UM2308 User manual

STM32CubeG0 Nucleo demonstration firmware

Introduction

STM32Cube™ is an STMicroelectronics original initiative to make developers' lives easier by reducing development effort, time and cost. STM32Cube covers the whole STM32 portfolio.

The STM32Cube includes:

- STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per Series (such as STM32CubeG0 for STM32G0 Series)
  - The STM32Cube HAL, STM32 abstraction layer embedded software ensuring maximized portability across the STM32 portfolio
  - The Low-Layer APIs (LL) offering a fast light-weight expert-oriented layer which is closer to the hardware than the HAL. LL APIs are available only for a set of peripherals
  - A consistent set of middleware components such as FAT file system, RTOS, USB PD
  - All embedded software utilities, delivered with a full set of examples

The STM32CubeG0 Nucleo demonstration firmware is built around the STM32Cube hardware abstraction layer (HAL) and low-layer (LL) APIs, and board support package (BSP) components, and uses almost the whole STM32 capability to load and display full color bitmaps from a microSD™ card.

Contents

1 **STM32CubeG0 main features 5**

2 **Getting started with the demonstration 6**

2.1 Hardware requirements 6

2.1.1 NUCLEO-G071RB Nucleo board 6

2.1.2 Adafruit 1.8” TFT shield 7

2.2 Hardware configuration 7

2.2.1 STM32 Nucleo board configuration 7

2.2.2 Assembling the Adafruit shield 8

3 **Demonstration firmware package 9**

3.1 Demonstration repository 9

3.2 STM32 Nucleo board BSP 9

3.2.1 Joystick 11

3.2.2 LCD 11

3.2.3 microSD 11

4 **Functional description of the demonstration 12**

4.1 Programming firmware application 15

5 **FAQs 16**

6 **Revision history 17**

List of tables

Table 1. Bitmap image properties 12
Table 2. Document revision history 17

List of figures

Figure 1. STM32CubeG0 firmware components 5
Figure 2. NUCLEO-G071RB Nucleo board 6
Figure 3. Adafruit 1.8” TFT shield 7
Figure 4. Assembling the Adafruit 1.8” TFT shield. 8
Figure 5. Folder structure. 9
Figure 6. Nucleo BSP architecture. 10
Figure 7. Demonstration application menu on NUCLEO-G071RB 12
Figure 8. Reading the Adafruit shield joystick 13
Figure 9. Demonstration application error messages 14
Figure 10. Demonstration running 14

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 to the STM32G0 Series but also to other STM32 Series.

STM32CubeG0 is fully compatible with the STM32CubeMX code generator that allows to generate initialization code.

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

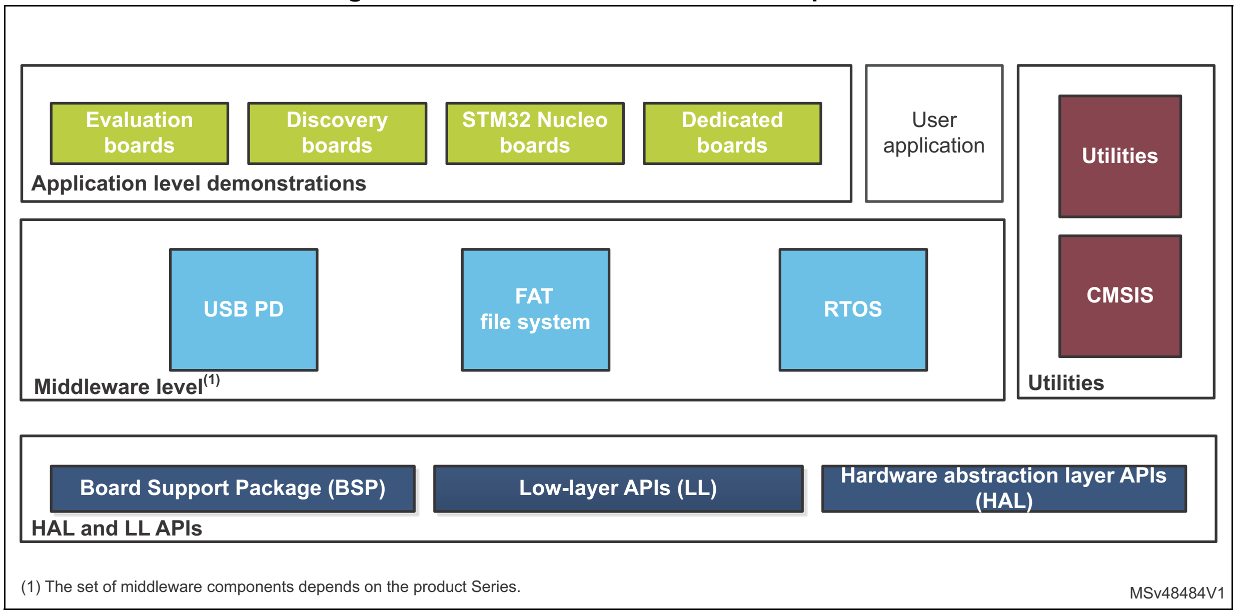
STM32CubeG0 package also contains a set of middleware components with the corresponding examples. They come in free user-friendly license terms:

