User manual

STM32CubeL4 STM32L4P5G-DK demonstration firmware

Introduction

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

STM32Cube includes:

- A set of user-friendly software development tools to cover project development from the conception to the realization, among which:
  - STM32CubeMX, a graphical software configuration tool that allows the automatic generation of C initialization code using graphical wizards
  - STM32CubeIDE, an all-in-one development tool with peripheral configuration, code generation, code compilation, and debug features
  - STM32CubeProgrammer (STM32CubeProg), a programming tool available in graphical and command-line versions
  - STM32CubeMonitor (STM32CubeMonitor, STM32CubeMonPwr, STM32CubeMonRF, STM32CubeMonUCPD) powerful monitoring tools to fine-tune the behavior and performance of STM32 applications in real-time
- STM32Cube MCU & MPU Packages, comprehensive embedded-software platforms specific to each microcontroller and microprocessor series (such as STM32CubeL4 for the STM32L4+ Series), which include:
  - STM32Cube hardware abstraction layer (HAL), ensuring maximized portability across the STM32 portfolio
  - STM32Cube low-layer APIs, ensuring the best performance and footprints with a high degree of user control over the HW
  - A consistent set of middleware components such as FAT file system, RTOS, USB Host and Device, TCP/IP, Touch library, and Graphics
  - All embedded software utilities with full sets of peripheral and applicative examples
- STM32Cube Expansion Packages, which contain embedded software components that complement the functionalities of the STM32Cube MCU & MPU Packages with:
  - Middleware extensions and applicative layers
  - Examples running on some specific STMicroelectronics development boards

The STM32L4P5G-DK watermark demonstration described in this document runs on a platform built around the STM32Cube HAL, BSP, FAT file system and USB Device middle-ware components as well as PNG file encoding/decoding libraries.

With external Octo-SPI PSRAM, Octo-SPI NOR Flash memories and on-board eMMC, an STLINK-V3 debugger/programmer and STM32L4P5AG microcontroller, this platform is the ideal environment to evaluate STM32L4+ external memory access capabilities.

The STM32L4P5G-DK demonstration firmware architecture is defined to illustrate that the demonstration core is an independent central component. This core is usable with several third party firmware libraries through a series of abstraction layers inserted between the core and the modules, and libraries running around it.

The STM32L4P5G-DK watermark demonstration supports the STM32L4P5xx microcontroller devices and runs on the STM32L4P5G-DK board.

1 General information

This document applies to the STM32 Arm®-based microprocessor devices.

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

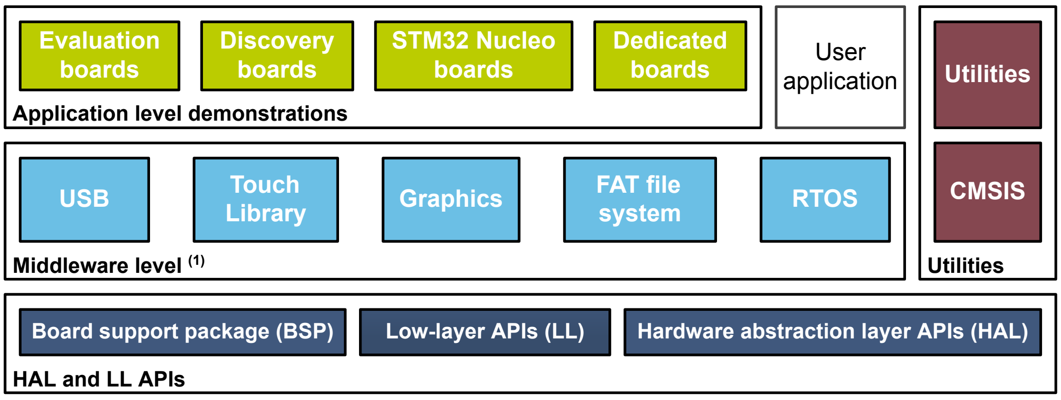
2 STM32Cube overview

The STM32Cube initiative is implemented by STMicroelectronics to reduce development efforts, time and cost. The STM32CubeTM covers the whole of the STM32 portfolio. The STM32Cube block diagram is illustrated in Figure 1.

STM32Cube Version 1.x includes:

- The STM32CubeMX: a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered for each of the Series (such as STM32CubeL4 for STM32L4 and STM32L4+ Series) which includes:
  – The STM32CubeL4 HAL: an STM32 abstraction layer embedded software, ensuring maximized portability across STM32 portfolio
  – A consistent set of middle-ware components such as RTOS, USB, FatFS, Graphics
  – All embedded software utilities coming with a full set of examples.

Figure 1. STM32Cube block diagram



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

3 Getting started with the watermark demonstration

3.1 Hardware requirements

The following hardware parts are required to run the watermark demonstration:

- STM32L4P5G-DK board (see Figure 2). Refer to *Discovery kit with STM32L4P5AG MCU* user manual (UM2651) for Discovery board description.
- One “USB type A to micro-B” cable to power the STM32L4P5G-DK board from the USB ST-LINK (micro-B USB connector CN11)
- One “USB type A to micro-B” cable to connect a PC to the USB OTG connector of the STM32L4P5G-DK board (USB OTG connector CN7)

The STM32L4P5G-DK board provides the user easy access to STM32L4+ Series external memory in order to understand the interface. It offers everything required for beginners and experienced users to quickly and easily get started and develop applications.

3.2 Hardware configuration to run the demonstration firmware

Table 1 lists the jumper configuration to run the demonstration firmware on the STM32L4P5G-DK board.

