

STM32 متّحكمات

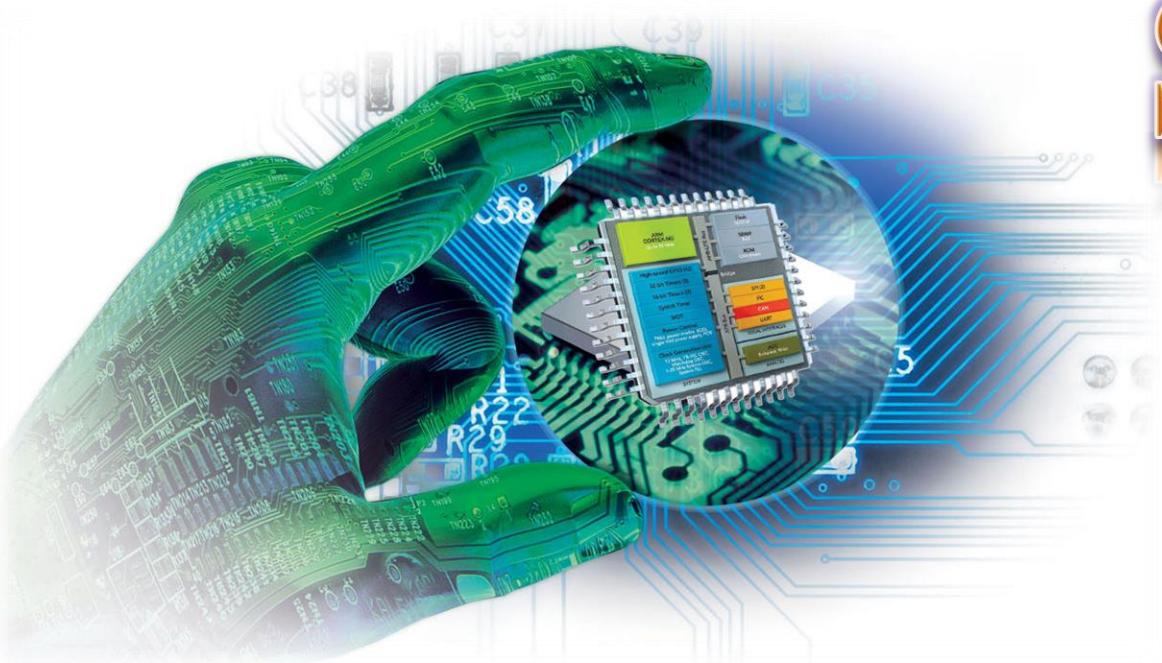
سننطرق في هذا الكورس إلى المواقعيّات التالية:

- مقدمة عن المتحكمات المصغرة ثم متحكمات STM32، بالإضافة إلى تغذيتها ومصادر الساعة لها
- مداخل ومخارج متحكمات stm32 وطرق برمجتها
- المقاطعات في متحكمات stm32
- المؤقتات في متحكمات stm32
- المنفذ التسلسلي USART/UART
- المبدلات التشابهية الرقمية ADC
- المبدلات الرقمية التشابهية DAC والوصول المباشر للذاكرة DMA
- بروتوكول الاتصال I2C
- بروتوكول الاتصال SPI
- مؤقتات متقدمة
- أنماط الطاقة في متحكمات stm32

مُتَحَكِّمَات

STM32

1

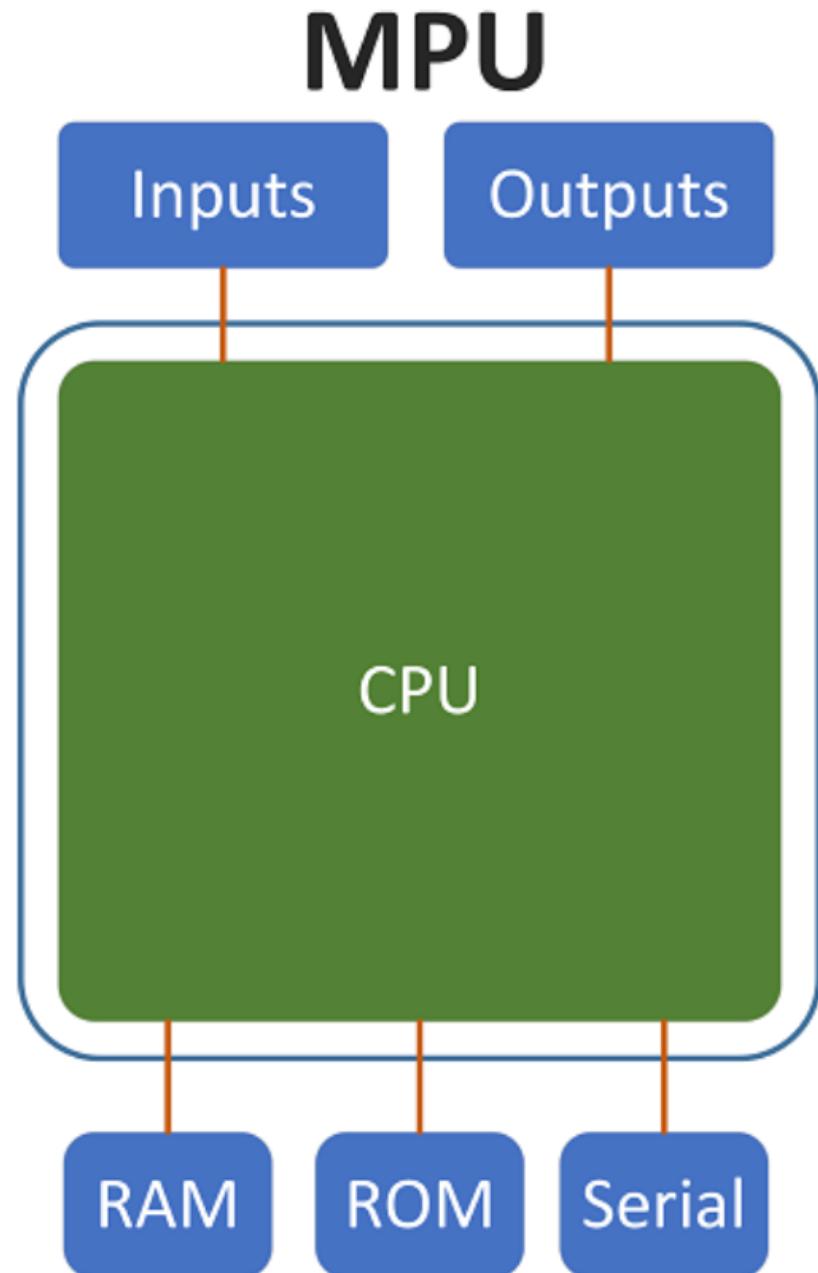
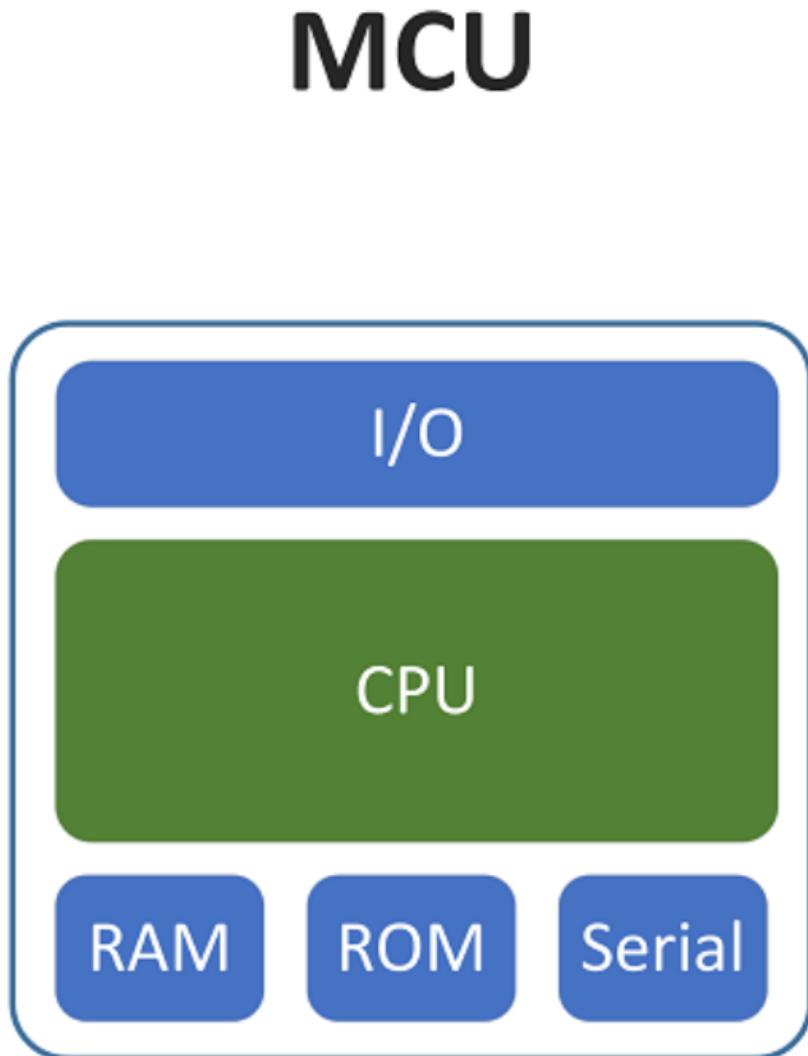


- العوامل المؤثرة في تصميم الأنظمة المدمجة.
- ما هو المتحكم المصغر.
- معاييرة تصميم بنية المعالجات.
- بني مسجلات التعليمات في المعالجات.
- مقارنة بين المتحكم المصغر والمعالج المصغر
- المعالجات المبنية بواسطة ARM:
- الهيكلية CORTEX-ARM
- مزایا المتحكم STM32G0
- أنواع اللوحات التطويرية المتوفرة Boards type
- تغذية متحكمات STM32
- مصادر الساعة في متحكمات STM32

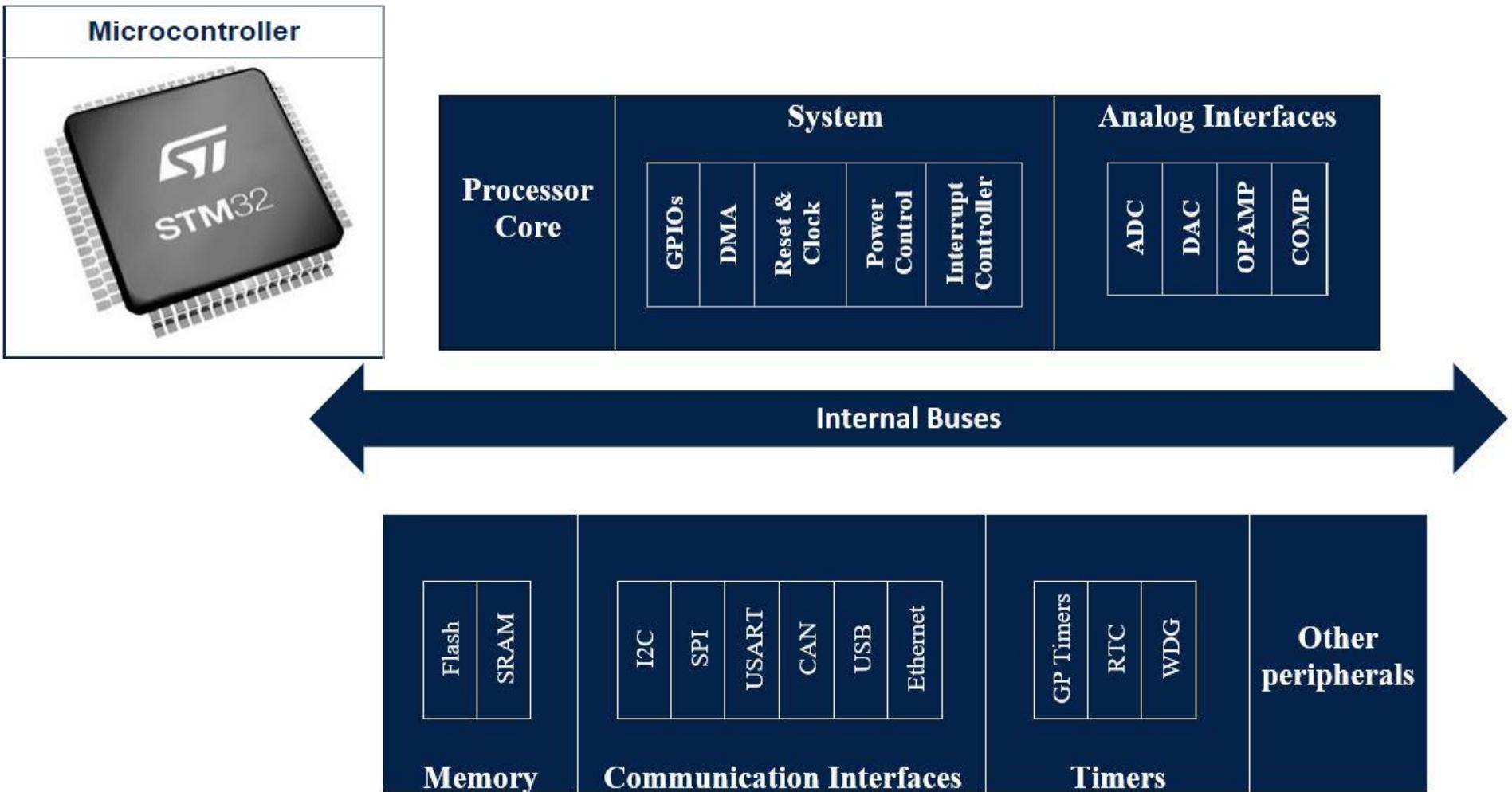
العوامل المؤثرة في تصميم الأنظمة المدمجة:

- ❖ سعة المعالجة (MIPS – Processing Power)
- ❖ عرض الناقل الداخلي (Data-Bus) 4bit, 8bit, 16bit, 32bit
- ❖ حجم الذاكرة (Memory Space) Flash, RAM, EEPROM
- ❖ استهلاك الطاقة (Power Consumption) mW/MIPS
- ❖ كلفة التطوير (Development Cost) HW+SW
- ❖ حياة المنتج (Lifetime) – يؤثر في جميع قرارات التصميم
- ❖ الوثوقية (Reliability) – مقدرة النظام على الاستجابة في مختلف الظروف؟
- ❖ متطلبات وظيفية أخرى خاصة تتعلق بـ بهوية النظام – المعالجة في الزمن الحقيقي

:Microcontroller VS microprocessor



What is Microcontroller ?



1) المعالج CPU : يقوم بالعمليات الحسابية والمنطقية والنقل والتحكم.

2) مسجلات Registers :

هي عبارة عن ذاكرة مؤقتة يعتمد حجمها على نوع معالج CPU ولها عدة أنواع :

- مسجلات ذات أغراض عامة.
- مسجلات الخاصة.
- مسجلات التحكم.
- مسجلات الحالة.
- مسجلات المعطيات.

(3) ذاكرة Flash memory : هي ذاكرة دائمة تستخدم لتخزين برنامج المتحكم المصغر.

(4) ذاكرة RAM memory : هي عبارة عن ذاكرة مؤقتة للبيانات Data التي يقوم معالجتها CPU.

(5) ذاكرة EEPROM memory : هي ذاكرة دائمة تستخدم لتخزين معطيات المستخدم.

6) منافذ رقمية Ports:

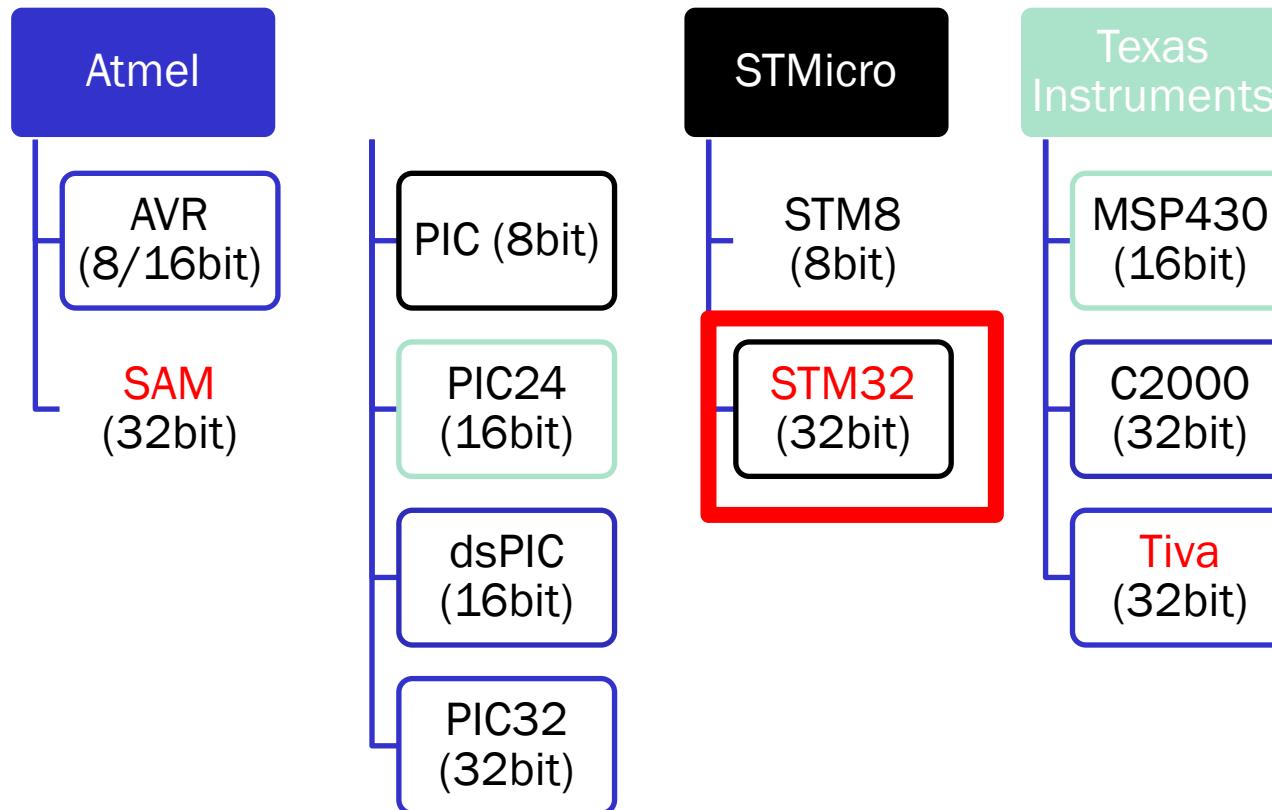
تستخدم لتبادل المعطيات الرقمية مع العالم الخارجي.

7) مؤقتات وعدادات Timer & Counter

وقد تحتوي أيضاً:

- محولات تشابهية رقمية ADC.
- محولات رقمية تشابهية DAC
- طرفيات اتصال تسلسلي.
- طرفيات أخرى.

