

16.08.2019 CESE_2019_PROY-FINAL

R. Oliva

Tareas de Avance con sAPI (de E.Pernia) – Upd (f) 16-08-19

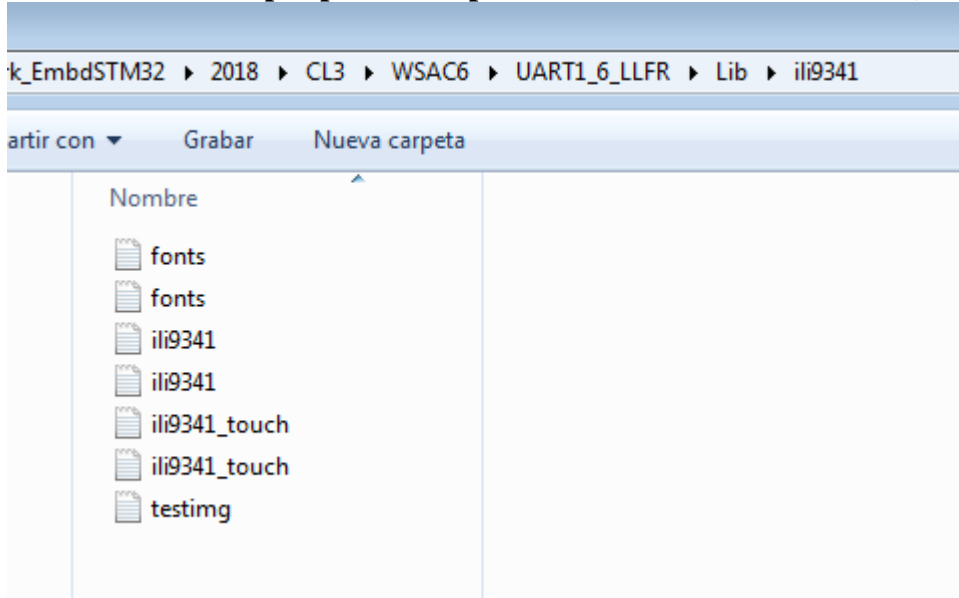
ENSAYO EJ03 – Incorporar Display al _ej2 en /lib/external_peripherals/inc/sapi3c_display.h
Y /lib/external_peripherals/src/sapi3c_display.c

(✓ Indica Realizado)

D.0) Requerimientos display:

-D.0.1) Tomamos modelo ya usado en UART_LL_6, pero en ese momento sin librerías sapi3c.

- En: /lib/external_peripherals/ copiar el directorio actual /ili9341 (usado en LL6)



- A lo que ya hay en _ej2, se agrega, dentro de /lib:

external_peripherals/ili9341
external_peripherals/inc/sapi3c_display.h
external_peripherals/src/sapi3c_display.c

- Primero lo hacemos en DB01 (directorio de ensayo) ✓

C:\CESE2018_TrFinal\2019\Tareas_dde062019\sAPI_3C\sapi3c\external_peripherals\inc\sapi3c_display.h

a) Inclusión de definiciones y librerías requeridas

a.1) en sapi3c_display.h ✓

```
/* typedef enum {  
    ILI9341 = 0, // Default Display  
    DISP2,      // Alternative 2  
    DISP3       // Alternative 3  
} display_t;  
  
/* typedef enum {  
    WELCOME = 0, // Default Welcome Screen  
    SCREEN1,     // Alternative Screen 1  
    SCREEN2,     // Alternative Screen 2  
    SCREEN3,     // Alternative Screen 3  
    SCREEN4      // Alternative Screen 3  
} screen_t;
```

a.2) en sapi3c_display.c

```
uint8_t displaycl3_Config(display_t distype);
debe incluir equivalentes en LL6 de:
    MX_SPI_Init(); //prepara SPI para acceso a display, define Handle hspi1
    init(); // de Afiskon, llama a Unselect, Touchunselect, luego ILI9341_Init()
    HAL_SPI_DeInit(&hspi1);
    Seleccionar Prescaler SPI_2
    HAL_SPI_Init(&hspi1);
```



b) en main.c, se agrega:

```
displaycl3_Config(ILI9341);
displaycl3_Screen(WELCOME);
switch(Key_apretada){
    case F1:
        displaycl3_Screen(SCREEN1);
        break;
    // etc..
```



-D.0.2) Cambiamos main.c en DB01 – Ej3, que es (final):

```
/**
*****
* @author E.Pernia / adapted by R.Oliva
* @brief Ej 03 This file Tests UART+Display for CL3
* rev 10.8.19
*
**/
/* USER CODE END Header */

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

#define MAX_LOOPS 15
//global space for debug string
char usr_msg[100]={0};
bool_t valor = 1;
uint8_t btest = 0;
uint8_t loop_count = MAX_LOOPS;

/**
* @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);
    btest = displaycl3_Config(ILI9341);
    if(btest) {
        sprintf(usr_msg, "\r\n Error Init Display %d \r\n", btest);
        printmsg_cl3(usr_msg);
    }
}
```

```

displaycl3_Screen(WELCOME);

while (1)
{
    valor = !gpioRead(KBD_IZQ);
    if(valor) {
        sprintf(usr_msg, "\r\n Tecla Izquierda Presionada (F1)");
        printmsg_cl3(usr_msg);
        displaycl3_Screen(SCREEN1);
    }
    valor = !gpioRead(KBD_ARR);
    if(valor) {
        sprintf(usr_msg, "\r\n Tecla Arriba Presionada (F2)");
        printmsg_cl3(usr_msg);
        displaycl3_Screen(SCREEN2);
    }
    valor = !gpioRead(KBD_ABJ);
    if(valor) {
        sprintf(usr_msg, "\r\n Tecla Abajo Presionada (F3)");
        printmsg_cl3(usr_msg);
        displaycl3_Screen(SCREEN3);
    }
    valor = !gpioRead(KBD_DER);
    if(valor) {
        sprintf(usr_msg, "\r\n Tecla Derecha Presionada (F4)");
        printmsg_cl3(usr_msg);
        displaycl3_Screen(SCREEN4);
    }

    HAL_Delay(500);
    gpioWrite(OLED_PB2, ON);
    HAL_Delay(500);
    gpioWrite(OLED_PB2, OFF);
    if(--loop_count == 0){
        loop_count = MAX_LOOPS;
        displaycl3_Screen(WELCOME);
    }
}
}

```

-D.0.3) WelcomeSCREEN:

```

/*y_coord, *str          Font      Color      BgColor      */
ILI9341_WriteString (20, /*      x_coord,*/
10, /*      x_coord,*/
"CESE - Especializacion en      Sistemas Embebidos - FIUBA",
Font_7x10, /* Font */
ILI9341_RED, /* Font color */
ILI9341_BLACK); /* BG color */
ILI9341_WriteString(10, 6*10, " Rafael Oliva - CL3", Font_11x18, ILI9341_GREEN, ILI9341_BLACK);
ILI9341_WriteString(20, 4*10+4*18, "Trabajo Final", Font_16x26, ILI9341_BLUE, ILI9341_BLACK);
ILI9341_WriteString(80, 4*10+6*18, "2019", Font_16x26, ILI9341_BLUE, ILI9341_BLACK);

```

Agregar: ..

```

ILI9341_WriteString (20, /*      x_coord,*/
120, /*      x_coord,*/
"Presione una Tecla F1-F4",
Font_7x10, /* Font */
ILI9341_RED, /* Font color */
ILI9341_BLACK); /* BG color */

```



-D.0.4) Modificar sapi3c.h para agregar inicializacion display:

```
#include "sapi3c_datatypes.h"

// Peripheral Drivers

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

#include "sapi3c_gpio.h"           // Use GPIO peripherals
#include "sapi3c_delay.h"          // Use sapi_tick module
#include "sapi3c_uart.h"           // Use UART peripherals

#include "sapi3c_display.h"        // Use DISPLAY peripheral - includes SPI1 init

#ifdef SECOND_TEST
```

D0.5 Formas Finales Menus ej03_ en sapi3c_display.c

```
/**
 * @brief Welcome_Screen()
 *        Sets up initial border, content message.
 * @param none
 * @retval none
 */

void Welcome_Screen(void){

    for(int x = 0; x < ILI9341_WIDTH; x++) {
        ILI9341_DrawPixel(x, 0, ILI9341_RED);
        ILI9341_DrawPixel(x, ILI9341_HEIGHT-1, ILI9341_RED);
    }

    for(int y = 0; y < ILI9341_HEIGHT; y++) {
        ILI9341_DrawPixel(0, y, ILI9341_RED);
        ILI9341_DrawPixel(ILI9341_WIDTH-1, y, ILI9341_RED);
    }
}
```

```

}

HAL_Delay(500);

// Check fonts
ILI9341_FillScreen(ILI9341_BLACK);
/*      x_coord, y_coord, *str
Font      Color      BGColor      */
ILI9341_WriteString(20, 10, "CESE - Especializacion en          Sistemas Embebidos - FIUBA",
Font_7x10, ILI9341_RED, ILI9341_BLACK);
ILI9341_WriteString(10, 6*10, " Rafael Oliva - CL3", Font_11x18, ILI9341_GREEN,
ILI9341_BLACK);
ILI9341_WriteString(20, 4*10+4*18, "Trabajo Final", Font_16x26, ILI9341_BLUE,
ILI9341_BLACK);
ILI9341_WriteString(80, 4*10+6*18, "2019", Font_16x26, ILI9341_BLUE, ILI9341_BLACK);

HAL_Delay(1000);
ILI9341_WriteString(20, 4*10+12*18, " Presione F1 a F4 ", Font_11x18, ILI9341_GREEN,
ILI9341_BLACK); // Agregado 16.8.2019
// ILI9341_InvertColors(true);
// HAL_Delay(1000);
// ILI9341_InvertColors(false);

}

static void Screen1(void)
{
    // F1 - Green Screen - IZQ
    ILI9341_FillScreen(ILI9341_CYAN);
    ILI9341_WriteString(20, 4*10+4*18, "Tecla F1 pres", Font_16x26, ILI9341_BLACK,
ILI9341_CYAN ); // Agregado 16.8.2019
    HAL_Delay(1000);
}

static void Screen2(void)
{
    // F2 - RED Screen - ARRIBA
    ILI9341_FillScreen(ILI9341_RED);
    ILI9341_WriteString(20, 4*10+4*18, "Tecla F2 pres", Font_16x26, ILI9341_BLACK,
ILI9341_RED); // Agregado 16.8.2019
    HAL_Delay(1000);
}

static void Screen3(void)
{
    // F3 - BLUE Screen - ABJ
    ILI9341_FillScreen(ILI9341_BLUE);
    ILI9341_WriteString(20, 4*10+4*18, "Tecla F3 pres", Font_16x26, ILI9341_BLACK,
ILI9341_BLUE); // Agregado 16.8.2019
/* " Presione F1 a F4 " */
    HAL_Delay(1000);
}

static void Screen4(void)
{
    // F4 - WHITE Screen - DER
    ILI9341_FillScreen(ILI9341_BLACK);
    ILI9341_WriteString(20, 4*10+4*18, "Tecla F4 pres", Font_16x26, ILI9341_WHITE,
ILI9341_BLACK); // Agregado 16.8.2019
    HAL_Delay(1000);
}
}

```

-D.0.6) Recordar habilitar HAL_SPI en HAL_Conf.h

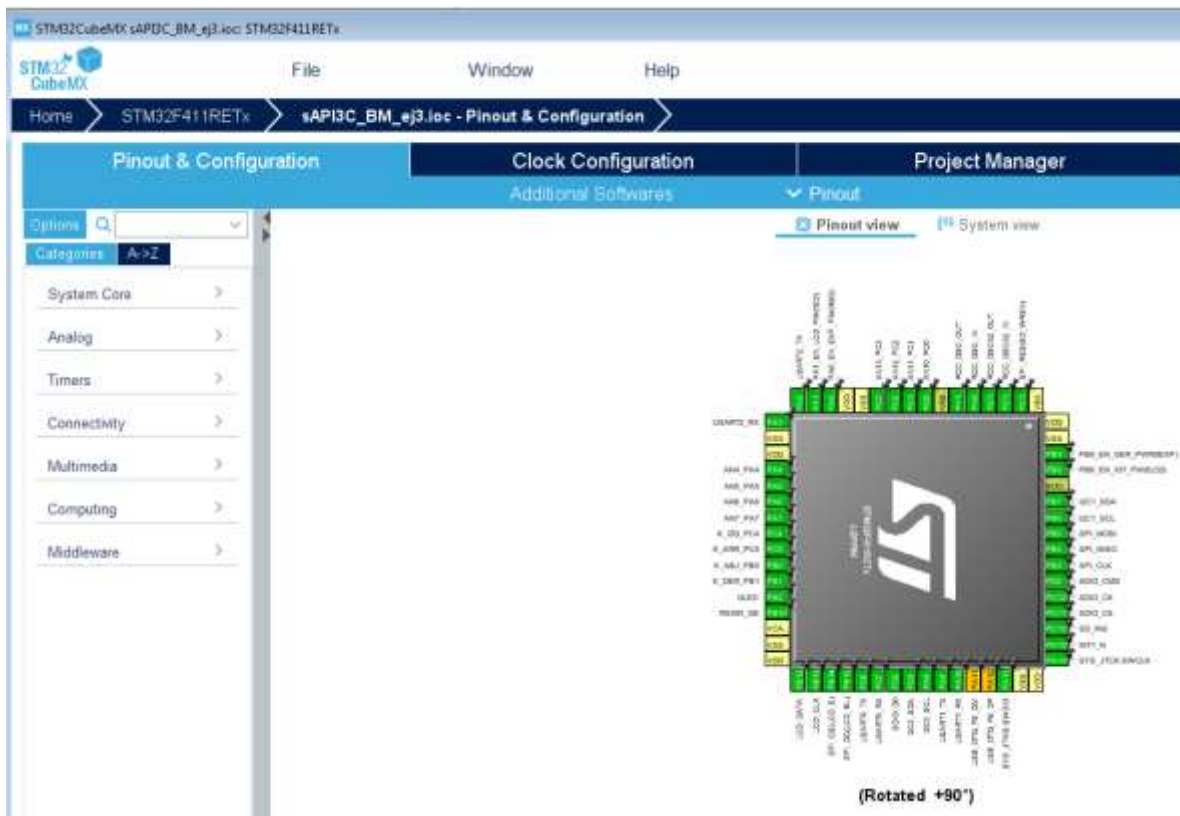
D.1) sAPI3C_ej3

Ahora Intentamos replicar el Proyecto sAPI3C_ej2, en un sAPI3C_ej3, dentro del mismo Workspace. Para esto, no es posible copiar directamente.

