UM2776

User manual

STM32CubeG0 STM32G0C1E-EV demonstration firmware

Introduction

STM32Cube is an STMicroelectronics original initiative to significantly improve designer's productivity by reducing development effort, time, and cost. STM32Cube covers the whole STM32 portfolio.

STM32Cube includes:

- A set of user-friendly software development tools to cover project development from conception to realization, among which are:
  - STM32CubeMX, a graphical software configuration tool that allows the automatic generation of C initialization code using graphical wizards
  - STM32CubeIDE, an all-in-one development tool with peripheral configuration, code generation, code compilation, and debug features
  - STM32CubeProgrammer (STM32CubeProg), a programming tool available in graphical and command-line versions
  - STM32CubeMonitor (STM32CubeMonitor, STM32CubeMonPwr, STM32CubeMonRF, STM32CubeMonUCPD) powerful monitoring tools to fine-tune the behavior and performance of STM32 applications in real-time
- STM32Cube MCU and MPU Packages, comprehensive embedded-software platforms specific to each microcontroller and microprocessor series (such as STM32CubeG0 for the STM32G0 Series), which include:
  - STM32Cube hardware abstraction layer (HAL), ensuring maximized portability across the STM32 portfolio
  - STM32Cube low-layer APIs, ensuring the best performance and footprints with a high degree of user control over hardware
  - A consistent set of middleware components such as FAT file system, RTOS, OpenBootloader, USB Host, USB Device, and USB Power Delivery
  - All embedded software utilities with full sets of peripheral and applicative examples
- STM32Cube Expansion Packages, which contain embedded software components that complement the functionalities of the STM32Cube MCU and MPU Packages with:
  - Middleware extensions and applicative layers
  - Examples running on some specific STMicroelectronics development boards

The STM32CubeG0 demonstration firmware running on the STM32G0C1E-EV Evaluation board is built around the STM32Cube hardware abstraction layer (HAL) and low-layer (LL) APIs, and board support package (BSP) components. It embeds several applications that demonstrate various features of the STM32G0C1VET6 device and exercise some peripherals of the STM32G0C1E-EV Evaluation board. These applications are:

- UCPD application
- Low-power application
- Image viewer application
- Audio application
- Calendar application
- Thermometer application
- File browser application

1 STM32CubeG0 main features

STM32CubeG0 gathers, in a single package, all the generic embedded software components, required to develop an application on STM32G0 microcontrollers. In line with the STM32Cube initiative, this set of components is highly portable, not only to the STM32G0 Series but also to other STM32 series.

STM32CubeG0 is fully compatible with the STM32CubeMX code generator that allows the generation of initialization code.

The package includes a driver layer (HAL) proposing a set of abstraction services and a low-level hardware layer (LL) proposing a set of register-level functions, together with an extensive set of examples running on STMicroelectronics boards. HAL is available in an open-source BSD license for user convenience.

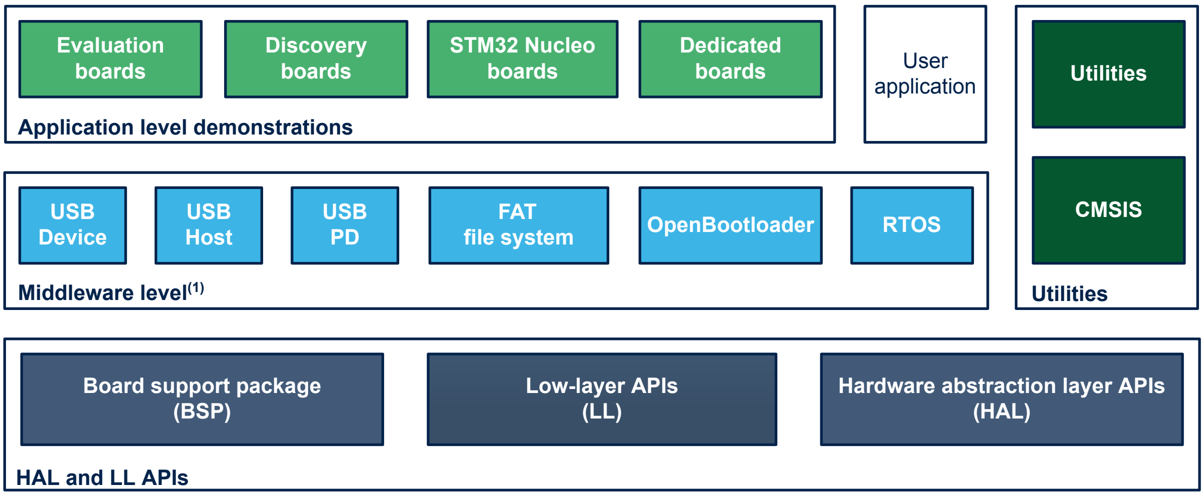
The STM32CubeG0 MCU Package also contains a set of middleware components with the corresponding examples. They come in free user-friendly license terms:

- FAT file system based on open source FatFS solution
- CMSIS-RTOS implementation with FreeRTOS™ open source solution
- USB PD Devices and Core libraries
- USB Host and Device libraries
- OpenBootloader (OpenBL)

Several applications and demonstrations implementing all these middleware components are also provided in the STM32CubeG0 MCU Package.

The block diagram of STM32Cube is shown in Figure 1.

Figure 1. STM32CubeG0 firmware components



(1) The set of middleware components depends on the product Series.

The demonstration firmware for the STM32G0C1E-EV Evaluation board comes on top of the STM32CubeG0 MCU Package, which is based on a modular architecture allowing the reuse of software components separately in standalone applications. All modules are managed by the STM32Cube demonstration kernel, which allows the addition of new modules dynamically and access to the storage, graphical, and components common resources.

The demonstration firmware is built on the light kernel and services provided by the BSP as well as components based on the STM32Cube HAL It makes extensive use of the STM32G0 device capability to offer a wide scope of usages.

The STM32G0 microcontrollers are based on the Arm® 32-bit Cortex®-M0+ processor.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2 Getting started with the demonstration

2.1 Hardware requirements

The hardware requirements to start the demonstration are the following:

- One STM32G0C1E-EV Evaluation board (MB1581)
- One microSD™ card
- One USB cable to power the STM32G0C1E-EV board from its ST-LINK USB

The STM32G0C1E-EV (MB1581) Evaluation board is connected to the host PC using the USB cable. It does not require any external power supply.

The demonstration displays icons that are stored on the microSD™ card. The microSD™ card must be loaded with several files (`*.bmp`, `*.txt`, `*.bin`) that are provided in the `/Projects/STM32G0C1E-EV/Demonstrations/Binary/SD_card` firmware package directory.

2.2 Hardware settings

The STM32CubeG0-based demonstration supports the STM32G0C1VET6 device and runs on the STM32G0C1E-EV Evaluation board (MB1581) from STMicroelectronics mounted with one of the two possible daughterboards, as shown in Figure 2 and Figure 3:

- The legacy peripheral daughterboard (MB1351), referenced as the legacy daughterboard in this document.
- The USB-C® and Power Delivery daughterboard (MB1352), referenced as UCPD daughterboard in this document.

Figure 2. STM32G0C1E-EV with legacy daughterboard (MB1351)

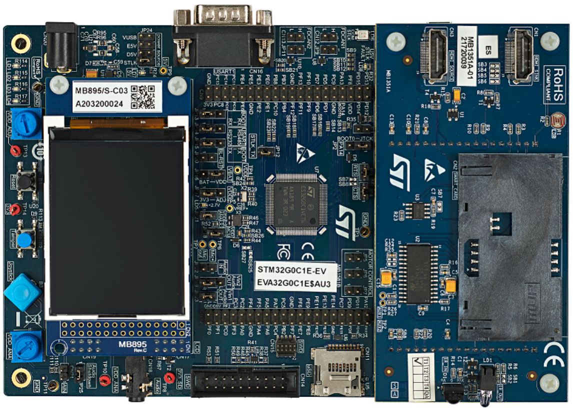
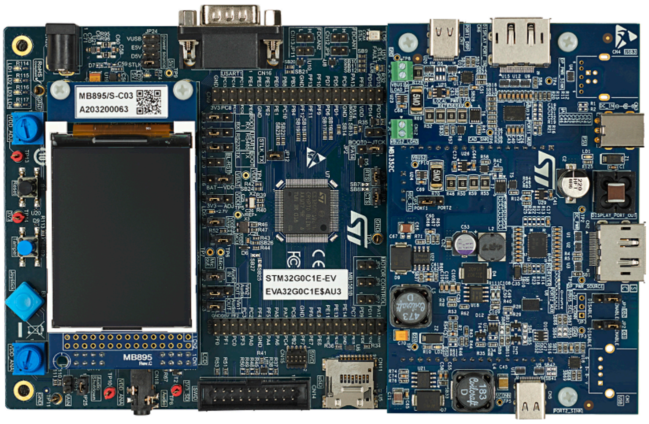


Figure 3. STM32G0C1E-EV with UCPD daughterboard (MB1352)



Pictures are not contractual.

The default jumper settings must be used to run the demonstration. Refer to [2] for details.

2.2.1 STM32G0C1E-EV Evaluation board settings

Figure 4. STM32G0C1E-EV Evaluation board (MB1581)

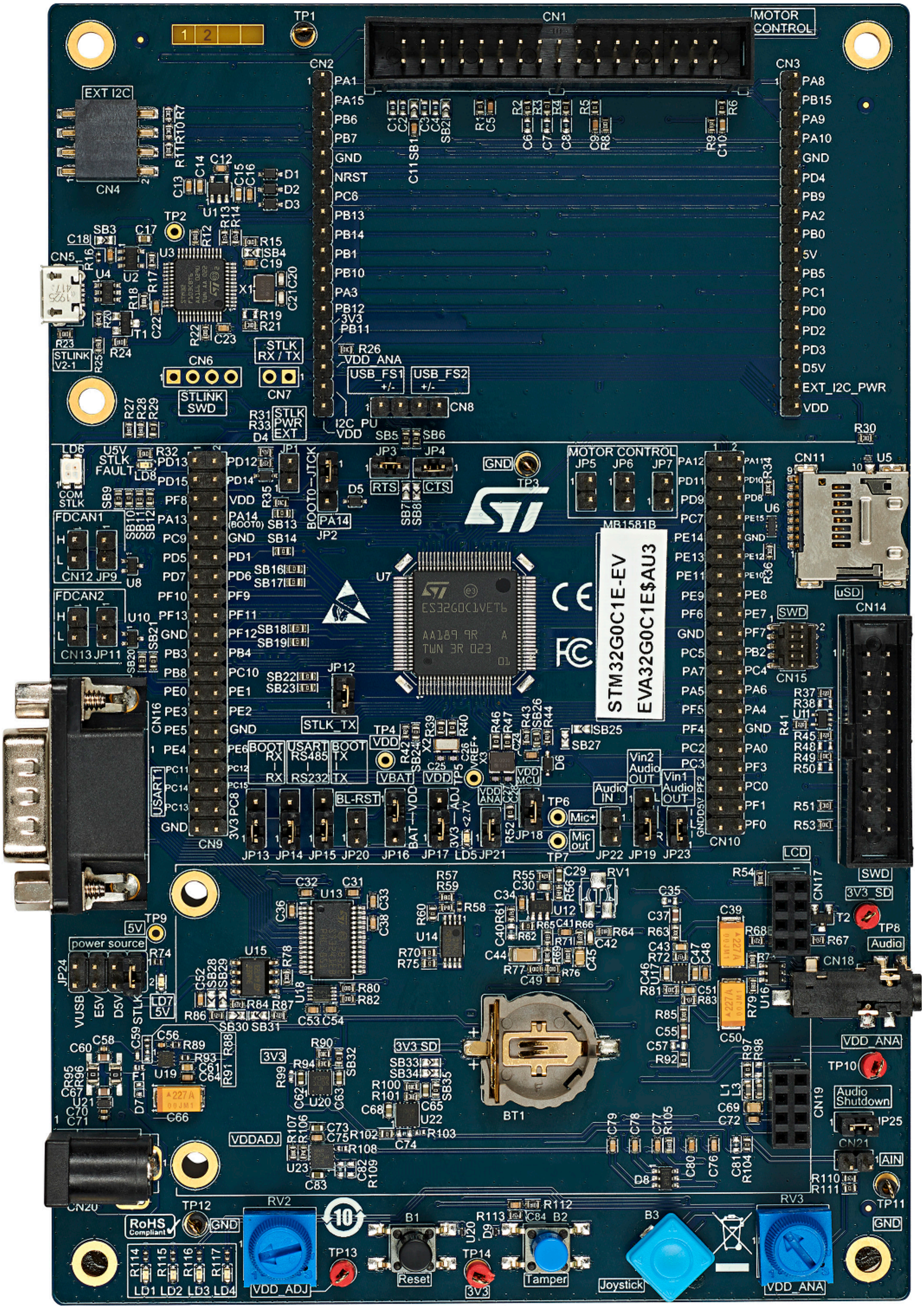


Table 1. STM32G0C1E-EV jumper settings

Jumper	Position description
JP1	JTAG/SWD Interface (Test Data Out pin connection): OFF (grounded)
JP2	PA14-BOOT0 usage: PA14-BOOT0 is used as SWCLK
JP3	USB_FS_P routing: ON (USART1_RTS)
JP4	USB_FS_N routing: ON (USART1_CTS)
JP5	MC Heatsink Temperature: OFF
JP6	MC Bus Voltage: OFF
JP7	MC Emergency STOP: OFF
JP9	FDCAN 1: OFF
JP11	FDCAN 2: OFF
JP12	VCP connection: ON (STLINK TX connected to VCP RX)
JP13	BOOT RX/RX: PC5 is connected as RX signal without bootloader being supported
JP14	RS-232 vs. RS-485: RS-485
JP15	BOOT TX/TX: PC4 is connected as TX signal without bootloader being supported
JP16	VBAT: connected to VDD
JP17	VDD: connected to 3.3 V
JP18	VDD1 MCU: connected to VDD
JP19	Audio playback mode: stereo playback (audio amplifier VIN2 connected to PA5)
JP20	Bootloader Reset: OFF (not connected to NRST pin)
JP21	VDD ANA: ON (connected to VDD source, 3.3 V)
JP22	Audio input: ON (CN18 microphone input connected to PA6)
JP23	Audio amplifier VIN1 connection: ON (connected to PA4)
JP24	Power supply: STLK
JP25	Audio amplifier: ON (enabled)

2.2.2 STM32G0C1E-EV legacy daughterboard

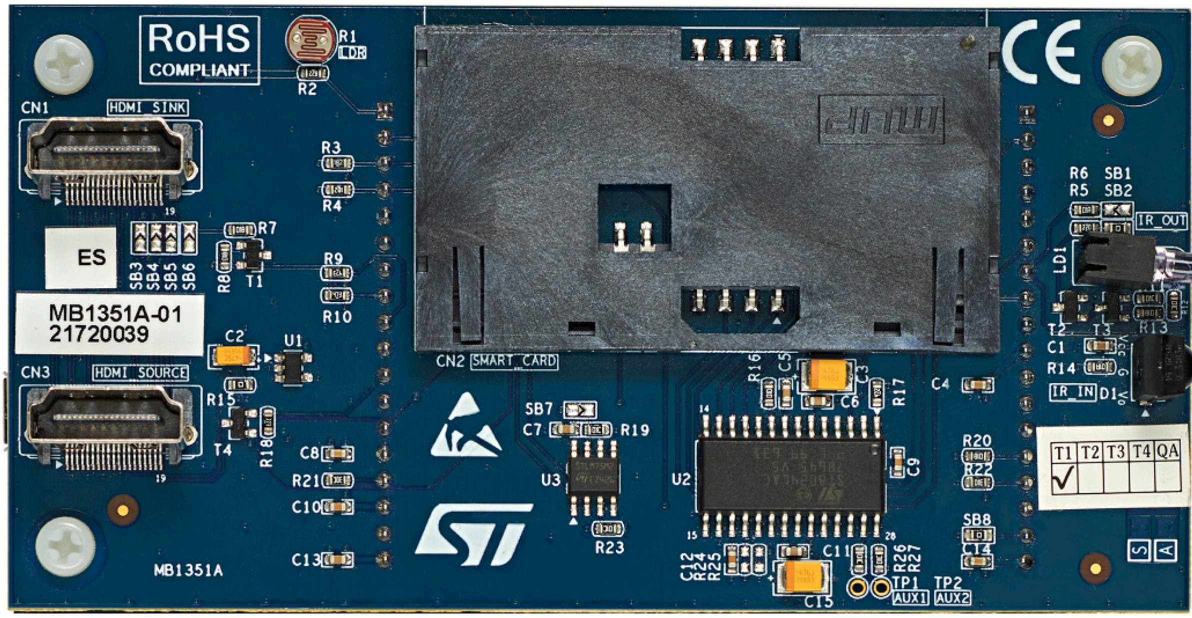


Figure 5. STM32G0C1E-EV legacy daughterboard (MB1351)

