

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:

- STM32CubeMX, a graphical software configuration tool that allows the automatic generation of C initialization code using graphical wizards
- STM32Cube MCU Packages, comprehensive embedded-software platforms specific to each microcontroller 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, and USB Power Delivery
 - All embedded software utilities with full sets of peripheral and applicative examples

The STM32CubeG0 demonstration platform running on the STM32G081B-EVAL evaluation board is built around the STM32Cube hardware abstraction layer (HAL) and low-layer (LL) APIs, and board support package (BSP) components, FatFS and USB PD middleware components. This demonstration embeds several applications showing several features supported by the STM32G081 device and using some of the peripherals present on the STM32G081B-EVAL evaluation board and associated daughterboard. These applications are:

- Low-power application
- Calendar application
- Image Viewer application
- Audio application
- Thermometer application
- Files browser application
- UCPD application



Contents

1	STM32CubeG0 main features	8
2	Getting started with the demonstration	10
2.1	Hardware requirements	10
2.2	Hardware settings of the STM32G081B-EVAL	10
2.2.1	Legacy daughterboard	12
2.2.2	USB-C Power Delivery (UCPD) daughterboard	12
2.3	microSD status	13
2.4	Programming demonstration firmware	14
2.4.1	Using binary files	14
2.4.2	Using preconfigured projects	14
3	Demonstration firmware package	15
3.1	Demonstration repository	15
3.1.1	Demonstration loader folder organization	16
3.1.2	Legacy folder organization	16
3.1.3	UCPD folder organization	18
3.2	Demonstration architecture overview	18
3.2.1	DemoLoader	19
3.2.2	Legacy	19
3.2.3	UCPD	20
3.2.4	HAL level	20
3.2.5	Kernel	20
3.2.6	Middleware	20
3.3	STM32G081RBT6 resources	20
3.3.1	Peripherals	20
3.3.2	Interrupts	24
3.3.3	Internal memory size	24
3.3.4	External memory organization	25
4	Running the demonstration	26
4.1	Demonstration startup	26
4.1.1	Normal processing	26
4.1.2	Error cases	26

4.2	Legacy demonstration	28
4.2.1	Overview	28
4.2.2	Main menu	28
4.2.3	Navigation	29
4.2.4	Calendar application	29
4.2.5	Image viewer application	34
4.2.6	Audio application	34
4.2.7	Thermometer & LDR application	39
4.2.8	Low-power mode application	40
4.2.9	Files browser application	44
4.2.10	Help application	44
4.2.11	About application	45
4.3	UCPD demonstration	45
4.3.1	Warning	45
4.3.2	Hardware checks	45
4.3.3	Emergency state	46
4.3.4	Overall presentation	46
4.3.5	Navigation in the menus	47
4.3.6	Available panels	47
4.3.7	Display Port demonstration	51
5	Software architecture	55
5.1	Demonstration loader	55
5.2	Legacy demonstration	56
5.3	UCPD demonstration	57
5.4	Kernel API overview	59
5.4.1	k_demo	59
5.4.2	k_menu	59
5.4.3	k_module	60
5.4.4	k_storage	60
5.4.5	k_window	61
5.4.6	k_widgets	61
5.4.7	k_tools	61
5.5	FatFS API overview	62
5.6	USBPD API overview	63
5.7	BSP API overview	64

5.7.1	BSP EVAL	64
5.7.2	BSP TSENSOR	65
5.7.3	BSP SD	65
5.7.4	BSP LCD	65
5.7.5	BSP PWR	66
5.7.6	BSP MUX	67
6	Memory footprint	69
6.1	Demonstration loader memory footprint	69
6.2	Legacy demonstration memory footprint	70
6.3	UCPD demonstration memory footprint	73
7	Acronyms	76
8	Appendix	77
8.1	Module detail description	77
8.1.1	Module control	77
8.1.2	Module menu description and Graphical interface	77
8.1.3	Functionality	79
8.2	Adding a new module	80
	References	81
	Revision history	82

List of tables

Table 1.	STM32G081B-EVAL jumpers default settings	11
Table 2.	USB-C daughterboard jumpers default settings	13
Table 3.	STM32G081RBT6 peripherals used by the demonstration loader	21
Table 4.	STM32G081RBT6 peripherals used by the legacy demonstration	22
Table 5.	STM32G081RBT6 peripherals used by the UCPD demonstration	23
Table 6.	STM32G081RBT6 demonstration interrupts usage	24
Table 7.	k_demo API	59
Table 8.	k_menu API	59
Table 9.	k_module API	60
Table 10.	k_storage API	61
Table 11.	k_window API	61
Table 12.	k_widgets API	61
Table 13.	k_tools API	61
Table 14.	FatFS API	62
Table 15.	USBPD API	63
Table 16.	BSP EVAL API	64
Table 17.	BSP TSENSOR API	65
Table 18.	BSP SD API	65
Table 19.	BSP LCD API	65
Table 20.	BSP PWR API	66
Table 21.	BSP MUXAPI	67
Table 22.	Demonstration loader memory footprint	69
Table 23.	Legacy demonstration memory footprint	70
Table 24.	UCPD demonstration memory footprint	73
Table 25.	Table of acronyms	76
Table 26.	K_ModuleItem_Typedef structure description	77
Table 27.	tMenu structure description	78
Table 28.	tMenuItem structure description	78
Table 29.	Document revision history	82

List of figures

Figure 1.	STM32CubeG0 firmware components	8
Figure 2.	SMT32G0 Evaluation board (MB1350) with legacy daughterboard (MB1351)	11
Figure 3.	Legacy daughterboard (MB1351)	12
Figure 4.	USB PD daughterboard (MB1352)	13
Figure 5.	SD Card directory organization	13
Figure 6.	Demonstration folder organization	15
Figure 7.	Demonstration loader folder organization	16
Figure 8.	DemoLegacy folder organization	17
Figure 9.	DemoUCPD folder organization	18
Figure 10.	STM32G081B-EVAL demonstration firmware - Software architecture	19
Figure 11.	Color code for block diagrams	20
Figure 12.	STM32G081RBT6 peripherals used by the demonstration loader	21
Figure 13.	STM32G081RBT6 peripherals used by the legacy demonstration	21
Figure 14.	STM32G081RBT6 peripherals used by the UCPD demonstration	23
Figure 15.	STM32G081RBT6 internal flash memory organization	25
Figure 16.	Welcome screen	26
Figure 17.	Loading...	26
Figure 18.	SD card detection error	27
Figure 19.	Binary not found	27
Figure 20.	Failed to load	27
Figure 21.	Legacy demonstration welcome screen	28
Figure 22.	Main menu	28
Figure 23.	Demonstration menu structure	29
Figure 24.	Calendar sub-menu	30
Figure 25.	Date sub-menu	30
Figure 26.	Date set	31
Figure 27.	Date show	31
Figure 28.	Time sub-menu	31
Figure 29.	Time set	32
Figure 30.	Time show	32
Figure 31.	Time sub-menu	33
Figure 32.	Alarm set	33
Figure 33.	Alarm disable	34
Figure 34.	Audio sub-menu	34
Figure 35.	Playlist	35
Figure 36.	Playing	35
Figure 37.	Paused	36
Figure 38.	Stopped	36
Figure 39.	Wave player data path	37
Figure 40.	Wave recorder start	37
Figure 41.	Wave recording	38
Figure 42.	Wave recorder stopped	38
Figure 43.	Wave recorder data path	39
Figure 44.	Temperature/Light	39
Figure 45.	Temperature/Light (missing legacy daughterboard)	40
Figure 46.	Low power menu	40
Figure 47.	STOP mode menu	41
Figure 48.	Exit from STOP mode - EXTI (WFI)	42

Figure 49.	Exit from STOP mode - RTC Alarm (WFI)	42
Figure 50.	STANDBY mode menu	43
Figure 51.	Exit from STANDBY - Wakeup	43
Figure 52.	Exit from STANDBY - RTC Alarm	44
Figure 53.	File browser	44
Figure 54.	Help	45
Figure 55.	About	45
Figure 56.	USB PD FW revision compatibility	46
Figure 57.	USB PD Emergency state screen	46
Figure 58.	USB PD demonstration main screen	47
Figure 59.	USB PD - available commands	48
Figure 60.	USB PD - Src/Snk capabilities	49
Figure 61.	USB PD - Power profile selection	50
Figure 62.	USB PD - Power profile selection	50
Figure 63.	Type-C to Display Port	51
Figure 64.	USB PD - Hot Plug Detect	52
Figure 65.	Display Port to Type-C	52
Figure 66.	USB PD - Vendor Define Message	53
Figure 67.	Display Port to Display Port (through Type-C)	53
Figure 68.	USB PD - DP contract established	54
Figure 69.	Loader software architecture block diagram	55
Figure 70.	Legacy demonstration software architecture block diagram	56
Figure 71.	Application startup sequence diagram	57
Figure 72.	UCPD demonstration software architecture block diagram	58
Figure 73.	Application processing	58
Figure 74.	KMenu sequence diagram	60
Figure 75.	Module menu architecture example	79

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 STMCube initiative, this set of components is highly portable, not only within STM32G0 Series but also to other STM32 Series.

STM32CubeG0 is fully compatible with STM32CubeMX code generator that allows generating initialization code. The package includes low-layer (LL) and hardware abstraction layer (HAL) APIs that cover the microcontroller hardware, together with an extensive set of examples running on STMicroelectronics boards. The HAL and LL APIs are available in open-source BSD license for user convenience.

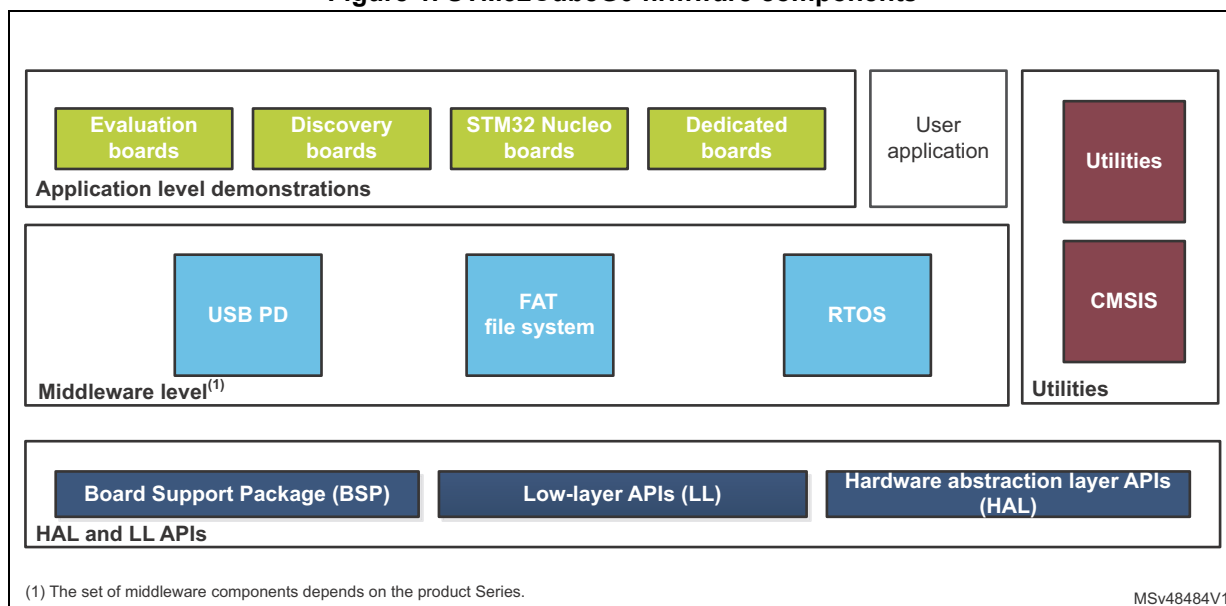
STM32CubeG0 MCU Package also contains a set of middleware components with the corresponding examples. They come in free user-friendly license terms:

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

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

The block diagram of STM32CubeG0 is shown in [Figure 1](#).

Figure 1. STM32CubeG0 firmware components



The STM32Cube G0 Eval Demonstration comes on top of the STM32CubeG0 MCU Package based on a modules architecture allowing re-using software components separately in standalone applications. All these modules are managed by the STM32Cube Demonstration kernel allowing to dynamically add new modules and access to common resources (storage, graphical and components).

The STM32Cube G0 Eval Demonstration is built on the light kernel and on services provided by BSP, components, based on the STM32Cube HAL, using almost the whole STM32 capability to offer a large scope of usages.

The STM32G0 microcontrollers are based on the Arm^{®(a)} 32-bit Cortex[®]-M0+ processor.

The logo for Arm, consisting of the word "arm" in a bold, lowercase, sans-serif font.

a. 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:

- STM32G081B-EVAL evaluation board (MB1350)
- One microSD™ card
- One USB cable to power up the STM32G081B-EVAL board from the ST-LINK USB
- One of the two STM32G081B-EVAL daughterboards:
 - Legacy daughterboard (MB1351)

or

- UCPD daughterboard (MB1352)
 - Type-C cable
 - 19 V wall charger

The STM32G081B-EVAL(MB1350) evaluation board is connected to the USB cable to the host PC and does not need any external power supply.

Only the UCPD daughterboard(MB1352) requires external power supply.

The demonstration displays icons which are stored on a microSD card. The microSD card must contain several files (.bmp, *.txt, *.bin) available under the firmware package directory: /Projects/STM32G081B-EVAL/Demonstrations/Binary/SD_card folder.

2.2 Hardware settings of the STM32G081B-EVAL

The STM32Cube demonstration supports the STM32G081RB device and runs on the STM32G081B-EVAL board from STMicroelectronics.

Figure 2. SMT32G0 Evaluation board (MB1350) with legacy daughterboard (MB1351)

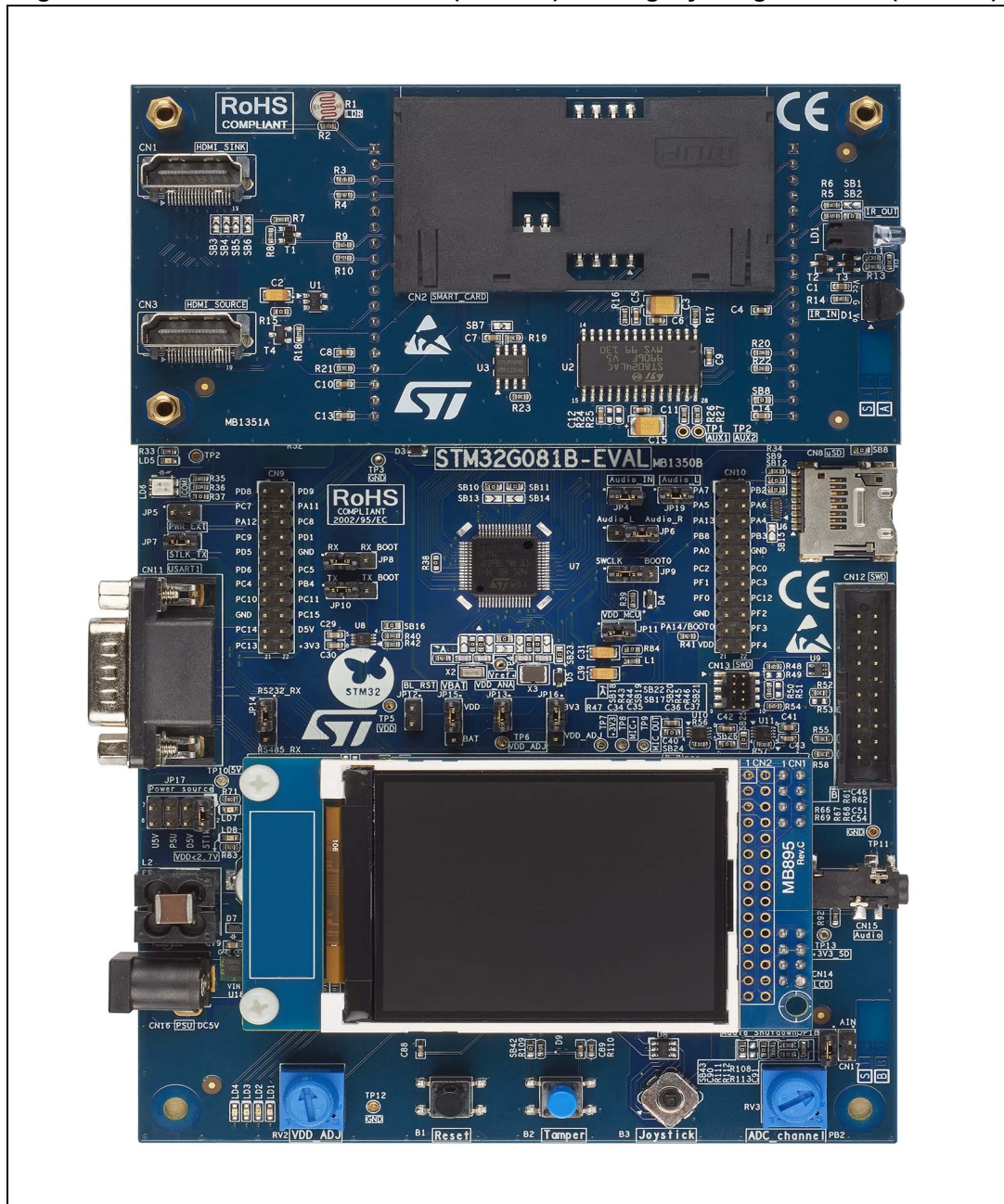


Table 1 indicates the default settings of the STM32081B-EVAL board jumpers.

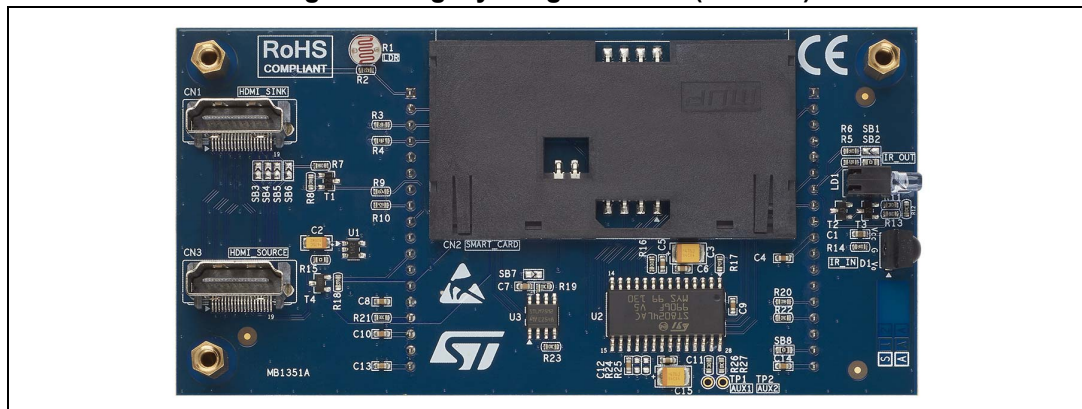
Table 1. STM32G081B-EVAL jumpers default settings

Jumper	Position description
JP1	Motor control signal selection
JP2	Motor control signal selection
JP3	Motor control signal selection
JP4	PA6 is connected to Audio_IN

Table 1. STM32G081B-EVAL jumpers default settings (**continued**)

Jumper	Position description
JP5	D5V from USB-C daughterboard is generated from external 19V or VBUS on Port1
JP6	Stereo playback is enabled
JP7	PC11 is connected to VCP_USART_3_RX
JP8	PC5 is connected as RX signal without bootloader being supported
JP9	PA14-BOOT0 is used as SWCLK
JP10	PC4 is connected as TX signal without bootloader being supported
JP11	VDD_MCU I _{dd} measurement enabled
JP12	Bootloader reset disabled
JP13	VDD_ANA connected to VDD
JP14	RS232_RX is connected to RS232 transceiver and RS232 communication is enabled
JP15	STM32G081R Vbat pin of is connected to VDD
JP16	STM32G081R VDD pin is connected to 3.3V
JP17	Power supply from the ST-link (CN6)
JP18	Speaker amplifier U17 is enabled
JP19	PA4 is connected to VIN1 of Audio amplifier

2.2.1 Legacy daughterboard

Figure 3. Legacy daughterboard (MB1351)

2.2.2 USB-C Power Delivery (UCPD) daughterboard

To be able to run the USBPD demo, you need to plug the 19V power supply into the CN3 socket.

Caution: There are two ways of generating the power from the DCDC converter:

1. with GPIO
2. PWM (Pulse width modulation)

In our demo, the PWM mode has been selected to give the possibility to select precise voltages and demonstrate PPS (Programmable power supply) capabilities. Therefore you need to check the solder bridges configuration at the back of the daughterboard. SB13, SB14, and SB15 must be open. And SB2, SB3, SB23 and SB26 must be closed.

