STM32CubeG0 STM32G0316-DISCO demonstration firmware

Introduction

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

STM32Cube includes:

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

The STM32CubeG0 demonstration firmware running on the STM32G0316-DISCO board is built around the STM32Cube hardware abstraction layer (HAL) and low-layer (LL) APIs, and board support package (BSP) components. It embeds several applications that demonstrate various features of the STM32G030J6 device.

1 STM32CubeG0 main features

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

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

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

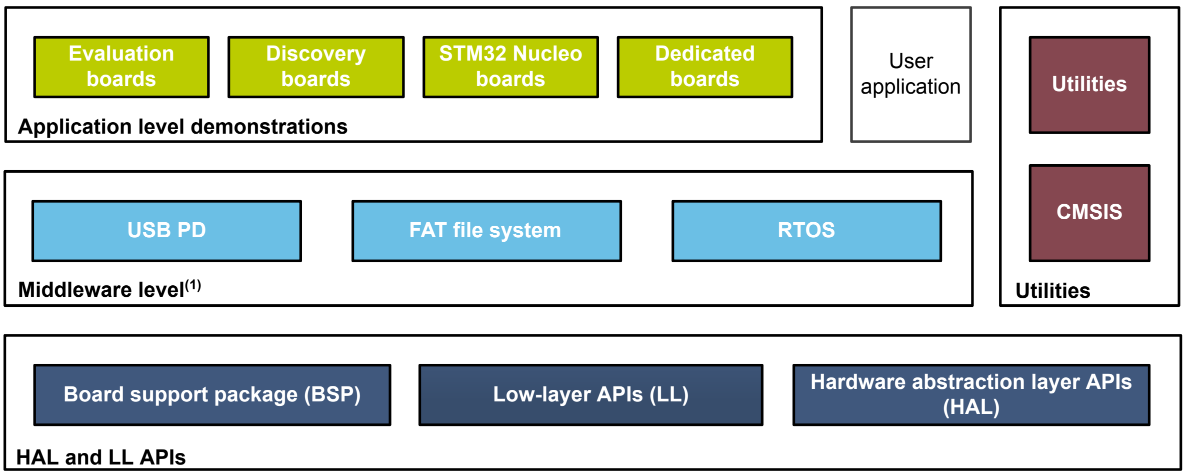
The STM32CubeG0 MCU Package also contains a set of middleware components with the corresponding examples. They come in free user-friendly license terms:

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

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

Figure 1 shows the block diagram of STM32Cube.

Figure 1. STM32CubeG0 firmware components



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

The demonstration firmware for the STM32G0316-DISCO board comes on top of the STM32CubeG0 MCU Package, which is based on a modular architecture allowing the reuse of software components separately in standalone applications. The STM32Cube demonstration kernel manages all the modules, allows the addition of new modules dynamically, and the access to the storage, graphical, and components common resources.

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

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

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

2 Getting started with the demonstration

2.1 Hardware requirements

The hardware requirements to start the demonstration are the following:

- One STM32G0316-DISCO board (MB1454)
- One USB cable to power the STM32G0316-DISCO board from its ST-LINK USB