2.2.3 STM32G0C1E-EV UCPD daughterboard settings

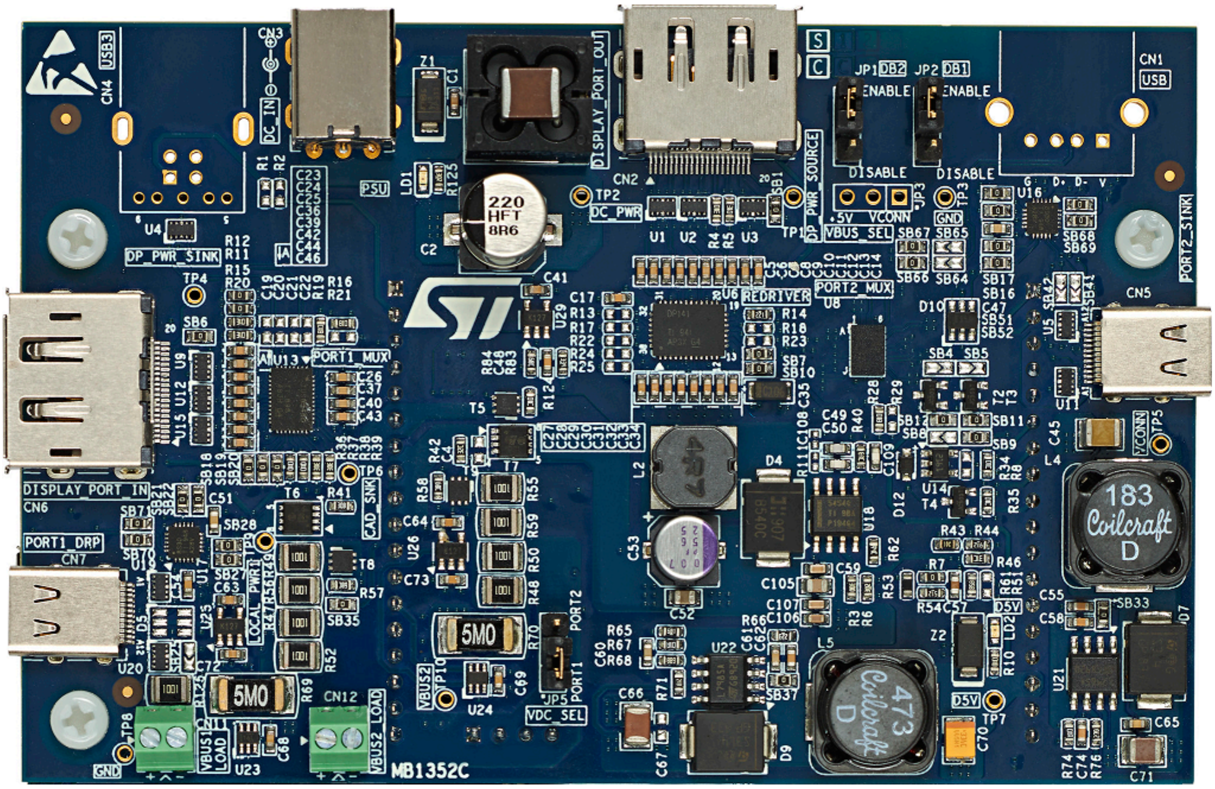


Figure 6. STM32G0C1E-EV UCPD daughterboard (MB1352)

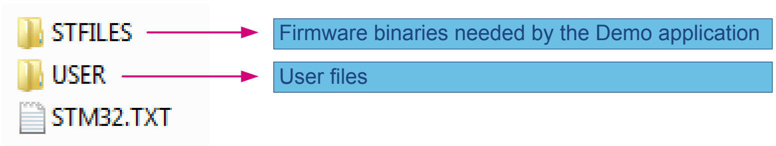
Table 2. STM32G0C1E-EV UCPD daughterboard jumper settings

Jumper	Position description
SW1	Under UCPD shield, in the opposite position indicated by default marking
CN3	19V power supply must be plugged into the UCPD shield.
JP1	Dead battery function is enabled.
JP2	Dead battery function is enabled.
JP5	UCPD daughterboard D5V is generated from external 19 V or V_{BUS} on Port 1.

2.3 microSD™ status

The STM32G0C1E-EV board comes with a microSD™ memory card pre-programmed with the image resources, text files, and directory trees used by the demonstration firmware. However, the user may load his image files in the USER directory, assuming that file formats are supported by the demonstration (`*.bmp`).

Figure 7. microSD™ card directory organization



If the microSD™ card is not correctly inserted or not well programmed, an error message is displayed.

2.4 Demonstration firmware

First, select the folder corresponding to the demonstration, then inspect the folder corresponding to the preferred toolchain (EWARM, MDK-ARM, or STM32CubeIDE):

- Open the corresponding project
- Rebuild all sources
- Load the project image using the debugger
- Restart the Evaluation board (press reset button B3)

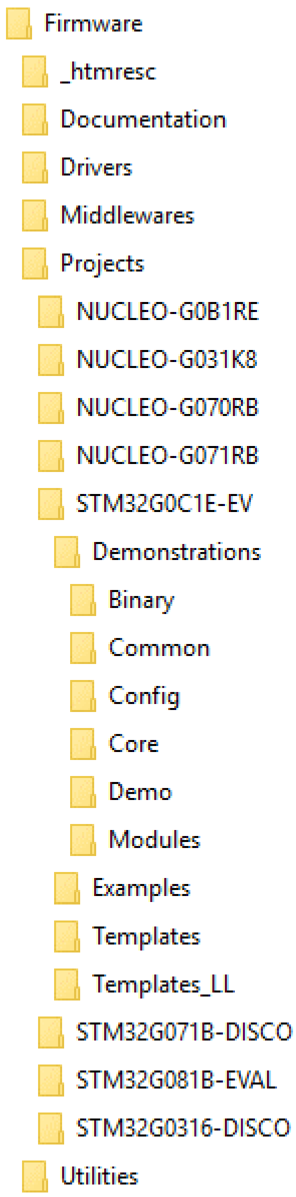
3 Demonstration description

The demonstration targets the following objectives:
- Toolkit with low memory consumption
- Independent modular applications with a high level of reuse
- Basic menu navigation with the joystick
- Comprehensive STM32G0 functional coverage

3.1 Demonstration package

Figure 8 shows the demonstration folder organization.

Figure 8. Demonstration folder organization



The demonstration sources are located in the Projects folder of the STM32CubeG0 MCU Package for each supported board. The folder selected here is STM32G0C1E-EV for the Evaluation board.

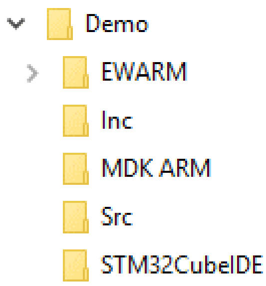
The files making up the demonstration firmware are spread over the following sub-directories:

- Binary: includes the SD_card sub-folder containing the resource files (such as bitmaps and binaries) required by the demonstration firmware to be copied on the microSD™ card.
- Common: C source files implementing common services used by the demonstration firmware
- Config: FatFS, HAL, and demonstration kernel configuration files
- Core: demonstration kernel implementation files
- Demo: demonstration firmware management implementation, with the main routine, the interrupt handlers, and the MSP initialization
- Modules: applications implementation, with one folder per application
  - main_app[1]: main menu management. Refer to Section 4.2.2 .
  - calendar[1]: date, time, and alarm setting. Refer to Section 4.2.4 .
  - image viewer: bitmap images slide show. Refer to Section 4.2.5 .
  - audio[1]: audio record and playback. Refer to Section 4.2.6 .
  - thermometer and LDR: temperature and daylight intensity measurement. Refer to Section 4.2.7 .
  - low power[1]: low power modes. Refer to Section 4.2.8 .
  - file browser[1]: navigation through a folder tree. Refer to Section 4.2.9 .
  - help: mother board jumpers description. Refer to Section 4.2.10 .
  - ucpd: demonstrates how UCPD version PD3.0 is implemented in the context of STM32G0 devices

1. *Legacy part of the demonstration firmware providing a sub-set of the applications supported by the STM32072B-EVAL demonstration firmware; (STM32G0 and STM32F0 share the same feature footprint)*

Figure 9 illustrates further the organization of the Modules folder, which contains one sub-folder for each elementary demonstration plus one for the main menu of the demonstration application, and Demo folder, which is dedicated to software development environments:

Figure 9. Demonstration module folder organization



The Modules folder contains the following sub-folders:

- Inc: demonstration header files
- Src: demonstration implementation

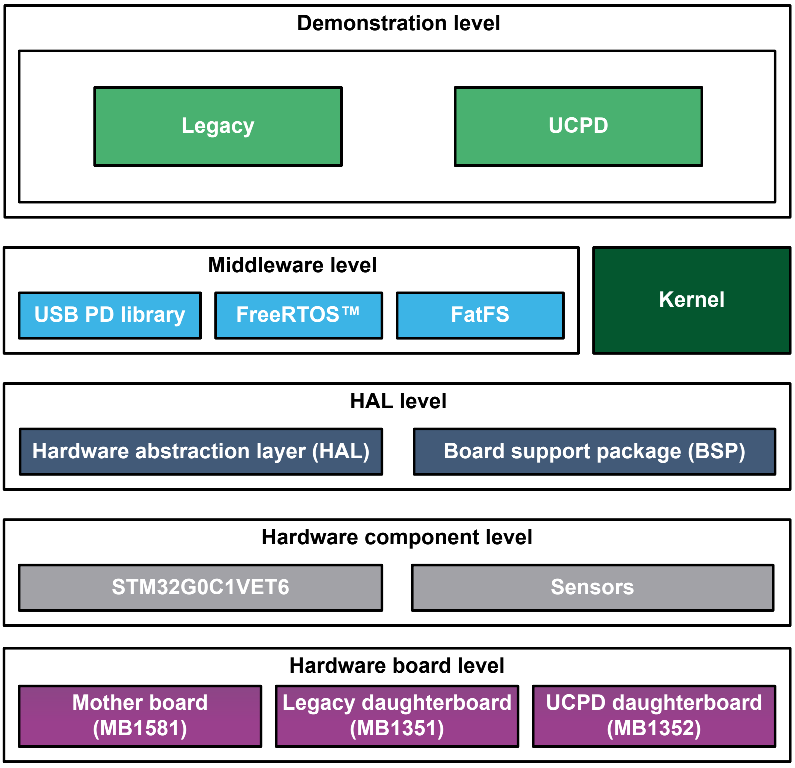
The Demo folder contains the following sub-folders dedicated to software development environments:

- EWARM: IAR Embedded Workbench®
- MDK-ARM: Keil® Microcontroller Development Kit
- STM32CubeIDE: IDE (integrated development environment) for STM32

3.2 Demonstration architecture overview

The top-level software architecture of the STM32G0C1E-EV demonstration firmware is represented in Figure 10. The software elements mentioned in this diagram are briefly depicted in dedicated sub-sections.

Figure 10. STM32G0C1E-EV demonstration firmware architecture



3.2.1 Legacy application

The legacy application is launched when the legacy daughterboard is connected to the mother board. It contains many applications that can be easily reused, such as RTC calendar, file system FAT implementation on microSD™ card, wave player using DAC and DMA peripherals, voice recorder using ADC and DMA peripherals, low power modes, temperature sensor interfacing, and TFT LCD.

3.2.2 UCPD

The UCPD application is launched when the UCPD daughterboard is connected to the mother board. It manages both USB-PD ports of the UCPD daughterboard which mainly consists of USB Type-C® connection/disconnection detection and USB Type-C® power contract negotiation. UCPD demo is also responsible for USB Type-C® pins reconfiguration when a USB Type-C® port is configured in DisplayPort (DP) alternate mode(1) or when the UCPD daughter board is used as USB Type-C® to USB-3.0 adapter. (1) DP (DisplayPort™) over USB Type-C® alternate Mode allows simultaneous transport of streaming video and USB data through a common USB Type-C® connector.

3.2.3 HAL level

The HAL level layer consists of the `stm32g0xx` HAL drivers together with the STM32G0C1E-EV board support package (BSP).

3.2.4 Kernel

The kernel is a suite of components providing high-level services to the applications to facilitate the application module integration and execution.

3.2.5 Middleware

Middleware provides the following modules:

- FreeRTOS: FreeRTOS™ open-source solution. The UCPD application is based on FreeRTOS™.
- FatFS: generic FAT file system module intended for small embedded systems. FatFS file control functions are used by the loader and the legacy applications to get access to the files stored in the microSD™ card.
- USBPD: USB-PD software stack

3.3 STM32G0C1VET6 resources

3.3.1 Peripherals

The following sections detail which peripherals of the STM32G0C1VET6 microcontroller are used by the legacy and UCPD applications.

3.3.1.1 Peripherals used by the legacy application

Figure 11. STM32G0C1VET6 peripherals used by the legacy application

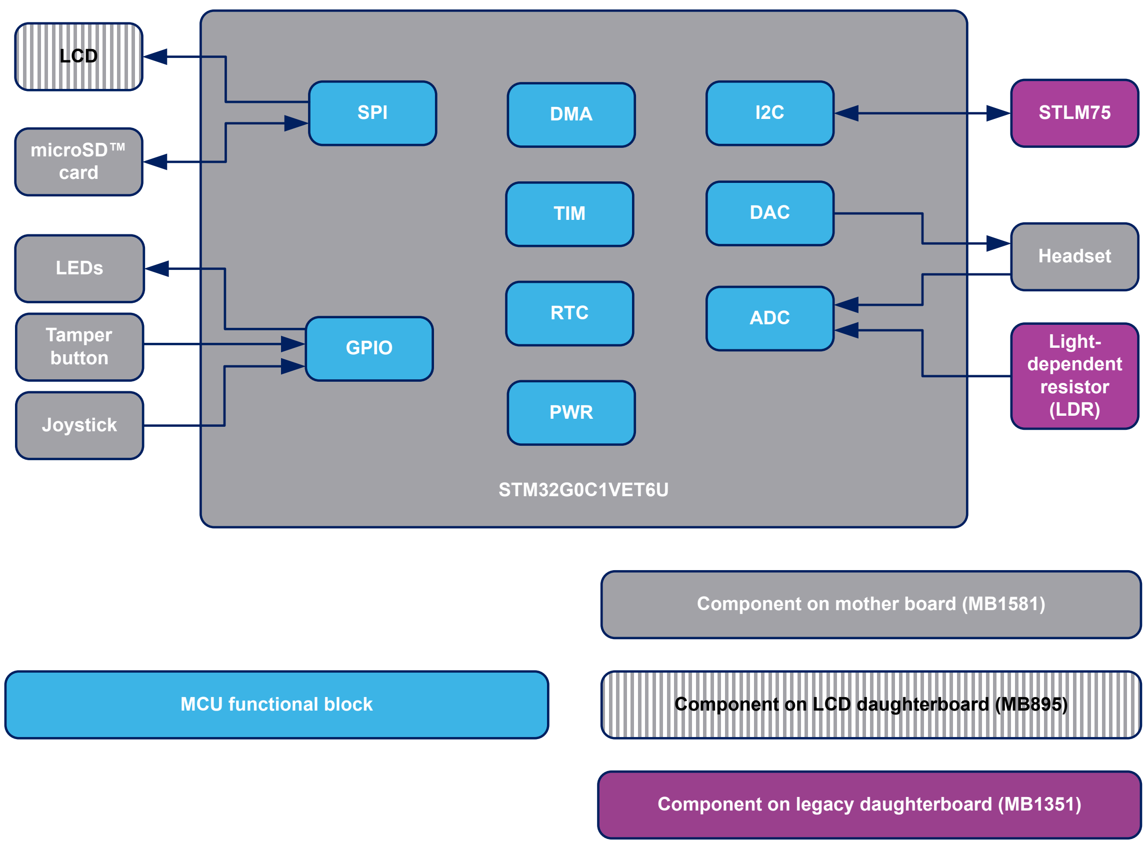


Table 3. STM32G0C1VET6 peripherals used by the legacy application

Peripheral	Usage description
SPI	microSD™ card and LCD are controlled through SPI1. Read accesses to the microSD™ card are performed to retrieve the bitmaps to display on the LCD screen. microSD™ card is also read accessed by the audio playback application and write accessed by the audio record application. Write accesses to the LCD are performed to display strings and bitmaps during the legacy application execution.
GPIO	GPIOs are used to toggle the mother board LEDs when an error is detected during the legacy application.

Peripheral	Usage description
	Also, the GPIO pins connected to the joystick and the USER button are used to interact with the legacy application (menu navigation).
ADC	ADC channel 1 is connected to the light-dependent resistor (LDR) supported by the legacy daughterboard. It is used by the thermometer/LDR application. ADC channel 6 is connected to the microphone input (external headset connected to the audio jack). It is used by the audio record application.
DAC	DAC outputs are connected to the left and right channels of the stereo audio jack. DAC peripheral is used by the audio playback application.
RTC	RTC peripheral is used by the calendar application to set the time, date, and alarm. It is also used by the power application to set the wakeup alarm.
DMA	DMA transfers are used to transfer audio samples from the audio playback buffer to the DAC data register or to transfer audio samples from the ADC data register to the audio record buffer.
TIM	TIM6 is used by both the audio playback and audio record applications to trigger the DAC/ADC conversions at the required audio sampling rate.
PWR	The PWR peripheral is used by the power application to enter Power or Standby mode. MCU exit Low-power mode either by pressing the joystick selection key (mapped on WKUP1) or when the programmed RCT alarm expires.
I2C	I2C1 is used by the thermometer/LDR application to control the STLM75 component (digital temperature sensor).

