

STM32CubeG4 STM32G474E-EVAL 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 the conception to the realization, among which:
 - [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 IP configuration, code generation, code compilation, and debug features
 - STM32CubeProgrammer ([STM32CubeProg](#)), a programming tool available in graphical and command-line versions
 - STM32CubeMonitor-Power ([STM32CubeMonPwr](#)), a monitoring tool to measure and help in the optimization of the power consumption of the MCU
 - STM32CubeMonitor-UCPD ([STM32CubeMonUCPD](#)), a monitoring tool to measure and help in the configuration of USB Type-C™ and Power Delivery applications
- [STM32Cube MCU & MPU Packages](#), comprehensive embedded-software platforms specific to each microcontroller and microprocessor series (such as STM32CubeG4 for the STM32G4 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 the HW
 - A consistent set of middleware components such as FAT file system, RTOS, 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 & MPU Packages with:
 - Middleware extensions and applicative layers
 - Examples running on some specific STMicroelectronics development boards

The [STM32CubeG4](#) demonstration firmware running on the [STM32G474E-EVAL](#) 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 STM32G474QET6 device and exercise some peripherals of the STM32G474E-EVAL Evaluation board. These applications are:

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



1 STM32CubeG4 main features

STM32CubeG4 gathers, in a single package, all the generic embedded software components, required to develop an application on STM32G4 microcontrollers. In line with the STM32Cube initiative, this set of components is highly portable, not only to the STM32G4 Series but also to other STM32 series.

STM32CubeG4 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 open-source BSD license for user convenience.

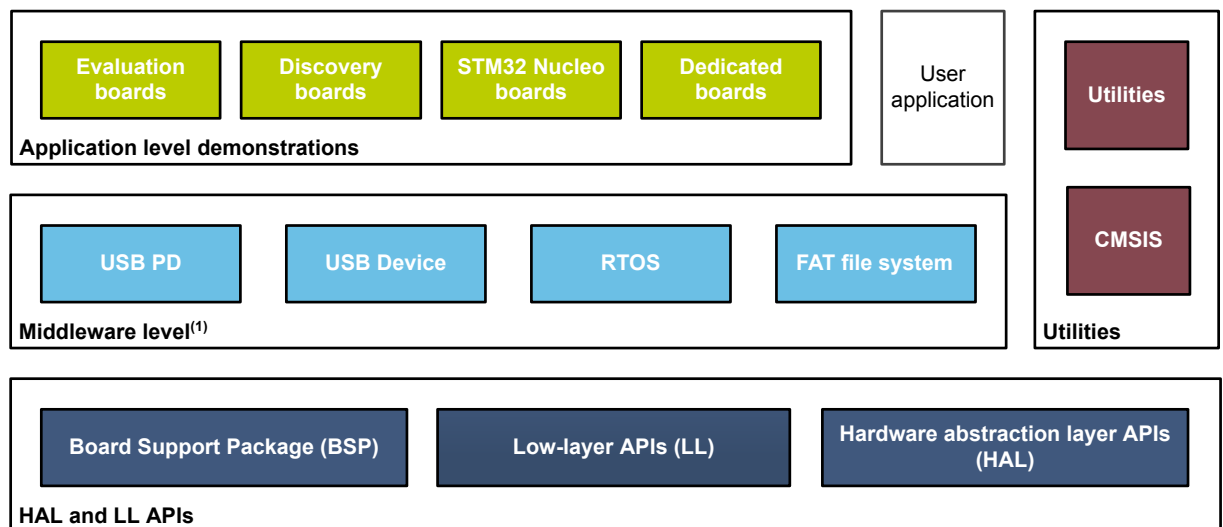
The STM32CubeG4 MCU Package also contains a set of middleware components with the corresponding examples. They come in free user-friendly license terms:

- FAT file system based on open source FatFS solution
- CMSIS-RTOS implementation with FreeRTOS™ open source solution
- USB PD Devices and Core libraries
- USB Device library

Several applications and demonstrations implementing all these middleware components are also provided in the STM32CubeG4 MCU Package.

The block diagram of STM32Cube is shown in Figure 1.

Figure 1. STM32CubeG4 firmware components



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

The demonstration firmware for the STM32G474E-EVAL Evaluation board comes on top of the STM32CubeG4 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 STM32G4 device capability to offer a wide scope of usages.

The STM32G4 microcontrollers are based on the Arm® 32-bit Cortex®-M4 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 [STM32G474E-EVAL](#) Evaluation board (MB1397)
- One microSD™ card
- One USB cable to power the STM32G474E-EVAL board from its ST-LINK USB

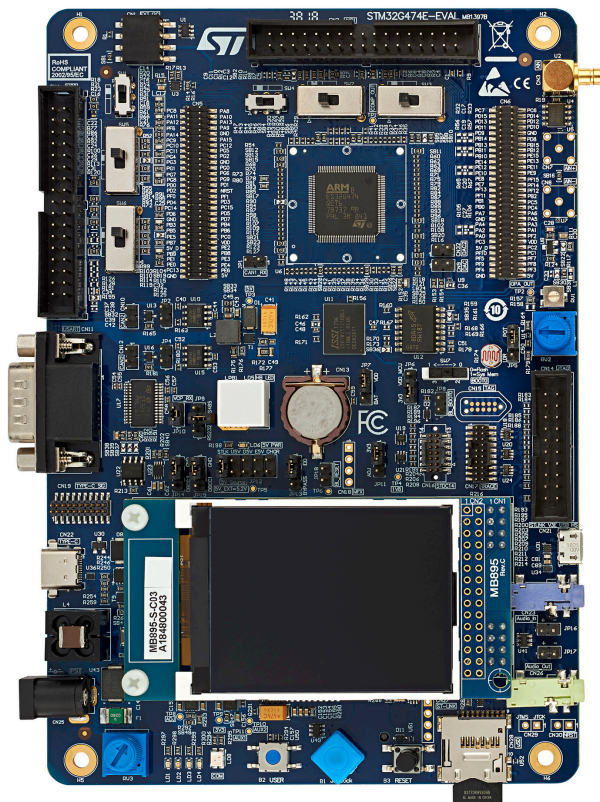
The [STM32G474E-EVAL](#) (MB1397) Evaluation board is connected to the host PC by means of the USB cable. It does not require any external power supply.

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/STM32G474-EVAL/Demonstrations/Binary/SD_card` firmware package directory.

2.2 Hardware settings

The [STM32CubeG4](#)-based demonstration supports the STM32G474QET6 device and runs on the [STM32G474E-EVAL](#) Evaluation board from STMicroelectronics presented in [Figure 2](#).

Figure 2. STM32G474E-EVAL front view (MB1397)

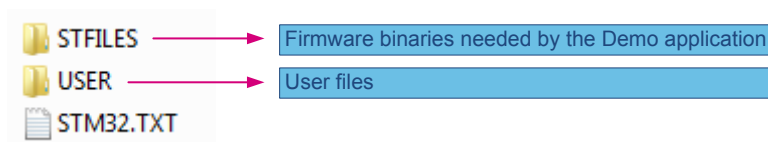


The default jumper settings must be used to run the demonstration. Refer to [\[2\]](#) for details.

2.3 microSD™ status

The [STM32G474E-EVAL](#) board comes with a microSD™ memory card pre-programmed with the image resources, text files and directory trees used by the demonstration FW. However, the user may load his own image files in the USER directory, assuming that file formats are supported by the demonstration (* .bmp).

Figure 3. 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 (MDK-ARM, EWARM or [SW4STM32](#)):

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

3 Demonstration firmware package

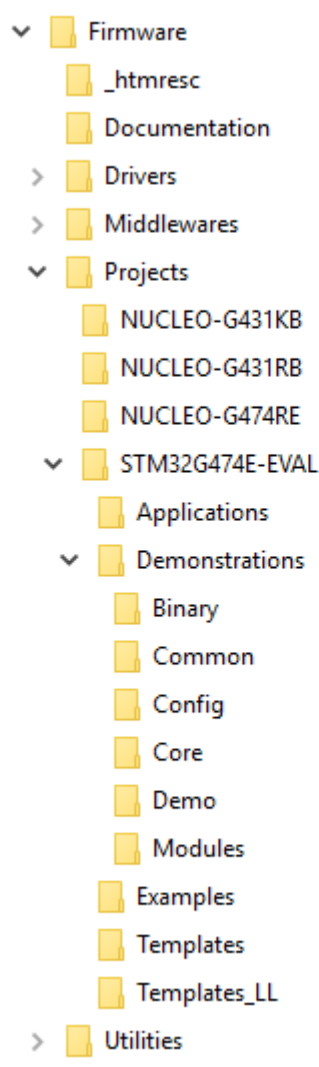
The demonstration targets the following objectives:

- Toolkit with low memory consumption
- Independent modular applications with a high level of reuse
- Basic menu navigation with the joystick
- Comprehensive STM32G4 functional coverage

3.1 Demonstration repository

Figure 4 shows the demonstration folder organization.

Figure 4. Demonstration folder organization



The demonstration sources are located in the `Projects` folder of the [STM32CubeG4](#) MCU Package for each supported board. The folder selected here is `STM32G474E-EVAL` for the Evaluation board.

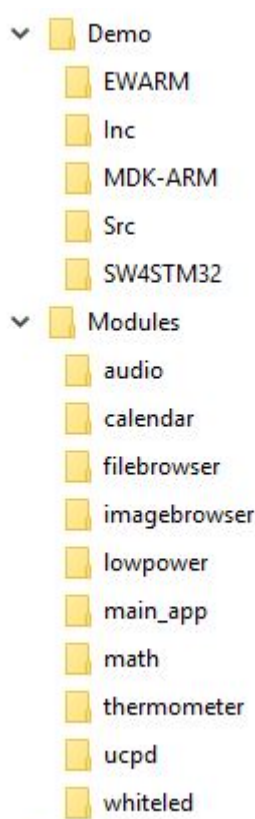
The files making up the demonstration firmware are spread over the following sub-directories:

- `Common`: C source files implementing common services used by the demonstration firmware

- **Config:** FatFS, HAL, BSP and demonstration kernel configuration files
- **Core:** demonstration kernel implementation files
- **Demo:** demonstration firmware
- **Modules:**
- **SD_card:** resource files (bitmaps) required by the demonstration firmware to be copied into the microSD™ card

Figure 5 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 5. Demonstration module folder organization



The **Modules** folder contains the following sub-folders:

- **main_app:** main menu management
- **audio:** audio record and playback
- **calendar:** date, time and alarm setting
- **filebrowser:** navigation through a folder tree
- **imagebrowser:** bitmap images slide show
- **lowpower:** low-power modes
- **math:** signal filtering
- **thermometer:** temperature measurement
- **ucpd:** USB Power Delivery
- **whiteled:** LED control

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

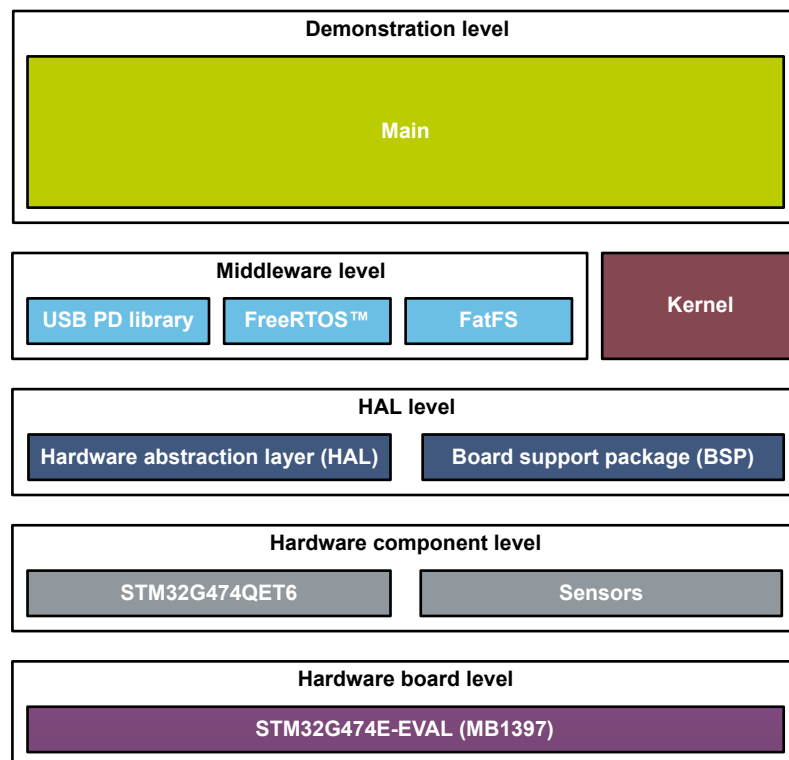
- **EWARM:** IAR Embedded Workbench™

- MDK-ARM: Keil® Microcontroller Development Kit
- SW4STM32: [SW4STM32](#), system workbench for STM32

3.2 Demonstration architecture overview

The top level software architecture of the [STM32G474E-EVAL](#) demonstration firmware is represented in [Figure 6](#). The software elements mentioned in this diagram are briefly depicted in dedicated sub-sections.

Figure 6. STM32G474E-EVAL demonstration firmware architecture



3.2.1 Demonstrations

The application contains many demonstrations that are easily reusable, such as RTC calendar, FAT file system implementation on microSD™ card, wave player and voice recorder using SAI and DMA peripherals, low-power modes, temperature sensor interfacing, and TFT LCD.

3.2.2 UCPD

The UCPD application consists in USB Type-C™ connection/disconnection detection and USB Type-C™ power contract negotiation. The UCPD demonstration is also capable of changing the power role. Protocol message exchanges can be displayed by means of the STM32CubeMonitor-UCPD tool ([STM32CubeMonUCPD](#)).

3.2.3 HAL level

The HAL level layer consists of the `stm32g4xx_hal_ppp.c` HAL drivers together with the [STM32G474E-EVAL](#) board support package (BSP).

3.2.4 Kernel

The kernel is a suite of components providing high-level services to the applications in order to facilitate 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 demonstration applications to get access to the files stored in the microSD™ card.
- USBPD: USB-PD software stack.

3.3 Microcontroller resources

3.3.1 Peripherals

The following sections detail which peripherals of the STM32G474QET6 microcontroller are used by the demonstrations.