Figure 4. USB PD daughterboard (MB1352)

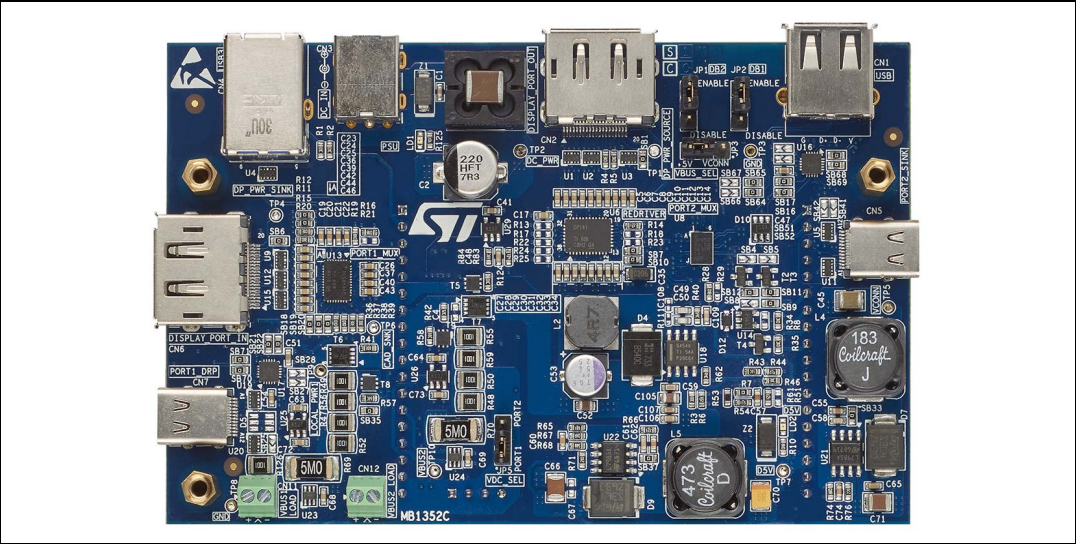


Table 2 indicates the default settings of the USB-C daughterboard jumpers.

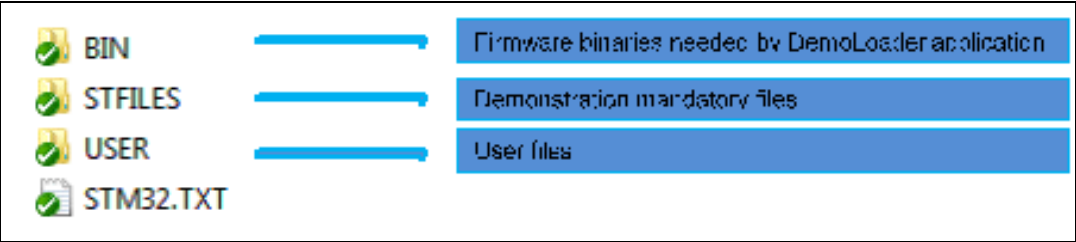
Table 2. USB-C daughterboard jumpers default settings

Jumper	Position description
JP1	Dead battery function is enabled
JP2	Dead battery function is enabled
JP3	VBUS is connected to VCONN
JP5	D5V from USB-C daughterboard is generated from external 19V or VBUS on Port1

2.3 microSD status

The STM32G081B-EVAL board comes with a microSD card memory preprogrammed with FW binaries, image resources, text files and directories tree used by 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 5. SD Card directory organization



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

2.4 Programming demonstration firmware

The user programs demonstration with two methods:

2.4.1 Using binary files

To program Demonstration Loader binary image into the internal Flash memory, the user must load the STM32CubeG0_Demo_Loader_STM32G081B_EVAL.hex file located under Project\STM32G081B-EVAL\Binary, using ST Link Utility programming tool for instance.

Depending on mounted daughterboard, UCPD or legacy demonstration runs.

2.4.2 Using preconfigured projects

Select first folder corresponding to desired demonstration: DemoLegacy, DemoLoader or DemoUCPD. Then inspect folder corresponding to the user's preferred toolchain (MDK-ARM, EWARM or SW4STM32).

- Open corresponding project. Warning: to use execute legacy or UCPD demonstration in standalone mode, i.e. without using Demonstration Loader, the user must select 'Standalone' workspace.
- Rebuild all sources.
- Load the project image through the debugger.
- Restart the evaluation board (press B1: reset button).

3 Demonstration firmware package

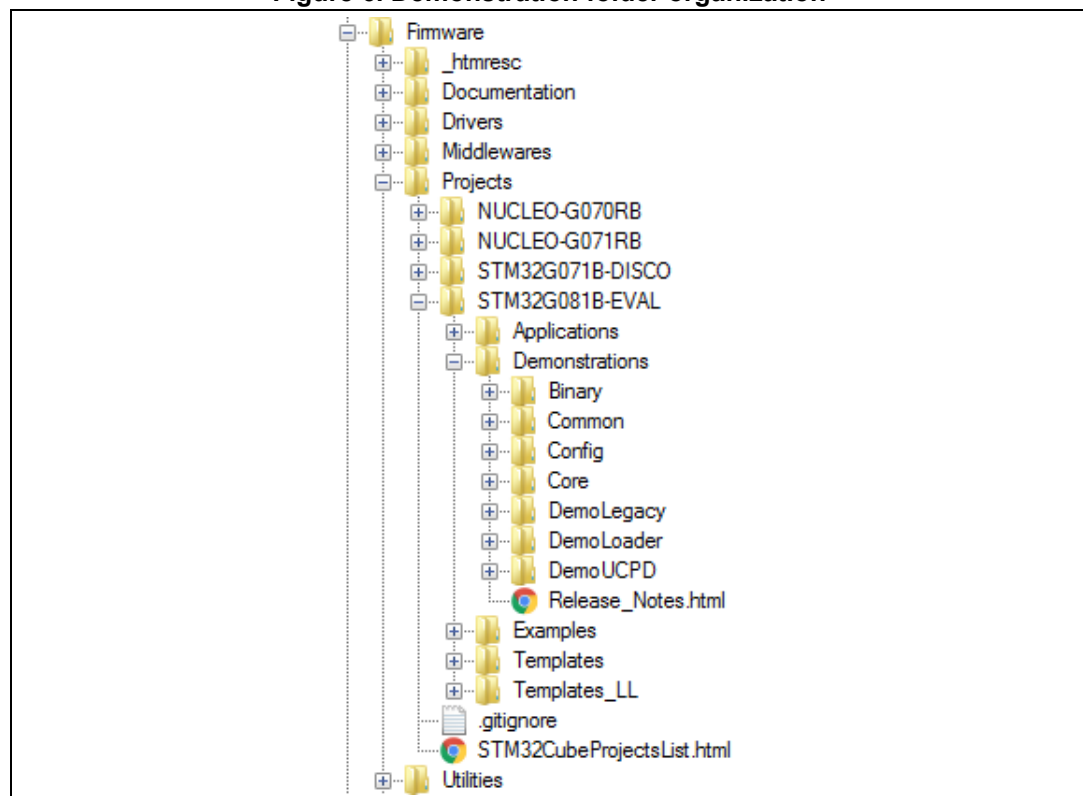
The demonstration has been designed with following objectives:

- Toolkit with low memory consumption
- Modular applications: independents with high level of reuse
- Basic menu navigation through joystick
- Comprehensive G0 functional coverage

3.1 Demonstration repository

Figure 6 shows the demonstration folder organization:

Figure 6. Demonstration folder organization



The demonstration sources are located in the 'Projects' folder of the STM32Cube package for each supported board, here in the STM32G081B-EVAL folder.

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

- Binary: demonstration binary file in Hex format of demonstration loader using Legacy.bin & Ucpd.bin.

Note: *Note: Both binaries Legacy.bin & Ucpd.bin are specifically linked to be used with demonstration loader. They are not usable in standalone mode.*

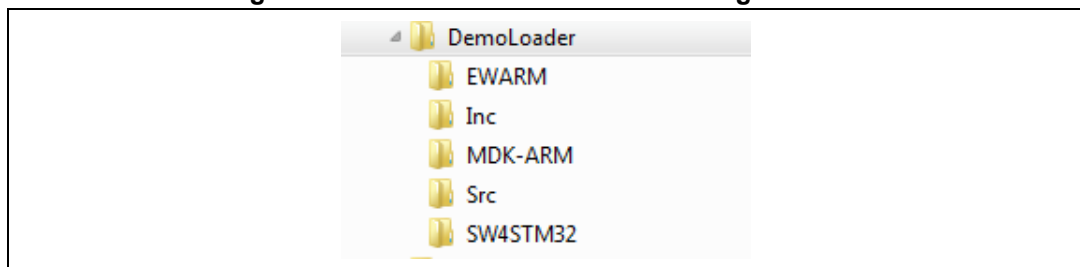
- Common: C source files implementing common services used by demonstration firmware sub-parts (loader, legacy and UCPD).
- Config: FatFS, HAL and demonstration kernel configuration files.
- Core: demonstration kernel implementation files.
- DemoLegacy: demonstration firmware legacy sub-part implementation.
- DemoLoader: demonstration firmware loader sub-part implementation.
- DemoUCPD: demonstration firmware USB Type-C™ power delivery (UCPD) sub-part implementation.
- SD_card: resource files (bitmaps, binaries...) required by the demonstration firmware to be copied on the SD card.

Each demonstration firmware sub-part related folder organization is detailed in sections below.

3.1.1 Demonstration loader folder organization

The role of the demonstration loader is to load the legacy or the UCPD binary according to the daughterboard connected to the extension connector of the mother board. [Figure 7](#) illustrates the demonstration loader folder organization.

Figure 7. Demonstration loader folder organization



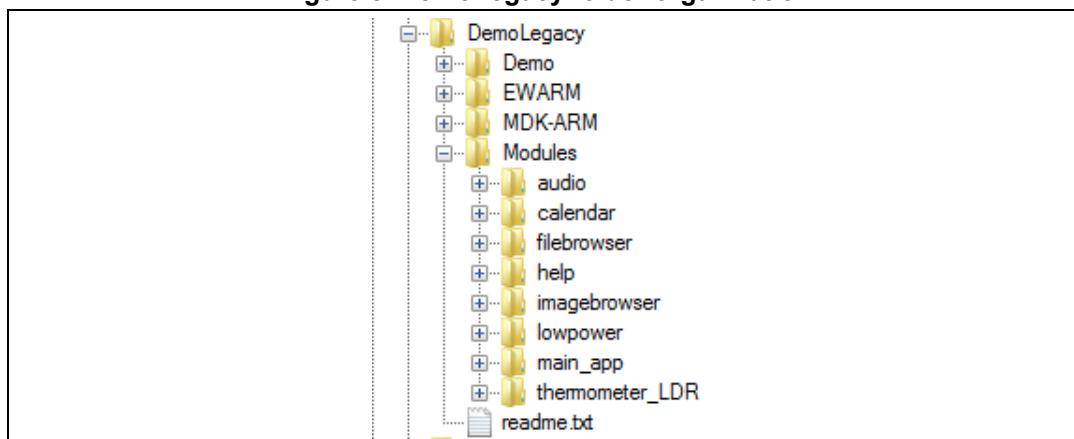
DemoLoader sub-folders:

- Inc: demonstration loader header files
- Src: demonstration loader implementation
- Software development environments:
 - EWARM: IAR™ embedded workbench,
 - MDK ARM: Keil® Microcontroller Development Kit
 - SW4STM32: System workbench for STM32

3.1.2 Legacy folder organization

The legacy part of the demonstration firmware provides a sub-set of the applications supported by the STM32072B-EVAL demonstration firmware; (STM32 G0 and STM32 F0 share the same feature footprint). [Figure 8](#) illustrates the organization of the legacy folder.

Figure 8. DemoLegacy folder organization



DemoLegacy sub-folders:

- Demo: demonstration loader header files
- Modules: legacy applications implementation (one folder per application)
 - main_app: main menu management (see [Section 4.2.2](#))
 - Calendar: date, time and alarm setting (see [Section 4.2.4](#))
 - Image viewer: bitmap images slide show (see [Section 4.2.5](#))
 - Audio: audio record and playback (see [Section 4.2.6](#))
 - Thermometer & LDR: temperature and daylight intensity measurement (see [Section 4.2.7](#))
 - Low power: low-power modes (see [Section 4.2.8](#))
 - File browser: navigation through a folder tree (see [Section 4.2.9](#))
 - Help: mother board jumpers description (see [Section 4.2.10](#))
 - About: mother board jumpers description (see [Section 4.2.11](#))
- Software development environments:
 - EWARM: IAR embedded workbench
 - MDK ARM: Keil Microcontroller Development Kit
 - SW4STM32: System workbench for STM32

Note: *Each software development environment provides two configurations:*

1. Standalone: once built, the legacy demonstration software is flashed directly on the board and executed right after. In that mode user has to make sure that the legacy daughterboard (MB1351) or no daughterboard is connected to the extension connector of the mother board (MB1350). No binary file is provided for standalone mode. It has to be rebuild based on the user's preferred IDE.
2. Relocate: once built, the legacy demonstration software executable must be copied in a specific folder of the SD card. It is copied from SD card to the flash memory by the demonstration loader if the legacy daughterboard (MB1351) or no daughterboard is connected to the extension connector of the mother board (MB1350).

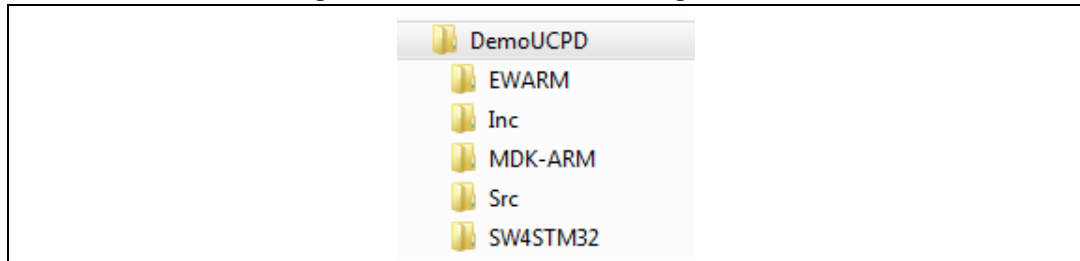
The demonstration loader automatically launches the execution of the legacy demonstration software after the flash programming has completed.

3.1.3 UCPD folder organization

The UCPD part of the demonstration firmware aims at demonstrating how USB-PD version PD3.0 has been implemented in the context of STM32G0xx devices.

Figure 9 Illustrates the organization of the UCPD folder.

Figure 9. DemoUCPD folder organization



DemoUCPD sub-folders:

- Inc: UCPD application header files
- Src: UCPD application implementation
- Software development environments:
 - EWARM: IAR embedded workbench,
 - MDK ARM: Keil Microcontroller Development Kit
 - SW4STM32: System workbench for STM32

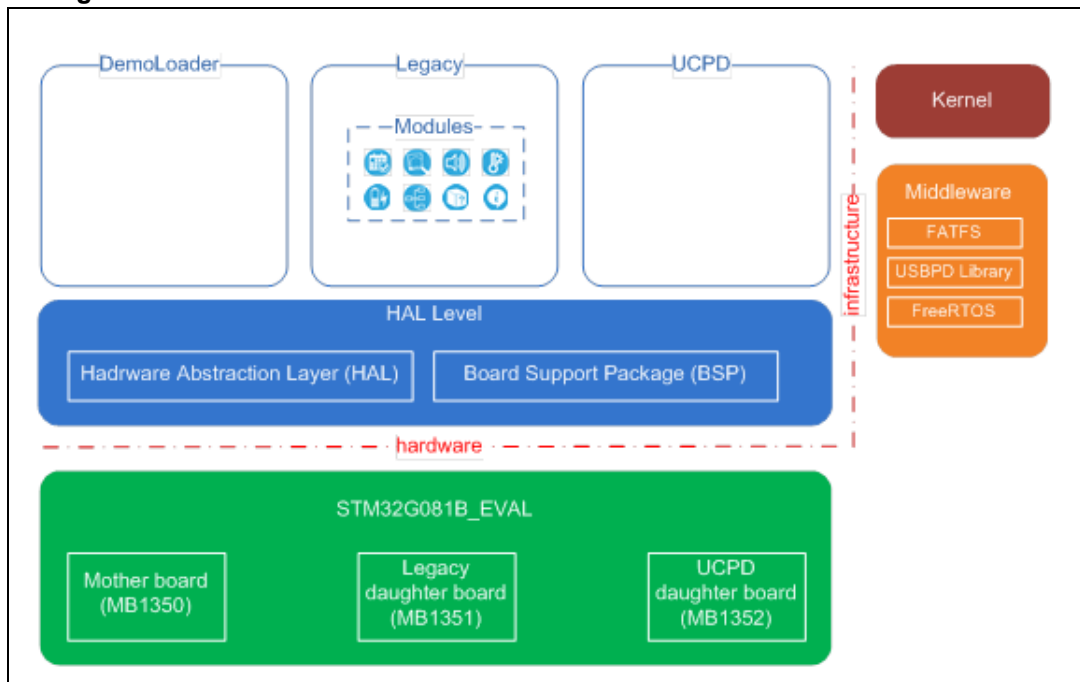
Note: Each software development environment provides two configurations:

1. Standalone: once built, the UCPD demonstration software is flashed directly on the board and executed right after. In that mode user has to make sure that the UCPD daughterboard (MB1352) is connected to the extension connector of the mother board (MB1350). No binary file is provided for standalone mode. It has to be rebuild based on the user's preferred IDE.
2. Relocate: once built, the UCPD demonstration software executable must be copied in a specific folder of the SD card. It is copied from SD card to the flash memory by the demonstration loader if the UCPD daughterboard (MB1352) is connected to the extension connector of the mother board (MB1350). The demonstration loader automatically launches the execution of the UCPD demonstration software after the flash programming has completed.

3.2 Demonstration architecture overview

The top level software architecture of the STM32G081B-EVAL demonstration firmware is represented on *Figure 10*. The software elements mentioned in this diagram are briefly depicted in dedicated sections.

Figure 10. STM32G081B-EVAL demonstration firmware - Software architecture



3.2.1 DemoLoader

The role of the demonstration loader is to load in flash the part of the demonstration software corresponding to the hardware configuration detected after reset. Three cases have to be considered:

1. No daughterboard connected to the extension connector of the mother board. In that case the legacy demonstration image is loaded in flash but only a sub-set of the legacy applications is available (applications requiring the hardware resources of the legacy daughterboard is disabled).
2. The legacy daughterboard is connected to the extension connector of the mother board. In that case the legacy demonstration image is loaded in flash and all the legacy applications are available.
3. The UCPD daughterboard is connected to the extension connector of the mother board. In that case the UCPD demonstration image is loaded in flash.

Note: The binary images of both the legacy and the UCPD applications are copied into a specific folder of the SD card.

3.2.2 Legacy

The legacy application runs when the mother board operates in standalone mode (no daughterboard connected to the extension connector) or when the legacy daughterboard is connected to the mother board. It contains many applications that are easily reusable, such as, RTC calendar, FAT file system implementation on SD Card, wave player using DAC and DMA peripherals, voice recorder using ADC and DMA peripherals, low-power modes, temperature sensor interfacing and TFT LCD.

3.2.3 UCPD

The UCPD application runs when the UCPD daughterboard is connected to the mother board. It manages both USB-PD ports of the UCPD daughterboard which mainly consists in Type-C connection/disconnection detection and Type-C power contract negotiation. UCPD demonstration is also responsible for Type-C pins reconfiguration when a Type-C port is configured in DisplayPort (DP) alternate mode(1) or when the UCPD daughterboard is used as USB Type-C to USB-3.0 adapter.

(1) DP over Type-C Alt Mode allows streaming video as well as USB data to transfer simultaneously through a common Type-C connector.

3.2.4 HAL level

HAL level layer consists in the stm32g0xx.HAL drivers together with the STM32G081B-EVAL board support package (BSP).

3.2.5 Kernel

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

3.2.6 Middleware

The middleware provides the following modules:

- FreeRTOS: FreeRTOS open source solution. 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 application to access to the files stored on the SD card.
- USBPD: USB-PD software stack

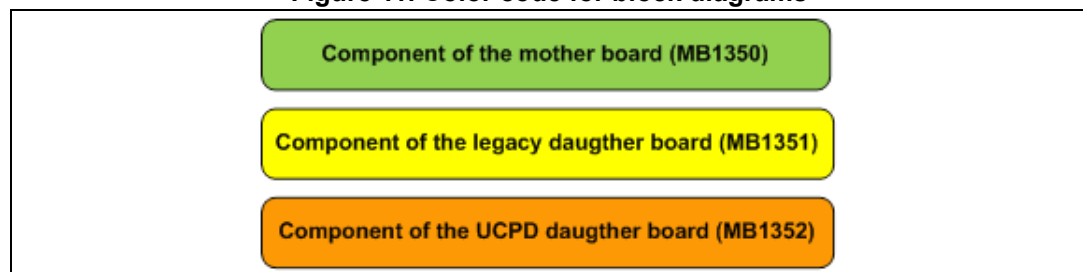
3.3 STM32G081RBT6 resources

3.3.1 Peripherals

The following sections detail what peripherals of the STM32G081RBT6 microcontroller are used by the demonstration loader, the legacy and the UCPD demonstrations.