3.3.1.2 Peripherals used by the UCPD application

Figure 12. STM32G0C1VET6 peripherals used by the UCPD application

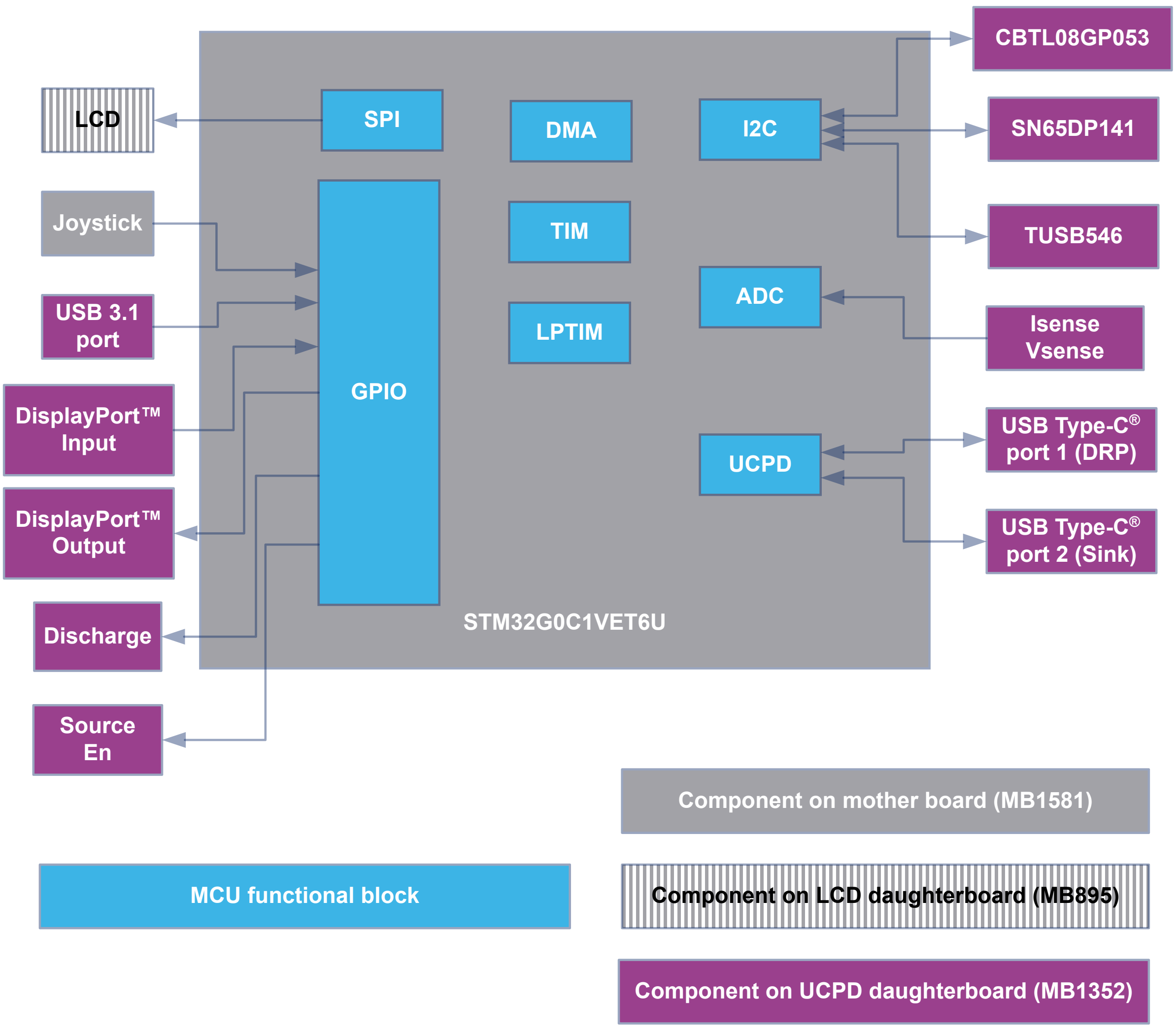


Table 4. STM32G0C1VET6 peripherals used by the UCPD application

Peripheral	Usage description
SPI	LCD is controlled through SPI1. Write accesses to the LCD are performed to display strings and bitmaps during the UCPD application execution.
GPIO	The GPIO pins connected to the joystick are used to interact with the UCPD application, such as menu navigation: • One GPIO pin is used to detect USB3.1 cable connect/disconnect. • One GPIO pin is used to detect the HPD (hot-plug-detect) signal coming from a DisplayPort™ input connector. • One GPIO pin is used to notify the presence of a Source Device to the Sink device through the HPD pin of the DisplayPort™ output connector. • One GPIO pin is used to enable the V_{BUS} on the USB Type-C® port 1 (DRP).

Peripheral	Usage description
	• One GPIO pin is used to control the USB V_{BUS} discharge mechanism on the USB Type-C® port 1 (DRP).
I2C	I2C1 is used to control the following components of the UCPD daughterboard (MB1352): • CBTL08GP053: USB Type-C®, multiplexer, switch, USB 3.1, DisplayPort™ • SN65DP141: DisplayPort™ Linear Redriver • TUSB546: USB Type-C® DP ALT Mode Linear Redriver Crosspoint Switch
ADC	• ADC channel 9 is used to measure the voltage level on the V_{BUS} line of the USB Type-C® port 1 (DRP). • ADC channel 11 is used to measure the current level on the V_{BUS} line of the Type-C port 1 (DRP). • ADC channel 3 is used to measure the voltage level on the V_{BUS} line of the Type-C port 2 (Sink). • ADC channel 16 is used to measure the current level on the V_{BUS} line of the Type-C port 2 (Sink). • ADC channel 15 is used to measure the DCDC voltage level[1].
UCPD	UCPD is used to manage the USB Type-C® communication over the USB Type-C® ports.
DMA	DMA is used for ADC conversions.
TIM	TIM6 is used for USB 3.1 detect and DisplayPort™ hot-plug detect debouncing.
LPTIM	LPTIM1 is used to generate the PWM signal controlling the voltage level on the V_{BUS} line of the USB Type-C® port 1 (when the duty cycle of the PWM signal is 0%, the V_{BUS} voltage level is 15 V; when the duty cycle is 10%, the V_{BUS} voltage level is 5 V).

1. ADC channel 15 is also used at the beginning of the execution to detect the type of daughterboard (MB1351 or MB1352).

3.3.2 Interrupts

Table 5 shows all the external interrupts used by the demonstration.

Table 5. STM32G0C1VET6 demonstration interrupt usage

Interrupt	Usage description	Legacy	UCPD
EXTI line 0	Joystick SELECT (interrupt mode, rising edge)	YES	YES
EXTI line 2	Joystick UP (interrupt mode, rising edge)	YES	YES
EXTI line 3	Joystick DOWN (interrupt mode, rising edge)	YES	YES
EXTI line 7	Joystick RIGHT (interrupt mode, rising edge)	YES	YES
EXTI line 8	Joystick LEFT (interrupt mode, rising edge)	YES	YES
EXTI line 9	microSD™ card detect (interrupt mode, rising and falling edge)	YES	NO
EXTI line 13	Tamper (interrupt mode, rising edge)	YES	YES
DMA1 Channel1	DAC/ADC conversions completion	YES	YES
ADC1_COMP	ADC analog watchdogs	NO	YES
UCPD	UCPD related interrupts, such as Rx message received, Rx ordered set detected, and Transmit message sent	NO	YES

3.3.3 External memory organization

The legacy part of the STM32G0C1E-EV demonstration is based on the FatFS embedded free FAT file system. The file system is needed by the legacy application to read all media information from the on-board microSD™ card memory. The microSD™ memory card is organized in two subdirectories:

- `STFILES`: this directory contains all the bitmaps required by the demonstration firmware.
- `USER`: this is a user folder. The user may add his files here to be played inside the demo menus (pictures and `.wav` files). This folder is used only by the file browser (refer to File browser application), image viewer (refer to Image viewer application), and wave player applications (refer to Wave player). This folder also contains the voice recorded wave file `rec.wav`, which is created when the voice recording application is run.

Note: The microSD™ memory card provided with STM32G0C1E-EV is already programmed with the media files to run the demonstration. These files are also available within the demonstration firmware package in the `Projects\STM32G0C1E-EV\Demonstrations\Binary\SD_card\` *folder.*

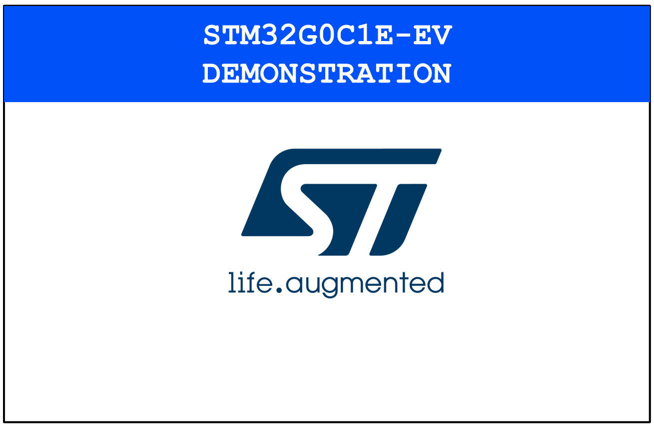
4 Running the demonstration

4.1 Demonstration startup

4.1.1 Normal processing

After a board reset, at demonstration startup, the welcome screen is displayed and the STMicroelectronics logo appears on the LCD as illustrated in Figure 13.

Figure 13. Welcome screen

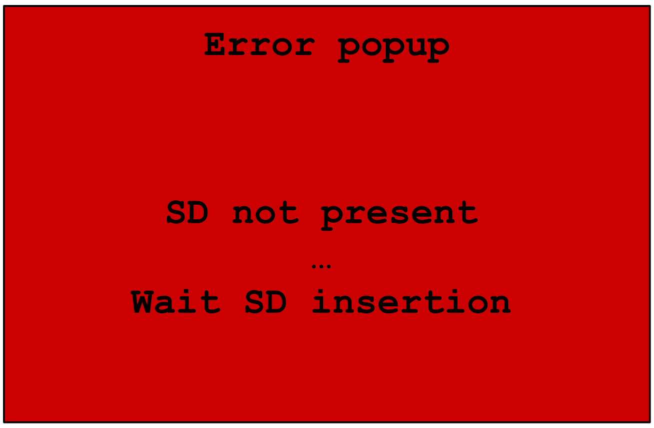


Then the demonstration program launches the daughterboard recognition sequence and executes the legacy or the UCPD application according to the detected daughterboard. When the MB1352 is plugged into the mother board, the UCPD application is executed. In all other cases, the legacy application is executed. The legacy application first checks the presence of the microSD™ card in the CN11 connector.

4.1.2 Error cases

If no card is detected, the demonstration does not start and the message shown in Figure 14 is displayed on the LCD screen. The demonstration waits for the microSD™ card insertion to proceed.

Figure 14. microSD™ card detection error



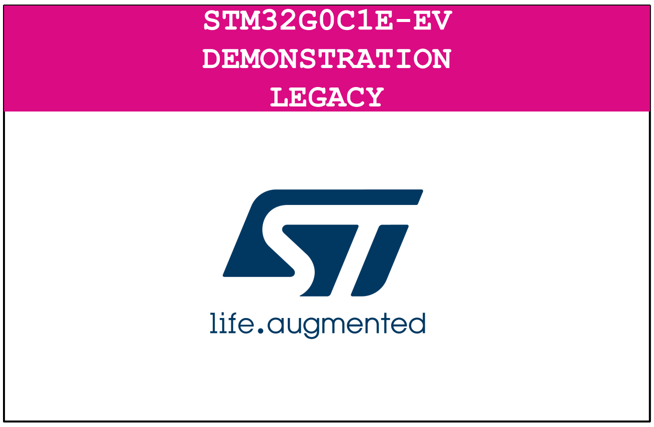
Note: Hot-plug detection of the daughterboard is not supported. If the user changes the daughterboard while the demonstration firmware is being executed, he must press the reset button of the mother board to restart execution from the beginning.

4.2 Legacy application

4.2.1 Overview

The purpose of the legacy application is to bring out the capabilities of the microcontroller and Evaluation board peripherals. It runs only on the MB1581 STM32G0C1E-EV mother board mounted with the MB1351 legacy daughterboard.

Figure 15. Legacy demonstration welcome screen



4.2.2 Main menu

The main menu is displayed in the form of a set of icons. It shows all submenus on the same screen. Menu navigation is achieved by pressing the joystick keys:

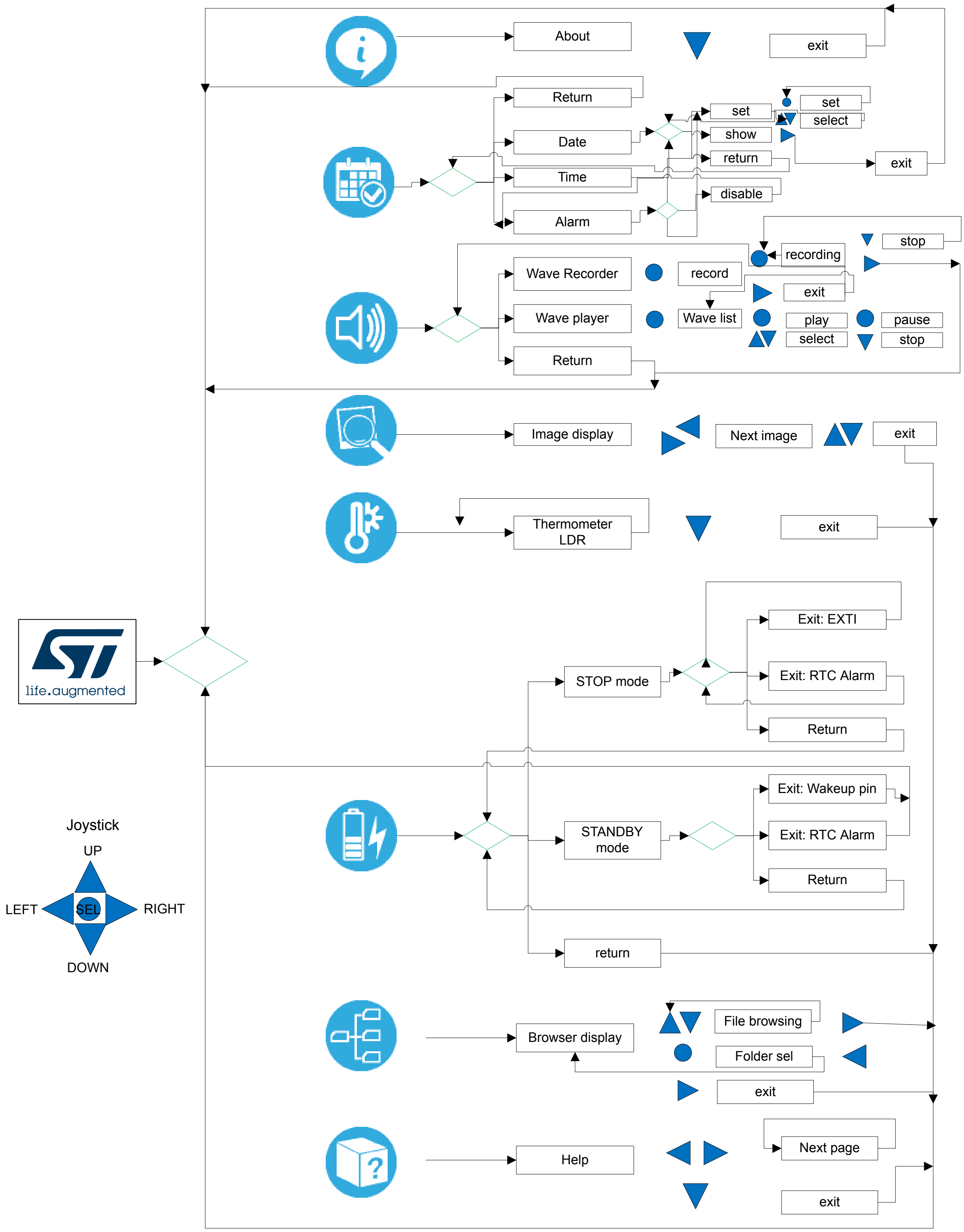
- UP, DOWN, LEFT, or RIGHT key to browse among submenus
- SELECT to select and enter a submenu

When a submenu is selected, the submenu title, which is the name of the attached application, is mentioned at the top of the LCD (refer to Figure 16).

Figure 16. Main menu

4.2.3 Navigation

Figure 17. **Demonstration menu structure**



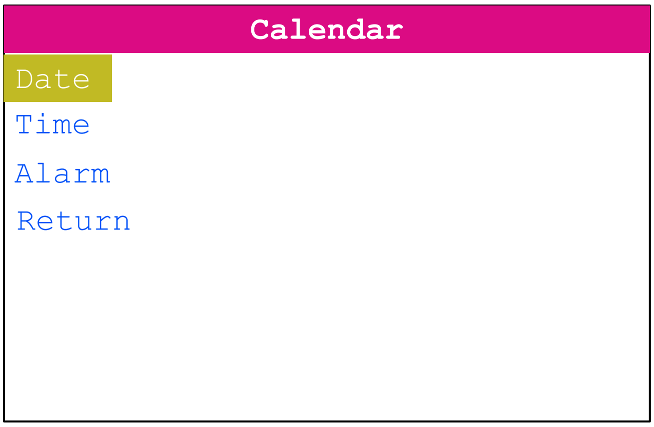
4.2.4 Calendar application

The STM32G0C1VET6 features a real-time clock (RTC), which is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar, two programmable alarm interrupts, and a periodic programmable wake-up flag with interrupt capability.