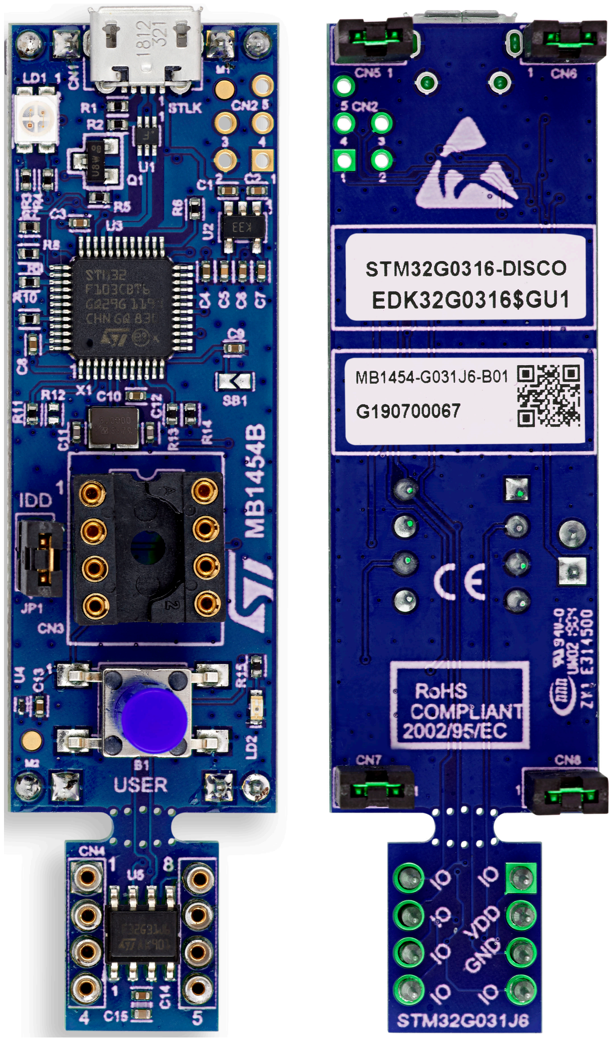
The STM32G0 Discovery board connects to the host PC by means of the USB cable. It does not require any external power supply.

For detailed information on the hardware part, refer to UM2603.

2.2 Hardware settings

The STM32CubeG0-based demonstration supports the STM32G030J6 device and runs on the STM32G0316-DISCO board from STMicroelectronics presented in Figure 2.

Figure 2. STM32G0316-DISCO top and bottom views (MB1454)



2.3 Demonstration firmware programming

The user may program the demonstration using the two methods detailed below.

2.3.1 Using binary file

To program the demonstration's binary image in the internal Flash memory, the user must handle the `STM32CubeG0_Demo_STM32G0316-DISCO.hex` file located under `Projects\STM32G0316-DISCO\Demonstrations\Binary`, using tool such as *STM32Cube Programmer*.

2.3.2 Using preconfigured projects

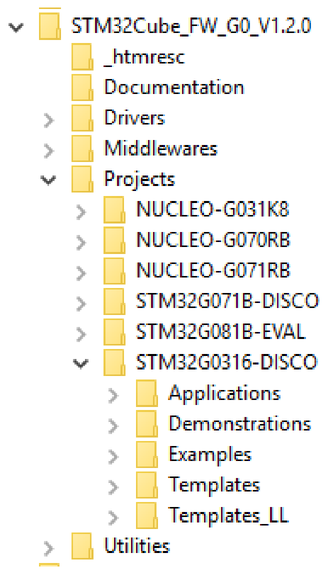
The user selects the folder corresponding to his preferred toolchain (MDK-ARM, EWARM or SW4STM32). Then the user:

1. Opens the demonstration project and rebuild all sources.
2. Loads the project image through his debugger.
3. Restarts the Discovery kit.

3 Demonstration firmware package

3.1 Demonstration repository

Figure 3 shows the demonstration folder organization.



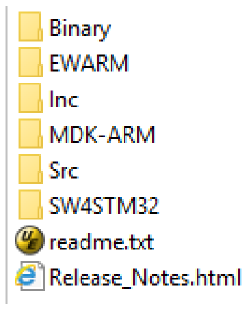
```
STM32Cube_FW_G0_V1.2.0
    _htmresc
    Documentation
    Drivers
    Middlewares
    Projects
        NUCLEO-G031K8
        NUCLEO-G070RB
        NUCLEO-G071RB
        STM32G071B-DISCO
        STM32G081B-EVAL
        STM32G0316-DISCO
            Applications
            Demonstrations
            Examples
            Templates
            Templates_LL
    Utilities
```

Figure 3. STM32CubeG0 Firmware package folder organization

The demonstration sources are located in the `Projects` folder of the STM32CubeG0 MCU Package for each supported board. The folder selected here is `STM32G0316-DISCO` for the Discovery board.