3.3.1.1 Peripherals used by the main demonstration

Figure 7. STM32G474QET6 peripherals used by the main demonstration

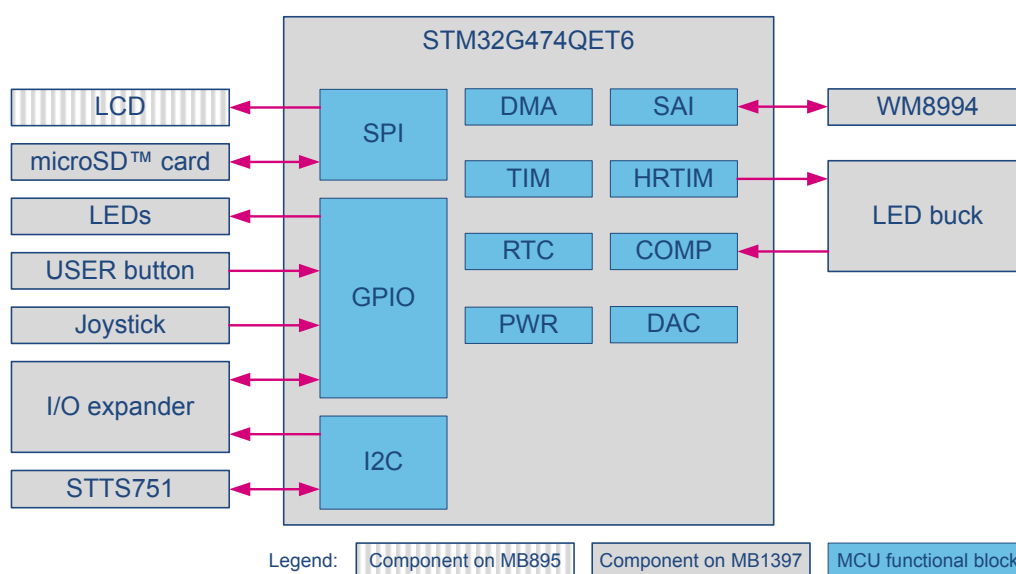


Table 1. STM32G474QET6 peripherals used by the main demonstration

Peripheral	Usage description
SPI	Both LCD and microSD™ card 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 demonstration execution.
GPIO	GPIOs are used to toggle the LEDs when an error is detected during the demonstration. Also the GPIO pins connected to the joystick and USER button are used to interact with the demonstration (menu navigation).
SAI	SAI is used for the audio data link with the WM8994 codec.
COMP	COMP is used to sense and limit the current of LEDbuck.
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 transfer are used to transfer audio samples from audio playback buffer to the DAC data register or to transfer audio sample from the ADC data register to the audio record buffer.
HRTIM	HRTIM is used to drive the LED buck.

Peripheral	Usage description
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 application to control the STTS751 component (digital temperature sensor).

3.3.1.2 Peripherals used by the UCPD demonstration

Figure 8. STM32G474QET6 peripherals used by the UCPD demonstration

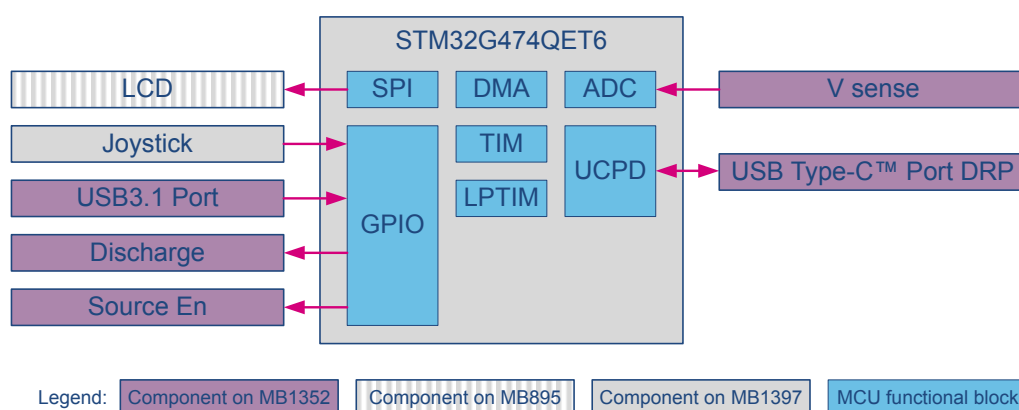


Table 2. STM32G474QET6 peripherals used by the UCPD demonstration

Peripheral	Usage description
SPI	LCD is controlled through SPI1. Write accesses to the LCD are performed to display strings and bitmaps during the UCPD demonstration execution.
GPIO	The GPIO pins connected to the joystick are used to interact with the UCPD demonstration, such as menu navigation: <ul style="list-style-type: none"> One GPIO pin is used to detect USB3.1 cable connect/disconnect One GPIO pin is used to enable the VBUS on the USB Type-C™ port (DRP) One GPIO pin is used to control the USB VBUS discharge mechanism on the USB Type-C™ port (DRP)
ADC	<ul style="list-style-type: none"> ADC channel 9 is used to measure the voltage level on the VBUS line of the USB Type-C™ port (DRP)
UCPD	UCPD is used to manage the USB Type-C™ communication over the USB Type-C™ ports.
DMA	DMA is used for ADC conversions.
LPTIM	LPTIM1 is used to generate the PWM signal controlling the voltage level on the VBUS line of USB Type-C™ port 1 (when the duty cycle of the PWM signal is 0%, the VBUS voltage level is 15 V; when the duty cycle is 10%, the VBUS voltage level is 5 V).

3.3.2 Interrupts

Table 3 shows all the external interrupts used by the demonstration.

Table 3. STM32G474QET6 demonstration interrupt usage

Interrupt	Usage description	Main	UCPD
Systick	Delay management.	YES	YES
EXTI line 0	IO Expander (Joystick, microSD™ card).	YES	YES
EXTI line 13	User button.	YES	YES
DMA1_Channel3	Math CORDIC write.	YES	NO
DMA1_Channel4	Math CORDIC read.	YES	NO
DMA1_Channel5	Math FMAC with DMA In.	YES	NO
DMA1_Channel6	Math FMAC with DMA Out.	YES	YES
DMA1_Channel7	Math FMAC with DMA memory to memory.	YES	NO
DMA2_Channel1	SAI Audio output.	YES	NO
DMA2_Channel2	SAI Audio Input.	YES	NO
RTC_TAMP_LSECSS	RTC tamper.	YES	NO
RTC_Alarm	RTC alarm.	YES	NO
HRTIM1_Master	White LED burst period.	YES	NO
UCPD1	UCPD IT.	NO	YES
USART1	USB-PD trace Tx.	NO	YES

3.3.3 Internal memory size

The system memory areas are used in the demonstration firmware as described in [Table 4](#).

Table 4. Internal memory configuration

Memory	Start address	Application
Internal Flash	0x0800 0000	Demonstration firmware run code and constants.
Internal SRAM1	0x2000 0000	Demonstration firmware data.

3.3.4 External memory organization

The [STM32G474E-EVAL](#) demonstration is based on the FatFS embedded free FAT file system. The file system is needed by the demonstration loader to retrieve the binary file to load in the Flash memory and read all media information from the on-board microSD™ card memory. The microSD™ memory card is organized in three subdirectories:

- **BIN:** this directory contains the binary images of the main and UCPD demonstrations (`Legacy.bin` and `Ucpd.bin`), built in *relocate* mode.
- **STFILES:** this directory contains all the bitmaps required by the demonstration firmware.
- **USER:** this is a user folder. The user may add his own files here to be played inside the demo menus (pictures and `.wav` files). This folder is used only by the file browser (refer to [Section 4.11 File browser application](#)), image viewer (refer to [Section 4.7 Image viewer application](#)), and wave player applications (refer to [Section 4.8.2 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 [STM32G474E-EVAL](#) is already programmed with the media files to run the demonstration. These files are also available within the demonstration firmware package in the `Project s\STM32G474E-EVAL\Demonstrations\Binary\SD_card\` folder.

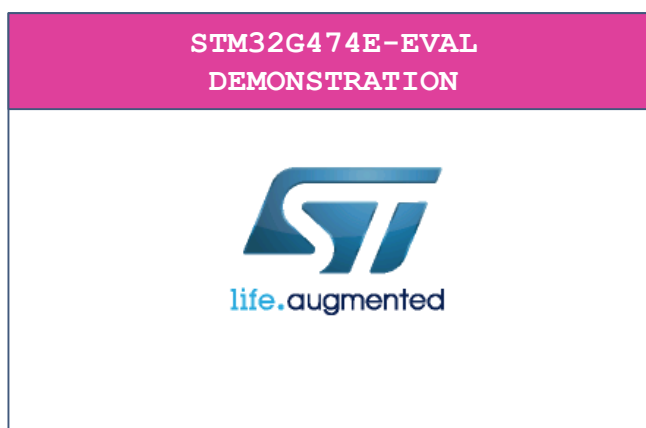
4 Running the demonstration

4.1 Demonstration startup

4.1.1 Normal processing

After a board reset, the welcome screen is displayed and the STMicroelectronics logo appears on the LCD as illustrated in [Figure 9](#).

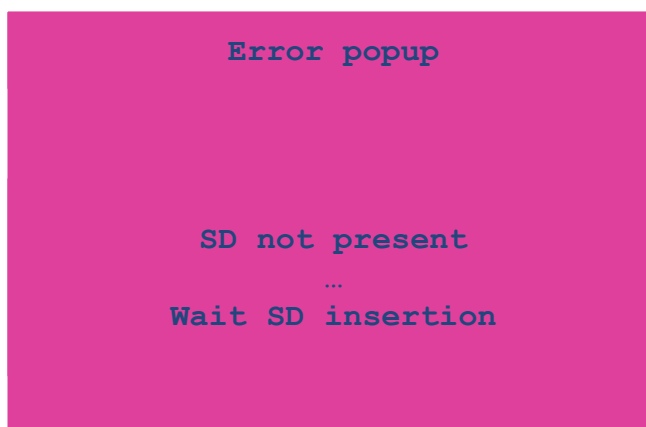
Figure 9. Welcome screen



4.1.2 Error cases

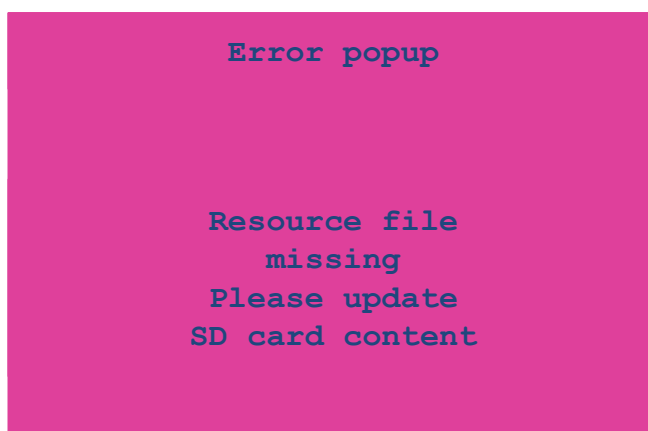
If no card is detected, the demonstration does not start and the message shown in [Figure 10](#) is displayed on the LCD screen. The demonstration waits for the microSD™ card insertion to proceed.

Figure 10. microSD™ card detection error



If the microSD™ card does not contain appropriate icons and pictures for the demonstration, the message shown in [Figure 11](#) is displayed on the LDC screen.

Figure 11. microSD™ card content error



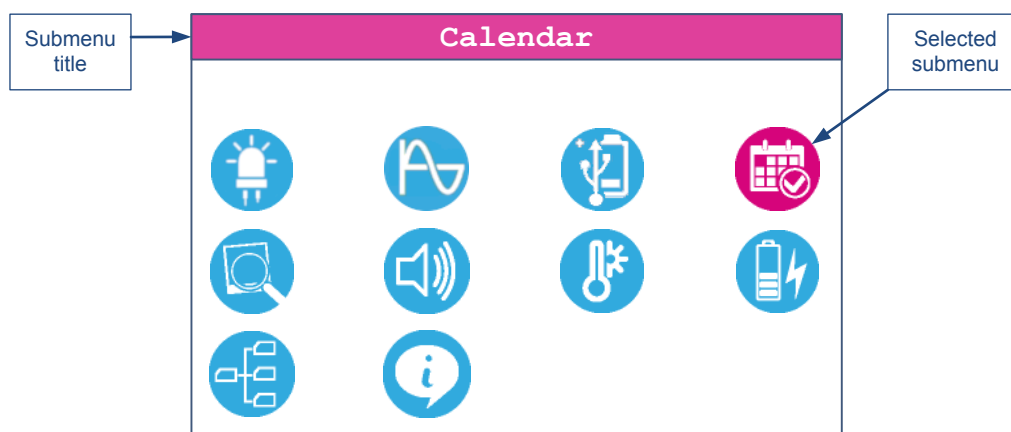
4.2 Main menu

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
- SEL 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 display (refer to Figure 12).

Figure 12. Main menu



4.3 White LED

The white LED application demonstrates the possibility to drive LED brightness using various on-chip peripherals:

- The high-resolution timer (HRTIM) to drive LED with PWM
- The fast digital-to-analog converter (DAC) coupled with HRTIM to generate sawtooth
- The comparator (COMP)

For technical details, refer to [4], which has been written for the 32F3348DISCOVERY Discovery kit. The same techniques are in the white LED application used to avoid flickering:

- Slope compensation
- Dithering
- High-brightness LED current regulation soft start

The main differences are:

- Instead of DMA, the white LED application takes advantage of STM32G474QET6 HRTIM capability to control internal DAC sawtooth signal generation.
60 steps are used to generated the sawtooth signal.
- Signals on [STM32G474E-EVAL](#) are mapped as follows:
 - HRTIM E channel 1 output on PC8 is used to drive the MOSFET (white LED)
 - Fast DAC4 channel 2 is used to generate the sawtooth signal
 - Sense-resistor voltage level is fed back to internal comparator COMP6 on PB11

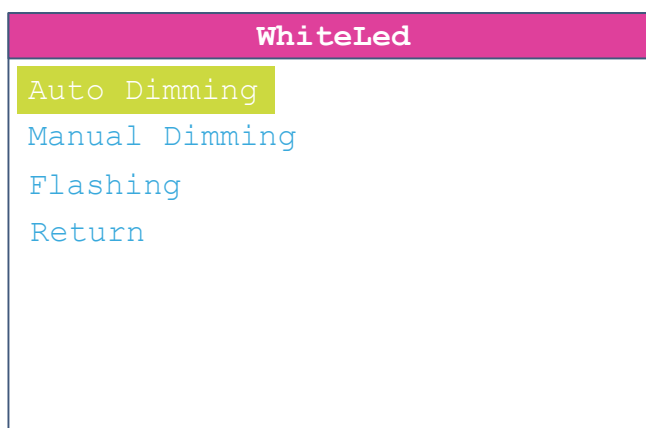
Caution: For safety reasons, the maximum high-brightness LED forward current is limited by software to 250 mA, and an optical cube-shaped protection is placed over the LED. Do not override this current limitation and do not remove the optical protection while the LED is operating. These precautions must be taken due to the high luminous flux emitted by the high-brightness LED. Users must not expose their eyes to the LED direct light.

