Programming procedure and examples (i) 2022
L&R Ing. CL3 CPU Board

L&R Ingeniería – Rev. 2b 06-22 - Rafael Oliva

1. INTRODUCTION

The CL3 semi-industrial board module from L&R Ing. (Figure 1) is an STM32 ARM-Cortex M4/F based unit, integrating the STM32F411RET (LQFP64 package) [ref00] microcontroller. It can be programmed in C or C++ using the STM32CubeIDE [ref01], which deploys the gcc arm-none-eabi cross-compiler toolchain [ref02]. In-circuit programming is possible through a standard connector for STLink/v2 or compatible USB programmers [ref03]. The board is slightly smaller than its 8-bit CL2 predecessor [ref04], but shares many features such as a TCXO DS32kHz chip to keep accurate timing on the internal Real Time Clock IC, a compatible mechanical layout and the industrial SD card interface. A TPS54302 switching regulator supplies power for the circuit from an external source between 7 and 20 Vdc, and is coupled to a TPS2085 Power Management IC to reduce power consumption by turning off unused peripherals. Consumption is typically 0.03 A @ 12.8 V. A complete schematic diagram can be found in: https://github.com/LyRIng/CL3board/blob/master/doc/cl3schem/Cl3_Schem.pdf [ref05]

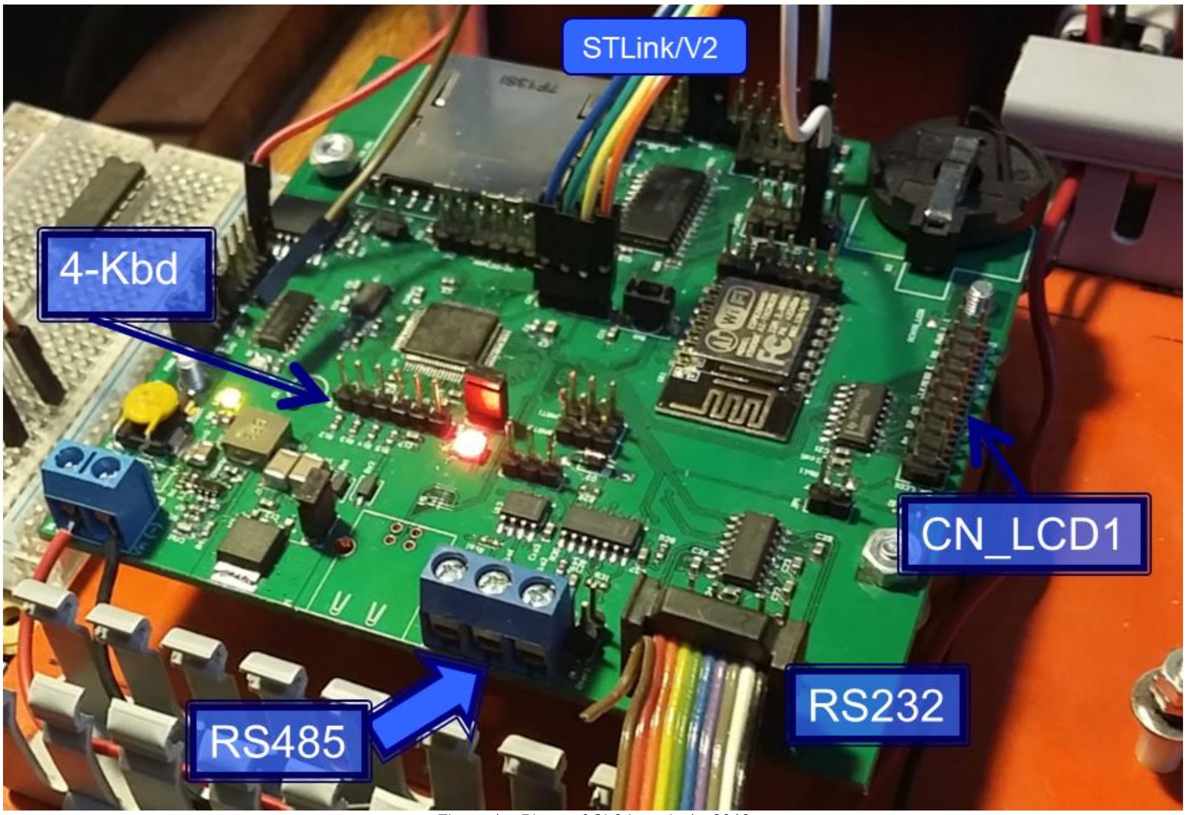


Figure 1 – Photo of CL3 board v1 - 2018

2. BLOCK DIAGRAM AND PROGRAMMING MODEL

2.a Block Diagram: L&R Ing. CL3 boards were designed from 2017 to 2018, and integrated into applications mainly for data logging and other control functions, usually teamed with peripheral boards such as the METEO board/module [ref06]. The block diagram of the board can be seen in Figure 2.

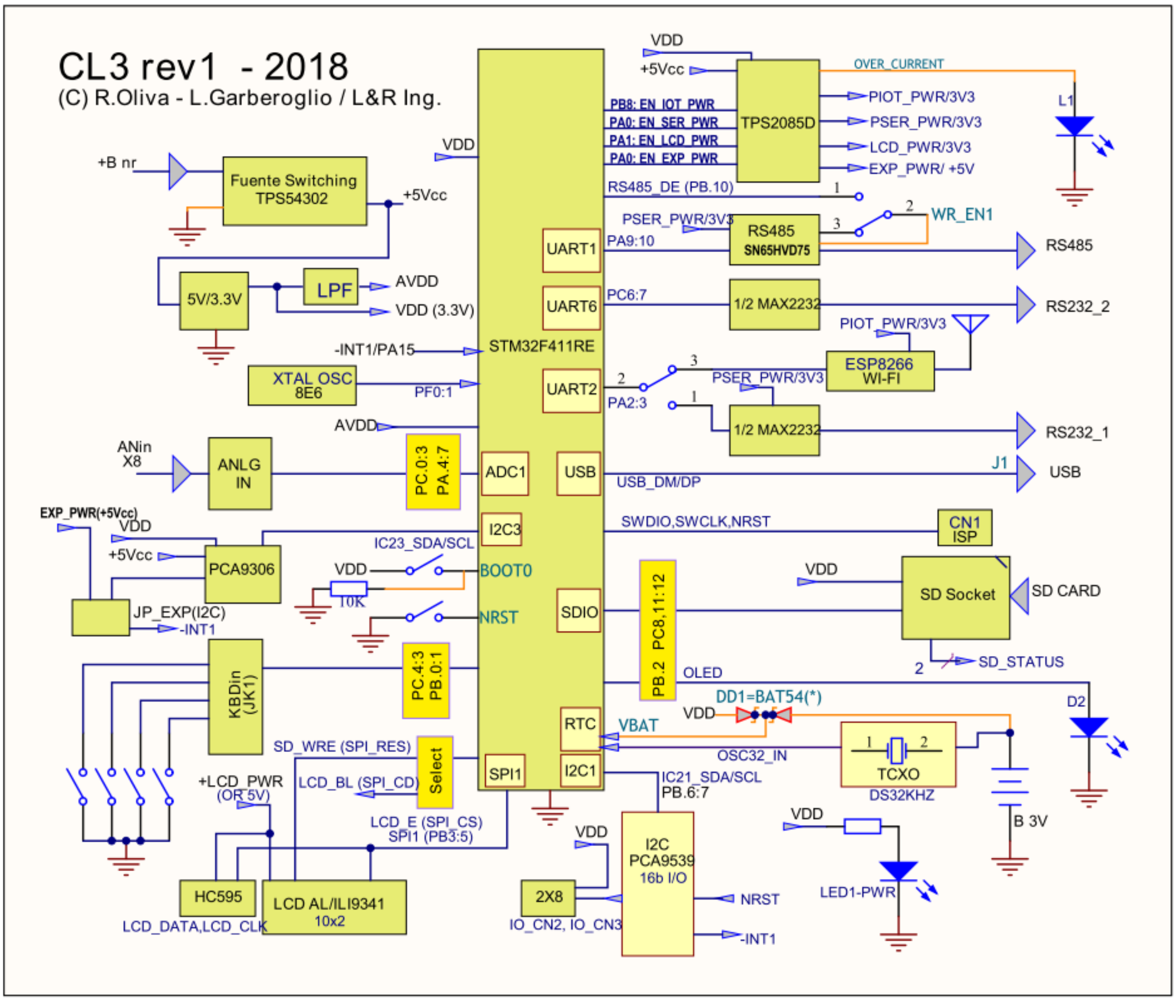


Figure 2 – Block diagram of the CL3 board

The boards are fitted with the STM32F411RET (ARM-Cortex M4/F core) controller, with 512 KB of Flash, 128 KB of RAM, three serial ports, running internally at 100 MHz from an external 8 MHz crystal oscillator. Their main components are:

a) A switching power supply with TPS54302 regulator for +5 V, an LDO LM117 for 3.3V, and a TPS2085 power manager IC. The board can be powered with up to 20 Vdc input.
b) 8 Analog inputs for onboard 12-bit ADC (0-3.3V), filtered AVCC supply.
c) 4 contact membrane keyboard interface
d) An LCD interface, can be adapted to alphanumeric or graphical ILI9341 units.
e) An I^2C (I^2C_3) expansion port with 3.3 to 5V level shifter.
f) A Lithium battery-backed Real Time Clock with DS32kHz TCXO.
f) An Industrial SD card interface
g) 3 serial ports (UART6 as a main terminal with RS232, UART1 for RS485 and UART2 which can be switched to a native RS232 interface or to an ESP12-F (ESP8266 based) WiFi module.
h) A USB interface.
h) A PCA9539 16-bit I/O port on a separate I^2C_1 bus port.

