

```

#ifndef _SAPI_H_
#define _SAPI_H_

/*===== [inclusions] =====*/

#include "sapi3c_datatypes.h"

// Peripheral Drivers

#include "sapi3c_board.h"           // Use clock peripheral

#include "sapi3c_gpio.h"           // Use GPIO peripherals

#ifdef SECOND_TEST
#include "sapi3c_uart.h"           // Use UART peripherals
#include "sapi3c_adc.h"           // Use ADC1 peripheral
#include "sapi3c_i2c.h"           // Use I2C1 peripheral
#include "sapi3c_spi.h"           // Use SPI1 peripheral
#include "sapi3c_rtc.h"           // Use RTC peripheral

// High Level drivers

#include "sapi3c_stdio.h"          // Use sapi_uart module
#include "sapi3c_consolePrint.h"   // Use sapi_print module
#include "sapi3c_convert.h"        // Use <string.h>

#include "sapi3c_delay.h"          // Use sapi_tick module
#include "sapi3c_circularBuffer.h" // It has no dependencies

// External Peripheral Drivers

#include "sapi3c_keypad.h"         // Use sapi_gpio and sapi_delay modules
#include "sapi3c_ili9341.h"        // Use sapi_gpio peripherals

// #include "sapi_esp8266.h"       // Use sapi_uart module

#endif

#ifdef __cplusplus

```

```
extern "C" {
#endif

#ifdef __cplusplus
}
#endif
#endif /* #ifndef _SAPI_H_ */
```

### C.3.1d) Dentro de sapi3c\_board.c ->

**Modificar la función boardInitCL3() a utilizar para que llame primero a HAL\_Init(), luego a SystemClock\_Config(), luego los GPIOs para CL3. Editar lo siguiente para que quede así:**



```
/**
 * *****
 * @file sapi3c_board.c
 * @author E.Pernia / adapted by R.Oliva
 * @brief This file holds board-specific configuration for CL3
 * rev. 4.8.19
 *
 */

/* Date: 2019-06-10 */

/*=====[inclusions]=====*/

#include "sapi3c_board.h"
#include "sapi3c_gpio.h"

/*=====[external functions definition]=====*/

/**
 * @brief boardInitCL3() - Set up and initialize CL3 board hardware
 * First call the system Clock configuration,
 * Then enable all gpio_inputs (clocks & reset outputs)
 * Then configure all gpio_inputs, all gpio_outputs
 * Finally the -INT1 EXTI interrupt
 * @param None
 * @retval None
 */
void boardInitCL3(void)
{
    // STM32 HAL initialization
    HAL_Init();
    // STM32 System Clock Config STM32F411RE on CL3
    SystemClock_Config();

    // Configure GPIO pins for CL3 board

    // Inicializar GPIOs
    // en sAPI: gpioInit( 0, GPIO_ENABLE );
    gpioInitEnable();

    // Configuracion de pines de entrada de CL3
    // Similar to gpioInit( DI0, GPIO_INPUT ); in EDUCIAA
    /*
        KBD_ABJ    = 0,    // K_ABJ_PB0_Pin
        KBD_DER    = 1,    // K_DER_PB1_Pin
        KBD_IZQ    = 2,    // K_IZQ_PC4_Pin
        KBD_ARR    = 3,    // K_ARR_PC5_Pin
        SDIO_CD     = 4,    // SDIO_CD_Pin_PC11 (former CS)
        SDIO_INS    = 5     // SD_INS_Pin_PC10
        and config_pull can be LL_GPIO_PULL_UP or LL_GPIO_PULL_NO
    */
```



```

    gpioInitInput( KBD_ABJ, LL_GPIO_PULL_UP);
    gpioInitInput( KBD_DER, LL_GPIO_PULL_UP);
    gpioInitInput( KBD_IQZ, LL_GPIO_PULL_UP);
    gpioInitInput( KBD_ARR, LL_GPIO_PULL_UP);
    gpioInitInput( SDIO_CD, LL_GPIO_PULL_UP);
    gpioInitInput( SDIO_INS, LL_GPIO_PULL_NO);

// Configuración de pines de salida de CL3
// bool_t gpioInitOutput( outputMap_t output);
// Similar a gpioInit( DO0, GPIO_OUTPUT );
// EXP_PW_PA0 = 0, // PA0_EN_EXP_PW_Pin
// LCD_PW_PA1 , // PA1_EN_LCD_PW_Pin
// OLED_PB2 , // PB2_OLED_Pin
// IOT_PW_PB8 , // PB8_EN_IOT_PW_Pin
// SER_PW_PB9 , // PB9_EN_SER_PWR_Pin
// RS485_DE_PB10, // PB10_RS485_DE_Pin
// LCD_CLK_PB12 , // PB12_LCD_CLK_pin
// LCD_DATA_PB13, // PB13_LCD_DATA_pin
// SPI_CS_PB14 , // PB14_(exLDC_E)_SPI_CS_Pin
// SPI_DC_PB15 , // PB15_(exLDC_BL)_SPI_DC_Pin
// SPI_RES_PC13 // PC13_(exSD_WREN)_SPI_RES_Pin

    gpioInitOutput(EXP_PW_PA0);
    gpioInitOutput(LCD_PW_PA1);
    gpioInitOutput(OLED_PB2);
    gpioInitOutput(IOT_PW_PB8);
    gpioInitOutput(SER_PW_PB9);
    gpioInitOutput(RS485_DE_PB10);
    gpioInitOutput(LCD_CLK_PB12);
    gpioInitOutput(LCD_DATA_PB13);
    gpioInitOutput(SPI_CS_PB14);
    gpioInitOutput(SPI_DC_PB15);
    gpioInitOutput(SPI_RES_PC13);

// Configuración Entrada _INT1 de Interrupcion
gpioInit_INT1();

}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
    RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
    */
    __HAL_RCC_PWR_CLK_ENABLE();
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
    /** Initializes the CPU, AHB and APB busses clocks
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE|RCC_OSCILLATORTYPE_LSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.LSEState = RCC_LSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = 4;
    RCC_OscInitStruct.PLL.PLLN = 84;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 4;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }
    /** Initializes the CPU, AHB and APB busses clocks
    */

```

```

RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
{
    Error_Handler();
}
PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_RTC;
PeriphClkInitStruct.RTCClockSelection = RCC_RTCCCLKSOURCE_LSE;
if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
{
    Error_Handler();
}
}

/**
 * @brief Period elapsed callback in non blocking mode
 * @note This function is called when TIM1 interrupt took place, inside
 * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
 * a global variable "uwTick" used as application time base.
 * @param htim : TIM handle
 * @retval None
 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM1) {
        HAL_IncTick();
    }
}

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
}

#ifdef USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line number
 * where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
}
#endif /* USE_FULL_ASSERT */

/*=====[end of file]=====*/

```

### C.3.1e) El archivo sapi3c\_board.h tiene lo siguiente ahora:

```

#ifndef _SAPI_BOARD_H_
#define _SAPI_BOARD_H_

/*=====[inclusions]=====*/

#include "sapi3c_datatypes.h"

```

```

/*=====[cplusplus]=====*/

#ifdef __cplusplus
extern "C" {
#endif

/*=====[macros]=====*/

#define boardConfigCL3 boardInitCL3

/* Private defines -----*/
#define SPI_RES_Pin LL_GPIO_PIN_13
#define SPI_RES_GPIO_Port GPIOC
#define AN10_PC0_Pin LL_GPIO_PIN_0
#define AN10_PC0_GPIO_Port GPIOC
#define AN11_PC1_Pin LL_GPIO_PIN_1
#define AN11_PC1_GPIO_Port GPIOC
#define AN12_PC2_Pin LL_GPIO_PIN_2
#define AN12_PC2_GPIO_Port GPIOC
#define AN13_PC3_Pin LL_GPIO_PIN_3
#define AN13_PC3_GPIO_Port GPIOC
#define PA0_EN_EXP_PW_Pin LL_GPIO_PIN_0
#define PA0_EN_EXP_PW_GPIO_Port GPIOA
#define PA1_EN_LCD_PW_Pin LL_GPIO_PIN_1
#define PA1_EN_LCD_PW_GPIO_Port GPIOA
#define AN4_PA4_Pin LL_GPIO_PIN_4
#define AN4_PA4_GPIO_Port GPIOA
#define AN5_PA5_Pin LL_GPIO_PIN_5
#define AN5_PA5_GPIO_Port GPIOA
#define AN6_PA6_Pin LL_GPIO_PIN_6
#define AN6_PA6_GPIO_Port GPIOA
#define AN7_PA7_Pin LL_GPIO_PIN_7
#define AN7_PA7_GPIO_Port GPIOA
#define K_IZQ_PC4_Pin LL_GPIO_PIN_4
#define K_IZQ_PC4_GPIO_Port GPIOC
#define K_ARR_PC5_Pin LL_GPIO_PIN_5
#define K_ARR_PC5_GPIO_Port GPIOC
#define K_ABJ_PB0_Pin LL_GPIO_PIN_0
#define K_ABJ_PB0_GPIO_Port GPIOB
#define K_DER_PB1_Pin LL_GPIO_PIN_1
#define K_DER_PB1_GPIO_Port GPIOB
#define OLED_Pin LL_GPIO_PIN_2
#define OLED_GPIO_Port GPIOB
#define RS485_DE_Pin LL_GPIO_PIN_10
#define RS485_DE_GPIO_Port GPIOB
#define LCD_DATA_Pin LL_GPIO_PIN_12
#define LCD_DATA_GPIO_Port GPIOB
#define LCD_CLK_Pin LL_GPIO_PIN_13
#define LCD_CLK_GPIO_Port GPIOB
#define SPI_CS_Pin LL_GPIO_PIN_14
#define SPI_CS_GPIO_Port GPIOB
#define SPI_DC_Pin LL_GPIO_PIN_15
#define SPI_DC_GPIO_Port GPIOB
#define INT1_N_Pin LL_GPIO_PIN_15
#define INT1_N_GPIO_Port GPIOA
#define SD_INS_Pin LL_GPIO_PIN_10
#define SD_INS_GPIO_Port GPIOC
#define SDIO_CD_Pin LL_GPIO_PIN_11
#define SDIO_CD_GPIO_Port GPIOC
#define SPI_CLK_Pin LL_GPIO_PIN_3
#define SPI_CLK_GPIO_Port GPIOB
#define SPI_MISO_Pin LL_GPIO_PIN_4
#define SPI_MISO_GPIO_Port GPIOB
#define SPI_MOSI_Pin LL_GPIO_PIN_5
#define SPI_MOSI_GPIO_Port GPIOB
#define PB8_EN_IOT_PW_Pin LL_GPIO_PIN_8
#define PB8_EN_IOT_PW_GPIO_Port GPIOB
#define PB9_EN_SER_PWR_Pin LL_GPIO_PIN_9
#define PB9_EN_SER_PWR_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */

```

```

/*=====[typedef]=====*/

/* Defined for sapi3c_uart.h */

typedef enum {
    UART_TER = 0, // Hardware UART6 (RS232_2 connector - Terminal)
    UART_485 = 1, // Hardware UART1 via RS_485 A, B and GND Borns
    UART_2 = 2 // Hardware UART2 (RS232_1 connector or ESP8266)
} uartMap_t;

typedef enum {
    KBD_ABJ = 0, // K_ABJ_PB0_Pin
    KBD_DER = 1, // K_DER_PB1_Pin
    KBD_IQZ = 2, // K_IQZ_PC4_Pin
    KBD_ARR = 3, // K_ARR_PC5_Pin
    SDIO_CD = 4, // SDIO_CD_Pin_PC11 (former CS)
    SDIO_INS = 5 // SD_INS_Pin_PC10
} inputMap_t;

// CL3_Board output map
// LL_GPIO_ResetOutputPin(GPIOA, PA0_EN_EXP_PW_Pin|PA1_EN_LCD_PW_Pin);
// LL_GPIO_ResetOutputPin(GPIOB, OLED_Pin = PB.2
// |PB8_EN_IOT_PW_Pin
// |PB9_EN_SER_PWR_Pin);
// |RS485_DE_Pin = PB.10
// |LCD_DATA_Pin = PB.12
// |LCD_CLK_Pin = PB.13
// |SPI_CS_Pin = oldLCD_E =PB.14
// |SPI_DC_Pin = oldLCD_BL =PB.15
// LL_GPIO_ResetOutputPin(GPIOC, SPI_RES_Pin); oldSD_WREN = PC.13

typedef enum {
    EXP_PW_PA0 = 0, // PA0_EN_EXP_PW_Pin
    LCD_PW_PA1 , // PA1_EN_LCD_PW_Pin
    OLED_PB2 , // PB2_OLED_Pin
    IOT_PW_PB8 , // PB8_EN_IOT_PW_Pin
    SER_PW_PB9 , // PB9_EN_SER_PWR_Pin
    RS485_DE_PB10, // PB10_RS485_DE_Pin
    LCD_CLK_PB12 , // PB12_LCD_CLK_pin
    LCD_DATA_PB13, // PB13_LCD_DATA_pin
    SPI_CS_PB14 , // PB14_(exLDC_E)_SPI_CS_Pin
    SPI_DC_PB15 , // PB15_(exLDC_BL)_SPI_DC_Pin
    SPI_RES_PC13 // PC13_(exSD_WREN)_SPI_RES_Pin
} outputMap_t;

/*=====[external data declaration]=====*/
/*=====[external functions declaration]=====*/

void boardInitCL3(void);
void SystemClock_Config(void);
void Error_Handler(void);

/*=====[cplusplus]=====*/
#ifdef __cplusplus
}
#endif
/*=====[end of file]=====*/
#endif /* #ifndef _SAPI_BOARD_H_ */

```

### C.3.1f) El archivo `sapi3c_gpio.h` tiene lo siguiente ahora (ver agregado resaltado): ✓

```

/* Define to prevent recursive inclusion -----*/
#ifndef __gpio_H
#define __gpio_H
#ifdef __cplusplus

```

```

extern "C" {
#endif

/* Includes -----*/
#include "sapi3c_datatypes.h"
#include "sapi3c_board.h"

// void MX_GPIO_Init(void);
void gpioInitEnable(void);
bool_t gpioInitInput( inputMap_t input, uint32_t config_pull );
bool_t gpioInitOutput( outputMap_t output);
void gpioInit_INT1(void);
bool_t gpioRead( inputMap_t input );
bool_t gpioWrite( outputMap_t output, bool_t value);

/* USER CODE BEGIN Prototypes */

/* USER CODE END Prototypes */

#ifdef __cplusplus
}
#endif
#endif /*__ pinoutConfig_H */

```

**C.3.1g) El archivo sapi3c\_gpio.c** ✓ -> ver detalles en: Tareas\_Avances(sAPI)\_Gpio(B)\_04-08-2019.docx

### **C.3.2) Ensayos –**

**C.3.2a ) Desde WindowsExplorer, 1ro crear directorio /lib en Workspace a utilizar (eliminamos el anterior, creado internamente desde el Eclipse), en**













**C:\Work\_EmbdSTM32\2019\CL3\sAPI3C\_BM\_ej1**  
**(o sea debe aparecer:**

**C:\Work\_EmbdSTM32\2019\CL3\sAPI3C\_BM\_ej1\lib )** ✓







**C.3.2b ) copiar lo modificado en DB01, al Workspace, dentro del directorio**

**C:\Work\_EmbdSTM32\2019\CL3\sAPI3C\_BM\_ej1\lib** ✓











**C.3.2c ) en el directorio /Src principal, dejar solamente: main.c, y los archivos propios de stm32f4xx mas syscalls.c (por ahora, después esos irían al /sapi3c\_board/src )** ✓

Work_EmbdSTM32 ▶ 2019 ▶ CL3 ▶ sAPI3C_BM_ej1 ▶ Src				
Compartir con ▼ Grabar Nueva carpeta				
Nombre	Fecha de modifica...	Tipo	Tamaño	
 adc	01/08/2019 8:28	Archivo C	5 KB	
 i2c	01/08/2019 8:28	Archivo C	5 KB	
 main	03/08/2019 19:19	Archivo C	7 KB	
 rtc	01/08/2019 8:28	Archivo C	3 KB	
 sdio	01/08/2019 8:28	Archivo C	4 KB	
 spi	01/08/2019 8:28	Archivo C	4 KB	
 stm32f4xx_hal_msp	01/08/2019 8:28	Archivo C	3 KB	
 stm32f4xx_hal_timebase_tim	01/08/2019 8:28	Archivo C	5 KB	
 stm32f4xx_it	01/08/2019 8:28	Archivo C	7 KB	
 syscalls	01/08/2019 8:28	Archivo C	5 KB	
 system_stm32f4xx	13/04/2019 11:29	Archivo C	28 KB	
 usart	01/08/2019 8:28	Archivo C	6 KB	

### C.3.2d) Borrar este main.c actual y copiar el nuevo muy sencillo generado en C.3.1b) ✓

Work_EmbdSTM32 ▶ 2019 ▶ CL3 ▶ sAPI3C_BM_ej1 ▶ Src				
Compartir con ▼ Grabar Nueva carpeta				
Nombre	Fecha de modifica...	Tipo	Tamaño	
 main	04/08/2019 12:41	Archivo C	2 KB	
 stm32f4xx_hal_msp	01/08/2019 8:28	Archivo C	3 KB	
 stm32f4xx_hal_timebase_tim	01/08/2019 8:28	Archivo C	5 KB	
 stm32f4xx_it	01/08/2019 8:28	Archivo C	7 KB	
 syscalls	01/08/2019 8:28	Archivo C	5 KB	
 system_stm32f4xx	13/04/2019 11:29	Archivo C	28 KB	

### C.3.2e) Ir al directorio /Inc principal, dejar solamente main.h, y los archivos propios de stm32f4xx (por ahora, después esos irían al /sapi3c\_board/inc ):

Work_EmbdSTM32 ▶ 2019 ▶ CL3 ▶ sAPI3C_BM_ej1 ▶ Inc				
Compartir con ▼ Grabar Nueva carpeta				
Nombre	Fecha de modifica...	Tipo	Tamaño	
 adc	01/08/2019 8:28	C++ Header File	2 KB	
 i2c	01/08/2019 8:28	C++ Header File	2 KB	
 main	01/08/2019 8:28	C++ Header File	5 KB	
 rtc	01/08/2019 8:28	C++ Header File	2 KB	
 sdio	01/08/2019 8:28	C++ Header File	2 KB	
 spi	01/08/2019 8:28	C++ Header File	2 KB	
 stm32_assert	01/08/2019 8:28	C++ Header File	3 KB	
 stm32f4xx_hal_conf	01/08/2019 8:28	C++ Header File	17 KB	
 stm32f4xx_it	01/08/2019 8:28	C++ Header File	3 KB	
 usart	01/08/2019 8:28	C++ Header File	2 KB	

**C.3.2f) En este directorio /inc editar el main.h. Por ahora dejamos los .h del sistema, pero eliminamos el prototipo del ErrorHandler() (ahora estará en sapi3c\_board.h) y los private defines de los pines que van al mismo lugar. ✓**

```
/* Define to prevent recursive inclusion -----*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----*/
#include "stm32f4xx_hal.h"
#include "stm32f4xx_hal.h"
#include "stm32f4xx_ll_usart.h"
#include "stm32f4xx_ll_rcc.h"
#include "stm32f4xx.h"
#include "stm32f4xx_ll_system.h"
#include "stm32f4xx_ll_gpio.h"
#include "stm32f4xx_ll_exti.h"
#include "stm32f4xx_ll_bus.h"
#include "stm32f4xx_ll_cortex.h"
#include "stm32f4xx_ll_utils.h"
#include "stm32f4xx_ll_pwr.h"
#include "stm32f4xx_ll_dma.h"

/* Exported functions prototypes -----*/
// void ErrorHandler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines -----*/
// #define SPI_RES_Pin LL_GPIO_PIN_13
// .. etc





/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif

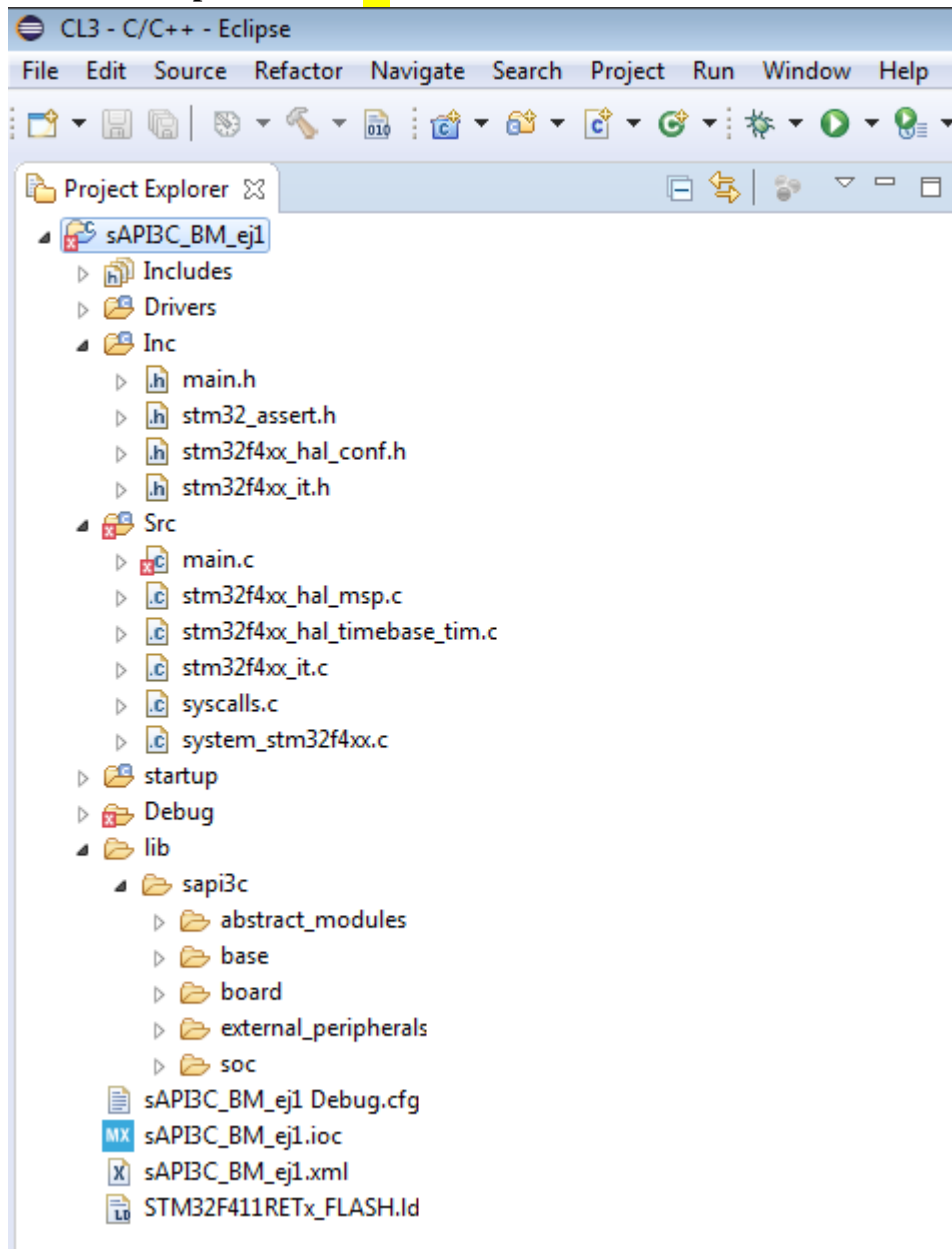
#endif /* __MAIN_H */
```

Work\_EmbdSTM32 ▶ 2019 ▶ CL3 ▶ sapi3C\_BM\_ej1 ▶ Inc

Compartir con ▼   Grabar   Nueva carpeta

Nombre	Fecha de modifica...	Tipo	Tamaño
 main	04/08/2019 12:36	C++ Header File	3 KB
 stm32_assert	01/08/2019 8:28	C++ Header File	3 KB
 stm32f4xx_hal_conf	01/08/2019 8:28	C++ Header File	17 KB
 stm32f4xx_it	01/08/2019 8:28	C++ Header File	3 KB

**C.3.2g) Compilación inicial: Cerrar editores y abrir AC6 en el proyecto sAPI3C\_BM\_ej1, y dar un “refresh” – Aparece así: .** ✓

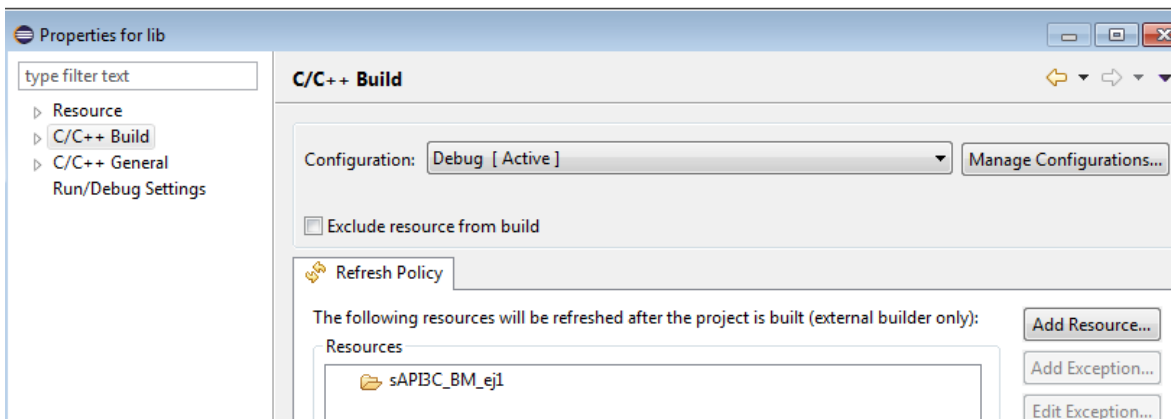


**C.3.2h) Ahora cumplimos los pasos 5) y 6) de C.1) o sea:**

1. Se va para cada directorio “nuevo” eliminando el “exclude from build” (right click en cada directorio, C/C++Build, destildar el “exclude resource from build”).
2. Finalmente, parado en el directorio del proyecto, en properties del Proyecto, C/C++ Build-> settings->MCU GCC-> Includes --- presionar el “+” para agregar los .h que se incorporan de los nuevos directorios.