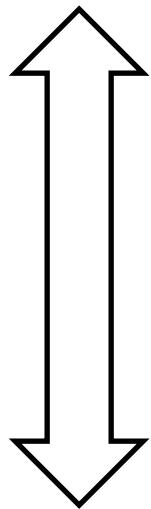
أشهر عوائل الميكانيات المصفرة



ARM Cortex-M based

أشهر لغات البرمجة الشائعة لبرمجة الميكانيات المصغرة :

Low level

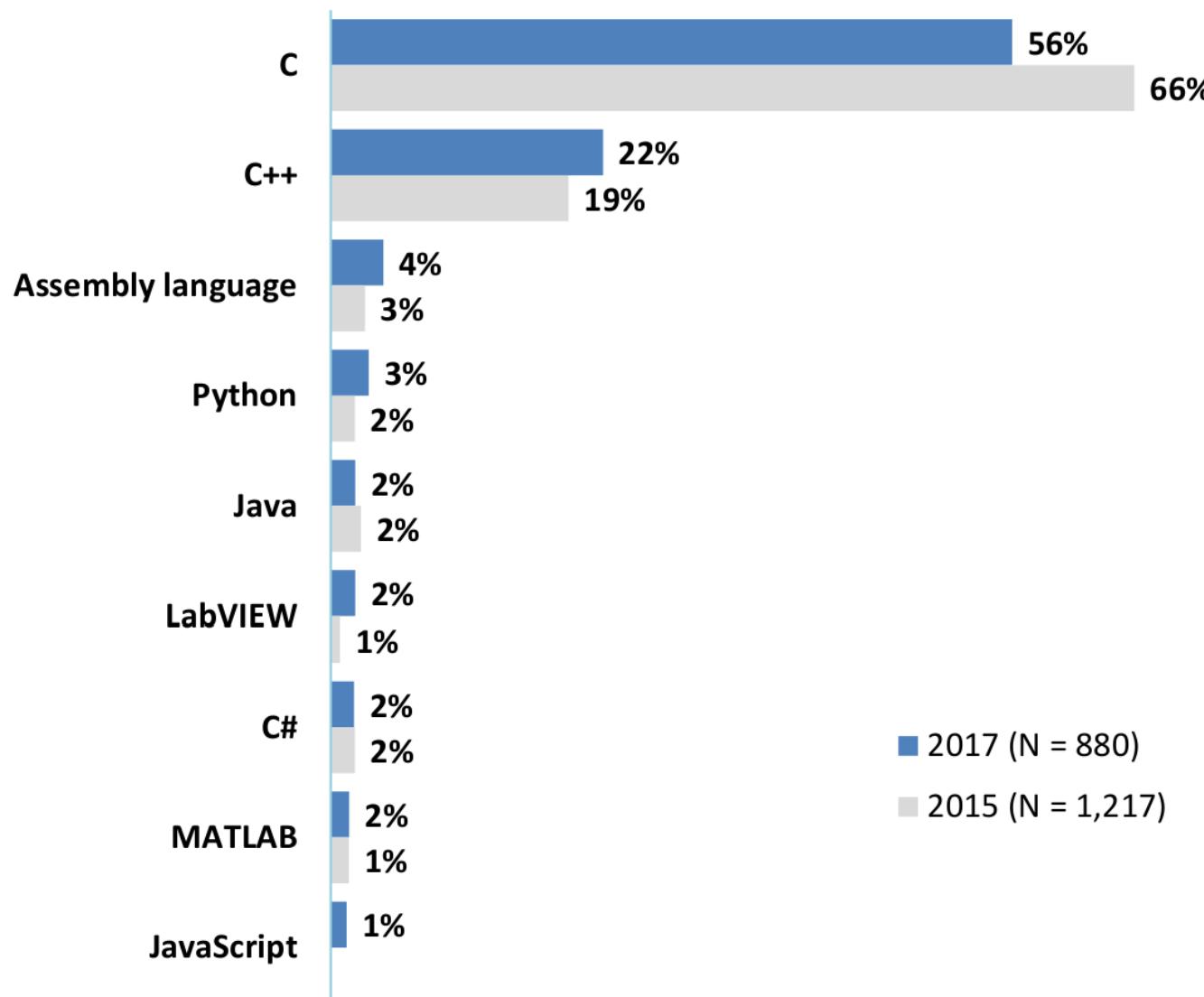


High level

Visual programming (Labview, Simulink, etc.)

- Assembly
- C
- C++
- Basic
- Java
- Python
- Matlab
-

أشهر لغات البرمجة الشائعة لبرمجية الميكانيات المصغرة :

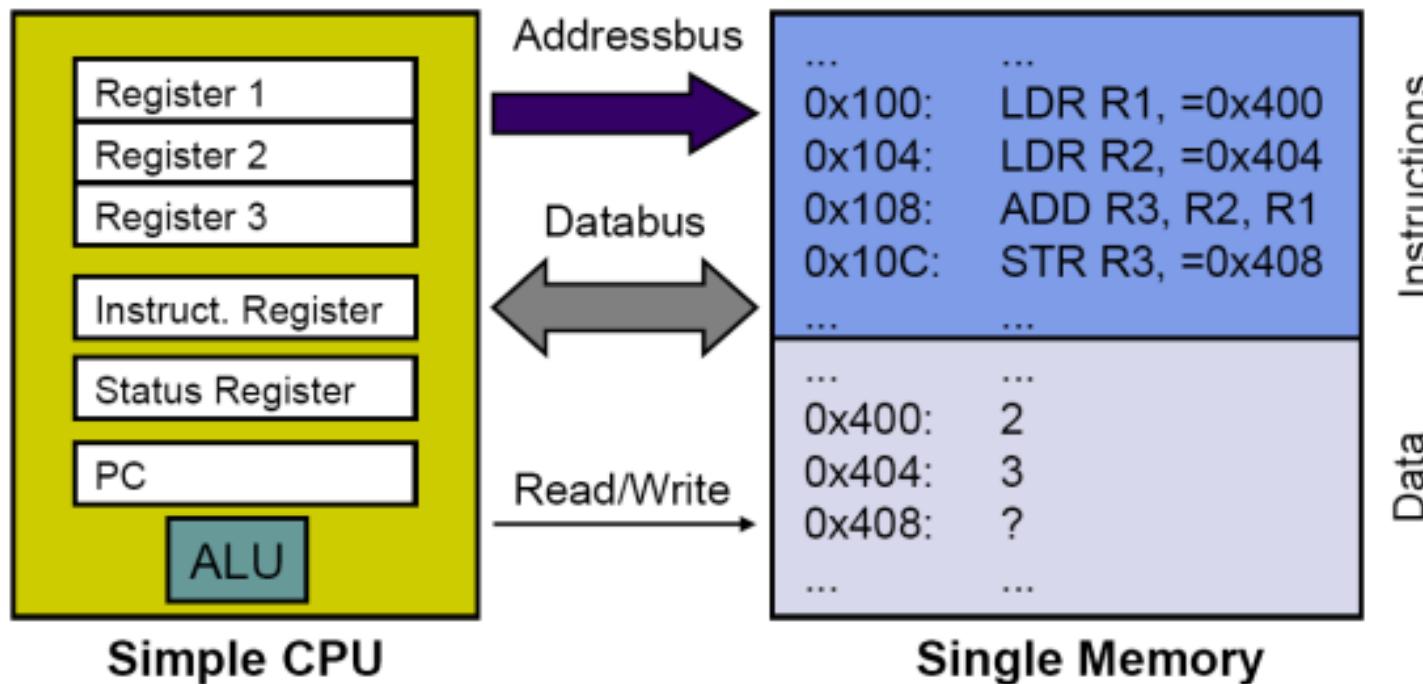


Source: <https://www.cnx-software.com/2017/08/15/aspencore-2017-embedded-markets-study-programming-languages-operating-systems-mcu-vendors-and-more/>

معاييرية تصميم بنية المعالجات :

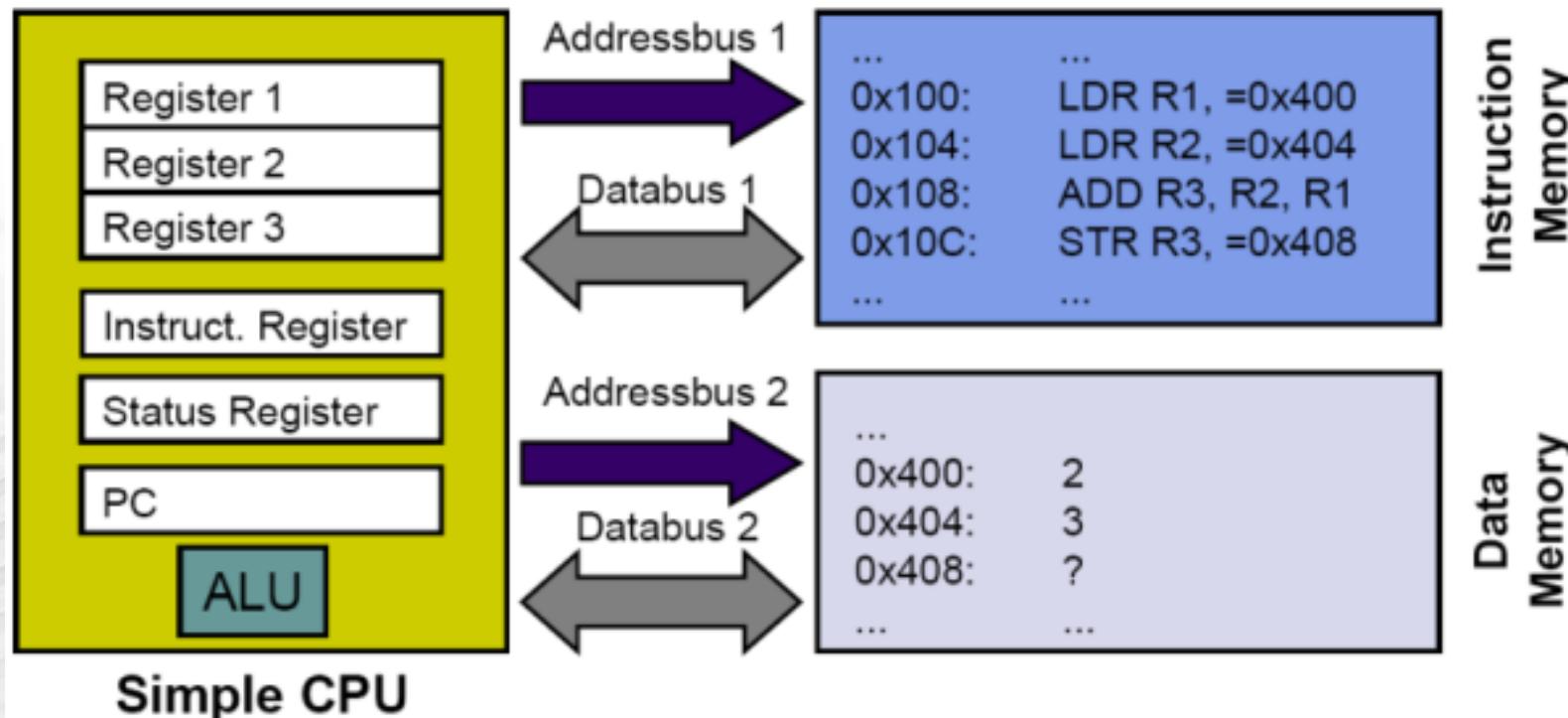
معاييرة Von-Neumann: تعتمد على ناقل وحيد لنقل التعليمات والبيانات بين الذاكرة (الوحيدة) ووحدة المعالجة المركزية بحيث:

- (1) يقوم المعالج بجلب كود التعليمات من الذاكرة.
- (2) يقوم بقراءة البيانات من الذاكرة.
- (3) إجراء العمليات على البيانات.
- (4) إعادة كتابة تلك البيانات على الذاكرة.



معاييرية تصميم بنية المعالجات :

معاييرة Harvard: ناقلين منفصلين أحدهما لنقل التعليمات والآخر لنقل البيانات وتحتلت ذاكرة البيانات عن ذاكرة التعليمات حيث أن لكل ذاكرة خطوط عنونة وتحكم ومحركات مختلفة، وبالتالي تم عملية قراءة التعليمات والبيانات في نفس الوقت ...



RISC:

Reduced Instruction Set Computer.

30 ~ 130 Instruction

CISC:

Complex Instruction Set Computer.

150 ~ 1000 Instruction

MISC:

Minimum Instruction Set Computer.

15 ~ 30 Instruction

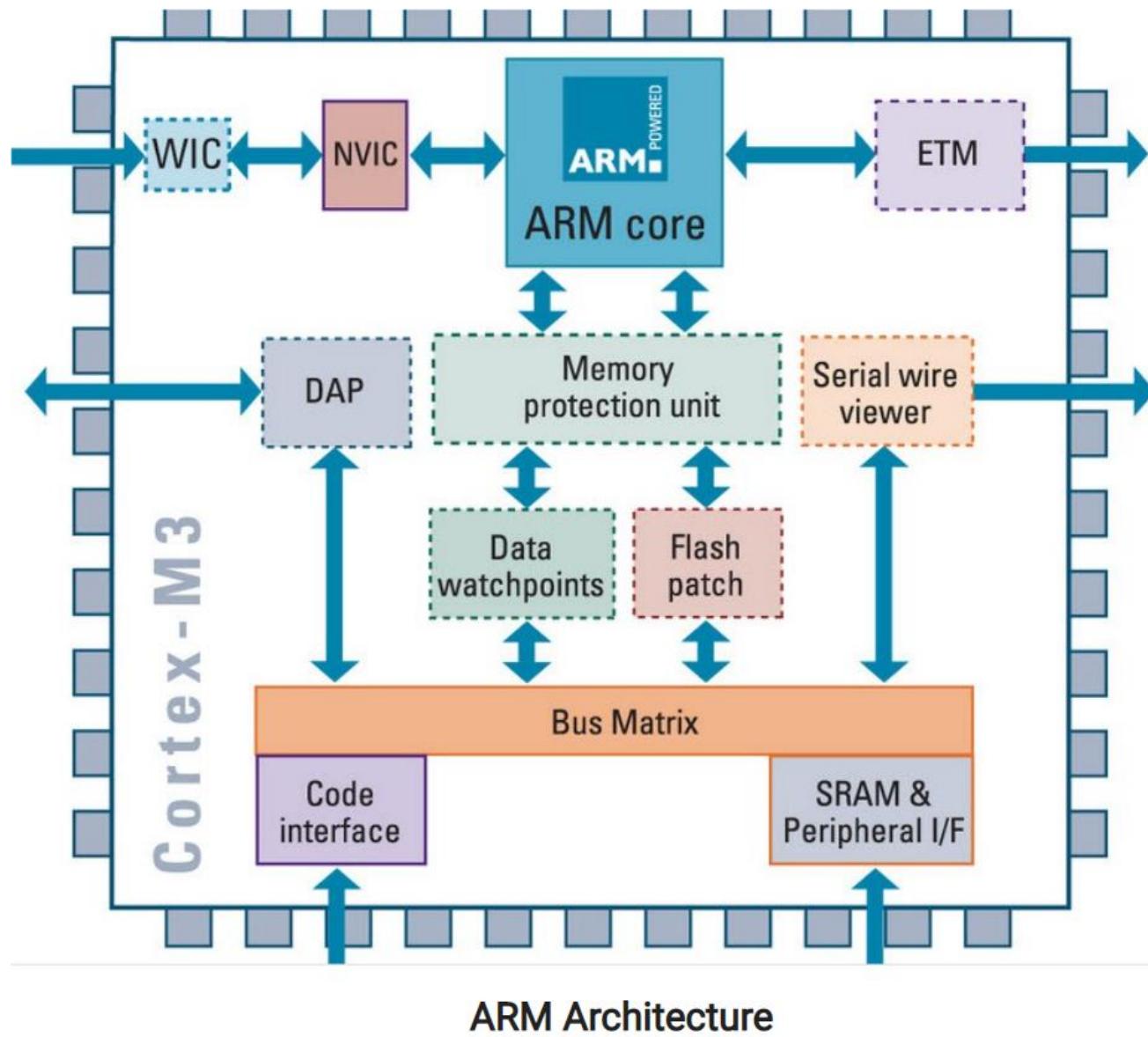
المعالجات المبنية بواسطة ARM

ARM هي اختصار لـ Advanced RISC Machines ✦

تقوم شركة ARM بتطوير الهياكل المعمارية للمعالجات المختلفة وأيضاً تصميم نوى المعالجات المبنية على معمارية RISC ومن ثم تقوم بإعطاء رخص للشركات لاستخدامها في تصميم منتجاتها المختلفة على سبيل المثال system on a chip . system on module (SOM) و (SOC)

تمتاز معالجات ARM بكلفتها المنخفضة واستهلاكها المنخفض للطاقة وأيضاً توليد حرارة أقل مقارنةً مع نظيراتها من المعالجات، لذا تعتبر المعالجات المثلالية لاستخدامها في الأجهزة المحمولة التي تعتمد على البطاريات في تغذيتها كالهواتف الذكية والكمبيوترات اللوحية computer tap وأيضاً أجهزة الكمبيوتر المحمولة laptop وغيرها العديد من الأنظمة المدمجة.

ARM Architecture



الهيكلية ARM - CORTEX هي عبارة عن مجموعة كبيرة من المعماريات والأنوية 32/64bit المنتشرة في عالم الأنظمة المدمجة، حيث تقسم المعالجات المبنية على معمارية CORTEX إلى ثلاثة عائلات فرعية وهي:

□ **CORTEX-A** : ويرمز الحرف A إلى التطبيقات Applications ، وهي عبارة عن سلسلة من المعالجات توفر مجموعة من الحلول التي للأجهزة التي تتطلب إنجاز مهام حوسبة معقدة مثل استضافة نظام تشغيل كامل ك Linux أو Android وغيرها والتي تدعم العديد من التطبيقات، وتستخدم هذه المعمارية في أغلب الهواتف الذكية

□ **Cortex-M** : والتي ترمز إلى Embedded وتميز هذه المعمارية بالعديد من الخصائص منها الكفاءة في استخدام الطاقة أيضاً التكلفة المنخفضة للمعالجات التي تستخدم هذه المعمارية وهي مصممة من أجل المتحكمات المستخدمة في تطبيقات إنترنت الأشياء IOT، التحكم في المحركات

□ **Cortex-R** : والتي ترمز إلى Real Time ، حيث تقوم المعالجات التي تستخدم هذه المعمارية بتقديم أداء عالي في مجالات أنظمة الزمن الحقيقي