The demonstration firmware highlights the possibility to use PF2 pin for different purposes (Reset or GPIO), but also the possibility to use different pins on the same pad (multi-bonding).

Figure 4 illustrates the organization of the `Demonstrations` folder.



```
Binary
EWARM
Inc
MDK-ARM
Src
SW4STM32
readme.txt
Release_Notes.html
```

Figure 4. STM32G0316-DISCO Demonstrations folder organization

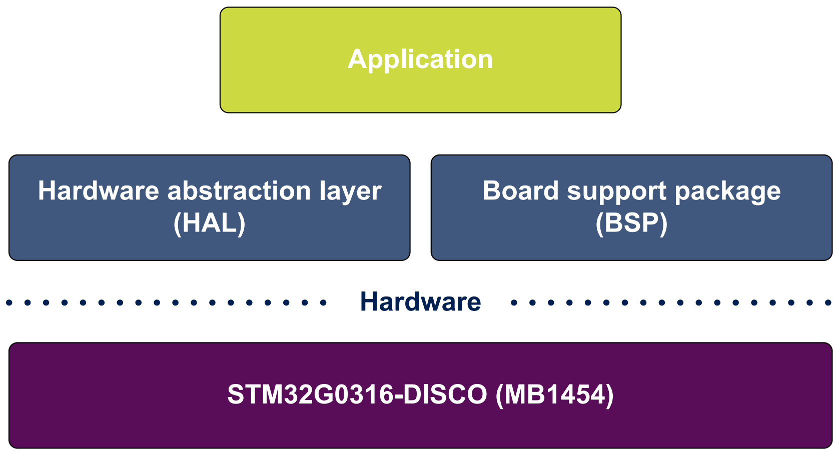
The `Demonstrations` folder contains the following sub-folders:

- `Binary`: contains link to demo compiled binary file in Hex format
- `Inc`: Demonstration application header files
- `Src`: Software development environments
  - `EWARM`: IAR Embedded Workbench™
  - `MDK-ARM`: Keil® Microcontroller Development Kit
  - `SW4STM32`: SW4STM32, system workbench for STM32

3.2 Demonstration architecture overview

Figure 5 shows the top-level software architecture of the STM32G0316-DISCO demonstration firmware. Dedicated sub-sections briefly describe the software elements mentioned in this diagram.

Figure 5. STM32G0316-DISCO demonstration firmware architecture


Application
Hardware abstraction layer (HAL)
Board support package (BSP)
Hardware
STM32G0316-DISCO (MB1454)

3.2.1 Application

The main application initializes the HAL, configures the clock and then starts the demonstration.

3.2.2 HAL and BSP level

HAL level layer consists in the *stm32g0xx.HAL* drivers together with the STM32G0316-DISCO board support package (BSP).

3.3 Microcontroller resources

The following sections detail the peripherals of the STM32G030J6 microcontroller used by the demonstration.

3.3.1 Peripherals

Figure 6 shows the peripherals used by the demonstration.