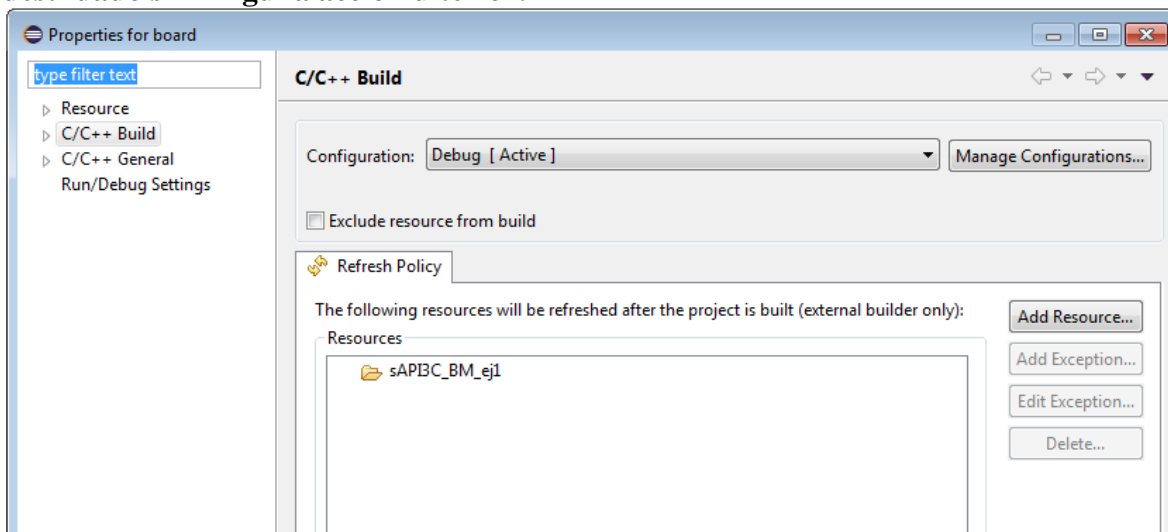
**C.3.2h.1) Sería como que repetir los pasos de A.4.5) Una vez copiado /lib/sapi3c desde el directorio preliminar A3 dentro de Eclipse, en properties de /lib->C/C++Build destildamos Exclude, o sea hacemos:**



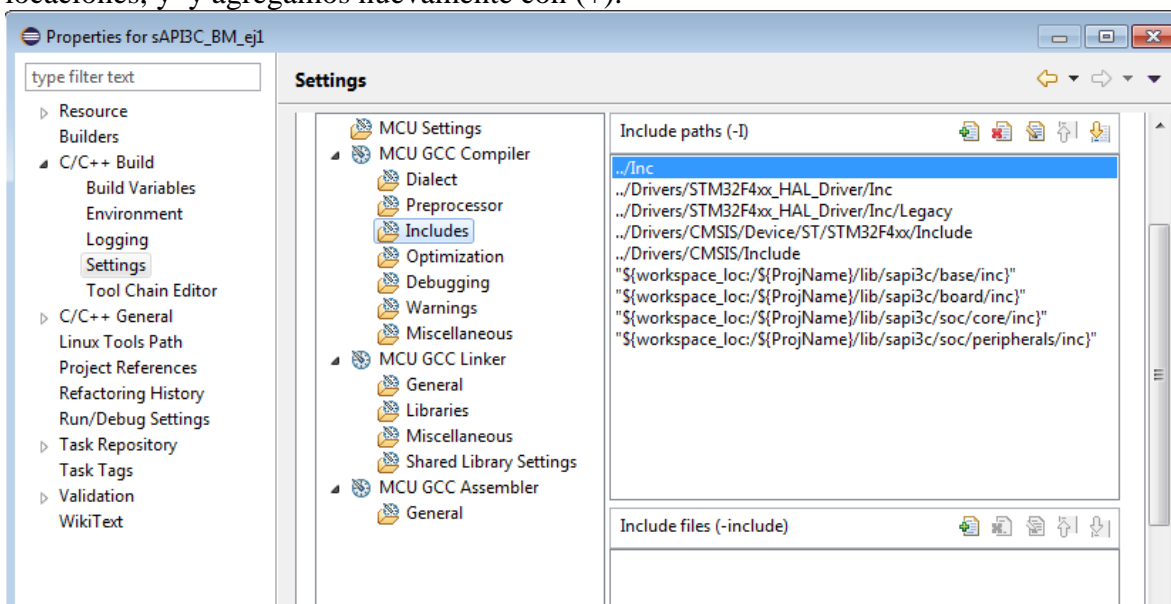


Y apretar Apply -> OK. . ✓

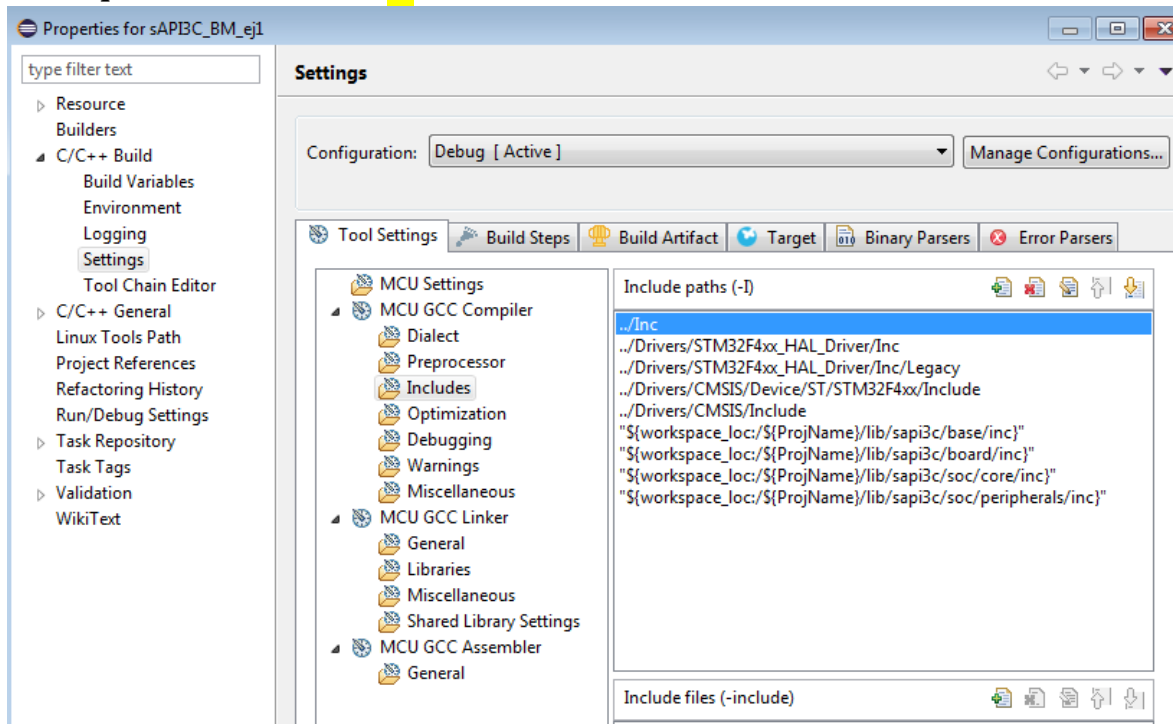
Esto aparentemente incluye automáticamente los subdirectorios.. por ejemplo / board aparece destildado sin ninguna acción ulterior:



**C.3.2h.2) Ahora aplicamos el C.1.6 “..parado en el directorio del proyecto, en properties del Proyecto, C/C++ Build-> settings->MCU GCC-> Includes --- presionar el “+” para agregar los .h que se incorporan de los nuevos directorios.”** Esto ya estaba hecho como en A.4, pero por las dudas deshacemos con (-) para las 4 locaciones, y y agregamos nuevamente con (+).



Y nos queda casi idéntico: . ✓



### C.3.3) Compilación y Arreglos – Compilación 1

ERRORES:

```
../lib/sapi3c/soc/peripherals/src/sapi3c_gpio.c:55:42: note: in expansion of macro 'RS485_DE_Pin'
  LL_GPIO_ResetOutputPin(GPIOB, OLED_Pin|RS485_DE_Pin|LCD_DATA_Pin|LCD_CLK_Pin
  ~~~~~
C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/lib/sapi3c/board/inc/sapi3c_board.h:90:22: error:
'LL_GPIO_PIN_12' undeclared (first use in this function); did you mean 'LL_GPIO_PIN_10'?
#define LCD_DATA_Pin LL_GPIO_PIN_12
```

#### C.3.3.1) Correccion 1 Intentamos: Agregar en sapi3c\_board.h lo que esta en main.h

```
/* Includes -----*/
#include "stm32f4xx_hal.h"
#include "stm32f4xx_hal.h"
#include "stm32f4xx_ll_usart.h"
#include "stm32f4xx_ll_rcc.h"
#include "stm32f4xx.h"
#include "stm32f4xx_ll_system.h"
#include "stm32f4xx_ll_gpio.h"
#include "stm32f4xx_ll_exti.h"
#include "stm32f4xx_ll_bus.h"
#include "stm32f4xx_ll_cortex.h"
#include "stm32f4xx_ll_utils.h"
#include "stm32f4xx_ll_pwr.h"
#include "stm32f4xx_ll_dma.h"
```

Que tiene todas estas definiciones

#### C.3.3.2) Compilación 2

```
15:05:57 **** Incremental Build of configuration Debug for project sAPI3C_BM_ej1 ****
make all
Building file: ../lib/sapi3c/soc/peripherals/src/sapi3c_gpio.c
Invoking: MCU GCC Compiler
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULL_LL_DRIVER '-D_weak=__attribute__((weak))' '-D_packed=__attribute__((packed__))' -DUSE_HAL_DRIVER -DSTM32F411xE -I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/Inc" -
```

```

I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/Drivers/STM32F4xx_HAL_Driver/Inc" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/Drivers/STM32F4xx_HAL_Driver/Inc/Legacy" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/Drivers/CMSIS/Device/ST/STM32F4xx/Include" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/Drivers/CMSIS/Include" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/lib/sapi3c/base/inc" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/lib/sapi3c/board/inc" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/lib/sapi3c/soc/core/inc" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/lib/sapi3c/soc/peripherals/inc" -O3 -g3 -Wall -fmessage-
length=0 -ffunction-sections -c -fmessage-length=0 -MMD -MP -
MF"lib/sapi3c/soc/peripherals/src/sapi3c_gpio.d" -MT"lib/sapi3c/soc/peripherals/src/sapi3c_gpio.o" -o
"lib/sapi3c/soc/peripherals/src/sapi3c_gpio.o" "../lib/sapi3c/soc/peripherals/src/sapi3c_gpio.c"
../lib/sapi3c/soc/peripherals/src/sapi3c_gpio.c: In function 'gpioInitEnable':
../lib/sapi3c/soc/peripherals/src/sapi3c_gpio.c:39:23: warning: unused variable 'GPIO_InitStruct' [-
Wunused-variable]
    LL_GPIO_InitTypeDef GPIO_InitStruct = {0};
                        ^~~~~~
../lib/sapi3c/soc/peripherals/src/sapi3c_gpio.c: In function 'gpioInitInput':
../lib/sapi3c/soc/peripherals/src/sapi3c_gpio.c:107:28: error: 'SDIO_CD_Pin_PC11' undeclared (first use
in this function); did you mean 'SDIO_CD_Pin'?
    GPIO_InitStruct.Pin = SDIO_CD_Pin_PC11;
                        ^~~~~~
                        SDIO_CD_Pin
../lib/sapi3c/soc/peripherals/src/sapi3c_gpio.c:107:28: note: each undeclared identifier is reported only
once for each function it appears in
../lib/sapi3c/soc/peripherals/src/sapi3c_gpio.c: In function 'gpioRead':
../lib/sapi3c/soc/peripherals/src/sapi3c_gpio.c:286:44: error: 'SDIO_CD_Pin_PC11' undeclared (first use
in this function); did you mean 'SDIO_CD_Pin'?
    ret_val = LL_GPIO_IsInputPinSet(GPIOC, SDIO_CD_Pin_PC11);
                                           ^~~~~~
                                           SDIO_CD_Pin
lib/sapi3c/soc/peripherals/src/subdir.mk:18: recipe for target
'lib/sapi3c/soc/peripherals/src/sapi3c_gpio.o' failed
make: *** [lib/sapi3c/soc/peripherals/src/sapi3c_gpio.o] Error 1

```

### C.3.3.3) Correccion 2

-Elimino en la función gpioInitEnable() la variable no usada 'GPIO\_InitStruct'  
 -Corrijo SDIO\_CD\_Pin\_PC11 por SDIO\_CD\_Pin

### C.3.3.4) Compilación 3

```

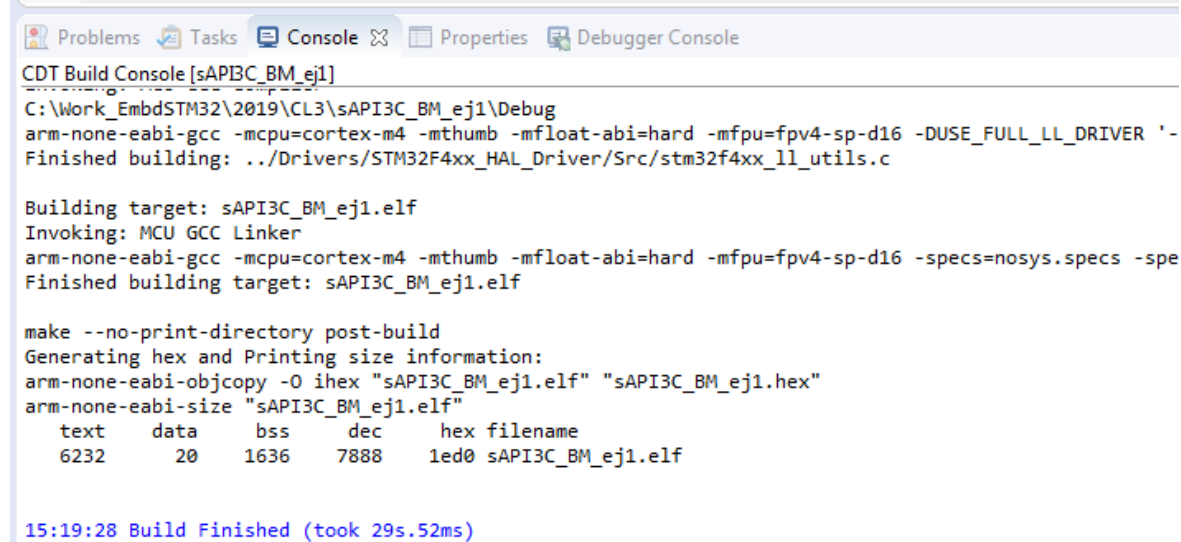
Building file: ../lib/sapi3c/soc/core/src/sapi3c_sleep.c
Invoking: MCU GCC Compiler
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULL_LL_DRIVER '-
D__weak=__attribute__((weak))' '-D__packed=__attribute__((__packed__))' -DUSE_HAL_DRIVER -DSTM32F411xE -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/Inc" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/Drivers/STM32F4xx_HAL_Driver/Inc" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/Drivers/STM32F4xx_HAL_Driver/Inc/Legacy" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/Drivers/CMSIS/Device/ST/STM32F4xx/Include" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/Drivers/CMSIS/Include" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/lib/sapi3c/base/inc" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/lib/sapi3c/board/inc" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/lib/sapi3c/soc/core/inc" -
I"C:/Work_EmbdSTM32/2019/CL3/sAPI3C_BM_ej1/lib/sapi3c/soc/peripherals/inc" -O3 -g3 -Wall -fmessage-
length=0 -ffunction-sections -c -fmessage-length=0 -MMD -MP -MF"lib/sapi3c/soc/core/src/sapi3c_sleep.d" -
MT"lib/sapi3c/soc/core/src/sapi3c_sleep.o" -o "lib/sapi3c/soc/core/src/sapi3c_sleep.o"
"../lib/sapi3c/soc/core/src/sapi3c_sleep.c"
In file included from ../lib/sapi3c/soc/core/src/sapi3c_sleep.c:41:0:
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\lib\sapi3c\soc\core\inc\sapi3c_sleep.h:45:10: fatal error:
sapi3c_peripheral_map.h: No such file or directory
#include "sapi3c_peripheral_map.h"
      ^~~~~~
compilation terminated.
make: *** [lib/sapi3c/soc/core/src/sapi3c_sleep.o] Error 1
lib/sapi3c/soc/core/src/subdir.mk:18: recipe for target 'lib/sapi3c/soc/core/src/sapi3c_sleep.o' failed

```

### C.3.3.5) Correccion 4

-Elimino el include de "sapi3c\_peripheral\_map.h" por el de sapi3c\_board.h en sapi3c\_sleep.c

### C.3.3.5) Compilación 4 – Genera Salida OK!



```
CDT Build Console [sAPI3C_BM_ej1]
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULL_LL_DRIVER '-
Finished building: ../Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_ll_utils.c

Building target: sAPI3C_BM_ej1.elf
Invoking: MCU GCC Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -specs=nosys.specs -spe
Finished building target: sAPI3C_BM_ej1.elf

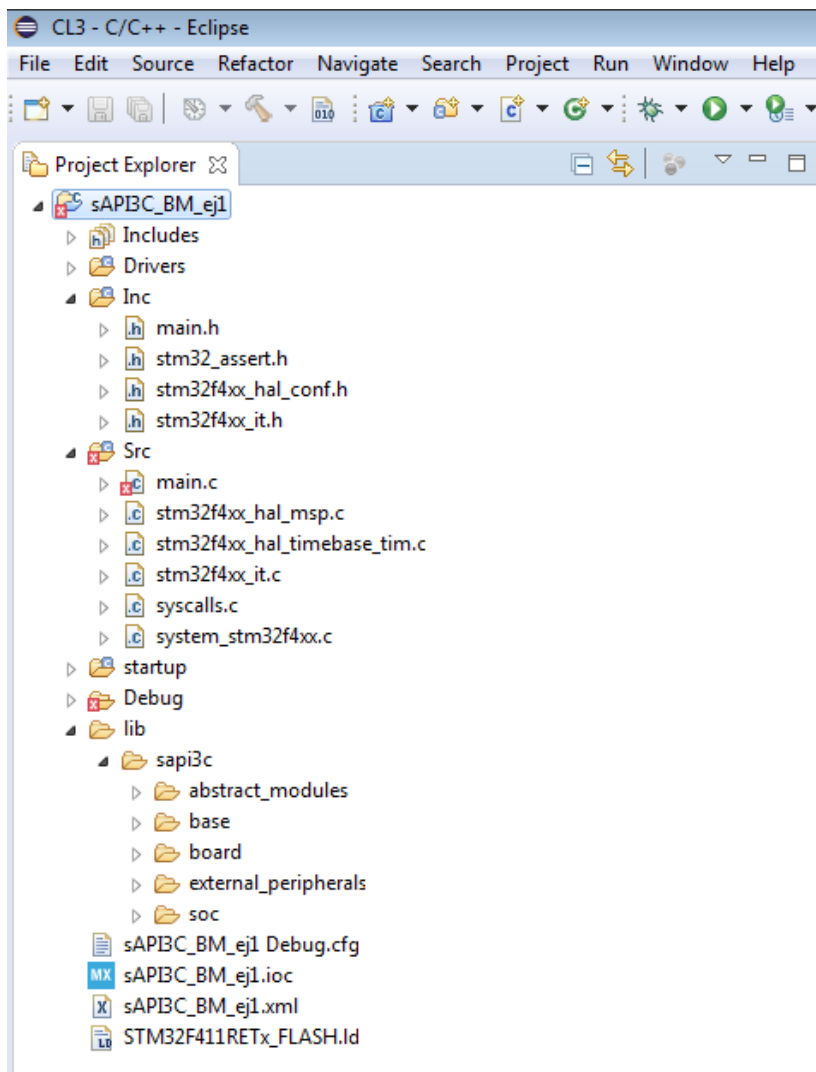
make --no-print-directory post-build
Generating hex and Printing size information:
arm-none-eabi-objcopy -O ihex "sAPI3C_BM_ej1.elf" "sAPI3C_BM_ej1.hex"
arm-none-eabi-size "sAPI3C_BM_ej1.elf"
   text    data     bss     dec     hex filename
   6232     20    1636    7888    1ed0 sAPI3C_BM_ej1.elf