One of the I^2C bus ports communications is used to communicate with auxiliary boards based on PSoC-1 family of microcontrollers. The most common board is the already mentioned M4/E. A separate I^2C port also offers an I/O interface using the PCA9539 device.

2.b Programming model: The STM32 controller on CL3 can be programmed in C or C++ using compatible cross-compilers. The examples in Section 3 will use either STM32CubeIDE [ref01], available for Windows, MacOS and Linux, or the older STM32 AC6 System Workbench [ref07] with integrated debuggers. To ease end-user development, a higher-level sAPI-3C firmware library written in C (Section 3) is offered and integrated into the example projects, the simplest ones using bare-metal infinite loops and a more complex example running FreeRTOS, in communication with an external METEO peripheral. Programs are cross-compiled on a standard PC and downloaded to the board via a standard STLink/v2 4-pin port using a low cost USB STLink/v2 interface [ref03]. This interface also serves as OpenOCD debugger to ease developer work. ST thus offers a completely free toolchain using the arm-gcc cross compiler / linker and associated tools integrated in a user-friendly IDE.

The CL3 internal peripheral and basic modules are routed and preconfigured in the examples using the CubeMX32 [ref08] graphical tool provided by ST but can be user-altered if required. In Figure 3 the layout of pins and peripherals for the CL3 board is shown. The CubeMX32 is provided as a separate tool but was recently (2019) integrated within the STM32CubeIDE. The configuration files (.ioc) are part of the supplied programming example projects and are adapted to the CL3 schematic of [ref03].

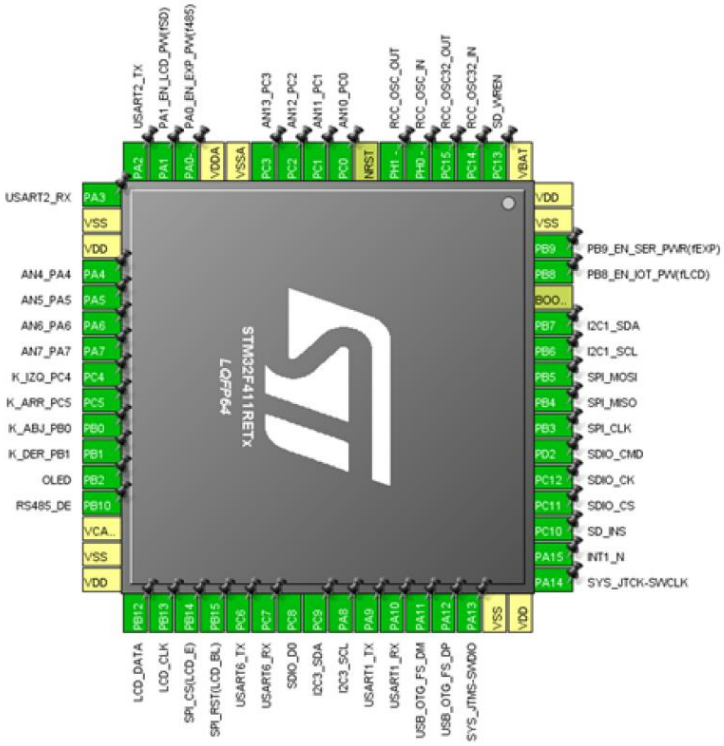


Figure 3 – Low-level configuration of the STM32F411RET unit using CubeMX32.

The complexity of the ARM M4/F microcontroller core (licensed by ST and other vendors) and its peripherals (vendor-specific) requires a series of firmware layers to deploy embedded applications. A generic distribution of these layers for the CL3 boards is shown in Figure 4. The upper application layer is user-defined and the intermediate layer called sAPI-3C -inspired in the original sAPI (*simplified Application Program Interface*, by E. Pernia [ref09] written for the CIAA project [ref10]) - provides a number of modules which solve many typical requirements of embedded applications. In the lower levels, directly on top of the hardware are the standard CMSIS layers produced by ARM, and some of the *Low Level* (LL_) driver libraries produced by ST for the STM32 series of microcontrollers. Some of the more complex modules (SDIO, USB) require higher level (HAL_ for *Hardware Abstraction Layer*) libraries also produced and continuously updated by ST. The middle level contains open-source intermediate level packages (*middleware*) from third-party sources. In Figure 4 these are FreeRTOS [ref10] and FATFS [11] which can be used on many different platforms. At the same level, the BSP (Board Support Package) contains specific peripheral initialization required by the board with functions such as internal clock configuration (*SystemClockConfig()*), and is run on startup for all applications.

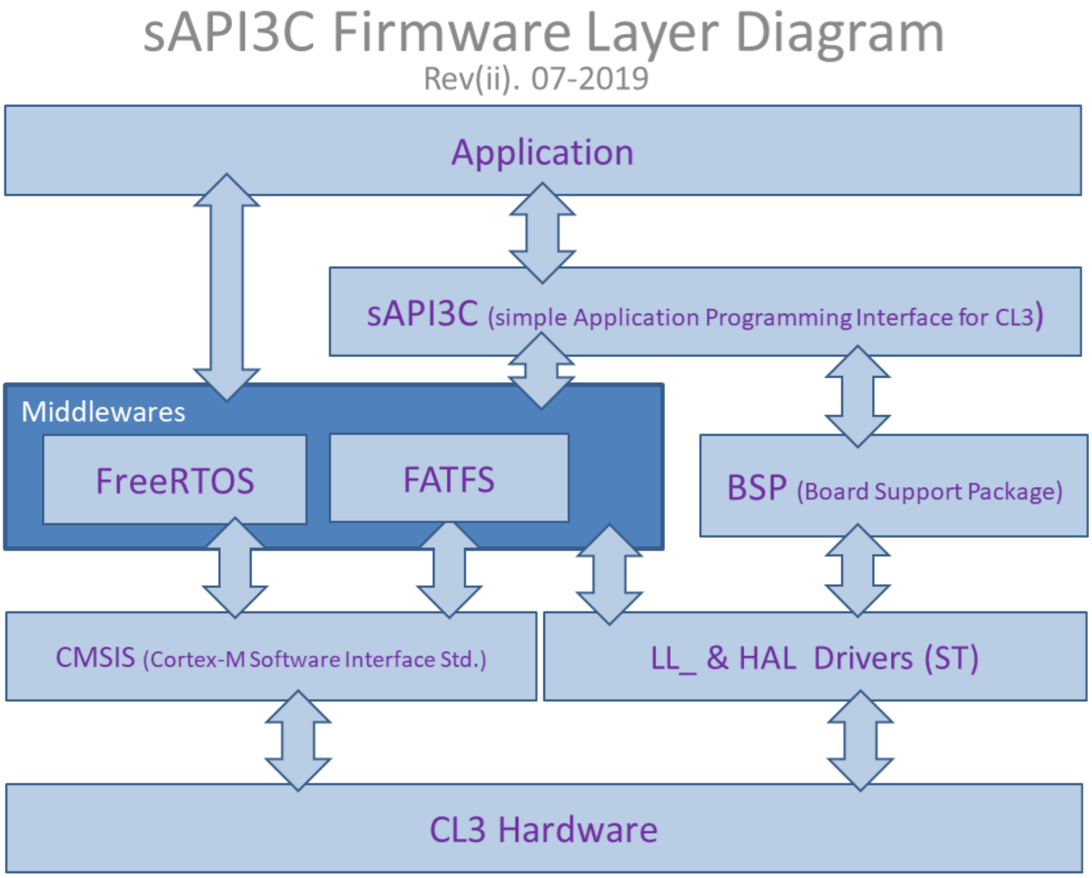


Figure 4 – Programming model layers for CL3 including sAPI-3C

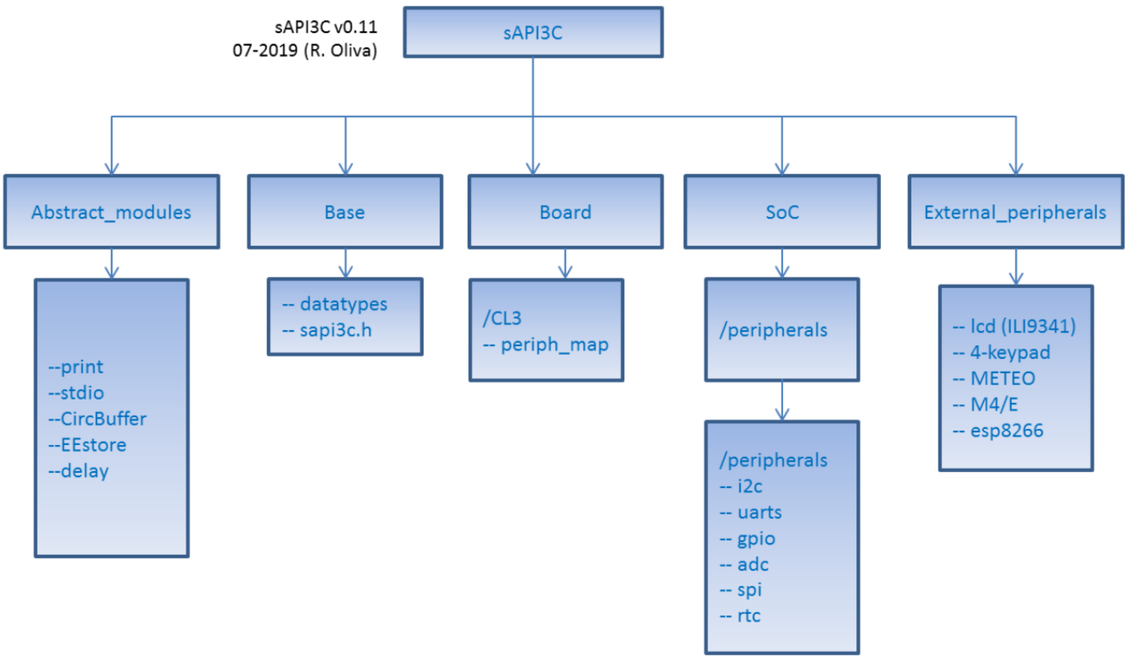


Figure 5 – Layers for CL3 within sAPI-3C

See APPENDIX I for typical hardware settings on CL3 board and connections required for the examples. The three examples which are numbered from 2 to 4 appear according to an increasing level of complexity. The first two are Bare Metal architecture. Example 4 shows the use of FreeRTOS as operating system.