4.3.1 White LED submenu

The white LED submenu allows the selection among several demonstrations:

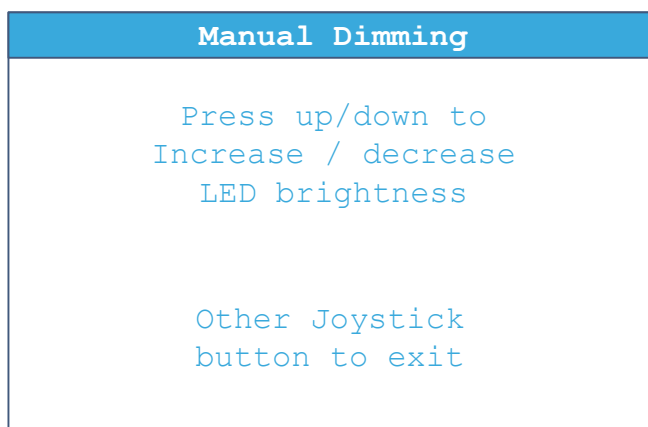
- Automatic dimming
- Manual dimming
- Flashing
- Return to the main menu

Figure 13. White LED submenu



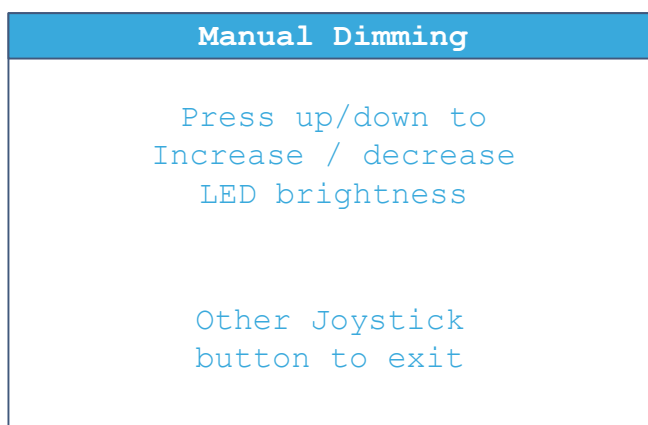
4.3.2 Automatic dimming

The **[Auto Dimming]** option starts automatic white LED dimming: this mode scans the entire brightness range, up and down successively. Press any joystick key to return to the white LED submenu.

Figure 14. White LED automatic dimming


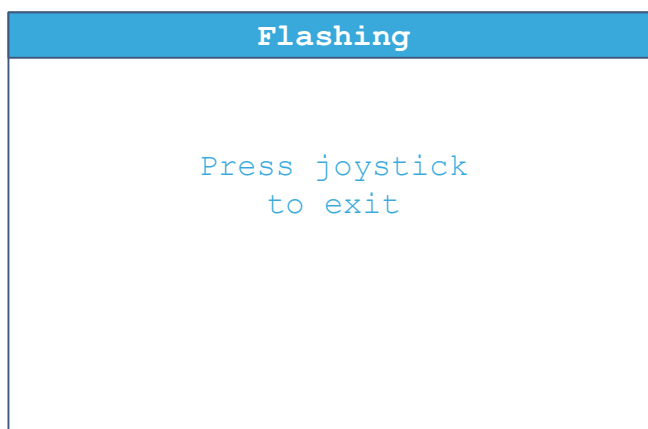
4.3.3 Manual dimming

The **[Manual Dimming]** option starts manual control of the white LED brightness: press the UP or DOWN joystick keys to increase or decrease white LED brightness. Press any other joystick key to return to the white LED submenu.

Figure 15. White LED manual dimming


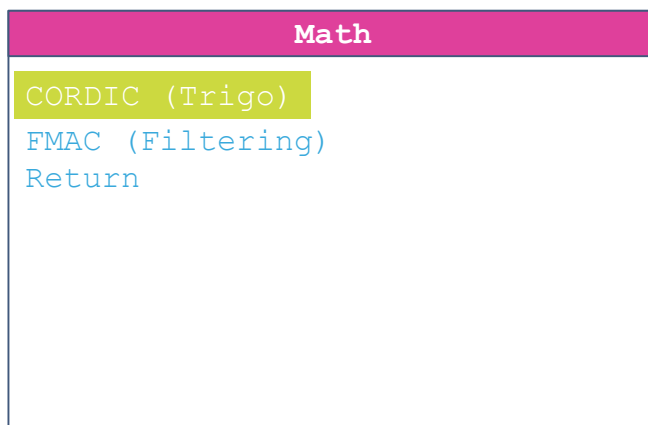
4.3.4 Flashing

The **[Flashing]** option starts manual control of the white LED brightness: the high-brightness LED is turned on with a 10 Hz frequency at full power ($t_{ON} = 20$ ms, $t_{OFF} = 80$ ms). Press any joystick key to return to the white LED submenu.

Figure 16. White LED flashing


4.4 Math

The Math application demonstrates the capabilities of the CORDIC and FMAC peripherals to compute trigonometric functions and filter signals, in order to offload the Cortex[®]-M4 processor of the STM32G474QET6.

Figure 17. Math application submenu


For technical details about the CORDIC and FMAC peripherals, refer to [\[3\]](#).

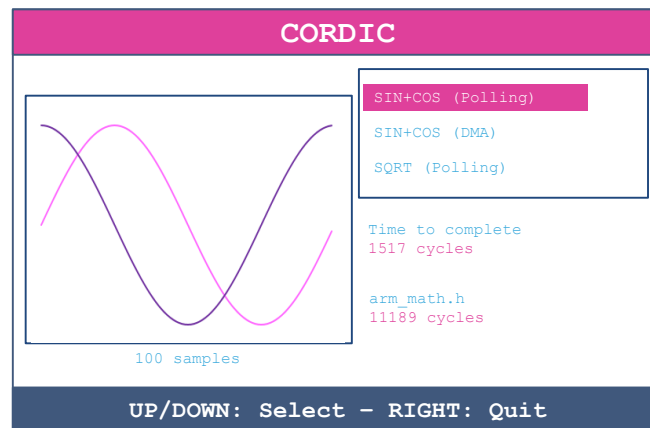
4.4.1 CORDIC

The CORDIC application is accessed through the **[CORDIC (Trigo)]** option of the Math submenu.

It is possible to compute:

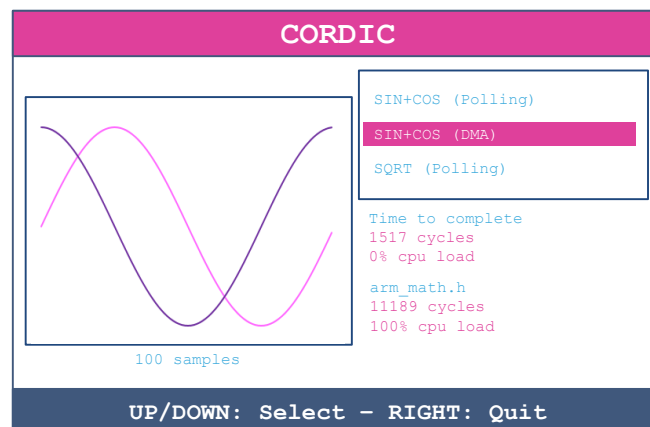
- Sine and cosine in polling mode⁽¹⁾

Figure 18. CORDIC sine and cosine (polling mode)



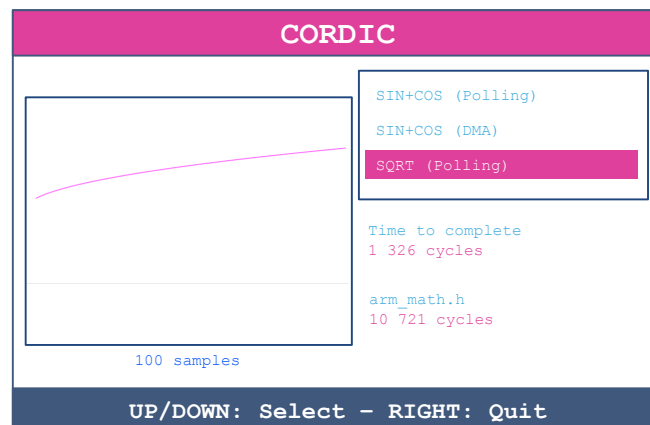
- Sine and cosine in DMA mode⁽²⁾

Figure 19. CORDIC sine and cosine (DMA mode)



- Square root in polling mode⁽³⁾

Figure 20. CORDIC sine and cosine (polling mode)



1. Sine plotted in pink and cosine plotted in purple.

2. Sine plotted in pink and cosine plotted in purple.
3. Square root plotted in pink.

The demonstration also compares the performance measured in number of cycles between the CORDIC peripheral and Arm® mathematical library.

4.4.2 FMAC

The FMAC application is accessed through the **[FMAC (Filtering)]** option of the Math submenu.

It is possible to apply either a “FIR 51” filter or an “IIR 8” filter on the input signal, which is the sum of a 1 kHz sinusoid and of a 21 kHz sinusoid.. Both filters remove the 21 kHz sinusoid and keep the 1 kHz sinusoid (refer to [Figure 21](#) and [Figure 22](#)).

As the order of the FIR filter is greater than the one of the IIR, more samples are required to get the result and the filtered graph appears delayed.

The application also shows the gain in CPU load brought by the FMAC peripheral when compared to the Arm® mathematical library.

Figure 21. FMAC FIR filtering

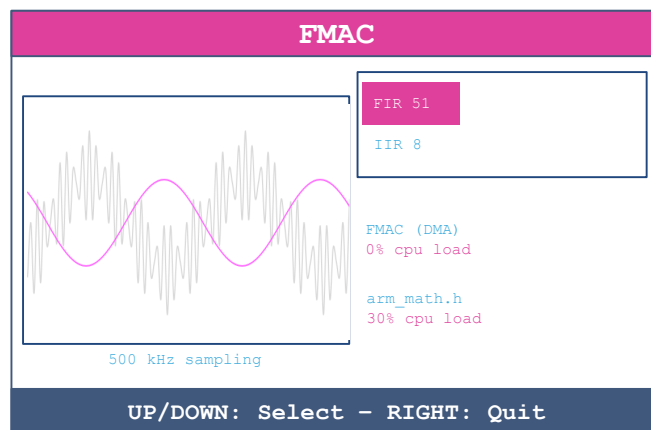
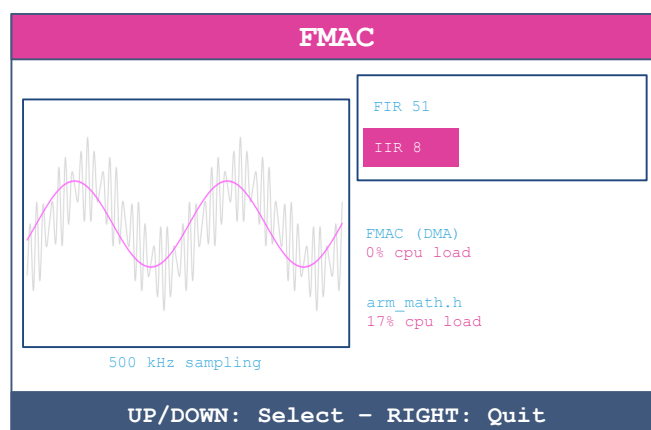


Figure 22. FMAC IIR filtering



4.5 UCPD demonstration

4.5.1 Warning

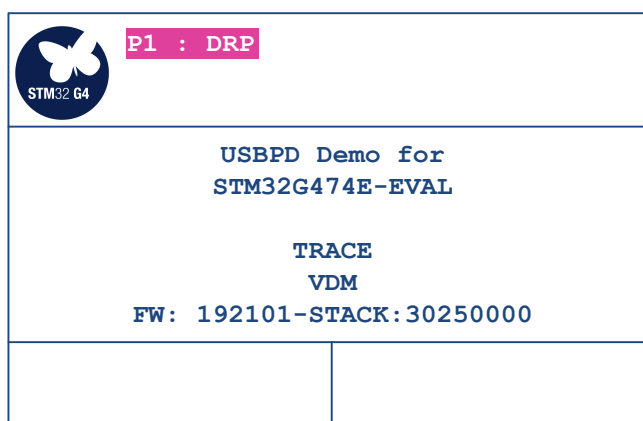
Running the UCPD demonstration may cause the power delivered to damage the device plugged onto the [STM32G474E-EVAL](#) Evaluation board. STMicroelectronics may not be held responsible for all issues encountered.

4.5.2 Hardware checks

Before running the USB-PD demonstration, make sure that the configuration is correct (refer to [Section 2.2 Hardware settings](#)).

Check that the right firmware is programmed.

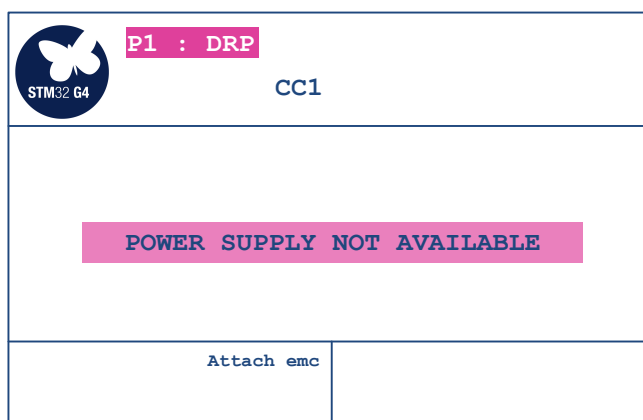
Figure 23. USB-PD firmware revision compatibility



4.5.3 Emergency state

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

Figure 24. USB-PD emergency state screen

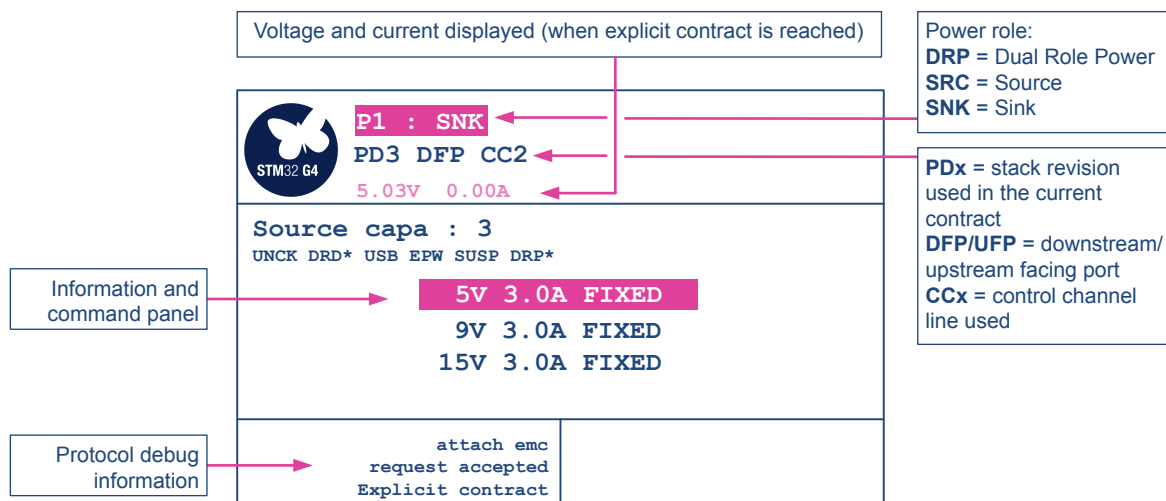


To exit from this state, the user must check the power supply and reset firmware.