Figure 6. STM32G031J6 peripherals

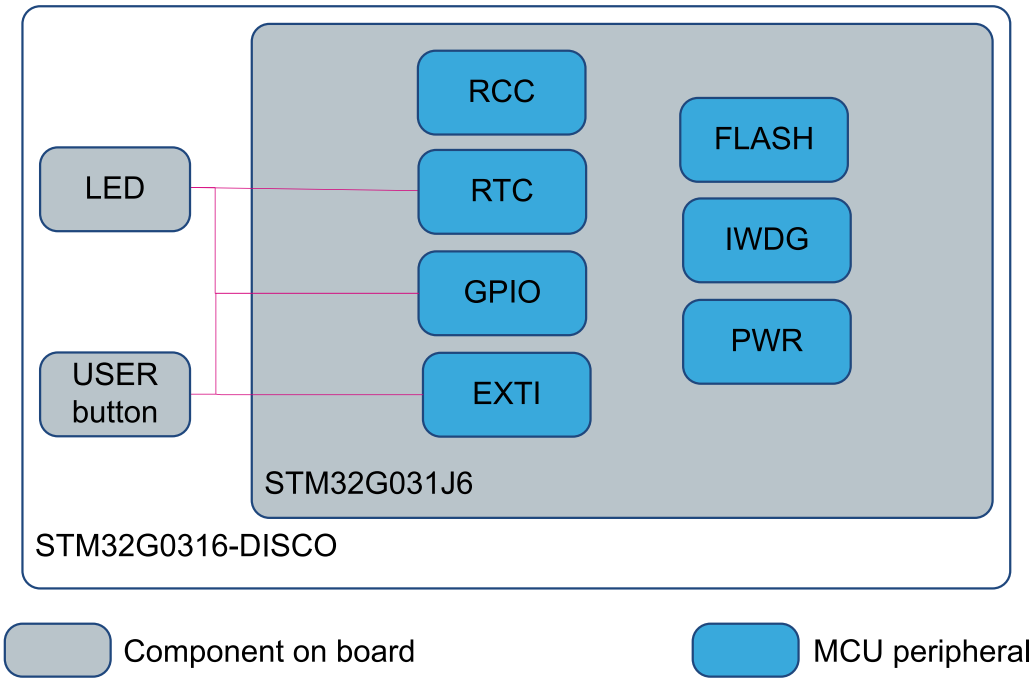


Table 1 indicates which STM32G030J6 peripherals are used by the demonstration firmware and for which purpose.

Table 1. STM32G030J6 peripherals used by the demonstration firmware

Peripheral	Usage description
RCC	Configures the system clock, checks and resets flags status when the demonstration firmware is started.
IWDG	When the demonstration firmware operates in PF2-NRST mode, IWDG is used as internal reset source so that the user may check generation of a pulse on the PF2 pin.
FLASH	Changes the PF2 pin configuration (NRST or GPIO) through option bytes programming.
GPIO	When the demonstration firmware operates in PF2-GPIO mode, PF2 can be used to enter standby mode or to trigger option-bytes re-configuration.
EXTI	When the demonstration firmware operates in PF2-GPIO mode PF2 related EXTI line is configured to generate an interrupt upon PF2 rising and falling edges.
RTC	Controls LED2 using the auto-reload wakeup timer capability of this peripheral.
PWR	When the demonstration firmware operates in PF2-GPIO mode, PWR configures the wakeup pin and enters standby mode.

3.3.2 Interrupts

Table 2 shows all the external interrupts used by the demonstration.

Table 2. **STM32G031J6 demonstration interrupts usage**

Interrupt	Usage description
SysTick	Delay management
RTC_TAMP_IRQHandler	Wakeup timer interrupt management
EXTI line 0	PA0 interrupt mode, USER button management
EXTI line 1	PA1 interrupt mode, USER button management
EXTI line 2	PF2 interrupt mode, falling edge, USER button management

4 Running the demonstration

4.1 Overview

This demonstration provides the possibility to use PF2 pin for different purposes (RESET or GPIO), and the possibility to use different pins on the same pad (multi-bonding).

When the Discovery kit is powered on, the demonstration firmware checks the PF2 pin capability by reading the option bytes. The demonstration firmware has two functioning modes, which depend on the PF2 pin capability:

- When PF2 pin is configured in RESET Input/Output mode, the demonstration firmware enters the PF2-NRST mode. In this mode, reset can be forced by grounding the PF2 pin and internal resets are propagated to the PF2 pin (20 µs pulse generation).
- When PF2 is configured in GPIO mode, the demonstration firmware enters the PF2-GPIO mode. In this mode, PF2 can be used as a standard GPIO, with all functionalities (Input/Output/Alternate functions/ Analog) available.

4.2 Modes description

4.2.1 PF2-NRST mode

When the demonstration firmware operates in PF2-NRST mode, the independent watchdog is configured to regularly reset the system. The user may then check that a pulse occurs on the PF2 pin.