D.1) Pasos de replicación:

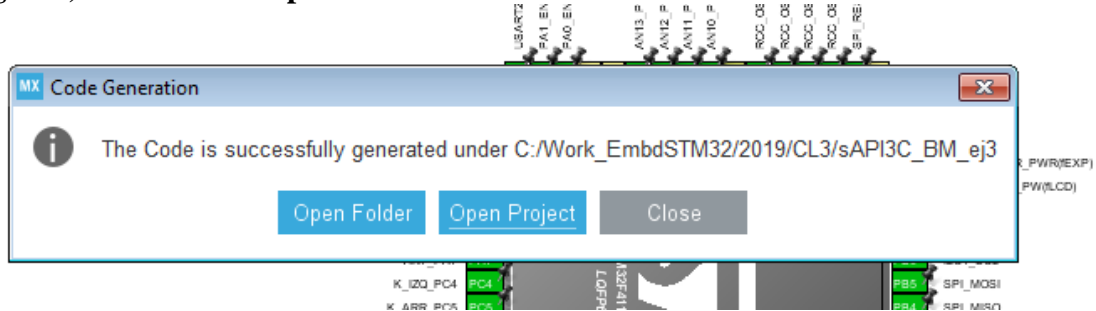
D.1a) Cerrar AC6

D.1b) Abrir el Proyecto desde CubeMX, y hacerle un Guardar Como \sAPI3C_BM_ej3, en el mismo Workspace: C:\Work_EmbdSTM32\2019\CL3 .. ✓

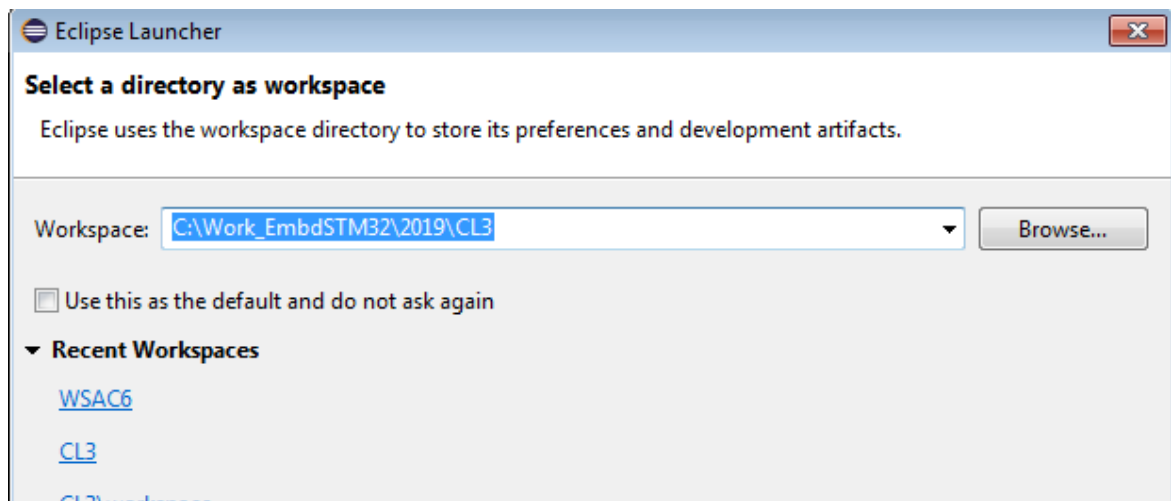


Si le damos aquí “Generate Code”, me reproduce todo lo del ejemplo 2, aunque después tendremos que hacer lo siguiente:

D.1c) Cuando termine el GenerateCode en CubeMX, abrir el Proyecto (dar la opción Open cuando pregunta) en AC6. La importación entonces es automática..

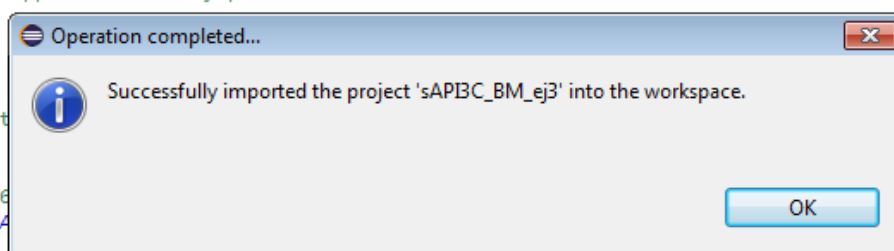


Workspace \2019\CL3

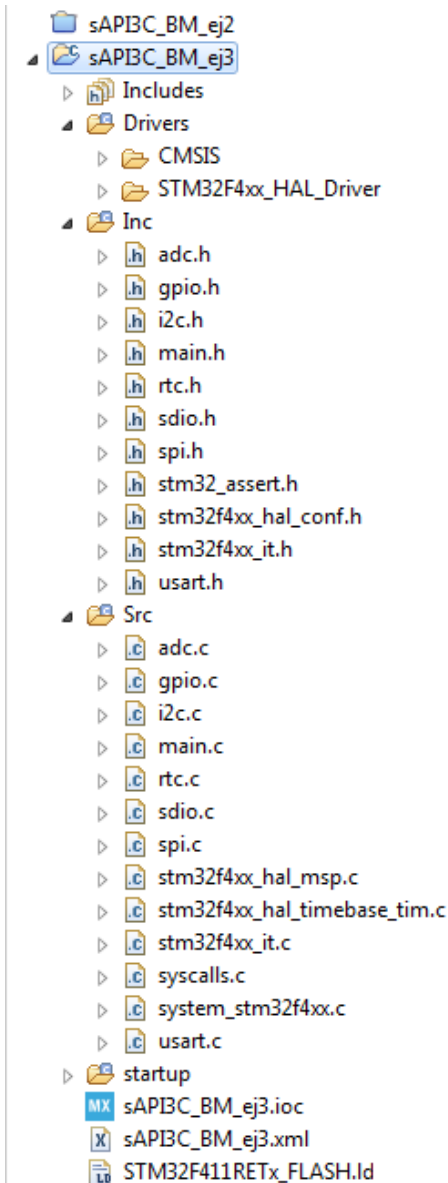


..paciencia porque tarda..

```
main() - The application entry point.
int
id)
ard configurat
CL3();
rminal @UART 6
g(UART_TER, BA
sr_msg, "\r\n Trabajo Final R.Oliva 2019 \r\n");
```



D.1d) Cuando Abrirlo aquí y ver que está todo, dar un refresh. ✓



D.2) Para replicar el mismo proyecto _ej2, con otro nombre, se requiere:

D.2.1) Cerrar AC6 y CubeMX. ✓

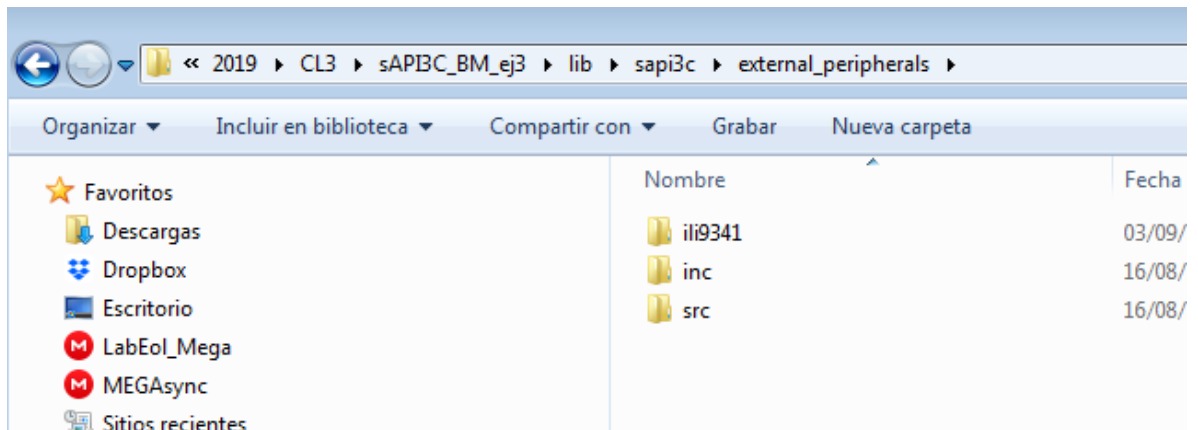
D.2.2) Desde el Explorer dentro del Workspace Workspace \2019\CL3, copiar /lib completo desde _ej2 a _ej3 ✓.

D.2.3) Borrar todos los archivos nuevos de /Src e /Inc, generados por CubeMX. ✓.

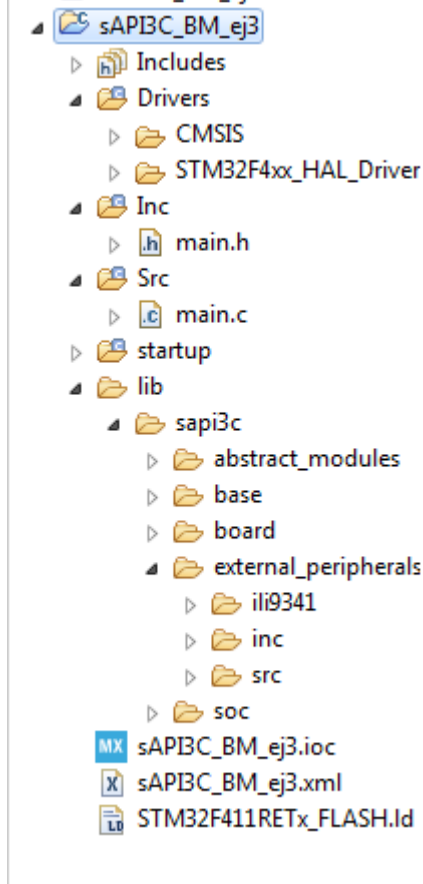
D.2.4) Copiar desde _ej2 a _ej3: main.c a /Src, main.h a /Inc. ✓.

D.2.5) - A lo que ya hay en _ej3, se agrega, dentro de /lib (ya generado en DB01):

external_peripherals/ili9341
external_peripherals/inc/sapi3c_display.h
external_peripherals/src/sapi3c_display.c



D.2.6) Entrar a AC6 de nuevo, dar Refresh (F5) con _ej3 abierto:



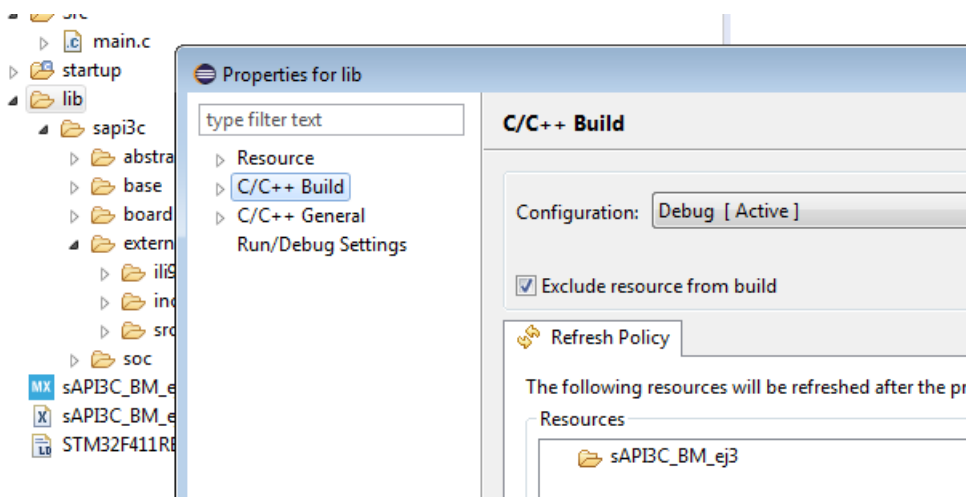
D.2.7) Aplicar (A.E.1) a /lib

- 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”).

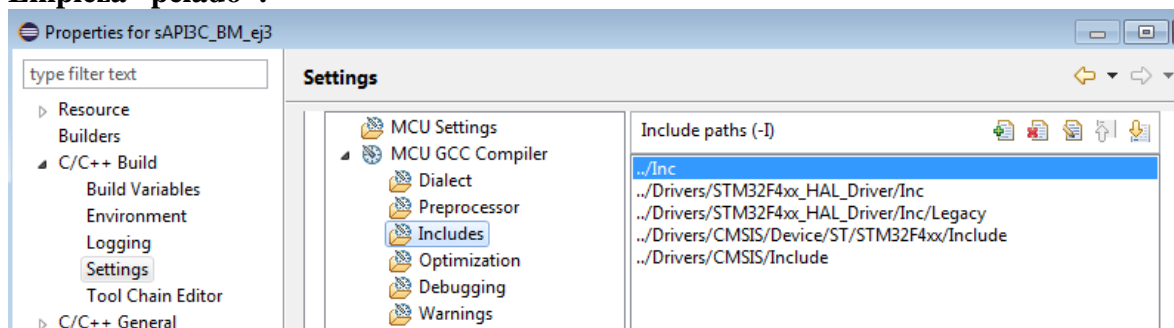
Por ejemplo para este /lib..destildar el “exclude”...

+Apply

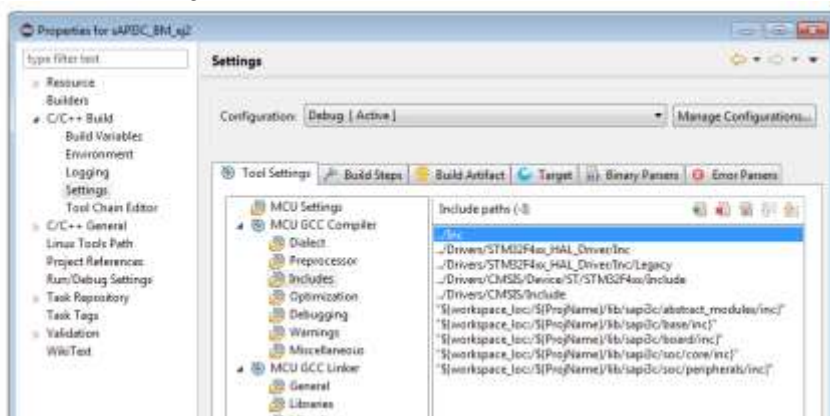
+OK . ✓



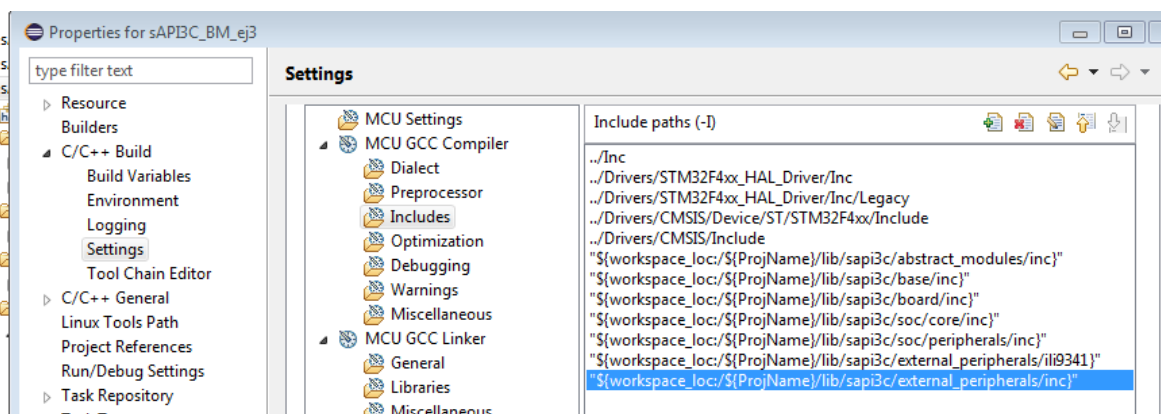
- Finalmente, parado en el directorio del proyecto _ej3 , en properties del Proyecto, C/C++ Build-> settings->MCU GCC-> Includes --- presionar el “+” para agregar los .h que se incorporan de los nuevos directorios.
Empieza “pelado”:



Este era el del ej02_



y agregamos los de display..