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

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

The block diagram of STM32Cube is shown in *Figure 1: STM32CubeG0 firmware components*

Figure 1. STM32CubeG0 firmware components



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

2 Getting started with the demonstration

2.1 Hardware requirements

The hardware requirements to start the demonstration application are the following:

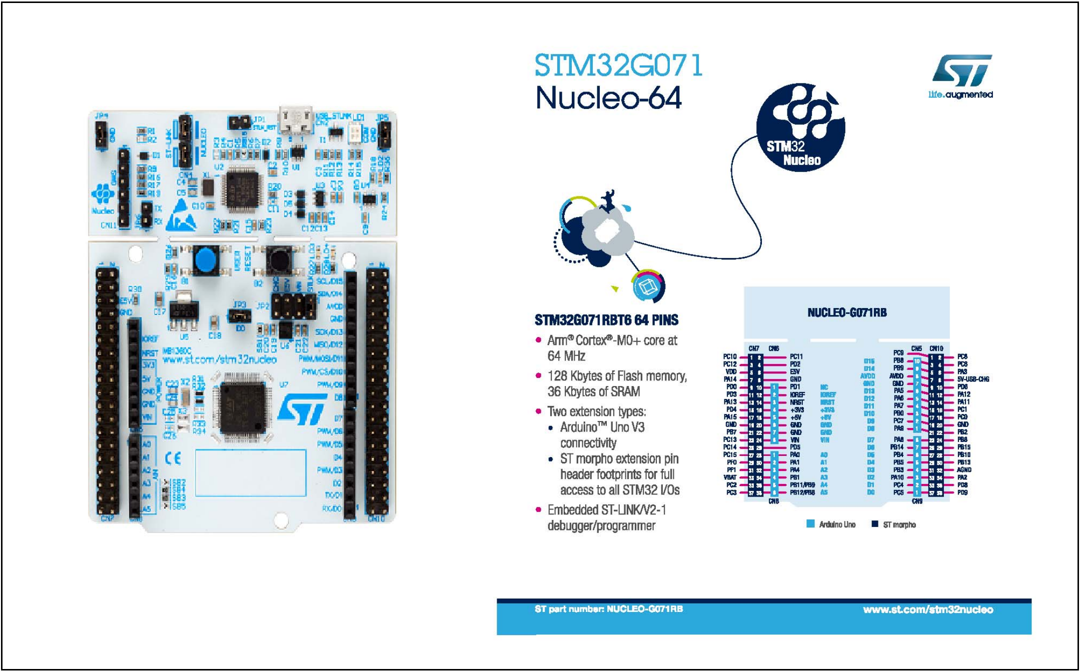
- NUCLEO-G071RB Nucleo board
- Adafruit 1.8" TFT shield with joystick and microSD (reference ID: 802)
- one 'USB Type A to Mini-B' cable to power up the STM32 Nucleo board from the USB ST-LINK (USB connector CN2)
- a standard capacity SD card (SDSC up to 4 Gbytes) or a high capacity SD card (HDSC up to 32 Gbytes).

2.1.1 NUCLEO-G071RB Nucleo board

The STM32 Nucleo board is a low-cost and easy-to-use development kit, to evaluate and start quickly some development with the Arm®[a] 32-bit Cortex®-M microcontrollers of the STM32 Series. The *Figure 2* shows the NUCLEO-G071RB and its package. Before installing and using the product, accept the Evaluation Product License Agreement available at www.st.com/epla.

For more information on the STM32 Nucleo boards visit www.st.com/stm32nucleo.