If the user presses the RST/USER button, option bytes are updated to configure PF2 in GPIO mode. When option bytes loading operation ends, system resets and the demonstration firmware switches to the PF2-GPIO mode.

4.2.2 PF2-GPIO mode

When the demonstration firmware operates in PF2-GPIO mode, RST/USER button selects amongst the following sub-modes:

- Short press on the RST/USER button: the system goes in Standby mode. The user may then measure the power consumption using the JP1 pins. Another press on the RST/USER button wakes up the system from Standby mode.
- Double press on the RST/USER button: independent watchdog is configured to reset the system after a short delay. It shows that even without Reset input, system reset can be triggered.
- Long press on the RST/USER button: option bytes are updated to configure PF2 in RESET mode. When option bytes loading operation ends, system resets and the demonstration firmware switches back to the PF2-NRST.

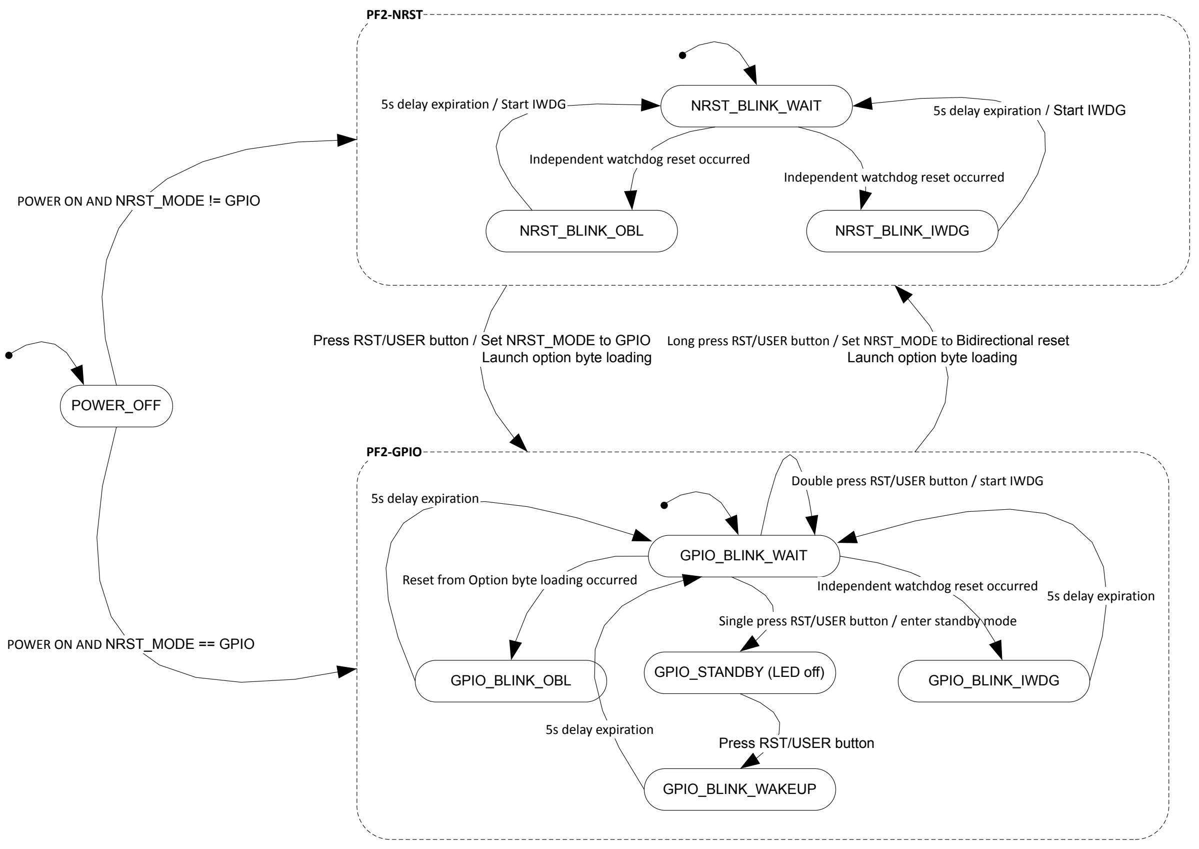
At each new reset in GPIO mode, a new GPIO is configured as input mode. Corresponding external interrupt and interrupt vector are configured accordingly. This demonstrates G0 multi-bonding feature, in which several IOs can be redirected to a single PAD. PA0, PA1, PA2 and PF2 are all linked to PAD 4.