D.2.8) Con esto ensayamos compilar con martillito..errores:

-- Nos faltaron agregar en sapi3c_display.h

D.2.8.1) la función void CL3_SPI1_Init(void);

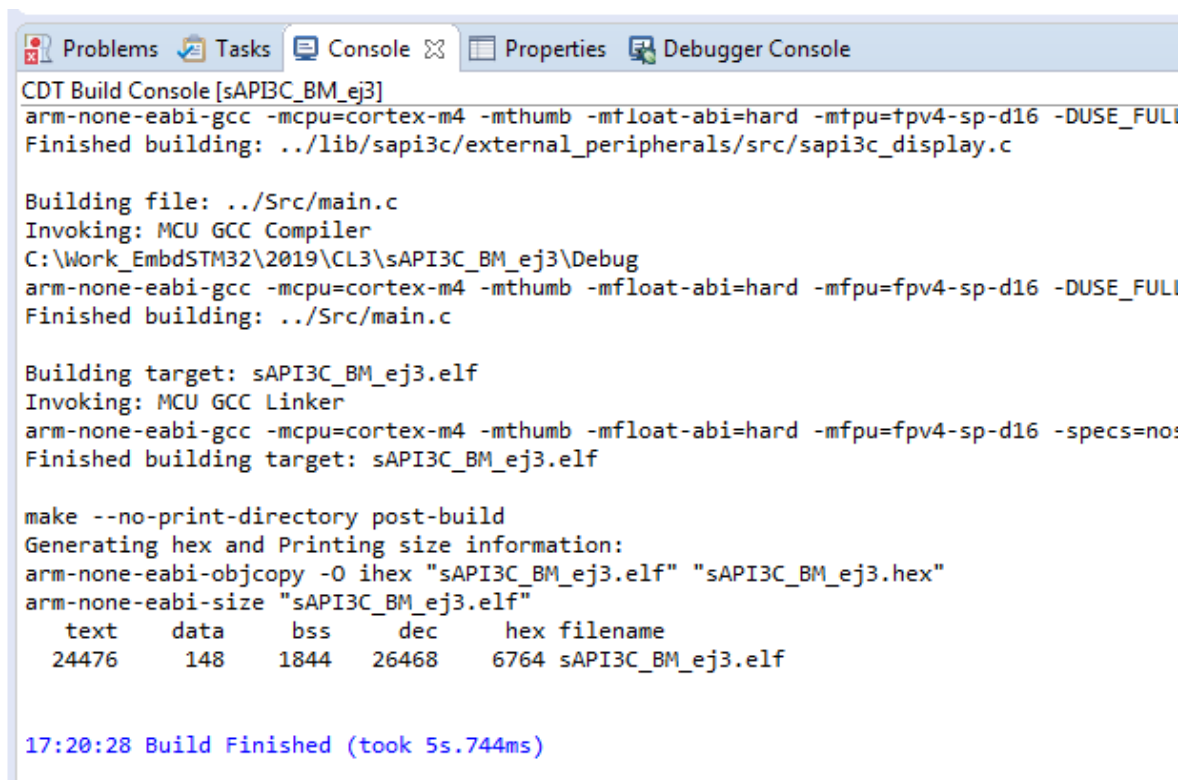
D.2.8.2) las funciones para el display (ver inicio de U_LL6, main.c):

```
#include <stdarg.h>
#include "ili9341.h"
#include "ili9341_touch.h"
#include "fonts.h"
#include "testing.h"
```

D.2.8.3) La antigua función init() en Afiskon, la reemplazamos por esta..

```
/**
 * @brief CL3_ILI9341_init() function replaces init() in Afiskon lib
 *        initializes the Display after resetting it.
 * @param none
 * @retval none
 */
void CL3_ILI9341_init(){
    ILI9341_Unselect();
    ILI9341_TouchUnselect();
    ILI9341_Init();
}
```

D.2.8.3) tuvimos que sacar el #include “testing.h” porque da errores de “redefinición”, aunque sólo se la referencia allí..



```
CDT Build Console [sAPI3C_BM_ej3]
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULLI
Finished building: ../lib/sapi3c/external_peripherals/src/sapi3c_display.c

Building file: ../Src/main.c
Invoking: MCU GCC Compiler
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej3\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULLI
Finished building: ../Src/main.c

Building target: sAPI3C_BM_ej3.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_ej3.elf

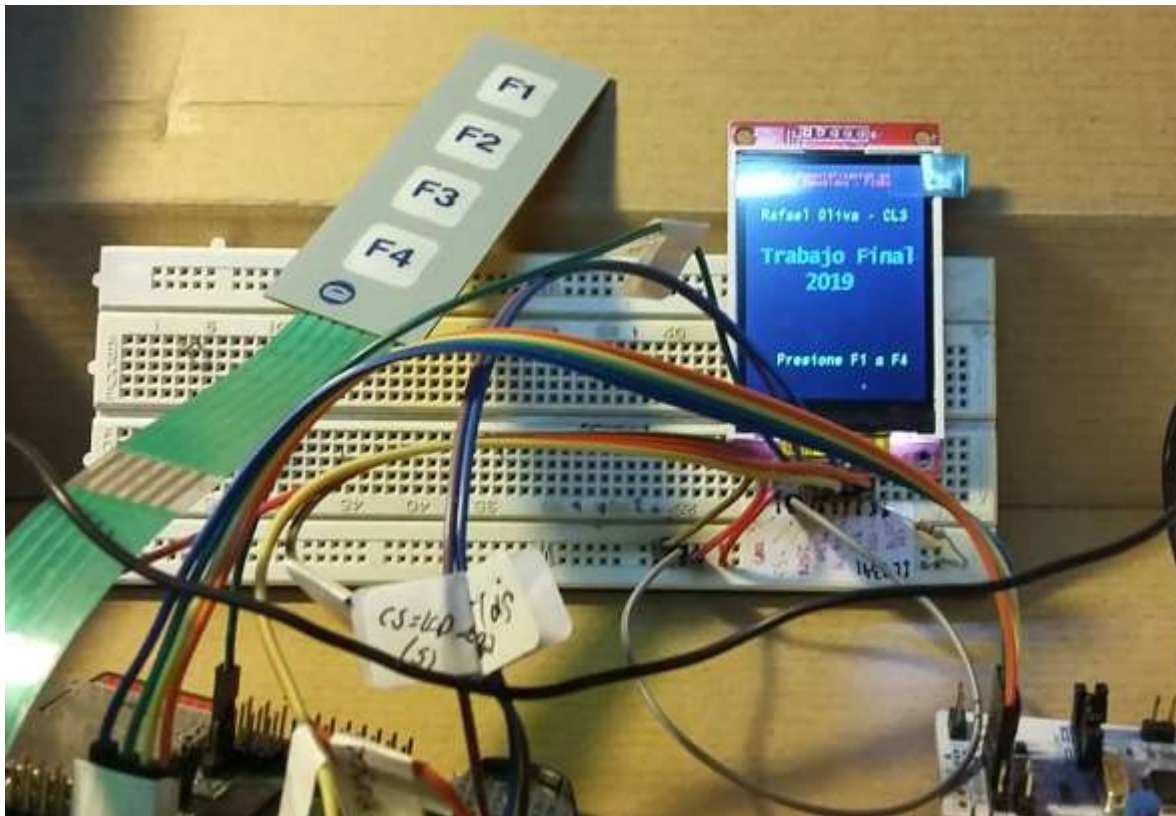
make --no-print-directory post-build
Generating hex and Printing size information:
arm-none-eabi-objcopy -O ihex "sAPI3C_BM_ej3.elf" "sAPI3C_BM_ej3.hex"
arm-none-eabi-size "sAPI3C_BM_ej3.elf"
  text    data    bss     dec     hex filename
 24476    148    1844    26468    6764 sAPI3C_BM_ej3.elf

17:20:28 Build Finished (took 5s.744ms)
```

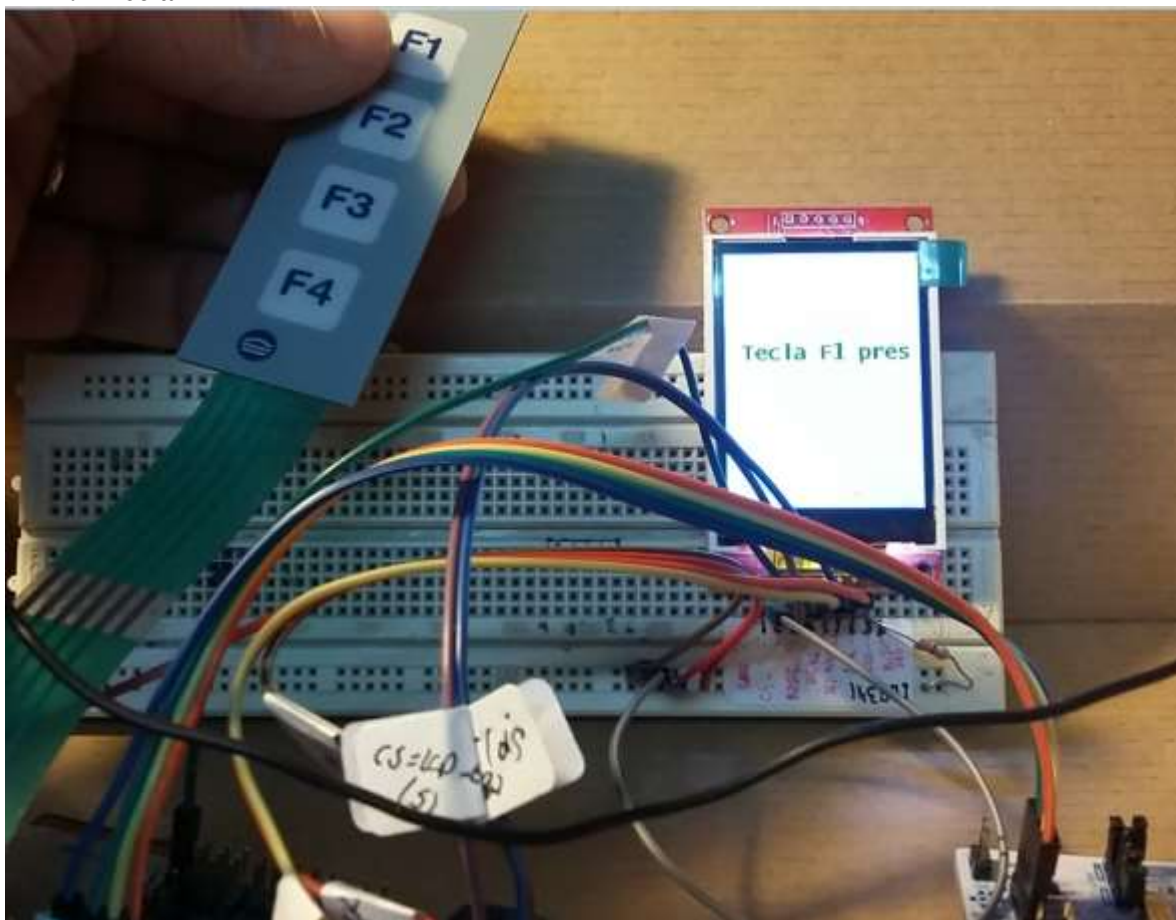
D.2.9) Compila . ✓

D.2.7) Ensayamos: OK! Funciona, aunque hay que corregir la ubicación del mensaje “Presione una Tecla (quedó en primera línea?) – y el reseteo periódico del WelcomeScreen()

