

Simplifying Embedded Microcontroller Design using the STM32G0 and STM32CubeMX 5.0



Technology Tour 2019
Anaheim, CA | March 26



Agenda

2

9:00 AM – 12:00 PM

Start Tools Installation

STM32G0 Overview

Lab: Blink an LED by software

Lab: Use hardware (PWM timer) to blink an LED

Lab: External Interrupt

Lab: Low power

Optional Lab: Power Consumption Estimation

Optional Lab: Printf

Begin Tools Installation

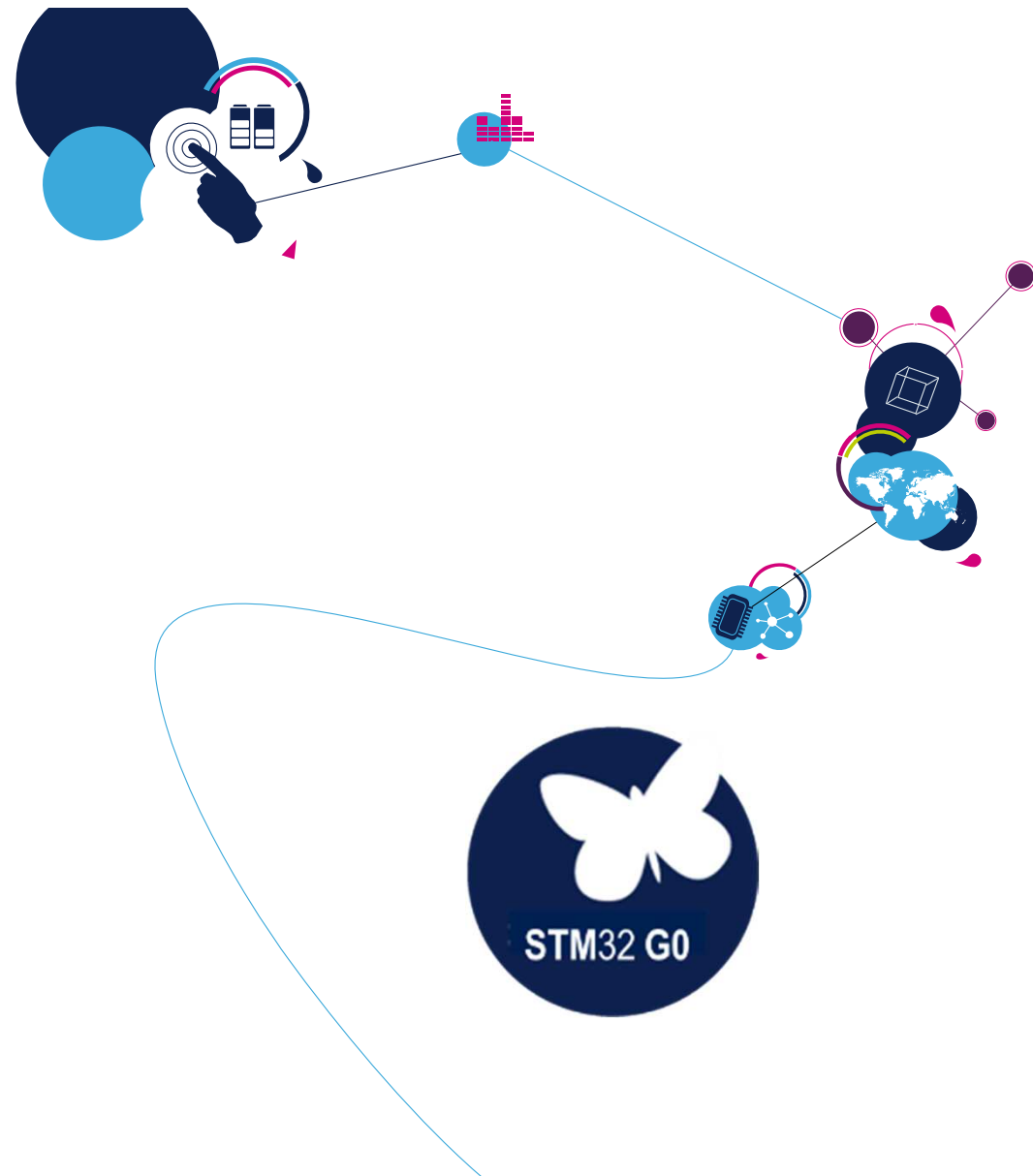
3

- Copy the content of the USB drive to a temporary directory on your machine (i.e. C:\temp)
- Install Java
- Install Keil uVision 5

For more details on the tools installation, refer to the document “**STM32G0 Workshop Installation from USB drive_v1.0.pdf**” part of the files you copied from the USB drive

STM32G0 MCU series

Efficiency at its best



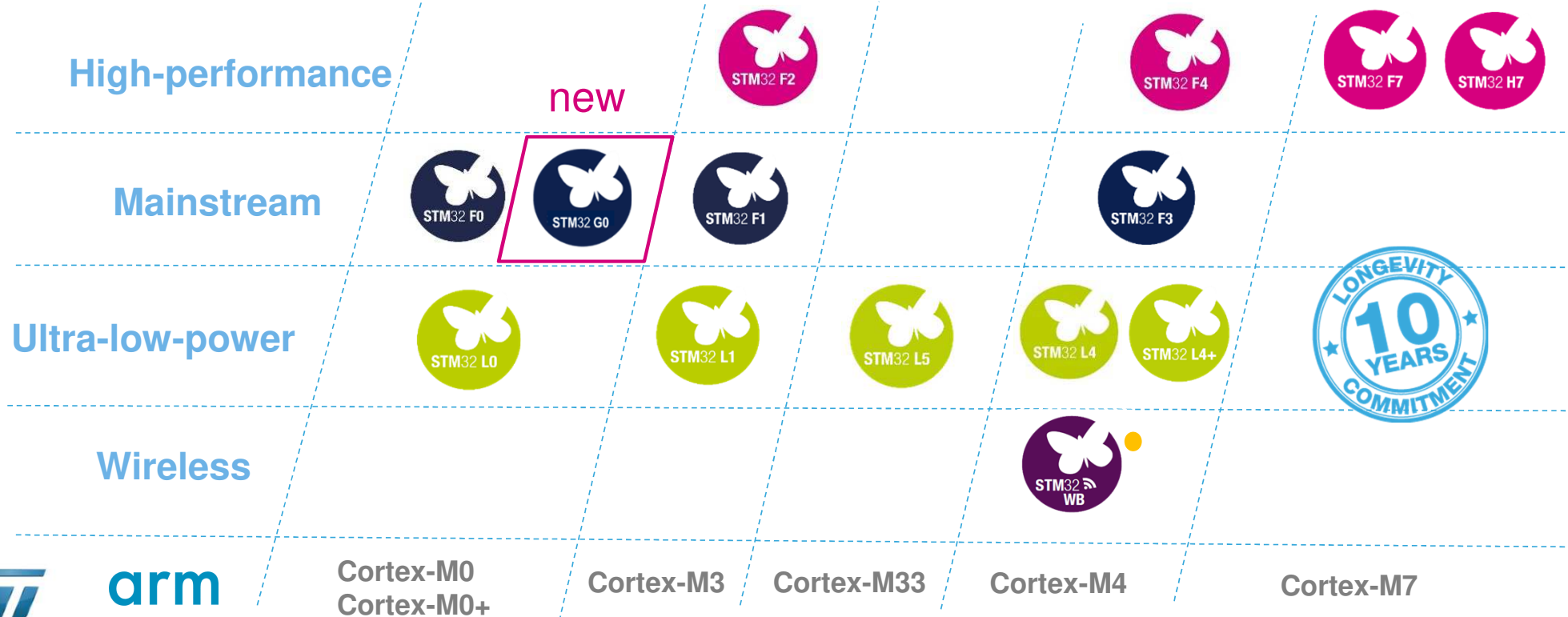


STM32G0: great investment

5

Keep releasing your growing creativity

Note ●: Cortex-M0+ Radio Co-processor



arm

Cortex-M0
Cortex-M0+

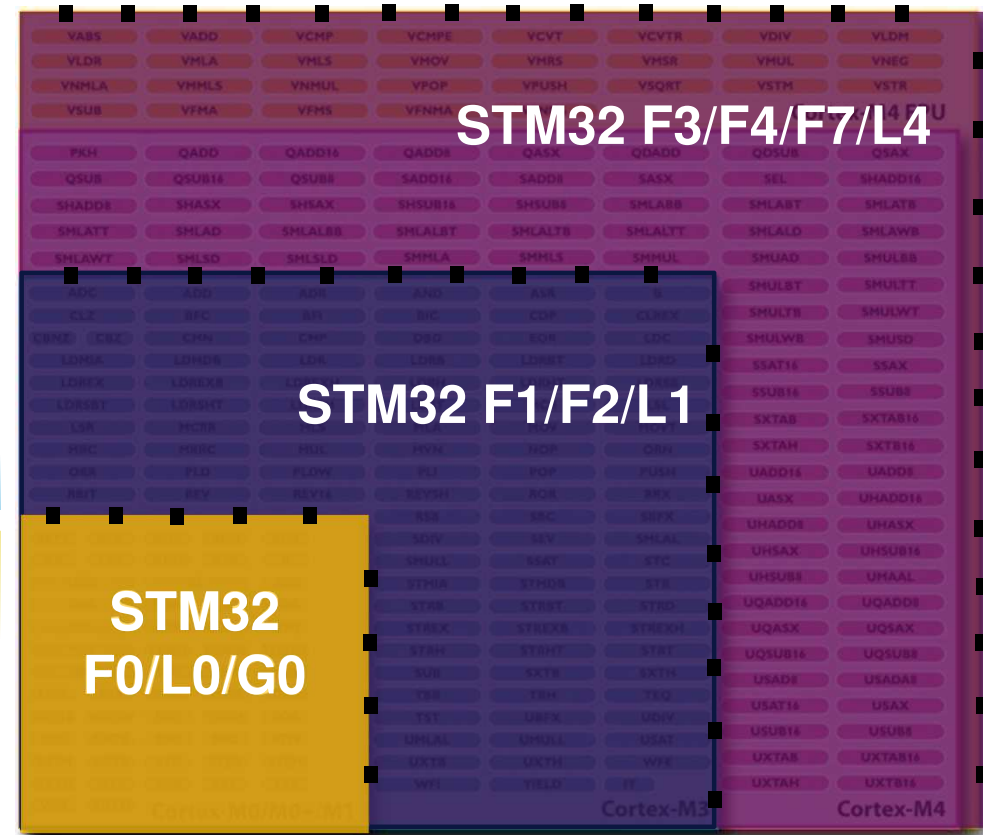
Cortex-M3

Cortex-M33

Cortex-M4

Cortex-M7

6





Key messages of STM32G0 series

7

1

Efficient

- ARM Cortex M0+ at 64MHz
- Compact cost: maximum I/Os count
- Best RAM/Flash Ratio
- Smallest possible package down to 8-pin
- Very low power consumption (3µA in stop, <100µA/MHZ in Run)
- Accurate internal high-speed clock 1% RC
- Best optimization, down to each and every detail
- Offers the best value for money

2

Robust

- Low electromagnetic susceptibility, EMC
- Clock Monitoring and 2 Watchdogs
- Error correction on Flash
- IoT ready with embedded security
- Hardware AES-256 encryption
- New Securable Memory Area
- Safe Firmware upgrade / Install

3

Simple

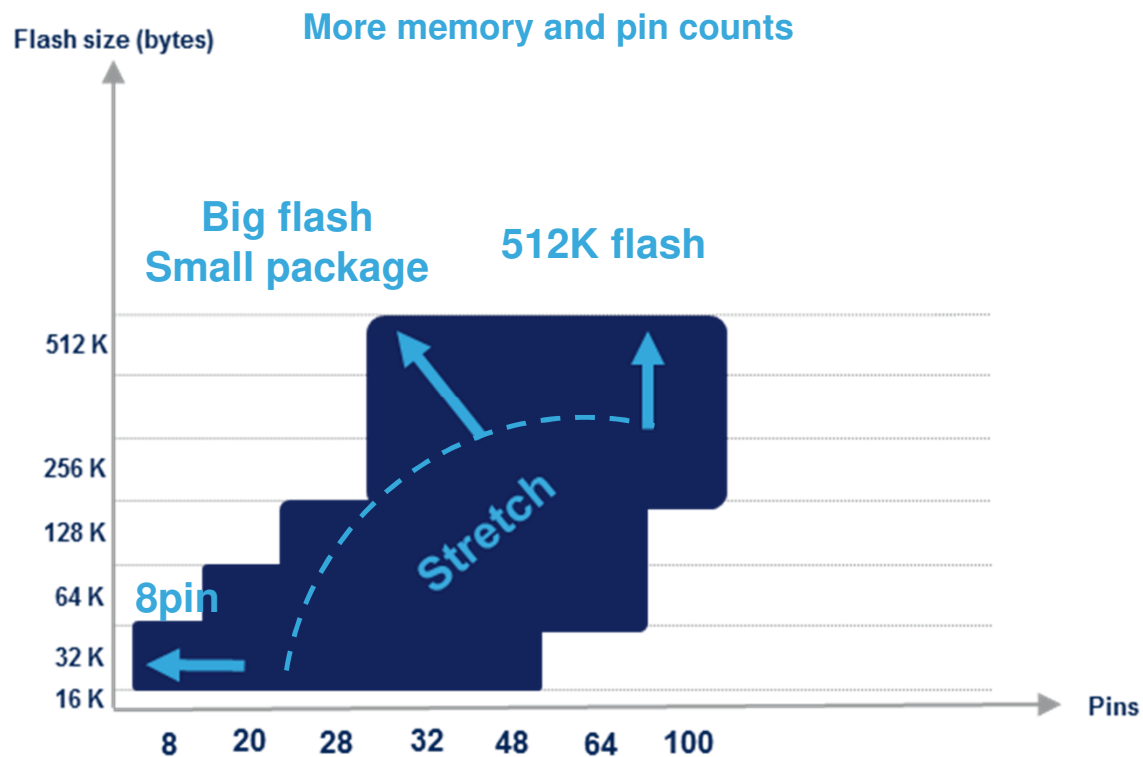
- Easy to configure thanks to the intuitive and graphical STM32CubeMX configuration tool
- Easy to develop based on the Hardware Abstraction Layer library (HAL) or the low-layer library (LL) allowing maximum re-use and faster time-to-market



Wider platform

8

Portfolio streeeeeeetched for efficient budget applications



More packages

SO / TSSOP

WLCSP

BGA

QFN

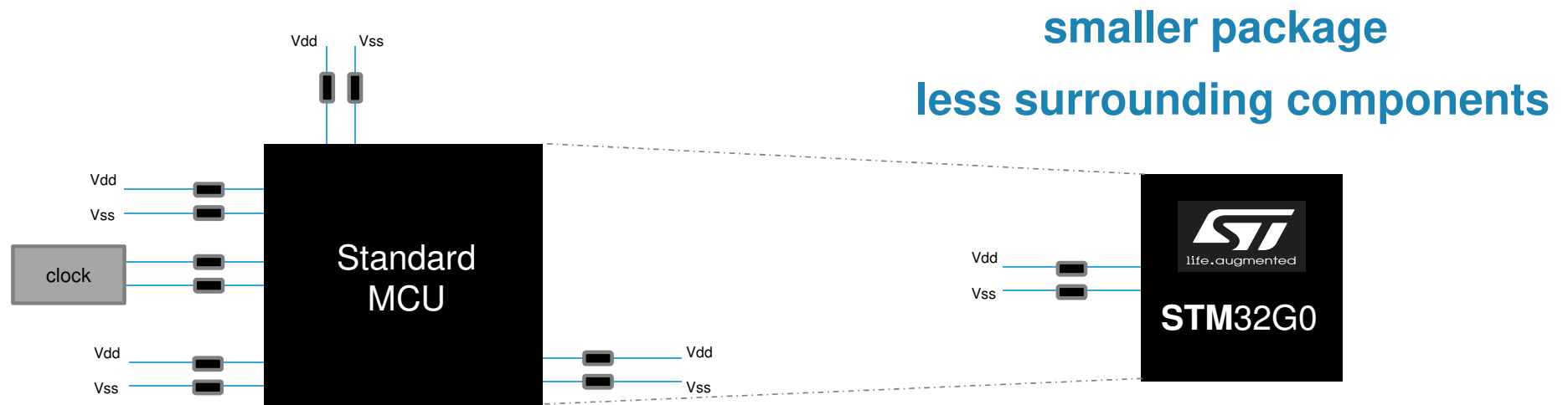
LQFP



Reducing BOM cost

9

New platform optimized with 1 power supply pair only up to 64pin packages





Innovations for your benefit

10

- **No external clock** **-10cts**
Accurate internal high speed clock +/-1% for 0 / 90°C
- **No decoupling capacitances** **-4cts**
Remove up to 6 decoupling capacitors for supply and clocks
- **Smaller PCB** **-1cts**
Smaller package, less components: save on PCB area

Additional benefits for your convenience:

- **USB-C power delivery** **-15cts**
Integrated transceivers, pull-up/down resistors and digital
- **Secure programming** **-25cts**
In house or at 3rd parties

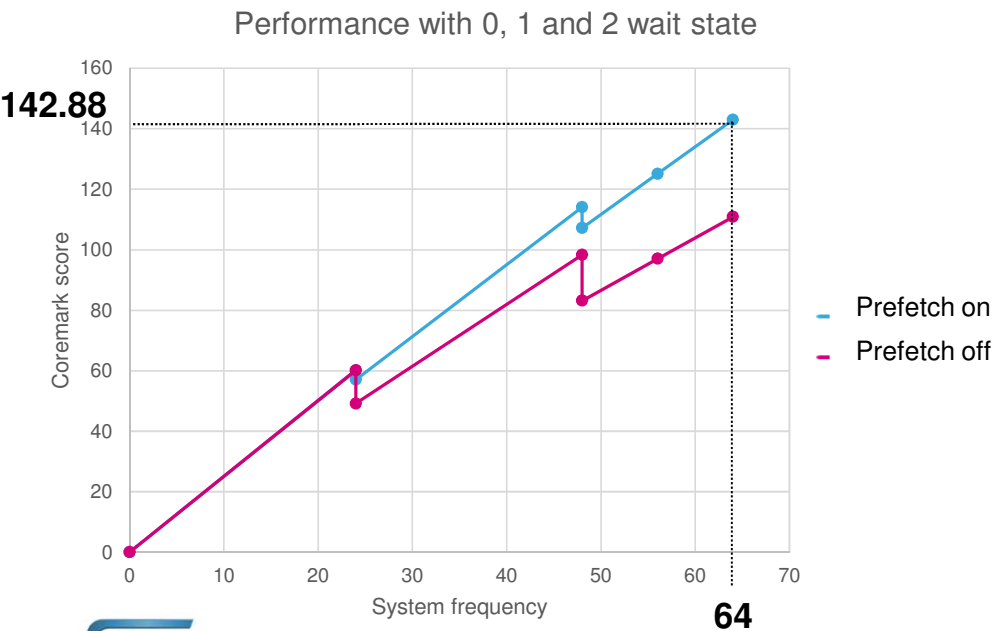




Providing more performance

11

Do not compromise on performance with STM32G0



- Up to 64 MHz/ 59 DMIPS
- Up to >142 CoreMark Result
- ARM Cortex-M0+ with Memory Protection Unit (MPU)
- Flexible **DMA** up to 12 channels



Low-power modes efficiency

12

When Mainstream MCU Series meets low-power requirements

Wake-up time

258 μ s

14 μ s

5 μ s

6 cycles

VBAT

10 nA / 400 nA*

Tamper: few I/Os, RTC

SHUTDOWN

40 nA / 500 nA*

Wake-up sources: reset pin, few I/Os, RTC

STANDBY

200 nA / 500 nA*

Wake-up sources: + BOR, IWDG

STOP

Flash-RTC off-off/off-on/on-off

3.0 μ A / 5 μ A / 8 μ A

Wake-up sources: + all I/Os, PVD, COMPs, LPUART, LPTIM, I²C, UART, USB-PD

SLEEP

24MHz, Vdd=3V, PLL=on

800 μ A

Wake-up sources: any interrupt or event

RUN at 64 MHz

<100 μ A / MHz

Conditions: 25°C, Vdd = 3V

Note : * without RTC / with RTC

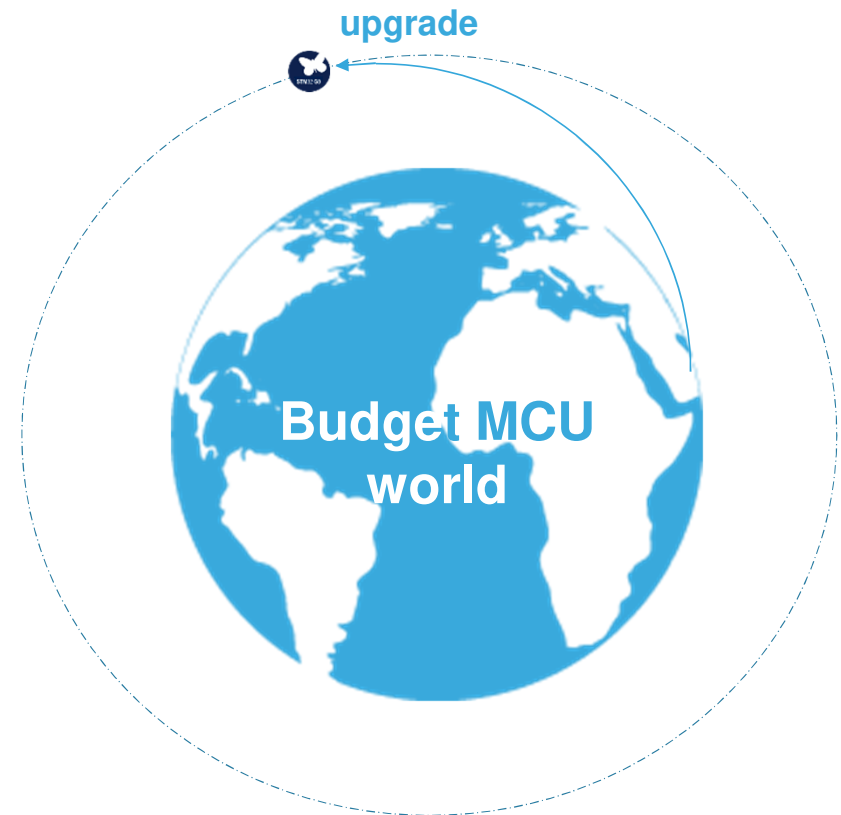


Ready for tomorrow

13

Faster, more accurate analog and digital functions

- More **RAM** for Flash
 - Up to 36KB SRAM for 128KB and 64KB Flash
- **Timers** frequency up to **128MHz** resolution (<8ns)
 - **Advanced control** capabilities
- **12-bit ADC** up to **2.5MSPS** (0.4μs) conversion time
 - **16-bit** oversampling by hardware
- **32Mbit/s SPI**, 7 Mbaud/s USART, 1Mbit/s I²C communication





OBD

FD CAN

Up to 2 instances



Industrial

V_{BAT} with RTC

for battery backup

400 nA in V_{BAT} mode
for RTC and
20x 32-bit backup registers



TRNG & AES

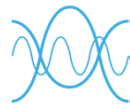
for Security

128-/256-bit AES
key encryption hardware
accelerator



Comparators

2 instances
Down to 30ns propagation delay



DAC

2x 12-bit DAC,

ADC

16x12-bit, 16-bit oversampling
2.5MSPS (0.4μs)



Timers

8ns PWM resolution
Advanced control
16- and 32-bit



Smart peripherals

14



USB-C Power Delivery

Up to 2 ports with dead-battery management



USB

USB 2.0
Full speed
Device / Host



SPI / UART / I²C

4x SPIs
8 UARTs (ISO 7816, LIN, IrDA, modem)
3 I²C

I/Os Up to 92 fast I/Os



Smart integration

15

Save on battery life

Low consumption process and design
Low-Power UART: wake-up on frame
Low-Power Timer: counts and generate signals
I²C wake-up on address



Save on BOM cost



+/-1% high speed clock internal from 0 to 90°C
+/-2% high speed clock internal from -40 to 125°C
IO maximization: smaller package footprint

More flexibility



More RAM or **more safety** with parity enable/disable
Dynamic DMA assignement on **DMAMUX**
All IOs with external interrupt capability



Always keep control Diagnose, react



Main Clock monitoring
Backup clock and interrupts
Voltage monitoring: programmable interrupts and reset
Window watchdog on CPU clock
Independant watchdog on independant clock
Checksum by hardware
ECC on Flash, **Parity** on RAM

High temperature



from -40°C
up to + 125°C

High robustness

Highly immune to fast-transients
Robust IOs against negative injections





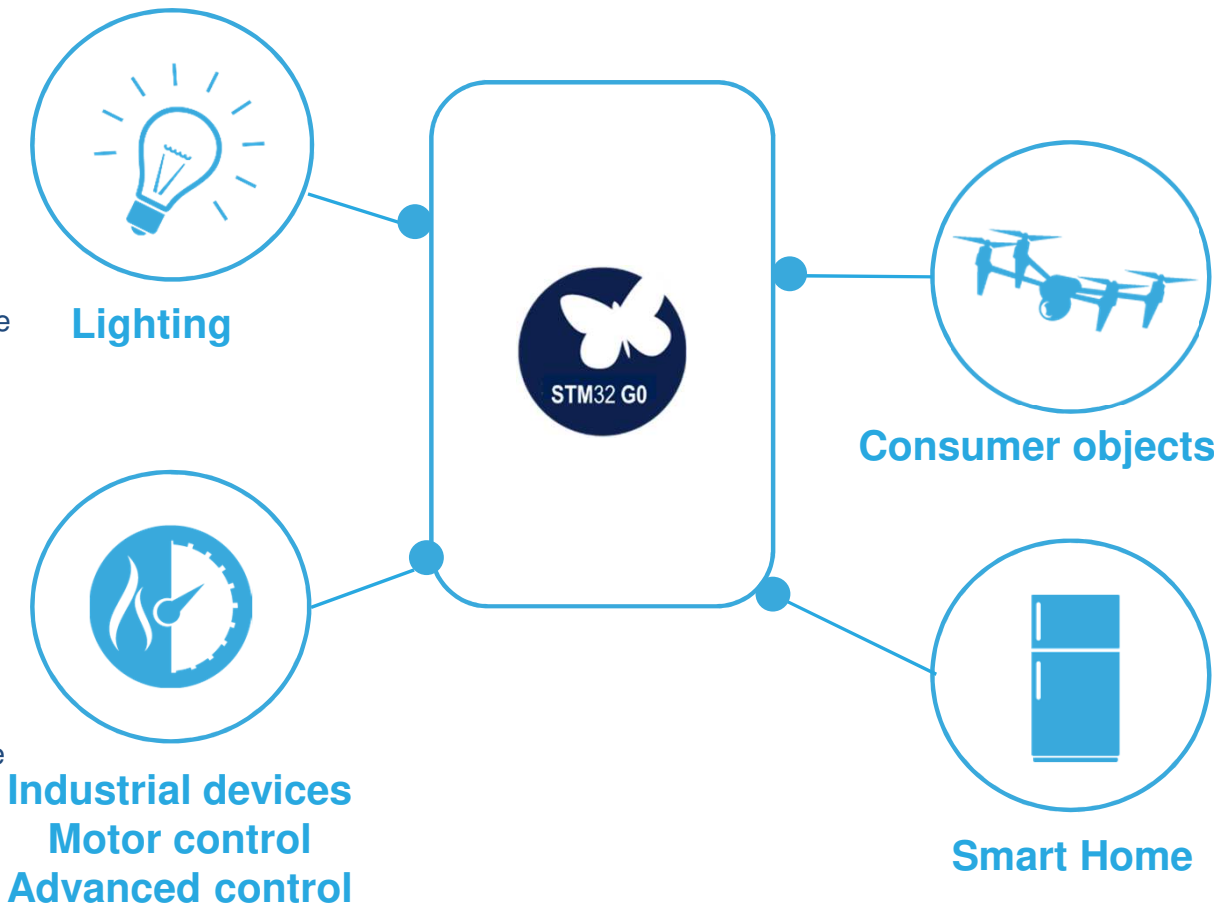
Smart applications

16

- High temperature 125°C
- Fast CPU 64MHz
- Advanced timers with high-resolution 7.8ns
- Fast comparators
- ADC-12bit, DAC-12bit
- Low-thickness packages
- AES & security for secure upgrades

Air conditioning, e-bikes, industrial equipments

- High temperature 125°C
- CANFD support
- SPI, USART, I²C
- Advanced timers with high-resolution 7.8ns
- Real Time Clock with backup registers
- AES & security for secure upgrades