4.3 Functional description

4.3.1 State machine

The state machine in Figure 7 illustrates the functioning of the demonstration firmware.

Figure 7. Functional state machine of the STM32G0316-DISCO demonstration firmware


PF2-NRST
NRST_BLINK_WAIT
5s delay expiration / Start IWDG
5s delay expiration / Start IWDG
Independent watchdog reset occurred
Independent watchdog reset occurred
NRST_BLINK_OBL
NRST_BLINK_IWDG
POWER ON AND NRST_MODE != GPIO
POWER_OFF
POWER ON AND NRST_MODE == GPIO
Press RST/USER button / Set NRST_MODE to GPIO
Launch option byte loading
Long press RST/USER button / Set NRST_MODE to Bidirectional reset
Launch option byte loading
PF2-GPIO
Double press RST/USER button / start IWDG
5s delay expiration
GPIO_BLINK_WAIT
Reset from Option byte loading occurred
Independent watchdog reset occurred
5s delay expiration
Single press RST/USER button / enter standby mode
GPIO_BLINK_OBL
GPIO_STANDBY (LED off)
GPIO_BLINK_IWDG
5s delay expiration
Press RST/USER button
GPIO_BLINK_WAKEUP

Whatever the operating mode is (PF2-NRST or PF2-GPIO), the demonstration firmware performs the following steps:

- Identification of the reset origin, reported during five seconds to the user, thanks to LED2 (one toggling scheme per reset origin).
- Exit from Standby mode reset: four fast successive blinks of LED2 in one second
- Independent watchdog reset: three fast successive blinks of LED2 in one second
- Option byte loader reset: two fast successive blinks of LED2 in one second: Power reset

After five seconds, LED-blinking scheme indicates to the user in which mode the PF2 pin is configured:

- PF2-GPIO: slow blinking (two seconds)
- PF2-NRST: fast blinking (one second)

4.3.2 LED toggling scheme

Figure 8 illustrates the LED2-toggling scheme used by the demonstration firmware.