15:19:28 Build Finished (took 29s.52ms)
```

### C.3.3.6) Funciona OK! .. ✓

(parpadea OLED cada 0.5 seg, nada mas)– Hacemos backup de este Project en \_Test01ok1

C.4 – Nos queda intentar “limpiar” los /Src y /Inc, como se muestra en la figura:



**llevando los archivos stm32f4xx .h al directorio /lib/board/inc, y los stm32f4xx .c, syscalls.c y system\_xx.c al directorio /lib/board/src, – y dejando en /Src, /Inc solo los main.c/.h**

#### C.4.1 – Lo intentamos, siempre desde el Windows Explores y con el AC6 cerrado. Abrimos de nuevo AC6, damos refresh, y con el martillito..

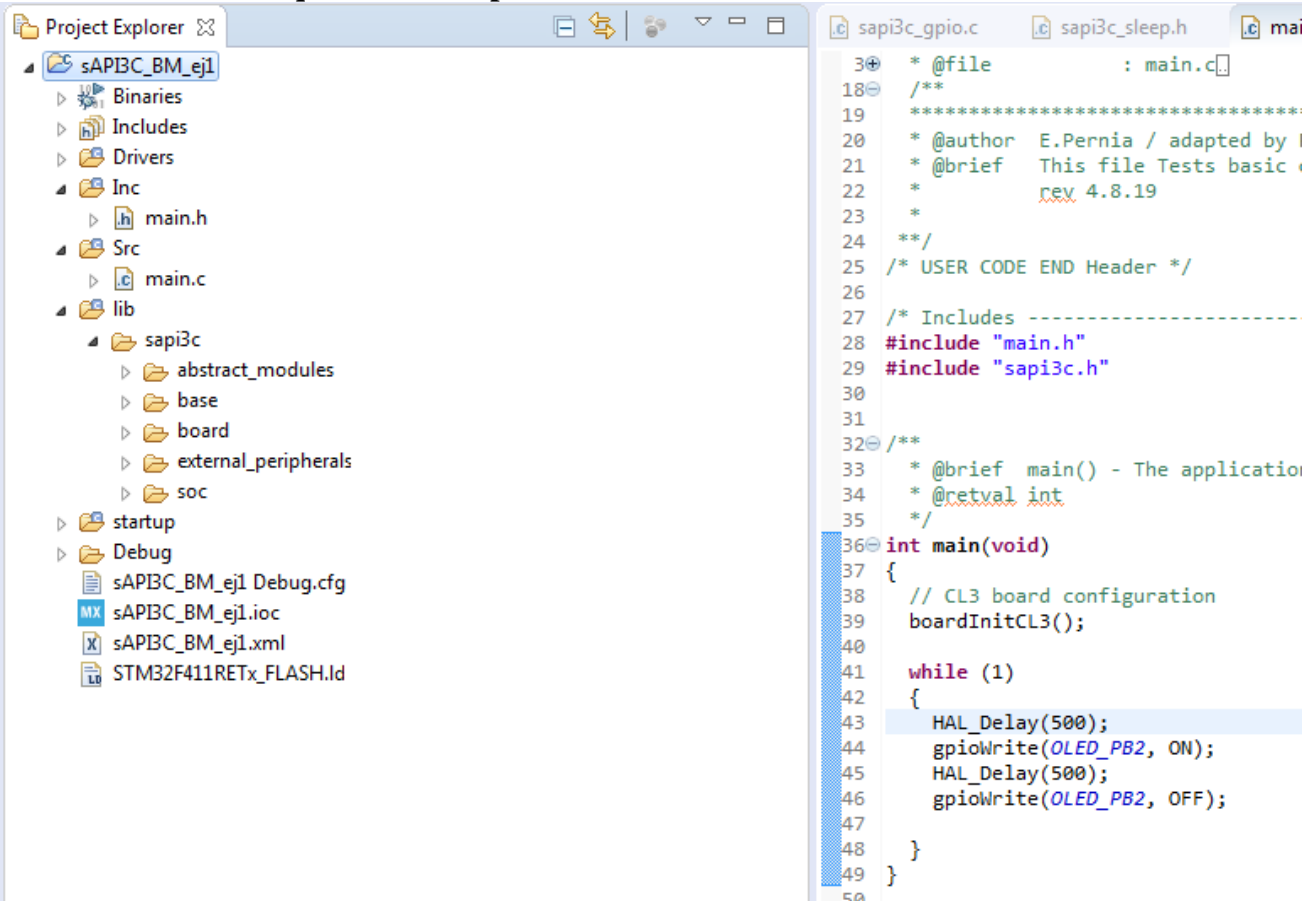
```
Problems Tasks Console Properties Debugger Console
CDT Build Console [sAPI3C_BM_ej1]
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -DUSE_FULL_LL
Finished building: ../Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_ll_utils.c

Building target: sAPI3C_BM_ej1.elf
Invoking: MCU GCC Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -specs=nosys.:
Finished building target: sAPI3C_BM_ej1.elf

make --no-print-directory post-build
Generating hex and Printing size information:
arm-none-eabi-objcopy -O ihex "sAPI3C_BM_ej1.elf" "sAPI3C_BM_ej1.hex"
arm-none-eabi-size "sAPI3C_BM_ej1.elf"
text    data    bss     dec     hex filename
6232     20     1636    7888    1ed0 sAPI3C_BM_ej1.elf

15:44:16 Build Finished (took 28s.210ms)
```

**OK! – La estructura queda mas simple:**



The screenshot shows an IDE interface. On the left is the 'Project Explorer' showing a project named 'sAPI3C\_BM\_ej1'. The project structure includes folders for 'Binaries', 'Includes', 'Drivers', 'Inc', 'Src', and 'lib'. The 'lib' folder contains a sub-folder 'sapi3c' with files 'abstract\_modules', 'base', 'board', 'external\_peripherals', and 'soc'. Other files in the project include 'main.h', 'main.c', 'sAPI3C\_BM\_ej1 Debug.cfg', 'sAPI3C\_BM\_ej1.ioc', 'sAPI3C\_BM\_ej1.xml', and 'STM32F411RETx\_FLASH.ld'. On the right, the 'sapi3c\_gpio.c' file is open, showing C code. The code includes comments for file, author, and brief, followed by include directives for 'main.h' and 'sapi3c.h'. The main function is defined as 'int main(void)' and contains a loop that calls 'boardInitCL3()', 'HAL\_Delay(500)', and 'gpioWrite(OLED\_PB2, ON)' and 'OFF'.

```
sapi3c_gpio.c sapi3c_sleep.h mai
30 * @file : main.c
18 /**
19 *****
20 * @author E.Pernia / adapted by I
21 * @brief This file Tests basic
22 * rev 4.8.19
23 *
24 **/
25 /* USER CODE END Header */
26
27 /* Includes -----
28 #include "main.h"
29 #include "sapi3c.h"
30
31
32 /**
33 * @brief main() - The application
34 * @retval int
35 */
36 int main(void)
37 {
38 // CL3 board configuration
39 boardInitCL3();
40
41 while (1)
42 {
43 HAL_Delay(500);
44 gpioWrite(OLED_PB2, ON);
45 HAL_Delay(500);
46 gpioWrite(OLED_PB2, OFF);
47 }
48 }
49
50
```

**Verificamos con un Clean,**

```

CDT Build Console [sAPI3C_BM_ej1]
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FUL
Finished building: ../Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_ll_utils.c

Building target: sAPI3C_BM_ej1.elf
Invoking: MCU GCC Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -specs=no
Finished building target: sAPI3C_BM_ej1.elf

make --no-print-directory post-build
Generating hex and Printing size information:
arm-none-eabi-objcopy -O ihex "sAPI3C_BM_ej1.elf" "sAPI3C_BM_ej1.hex"
arm-none-eabi-size "sAPI3C_BM_ej1.elf"
text      data      bss      dec      hex filename
6232      20      1636    7888    1ed0 sAPI3C_BM_ej1.elf

15:50:11 Build Finished (took 26s.643ms)

```

y luego descarga a la placa - OK! .. ✓

Hacemos backup de este Project en C:\Work\_EmbdSTM32\2019\Backups\del04-08-2019\_Test01okclean (este puede ir como primer ejemplo del Repo) –

En realidad, le vamos a cambiar el HAL\_Delay(500) por algo mas parecido a la sAPI, como delay\_cl3(500); . ✓

```

19  *****
20  * @author   E.Pernia / adapted by R.Oliva
21  * @brief    This file Tests basic operations for CL3
22  *           rev 4.8.19
23  *
24  **/
25  /* USER CODE END Header */
26
27  /* Includes -----
28  #include "main.h"
29  #include "sapi3c.h"
30
31
32  /**
33  * @brief main() - The application entry point.
34  * @retval int
35  */
36  int main(void)
37  {
38      // CL3 board configuration
39      boardInitCL3();
40
41      while (1)
42      {
43          delay_cl3(500);
44          gpioWrite(OLED_PB2, ON);
45          delay_cl3(500);
46          gpioWrite(OLED_PB2, OFF);
47      }
48  }
49  }
50

```

C4.2) En /abstract\_modules, desde fuera de AC6 agregamos los archivos sapi3c\_delay.c /h en sus respectivos /src y /inc. Primer lo hacemos en el directorio DB01:

C4.2.1) La función delay\_cl3 (uint32t delay) [en ms] es un wrapper para HAL\_Delay() y nada mas, en sapi3c\_delay.c. ✓

```

#include "sapi3c_board.h"
#include "sapi3c_delay.h"

/*=====[external functions definition]=====*/

/**
 * @brief delay_cl3() -
 * @param uint32_t delay in ms
 * @retval None
 */
void delay_cl3(uint32_t delay)
{
    // STM32 Delay with HAL
    HAL_Delay(uint32_t delay);
}

*****
* @file sapi3c_delay.h
* @author E.Pernia / adapted by R.Oliva
* @brief This file holds gpio header info for CL3 - rev 4.8.19
*
*
**/

```

#### C4.2.2) La función delay\_cl3 (uint32t delay) se declara en sapi3c\_delay.h . ✓

```

/* Date: 2019-06-10 */

/* Define to prevent recursive inclusion -----*/
#ifndef __delay3_H
#define __delay3_H
#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----*/
#include "sapi3c_datatypes.h"
#include "sapi3c_board.h"

void delay_cl3(uint32_t delay);

/* USER CODE BEGIN Prototypes */

/* USER CODE END Prototypes */

#ifdef __cplusplus
}
#endif
#endif /*__ pinoutConfig_H */

```

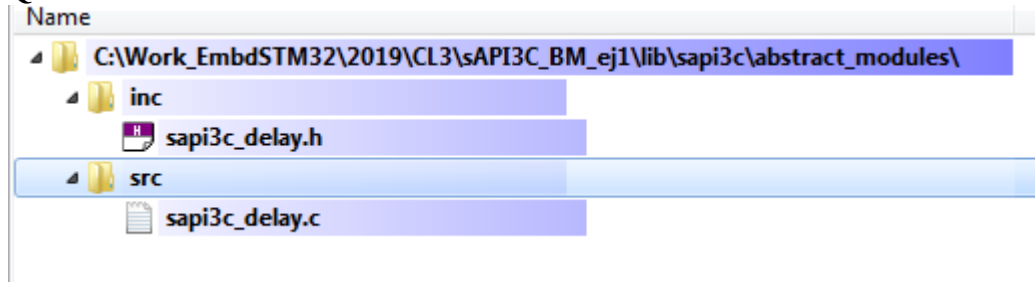
### C4.3) Copiamos al Project C:\Work\_EmbdSTM32\2019\CL3\sAPI3C\_BM\_ej1\lib\

#### C4.3.1) a C:\Work\_EmbdSTM32\2019\CL3\sAPI3C\_BM\_ej1\lib\sapi3c\abstract\_modules\inc va sapi3c\_delay.h . ✓

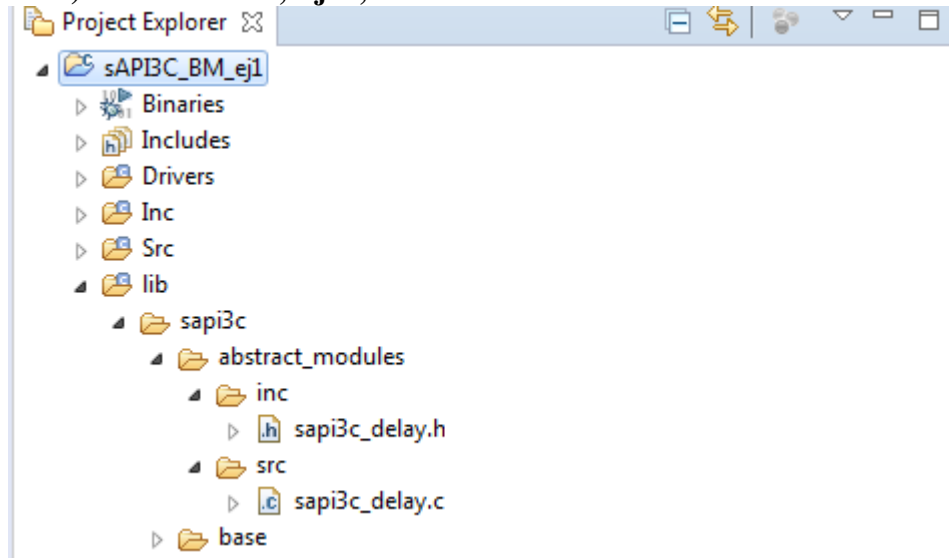


C4.3.2) a C:\Work\_EmbdSTM32\2019\CL3\sAPI3C\_BM\_ej1\lib\sapi3c\abstract\_modules\src va sapi3c\_delay.c . ✓

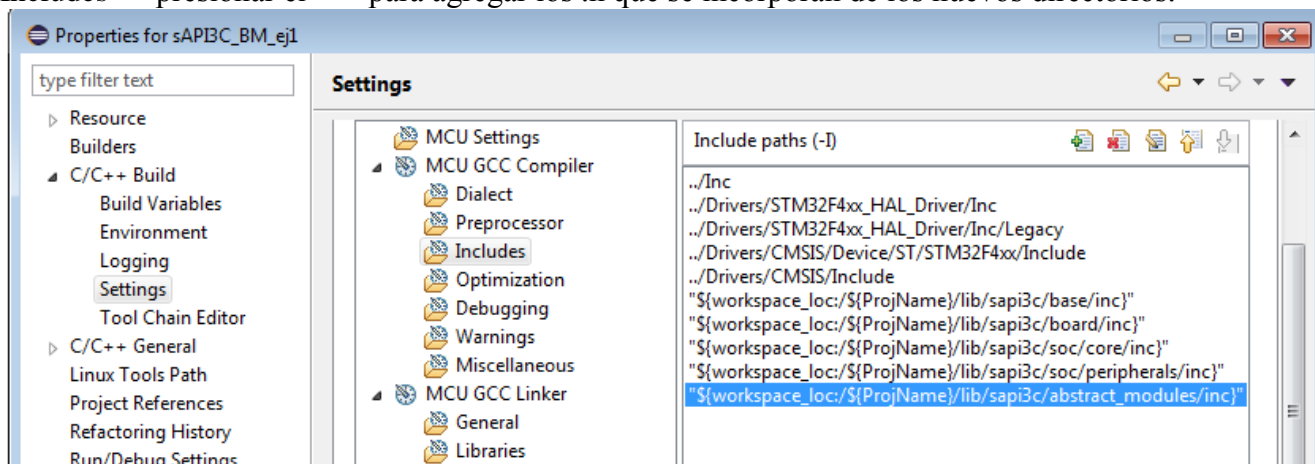
Quedando asi:



C4.4) Abrimos AC6, Ej01, Damos Refresh.



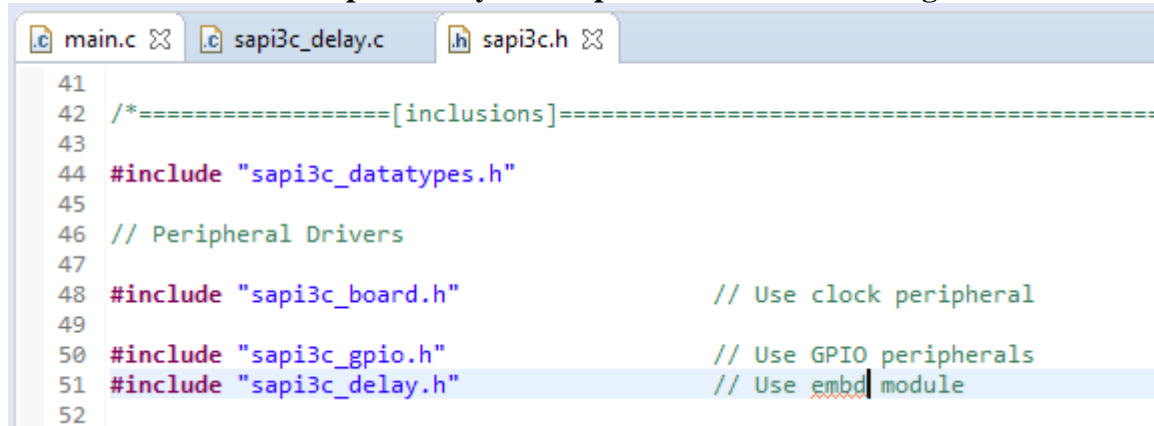
C4.4.1) Sobre el directorio Ej01, mouse derecho, properties. Aplicamos a este nuevo la regla C.1.6 “..parado en el directorio del proyecto, en properties del Proyecto, C/C++ Build-> settings->MCU GCC-> Includes --- presionar el “+” para agregar los .h que se incorporan de los nuevos directorios.”



C4.5) Ensayos finales de compilación

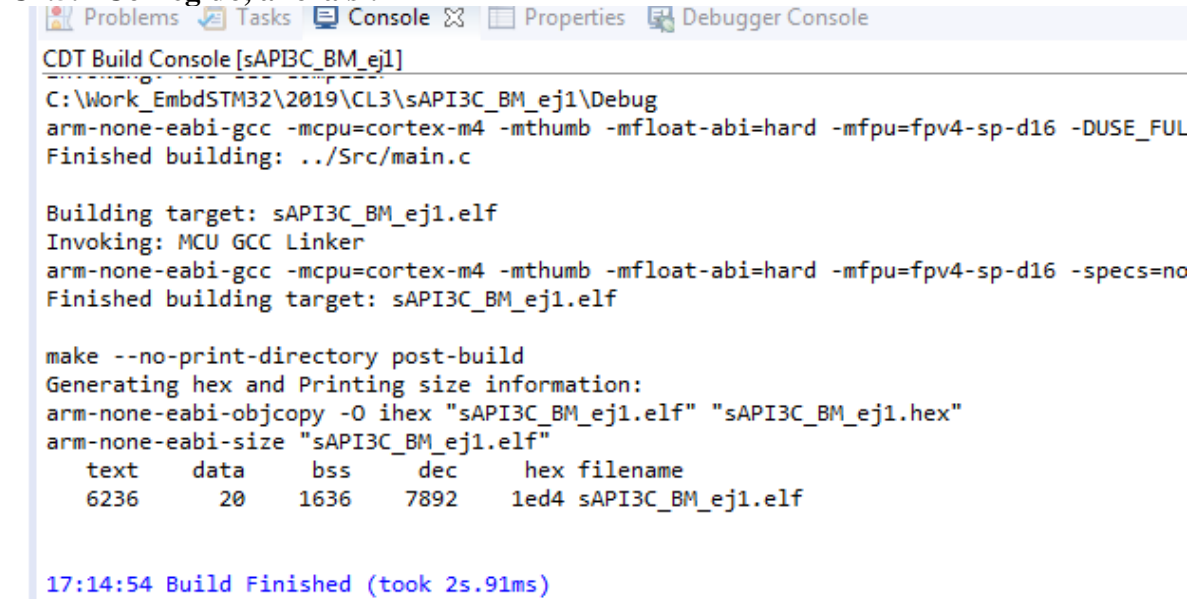


#### C4.5.1 nos faltó incluir sapi3c\_delay.h en sapi3c.h -- tira un warning..



```
41
42 /*=====[inclusions]=====
43
44 #include "sapi3c_datatypes.h"
45
46 // Peripheral Drivers
47
48 #include "sapi3c_board.h"           // Use clock peripheral
49
50 #include "sapi3c_gpio.h"           // Use GPIO peripherals
51 #include "sapi3c_delay.h"         // Use em module
52
```

#### C4.5.2 Corregido, ahora si:



```
CDT Build Console [sAPI3C_BM_ej1]
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -DUSE_FUL
Finished building: ../Src/main.c

Building target: sAPI3C_BM_ej1.elf
Invoking: MCU GCC Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -specs=no
Finished building target: sAPI3C_BM_ej1.elf

make --no-print-directory post-build
Generating hex and Printing size information:
arm-none-eabi-objcopy -O ihex "sAPI3C_BM_ej1.elf" "sAPI3C_BM_ej1.hex"
arm-none-eabi-size "sAPI3C_BM_ej1.elf"
  text    data     bss     dec     hex filename
  6236     20    1636    7892    1ed4 sAPI3C_BM_ej1.elf