Lighting

Consumer objects

Smartphones, IoT devices, rechargeable connected devices, drones, toys

- Low-thickness, small form-factor
- 64MHz CPU with DMA
- Low consumption in run and low-power, fast wake-up
- USB type-C Power Delivery 3.0
- USB FS 2.0 dev/host crystal-less

Home appliances, alarms and safety, advanced user interfaces

- High temperature 125°C
- Safety monitoring features
- More RAM for flash
- Low consumption <100µA/MHz in run

Industrial devices Motor control Advanced control

Smart Home

STM32 G0 product lines

17

Common peripherals and architecture:

ARM Cortex-M0+
64MHz
0.93 DMIPS/MHz

MPU

Communication Peripheral:
USART, SPI, I²C

Multiple general-purpose
16-bit timers

Integrated reset and brown-
out warning

DMA channels

2x watchdogs
Real-time clock (RTC)

Integrated regulator
PLL and clock circuit

Main oscillator and
32 kHz oscillator

Internal RC oscillators
32kHz , 16 MHz

-40 to +85 °C

Low voltage 1.65 to 3.6 V
(Value Line: 2.0 to 3.6V)

Temperature sensor

Up to +125°C

STM32G0x0 – Value Line (ex: STM32G070)

Up to 512KB Flash	Up to 64KB SRAM	12-bit ADC 2.5MSPS
-------------------------	--------------------	--------------------------

STM32G0x1 – Access Line (ex: STM32G071)

Up to 512KB Flash	Up to 80KB SRAM	12-bit ADC 2.5MSPS	2x Comp 2x 12-bit DAC	1x 32-bit Timer	1x 16-bit MC Timer >100MHz	1x 16-bit Timer >100MHz	USB-PD	USB OTG 2.0 FS	CAN-FD	Securable Memory Area
-------------------------	-----------------------	--------------------------	-----------------------------	--------------------	----------------------------------	-------------------------------	--------	----------------------	--------	-----------------------------

STM32G0+11 – Access Line & Encryption (ex: STM32G081)

Up to 512KB Flash	Up to 80KB SRAM	12-bit ADC 2.5MSPS	2x Comp 2x 12-bit DAC	1x 32-bit Timer	1x 16-bit MC Timer >100MHz	1x 16-bit Timer >100MHz	USB-PD	USB OTG 2.0 FS	CAN-FD	Securable Memory Area	AES TRNG
-------------------------	-----------------------	--------------------------	-----------------------------	--------------------	----------------------------------	-------------------------------	--------	----------------------	--------	-----------------------------	-------------

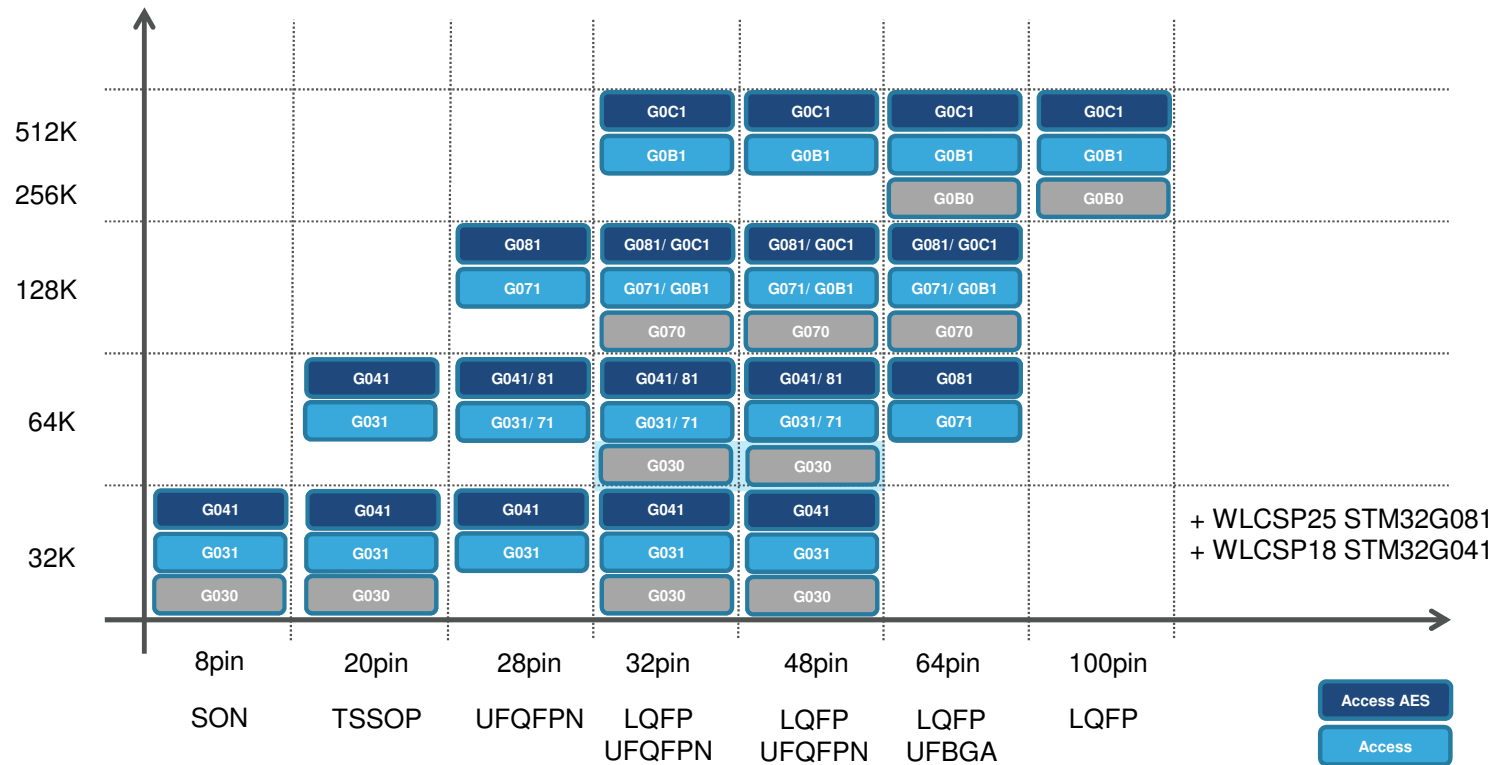


On some part-numbers



STM32G0 Portfolio

18

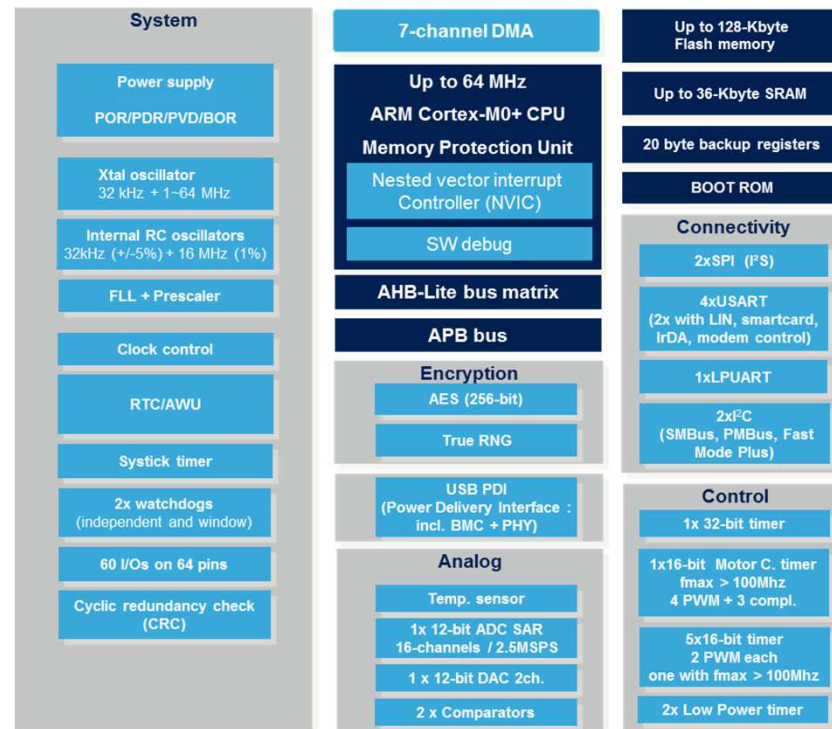


+ WLCSP25 STM32G081
+ WLCSP18 STM32G041



Advanced features and solutions

- Arm 32-bit Cortex-M0+ core
- 1.7 to 3.6V power supply
- RAM maximization
- 1% internal clock
- Direct Memory Access (DMA)
- Communication peripherals
- USB-C Power Delivery

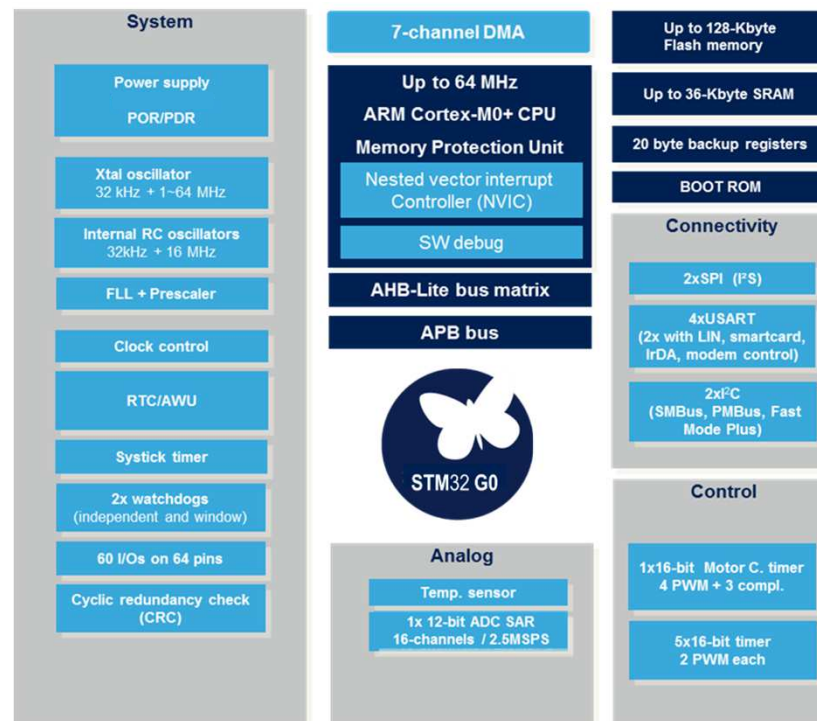


- Timers up to 2xfcpu resolution
- Real Time Clock
- I/O ports maximization
- ADC 12-bit Ultra-fast
- DAC 12-bit
- Comparators
- Safety features
- Advanced Security features



No compromise on what matters

- Arm 32-bit Cortex-M0+ core
- 2.0 to 3.6V power supply
- RAM maximization
- 1% internal clock
- Direct Memory Access (DMA)
- Communication peripherals



- Timers
- Real Time Clock
- I/O ports maximization
- ADC 12-bit Ultra-fast
- Safety features



More security

21

Integrated security features, ready for tomorrow's needs

Firmware IP protection

Mutual distrustful

Secret key storage

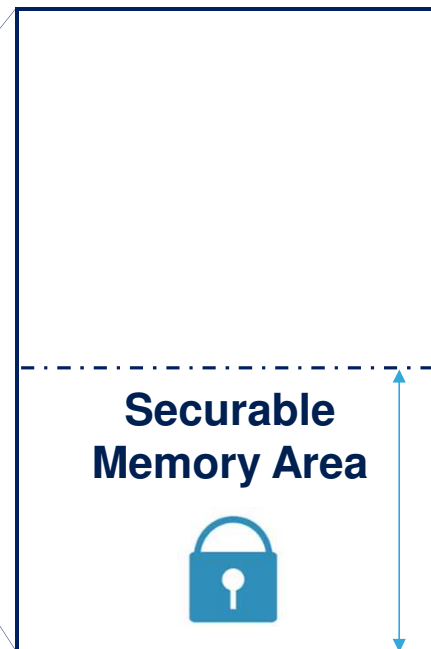
Authentication

Secure firmware upgrade



Securable Memory Area
Execute-only Protection
Read-out Protection
Write Protection
Memory Protection Unit (MPU)
AES-256 / SHA-256 Encryption
True Random Number Generator
Unique ID

User flash



Standard user flash by default

Can be secured once exiting
No more access nor debug

Configurable size

Good fit to store critical data

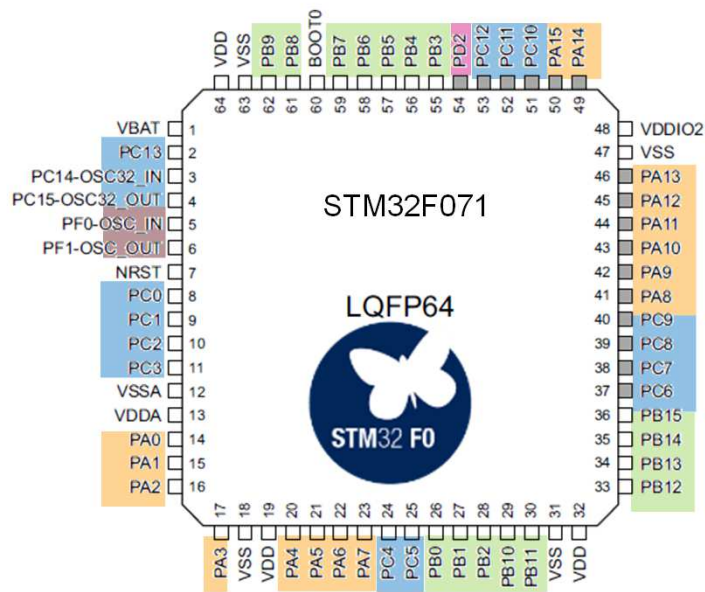
- Critical routines
- Keys



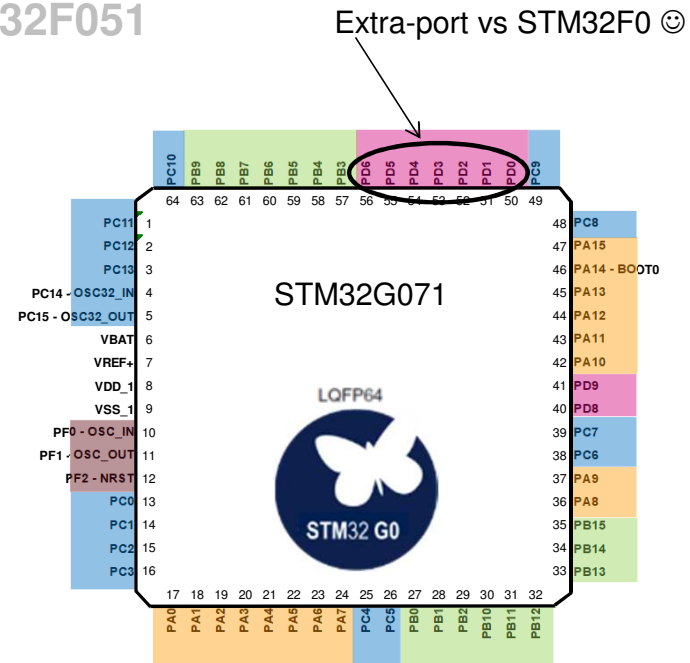
LQFP64 pin-to-pin comparison

22

- 9 IOs more on STM32G071 vs STM32F071
- 5 IOs more on STM32G071 vs STM32F051



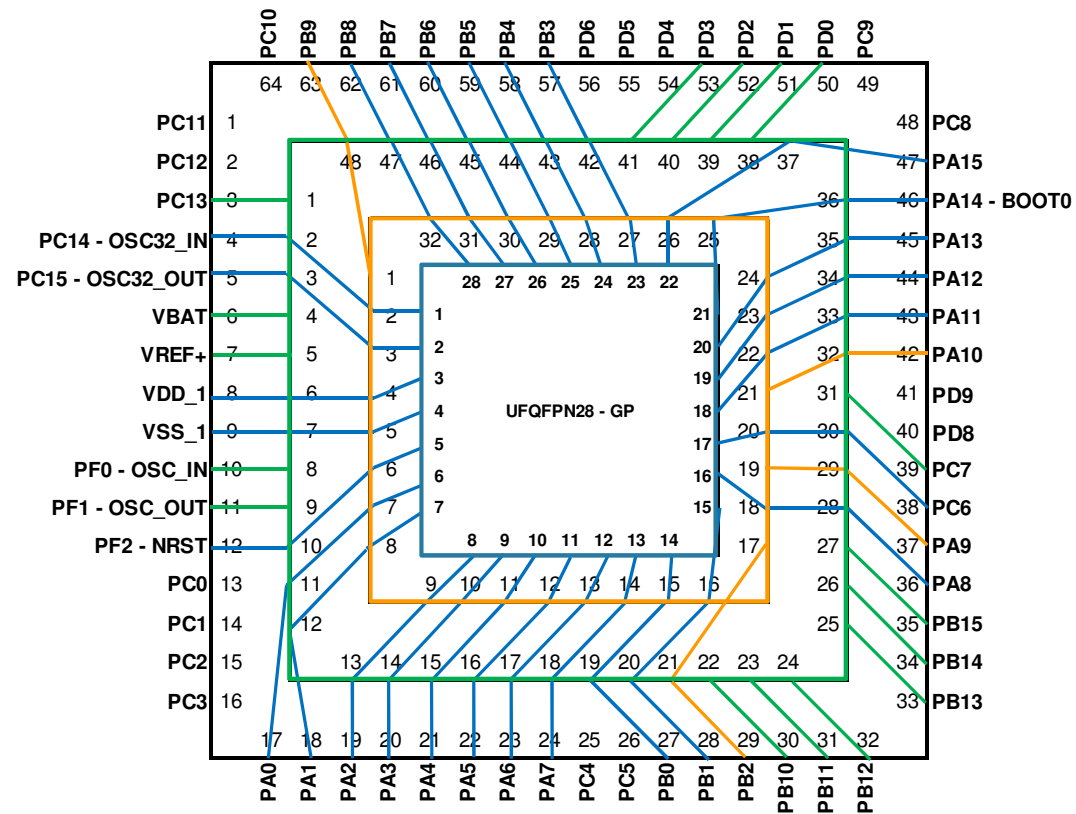
51 I/Os



60 I/Os



Consistent and optimized pinout





STM32G0 ecosystem

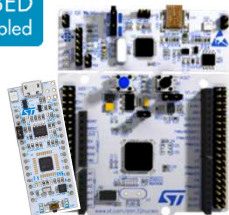
24

Go fast, be first

HARDWARE TOOLS

STM32 Nucleo

arm
MBED
Enabled



Flexible
prototyping

Discovery kit



Key feature
prototyping

Evaluation board

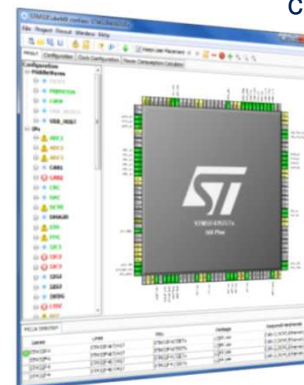


Full feature
evaluation

SOFTWARE TOOLS



STM32CubeMX featuring intuitive pin selection, clock tree configuration, code generation and power consumption calculation

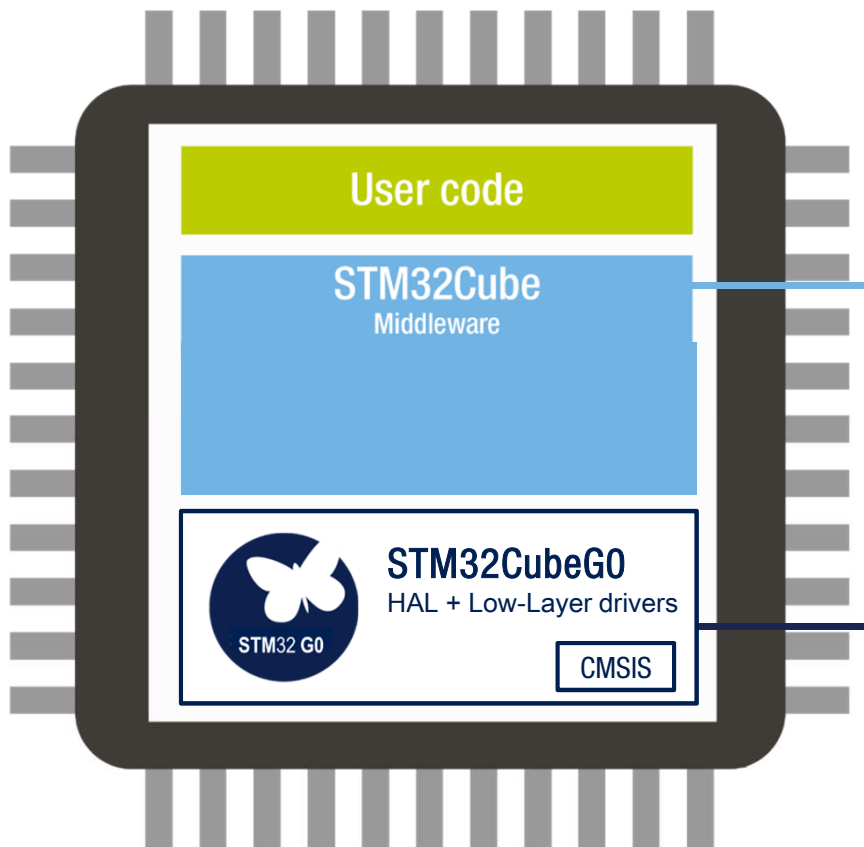




STM32G0 ecosystem

25

Platform approach or custom code: you choose



- Open-source FAT file system (FatFs)
- Open-source real-time OS (FreeRTOS)
- Dozens of examples

- STM32G0 Hardware Abstraction Layer (HAL) portable APIs
- **High-performance, light-weight low-layer (LL) APIs**
- High coverage for most STM32 peripherals
- Production-ready and fully qualified
- Dozens of usage examples
- Open-source BSD license





SUMMARY

3 Keys of STM32G0 series

26

1 Efficient

2 Robust

3 Simple



Continue with the tools installation

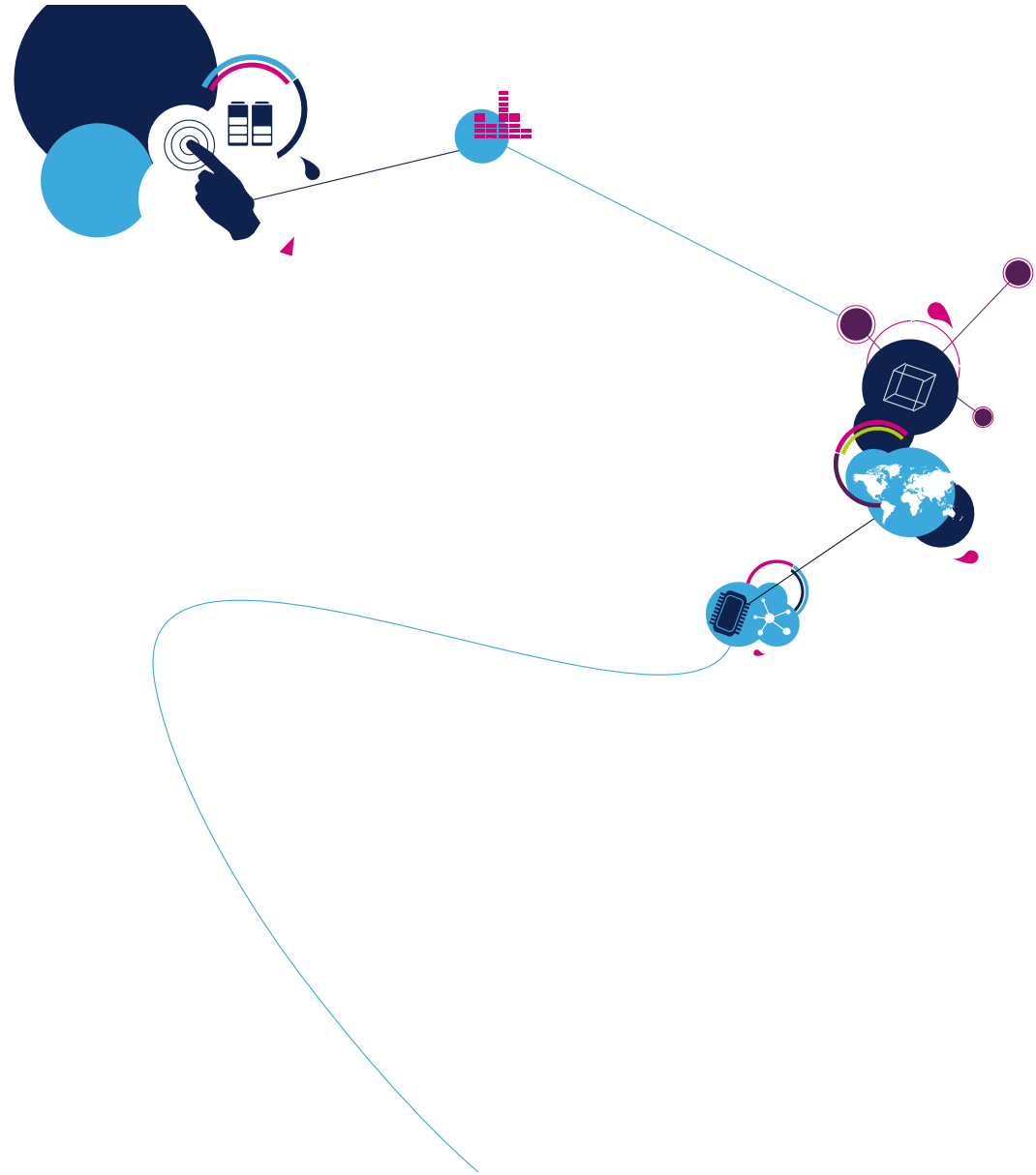
27

- By this time Keil uVision 5 should be installed
- Install STM32G0 Pack for Keil uVision5 (From Pack Installer)
- Activate Keil uVision5 (from Keil uVision5)

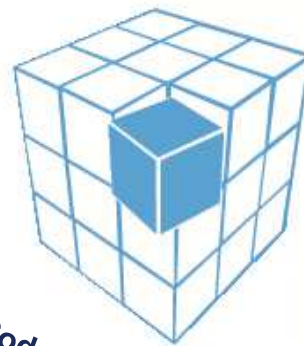
For more details on the tools installation, refer to the document “**STM32G0 Workshop Installation from USB drive_v1.0.pdf**” part of the files you copied from the USB drive

STM32CubeMX 5.0

STM32CubeMX graphical software configuration tool



STM32CubeMX



Initialization Code generation based on user choices



STM32CubeL0



STM32CubeL1



STM32CubeL4



STM32CubeF0



STM32CubeF1



STM32CubeF2



STM32CubeF3



STM32CubeF4



STM32CubeF7



STM32CubeH7

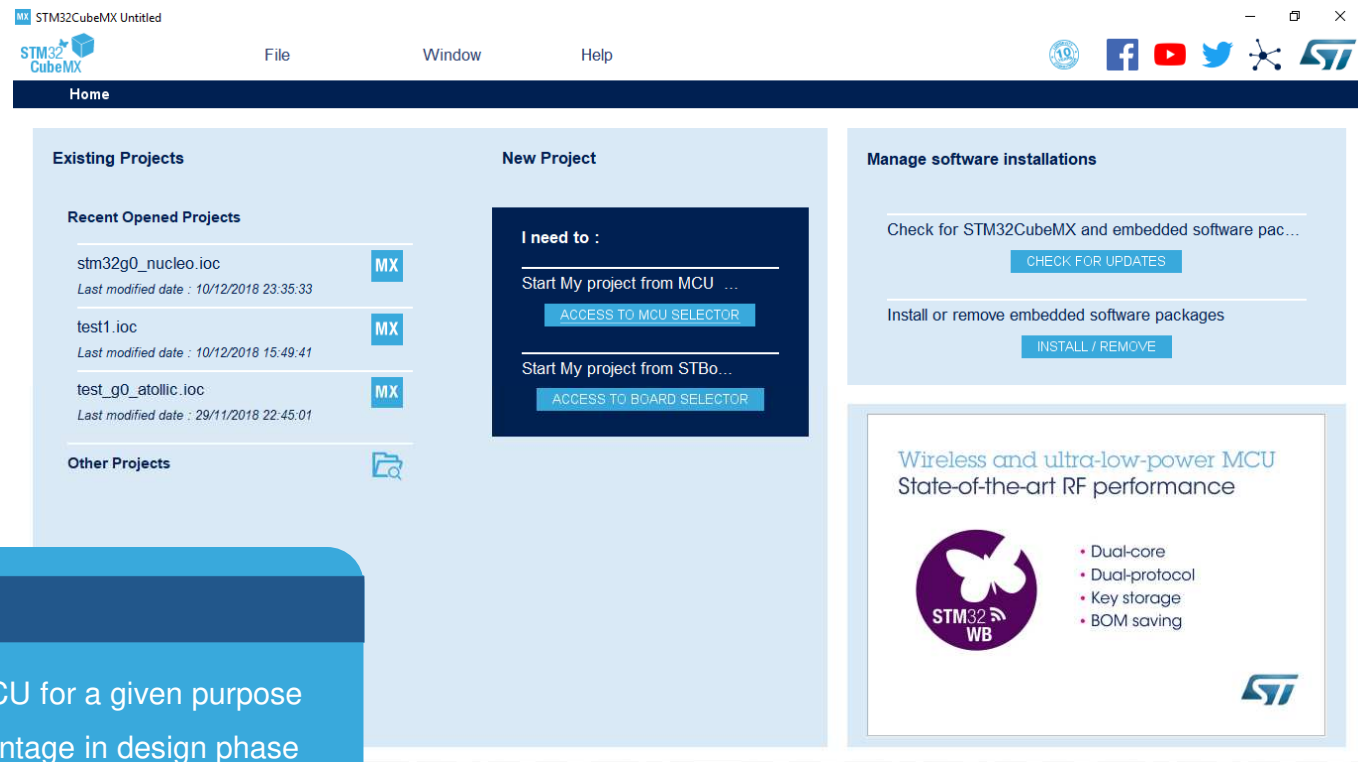


STM32CubeG0

STM32Cube HAL: Portable API within all series - Middleware stacks when applicable: RTOS, USB, TCP/IP, Graphics, ...