4.5.4 Overall presentation

Plug the USB Type-C™ cable to a USB Type-C™ device. The information on the LCD screen is presented as follows:

Figure 25. USB-PD demonstration screen



4.5.5 Navigation in the menus

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
- SEL: executes the selected action
- Blue USER button: exits the UCPD demonstration

4.5.6 Available panels


One command panel and several information panels are available.

Command panel

Use the joystick up and down keys to scroll across the following available commands (refer to Figure 26):

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


Figure 26. USB-PD - Available commands

 P1 : SRC PD3 DFP CC2 5.03V 0.00A	
Commands : HARD RESET GOTO MIN GET SOURCE CAPA GET SINK CAPA	
attach emc request accepted Explicit contract	

Source capabilities / Sink capabilities

This information panel depends on the role of the port. Figure 27 shows an example of source capabilities:

Figure 27. USB-PD - Source / sink capability

 P1 : SNK PD3 DFP CC2 5.03V 0.00A	
Source capa : 3 UNCK DRD* USB EPW SUSP DRP 5V 3.0A FIXED 9V 3.0A FIXED 15V 3.0A FIXED	
attach emc request accepted Explicit contract	

- 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:
 - UNCK: Unchunked message support
 - DRD: Dual-Role Data (possibility to swap data role)
 - USB: USB available
 - EPW: Externally powered
 - SUSP: USB Suspend Supported
 - 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 28.

Figure 28. USB-PD - Power profile selection

	P1 : SRC PD3 DFP CC2 9.03V 0.00A
	Source capa : 3 FRS DRD* USB EPW SUSP DRP 5V 3.0A FIXED 9V 3.0A FIXED 15V 3.0A FIXED
request sent request accepted Explicit contract	

- Pressing the down key of the joystick once reaches the 9 V profile
- Selecting the 9 V profile with the SEL button of the joystick sends a request to switch to 9 V from the port 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 29 shows the extended capabilities available when PD3 is supported (scroll with the joystick).

Figure 29. USB-PB - Extended capabilities

	P1 : SRC PD3 DFP CC2 9.03V 0.00A
	Extended capa : VID:0x1 PID:0x2 XID:0xf0000003 F rev:0x1 H rev:0x2
attach request accepted Explicit contract	

- 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 29:

- 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].

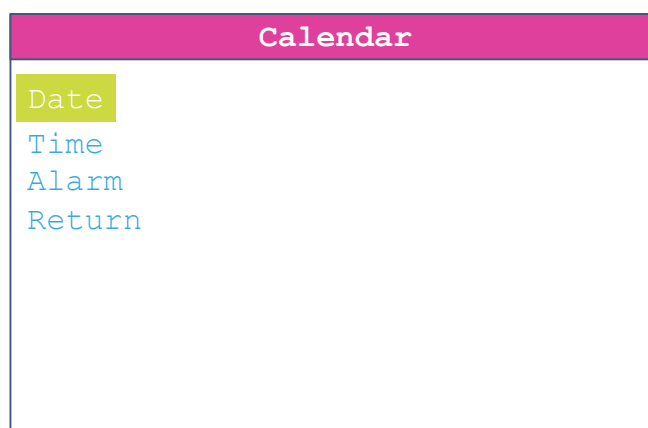
4.6 Calendar application

The STM32G474QET6 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 wakeup flag with interrupt capability.

4.6.1 Calendar submenu

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

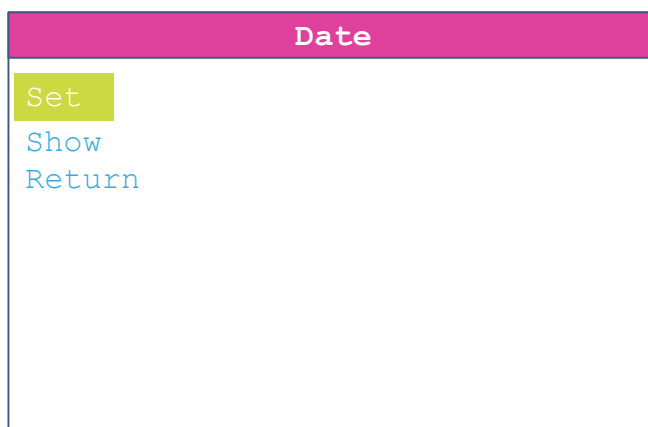
Figure 30. Calendar submenu



4.6.2 Date setting

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

Figure 31. 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 right side of the upper line. Press the SEL joystick key when the desired year is reached.

Figure 32. Setting the year

NOVEMBER				2017		
WEEK: 44				DAY: 306		
Mo	Tu	We	Th	Fr	Sa	Su
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			
UP/DOWN: Select Year						
Sel: to set						

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.

Figure 33. Setting the month

NOVEMBER				2017		
WEEK: 44				DAY: 306		
Mo	Tu	We	Th	Fr	Sa	Su
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			
UP/DOWN: Select Month						
Sel: to set						

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

Figure 34. Setting the day

November 2017						
WEEK: 46				DAY: 320		
Mo	Tu	We	Th	Fr	Sa	Su
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			
UP/DOWN/LEFT/RIGHT: Select Day						
Sel: to set						

The current date can be consulted through the **[Show]** option of the Date submenu.

Figure 35. Consulting the date

Date
Wed. 16.11.2017

4.6.3 Time setting

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

Figure 36. Time submenu

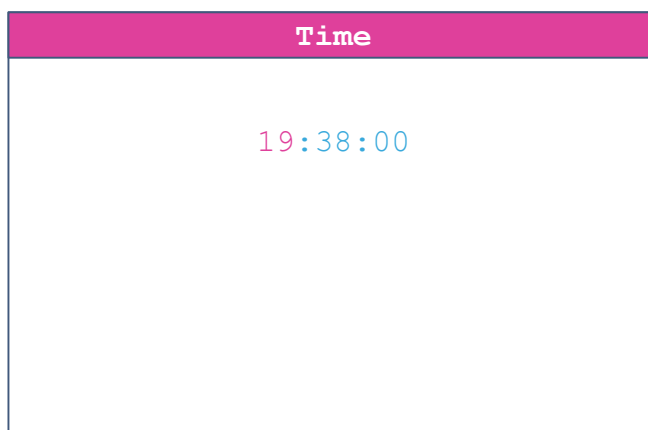
Time
Set
Show
Return

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 DOWN or UP joystick keys to increase or decrease the selected number. Keep continuously pressing for incrementing or decrementing faster.

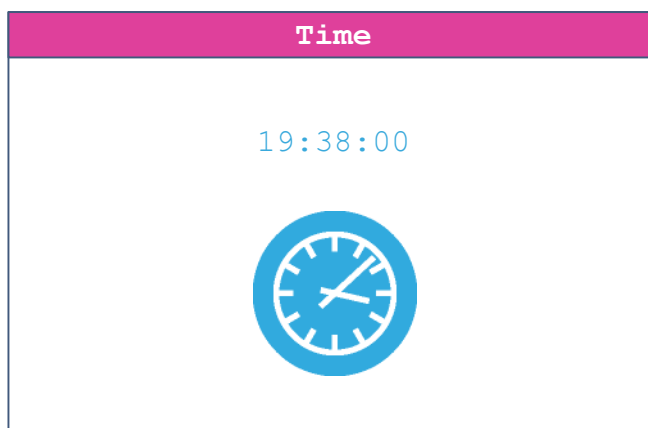
When the desired values are selected, press the SEL joystick key to set these as the current time.

Figure 37. Time setting



The current time can be consulted through the **[Show]** option of the Time submenu.

Figure 38. Consulting the time

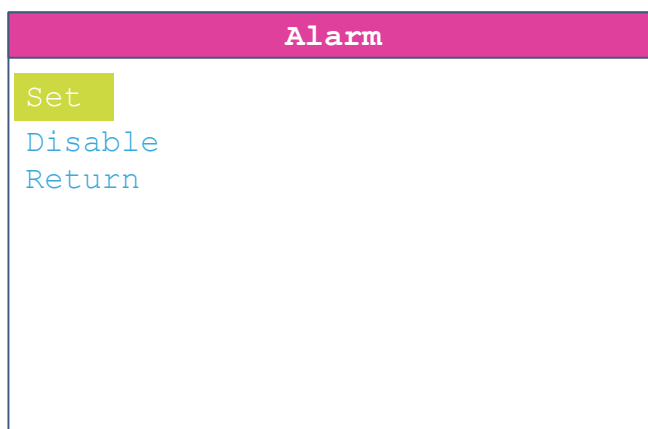


4.6.4

Alarm setting

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

Figure 39. Alarm submenu

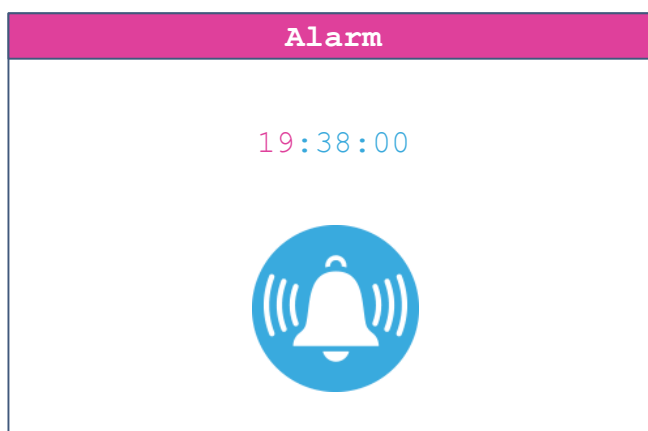


The alarm is set by selecting hours (in the 24-hour format), minutes, and seconds:

- 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.

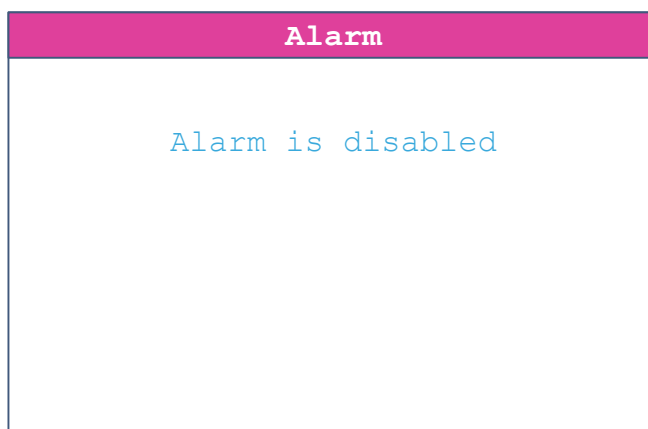
When the desired values are selected, press the SEL joystick key to set the alarm accordingly.

Figure 40. Alarm setting



When the current time reaches the value set for the 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 41. Alarm disable



4.7 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 in the list. Quit the application by pressing the DOWN key of the joystick and then return to the main menu.

4.8 Audio application

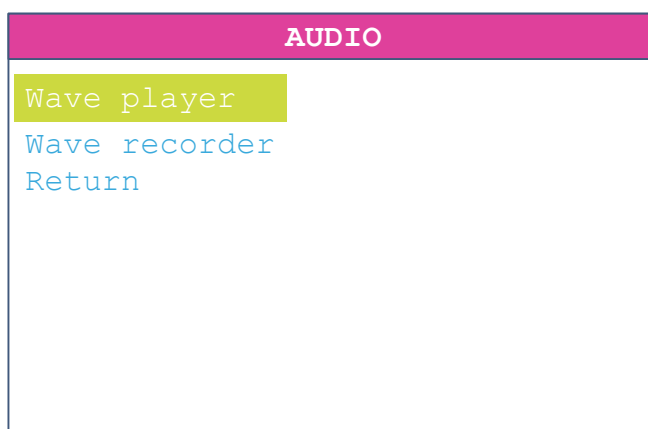
The audio application provides means to playback .wav files and record voice using STM32G474QET6 SAI peripheral connected to the WM8994 external codec.

Connect the Micro or IN line to the CN23 connector, and the headphone or Output line to the CN26 connector.

4.8.1 Audio submenu

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

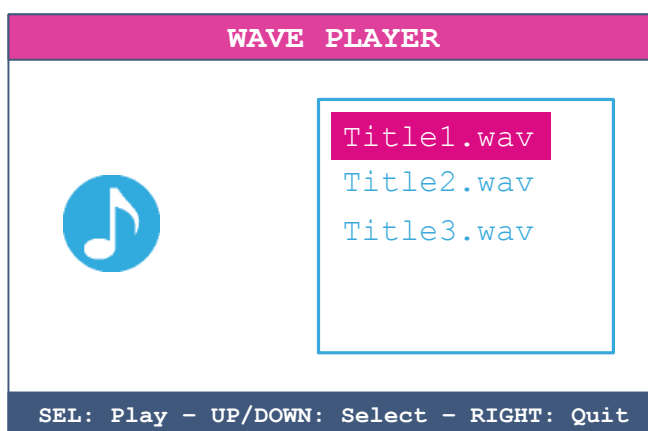
Figure 42. Audio submenu



4.8.2 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 in a listing the .wav files in the USER folder of the microSD™ card. The playlist size is limited to four titles.

Figure 43. Audio playlist



Use the UP and DOWN keys of the joystick to browse among available titles. Press the SEL key of the joystick to play the selected title. Figure 44 represents the aspect of the LCD screen when the wave file is being played.

Figure 44. Audio playing



Pause the playback by pressing the SEL key of the joystick (press the SEL key again to resume playback). Press the DOWN key of the joystick to stop the playback (press the SEL key again restart playback from the beginning). Figure 45 represents the aspect of the LCD screen when playback is paused.