17:14:54 Build Finished (took 2s.91ms)
```

#### C4.5.3 Bajado y probado—OK . ✓ -> Subido al Repo

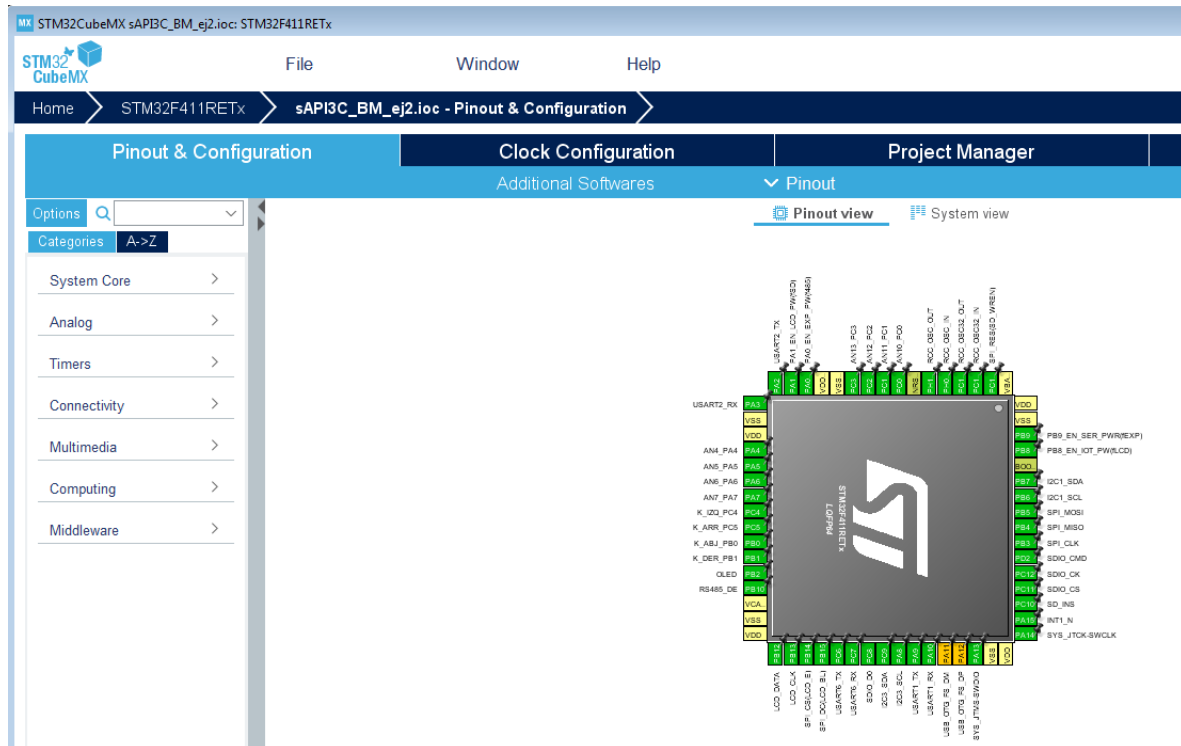
## C.5) sAPI3C\_ej2

Ahora Intentamos replicar el Proyecto, en un sAPI3C\_ej2, dentro del mismo WorkSpace. Para esto, no es posible copiar directamente.

### C.5.1) Pasos de replicación:

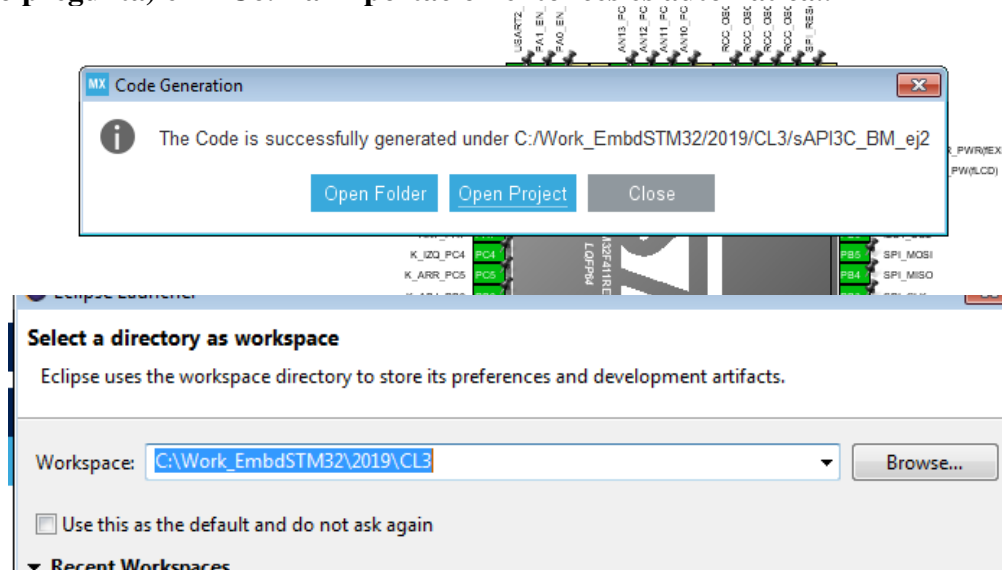
#### C.5.1a) Cerrar AC6

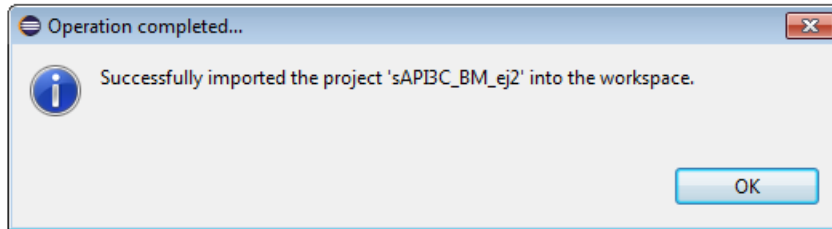
C.5.1b) Abrir el Proyecto desde CubeMX, y hacerle un Guardar Como \sAPI3C\_BM\_ej2, en el mismo Workspace: C:\Work\_EmbdSTM32\2019\CL3



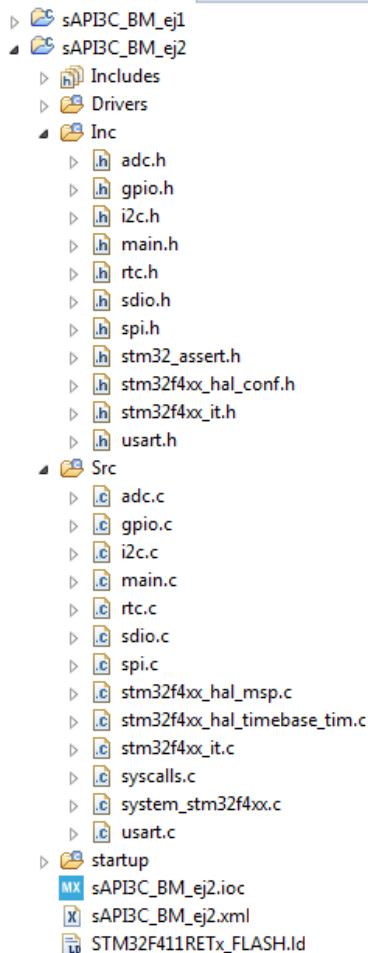
Si le damos aquí “Generate Code”, me reproduce todo lo del ejemplo 1, aunque después tendremos que hacer lo siguiente:

C.5.1c) Cuando termine el GenerateCode en CubeMX, abrir el Proyecto (dar la opción Open cuando pregunta) en AC6. La importación entonces es automática..





**Abrirlo aquí y ver que está todo, dar un refresh. ✓**



**C.5.2) Para replicar el mismo proyecto, con otro nombre, se requiere:**

**C.5.2.1) Cerrar AC6.**

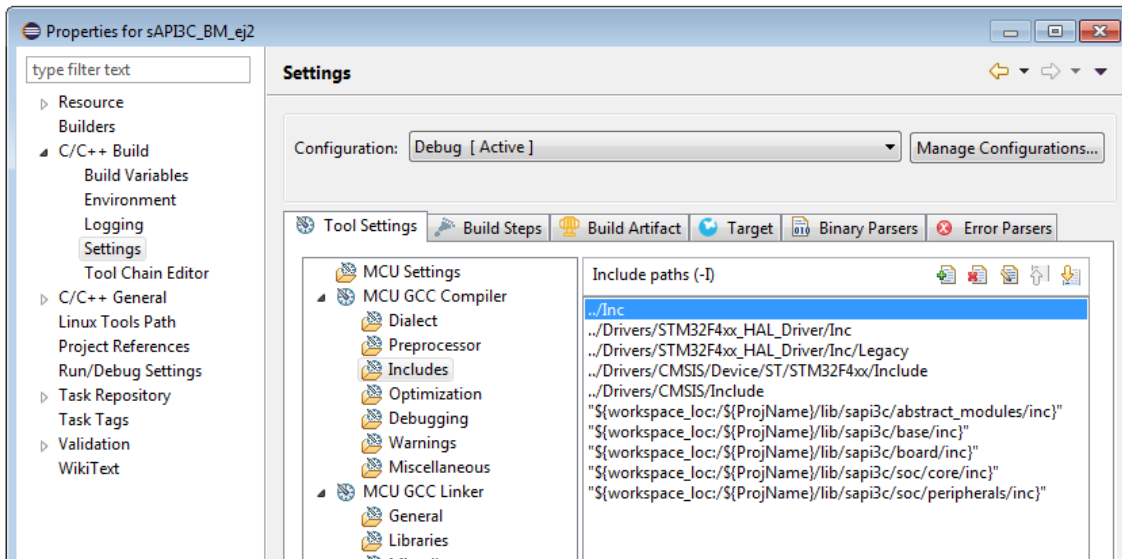
**C.5.2.2) Desde el Explorer, copiar /lib completo desde \_ej1 a \_ej2. ✓**

**C.5.2.3) Borrar todos los archivos nuevos de /Src e /Inc, generados por CubeMX. ✓**

**C.5.2.4) Copiar desde \_ej1 a \_ej2: main.c a /Src, main.h a /Inc. ✓**

**C.5.2.5) Entrar a AC6 de nuevo, dar Refresh (F5) con \_ej2 abierto, Aplicar C.1 a /lib**  
**- Se va para directorio /lib “nuevo” eliminando el “exclude from build” (right click en cada directorio, C/C++Build, destildar el “exclude resource from build”). . ✓**  
**- Finalmente, parado en el directorio del proyecto \_ej2 , en properties del Proyecto, C/C++ Build-> settings->MCU GCC-> Includes --- presionar el “+” para agregar los .h que se incorporan de los nuevos directorios.**

Termina quedando asi:



### C.5.2.6) Con esto compila correctamente:

```
Problems Tasks Console Properties Debugger Console
CDT Build Console [sAPI3C_BM_ej2]
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej2\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -DUSE_FULL_LL_DRIVER -D_
Finished building: ../Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_ll_utils.c

Building target: sAPI3C_BM_ej2.elf
Invoking: MCU GCC Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -specs=nosys.specs -specs
Finished building target: sAPI3C_BM_ej2.elf

make --no-print-directory post-build
Generating hex and Printing size information:
arm-none-eabi-objcopy -O ihex "sAPI3C_BM_ej2.elf" "sAPI3C_BM_ej2.hex"
arm-none-eabi-size "sAPI3C_BM_ej2.elf"
text data bss dec hex filename
6236 20 1636 7892 1ed4 sAPI3C_BM_ej2.elf

18:29:27 Build Finished (took 29s.809ms)
```

### C.5.2.7) Ensayamos: Funciona OK!

**D.1) Ensayo elemental con UART** Aquí tomamos como base el ejemplo que usa la UART solo para salida, con la función `printfmsg(char *msg)` de `UART_LL_5`

La renombramos a `printfmsg_cl3(char *msg)`

#### D.1.1) En -DB01 creamos la función `sapi3c_uart.h`

```
/* Date: 2019-06-10 */

#ifndef _SAPI_UART_H_
#define _SAPI_UART_H_

/*=====[inclusions]=====*/
#include "sapi3c_delay.h"
#include "sapi3c_datatypes.h"

/*=====[cplusplus]=====*/

#ifdef __cplusplus
extern "C" {
#endif

/*=====[macros]=====*/
```

```
#define uartConfig uartInit

/*=====[typedef]=====*/

/* In sapi3c_board.h */
/* typedef enum {
    UART_TER = 0, // Hardware UART6 (RS232_2 connector - Terminal)
    UART_485 = 1, // Hardware UART1 via RS_485 A, B and GND Borns
    UART_2 = 2 // Hardware UART2 (RS232_1 connector or ESP8266)
} uartMap_t;

typedef enum {
    BAUD_9600 = 0, // Lowest Baud rate 9600
    BAUD_19200, // Intermediate 19200
    BAUD_38400, // For METEO
    BAUD_115200 // Highest 115200
} uartBaudR_t; */

/*=====[external data declaration]=====*/

/*=====[external functions declaration]=====*/

//-----
// UART Initialization
void uartInit( uartMap_t uart, uartBaudR_t baudR);

// Prints Debug message out on Terminal - UART6..
// 18.4.2019 Use LL_USART functions as in STM32F4 LL_Examples

void printmsg_cl3(char *msg); . ✓
```

**D.1.2) y la copiamos a Workspace de \_ej2 .** ✓

**D.1.3) En –DB01 creamos la función sapi3c\_uart.c** ✓

```
#include "sapi3c_uart.h"
#include "string.h"

/*=====[macros]=====*/

/* on sapi3c_board.h */
/* typedef enum {
    UART_TER = 0, // Hardware UART6 (RS232_2 connector - Terminal)
    UART_485 = 1, // Hardware UART1 via RS_485 A, B and GND Borns
    UART_2 = 2 // Hardware UART2 (RS232_1 connector or ESP8266)
} uartMap_t;

typedef enum {
    BAUD_9600 = 0, // Lowest Baud rate 9600
    BAUD_19200, // Intermediate 19200
    BAUD_38400, // For METEO
    BAUD_115200 // Highest 115200
} uartBaudR_t; */

/*=====[typedef]=====*/

//-----
// UART Initialization
bool_t uartInit( uartMap_t uart, uartBaudR_t baudR ){

bool_t error = 0;
uint32_t baudRate;
switch(baudR){
```

```

        case BAUD_9600:
            baudRate = 9600;
            break;
        case BAUD_19200:
            baudRate = 19200;
            break;
        case BAUD_38400:
            baudRate = 38400;
            break;
        case BAUD_115200:
            baudRate = 115200;
            break;
        default:
            baudRate = 115200;
            error = 1;
            break;
    }
    LL_USART_InitTypeDef USART_InitStruct = {0};
    LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

/* USART1=UART_485 init function */
if( uart == UART_485 ) {
    /* Peripheral clock enable */
    LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_USART1);
    LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
    /**USART1 GPIO Configuration
    PA9  -----> USART1_TX
    PA10 -----> USART1_RX
    */
    GPIO_InitStruct.Pin = LL_GPIO_PIN_9|LL_GPIO_PIN_10;
    GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
    GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
    GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
    GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
    LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    /* USART1 interrupt Init */
    NVIC_SetPriority(USART1_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),5, 0));
    NVIC_EnableIRQ(USART1_IRQn);

    USART_InitStruct.BaudRate = baudRate;
    USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
    USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
    USART_InitStruct.Parity = LL_USART_PARITY_NONE;
    USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
    USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
    USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
    LL_USART_Init(USART1, &USART_InitStruct);
    LL_USART_ConfigAsyncMode(USART1);
    LL_USART_Enable(USART1);
}

else if (uart == UART_TER ){
    /* USART6 init function */
    /* Peripheral clock enable */
    LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_USART6);
    LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
    /**USART6 GPIO Configuration
    PC6  -----> USART6_TX
    PC7  -----> USART6_RX
    */
    GPIO_InitStruct.Pin = LL_GPIO_PIN_6|LL_GPIO_PIN_7;
    GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
    GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
    GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
    GPIO_InitStruct.Alternate = LL_GPIO_AF_8;
    LL_GPIO_Init(GPIOC, &GPIO_InitStruct);
}

```

```

/* USART6 interrupt Init */
NVIC_SetPriority(USART6_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),5, 0));
NVIC_EnableIRQ(USART6_IRQn);

USART_InitStruct.BaudRate = BaudRate;
USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
USART_InitStruct.Parity = LL_USART_PARITY_NONE;
USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
LL_USART_Init(USART6, &USART_InitStruct);
LL_USART_ConfigAsyncMode(USART6);
LL_USART_Enable(USART6);
}
else {
    /* UART2 not implem yet */
    error = 1;
}
return(error);
}

// Prints message out on Terminal UART6 - blocking
// 18.4.2019 Use LL_USART functions as in STM32F4 LL_Examples
void printmsg_cl3(char *msg)
{
    for(uint32_t i=0; i < strlen(msg); i++)
    {
        while(!LL_USART_IsActiveFlag_TXE(USART6)){
            ; // Wait forever
        }
        // while (USART_GetFlagStatus(USART6,USART_FLAG_TXE) != SET);
        LL_USART_TransmitData8(USART6,msg[i]);
    }

    while (!LL_USART_IsActiveFlag_TC(USART6)){
        ; // Wait again forever
    }
}

```

**D.1.4) y la copiamos a Workspace de \_ej2 .** 

**D.1.5) Cambiamos main.c en DB01 – Ej2, que es:**

```

/**
*****
* @author E.Pernia / adapted by R.Oliva
* @brief Ej 02 This file Tests basic operations for CL3
* rev 4.8.19
*
**/
/* USER CODE END Header */

/* Includes -----*/
#include "main.h"
#include "sapi3c.h"

//global space for debug string
char usr_msg[100]={0};
bool_t valor = 1;

/**
* @brief main() - The application entry point.
* @retval int
*/

```

```

int main(void)
{
    // CL3 board configuration
    boardInitCL3();

    // CL3 Terminal @UART 6, 115200 configuration
    uartConfig(UART_TER, BAUD_115200);

    sprintf(usr_msg, "\r\n Trabajo Final R.Oliva 2019 \r\n");
    printmsg_cl3(usr_msg);

    while (1)
    {
        valor = !gpioRead(KBD_ABJ);
        if(valor) {
            sprintf(usr_msg, "\r\n Tecla Abajo Presionada");
            printmsg_cl3(usr_msg);
        }
        HAL_Delay(500);
        gpioWrite(OLED_PB2, ON);
        HAL_Delay(500);
        gpioWrite(OLED_PB2, OFF);
    }
}

```

**D.1.6) y copiamos main.c a Workspace de \_ej2 .** ✓

**D.1.7) Abrimos AC6, \_ej2 y vemos si compila..**

**D.1.8) Error, no encuentra las definiciones nuevas de board\_cl3, Faltó copiar la sapi3c\_board.h nueva. Salimos de AC6 y copiamos desde DB0 a WS de \_ej2, y volvemos**

## Reinicio

**D.1.9) Errores detectados: Comparamos con TreeSize free los directorios, vemos que sapi3c\_board.h tiene un tamaño menor en ej\_2 que en ej\_1 - Faltó incluir en la versión del ej\_2:**

```

/* sapi3c_board.h
 * Adapted for CL3 board: R.Oliva 06-2019 rev 4.8.19
 *
 */
/* Date: 2019-06-10 */

#ifndef _SAPI_BOARD_H_
#define _SAPI_BOARD_H_

/*===== [inclusions] =====*/
#include "sapi3c_datatypes.h"

/*===== [cplusplus] =====*/

#ifdef __cplusplus
extern "C" {
#endif

/*===== [macros] =====*/

```

**Faltaron Los includes de stm32, como estaban en la versión anterior (probablemente no quedaron copiados?) – como aquí de \_ej1:**



```

30  * POSSIBILITY OF SUCH DAMAGE.
31  *
32  * sapi3c_board.h
33  * Adapted for CL3 board: R.Oliva 06-2019 rev 4.8.19
34  *
35  */
36  /* Date: 2019-06-10 */
37
38  #ifndef _SAPI_BOARD_H_
39  #define _SAPI_BOARD_H_
40
41  /*===== [inclusions] =====*/
42
43  #include "sapi3c_datatypes.h"
44  /* Includes -----*/
45  #include "stm32f4xx_hal.h"
46  #include "stm32f4xx_hal.h"
47  #include "stm32f4xx_ll_usart.h"
48  #include "stm32f4xx_ll_rcc.h"
49  #include "stm32f4xx.h"
50  #include "stm32f4xx_ll_system.h"
51  #include "stm32f4xx_ll_gpio.h"
52  #include "stm32f4xx_ll_exti.h"
53  #include "stm32f4xx_ll_bus.h"
54  #include "stm32f4xx_ll_cortex.h"
55  #include "stm32f4xx_ll_utils.h"
56  #include "stm32f4xx_ll_pwr.h"
57  #include "stm32f4xx_ll_dma.h"
58
59  /*===== [cplusplus] =====*/
60
61

```

**D.1.10) Acción – Desde ej\_1 en WS (Explorer), editamos con NP++ y copiamos todos los faltantes a sapi3c\_board.h del ej\_2 .** ✓

**D.1.11) Cerramos los WS, TSFree y Ensayamos -> ver que está el Archivo actualizado:**

**➔ Damos Clean, y nuevamente compilación desde AC6..**

**D.1.12 – Error 1 - En uart\_init() lo habíamos declarado como void, y después lo cambiamos en el .c a bool\_t para devolver un error si falla.. Además le erramos en una letra a baudRate, variable auxiliar que se puede configurar como 9600, 19200,38400 o 115200.**

```
Building file: ../lib/sapi3c/soc/peripherals/src/sapi3c_uart.c
Invoking: MCU GCC Compiler
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej2\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULL_LL_DRIVER
../lib/sapi3c/soc/peripherals/src/sapi3c_uart.c:66:8: error: conflicting types for 'uartInit'
bool_t uartInit( uartMap_t uart, uartBaudR_t baudR ){
      ^~~~~~
In file included from ../lib/sapi3c/soc/peripherals/src/sapi3c_uart.c:41:0:
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej2\lib/sapi3c/soc/peripherals/inc/sapi3c_uart.h:83:6: note: |
void uartInit( uartMap_t uart, uartBaudR_t baudR);
      ^~~~~~
../lib/sapi3c/soc/peripherals/src/sapi3c_uart.c: In function 'uartInit':
../lib/sapi3c/soc/peripherals/src/sapi3c_uart.c:145:31: error: 'BaudRate' undeclared (first use in
USART_InitStruct.BaudRate = BaudRate;
                        ^~~~~~
```

Corregimos el sapi3c\_uart.h .



**D.1.13 – Error 2 - Ahora no estamos reconociendo el define de uartConfig(), ni stdio.h para sprintf()**

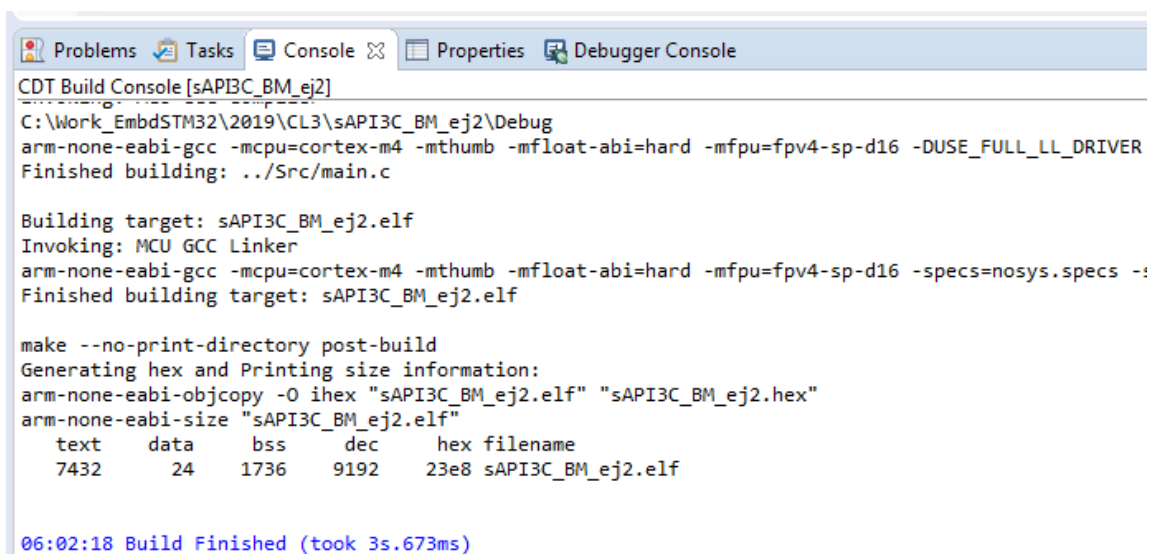
```
Building file: ../Src/main.c
Invoking: MCU GCC Compiler
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej2\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULL_LL_DRIVER '-D__weak=__at
../Src/main.c: In function 'main':
../Src/main.c:46:3: warning: implicit declaration of function 'uartConfig'; did you mean 'boardConfigCL3'? [-Wim
uartConfig(UART_TER, BAUD_115200);
      ^~~~~~
boardConfigCL3
../Src/main.c:48:3: warning: implicit declaration of function 'sprintf' [-Wimplicit-function-declaration]
sprintf(usr_msg, "\r\n Trabajo Final R.Oliva 2019 \r\n");
      ^~~~~~
../Src/main.c:48:3: warning: incompatible implicit declaration of built-in function 'sprintf'
../Src/main.c:48:3: note: include '<stdio.h>' or provide a declaration of 'sprintf'
../Src/main.c:49:3: warning: implicit declaration of function 'printmsg_cl3' [-Wimplicit-function-declaration]
printmsg_cl3(usr_msg);
      ^~~~~~
Finished building: ../Src/main.c
```

**Error2.1 – nos faltó en sapi3c.h “subir” fuera del #define el sapi3c\_uart.h:**

```
h sapi3c_board.h c sapi3c_uart.c h sapi3c_uart.h h sapi3c.h x
1+ /* Copyright 2015-2017, Eric Pernia.
36
37 /* Date: 2019-06-10 */
38
39 #ifndef _SAPI_H_
40 #define _SAPI_H_
41
42 /*===== [inclusions] =====
43
44 #include "sapi3c_datatypes.h"
45
46 // Peripheral Drivers
47
48 #include "sapi3c_board.h" // Use clock peripheral
49
50 #include "sapi3c_gpio.h" // Use GPIO peripherals
51 #include "sapi3c_delay.h" // Use embd module
52
53 #ifdef SECOND_TEST
54 #include "sapi3c_uart.h" // Use UART peripherals
55 #include "sapi3c_adc.h" // Use ADC1 peripheral
```

**Error2.2 en main.h incluimos el stdio.h**

### D.1.13 – Recompilamos, y genera el archivo OK. ✓



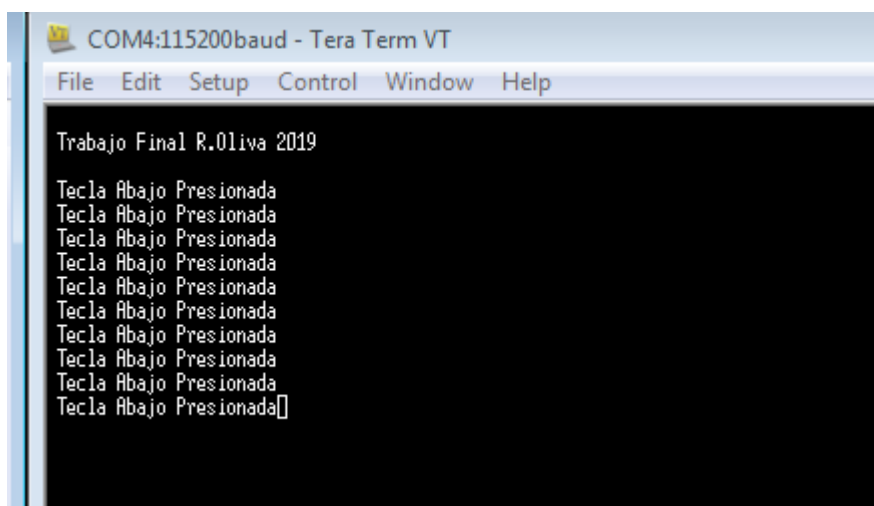
```
CDT Build Console [sAPI3C_BM_ej2]
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej2\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -DUSE_FULL_LL_DRIVER
Finished building: ../Src/main.c