Figure 2. NUCLEO-G071RB Nucleo board


STM32G071
Nucleo-64
life.augmented
STM32 Nucleo
STM32G071RBT6 64 PINS
Arm® Cortex®-M0+ core at 64 MHz
128 Kbytes of Flash memory, 36 Kbytes of SRAM
Two extension types:
Arduino™ Uno V3 connectivity
ST morpho extension pin header footprints for full access to all STM32 I/Os
Embedded ST-LINK/V2-1 debugger/programmer
NUCLEO-G071RB
CN7
CN6
CN8
CN5
CN10
CN9
Arduino Uno
ST morpho
ST part number: NUCLEO-G071RB
www.st.com/stm32nucleo

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

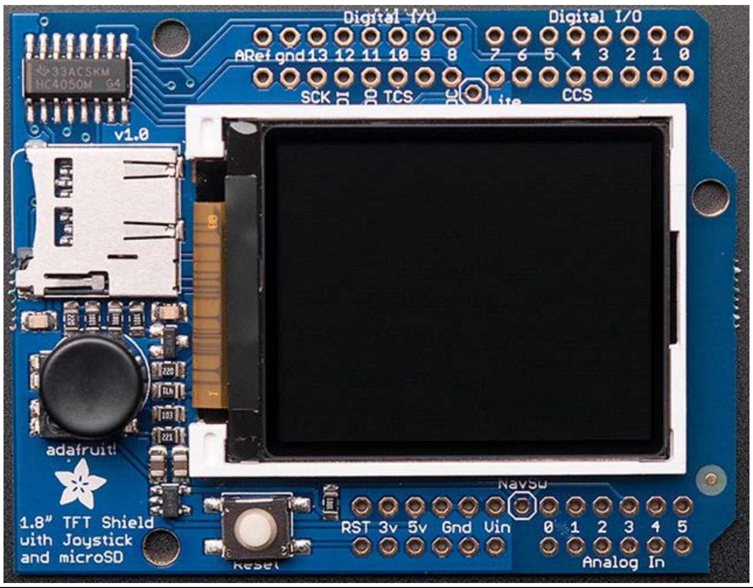
2.1.2 Adafruit 1.8” TFT shield

The STM32 Nucleo board supports Arduino™ connectivity.

This Adafruit 1.8" TFT shield may be found on the Adafruit website (reference ID 802) with the following features (see *Figure 3*):

- one 1.8” 128x160 TFT color display
- one microSD card interface slot
- one 5-way joystick navigation switch (left, right, up, down and select)

Figure 3. Adafruit 1.8” TFT shield



2.2 Hardware configuration

To start using the Adafruit 1.8” TFT shield with the STM32 Nucleo board, follow the recommendations in *Section 2.2.1* in addition to gather the hardware.

2.2.1 STM32 Nucleo board configuration

Check the positions of the jumpers on the STM32 Nucleo board, as follows:

- JP1 OFF
- JP2 ON: PIN1 and PIN2 connected (ST-LINK configuration)
- JP3 (I_{DD}) ON