In the block diagrams [Figure 12](#), [Figure 13](#) and [Figure 14](#), the following notation applies:

Figure 11. Color code for block diagrams



Peripherals used by the demonstration loader

Figure 12. STM32G081RBT6 peripherals used by the demonstration loader

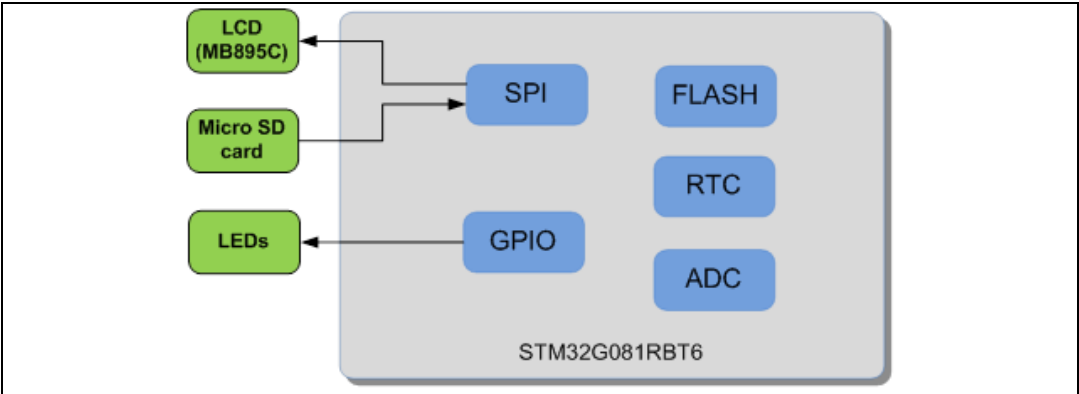


Table 3. STM32G081RBT6 peripherals used by the demonstration loader

Peripheral	Usage description
SPI	Both LCD and micro SD card are controlled through SPI1. Read accesses to the Micro SD card are performed during the demonstration loading process. Write accesses to the LCD are performed to display information messages during the demonstration loader execution.
FLASH	The flash peripheral is used to program the flash memory with the demonstration firmware associated with the detected daughterboard.
GPIO	GPIO are used to toggle the LEDS of the mother board when an error is detected during by the demonstration loader.
ADC	ADC channel 15 is used for the daughterboard detection.
RTC	RTC backup registers are used to store the following context information: DR0: image ID of the demonstration firmware loaded in the flash memory (legacy or UCPD).

Peripherals used by the legacy demonstration

Figure 13. STM32G081RBT6 peripherals used by the legacy demonstration

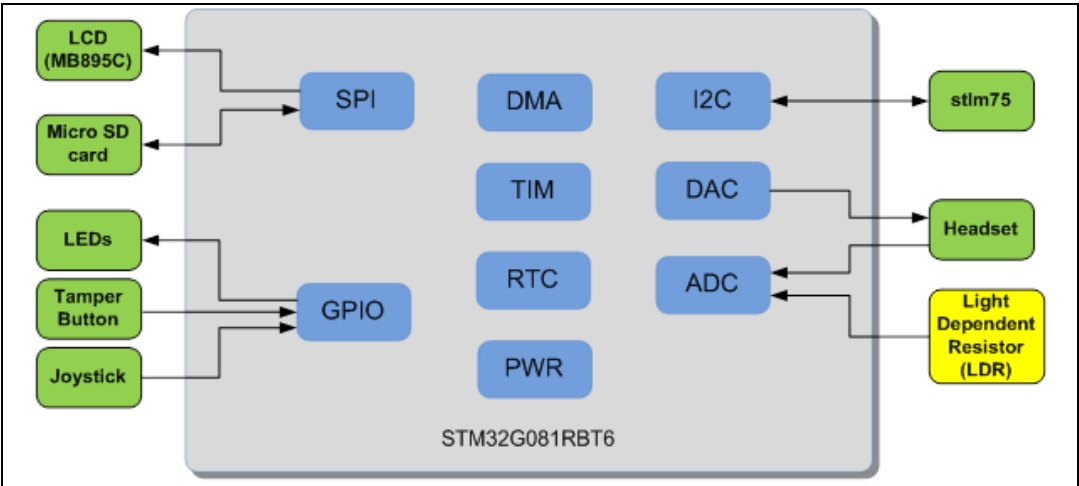


Table 4. STM32G081RBT6 peripherals used by the legacy demonstration

Peripheral	Usage description
SPI	Both LCD and micro SD card are controlled through SPI1. Read accesses to the Micro SD card are performed to retrieve the bitmaps to display on the LCD screen. Micro SD 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 demonstration execution.
GPIO	GPIO are used to toggle the LEDS of the mother board when an error is detected during by the legacy demonstration. Also the GPIO pins connected to the joystick and the tamper button are used to interact with the legacy demonstration (e.g. menu navigation).
ADC	ADC channel 1 is connected to the light dependent resistor (LDR) supported by the legacy daughterboard. It used by the Thermometer/LDR application. ADC channel 6 is connected to the microphone input (external headset connected on 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 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.
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).

Peripherals used by the UCPD demonstration

Figure 14. STM32G081RBT6 peripherals used by the UCPD demonstration

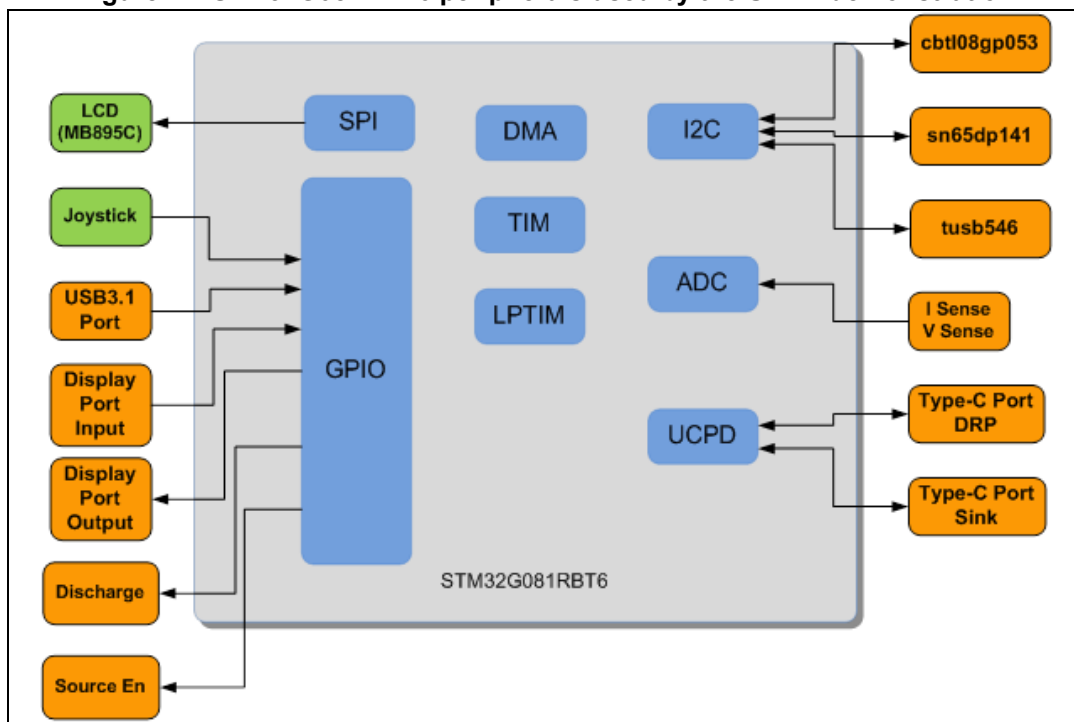


Table 5. STM32G081RBT6 peripherals used by the UCPD demonstration

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	<p>The GPIO pins connected to the joystick are used to interact with the UCPD demonstration (e.g. menu navigation).</p> <p>One GPIO pin is used to detect USB3.1 cable connection/disconnection.</p> <p>One GPIO pin is used to detect the HPD (hot-plug-detect) signal coming from a Display Port input connector.</p> <p>One GPIO pin is used to notify the presence of a Source Device to the Sink device through the HPD pin of the Display Port output connector.</p> <p>One GPIO pin is used to enable the VBUS on the Type-C port 1 (DRP).</p> <p>One GPIO pin is used to control the USB VBUS discharge mechanism on the Type-C port 1 (DRP).</p>
I2C	<p>I2C1 is used to control the following components of the UCPD daughterboard (MS1352):</p> <p>CBTL08GP053: USB Type-C, multiplexer, switch, USB 3.1, DisplayPort</p> <p>SN65DP141: DisplayPort Linear Redriver</p> <p>TUSB546: USB Type-C DP ALT Mode Linear Redriver Crosspoint Switch</p>
ADC	<p>ADC channel 9 is used to measure the voltage level on the VBUS line of the Type-C port 1 (DRP).</p> <p>ADC channel 11 is used to measure the current level on the VBUS line of the Type-C port 1 (DRP).</p> <p>ADC channel 3 is used to measure the voltage level on the VBUS line of the Type-C port 2 (Sink).</p> <p>ADC channel 16 is used to measure the current level on the VBUS line of the Type-C port 2 (Sink).</p>
UCPD	UCPD is used to manage the USB Type-C communication over the Type-C ports.

Table 5. STM32G081RBT6 peripherals used by the UCPD demonstration (continued)

Peripheral	Usage description
DMA	DMA is used for ADC conversions.
TIM	TIM6 is used for USB3.1 detect and DisplayPort hot-plug detect debouncing.
LPTIM	LPTIM1 is used to generate the PWM signal controlling the voltage level on the VBUS line of the Type-C port 1 (when the duty cycle of the PWM signal is 0%, VBUS voltage level is 15V, when the duty cycle is 10%, VBUS voltage level is 5V).

3.3.2 Interrupts

[Table 6](#) shows all the external interrupts used by the demonstration

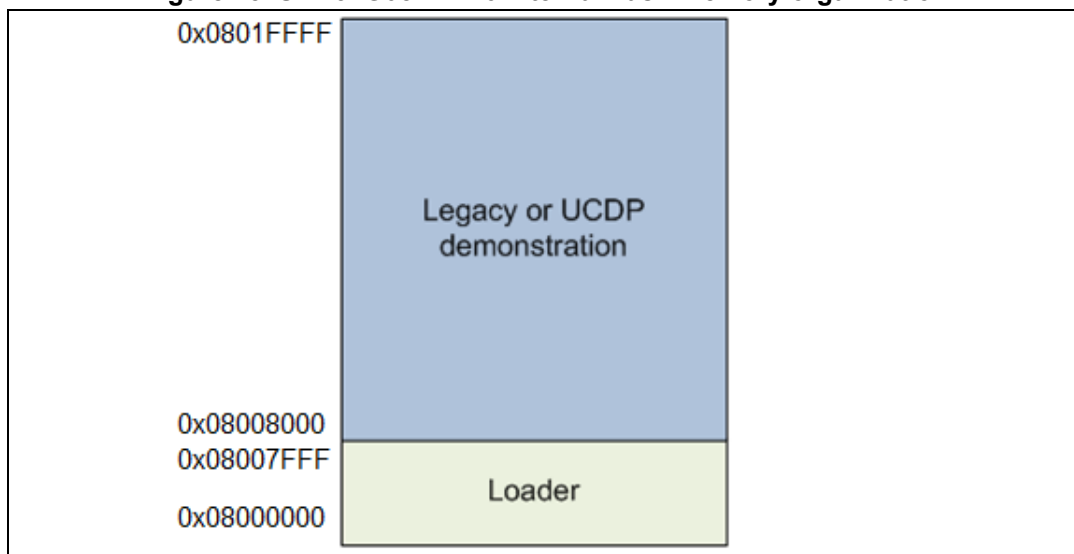
Table 6. STM32G081RBT6 demonstration interrupts usage

Interrupt	Usage description	Loader	Legacy	UCPD
Systick	Delay management	YES	YES	YES
EXTI line 0	Joystick SEL (interrupt mode, rising edge)	NO	YES	YES
EXTI line 2	Joystick UP (interrupt mode, rising edge)	NO	YES	YES
EXTI line 3	Joystick DOWN (interrupt mode, rising edge)	NO	YES	YES
EXTI line 7	Joystick RIGHT (interrupt mode, rising edge)	NO	YES	YES
EXTI line 8	Joystick LEFT (interrupt mode, rising edge)	NO	YES	YES
EXTI line 9	SD Card detect (interrupt mode, rising and falling edge)	YES	YES	NO
EXTI line 13	Tamper (interrupt mode, rising edge)	NO	YES	YES
DMA1 Channel1	DAC/ADC conversions completion	NO	YES	YES
ADC1_COMP	ADC analog watchdogs	NO	NO	YES
UCPD	UCPD related interrupts (e.g. Rx message received, Rx ordered set detected, Transmit message sent, ...)	NO	NO	YES

3.3.3 Internal memory size

[Figure 15](#) represents the image of the STM32G081RBT6 flash memory. The demonstration loader is loaded in the first 16 pages flash memory (32 Kbytes). The demonstration loader loads the legacy or the UCPD demonstration from page 21 onwards. The size of the legacy or the UCPD demonstration must not exceed 96 Kbytes.

Figure 15. STM32G081RBT6 internal flash memory organization



3.3.4 External memory organization

The STM32G081B-EVAL demonstration is based on an embedded free FAT file system, FatFS. The file system is needed by the demonstration loader to retrieve the binary file to load in the flash memory and to read all media information from the on-board microSD card memory. The SD card memory is organized in three subdirectories:

- **BIN:** this directory contains the binary images of the legacy and the 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/her own files here to be played inside the demonstration menus (pictures and wav files). This folder is used only by the File Browser (see [Section 4.2.9](#)), Image Viewer (see [Section 4.2.5](#)) and Wave Player applications (see [Section 4.2.6](#)). This folder also contains the voice recorded wave file 'rec.wav'. (This Folder is created when the Voice Recording application is run).

Note: *The microSD card memory provided within STM32G081B-EVAL board is already programmed with the media files to run the demonstration. These files are also available within the demonstration firmware package the following folder:*
Projects\STM32G081B_EVAL\Demonstrations\Binary\SD_card

4 Running the demonstration

4.1 Demonstration startup

4.1.1 Normal processing

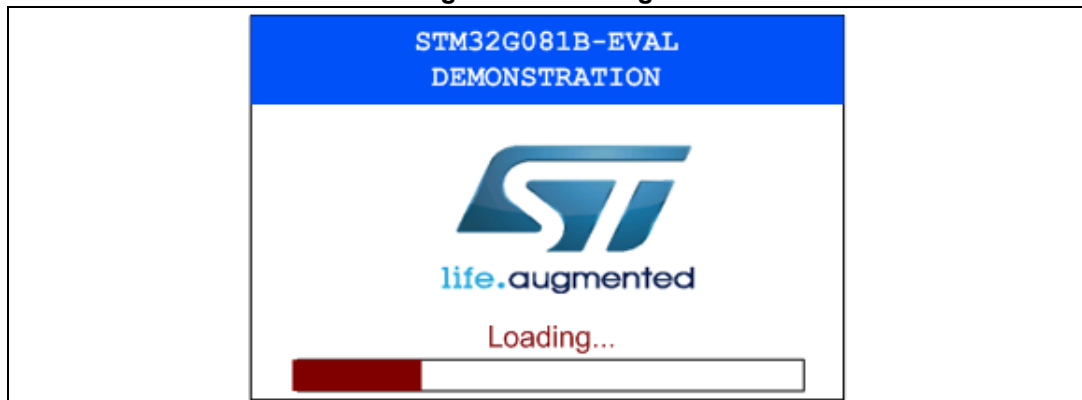
After a board reset, at demonstration startup the welcome screen is displayed and the ST logo appears on the LCD (see [Figure 16](#)).

Figure 16. Welcome screen



Then the demonstration programs the flash memory with the legacy or the UCPD application according to the daughterboard connected to the extension connector of the mother board. Note that if the required software is already present in the flash memory the loading step is skipped and legacy or UCPD application execution starts immediately. During the loading process a progress bar is displayed on the LCD screen (see [Figure 17](#)).

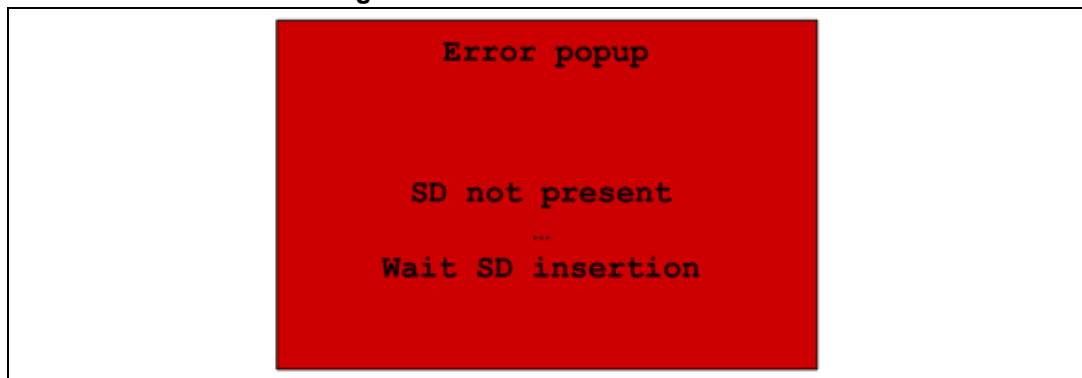
Figure 17. Loading...



4.1.2 Error cases

If no card is detected, the demonstration doesn't start and the message shown in [Figure 18](#) is displayed on the LCD screen. The demonstration waits for the SD card insertion to proceed.

Figure 18. SD card detection error



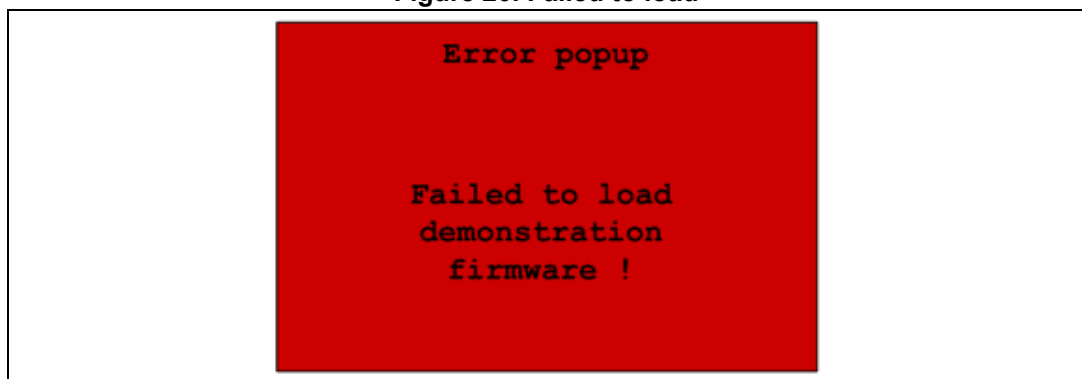
Both the legacy and UCPD binary files must be copied to the BIN folder of the SD card. If binary file to download is missing on the SD card, the message shown in [Figure 19](#) is displayed on the LDC screen.

Figure 19. Binary not found



If the loading process fails the message shown in [Figure 19](#) is displayed on the LCD.

Figure 20. Failed to load



4.2 Legacy demonstration

4.2.1 Overview

The purpose of the legacy demonstration is to bring out the capabilities of microcontroller and Evaluation board peripherals. It runs only on the STM32G081B-EVAL board MB1350 mounted with daughterboard MB1351.

Figure 21. Legacy demonstration welcome screen



The demonstration contains the following applications. (See [Figure 22](#))

4.2.2 Main menu

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

- UP/DOWN/LEFT/RIGHT keys to select a submenu.
- SELECT to enter a submenu.

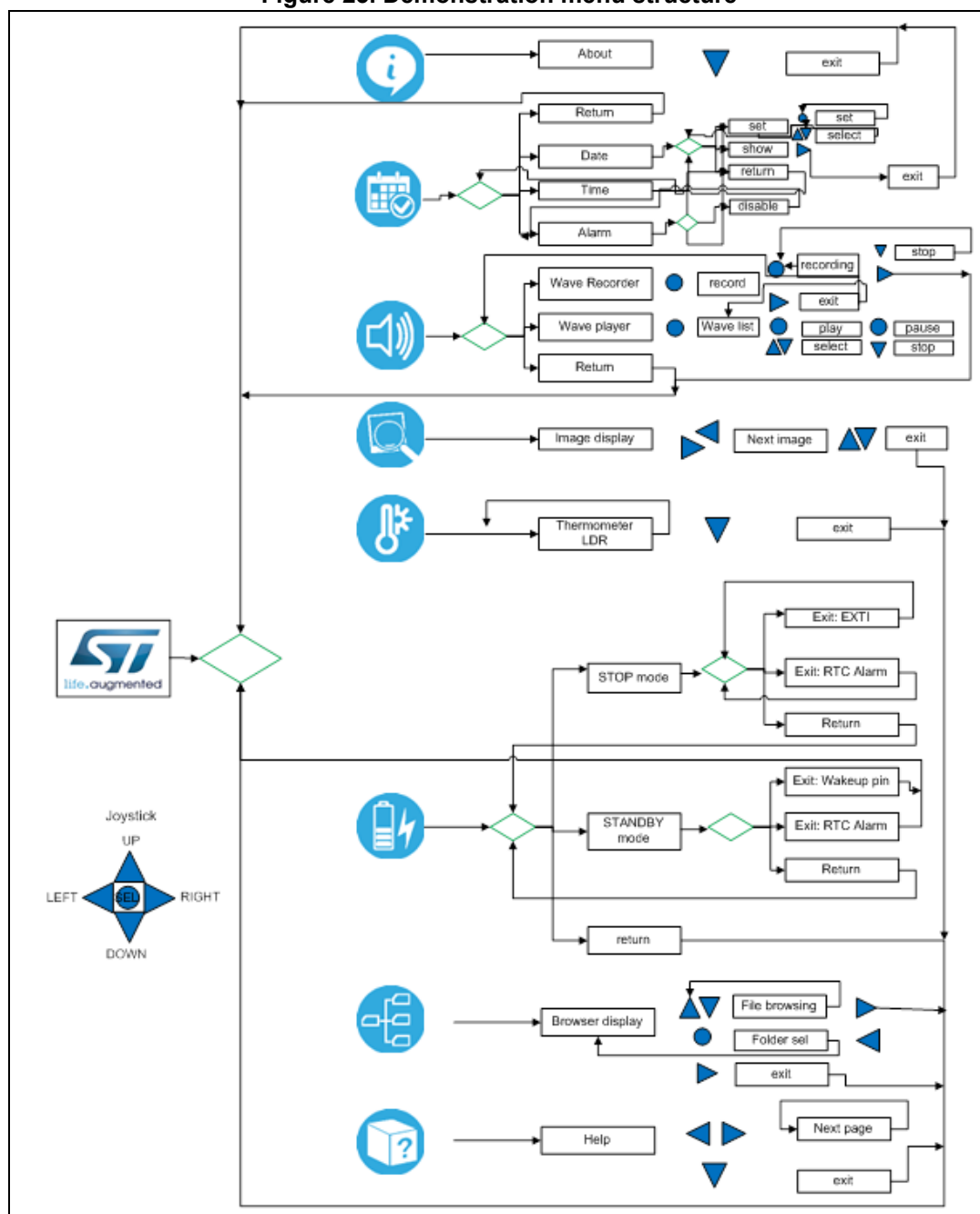
When a submenu is selected, the submenu title (i.e. the name of the attached application) is mentioned at the top of the LCD display (see [Figure 22](#)).