Figure 45. Audio paused

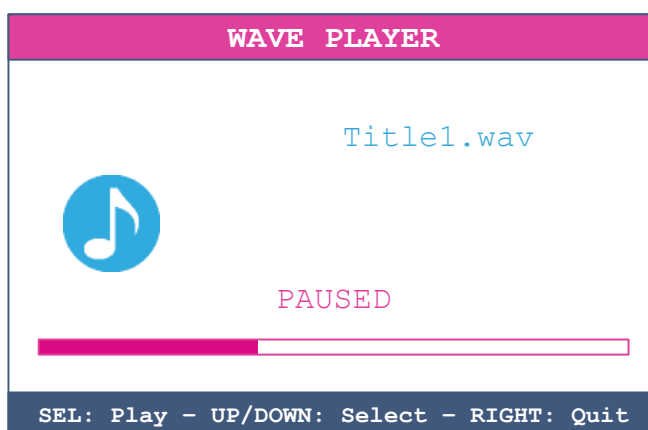
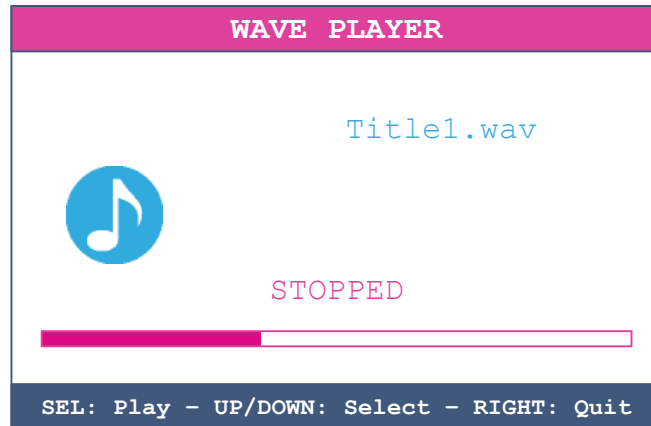


Figure 46 represents the aspect of the LCD screen when playback is stopped.

Figure 46. Audio stopped



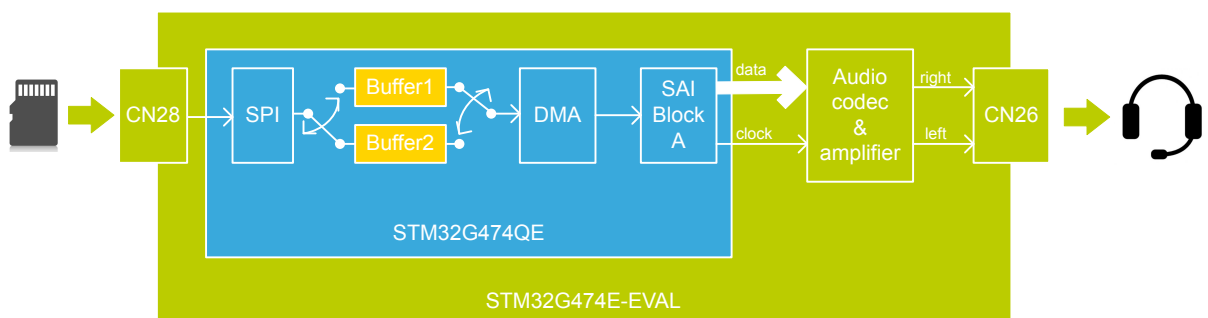
Note: The audio files provided within this package are based on free-music download from the danosongs.com website.

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 SAI. The SAI triggers the DAC conversions of the codec to generate the wave signal at the desired sample rate.

The wave player data path is shown in Figure 47.

Figure 47. Wave player data path



4.8.3

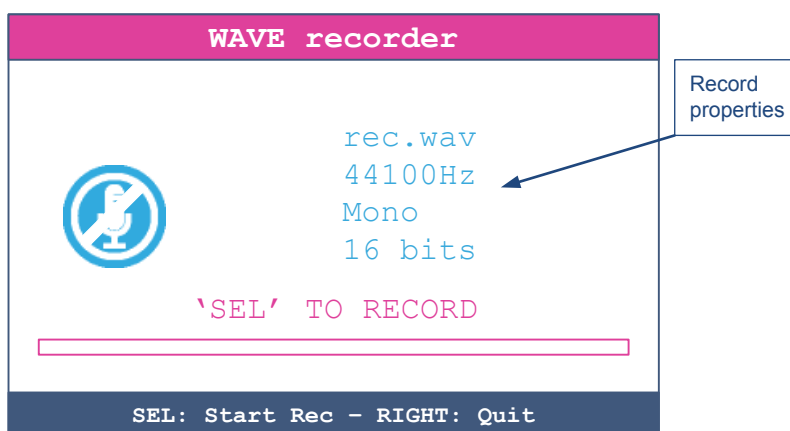
Wave recorder

When the **[Wave Recorder]** option of the audio submenu is selected, the LCD displays the LCD audio record page (refer to Figure 48). When recording is started, the `rec.wav` file is created in the `USER` folder of the microSD™ card. If this file exists already, its content is erased and the file is considered as a new empty file.

The recorder audio file has the following properties:

- Sampling rate: 44100 samples/s
- Channels: stereo (left = right)
- Resolution: 16 bits/sample

Figure 48. Audio recorder start



Start recording by pressing the SEL 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 49. Wave recording



Press the SEL key of the joystick to stop the recording (refer to Figure 50). Pressing the SEL key again starts a new recording.

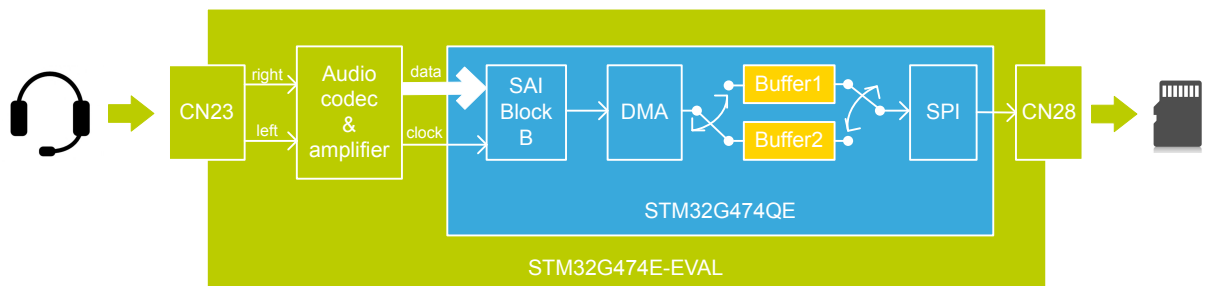
Figure 50. Recording stopped



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 44100 Hz rate. ADC conversions are triggered by the SAI in order 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 51. Wave recorder data path



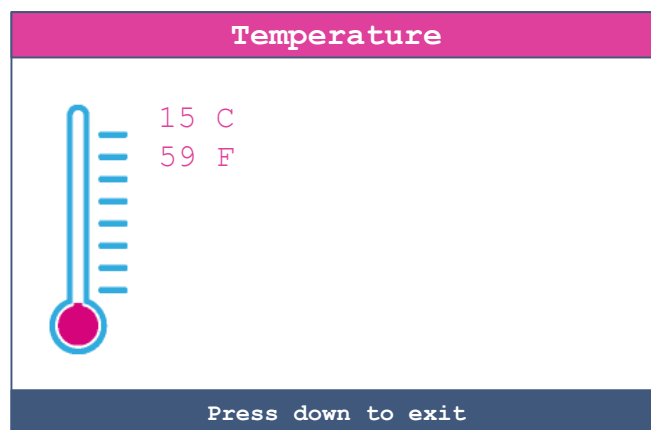
4.9 Thermometer

The STM32G474QET6 microcontroller has four embedded I²C peripherals for connection to any device supporting the I²C protocol.

In this demonstration, the [STTS751](#) I²C temperature sensor mounted on the [STM32G474E-EVAL](#) Evaluation board is used to capture the external temperature (-40°C to +125°C).

Select the **[Thermometer]** submenu of the demonstration menu by pressing the SEL push-button of the joystick. The temperature value is displayed in Celsius and Fahrenheit degrees as shown in [Figure 52](#).

Figure 52. Temperature screen



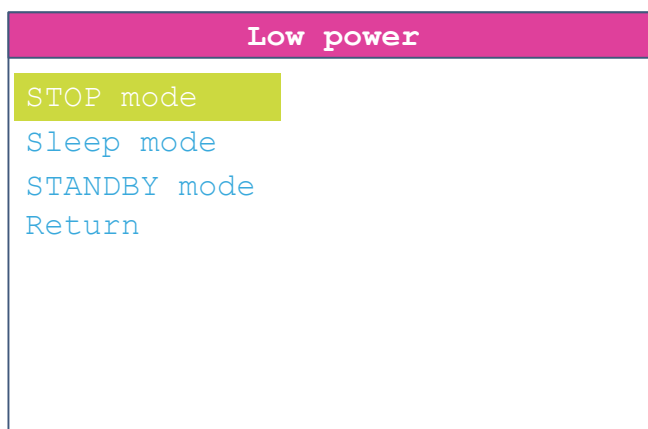
4.10 Low-power mode application

The STM32G474QET6 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, Standby, and Sleep modes are used as examples.

4.10.1 Low-power mode submenu

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

Figure 53. Low-power submenu



4.10.2

Stop mode

The Stop mode submenu (refer to Figure 54) allows the user to put the STM32G474QET6 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 54. Stop mode submenu

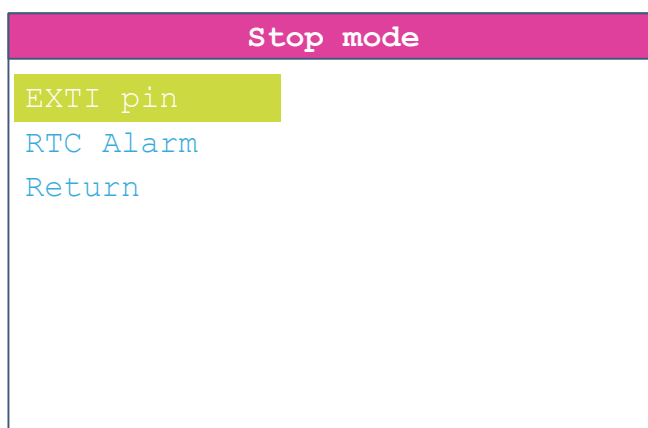


Figure 55. Start Stop mode (EXTI)



Figure 56. Stop mode activated (EXTI)

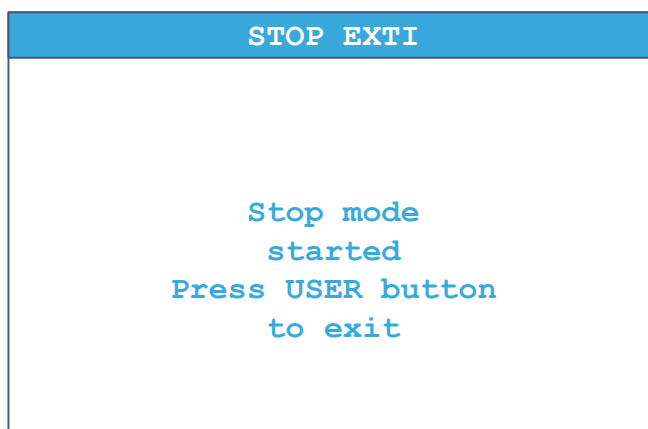


Figure 57. Exit from Stop mode (EXTI)



Figure 58. Start Stop mode (alarm)

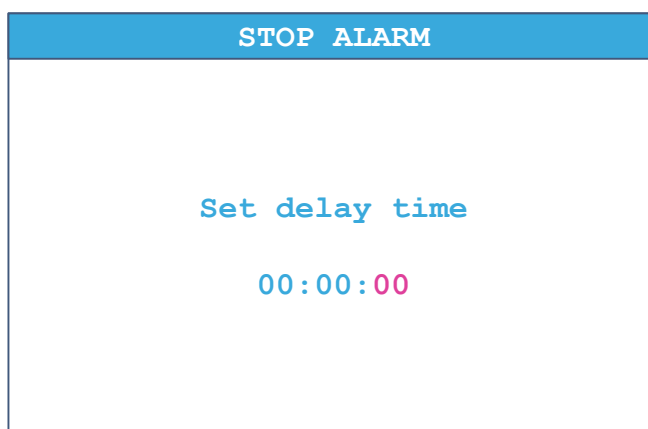
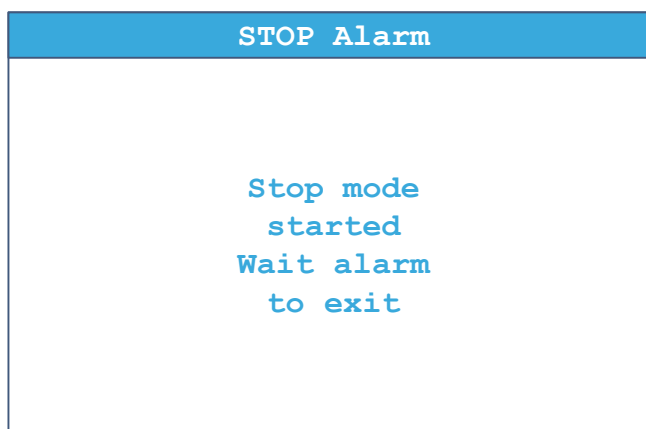


Figure 59. Exit from Stop mode (alarm)

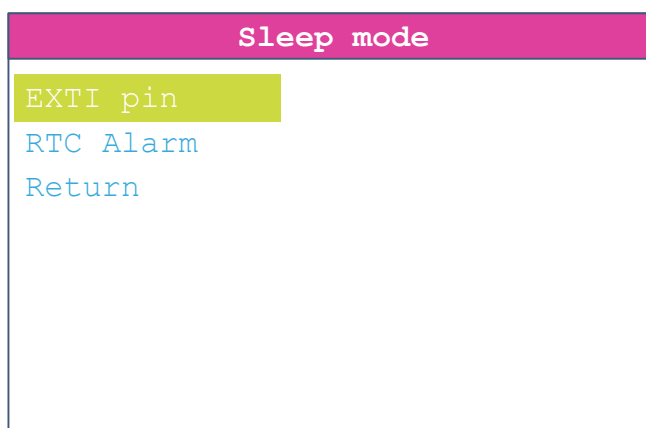


4.10.3

Sleep mode

The Sleep mode submenu (refer to Figure 60) allows the user to put the STM32G474QET6 in Sleep mode with its low-power regulator on. It is possible to select between the EXTI (WFI) or RTC alarm (WFI) to exit from the Sleep mode.

Figure 60. Sleep mode submenu



4.10.4

Standby mode

The Standby mode submenu allows the user to put the STM32G474QET6 in Standby mode. It is possible to select between the Wakeup pin or RTC alarm to exit from the Standby mode.