4.2.4.1 Calendar submenu

The calendar submenu is used to select the date, time, and alarm settings, or to return to the main menu.

Figure 18. Calendar submenu



4.2.4.2 Date setting

The date setting submenu allows the user to set or see the current date, and get back to the calendar submenu.

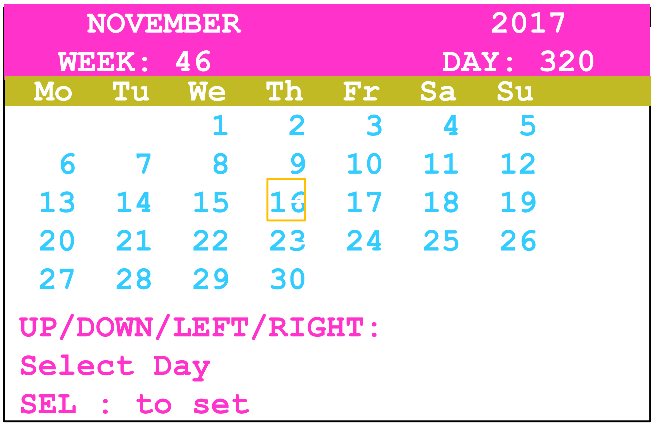
Figure 19. Date submenu

Setting the date is performed in three consecutive steps:

1. Select the current year with the UP and DOWN joystick keys. The current year is displayed on the right side of the upper line. Press the SELECT joystick key when the desired year is reached.
2. Select the month with the UP and DOWN joystick keys. The current month is displayed on the left side of the upper line. Press the SEL joystick key when the desired month is reached.

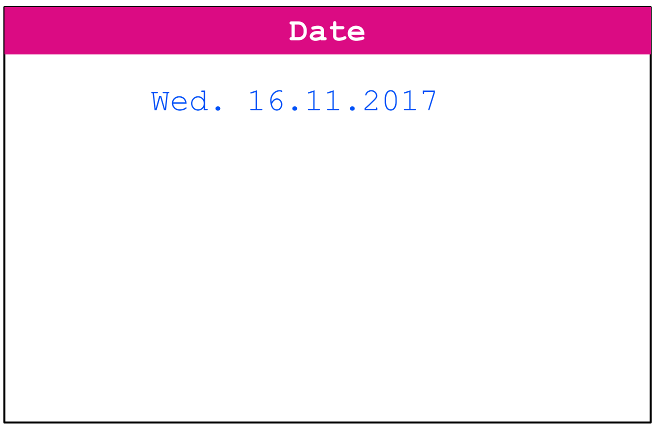
3. Select the day number with the UP, DOWN, LEFT, and RIGHT joystick keys. The currently selected date is framed. Press the SEL joystick key when the desired day is reached.

Figure 20. Setting the day



The current date can be watched through this menu:

Figure 21. Consulting the date



4.2.4.3 *Time setting*

The time setting submenu allows setting or consulting the current date, and getting back to the calendar submenu.

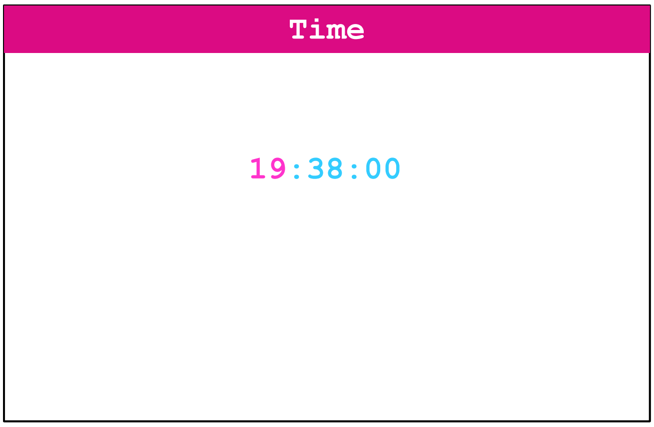
Figure 22. Time submenu

The time is set by selecting hours (in the 24-hour format), minutes, and seconds:

- Use the LEFT or RIGHT joystick keys to switch among hours, minutes, and seconds. The current selection is highlighted in pink.
- Press the UP or DOWN joystick keys to increase or decrease the selected number. Keep continuously pressing for incrementing or decrementing faster.

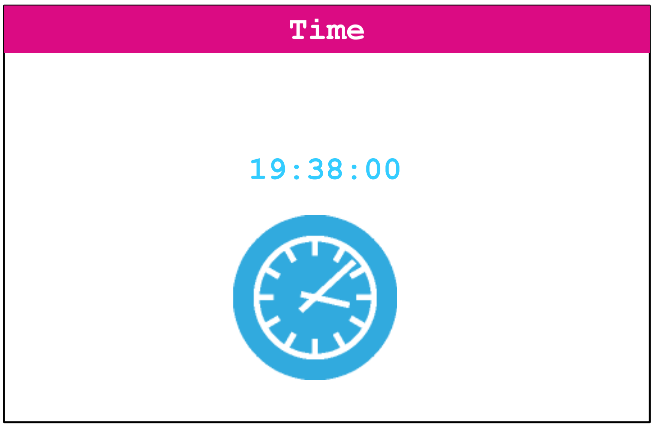
When the desired values are selected, press the SELECT joystick key to choose them as the current time.

Figure 23. Time setting



The current time can be consulted through this menu:

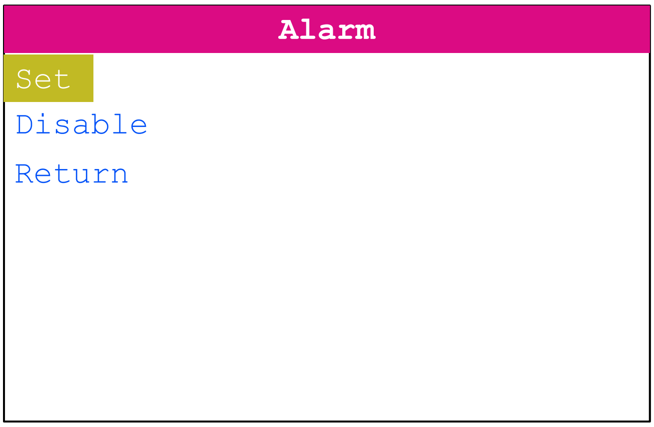
Figure 24. **Consulting the time**



4.2.4.4 *Alarm setting*

The alarm setting submenu allows the user to set or disable the A alarm, and get back to the calendar submenu.

Figure 25. **Alarm submenu**



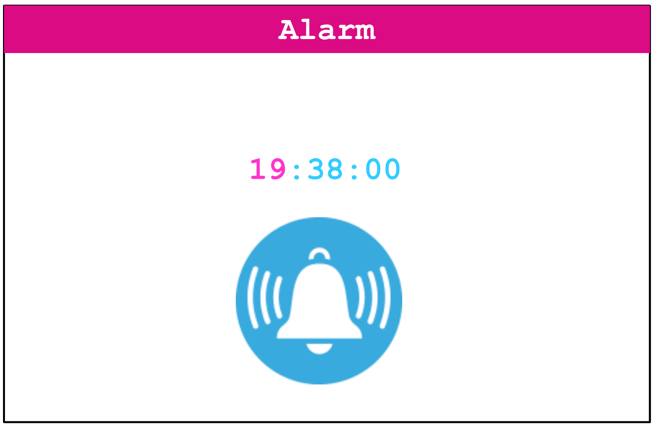
The A alarm is set by selecting hours (in the 24-hour format), minutes, and seconds:

- Use the LEFT or RIGHT joystick keys to switch among hours, minutes, and seconds. The current selection is highlighted in pink.
- Press the DOWN or UP joystick keys to increase or decrease the selected number. Keep continuously pressing for incrementing or decrementing faster.

When the desired values are selected, press the SEL joystick key to set the A alarm accordingly.

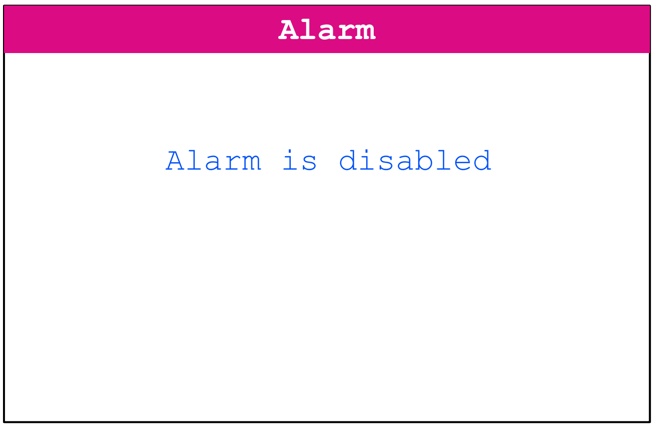
When the current time reaches the value set for the A alarm, the alarm is triggered , making all LEDs blink alternatively for a few seconds. The alarm is configured to be triggered every day at the same time. If the user wants to disable it, he must do it manually through the disable option of the alarm submenu. Upon selection of the [**Disable**] option, the alarm is disabled.

Figure 26. Alarm setting



When selecting the Disable submenu, Alarm A is disabled.

Figure 27. Alarm disable



4.2.5 Image viewer application

The image viewer submenu is used to demonstrate LCD control performance using the embedded SPI interface. The application performs a slideshow of the bitmap files stored in the USER folder of the microSD™ card. Only the `.bmp` files having the following properties are considered:

- Bit depth: 16 bits (RGB)
- Size: 240×320

Press the RIGHT or LEFT keys of the joystick to display the next or previous bitmap on the list. Quit the application by pressing the DOWN key of the joystick and then return to the main menu.

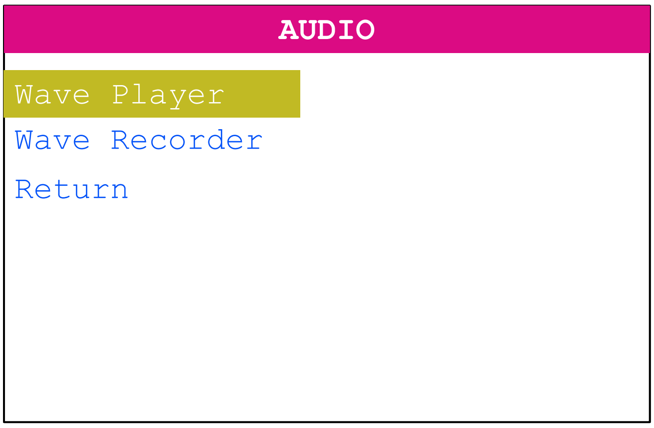
4.2.6 Audio application

The audio application provides means to playback `.wav` files and record voice using STM32G0C1VET6 DAC and DAC peripherals connected to the headphone plugged into the CN15 connector.

4.2.6.1 Audio submenu

The audio submenu is used to select among the wave player or wave recorder applications.

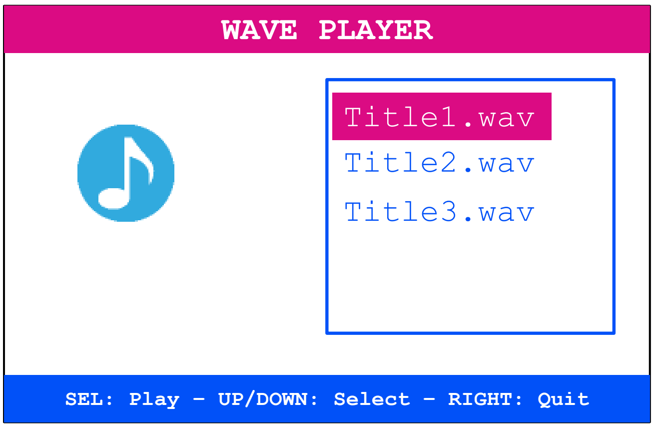
Figure 28. **Audio submenu**



4.2.6.2 Wave player

When the [**Wave Player**] menu option of the audio submenu is selected, the playlist is displayed on the LCD screen. The playlist consists of listing the `.wav` files in the USER folder of the microSD™ card. The playlist size is limited to four titles.

Figure 29. **Audio playlist**



Use the UP and DOWN keys of the joystick to browse among available titles. Press the SELECT key of the joystick to play the selected title. Figure 30 represents the aspect of the LCD screen when the wave file is being played.

Figure 30. Audio playing

Pause the playback by pressing the SELECT key of the joystick (Press the SELECT key again to resume playback). Press the DOWN key of the joystick to stop the playback (Press the SELECT key again to restart the playback from the beginning). Figure 31 represents the aspect of the LCD screen when playback is paused.

Figure 31. Audio paused

Figure 32 represents the aspect of the LCD screen when playback is stopped.

Figure 32. **Audio stopped**

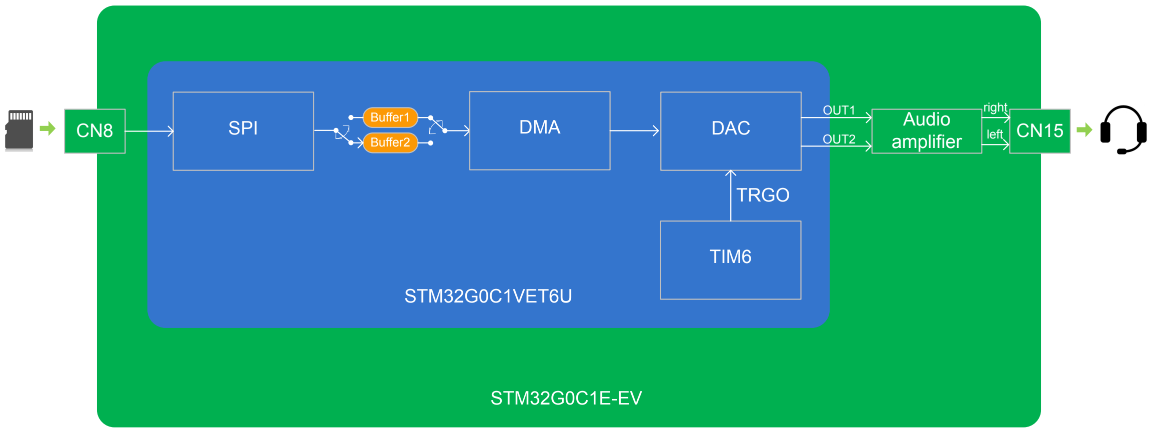
Note: The audio files provided within this package are based on free-music downloads from the danosongs.com website.

Wave player implementation overview

Access to the selected wave file is done through the FatFS interface. Audio samples are transferred to the internal SRAM by blocks (512 bytes each) using the SPI interface. Audio samples go through a flip-flop buffer, meaning that one half of the buffer is fed by the SPI while the other half is read by the DMA to feed the DAC. The TIM6 triggers the DAC conversions to generate the wave signal at the desired sample rate. The DAC is used in conjunction with the DMA controller to reduce the Cortex CM0+ workload.

The wave player data path is shown in Figure 33.

Figure 33. **Wave player data path**



4.2.6.3 *Wave recorder*

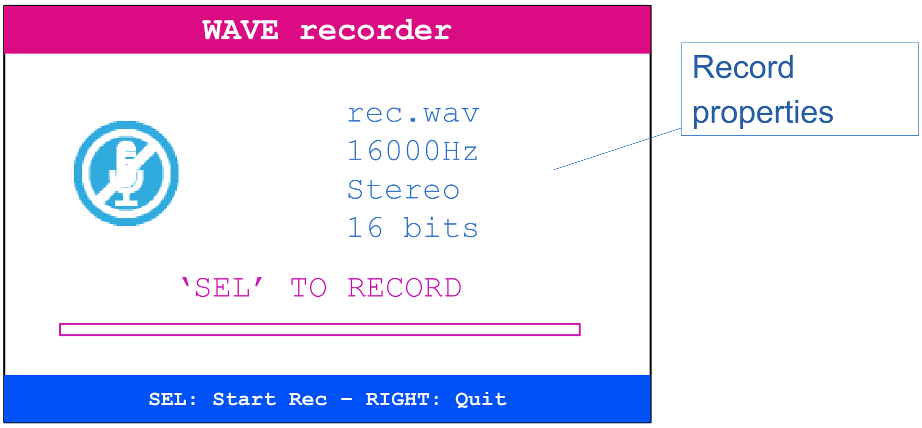
When the [**Wave Recorder**] option of the audio submenu is selected, the LCD shows the LCD audio record page (refer to Figure 34). When recording is started, the `rec.wav` file is created in the `USER` folder of the microSD™ card. If this file already exists, its content is erased and the file is considered as a new empty file.