Figure 22. Main menu



4.2.3 Navigation

Figure 23. Demonstration menu structure



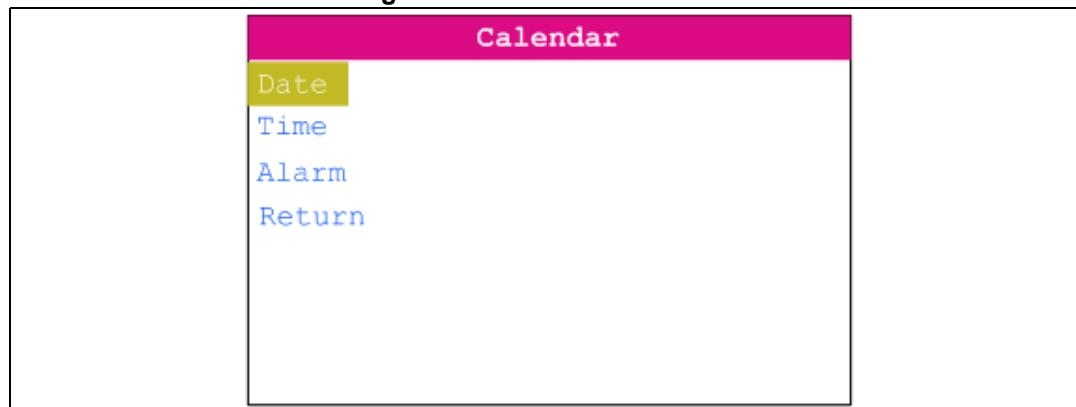
4.2.4 Calendar application

The STM32G081RB 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.

Calendar submenu

Calendar submenu is used to select Date, Time, and Alarm settings or to return to main menu.

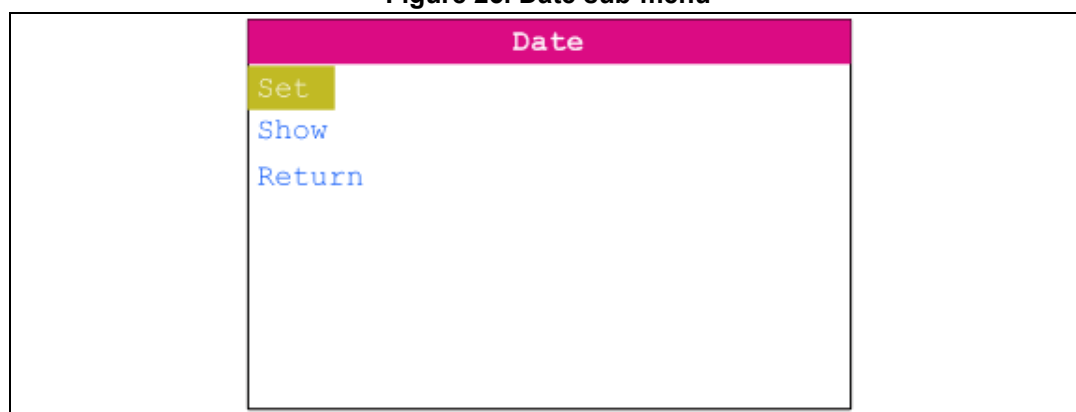
Figure 24. Calendar sub-menu



Date setting

Date setting submenu allows to set or see current date and to get back to Calendar submenu.

Figure 25. Date sub-menu



Date set is done in three steps:

1. First, select current year with UP & DOWN joystick keys. Current year is displayed on right side of upper line. Press SEL joystick key when selected year fits the user's requirement.
2. Second, select month with UP & DOWN joystick keys. Current month is displayed on left side of upper line. Press SEL joystick key when selected month fits the user's requirement.
3. Then, select day number with UP/DOWN/LEFT/RIGHT joystick keys. Current selected date is framed. Press SEL joystick key when selected day fits the user's requirement.

Figure 26. Date set

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						

Current date is watched through this menu.

Figure 27. Date show

Date
Wed. 16.11.2017

Time Setting

Time setting submenu allows to set or see current date and to get back to Calendar submenu.

Figure 28. Time sub-menu

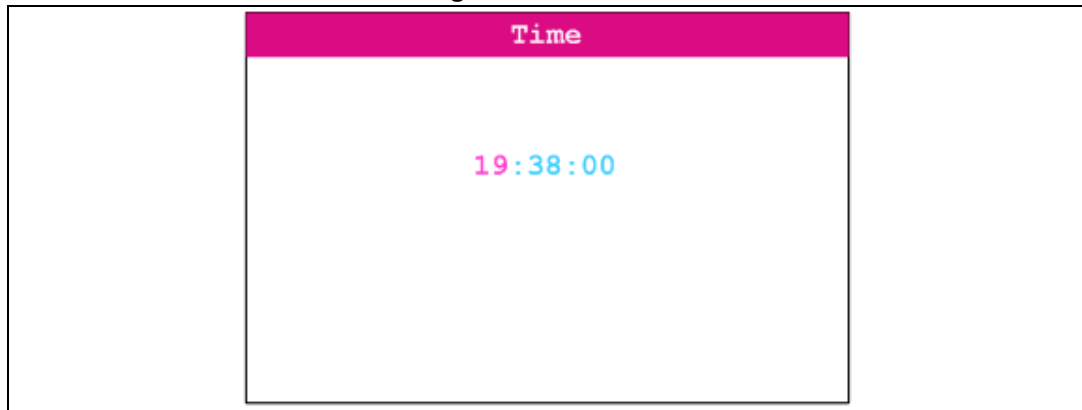
Time
Set
Show
Return

Time is set selecting hours (in 24 hours format), minutes and seconds. Use LEFT or RIGHT joystick keys to switch from one to another. Current selection is highlighted in pink.

Press DOWN or UP joystick keys to increase or decrease selected number. Keep pressing allows to increment or decrement faster.

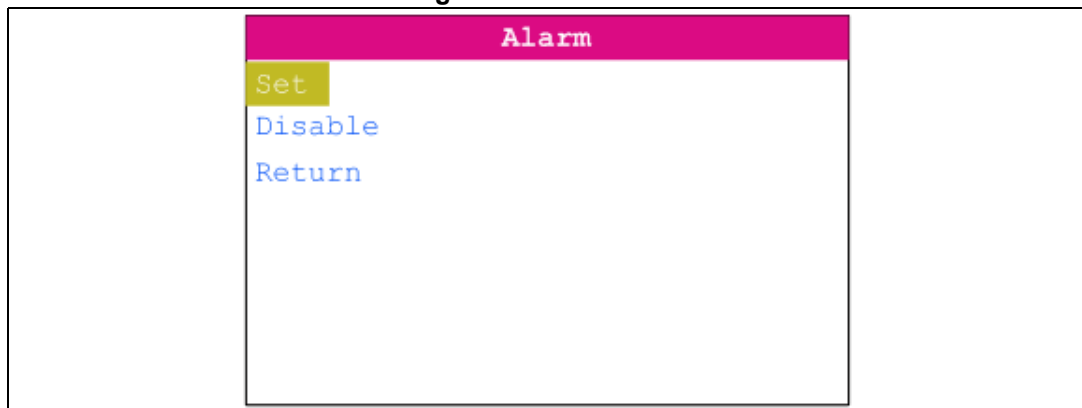
When the user's settings are done, pressing SEL joystick key saves this time as current one.

Figure 29. Time set



Current date is watched through this menu.

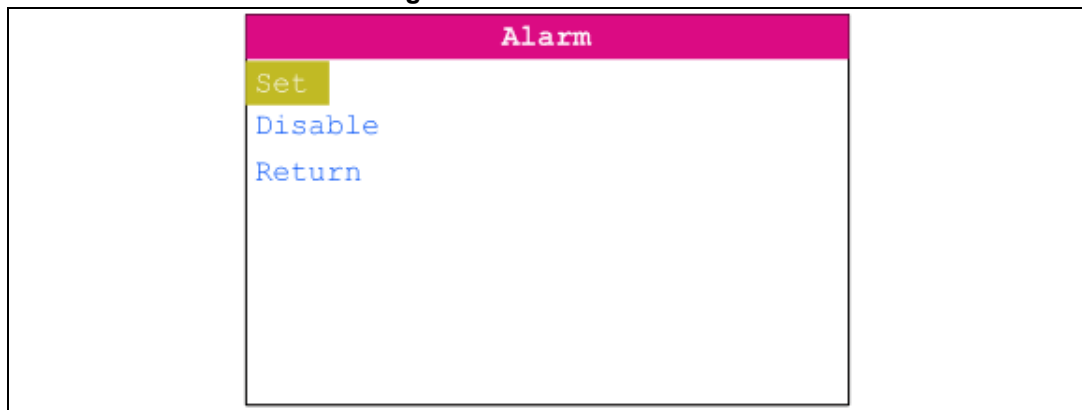
Figure 30. Time show



Alarm setting

Alarm setting submenu allows the user to set or disable Alarm A and to get back to Calendar submenu.

Figure 31. Time sub-menu



Alarm A is set selecting hours (in 24 hours format), minutes and seconds. Use LEFT or RIGHT joystick keys to switch from one to another. Current selection is highlighted in pink.

Press DOWN or UP joystick keys to increase or decrease selected number. Keep pressing allows the user to increment or decrement faster.

When the user's settings are done, pressing SEL joystick key saves this time for Alarm A.

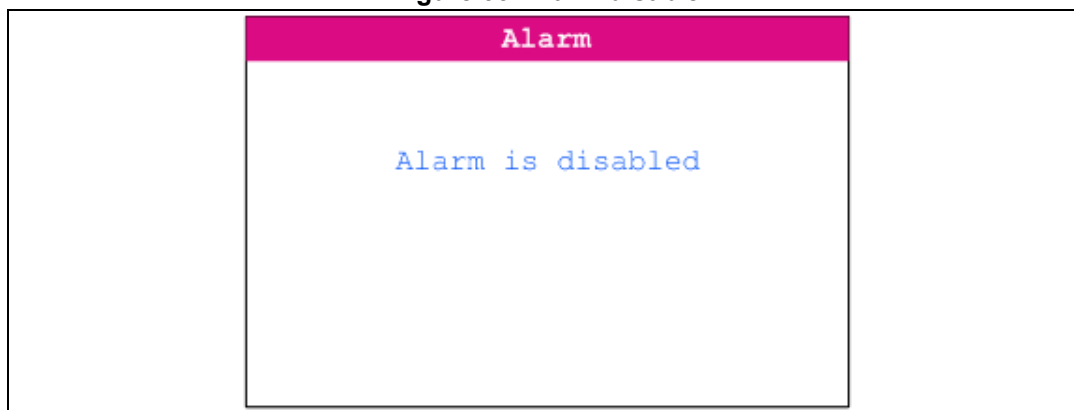
When selected time is reached Alarm A is triggered making all LEDs blinking alternatively few seconds. Alarm is configured to be triggered every day at same hour. If the user wants to disable it, he has to do it manually through 'disable' submenu.

Figure 32. Alarm set



When selecting Disable submenu, Alarm A is disabled.

Figure 33. Alarm disable



4.2.5 Image viewer application

The Image Viewer submenu is used to demonstrate the LCD control performance using the embedded SPI interface. The application acts as a slideshow of the bitmap files stored in the USER folder of the SD card. Only the BMP files having the following properties are considered:

- Bit depth: 16 bits (RGB)
- Size: 240x320

The user presses the RIGHT or LEFT keys of the joystick to display the next or the previous bitmap of the list. He quits this application by pressing the DOWN key of the joystick and then return to the main menu.

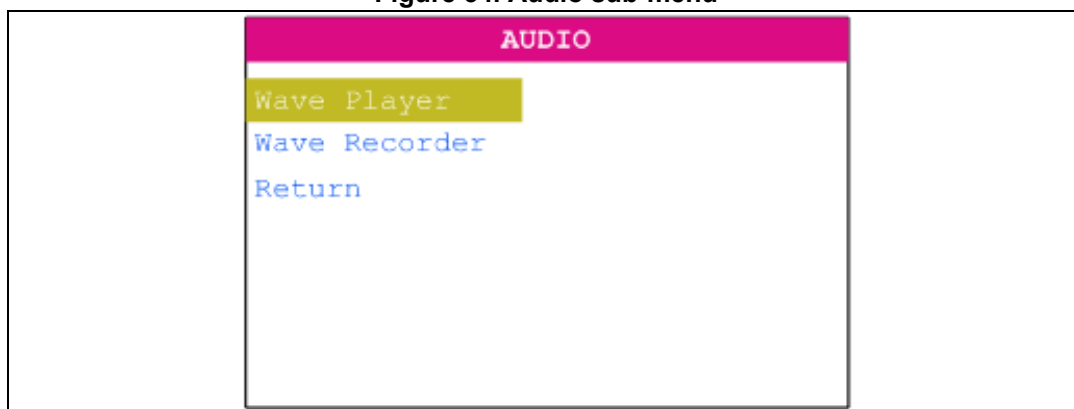
4.2.6 Audio application

The audio application provides means to perform wav playback and voice recording by means of STM32G081RB DAC and ADC peripherals connected to a headphone plugged to the CN15 connector.

Audio submenu

Audio submenu is used to select wave player or wave recorder applications (see [Figure 34](#))

Figure 34. Audio sub-menu



Wave player

When the 'Wave player' menu option of the Audio menu is selected the playlist is displayed on the LCD screen (see [Figure 35](#)). The playlist consists in listing all the wav files found in the USER folder of the Micro SD. Note that the playlist size is limited to 4 titles.

Figure 35. Playlist



Using the UP/DOWN key of the joystick selects the title. Press the SELECT key of the joystick to play the selected title. [Figure 36](#) represents the aspect of the LCD screen when the wave file is being played.

Figure 36. Playing



The user pauses the playback by pressing the SELECT key of the joystick (pressing the SELECT key again then resumes the playback). Press the DOWN key of the joystick to stop the playback (pressing the SELECT key again restarts the playback from the beginning). [Figure 37](#) represents the aspect of the LCD screen when playback is paused.

Figure 37. Paused



Figure 38 represents the aspect of the LCD screen when playback is stopped.

Figure 38. Stopped



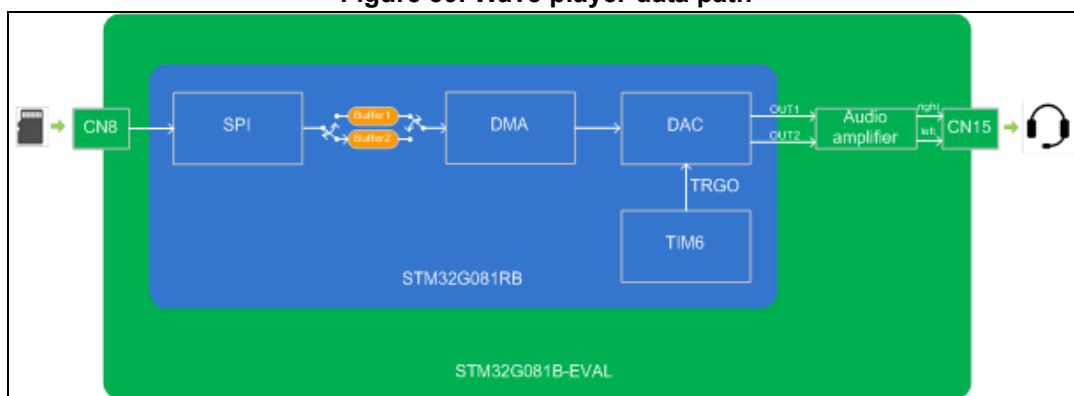
Note: The audio files provided within this package are based on a free music download from www.DanoSongs.com website.

Wave player implementation overview:

The selected wave file is accessed through the FatFS interface. Audio samples are transferred to the internal SRAM by blocks (512 byte each) using the SPI interface. Audio samples go through a flip flop buffer, i.e. one half of the buffer is fed by the SPI while the other one is accessed by the DMA to feed the DAC. 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.

Wave player data path is represented by Figure 39.

Figure 39. Wave player data path



Wave recorder

When the 'Wave Recorder' option of the Audio menu is selected the LCD displays the LCD audio record page (see [Figure 40](#)). When the record is started, the rec.wav file is created under the USER folder of the SD card. Note that if this file already exists, its content is erased and the file is considered as a new empty file.

Recorder audio file has the following properties:

- Sampling rate: 16000 samples/s
- Channels: Stereo (left = right)
- Resolution: 16 bits/sample

Figure 40. Wave recorder start



The user starts the playback by pressing the SELECT key of the joystick. The record duration is limited to 30 seconds. During the recording the progress bar gives an idea of the 'record tim budget' consumption (see [Figure 41](#)).

Figure 41. Wave recording



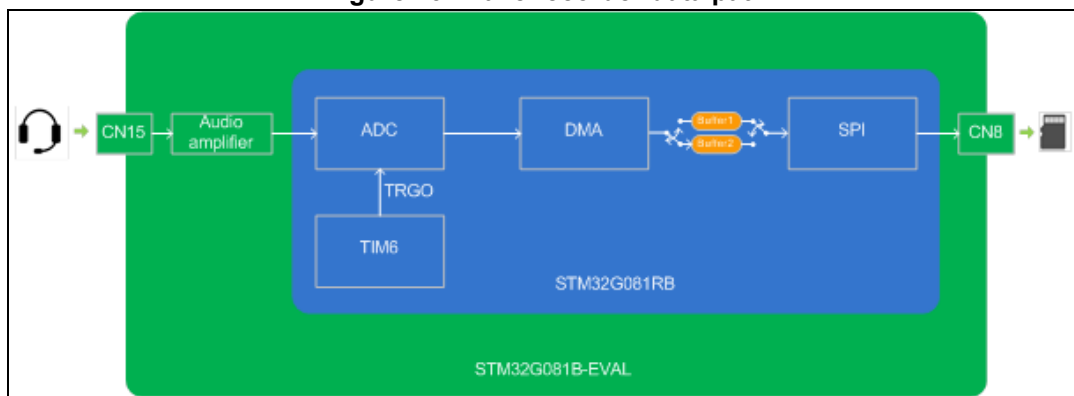
Press the SELECT key of the joystick to stop the recording (see [Figure 42](#)). Pressing the SELECT key again starts a new recording.

Figure 42. Wave recorder stopped

**Wave recorder implementation overview:**

The ADC is connected to the headphone through a microphone amplifier. The ADC converts incoming audio stream into audio samples at a 16 kHz rate. ADC conversion are triggered by TIM6 in order to reach the targeted sample rate. Converted data go through a flip flop buffer, i.e. one half of the buffer is fed by the DMA while the other one is copied to the rev.wav file using the FatFS interface (in fine the SD card is accessed through the SPI). Audio samples are written to the record file by blocks (512 byte each).

Figure 43. Wave recorder data path



4.2.7 Thermometer & LDR application

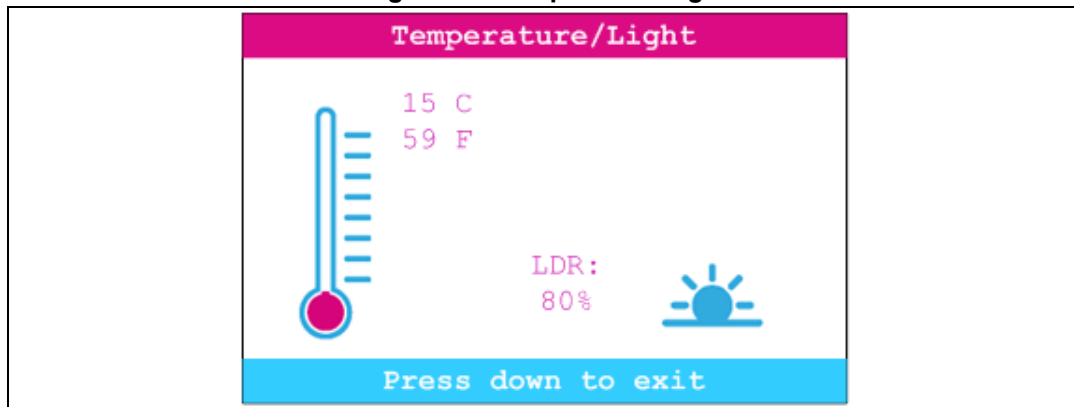
The STM32G081RBT6 microcontroller has two embedded I2C peripherals that may be connected to any device supporting the I2C protocol.