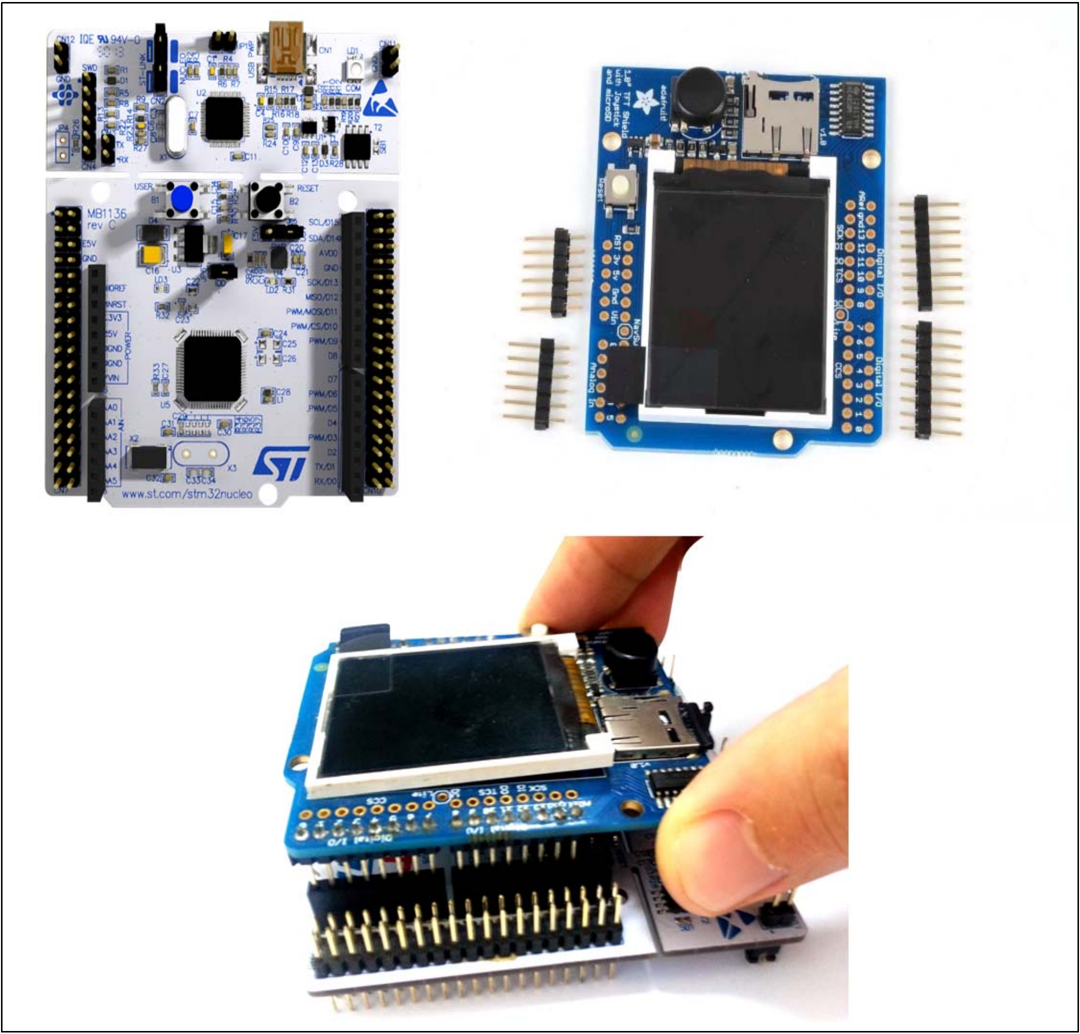
2.2.2 Assembling the Adafruit shield

The Adafruit TFT shield comes with all the surface parts soldered. User can install the headers following the next steps:

- Cut the breakaway header strip into sections, to fit the holes on the edge of the shield: two sections of six pins and two other sections of eight pins are needed.
- To align the header strips for soldering, insert them (long pins down) into the headers of the STM32 Nucleo board using the connectors CN7, CN8, CN10 and CN12.
- Place the shield over the header strips, so that the short pins stick up through the holes.
- Solder on each pin of the header onto the shield PCB to ensure good electrical contact.

The sequence is shown in *Figure 4*.