arm CORTEX®-M0

Nested vectored
interrupt controller

Wake-up interrupt
controller

CPU
Armv6-M

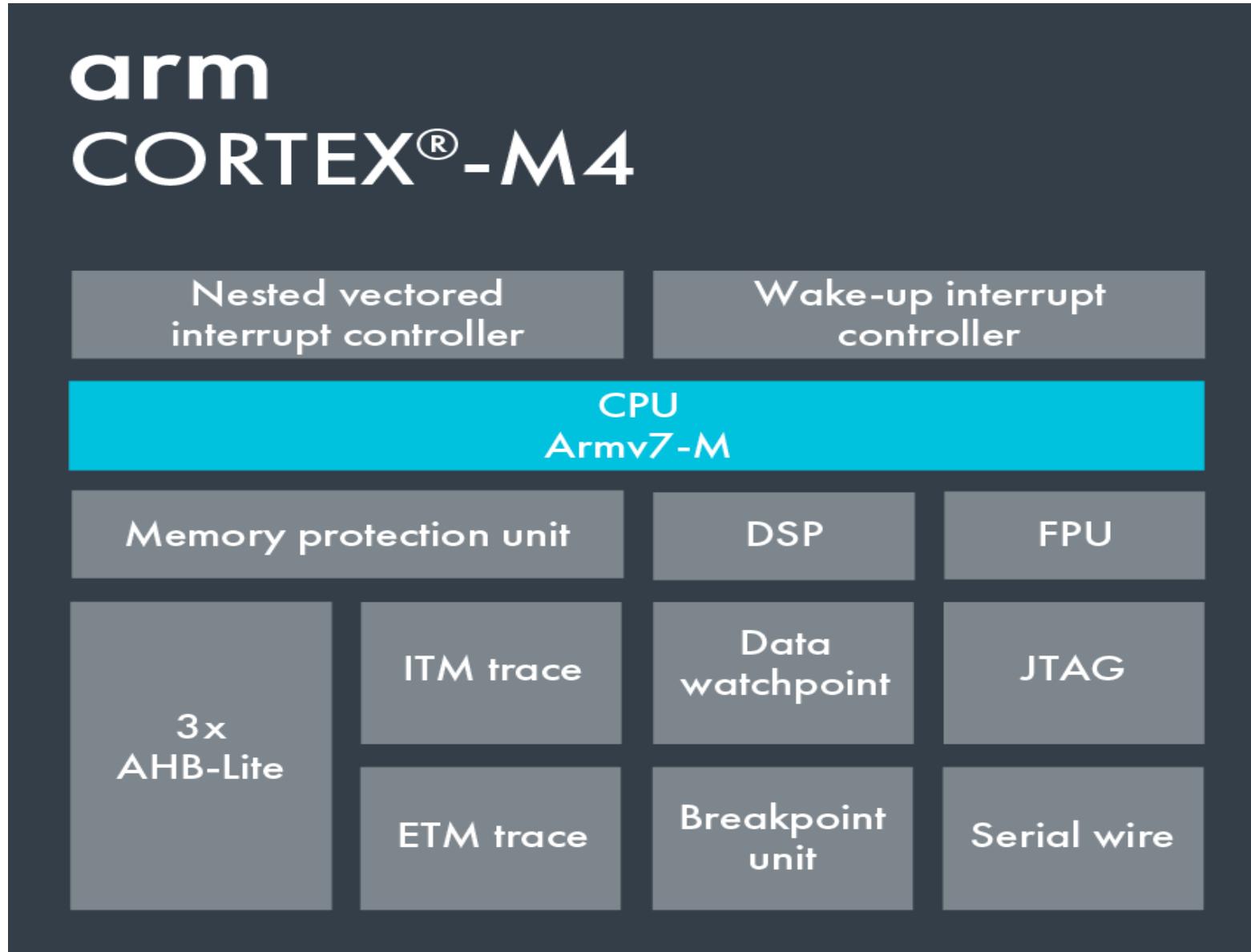
AHB-Lite

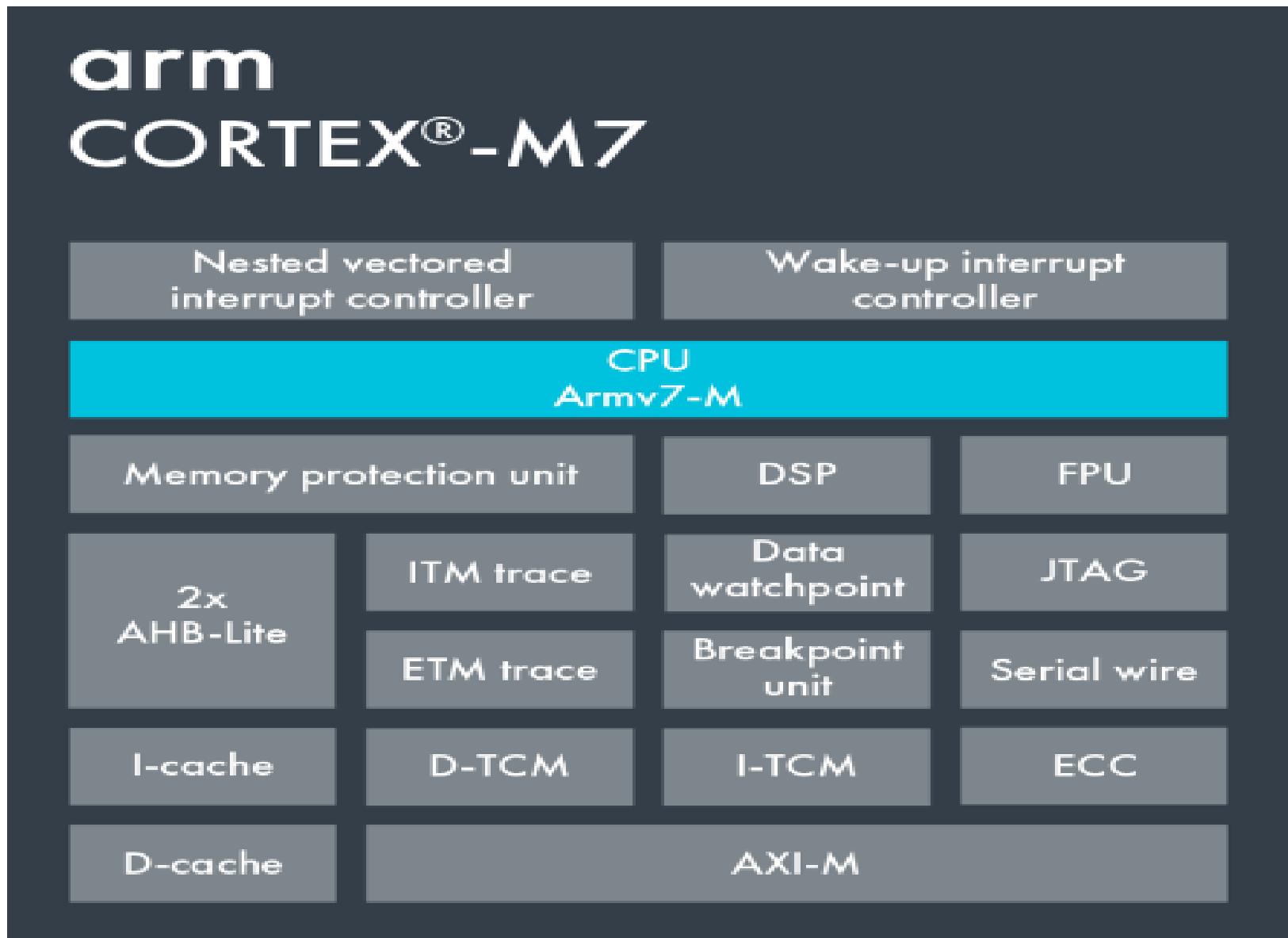
Data
watchpoint

JTAG

Breakpoint
unit

Serial wire





المتحكم STM32G0

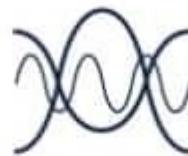


STM32 Mainstream MCUs 32-bit Arm® Cortex®-M



STM32G4

- Arm Cortex-M4 + FPU at 170 MHz – 213 DMIPS
- Rich analog peripheral set
- High-resolution timer
- Mathematical accelerators



Instrumentation
& Measurement



Digital Power



Motor Control

Mixed-signal MCUs

STM32F3

- Arm Cortex-M4 + FPU at 72 MHz – 90 DMIPS
- Rich analog peripheral set
- High-resolution timer

STM32F1

- Arm Cortex-M3 at 72 MHz – 61 DMIPS
- STM32 Foundation line
- Wide range of performance and peripherals, easy-to-use tools

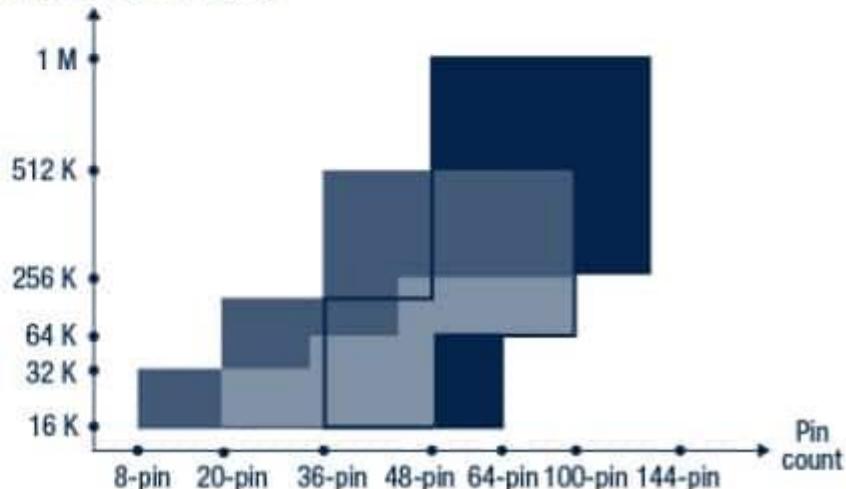
STM32G0

- Arm Cortex-M0+ at 64 MHz – 59 DMIPS
- Maximum I/O count per package
- Advanced function is analog, low-power, control

STM32F0

- Entry-level MCU for cost-sensitive operations
- Arm Cortex-M0 at 48 MHz – 38 DMIPS

Flash memory size (bytes)

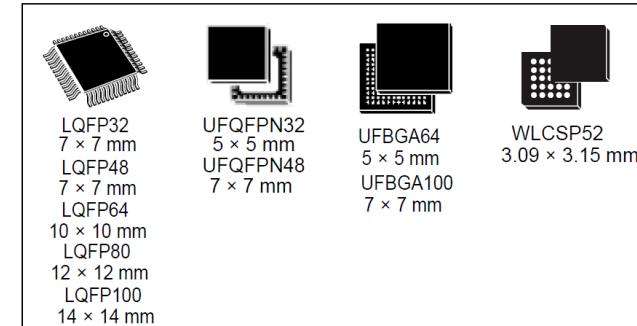
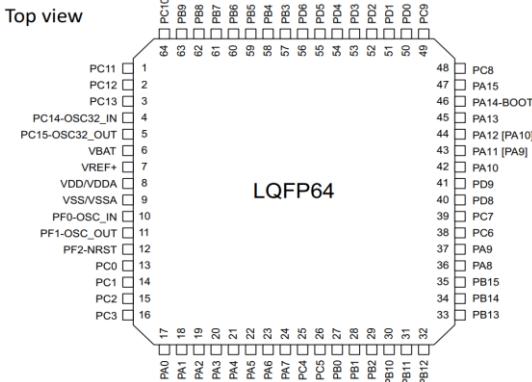


STM32G0B1CE المتحكم

<https://www.st.com/en/microcontrollers-microprocessors/stm32g0b1ce.html>

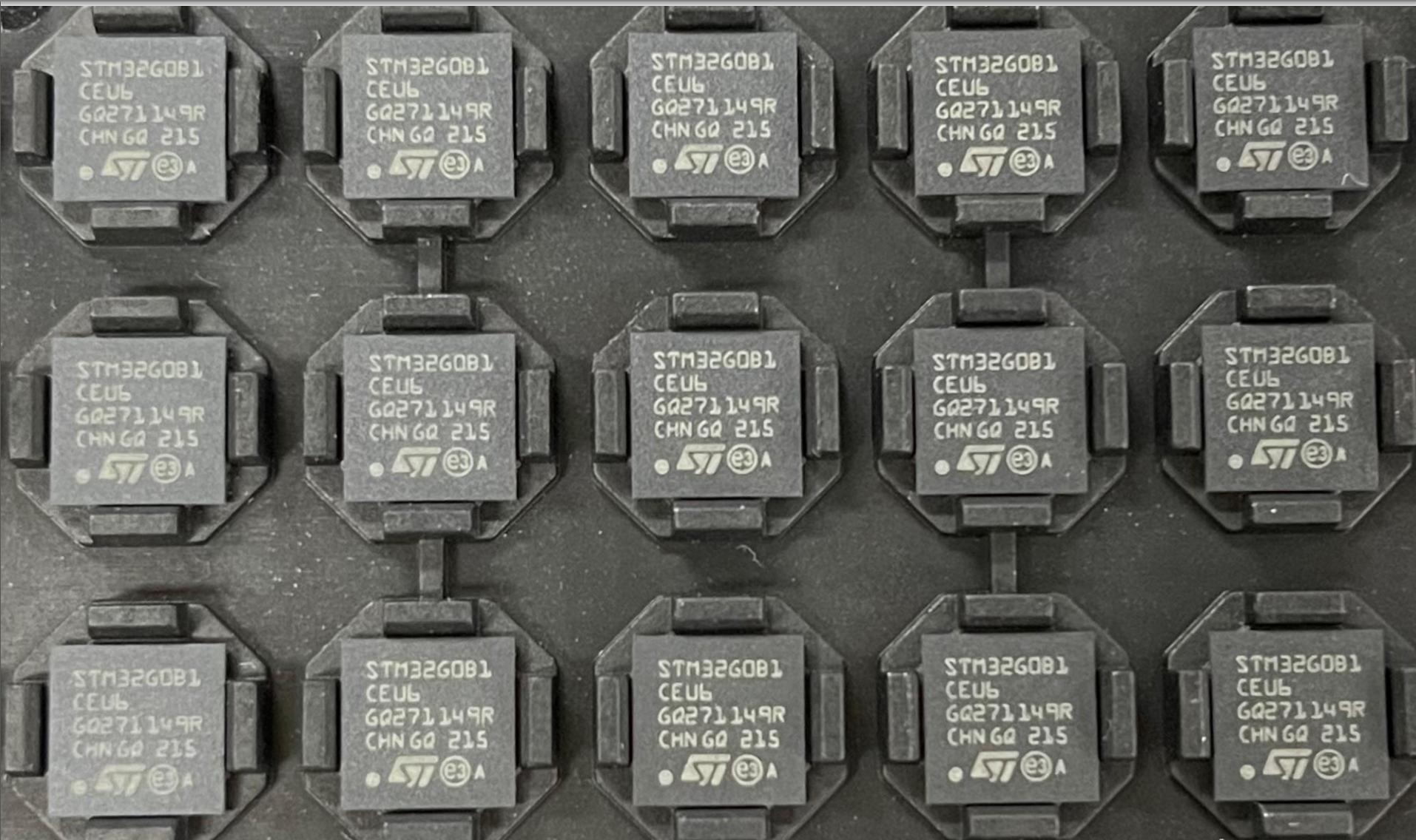
System	
Power supply	POR/PDR/PVD/BOR
Xtal oscillator	32 kHz + 4 to 48 MHz
Internal RC oscillators	32 kHz ($\pm 5\%$) + 16 MHz ($\pm 1\%$)
Internal RC oscillator	48 MHz (auto trimming on ext. synchro)
PLL + Prescaler	
Clock control	
RTC/AWU	
Systick timer	
2x watchdogs (independent and window)	
94 I/Os on 100 pins	
Cyclic redundancy check (CRC)	
Arm® Cortex®-M0+ CPU Up to 64 MHz	
Nested vector interrupt Controller (NVIC)	
SW debug	
Memory Protection Unit	
AHB-Lite bus matrix	
APB bus	
Up to 512-Kbyte Flash memory	
Up to 144-Kbyte SRAM	
20-byte backup registers	
Boot ROM	
12-channel DMA	
Analog	
Temp. sensor	
1x 12-bit ADC SAR 16-channels / 2.5 MSPS	
1x 12-bit DAC 2ch	
3x comparators	
Connectivity	
3x SPI (I ² S)	
6x USART (3x with LIN, smartcard, IrDA, modem control)	
2x LPUART	
3x I ² C Fast Mode Plus (2x SMBus, PMBus)	
2x FDCAN	
USB FS 2.0 Device (crystal less) Host	
USB Power Delivery (incl. BMC + PHY)	
Control	
1x 32-bit timer	
1x 16-bit Motor C. timer $f_{MAX} = 128$ MHz 4 PWM + 3 compl.	
6x 16-bit timers one with $f_{MAX} = 128$ MHz	
2x Low-power timers	

Top view



STM32G0B1CE المتحكم

<https://www.st.com/en/microcontrollers-microprocessors/stm32g0b1ce.html>



- ❖ Core: Arm® 32-bit Cortex®-M0+ CPU, frequency up to 64 MHz
- ❖ -40°C to 85°C/105°C/125°C operating temperature
- ❖ Memories
 - Up to 512 Kbytes of Flash memory
 - 144 Kbytes of SRAM

❖ Clock management

- 4 to 48 MHz crystal oscillator
- 32 kHz crystal oscillator with calibration
- Internal 16 MHz RC with PLL option ($\pm 1\%$)
- Internal 32 kHz RC oscillator ($\pm 5\%$)

❖ Reset and power management

❖ Voltage range: 1.7 V to 3.6 V

- ❖ Up to 94 fast I/Os
 - All mappable on external interrupt vectors
- ❖ 12-channel DMA controller with flexible mapping
- ❖ 12-bit, 0.4 µs ADC (up to 16 ext. channels)
- ❖ 12-bit DACs, low-power sample-and-hold
- ❖ Low-power modes:
 - Sleep, Stop, Standby, Shutdown

- ❖ Two fast low-power analog comparators
- ❖ 14 timers (two 128 MHz capable)
- ❖ Communication interfaces
- ❖ Three I2C-bus
- ❖ 6× USARTs with master/slave
- ❖ Two low-power UART
- ❖ Three SPIs (32 Mbit/s) with 4- to 16-bit

كيفية الحصول على معلومات فنية عن المتحكم؟

موقع الشركة المصنعة

<https://www.st.com/en/microcontrollers-microprocessors/stm32g0b1ce.html>

تحتوي ال Datasheet على المواصفات الكهربائية وتوزع الأقطاب

تحتوي ال User Manual على المسجلات الداخلية وتفاصيل برمجة المتحكم

المنتديات والمواقع التعليمية المختلفة



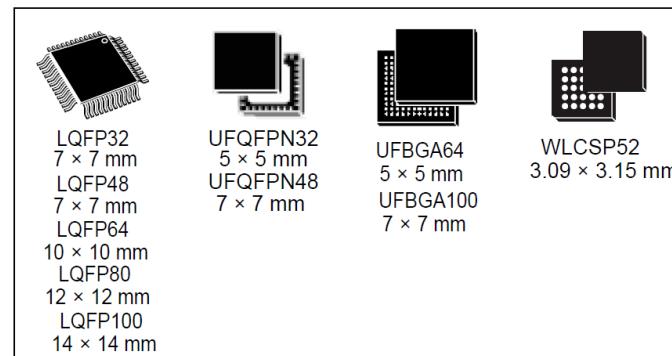
STM32G0B1xB/xC/xE

Arm® Cortex®-M0+ 32-bit MCU, up to 512KB Flash, 144KB RAM,
6x USART, timers, ADC, DAC, comm. I/Fs, 1.7-3.6V

Datasheet - production data

Features

- Core: Arm® 32-bit Cortex®-M0+ CPU, frequency up to 64 MHz
- 40°C to 85°C/105°C/125°C operating temperature
- Memories
 - Up to 512 Kbytes of Flash memory with protection and securable area, two banks, read-while-write support
 - 144 Kbytes of SRAM (128 Kbytes with HW parity check)
- CRC calculation unit
- Reset and power management
 - Voltage range: 1.7 V to 3.6 V
 - Separate I/O supply pin (1.6 V to 3.6 V)
 - Power-on/Power-down reset (POR/PDR)
 - Programmable Brownout reset (BOR)
 - Programmable voltage detector (PWD)



- Communication interfaces
 - Three I²C-bus interfaces supporting Fast-mode Plus (1 Mbit/s) with extra current sink, two supporting SMBus/PMBus and wakeup from Stop mode
 - Six USARTs with master/slave synchronous SPI; three supporting ISO7816 interface, LIN, IrDA capability, auto baud rate detection and wakeup feature



life.augmented

RM0444 Reference manual

STM32G0x1 advanced Arm[®]-based 32-bit MCUs

Introduction

This reference manual complements the datasheets of the STM32G0x1 microcontrollers, providing information required for application and in particular for software development. It pertains to the superset of feature sets available on STM32G0x1 microcontrollers.

For feature set, ordering information, and mechanical and electrical characteristics of a particular STM32G0x1 device, refer to its corresponding datasheet.

For information on the Arm[®] Cortex[®]-M0+ core, refer to the Cortex[®]-M0+ technical reference manual.

Related documents

- “Cortex[®]-M0+ Technical Reference Manual”, available from: <http://infocenter.arm.com>
- PM0223 programming manual for Cortex[®]-M0+ core^(a)
- STM32G0x1 datasheets^(a)
- AN2606 application note on booting STM32 MCUs^(a)

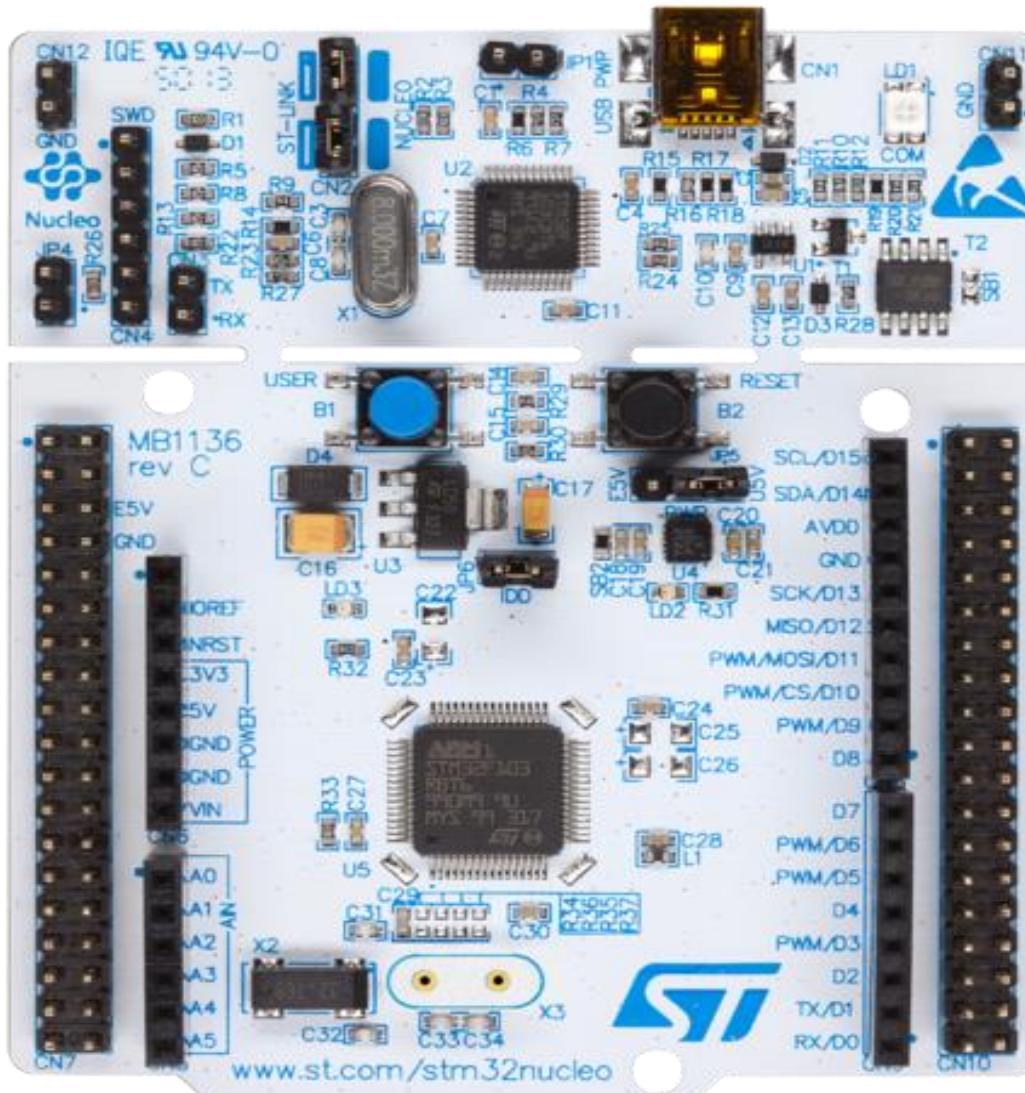
STM32 متحكمات استخدام استخدام كيفية

هناك طريقتين أساسيتين لاستخدام متحكمات STM32:

- من خلال بناء وتصميم لوحة خاصة
- من خلال استخدام إحدى اللوحة التطويرية المتاحة

أنواع اللوحة التطويرية المتوفرة Boards type

Nucleo board



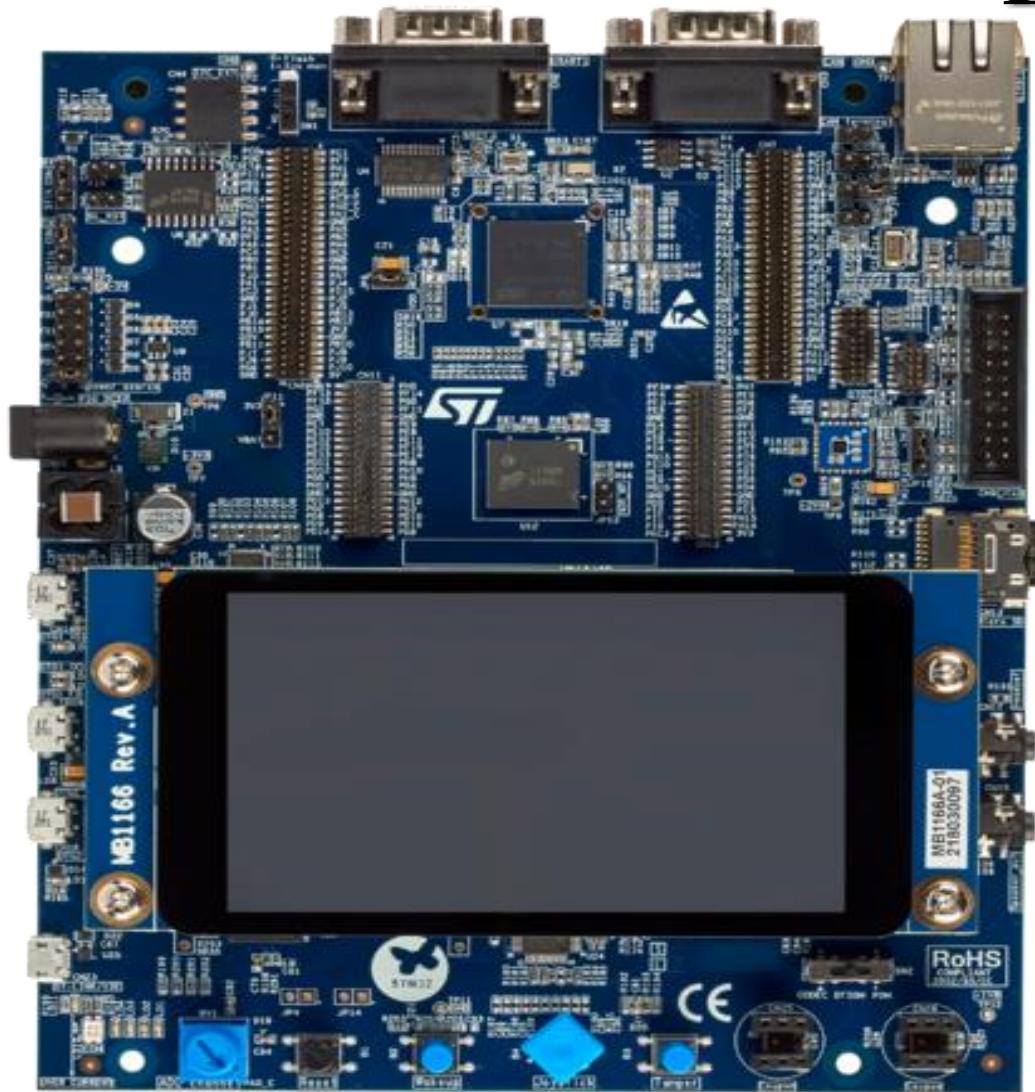
أنواع اللوحة التطويرية المتوفرة Boards type

Discovery kit



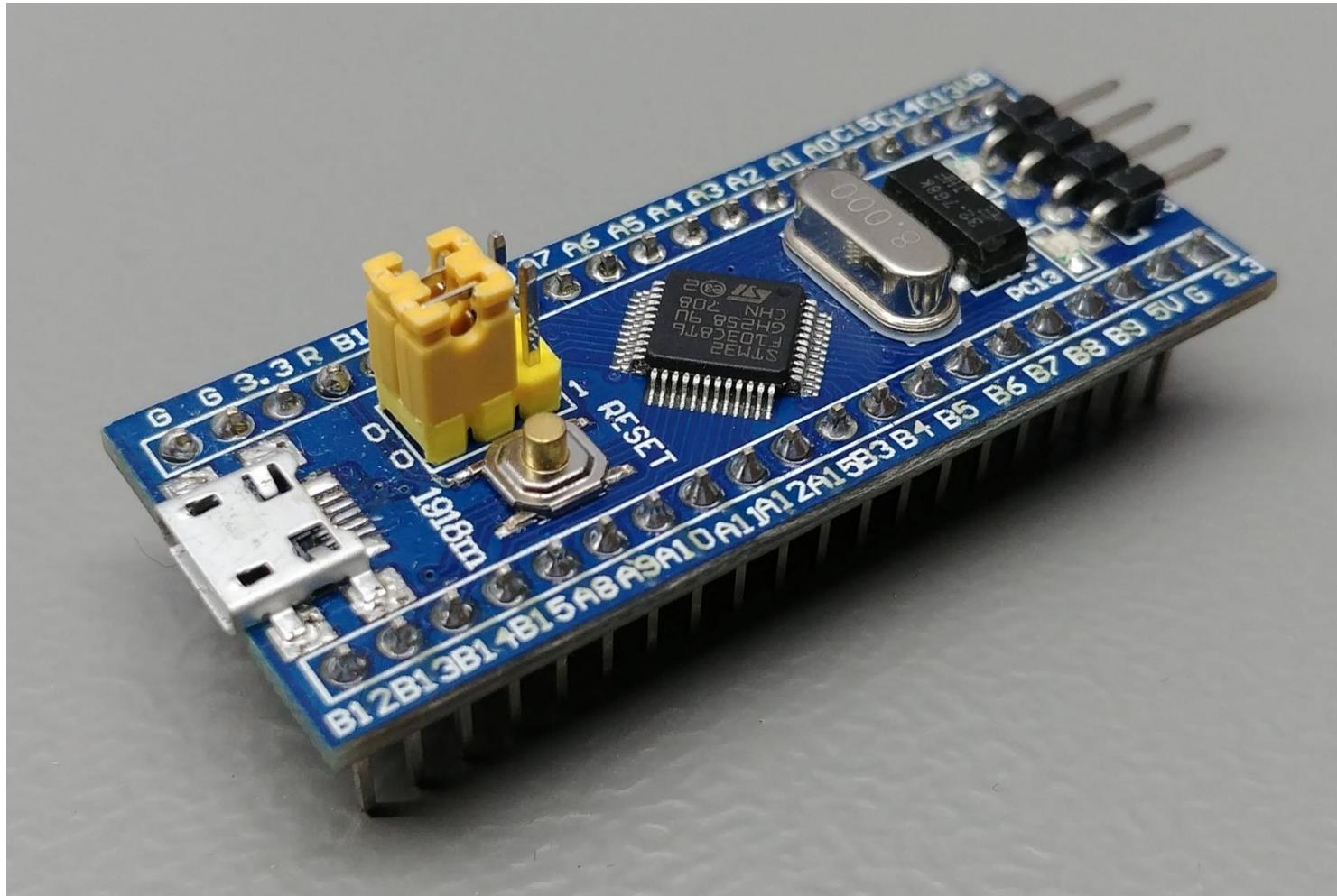
أنواع اللوحة التطويرية المتوفرة Boards type

Eval board



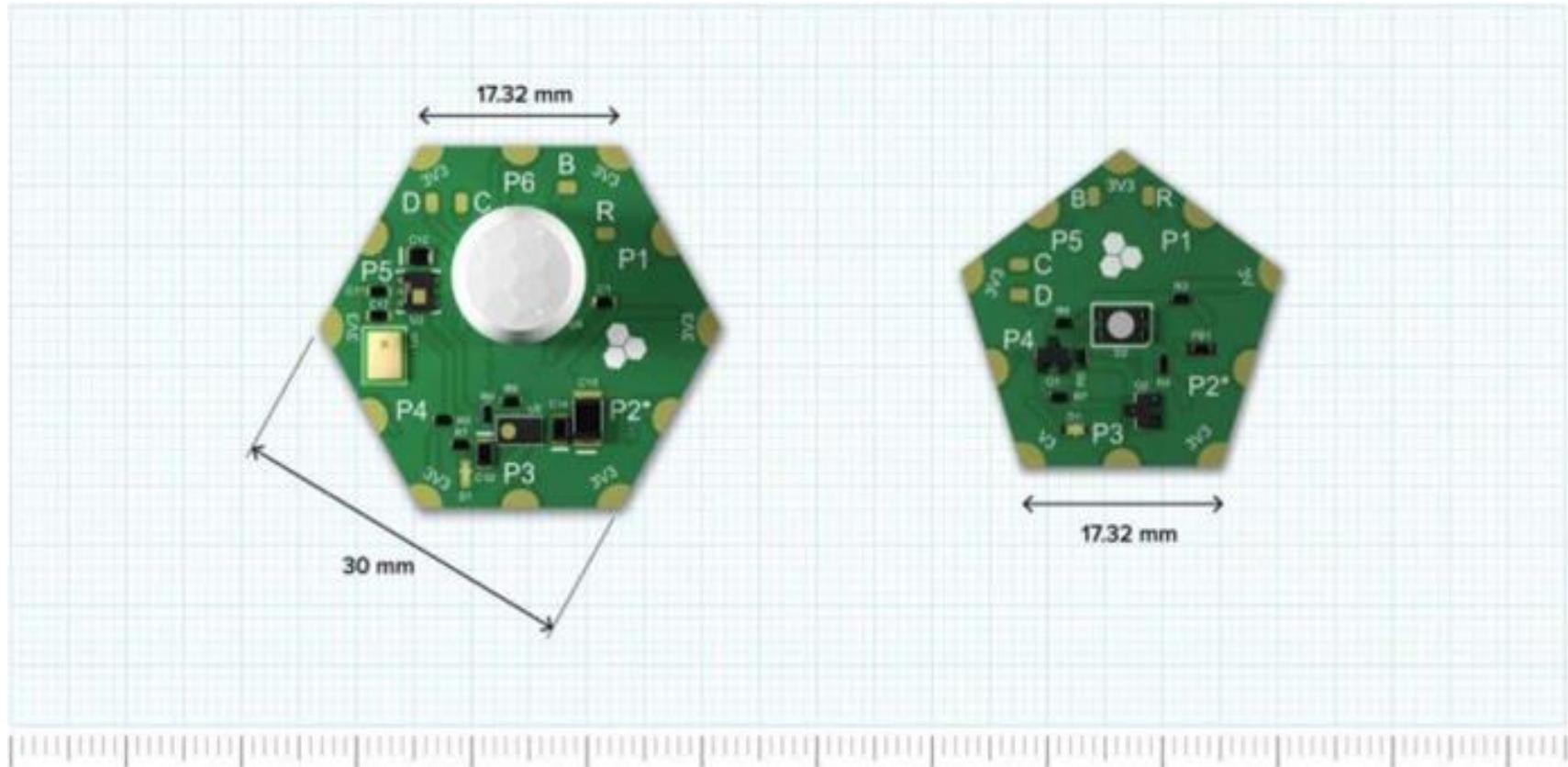
أنواع اللوحة التطويرية المتوفرة Boards type

Blue Pill 



أنواع اللوحات التطويرية المتوفرة Boards type

مودولات Hexabitz



STM32 تغذية متحكمات

أقطاب التغذية في متحكمات STM32 هي :

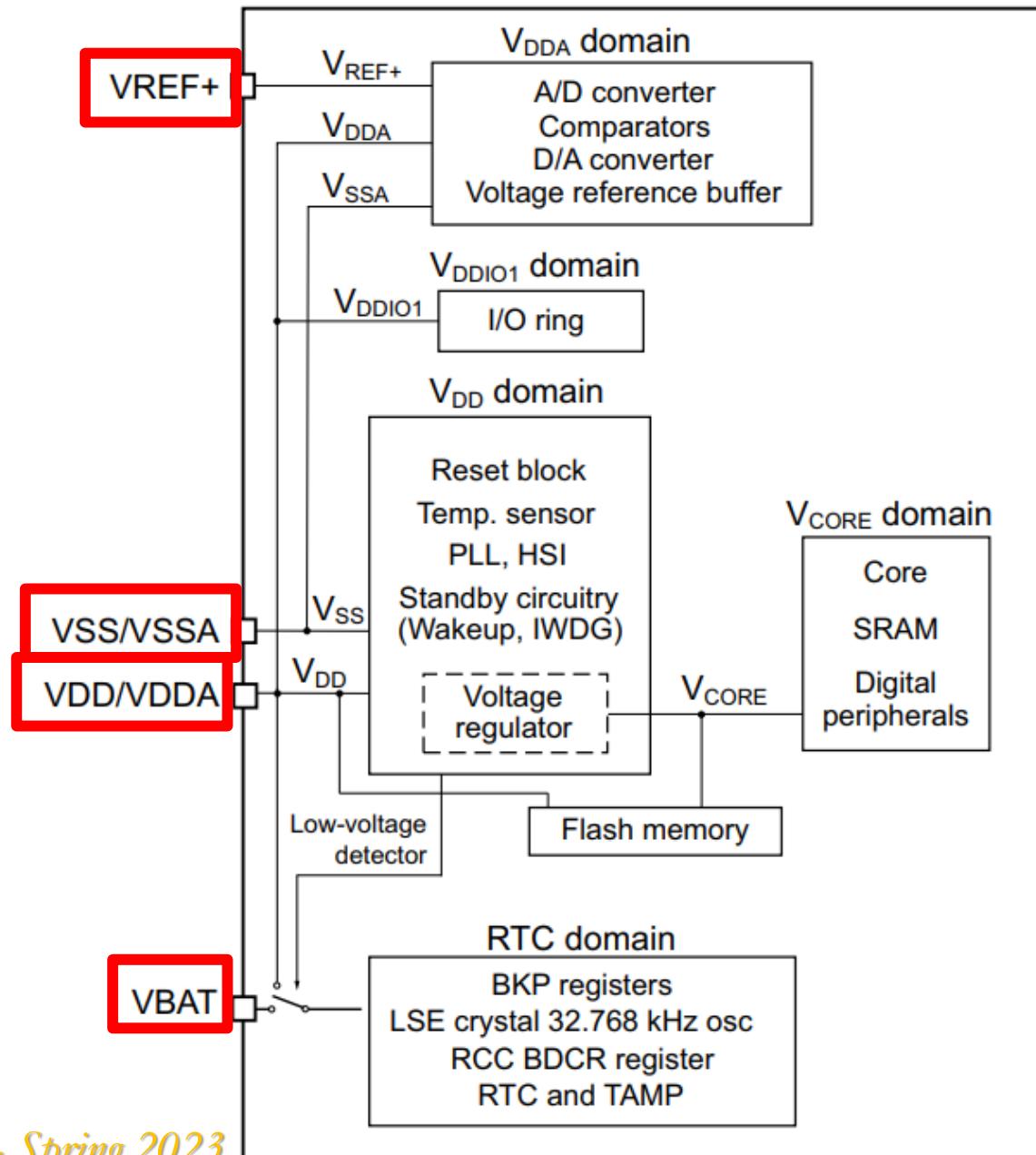
VDD/VDDA

VSS/VSSA

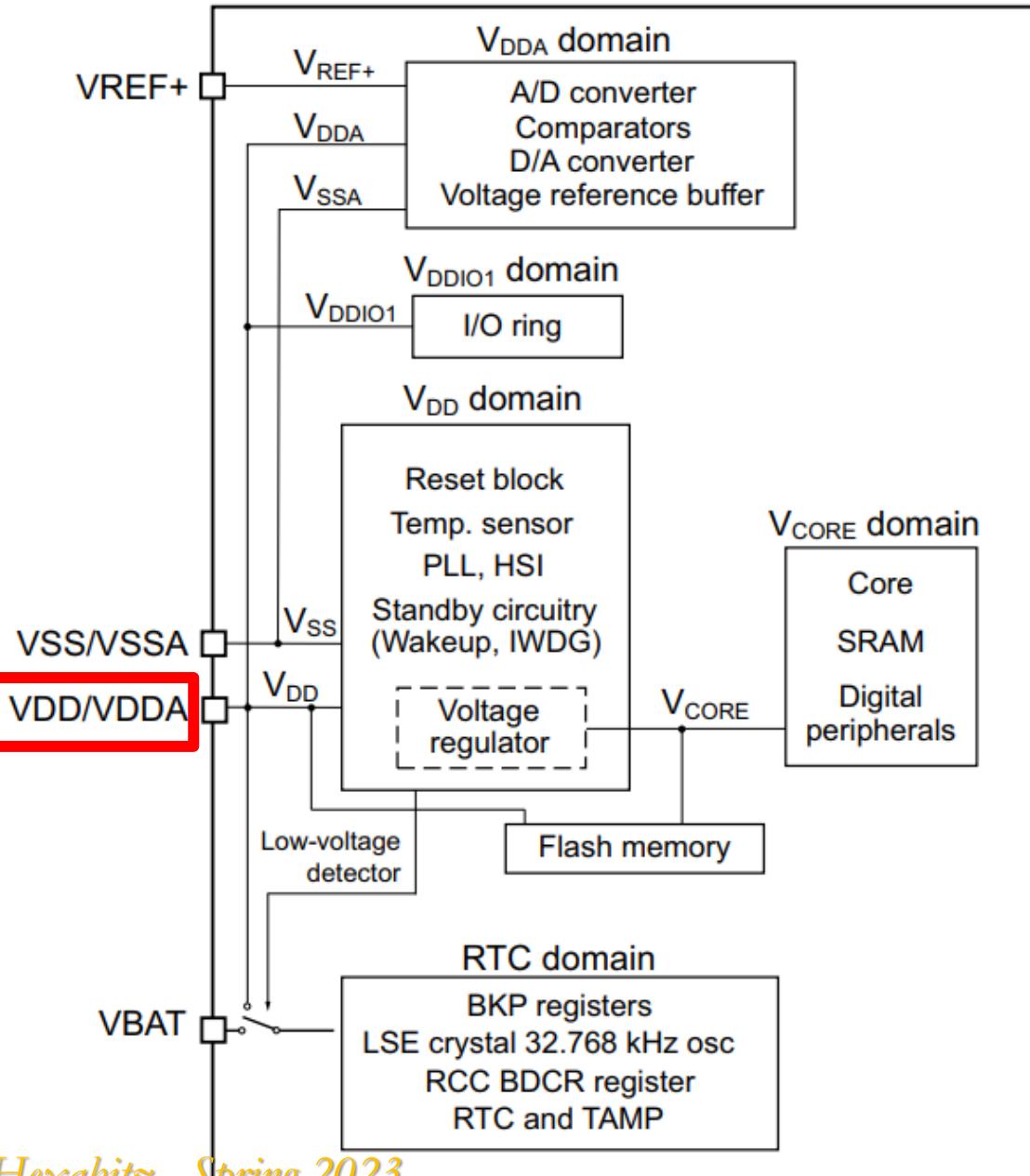
VREF+

VBAT

تغذية متحكمات STM32



تغذية متحكمات STM32



:VDD/VDDA

- يتراوح الجهد اللازم

- تزويده

لتشغيل متحكم بين stm32

1.7V to 3.6V

- أقل جهد يمكن تغذية

- المتحكم به هو 1.7V

- وهو العتبة الدنيا

المسوحة(min) VPDR(min)