Figure 61. Standby mode

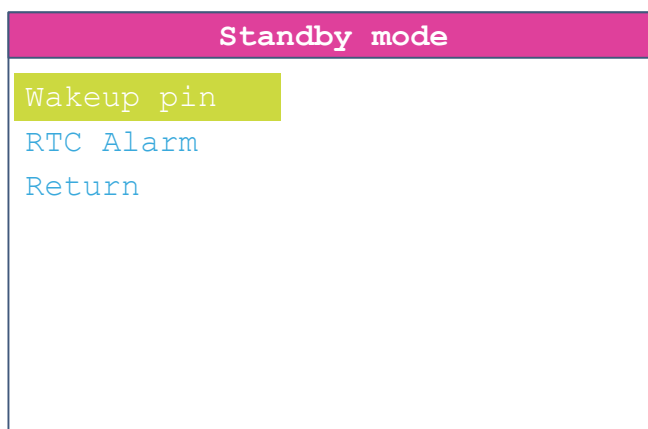


Figure 62. Standby mode entry (wake-up pin)



Figure 63. Standby mode exit (wake-up pin)



Figure 64. Standby mode entry (alarm)

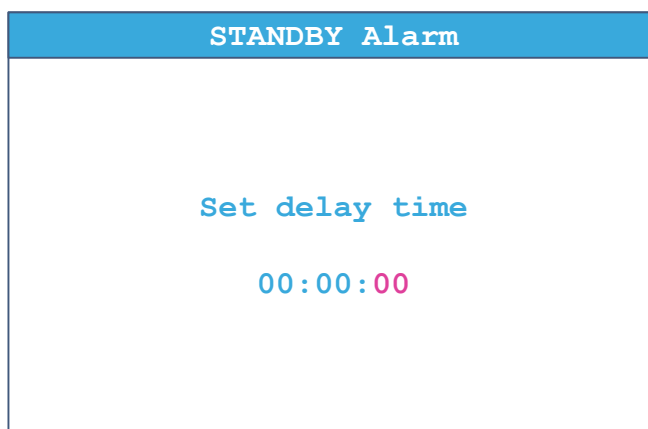
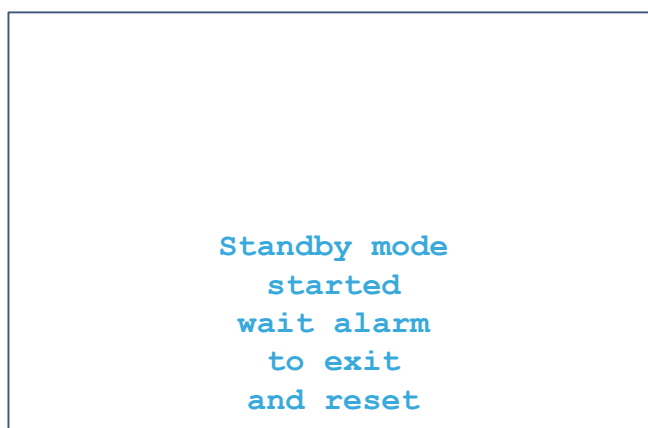


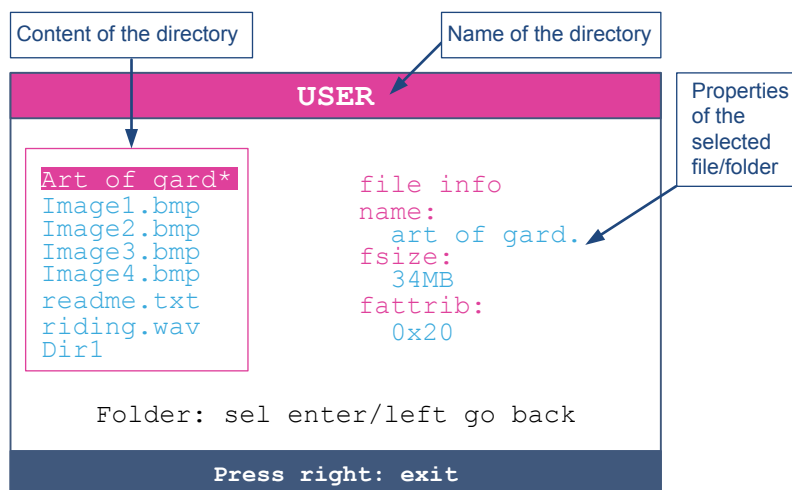
Figure 65. Standby mode exit (alarm)



4.11 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 66.

Figure 66. File browser



Use the UP and DOWN keys of the joystick to select an item. Press the SEL 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.

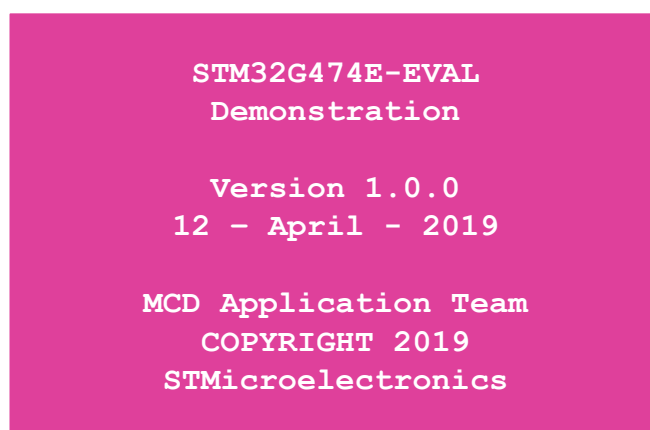
When a file is selected, the following fields of the FatFS file information structure are displayed on the right side of the LCD screen:

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

4.12 About application

The About submenu shows the version and date of the [STM32G474E-EVAL](#) demonstration firmware. When the About is selected, the message shown in [Figure 67](#) is displayed on the LCD screen. Press the DOWN key to return to the main menu.

Figure 67. About submenu

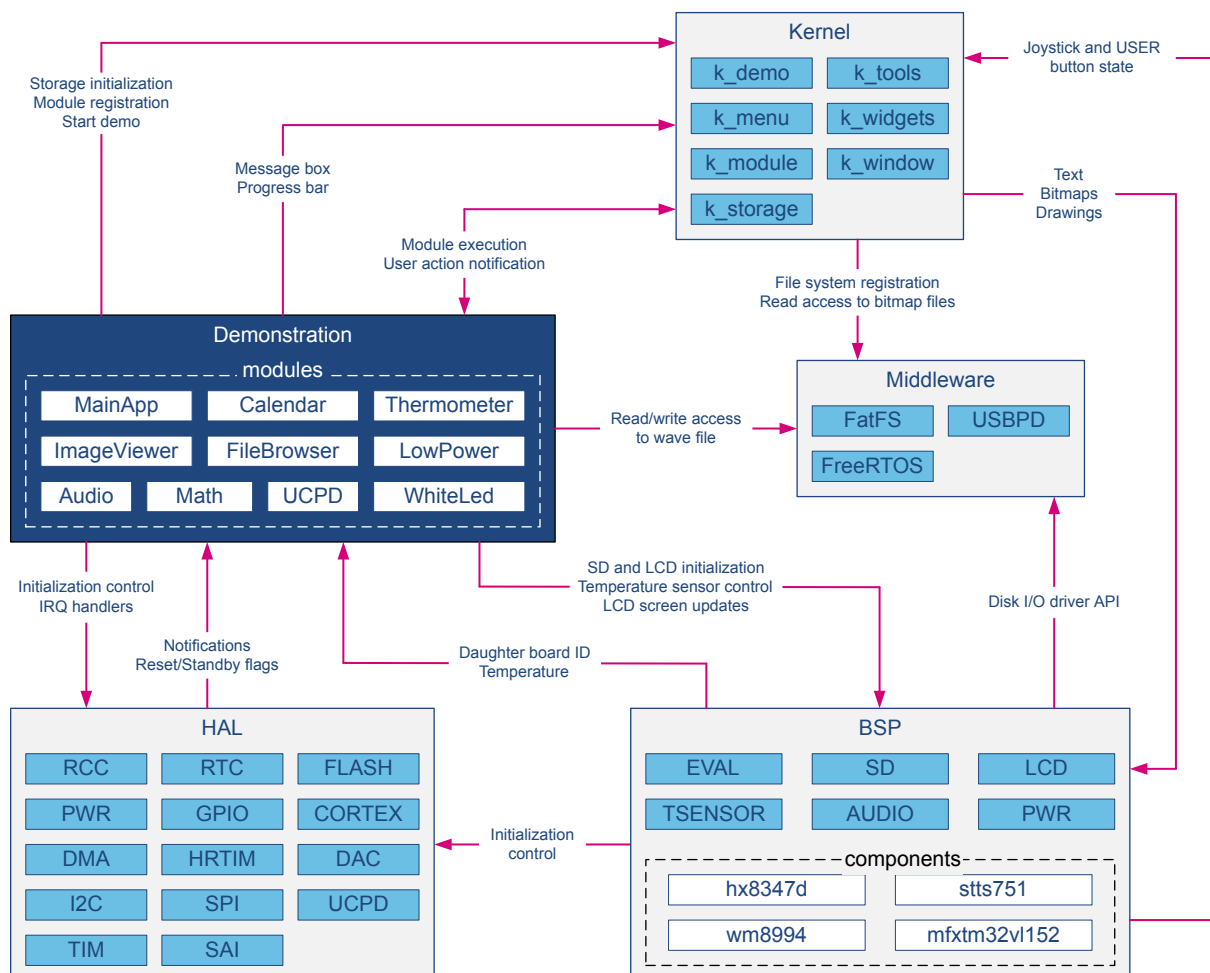


5 Software architecture

5.1 Demonstration

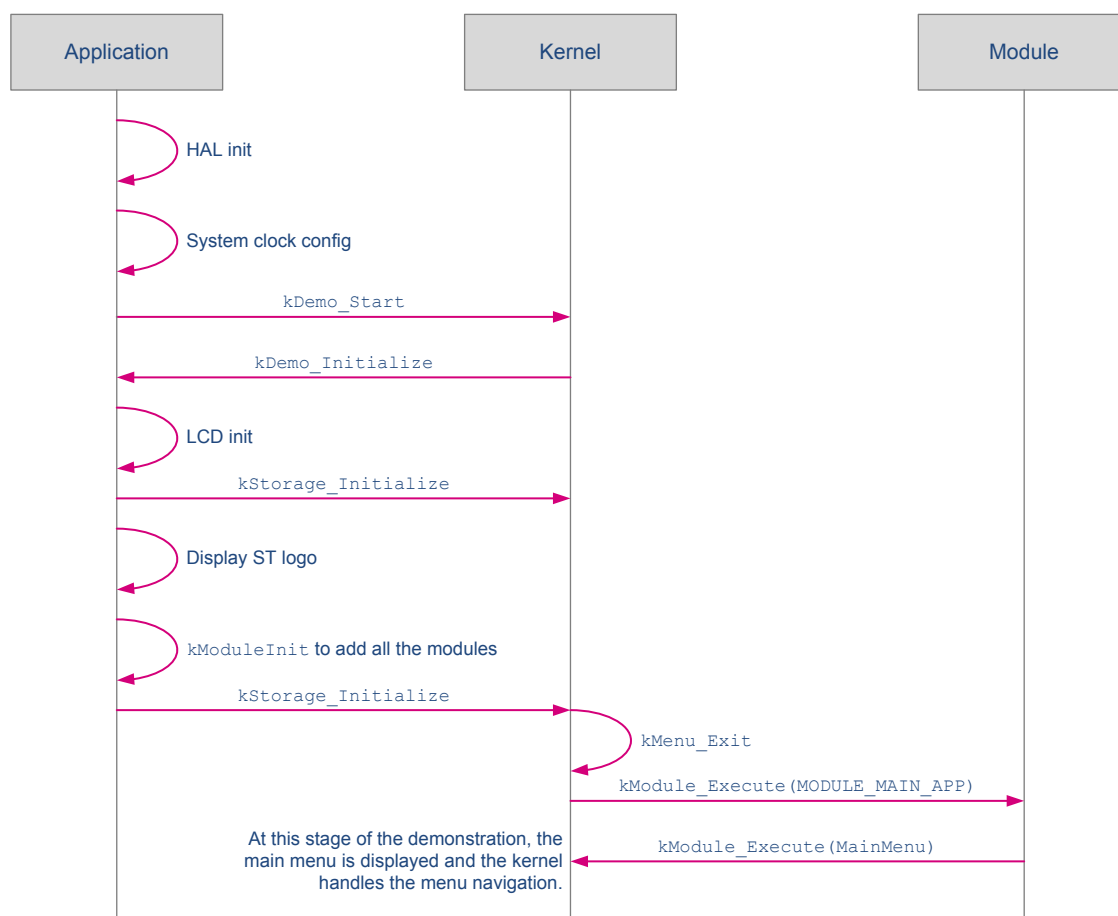
Figure 68 represents the main interactions between the demonstration and the surrounding software modules. The demonstration is a set of modules accessible through a main menu displayed on the LCD screen. At application startup, the system clock is configured at 170 MHz, and the file system, SD card, and LCD BSPs are initialized. If all the graphic resources (bitmap files) needed by the modules are present in the microSD™ card, each module is declared to the kernel (registration step). From this point onward, the kernel handles menu navigation, whatever the menu level (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 STM32G474QET6 interrupts (RTC, EXTI, HRTIM and DMA) are handled by the application (refer to Figure 68) and are used, as usual, to map the interrupt vector on the driver HAL driver, depending on the module requirement.

Figure 68. Main demonstration software architecture block diagram



The sequence diagram in Figure 69 summarizes the application startup until the kernel takes the lead.