a) Example2: (**sAPI3C_BM_ej2**) It blinks the OLED output, which as seen in the CL3 schematic [ref03] is mapped to PB.2 port as output and connected to D2, turning it on and off using simple blocking delays and the sapi3c_gpio() functions. BM indicates “Bare Metal”, meaning an infinite loop, no OS. This example requires JPL1 to be jumpered as shown in Figure 6. It also adds the use of: **a.1)** the UART6 serial port as a terminal at 115200,N,8,1 and from there to the RS232#0 converter, and **a.2)** the 4 contact membrane keyboard connected to the JK1 connector, which maps to controller ports PB.0, PB.1, PC.4 and PC.5. A low cost - general purpose RS232 serial to USB (e.g. Manhattan) converter is supposed to be available and attached to the RS232_2 connector and to the user PC as shown in Figure A.1 of APPENDIX I. A conventional Terminal program such as TeraTerm should be running on the PC.

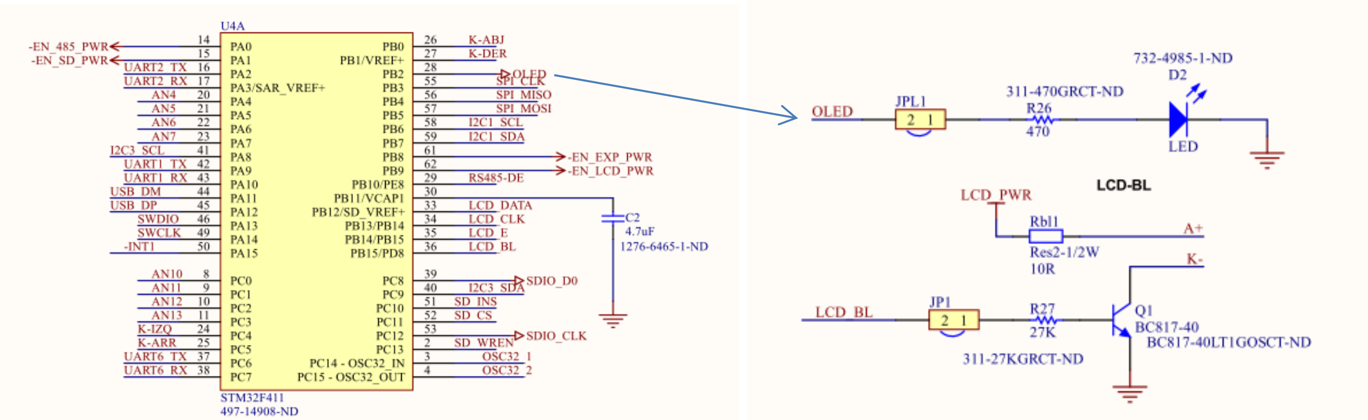


Figure 6 – OLED connection from PB2 on main controller (sheet 7) to OnBoard general purpose LED (OLED, sheet 3) of CL3 schematic

b) Example3: (sAPI3C_BM_ej3) It builds on Example 2, adds use of: **b.1)** the graphical ILI9341 LCD [ref10] display attached to the SPI port, for which the original board schematic was slightly modified. The display shows different static screens as each key F1 to F4 are pressed. As before, a low cost - general purpose RS232 serial to USB (e.g. Manhattan) converter is supposed to be available and attached to the RS232_2 connector and to the user’s PC as shown in Figure A.1 of APPENDIX I. A conventional Terminal program such as TeraTerm should be running on the PC.

c) Example4: (sAPI3C_FR_ej4) This example is based on the FreeRTOS real-time operating system. It uses **c.1)** the same UART6 serial port as a terminal, **c.2)** the graphical ILI9341 LCD display, **c.3)** the 4 contact membrane keyboard connected to the JK1 connector, and **c.4)** the METEO [ref06] board with a TTL-to-RS485 adapter such as RS485-BRD [ref09] connected to CN1(485) as shown in Figure A.2 of APPENDIX I.

3. EXAMPLES

These examples are prepared to be used with either the **AC6 System Workbench** [ref07] or the **STM32CubeIDE** [ref01]. Both programs are based on the Eclipse IDE, are free to download at the referenced locations, and only require user pre-registration on the company websites. The STMCubeIDE has much more frequent updates and includes the graphical CubeMX32 configurator if required. The initial interface for this program is shown in Figure 7. The examples can be downloaded directly as .zip or by cloning the repo at https://github.com/LyRIng/CL3board [ref11] to a directory in your local machine.

The examples can be freely modified or used with no changes. As with most Eclipse IDEs, an initial user workspace should be defined, then the downloaded examples can be imported using the menu items as in Figure 8. First go to **File→Open projects from File System**, and navigate to the local directory where the examples were downloaded. In Figure 8 Example 3 is shown and the indicated Import preferences should work. Press **Finish** and then **OK** to accept the import changes. The Project Explorer on the left pane can be expanded to observe the file structure of the example as in Figure 9. Once imported, the examples can be opened or closed by right-clicking on the Project Explorer as in Figure 10.