Building target: sAPI3C_BM_ej2.elf
Invoking: MCU GCC Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -specs=nosys.specs -:
Finished building target: sAPI3C_BM_ej2.elf

make --no-print-directory post-build
Generating hex and Printing size information:
arm-none-eabi-objcopy -O ihex "sAPI3C_BM_ej2.elf" "sAPI3C_BM_ej2.hex"
arm-none-eabi-size "sAPI3C_BM_ej2.elf"
   text    data     bss     dec     hex filename
   7432     24    1736    9192    23e8 sAPI3C_BM_ej2.elf

06:02:18 Build Finished (took 3s.673ms)
```

Recordamos que en D.1.5) definimos que el nuevo programa imprima un mensaje inicial, y luego en el lazo del LED otro si detecta presionada la tecla KBD\_ABJ = F3



```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

Trabajo Final R.Oliva 2019
Tecla Abajo Presionada
Tecla Abajo Presionada
Tecla Abajo Presionada
Tecla Abajo Presionada
Tecla Abajo Presionada
Tecla Abajo Presionada
Tecla Abajo Presionada
Tecla Abajo Presionada
Tecla Abajo Presionada
Tecla Abajo Presionada
```

Funciona OK! 05.0819 – 6:09. – Subimos Ej02 al Repo, y enviamos a Eric

**D.2) Ensayo elemental con UART** Aquí tomamos como base el ejemplo que usa la UART con interrupciones, para preparar el ejemplo de RTOS funcionando

C:\CESE2019\EDU\_CIAA\_Repo\cese-edu-ciaa-template\examples\c\sapi\uart\rx\_interrupt\src\rx\_interrupt.c

```
#include "sapi.h"

void onRx( void *noUsado )
{
    char c = uartRxRead( UART_USB );
    printf( "Recibimos <<%c>> por UART\r\n", c );
}

int main(void)
{
```

```

/* Inicializar la placa */
boardConfig();

/* Inicializar la UART_USB junto con las interrupciones de Tx y Rx */
uartConfig(UART_USB, 115200);
// Seteo un callback al evento de recepcion y habilito su interrupcion
uartCallbackSet(UART_USB, UART_RECEIVE, onRx, NULL);
// Habilito todas las interrupciones de UART_USB
uartInterrupt(UART_USB, true);

while(TRUE) {
    // Una tarea muy bloqueante para demostrar que la interrupcion funciona
    gpioToggle(LED_B);
    delay(1000);
}
return 0;
}

```

Para esto, deberíamos tener las **versiones 3c de las funciones resaltadas** para inicializar las UARTs usadas en el ejemplo UART\_LL\_6, o sea según sapi3c\_board.h, e implementadas en sapi3c\_uart.h/.c :

```

/* Defined for sapi3c_uart.h */

typedef enum {
    UART_TER = 0, // Hardware UART6 (RS232_2 connector - Terminal)
    UART_485 = 1, // Hardware UART1 via RS_485 A, B and GND Borns
    UART_2 = 2 // Hardware UART2 (RS232_1 connector or ESP8266)
} uartMap_t;

```

**E) Para el periférico externo Display** Aquí tomamos como base el ejemplo /sAPI que usa el display de 7 segmentos. Como sólo tenemos un LED, el display (ILI9341 que funciona bien) debería tener una inicialización parecida a la del ejemplo de Eric:

```

/* FUNCION PRINCIPAL, PUNTO DE ENTRADA AL PROGRAMA LUEGO DE RESET. */
int main(void){

```

```

/* ----- INICIALIZACIONES ----- */

```

```

/* Inicializar la placa */
boardConfig();

```

```

/* Configuracion de pines para el display 7 segmentos */
/*

```

Segmento encendido	Valor BIN	Valor HEX	GPIO resultado
Enciende el segmento 'a'	0b00000001	0x20	GPIO5
Enciende el segmento 'b'	0b00000010	0x80	GPIO7
Enciende el segmento 'c'	0b00000100	0x40	GPIO6
Enciende el segmento 'd'	0b00001000	0x02	GPIO1
Enciende el segmento 'e'	0b00010000	0x04	GPIO2
Enciende el segmento 'f'	0b00100000	0x10	GPIO4
Enciende el segmento 'g'	0b01000000	0x08	GPIO3
Enciende el segmento 'h'	0b10000000	0x80	GPIO8

```

      a
    -----
  f /      / b
    /  g  /
    -----
e /      / c
 /  d  /
-----  0 h = dp (decimal point).

```

```

*/
uint8_t display7Segment[8] = {

```

```

    GPIO5, // Segment 'a'
    GPIO7, // Segment 'b'
    GPIO6, // Segment 'c'
    GPIO1, // Segment 'd'
    GPIO2, // Segment 'e'
    GPIO4, // Segment 'f'
    GPIO3, // Segment 'g'
    GPIO8 // Segment 'h' or 'dp'
};

```

**display7SegmentPinConfig( display7Segment ); (E.A.1)**

```

/* Configuración de pines para el Teclado Matricial*/

```

```

// Teclado
keypad_t keypad;

```

```

// Filas --> Salidas
uint8_t keypadRowPins1[4] = {
    RS232_TXD, // Row 0
    CAN_RD,    // Row 1
    CAN_TD,    // Row 2
    T_COL1     // Row 3
};

```

```

// Columnas --> Entradas con pull-up (MOD0 = GPIO_INPUT_PULLUP)
uint8_t keypadColPins1[4] = {
    T_FIL0,    // Column 0
    T_FIL3,    // Column 1
    T_FIL2,    // Column 2
    T_COL0     // Column 3
};

```

**keypadConfig( &keypad, keypadRowPins1, 4, keypadColPins1, 4 ); (E.A.2)**

```

// Vector de conversión entre índice de tecla presionada y el índice del
// display 7 segmentos

```

```

uint16_t keypadToDisplayKeys[16] = {
    1,    2,    3, 0x0a,
    4,    5,    6, 0x0b,
    7,    8,    9, 0x0c,
    0x0e, 0, 0x0f, 0x0d
};

```

```

// Variable para guardar la tecla leída
uint16_t tecla = 0;

```

```

/* ----- REPETIR POR SIEMPRE ----- */
while(1) {

```

```

    if( keypadRead( &keypad, &tecla ) ){
        display7SegmentWrite( display7Segment,
                               keypadToDisplayKeys[ (uint8_t)tecla ] ); (E.A.3)
    } else{

```

```

        display7SegmentWrite( display7Segment, DISPLAY_7_SEGMENT_OFF );
    }
}

```

```

/* NO DEBE LLEGAR NUNCA AQUÍ, debido a que a este programa no es llamado
por ningún S.O. */
return 0 ;
}

```

Si se observa deberíamos tener en primer lugar funciones al estilo de **(E.A.1,2)** que inicializan los periféricos externos específicos, y además para el display una función de tipo **(E.A.3)**, que permita escribir en el mismo.

**F) Para el uso de FreeRTOS, tomamos:** El ejemplo de implementación en la sAPI podría ser el de:  
 C:\CESE2019\EDU\_CIAA\_Repo\cese-edu-ciaa-  
 template\examples\c\sapi\rtos\_freertos\dynamic\_mem\freeRTOS\_01\_blinky\src\freeRTOS\_blinky.c

```
// sAPI header
#include "sapi.h"
DEBUG_PRINT_ENABLE;

// Prototipo de funcion de la tarea
void myTask( void* taskParmPtr );

int main(void)
{
    boardConfig();

    // UART for debug messages
    debugPrintConfigUart( UART_USB, 115200 );
    debugPrintlnString( "Blinky con freeRTOS y sAPI." );

    // Led para dar se al de vida
    gpioWrite( LED3, ON );

    // Crear tarea en freeRTOS
    xTaskCreate(
        myTask,                // Funcion de la tarea a ejecutar
        (const char *)"myTask", // Nombre de la tarea como String amigable para el usuario
        configMINIMAL_STACK_SIZE*2, // Cantidad de stack de la tarea
        0,                    // Parametros de tarea
        tskIDLE_PRIORITY+1,    // Prioridad de la tarea
        0,                    // Puntero a la tarea creada en el sistema
    );

    // Iniciar scheduler
    vTaskStartScheduler();

    // ----- REPETIR POR SIEMPRE -----
    while( TRUE ) {
        // Si cae en este while 1 significa que no pudo iniciar el scheduler
    }
    return 0;
}

// Implementacion de funcion de la tarea
void myTask( void* taskParmPtr )
{
    // ----- CONFIGURACIONES -----
    printf( "Blinky con freeRTOS y sAPI.\r\n" );

    gpioWrite( LED1, ON );
    // Env a la tarea al estado bloqueado durante 1 s (delay)
    vTaskDelay( 1000 / portTICK_RATE_MS );
    gpioWrite( LED1, OFF );

    // Tarea periodica cada 500 ms
    portTickType xPeriodicity = 500 / portTICK_RATE_MS;
    portTickType xLastWakeTime = xTaskGetTickCount();

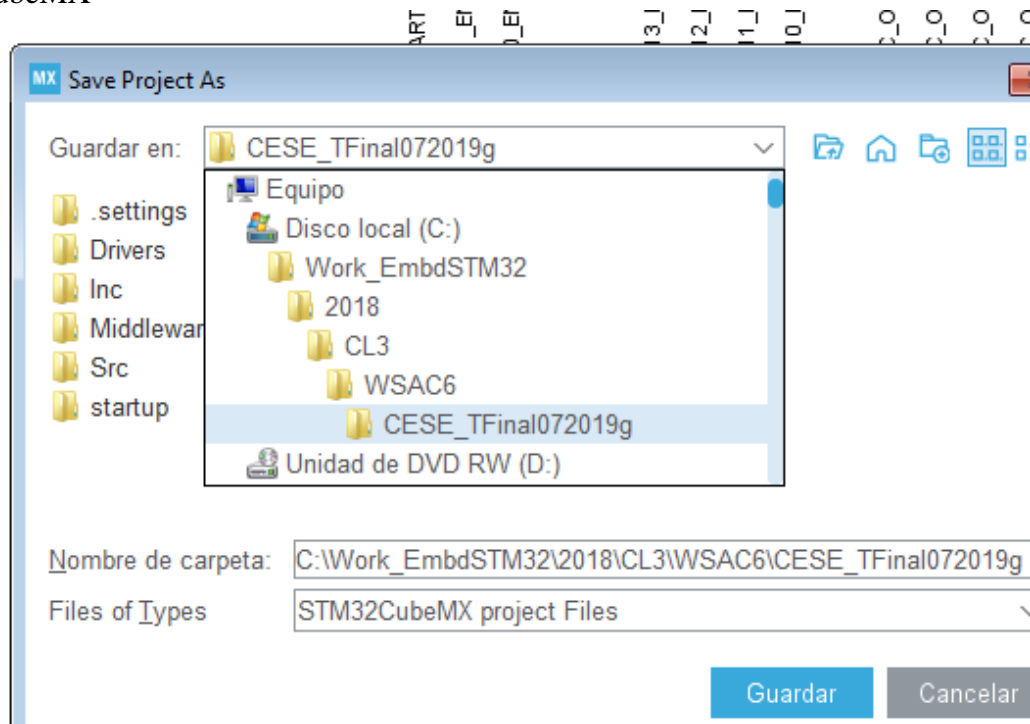
    // ----- REPETIR POR SIEMPRE -----
    while(TRUE) {
        gpioToggle( LEDB );
        printf( "Blink!\r\n" );
        // Env a la tarea al estado bloqueado durante xPeriodicity (delay periodico)
        vTaskDelayUntil( &xLastWakeTime, xPeriodicity );
    }
}
```

}

## ANEXO

### 31.07.2019 Creación de un proyecto con sAPI3C BareMetal (ejemplo sencillo de LED y Consola UART6)

- A.1)** Tomamos de versión g 0719 (placa configurada completa)  
Desde el CubeMX



(que es la versión 5.1)

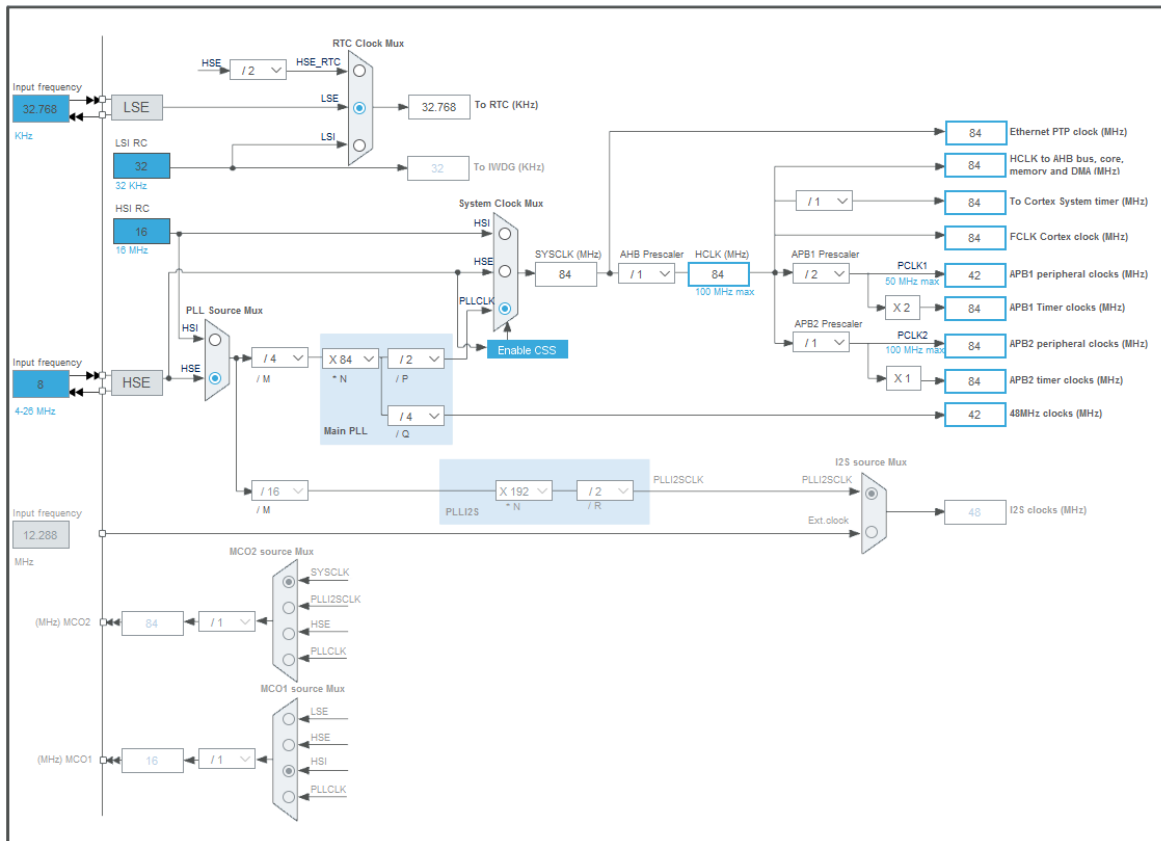






10





STM32CubeMX sAPI3C\_BM\_aj1loc - Project Manager

File Window Help

Home > STM32F411REt > sAPI3C\_BM\_aj1loc - Project Manager

GENERATE CODE

Pinout & Configuration Clock Configuration Project Manager Tools

Project

Code Generator

Advanced Settings

Generated Function Calls

Rank	Function Name	IP Instance Name	Not Generate Function Call	Visibility (Static)
1	MX_GPIO_Init	GPIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	SystemClock_Config	RCC	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	MX_DC1_Init	DC1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4	MX_USART6_UART_Init	USART6	<input type="checkbox"/>	<input checked="" type="checkbox"/>
5	MX_USART1_UART_Init	USART1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6	MX_ADC1_Init	ADC1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
7	MX_SPI1_Init	SPI1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
8	MX_USART2_UART_Init	USART2	<input type="checkbox"/>	<input checked="" type="checkbox"/>
9	MX_SDIO_SD_Init	SDIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
10	MX_DC3_Init	DC3	<input type="checkbox"/>	<input checked="" type="checkbox"/>
11	MX_RTC_Init	RTC	<input type="checkbox"/>	<input checked="" type="checkbox"/>

MX STM32CubeMX sAPI3C\_BM\_ej1.ioc\*: STM32F411RETx

STM32CubeMX File Window Help

Home > STM32F411RETx > sAPI3C\_BM\_ej1.ioc - Project Manager >

	Pinout & Configuration	Clock Configuration
Project	<p>STM32Cube Firmware Library Package</p> <p><input type="radio"/> Copy all used libraries into the project folder</p> <p><input checked="" type="radio"/> Copy only the necessary library files</p> <p><input type="radio"/> Add necessary library files as reference in the toolchain project configuration file</p>	
Code Generator	<p>Generated files</p> <p><input checked="" type="checkbox"/> Generate peripheral initialization as a pair of '.c/.h' files per peripheral</p> <p><input type="checkbox"/> Backup previously generated files when re-generating</p> <p><input checked="" type="checkbox"/> Keep User Code when re-generating</p> <p><input checked="" type="checkbox"/> Delete previously generated files when not re-generated</p>	
Advanced Settings	<p>HAL Settings</p> <p><input type="checkbox"/> Set all free pins as analog (to optimize the power consumption)</p> <p><input type="checkbox"/> Enable Full Assert</p>	
	<p>Template Settings</p> <p>Select a template to generate customized code</p> <p><a href="#">Settings...</a></p>	

## Opción Toolchain es AC6 (SW4STM32) y Firmware v1\_24\_0

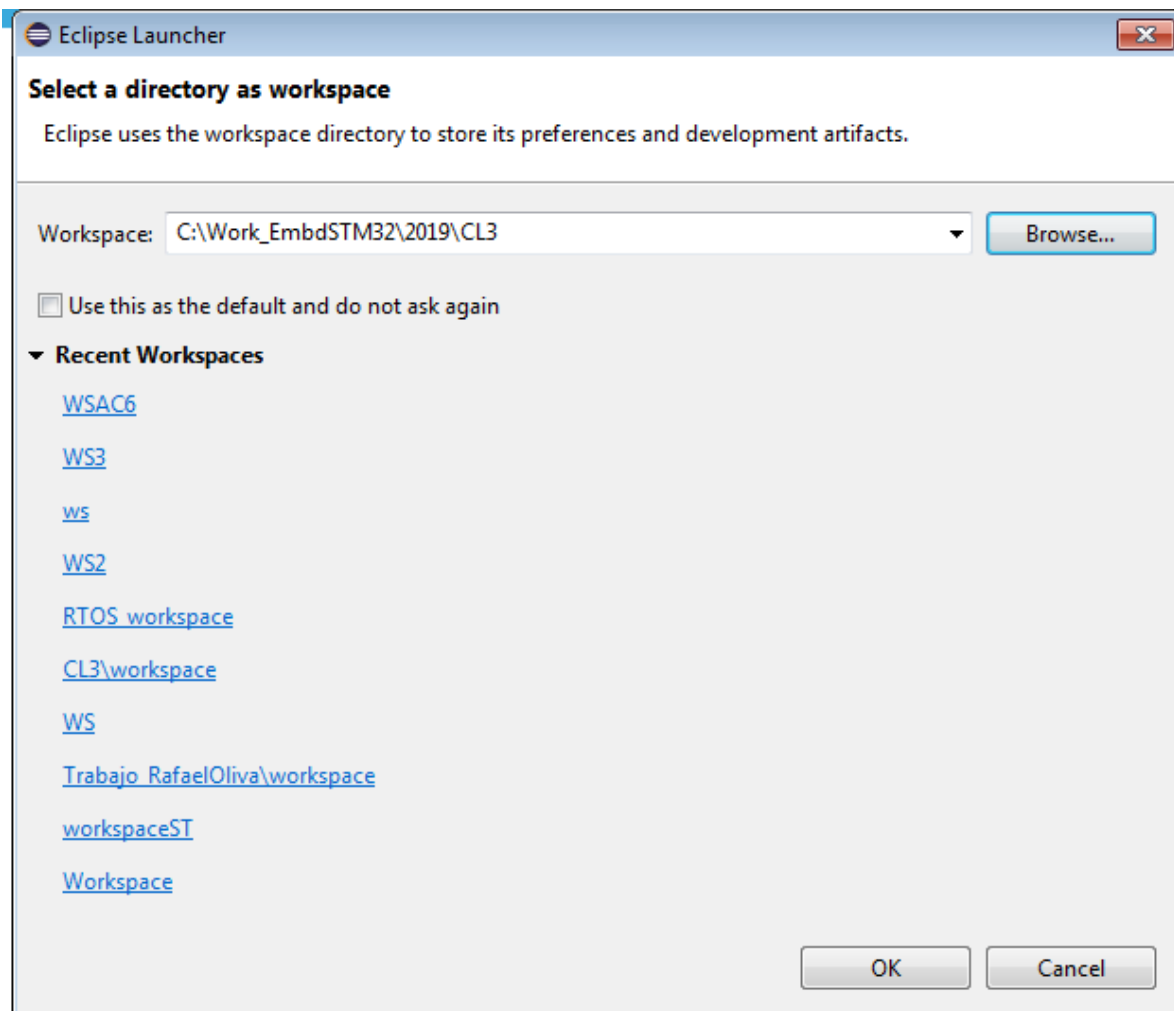
MX STM32CubeMX sAPBC\_BM\_ej1.ioc\*: STM32F411RETx

STM32CubeMX File Window Help

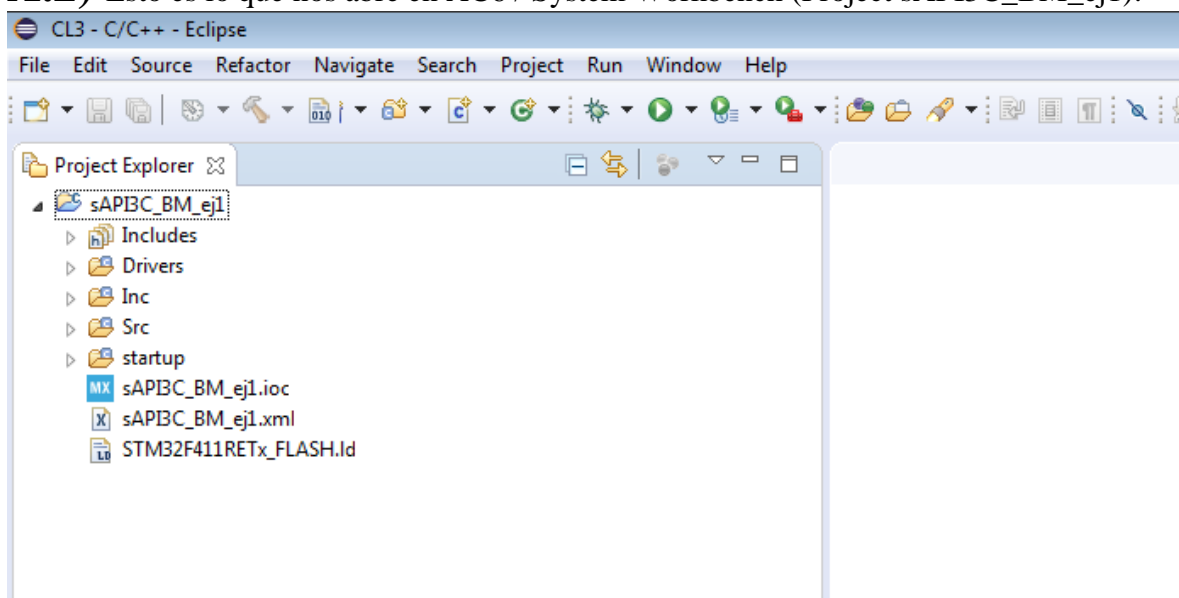
Home > STM32F411RETx > sAPI3C\_BM\_ej1.ioc - Project Manager >

	Pinout & Configuration	Clock Configuration
Project	<p>Project Settings</p> <p>Project Name sAPI3C_BM_ej1</p> <p>Project Location C:\Work_EmbdSTM32\2019\CL3</p>	
Code Generator	<p>Application Structure Basic <input type="checkbox"/> Do not generate the main()</p> <p>Toolchain Folder Location C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\</p> <p>Toolchain / IDE SW4STM32 <input checked="" type="checkbox"/> Generate Under Root</p>	
Advanced Settings	<p>Linker Settings</p> <p>Minimum Heap Size <input type="text" value="0x200"/></p> <p>Minimum Stack Size <input type="text" value="0x400"/></p>	
	<p>Mcu and Firmware Package</p> <p>Mcu Reference STM32F411RETx</p> <p>Firmware Package Name and Version STM32Cube FW_F4 V1.24.0 <input type="checkbox"/> Use latest available version</p> <p><input checked="" type="checkbox"/> Use Default Firmware Location C:/Users/Rafael/STM32Cube/Repository/STM32Cube_FW_F4_V1.24.0 <a href="#">Browse</a></p>	

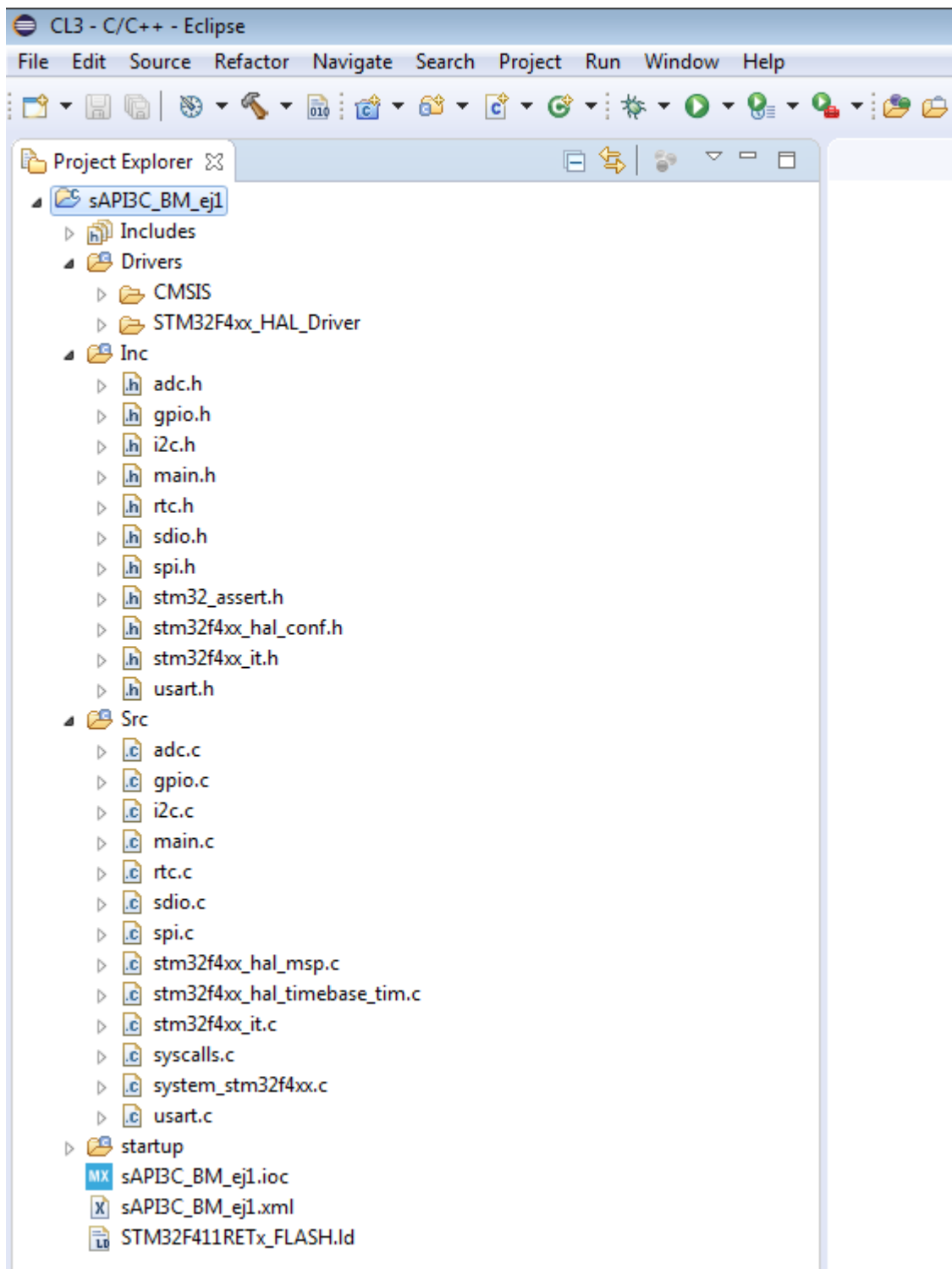




**A.2)** Esto es lo que nos abre en AC6 / System Workbench (Project sAPI3C\_BM\_ej1):

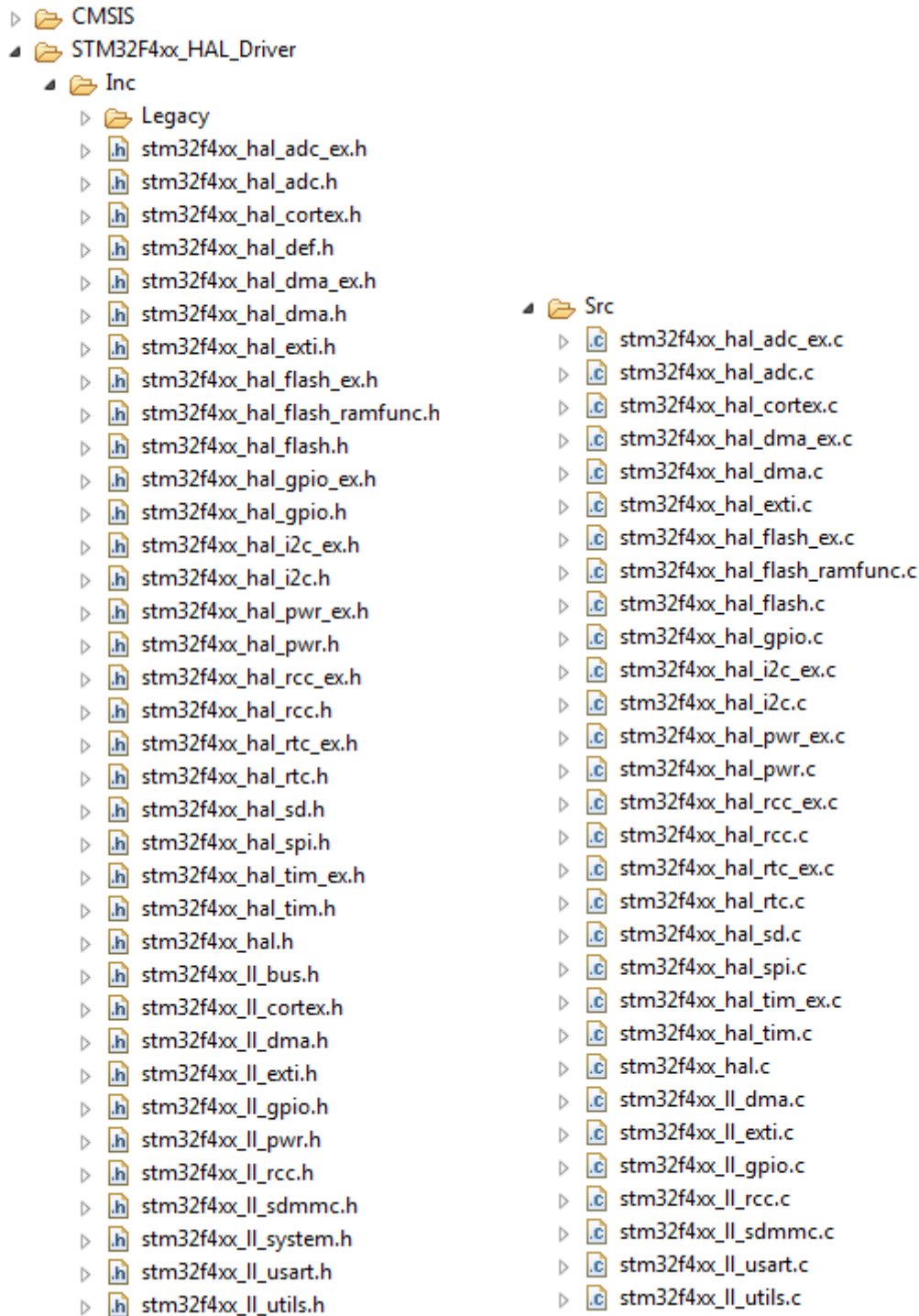


Con la siguiente estructura:



A.2.1) Y dentro de los Drivers, STM32F4xx\_HAL\_Driver queda así:

/INC y /SRC



A.2.2) El main.c contiene la función main() como sigue:

```

72 int main(void)
73 {
74     /* USER CODE BEGIN 1 */
75     /* USER CODE END 1 */
76
77     /* MCU Configuration-----*/
78     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
79     HAL_Init();
80
81     /* USER CODE BEGIN Init */
82     /* USER CODE END Init */
83
84     /* Configure the system clock */
85     SystemClock_Config();
86
87     /* USER CODE BEGIN SysInit */
88     /* USER CODE END SysInit */
89
90     /* Initialize all configured peripherals */
91     MX_GPIO_Init();
92     MX_I2C1_Init();
93     MX_USART6_UART_Init();
94     MX_USART1_UART_Init();
95     MX_ADC1_Init();
96     MX_SPI1_Init();
97     MX_USART2_UART_Init();
98     MX_SDIO_SD_Init();
99     MX_I2C3_Init();
100    MX_RTC_Init();
101    /* USER CODE BEGIN 2 */
102    /* USER CODE END 2 */
103
104    /* Infinite loop */
105    /* USER CODE BEGIN WHILE */
106    while (1)
107    {
108        /* USER CODE END WHILE */
109        /* USER CODE BEGIN 3 */
110    }
111    /* USER CODE END 3 */
112 }
113

```

Si sobre el proyecto Nuevo así creado, damos Build con el martillito, nos da:



```

88  /* USER CODE END SysInit */
89
90  /* Initialize all components */
91  MX_GPIO_Init();
92  MX_I2C1_Init();
93  MX_USART6_UART_Init();
94  MX_USART1_UART_Init();
95  MX_ADC1_Init();
96  MX_SPI1_Init();
97  MX_USART2_UART_Init();
98  MX_SDIO_SD_Init();
99  MX_I2C3_Init();
100 MX_RTC_Init();
101 /* USER CODE BEGIN 2 */
102 /* USER CODE END 2 */
103
104 /* Infinite loop */
105 /* USER CODE BEGIN WHILE */

```

**Build Project**

Building project...

Invoking Command: make all

☐ Always run in background

CDT Build Console [sAPI3C\_BM\_ej1]

```

arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -DUSE_FULL_LL_DRIVER '-D__weak=__
Finished building: ../Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_dma.c

Building file: ../Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_dma_ex.c
Invoking: MCU GCC Compiler
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -DUSE_FULL_LL_DRIVER '-D__weak=__
Finished building: ../Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_dma_ex.c

Building file: ../Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_exti.c

```

Y compila correctamente:

Problems Tasks Console Properties

CDT Build Console [sAPI3C\_BM\_ej1]

```

C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -DUSE_FULL_LL_DRIVER
Finished building: ../Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_ll_utils.c

Building target: sAPI3C_BM_ej1.elf
Invoking: MCU GCC Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpv4-sp-d16 -specs=nosys.spec
Finished building target: sAPI3C_BM_ej1.elf

make --no-print-directory post-build
Generating hex and Printing size information:
arm-none-eabi-objcopy -O ihex "sAPI3C_BM_ej1.elf" "sAPI3C_BM_ej1.hex"
arm-none-eabi-size "sAPI3C_BM_ej1.elf"
  text    data    bss     dec     hex filename
 12540     20    2124   14684   395c sAPI3C_BM_ej1.elf

08:45:06 Build Finished (took 34s.108ms)

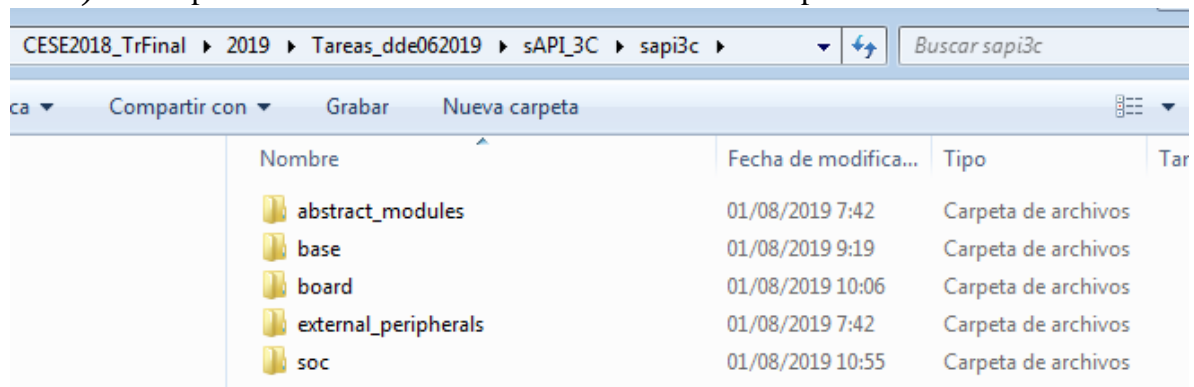
```

Ya no tienen Middlewares (FATFS ni FreeRTOS) y las rutinas de inicialización que muestra main.c están dispersadas en los pares gpio.c / .h, spi.c / .h etc..

Cerramos AC6 y CubeMX. Hacemos backup en el Seagate de Work\_EmbdST32/2019

Volvemos a abrir AC6 e intentamos migrar a una estructura de /lib/sapi3c, de forma que en el /src y /inc sólo queden los archivos main.c y main.h, para el caso del ejemplo sencillo de sapi3cBM

### A.3) Arbol preliminar: Construimos el árbol de directorios primero en:



Esto, dentro de nuestro sAPI3C\_BM\_ej1, estará agregado como /lib/sapi3c

En todos los casos, tomamos como base el archivo sapixxx de Eric, renombrando con 3c agregado, y agregando al final del header el nombre del archivo y la fecha, por ejemplo para sapi\_board.h:

```
*
* sapi3c_board.h
* Adapted for CL3 board: R.Oliva 06-2019
*
*/
```

```
/* Date: 2019-06-10 */ (B)
```

En el caso de este archivo en particular, definimos los pines según lo que teníamos originalmente en main.h, por lo cual después de (B) queda:

```
#ifndef _SAPI_BOARD_H_
#define _SAPI_BOARD_H_

/*=====[inclusions]=====*/

#include "sapi3c_datatypes.h"

/*=====[cplusplus]=====*/

#ifdef __cplusplus
extern "C" {
#endif

/*=====[macros]=====*/

#define boardConfigCL3 boardInitCL3

/* Private defines -----*/
#define SPI_RES_Pin LL_GPIO_PIN_13
#define SPI_RES_GPIO_Port GPIOC
#define AN10_PC0_Pin LL_GPIO_PIN_0
#define AN10_PC0_GPIO_Port GPIOC
#define AN11_PC1_Pin LL_GPIO_PIN_1
#define AN11_PC1_GPIO_Port GPIOC
#define AN12_PC2_Pin LL_GPIO_PIN_2
#define AN12_PC2_GPIO_Port GPIOC
#define AN13_PC3_Pin LL_GPIO_PIN_3
#define AN13_PC3_GPIO_Port GPIOC
#define PA0_EN_EXP_PW_Pin LL_GPIO_PIN_0
```

```

#define PA0_EN_EXP_PW_GPIO_Port GPIOA
#define PA1_EN_LCD_PW_Pin LL_GPIO_PIN_1
#define PA1_EN_LCD_PW_GPIO_Port GPIOA
#define AN4_PA4_Pin LL_GPIO_PIN_4
#define AN4_PA4_GPIO_Port GPIOA
#define AN5_PA5_Pin LL_GPIO_PIN_5
#define AN5_PA5_GPIO_Port GPIOA
#define AN6_PA6_Pin LL_GPIO_PIN_6
#define AN6_PA6_GPIO_Port GPIOA
#define AN7_PA7_Pin LL_GPIO_PIN_7
#define AN7_PA7_GPIO_Port GPIOA
#define K_IZQ_PC4_Pin LL_GPIO_PIN_4
#define K_IZQ_PC4_GPIO_Port GPIOC
#define K_ARR_PC5_Pin LL_GPIO_PIN_5
#define K_ARR_PC5_GPIO_Port GPIOC
#define K_ABJ_PB0_Pin LL_GPIO_PIN_0
#define K_ABJ_PB0_GPIO_Port GPIOB
#define K_DER_PB1_Pin LL_GPIO_PIN_1
#define K_DER_PB1_GPIO_Port GPIOB
#define OLED_Pin LL_GPIO_PIN_2
#define OLED_GPIO_Port GPIOB
#define RS485_DE_Pin LL_GPIO_PIN_10
#define RS485_DE_GPIO_Port GPIOB
#define LCD_DATA_Pin LL_GPIO_PIN_12
#define LCD_DATA_GPIO_Port GPIOB
#define LCD_CLK_Pin LL_GPIO_PIN_13
#define LCD_CLK_GPIO_Port GPIOB
#define SPI_CS_Pin LL_GPIO_PIN_14
#define SPI_CS_GPIO_Port GPIOB
#define SPI_DC_Pin LL_GPIO_PIN_15
#define SPI_DC_GPIO_Port GPIOB
#define INT1_N_Pin LL_GPIO_PIN_15
#define INT1_N_GPIO_Port GPIOA
#define SD_INS_Pin LL_GPIO_PIN_10
#define SD_INS_GPIO_Port GPIOC
#define SDIO_CD_Pin LL_GPIO_PIN_11 // Cambiamos CS a CD.. PC.11
#define SDIO_CD_GPIO_Port GPIOC
#define SPI_CLK_Pin LL_GPIO_PIN_3
#define SPI_CLK_GPIO_Port GPIOB
#define SPI_MISO_Pin LL_GPIO_PIN_4
#define SPI_MISO_GPIO_Port GPIOB
#define SPI_MOSI_Pin LL_GPIO_PIN_5
#define SPI_MOSI_GPIO_Port GPIOB
#define PB8_EN_IOT_PW_Pin LL_GPIO_PIN_8
#define PB8_EN_IOT_PW_GPIO_Port GPIOB
#define PB9_EN_SER_PWR_Pin LL_GPIO_PIN_9
#define PB9_EN_SER_PWR_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/*=====[typedef]=====*/
/*=====[external data declaration]=====*/
/*=====[external functions declaration]=====*/

void boardInit3c(void);