- Choose ideal MCU and simply configure

- Pinouts
- Clocks and oscillators
- Peripherals
- Low-power modes
- Middleware



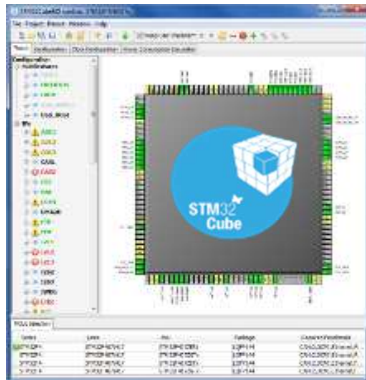
Application benefits

- Helps choose the correct MCU for a given purpose
- Simulation provides an advantage in design phase
- Boosts development speed with a headstart

- Peripheral and middleware parameters
- Power consumption calculator
- Code generation
 - Possible to re-generate code while keeping user code intact.
- Option of command-line and batch operation
- Expandable by plugins
- MCU selector
 - Filter by family, package, peripherals or memory sizes.
 - Search for similar product.
- Pinout configuration
 - Choose peripherals to use and assign GPIO and alternate functions to pins.
- Configure NVIC and DMA
- Clock tree initialization
 - Choose oscillator and set PLL and clock dividers.

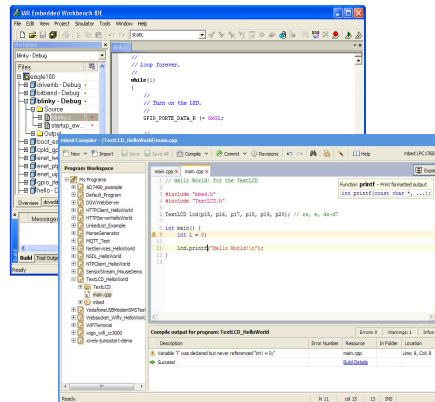
Comprehensive choice of IDEs

32



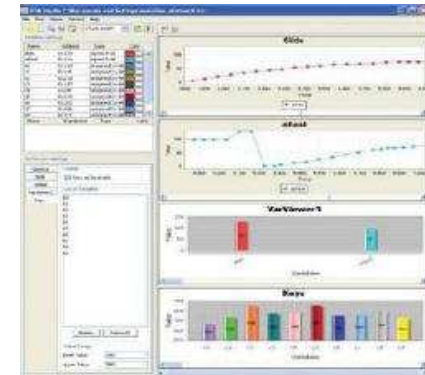
STM32CubeMX

Generate Code



Partners IDEs

Compile & Debug



STMStudio

Monitor

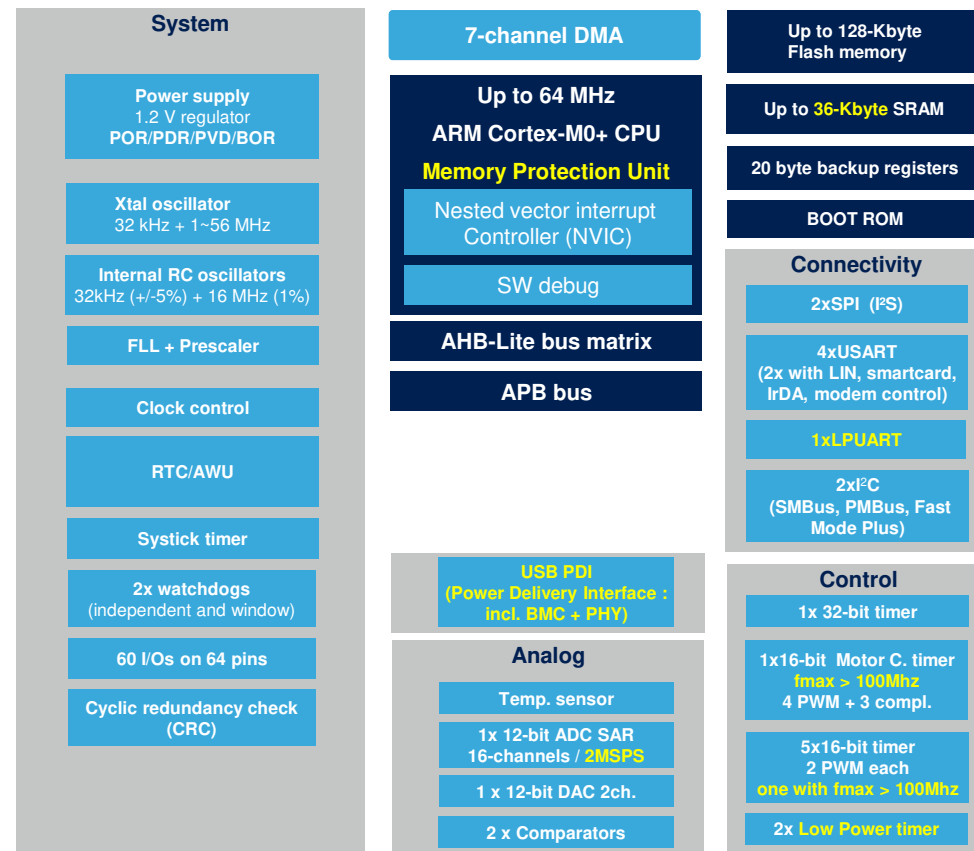


STM32G071 Block diagram

Main specification

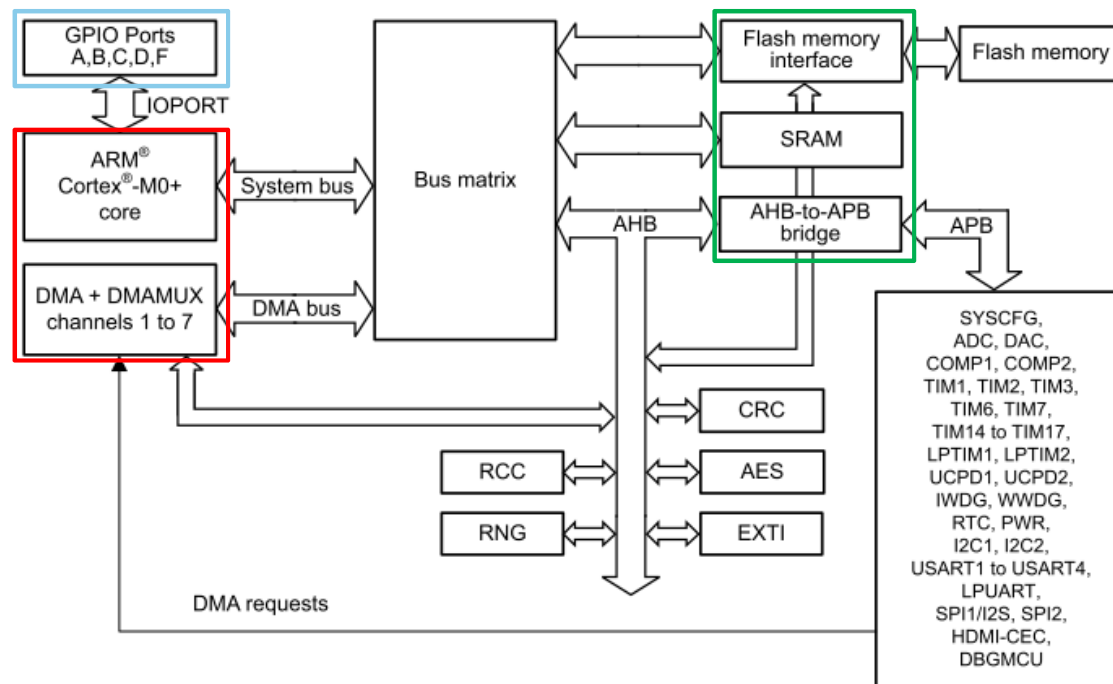
- 1.65V to 3.6V
- 0.93 DMIPS/MHz
- **Vbat** supply
- Vref+ pin
- **Max ambient temp 125°C**
- One Supply pair
- Securable Memory Area
- High sink I/Os
- <100µA/MHz run mode
- **Stand-by** <1µA @ room temperature
- Stop 5 µA @ room temperature
- **Shutdown mode**
- Low EMI SAE (2.5@24MHz)
- Robust EMC/ESD/EMS
- 28/32/48 and 64 pins

features highlight



System architecture overview

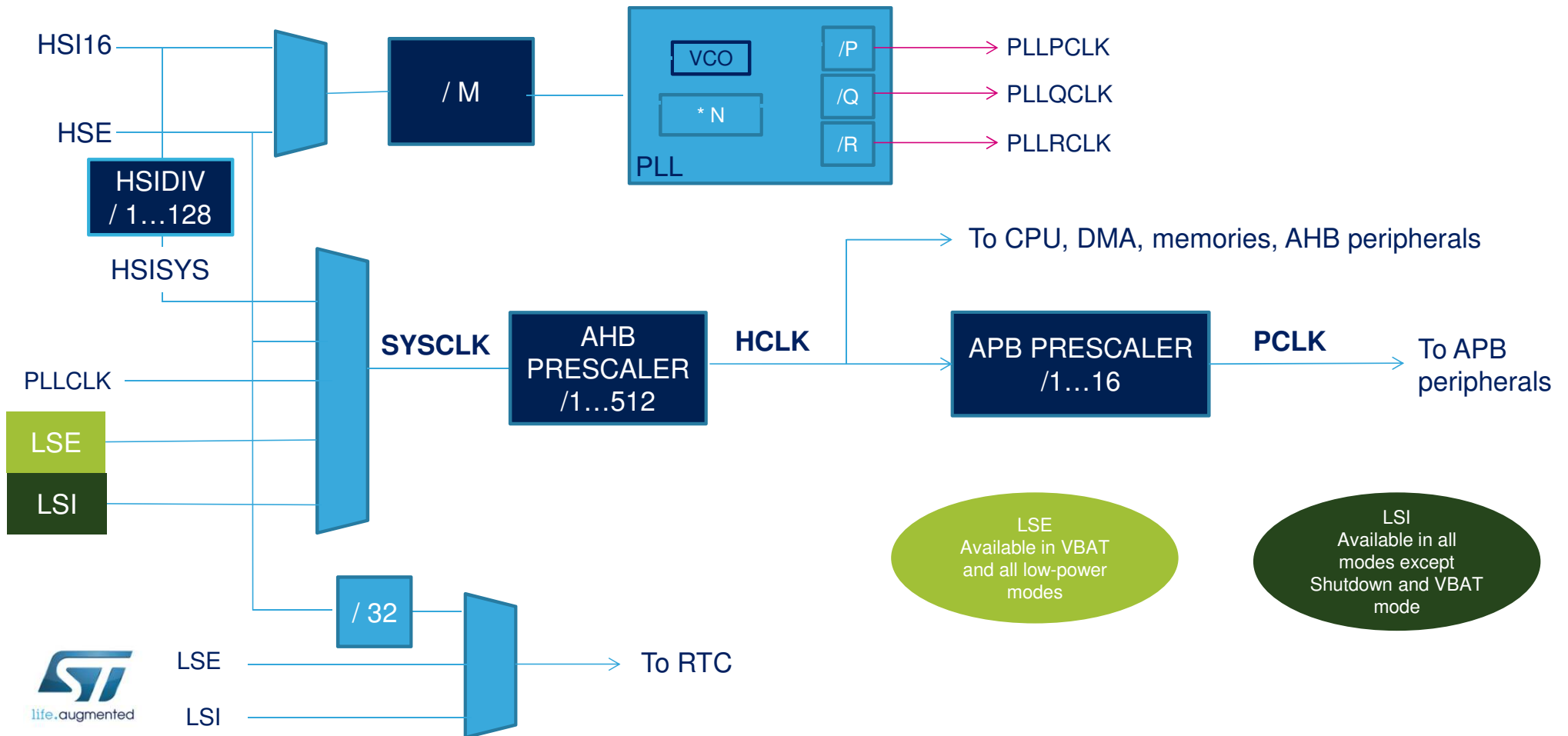
34



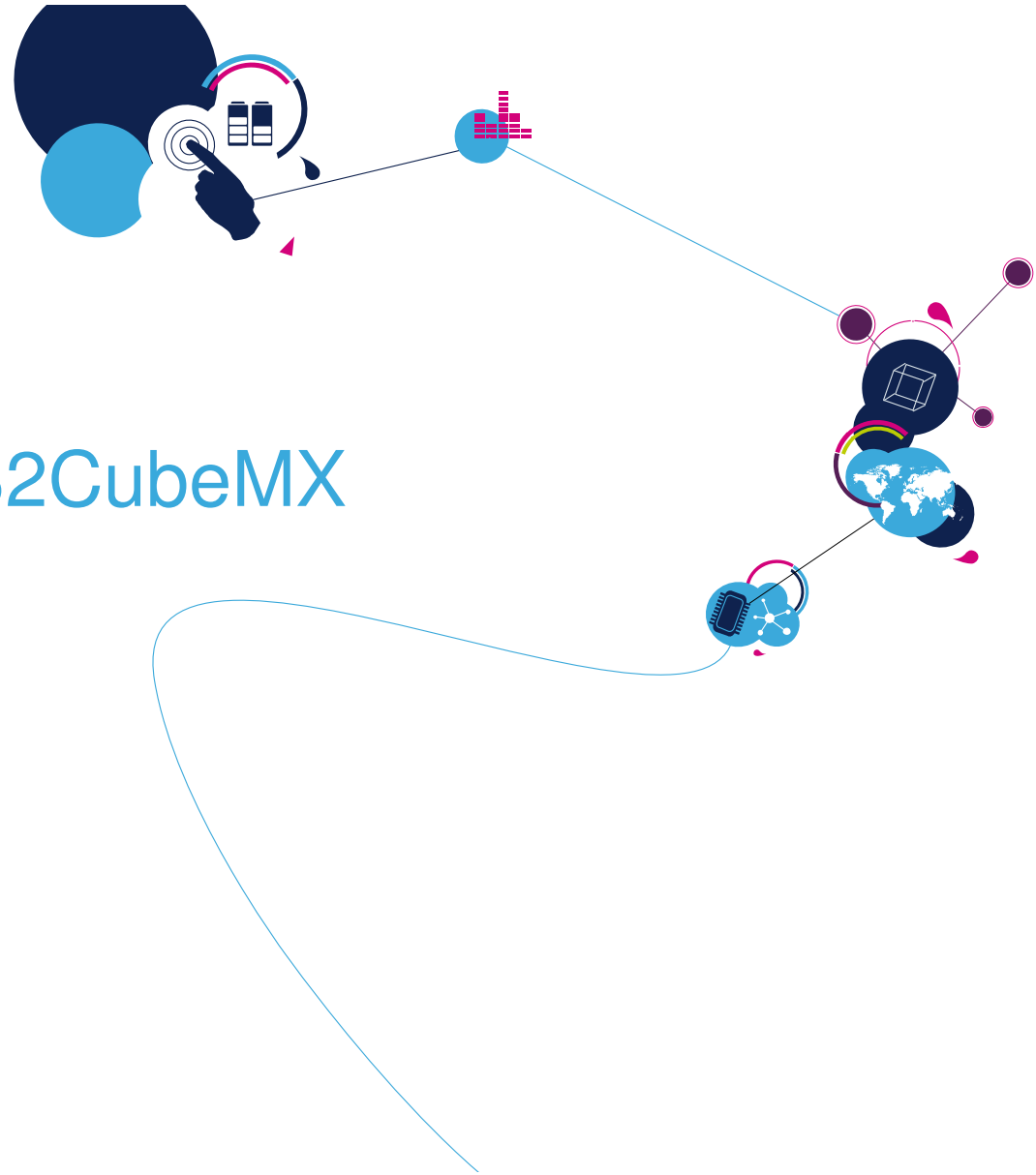
- Two masters:
 - Cortex®-M0+ core
 - General-purpose DMA
- Three slaves:
 - Internal SRAM
 - Internal Flash memory
 - AHB with AHB-to-APB bridge that connects all the APB peripherals
- Dedicated IOPORT for accessing the GPIOs

Simplified clock tree

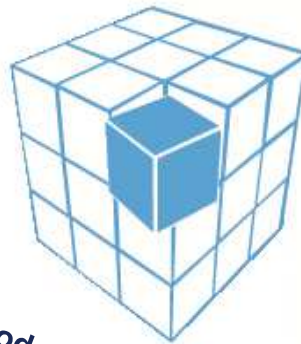
35



STM32Cube and STM32CubeMX



STM32CubeMX



Initialization Code generation based on user choices



STM32CubeL0



STM32CubeL1



STM32CubeL4



STM32CubeF0



STM32CubeF1



STM32CubeF2



STM32CubeF3



STM32CubeF4



STM32CubeF7



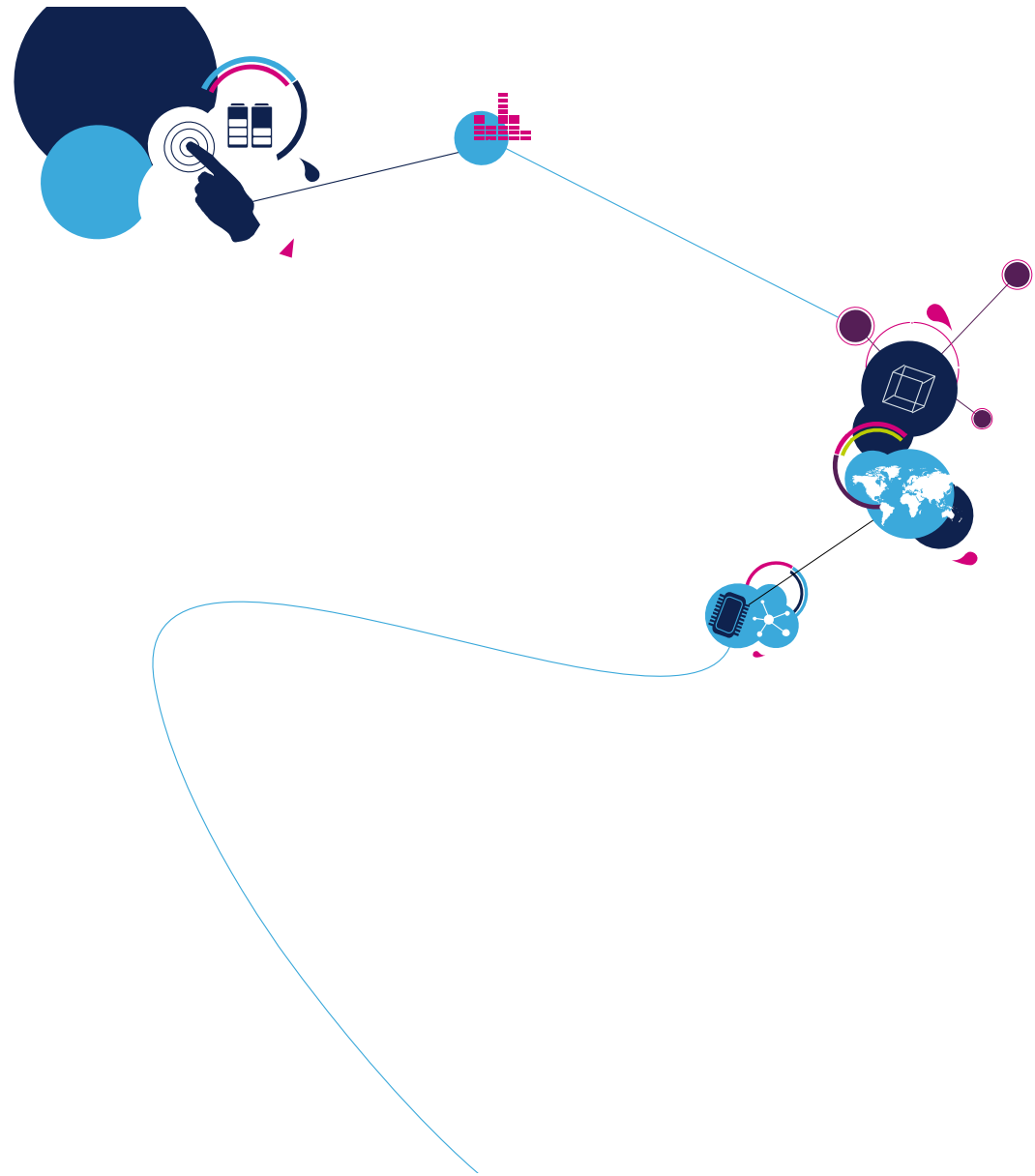
STM32CubeH7

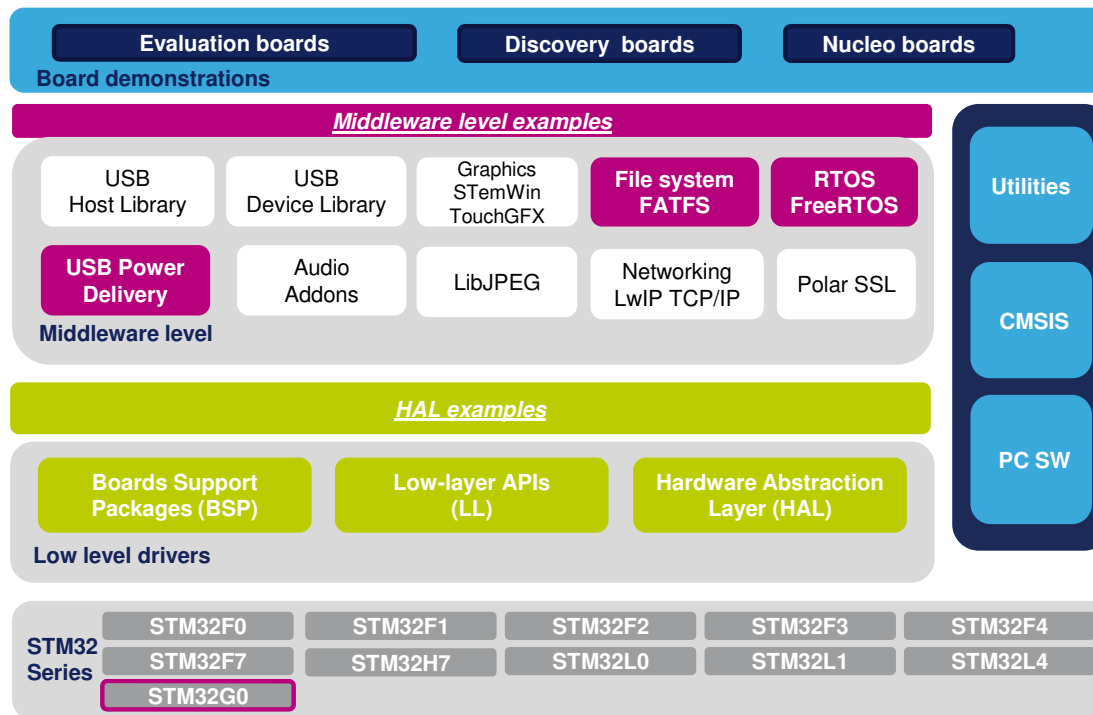


STM32CubeG0

STM32Cube HAL: Portable API within all series - Middleware stacks when applicable: RTOS, USB, TCP/IP, Graphics, ...