Figure 7 – STM32CubeIDE initial interface

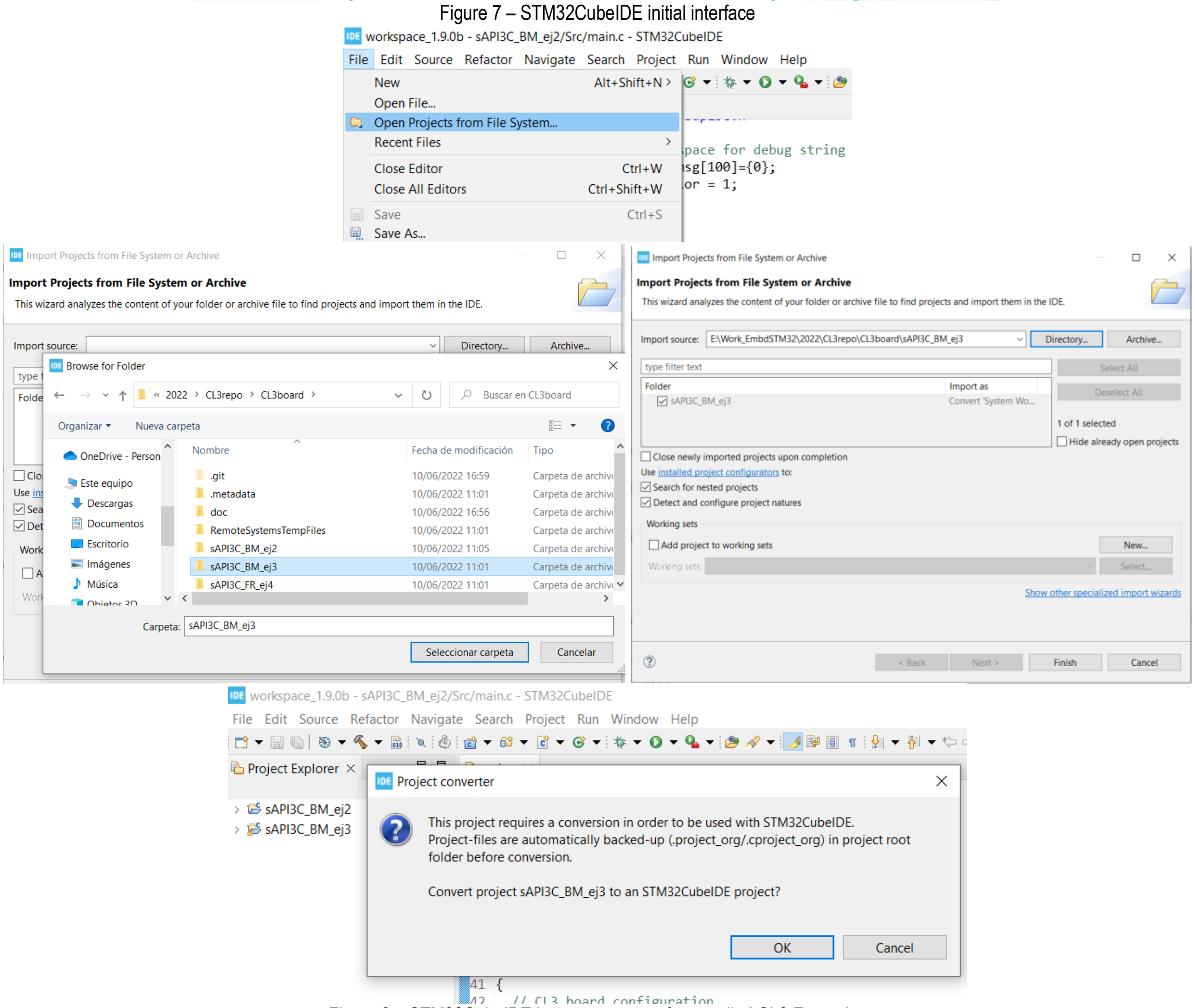


Figure 8 – STM32CubeIDE import sequence for supplied CL3 Examples

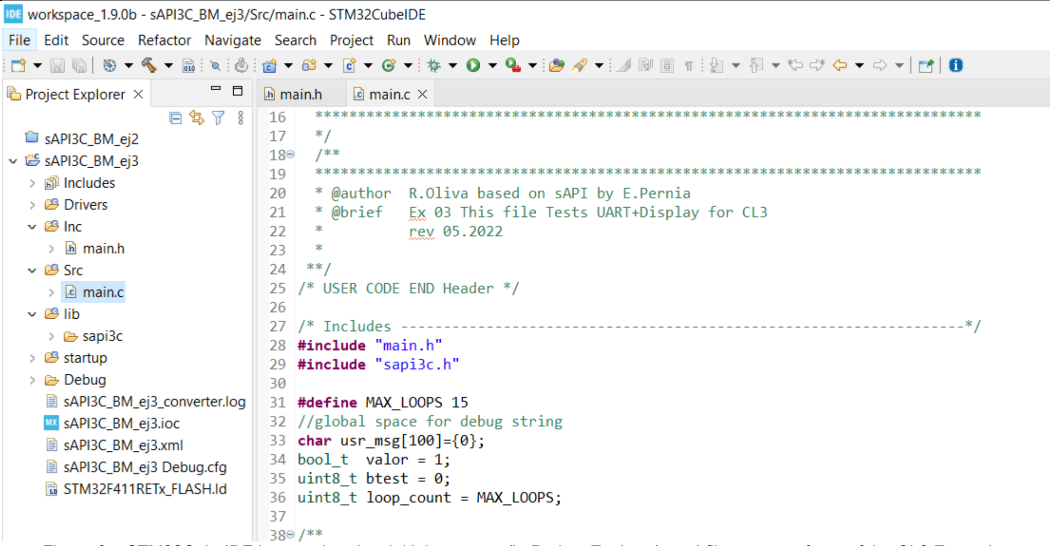


Figure 9 – STM32CubeIDE imported project initial structure (in Project Explorer), and file content of one of the CL3 Examples

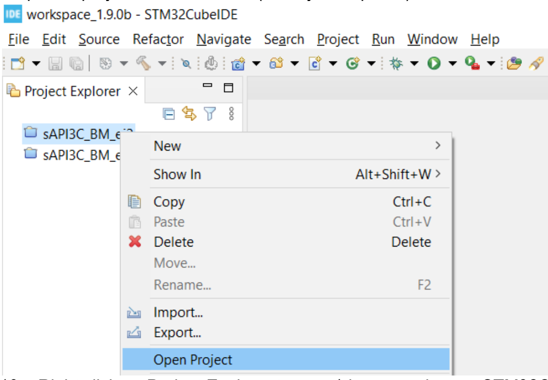


Figure 10 – Right click on Project Explorer to open/close a project on STM32CubeIDE

3.a Example 2 (sAPI3C_BM_ej2) : This simple program simply blinks the OLED general purpose on-board LED. Open the sAPI3C_BM_ej2 example as in Figure 10 and expand the directory structure as shown in Figure 11. The upper-layer user defined programs are in the src/ and /inc directories. The sapi3c library is under /lib and the controller specific files and startup code.

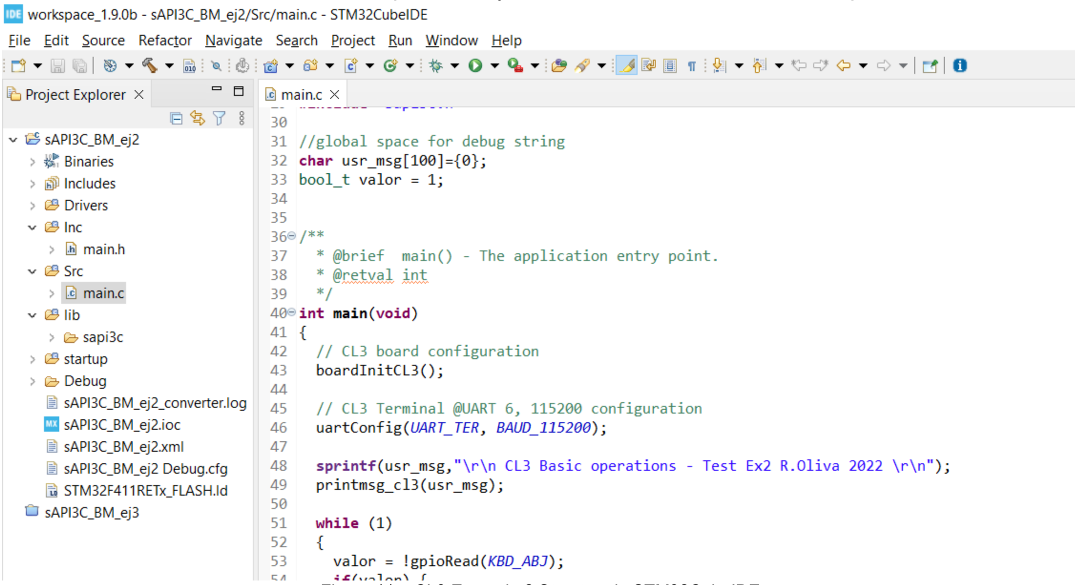


Figure 11 – CL3 Example 2 Structure in STM32CubeIDE

In disk space, the sAPI3C_BM_ej2 example has the layered structure and components which were mentioned in section 2, Figures 4 and 5, shown in Figure 12. It is not recommended to modify or edit these files outside the Eclipse IDE.

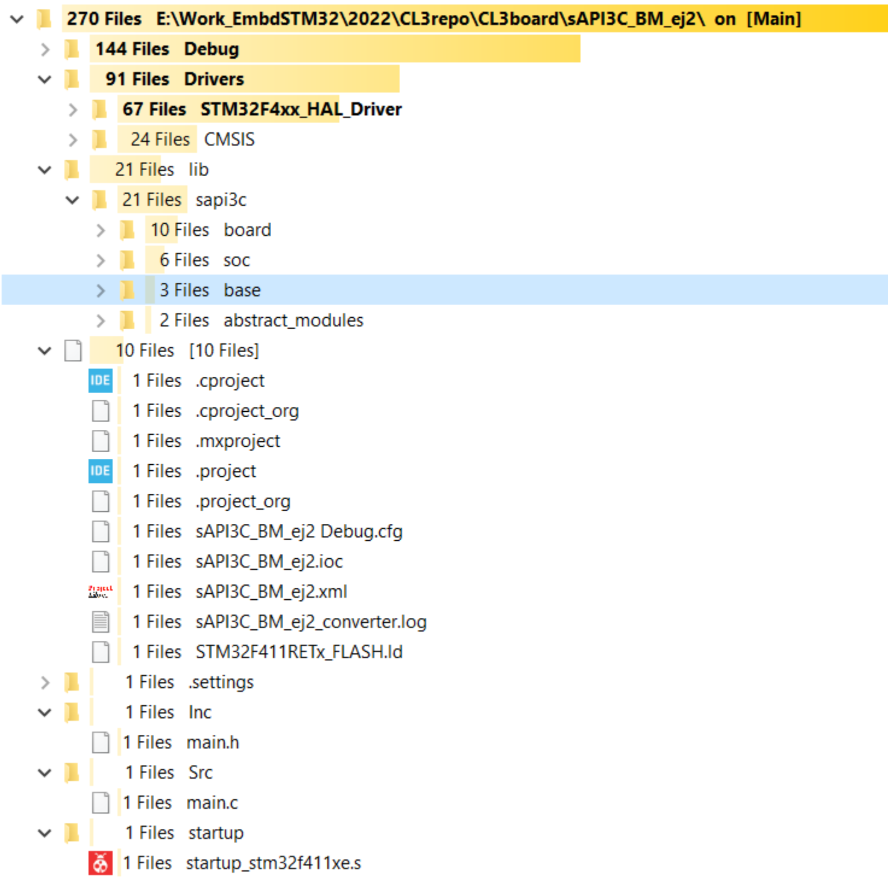


Figure 12 – Example 2 directory structure

3.a.1 Example 1 listing: This program is quite elementary but can be used to ensure that all board jumper settings (APPENDIX I) and interfaces are correct and that the IDE and compiler are working as expected.

```
  ******************************************************************************
  * @author  R.Oliva based on sAPI by E.Pernia
  * @brief   Example 02 - This file Tests basic operations for CL3
  *          rev 05.2022
  *
 **/
/* 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 CL3 Basic operations - Test Ex2 R.Oliva 2022 \r\n");
  printmsg_cl3(usr_msg);

  while (1)
  {
    valor = !gpioRead(KBD_ABJ);
      if(valor) {
            sprintf(usr_msg,"\r\n F3 Key pressed");
        printmsg_cl3(usr_msg);
      }
      delay_cl3(500);
    gpioWrite(OLED_PB2, ON);
    delay_cl3(500);
    gpioWrite(OLED_PB2, OFF);

  }
}
```

Listing Example 1

3.a.2 Using STM32CubeIDE to run Ex2: To run the example, connect all components as in Figure A.1 of APPENDIX I, and first go to Run→Debug Configurations, as in Figure 13.