Table 1. **Jumpers configuration**

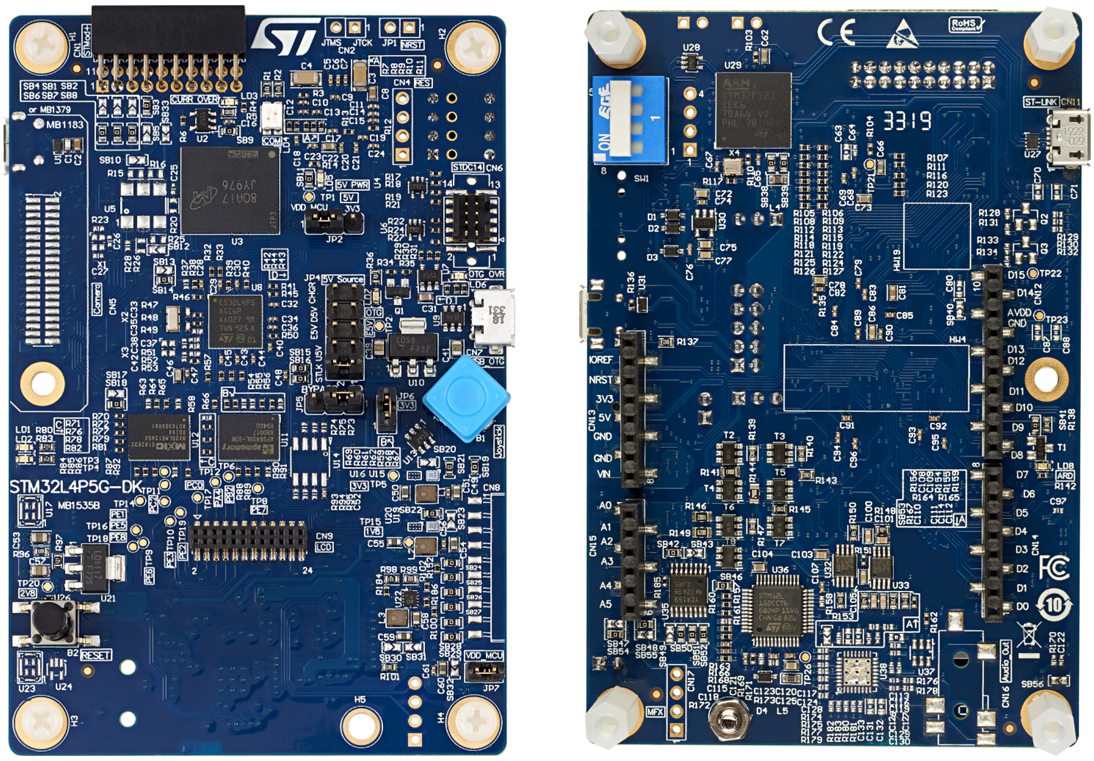
Jumper/connector number	Position (note)
JP2	1-2 (VDD_MCU)
JP4	STLK
JP5	1-2 (IDD)
JP6	CLOSED
JP7	CLOSED

Note: *Position 1 corresponds to jumper side with a dot marking.*
SW1 switch must be set to ON.
Refer to *Discovery kit with STM32L4P5AG MCU* user manual (UM2651) for complete jumper settings description.