The recorder audio file has the following properties:

- Sampling rate: 16000 samples/s
- Channels: stereo (left = right)

- Resolution: 16 bits/sample

Figure 34. **Wave recorder start**



Start recording by pressing the SELECT key of the joystick. The record duration is limited to 30 seconds. During the recording, the progress bar allows an estimation of the record time budget consumption:

Figure 35. **Wave recording**

Press the SELECT key of the joystick to stop the recording (refer to Figure 36). Pressing the SELECT key again starts a new recording.

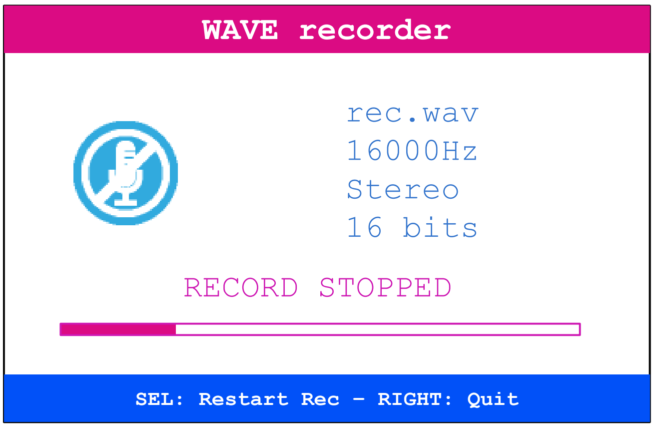
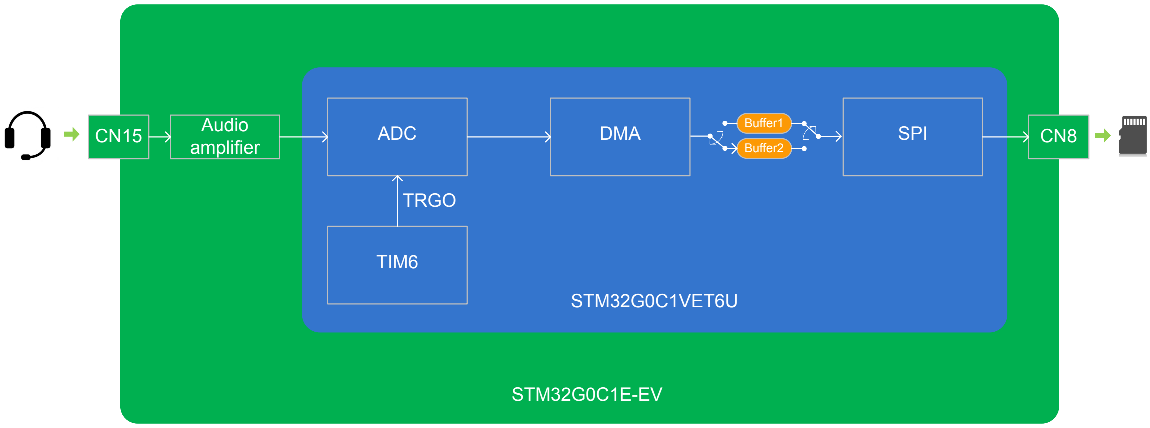


Figure 36. Wave recorder stopped

Wave recorder implementation overview

The ADC is connected to the headphone through a microphone amplifier. The ADC converts the incoming audio stream into audio samples at a 16000 Hz rate. ADC conversions are triggered by TIM6 to reach the target sample rate. Converted data go through a flip-flop buffer: one half of the buffer is fed by the DMA while the other half is copied to the `rec.wav` file using the FatFS interface (the microSD™ card is accessed through the SPI). Audio samples are written to the recorded file by blocks (512 bytes each).

Figure 37. Wave recorder data path



4.2.7 Thermometer/LDR application

The STM32G0C1VET6 microcontroller has two embedded I²C peripherals for connection to any device supporting the I²C protocol.

In this demonstration, an STLM75 (or a compatible device) I²C temperature sensor mounted on the MB1351 daughterboard is used to capture the external temperature (-40°C to +125°C).

A light-dependent resistor (LDR) is also present on the MB1351 daughterboard (connected to ADC1 peripheral) and captures luminosity (0% to 100%).

Select the [**Thermometer/Light**] submenu of the demonstration menu by pressing the SELECT push-button of the joystick. The temperature value is displayed in Celsius and Fahrenheit degrees and the light value is displayed in percent as shown in Figure 38.

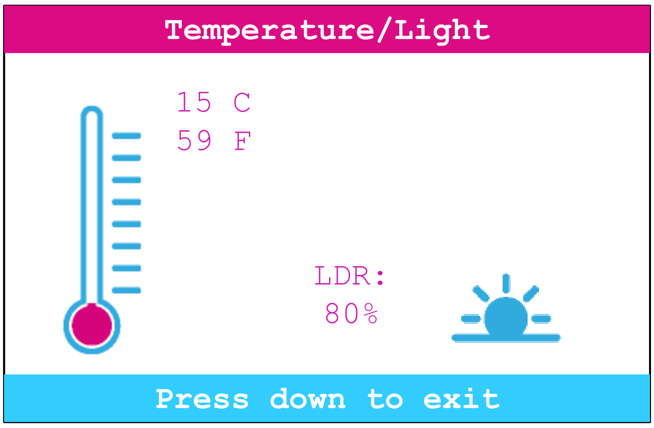


Figure 38. Temperature/Light screen

The thermometer/LDR application requires hardware resources of the MB1351 legacy daughterboard.

In case the MB1351 board is not mounted on the MB1581 mother board, an error is displayed as shown in Figure 39.

Figure 39. Temperature/Light (missing legacy daughterboard)

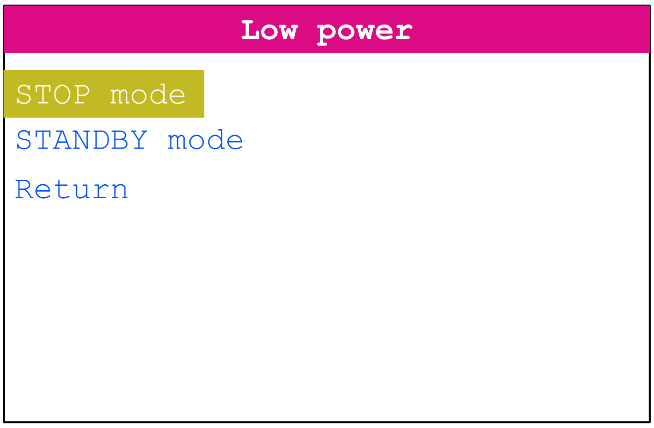
4.2.8 Low-power mode application

The STM32G0C1VET6 microcontroller provides different operating modes in which the power consumption is reduced. The purpose of this menu is to show the behavior of the microcontroller in different low-power modes. The Stop and Standby modes are used as examples.

4.2.8.1 Low-power mode submenu

The Low-power mode submenu is used to select among low-power modes Stop and Standby or return to the main menu.

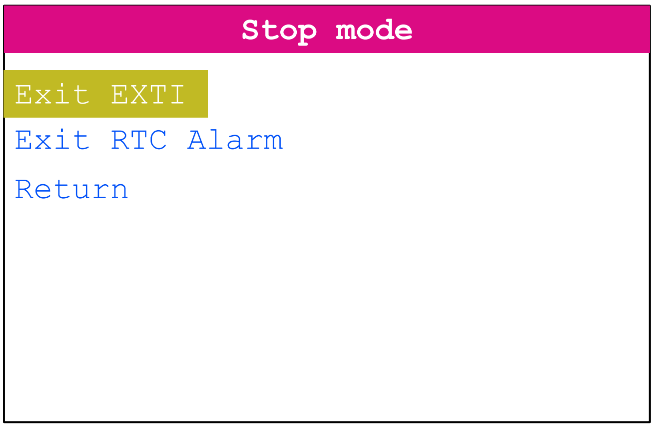
Figure 40. Low-power submenu



4.2.8.2 Stop mode

The Stop mode submenu (refer to Figure 41) allows the user to put the STM32G0C1VET6 in Stop mode with its low-power regulator on. It is possible to select between the EXTI (WFI) or RTC alarm (WFI) to exit from the Stop mode.

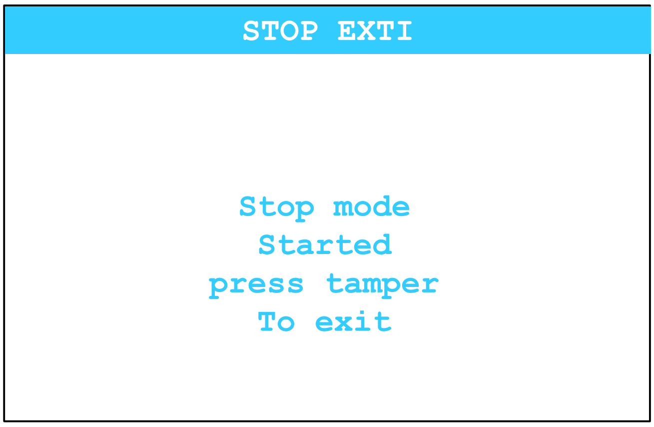
Figure 41. Stop mode submenu



EXTI (WFI)

The tamper button is configured as an external interrupt. Pressing SELECT enters Stop mode. When the MCU is in Stop mode, the message shown in Figure 42 is displayed on the LCD. MCU remains in Stop mode until the tamper push-button is pressed. The system clock is then set to 56 MHz and the application resumes execution.

Figure 42. Exit from Stop mode - EXTI (WFI)



RTC alarm (WFI)

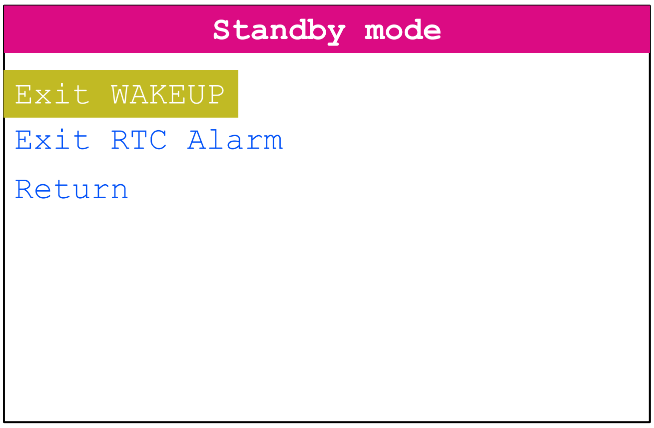
When selecting this submenu, the user sets the alarm delay when MCU has to exit from Stop mode. Figure 43 shows how to set the wake-up time, using UP, DOWN, LEFT, and RIGHT joystick keys. The wake-up time is validated after a press on SELECT, and MCU enters then in Stop mode. RTC Alarm wakes up MCU from Stop mode after programmed time has elapsed. The system clock is then restored to 56 MHz and the application resumes execution.

Figure 43. Exit from Stop mode - RTC alarm (WFI)

4.2.8.3 *Standby mode*

The Standby mode submenu allows the user to put the STM32G0C1VET6 in Standby mode. It is possible to select between the wake-up pin and the RTC alarm to exit from the Standby mode.

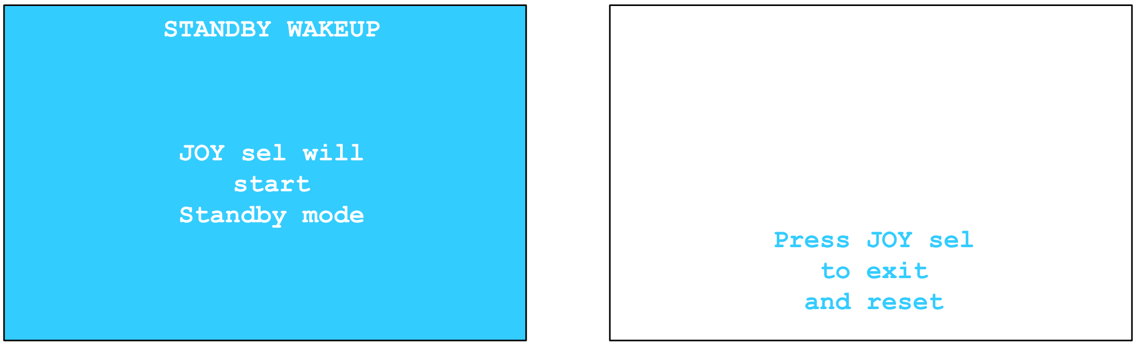
Figure 44. **Standby mode**



WAKEUP

Pressing again SELECT joystick key enters the Standby mode. Doing this, the button is configured to wake up the MCU from the Standby mode. Once the EXIT Wakeup submenu is selected, the system enters Standby mode pressing the SELECT key of the joystick . When the SELECT key of the joystick is pressed again, the MCU exits Standby mode. Then, the system reset is generated, and the legacy application execution restarts from the beginning (main menu).

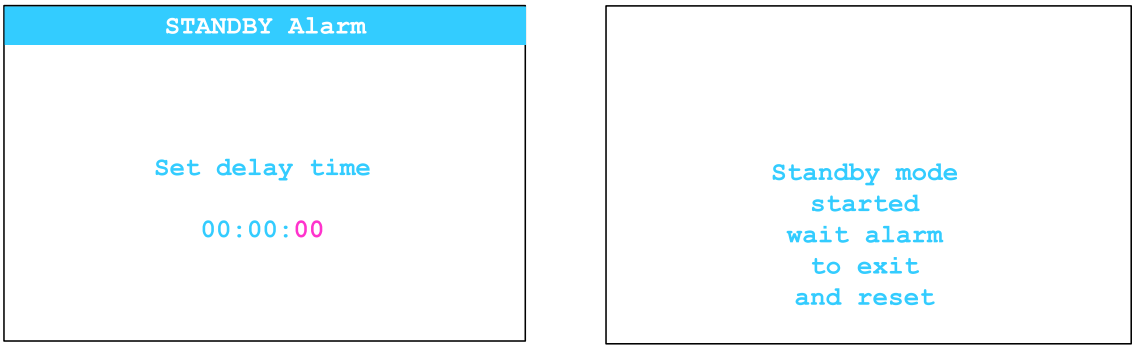
Table 6. **Exit from Standby mode - Wake-up**



RTC alarm

When selecting this submenu, the user sets the alarm delay when MCU has to exit from Standby mode. The RTC alarm wakes up the MCU from the Standby mode after the programmed time has elapsed. The user sets the wakeup time, using UP, DOWN, LEFT, and RIGHT joystick keys. Once the wake-up time is validated with a press on SELECT, MCU enters in Standby mode. After the programmed timing has elapsed, the system exits the Standby mode and the program execution resets.

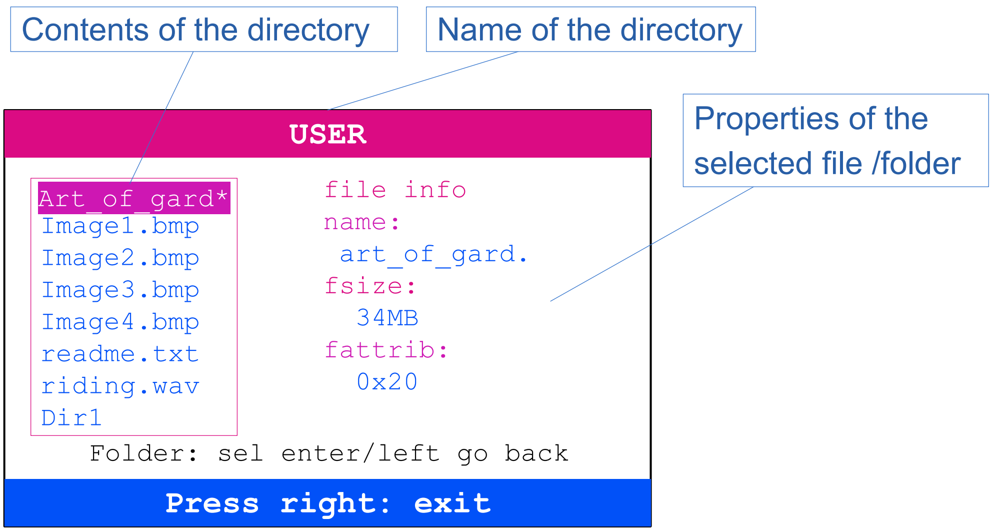
Table 7. **Exit from Standby mode - RTC alarm**



4.2.9 File browser application

The file browser application demonstrates the possibility to have access to the microSD™ card file system through the FatFS interface. It can be used to navigate through the `USER` folder of the microSD™ card and display its contents and subfolders as shown in Figure 49.

Figure 45. **File browser**



Use the UP and DOWN keys of the joystick to selects an item. Press the SELECT key of the joystick to enter a sub-directory. Press the LEFT key of the joystick to quit a sub-directory and go back to the parent directory.