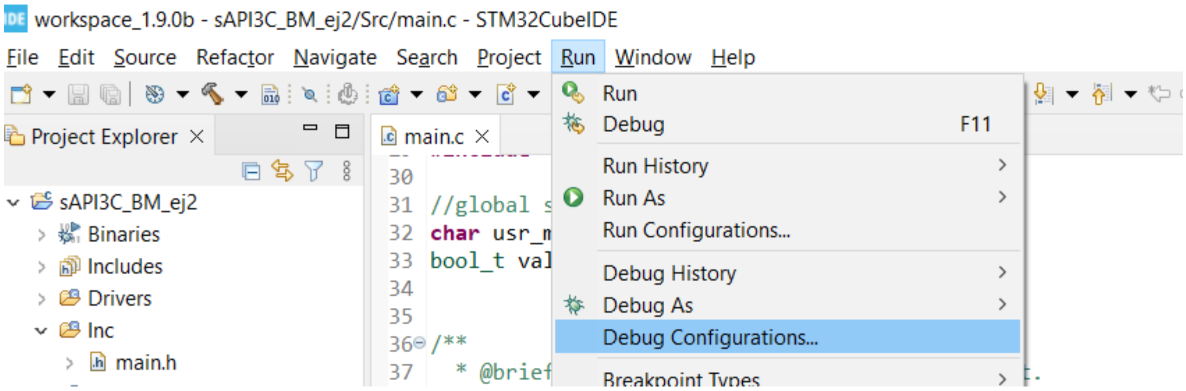


Figure 13 – Running and debugging the Example 2 project

Double click on the blue icon IDE STM32 Cortex M C/C++ application and the preconfigured sAPI3C_BM_ej2 Debug configuration should appear. Most of the options will work without modifications, as in Figure 14, but the debugger configuration might require changes, depending on the model of your debugger probe. Our examples were tested with the low cost STLink/v2 debugger using the "STLINK(OpenOCD)" probe, which works well with the options as shown in Figure 15. After setting this, click Apply. Power on the CL3 board and connect the probe to a free USB port on the PC. Then click Debug, and the Eclipse IDE should build the program with the results shown in Figure 15 (lower) and then switch to Debug view as in Figure 16. Here the **Play** button or **Step Over**, **Step Into** buttons can be used to observe program execution, step by step, or in normal running conditions.

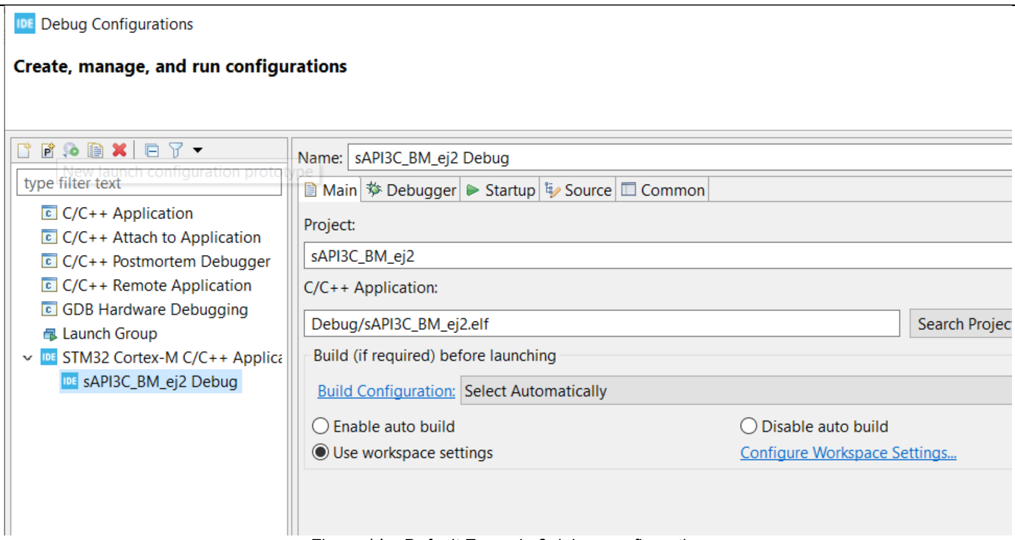


Figure 14 – Default Example 2 debug configuration

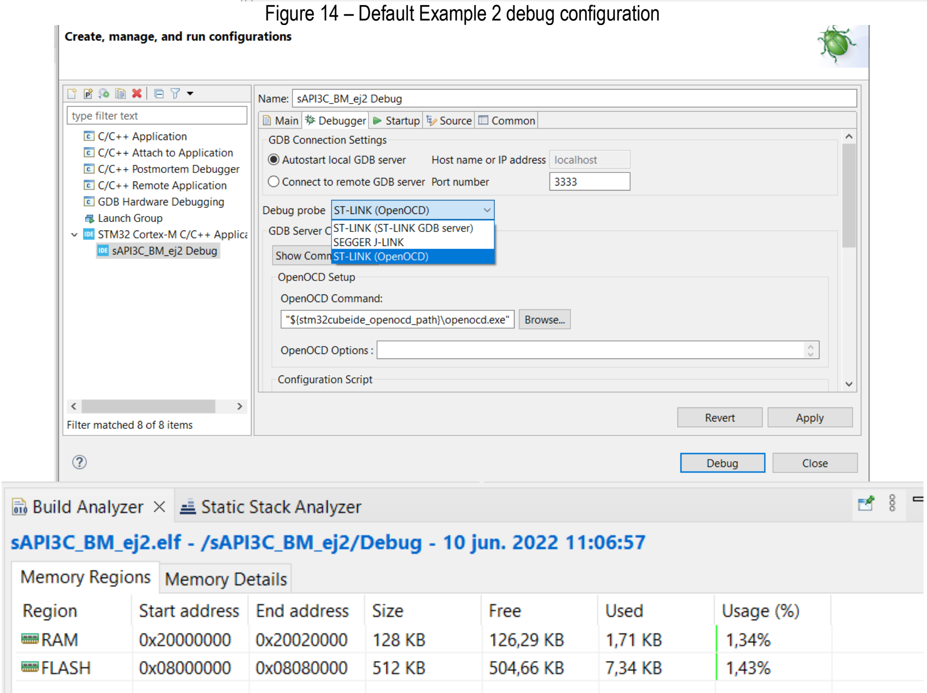


Figure 15 – Setting Debugger options for CL3 with STLink/V2 and openOCD. Clicking Debug should first build the program with the results shown.

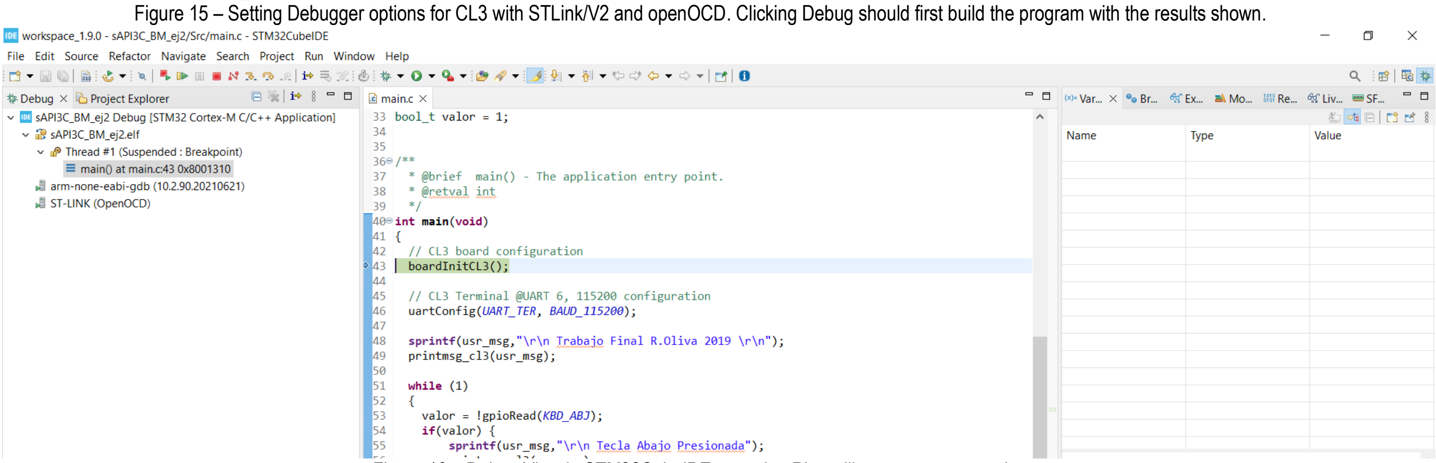


Figure 16 – Debug View in STM32CubeIDE, pressing Play will start program running

Once “Play” or Resume (F8) is clicked, with the main UART6 output connected to a PC via the RS232 to USB adapter, a terminal program such as TeraTerm should show (configured at 115200, N, 8, 1), a similar output as in Figure 17, when the F3 key is pressed in Example 2.