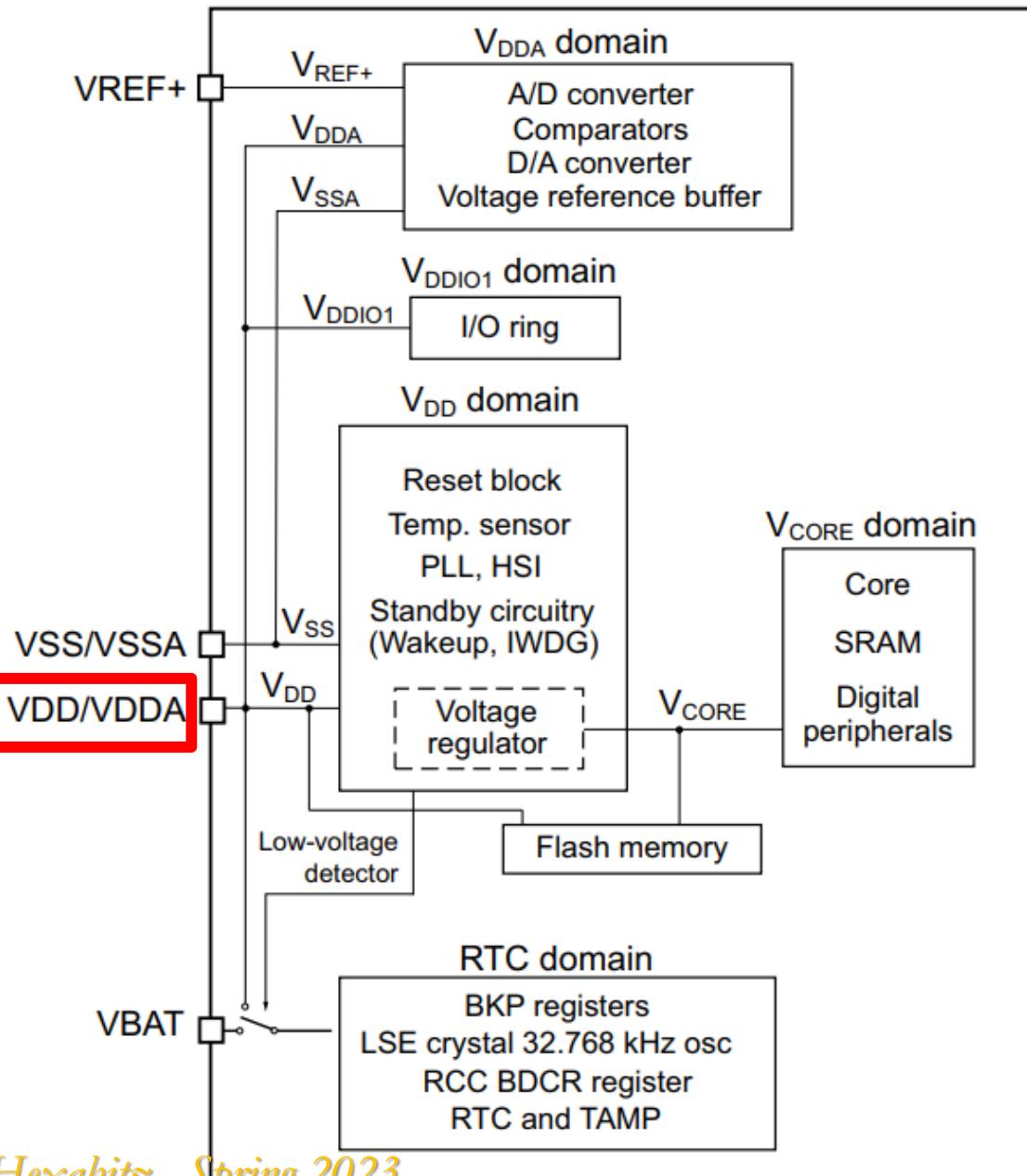
- أكبر جهد يمكن تغذية

- المتحكم به هو 3.6V

- وهو العتبة العليا

المسوحة(max) VPoR(max)

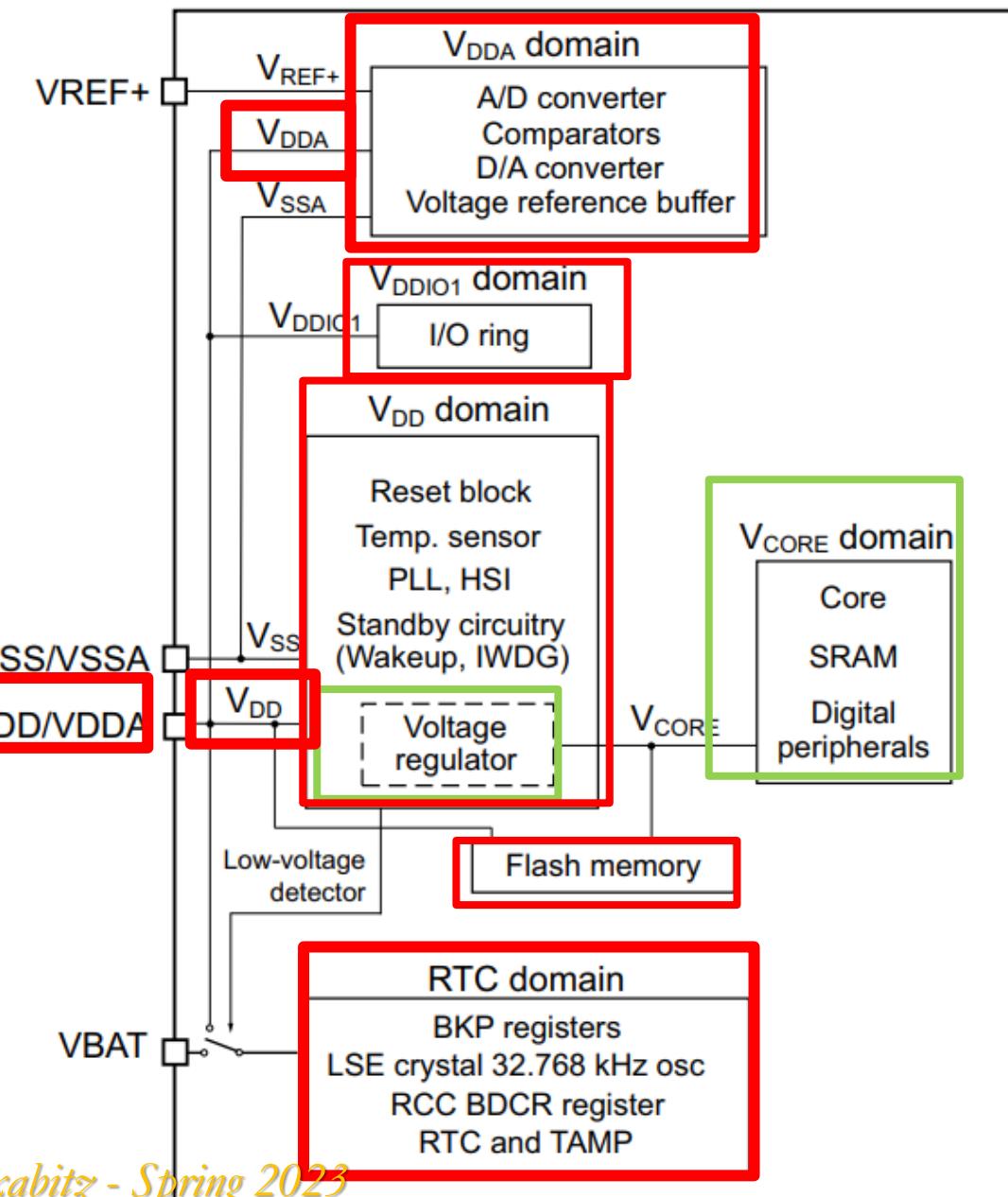
تغذية متحكمات STM32



:VDD/VDDA

• عندما ينخفض جهد تغذية المتحكم عن الدنيا العتبة أو VPDR(min) عندما يرتفع عن العتبة العليا (VPoR(max) reset يتم عمل للمتحكم

تغذية متحكمات STM32



:VDD/VDDA

ويقوم بتغذية كلاً من :

منظم الجهد الداخلي

المسؤول عن تغذية
المعالج

المبدلات التشابهية

والمقارنات

Flash memory

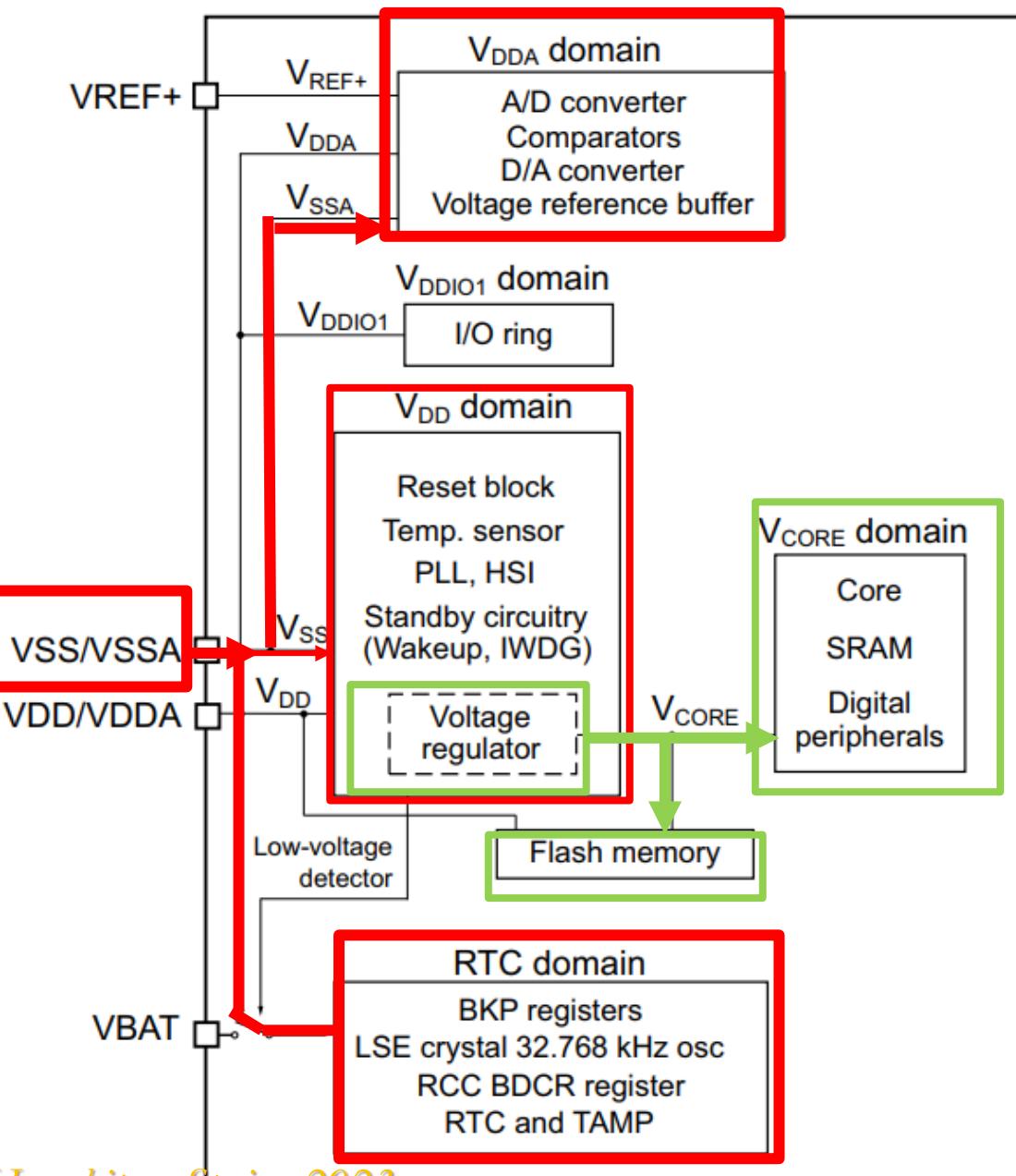
RTC

المدخل والمخرج

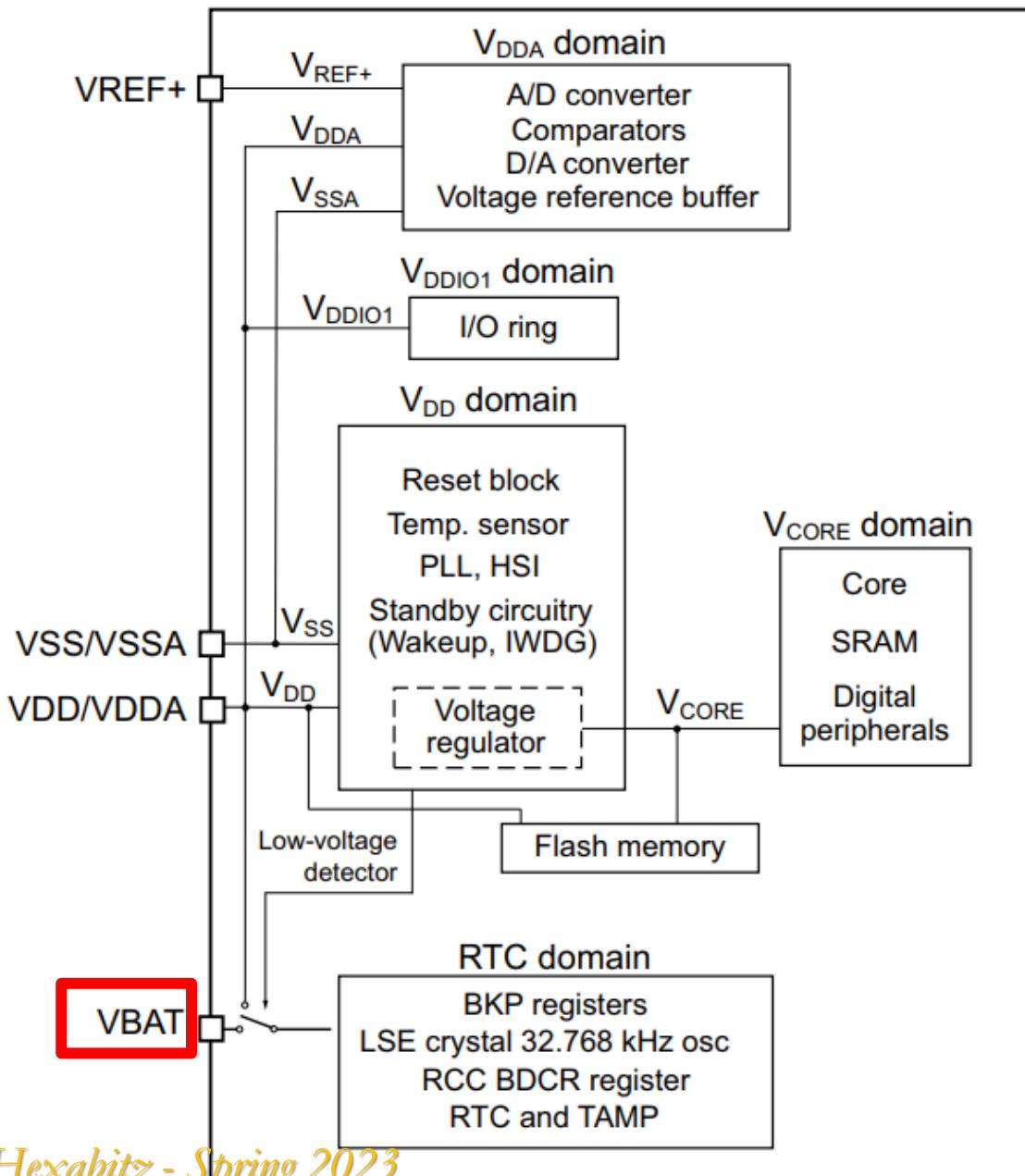
I/O

تغذية متحكمات STM32

:VSS/VSSA



تغذية متحكمات STM32



:VBAT

• يتراوح بين

• 1.55V to 3.6V

• وهو عبارة عن جهد
التغذية الخاص بالـ:

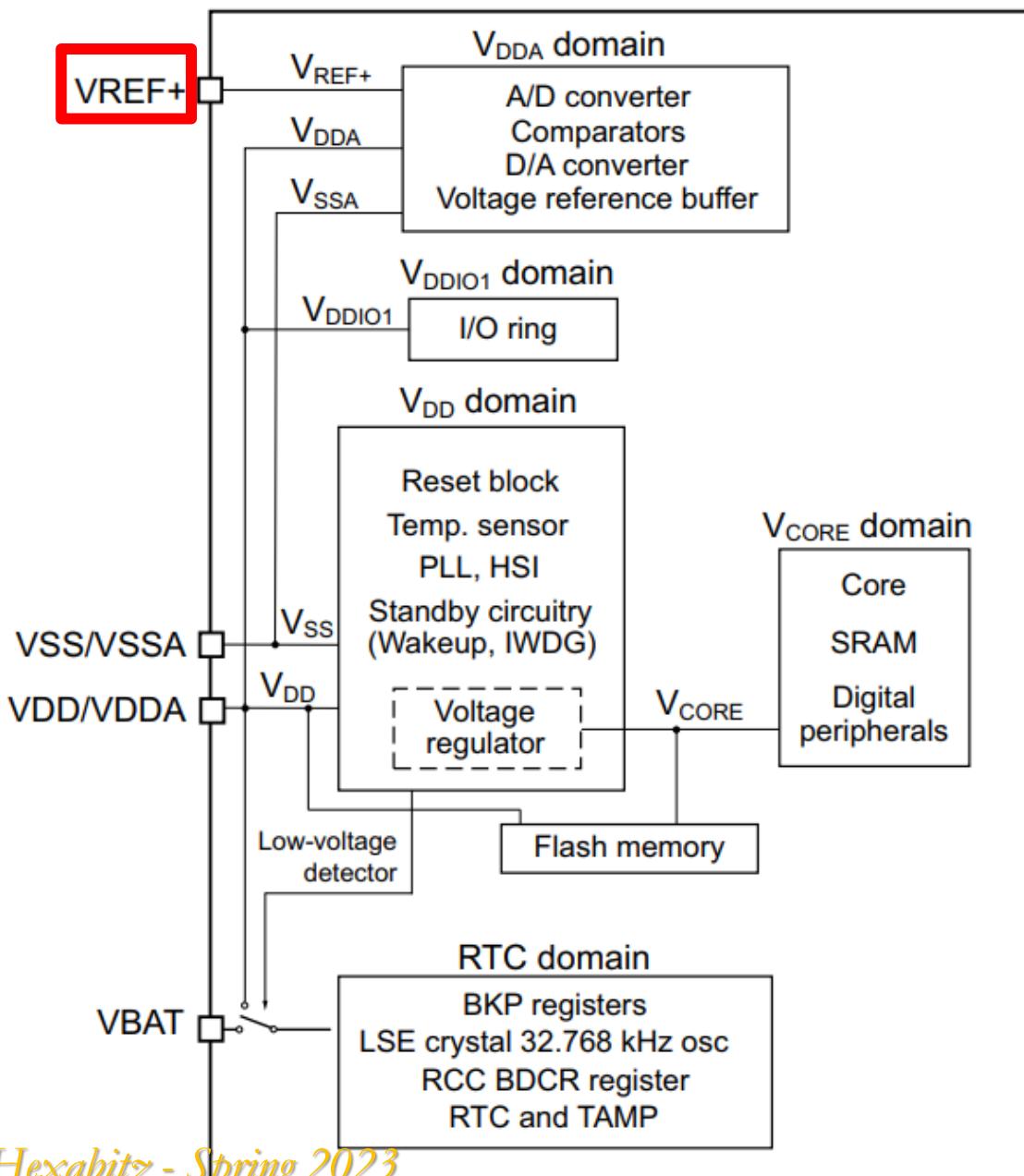
• RTC

• backup registers

• وذلك عند عدم وجود

• VDD

تغذية متحكمات STM32



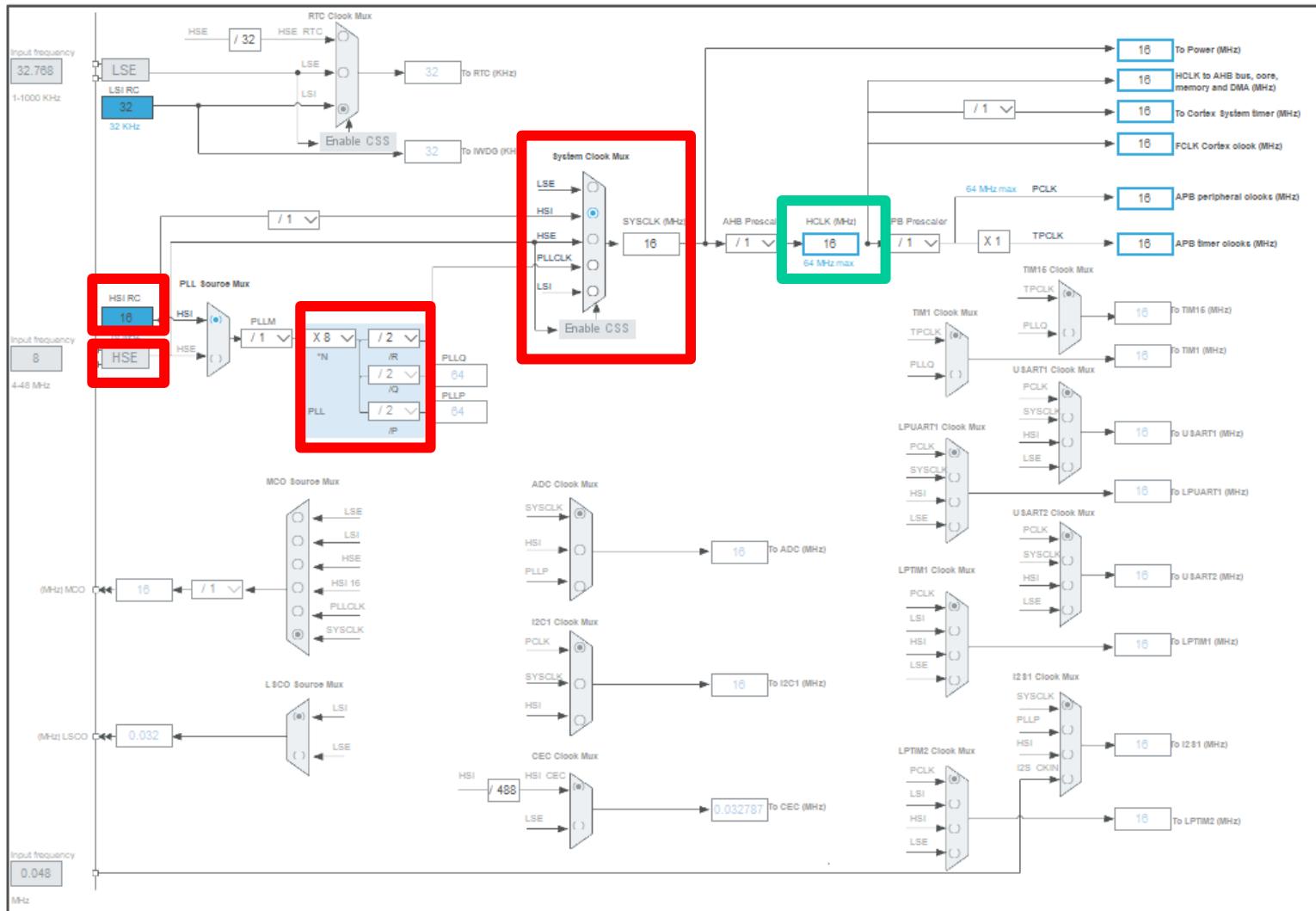
:V_{REF+}

وهو عبارة عن جهد
الدخل المرجعي
بالمبدلات الخاصة
التشابهية

مصادر الساعة في متحكمات STM32

لمتحكمات STM32 ثلاثة مصادر مختلفة للساعة هي:

- HSE
- HSI
- PLL

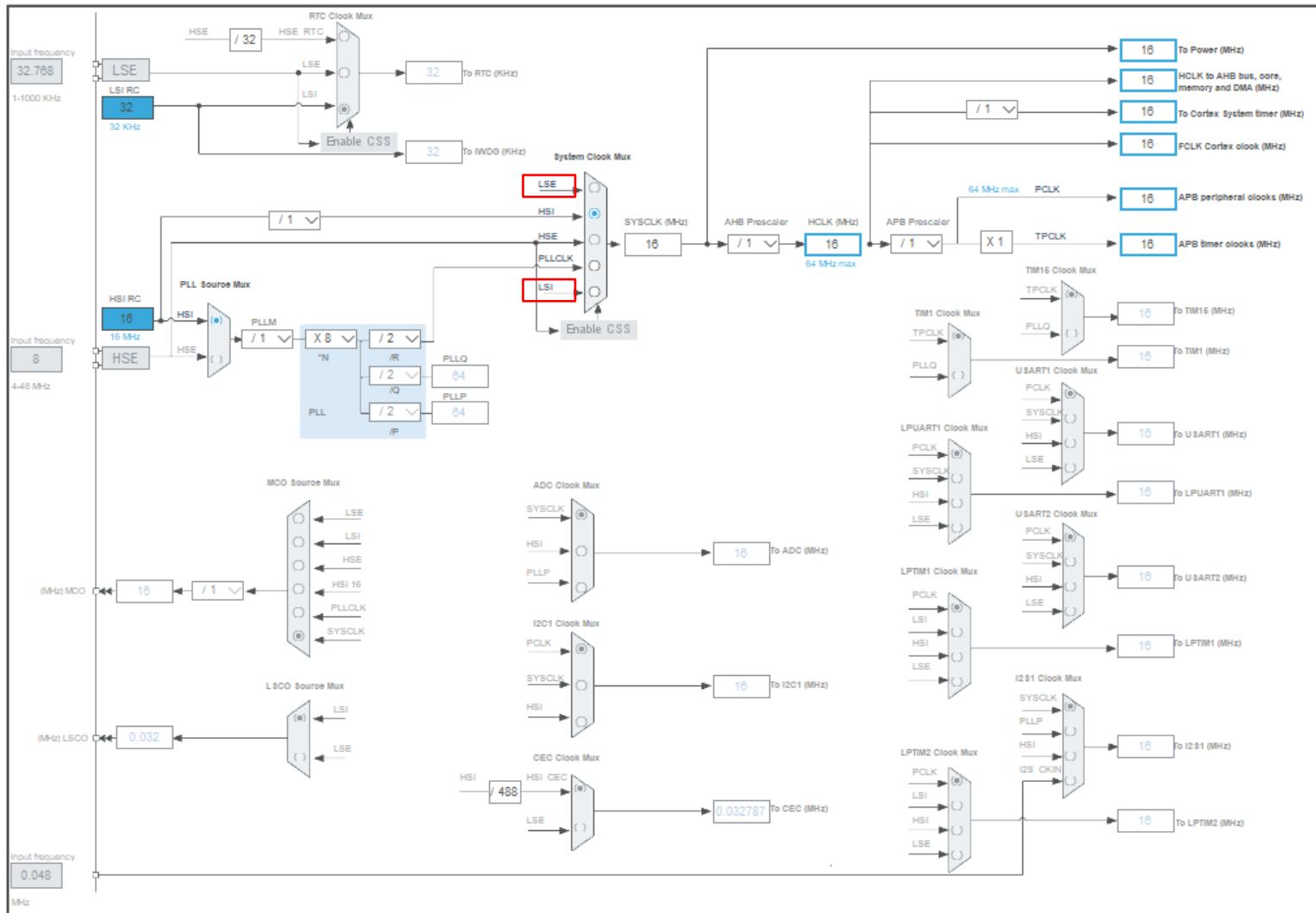


لمتحكمات STM32 ثلاثة مصادر مختلفة للساعة هي: □

- **HSE:** 4-48 MHz high-speed oscillator with external crystal or ceramic resonator , It can supply clock to system PLL
- **HSI:** 16 MHz high-speed internal RC oscillator (HSI16), trimmable by software. It can supply clock to system PLL
- **PLL:** System PLL with maximum output frequency of 64 MHz. It can be fed with HSE or HSI16 clocks

مصادر الساعة في متحكمات STM32

لمتحكمات STM32 مصدري ساعة مساعدين من أجل الـ RTC هما:



LSE



LSI



مصادر الساعة في متحكمات STM32

لتحكمات STM32 مصدري ساعة مساعدين من أجل الـ RTC هما:

- LSE: – 32.768 kHz low-speed oscillator with external crystal
- LSI: 32 kHz low-speed internal RC oscillator (LSI) with $\pm 5\%$ accuracy, also used to clock an independent watchdog

مصادر التصفير Reset في متحكمات STM32

لمتحكمات STM32 ثلاثة مصادر للتصفير هي :

System Reset

Power Reset

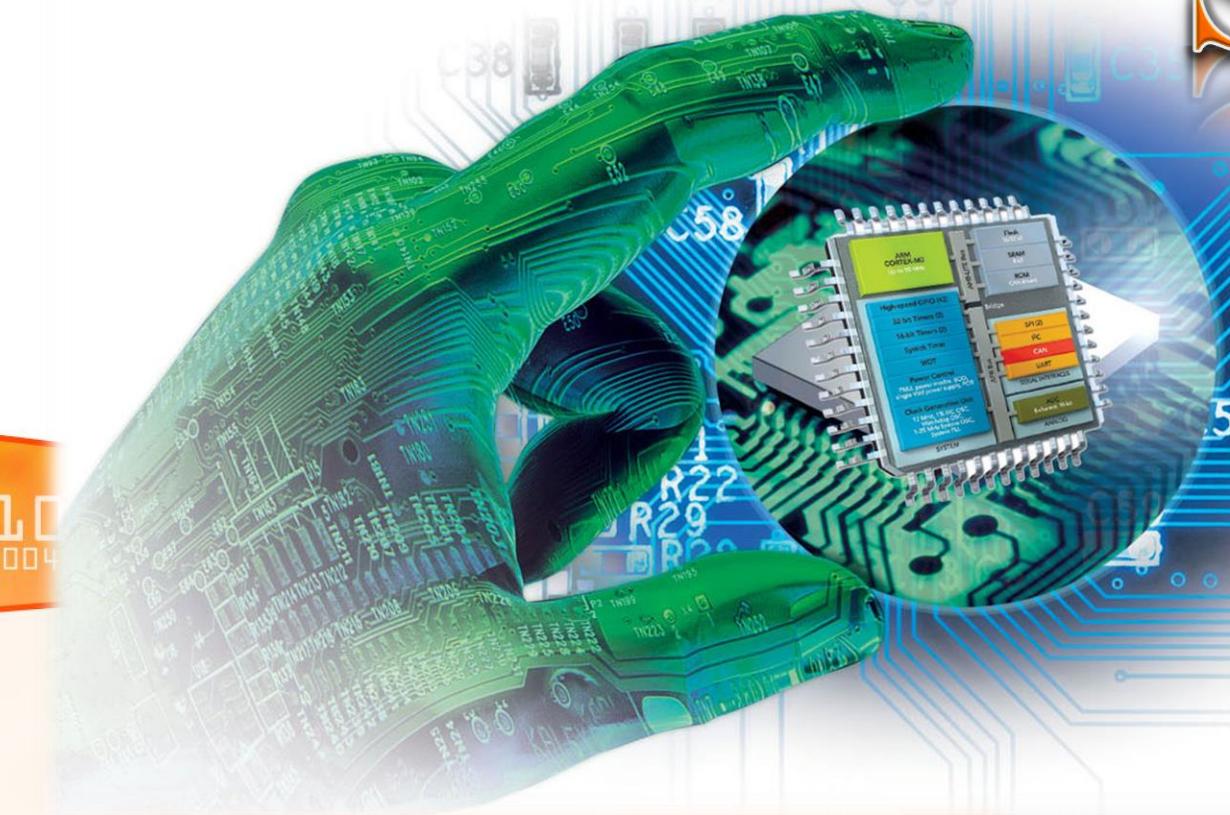
RTC Domain Reset

Thank you for listening

مُتَحَكِّمَات

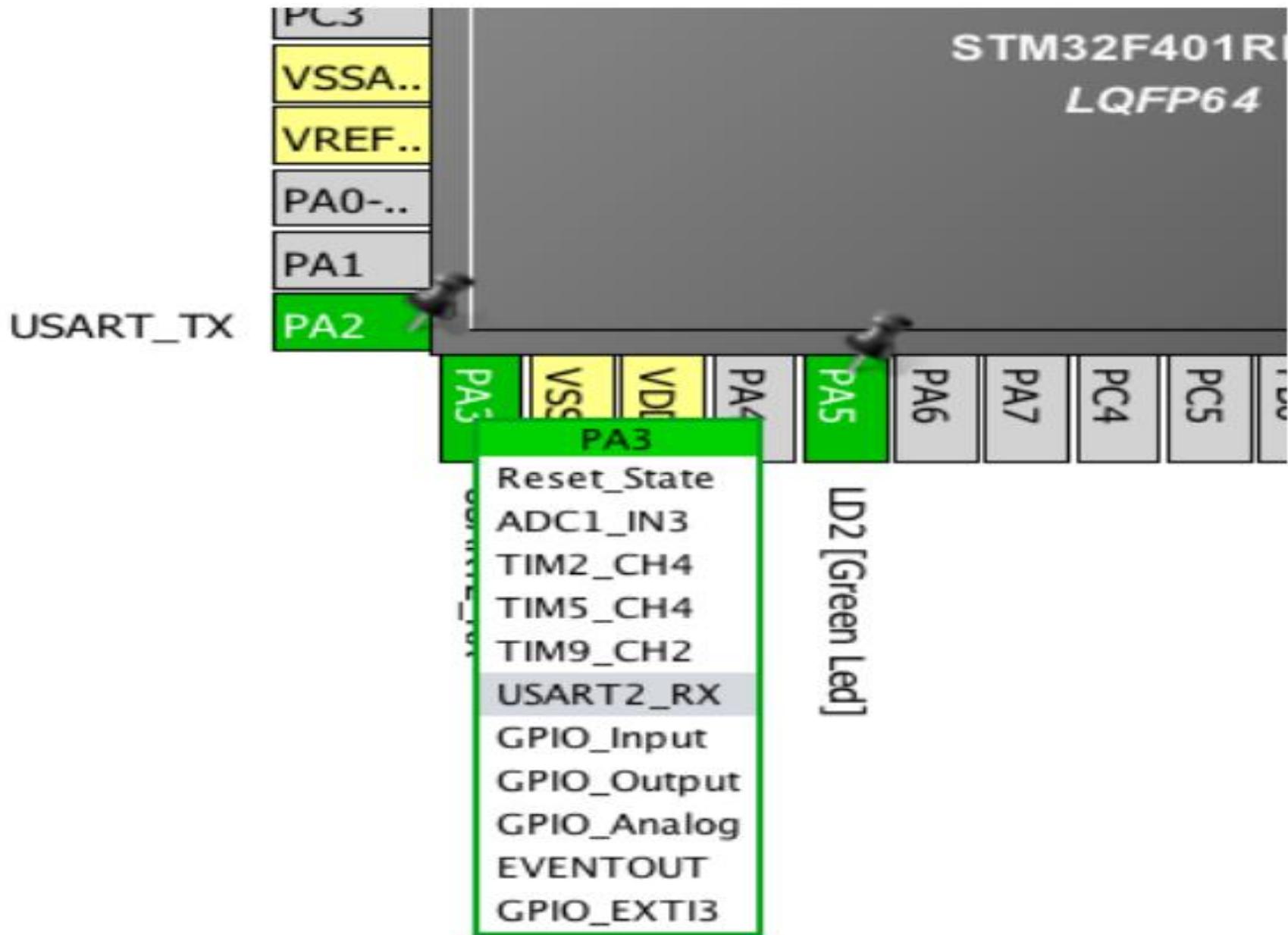
STM32

2



- أنماط عمل أقطاب المتحكم GPIO Mode
- برمجة أقطاب الخرج في متحكمات stm32
- التوابع المستخدمة من مكتبة HAL للتحكم بالمخارج الرقمية في متحكم STM32.
- بناء أول تطبيق لإضاءة ليد باستخدام متحكمات stm32 و مكتبة HAL
- برمجة أقطاب الدخل في متحكمات stm32
- التوابع المستخدمة من مكتبة HAL للتحكم بالمدخلات الرقمية في متحكم STM32.
- بناء تطبيق لإضاءة ليد من خلال مفتاح لحظي باستخدام متحكمات stm32 و مكتبة HAL

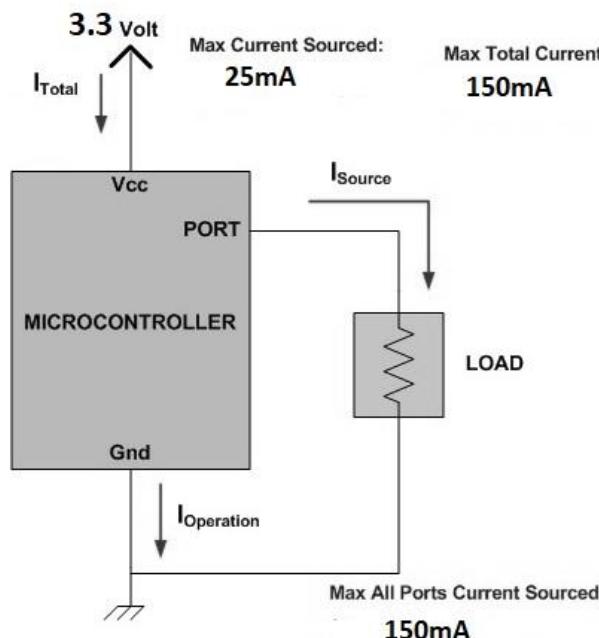
أنماط عمل أقطاب المتحكم .1 GPIO Mode



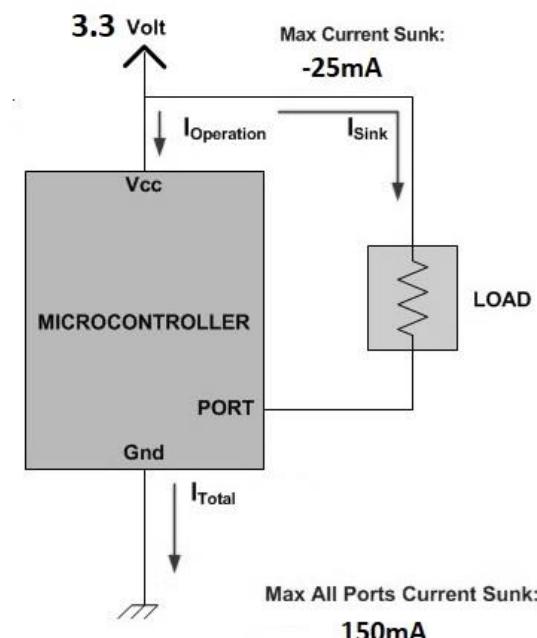
ربط الأحمال مع مخارج المتحكم

إن وصل الأحمال مع أقطاب المتحكم يكون بطريقتين:

- A. يعمل القطب كمنبع لتيار تشغيل الحمل (Source).
- B. يعمل القطب كمصرف لتيار تشغيل الحمل (Sink).