Figure 4. Assembling the Adafruit 1.8” TFT shield

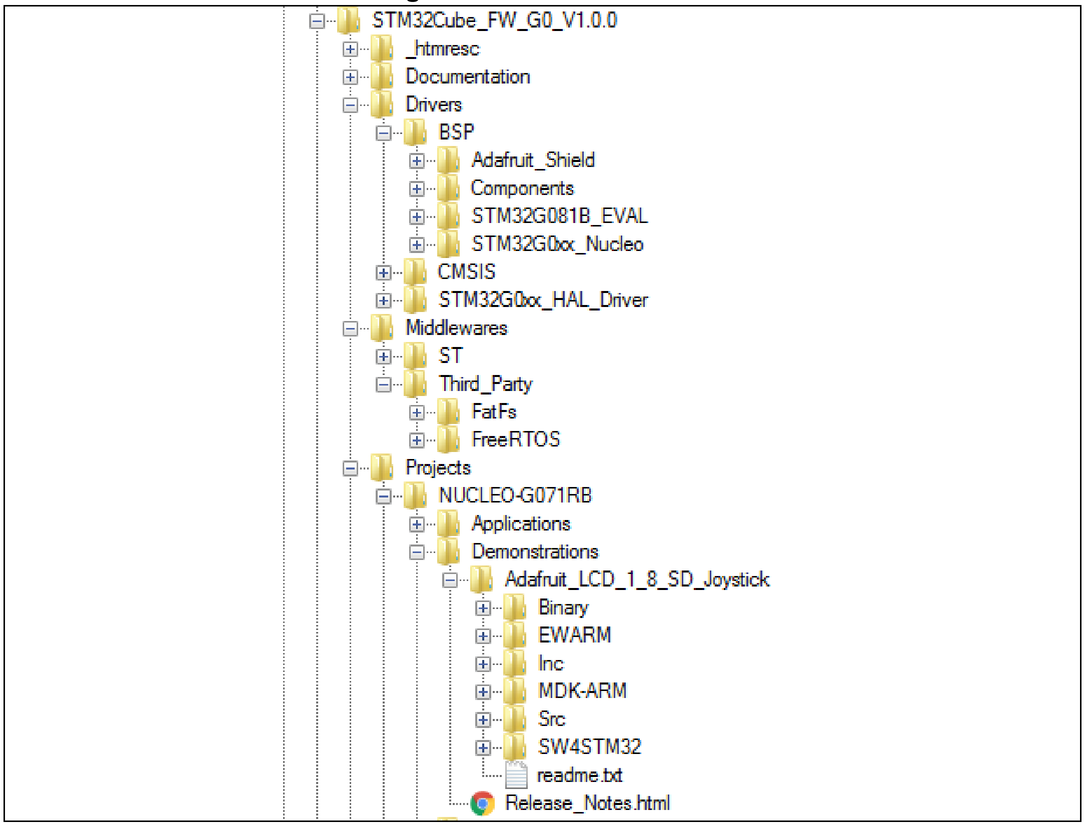


3 Demonstration firmware package

3.1 Demonstration repository

The demonstration *Adafruit_LCD_1_8_SD_Joystick* is provided within the STM32CubeG0 firmware package, as shown in *Figure 5: Folder structure*.

Figure 5. Folder structure



```
STM32Cube_FW_G0_V1.0.0
    _htmresc
    Documentation
    Drivers
        BSP
            Adafruit_Shield
            Components
            STM32G081B_EVAL
            STM32G0xx_Nucleo
        CMSIS
        STM32G0xx_HAL_Driver
    Middlewares
        ST
        Third_Party
            FatFs
            FreeRTOS
    Projects
        NUCLEO-G071RB
            Applications
            Demonstrations
                Adafruit_LCD_1_8_SD_Joystick
                    Binary
                    EWARM
                    Inc
                    MDK-ARM
                    Src
                    SW4STM32
                    readme.txt
                Release_Notes.html
```

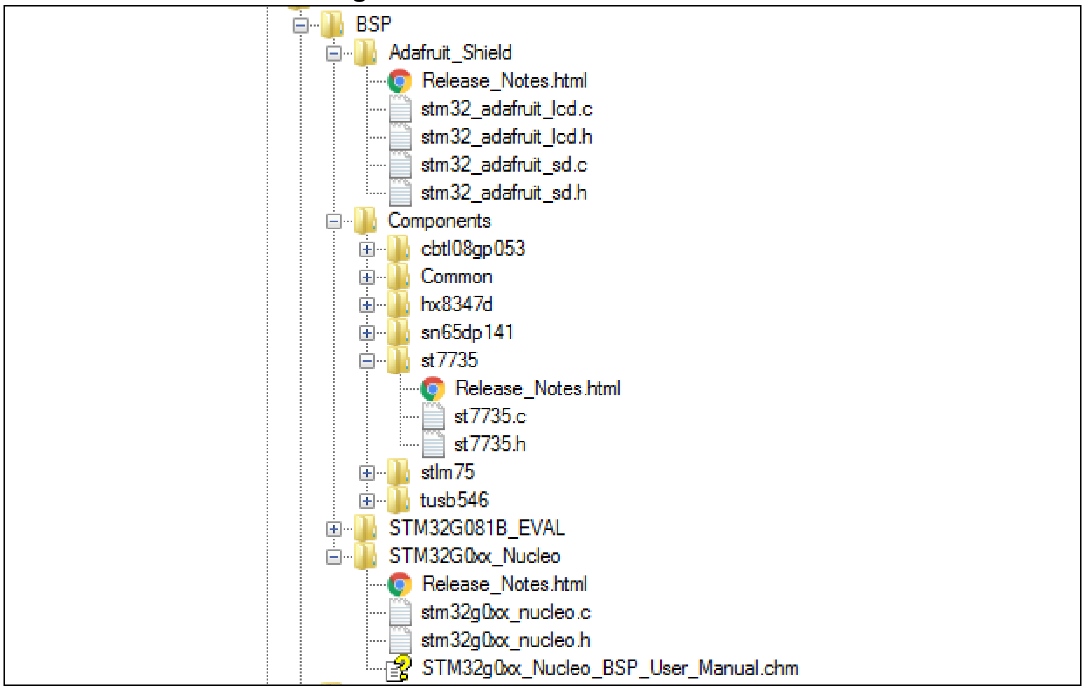
The demonstration sources are located in the project folder of the STM32Cube package for the NUCLEO-G071RB Nucleo board. The sources are divided into five groups, described as follows:

1. Binary: demonstration binary file in Hex format
2. Media: the media files (*.bmp) required to run the demonstration are available under the firmware package Utilities\Media\Pictures folder
3. Inc: contains the demonstration header files
4. Src: contains the demonstration source files
5. Project settings: a folder per toolchain containing the project settings and the linker files

3.2 STM32 Nucleo board BSP

The *Figure 6: Nucleo BSP architecture* show the BSP architecture. For each board, a set of drivers for button, LED and joystick is available within the *_nucleo.c/.h* files, implementing the board capabilities and the bus-link mechanism.

Figure 6. Nucleo BSP architecture



```
BSP
  Adafruit_Shield
    Release_Notes.html
    stm32_adafruit_lcd.c
    stm32_adafruit_lcd.h
    stm32_adafruit_sd.c
    stm32_adafruit_sd.h
  Components
    cbtl08gp053
    Common
    hx8347d
    sn65dp141
    st7735
      Release_Notes.html
      st7735.c
      st7735.h
    stlm75
    tusb546
  STM32G081B_EVAL
  STM32G0xx_Nucleo
    Release_Notes.html
    stm32g0xx_nucleo.c
    stm32g0xx_nucleo.h
    STM32g0xx_Nucleo_BSP_User_Manual.chm
```

3.2.1 Joystick

The 5-way joystick on the shield is based on a resistor trick, to permit all the switches to share one analog pin. Each movement of the joystick control connects a different resistor and results in a different voltage reading.

The ADC peripheral is configured within the *stm32g0xx_nucleo.c/.h* driver, to get analog voltage values through the analog I/O pin 3.

The *BSP_JOY_GetState()* function reads the analog pin and compares the result with five different ranges, to determine in which direction (if any) the stick has been moved (left, right, up, down, select).

3.2.2 LCD

The LCD available on the Adafruit 1.8" TFT shield uses 4-wire SPI to communicate with the STM32G071RBT6 chipset (Digital I/O pins 13, 12, 11 and 10) and has its own pixel-addressable frame buffer to display text, shapes, lines, pixels and others.

The SPI peripheral is configured within the *stm32g0xx_nucleo.c/.h* driver, which contains also the SPI bus-link mechanism and I/O operations.

The LCD is controlled by a dedicated BSP LCD driver *stm32_adafruit_lcd.c/.h,* which uses the st7735 component that exports the LCD I/O operations needed for its process.

3.2.3 microSD

The microSD slot available on the Adafruit 1.8" TFT shield uses 4-wire SPI to communicate with the STM32G071RBT6 chipset (Digital I/O pins 13, 12, 11 and 10).

The SPI peripheral is configured within the *stm32g0xx_nucleo.c/.h* driver which contains also the SPI bus-link mechanism and I/O operations.

The microSD is controlled by a dedicated BSP SD driver *stm32_adafruit_sd.c/.h*, which exports in a generic way the SD I/O operations needed for its process.

4 Functional description of the demonstration

This demonstration application shows how to use the STM32CubeG0 firmware package with the NUCLEO-G071RB board and the Adafruit 1.8” TFT shield, to display a 128x160-pixel, full-color bitmap from a microSD card using the FatFS file system.

To start with this demonstration application, the user has to copy the provided 128x160-pixel bitmap pictures, available within the firmware package under “\Media” folder, to the root directory of a FAT formatted microSD card and insert the microSD card into the Adafruit shield microSD holder.

Note that the microSD card can have a storage capacity up to 4 Gbytes (SDSC) and that the bitmap images must have the properties detailed as in *Table 1*.

Table 1. Bitmap image properties

	Dimensions	128 x 160
	Width	128 pixels
	Height	160 pixels
	Bit depth	16
	Item type	BMP file
	Name	Must not exceed 11 characters (including bmp extension).