A STLM75 (or a compatible device) I2C temperature sensor is mounted on the MB1351 daughterboard and is used to capture the external temperature (-55°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%).

Once Thermometer & LDR application has been selected by pressing JOYSEL push-button, temperature value is displayed in Celsius and Fahrenheit and Light value (LDR) is displayed in percent as shown in [Figure 44](#).

Figure 44. Temperature/Light



Thermometer & LDR application do require hardware resources of the legacy daughterboard MB1351.

In case of board MB1351 is not mounted on motherboard MB1350, an error is displayed as shown in [Figure 45](#).

Figure 45. Temperature/Light (missing legacy daughterboard)



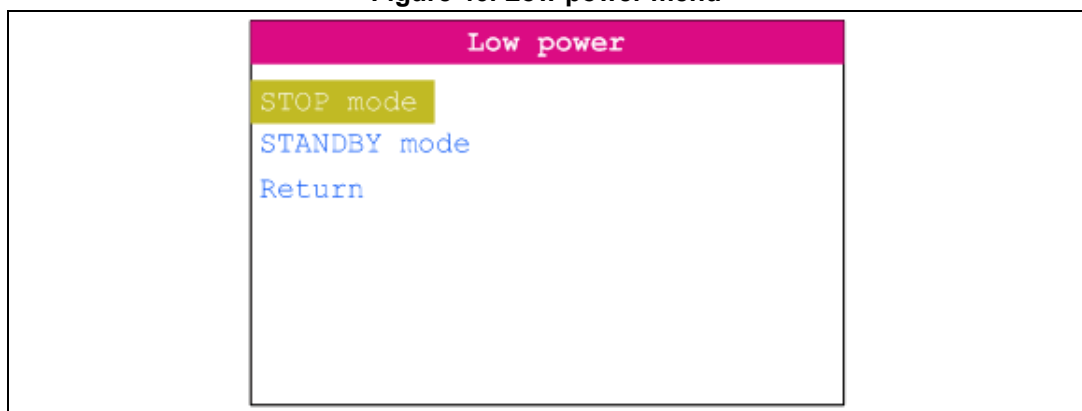
4.2.8 Low-power mode application

The STM32G081 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 taken as examples.

Low-power mode submenu

Low power submenu is used to select low power mode STOP or STANDBY or to return to main menu.

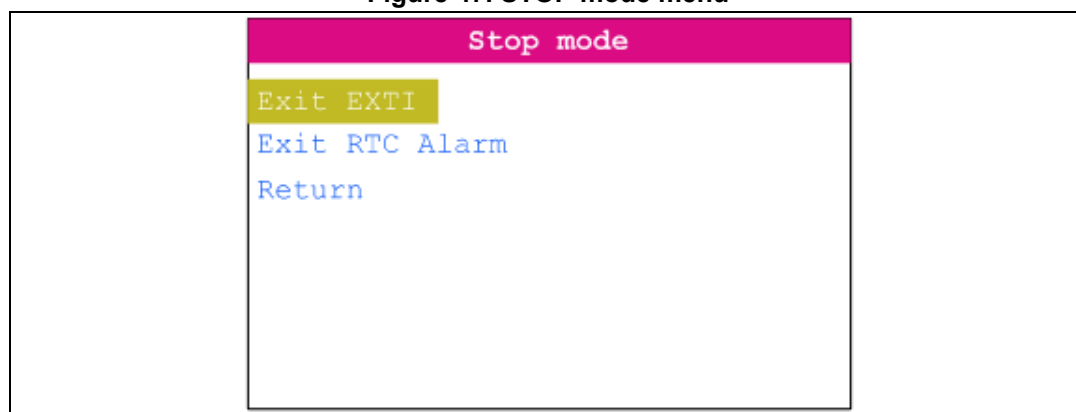
Figure 46. Low power menu



STOP mode

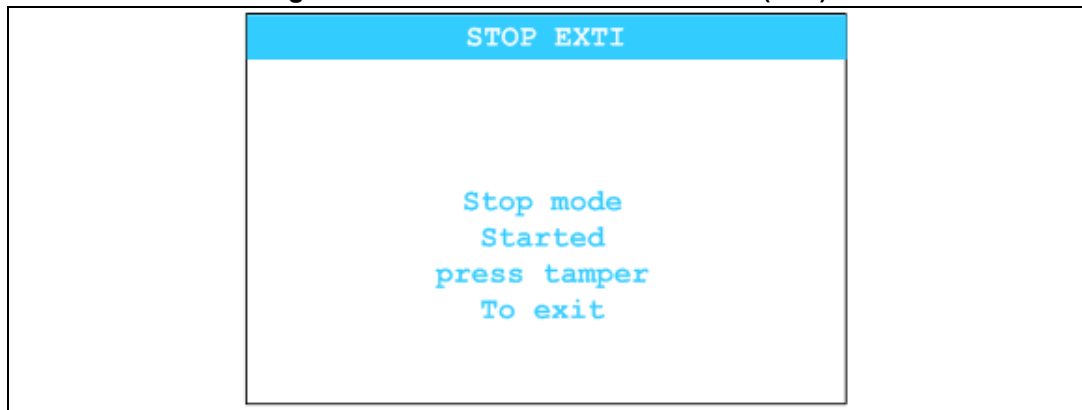
STOP mode sub-menu (see [Figure 47](#)) allows the user to put STM32G081 device in STOP mode with low power regulator ON. It is possible to select between EXTI (WFI) or RTC alarm (WFI) to exit from STOP mode.

Figure 47. STOP mode menu



EXTI (WFI)

Tamper button configured as External Interrupt. Press SEL to enter STOP mode. When the MCU is in Stop mode, the message shown in [Figure 48](#) below is displayed on the LCD. MCU remains in Stop mode until Tamper push-button is pressed. System clock is then set to 64 MHz and the application resumes execution.

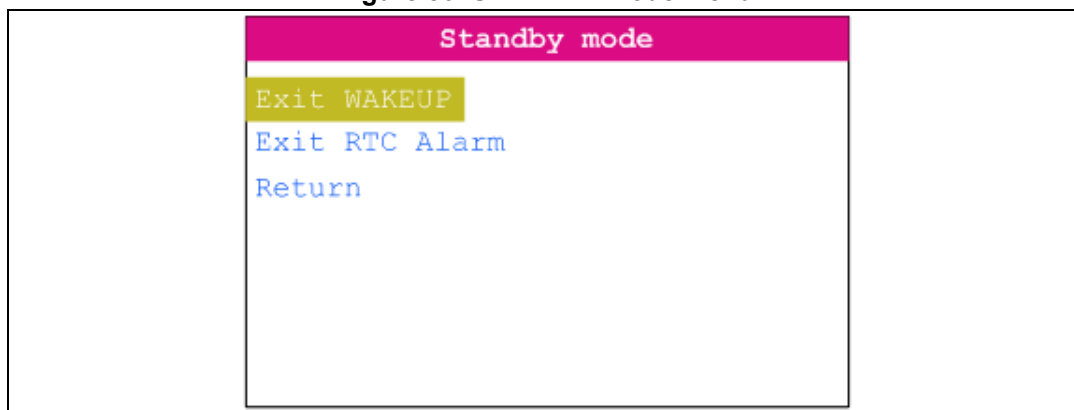
Figure 48. Exit from STOP mode - EXTI (WFI)**RTC Alarm (WFI)**

When selecting this submenu, user has to set alarm delay when MCU has to exit from Stop mode. [Figure 49](#) below shows how to set the wakeup time, using UP, DOWN, LEFT, RIGHT joystick keys. Wakeup time is validated after a press on SEL, 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 64 MHz and application resumes execution.

Figure 49. Exit from STOP mode - RTC Alarm (WFI)**STANDBY mode**

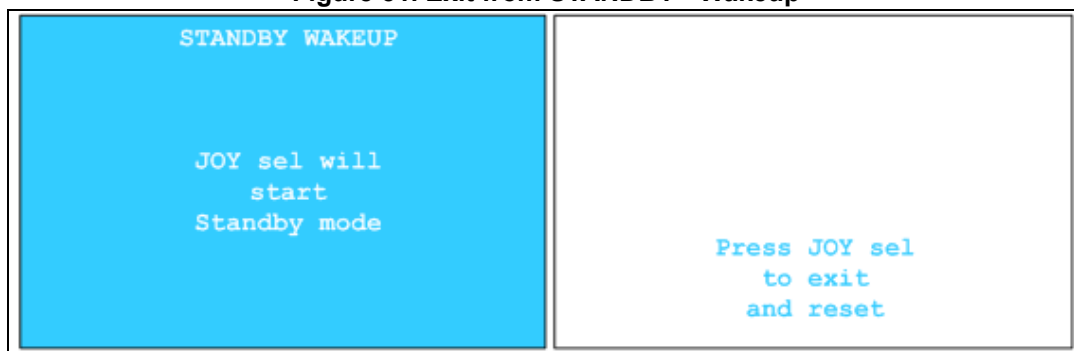
STANDBY mode submenu allows the user to put STM32G081 device in Standby mode. It is possible to select between Wakeup pin or RTC alarm to exit from Standby mode.

Figure 50. STANDBY mode menu

**WAKEUP**

Press again SEL joystick key to enter standby mode. Doing this, button is configured as Wake-up pin in order to wake up the MCU from the Standby mode. Once the EXIT Wakeup submenu has been selected, press the SELECT key of the joystick to allow the system entering Standby mode. When the SELECT key of the joystick is pressed again the MCU exits Standby mode, i.e. a system reset is generated and legacy demonstration execution is restarts from the beginning (main menu).

Figure 51. Exit from STANDBY - Wakeup

**RTC Alarm**

When selecting this submenu, user must set alarm delay when MCU has to exit from Standby mode. RTC Alarm wakes up the MCU from the Standby mode after the programmed time has elapsed. User may set the wakeup time, using UP, DOWN, LEFT, RIGHT joystick keys. Once Wakeup time is validated with a press on SEL, MCU enters then in STANDBY mode. After the programmed timing has elapsed, the system exits the Standby mode and program execution resets.

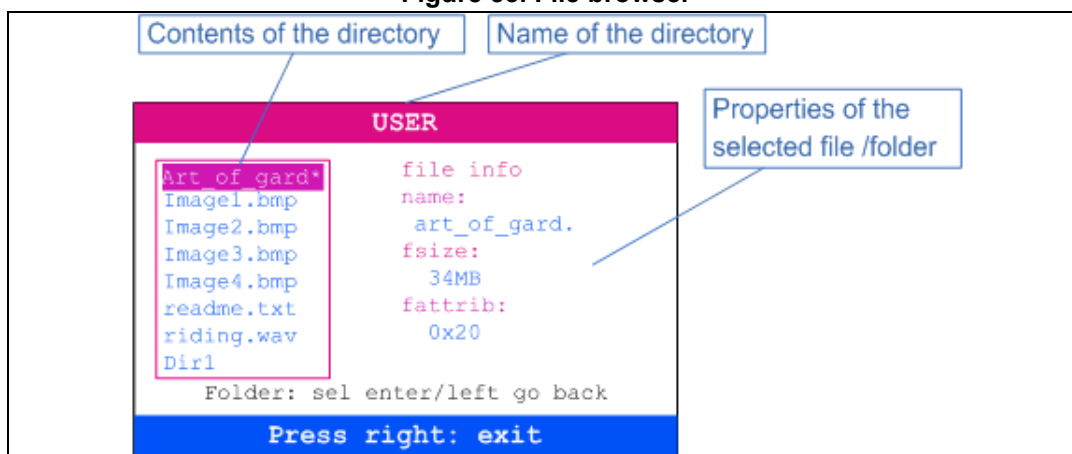
Figure 52. Exit from STANDBY - RTC Alarm



4.2.9 Files browser application

The File browser application is used to demonstrate the possibility to access to the microSD card file system through the FatFS interface. It may be used to navigate the USER folder of the microSD card and to display its contents and subfolders (see [Figure 53](#)).

Figure 53. File browser



Using the UP/DOWN key of the joystick selects the title. Press the SELECT key of the joystick to enter a sub-directory. Press the LEFT key of the joystick to leave 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 Byte, KByte or MByte units)
- file attributes

4.2.10 Help application

Help submenu showing different information regarding motherboard MB1350. Use Right and Left button to see next/previous slide. [Figure 54](#) shows first slide of Help submenu.

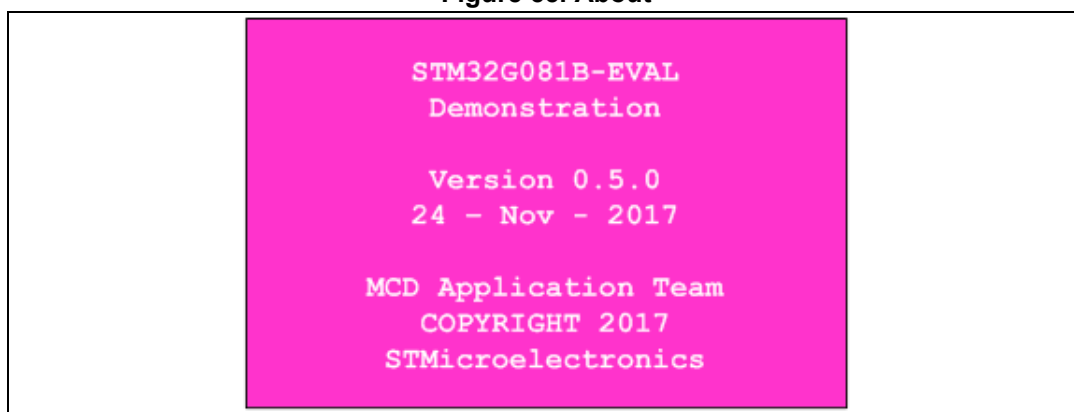
Figure 54. Help



4.2.11 About application

About submenu shows version & date of the STM32G081B-EVAL demonstration firmware. When 'About menu' is selected, the message shown in [Figure 55](#) is displayed on the LCD screen. Press DOWN key to return to the main menu.

Figure 55. About



4.3 UCPD demonstration

In the following USB-PD presentations, the port 1 (P1) and port 2 (P2) are connected through a USB Type-C cable

4.3.1 Warning

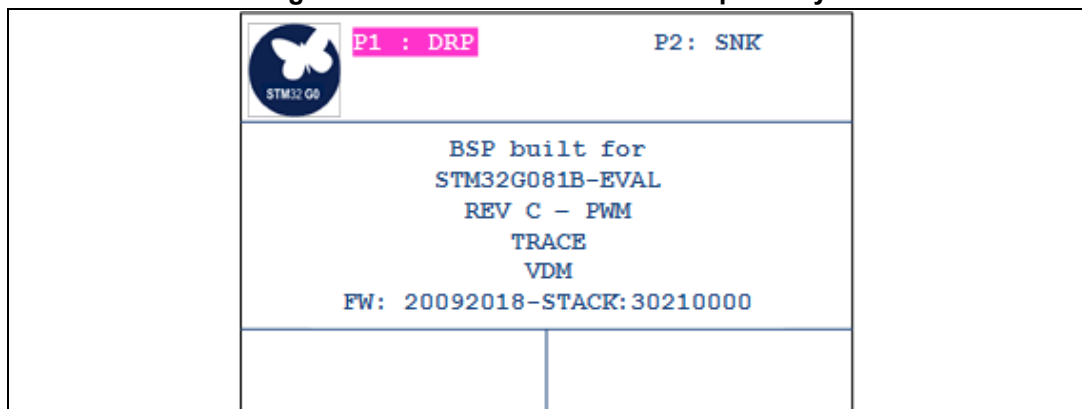
Be aware that using this demo, the delivered power may damage the device that the user plugs onto the G0 Evaluation board. ST may not be held responsible for all issues encountered.

4.3.2 Hardware checks

Before running USB PD demo, please insure HW configuration is correct. Please refer to [Section 2.2.2](#).

The user must check that the correct firmware is flashed:

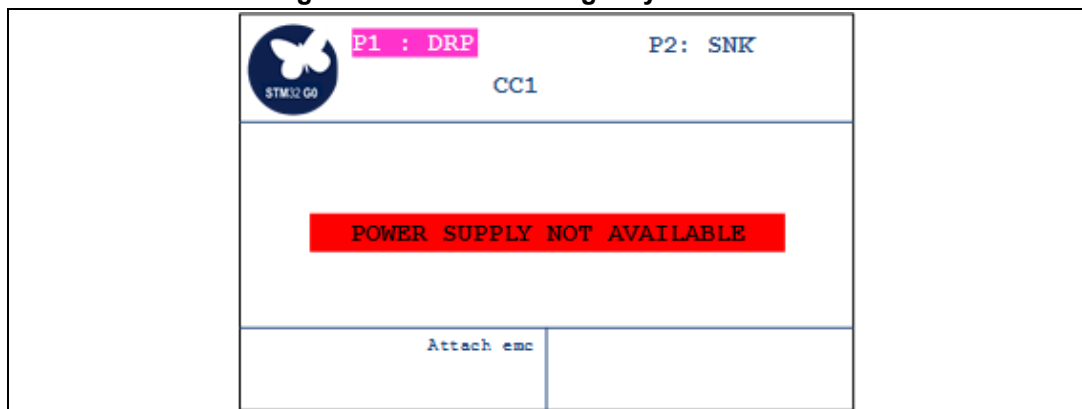
Figure 56. USB PD FW revision compatibility



4.3.3 Emergency state

If the user faces a power issue, the demonstration goes in emergency state:

Figure 57. USB PD Emergency state screen



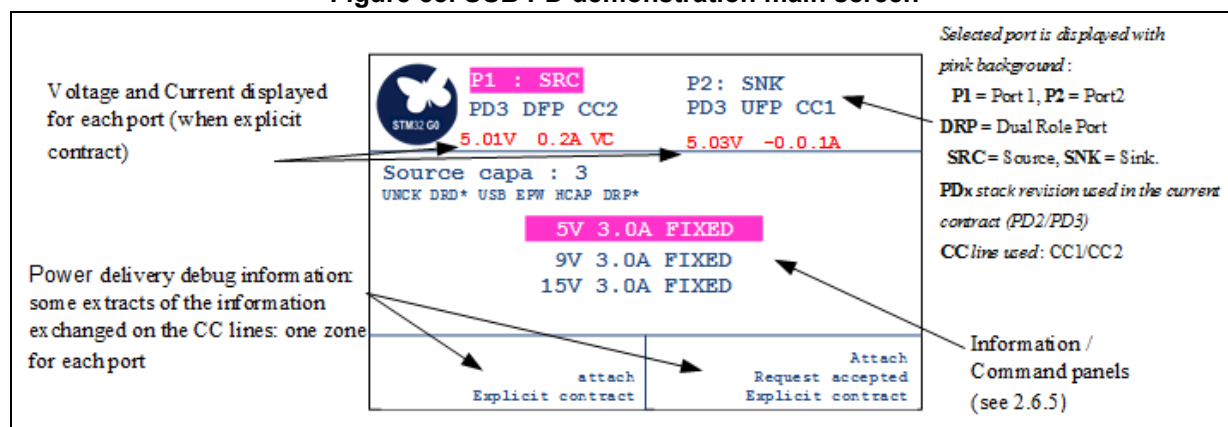
To exit from this state, the user needs to check the power supply and reset the firmware.

4.3.4 Overall presentation

The Port 1 (dual role capability) or the Port 2 (Sink only) are pluggable to an external device, or even connected together.

The information on the LCD is presented as follow:

Figure 58. USB PD demonstration main screen



4.3.5 Navigation in the menus

The tamper button is used to select the port the user wants to get control of, or to get information.