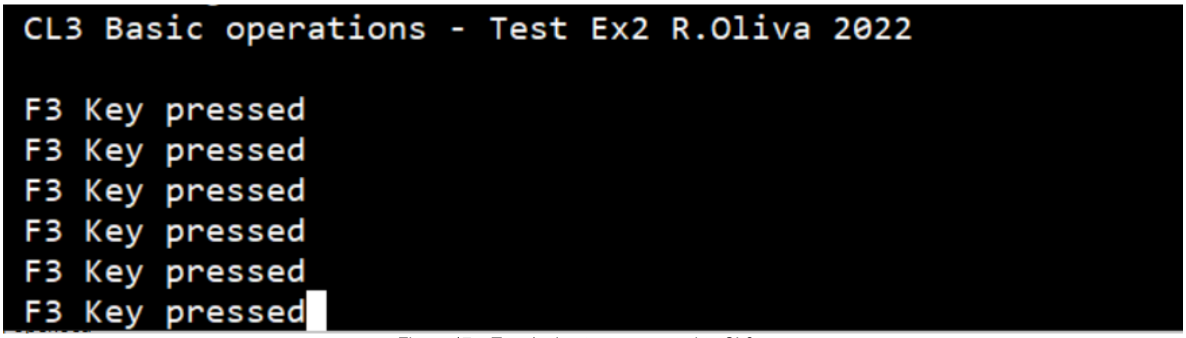


Figure 17 – Terminal output connected to CL3 port

3.b Example 3 (sAPI3C_BM_ej3) : This program is very similar to Example 2, but adds the use of the LCD Display element. In this case, an infinite loop is implemented with the serial output and LED blinking, but the ILI9341 display is initialized and screens changed upon pressing of different keys. The sequence is exactly the same as explained in Example 2.

3.b.1 Program Listing: Example 2 listing is shown in Listing 2.

```
/**
---
  ******************************************************************************
  */
  /**
  ******************************************************************************
  * @author  R.Oliva based on sAPI by E.Pernia
  * @brief   Ex 03 This file Tests UART+Display for CL3
  *          rev 05.2022
  *
 **/
/* 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 CL3 Basic operations - Test Ex3 (Display) R.Oliva 2022 \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 F1 (Left) Key pressed");
          printmsg_cl3(usr_msg);
          displaycl3_Screen(SCREEN1);
      }
    valor = !gpioRead(KBD_ARR);
      if(valor) {
            sprintf(usr_msg,"\r\n F2 (Up) Key Pressed");
        printmsg_cl3(usr_msg);
            displaycl3_Screen(SCREEN2);
      }
      valor = !gpioRead(KBD_ABJ);
      if(valor) {
            sprintf(usr_msg,"\r\n F3 (Dn) Key Pressed");
        printmsg_cl3(usr_msg);
            displaycl3_Screen(SCREEN3);
      }
      valor = !gpioRead(KBD_DER);
      if(valor) {
            sprintf(usr_msg,"\r\n F4 (Right) Key Pressed");
        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);
      }

  }
}
```

Listing 2

3.b.2 Using STM32CubeIDE to run Ex3: To run this example, connect all components as in Figure A.1 of APPENDIX I, and repeat the sequence from Run→Debug Configurations, as in Figure 13 through Figure 16 with the _Ej3 project. Figure 18 shows the display output of a CL3 (to the right) connected to a ILI9341 display and executing through an STLink/V2 debugger probe on the left section of a Nucleo Development board. Figure 19 shows the F4 screen, the detail of the STLink/V2 probe (electrically detached from a generic STM32 Nucleo board) and the terminal output for Example 3.

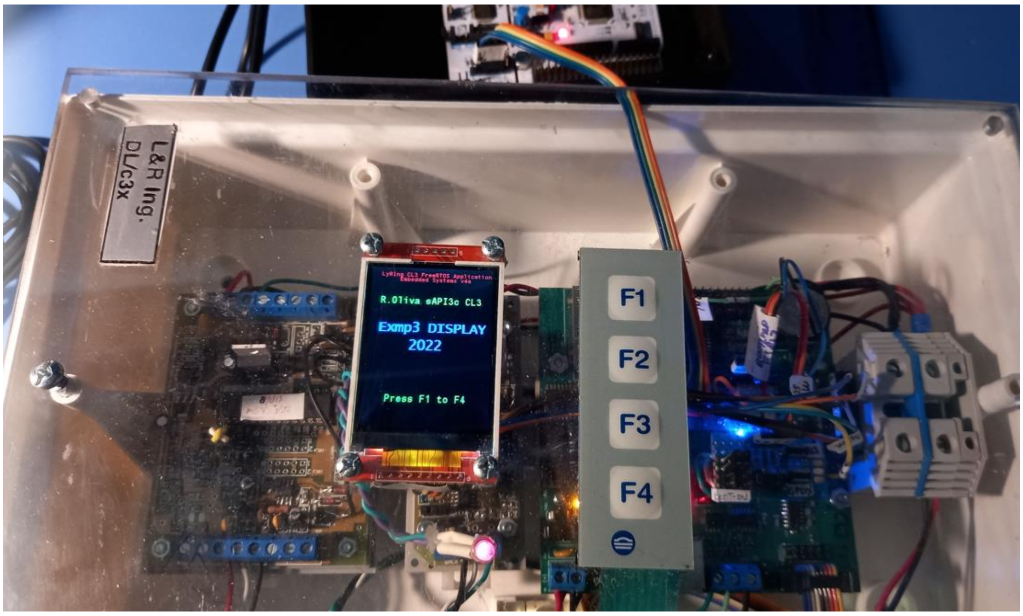
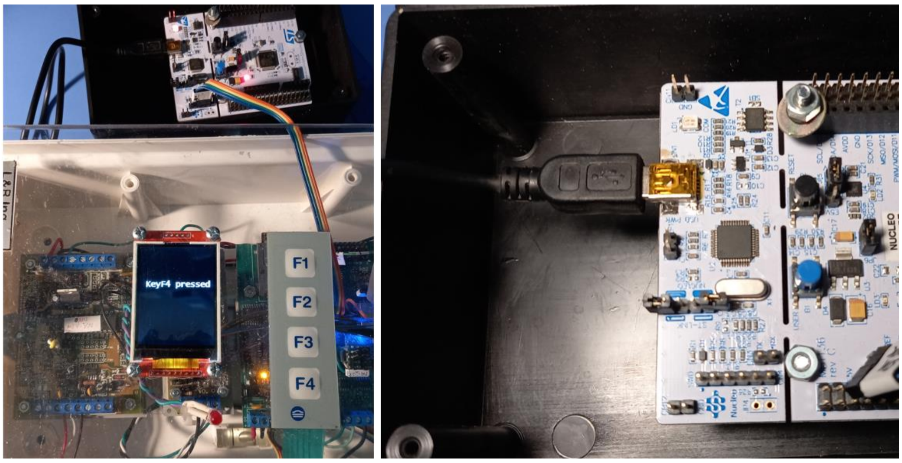


Figure 18 – Display connected to CL3 (right), with Example 3 program executing through STLink/v2 probe (top)




```
CL3 Basic operations - Test Ex3 (Display) R.Oliva 2022

F1 (Left) Key pressed
F2 (Up) Key Pressed
F3 (Dn) Key Pressed
F4 (Right) Key Pressed
F1 (Left) Key pressed
```

Figure 19 – Display for F4, STLink/V2 probe and Terminal output for Example 3

3.c Example 4 (sAPI3C_FR_ej4) : This program is a more complex application using FreeRTOS as middleware and connecting the CL3 board to a METEO peripheral using an RS485 link (see Figure A.2 in APPENDIX I). The structure of the project is shown in Figure 20, and is considerably longer since it includes more files in the /src and /inc directories. Figure 21 shows the main display output and the internal connection to a METEO board (left) with an RS485 adapter in the middle. The CL3 board is being debugged as in prior examples using the STLink/v2 interface to a USB port on a PC running STM32CubeIDE. A second USB port is used to connect the Terminal port to show the main user interface (Figure 22).