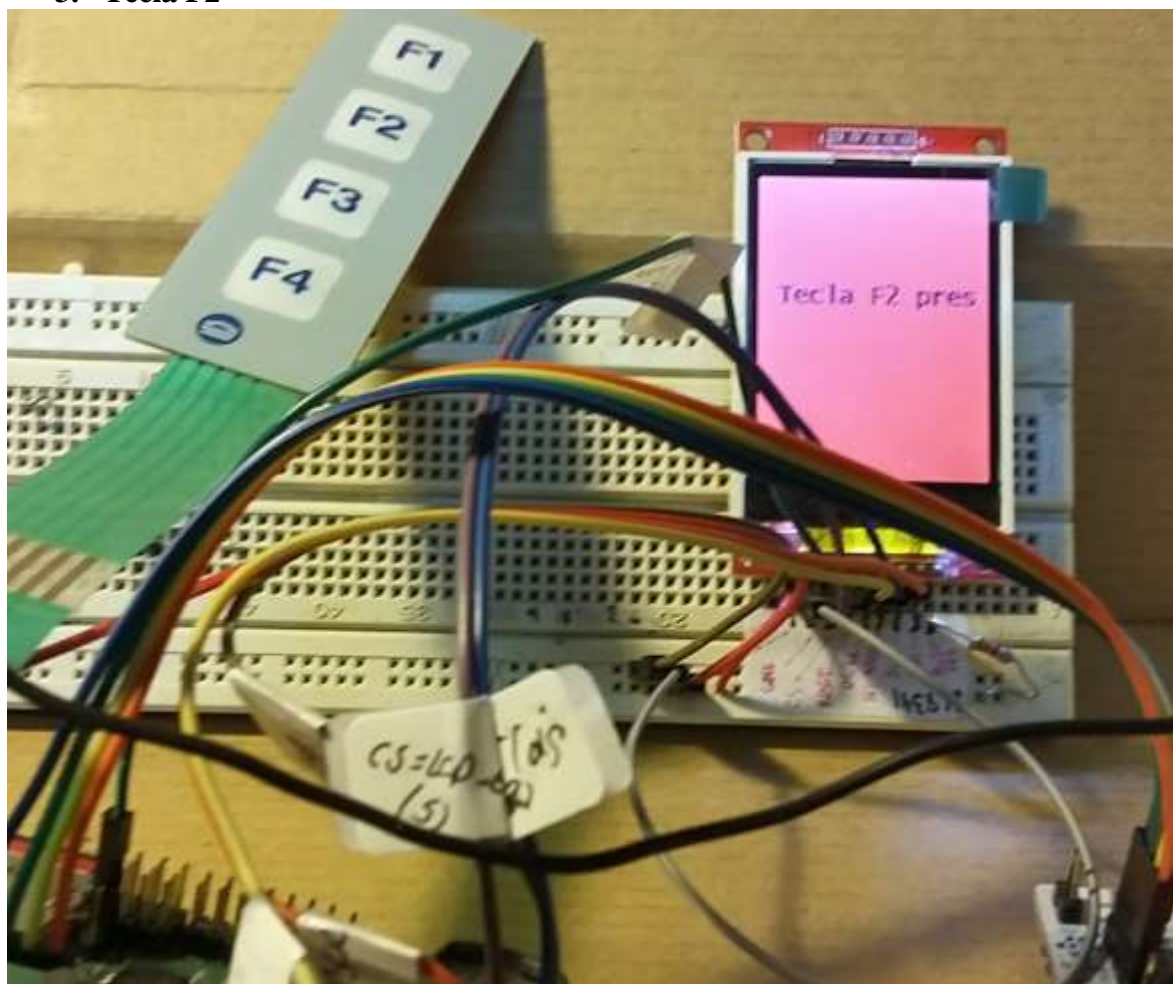
1. PPAL



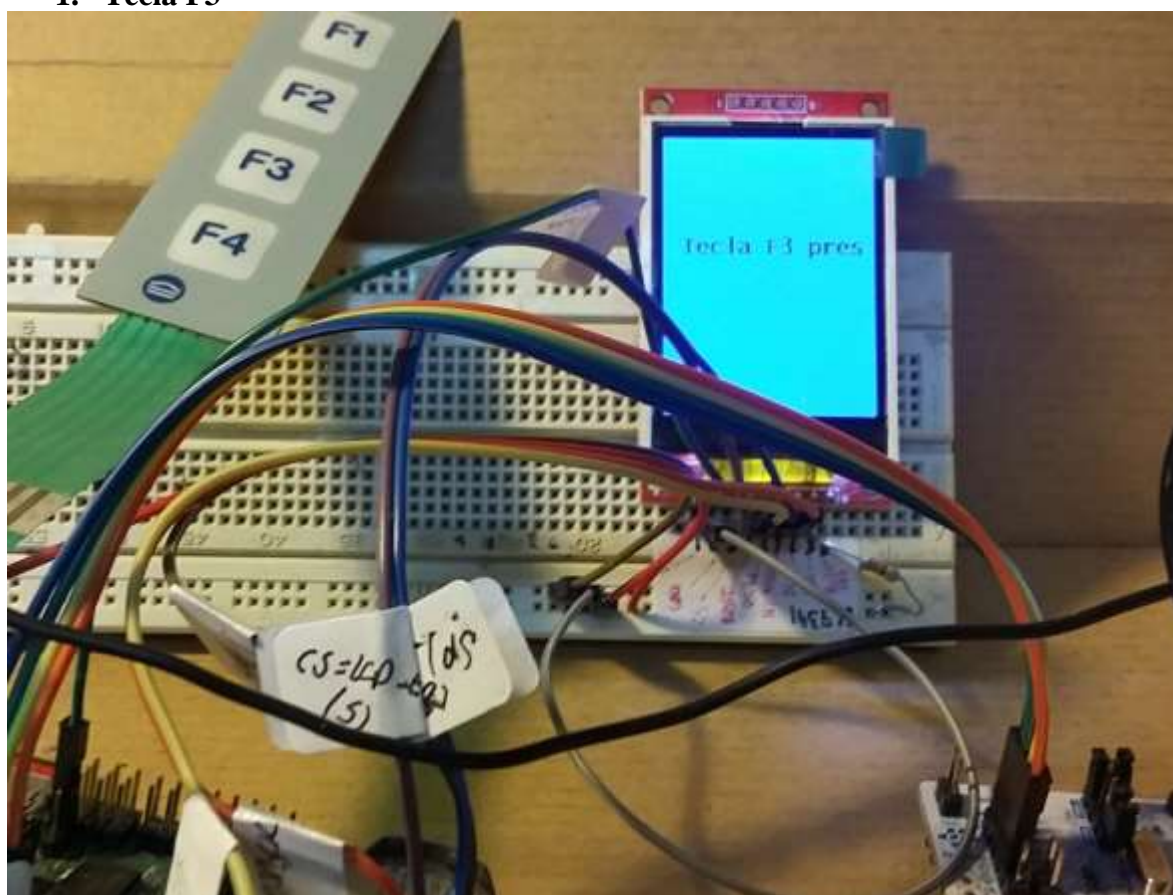
2. Tecla F1



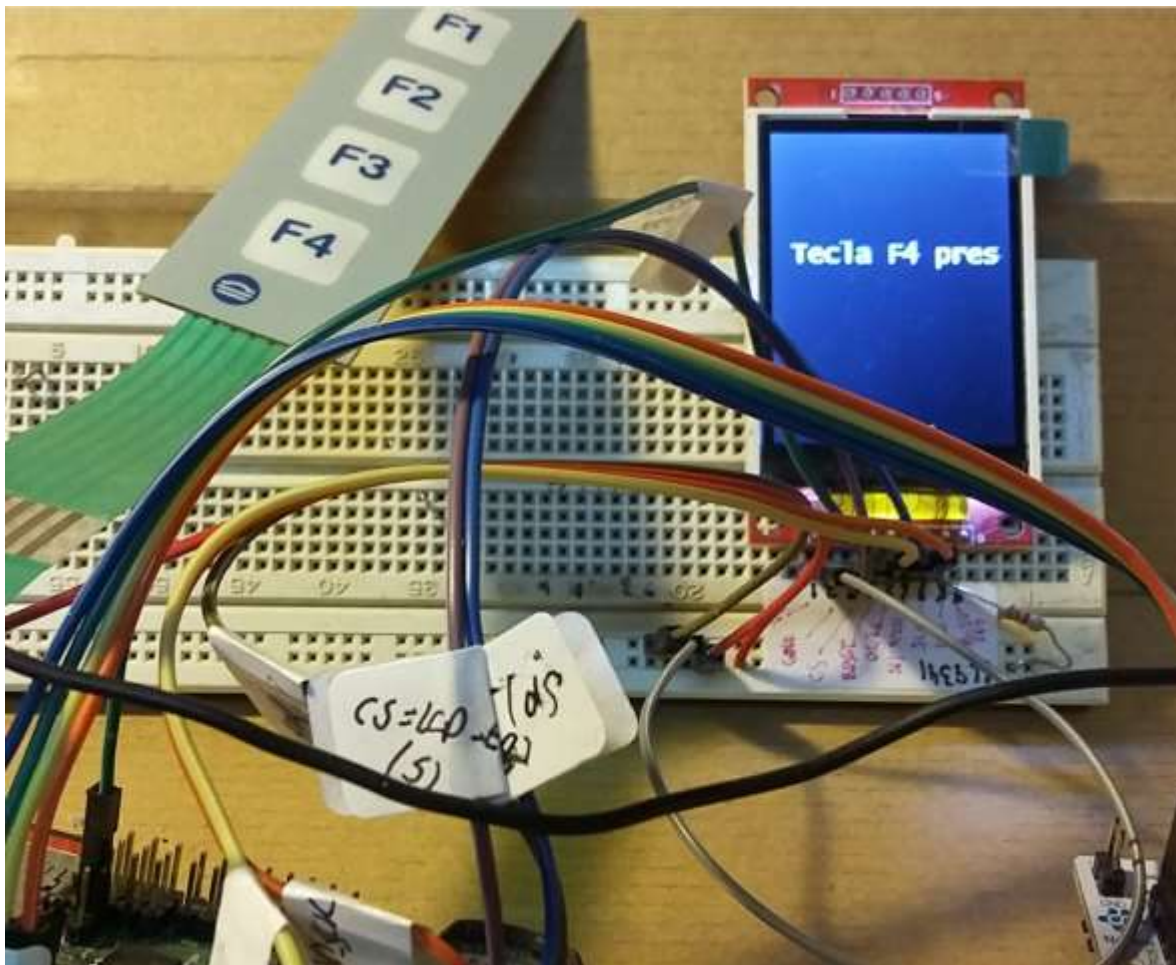
3. Tecla F2



1. Tecla F3



1. Tecla F4

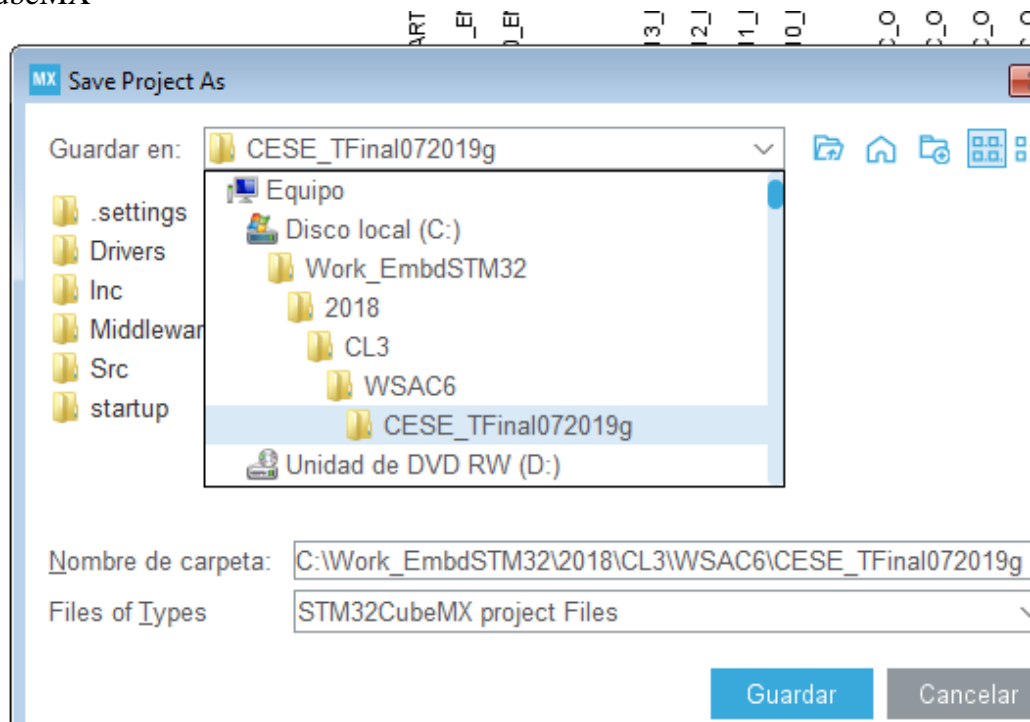


NOTA:

ANEXO AB)

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

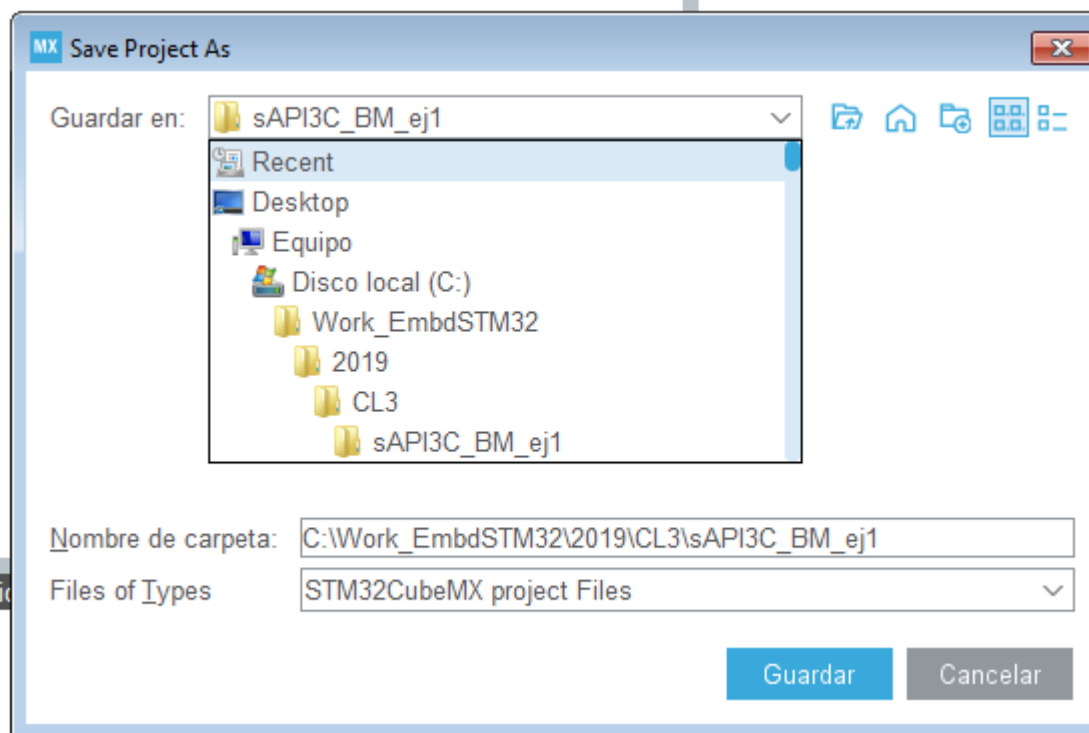
AB.1) Tomamos de versión g 0719 (placa configurada completa)
Desde el CubeMX



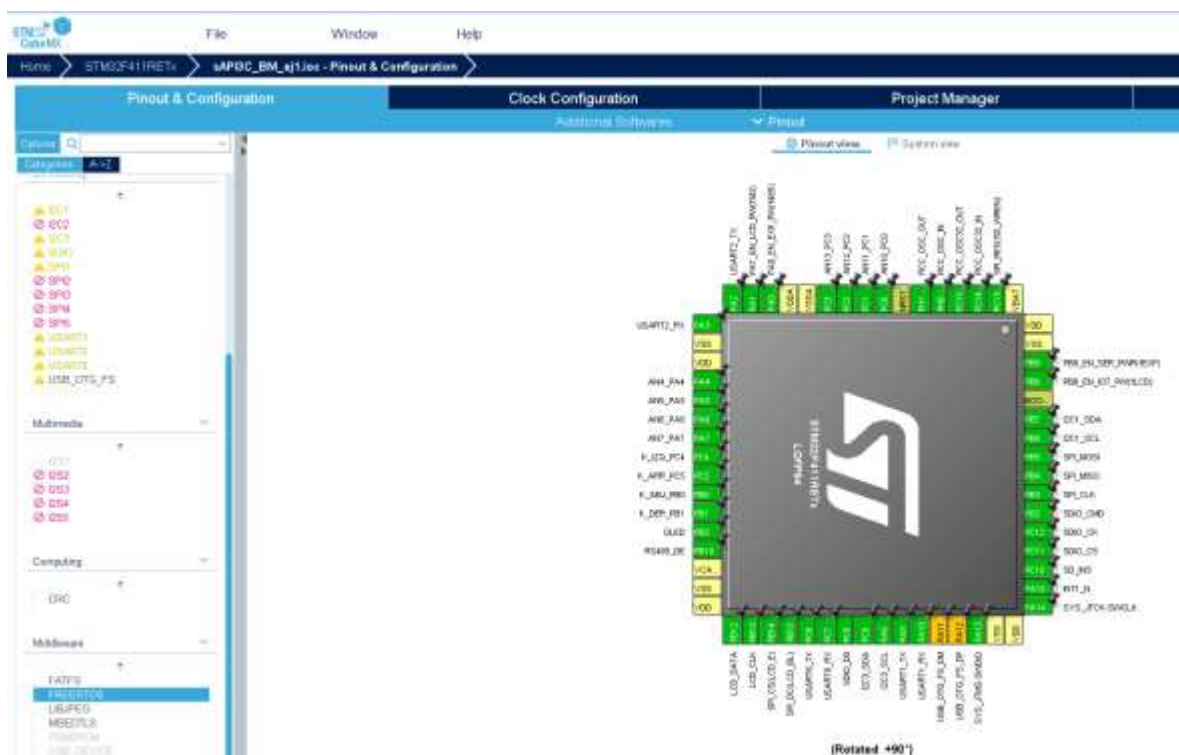
(que es la versión 5.1)



Y la guardamos sin FATFS y sin FreeRTOS como:



Por lo demás, mantenemos el Pinout Fijo como en 0719g:



Con los siguientes Clocks:



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>Settings...</p>	

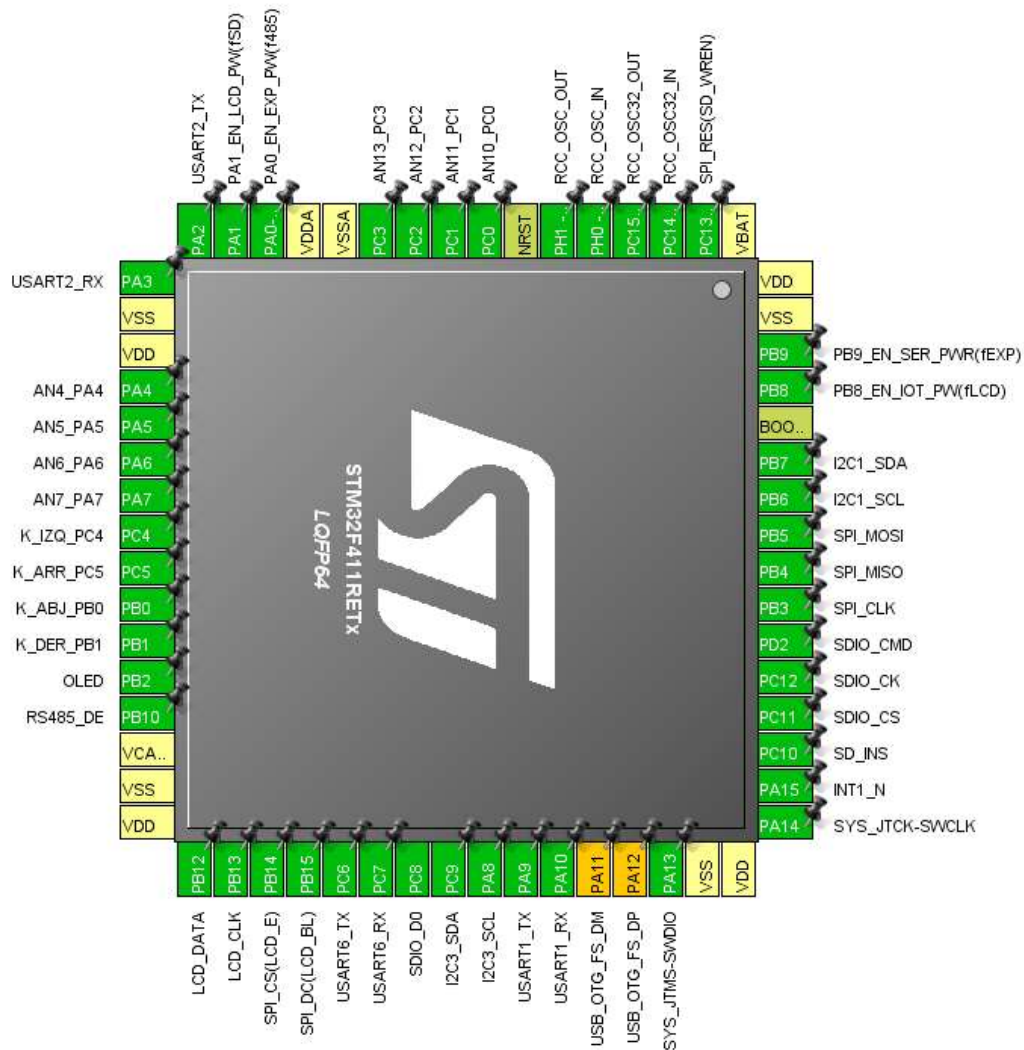
Opción Toolchain es AC6 (SW4STM32) y Firmware v1_24_0

MX STM32CubeMX sAP3C_BM_ej1.ioc*: STM32F411RETx

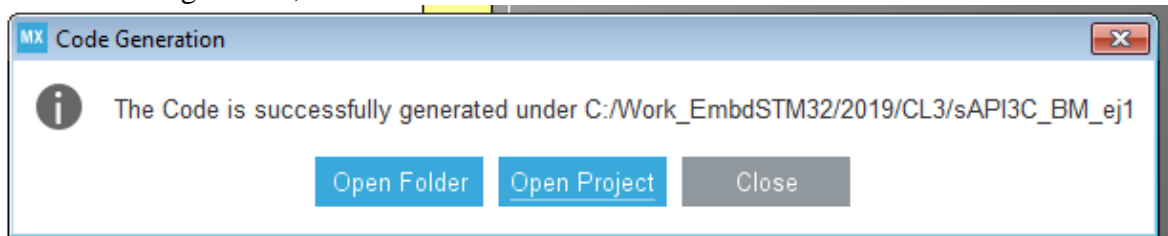
STM32CubeMX File Window Help

Home > STM32F411RETx > sAPI3C_BM_ej1.ioc - Project Manager >

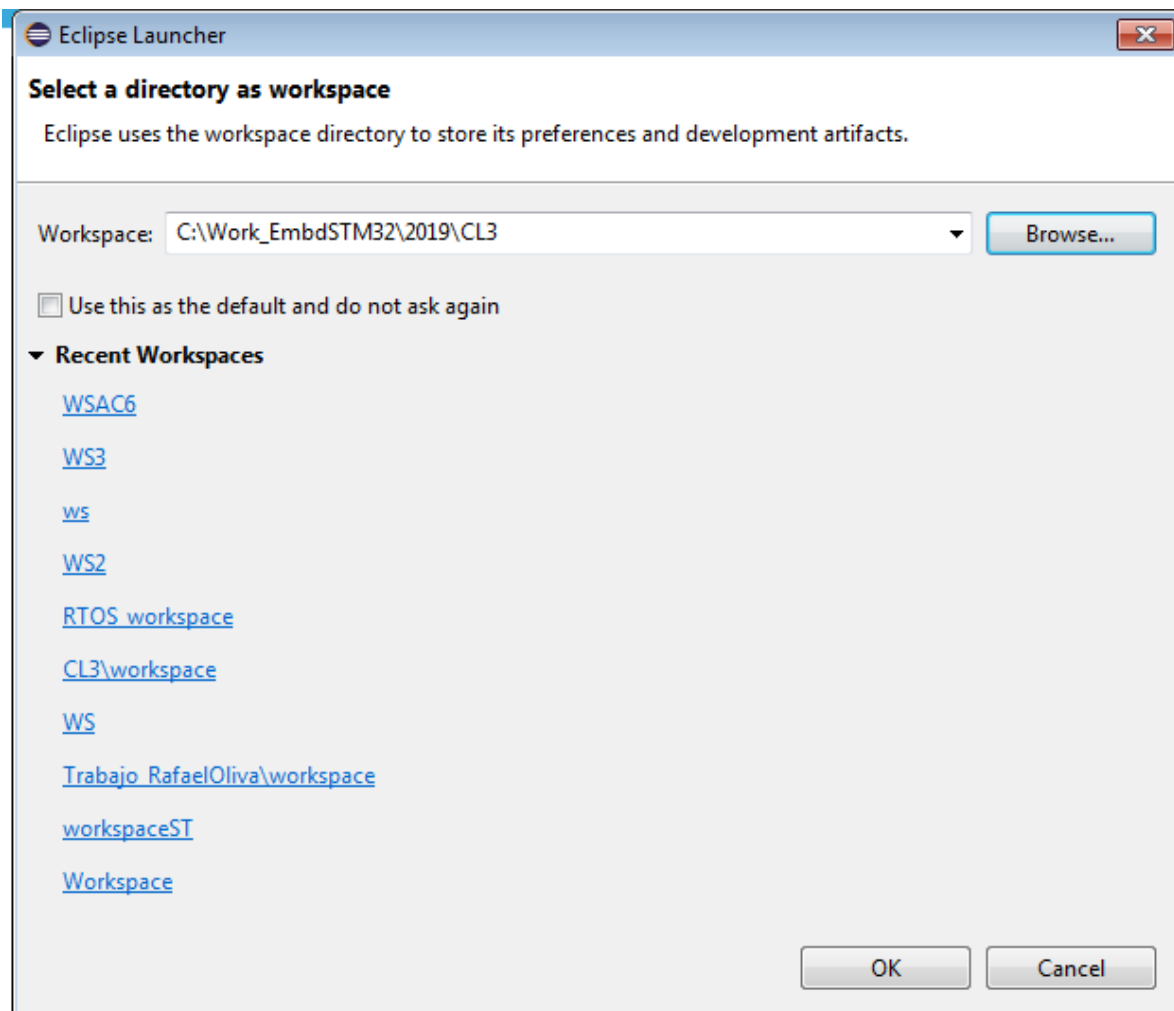
	Pinout & Configuration	Clock Configuration
Project	<p>Project Settings</p> <p>Project Name sAPI3C_BM_ej1</p> <p>Project Location C:\Work_Embd\STM32\2019\CL3</p> <p>Application Structure Basic <input type="checkbox"/> Do not generate the main()</p>	
Code Generator	<p>Toolchain Folder Location C:\Work_Embd\STM32\2019\CL3\sAPI3C_BM_ej1\</p> <p>Toolchain / IDE SW4STM32 <input checked="" type="checkbox"/> Generate Under Root</p>	
Advanced Settings	<p>Linker Settings</p> <p>Minimum Heap Size 0x200</p> <p>Minimum Stack Size 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\Gabriel\STM32Cube\Repository\STM32Cube_FW_F4_V1.24.0 Browse</p>	



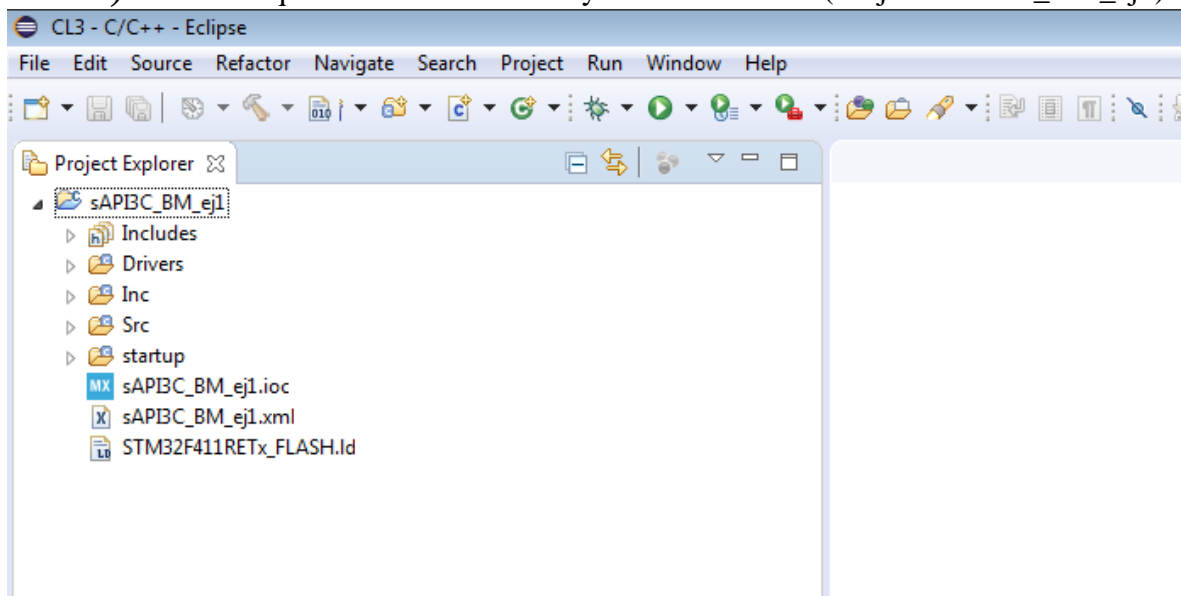
Con esta configuración, damos GenerateCode



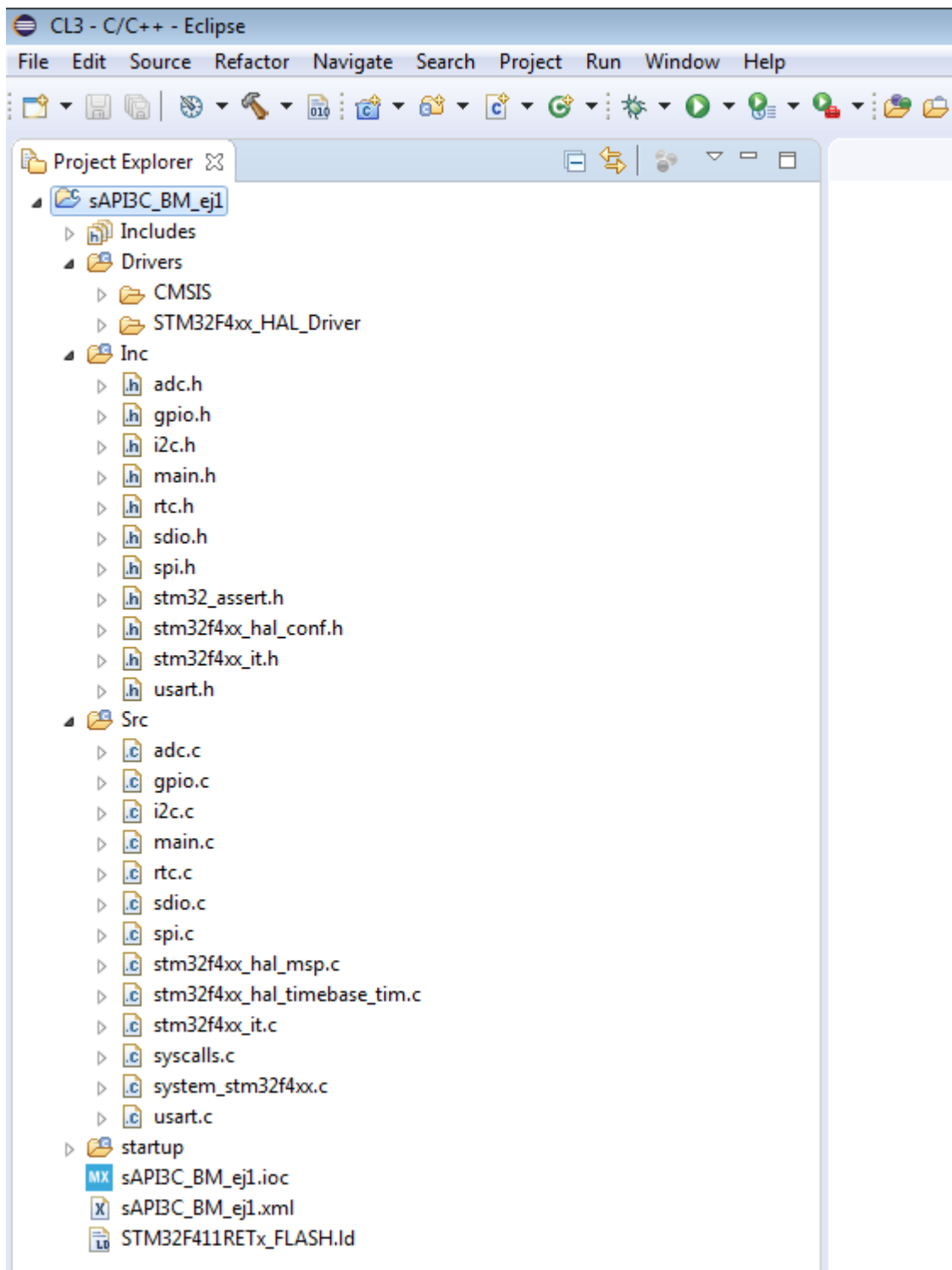
Y aquí damos “OpenProject”, usando el Workspace nuevo 2019/CL3



AB.2) Esto es lo que nos abre en AC6 / System Workbench (Project sAPI3C_BM_ej1):