Figure 2. STM32L4P5G-DK board

STM32L4P5G-DK top view

STM32L4P5G-DK bottom view



4 Demonstration firmware package

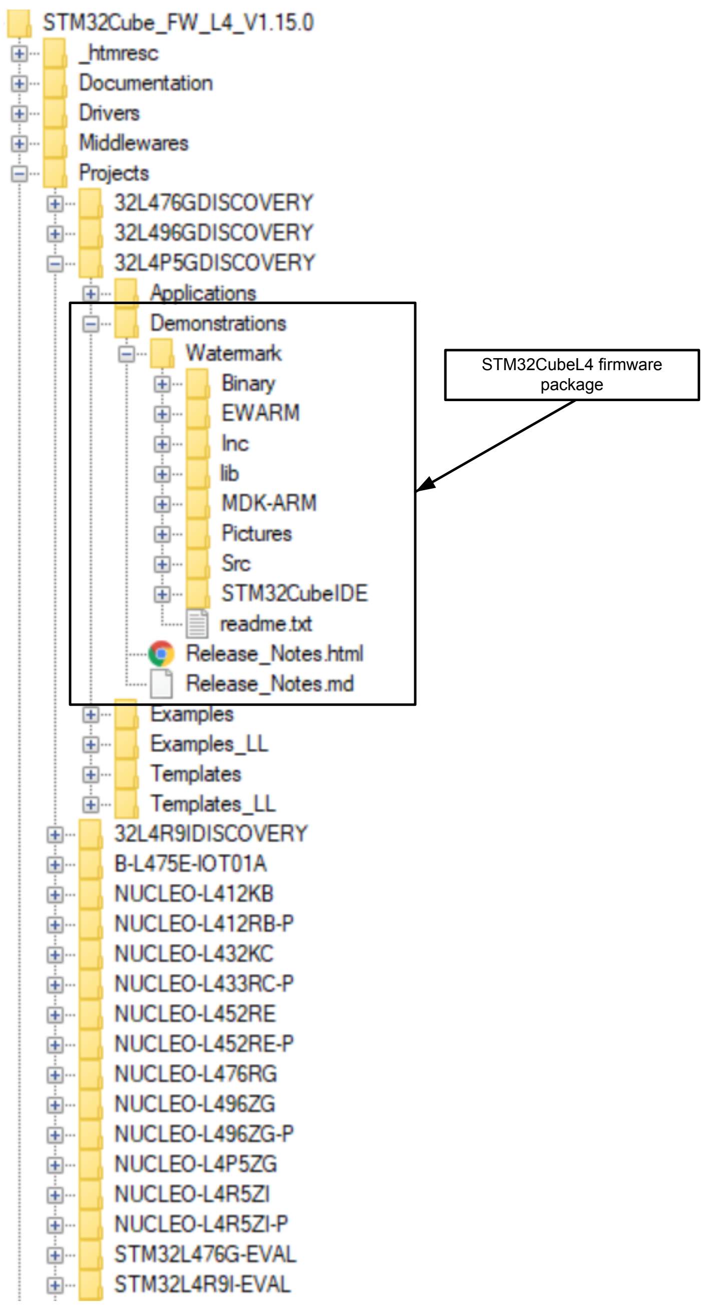
4.1 Demonstration repository

The STM32CubeL4 watermark demonstration firmware for STM32L4P5G-DK board is provided with the STM32CubeL4 firmware package as shown in Figure 3.

The watermark demonstration sources are located in the project folder of the STM32CubeL4 package for each of the supported boards. The sources and other material are divided into the six following groups:

- Binary: this folder contains the readme.txt and refers to the location on the official STMicroelectronics web page from where the demonstration binary file in hex format is available for download.
- Inc: contains all the watermark demonstration header files.
- lib: contains all the libraries files needed to encode/decode PNG files.
- Pictures: PNG format logos and images samples. Users are free to use any other logo and image files as long as the image fits the required sizes detailed in the readme.txt file.
- Src: contains all the watermark demonstration source files.
- EWARM, MDK-ARM, STM32CubeIDE: respectively contain the configuration files for IAR™, Keil® and GCC-based STM32CubeIDE development tool-chains.

Figure 3. **Folder structure**



4.2 Watermark demonstration architecture overview

The STM32CubeL4 watermark demonstration firmware for STM32L4P5G-DK board is composed of a **main** function based on a set of firmware and hardware services. These services are offered by the STM32Cube middle-ware, the evaluation board drivers and a set of libraries used for image decoding/encoding. Each module is subsequently reused separately as a standalone application. The full set of modules is managed by the **main** function which provides access to all common resources and facilitates the addition of new modules as shown in Figure 4 below.

In a nutshell, the software scans in an infinite loop whether non-marked PNG files are provided by the user and watermarks them with the logo PNG file.

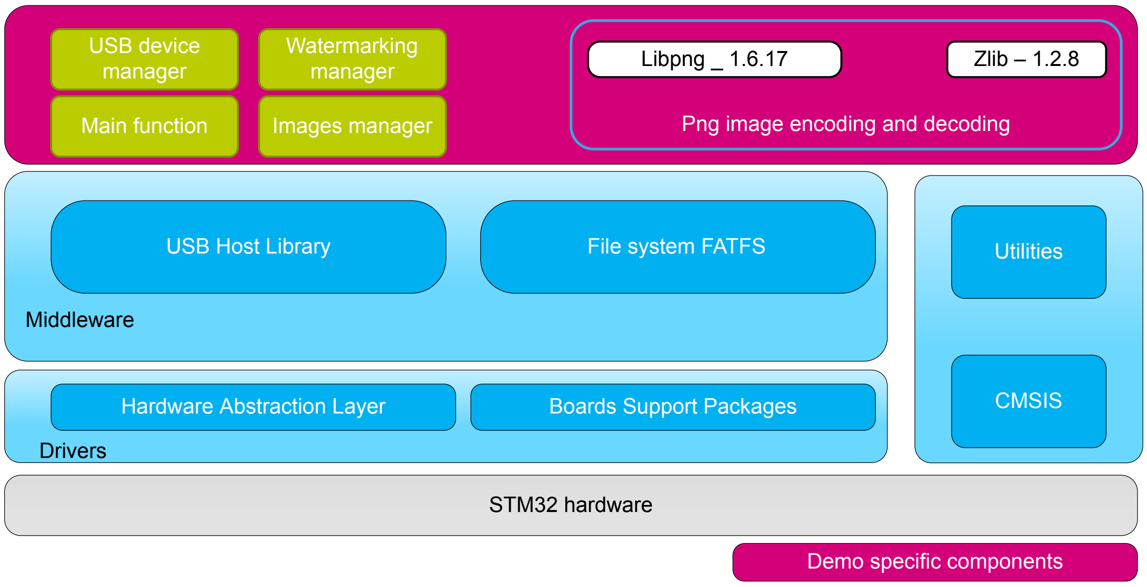
In other words, new files are created and renamed as follows;

The source PNG files is blended with the logo PNG to give the source_logo PNG file.

To achieve the watermarking operation:

1. A FAT file system (FatFS) is mounted on the eMMC.
2. The files are shared with the PC through the USB device feature (read/write operations).
3. The external PSRAM AP-memory accessible through the Octo-SPI is used to store the raw (decoded) image buffer.
4. The full software libPNG library is used to decode/encode PNG files.
5. The DMA2D peripheral is used to blend logo into images.