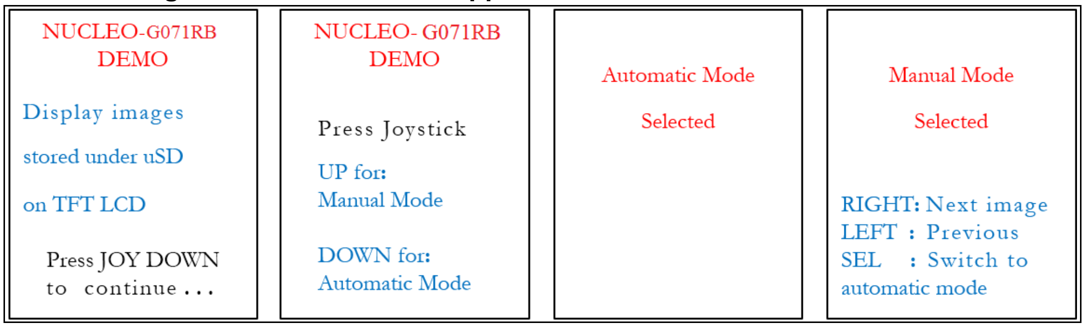
Once started, the application checks the availability of the Adafruit 1.8" TFT shield on top of the STM32 Nucleo board. This is done by reading the state of the I/O PC.13 pin (mapped to the joystick available on the shield). If the state of PB.00 is high then the shield is available.

If the Adafruit 1.8" TFT shield is not available, the LED4 blinks with a frequency of ~1Hz. A first press on the user button makes the LED4 blinking with a frequency of ~5Hz. At the second press, the LED4 blinks with a frequency of ~10Hz and at the third press, the LED4 blinks with a frequency of ~5Hz. The described process is done in an infinite loop.

If the Adafruit 1.8" TFT shield is available, the LED4 is turned ON, because it is sharing the same pin with the SPI CLK signal, used to communicate with the LCD and microSD available on the shield.

A menu is displayed on Adafruit 1.8" TFT describing the demonstration application, as shown in *Figure 7: Demonstration application menu on NUCLEO-G071RB*.

Figure 7. Demonstration application menu on NUCLEO-G071RB



NUCLEO-G071RB
DEMO

Display images
stored under uSD
on TFT LCD

Press JOY DOWN
to continue . . .

NUCLEO- G071RB
DEMO

Press Joystick
UP for:
Manual Mode

DOWN for:
Automatic Mode

Automatic Mode
Selected

Manual Mode
Selected

RIGHT: Next image
LEFT : Previous
SEL : Switch to
automatic mode

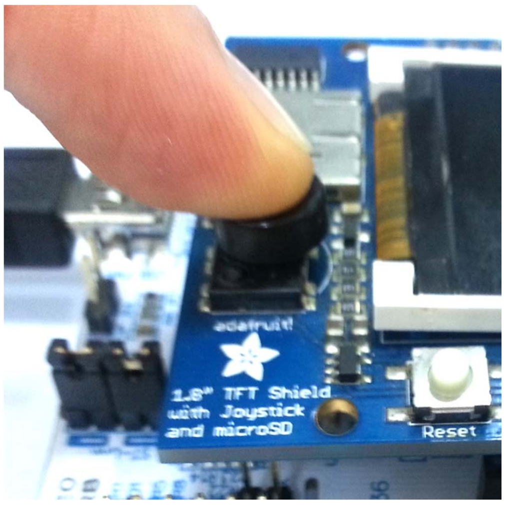
User has to follow the instructions below:

- Press the joystick DOWN to keep displaying the menu (see *Figure 8: Reading the Adafruit shield joystick*)
- Choose one of the available display modes (manual or automatic) using the joystick button:
  - **Automatic mode**: by pressing joystick DOWN

    The bitmap images available on the microSD card are displayed sequentially in a forever loop.
  - **Manual mode**: by pressing joystick UP

    The bitmap images available on the microSD card are displayed by pressing joystick right to display next image, or joystick left to display the previous one. Keeping the joystick pressed in select for ~1s, switches the display mode from manual to automatic.

Figure 8. Reading the Adafruit shield joystick

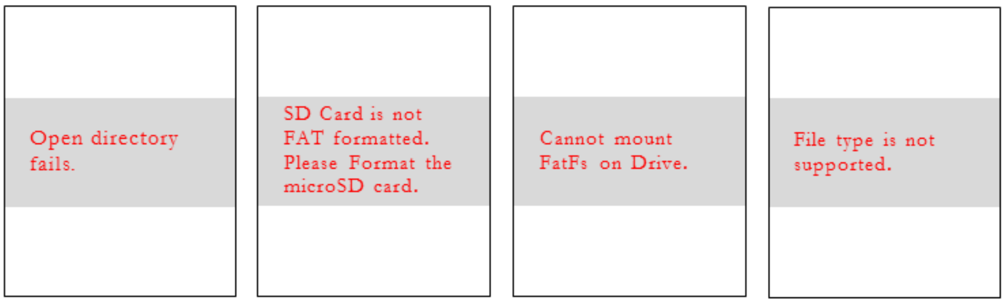


Note that the application manages some errors (refer to *Figure 9*), that can occur during the access to the microSD card to load the bitmap images:

- If the microSD card is not FAT formatted, a message is displayed on TFT. In this case, format the microSD card and put into its root directory the *.bmp files available within the firmware package under \Media folder.
- If the format of the files stored on the microSD card is not bitmap, an error message is displayed on TFT, mentioning that the format is not supported. User has to check carefully that the files, available under the microSD card root directory, comply with the above described bitmap properties.