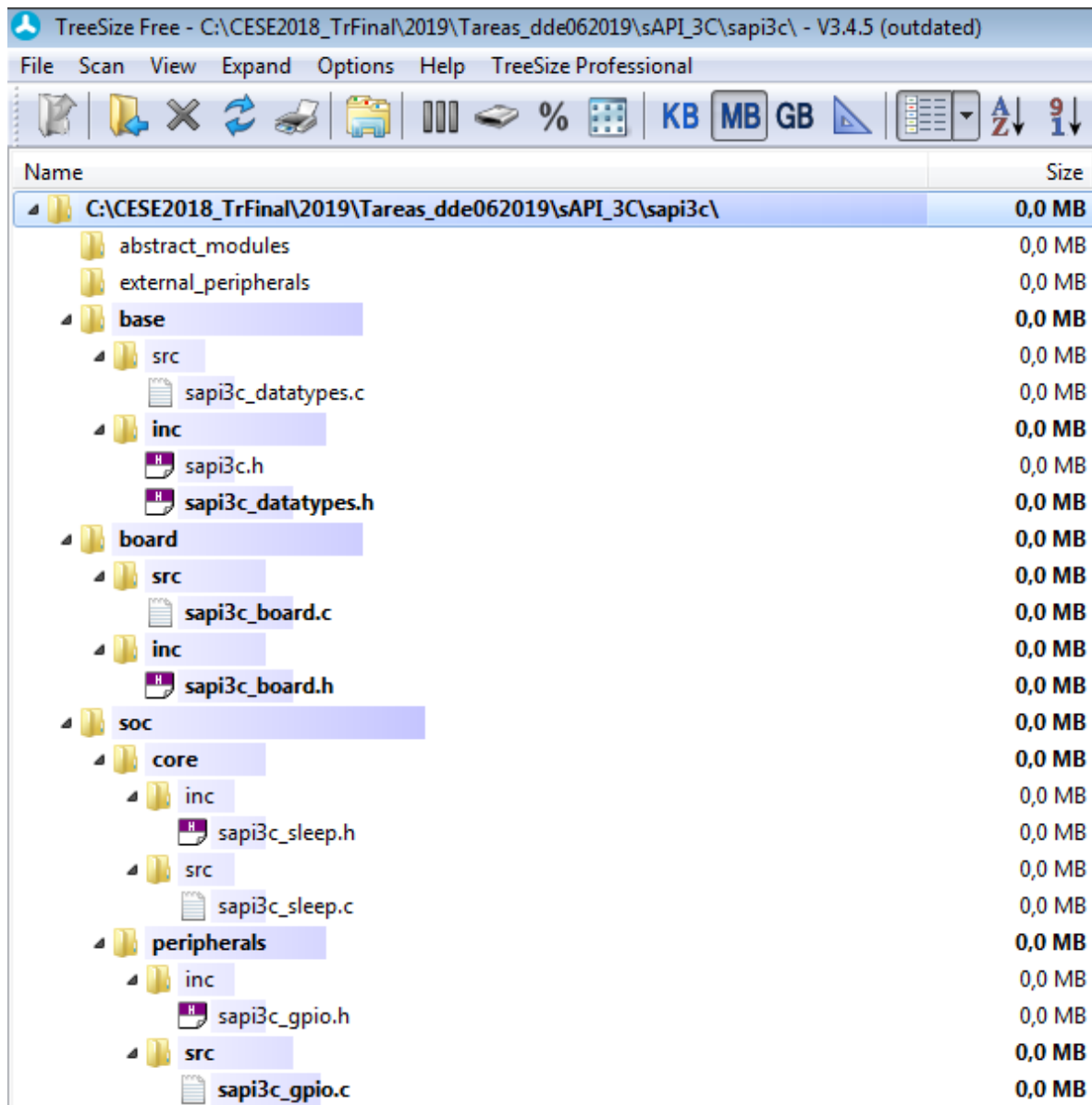
/*=====[cplusplus]=====*/

#ifdef __cplusplus
}
#endif

/*=====[end of file]=====*/
#endif /* #ifndef _SAPI_BOARD_H_ */

```

Hasta ahora nos viene quedando así:



The screenshot shows the TreeSize Free application window. The title bar reads "TreeSize Free - C:\CESE2018\_TrFinal\2019\Tareas\_dde062019\sAPI\_3C\sapi3c\ - V3.4.5 (outdated)". The menu bar includes "File", "Scan", "View", "Expand", "Options", and "Help". The toolbar contains icons for file operations and unit selection (KB, MB, GB). The main window displays a tree view of the directory structure. The root directory is "C:\CESE2018\_TrFinal\2019\Tareas\_dde062019\sAPI\_3C\sapi3c\" with a size of 0,0 MB. It contains subdirectories: "abstract\_modules" (0,0 MB), "external\_peripherals" (0,0 MB), "base" (0,0 MB), "board" (0,0 MB), and "soc" (0,0 MB). The "base" directory contains "src" (0,0 MB) with "sapi3c\_datatypes.c" (0,0 MB) and "inc" (0,0 MB) with "sapi3c.h" (0,0 MB) and "sapi3c\_datatypes.h" (0,0 MB). The "board" directory contains "src" (0,0 MB) with "sapi3c\_board.c" (0,0 MB) and "inc" (0,0 MB) with "sapi3c\_board.h" (0,0 MB). The "soc" directory contains "core" (0,0 MB) with "inc" (0,0 MB) containing "sapi3c\_sleep.h" (0,0 MB) and "src" (0,0 MB) containing "sapi3c\_sleep.c" (0,0 MB), and "peripherals" (0,0 MB) with "inc" (0,0 MB) containing "sapi3c\_gpio.h" (0,0 MB) and "src" (0,0 MB) containing "sapi3c\_gpio.c" (0,0 MB).

Name	Size
C:\CESE2018_TrFinal\2019\Tareas_dde062019\sAPI_3C\sapi3c\	0,0 MB
abstract_modules	0,0 MB
external_peripherals	0,0 MB
base	0,0 MB
src	0,0 MB
sapi3c_datatypes.c	0,0 MB
inc	0,0 MB
sapi3c.h	0,0 MB
sapi3c_datatypes.h	0,0 MB
board	0,0 MB
src	0,0 MB
sapi3c_board.c	0,0 MB
inc	0,0 MB
sapi3c_board.h	0,0 MB
soc	0,0 MB
core	0,0 MB
inc	0,0 MB
sapi3c_sleep.h	0,0 MB
src	0,0 MB
sapi3c_sleep.c	0,0 MB
peripherals	0,0 MB
inc	0,0 MB
sapi3c_gpio.h	0,0 MB
src	0,0 MB
sapi3c_gpio.c	0,0 MB

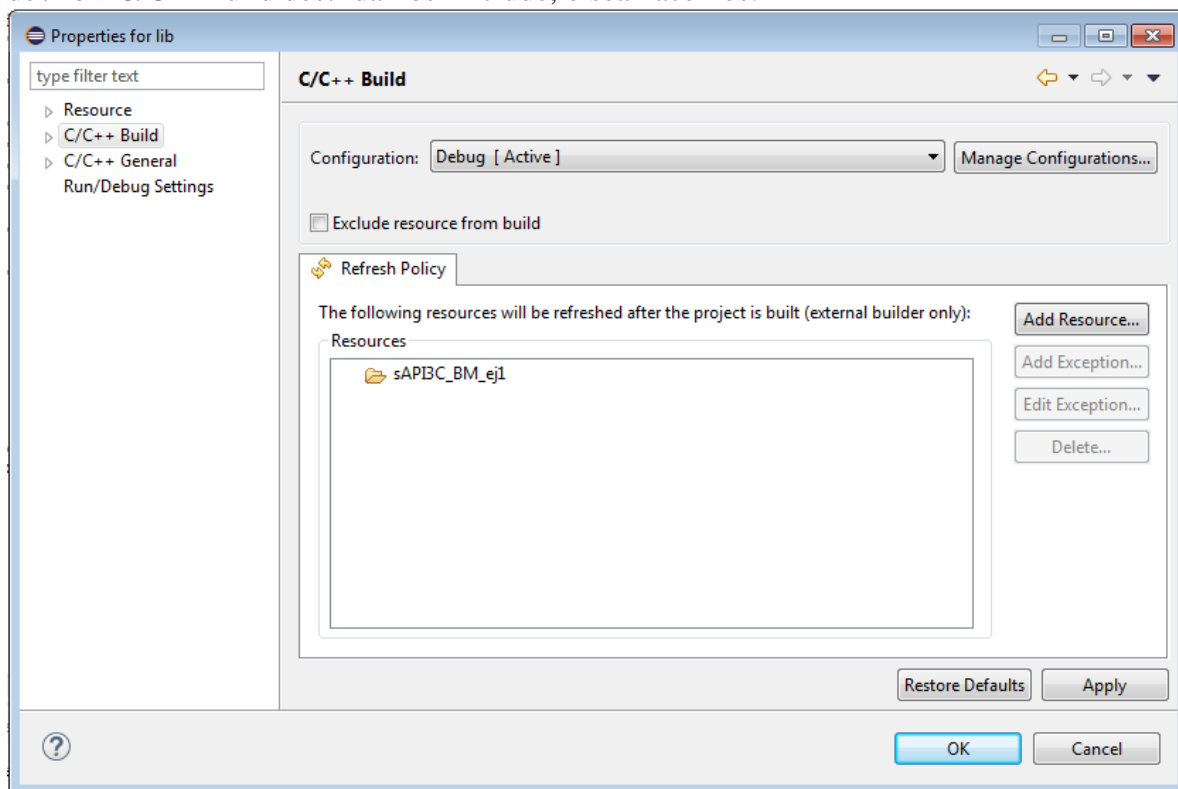
Lo que hacemos, a modo de prueba, es armar un nuevo directorio dentro del Project, sAPI3C\_BM\_ej1, y agregar esta estructura como /lib/sapi3c

## A.4) Ensayo con nuevo directorio..

- A.4.1) Por ahora, sapi3c\_gpio.c /.h serán los mismos gpio.c/.h generados antes, solo que no los incluimos en el /src, /inc principales sino dentro de la /lib/sapi3c.
- A.4.2) Además las definiciones de pines que estaban en main.h ahora estarán en sapi3c\_board.h
- sapi3c\_board.h
- A.4.3) Por ahora, sapi3c\_board.c no contendrá inicialización
- A.4.4) Lazo ppal del main() A.2.2: `/* USER CODE BEGIN WHILE */`

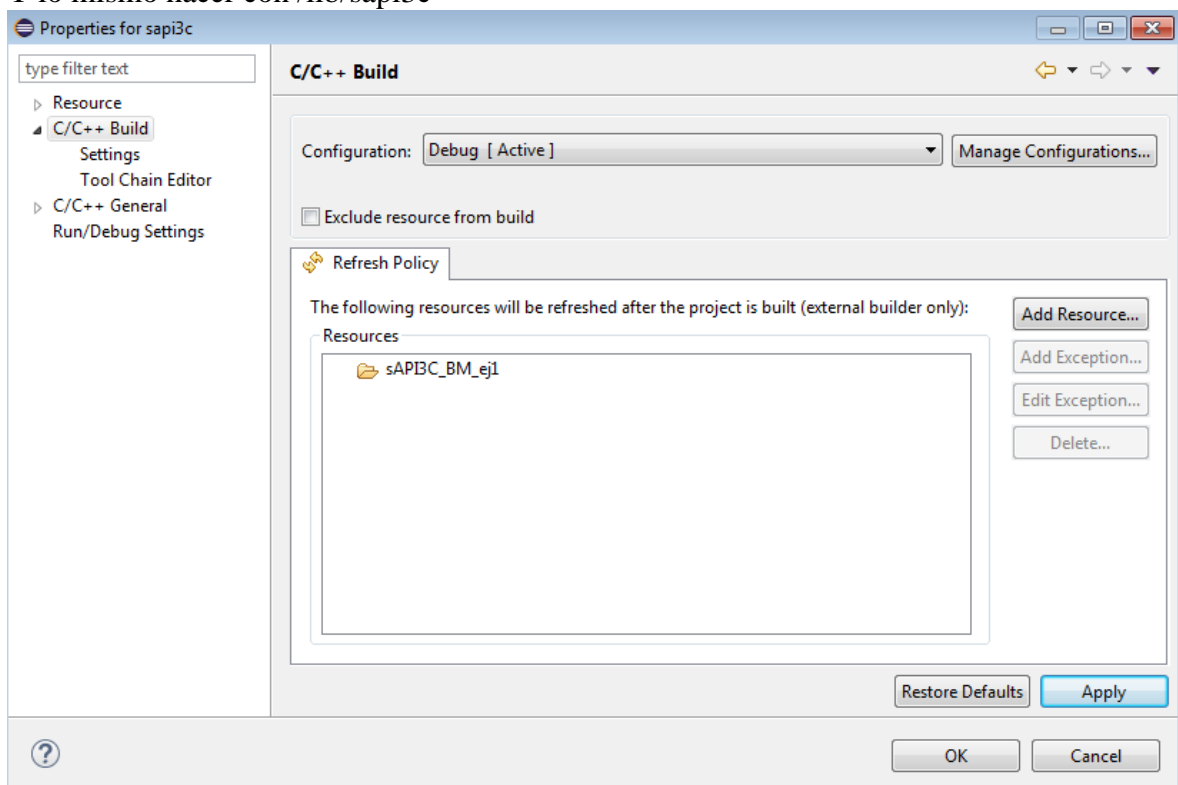
```
while (1)
{
    /* USER CODE END WHILE */
    // from stm32f4xx_ll_utils.h /.c
    // void LL_mDelay(uint32_t Delay)
    LL_mDelay(1000);
    LL_GPIO_TogglePin(OLED_GPIO_Port,OLED_Pin);
    /* USER CODE BEGIN 3 */
}
```

A.4.5) Una vez copiado `/lib/sapi3c` desde el directorio preliminar A3 dentro de Eclipse, en properties de `/lib->C/C++Build` destildamos Exclude, o sea hacemos:



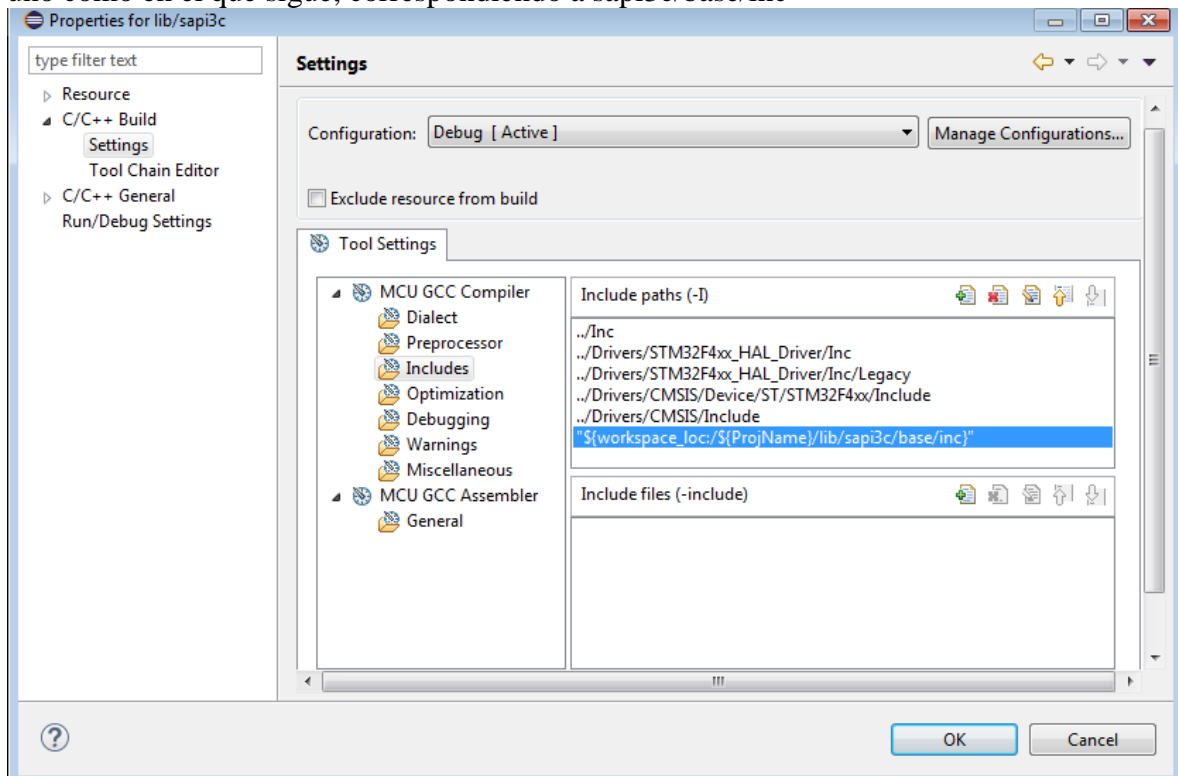
Apply-> OK,

Y lo mismo hacer con `/lib/sapi3c`

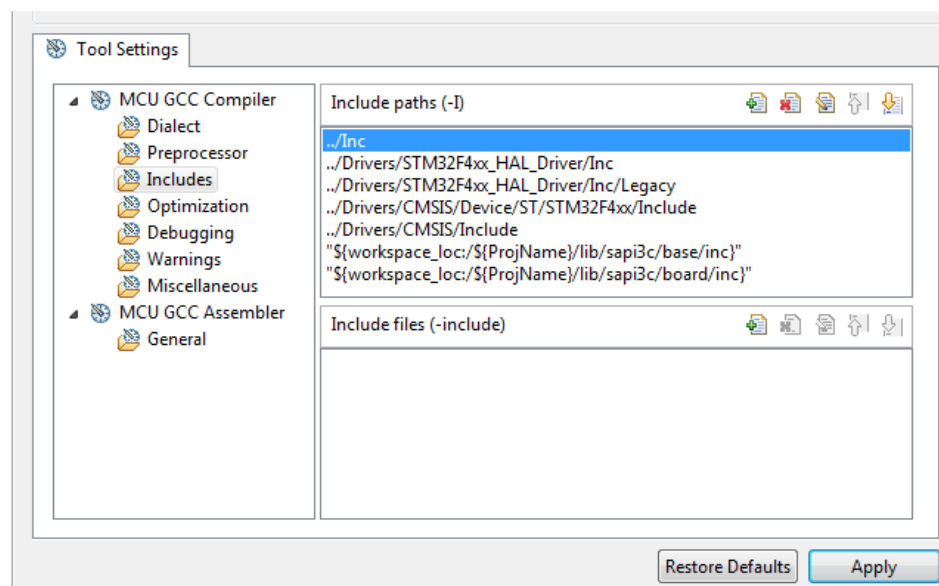


Aparentemente no es necesario con los subdirectorios de `sapi3c` (no aparecen Excluidos, al mirar sus properties).

A.4.6) Ahora tenemos que incluir con (+) en los C/C++Build/Settings/los Includes agregados, de a uno como en el que sigue, correspondiendo a sapi3c/base/inc



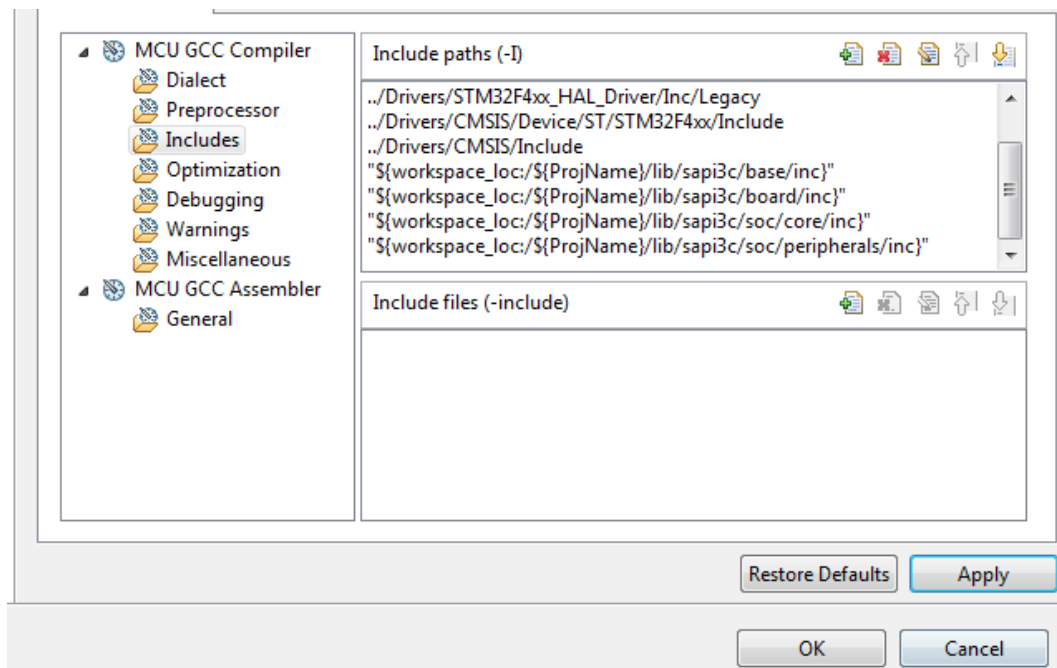
Con /board/inc  
Run/Debug Settings



(si lo pide, Appy -> OK para reconstruir).

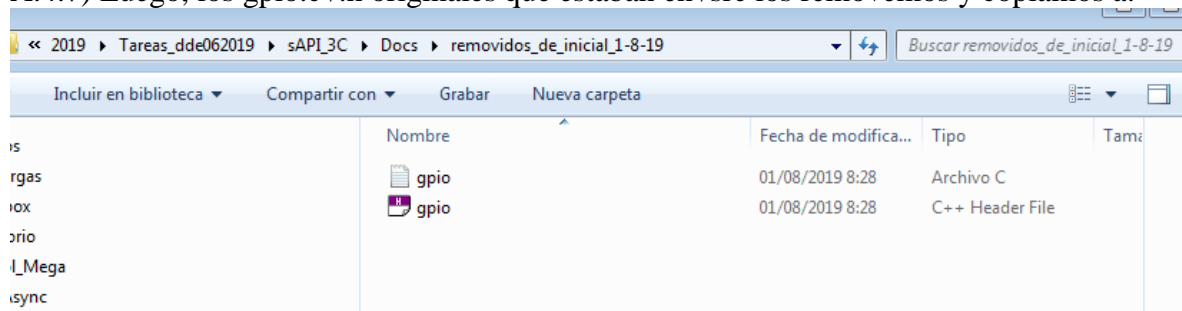
Luego con /sapi3c/soc/core/inc

Y con /sapi3c/soc/peripherals/inc

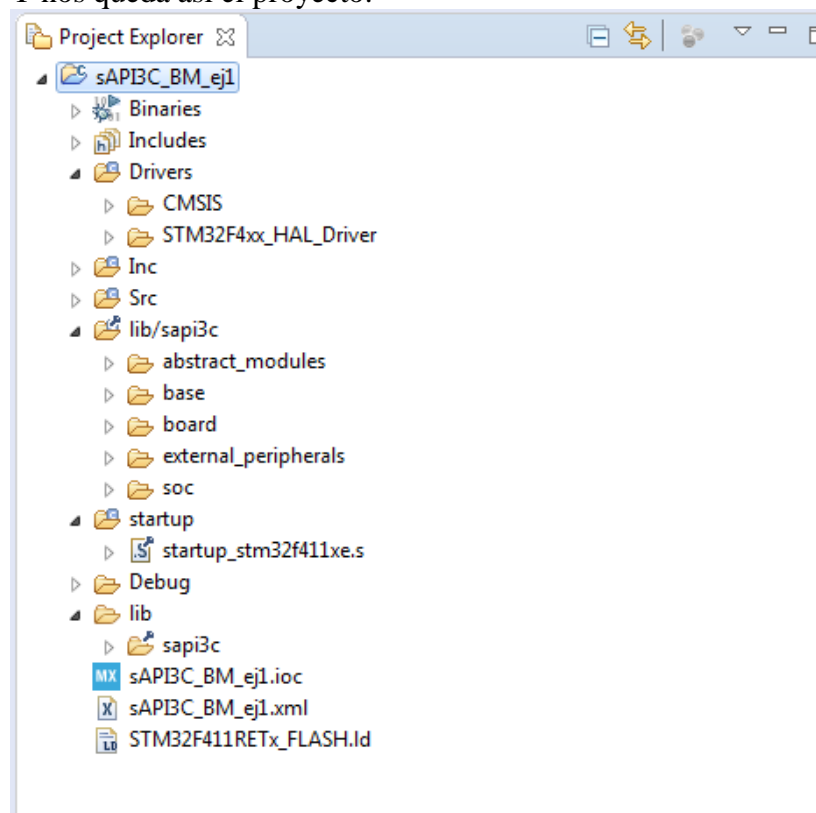


Quedando así..

A.4.7) Luego, los gpio.c /h originales que estaban en /src los removemos y copiamos a:



Y nos queda así el proyecto:



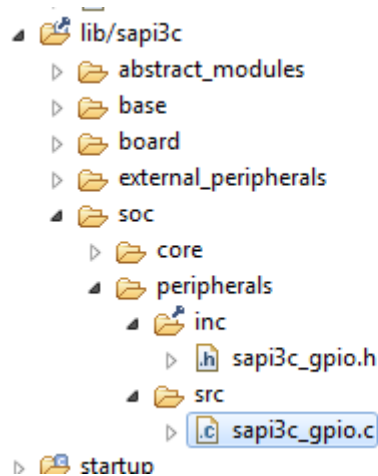
Al intentar compilar, nos falta el sapi3c\_peripheralmap, en sleep:

```

C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULL_LL_DRIVER '-D__weak=__attribute__((weak))' '-D__packed=__attribut
In file included from ../lib/sapi3c/soc/core/src/sapi3c_sleep.c:41:0:
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\lib/sapi3c/soc/core/inc/sapi3c_sleep.h:45:10: fatal error: sapi3c_peripheral_map.h: No such file or directory
#include "sapi3c_peripheral_map.h"
^~~~~~
lib/sapi3c/soc/core/src/subdir.mk:18: recipe for target 'lib/sapi3c/soc/core/src/sapi3c_sleep.o' failed
compilation terminated.
make: *** [lib/sapi3c/soc/core/src/sapi3c_sleep.o] Error 1

```

Y en main.c, el gpio.h ahora se tendría que llamar sapi3c\_gpio.h – NO!



Ya está incluido por sapi3c\_gpio.c

```

CDT Build Console [sAPI3C_BM_ej1]
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULL_LL_DRIVER '-D__weak=__attribute__((weak))' '-D__packed=__attribute__((packed))'
Finished building: ../lib/sapi3c/board/src/sapi3c_board.c

Building target: sAPI3C_BM_ej1.elf
Invoking: MCU GCC Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -specs=nosys.specs -specs=nano.specs -T"../STM32F411RETx_FLASH.ld" -o sAPI3C_BM_ej1.elf
Finished building target: sAPI3C_BM_ej1.elf

make --no-print-directory post-build
Generating hex and Printing size information:
arm-none-eabi-objcopy -O ihex "sAPI3C_BM_ej1.elf" "sAPI3C_BM_ej1.hex"
arm-none-eabi-size "sAPI3C_BM_ej1.elf"
text    data    bss    dec    hex filename
12600    20      2124    14744    3998 sAPI3C_BM_ej1.elf