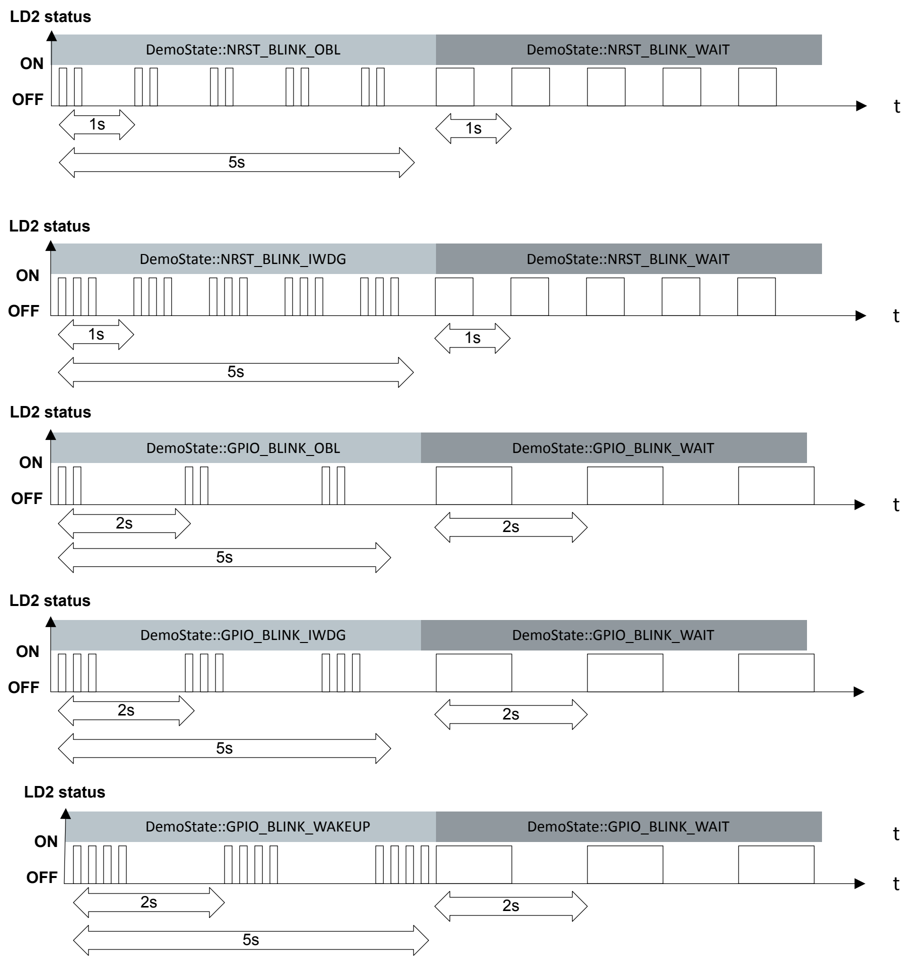

LD2 status
ON
OFF
DemoState::NRST_BLINK_OBL
DemoState::NRST_BLINK_WAIT
1s
1s
5s
t
LD2 status
DemoState::NRST_BLINK_IWDG
DemoState::NRST_BLINK_WAIT
1s
1s
5s
t
LD2 status
DemoState::GPIO_BLINK_OBL
DemoState::GPIO_BLINK_WAIT
2s
2s
5s
t
LD2 status
DemoState::GPIO_BLINK_IWDG
DemoState::GPIO_BLINK_WAIT
2s
2s
5s
LD2 status
DemoState::GPIO_BLINK_WAKEUP
DemoState::GPIO_BLINK_WAIT
2s
2s
5s
t
t

Figure 8. STM32G0316-DISCO demonstration firmware LED blinking scheme

5 Software architecture

Following sections detail the application in charge of initializing demonstration application, HAL, interrupt handler and launching the main module.

5.1 Clock control

Here is the configuration of the various system clocks in this demonstration application:

- STM32G031 internal clocks are derived from the HSI running at 16 MHz in RST mode.
- PLL is used as the system clock source in GPIO mode. System clock and HCLK frequency are set to 64 MHz.

Only the RTC is clocked by a 32 kHz external oscillator.

5.2 Main application API overview

The application goal is to prepare demonstration startup, by initializing all the hardware and software. Table 3 provides a description of all the actions performed by the different functions in `main.c`.

Table 3. Main function description

Functions (main.c)	Description
Main	Initialize the HAL, configure the clock and then start the demonstration

The file `stm32g0xx_it.c` is also part of the application and is used, as usual, to map the interrupt vector on the HAL driver, depending on the module requirement.

Table 4 explains the main demonstration functionalities in `main.c`.

Table 4. Main applications functions description

Functions (main.c)	Description
MainNrstDemoMode	Main function of PF2-NRST demonstration mode
MainGpioDemoMode	Main function of PF2-GPIO demonstration mode
GpioDemoModeShortPress	GPIO Sub Mode short press handler
GpioDemoModeLongPress	GPIO Sub Mode Long press handler
GpioDemoModeDoublePress	GPIO Sub Mode double press handler
LedBlinkSchemeOn	Configure and Turn on LED blinking scheme
LedBlinkSchemeOff	Turn off Led blinking scheme
PushButtonConfig	Configure push button (PF2)
PushButtonInputScan	Scan push button input signal in order to determine short, long or double key press
GetDemoMode	Get current demo sub-mode
SetDemoMode	Set demo sub-mode: SUBDEMO_MODE_GPIO or SUBDEMO_MODE_NRST
BackupDomainAccessEnable	Enable backup domain access
CheckBootReason	Check boot reason and save them