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

- Left / right: lets select the information / command panels.
- Up/ down: when possible, allows to scroll the information / action
- Center: executes the selected action


4.3.6 Available panels

Three information panels and one command panel are available:

1. Command panel:

Available commands (use the joystick up/down to scroll):

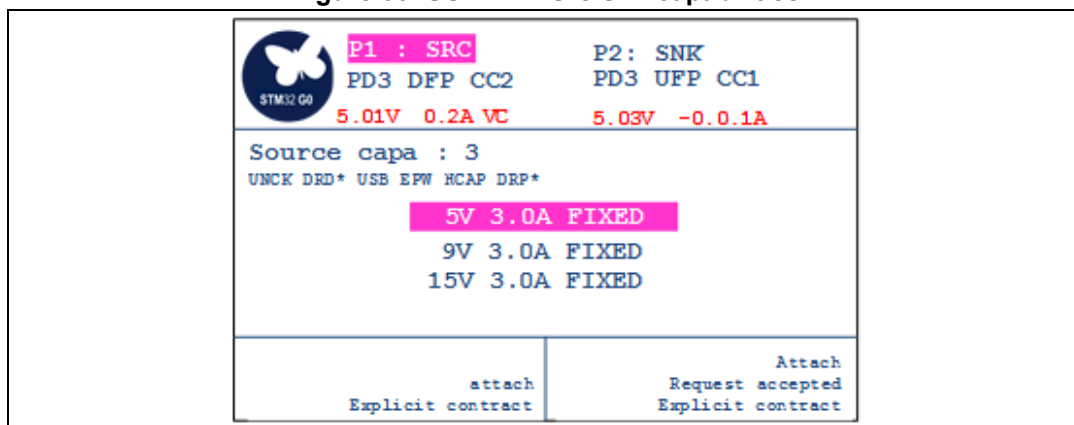
Figure 59. USB PD - available commands

	P1 : SRC PD3 DFP CC2 5.01V 0.2A VC	P2 : SNK PD3 UFP CC1 5.03V -0.0.1A
Commands : HARD RESET GOTO MIN GET SOURCE CAPA GET SINK CAPA		
Detach attach Request accepted Explicit contract		Request sent Request accepted Explicit contract detach

- Hardware reset
 - Goto Min voltage
 - Get Source Capabilities
 - Get Sink Capabilities
 - Data Role Swap
 - Power Role Swap
 - Software reset
 - Get Extended capabilities
2. Source capabilities/ Sink capabilities (depending of the role of the port)

Here is an example of source capabilities:

Figure 60. USB PD - Src/Snk capabilities

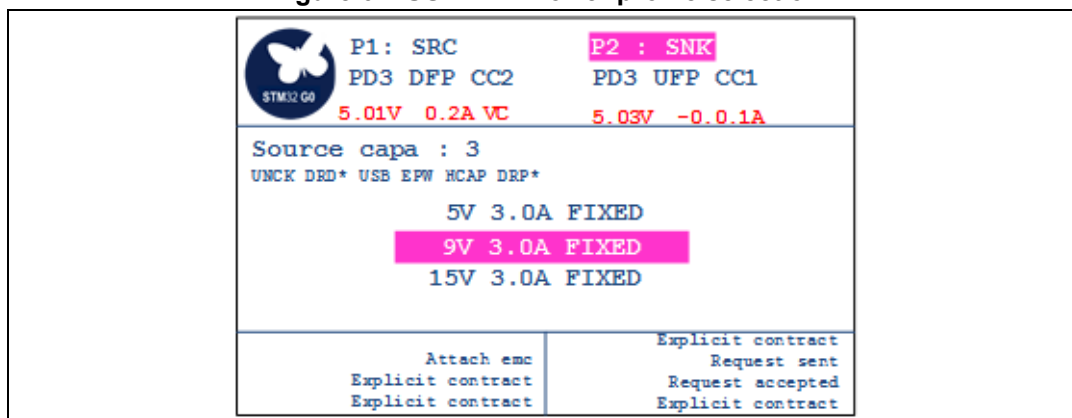


- In this example we see that the source proposes 3 PDO (Power Data Objects: see 6.4.1 (USB-IF, 2017)): 5V, 9V, 15V.
- Power delivery available features are also displayed. A star '*' next to the capability indicated that the feature is supported:
 - UNCK: Unchunked messages support
 - DRD: Dual Role 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)

Using the joystick, the user presses up/down, to go on a power profile, and selecting the profile sends a power request.

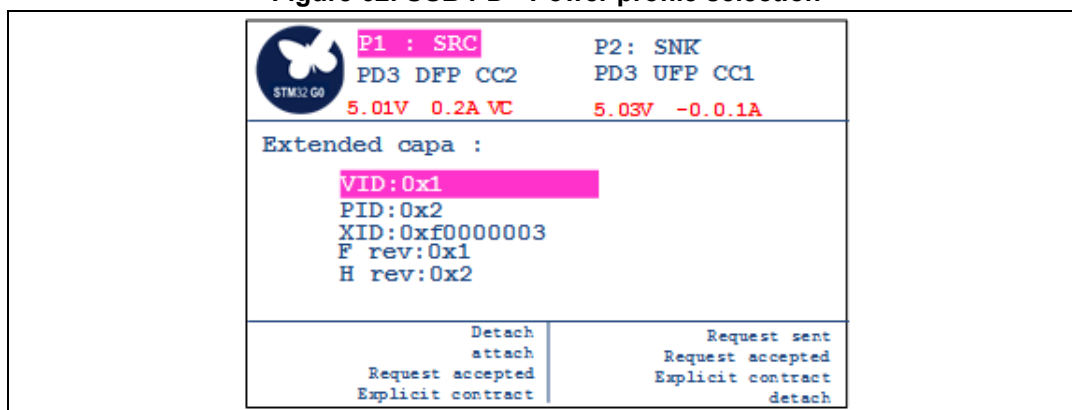
Example:

Figure 61. USB PD - Power profile selection



- We press down to select the 9V line.
 - Once we press the center button of the joystick, the request to switch to 9V is sent by the selected port (Port2) to the Source.
 - We see the line 'Request sent / Request accepted' in the debug info panel.
 - And then the power switches to 9V, as indicated by the port current/ voltage information line.
3. Extended capabilities

Figure 62. USB PD - Power profile selection



This section details the extended capabilities available when PD3 is supported: (to scroll with joystick)

- 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

Other information are displayable: (when pressing the down joystick)

- V reg: The Voltage Regulation field contains bits covering Load Step Slew Rate and Magnitude.
- See Section 7.1.12.1 of (USB-IF, 2017) for further details.
- Htime: The Holdup Time field shall contain the Source's holdup time (see Section 7.1.12.2).
- Compliance: The Compliance field shall contain the standards the Source is compliant with (see Section 7.1.12.3 of (USB-IF, 2017)).
- 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 shall contain the combinations of Peak Current that the Source supports (see Section 7.1.12.4 of (USB-IF, 2017)).
- Peak2: Peak current 2
- Peak3: Peak current 3
- Ttemp: The Touch Temp field shall report the IEC standard used to determine the surface temperature of the Source's enclosure.
- SRCin: The Source Inputs field shall identify the possible inputs that provide power to the Source. Note some Sources are only powered by a Battery (e.g. an automobile) rather than the more common mains.
- Nbbatt: The Batteries field Shall report the number of batteries the source supports
- PDP: Source's rated
- PDP: The Source PDP field shall report the Source's rated PDP as defined in PD3 spec of (USB-IF, 2017) Table 10-2.

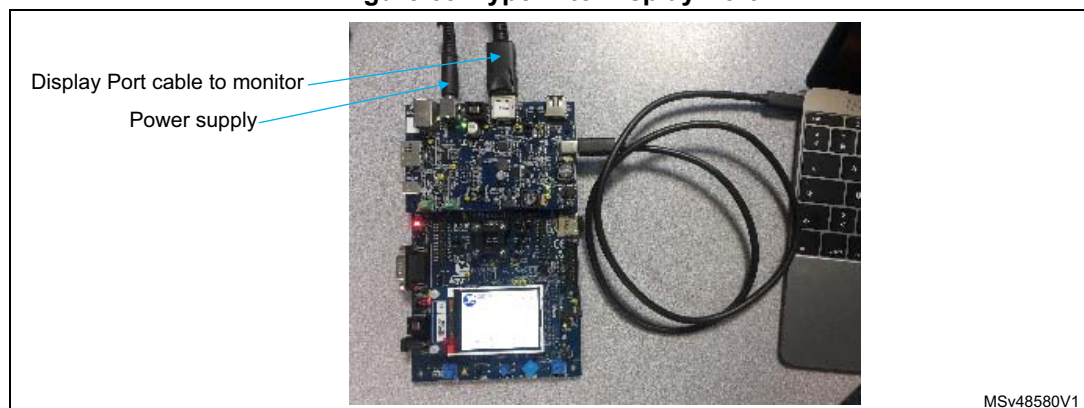
4.3.7 Display Port demonstration

Type-C to Display Port

For this demonstration the user needs a device that is able to output video from a Type-C port. This is possible with a MacBook®, or a Samsung Galaxy S8® for example.

First, the user connects the Display Port monitor cable on 'Display Port Out', and then connects the Type-C device on Port, as below.

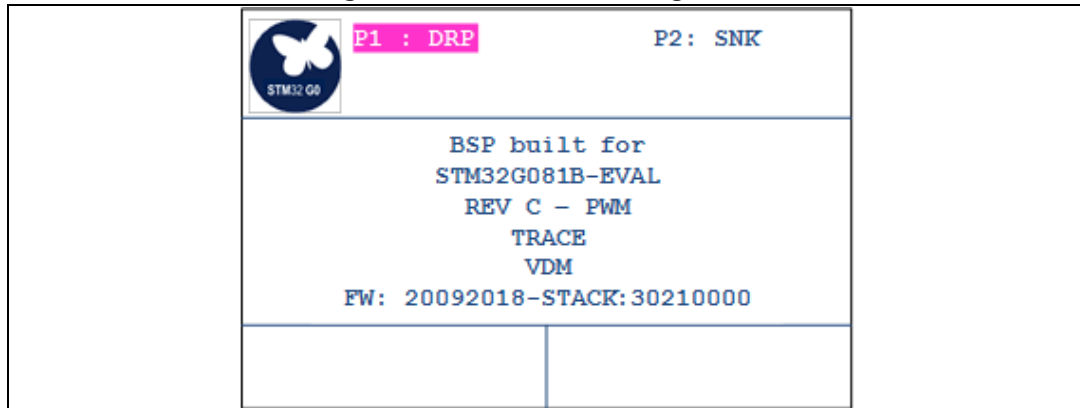
Figure 63. Type-C to Display Port



When the Display Port cable is connected, the user may see 'HPD' (hot plug detection) displayed on top right of the LCD.

And in the debug window, the user may get a contract, and see his device screen on the Display Port monitor.

Figure 64. USB PD - Hot Plug Detect

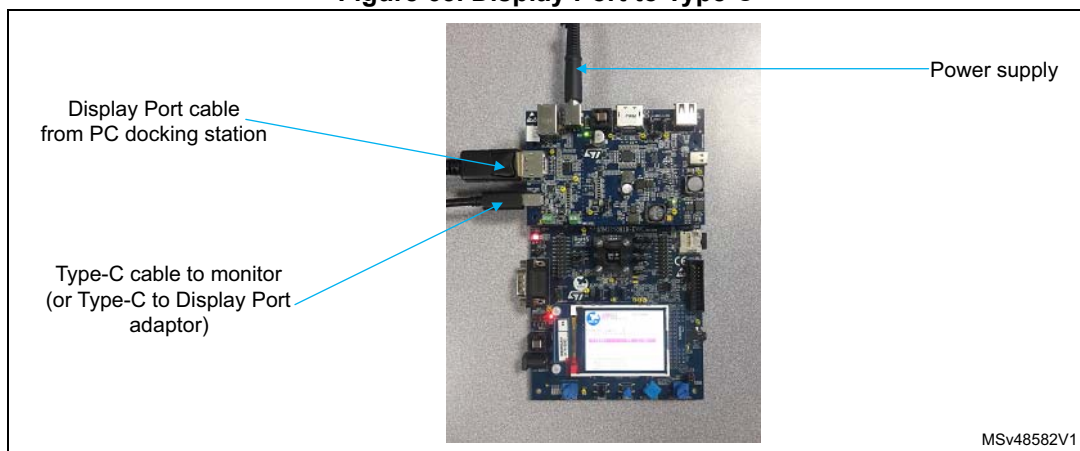


Display Port to Type-C

For this demonstration the user needs a Type-C monitor (or a Type-C to Display Port adapter)

First, the user may connect his Display Port cable from his PC docking station to 'Display Port In' on Eval G0 board. And then he may connect his Type-C monitor cable on Port 1 as below:


Figure 65. Display Port to Type-C



The user is able to see below debug information: 'Explicit contract', 'send vdm enter mode', and 'vdm enter mode ack'.

Which means that the VDM (vendor define message) capabilities have been read by the G0, and that the MUX path is correctly set on the G0 Eval board.

Figure 66. USB PD - Vendor Define Message

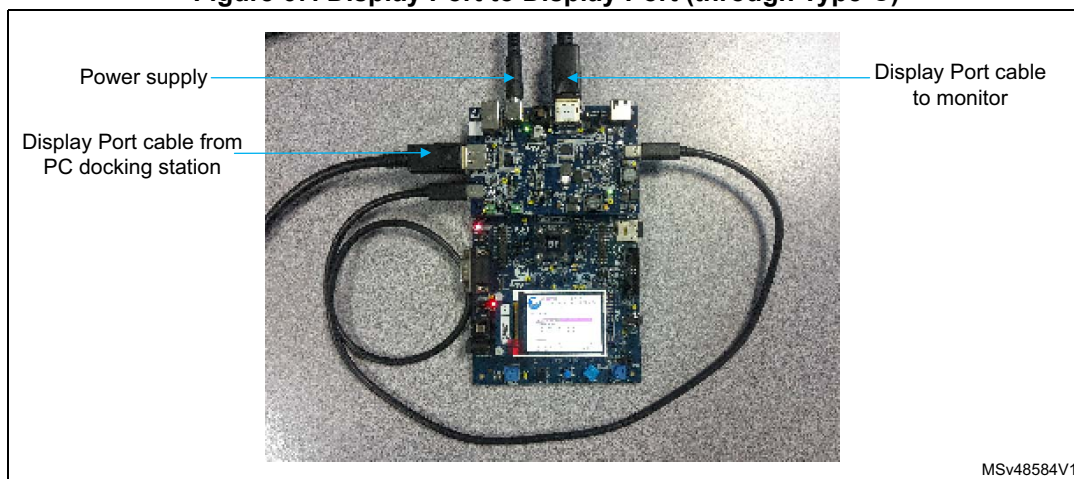
	P1 : SRC PD2 DFP CC2 5.01V 0.01A VC	P2 : SNK
Commands :		
HARD RESET GOTO MIN GET SOURCE CAPA GET SINK CAPA		
attach Explicit contract Send vdm enter mode vdm enter mode ACK		Request sent Request accepted Explicit contract detach

Display Port to Display Port (through Type-C)

This demonstration is available even if the user does not have any Type-C monitor, or has a device that may drive video through Type-C.

He may plug a Display Port cable out of a PC (or docking station) to 'Display Port In', a Display Port cable to a monitor on 'Display Port Out', and then a Type-C cable between Type-C port 1 and 2.

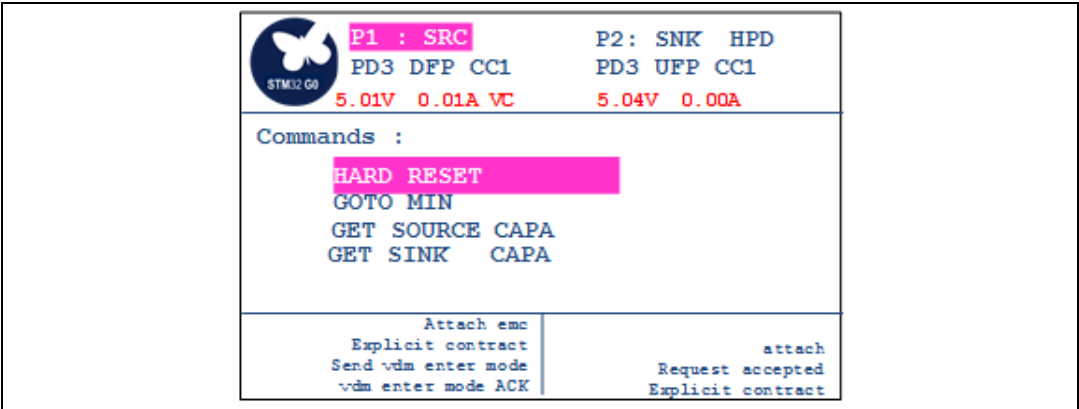
Figure 67. Display Port to Display Port (through Type-C)



MSv48584V1

A contract is established between the two Type-C ports as below:

Figure 68. USB PD - DP contract established

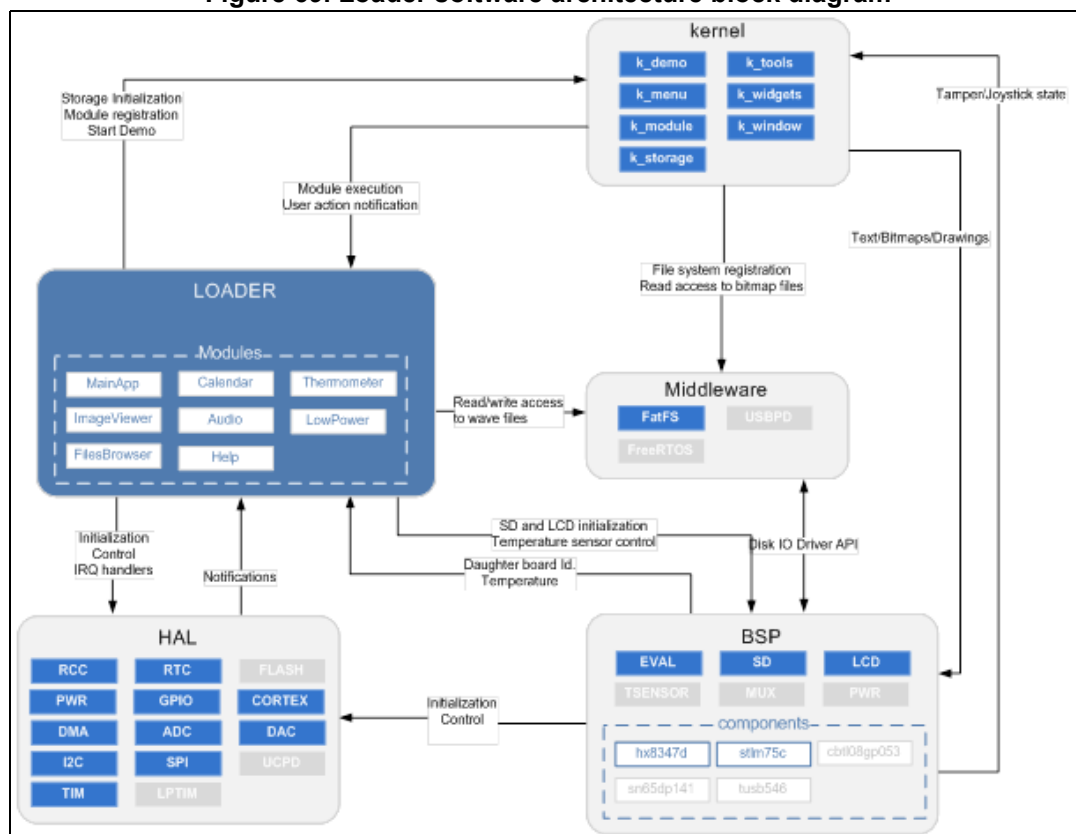


5 Software architecture

5.1 Demonstration loader

Figure 69 represents the main interactions between the demonstration loader and the surrounding software modules. The loader uses the services of the kernel to check whether or not the micro SD card is inserted, to initialize the file system and to retrieve the legacy or UCPD binary image related information. The loader also uses the kernel to manage the progress bar displayed during the flash loading process or to display an error message when a problem is detected. During the flash programming operation, the loader relies on the services provided by the FatFS to perform read accesses to the demonstration binary file. Each read block is 512 bytes large and is written to the flash memory using the fast flash programming mode allowing to reduce the flash page programming time. The Loader uses the HAL API mainly to configure the system clock at 64 MHz, enable access to the backup registers (used to store the loaded demonstration identifier) and to program the flash memory. Also HAL IRQ handlers are called from within the interrupts handlers implemented within the Loader (e.g. EXTI IRQ handler used to detect SD card insertion). The Loader uses the EVAL BSP to determine the type of daughterboard connected (or not) to the STM32G081B-EVAL evaluation board. SD and LCD BSP are used to initialize the hardware resources required to manage the Micro SD card and the hx8347d component (240x320 TFT color LCD).