STM32Cube

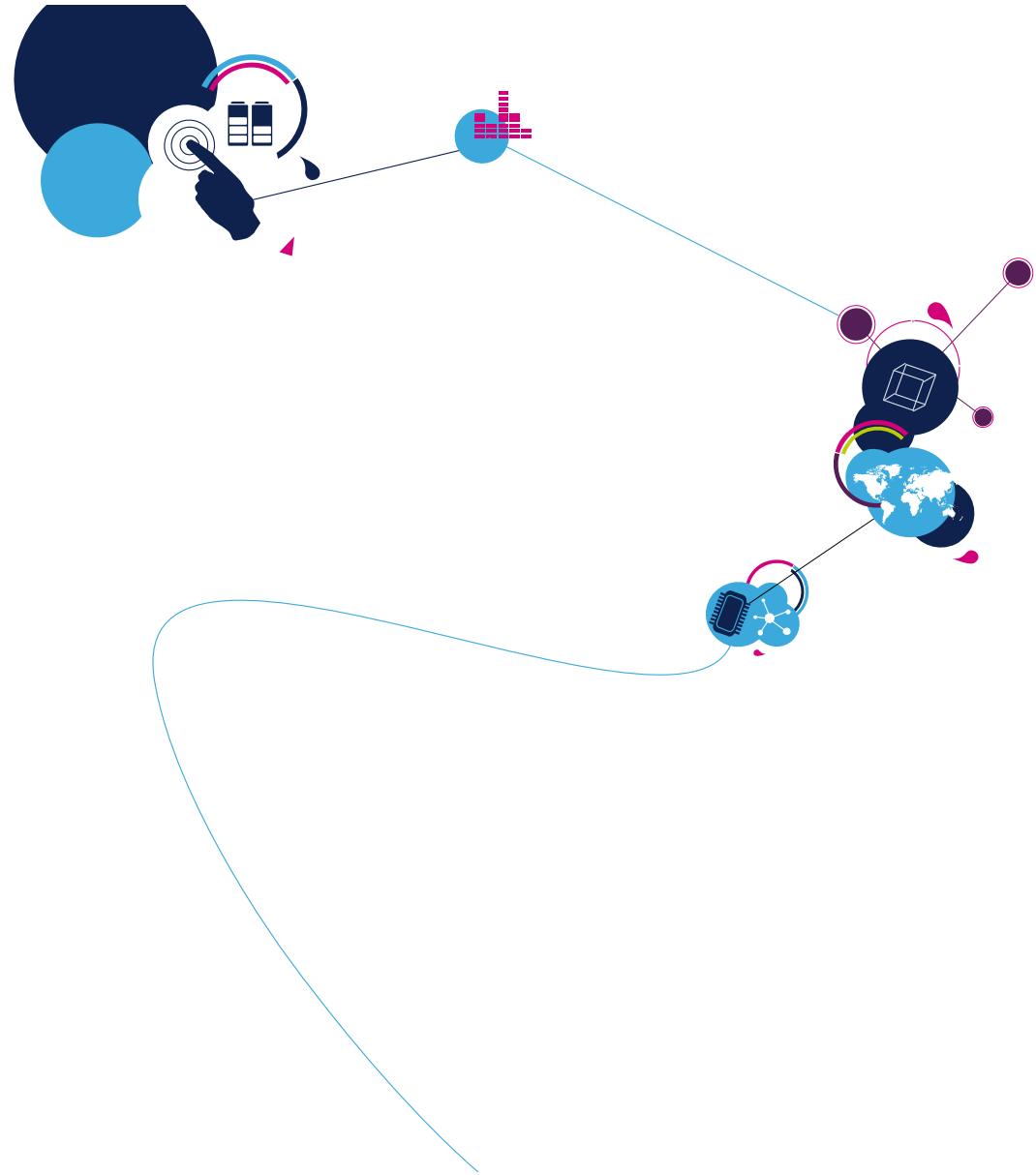




Application benefits

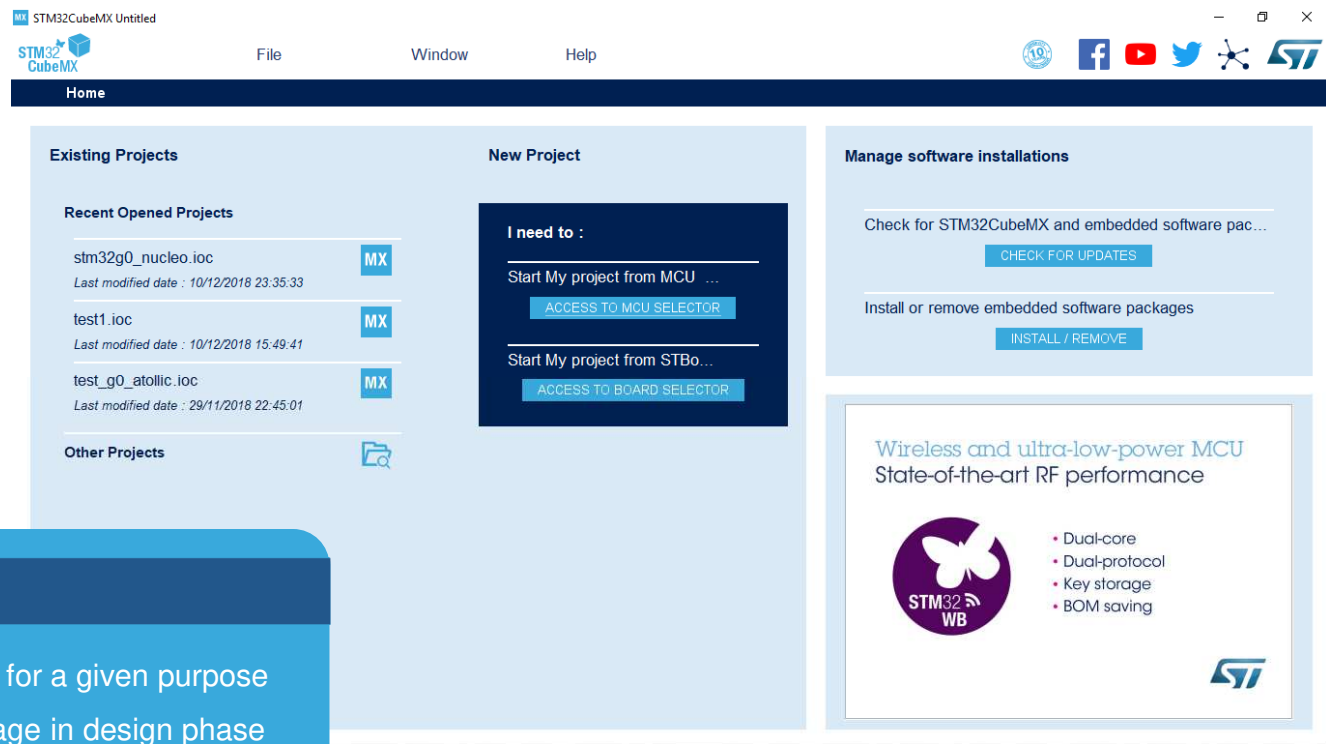
- Single package
- Compatible with all STM32 series
- Source code with open-source BSD license

STM32CubeMX



- Choose ideal MCU and simply configure

- Pinouts
- Clocks and oscillators
- Peripherals
- Low-power modes
- Middleware



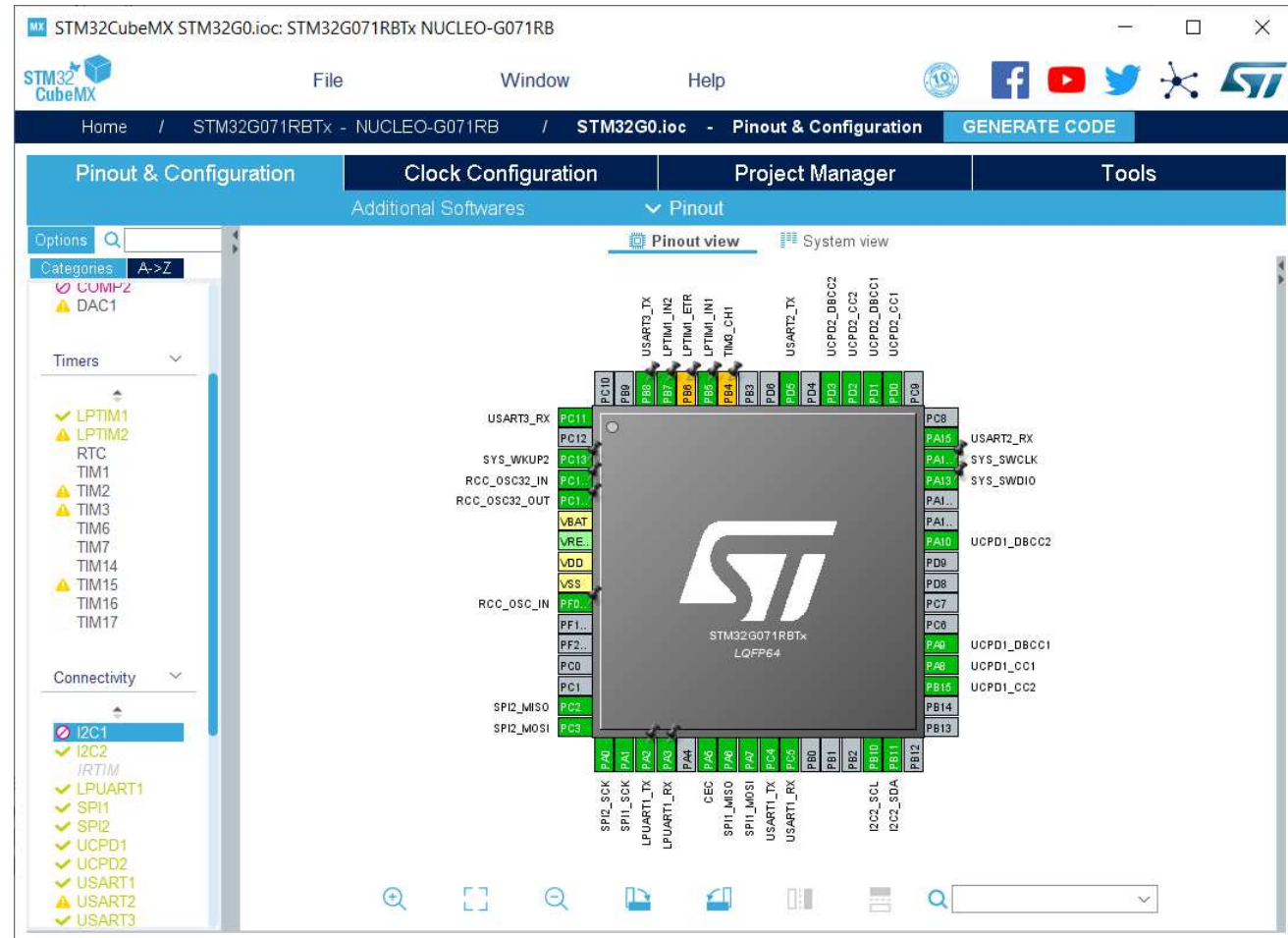
Application benefits

- Helps choose the correct MCU for a given purpose
- Simulation provides an advantage in design phase
- Boosts development speed with a headstart

Pin assignment

42

- Pinout from:
 - Peripheral tree
 - Manually
- Automatic signal remapping
- Management of dependencies between peripherals and/or middleware (FatFS, USB ...)



Peripheral and Middleware configuration

- Global view of used peripherals and middleware

- Highlight of configuration errors

+ Not configured

v OK

⚠ Non-blocking problem

x Error

GPIO configuration is considered incorrect, but code may be generated



The screenshot displays the STM32CubeMX Pinout & Configuration window. The interface includes a top menu bar with File, Window, and Help. Below this is a breadcrumb trail: Home / STM32G071RBTx - NUCLEO-G071RB / STM32G0.ioc - Pinout & Configuration. The main area is divided into tabs: Pinout & Configuration (selected), Clock Configuration, Project Manager, and Tools. A sub-tab bar shows Additional Softwares and Pinout (selected). The Pinout view is active, displaying a list of peripherals and their status. Callouts highlight specific features: 'Click to configure DMA' points to the DMA button; 'Quickly switch pinout and system view' points to the Pinout/System view toggle; 'Error in configuration, code generator will display a warning message' points to the TIM1 button which has a red 'X' icon; and 'Configuration is valid here' points to the LPUART1 button which has a green checkmark icon.

STM32CubeMX STM32G0.ioc*: STM32G071RBTx NUCLEO-G071RB

File Window Help

Home / STM32G071RBTx - NUCLEO-G071RB / STM32G0.ioc - Pinout & Configuration GENERATE CODE

Pinout & Configuration Clock Configuration Project Manager Tools

Additional Softwares Pinout System view

Options Pinout view System view

Categories A-Z

COMP2 DAC1

Timers

LPTIM1 LPTIM2 RTC TIM1 TIM2 TIM3 TIM6 TIM7 TIM14 TIM15 TIM16

System Core Analog Timers Connectivity Multimedia Computing

DMA GPIO NVIC RCC

LPTIM1 LPTIM2 TIM1

I2C2 LPUART1 SPI1 SPI2 UCPD1 UCPD2 USART1 USART2 USART3

HDMI_CEC CRC

Click to configure DMA

Quickly switch pinout and system view

Error in configuration, code generator will display a warning message

Configuration is valid here

Click to
configure DMA

Quickly switch
pinout and
system view

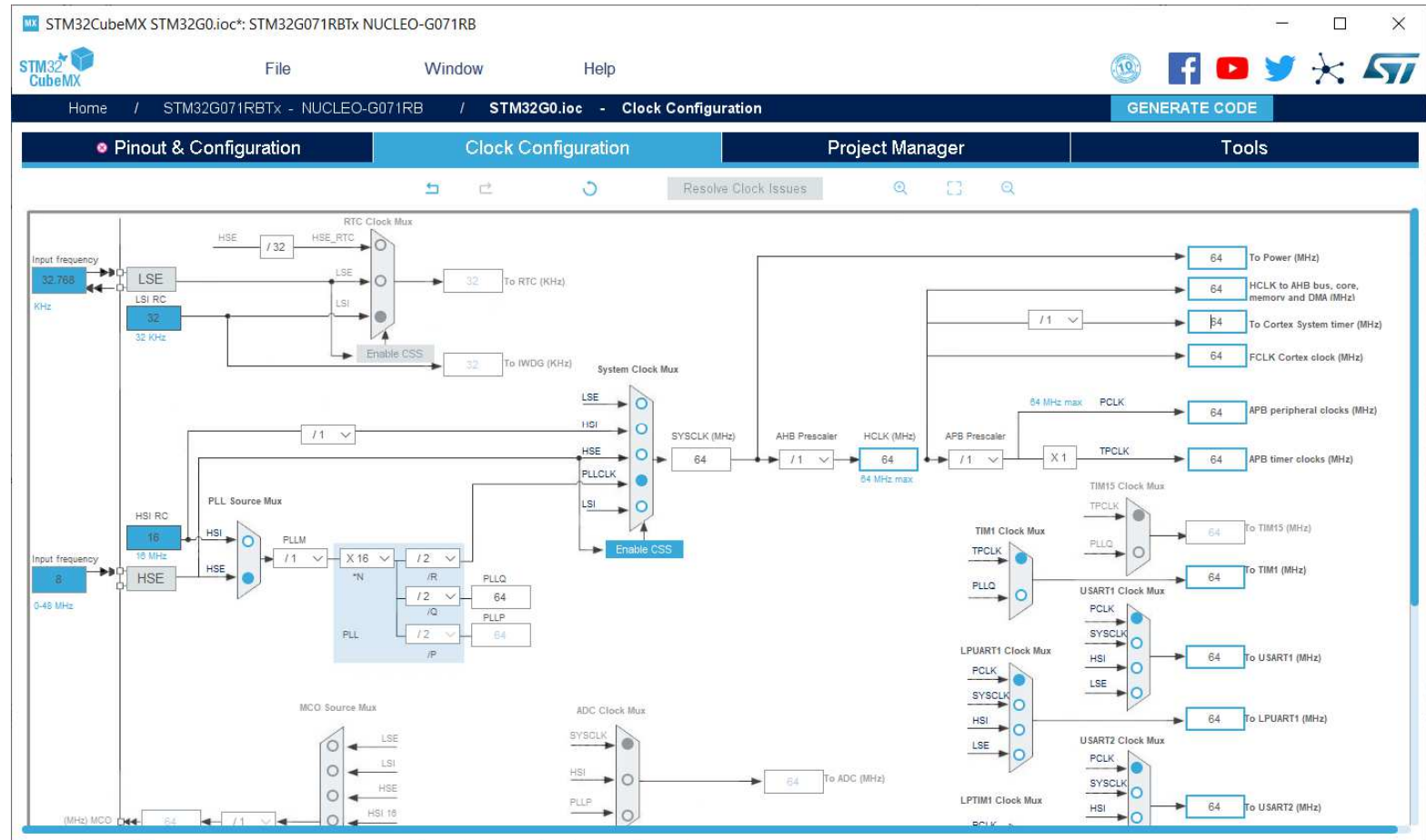
Configuration
is valid here

Error in configuration, code generator will display a warning message

Clock configuration

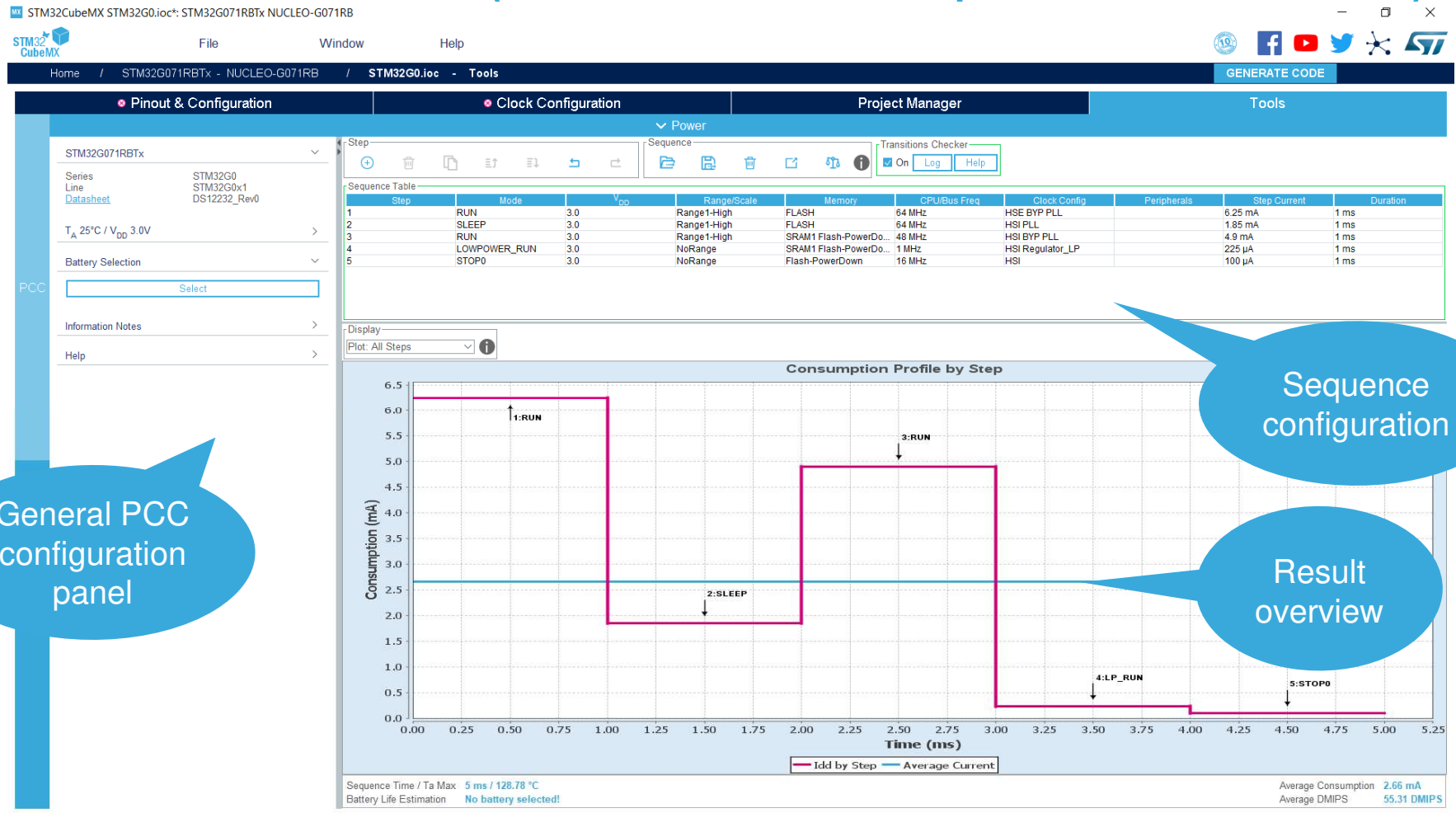
44

- Immediate display of all clock values
- Active and inactive clock paths are differentiated
- Management of clock constraints and features




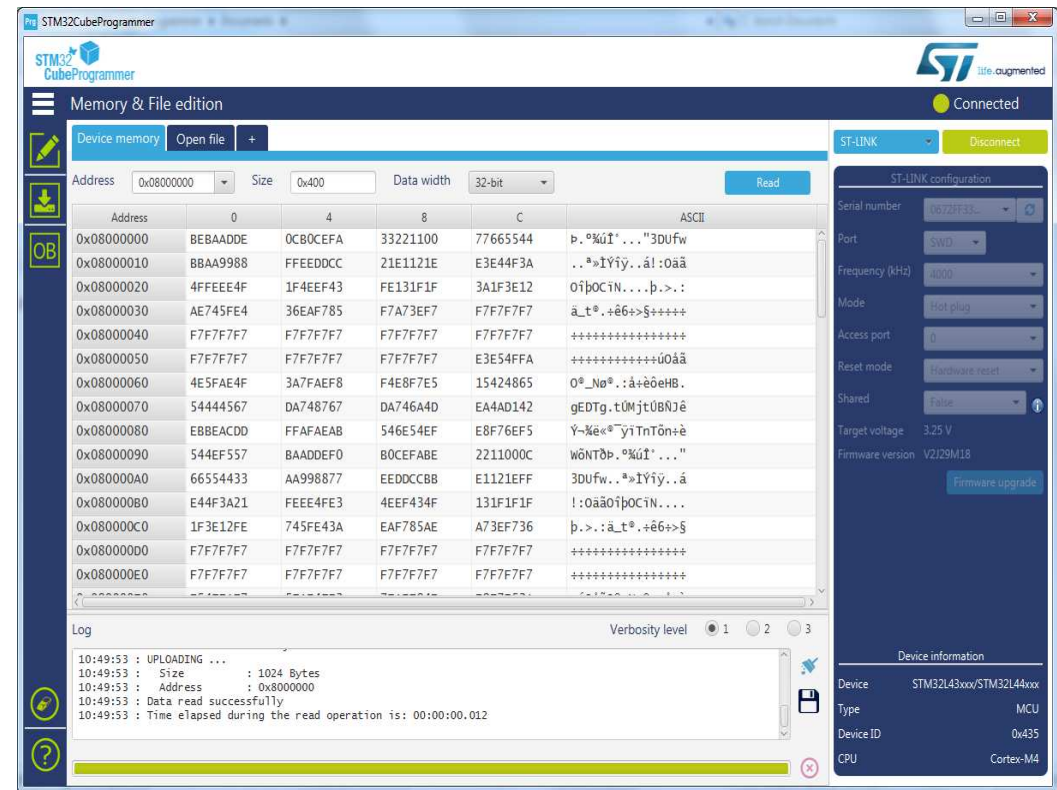
PCC (Power Consumption Calculator)

45



• Main Features:

- Unify existing programming software tools : Merge STVP, ST-Link Utility and Bootloader softwares tools in one solution.
- Multiplatform (Windows, Linux and macOS) 
- Debug and bootloader interfaces support: ST-LINK debug probe (JTAG/SWD), Bootloader interfaces (UART, USB DFU, SPI, I²C and CAN)
- Secure programming



Finish the tools installation

47

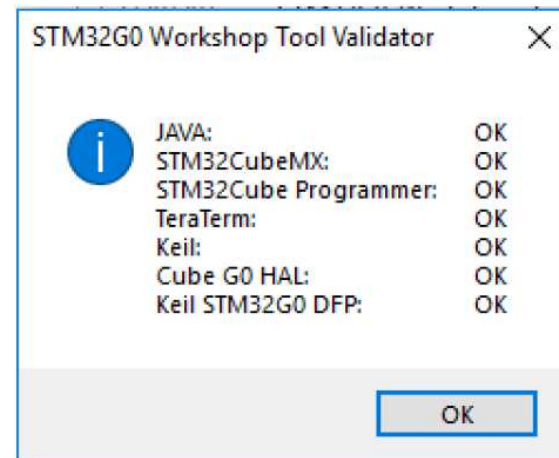
- Install STM32CubeMX
- Install STM32CubeProgrammer
- Install STM32CubeG0 FW Library
- Tera Term

For more details on the tools installation, refer to the document “**STM32G0 Workshop Installation from USB drive_v1.0.pdf**” part of the files you copied from the USB drive

Validate the Tools Installation

48

- Run The Tool Validator as administrator (part of the files you copied from the USB drive) to validate the tools installation.
- When running the tool you should get:

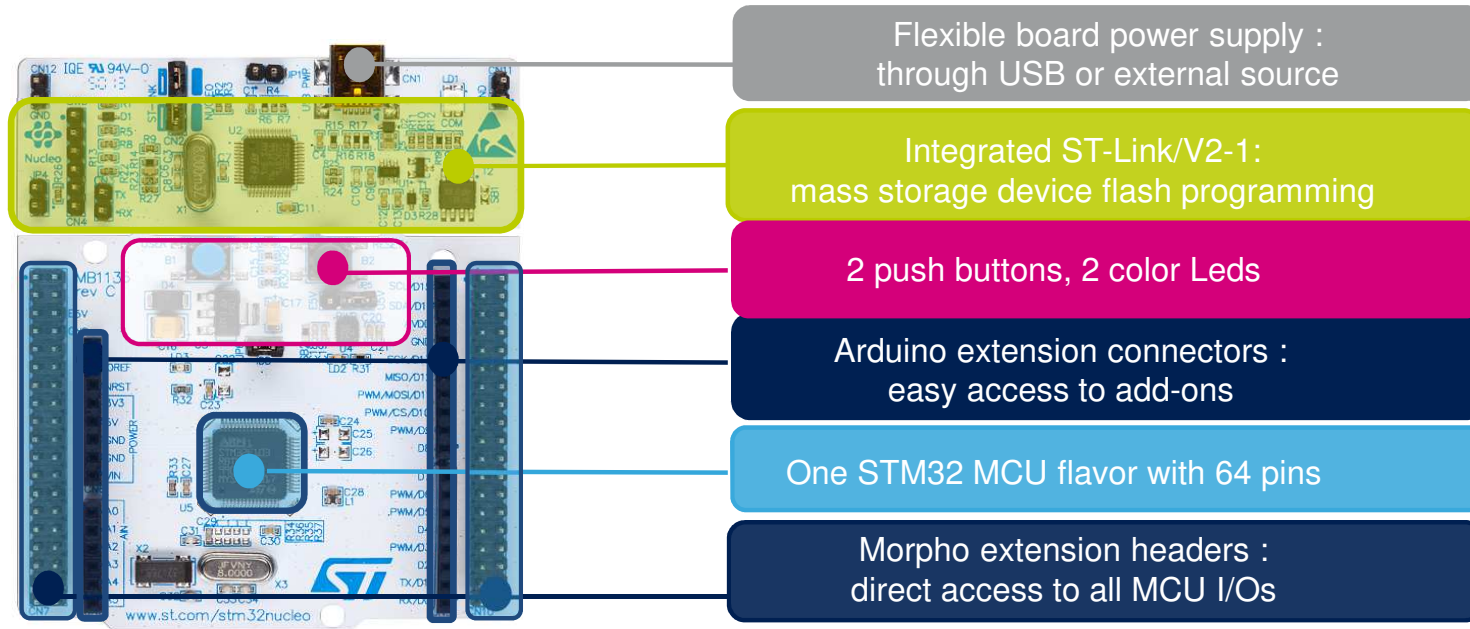


- **If you don't, please let us know**



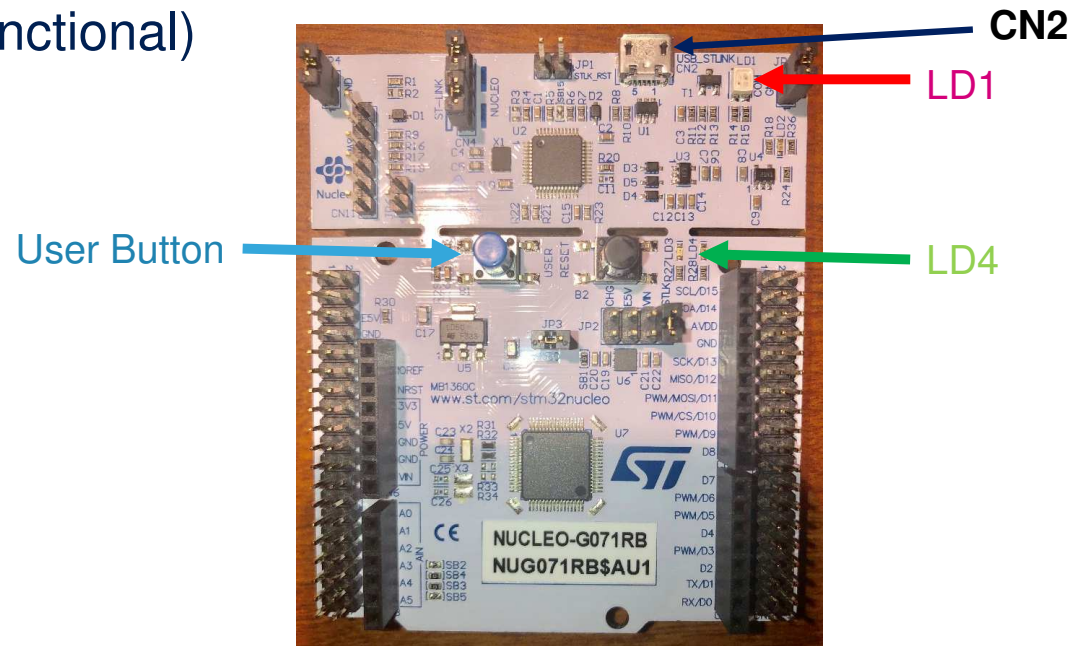
STM32 NUCLEO features

49

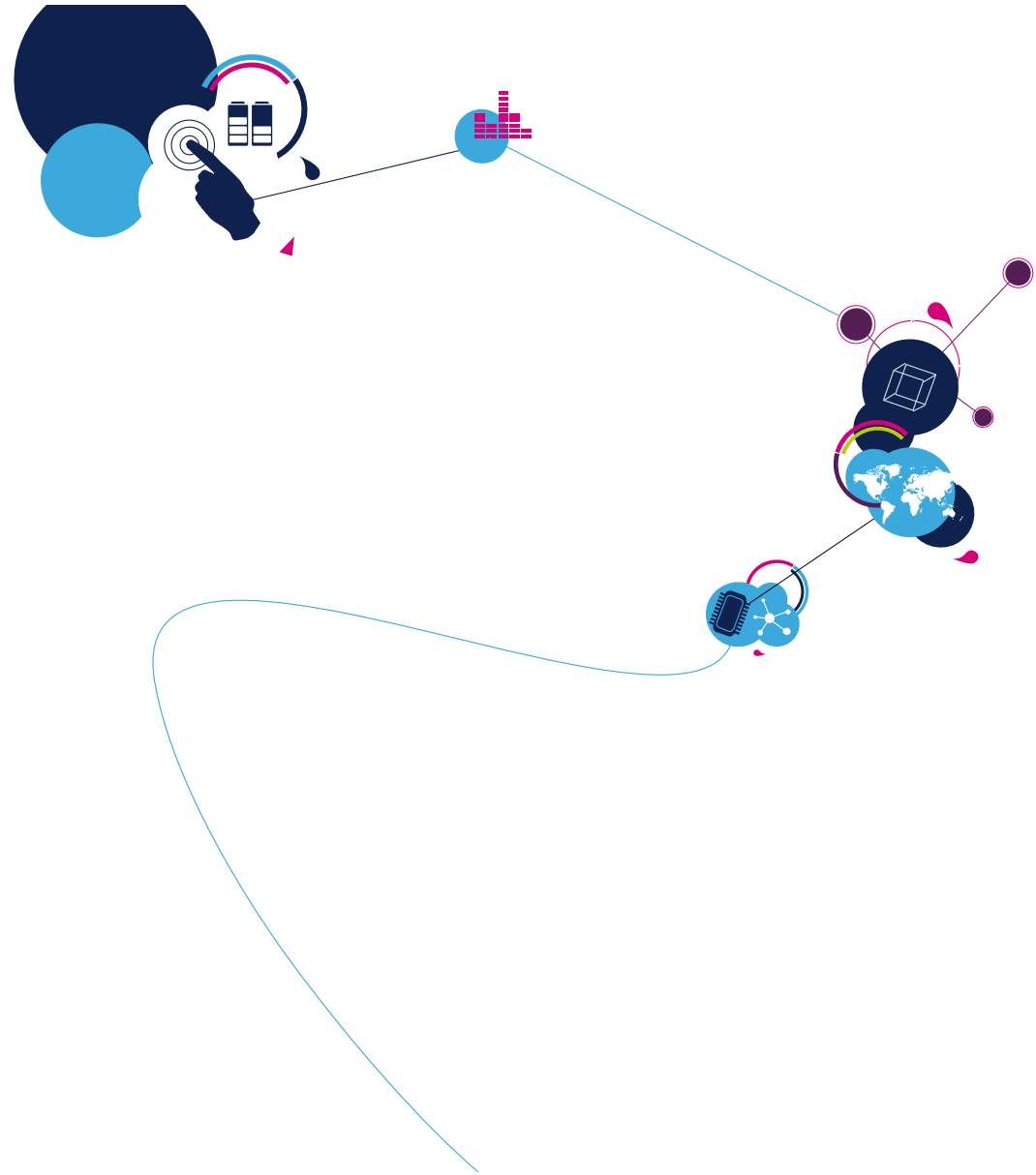


We are now going to provide you with the STM32G0 Nucleo board and the USB cable.