Functions (main.c)	Description
ExecuteBootReason	Check boot reason and warn the user in case of option byte reload reset, IWDG reset and Standby exit
HAL_GPIO_EXTI_Falling_Callback	EXTI line detection callback
HAL_RTCEx_WakeUpTimerEventCallback	Wakeup timer callback
ErrorHandler	Functional error report to the user: LED stays ON forever
RtcConfig	Configure RTC
SystemClock	Set the rights clocks for Flash and RCC

5.3 STM32G0316-DISCO BSP API overview

Table 5 shows STM32G0316-DISCO BSP API functions.

Table 5. STM32G0316-DISCO BSP API

Functions	Description
BSP_LED_Init	Configure LED
BSP_LED_DeInit	Put LEDs in unconfigured state
BSP_LED_Toggle	Toggle the selected LED
BSP_LED_On	Turn the selected LED on
BSP_LED_Off	Turn the selected LED off
BSP_GetVersion	Return the STM32G0316-DISCO BSP driver revision

5.4 Dynamic memory use

The demonstration is currently using:

- CSTACK = 0x400
- HEAP = 0x200

6 Memory footprint

This chapter sums up RAM and ROM consumptions per software blocks.
Table 6 also gives the total demonstration firmware consumption.

Table 6. RAM and ROM consumption

Full demonstration firmware	ROM code (Byte)	ROM data (Byte)	RAM data (Byte)
Demo Application	1298	58	208
BSP Drivers	118	1	4
HAL Drivers	3828	6	12
STM32G030J6 Device	338	2	4
Others (linker...)	420	32	1024
Grand Total	6002	99	1252

Revision history

Table 7. Document revision history

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

Contents

1 STM32CubeG0 main features 2
2 Getting started with the demonstration. 3
2.1 Hardware requirements 3
2.2 Hardware settings 3
2.3 Demonstration firmware programming 3
2.3.1 Using binary file 4
2.3.2 Using preconfigured projects. 4
3 Demonstration firmware package 5
3.1 Demonstration repository 5
3.2 Demonstration architecture overview 6
3.2.1 Application 6
3.2.2 HAL and BSP level 6
3.3 Microcontroller resources 7
3.3.1 Peripherals 7
3.3.2 Interrupts. 8
4 Running the demonstration 9
4.1 Overview 9
4.2 Modes description 9
4.2.1 PF2-NRST mode 9
4.2.2 PF2-GPIO mode 9
4.3 Functional description. 10
4.3.1 State machine 10
4.3.2 LED toggling scheme 11
5 Software architecture 12
5.1 Clock control 12
5.2 Main application API overview. 12
5.3 STM32G0316-DISCO BSP API overview 13
5.4 Dynamic memory use 13
6 Memory footprint 14

Revision history 15
Contents 16
List of tables 18
List of figures. 19

List of tables

Table 1. STM32G030J6 peripherals used by the demonstration firmware . . . 7
Table 2. STM32G031J6 demonstration interrupts usage . . . 8
Table 3. Main function description . . . 12
Table 4. Main applications functions description . . . 12
Table 5. STM32G0316-DISCO BSP API . . . 13
Table 6. RAM and ROM consumption . . . 14
Table 7. Document revision history . . . 15

List of figures

Figure 1. STM32CubeG0 firmware components . . . 2
Figure 2. STM32G0316-DISCO top and bottom views (MB1454) . . . 3
Figure 3. STM32CubeG0 Firmware package folder organization . . . 5
Figure 4. STM32G0316-DISCO Demonstrations folder organization . . . 5
Figure 5. STM32G0316-DISCO demonstration firmware architecture . . . 6
Figure 6. STM32G031J6 peripherals. . . . 7
Figure 7. Functional state machine of the STM32G0316-DISCO demonstration firmware. . . . 10
Figure 8. STM32G0316-DISCO demonstration firmware LED blinking scheme . . . 11