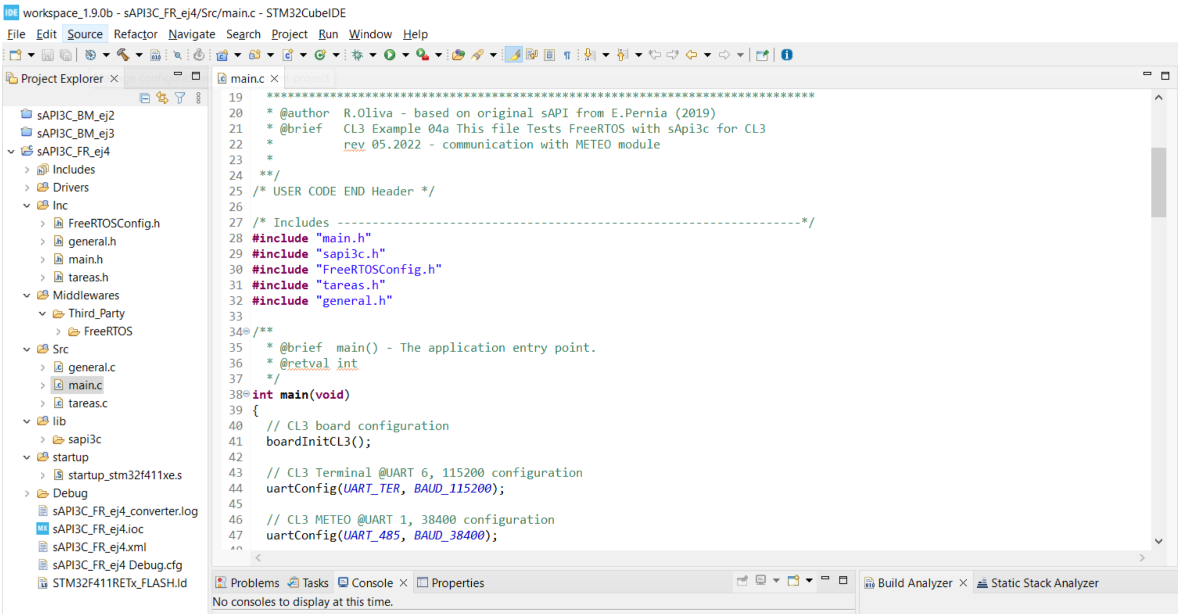


Figure 20 – Example 4 project structure with FreeRTOS as middleware

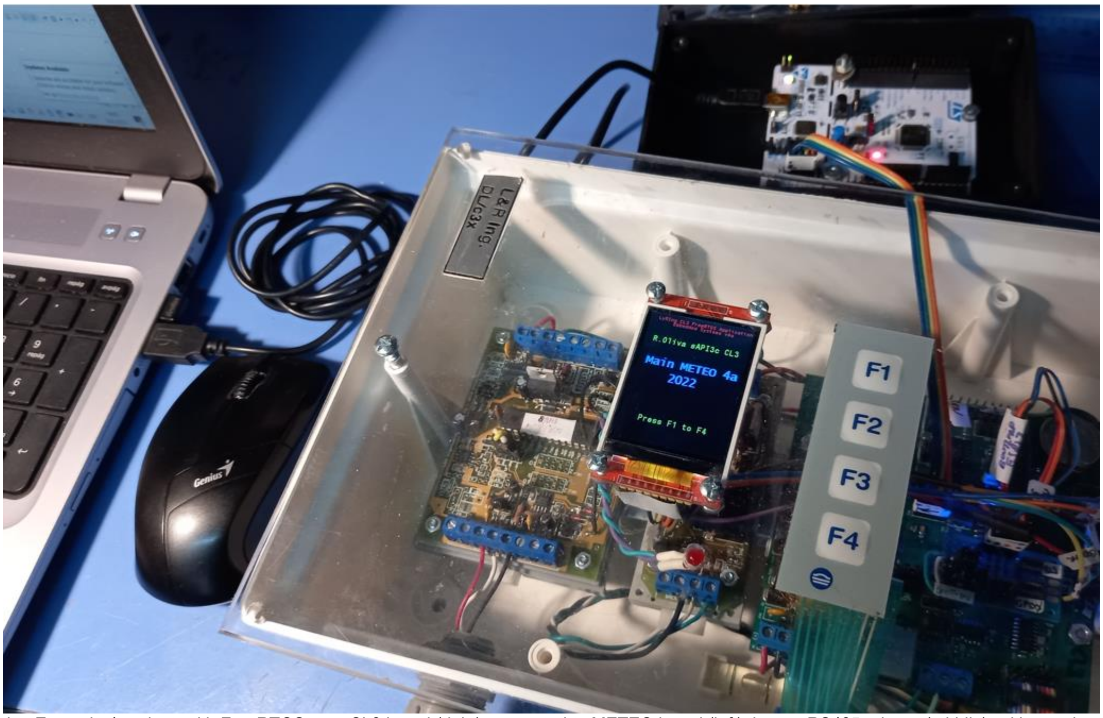


Figure 21 – Running Example 4 project with FreeRTOS on a CL3 board (right) connected to METEO board (left) thru an RS485 adaper (middle), with membrane Keyboard and ILI9341 display to the SPI port connector. The PC connects through the USB port to the STLink V2 debugger (top) and another USB port for the Terminal.

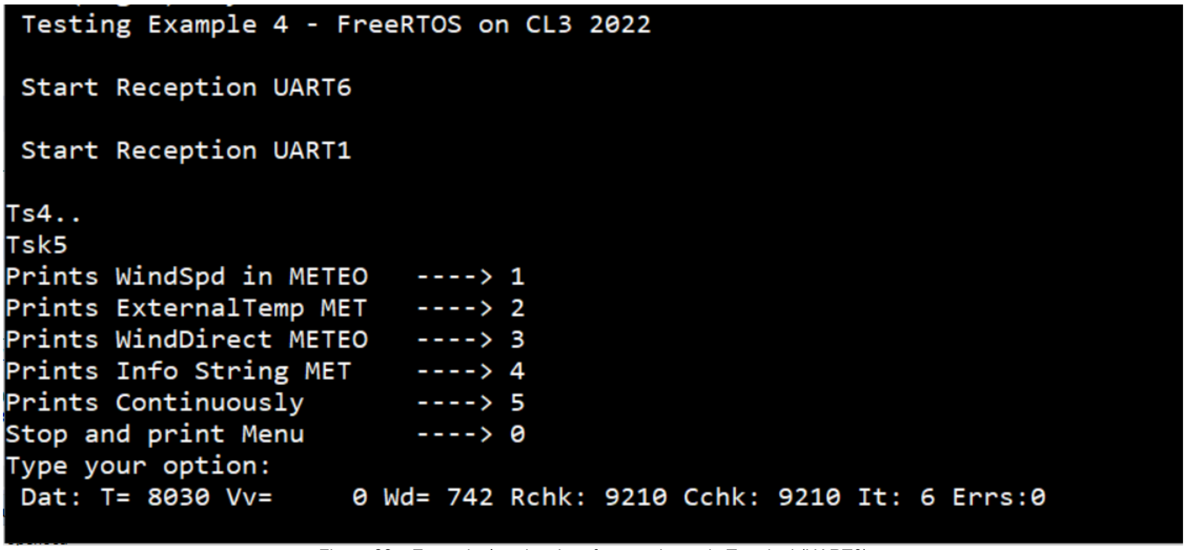


Figure 22 – Example 4 project Interface on the main Terminal (UART6)

In FreeRTOS applications the program initializes a series of tasks and starts a scheduler, which defines execution time-slots to each task. Example 4 has a quite complex Task distribution which is outlined in Figure 23. A more complete graphical task detail is shown in Figure A.3 of APPENDIX I. A thorough description can be found (in Spanish) in [ref12].

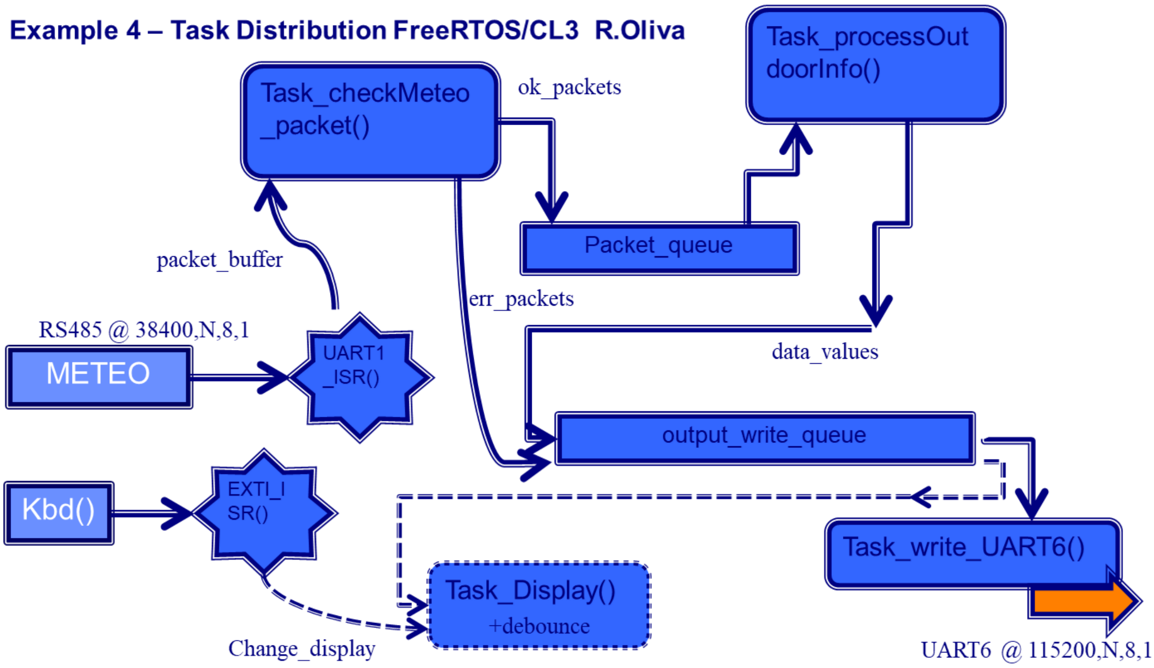


Figure 23 – Example 4 project task distribution

4. CONCLUSIONS

A set of 3 example programs for the CL3 board show the programming sequence using the STM32CubeIDE compiler. These examples in source code and documentation are available on the github repository for the CL3 board [ref11]: https://github.com/LyRIng/CL3board . Further references on FreeRTOS can be found at [ref13].