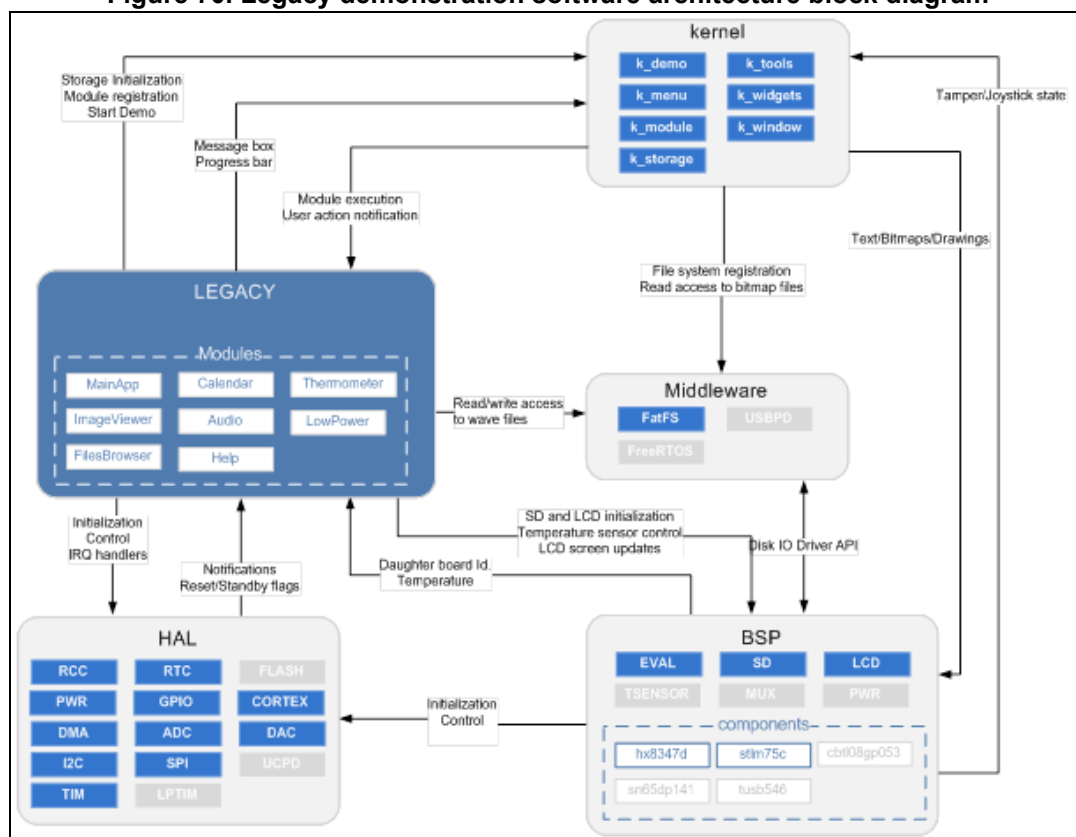
Figure 69. Loader software architecture block diagram



5.2 Legacy demonstration

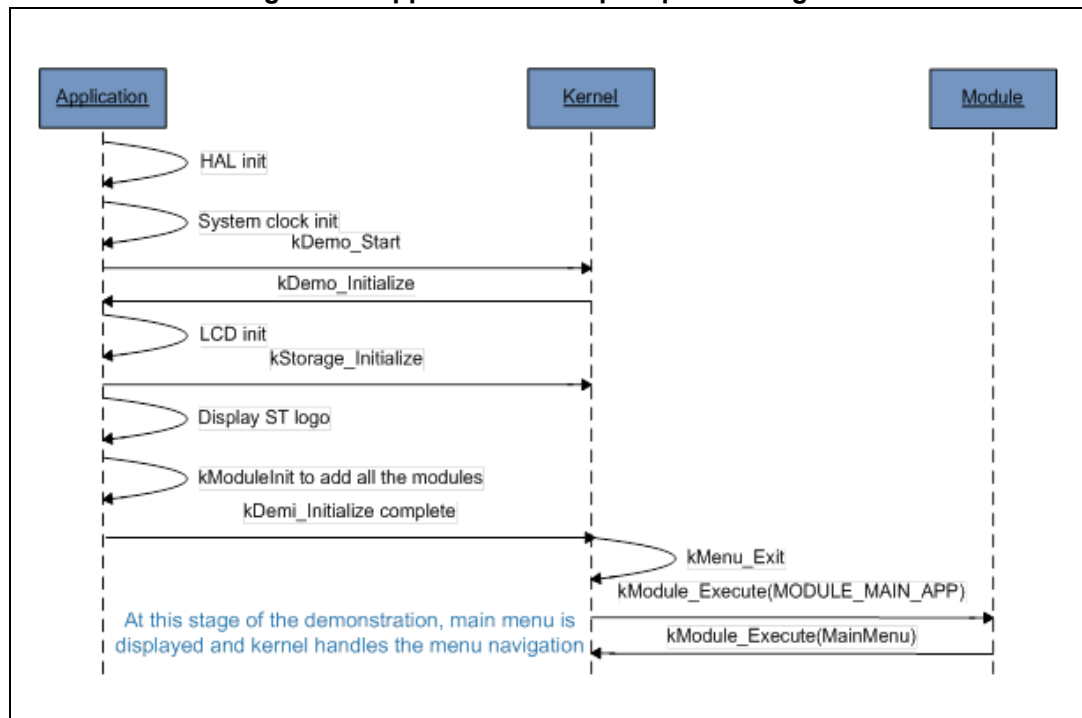
Figure 70 represents the main interactions between the legacy demonstration and the surrounding software modules. The legacy demonstration is a set of modules accessible through a main menu displayed on the LCD screen. At application startup the system clock is configured at 64 MHz and the file system, the BSD and the LCD BSPs are initialized. If all the graphic resources (bitmap files) needed by the modules are present on the SD 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. 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 STM32G0xx interrupts (TAMPER, EXTI, DMA) are handled by *Figure 70* is also part of the application and is used, as usual, to map the interrupt vector on the driver HAL driver, depending on the module requirement.

Figure 70. Legacy demonstration software architecture block diagram



The sequence diagram below summarizes the application startup until the Kernel takes the lead.

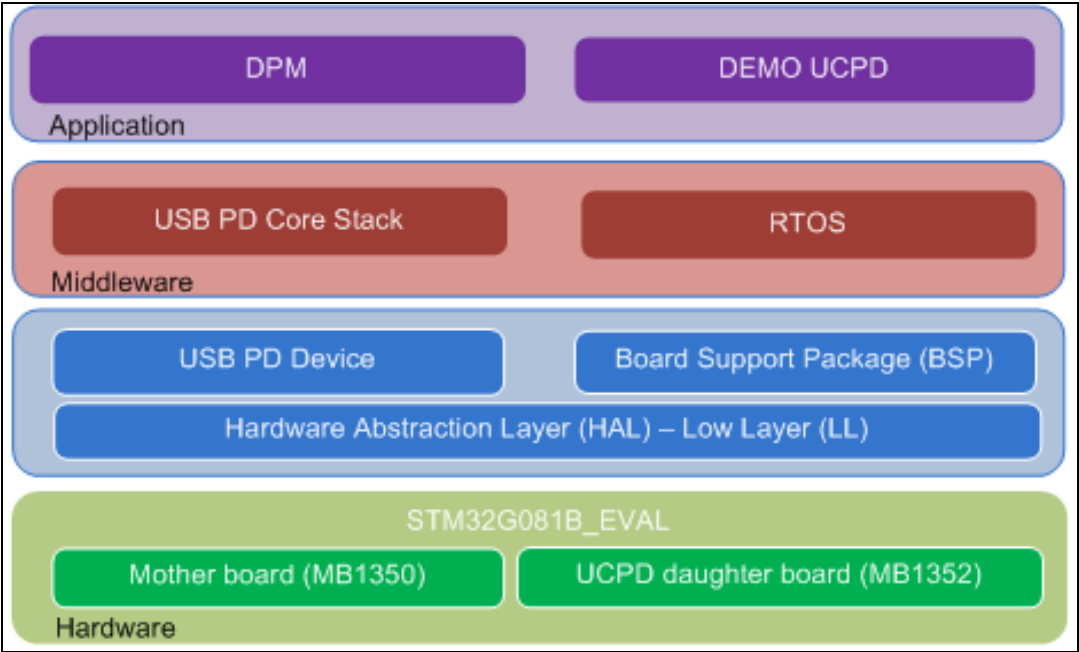
Figure 71. Application startup sequence diagram



5.3 UCPD demonstration

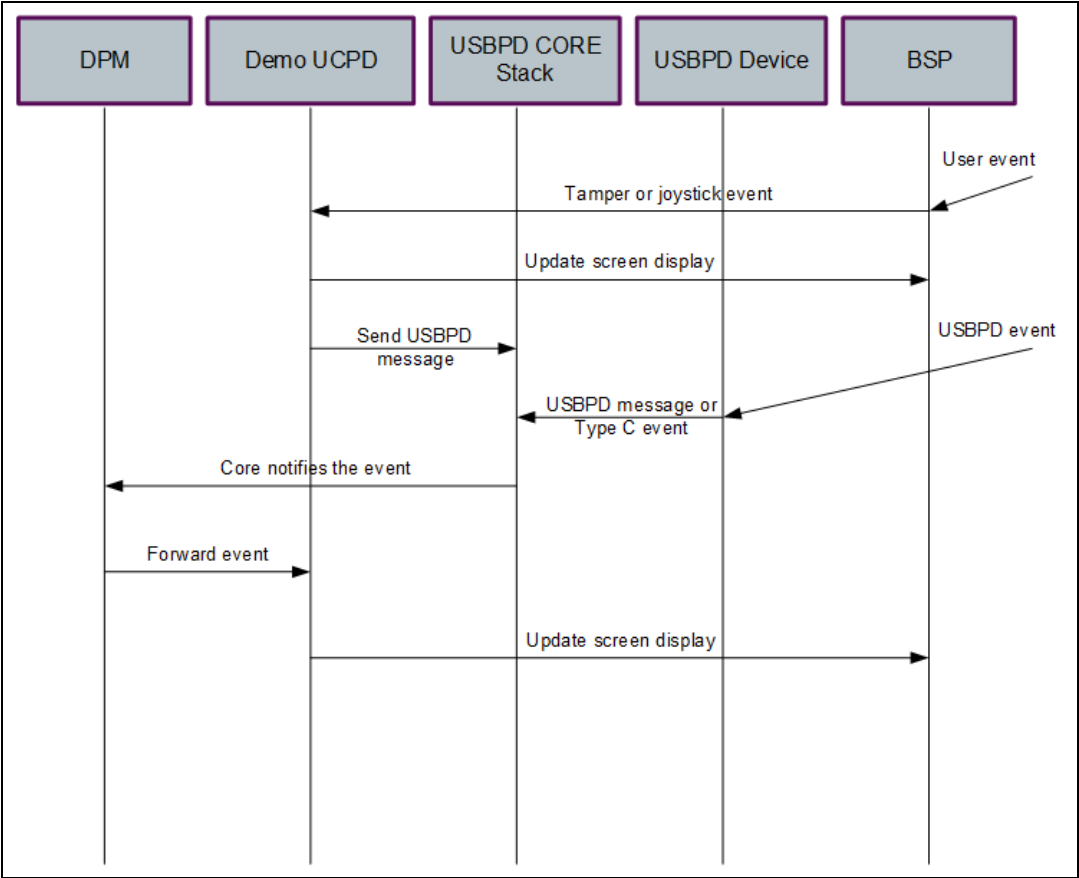
Figure 72 represents the main interactions between the UCPD demonstration and a standard USBPD Application. The UCPD demonstration is an RTOS task which receives events forwarded by DPM (Device Policy Manager), the events are mainly coming from the USBPD stack (like attachment, detachment ...) and manages the display of those information on the screen by using the BSP interface. At application startup the system clock is configured at 64 MHz. USBPD application initializes PORT1 as DRP, PORT2 as SYNC and the demonstration is ready to catch DPM events. User uses the TAMPER and JOYSYICK to navigate inside the menu and to execute actions.

Figure 72. UCPD demonstration software architecture block diagram



The sequence diagram below summarizes the application processing of UCPD application.

Figure 73. Application processing



5.4 Kernel API overview

5.4.1 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 application chapter for details about these functions)

Table 7. k_demo API

Function	Description
kDemo_Initialization	Initialize the demonstration (function defined on application side)
kDemo_UnInitialization	Initialize the demonstration (function defined on application side)
kDemo_Start	Start the demonstration (Initialize, Run, UnInitialize, Exit)
kDemo_Execute	Initialize kMenu and execute the module "MODULE_MAIN_APP"

5.4.2 k_menu

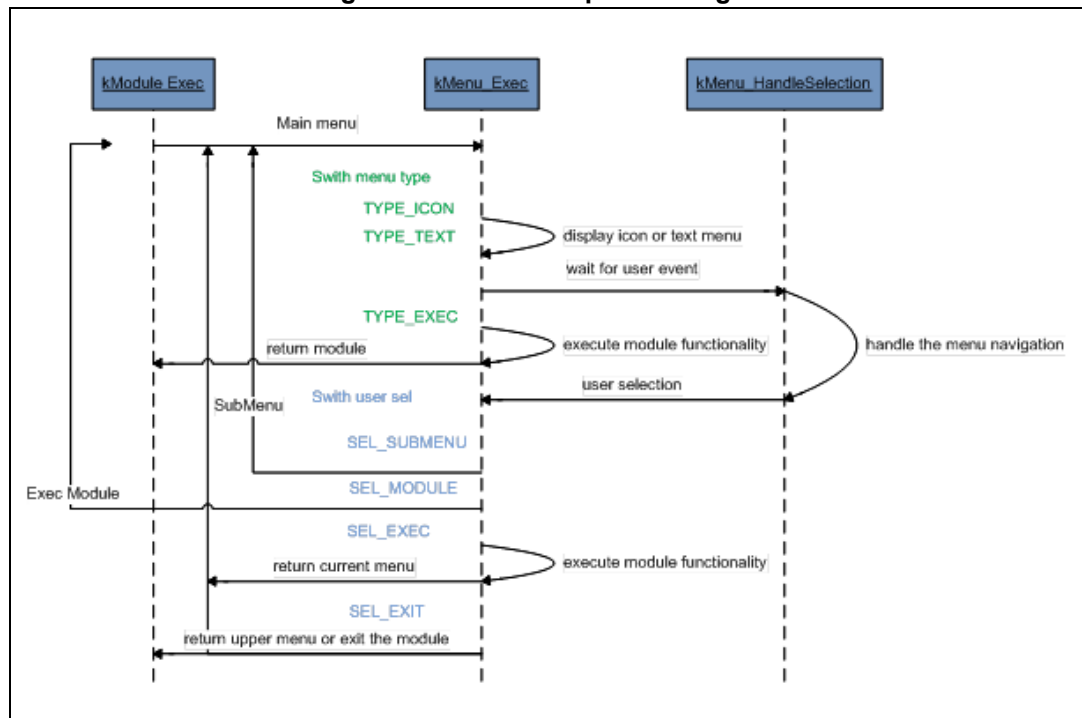
The module execution is started by a call to the function kMenu_Execute. This kernel function handles the menu navigation and the execution functionalities (thanks to the structure t_menu defined inside the module).

Table 8. k_menu API

Function	Description
kMenu_Init	Initialize the joystick
kMenu_EventHandler	GPIO Event handler
kMenu_Execute	Function to execute a menu
kMenu_Header	Function to display header information
kMenu_HandleSelectionExecute	The function handle the menu navigation

[Figure 74](#) shows the execution flow for a menu.

Figure 74. KMenu sequence diagram



5.4.3 k_module

This kernel part centralizes information about all modules available inside demonstration firmware: Function `kModule_Init` (defined on application side) aims to register all the modules present with the help of function `kModule_Init` (thanks to the structure `K_ModuleItem_Typedef` defined inside the module).

Table 9. k_module API

Function	Description
<code>kModule_Init</code>	Defined on application side
<code>kModule_Add</code>	Add module inside the module list
<code>kModule_CheckResource</code>	Check the module resource
<code>kModule_Execute</code>	Execute the module

5.4.4 k_storage

The `K_Storage` handles only the SD card storage and provides some services to simplify the modules development.

(To be completed)

Table 10. k_storage API

Function	Description
kStorage_Init	Mount the SD card file system
kStorage_DelInit	Unmount the file system
kStorage_GetStatus	Return SD card presence
kStorage_GetDirectoryFiles	Return the name of the file present inside a directory
kStorage_FileExist	Check if a file exist
kStorage_GetFileInfo	Return file information
kStorage_OpenFileDrawBMP	Open a bmp file and display it (only file of 8ko max)
kStorage_OpenFileDrawPixel	Open a bmp file and display it line by line
kStorage_GetExt	Return the file extension

5.4.5 k_window

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

Table 11. k_window API

Function	Description
kWindow_Popup	Display a popup
kWindow_Error	Display a red popup with an error message

5.4.6 k_widgets

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

Table 12. k_widgets API

Function	Description
kWidgets_ProgressBarCreate	Create and display an empty progress bar object
kWidgets_ProgressBarDestroy	Destroy the progress bar object
kWidgets_ProgressBarUpdate	Update the progress bar content
kWidgets_ProgressBarReset	Reset the progress bar content

5.4.7 k_tools

The K_Tools provides tools for module development.

Table 13. k_tools API

Function	Description
kTools_Buffercmp	Return 0 if the both buffers are identical

5.5 FatFS API overview

FatFS APIs function set given by the file system interface are listed in [Table 14](#).

Table 14. FatFS API

Function	Description
f_mount	Register/Unregister a work area
f_open	Open/Create a file
f_close	Close a file
f_read	Read file
f_write	Write file
f_lseek	Move read/write pointer, Expand file size
f_truncate	Truncate file size
f_sync	Flush cached data
f_opendir	Open a directory
f_readdir	Read a directory item
f_getfree	Get free clusters
f_stat	Get file status
f_mkdir	Create a directory
f_unlink	Remove a file or directory
f_chmod	Change attribute
f_utime	Change timestamp
f_rename	Rename/Move a file or directory
f_mkfs	Create a file system on the drive
f_forward	Forward file data to the stream directly
f_chdir	Change current directory
f_chdrive	Change current drive
f_getcwd	Retrieve the current directory
f_gets	Read a string
f_putc	Write a character
f_puts	Write a string
f_printf	Write a formatted string

5.6 USBPD API overview

Table 15. USBPD API

Function	Description
USBPD_CAD_AssertRd	Set as SNK
USBPD_CAD_AssertRp	Set as SRC
USBPD_CAD_EnterErrorRecovery	Set type C state machine in error recovery state
USBPD_CAD_Init	Initialize the CAD module for a specified port
USBPD_CAD_PortEnable	Enable or Disable CAD port
USBPD_CAD_Process	Main CAD task
USBPD_CAD_SetRpResistor	Set the default Rd resistor
USBPD_CAD_Task	-
USBPD_DPM_CADCallback	Callback reporting events on a specified port from CAD layer
USBPD_DPM_CADTaskWakeUp	Wake up CAD task
USBPD_DPM_Init	Initialize DPM (port power role, PWR_IF, CAD and PE initialization procedures)
USBPD_DPM_TimerCounter	Initialize DPM timer counter
USBPD_PE_CheckLIB	Check coherence between lib selected and the lib include inside the project
USBPD_PE_ExecFastRoleSwapSignalling	Execute an FRS signaling
USBPD_PE_GetMemoryConsumption	Return the need of the stack in terms of dynamized allocation
USBPD_PE_Init	Initialize policy engine layer for a port with a specified role
USBPD_PE_InitVDM_Callback	Initialize VDM callback functions in PE
USBPD_PE_IsCableConnected	Function called by DPM to set the cable status connected or disconnected
USBPD_PE_Request_CtrlMessage	Generic function to send a control message
USBPD_PE_Request_DataMessage	Generic function to send a data message
USBPD_PE_Request_HardReset	Called by DPM to request PE to perform a hard reset
USBPD_PE_Send_Request	Evaluate received Capabilities Message from Source port and prepare the request message
USBPD_PE_SendExtendedMessage	Send an chunked extended message and store data received inside a buffer
USBPD_PE_SetTrace	Set the trace pointer
USBPD_PE_StateMachine_DRP	Policy Engine dual role port main state machine
USBPD_PE_StateMachine_SNK	Policy Engine Sink port main state machine
USBPD_PE_StateMachine_SRC	Policy Engine Source port main state machine
USBPD_PE_SVDM_RequestAttention	Called by DPM to request the PE to perform a VDM attention

Table 15. USBPD API (continued)

Function	Description
USBPD_PE_SVDM_RequestIdentity	Called by DPM to request the PE to perform a VDM identity request
USBPD_PE_SVDM_RequestMode	Called by DPM to request the PE to perform a VDM Discovery mode message on one SVID
USBPD_PE_SVDM_RequestModeEnter	Called by DPM to request the PE to perform a VDM Discovery mode message on one SVID
USBPD_PE_SVDM_RequestModeExit	Called by DPM to request the PE to perform a VDM mode exit
USBPD_PE_SVDM_RequestSpecific	Called by DPM to request the PE to send a specific SVDM message
USBPD_PE_SVDM_RequestSVID	Called by DPM to request the PE to perform a VDM SVID request
USBPD_PE_Task	
USBPD_PE_TaskWakeUp	Wake up PE task
USBPD_PE_TimerCounter	Increment PE Timers tick
USBPD_PE_UVDM_RequestMessage	Called by DPM to request the PE to send a UVDM message
USBPD_PRL_TimerCounter	Decrement the PRL timers values
USBPD_TRACE_Add	Add information in debug trace buffer
USBPD_TRACE_Init	Initialize the TRACE module
USBPD_TRACE_TX_Process	Main Trace TX process to push data on the media

5.7 BSP API overview

5.7.1 BSP EVAL

Table 16. BSP EVAL API

Function	Description
BSP_JOY_Init	Configures joystick GPIO and EXTI modes
BSP_JOY_GetState	Returns the current joystick status
BSP_JOY_DeInit	Reconfigures all GPIOs used as buttons of the joystick
BSP_LED_Init	Configures LED GPIO
BSP_LED_DeInit	De-initializes LEDs
BSP_LED_Toggle	Toggles the selected LED
BSP_LED_On	Turns selected LED On
BSP_LED_Off	Turns selected LED Off
BSP_PB_Init	Configures Tamper Button GPIO or EXTI Line

Table 16. BSP EVAL API (continued)

Function	Description
BSP_PB_GetState	Returns the selected button state
BSP_COM_Init	Configures COM port
BSP_DB_GetId	Identifies the daughterboard plugged on daughterboard connectors
BSP_GetVersion	Returns the STM32G081B-EVAL BSP driver revision

5.7.2 BSP TSENSOR

Table 17. BSP TSENSOR API

Function	Description
BSP_TSENSOR_Init	Initializes peripherals used by the I2C Temperature Sensor driver
BSP_TSENSOR_ReadStatus	Returns the Temperature Sensor status
BSP_TSENSOR_ReadTemp	Read Temperature register of STLM75

5.7.3 BSP SD

Table 18. BSP SD API

Function	Description
BSP_SD_Init	Initializes the SD communication
BSP_SD_GetStatus	Returns the SD status
BSP_SD_IsDetected	Detects if SD card is correctly plugged in the memory slot or not
BSP_SD_GetCardInfo	Returns information about specific card
BSP_SD_ReadBlocks	Reads block(s) from a specified address in the SD card, in polling mode
BSP_SD_WriteBlocks	Writes block(s) to a specified address in the SD card, in polling mode
BSP_SD_Erase	Erases the specified memory area of the given SD card