Figure 4. Demonstration architecture overview



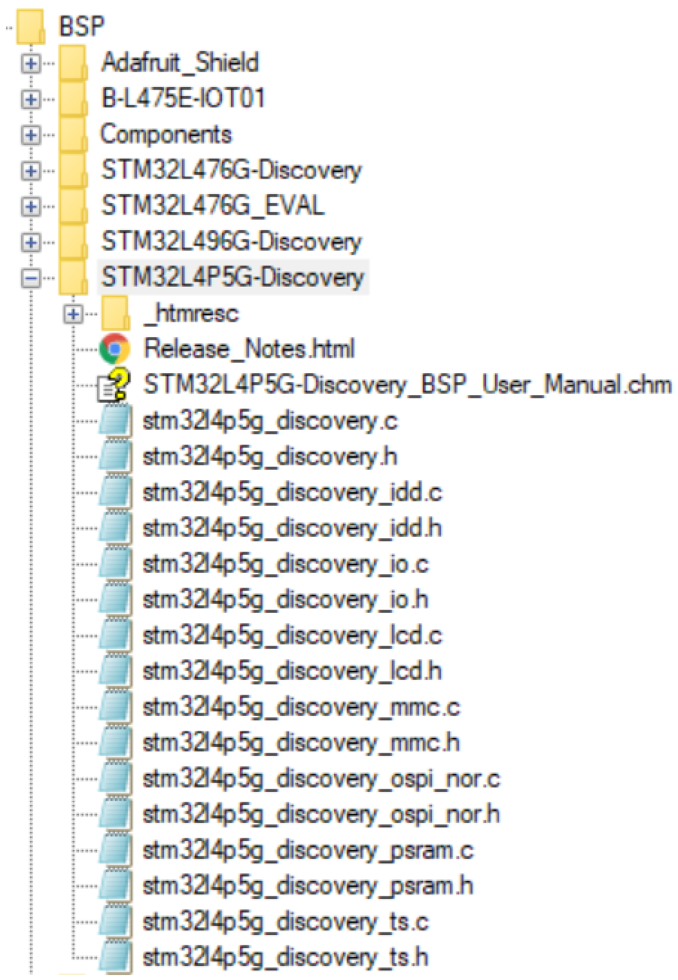
4.3 STM32L4P5G-DK board BSP

Board drivers are available in the `stm32l4p5g_discovery_XXX.c/.h` files as illustrated in Figure 5 to provide access to the on board services such as LEDs, buttons, power consumption measurements, external PSRAM and NOR flash memories, eMMC card, touch screen and LCD when the MB1609 daughter board is connected.

Components present on the STM32L4P5G-DK board are controlled by dedicated BSP drivers. These are:

- the IO expanders in `stm32l4p5g_discovery_io.c/.h`
- the 64-Mbit AP-Memory Octo-SPI PSRAM in `stm32l4p5g_discovery_psram.c/.h`
- the 512-Mbit Macronix MX25LM51245G Octo-SPI NOR flash memory in `stm32l4p5g_discovery_ospi_nor.c/.h`
- the eMMC card in `stm32l4p5g_discovery_mmc.c/.h`
- the built-in Idd circuitry for MCU current consumption measurement in `stm32l4p5g_discovery_idd.c/.h`
- the 30-mm diameter 240x240 MiP 64-color LCD panel with resistive touchscreen provided by the optional MB1609 daughter board in `stm32l4p5g_discovery_lcd.c/.h` and `stm32l4p5g_discovery_ts.c/.h`

Figure 5. **Discovery BSP structure**



```
BSP
  Adafruit_Shield
  B-L475E-IOT01
  Components
  STM32L476G-Discovery
  STM32L476G_EVAL
  STM32L496G-Discovery
  STM32L4P5G-Discovery
    _htmresc
    Release_Notes.html
    STM32L4P5G-Discovery_BSP_User_Manual.chm
    stm32l4p5g_discovery.c
    stm32l4p5g_discovery.h
    stm32l4p5g_discovery_idd.c
    stm32l4p5g_discovery_idd.h
    stm32l4p5g_discovery_io.c
    stm32l4p5g_discovery_io.h
    stm32l4p5g_discovery_lcd.c
    stm32l4p5g_discovery_lcd.h
    stm32l4p5g_discovery_mmc.c
    stm32l4p5g_discovery_mmc.h
    stm32l4p5g_discovery_ospi_nor.c
    stm32l4p5g_discovery_ospi_nor.h
    stm32l4p5g_discovery_psram.c
    stm32l4p5g_discovery_psram.h
    stm32l4p5g_discovery_ts.c
    stm32l4p5g_discovery_ts.h
```