5. REFERENCES

[ref00] STM23F411RET microcontroller: https://www.st.com/en/microcontrollers-microprocessors/stm32f411re.html
[ref01] STM32CubeIDE: https://www.st.com/en/development-tools/stm32cubeide.html
[ref02] ARM-GCC (GNU Arm Embedded Toolchain): https://developer.arm.com/downloads/-/gnu-rm
[ref03] ST-LINK/v2 In Circuit Debugger / Programmer: https://www.st.com/en/development-tools/st-link-v2.html
[ref04] CL2 Board from L&R Ing. Info, examples, documentation: https://github.com/LyRIng/CL2bm1
[ref05] CL3 Board Schematic: https://github.com/LyRIng/CL3board/blob/master/doc/cl3schem/Cl3_Schem.pdf
[ref06] METEO module https://github.com/LyRIng/METEO
[ref07] AC6 STM32 System Workbench https://www.ac6-tools.com/content.php/content_SW4MCU/lang_en_GB.xphp
[ref08] STM32CubeMX Graphical Initialization code generator https://www.st.com/en/development-tools/stm32cubemx.html
[ref09] RS485-BRD 3 board or module: https://www.lyringenieria.com.ar/productos/rs485-brd-rs-485-rs-232-ttl-board-module-3/
[ref10] ILI9341 LCD Graphical display (non-touch) https://esphome.io/components/display/ili9341.html
[ref11] CL3 Examples and documentation repo: https://github.com/LyRIng/CL3board
[ref12] Final CL3 work FIUBA / R.Oliva: http://laboratorios.fi.uba.ar/lse/tesis/LSE-FIUBA-Trabajo-Final-CESE-Rafael-Oliva-2019.pdf
[ref13] FreeRTOS Real Time OS: https://freertos.org/

Revision date: June, 2022

APPENDIX 1 : CL3 Typical connection for examples

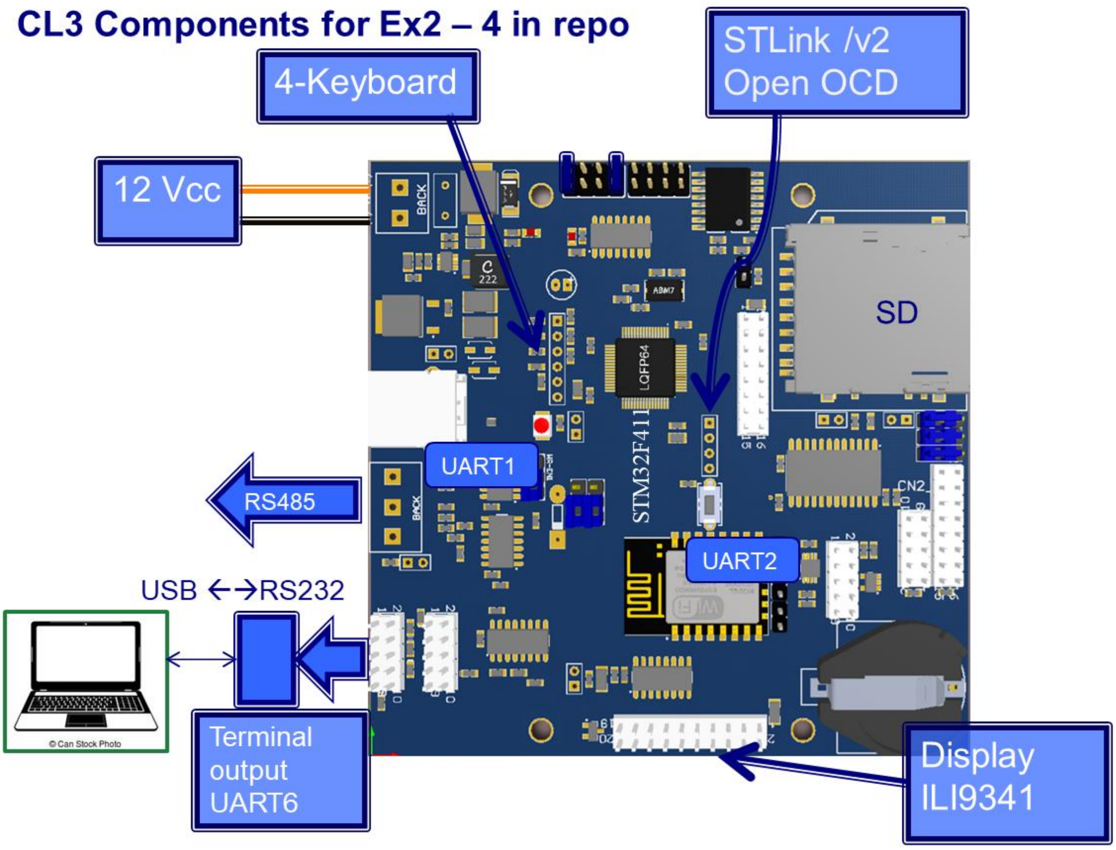


Figure A.1 – Connecting the CL3 board for Example 2

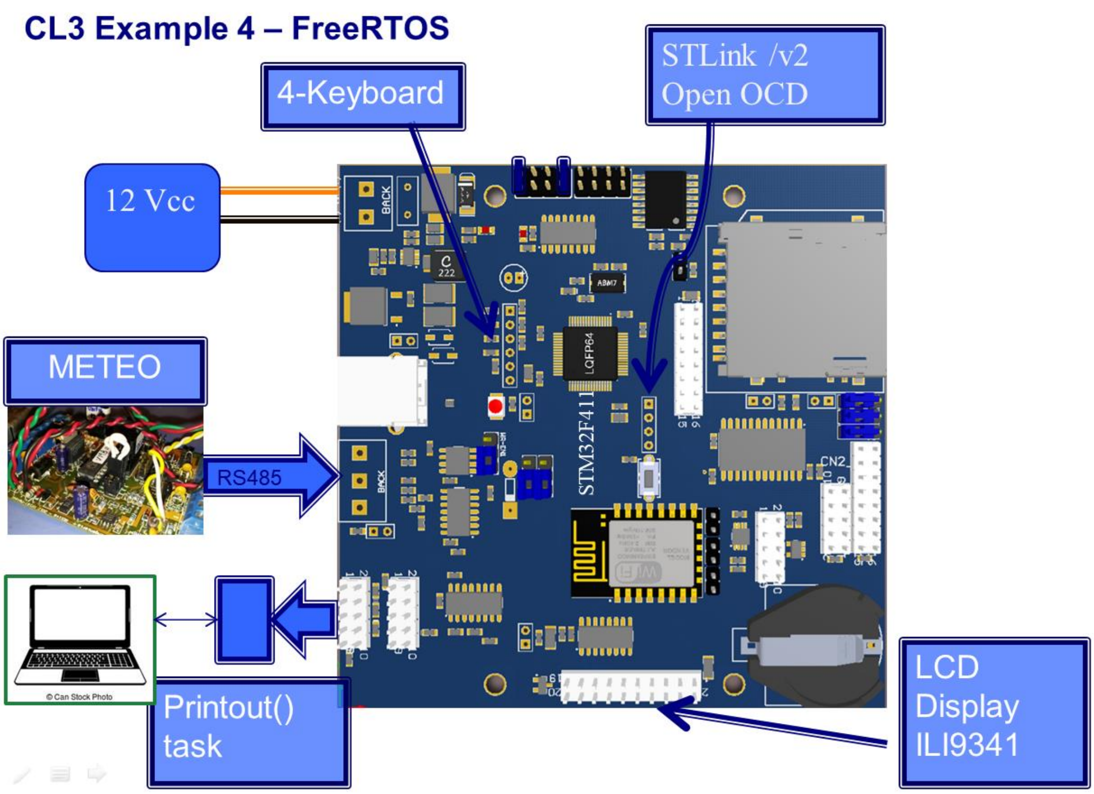


Figure A.2 – CL3 board connection for Example 4

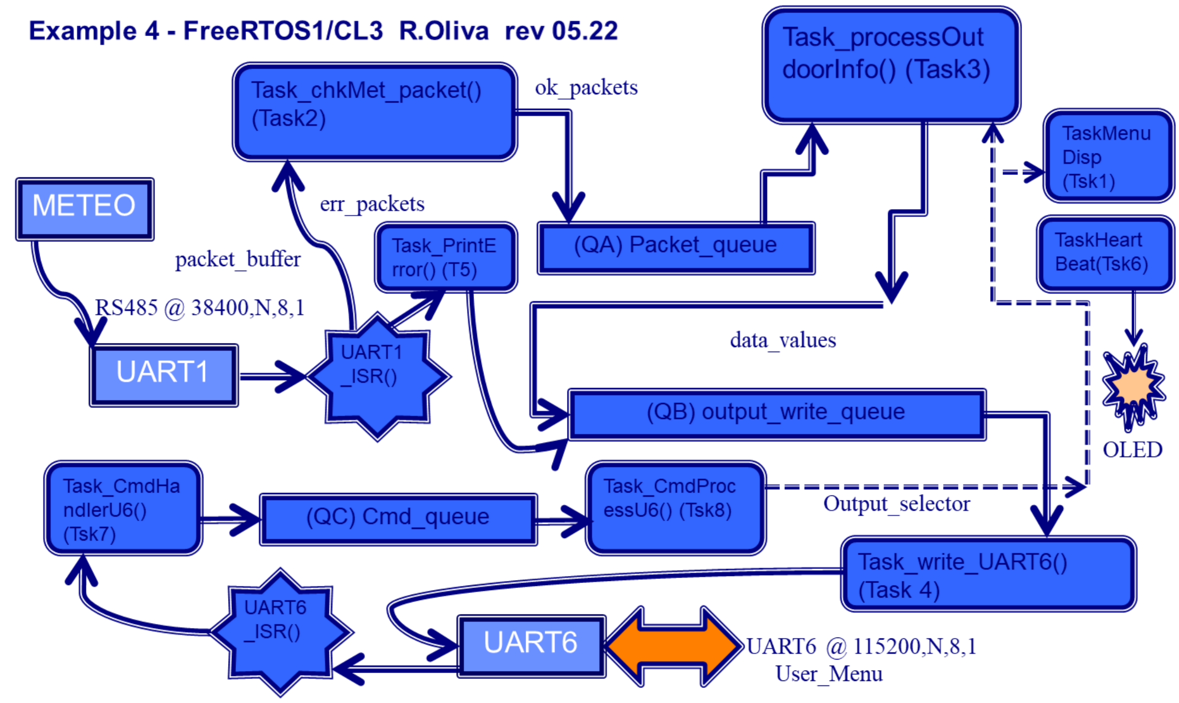


Figure A.3 – Example 4 on CL3 detailed Task Distribution