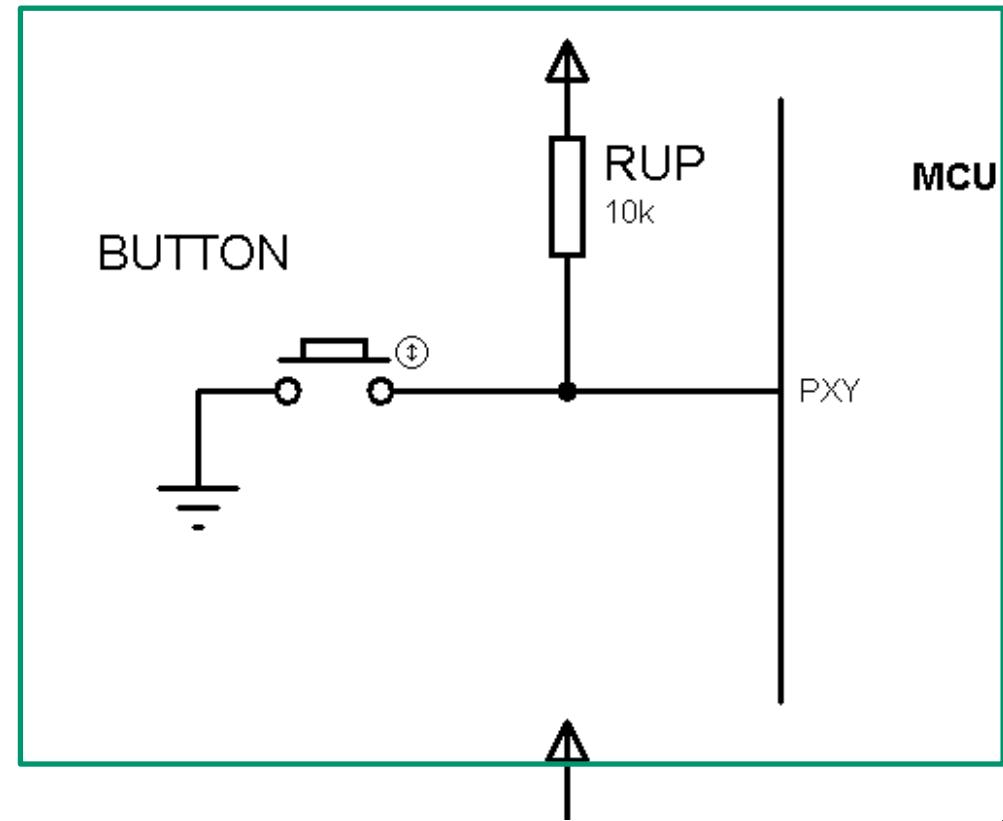
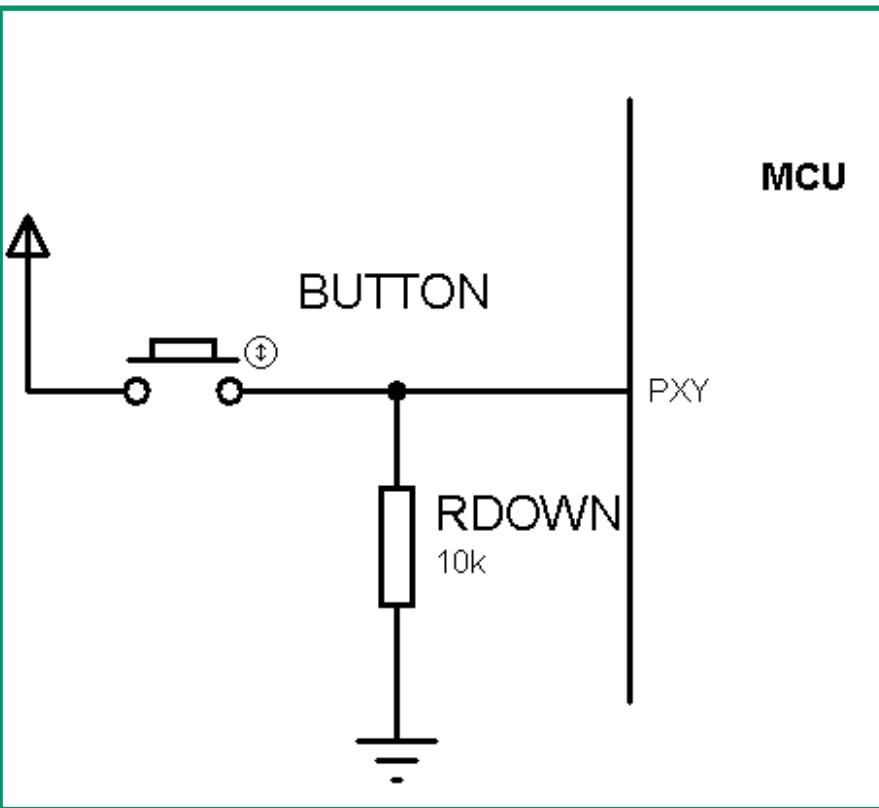


A. Microcontroller's Port is used as a Current Source



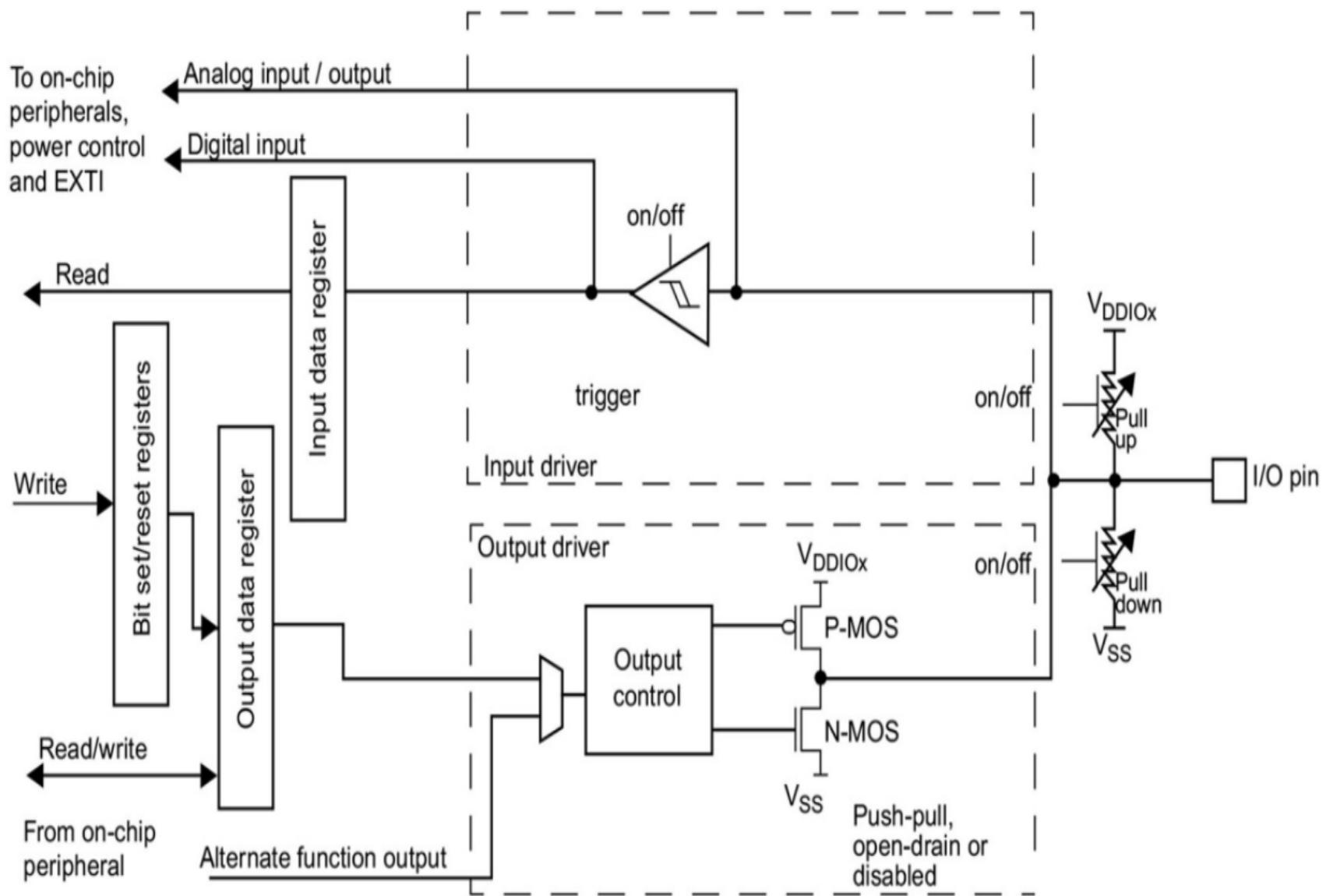
B. Microcontroller's Port is used to Sink Current

مفهوم مقاومة الرفع PULL_UP RES و مقاومة الخفض PULL_DOWN_RES

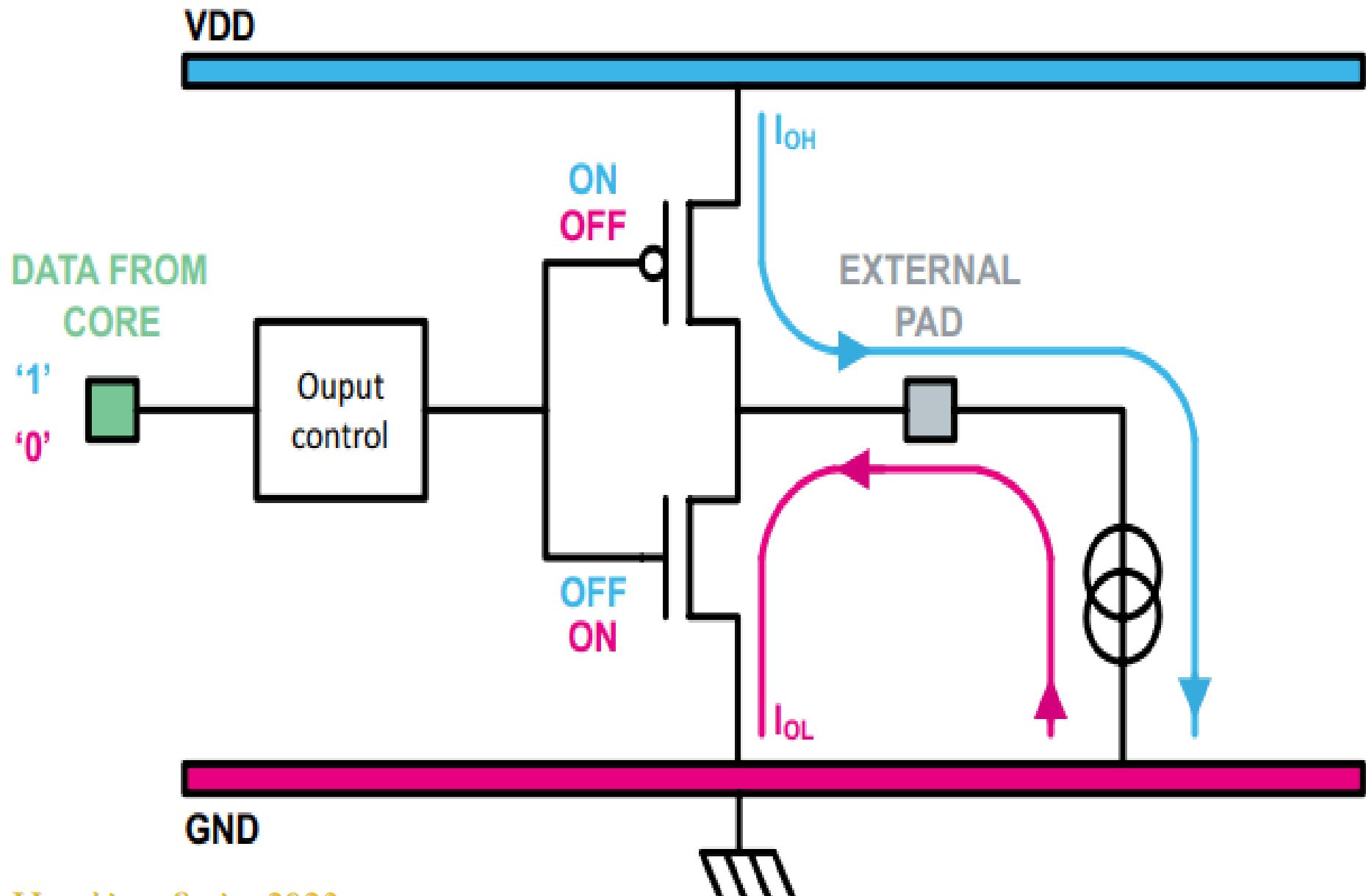


أنماط عمل أقطاب المتحكم .1

GPIO Mode



برمجة أقطاب الخرج في متحكمات stm32 .2



2. برمجة أقطاب الخرج في متحكمات stm32

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم :GPIO

□ **نمط PUSH-PULL:** فبفرض تم وصل مصد ليد مع قطب الإخراج للمتحكم المصغر، فعند تطبيق واحد منطقي على القطب يصبح الترانزستور PMOS بحالة تشغيل on وبالتالي يتم تطبيق جهد الـ VCC على مصد الليد ويضيء الليد، أما عند تطبيق صفر منطقي على القطب يصبح الترانزستور NMOS بحالة تشغيل on وبالتالي يتم تطبيق جهد الأرضي GND على مصد الليد ويطفأ الليد كما هو موضح بالشكل التالي:

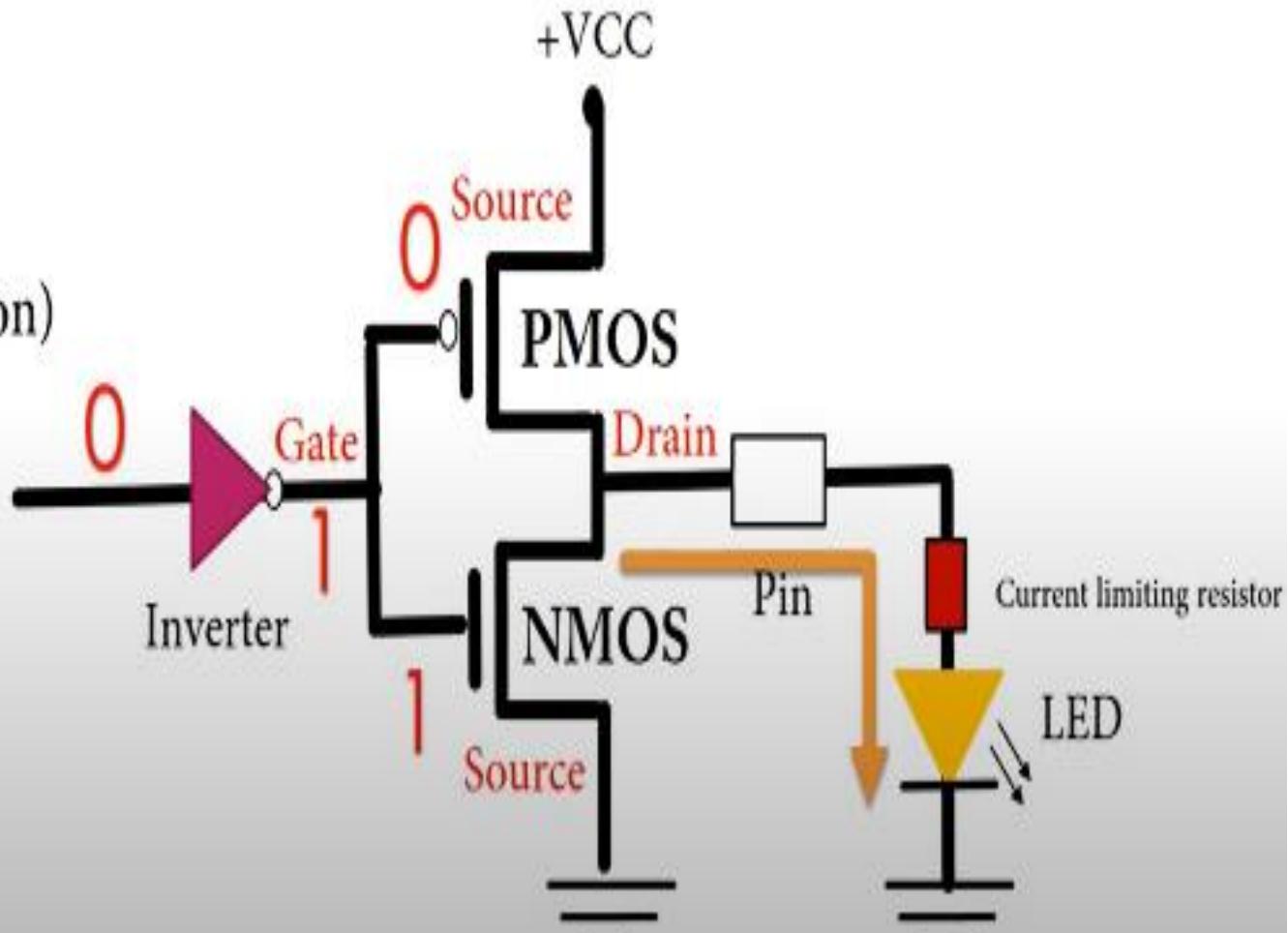
2. برمجة أقطاب الخرج في متحكمات stm32

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم :GPIO

:**PUSH-PULL**

Push-Pull

(Default Configuration)

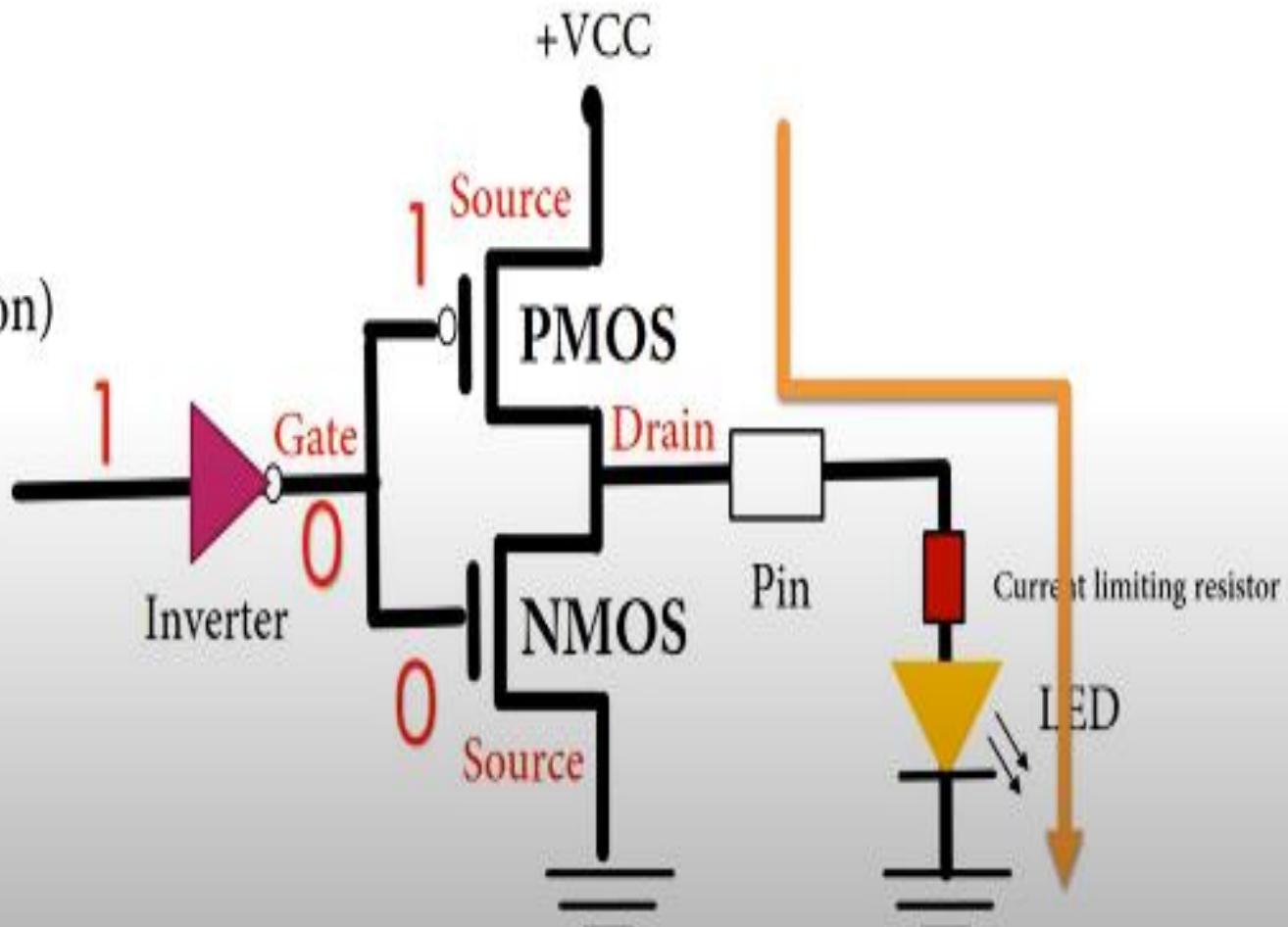


نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم :GPIO

:PUSH-PULL

Push-Pull

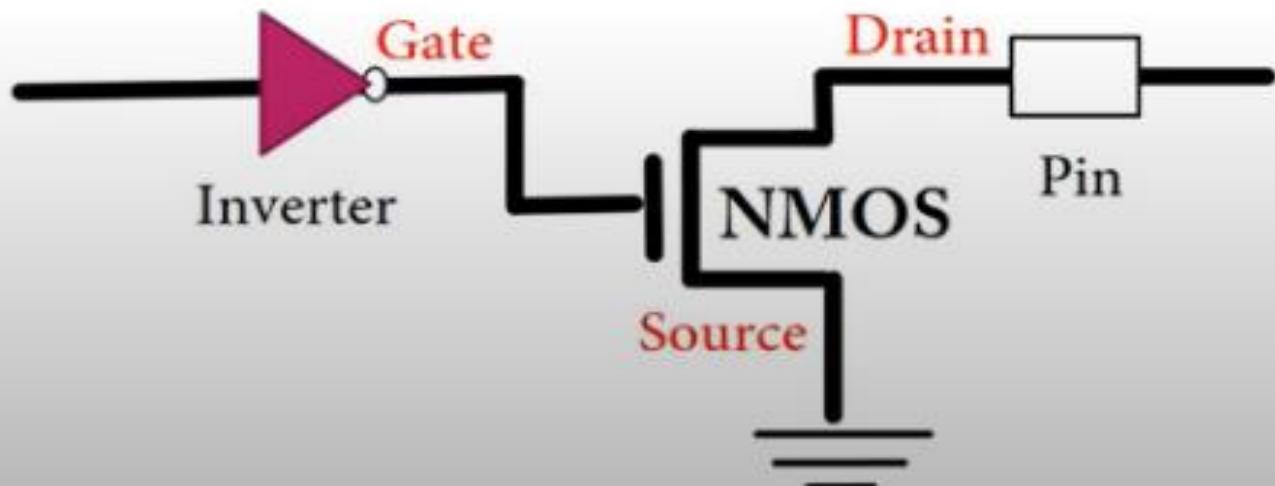
(Default Configuration)



نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم :GPIO:

- **نمط Open-Drain** في هذا النمط يمكن للمتحكم أن يعمل كصرف للتيار sink فقط، فيتم قيادة قطب الخرج من خلال ترانزستور واحد من نوع NMOS كما هو موضح في الأشكال التالية.