- Connect USB ST-LINK (**CN2**) to your PC
 - ST-LINK driver may be installed if this is the first time the board is plugged in.
- **LD1** should be **ON** and solid **RED** (indicating board power available and ST-Link is functional)



Lab: Blinky



Objective:

- The objective of this lab is to generate a very simple project using STM32CubeMX Software.
- In this example we are going to blink one of the LEDs present on the STM32G0 Nucleo board, connected to PA5 of the STM32G0 MCU.

Run STM32CubeMX

53

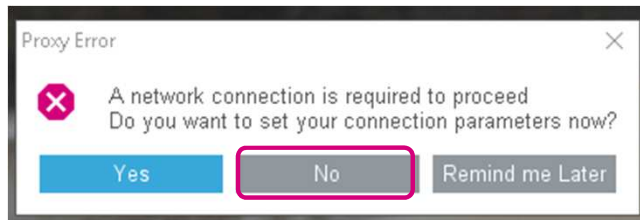
1

Double Click on STM32CubeMX icon on your desktop



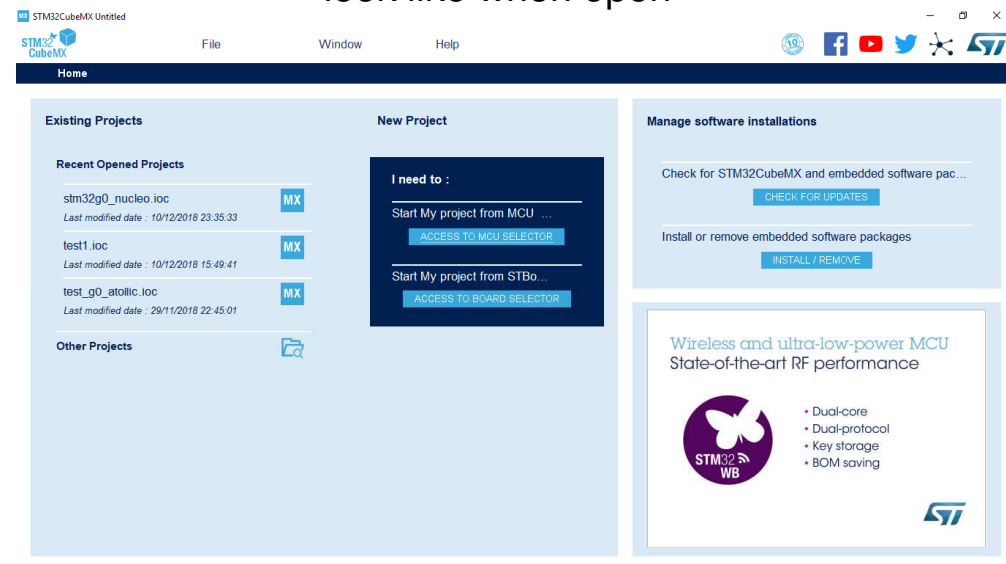
2

Click "No" when this window appears



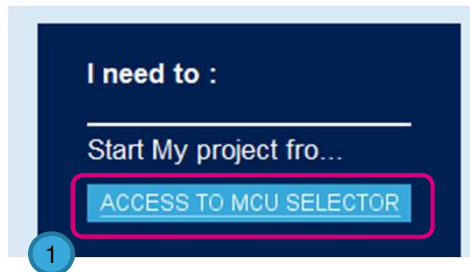
3

This is how STM32CubeMX will look like when open

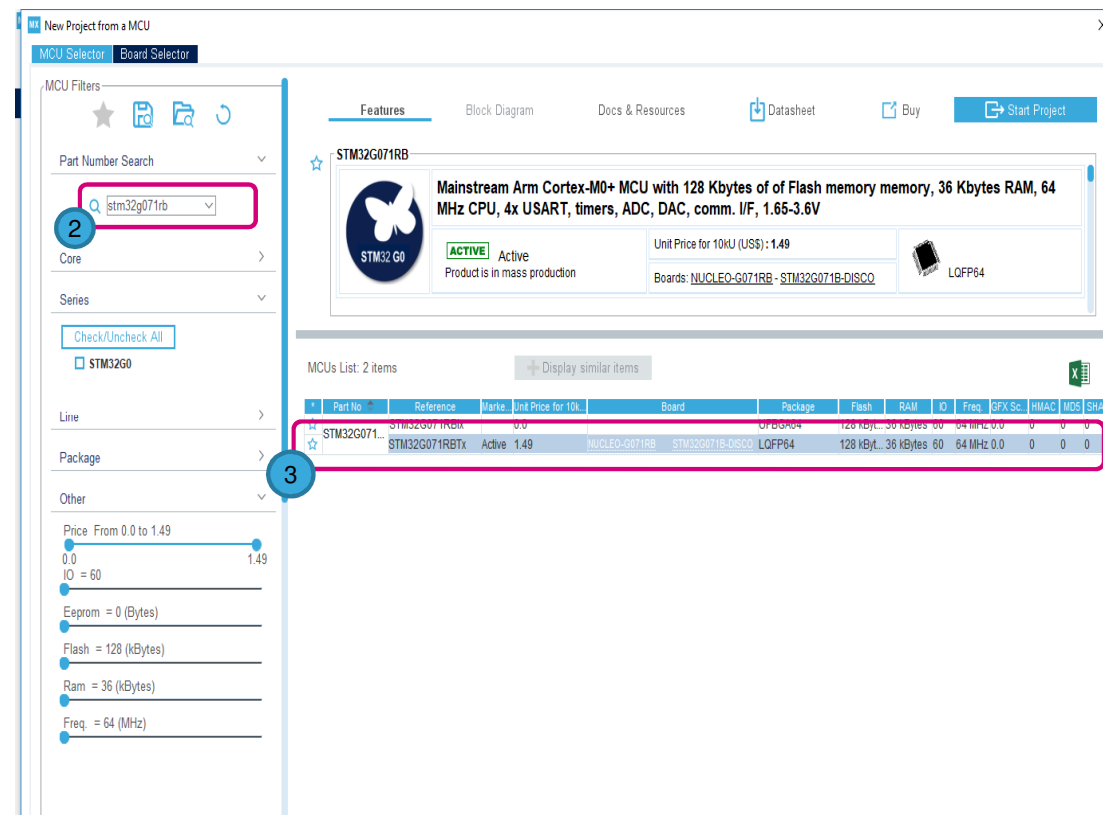


Step 1: Create New Project

54



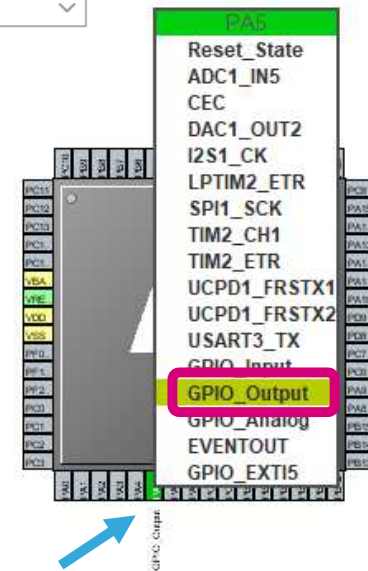
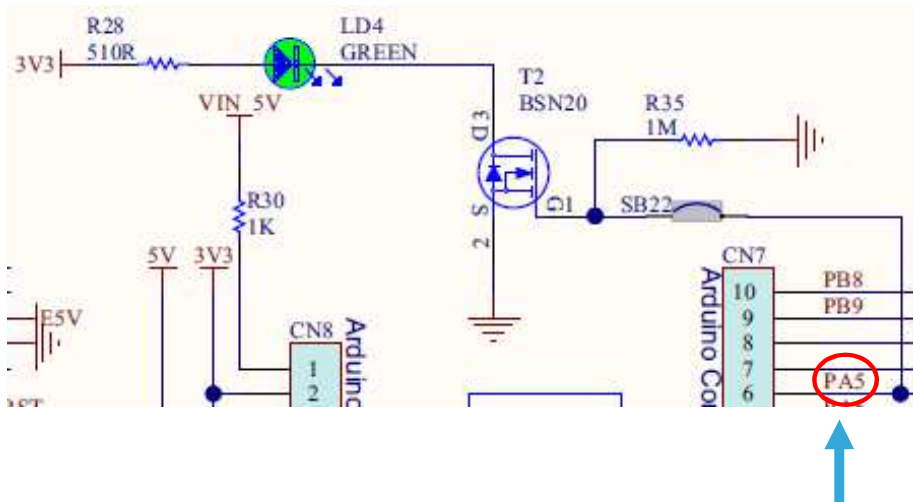
- Click **Access To MCU Selector** 1
- Type: “stm32g071rb” in the Part Number Search 2
- Select **STM32G071RBTx** 3
 - LQFP64, 128kB Flash
- Double Click



Step 2: Pin Configuration

55

- In this example we are going to use one of the LEDs present on the STM32G0 Nucleo board (connected to PA5 as seen in the schematic below)
- Search for **PA5** in the search window at the bottom right
- Left-click **PA5** and set it to **GPIO_Output** mode



Step 3: Generate Source Code

56

- Open **Project Manager**



- Set the project name (**blinky**) and the project location (**C:\STM32G0Workshop\HandsOn**)

1

2

- Set the IDE Toolchain to **MDK-ARM V5**

3

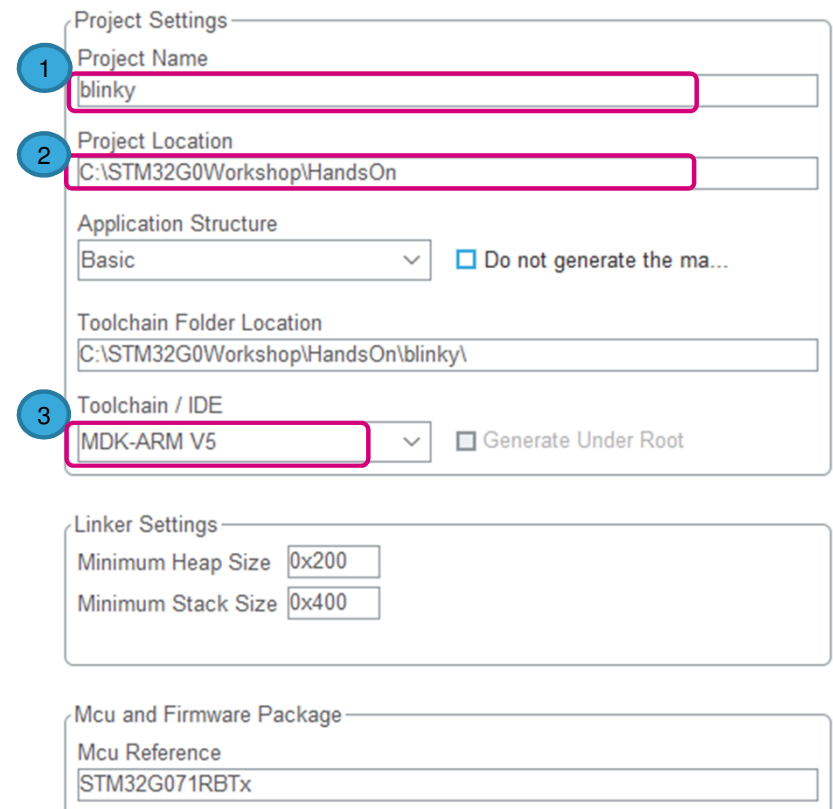
- **Generate Code**



- Click **Open Project**



Note: STM32CubeMX projects have an *ioc* file extension



Project Settings

1 Project Name
blinky

2 Project Location
C:\STM32G0Workshop\HandsOn

Application Structure
Basic ☐ Do not generate the ma...

Toolchain Folder Location
C:\STM32G0Workshop\HandsOn\blinky\

3 Toolchain / IDE
MDK-ARM V5 ☐ Generate Under Root

Linker Settings

Minimum Heap Size 0x200

Minimum Stack Size 0x400

Mcu and Firmware Package

Mcu Reference
STM32G071RBTx

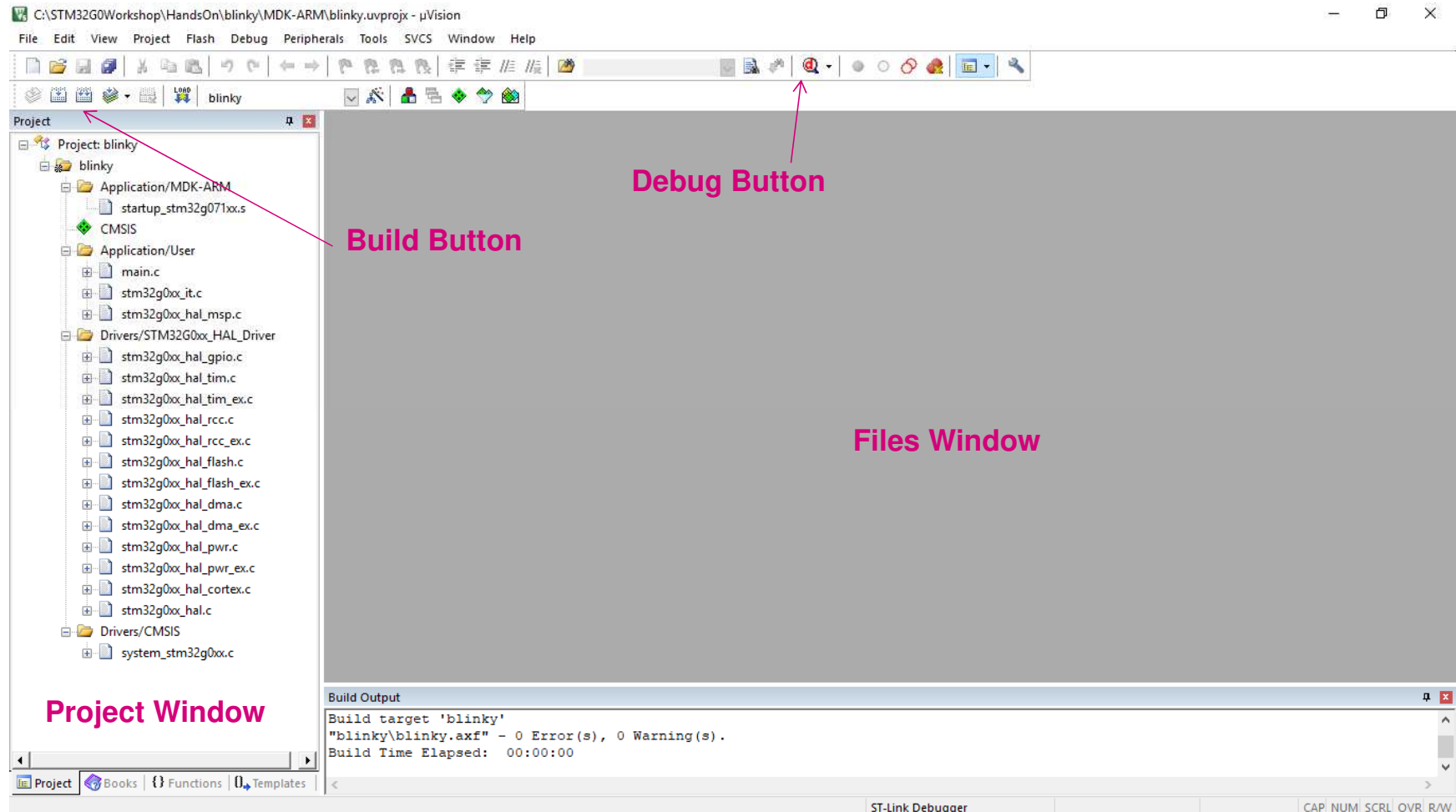
- Free licenses for STM32 devices based on Cortex-M0/M0+ cores :
 - Applicable immediately to all **STM32G0**, STM32F0 and STM32L0 MCUs.
 - PC-locked multi-year licenses.
 - No code size limit.
 - Multiple language support.
 - Technical support included.
- Direct download from Keil website :
 - No limit of number of downloads by customer.
 - Direct access to configuration files for STM32 and associated boards.
 - Free access to MDK-ARM periodic updates.

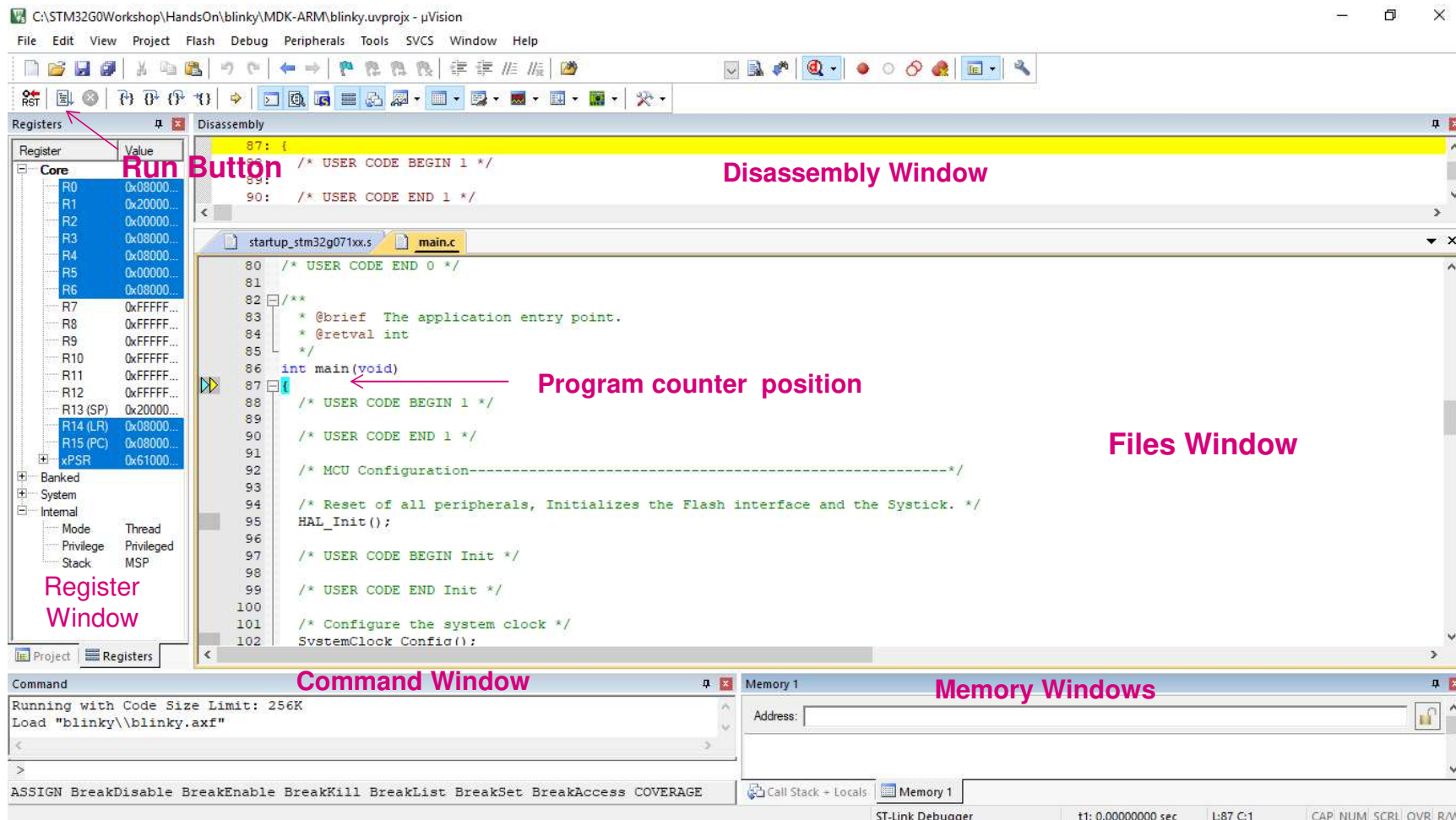
- To get a free MDK-ARM license for STM32F0, STM32L0 and **STM32G0**:

- Go to Keil website at : www.keil.com/mdk-st
- Download MDK-ARM tool chain.
- Activate the free license using this Product Serial Number (PSN) :
4PPFW-QBEHZ-M0D5M

Inside Keil uVision (ARM-MDK)

58





The screenshot displays the KEIL MDK-ARM IDE Debugger interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, debugging, and viewing. The main workspace is divided into several windows:

- Registers Window:** Located on the left, it shows a list of registers (R0 to R15, xPSR) and their values. A red arrow points to the 'Run' button (a green play icon) in the toolbar.
- Disassembly Window:** Located at the top right, it shows the disassembled code for the selected file (main.c). A red arrow points to the 'Program counter position' (PC) register value (0x00000000) in the Registers window.
- Files Window:** Located in the center, it shows the source code for the selected file (main.c). A red arrow points to the 'Program counter position' (PC) register value (0x00000000) in the Registers window.
- Command Window:** Located at the bottom left, it shows the command prompt with the text: "Running with Code Size Limit: 256K", "Load 'blinky\\blinky.axf'", and "ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE".
- Memory Windows:** Located at the bottom right, it shows the memory address and its contents.

Annotations in red text highlight the 'Run Button', 'Disassembly Window', 'Program counter position', 'Files Window', 'Register Window', 'Command Window', and 'Memory Windows'.

Step 4: Toggle The LED

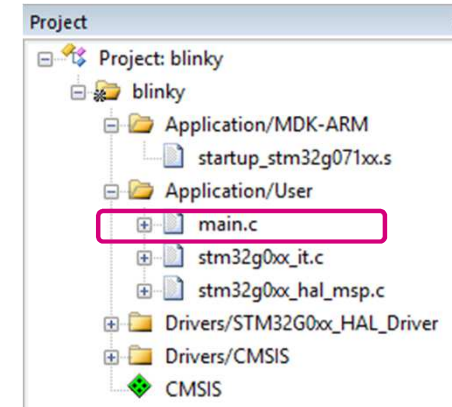
60

In the Keil uVision5 IDE:

- Expand the file tree and open the **main.c** file
- Add the following code **inside the while(1) loop** (~Line 118(*) in “main.c”)

```
HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);  
HAL_Delay(100);
```

```
114  /* Infinite loop */  
115  /* USER CODE BEGIN WHILE */  
116  while (1)  
117  {  
118      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);  
119      HAL_Delay(100);  
120  /* USER CODE END WHILE */
```



Note: Code within the “USER CODE BEGIN WHILE” / “USER CODE END WHILE” section will be preserved after regeneration by STM32CubeMX.

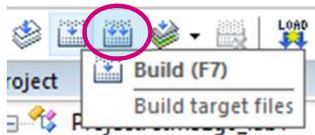
TIP: The code to be added for the labs are located in a text called “code_to_add_vx.x.txt”

(*) Line numbers throughout the presentation are given for STM32CubeMX v5.0.1 and STM32CubeG0 Library v1.0.0 and may vary with other versions.

Step 5: Build the Project and Debug

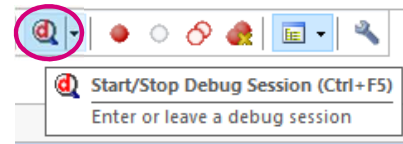
61

- Click the “Build” button (F7)



```
Build target 'blinky'  
"blinky\blinky.axf" - 0 Error(s), 0 Warning(s).  
Build Time Elapsed: 00:00:00
```

- Click the “Start/Stop Debug Session” button (Ctrl + F5)



Step 5: Build the Project

62

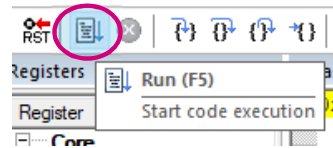
- If you see this warning message due to a minor Syntax error in the startup file, just press OK to continue.



Note: To correct the Syntax error in the startup_stm32g071x.s:

- Remove "<h2><center>©" from line 17.
- Remove "</center></h2>" from line 18.

- Click the "Run" button (F5)



- Enjoy the flashing Green LED (LD4)!

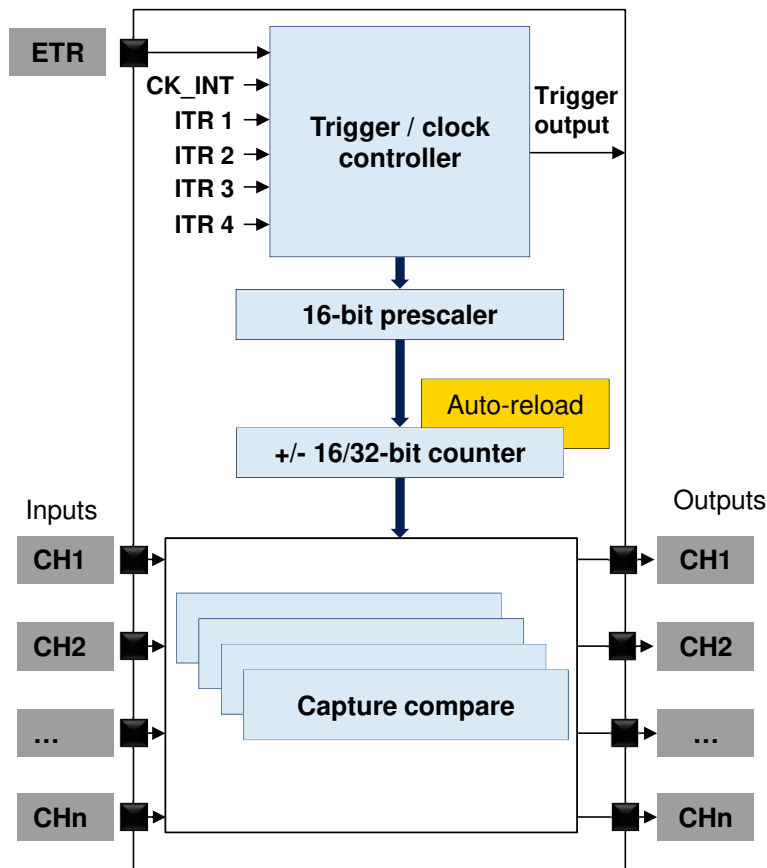
Lab: PWM (Pulse Width Modulation) Timer

Objective:

- Now let's use a more advanced peripheral like the Timer.
- In this lab we are going to configure a Timer in a PWM mode to blink the LED that we previously controlled with a GPIO.
- PA5 has an alternate Timer channel alternate function which is Timer 2 Channel 1: TIM2_CH1 that we will be using.