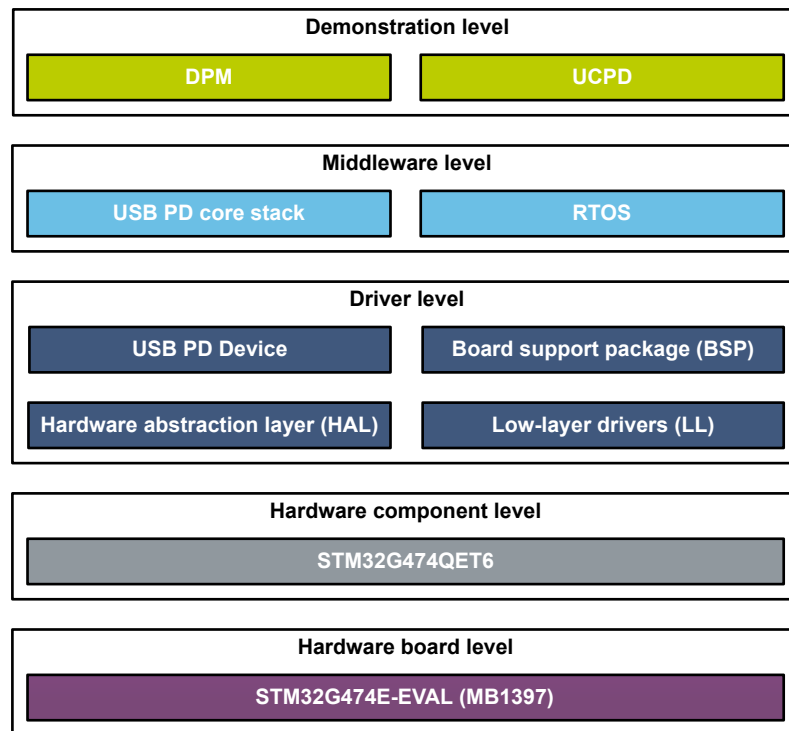
Figure 69. Simplified application startup sequence diagram



5.2 UCPD demonstration

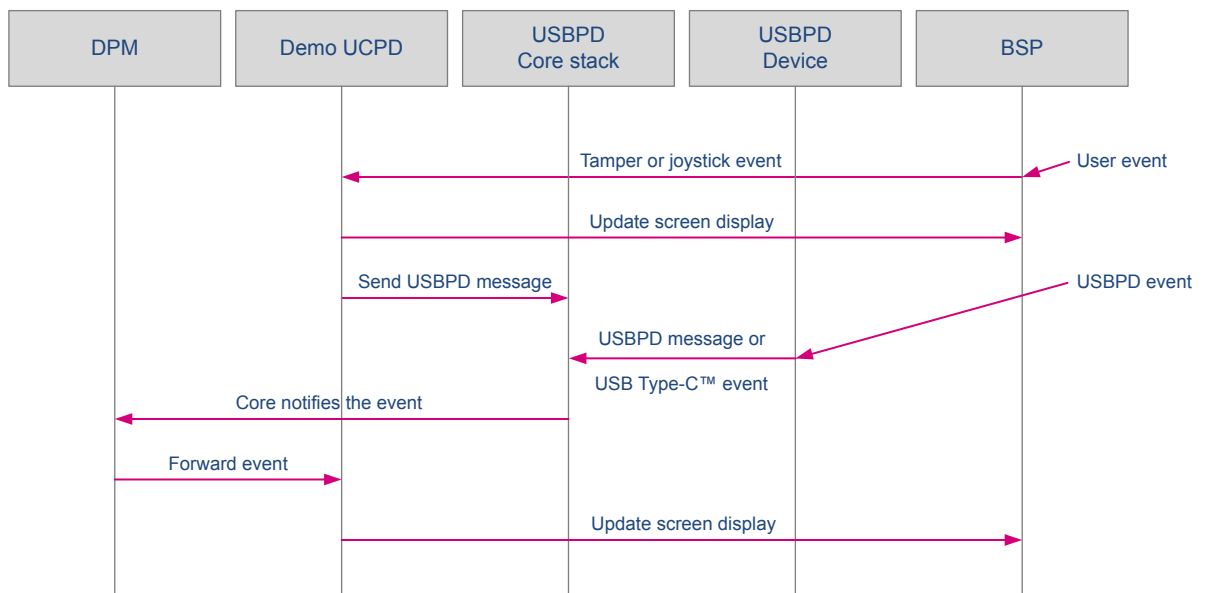
Figure 70 represents the main functional blocks involved in the UCPD demonstration. The UCPD demonstration is an RTOS task which receives events forwarded by DPM (Device Policy Manager). The events are mainly coming from the USBPD stack (such as attachment, detachment, and others). The demonstration 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 the port as DRP, and the demo is ready to catch DPM events. The user uses the USER button and joystick to navigate inside the menu and execute actions.

Figure 70. UCPD demonstration software architecture block diagram



The sequence diagram in Figure 71 summarizes the application processing of the UCPD application.

Figure 71. 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 application side and called by `kDemo_Start` (see the application chapter for details about these functions).

Table 5. `k_demo` API

Function	Description
<code>kDemo_Initialization</code>	Initializes the demo (function defined on application side).
<code>kDemo_UnInitialization</code>	Initializes the demo (function defined on application side).
<code>kDemo_Start</code>	Starts the demo (Initialize, Run, UnInitialize, Exit).

5.3.2

`k_menu`

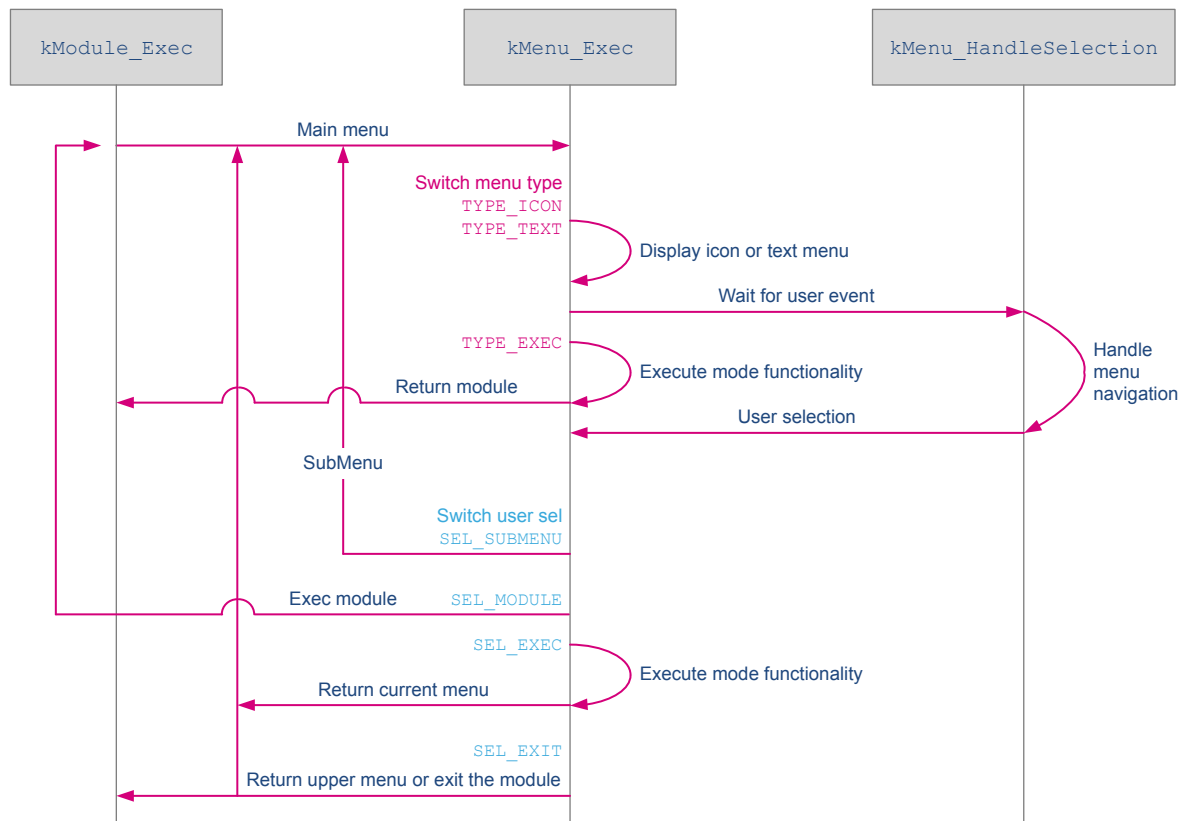
The module execution is started by a call to function `kMenu_Execute`. This kernel function handles navigation in the menu and the execution functionalities (by means of the structure `t_menu` defined inside the module).

Table 6. `k_menu` API

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

Figure 72 shows the execution flow for a menu.

Figure 72. kMenu sequence diagram



5.3.3 k_module

This kernel part centralizes information about all modules available inside demonstration firmware: function `kModule_Init` (defined on application side) registers all the modules present with the help of function `kModule_Init` (by means of the structure `K_ModuleItem_Typedef` defined inside the module).

Table 7. k_module API

Function	Description
<code>kModule_Init</code>	Defined on application side.
<code>kModule_Add</code>	Adds module inside the module list.
<code>kModule_CheckResource</code>	Checks the module resource.
<code>kModule_Execute</code>	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 8. k_storage API

Function	Description
<code>kStorage_Init</code>	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.
kStorage_GetFree	Gets free storage space.

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 9. 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 10. 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 11. k_demo API

Function	Description
kTools_Buffercmp	Returns 0 if both buffers are identical.

5.4 FatFS API overview

The set of functions exposed by the FatFS file system interface is described in [\[5\]](#).

5.5 USBPD API overview

The set of functions exposed by the USB-PD interface is described in [\[6\]](#).

5.6 BSP API overview

The set of functions exposed by the BSP interface is described in [\[7\]](#).

6 Demonstration memory footprint

Table 12 summarizes the RAM and Flash memory allocations for the demonstration:

- *ro code* is the number of bytes used in the read-only code memory
- *rw code* is the number of bytes used in the read-write code memory
- *ro data* is the number of bytes used in the read-only data memory
- *rw data* is the number of bytes used in the read-write data memory

Table 12. Demonstration memory footprint

Module	ro code	rw code	ro data	rw data
Core				
k_demo.o	24	-	-	-
k_menu.o	1032	-	-	-
k_module.o	112	-	-	220
k_storage.o	736	-	-	629
k_widgets.o	216	-	-	-
k_window.o	244	-	-	-
Demo				
main.o	1132	-	-	48
stm32g4xx_it.o	166	-	-	-
stm32g4xx_hal_msp.o	116	-	-	-
EWARM				
startup_stm32g474xx.o	840	-	-	-
Modules				
Audio				
app_audio.o	56	-	160	8758
wave_player.o	1916	-	92	472
wave_recorder.o	1536	-	-	8
Calendar				
app_calendar.o	3280	-	560	48
FileBrowser				
app_filesbrowser.o	1568	-	88	2586
ImageViewer				
app_imagesbrowser.o	232	-	52	2
LowPower				
app_lowpower.o	2964	-	620	2
MainApp				
app_main.o	544	-	857	321
Math				
app_math.o	4620	-	176	582

Module	ro code	rw code	ro data	rw data
Thermometer				
app_thermometer_LDR.o	1284	-	92	102
UCPD				
demo_application.o	14724	-	220	825
app_ucpd.o	196	-	20	12
WhiteLed				
app_whiteled.o	2596	-	218	384
Drivers				
BSP				
Components				
hx8347d.o	2632	-	58	760
hx8347d_reg.o	24	-	-	-
mfxstm321152.o	2140	-	57	120
mfxstm321152_reg.o	24	-	-	-
stts751.o	1204	-	25	52
stts751_regs.o	24	-	-	-
wm8994.o	3036	-	35	73
wm8994_reg.o	354	-	-	-
STM32G474E-EVAL				
stm32g474e_eval.o	1192	-	19	56
stm32g474e_eval_audio.o	2508	-	-	764
stm32g474e_eval_bus.o	1618	-	-	2180
stm32g474e_eval_env_sensor.o	320	-	-	60
stm32g474e_eval_io.o	726	-	-	60
stm32g474e_eval_lcd.o	974	-	44	56
stm32g474e_eval_sd.o	2978	-	4	16
stm32g474e_eval_usbpd_pwr.o	884	-	-	-
CMSIS				
system_stm32g4xx.o	44	-	10	4
STM32G4xx_HAL_Drivers				
stm32g4xx_hal.o	200	-	6	12
stm32g4xx_hal_comp.o	684	-	-	-
stm32g4xx_hal_cordic.o	452	-	-	-
stm32g4xx_hal_cortex.o	264	-	-	-
stm32g4xx_hal_dac.o	648	-	-	-
stm32g4xx_hal_dac_ex.o	96	-	-	-
stm32g4xx_hal_dma.o	920	-	-	-
stm32g4xx_hal_exti.o	72	-	-	-

Module	ro code	rw code	ro data	rw data
stm32g4xx_hal_fmac.o	1796	-	-	-
stm32g4xx_hal_gpio.o	720	-	-	-
stm32g4xx_hal_hrtim.o	3874	-	13	28
stm32g4xx_hal_i2c.o	1424	-	-	-
stm32g4xx_hal_i2c_ex.o	134	-	-	-
stm32g4xx_hal_opamp.o	682	-	-	-
stm32g4xx_hal_pwr.o	168	-	-	-
stm32g4xx_hal_pwr_ex.o	552	-	-	-
stm32g4xx_hal_rcc.o	1864	-	-	-
stm32g4xx_hal_rcc_ex.o	1764	-	-	-
stm32g4xx_hal_rtc.o	1442	-	-	-
stm32g4xx_hal_sai.o	2002	-	-	-
stm32g4xx_hal_spi.o	1880	-	-	-
STM32G4xx_LL_Drivers				
stm32g4xx_ll_dma.o	140	-	-	-
stm32g4xx_ll_rcc.o	492	-	-	-
stm32g4xx_ll_ucpd.o	98	-	-	-
stm32g4xx_ll_usart.o	308	-	-	-
Middleware				
Fat FS				
Core				
diskio.o	120	-	-	-
ff.o	7652	-	-	204
ff_gen_drv.o	68	-	-	16
Options				
syscall.o	8	-	-	-
unicode.o	1128	-	-	-
FreeRTOS				
cmsis_os.o	328	-	-	-
queue.o	1328	-	-	-
heap_4.o	432	-	-	9524
list.o	144	-	-	-
port.o	516	-	6	12
portasm.o	176	-	-	-
task.o	1756	-	-	280
USBPD				
usbpd_cad_hw_if.o	2244	-	-	12
usbpd_dpm_core.o	708	-	72	92

Module	ro code	rw code	ro data	rw data
usbpd_dpm_user.o	2940	-	4	252
usbpd_hw.o	336	-	-	-
usbpd_hw_if_it.o	320	-	-	1
usbpd_phy.o	336	-	-	8
usbpd_phy_hw_if.o	900	-	-	64
usbpd_pwr_hw_if.o	112	-	-	-
usbpd_pwr_if.o	1124	-	36	76
usbpd_timersserver.o	656	-	-	1
usbpd_trace.o	262	-	-	-
usbpd_vdm_user.o	1432	-	25	58
Others				
basic_gui.o	1192	-	14835	140
sd_diskio.o	136	-	20	1
tracer_emb.o	266	-	-	1040
tracer_emb_hw.o	888	-	-	4
USBPD_CORE_PD3_FULL_CM4_wc32.a	25054	-	-	44
dl7M_tlf.a	4742	-	68	144
iar_cortexM4lf_math.a	1170	-	2052	-
m7M_tls.a	2488	-	-	-
rt7M_tl.a	1670	-	-	-
shb_l.a	20	-	-	-
Gaps	86	-	8	-
Linker created	-	-	32	49152
Total				
Total of all modules	145462	0	20576	80268

7 Advanced module description

7.1 Module detailed description

A module is an autonomous application that runs directly from the launcher or from 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

7.1.1 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 a main function as listed in [Table 13](#).