The *Figure 10* shows the demonstration running.

Figure 9. Demonstration application error messages



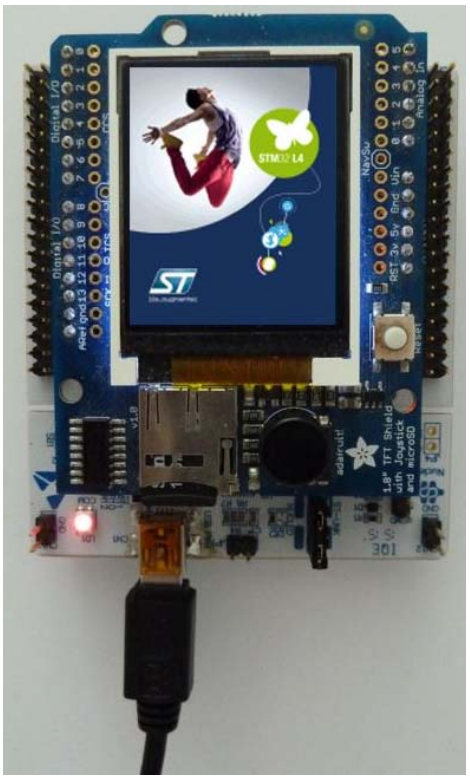
Open directory fails.

SD Card is not FAT formatted. Please Format the microSD card.

Cannot mount FatFs on Drive.

File type is not supported.

Figure 10. Demonstration running



4.1 Programming firmware application

To program the STM32 Nucleo board with the demonstration application, proceed as follows:

1. Install the preferred Integrated Development Environment (IDE)
2. Install the ST-LINK/V2.1 driver available on ST website.

There are two ways of programming the STM32 Nucleo board:

- Method 1:
  Using the preferred in-system programming tool and depending on the STM32 Nucleo board:
  – upload the STM32CubeG0_Demo_STM32G071RB_Nucleo.hex from the firmware package available under: Projects\STM32G071RB-Nucleo\Demonstrations\Binary
- Method 2:
  Choose one of the three supported tool chains (IAR™ / Keil® / AC6®) and follow the steps below:
  – Open the application folder either: Projects\SM32G071RB-Nucleo\Demonstrations\Adafruit_LCD_1_8_SD_Joystick
  – Chose the desired IDE project (EWARM for IAR, MDK-ARM for Keil or SW4STM32 for AC6)
  – Double click on the project file (for example Project.eww for EWARM)
  – Rebuild all files: go to **Project** and select **Rebuild all**
  – Load the project image: go to **Project** and select **Debug**
  – Run the program: go to **Debug** and select **Go**.

The demonstration software as well as other software examples that allow user to discover the STM32 microcontroller features, are available on ST website at www.st.com/stm32nucleo.

5 FAQs

How can I use this application to display my own images?

Use any image editing tool and crop/resize the image to be no larger than 160 pixels high and 128 pixels wide. Save it as a 16-bit color BMP format file.

Can I display more bitmap files?

Yes, it is possible. Display more pictures, by copying them under the microSD root directory. Modify the value of the MAX_BMP_FILES constant in the #define directive with the desired number of files. In this case, the user must fine-tune the _FS_LOCK value, defining the number of files that can be opened simultaneously, under “ffconf.h” the FatFS configuration file.

Can I store the bitmap files in a folder different from the root directory of the microSD?

If the bitmap files are stored in a folder different from the root directory, they cannot be accessed by the demonstration application. The “File type not supported” error message is displayed on the LCD.

To make it work, add the new directory path within f_open() and f_opendir() FatFS APIs calls under the fatfs_storage.c file.

6 Revision history

Table 2. Document revision history

Date	Revision	Changes
26-Sep-2018	1	Initial version
20-Nov-2018	2	Updated *Introduction* to STM32Cube standard