Timer - Overview

65



- Multiple timer units providing timing resources

- Internally (triggers and time-bases)
- Externally, for outputs or inputs:
 - For waveform generation (PWM)
 - For signal monitoring or measurement (frequency or timing)

Application benefits

- Versatile operating modes reducing CPU burden and minimizing interfacing circuitry needs
- A single architecture for all timer instances offers scalability and ease-of-use
- Also fully featured for motor control and digital power conversion applications

STM32G0 timer instance features

66

Feature		TIM1 (Advanced Control)	TIM2	TIM3	TIM6	TIM7	TIM14	TIM15	TIM16	TIM17
			(General-Purpose)		(Basic)		(General-Purpose)			
Clock source		CK_INT External input pin External trigger input ETR	CK_INT External input pin External trigger input ETR Internal trigger inputs		CK_INT		CK_INT	CK_INT External input pin Internal trigger inputs	CK_INT External input pin	
Resolution		16-bit	32-bit		16-bit		16-bit	16-bit		
Prescaler		16-bit								
Counter direction		Up, Down, Up&Down	Up, Down, Up&Down		Up		Up	Up		
Repetition counter		✓	-		-		-	✓		
Synchronization	Master	✓	✓		✓		-	✓		
	Slave	✓	✓		-		-	✓	-	
Number of channels		6: ➤ CH1/CH1N ➤ CH2/CH2N ➤ CH3/CH3N ➤ CH4 ➤ CH5 and CH6 output only, not available externally	4: ➤ CH1 ➤ CH2 ➤ CH3 ➤ CH4		0		1: ➤ CH1	2: ➤ CH1/CH1N ➤ CH2	1: ➤ CH1/CH1N	
Trigger input		✓	✓							

STM32G0 timer instance features

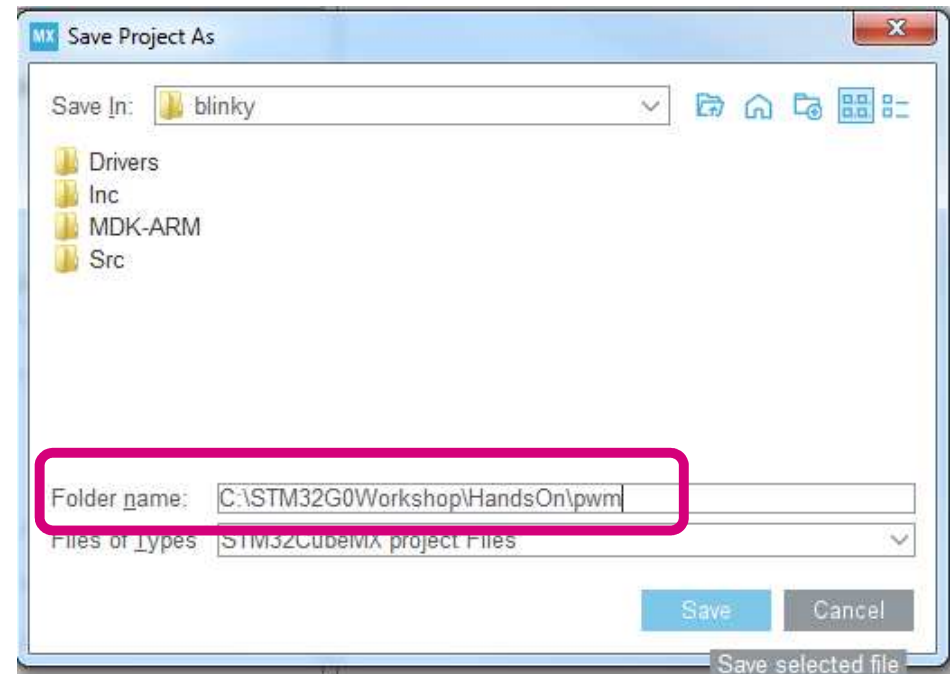
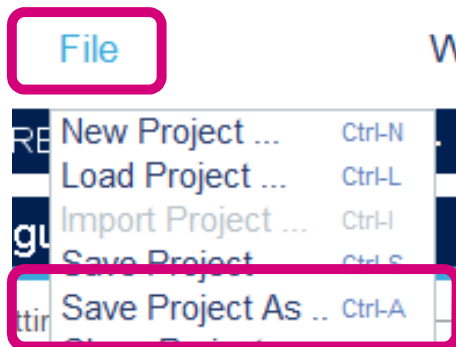
67

Feature	TIM1 (Advanced Control)	TIM2	TIM3	TIM6	TIM7	TIM14	TIM15	TIM16	TIM17
Input capture mode	✓	✓		-		✓		✓	
PWM input mode	✓	✓		-			✓		-
Forced output mode	✓	✓		-		✓		✓	
Output compare mode	✓	✓		-		✓		✓	
PWM	Standard Asymmetric Combined Combined 3-phase 6-step PWM	Standard Asymmetric Combined		-		Standard	Standard Asymmetric Combined		Standard
Programmable dead-time	✓ (CH1-3)	-		-		-	✓ (CH1)		-
Break inputs	2 bidirectional	0		0		0		1 bidirectional	
One-Pulse Mode	✓	✓		-		✓		✓	
Retriggerable one pulse mode	✓	✓		-		-	✓		-
Encoder interface mode	✓	✓		-		-		-	
Timer input XOR function	✓	-		-		-	✓		-
DMA	✓	✓		✓		-		✓	

Lab: Rename the project

68

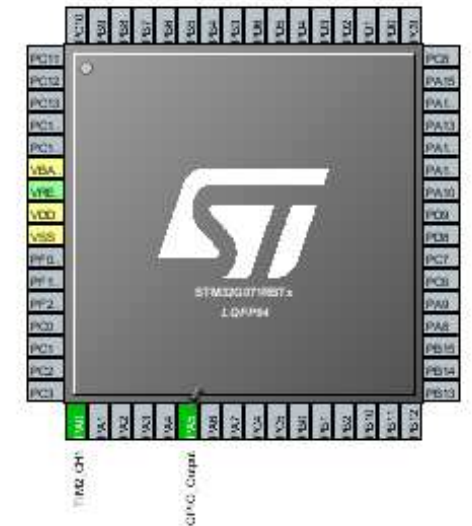
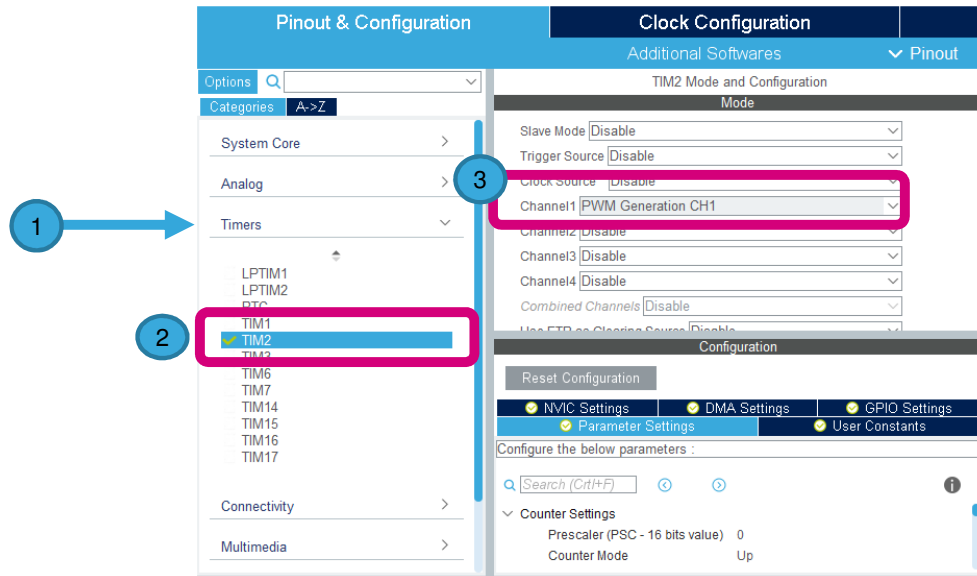
- Close Keil uVision5 IDE if it is open
- Open the last STM32CubeMX project (“**blinky**”) (using File->Recent Projects) and save it as a new project name “**pwm**” (using File -> Save Project As)



Lab: Timer 2 CH1 Configuration

69

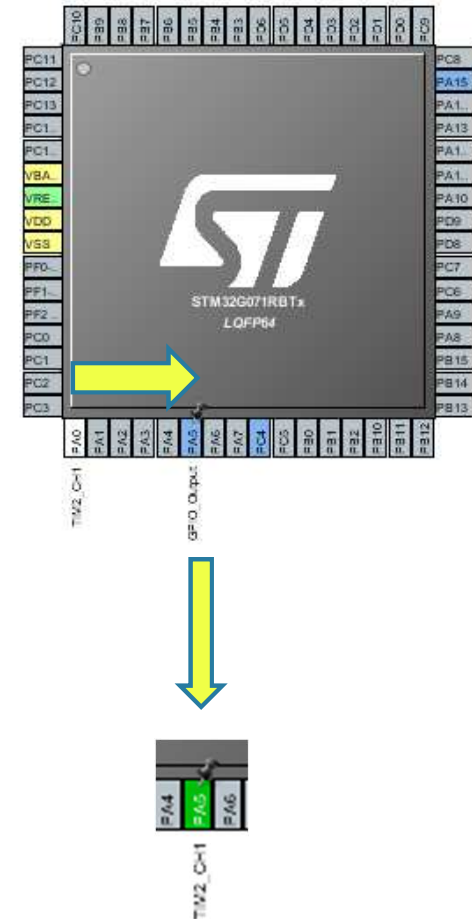
- In this STM32CubeMX project we are going to add Timer 2 Channel 1 to blink LD4 (PA5) on the Nucleo board.
- In the **Pinout & Configuration** tab, Expand **Timers** Categories, then click on **TIM2** peripheral and set Channel1 to “**PWM Generation CH1**”.



Remapping Timer 2 CH1 output to PA5

70

- By default the tool will configure Timer 2 CH1 to **PA0**
- We want to remap it to **PA5**
 - NOTE: PA5 is connected to LD4
- Hold “**Ctrl**” button and left mouse click on **PA0**
- Then drag the mouse pointer to **PA5** and then release



Timer Parameters Calculation

71

- We want the Timer's PWM output channel to be:
 - $T = 1$ second period (1 Hz)
 - $D = 50\%$ duty cycle (0.5)
- Timer input clock frequency (**TPCLK**) is set to 8 MHz.
- **Prescaler** for the Timer is set to **128**. The resulting timer counter clock is:
$$CK_CNT = TPCLK / Prescaler = 8MHz / 128 = 62500 \text{ Hz}$$
- To get $T=1$ Hz (or 1 sec period) the **Counter Period** needs to be set to: **62500**
 - **Counter Period** = $CK_CNT / T = 62500 / 1 = 62500$
- To get $D=50\%$ duty cycle the **Pulse** needs to be set to: **31250**
 - $Pulse = Counter \text{ Period} / 2 = 62500 / 2 = 31250$

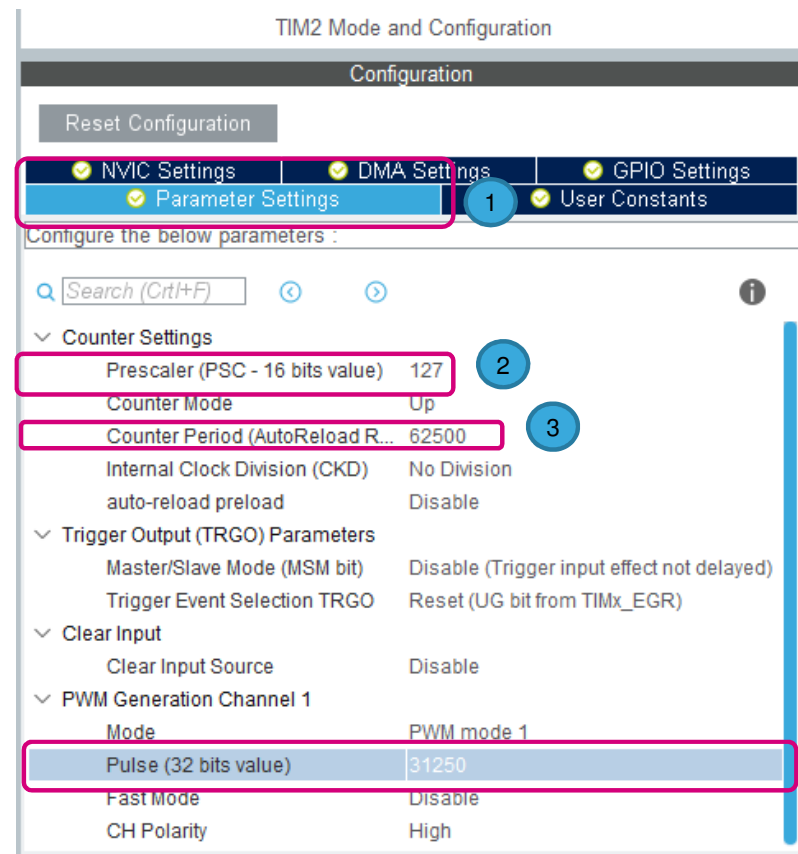
72



TIM2 Configuration – 4 steps

73

- Select the **Pinout & Configuration**
- In **Parameters Settings** of the TIM2 1
- Configure 1 Hz timer
 - PSC Prescaler -1 = 127 2
 - Counter Period = 62500 3
- Set CH1 PWM 4
 - Pulse = 31250



Generate Source Code

74

- **Generate Code**



- Click **Open Project**



- Open the **main.c**, Add the following code before the **while(1)** loop in order to start the PWM Timer:

Note : within "USER CODE BEGIN 2" / "USER CODE END 2" section (~Line 114)

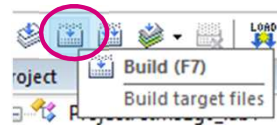
```
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
```

```
113 /* USER CODE BEGIN 2 */  
114 HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);  
115 /* USER CODE END 2 */  
...
```

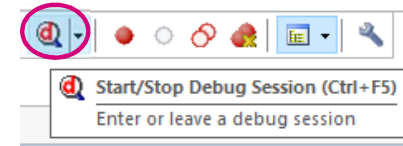
Build the Project

75

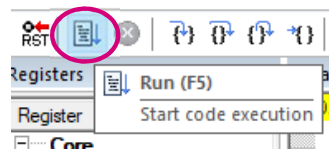
- Click the “Build” button



- Click the “Start/Stop Debug Session” button

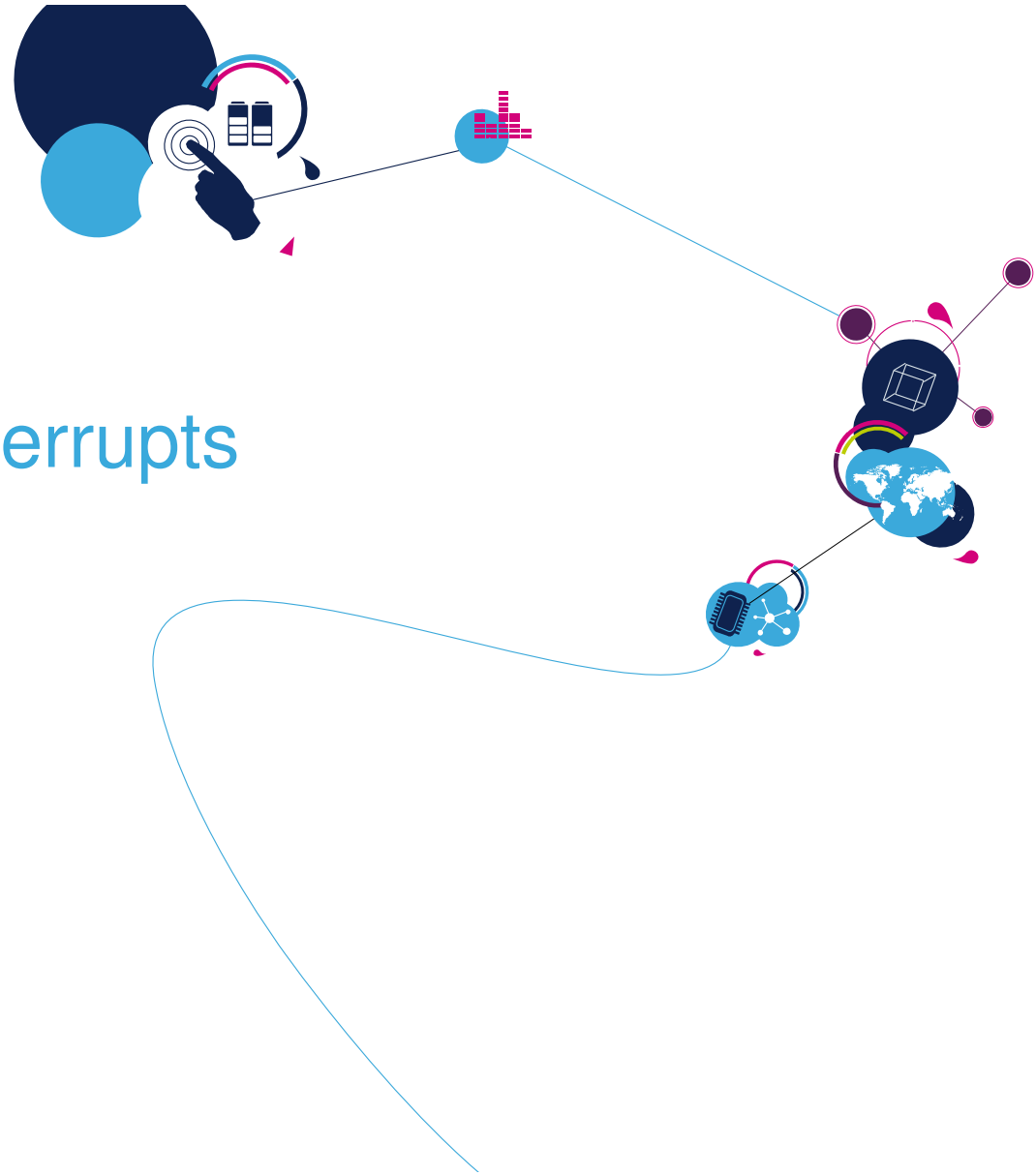


- Click “Run” button



- Enjoy the flashing LED (LD4)!
 - LD4 is flashing using the PWM Timer

Lab: NVIC + External Interrupts



Lab: NVIC + External Interrupts

77

Objective:

- In this project we are going to configure the GPIO that is connected to the user button as External Interrupt (EXTI) with rising edge trigger.
- We will also configure the Interrupt Controller: the NVIC.

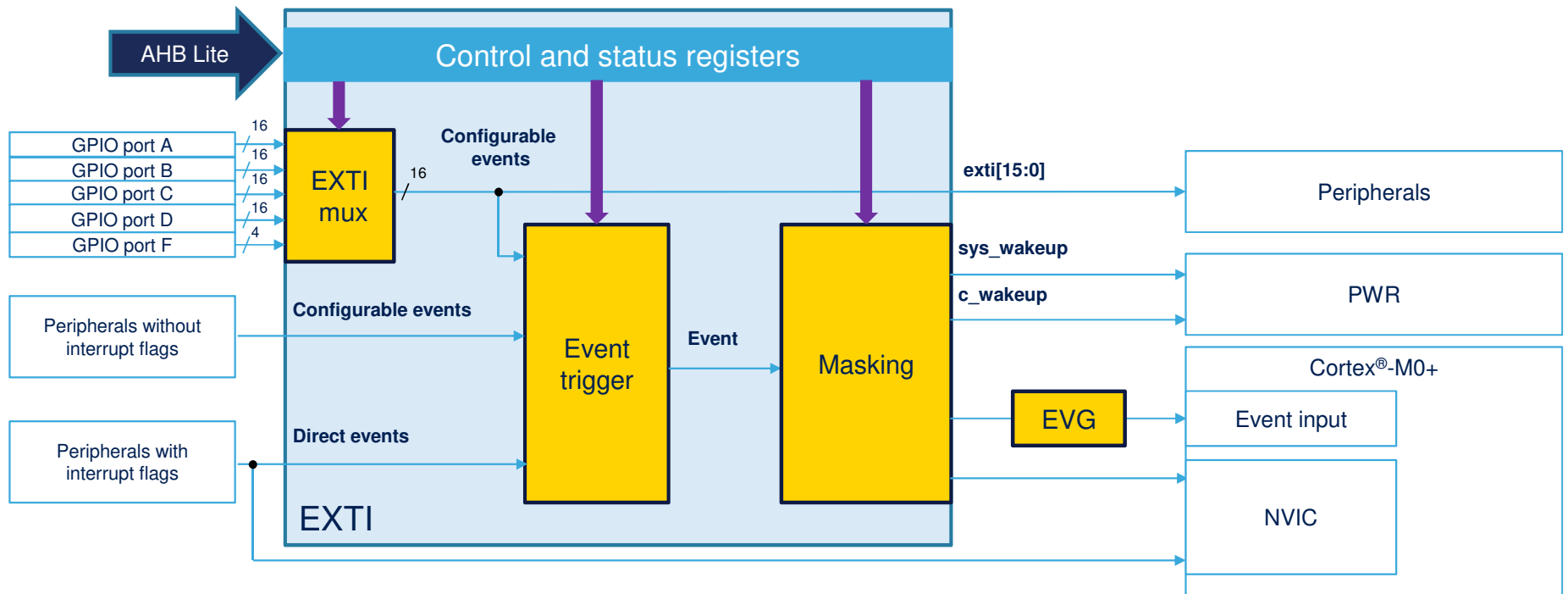
EXTI - Key features

78

- Wake-up from Stop mode, interrupts and events generation
 - Independent interrupt and event masks
- Configurable events
 - Active edge selection
 - Dedicated pending flag
 - Trigger-able by software
 - Linked to:
 - GPIO, PVD, and COMPx
- Direct events
 - Status flag provided by related peripheral
 - Linked to:
 - RTC, TAMP, I2C1, USARTx, CEC, LPUART1, LPTIMx, LSE_CSS and UCPDx

EXTI - block diagram

79



EVG: EVent Generator

- The NVIC (Nested vector Interrupt Controller) is integrated in the Cortex[®]-M0+ CPU:
 - 32 maskable interrupt channels
 - 4 programmable priority levels
 - Low-latency exception and interrupt handling
 - Power management control

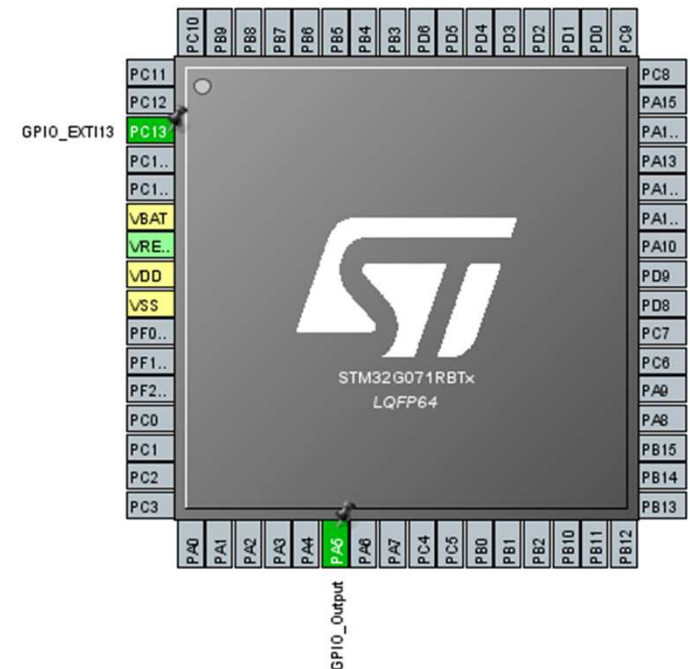
Application benefits

- Supports prioritization levels with dynamic control
- Fast response to interrupt requests
- Relocatable vector table

Lab: Pinout Configuration

81

- Close Keil uVision5 IDE if it is open; Open the “**blinky**” STM32CubeMX project (using File->Recent Projects) and save it as a new project named “**exti**”.
- Add configuration of the IO that is connected to the User Button (connected to **PC13**) to toggle the LED LD4 (connected to **PA5**) on the STM32G0 Nucleo board.
- **PA5** is already configured as **GPIO** output push-pull.
- Left-click on **PC13** and set it to **GPIO_EXTI13** mode.



GPIO Configuration

82

- Select **GPIO** under **System View** 1
- Click on Pin Name **PC13** 2
- Make sure GPIO mode is “External **Interrupt Mode** with Rising edge trigger detection” 3

The screenshot shows the STM32CubeMX software interface. At the top, the 'System view' tab is selected, and the 'GPIO' button is highlighted with a red box and a circled '1'. Below this, the 'Configuration' window is open. The 'Group By Peripherals' checkbox is unchecked. The 'GPIO' and 'NVIC' tabs are both selected. A table lists the configured pins, with the row for 'PC13' highlighted by a red box and a circled '2'. The 'PC13 Configuration' section below the table shows the 'GPIO mode' dropdown menu set to 'External Interrupt Mode with Rising edge trigger detection', which is also highlighted by a red box and a circled '3'. Other settings like 'GPIO Pull-u...' are set to 'No pull-up and no pull-down'.

Pin ...	Signal ...	GPIO P...	GPIO ...	GPIO P...	Maxim...	Fast M...	User L...	Modified
PA5	n/a	Low	Output ...	No pull...	Low	n/a		<input type="checkbox"/>
PC13	n/a	n/a	Extern...	No pull...	n/a	n/a		<input type="checkbox"/>

PC13 Configuration :

GPIO mode: External Interrupt Mode with Rising edge trigger detec...