5 Watermark demonstration functional description

5.1 Demonstration overview

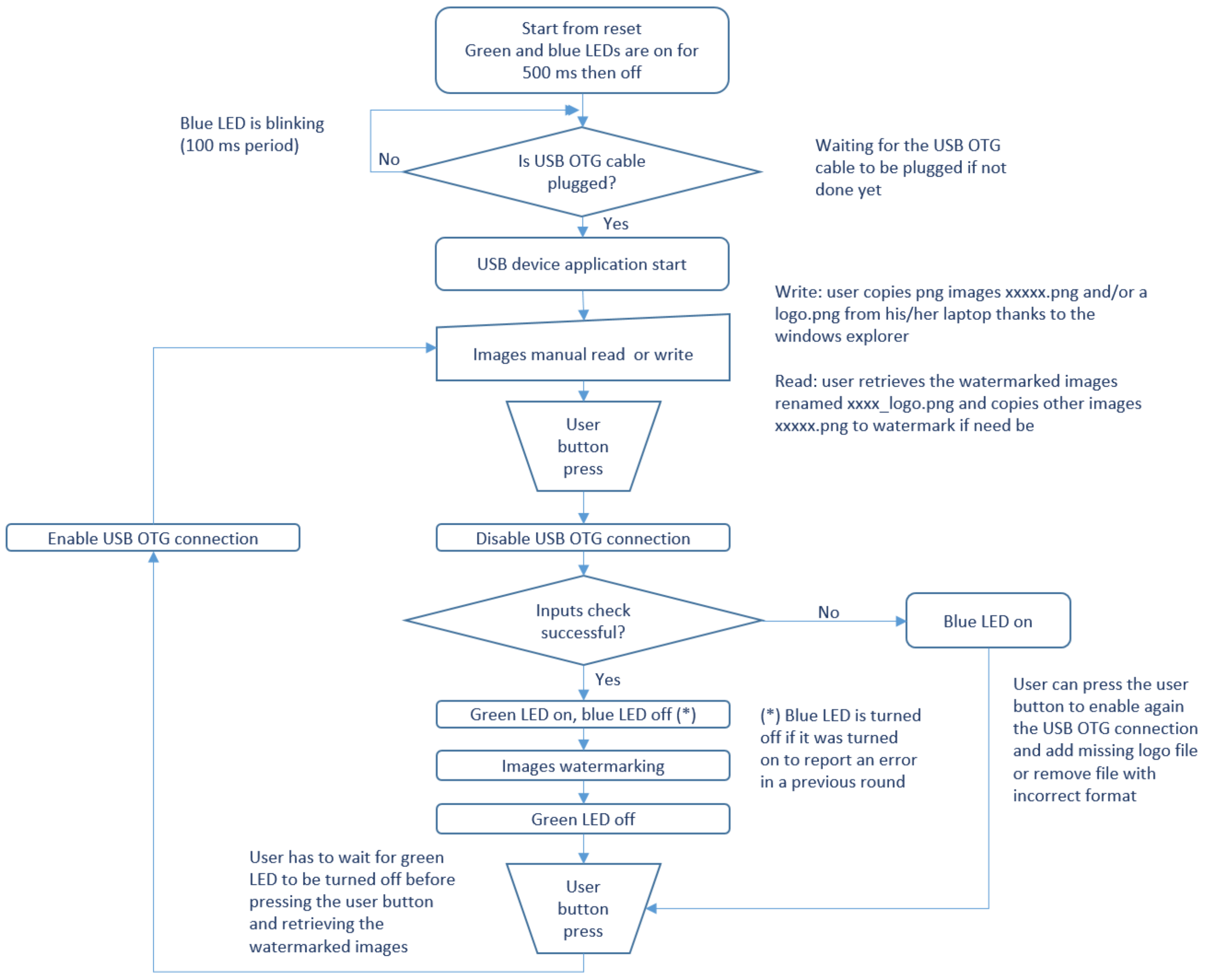
Broadly speaking, the watermark demonstration is defined as an infinite software loop. When PNG format images are found, they are automatically watermarked by the "logo.png" file and made available to the user.

Logo, plain, or watermarked images are read or written by the user through the USB OTG.

No real-time operating system is used.

Figure 6 provides the state machine used by the demonstration software.

Figure 6. Software state machine


Start from reset
Green and blue LEDs are on for 500 ms then off
Blue LED is blinking (100 ms period)
No
Is USB OTG cable plugged?
Waiting for the USB OTG cable to be plugged if not done yet
Yes
USB device application start
Write: user copies png images xxxxx.png and/or a logo.png from his/her laptop thanks to the windows explorer
Images manual read or write
Read: user retrieves the watermarked images renamed xxxx_logo.png and copies other images xxxxx.png to watermark if need be
User button press
Enable USB OTG connection
Disable USB OTG connection
Inputs check successful?
No
Blue LED on
Yes
Green LED on, blue LED off (*)
(*) Blue LED is turned off if it was turned on to report an error in a previous round
User can press the user button to enable again the USB OTG connection and add missing logo file or remove file with incorrect format
Images watermarking
Green LED off
User has to wait for green LED to be turned off before pressing the user button and retrieving the watermarked images
User button press

5.2 Demonstration start-up

When the application starts, both LED1 (green) and LED2 (blue) are turned on for half a second to indicate the software is up and running.

The software checks that a USB cable is connected to the USB OTG CN7 connector. If it is not the case, blue LED2 flashes with a 100 ms period until a cable is plugged.

After the USB OTG cable is connected, the device is automatically detected and the USB device application starts.

5.3 Logo and image files writing

Using a Windows® explorer, the user copies the files onto the eMMC embedded on the STM32L4P5G-DK board. This copy is done seamlessly as the eMMC appears as a simple external disk available on the PC.

Unless already copied, the user is expected to copy a "logo.png" file to be used to watermark the other PNG files that are copied at the same time.