Table 13. K_ModuleItem_Typedef structure description

Field	Description
kModuleId	Unique ID module. This has to be defined inside the <code>MODULES_INFO</code> enumeration in the <code>k_config.h</code> include file.
kModulePreExec	Function pointer used to initialize the low-layer driver, allocate memory or execute any specific action before module execution. This function is optional.
kModuleExec	Function pointer used to start the module main application. In the current kernel architecture, this function is used to call the <code>kModuleExecute</code> function that executes the module application based on the menu structure (refer to Section 7.1.2.1). This function is mandatory.
kModulePostExec	Function pointer used to de-initialize the low-layer driver, and release the resource allocation done inside <code>kModulePreExec</code> . This function is optional.
kModuleRessouceCheck	Function pointer 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.

7.1.2 Module menu description and graphical interface

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

7.1.2.1 tMenu structure

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

Table 14. tMenu structure description

pszTitle	Character string that contains the menu title to display.
psItems	Pointer on the menu description <code>tMenuItem</code> item (refer to Section 7.1.2.2).
nItems	Number of entry inside 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 (<code>TYPE_ICON</code>), basic text string (<code>TYPE_TEXT</code>), or only code execution (<code>TYPE_EXEC</code>).

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

7.1.2.2

tMenuItem structure

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

Table 15. 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 following defined values: <ul style="list-style-type: none"> 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	Function pointer to execute selected item.
pfActionFunc	Function pointer on a dedicated callback that is used to capture the user interface selection through 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 alternative icon name and path to be displayed when the item is selected by the user.

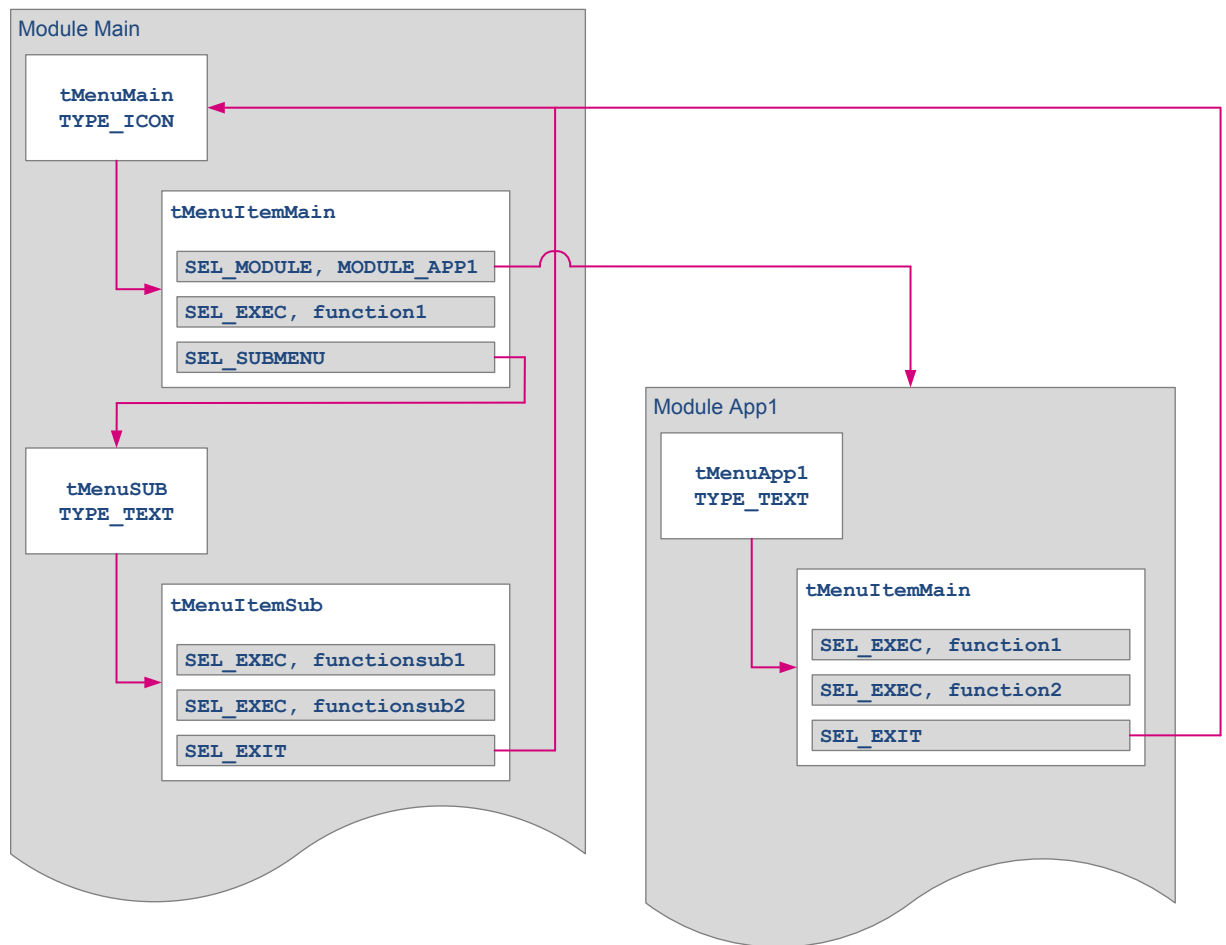
7.1.2.3

Menu example

Figure 73 illustrates the basic example of a menu structure with the use of two modules:

- The main module is built with two levels of 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 by means of `SEL_EXIT`.
- The module *App1* has a `TYPE_TEXT` context with 2 embedded functionalities: *function1* and *function2*.

Figure 73. Module menu architecture example



7.1.3 Functionality

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

- Menu event that is executed when 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).

7.2 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:

- Define the new module unique ID in file `k_config.h`
- The `kModule_Add()` function must be called in `KDemo_Initialization()`, with the module unique ID as parameter
- 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)
- Add any possible resource file into the microSD™ card as explained above

8 Acronyms

Table 16 presents the definitions of the acronyms that are relevant for a better contextual understanding of this document.

Table 16. 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.

9 References

Table 17. References

ID	Description
[1]	USB-IF (2017). <i>Universal Serial Bus Power Delivery Specification rev 3.0</i> .
[2]	<i>Evaluation board with STM32G4xxQE MCU</i> user manual (UM2514).
[3]	<ul style="list-style-type: none"> <i>Getting started with the CORDIC accelerator on STM32G4 Series microcontrollers</i> application note (AN5325) <i>Digital filter implementation with the FMAC using STM32CubeG4 MCU Package</i> application note (AN5305)
[4]	<i>High brightness LED dimming using the STM32F334 Discovery kit</i> application note (AN4885).
[5]	<ul style="list-style-type: none"> <i>Developing Applications on STM32Cube with FatFs</i> user manual (UM1721) <i>Developing Applications on STM32Cube with RTOS</i> user manual (UM1722)
[6]	<i>Managing USB Power Delivery systems with STM32 microcontrollers</i> user manual (UM2552).
[7]	<i>Getting started with STM32CubeG4 for STM32G4 Series</i> user manual (UM2492).
[8]	<i>Description of STM32G4 HAL and low-layer drivers</i> user manual (UM2570).

Revision history

Table 18. Document revision history

Date	Version	Changes
3-Jun-2019	1	Initial release.

Contents

1	STM32CubeG4 main features	2
2	Getting started with the demonstration	3
2.1	Hardware requirements	3
2.2	Hardware settings	3
2.3	microSD™ status	3
2.4	Demonstration firmware	4
3	Demonstration firmware package	5
3.1	Demonstration repository	5
3.2	Demonstration architecture overview	7
3.2.1	Demonstrations	7
3.2.2	UCPD	7
3.2.3	HAL level	7
3.2.4	Kernel	7
3.2.5	Middleware	7
3.3	Microcontroller resources	8
3.3.1	Peripherals	8
3.3.2	Interrupts	9
3.3.3	Internal memory size	10
3.3.4	External memory organization	10
4	Running the demonstration	11
4.1	Demonstration startup	11
4.1.1	Normal processing	11
4.1.2	Error cases	11
4.2	Main menu	12
4.3	White LED	12
4.3.1	White LED submenu	13
4.3.2	Automatic dimming	13
4.3.3	Manual dimming	14
4.3.4	Flashing	14
4.4	Math	15

4.4.1	CORDIC	15
4.4.2	FMAC	17
4.5	UCPD demonstration	17
4.5.1	Warning	17
4.5.2	Hardware checks	18
4.5.3	Emergency state	18
4.5.4	Overall presentation	18
4.5.5	Navigation in the menus	19
4.5.6	Available panels	19
4.6	Calendar application	22
4.6.1	Calendar submenu	22
4.6.2	Date setting	22
4.6.3	Time setting	24
4.6.4	Alarm setting	25
4.7	Image viewer application	27
4.8	Audio application	27
4.8.1	Audio submenu	27
4.8.2	Wave player	27
4.8.3	Wave recorder	29
4.9	Thermometer	31
4.10	Low-power mode application	31
4.10.1	Low-power mode submenu	31
4.10.2	Stop mode	32
4.10.3	Sleep mode	34
4.10.4	Standby mode	34
4.11	File browser application	36
4.12	About application	37
5	Software architecture	38
5.1	Demonstration	38
5.2	UCPD demonstration	39
5.3	Kernel API overview	40

5.3.1	k_demo	40
5.3.2	k_menu	41
5.3.3	k_module	42
5.3.4	k_storage	42
5.3.5	k_window	43
5.3.6	k_widgets	43
5.3.7	k_tools	43
5.4	FatFS API overview	44
5.5	USBPD API overview	44
5.6	BSP API overview	44
6	Demonstration memory footprint	45
7	Advanced module description	49
7.1	Module detailed description	49
7.1.1	Module control	49
7.1.2	Module menu description and graphical interface	49
7.1.3	Functionality	51
7.2	Adding a new module	51
8	Acronyms	52
9	References	53
	Revision history	54
	Contents	55
	List of tables	58
	List of figures	59

List of tables

Table 1.	STM32G474QET6 peripherals used by the main demonstration	8
Table 2.	STM32G474QET6 peripherals used by the UCPD demonstration	9
Table 3.	STM32G474QET6 demonstration interrupt usage	10
Table 4.	Internal memory configuration	10
Table 5.	k_demo API	41
Table 6.	k_menu API	41
Table 7.	k_module API	42
Table 8.	k_storage API	42
Table 9.	k_window API	43
Table 10.	k_widgets API	43
Table 11.	k_demo API	43
Table 12.	Demonstration memory footprint	45
Table 13.	K_ModuleItem_Typedef structure description	49
Table 14.	tMenu structure description	49
Table 15.	tMenuItem structure description	50
Table 16.	Acronyms	52
Table 17.	References	53
Table 18.	Document revision history	54

List of figures

Figure 1.	STM32CubeG4 firmware components	2
Figure 2.	STM32G474E-EVAL front view (MB1397)	3
Figure 3.	microSD™ card directory organization	4
Figure 4.	Demonstration folder organization	5
Figure 5.	Demonstration module folder organization	6
Figure 6.	STM32G474E-EVAL demonstration firmware architecture	7
Figure 7.	STM32G474QET6 peripherals used by the main demonstration	8
Figure 8.	STM32G474QET6 peripherals used by the UCPD demonstration	9
Figure 9.	Welcome screen	11
Figure 10.	microSD™ card detection error	11
Figure 11.	microSD™ card content error	12
Figure 12.	Main menu	12
Figure 13.	White LED submenu	13
Figure 14.	White LED automatic dimming	14
Figure 15.	White LED manual dimming	14
Figure 16.	White LED flashing	15
Figure 17.	Math application submenu	15
Figure 18.	CORDIC sine and cosine (polling mode)	16
Figure 19.	CORDIC sine and cosine (DMA mode)	16
Figure 20.	CORDIC sine and cosine (polling mode)	16
Figure 21.	FMAC FIR filtering	17
Figure 22.	FMAC IIR filtering	17
Figure 23.	USB-PD firmware revision compatibility	18
Figure 24.	USB-PD emergency state screen	18
Figure 25.	USB-PD demonstration screen	19
Figure 26.	USB-PD - Available commands	20
Figure 27.	USB-PD - Source / sink capability	20
Figure 28.	USB-PD - Power profile selection	21
Figure 29.	USB-PB - Extended capabilities	21
Figure 30.	Calendar submenu	22
Figure 31.	Date submenu	23
Figure 32.	Setting the year	23
Figure 33.	Setting the month	23
Figure 34.	Setting the day	24
Figure 35.	Consulting the date	24
Figure 36.	Time submenu	24
Figure 37.	Time setting	25
Figure 38.	Consulting the time	25
Figure 39.	Alarm submenu	26
Figure 40.	Alarm setting	26
Figure 41.	Alarm disable	27
Figure 42.	Audio submenu	27
Figure 43.	Audio playlist	28
Figure 44.	Audio playing	28
Figure 45.	Audio paused	28
Figure 46.	Audio stopped	29
Figure 47.	Wave player data path	29
Figure 48.	Audio recorder start	30
Figure 49.	Wave recording	30
Figure 50.	Recording stopped	30
Figure 51.	Wave recorder data path	31

Figure 52.	Temperature screen	31
Figure 53.	Low-power submenu	32
Figure 54.	Stop mode submenu	32
Figure 55.	Start Stop mode (EXTI)	32
Figure 56.	Stop mode activated (EXTI)	33
Figure 57.	Exit from Stop mode (EXTI)	33
Figure 58.	Start Stop mode (alarm).	33
Figure 59.	Exit from Stop mode (alarm).	34
Figure 60.	Sleep mode submenu	34
Figure 61.	Standby mode	35
Figure 62.	Standby mode entry (wake-up pin)	35
Figure 63.	Standby mode exit (wake-up pin)	35
Figure 64.	Standby mode entry (alarm).	36
Figure 65.	Standby mode exit (alarm)	36
Figure 66.	File browser	37
Figure 67.	About submenu	37
Figure 68.	Main demonstration software architecture block diagram	38
Figure 69.	Simplified application startup sequence diagram	39
Figure 70.	UCPD demonstration software architecture block diagram	40
Figure 71.	Application processing sequence diagram	40
Figure 72.	kMenu sequence diagram	42
Figure 73.	Module menu architecture example	51

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2019 STMicroelectronics – All rights reserved