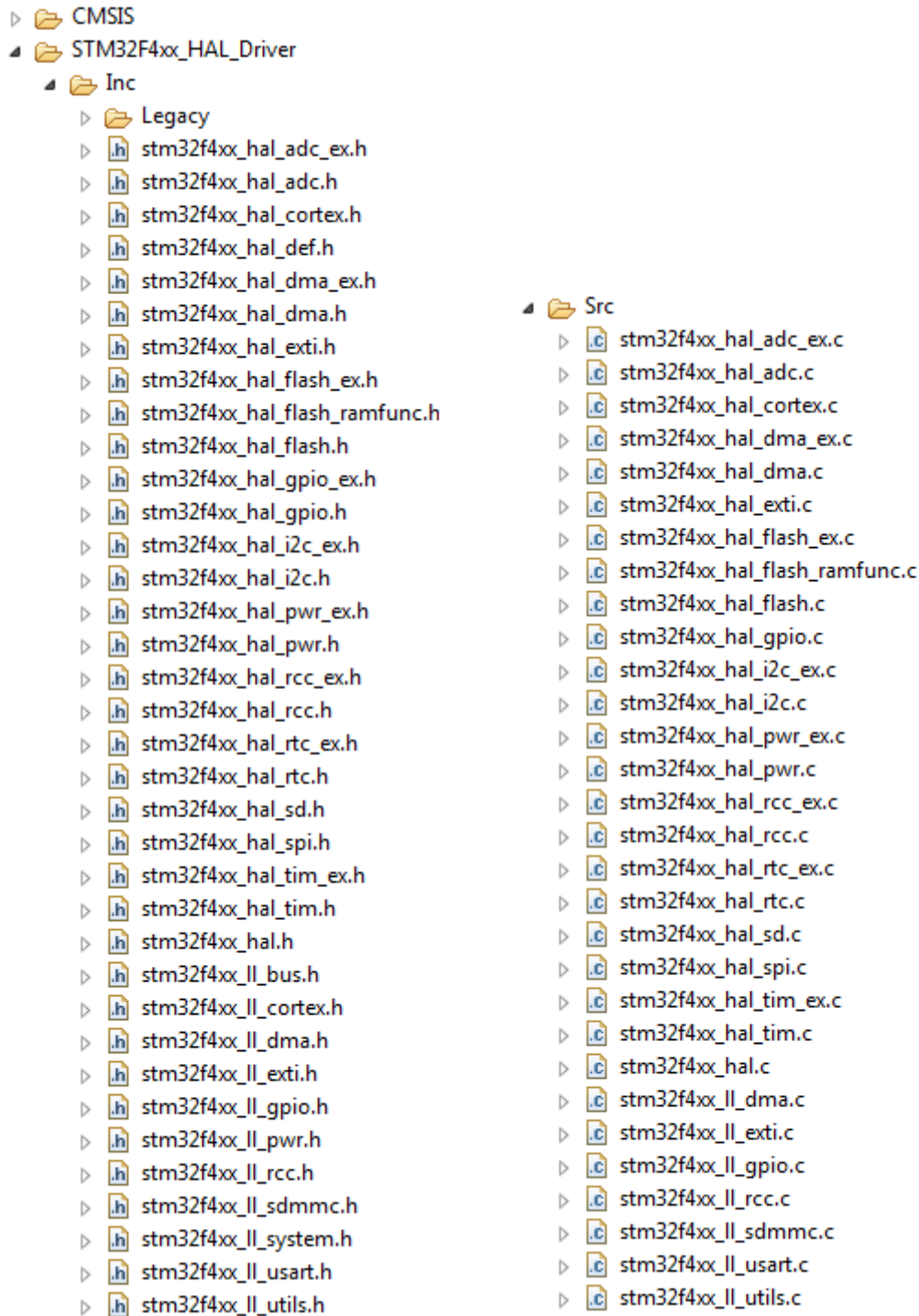


Con la siguiente estructura:



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

Problems Tasks Console Properties

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)

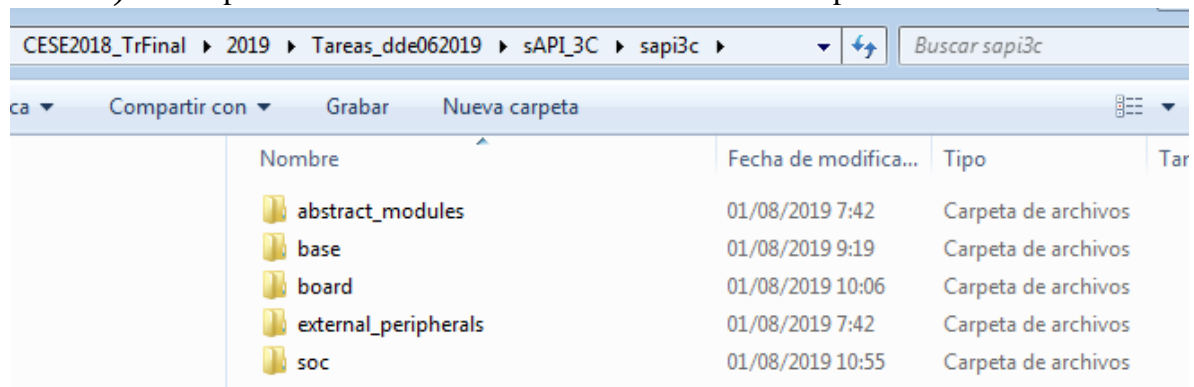
```

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

AB.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);

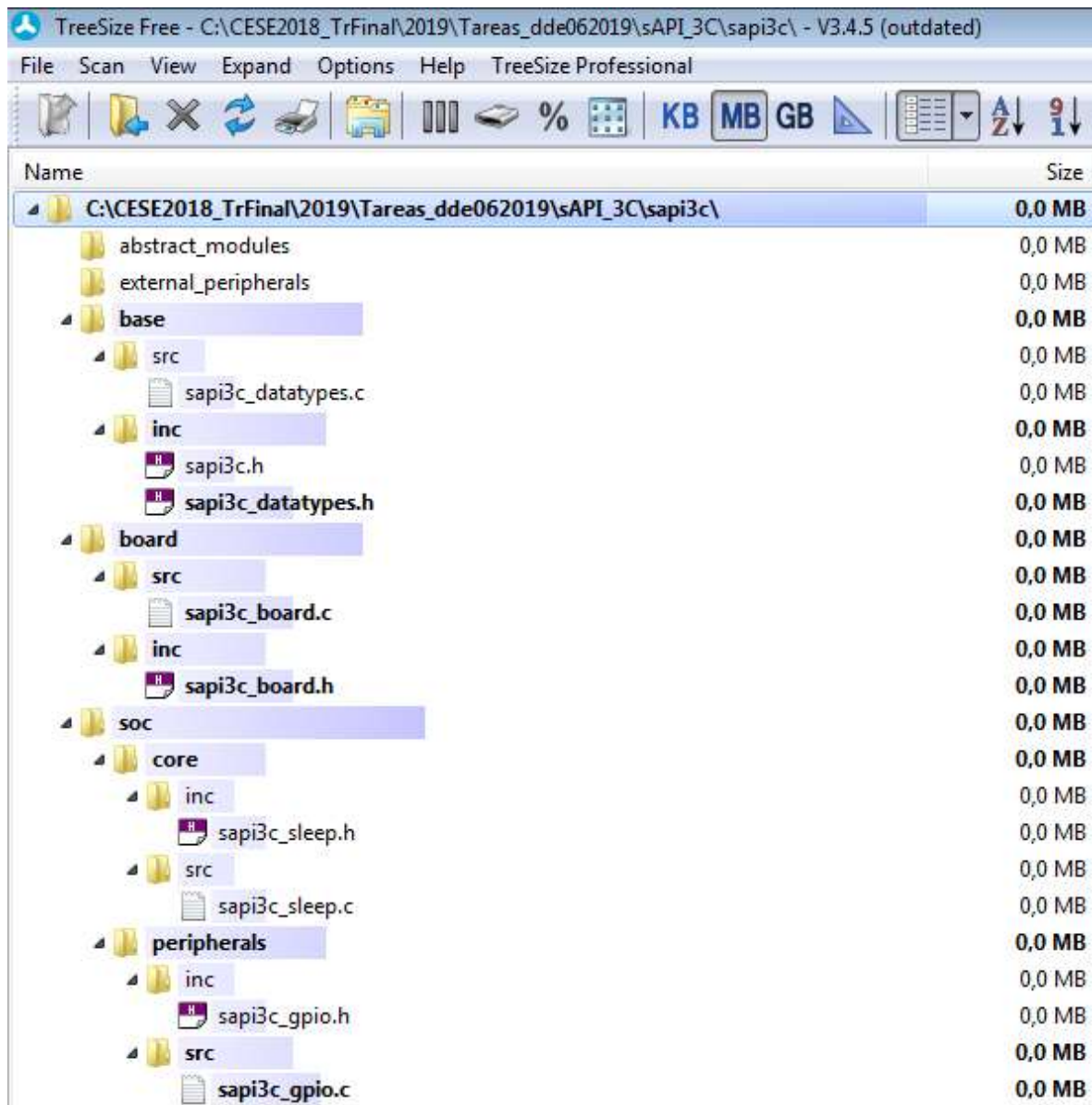
/*=====[cplusplus]=====*/

#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 pane 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

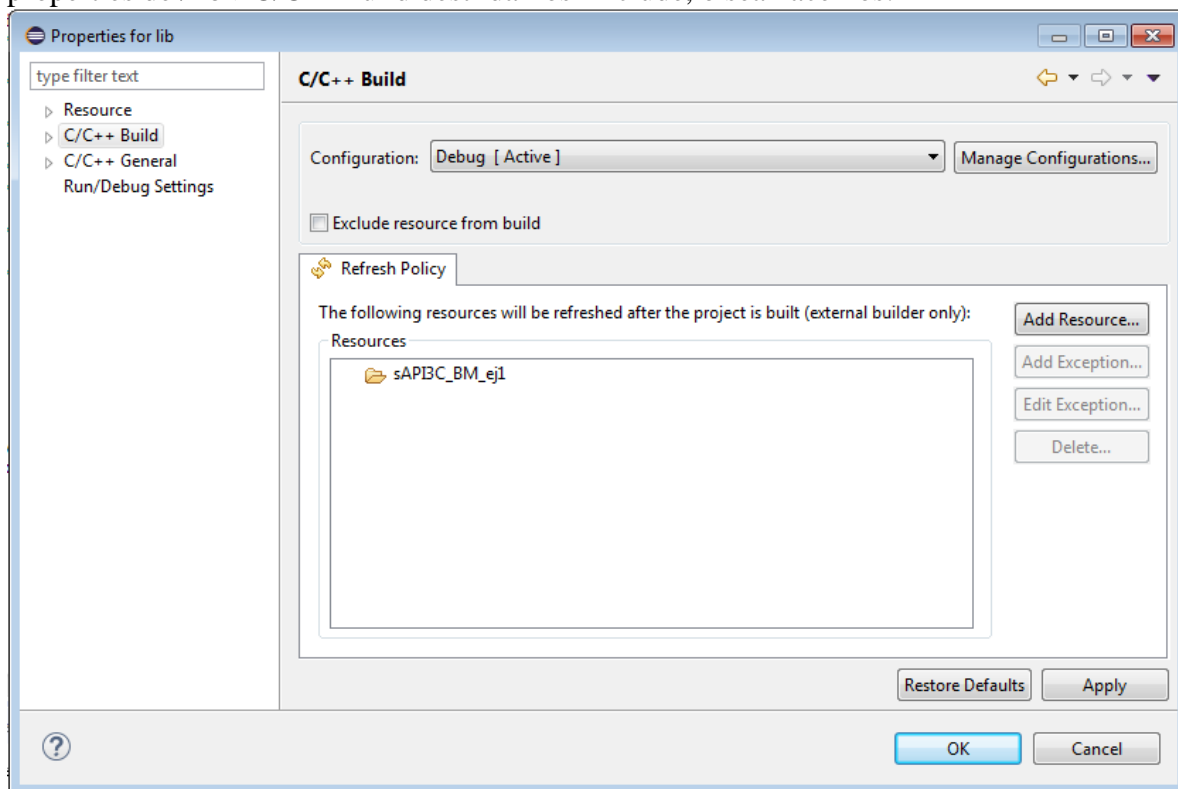
Lo que hacemos, a modo de prueba, es armar un nuevo directorio dentro del Project, sAPI3C_BM_ej1, y agregar esta estructura como /lib/sapi3c

AB.4) Ensayo con nuevo directorio..

- AB.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.
- AB.4.2) Además las definiciones de pines que estaban en main.h ahora estarán en sapi3c_board.h
- AB.4.3) Por ahora, sapi3c_board.c no contendrá inicialización
- AB.4.4) Lazo ppal del main() A.2.2:

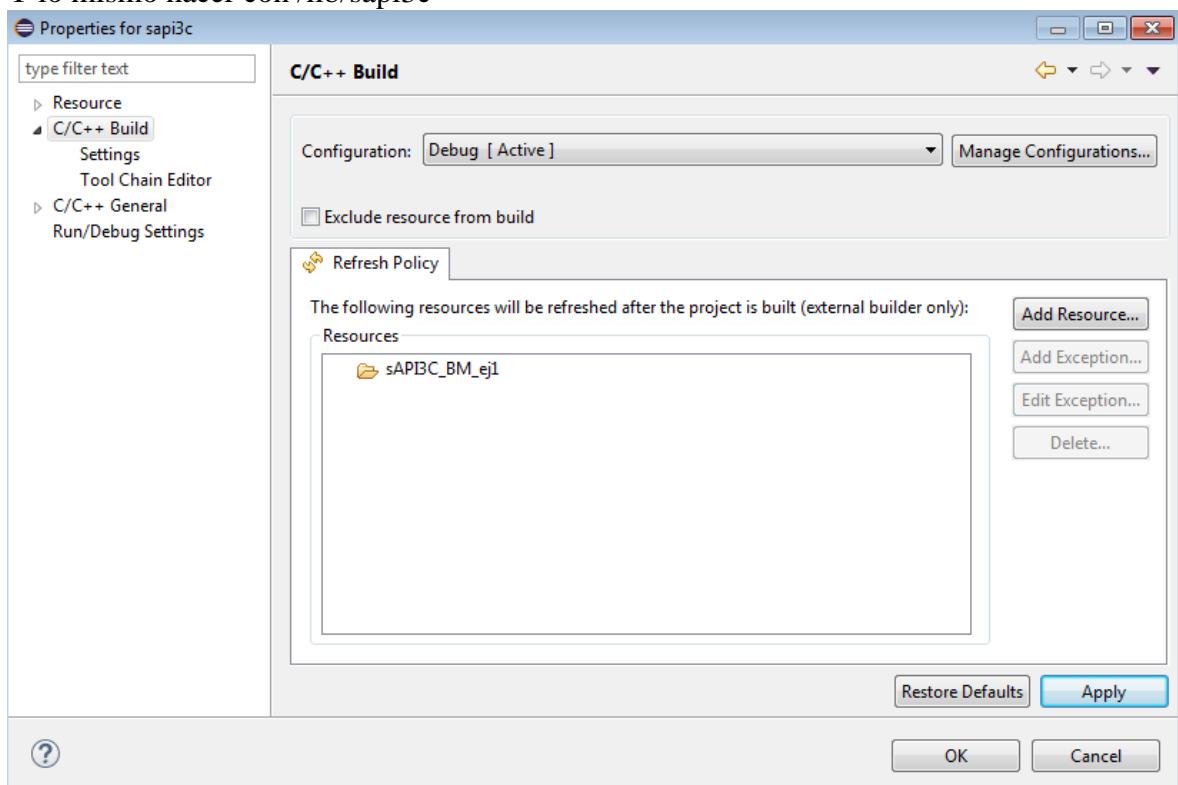
```
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */
    // from stm32f4xx_ll_utils.h /.c
    // void LL_mDelay(uint32_t Delay)
    LL_mDelay(1000);
    LL_GPIO_TogglePin(OLED_GPIO_Port,OLED_Pin);
    /* USER CODE BEGIN 3 */
}
```

AB.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:



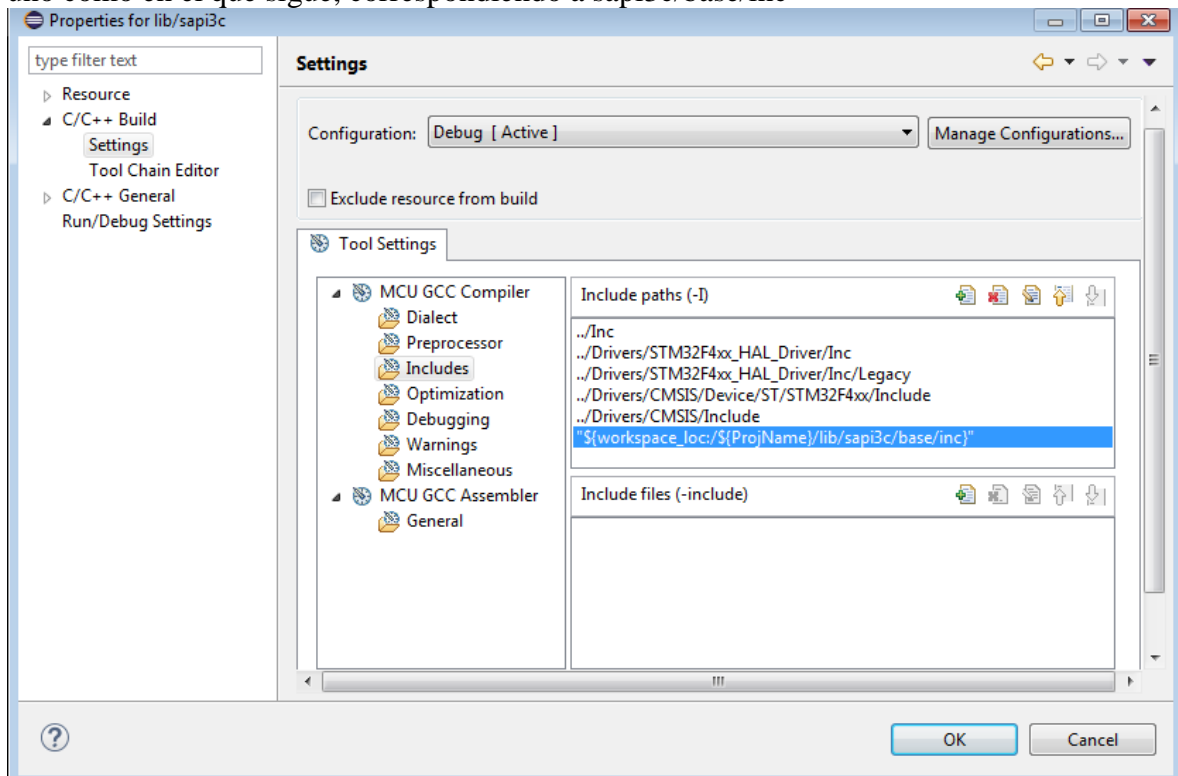
Apply-> OK,

Y lo mismo hacer con /lib/sapi3c

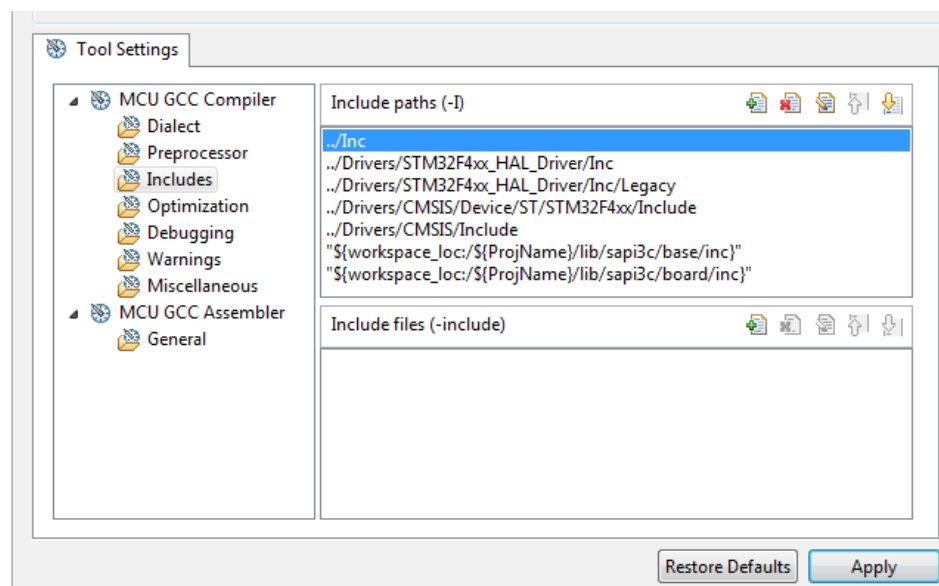


Aparentemente no es necesario con los subdirectorios de sapi3c (no aparecen Excluidos, al mirar sus properties).

AB.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



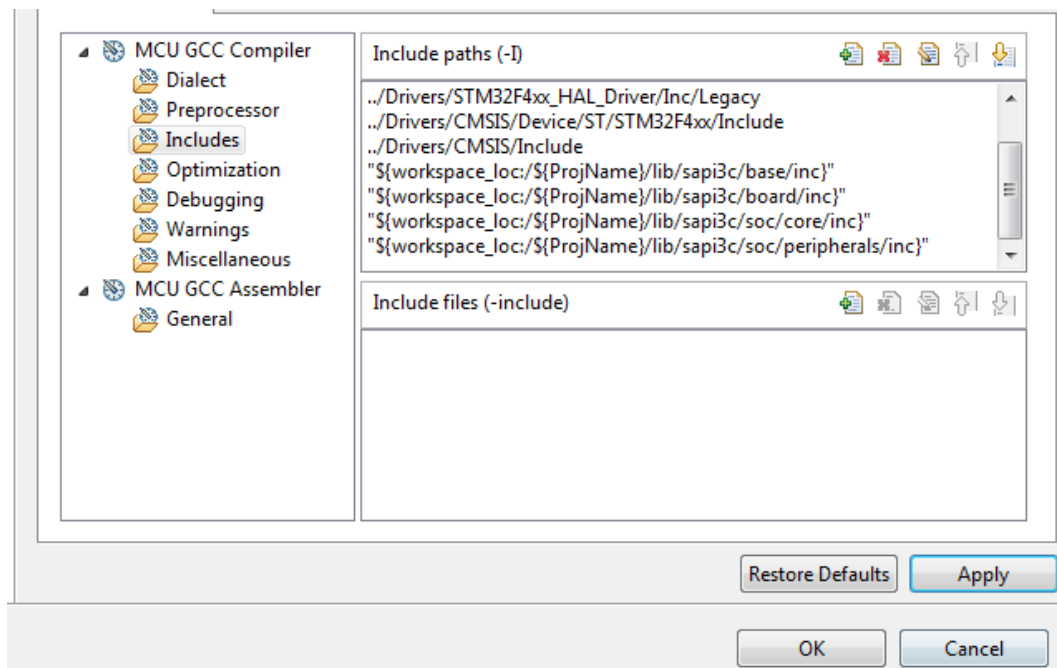
Con /board/inc
Run/Debug Settings



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

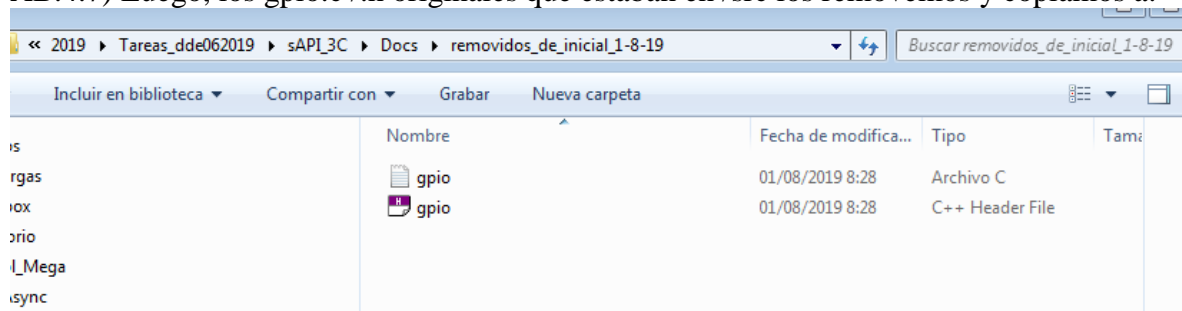
Luego con /sapi3c/soc/core/inc

Y con /sapi3c/soc/peripherals/inc

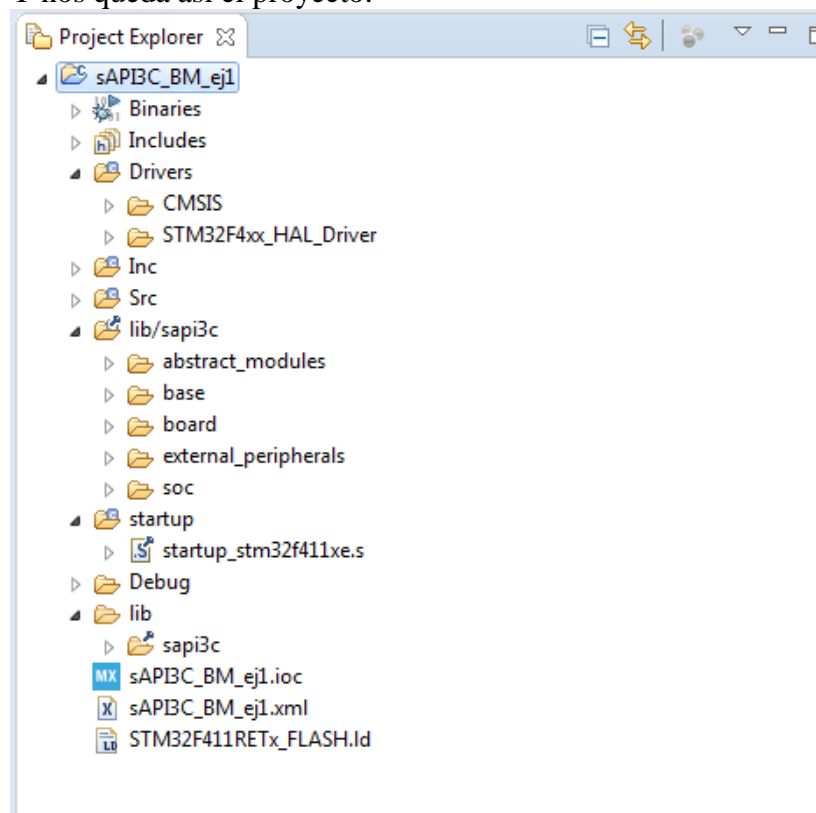


Quedando así..

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



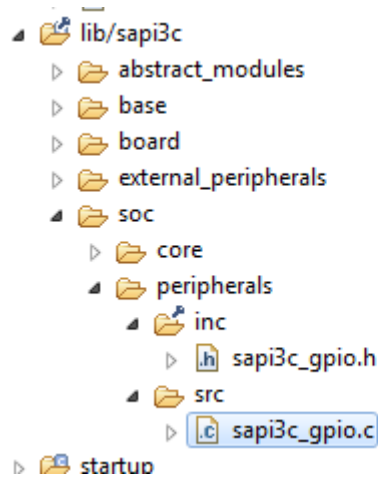
Y nos queda así el proyecto:



Al intentar compilar, nos falta el sapi3c_peripheralmap, en sleep:

```
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ejl\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))
In file included from ../lib/sapi3c/soc/core/src/sapi3c_sleep.c:41:0:
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ejl\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_ejl]
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ejl\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_ejl.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"
Finished building target: sAPI3C_BM_ejl.elf

make --no-print-directory post-build
Generating hex and Printing size information:
arm-none-eabi-objcopy -O ihex "sAPI3C_BM_ejl.elf" "sAPI3C_BM_ejl.hex"
arm-none-eabi-size "sAPI3C_BM_ejl.elf"
text    data    bss     dec     hex filename
12600    28      2124   14744   3998 sAPI3C_BM_ejl.elf

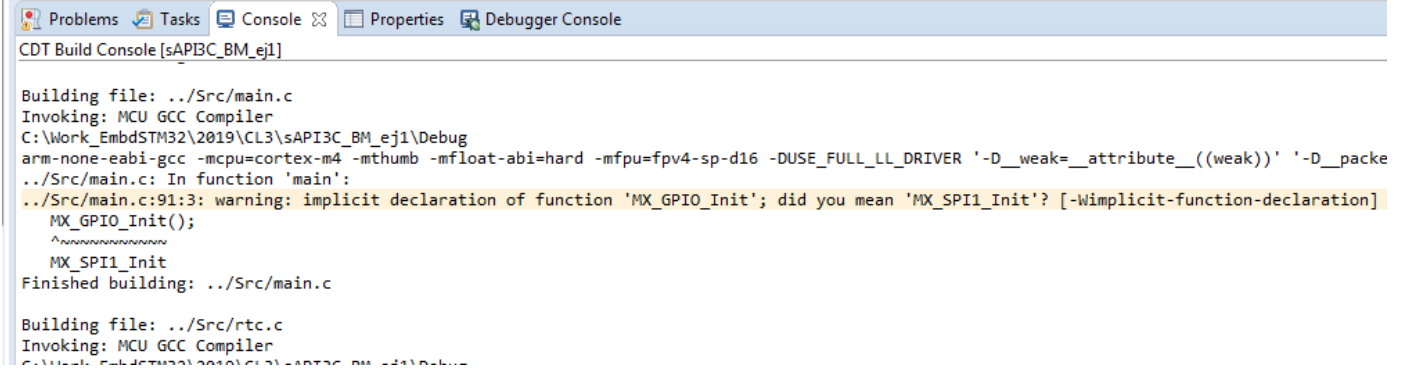
13:47:42 Build Finished (took 3s.344ms)
```

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"

AB.4. 8) PERFECTO!

AB.4. 9) 2.8.2019 – Al intentar recompilar, luego de abrir, me sigue dando el error de que no encuentra MG_GpioInit() al compilar el main..

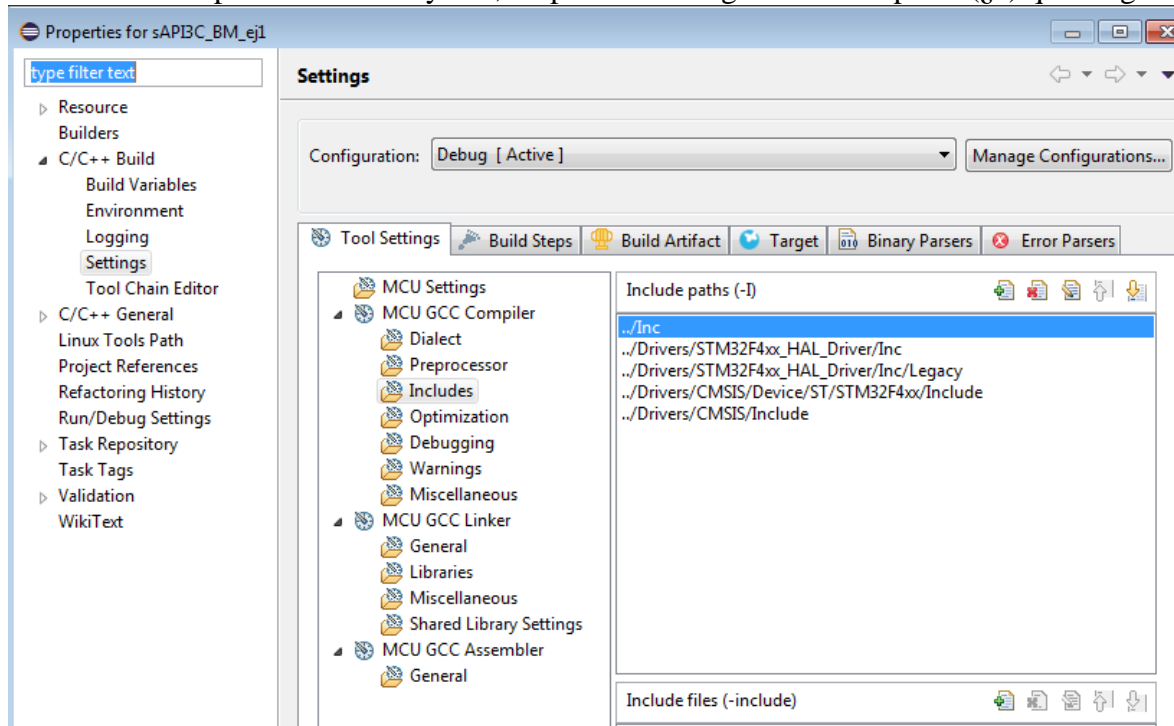