The PNG size limitation is due to memory constraint and the four channel decoding (32 bpp) lead to the following size requirements:

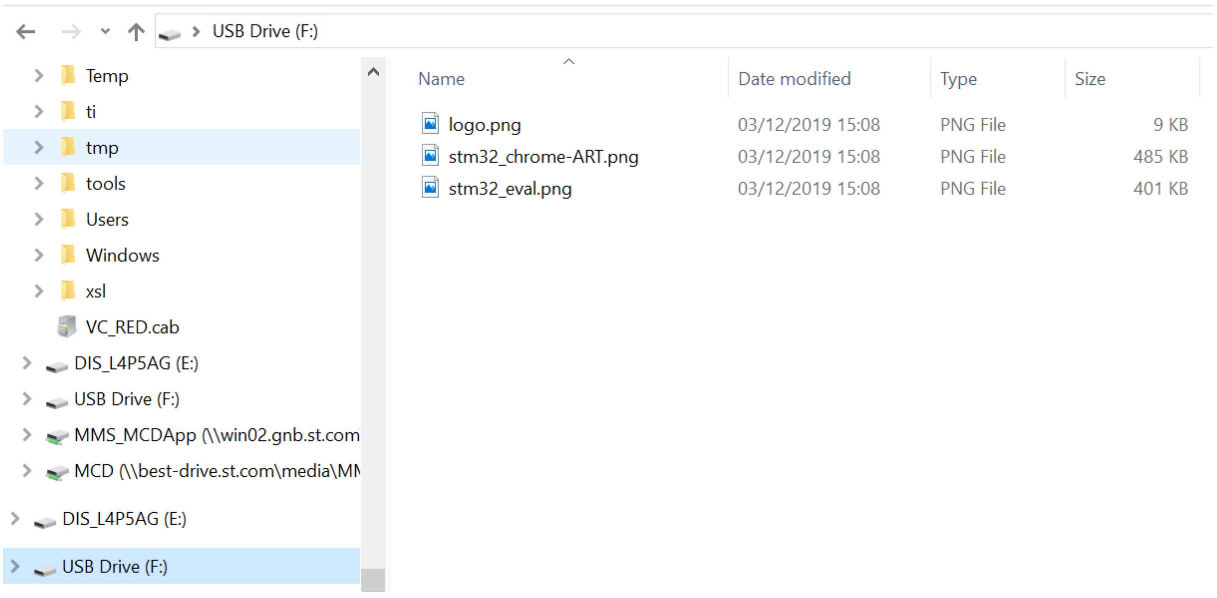
1. The size of the "logo.png" picture is limited to 16 000 pixels (for example a picture size of 150*100 pixels)
2. The PNG image size is limited to 2,000,000 pixels (for example, a picture size of 1500*1000 pixels is adequate for the demonstration).

The following "logo.png" file, illustrated in Figure 7, is provided as an example in the pictures demonstration folder but any other PNG with the proper resolution and renamed "logo.png" can be used for the watermarking operation.

Figure 7. Sample logo.png

Figure 8 below is a snapshot of theSTM32L4P5G-DK board picture folder content when seen as an external disk.

Figure 8. Logo and png files copy



Name	Date modified	Type	Size
logo.png	03/12/2019 15:08	PNG File	9 KB
stm32_chrome-ART.png	03/12/2019 15:08	PNG File	485 KB
stm32_eval.png	03/12/2019 15:08	PNG File	401 KB

5.4 Watermarking operation

To initiate the watermarking operation, the user must press the "User" button to indicate the write operation is completed. The USB OTG connection is disabled and the FatFS is mounted on the eMMC so that the relevant functions detect the logo.png and the PNG image files to watermark.

Unless already watermarked, each detected PNG image is decoded to the raw (uncoded) format and copied to the external Octo-SPI PSRAM.

Once done, the DMA2D peripheral is used to blend a horizontal banner made with the logo over the raw image. This watermarks the image.

Finally, the watermarked image is copied on the eMMC with a name based on the original image name: the watermarked xxxxx.png is named xxxxx_logo.png.

LED1 (green) is lit as long as the application is working on the eMMC. This comprises file detection, watermarking, and copying back to the eMMC. The whole watermarking operation may last several seconds depending on the number and the size of the files to be processed.

When all detected files have been watermarked, LED1 (green) turns off.

The user then presses the "User" button once again.

This unmounts the FatFS and enables the USB OTG connection again. The watermarked files are now available to the user.

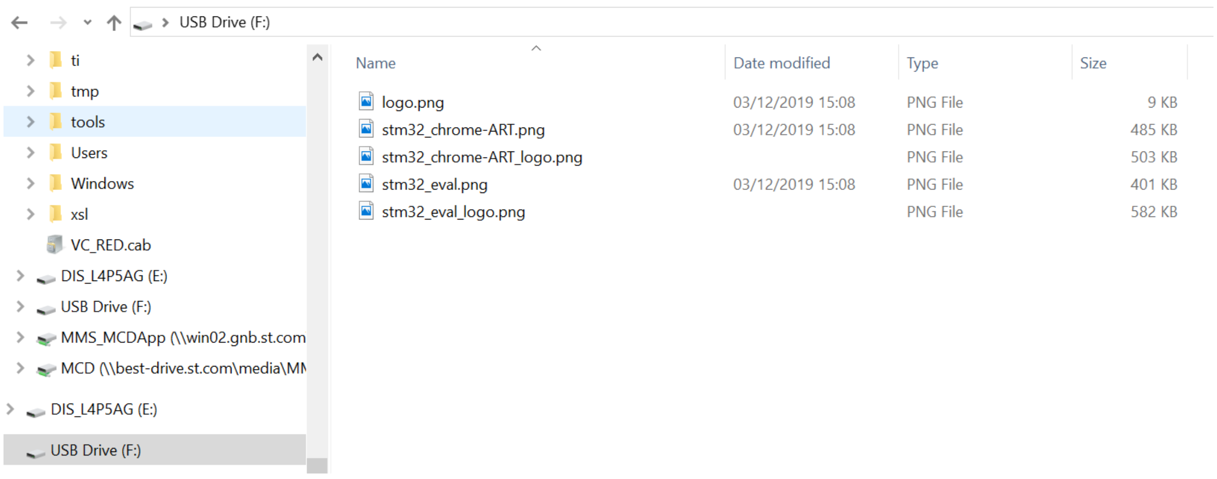
If LED2 (blue) lights up, this means the "logo.png" file is missing, too big, or that one of PNG files is too big.