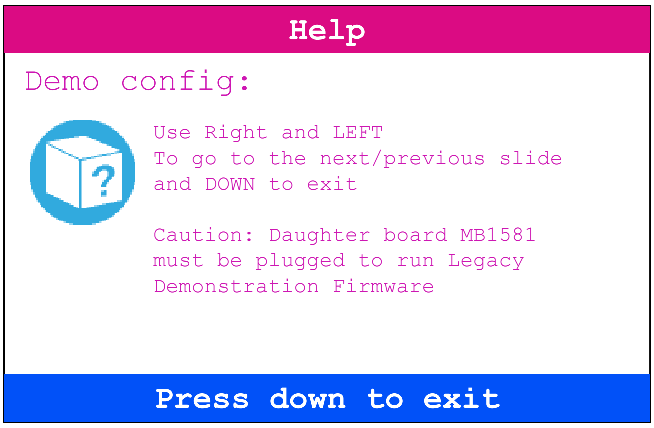
When a file is selected, the following fields of the FatFS file information structure are displayed on the right side of the LCD screen:

- Filename (only the first 12 characters are displayed)
- File size (in bytes, Kbytes, or Mbytes)
- File attributes

4.2.10 Help application

Help submenu shows different information regarding the MB1581 motherboard. Use the RIGHT or LEFT buttons to see the next or previous slide. Figure 50 shows the first slide of the Help submenu.

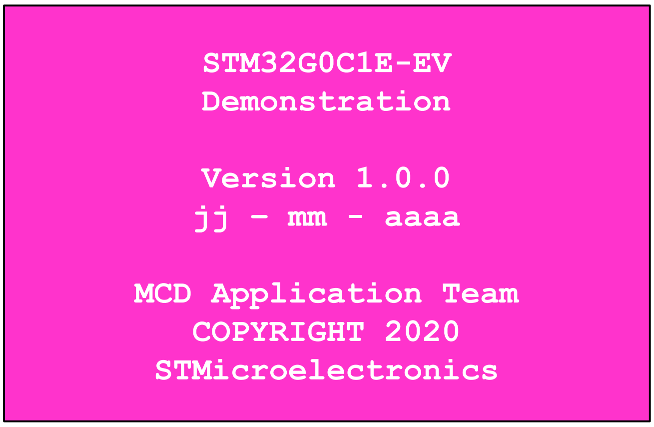
Figure 46. Help



4.2.11 About application

The About submenu shows the version and date of the STM32G0C1E-EV demonstration firmware. When the About is selected, the message shown in Figure 51 is displayed on the LCD screen. Press the DOWN key to return to the main menu.

Figure 47. About submenu



4.3 UCPD application

4.3.1 Warning

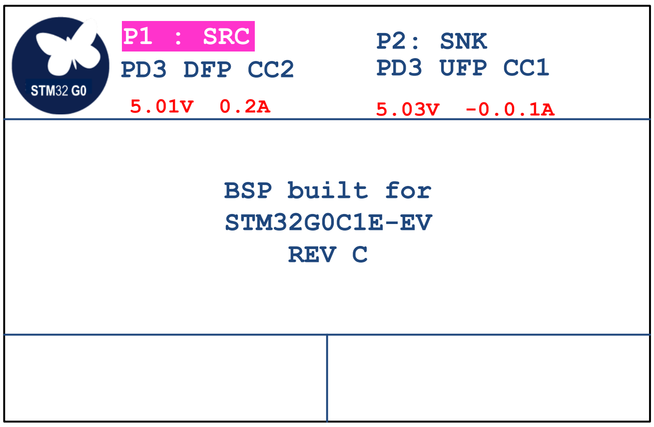
Running the UCPD demonstration may cause the power delivered to damage the device plugged onto the STM32G0C1E-EV Evaluation board. STMicroelectronics may not be held responsible for all issues encountered.

4.3.2 Hardware checks

Before running the UCPD demonstration, make sure that the hardware configuration is correct (refer to Hardware settings).

Depending on the board revision (RevA or RevB), check that the correct firmware is flashed:

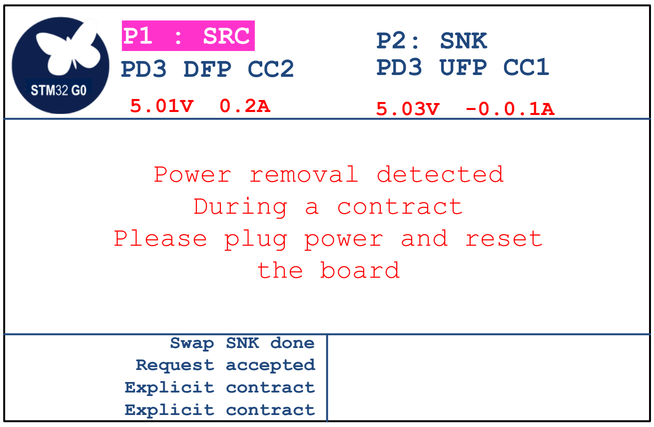
Figure 48. **UCPD firmware revision compatibility**



4.3.3 Emergency state

In case of a power issue, the demonstration enters the emergency state:

Figure 49. **UCPD emergency state screen**



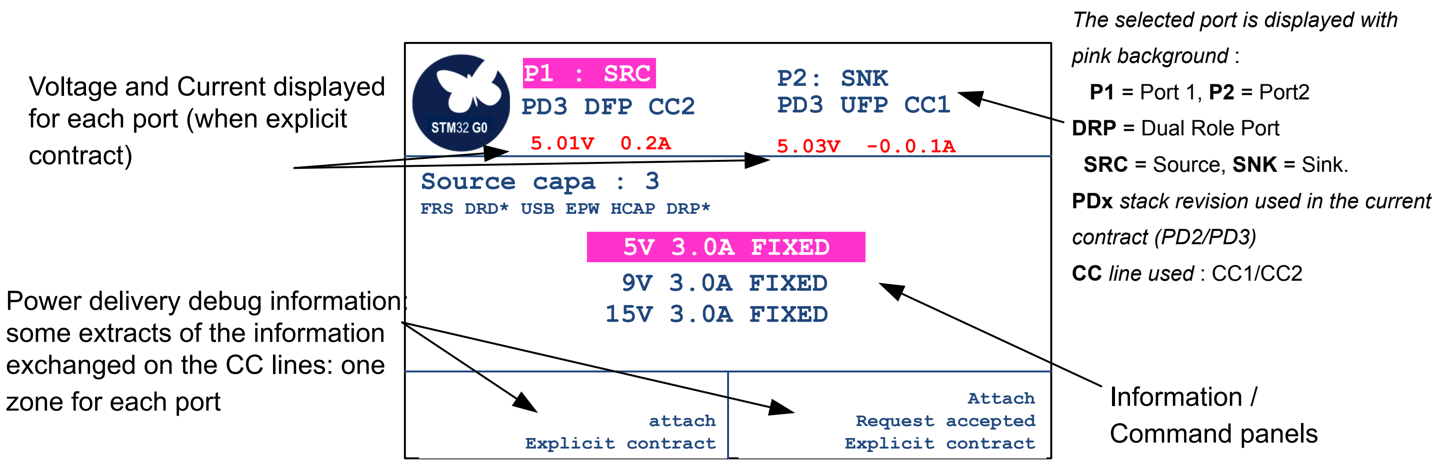
To exit from this state, the user must check the power supply and reset firmware.

4.3.4 Overall presentation

Plug the dual-role capability Port 1 or the sink-only Port 2 to an external device, or each other.

The information on the LCD screen is presented as follows:

Figure 50. UCPD demonstration main screen



4.3.5 Navigation in the menus

The tamper button is used to select the port to get control or to get information.

The joystick is used to navigate in the different USBPD demonstration panels:

- LEFT / RIGHT: selects the information / command panels.
- UP / DOWN: when possible, scrolls the information / action
- SELECT: executes the selected action

4.3.6 Available panels

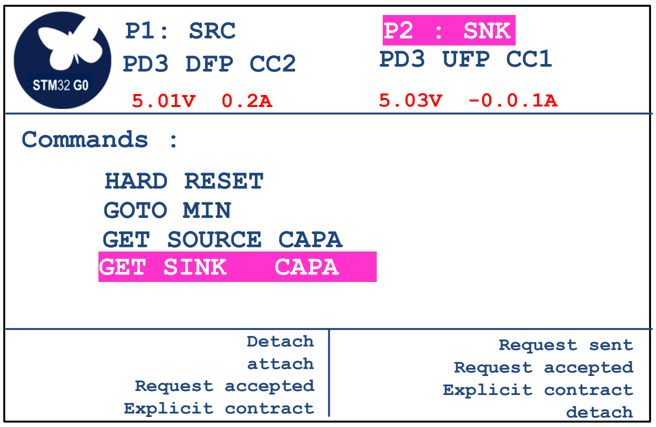
One command panel and three information panels are available.

Command panel

Use the joystick up and down keys to scroll across the following available commands (refer to Figure 55):

- Hardware reset
- Goto Min voltage
- Get Source Capabilities
- Get Sink Capabilities
- Data Role Swap
- Power Role Swap
- Software reset
- Get Extended capabilities

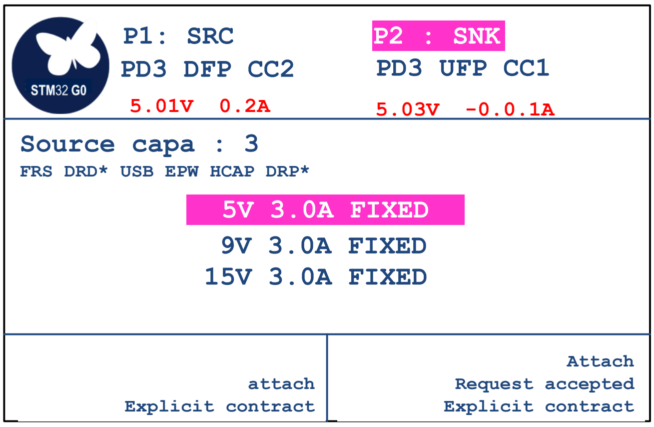
Figure 51. UVPD - Available commands



Source capabilities / Sink capabilities

This information panel depends on the role of the port. Figure 56 shows an example of source capabilities:

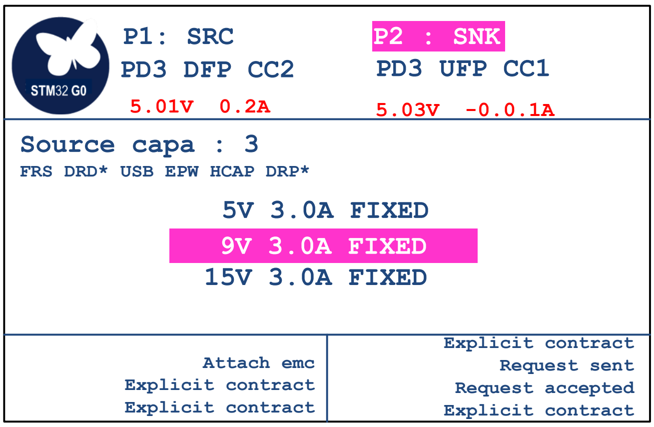
Figure 52. UCPD - Source/sink capability



- In this example, the source proposes 3 PDOs (Power Data Objects; refer to section *6.4.1* in [1]): 5 V, 9 V, and 15 V.
- The Power Delivery available features are also displayed. A star (*) next to the capability indicates that the feature is supported among:
  - FRS: Fast Role Swap support activated
  - DRD: Dual-Role Data (possibility to swap data role)
  - USB: USB available
  - EPW: Externally powered
  - HCAP: High Capabilities support
  - DRP: Dual-Role Power capability (possibility to change power role)

Press the up and down joystick keys to reach a power profile and select the profile to send a power request as illustrated by an example in Figure 57.

Figure 53. UCPD - Power profile selection

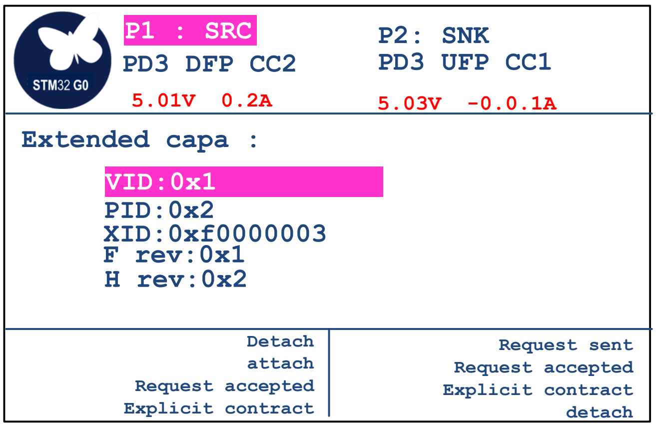


- Pressing the down key of the joystick once reaches the 9 V profile.
- Selecting the 9 V profile with the SELECT button of the joystick sends a request to switch to 9 V from the selected port (Port 2) to the source.
- The debug info panel shows `Request sent / Request accepted`.
- The power switches to 9 V as indicated by the port current/voltage information line.

Extended capabilities

Figure 58 shows the extended capabilities available when PD3 is supported (scroll with the joystick).

Figure 54. UCPD - Extended capabilities



- VID: Vendor ID (given by USB-IF)
- PID: Product ID (given by USB-IF)
- XID: Value provided by the USB-IF assigned to the product
- F rev: Firmware revision
- H rev: Hardware version

Press the down key of the joystick to display more information, which is not shown in Figure 58:

- V reg: the Voltage Regulation field contains bits covering Load Step Slew Rate and Magnitude.
  Refer to section *7.1.12.1* in [1] for details.
- Htime: the Holdup Time field must contain the source's holdup time.
  Refer to section *7.1.12.2* in [1] for details.
- Compliance: the Compliance field must contain the standards with which the source is compliant.
  Refer to section *7.1.12.3* in [1] for details.
- Tcurre: the Touch Current field reports whether the source meets certain leakage current levels and if it has a ground pin
- Peak1: the Peak Current field must contain the combinations of Peak Current that the source supports.
  Refer to section *7.1.12.4* in [1] for details.
- Peak2: Peak current 2
- Peak3: Peak current 3
- Ttemp: the Touch Temp field must report the IEC standard used to determine the surface temperature of the source's enclosure.
- SRCin: the source input field must identify the possible inputs that provide power to the source. Some sources are only powered by a battery (such as in an automobile) rather than by the more common mains.
- Nbbatt: the Batteries field must report the number of batteries that the source supports.
- PDP: The source PDP field must report the source's rated PDP as defined in the PD3 specification.
  Refer to table *10-2* in [1].

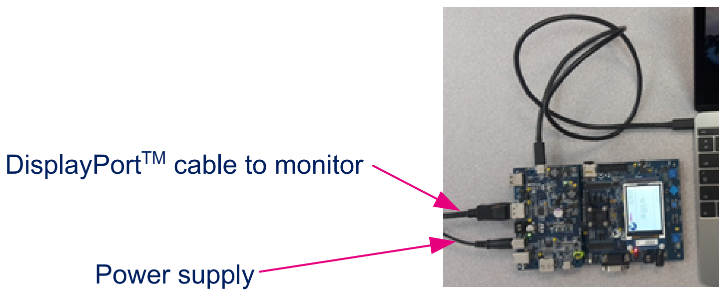
4.3.7 DisplayPort™ demonstration

4.3.7.1 USB Type-C® to DisplayPort™

This demonstration requires a device capable of outputting video from a USB Type-C® port. For example, this is performed with a MacBook© or a Samsung Galaxy S8©.

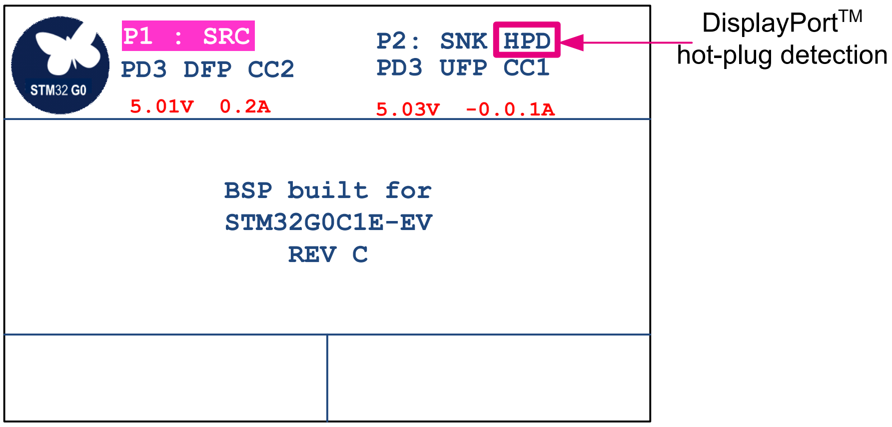
1. First, connect the DisplayPort™ monitor cable on 'Display Port Out', then connect the USB Type-C® device to the port as shown in Figure 59.

Figure 55. USB Type-C® to DisplayPort™



2. When the DisplayPort™ cable is connected, the `HPD` (hot-plug detection) mention is displayed in the top panel on the LCD as shown in Figure 60.