13:47:42 Build Finished (took 3s.344ms)

```

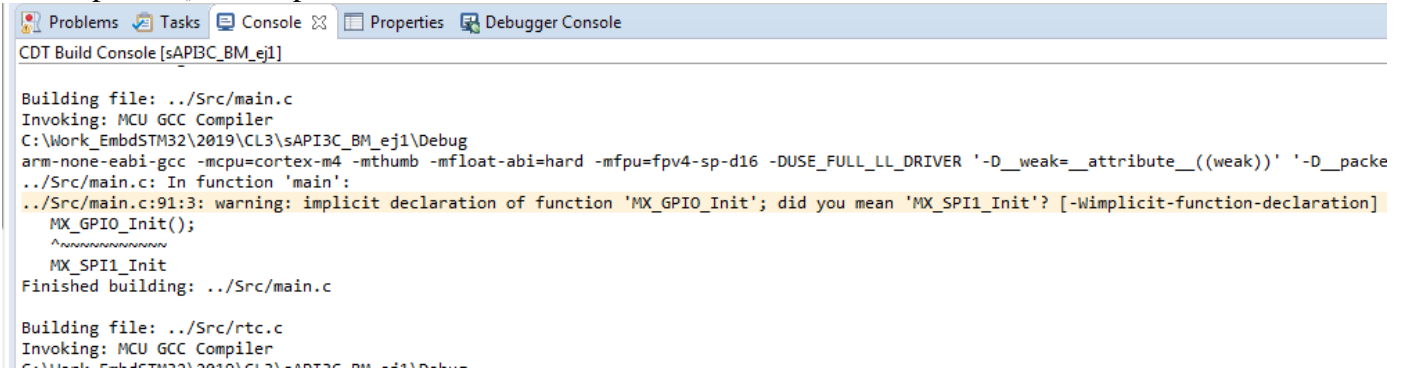
Ahora compila bien.. Vamos a ver si funciona.. (Solo parpadear OLED cada 1 segundo..)

Problema: dentro de sapi3c\_gpio.c aparecía #include <sapi3c\_gpio.h> en vez de #include "sapi3c\_gpio.h"

A.4. 8) PERFECTO!



A.4. 9) 8.2019 – Al intentar recompilar, luego de abrir, me sigue dando el error de que no encuentra MG\_GpioInit() al compilar el main..

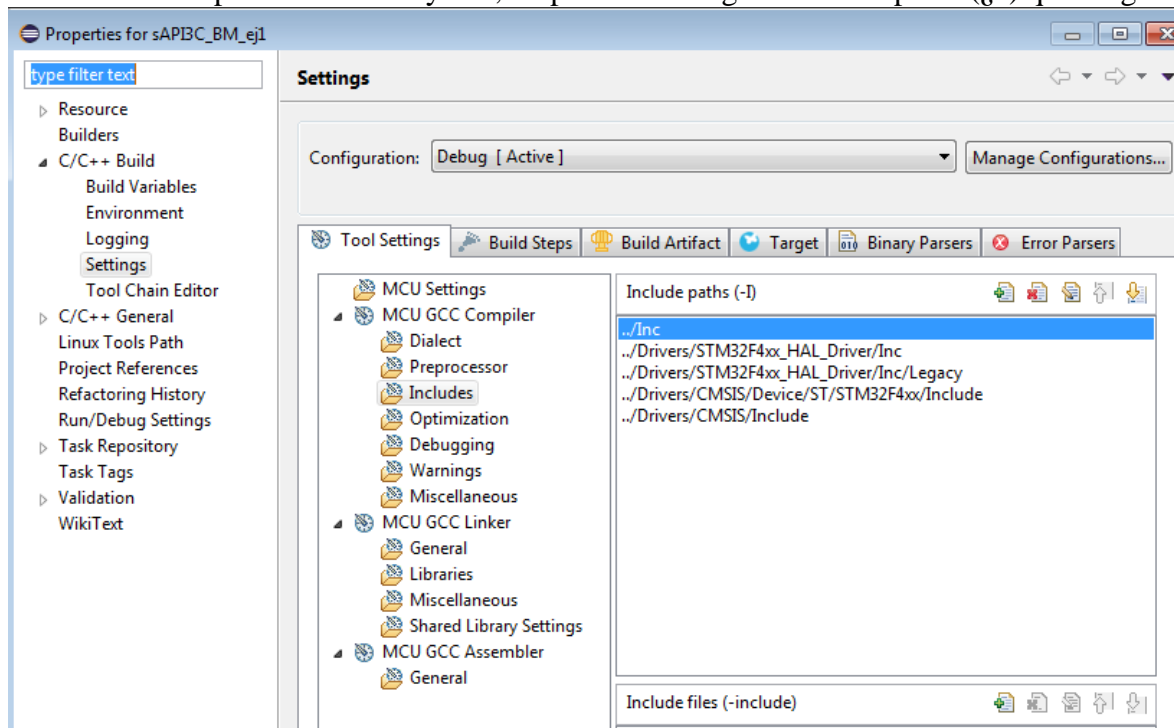


```
Problems Tasks Console Properties Debugger Console
CDT Build Console [sAPI3C_BM_ej1]

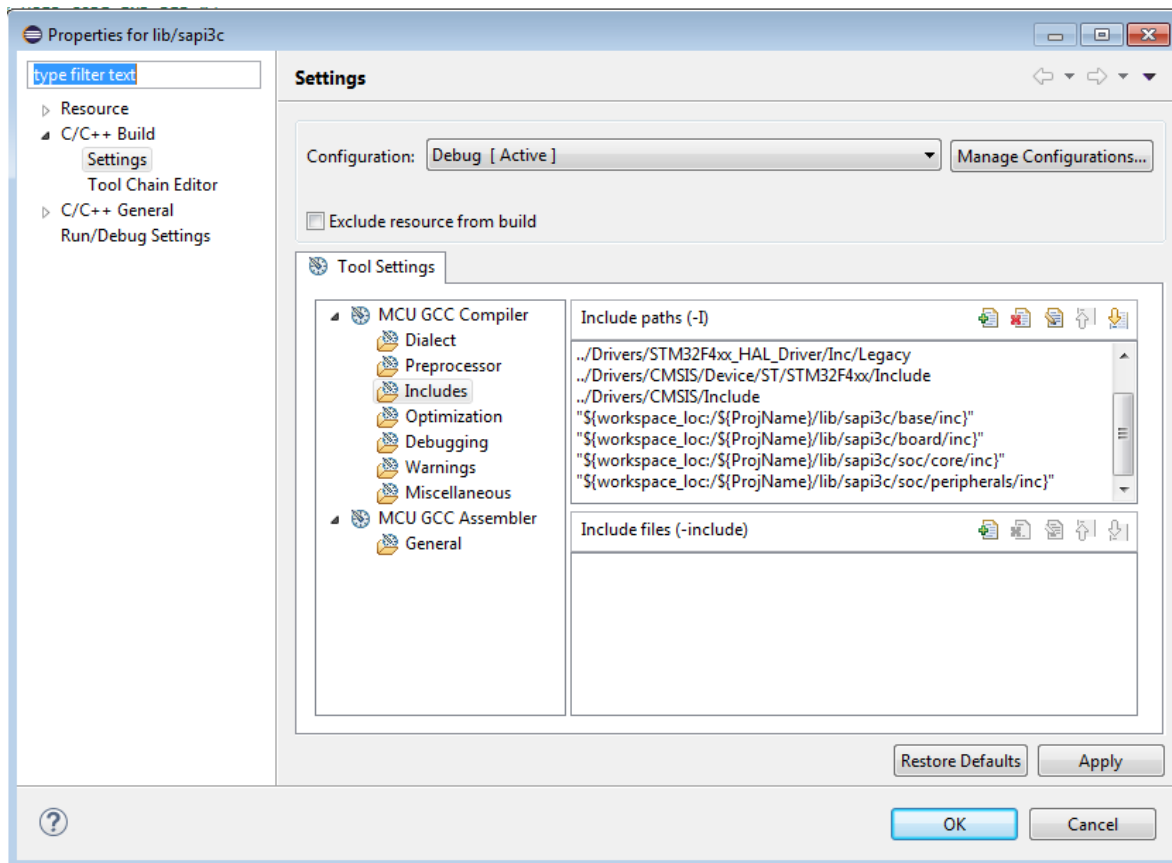
Building file: ../Src/main.c
Invoking: MCU GCC Compiler
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULL_LL_DRIVER '-D__weak=__attribute__((weak))' '-D__packe
../Src/main.c: In function 'main':
../Src/main.c:91:3: warning: implicit declaration of function 'MX_GPIO_Init'; did you mean 'MX_SPI1_Init'? [-Wimplicit-function-declaration]
    MX_GPIO_Init();
    ^
    MX_SPI1_Init
Finished building: ../Src/main.c

Building file: ../Src/rtc.c
Invoking: MCU GCC Compiler
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
```

Mirando las Propiedades del Proyecto, no parece haber guardado los paths (¿?) que cargamos antes:

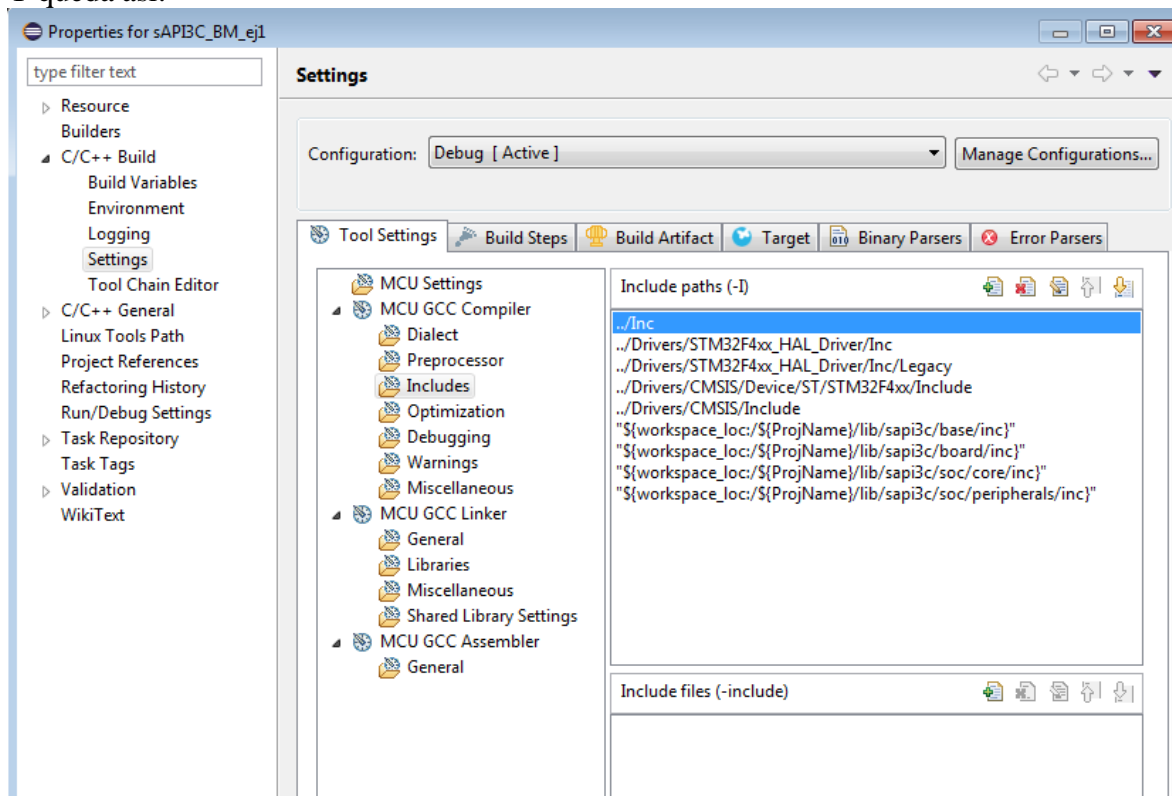


Pero sí están para el directorio que estuvimos trabajando / lib/sapi3c:



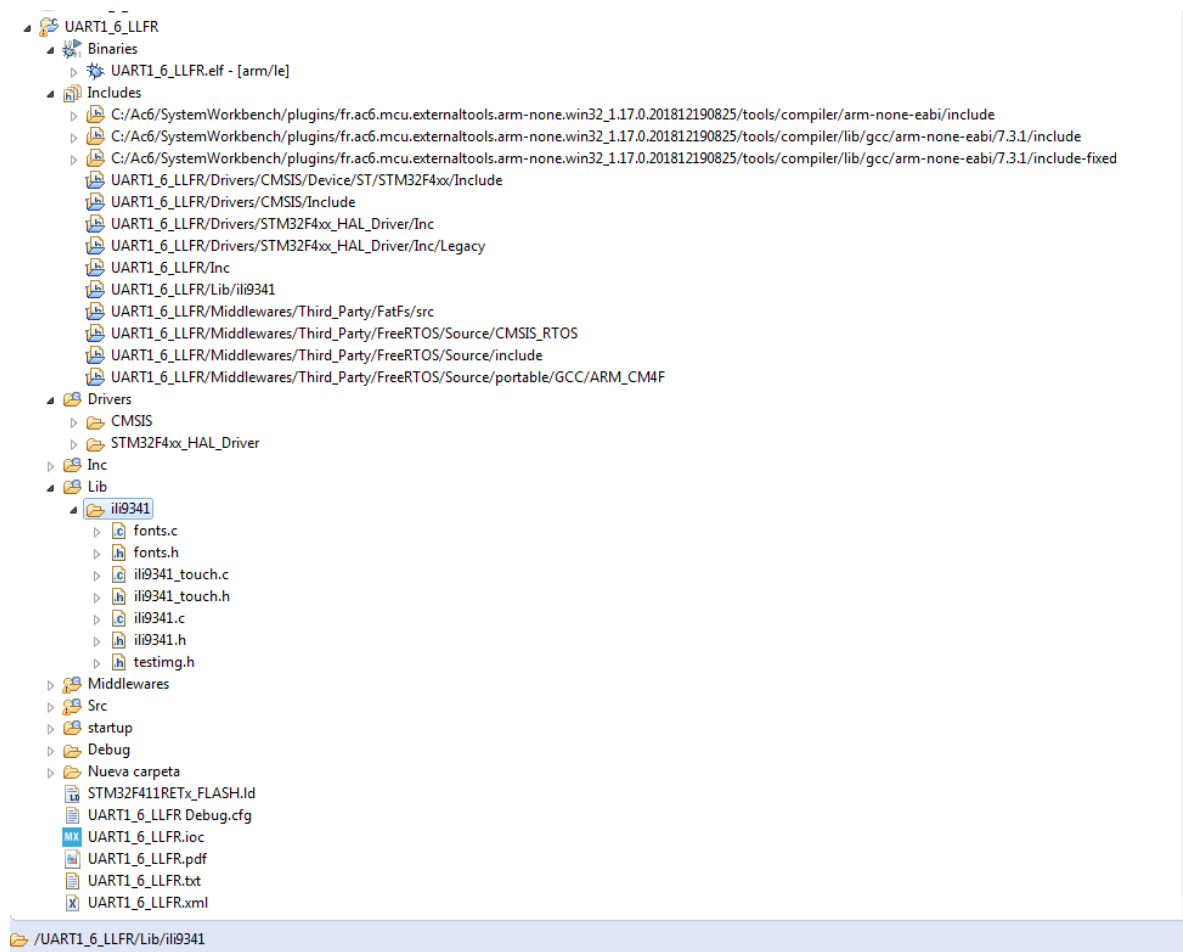
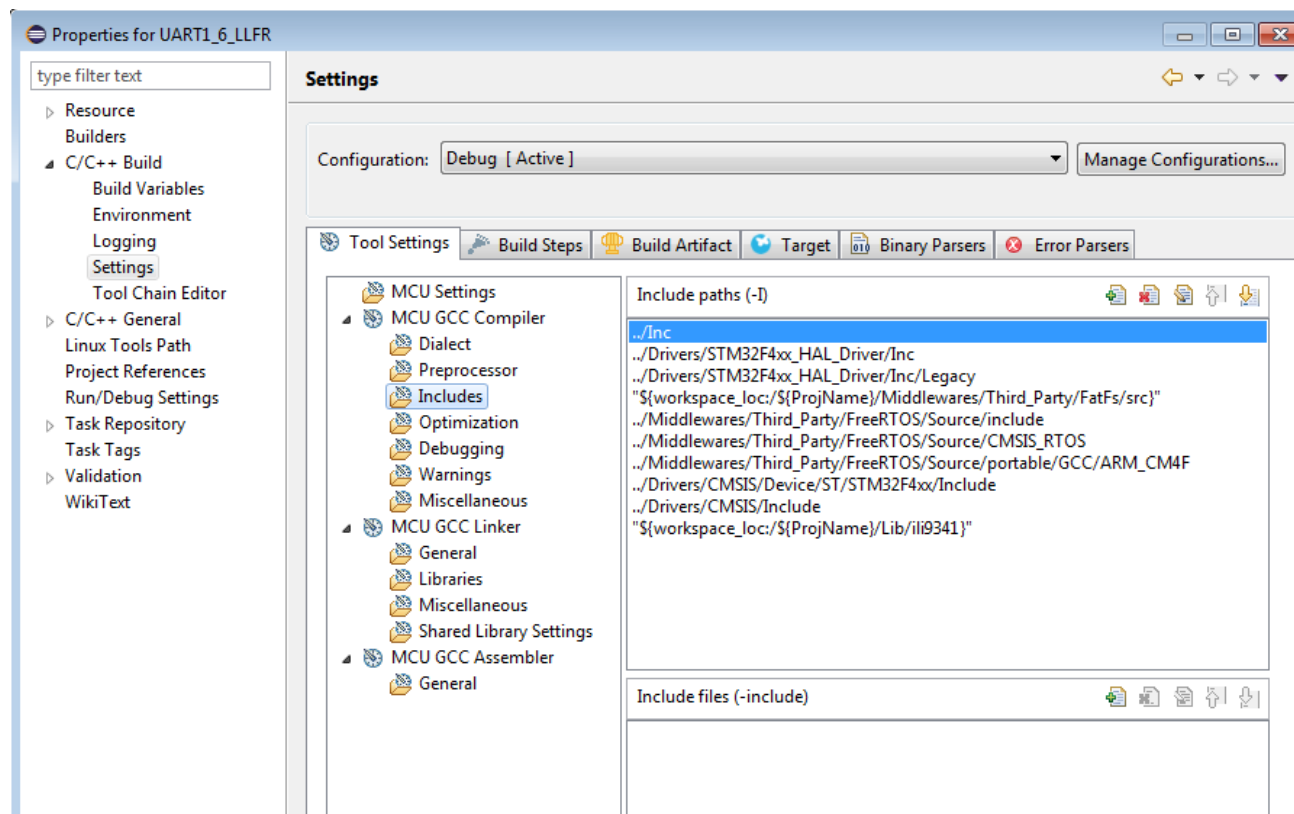
Entonces, lo que aparentemente nos faltó es agregarlo en las propiedades del proyecto completo, es decir pararse en Project ->sAPI32\_BM\_ej1 y agregar en los C/C++ build settings -> includes, los mismos paths que van dentro de /lib/sapi3c, etc..

Y queda así:



(siempre dar Apply, OK en cada paso de agregado..)

Probamos el efecto sobre el Martillito..  
 Sigue sin reconocerlo. Comparamos con UART\_6\_LL





## A.5) gpio..

sapi3c\_gpio.c /.h

```
void gpioInitEnable(void);
bool_t gpioInitInput( inputMap_t input, uint32_t config_pull );
bool_t gpioInitOutput( outputMap_t output);
void gpioInit_INT1(void);
```

Para implementar:

A.5.1) bool\_t gpioRead( inputMap\_t input )

podemos usar 1) LL

```
/**
 * @brief Return if input data level for several pins of dedicated port is high or low.
 * @rmtoll IDR IDy LL_GPIO_IsInputPinSet
 * @param GPIOx GPIO Port
 * @param PinMask This parameter can be a combination of the following values:
 * @arg @ref LL_GPIO_PIN_0
 * @arg @ref LL_GPIO_PIN_1
 * @arg @ref LL_GPIO_PIN_2
 * @arg @ref LL_GPIO_PIN_3
 * @arg @ref LL_GPIO_PIN_4
 * @arg @ref LL_GPIO_PIN_5
 * @arg @ref LL_GPIO_PIN_6
 * @arg @ref LL_GPIO_PIN_7
 * @arg @ref LL_GPIO_PIN_8
 * @arg @ref LL_GPIO_PIN_9
 * @arg @ref LL_GPIO_PIN_10
 * @arg @ref LL_GPIO_PIN_11
 * @arg @ref LL_GPIO_PIN_12
 * @arg @ref LL_GPIO_PIN_13
 * @arg @ref LL_GPIO_PIN_14
 * @arg @ref LL_GPIO_PIN_15
 * @arg @ref LL_GPIO_PIN_ALL
 * @retval State of bit (1 or 0).
 */
__STATIC_INLINE uint32_t LL_GPIO_IsInputPinSet(GPIO_TypeDef *GPIOx, uint32_t PinMask)
{
    return (READ_BIT(GPIOx->IDR, PinMask) == (PinMask));
}
```

Ó 2) HAL

```
/**
 * @brief Reads the specified input port pin.
 * @param GPIOx where x can be (A..K) to select the GPIO peripheral for STM32F429X device or
 *          x can be (A..I) to select the GPIO peripheral for STM32F40XX and
 *          STM32F427X devices.
 * @param GPIO_Pin specifies the port bit to read.
 *          This parameter can be GPIO_PIN_x where x can be (0..15).
 * @retval The input port pin value.
 */
GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
    GPIO_PinState bitstatus;

    /* Check the parameters */
    assert_param(IS_GPIO_PIN(GPIO_Pin));
}
```

```

if((GPIOx->IDR & GPIO_Pin) != (uint32_t)GPIO_PIN_RESET)
{
    bitstatus = GPIO_PIN_SET;
}
else
{
    bitstatus = GPIO_PIN_RESET;
}
return bitstatus;
}

```

Al final usamos el primero, y nos queda:

```

bool_t gpioRead( inputMap_t input )
{
    uint32_t ret_val = 0;
    switch(input){
        case KBD_ABJ:
            ret_val = LL_GPIO_IsInputPinSet(GPIOB, K_ABJ_PB0_Pin);
            break;
        case KBD_DER:
            ret_val = LL_GPIO_IsInputPinSet(GPIOB, K_DER_PB1_Pin);
            break;
        case KBD_IZQ:
            ret_val = LL_GPIO_IsInputPinSet(GPIOC, K_IZQ_PC4_Pin);
            break;
        case KBD_ARR:
            ret_val = LL_GPIO_IsInputPinSet(GPIOC, K_ARR_PC5_Pin);
            break;
        case SDIO_INS:
            ret_val = LL_GPIO_IsInputPinSet(GPIOC, SD_INS_Pin);
            break;
        case SDIO_CD:
            ret_val = LL_GPIO_IsInputPinSet(GPIOC, SDIO_CD_Pin_PC11);
            break;
        default:
            ret_val = 1;
            break;
    }
    return ((bool_t)ret_val);
}

```

#### A.5.2) Funcion original:

```

bool_t gpioWrite( gpioMap_t pin, bool_t value )
{
    bool_t ret_val = 1;

    int8_t pinNamePort = 0;
    int8_t pinNamePin = 0;

    int8_t func = 0;

    int8_t gpioPort = 0;
    int8_t gpioPin = 0;

    gpioObtainPinInit( pin, &pinNamePort, &pinNamePin, &func,

```

```

        &gpioPort, &gpioPin );

    Chip_GPIO_SetPinState( LPC_GPIO_PORT, gpioPort, gpioPin, value);

    return ret_val;
}

```

Para escribir las salidas, usamos:

```

/**
 * @brief Set several pins to high level on dedicated gpio port.
 * @rmtoll BSRR          BSy          LL_GPIO_SetOutputPin
 * @param GPIOx GPIO Port
 * @param PinMask This parameter can be a combination of the following values:
 *               @arg @ref LL_GPIO_PIN_0
 *
 * ...
 *               @arg @ref LL_GPIO_PIN_15
 *               @arg @ref LL_GPIO_PIN_ALL
 * @retval None
 */
__STATIC_INLINE void LL_GPIO_SetOutputPin(GPIO_TypeDef *GPIOx, uint32_t PinMask)
{
    WRITE_REG(GPIOx->BSRR, PinMask);
}

/**
 * @brief Set several pins to low level on dedicated gpio port.
 * @rmtoll BSRR          BRy          LL_GPIO_ResetOutputPin
 * @param GPIOx GPIO Port
 * @param PinMask This parameter can be a combination of the following values:
 *               @arg @ref LL_GPIO_PIN_0
 *
 * ..
 *               @arg @ref LL_GPIO_PIN_ALL
 * @retval None
 */
__STATIC_INLINE void LL_GPIO_ResetOutputPin(GPIO_TypeDef *GPIOx, uint32_t PinMask)
{
    WRITE_REG(GPIOx->BSRR, (PinMask << 16));
}

```