In that case, the user needs to press the "User" button to add the missing logo file, copy another one, or remove the flagged PNG file.

5.5 Image files reading

Once the "User" button is pressed and the UST OTG connection is enabled again, the watermarked files are visible through a windows explorer as shown in Figure 9.

Figure 9. **Retrieving watermarked files**



USB Drive (F:)

- ti
- tmp
- tools
- Users
- Windows
- xsl
- VC_RED.cab
- DIS_L4P5AG (E:)
- USB Drive (F:)
- MMS_MCDApp (\\win02.gnb.st.com
- MCD (\\best-drive.st.com\media\MM
- DIS_L4P5AG (E:)
- USB Drive (F:)

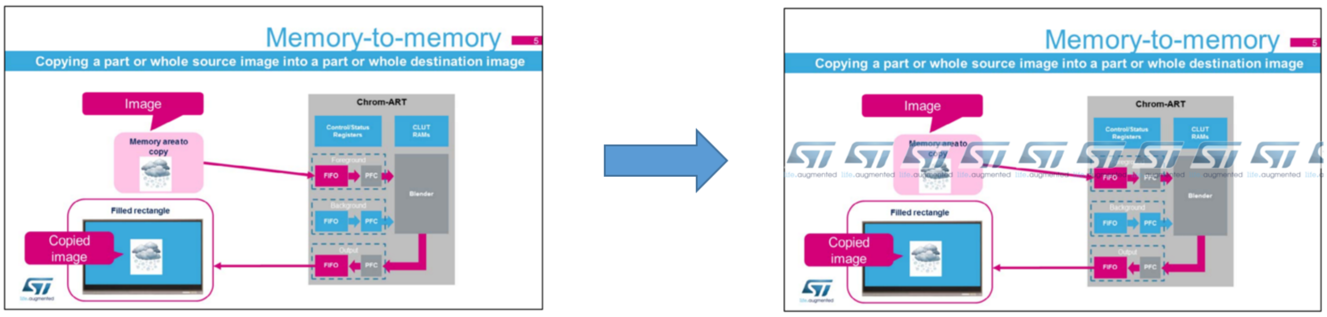
Name	Date modified	Type	Size
logo.png	03/12/2019 15:08	PNG File	9 KB
stm32_chrome-ART.png	03/12/2019 15:08	PNG File	485 KB
stm32_chrome-ART_logo.png		PNG File	503 KB
stm32_eval.png	03/12/2019 15:08	PNG File	401 KB
stm32_eval_logo.png		PNG File	582 KB

The user is now able to retrieve the xxxx_logo.png file, copy other files to watermark, change logo.png, and so on.

Note: *A watermarked file is not watermarked again if it is left on the eMMC.*

Figure 10 illustrates the watermarking operation with the stm32_chrome-ART.png file (this image is extracted from a DMA2D peripheral features presentation).

Figure 10. **Watermarking result illustration**



To restart another round of watermarking, the user presses the "User" button again once and LED1 (green) turns on to indicate the processing has re-started.

6 Demonstration firmware settings

6.1 Clock control

The following clock configurations are used in the demonstration firmware:

- SYSCLK: 120MHz (PLL) derived from 48MHz MSI (RUN voltage range 1)
- MSI (48 MHz) as PLL source clock and USB clock source
- PLL main output at 120Mhz
- SDMMC clock frequency at 25MHz.

6.2 Peripherals

The peripherals used in the demonstration firmware are listed in Table 2.

Table 2. **Peripherals list**

Used peripherals	Application/module
CORTEX	NVIC services
DMA2D	Image transfer on LCD internal buffer
EXTI	Push-button
FLASH	System settings
GPIO	All applications
I2C	IO expander usage on board
OSPI	Octo-SPI PSRAM read/write
RCC	System application and BSP driver clocking
SD	All applications with EMMC storage unit accesses
USB	USB OTG application

6.3 Interrupts / Wakeup pins

The interrupts used in the demonstration firmware are listed in Table 3.

Table 3. **Interrupts list**

Interrupts	Application/module	Priority, SubPriority (highest=0,0)
DMA2D_IRQn	DMA2D interrupt request	0,0
EXTI Line 13	User button interruption	15,0
OCTOSPI1_IRQn	OctoSPI interrupt request	15,0
OTG_FS_IRQn	USB Device application	7,0
SDMMC1_IRQn	All applications with eMMC storage	5,0
SysTick	CortexM4 System Timer for OS Tick	15, 0

6.4 System memory configuration

The system memory areas used in the demonstration firmware are listed in Table 4.

Table 4. **Memories areas**

Memory	Start Address	Application
Internal Flash	0x80000000	Demonstration firmware run code and constants
Internal SRAM1	0x20000000	Demonstration firmware data (128 Kbytes)
Internal SRAM2	0x10000000	Demonstration firmware data (64 Kbytes)
Internal SRAM3	0x20030000	Demonstration firmware data (128 Kbytes)
External OSPI	0x90000000	External Octo-SPI PSRAM where decoded png files are stored and watermarked

6.5 Programming firmware application

First of all, install the ST-LINK/V3 driver available on *www.st.com*.

There are two ways of programming the STM32L4P5G-DK board and are described in the two following sections.