Figure 56. UCPD - Hot-plug detection



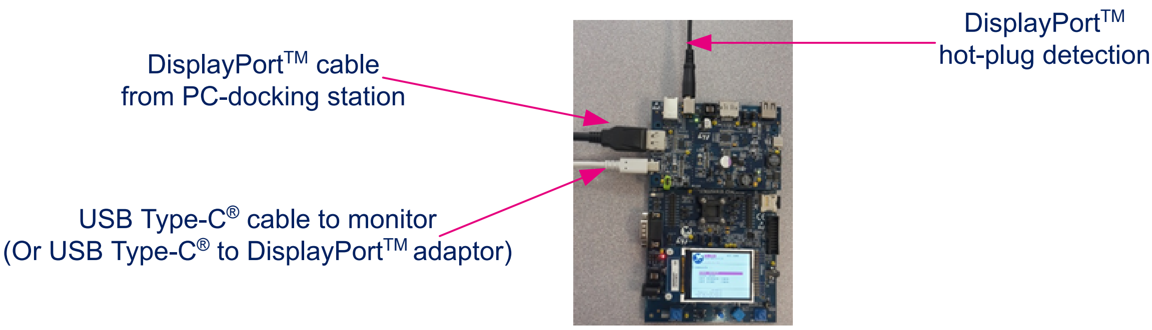
3. The debug window shows that a contract is obtained (refer to Figure 60), and the device screen is displayed on the DisplayPort™ monitor.

4.3.7.2 *DisplayPort™ to USB Type-C®*

For this demonstration, use a USB Type-C® monitor or a DisplayPort™ monitor through a USB Type-C® to DisplayPort™ adapter.

1. Connect the DisplayPort™ cable from the PC docking station to the ‘Display Port In’ connector of the STM32G0C1E-EV Evaluation board.
2. Connect the USB Type-C® monitor cable to Port 1 as shown in Figure 61.

Figure 57. DisplayPort™ to USB Type-C®



3. The following debug information is displayed:
   – Explicit contract
   – Send vdm enter mode
   – vdm enter mode ACK

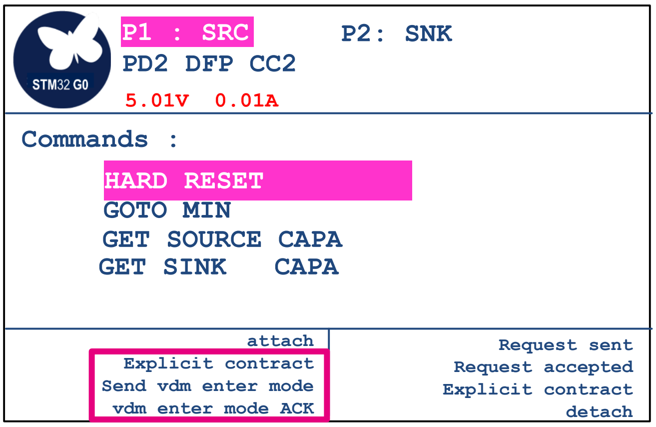


Figure 58. **UCPD - Vendor-defined message**

This information means that the VDM (vendor-defined message) capabilities are already read by the STM32G0C1VET6 MCU, and that the MUX path is correctly set on the STM32G0C1E-EV Evaluation board.

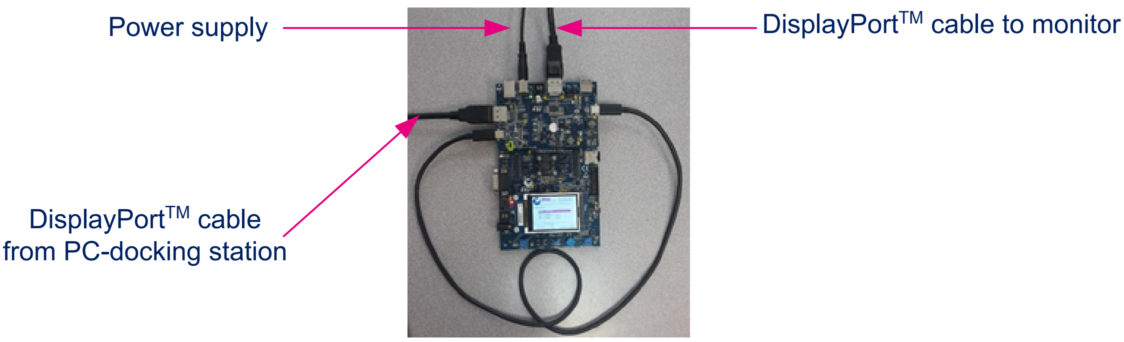
4.3.7.3 *DisplayPort™ to DisplayPort™ through USB Type-C®*

This demonstration requires neither a USB Type-C® monitor nor a device capable of driving video through USB Type-C®.

Plug as shown in Figure 63:

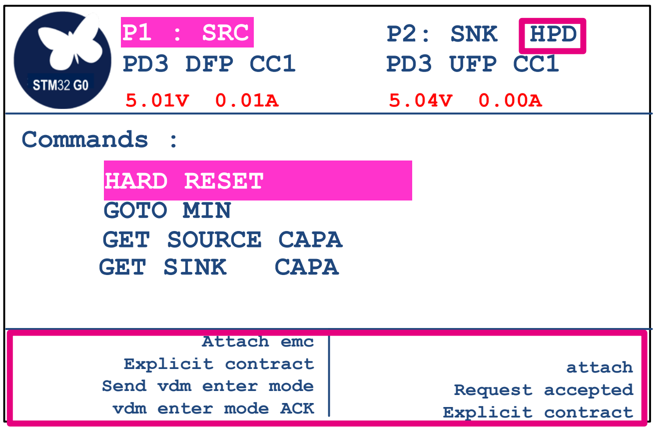
1. a DisplayPort™ cable from a PC or docking station to ‘Display Port In’
2. a DisplayPort™ cable to a monitor from ‘Display Port Out’
3. a USB Type-C® cable between USB Type-C® ports 1 and 2

Figure 59. **DisplayPort™ to DisplayPort™ through USB Type-C®**



A contract is established between the two USB Type-C® ports as shown in Figure 64.

Figure 60. UCPD - DP contract established

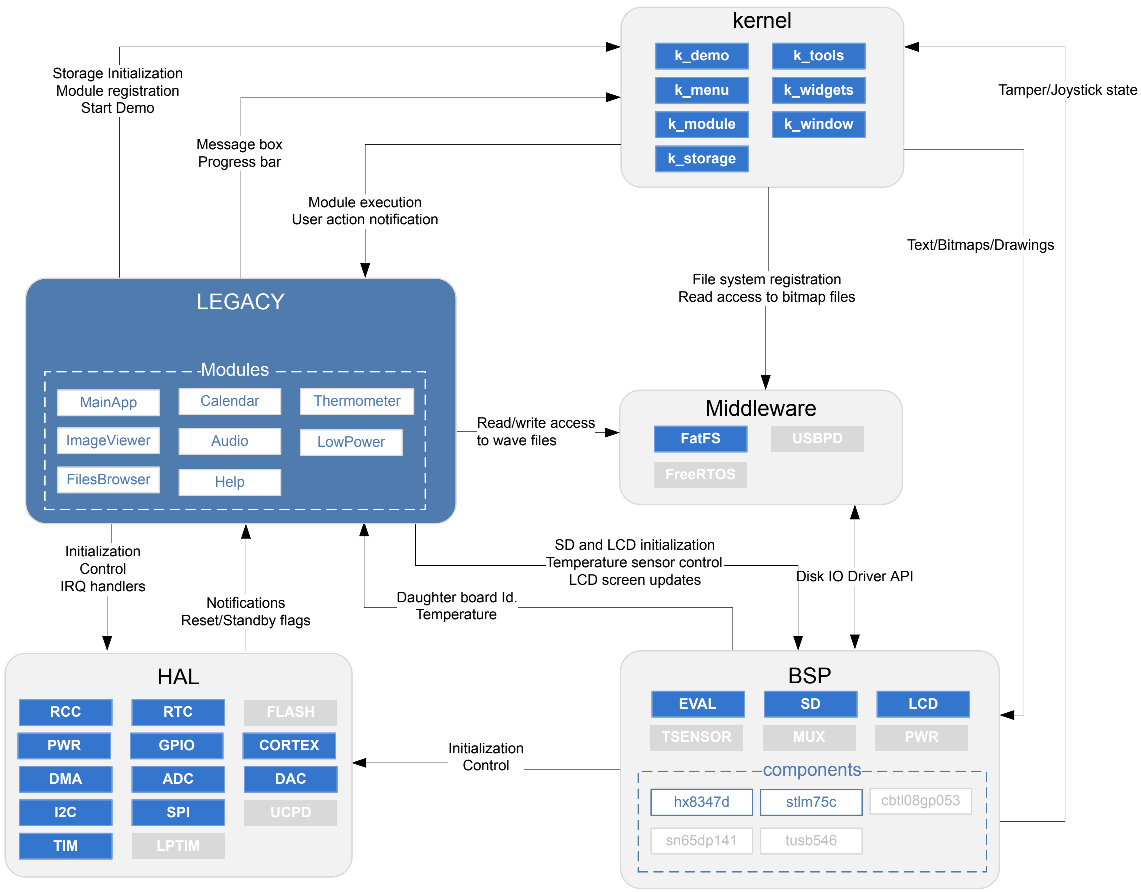


5 Software architecture

5.1 Legacy application

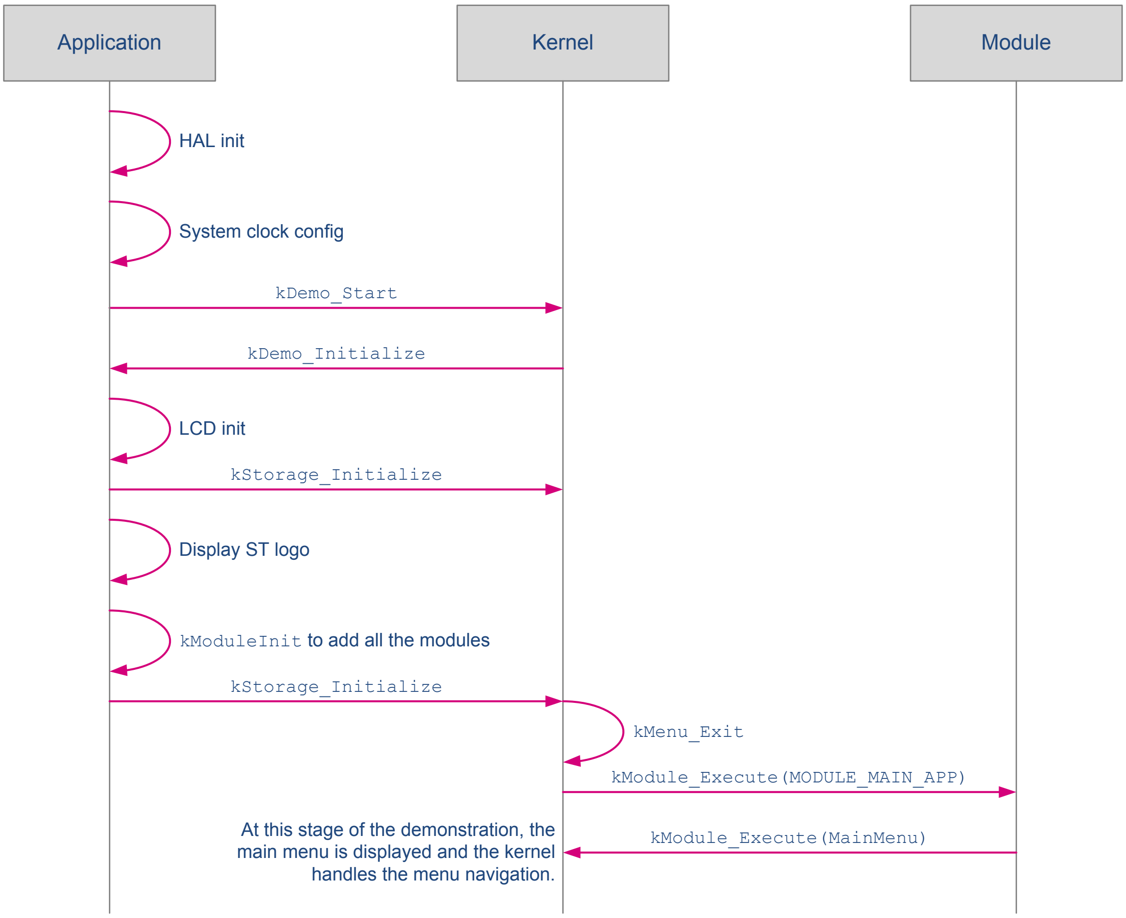
Figure 65 represents the main interactions between the legacy application and the surrounding software modules. The legacy application is a set of modules accessible through the main menu displayed on the LCD screen. At the application startup, the system clock is configured at 56 MHz and the file system, the BSD, and the LCD BSPs are initialized. If all the graphic resources (bitmap files) needed by the modules are present on the microSD™ card, each module is declared to the kernel (registration step). From this point onward, the kernel handles the menu navigation whatever the menu level is (root menu or module sub-menu) by managing an event loop. The user uses the joystick keys to move from one menu item to the other. When he presses the select joystick key from the main menu, the kernel launches the execution of the module. During module execution, the kernel forwards all joystick-key related actions to the running module. It is then up to the module to decide what to do. Enabled STM32G0C1VET6 interrupts (TAMPER, EXTI, DMA) are handled by the legacy application (refer to Figure 65) and are used, as usual, to map the interrupt vector on the HAL driver, depending on the module requirement.

Figure 61. Legacy application software architecture block diagram



The sequence diagram in Figure 66 summarizes the application startup until the kernel takes the lead.

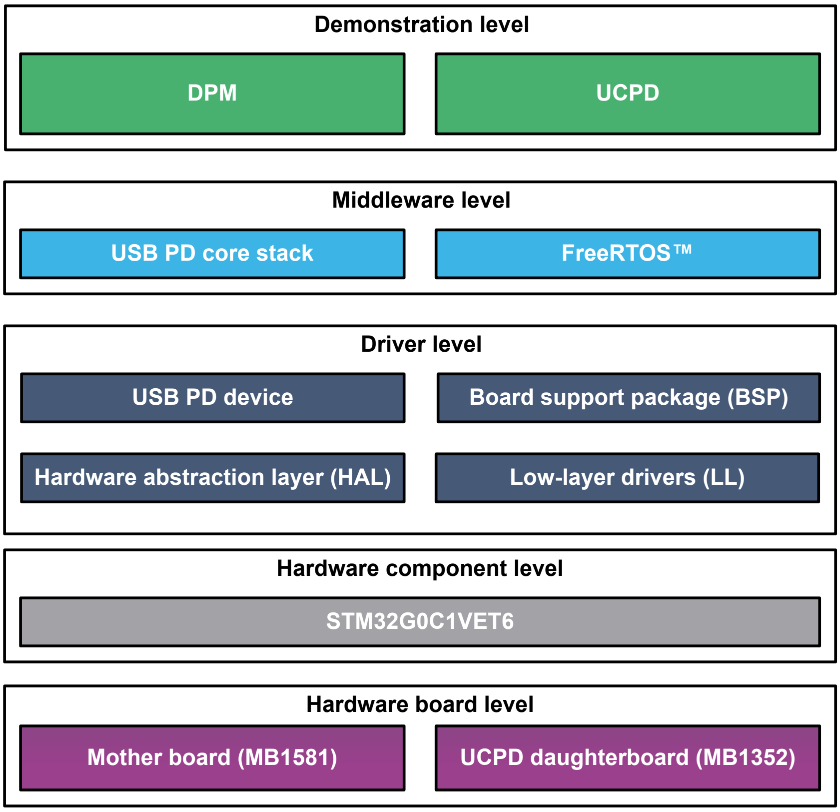
Figure 62. Simplified application startup sequence diagram



5.2 UCPD application

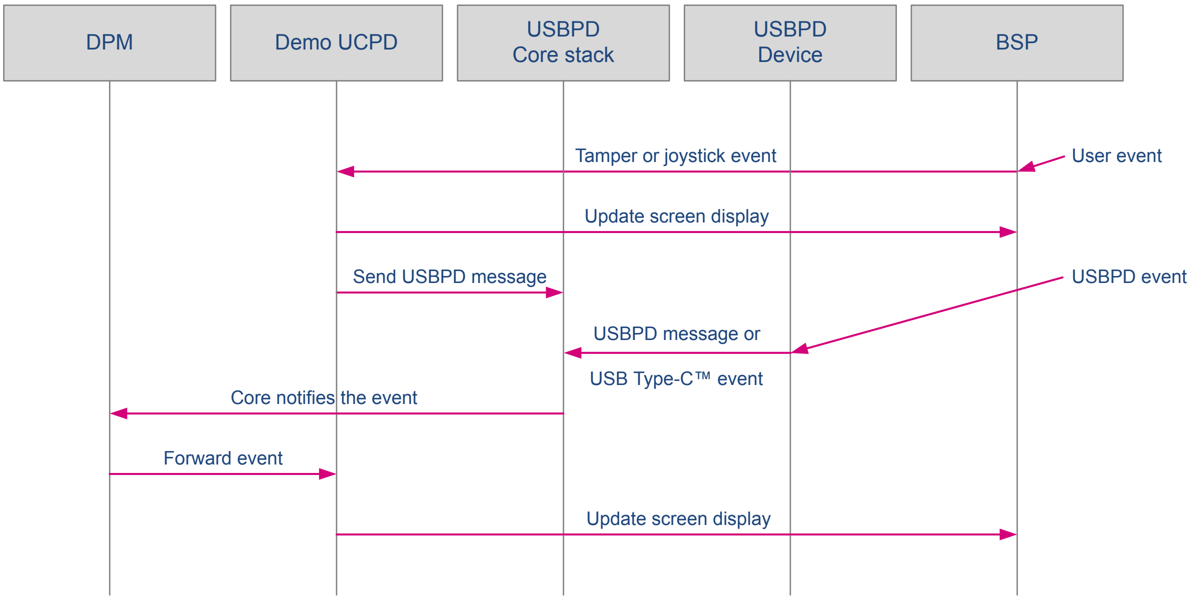
Figure 67 represents the main functional blocks involved in the UCPD application. The UCPD application is an RTOS task which receives events forwarded by DPM (Device Policy Manager). The events are mainly coming from the USBPD stack (such as attachment, detachment, and others). The application manages the display of the related information on the screen by using the BSP interface. At application startup, the system clock is configured at 56 MHz. The USBPD application initializes PORT1 as DRP, PORT2 as SYNC, and the demo is ready to catch DPM events. The user uses the TAMPER button and joystick to navigate inside the menu and execute actions.