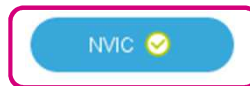
GPIO Pull-u...: No pull-up and no pull-down

User Label:

NVIC Configuration

83

- Select **NVIC** under **System View**



- Enable “**EXTI line 4 to 15 interrupts**” (by checking the box) 1

NVIC Mode and Configuration		
Configuration		
<input checked="" type="checkbox"/> NVIC	<input checked="" type="checkbox"/> Code generation	
<input type="checkbox"/> Sort by Preemption Priority and Sub Priority		
Search	<input type="text" value="Search (Ctrl+F)"/>	<input type="checkbox"/> Show only enabled interrupts
NVIC Interrupt Table	Enabled	Preemption Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0
Pendable request for system service	<input checked="" type="checkbox"/>	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0
Flash global interrupt	<input type="checkbox"/>	0
RCC global interrupt	<input type="checkbox"/>	0
EXTI line 4 to 15 interrupts	<input checked="" type="checkbox"/>	0

1

Generate Source Code

84

- Generate Code



- Click **Open Project**



- Open **main.c**, add the following code:
 - within "USER CODE BEGIN PV" / "USER CODE END PV" section (~Line 67)

```
uint8_t PC13_flag = 0;
```

```
66  /* USER CODE BEGIN PV */  
67  uint8_t PC13_flag=0;  
68  /* USER CODE END PV */
```

Add EXTI Rising Edge Callback Function

85

- Also in **main.c** add the following code,
 - within “USER CODE BEGIN 4” / “USER CODE END 4” section

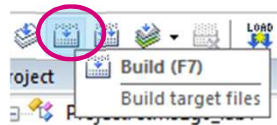
```
void HAL_GPIO_EXTI_Rising_Callback(uint16_t GPIO_Pin)
{
    PC13_flag++;
    if ( ( PC13_flag & 0x01 ) == 0x01 )
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
    }
    else
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
    }
}
```

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Rising_Callback(uint16_t GPIO_Pin)
{
    PC13_flag++;
    if ((PC13_flag & 0x01) == 0x01)
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
    }
    else
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
    }
}
/* USER CODE END 4 */
```

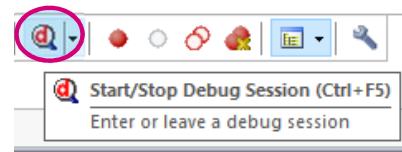
Build the Project

86

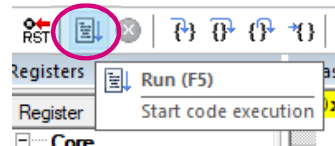
- Click the “Build” button



- Click the “Start/Stop Debug Session” button

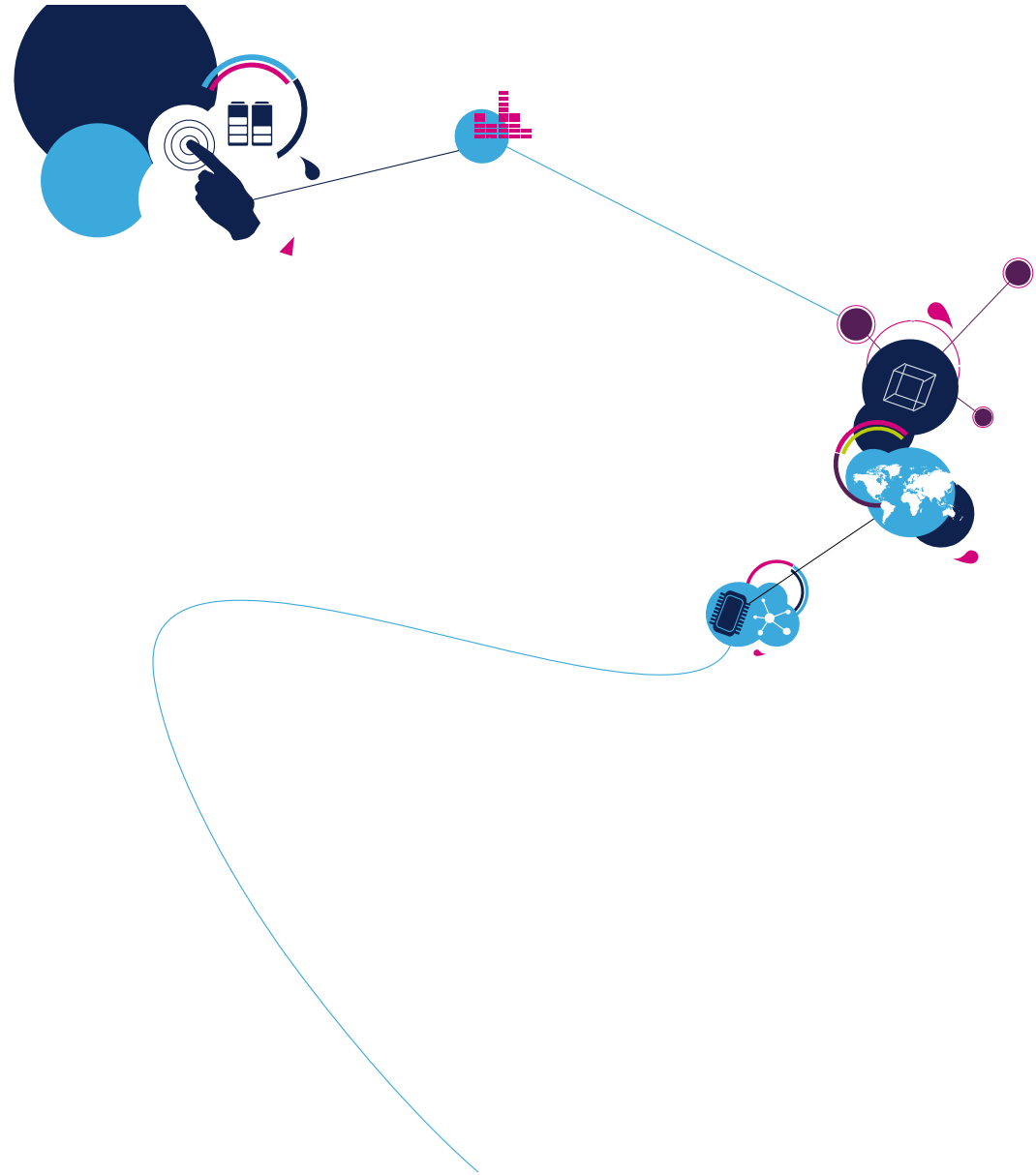


- Click “Run” button



- Push the Blue “USER” button to toggle the LED LD4!

Lab: Low Power



Objective:

- In this lab we are going use the STOP 1 mode and wakeup from RTC which is configured to wakeup the STM32 every 5 seconds.
- When the STM32 wakes up it will turn on the LED (LD4) for one second and then go back to STOP mode.
- The MCU can also wake-up using the user button which is configured as EXTI.

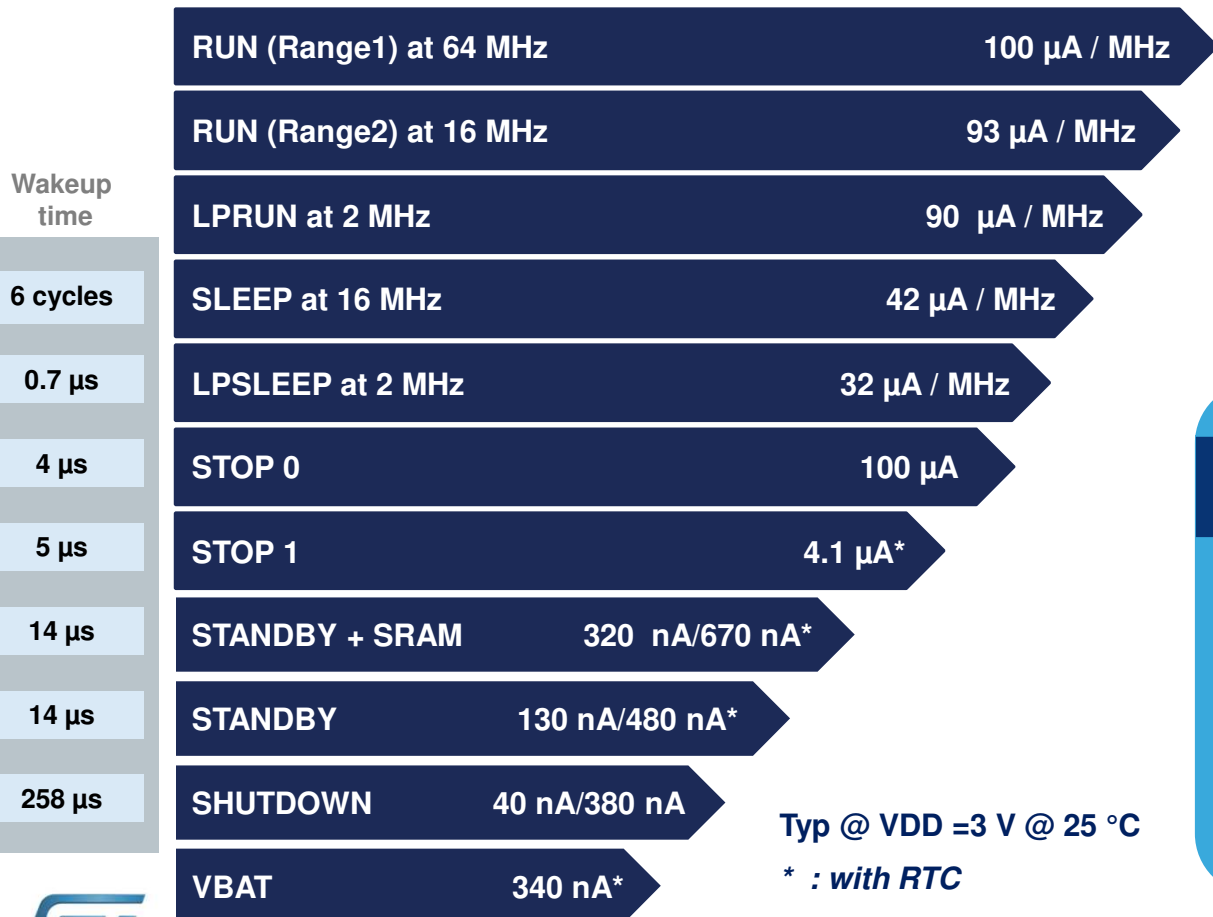
Low Power Modes

FlexPowerControl

- Efficient running
- 7 low-power modes, several sub-modes
- High flexibility

Application benefits

- High performance
→ CoreMark score = 142.88
- Outstanding power efficiency



Stop 1 mode

Available peripherals

GPIO
DMA
BOR
PVD
USART
LP UART
I2C 1
I2C 2
SPI
ADC
DAC
COMP
Temp Sensor
Timers
LPTIM 1
LPTIM 2
IWDG
WWDG
Systick Timer
UCPD
RNG
AES
CRC
CEC



I/Os kept, and configurable

Zzz

Cortex M0+

Flash
memory

SRAM
(36 Kbytes)

Available
clocks

HSI16
HSE
LSI
LSE

Flash memory not powered:

- w/o RTC: 1.3 μA @ 3.0 V
- w/ RTC: 4.1 μA @ 3.0 V

Flash memory powered:

- w/o RTC: 7.0 μA @ 3.0 V

Main regulator (MR)

Low Power regulator (LPR) up to 2 MHz

Backup domain

Backup Register (5x32 bits)

RTC & TAMPER

Active cell

Clocked-off
cell

Cell in power-
down

Wakeup time to 16 MHz:

- In SRAM: 5 μs
- In Flash ON: 5 μs
- In Flash OFF: 9 μs

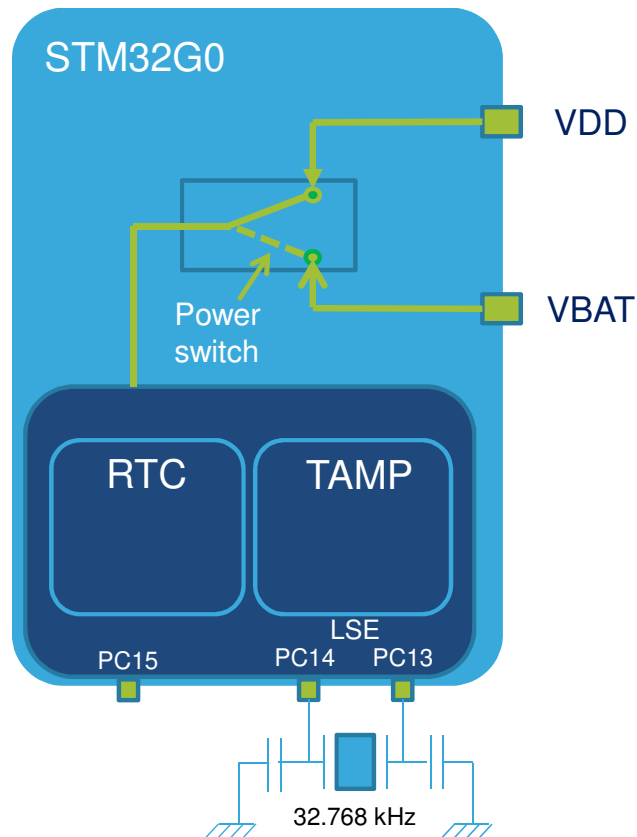
Wake-up event

NRST
BOR
PVD
RTC + Tamper
USART
LP UART
I2C 1
CEC
COMP
LPTIM 1
LPTIM 2
IWDG
GPIOs

Available
Periph and clock

RTC - Overview

91



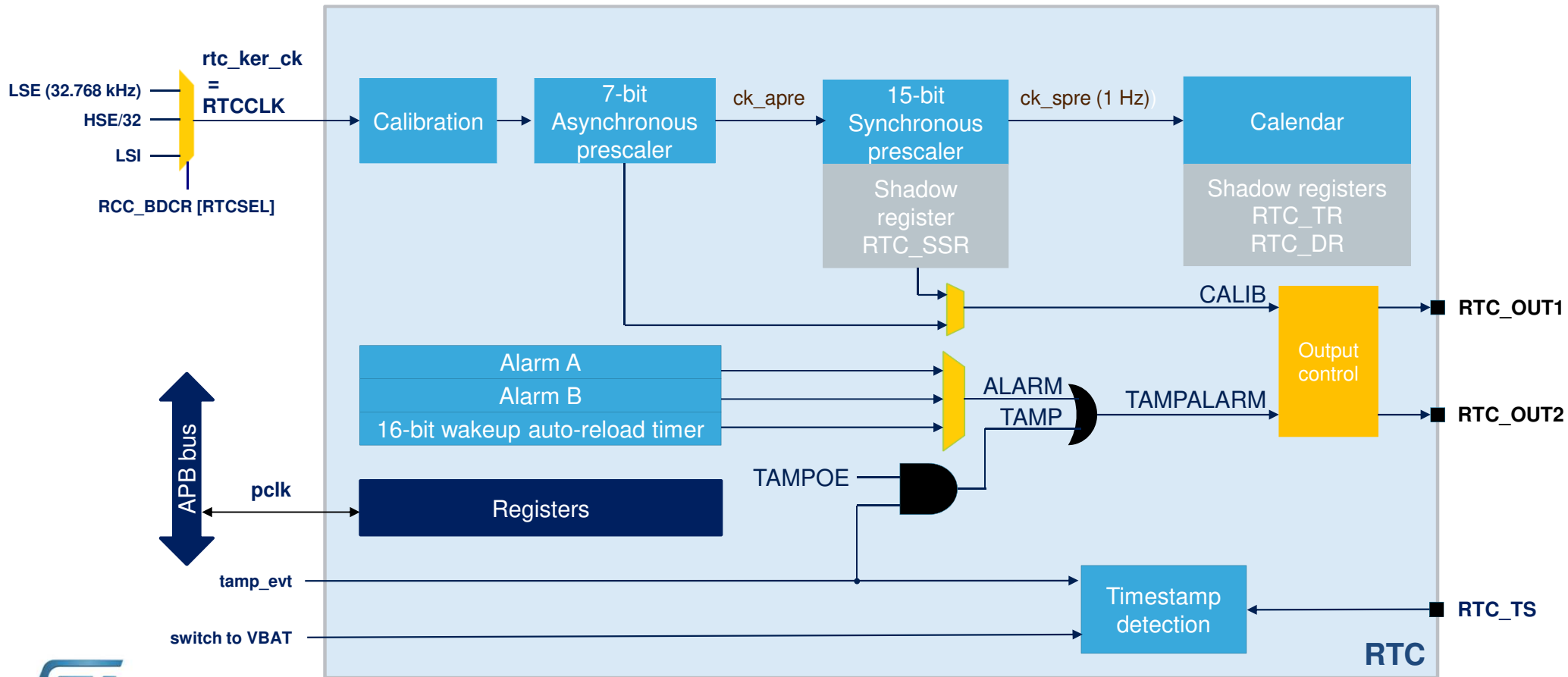
- The RTC provides an ultra-low-power hardware calendar with alarms, in all low-power modes
- It belongs to the Battery Backup Domain, so it is kept functional when the main supply is off and VBAT is present
- The TAMP peripheral features the backup registers and tamper detection

Application benefits

- Ultra-low power: 300 nA at 1.8 V
- Hardware BCD calendar to reduce software load

RTC - Block diagram

92



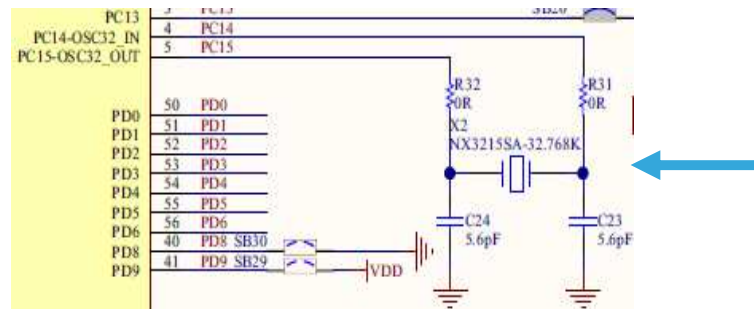
- RTC not affected by system reset when clocked by LSE

- Close Keil uVision5 IDE if it is open; In STM32CubeMX open the “**exti**” STM32CubeMX project save it as a new project like “**lowpower**”.

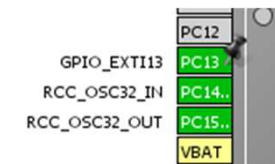
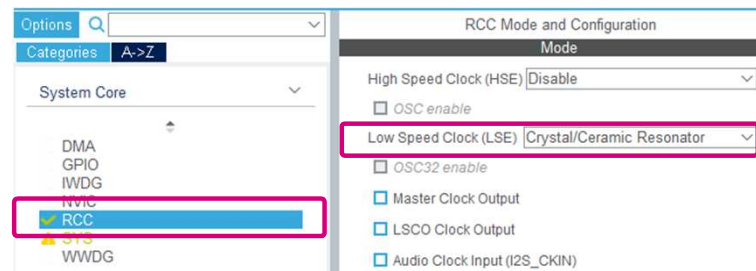
Enable LSE (Low Speed External) Clock

94

- We are going to use the 32 KHz Crystal that is on the Nucleo board (see schematic below) to clock the RTC:



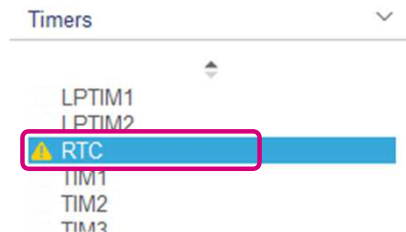
- In the **Pinout & Configuration** tab, under **System Core** expand **RCC** and choose Crystal/Ceramic Resonator for Low Speed Clock (LSE) clock:



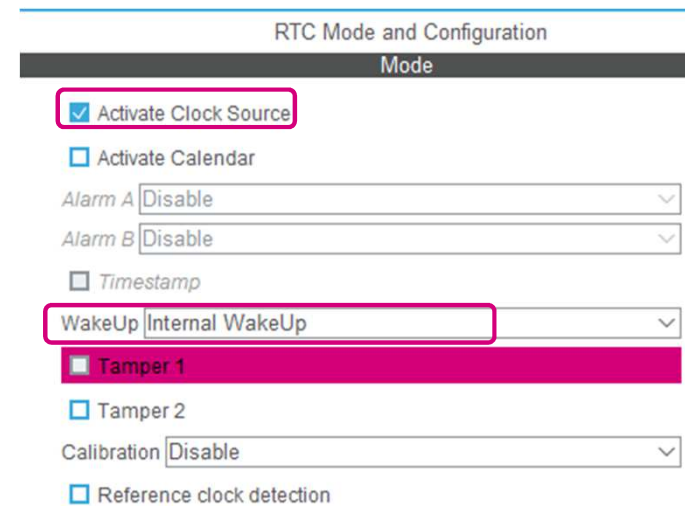
Enable and Configure the RTC

95

- In the **Pinout** tab, under **Timers**, expand **RTC**



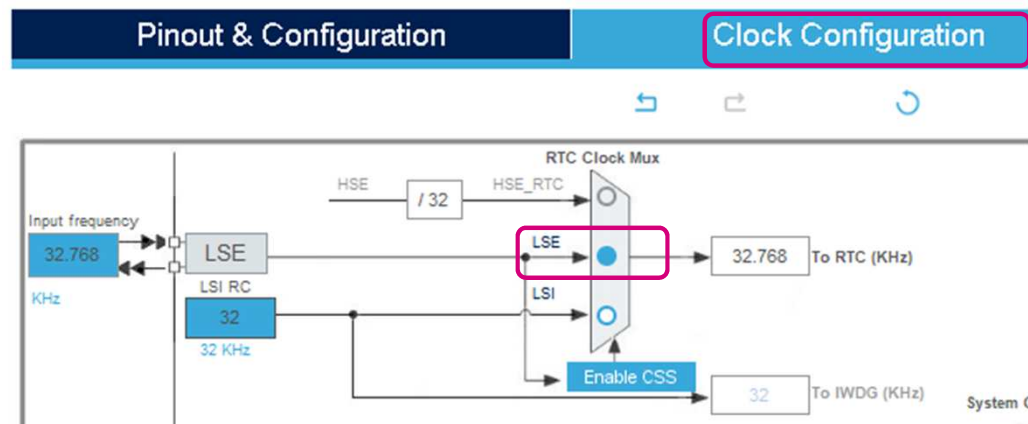
- Check the **Activate Clock Source**
- Select **Internal Wakeup** for the **Wakeup** mode



Choose RTC clock source

96

- In the **Clock Configuration** tab, select LSE as input clock for RTC



Note: For applications that do not require precise RTC timings the LSI (Low Speed Internal RC) can be used to clock the RTC

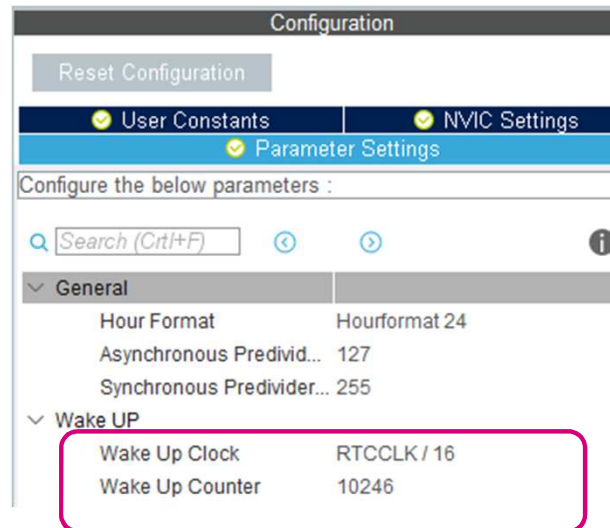
Wakeup Counter Calculation:

- To configure the wake up timer for 5s, the WakeUpCounter should be set to **10246** as calculated below:
- With RTC Clock set to **RTCCLK /16**
- Wakeup Time Base = $\text{RTC_PRESCALER} / \text{LSE} = 16 / (32.768\text{KHz}) = 0.488 \text{ ms}$
- Wakeup Time = Wakeup Time Base * WakeUpCounter = $0.488\text{ms} * \text{WakeUpCounter}$

$$\Rightarrow \text{WakeUpCounter} = 5\text{s} / 0.488\text{ms} = \mathbf{10246}$$

RTC configuration 98

- Based on previous calculation we will configure the RTC
- In the **Pinout & Configuration** tab, click on **RTC** (under System Core)
- Enter the following configuration:



The screenshot shows the 'Configuration' window for the RTC module. The 'Parameter Settings' tab is selected. The 'General' section is expanded, showing 'Hour Format' set to 'Hourformat 24', 'Asynchronous Predivid...' set to '127', and 'Synchronous Predivider...' set to '255'. The 'Wake UP' section is also expanded, showing 'Wake Up Clock' set to 'RTCCLK / 16' and 'Wake Up Counter' set to '10246'. A red rectangle highlights the 'Wake UP' section.

Configuration	
Reset Configuration	
✓ User Constants	✓ NVIC Settings
✓ Parameter Settings	
Configure the below parameters :	
Search (Ctrl+F) [Navigation icons]	
General	
Hour Format	Hourformat 24
Asynchronous Predivid...	127
Synchronous Predivider...	255
Wake UP	
Wake Up Clock	RTCCLK / 16
Wake Up Counter	10246

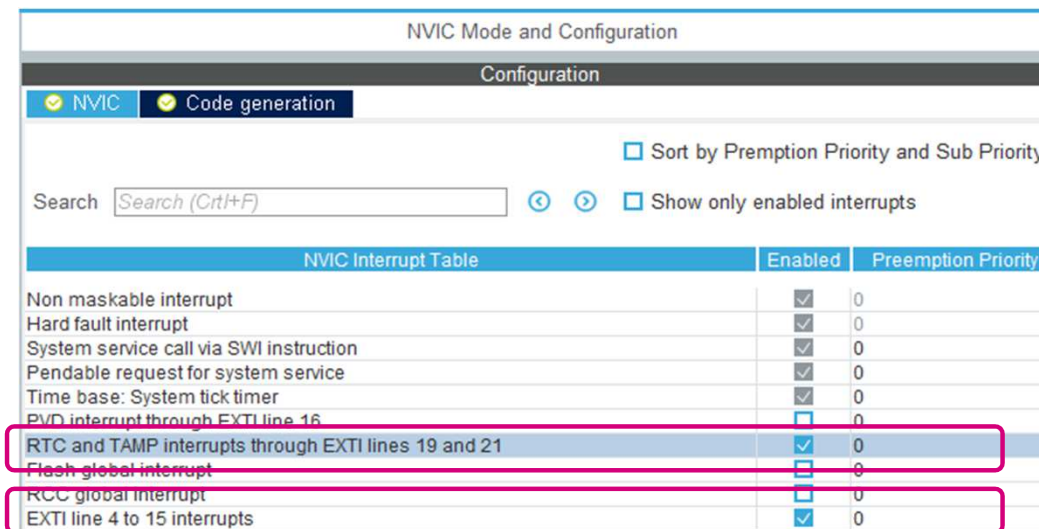
Enable Interrupts

99

- In the **Configuration** tab, go to **NVIC settings** and then enable the interrupt for **RTC**:



- In the “System View” in the NVIC, check that both RTC and EXTI[4...15] are enabled, if not re-enable them both:



Generate Source Code

100

- **Generate Code**



- Click **Open Project**



Add code – to main function

101

- Open the **main.c**, add the following code in the **while(1)** loop of the main function in the **USER CODE WHILE** section (~ line 121):

```
/* USER CODE BEGIN WHILE */
while (1)
{

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
    HAL_Delay(1000);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);

    // enter STOP mode
    HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);

    // reconfigure system clock
    SystemClock_Config();
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

```
117  /* Infinite loop */
118  /* USER CODE BEGIN WHILE */
119  while (1)
120  {
121      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
122      HAL_Delay(1000);
123      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
124
125      // enter STOP mode
126      HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);
127
128      // reconfigure system clock
129      SystemClock_Config();
130      /* USER CODE END WHILE */
131
132      /* USER CODE BEGIN 3 */
133  }
134      /* USER CODE END 3 */
135  }
```

Add code – to init function

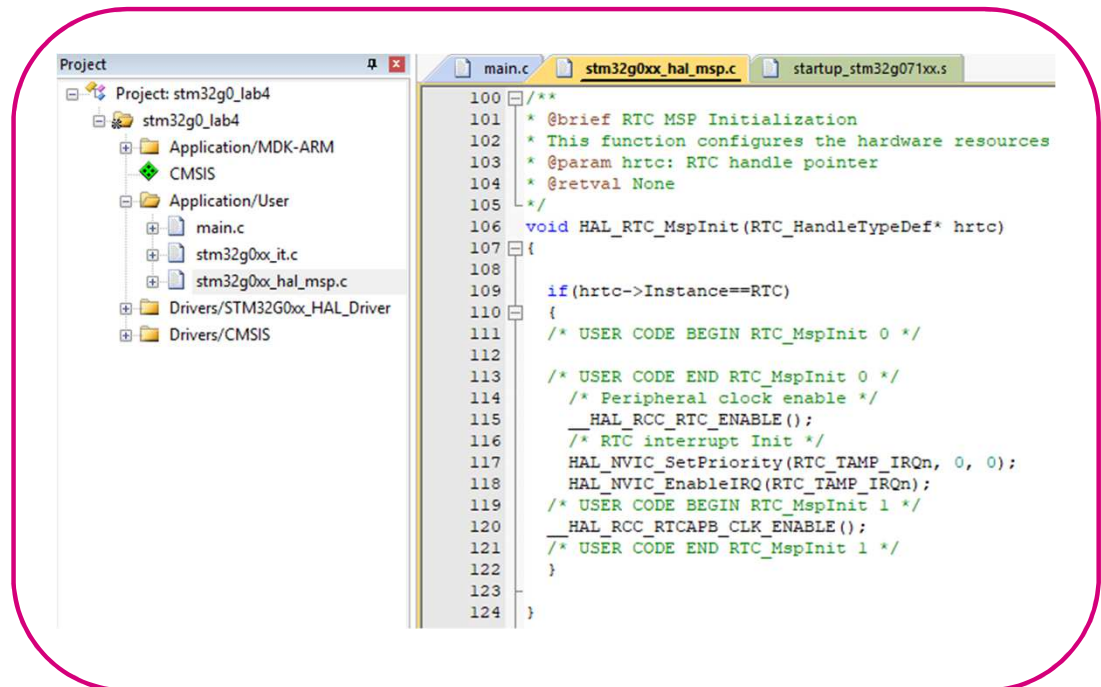
102

- Open the **stm32g0xx_hal_msp.c**, Add the following line of code (marked in red below) to the msp init function HAL_RTC_MspInit():

```
void HAL_RTC_MspInit(RTC_HandleTypeDef* hrtc)
{

    if (hrtc->Instance==RTC)
    {
        /* USER CODE BEGIN RTC_MspInit 0 */

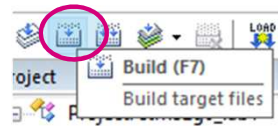
        /* USER CODE END RTC_MspInit 0 */
        /* Peripheral clock enable */
        __HAL_RCC_RTC_ENABLE();
        /* RTC interrupt Init */
        HAL_NVIC_SetPriority(RTC_TAMP_IRQn, 0, 0);
        HAL_NVIC_EnableIRQ(RTC_TAMP_IRQn);
        /* USER CODE BEGIN RTC_MspInit 1 */
        __HAL_RCC_RTCAPB_CLK_ENABLE();
        /* USER CODE END RTC_MspInit 1 */
    }
}
```



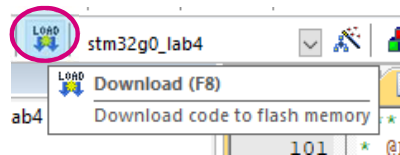
Build the Project

103

- Click the “Build” button; or use menu Project > Build target.



- Click the “**Load**” button (F8) to flash the code into the STM32 (not using the debug session because we are using low power modes)



- Press Reset on your board (black button) once the code is loaded and the application will work as follows:
 - RUN** mode for 1 second (LD4 LED on)
 - STOP** mode for 5 seconds (LD4 LED off) with wakeup by RTC
 - If during the **STOP** mode (LD4 LED off) you press the user button: the interrupt (EXTI) will wakeup from STOP mode

Optional Lab: Estimation of power consumption

Optional Lab: Estimation of power consumption

105

Objective:

- Use the Power tool inside the STM32CubeMX to estimate the average power consumption of the low power lab we just finished.