5.7.4 BSP LCD

Table 19. BSP LCD API

Function	Description
BSP_LCD_Init	Initializes the LCD
BSP_LCD_GetXSize	Gets the LCD X size
BSP_LCD_GetYSize	Gets the LCD Y size
BSP_LCD_SetBackColor	Sets the LCD background color
BSP_LCD_GetBackColor	Gets the LCD background color
BSP_LCD_SetTextColor	Sets the LCD text color

Table 19. BSP LCD API (continued)

Function	Description
BSP_LCD_GetTextColor	Gets the LCD text color
BSP_LCD_SetFont	Sets the LCD text font
BSP_LCD_GetFont	Gets the LCD text font
BSP_LCD_Clear	Clears the whole LCD
BSP_LCD_DisplayStringAt	Displays characters on the LCD
BSP_LCD_DisplayStringAtLine	Displays characters at a specific line on the LCD
BSP_LCD_ClearStringLine	Clears the selected line
BSP_LCD_DisplayChar	Displays one character
BSP_LCD_FillRect	Draws a full rectangle
BSP_LCD_FillCircle	Draws a full circle
BSP_LCD_FillEllipse	Draws a full ellipse
BSP_LCD_DrawBitmap	Draws a bitmap picture loaded in the internal Flash (32 bpp)
BSP_LCD_DrawPixel	Draws a pixel on LCD
BSP_LCD_ReadPixel	Reads a LCD pixel
BSP_LCD_DrawHLine	Draws an horizontal line
BSP_LCD_DrawVLine	Draws a vertical line
BSP_LCD_DrawLine	Draws an uni-line (between two points)
BSP_LCD_DrawRect	Draws a rectangle
BSP_LCD_DrawCircle	Draws a circle
BSP_LCD_DrawPolygon	Draws an poly-line (between many points)
BSP_LCD_DrawEllipse	Draws an ellipse
BSP_LCD_DisplayOff	Disables the display
BSP_LCD_DisplayOn	Enables the display

5.7.5 BSP PWR

Table 20. BSP PWR API

Function	Description
BSP_PWR_VBUSInit	Initializes the hardware resources used by the Type-C power delivery (PD) controller
BSP_PWR_VBUSDeInit	Releases the hardware resources used by the Type-C power delivery (PD) controller
BSP_PWR_VBUSIsGPIO	Get the VSENSE_DCDC measurement with DCDC_EN = 0 and = 1
BSP_PWR_VBUSOn	Enables power supply over VBUS
BSP_PWR_VBUSOff	Disables power supply over VBUS

Table 20. BSP PWR API (continued)

Function	Description
BSP_PWR_VBUSSetVoltage_Fixed	Set a fixed PDO and manage the power control
BSP_PWR_VBUSSetVoltage_Variable	Set a variable PDO and manage the power control
BSP_PWR_VBUSSetVoltage_Battery	Set a Battery PDO and manage the power control
BSP_PWR_VBUSSetVoltage_APDO	Set a APDO and manage the power control
BSP_PWR_VBUSGetVoltage	Gets actual voltage level measured on the VBUS line
BSP_PWR_VBUSGetCurrent	Gets actual current level measured on the VBUS line
BSP_PWR_VCONNInit	Initializes VCONN sourcing
BSP_PWR_VCONNDeInit	De-initializes VCONN sourcing
BSP_PWR_VCONNOn	Enables VCONN sourcing
BSP_PWR_VCONNOff	Disables VCONN sourcing
BSP_PWR_SetVBUSDisconnectThreshold	Sets the VBUS disconnection voltage threshold
BSP_PWR_RegisterVBUSDetectCallback	Registers USB Type-C Current callback function
BSP_PWR_VBUSIsOn	Gets actual VBUS status
BSP_PWR_VCONNIsOn	Gets actual VCONN status
BSP_PWR_DCDCGetVoltage	Gets actual DCDC voltage level

5.7.6 BSP MUX

Table 21. BSP MUXAPI

Function	Description
BSP_MUX_Init	Initializes the hardware resources used by the Type-C MUX assigned to a given Type-C MUX
BSP_MUX_DeInit	Frees the hardware resources used by the Type-C MUX
BSP_MUX_Enable	Power on the Type-C MUX
BSP_MUX_Disable	Power down the Type-C MUX
BSP_MUX_SetDPPinAssignment	Sets the pin assignment of the USB Type-C connector when it operates in one of the following mode
BSP_MUX_SetEQGain	Sets the equalizer gain for all the channels
BSP_MUX_SetHPDState	Sets the HPD level seen by the DP source device
BSP_MUX_GetHPDState	Retrieves actual HPD level of the DP source device
BSP_MUX_GetUSB3DetectState	Retrieves actual USB3 connection state
BSP_MUX_HPDIRQ	Generates an HPD_IRQ towards the DP source device

Table 21. BSP MUXAPI (continued)

Function	Description
BSP_MUX_Detect_HPD	Displays port hot plug detection (HPD) function
BSP_MUX_Detect_USB3	USB3 detection function
BSP_MUX_RegisterHPDCallbackFunc	HPD callback function registration
BSP_MUX_RegisterUSB3DetectCallbackFunc	USB3 detection callback function registration
BSP_MUX_DumpDeviceRegisters	Dump the register content of a device

6 Memory footprint

Following sections summarize memory allocations for RAM and flash for the loader the legacy demonstration and the UCPD 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

Note: values listed in [Table 22](#), [Table 23](#) and [Table 24](#) are extracted from the map file generated by the IAR toolchain for ARM.

6.1 Demonstration loader memory footprint

Table 22. Demonstration loader memory footprint

Module	ro code	rw code	ro data	rw data
Application	-	-	-	-
Common	-	-	-	-
sd_diskio.o	156	-	21	1
utils.o	32	-	-	-
Core	-	-	-	-
k_storage.o	368	-	-	569
k_widgets.o	188	-	-	-
k_window.o	228	-	-	-
EWARM	-	-	-	-
startup_stm32g081xx.o	270	-	-	-
User	-	-	-	-
main.o	1544	72	122	124
stm32g0xx_it.o	72	-	-	-
stm32g0xx_hal_msp.o	104	-	-	-
Drivers	-	-	-	-
BSP	-	-	-	-
Components	-	-	-	-
hx8347d.o	1108	-	21	697
STM32G081B_EVAL	-	-	-	-
stm32g081b_eval.o	1456	-	47	120
stm32g081b_eval_lcd.o	976	-	3809	896
stm32g081b_eval_sd.o	2128	-	-	3

Table 22. Demonstration loader memory footprint (continued)

Module	ro code	rw code	ro data	rw data
CMSIS	-	-	-	-
system_stm32g0xx.o	80	-	1	4
STM32G0xx_HAL_Drivers	-	-	-	-
stm32g0xx_hal.o	186	-	5	12
stm32g0xx_hal_adc.o	2012	-	-	-
stm32g0xx_hal_adc_ex.o	168	-	-	-
stm32g0xx_hal_cortex.o	140	-	-	-
stm32g0xx_hal_flash.o	44	-	-	-
stm32g0xx_hal_gpio.o	706	-	-	-
stm32g0xx_hal_pwr.o	16	-	-	-
stm32g0xx_hal_rcc.o	1496	-	-	-
stm32g0xx_hal_rcc_ex.o	548	-	-	-
stm32g0xx_hal_rtc.o	328	-	-	-
stm32g0xx_hal_rtc_ex.o	40	-	-	-
stm32g0xx_hal_spi.o	2156	-	-	-
Middlewares	-	-	-	-
Fat FS	-	-	-	-
Core	-	-	-	-
diskio.o	132	-	-	-
ff.o	6980	-	-	204
ff_gen_drv.o	72	-	-	16
Options	-	-	-	-
syscall.o	16	-	-	-
unicode.o	1132	-	-	-
Others	852	-	48	10252
Total	25734	72	4074	12898

6.2 Legacy demonstration memory footprint

Table 23. Legacy demonstration memory footprint

Module	ro code	rw code	ro data	rw data
Application	-	-	-	-
Common	-	-	-	-
sd_diskio.o	152	-	21	1
utils.o	220	-	-	-

Table 23. Legacy demonstration memory footprint (continued)

Module	ro code	rw code	ro data	rw data
Core	-	-	-	-
k_demo	22	-	-	-
k_menu	1108	-	-	4
k_module	128	-	-	200
k_storage.o	776	-	-	629
k_widgets.o	240	-	-	-
k_window.o	228	-	-	-
Demo	-	-	-	-
main.o	1080	-	-	52
stm32g0xx_it.o	132	-	-	-
stm32g0xx_hal_msp.o	208	-	-	-
EWARM	-	-	-	-
startup_stm32g081xx.o	262	-	-	-
Modules	-	-	-	-
Audio	-	-	-	-
app_audio.o	56	-	160	1713
wave_player.o	2444	-	225	604
wave_recorder.o	2128	-	-	164
Calendar	-	-	-	-
app_calendar	3244	-	560	48
FileBrowser	-	-	-	-
app_filesbrowser.o	1656	-	88	2586
Help	-	-	-	-
app_help.o	2600	-	556	3
ImageViewer	-	-	-	-
app_imagesbrowser.o	240	-	52	2
LowPower	-	-	-	-
app_lowpower.o	1504	-	468	2
MainApp	-	-	-	-
app_main.o	560	-	552	257
Thermometer_LDR	-	-	-	-
app_thermometer_LDR.o	1284	-	92	102
Drivers	-	-	-	-
BSP	-	-	-	-
Components	-	-	-	-

Table 23. Legacy demonstration memory footprint (continued)

Module	ro code	rw code	ro data	rw data
stlm75.c	204	-	4	16
hx8347d.o	1100	-	14	697
STM32G081B_EVAL	-	-	-	-
stm32g081b_eval.o	2224	-	57	228
stm32g081b_eval_lcd.o	1024	-	14832	920
stm32g081b_eval_sd.o	2132	-	-	3
stm32g081b_eval_tsensor.o	100	-	-	8
CMSIS	-	-	-	-
system_stm32g0xx.o	220	-	33	4
STM32G0xx_HAL_Drivers	-	-	-	-
stm32g0xx_hal.o	186	-	3	12
stm32g0xx_hal_adc.o	2480	-	-	-
stm32g0xx_hal_adc_ex.o	168	-	-	-
stm32g0xx_hal_cortex.o	172	-	-	-
stm32g0xx_hal_dac.o	830	-	-	-
stm32g0xx_hal_dac_ex.o	56	-	-	-
stm32g0xx_hal_dma.o	960	-	-	-
stm32g0xx_hal_gpio.o	690	-	-	-
stm32g0xx_hal_i2c.o	1908	-	-	-
stm32g0xx_hal_pwr.o	152	-	-	-
stm32g0xx_hal_rcc.o	1556	-	-	-
stm32g0xx_hal_rcc_ex.o	544	-	-	-
stm32g0xx_hal_rtc.o	1582	-	-	-
stm32g0xx_hal_rtc_ex.o	38	-	-	-
stm32g0xx_hal_spi.o	2244	-	-	-
stm32g0xx_hal_tim.o	368	-	-	-
stm32g0xx_hal_tim_ex.o	92	-	-	-
Middlewares	-	-	-	-
Fat FS	-	-	-	-
Core	-	-	-	-
diskio.o	132	-	-	-
ff.o	8308	-	-	204
ff_gen_drv.o	72	-	-	16
Options	-	-	-	-
syscall.o	16	-	-	-

Table 23. Legacy demonstration memory footprint (continued)

Module	ro code	rw code	ro data	rw data
unicode.o	1132	-	-	-
Others	13076	-	86	12920
Total	64038	0	17803	21395

6.3 UCPD demonstration memory footprint

Table 24. UCPD demonstration memory footprint

Module	ro code	rw code	ro data	rw data
Application	-	-	-	-
EWARM	-	-	-	-
system_stm32g0xx.o	220	-	33	4
User	-	-	-	-
demo_application.o	7132	-	9473	1033
main.o	252	-	-	-
stm32g0xx_it.o	156	-	-	-
usbpd_vdm_user.o	1940	-	18	108
usbpd_dpm_user.o	3912	-	4	596
usbpd_pwr_if.o	1488	-	-	128
Drivers	-	-	-	-
BSP	-	-	-	-
Components	-	-	-	-
cbtl08gp053.o	1112	-	9	24
hx8347d.o	1108	-	21	697
sn65dp141.o	578	-	11	28
tusb546.o	658	-	20	54
STM32G081B_EVAL	-	-	-	-
stm32g081b_eval.o	1364	-	78	212
stm32g081b_eval_lcd.o	1036	-	4192	904
stm32g081b_eval_mux.o	1008	-	49	132
stm32g081b_eval_pwr.o	3532	-	135	392
CMSIS	-	-	-	-
startup_stm32g081xx.o	210	-	-	-
STM32G0xx_HAL_Drivers	-	-	-	-
stm32g0xx_hal.o	186	-	4	12
stm32g0xx_hal_adc.o	3486	-	-	-

Table 24. UCPD demonstration memory footprint (continued)

Module	ro code	rw code	ro data	rw data
stm32g0xx_hal_adc_ex.o	168	-	-	-
stm32g0xx_hal_cortex.o	136	-	-	-
stm32g0xx_hal_dma.o	844	-	-	-
stm32g0xx_hal_flash.o	0	-	-	-
stm32g0xx_hal_flash_ex.	0	-	-	-
stm32g0xx_hal_gpio.o	684	-	-	-
stm32g0xx_hal_i2c.o	1598	-	-	-
stm32g0xx_hal_lptim.o	296	-	-	-
stm32g0xx_hal_rcc.o	40	-	-	-
stm32g0xx_hal_rcc_ex.o	548	-	-	-
stm32g0xx_hal_spi.o	2154	-	-	-
stm32g0xx_hal_tim.o	668	-	-	-
stm32g0xx_ll_dma.o	116	-	-	-
stm32g0xx_ll_ucpd.o	138	-	-	-
stm32g0xx_ll_utils.o	40	-	-	-
Middlewares	-	-	-	-
ST	-	-	-	-
STM32_USBPD_Library	-	-	-	-
Core	-	-	-	-
USBPD_CORE_CM0PLUS_PD3_IAR.a	29188	-	18	44
usbpd_dmp_core.a.o	1008	-	107	268
usbpd_trace.o	512	-	-	1040
Devices	-	-	-	-
STM32G0xx	-	-	-	-
usbpd_bsp_trace.o	988	-	-	8
usbpd_cad_hw_if.o	1992	-	-	32
usbpd_bsp_hw.o	396	-	-	-
usbpd_hw_if_it.o	320	-	-	1
usbpd_phy.o	348	-	-	16
usbpd_phy_hw_if.o	1172	-	-	128
usbpd_pwr_hw_if.o	196	-	-	-
usbpd_timersserver.o	580	-	-	-
Third_Party	-	-	-	-
FreeRTOS	-	-	-	-
cmsis_os.o	304	-	-	-

Table 24. UCPD demonstration memory footprint (continued)

Module	ro code	rw code	ro data	rw data
Portable	-	-	-	-
heap_4.o	392	-	-	9024
port.o	228	-	1	4
portasm.o	124	-	-	-
list.o	150	-	-	-
queue.o	1216	-	-	-
tasks.o	1540	-	-	280
Others	3306	-	83	2968
Total	80768	0	14256	18137

7 Acronyms

Table 25. Table of acronyms

Acronym	
CC	Configuration channel
USB PD	USB Power Delivery
DRP	Dual Role Port

8 Appendix

8.1 Module detail 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 used this part to handle input (user button) & output (display) to interact with the end-user.
- Functional behavior: execution function(s)

8.1.1 Module control

A module is described by a simple structure named `K_ModuleItem_Typedef`. This structure provides a unique ID, initialization and de-initialization function, a check resource application function, and main function as listed below:

Table 26. K_ModuleItem_Typedef structure description

Field	Description
kModuleId	Unique ID module. This has to be defined inside <code>MODULES_INFO</code> enumeration in <code>k_config.h</code> include file.
kModulePreExec	Function pointer used to initialize low layer driver, allocate memory or to execute any specific action before the module execution. This function is optional.
kModuleExec	Function pointer used to start module main application. In current kernel architecture, this function is nearly used to call <code>kMenuExecute</code> function that executes module application based on menu structure (see below). This function is mandatory.
kModulePostExec	Function pointer used to de-initialize low layer driver, 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 could be used for instance to check if specific resource file is present on SD card. This function is optional.

8.1.2 Module menu description and Graphical interface

Module menu is used to described module architecture and display.

tMenu structure

This structure is the main entry point used by function `kMenuExecute`, to display the module Man Machine Interface and execute the functionalities.

Table 27. tMenu structure description

Field	Description
pszTitle	String character which contains Menu title to display.
pslItems	Pointer on the menu description tMenuItem item (see below).
nItems	Number of entry inside above menu item data pointer.
nType	Menu type. This parameter allows 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.

tMenuItem structure

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

Table 28. tMenuItem structure description

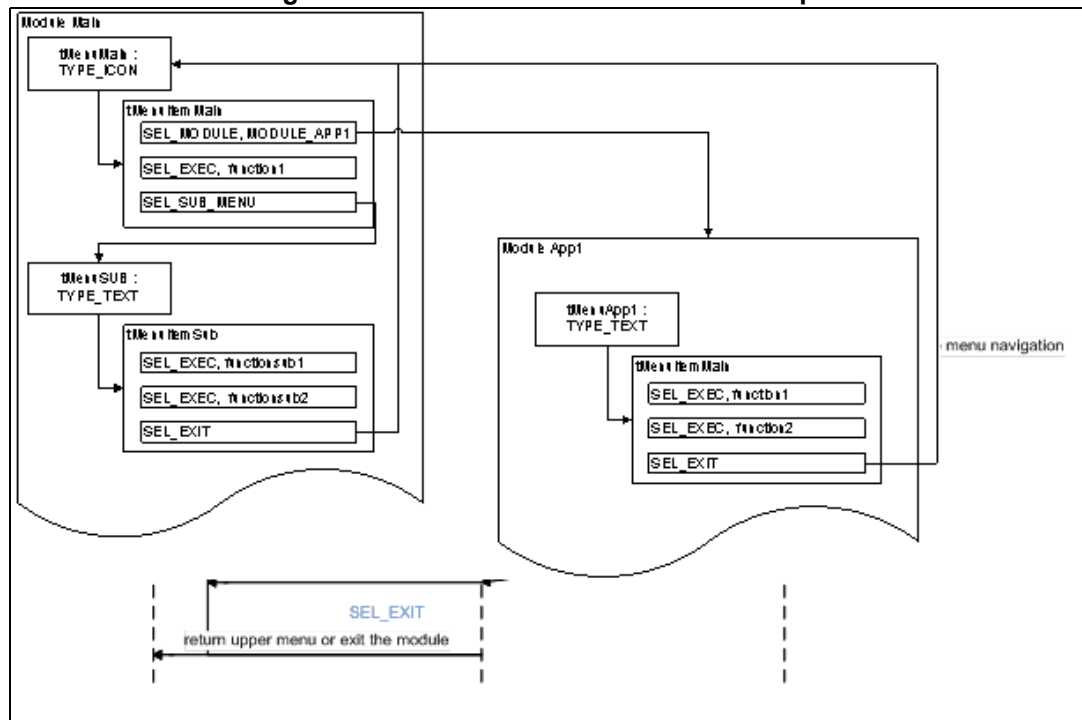
Field	Description
pszTitle	String character which contains item title to display.
x, y	In case menu is of icon, those numbers provide icon position on screen
SelType	Item selection type. This field is one of following defined values: SEL_EXEC: directly execute a functionality SEL_SUBMENU: select a sub-menu SEL_EXIT: exit the current menu and return to previous menu or module SEL_MODULE: execute 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 tMenu type
plconPath	Character string which contains icon name and path to be displayed
plconSelectedPath	Character string which contains alternative icon name and path to be displayed when item is selected by user.

Menu example

The [Figure 75](#) below shows an example of a basic 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 SEL_MODULE, MODULE_APP1 property, execute function1 or display a sub-menu with SEL_SUB_MENU property.
- This submenu may execute functionsub1, functionsub2 or go back on Main Menu, through SEL_EXIT.
- The module App1 has a TYPE_TEXT context with 2 embedded functionalities function1, function2.

Figure 75. Module menu architecture example



8.1.3 Functionality

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

1. Menu event that is executed when module is selected. Called function must respect the following prototype:
void functionExec(void)
This function is linked to module context by function pointer pfExecFunc field of tMenuItem structure (see above description)
On function exit, the kernel returns to the previous menu state.
2. In many cases, functionality could be stopped by a user event. As a consequence, kernel offers capability to return all possible events through a callback function with the following prototype:
void functionSel(uint8_t sel)

This function is linked to module context by function pointer pfActionFunc field of tMenuItem structure (see above description)

8.2 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 new module unique ID in k_config.h file
2. k_ModuleAdd() function must be called in KDemo_Initialization(), with module unique ID as parameter.
3. Modify main module item description table, adding dedicated line matching with new module description (MainMenuItems in main_app.c in our demonstration firmware).
4. Then any possible resource file into SD card as explained above.

References

USB-IF. (2017). Universal Serial Bus Power Delivery Specification rev 3.0. USB-IF.

Revision history

Table 29. Document revision history

Date	Revision	Changes
7-Nov-2018	1	Initial version
12-Apr-2019	2	Updated: <ul style="list-style-type: none">– Introduction and STM32CubeG0 main features chapters aligned to STM32Cube standards– FRS replaced by UNCK in Figure 58, Figure 60 and Figure 61– Clarified Figure 73

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