6.5.1 Using Binary file

Download the binary STM32CubeDemo_STM32L4P5G-Discovery-MB1535-B01_V1.0.0.hex from *www.st.com* and load the binary using any preferred method in-system programming tool.

The STM32CubeProgrammer v2.1.0 must be used.

6.5.2 Using preconfigured projects

Choose one of the supported tool chains and follow the steps below:

1. Open the application folder: Projects\STM32L4P5G-DK\Demonstrations\Watermark.
2. Chose an IDE project (EWARM for IAR, MDK-ARM for Keil, and STM32CubeIDE for gcc-based compilers).
3. Double click on the project file (for example Project.eww for EWARM).
4. Rebuild all files: go to "Project" and select "Rebuild all".
5. Load the project image: go to "Project" and select "Debug".
6. Run the program: go to "Debug" and select "Go".

7 Demonstration firmware footprints

Table 5 below provides the footprints of the demo (Flash size for the code and read-only memory) and internal SRAM for read/write data.

Table 5. **Demonstrations footprints**

Watermark demonstration	Requested size in Flash (in Kbytes)	Requested size in RAM (in Kbytes)
Read only code memory	156	-
Read only data memory	28	-
Read write data memory	-	320
STM32L4P5G-DK available size	1024	320

8 Storage unit

The STM32Cube demonstration uses an eMMC storage unit for the following purpose:

- to store the logo and the images to watermark
- to store the watermarked images.

The unit is accessible through the standard FatFS I/O operations implemented on the development platform. The unit is automatically mounted when the USB OTG is connected and dismounted again when the USB OTG is disconnected. The goal is to avoid two simultaneous mount set-ups, one for the eMMC access from the Windows® Explorer and one from the application. Such simultaneous accesses to the physical media only leads to conflicts.

Table 6 lists the implemented file system interface functions which deals with the physical storage unit.

Table 6. File system interface functions

Function	Description
disk_initialize	Initialize disk drive
disk_read	Interface function for a logical page read
disk_write	Interface function for a logical page write
disk_status	Interface function for testing if unit is ready
disk_ioctl	Control device dependent features

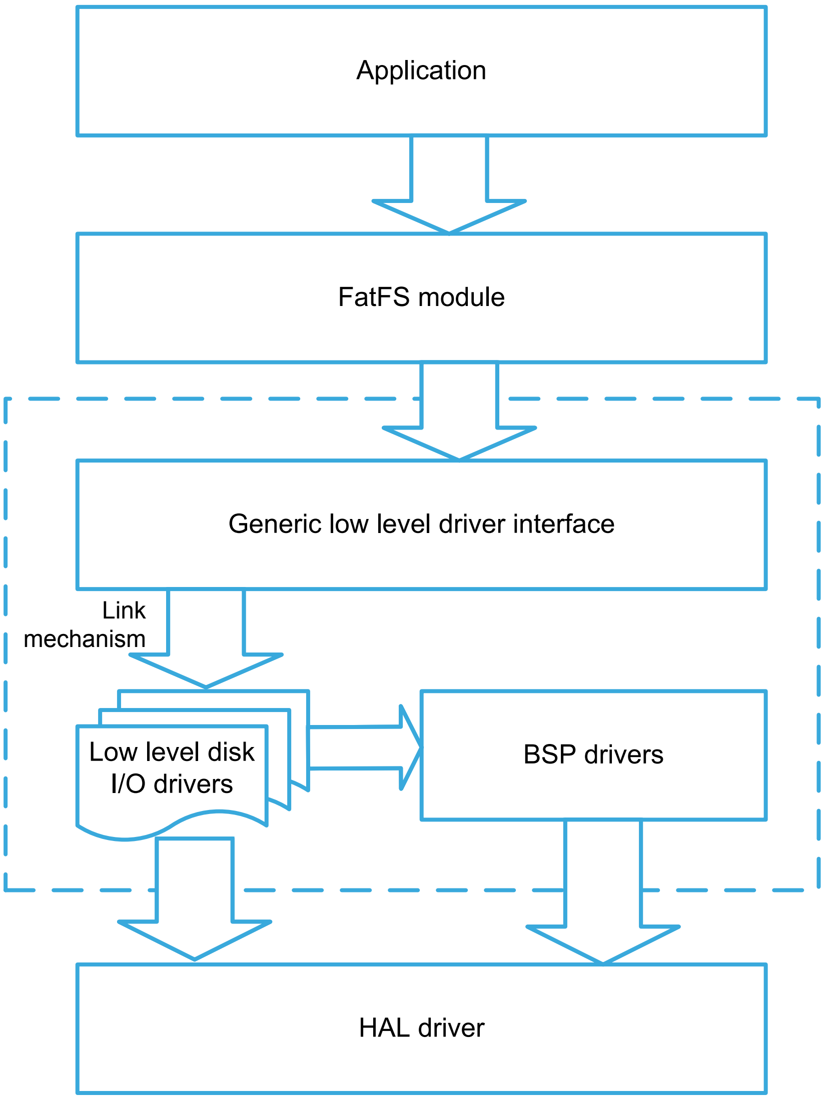
The full list of file system interface API functions set given in Table 7.

Table 7. **File system APIs**

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

For the FatFS file system, the page size is fixed to 512 bytes.
The file system architecture is described in Figure 11.

Figure 11. File System architecture


Application
FatFS module
Generic low level driver interface
Link mechanism
Low level disk I/O drivers
BSP drivers
HAL driver

Revision history

Table 8. Document revision history

Date	Version	Changes
18-Mar-2020	1	Initial release.

Contents

List of figures

List of tables