Power Supply and Power Source Selection

106

- Using the “**lowpower**” project in STM32CubeMX

- Click on the **Tools** tab in STM32CubeMX



- Select **3V** for **VDD**

T_A 25°C / V_{DD} 3.0V

$T_{Ambient}$ 25°C

V_{DD}

- In the **Battery Selection** section, select **AA Alkaline** batteries (2 in series, 1 in parallel) as the power source for the application

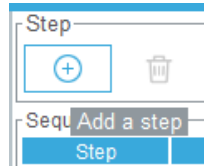
Alkaline(AA LR6) (2x1)

In Series In Parallel

Capacity	2850.0 mAh
Self Discharge	0.3 %/month
Nominal Voltage	3.0 V
Max Cont Current	1000.0 mA

- Add a step to our power sequence:

- Click: Step.. Add



- Configure a first step: RUN mode

- Mode: Run
- Power Range: Range2 -Medium
- Memory Fetch Type: Flash
- VDD: 3.0
- Voltage Source: Battery

- CPU Frequency: 16 MHz
- Clock Configuration: HSI

- Enable IPs from Pinout function

- Duration: 1 second

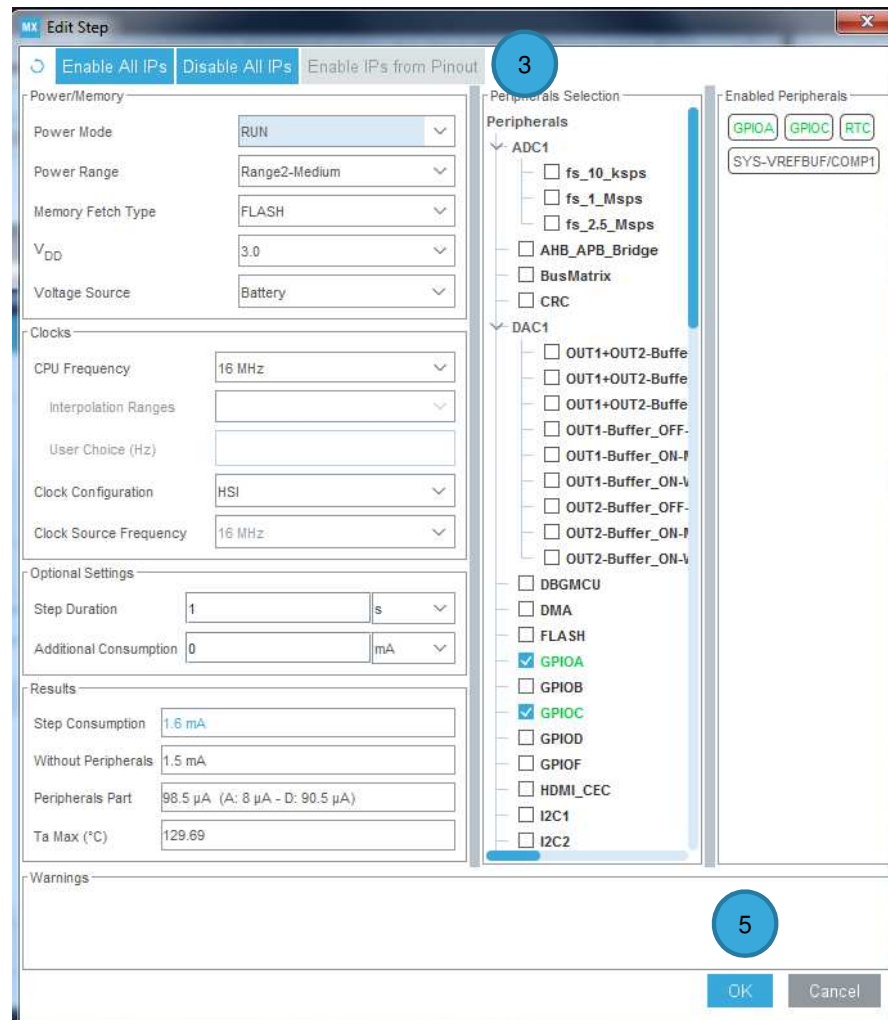
- Click "Add"



Resulting step consumption should be 1.6mA

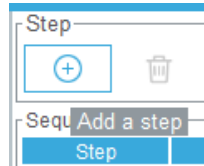
Adding a RUN mode step

107



• Add a step to our power sequence:

- Click: Step.. Add



• Add a second step: STOP1 mode

1

- Power Mode: STOP1
- Fetch Type: Flash -PowerDown
- VDD: 3V

2

- Clocks HSI 16 MHz

3

- RTC enabled (to wakeup the system)

4

- Step Duration: 5 seconds

5

- Click "Add"

- Resulting step consumption should be 3.4 uA

Add a STOP1 mode step

New Step

Enable All IPs | Disable All IPs | Enable IPs from Pinout

1 Power/Memory

- Power Mode: STOP1
- Power Range: NoRange
- Memory Fetch Type: Flash-PowerDown
- V_{DD}: 3.0
- Voltage Source: Battery

2 Clocks

- CPU Frequency: 16 MHz
- Interpolation Ranges:
- User Choice (Hz):
- Clock Configuration: HSI
- Clock Source Frequency: 16 MHz

3 Peripherals Selection

Peripherals

- ☐ IWDG*
- ☐ LPTIM1*
- ☐ LPTIM2*
- ☒ RTC*

Enabled Peripherals

RTC*

4 Optional Settings

- Step Duration: 5 s
- Additional Consumption: 0 mA

5 Results

- Step Consumption: 3.4 μ A
- Without Peripherals: 3.4 μ A
- Peripherals Part: 0 nA (A: 0 nA - D: 0 nA)
- Ta Max (°C): 130

Warnings

Add Cancel

Add a Wakeup from STOP1 mode step

109

1 Add a last step: Wakeup from STOP1 mode

- Power Mode: WU_FROM_STOP1
- VDD = 3V
- Voltage source: Battery

- ## 2
- Click “Add”

- Resulting step consumption should be 1.21 mA

Edit Step

Enable All IPs | Disable All IPs | Enable IPs from Pinout

Power/Memory

Power Mode: WU_FROM_STOP1
Power Range: NoRange
Memory Fetch Type: Flash-PowerDown
V_{DD}: 3.0
Voltage Source: Battery

Clocks

CPU Frequency: 16 MHz
Interpolation Ranges:
User Choice (Hz):
Clock Configuration: HSI
Clock Source Frequency: 16 MHz

Optional Settings

Wakeup time: 9.0 μs
Additional Consumption: 0 mA

Results

Step Consumption: 1.21 mA
Without Peripherals: 1.21 mA
Peripherals Part: 0 nA (A: 0 nA - D: 0 nA)
Ta Max (°C): 129.76

Peripherals Selection

Peripherals:
Enabled...:

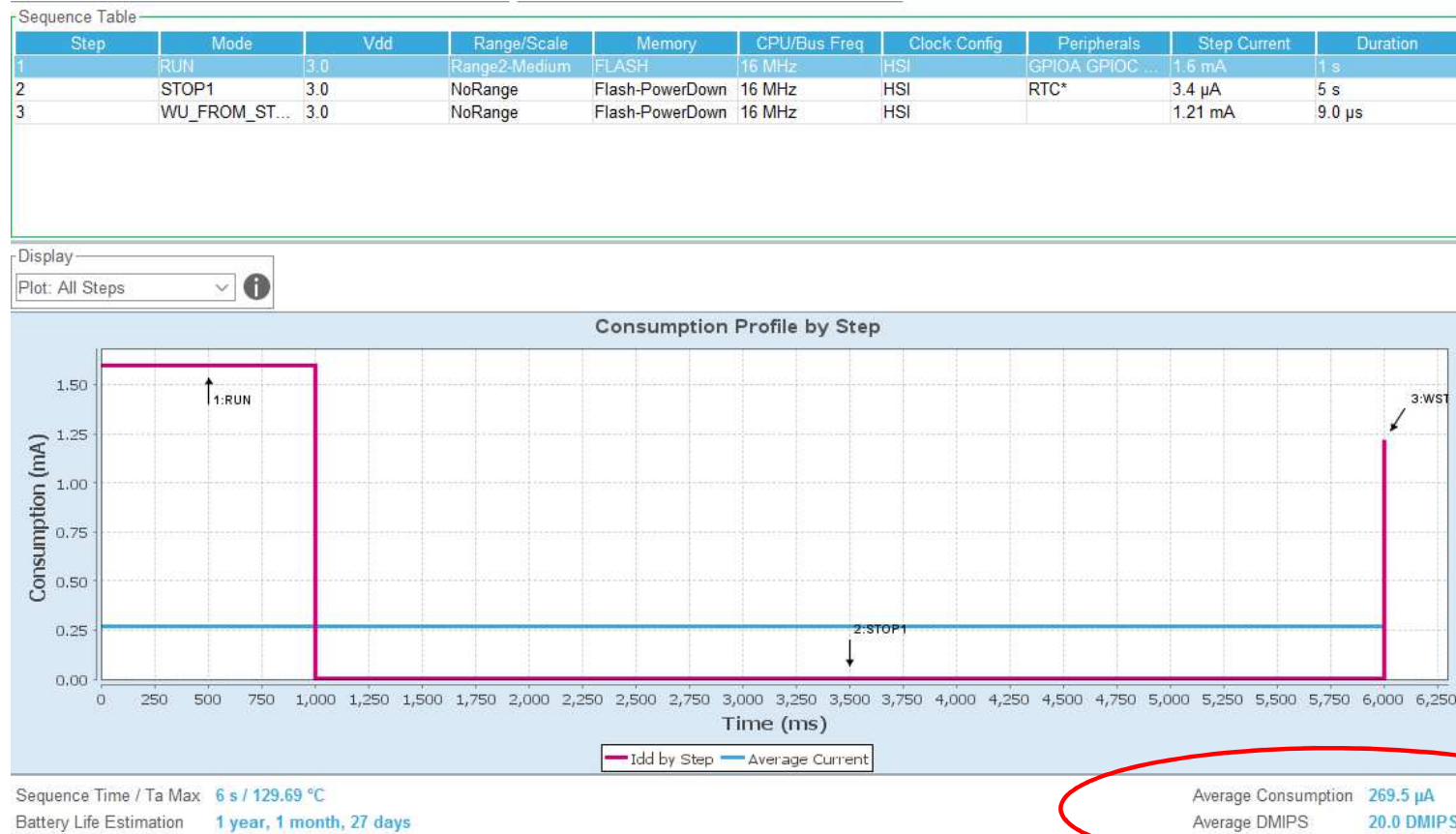
Warnings

OK Cancel

Average Current Consumption Result

110

- Note: the Current consumption numbers are for the MCU only.



Optional Lab: printf() debugging using UART

Lab: printf() debugging using UART

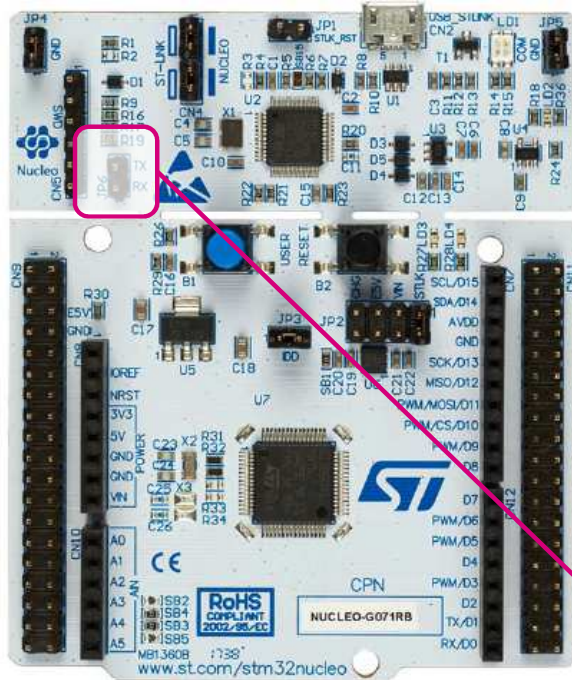
112

Objective:

- Redirect Printf output to LPUART1 which is connected to the ST-LINK Virtual COM port on the Nucleo board
- Using a Terminal like Teraterm we can view the printf output.

printf() debugging settings overview

113



LPUART1 debug will be used via the ST-LINK Virtual-COM port

Set up additional GPIO / Clocks:

PA2 – LPUART1, “LPUART1-TX”

PA3 – LPUART1, “LPUART1-RX”

LPUART1 Clock = PCLK1 (64MHz)

LPUART1 settings:

Asynchronous Mode - 115200 N/8/1, No HW Flow control

Tx/Rx, No advanced features

Teraterm Terminal will be used to display the printf output

LPUART1 is routed to the ST-LINK's USART, and brought via the USB Virtual-COM port class (SB16/18 located on the back on the board have been soldered)

STM32G0 USART/LPUART features

114

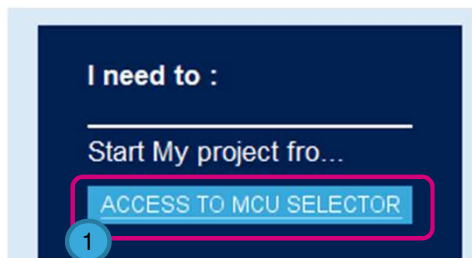
USART features	USART1/2	USART3/4	LPUART1
Hardware flow control for modem	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Slave/Master)	X	X	-
Smartcard mode	X	-	-
Single wire half duplex communication	X	X	X
IrDA SIR ENDEC	X	-	-
LIN mode	X	-	-
Dual clock domain and wakeup from Stop mode	X	-	X
Receiver timeout	X	-	-
Auto baudrate detection	X	-	-
Driver enable	X	X	X
Data length	7, 8 and 9 bits		
TX/RX FIFO	X	-	X
TX/RX FIFO size (data word)	8	-	8

Create New Project

115

- In STM32CubeMX, click “Home”

- Click **Access To MCU Selector** 1



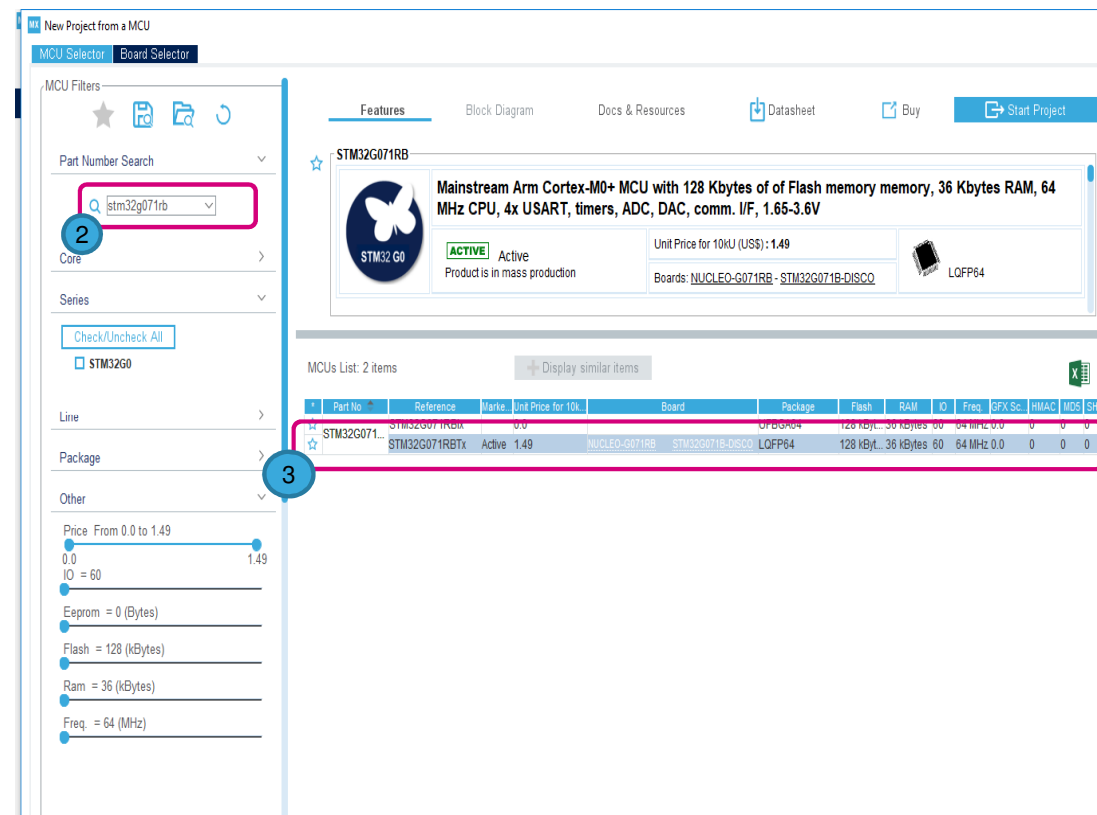
- Select **STM32G071RBTx** 2

- LQFP64, 128KB Flash

- Double Click 3



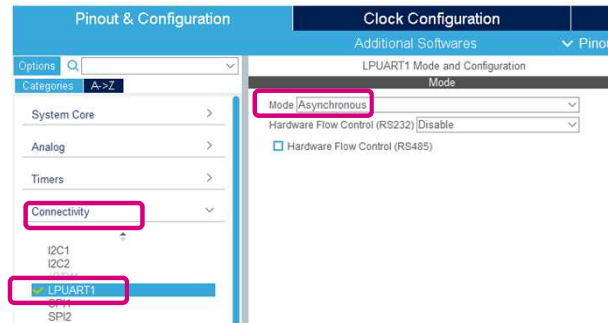
Home /



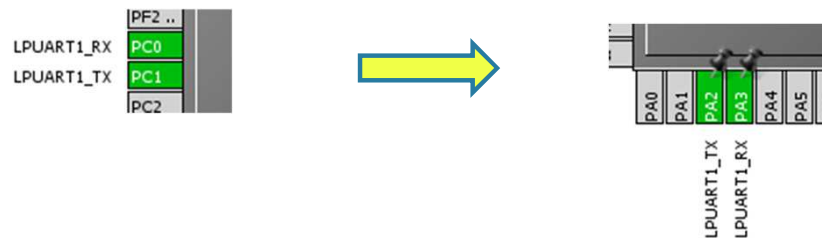
GPIO Configuration additions

116

- Click on **LPUART1** dialog (under Connectivity), and select **Asynchronous** mode:



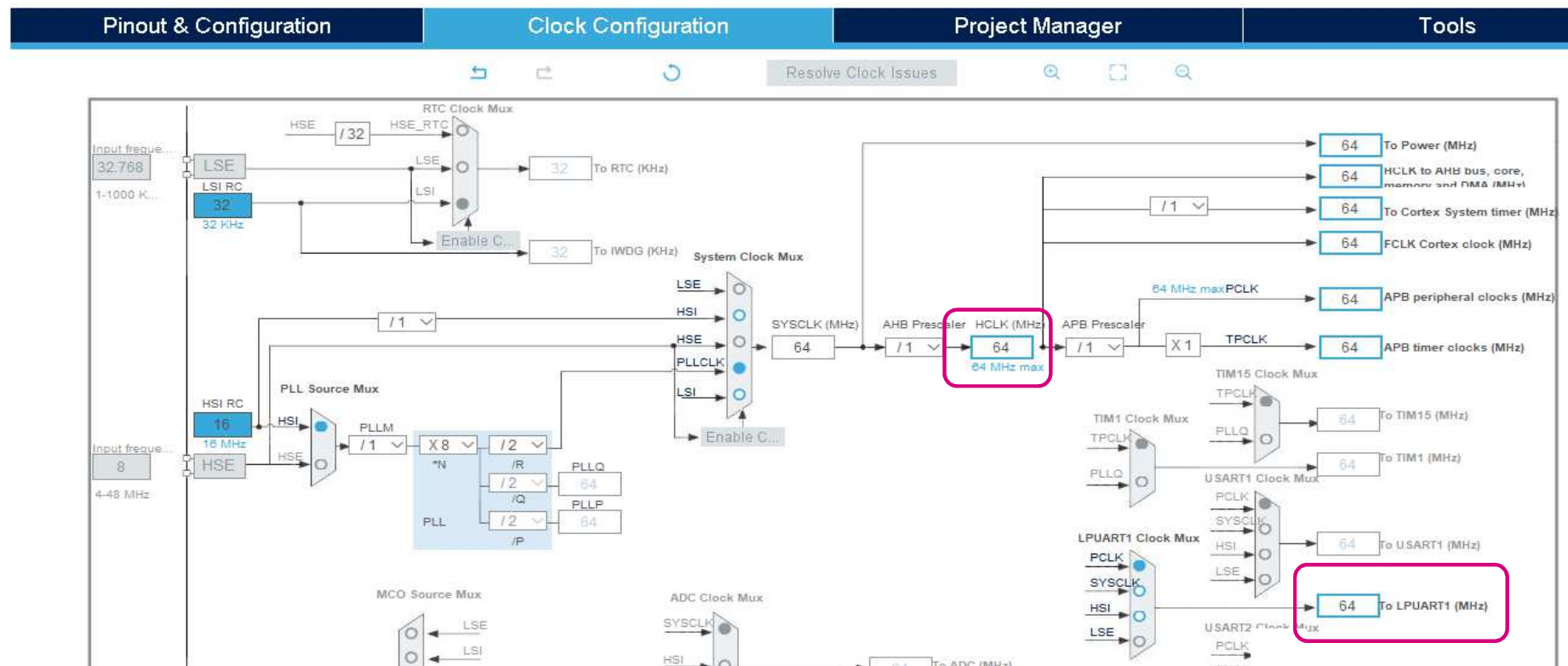
- Use PA2 & PA3 for Tx / Rx pins:
 - These are the alternate mapping pins (PC0/PC1 are default)
 - So need to remap



Clock Configuration

117

- Run the STM32G0 at 64 MHz for this lab, the LPUART1 clock also at 64 MHz.



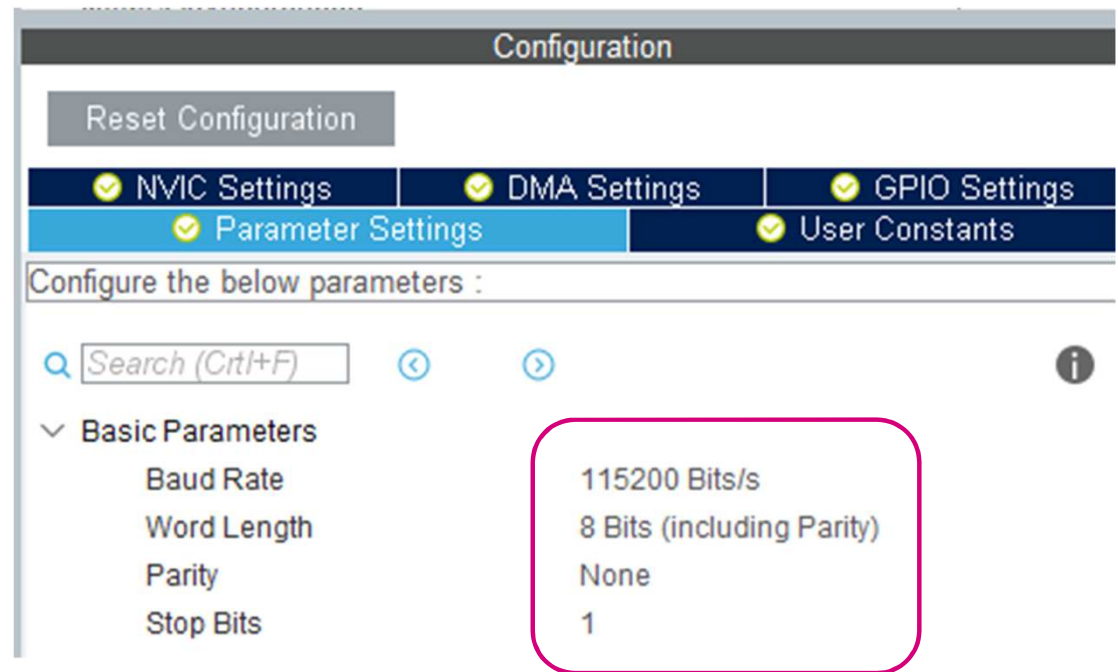
LPUART1 Configuration

118

- Click on the Configuration tab and select LPUART1

- Parameter Settings tab

- 115200 Bits/s
- 8-bit word length
- No parity bit
- 1 Stop bit
- Keep Default settings for the rest



Generate Source Code

119

- Open **Project Manager**
- Set the project name (**printf**) and the project location (**C:\STM32G0Workshop\STM32G0\HandsOn**)
- Set the IDE Toolchain to **MDK-ARM V5**

Project Manager

1 Project Name
printf

2 Project Location
C:\STM32G0Workshop\HandsOn

Application Structure
Basic ☐ Do not generate the ma...

Toolchain Folder Location
C:\STM32G0Workshop\HandsOn\printf\

3 Toolchain / IDE
MDK-ARM V5 ☐ Generate Under Root

Linker Settings
Minimum Heap Size 0x200
Minimum Stack Size 0x400

- **Generate Code**
- Click **Open Project**



Adding printf redirecting code in main.c

120

1- Add the stdio include:

```
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */
```

```
44 /* Private includes -----
45 /* USER CODE BEGIN Includes */
46 #include <stdio.h>
47 /* USER CODE END Includes */
```

2- Add following code in the section below:

```
/* USER CODE BEGIN PFP */
/* Private function prototypes -----*/
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
/* USER CODE END PFP */
```

```
75 /* USER CODE BEGIN PFP */
76 #define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
77 /* USER CODE END PFP */
```

3- Add following function in the section below:

```
/* USER CODE BEGIN 4 */
PUTCHAR_PROTOTYPE
{
    HAL_UART_Transmit(&hlpuart1, (uint8_t *)&ch, 1, 0xFFFF);
    return ch;
}
/* USER CODE END 4 */
```

```
241 /* USER CODE BEGIN 4 */
242 PUTCHAR_PROTOTYPE
243 {
244     HAL_UART_Transmit(&hlpuart1, (uint8_t *)&ch, 1, 0xFFFF);
245     return ch;
246 }
247 /* USER CODE END 4 */
```

Adding application code in main.c

121

Add application code in main loop:

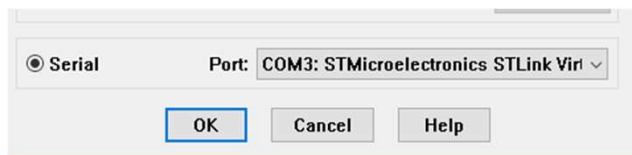
```
/* USER CODE BEGIN WHILE */
while (1)
{
    printf("** Hello World ** \n\r");
    HAL_Delay(1000);
/* USER CODE END WHILE */
```

```
117 /* Infinite loop */
118 /* USER CODE BEGIN WHILE */
119 while (1)
120 {
121     printf("** Hello World ** \n\r");
122     HAL_Delay(1000);
123
124     /* USER CODE END WHILE */
125
126     /* USER CODE BEGIN 3 */
127 }
128 /* USER CODE END 3 */
129 }
```

Build the Project and run the application

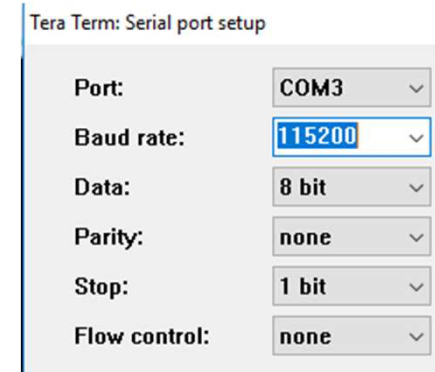
122

- Click the “Build” button; or use menu Project > Build target.
- Click the “Start/Stop Debug Session” button
- Click “Run” button
- Open a Terminal emulator like TeraTerm, using LPUART1 settings, connect ST-LINK Virtual COM port xx



- You should see the printf message being displayed.

```
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***  
*** Hello World ***
```



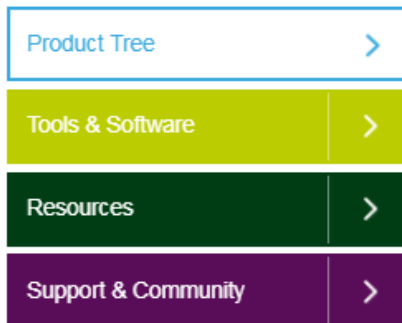


Summary

123

- 1 **Efficient** (Power, Performance and Cost)
- 2 **Robust** (EMS, ECC, Clock Monitoring/Watchdogs, Security)
- 3 **Simple** (Easy to configure and develop code)

www.st.com/stm32g0



We Greatly Value Your Feedback

Use your phone to scan the QR code or type the link into your browser.



<https://www.surveymonkey.com/r/8WBPJFF>

Thank you!