Figure 63. UCPD application software architecture block diagram



The sequence diagram in Figure 68 summarizes the application processing of the UCPD application.

Figure 64. Application processing sequence diagram



5.3 Kernel API overview

5.3.1 k_demo

The function Kdemo_Start is executed by the application to launch modules scheduling and the kernel starts by running the module with ID=MODULE_MAIN_APP. To keep the kernel independent with the HW/SW, functions kDemo_Initialization and kDemo_UnInitialization are defined on the application side and called by kDemo_Start (see the application chapter for details about these functions).

Table 8. k_demo API

Function	Description
kDemo_Initialization	Initializes the demo (the function is defined on the application side).
kDemo_UnInitialization	Initializes the demo (the function is defined on the application side).
kDemo_Start	Starts the demo (Initialize, Run, UnInitialize, Exit).
kDemo_Execute	Initializes kMenu and executes the MODULE_MAIN_APP module.

5.3.2 k_menu

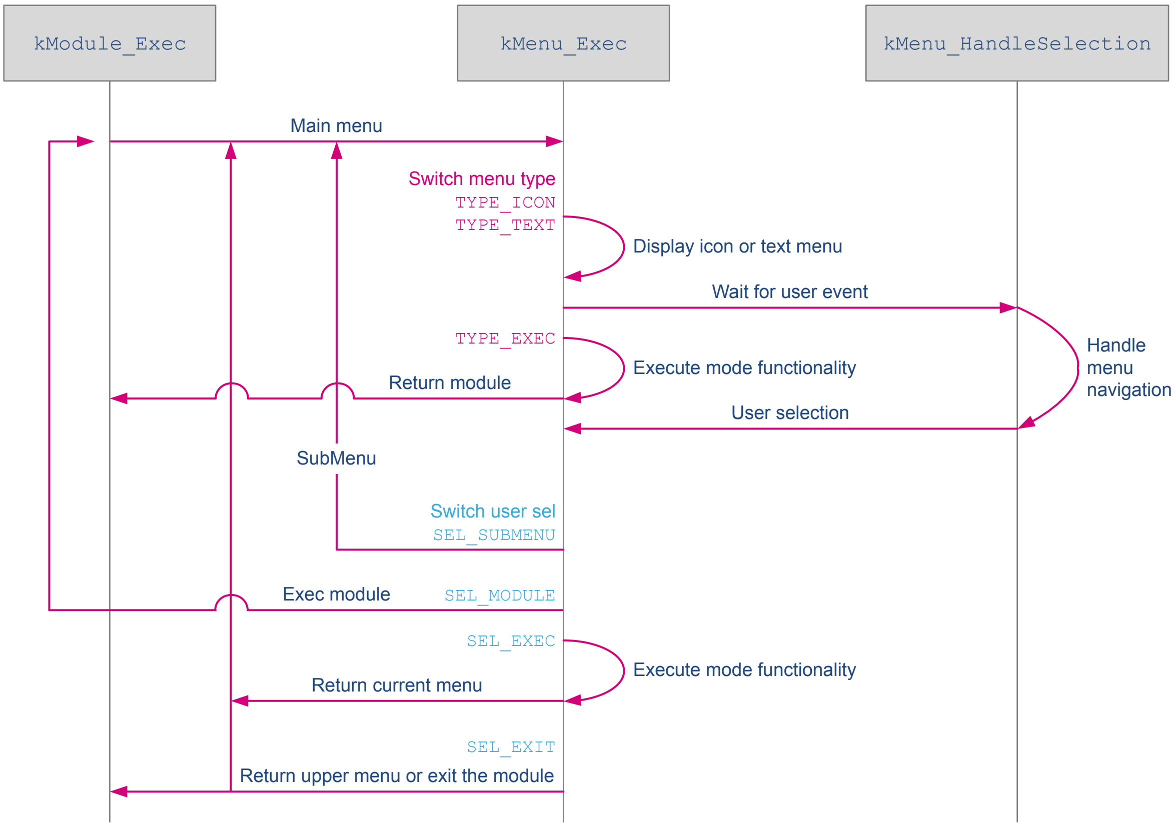
The module execution is started by a call to function kMenu_Execute. This kernel function handles navigation in the menu and the execution functionalities using the structure t_menu defined inside the module.

Table 9. k_menu API

Function	Description
kMenu_Init	Initializes the joystick.
kMenu_EventHandler	GPIO event handler
kMenu_Execute	Function to execute a menu
kMenu_Header	Function to display the header information
kMenu_HandleSelectionEx	The function handles the navigation in the menu.

Figure 69 shows the execution flow for a menu.

Figure 65. kMenu sequence diagram



5.3.3 k_module

This kernel part centralizes information about all modules available inside demonstration firmware. The `kModule_Init` function defined on the application side, registers all the modules present with the help of function `kModule_Init` using the structure `K_ModuleItem_Typedef` defined inside the module.

Table 10. `k_module` API

Function	Description
`kModule_Init`	Defined on the application side
`kModule_Add`	Adds module inside the module list.
`kModule_CheckRessource`	Checks the module resource.
`kModule_Execute`	Executes the module.

5.3.4 k_storage

The `k_storage` API handles only microSD™ card storage and provides some services to simplify module development.

Table 11. `k_storage` API

Function	Description
`kStorage_Init`	Mounts the SD card file system.

Function	Description
kStorage_DeInit	Unmounts the file system.
kStorage_GetStatus	Returns SD card presence.
kStorage_GetDirectoryFiles	Returns the name of the file present inside a directory.
kStorage_FileExist	Checks if a file exists.
kStorage_GetFileInfo	Returns file information.
kStorage_OpenFileDrawBMP	Opens a .bmp file and displays it (only files of 8 Kbytes max).
kStorage_OpenFileDrawPixel	Opens a .bmp file and displays it line by line.
kStorage_GetExt	Returns the file extension.

5.3.5 k_window

The k_window API provides services to display a popup window without user event management (must be handled outside, for example inside the module).

Table 12. k_window API

Function	Description
kWindow_Popup	Displays a popup.
kWindow_Error	Displays a red popup with an error message.

5.3.6 k_widgets

k_widgets provides a set of services allowing the creation of graphic objects to be displayed on the LCD screen.

Table 13. k_widgets API

Function	Description
kWidgets_ProgressBarCreate	Creates and displays an empty progress bar object.
kWidgets_ProgressBarDestroy	Destroys the progress bar object.
kWidgets_ProgressBarUpdate	Updates the progress bar content.
kWidgets_ProgressBarReset	Resets the progress bar content.

5.3.7 k_tools

The k_tools API provides tools for module development.

Table 14. k_demo API

Function	Description
kTools_Buffercmp	Returns 0 if both buffers are identical.

6 Advanced module description

6.1 Module detailed description

A module is an autonomous application that runs directly from the launcher or another module. The module contains two main parts:

- Module control: the kernel uses this part to handle input (user button) and output (display) to interact with the end-user.
- Functional behavior: execution functions

6.1.1 Module control

A module is described by a simple structure named `K_ModuleItem_Typedef`. This structure provides a unique ID, initialization and de-initialization functions, a function for checking resource application, and the main function as listed in Table 15.

Table 15. K_ModuleItem_Typedef structure description

Field	Description
`kModuleId`	Unique ID module. This has to be defined inside the `MODULES_INFO` enumeration in the `k_config.h` include file.
`kModulePreExec`	This function pointer is used to initialize the low-layer driver, allocate memory, or execute any specific action before module execution. This function is optional.
`kModuleExec`	This function pointer is used to start the main application module. In the current kernel architecture, this function is used to call the `kModuleExecute` function that executes the module application based on the menu structure (refer to Section 6.1.2.1). This function is mandatory.
`kModulePostExec`	This function pointer is used to de-initialize the low-layer driver, and release the resource allocation done inside `kModulePreExec`. This function is optional.
`kModuleRessouceCheck`	This function pointer is used to control if all required resources of the module are available. It is used for instance to check if a specific resource file is present on the microSD™ card. This function is optional.

6.1.2 Module menu description and graphical interface

The module menu is used to describe the module architecture and display.

6.1.2.1 tMenu structure

This structure is the main entry point used by function `kModuleExecute` to display the module MMI and execute the functionalities.

Table 16. tMenu structure description

`pszTitle`	Character string that contains the menu title to display.
`psItems`	Pointer on the menu description `tMenuItem` item (refer to Section 6.1.2.2).
`nItems`	Number of entry inside the above menu item data pointer.
`nType`	Menu type. This parameter allows the kernel to distinguish what kind of menu is executed. Menu type is used to display icon menu (`TYPE_ICON`), basic text string (`TYPE_TEXT`), or only code execution (`TYPE_EXEC`).

`line`	In case of icon display selection, number of icon lines.
`column`	In case of icon display selection, number of icon columns.

6.1.2.2 *tMenuItem structure*

This structure is used by the menu structure to describe all different items or sub-menus of the application module.

Table 17. tMenuItem structure description

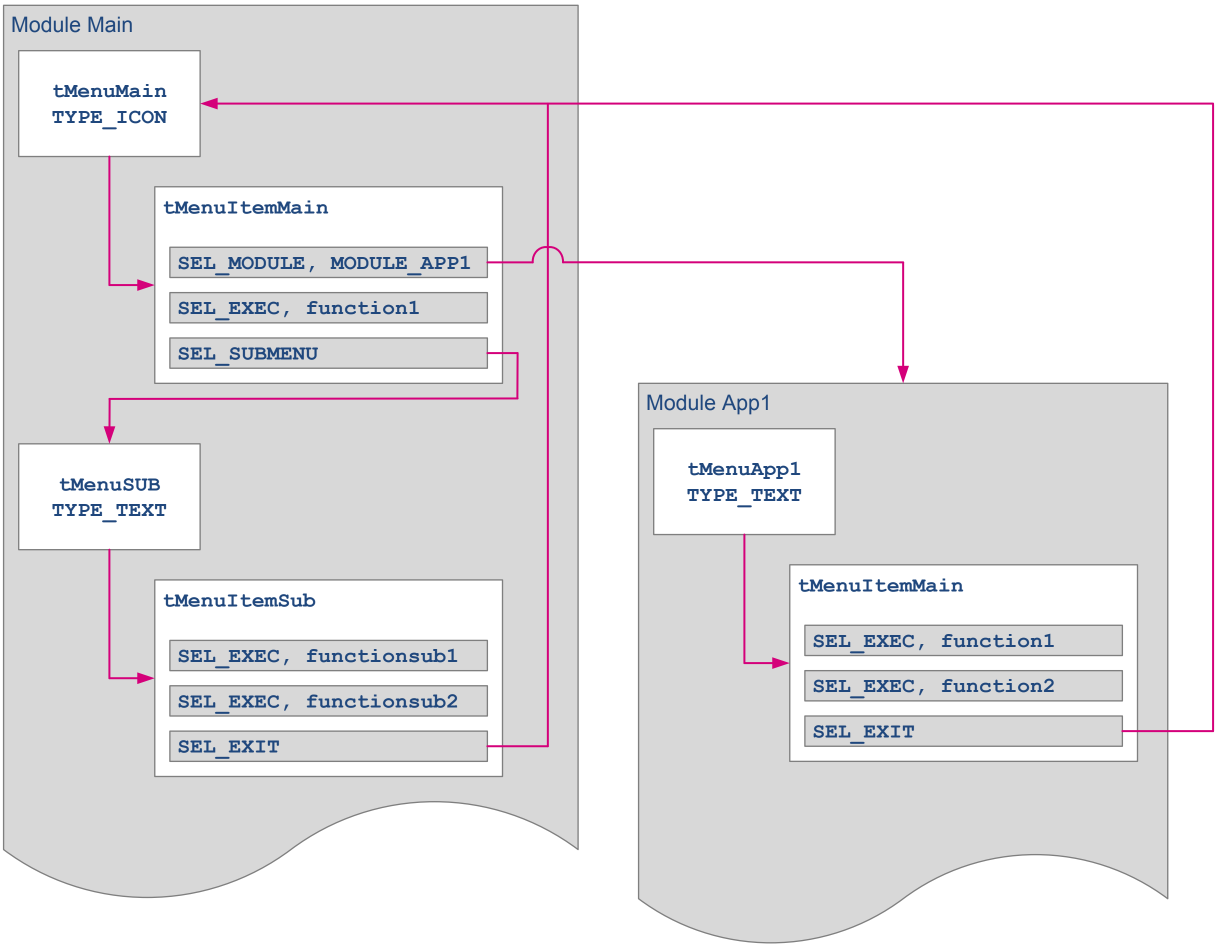
Field	Description
`pszTitle`	Character string that contains item title to display.
`x, y`	In case the menu is of the icon type, those numbers provide the icon position on the screen.
`SelType`	Item selection type. This field is one of the following defined values: • `SEL_EXEC`: directly executes a functionality • `SEL_SUBMENU`: selects a sub-menu • `SEL_EXIT`: exits the current menu and returns to the previous menu or module • `SEL_MODULE`: executes a module
`ModuleId`	Corresponding Module ID
`pfExecFunc`	This function pointer executes the selected item.
`pfActionFunc`	Function pointer on a dedicated callback that is used to capture the user interface selection through the joystick, button, or any other interface interrupt.
`psSubMenu`	Pointer on a sub-menu item of `tMenutype`
`pIconPath`	Character string, which contains the icon name and path to be displayed
`pIconSelectedPath`	Character string, which contains an alternative icon name and path to be displayed when the item is selected by the user.

6.1.2.3 *Menu example*

Figure 70 illustrates the basic example of a menu structure with the use of two modules:

- The main module is built with two levels of the menu and three functionalities. The level main is a `TYPE_ICON` menu, which may execute another *App1* module with the `SEL_MODULE` property, execute *function1*, or display a sub-menu with the `SEL_SUBMENU` property.
- The sub-menu may execute *functionsub1*, *functionsub2*, or go back to the main menu using `SEL_EXIT`.
- The module *App1* has a `TYPE_TEXT` context with 2 embedded functionalities: *function1* and *function2*.

Figure 66. **Module menu architecture example**



6.1.3 Functionality

Functionality is describing module behavior from the end-user point of view. Two cases are possible:

- Menu event that is executed when the module is selected. The called function must respect the following prototype: `void functionExec(void)`.

  This function is linked to the module context by function pointer `pfExecFunc` in the `tMenuItem` structure (see above description). On function exit, the kernel returns to the previous menu state.
- In many cases, functionality is stopped by a user event. As a consequence, the kernel offers the capability to return all possible events through a callback function with the following prototype: `void functionSel(uint8_t sel)`.

  This function is linked to the module context by function pointer `pfActionFunc` in the `tMenuItem` structure (see above description).

6.2 Adding a new module

Once the module appearance and functionality are defined and created, based on constraints described above, only the module is left to be added:

1. Define the new module unique ID in file `k_config.h`
2. The `kModule_Add()` function must be called in `KDemo_Initialization()`, with the module unique ID as parameter
3. Modify the main module item description table, adding a dedicated line matching the new module description (`MainMenuItems` in `main_app.c` in our demonstration firmware)
4. Add any possible resource file into the microSD™ card as explained above

7 Acronyms

Table 1 presents the definitions of the acronyms that are relevant for a better contextual understanding of this document.

Table 18. Acronyms

Acronym	Definition
BSP	Board support package
CC	Configuration channel
CORDIC	Coordinate rotation digital computer
DMA	Direct memory access
DPM	Device policy manager
DRD	Dual-role data
DRP	Dual-role power
FIR	Finite impulse response filter
FMAC	Filter math accelerator
HAL	Hardware abstraction layer
HRTIM	High-resolution timer
IIR	Infinite impulse response filter
LL	Low-layer drivers, low-layer API
PID	Product ID
RTOS	Real-time operating system
SAI	Synchronous audio interface
UCPD	Name of the demonstration dedicated to USB Power Delivery with USB Type-C®
USB-PD USB PD USBPD	USB Power Delivery
VID	Vendor ID

8 References

Table 19. References

ID	Description
[1]	USB-IF (2017) *Universal Serial Bus Power Delivery Specification rev 3.0*.
[2]	*Evaluation board with STM32G0C1VE MCU* user manual (UM2783).

Revision history

Table 20. **Document revision history**

Date	Version	Changes
10-Nov-2020	1	Initial release.

Contents

List of tables

List of figures