```
Problems Tasks Console Properties Debugger Console
CDT Build Console [sAPI3C_BM_ej1]

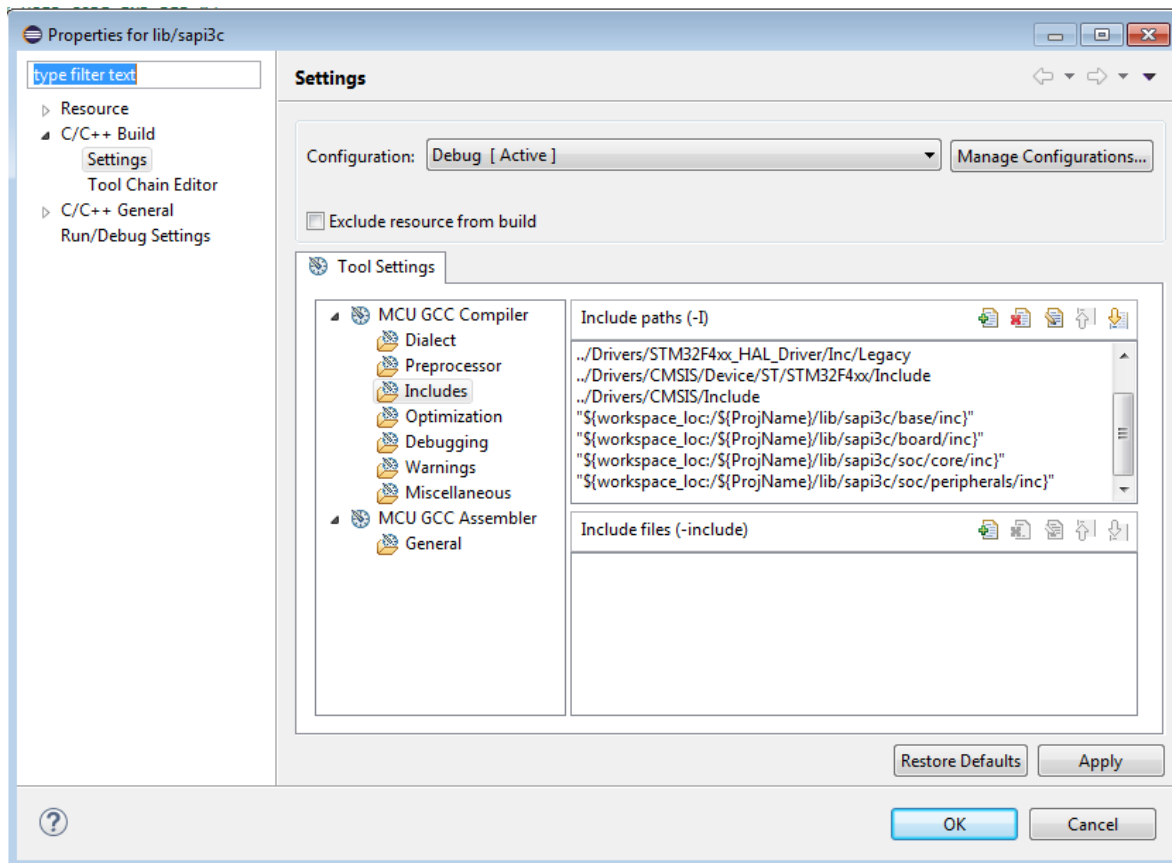
Building file: ../Src/main.c
Invoking: MCU GCC Compiler
C:\Work_EmbdSTM32\2019\CL3\sAPI3C_BM_ej1\Debug
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -DUSE_FULL_LL_DRIVER '-D__weak=__attribute__((weak))' '-D__packe
../Src/main.c: In function 'main':
../Src/main.c:91:3: warning: 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:

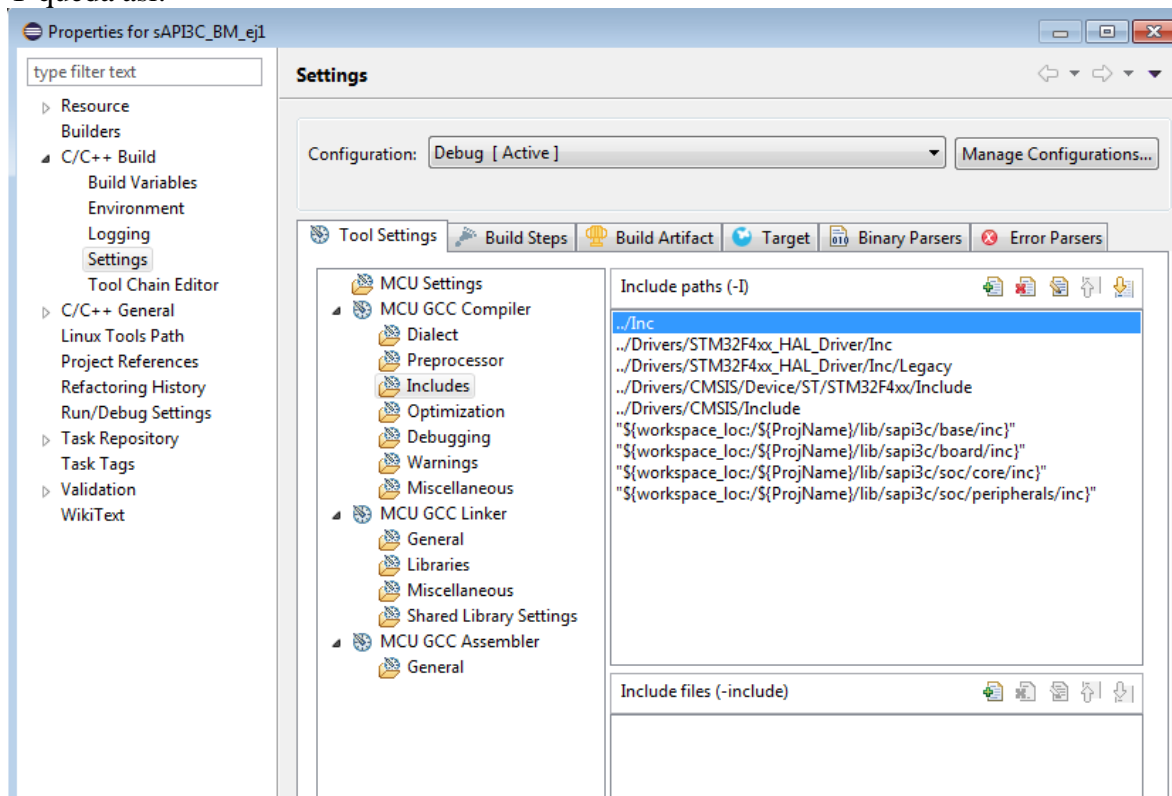


Pero sí están para el directorio que estuvimos trabajando / lib/sapi3c:



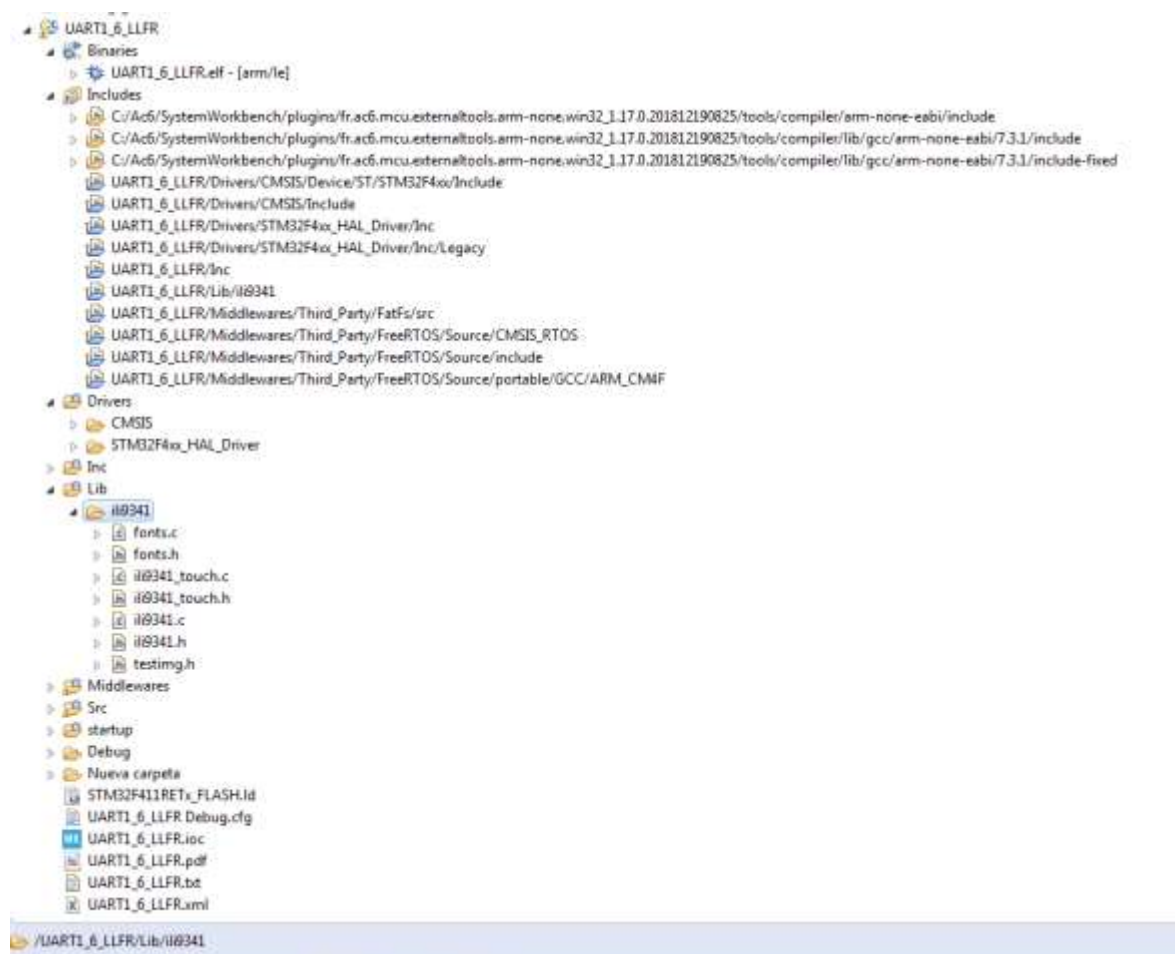
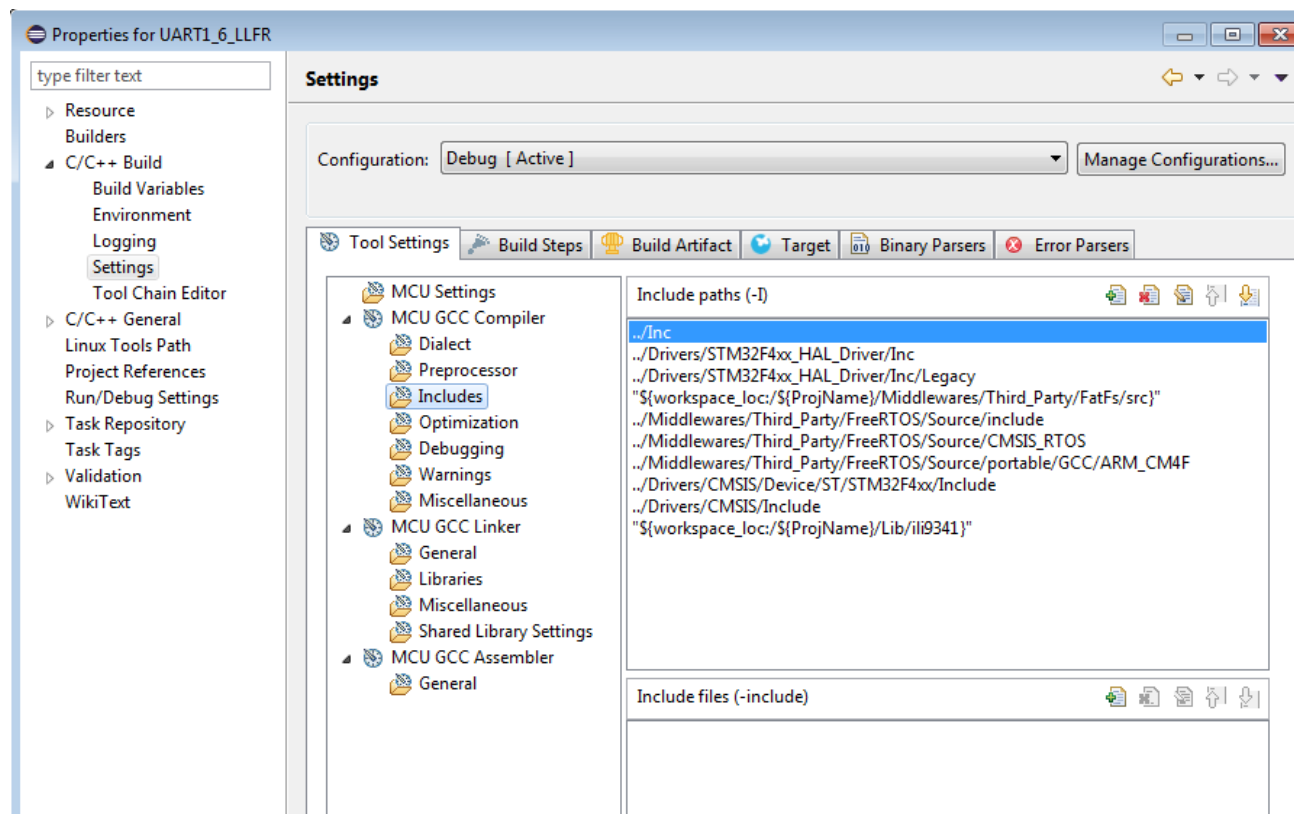
Entonces, lo que aparentemente nos faltó es agregarlo en las propiedades del proyecto completo, es decir pararse en Project ->sAPI32_BM_ej1 y agregar en los C/C++ build settings -> includes, los mismos paths que van dentro de /lib/sapi3c, etc..

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



AB.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));
}

```

APÉNDICE A.E) NUEVOS PROYECTOS

Un tema a tener en cuenta es lo que se mencionaba en el curso Udemy/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\Udemy\Admin\FreeRTOS\Printouts\CreacionProjectNvoResumido(16-03-2019).pdf

A.E.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.