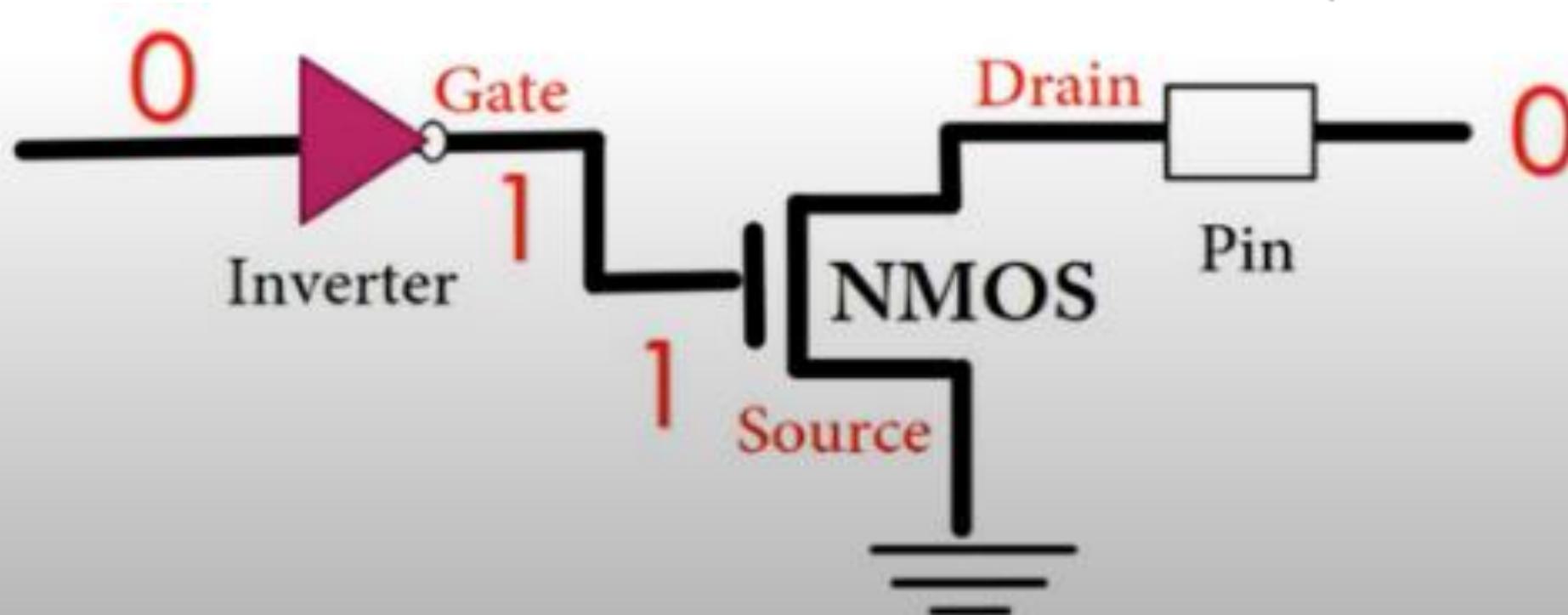
Open Drain



2. برمجة أقطاب الخرج في متحكمات stm32

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم :GPIO

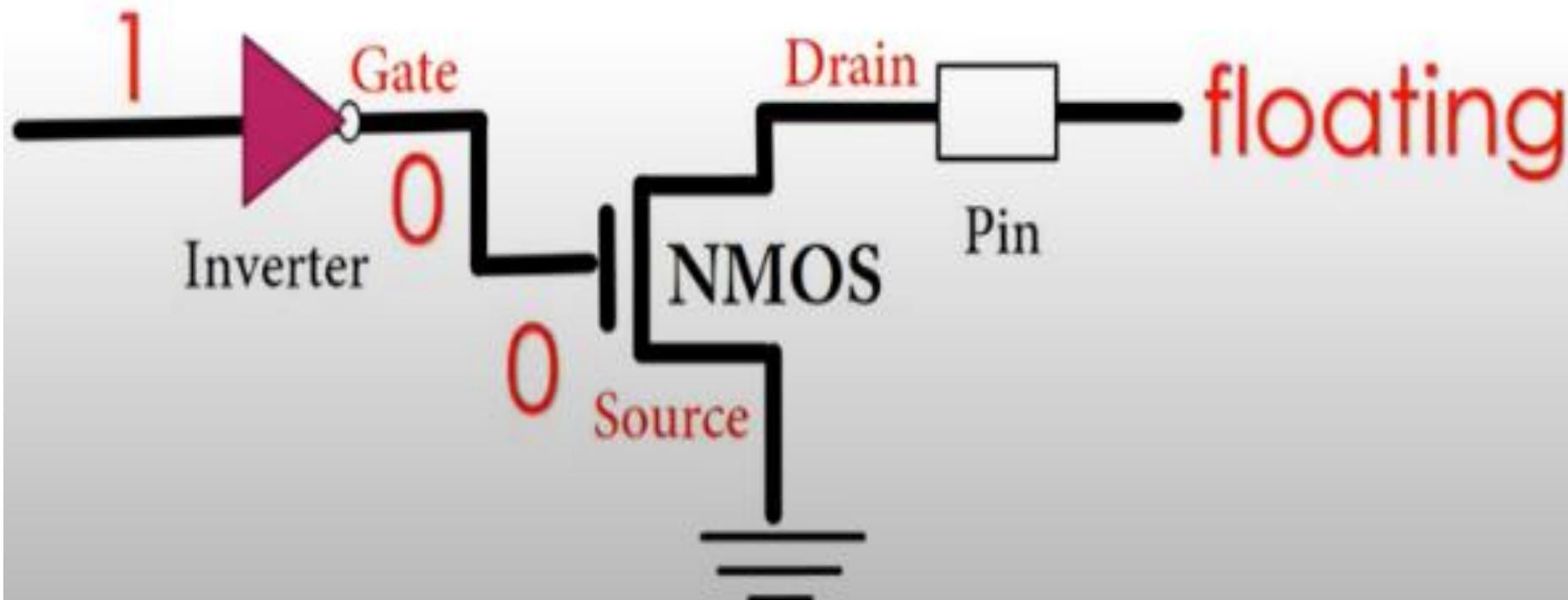
□ **نمط Open-Drain** في هذا النمط يمكن للمتحكم أن يعمل كصرف للتيار sink فقط، فيتم قيادة قطب الخرج من خلال ترانزستور واحد من نوع NMOS كما هو موضح في الأشكال التالية:



2. برمجة أقطاب الخرج في متحكمات stm32

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم :GPIO:

□ **نمط Open-Drain** في هذا النمط يمكن للمتحكم أن يعمل كصرف للتيار sink فقط، فيتم قيادة قطب الخرج من خلال ترانزستور واحد من نوع NMOS كما هو موضح في الأشكال التالية:



2. برمجة أقطاب الخرج في متحكمات stm32

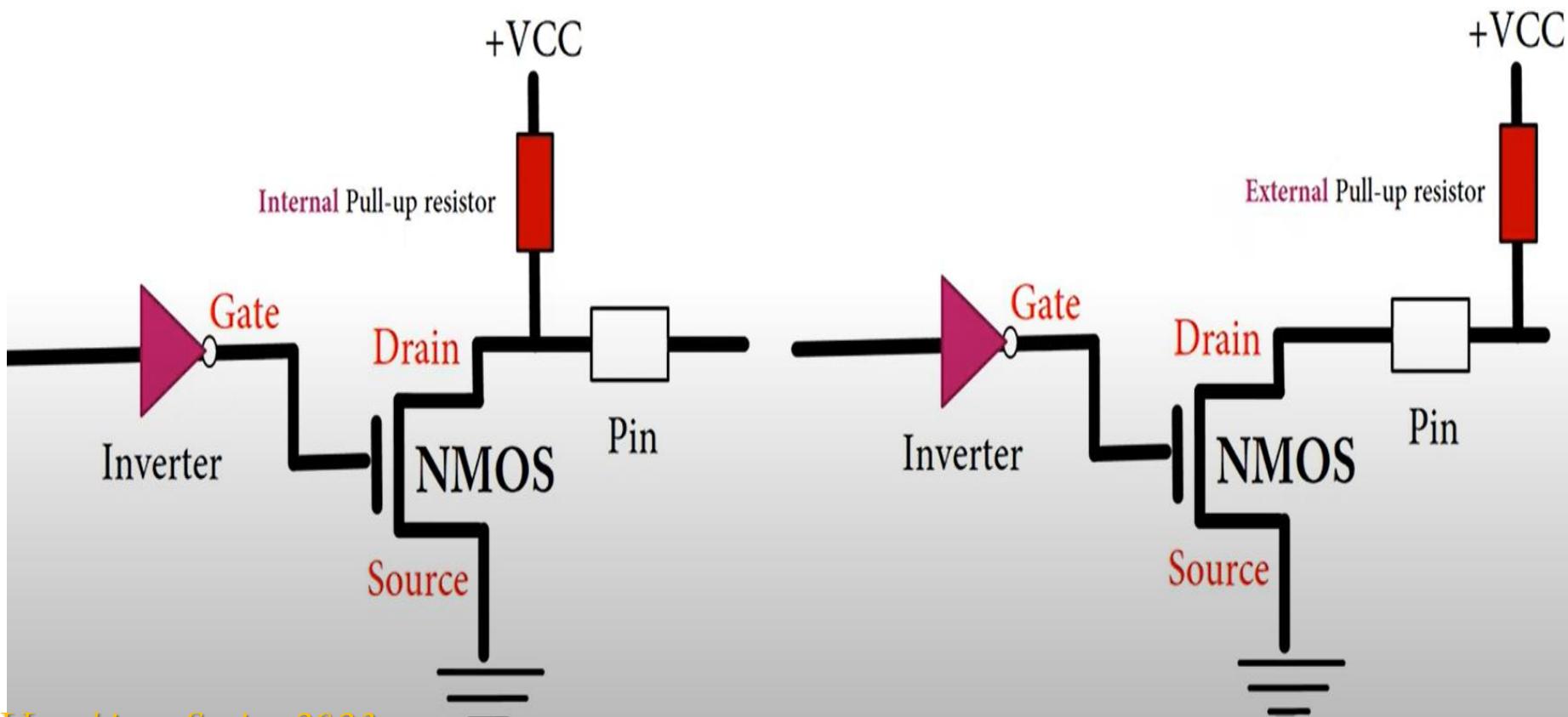
نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم :GPIO

□ **نمط Open-Drain** نلاحظ من الأشكال السابقة أنه عند تطبيق صفر منطقي على قطب المتحكم يصبح الترانزستور بحالة توصيل on وبالتالي يتم توصيل الحمل مع الأرضي GND، أما في حالة تطبيق واحد منطقي على قطب المتحكم يصبح الترانزستور بحالة فصل off وبالتالي يصبح الجهد المطبق على الحمل عائم غير محدد floating لذا ولحل هذه المشكلة لابد من توصيل مقاومة رفع مع قطب المتحكم ، يمكن توصيل هذه المقاومة خارجياً أو يمكن تفعيل مقاومة الرفع الداخلية الموجودة ضمن المتحكم كما في الشكل التالي:

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم :GPIO

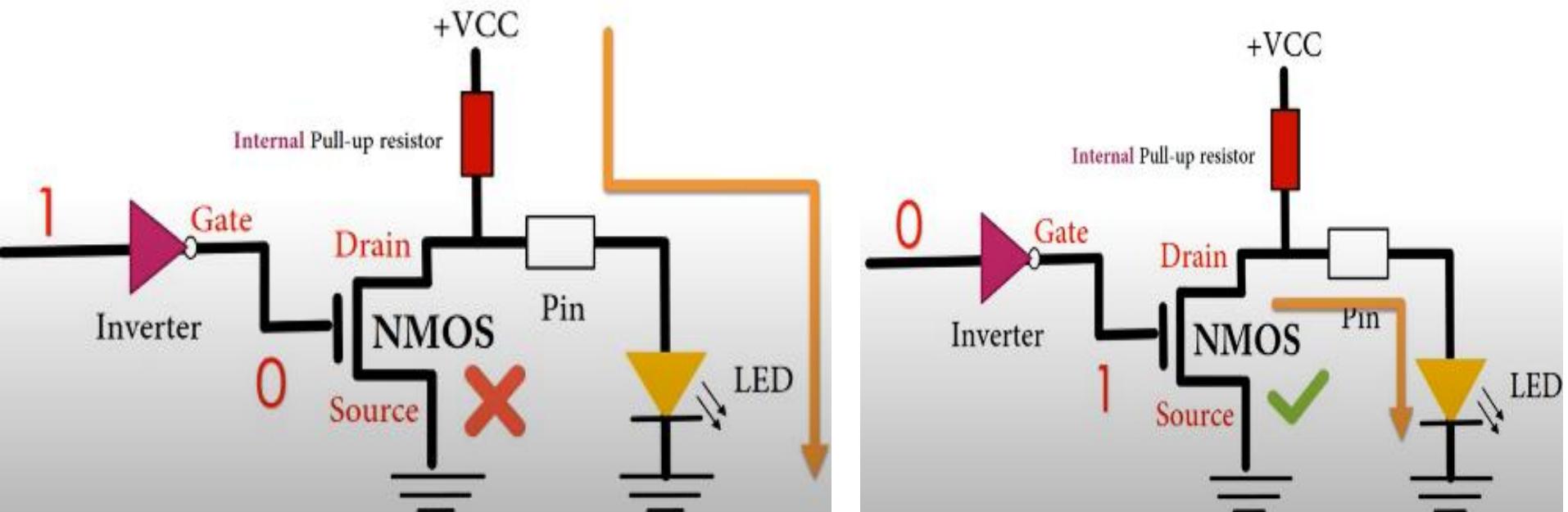
نمط Open-Drain □

Open Drain with Pull-up Resistor -



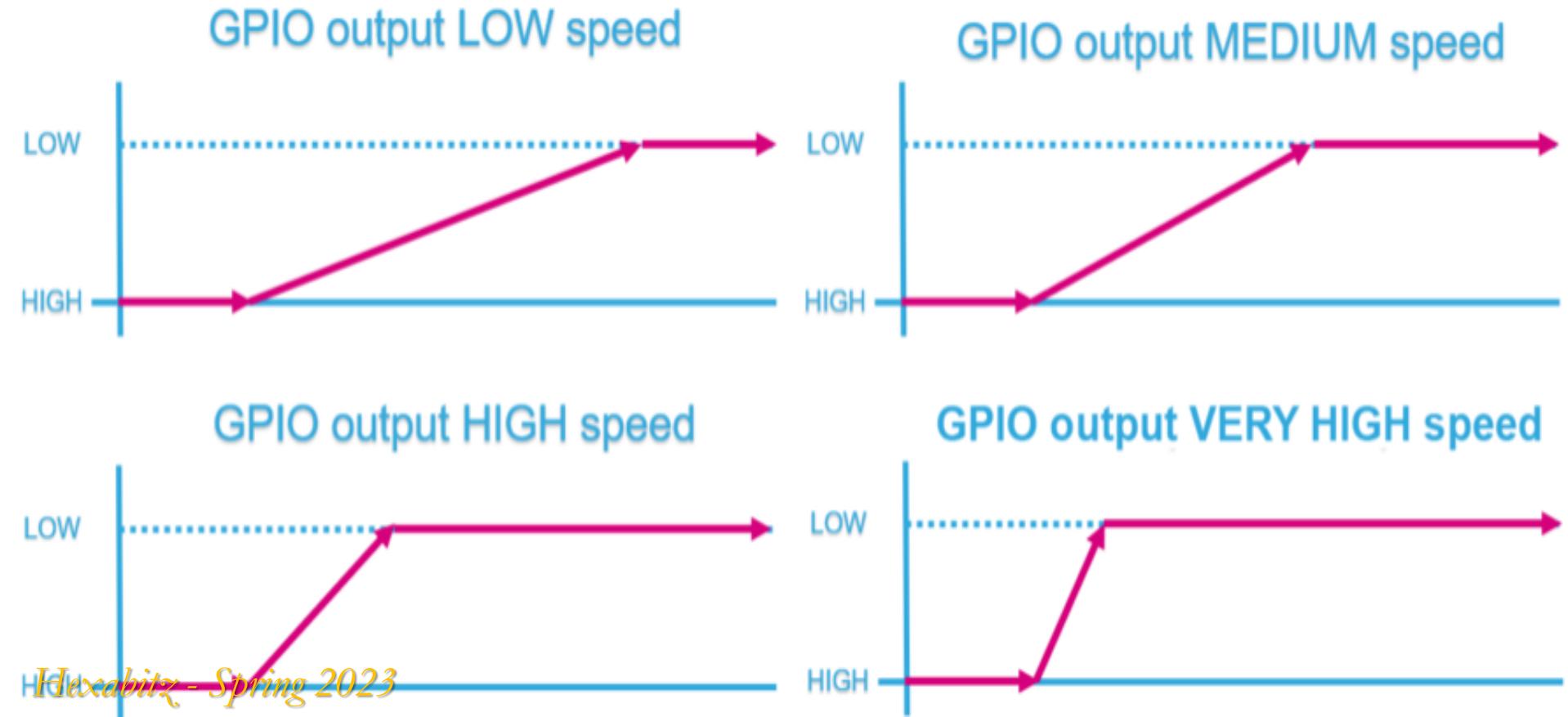
نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم :GPIO

نمط Open-Drain □



برمجة أقطاب الخرج في متحكمات stm32

: يعني سرعة انتقال الإشارة من الحالة المنخفضة low إلى الحالة المرتفعة HIGH وبالعكس وهو ما يسمى بزمن الصعود rise time وزمن الهبوط fall time كما هو موضح بالشكل التالي:



ربط الأحمال مع مخارج المتحكم

أهم الاعتبارات التي يجب أن تؤخذ بعين الاعتبار عن ربط أقطاب المتحكم إلى الأحمال هو:

- التيار الأعظمي المستهلك من قطب المتحكم (Vcc to Gnd) (Vcc to Gnd) الذي يمكن سحبه أو تصريفه عن طريق المتحكم بشكل كلي هو 150mA وفق المواصفات الكهربائية لعائلة متحكمات STM32.
- قيمة التيار التي يمكن سحبها أو تصريفها لقطب خرج من أقطاب المتحكم تتراوح عادة من 25mA حسب المواصفات الكهربائية للمتحكم المصغر STM32.

ربط الأحمال مع مخارج المتحكم

Symbol	Ratings	Max.	Unit
I_{VDD}	Total current into V_{DD}/V_{DDA} power lines (source) ⁽¹⁾	150	
I_{VSS}	Total current out of V_{SS} ground lines (sink) ⁽¹⁾	150	
I_O	Output current sunk by any I/O and control pin	25	mA
	Output current source by any I/Os and control pin	-25	

التوابع المستخدمة من مكتبة HAL للتحكم بالخارج الرقمية في متّحكم STM32

لإعطاء واحد منطقى set أو صفر منطقى reset لقطب محدد من أي منفذ من منافذ المتّحكم نقوم باستخدام التابع التالي من مكتبة :HAL

HAL_GPIO_WritePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState);



اسم المنفذ المراد التحكم بأحد
أقطابه على سبيل المثال
GPIOA

لإعطاء واحد منطقى نكتب رقم القطب المراد التحكم به

على سبيل المثال
GPIO PIN 5

ولإعطاء صفر منطقى
نكتب
GPIO_PIN_RESET

التوابع المستخدمة من مكتبة HAL للتحكم بالمخارج الرقمية في متّحكم STM32

مثال 1: لكتابة واحد منطقى على القطب رقم 10 من المنفذ D يستخدم
التابع التالي:

**HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10,
GPIO_PIN_SET);**

مثال 2: لكتابة صفر منطقى على القطب رقم 5 من المنفذ A يستخدم
التابع التالي:

**HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,
GPIO_PIN_RESET);**

التوابع المستخدمة من مكتبة HAL للتحكم بالمخارج الرقمية في متّحكم STM32

لعكس الحالة المنطقية لأحد الأقطاب نستخدم التابع التالي:

```
HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx,  
                      uint16_t GPIO_Pin);
```

مثال: لعكس الحالة المنطقية للقطب رقم 5 من المنفذ A نستخدم التابع التالي:

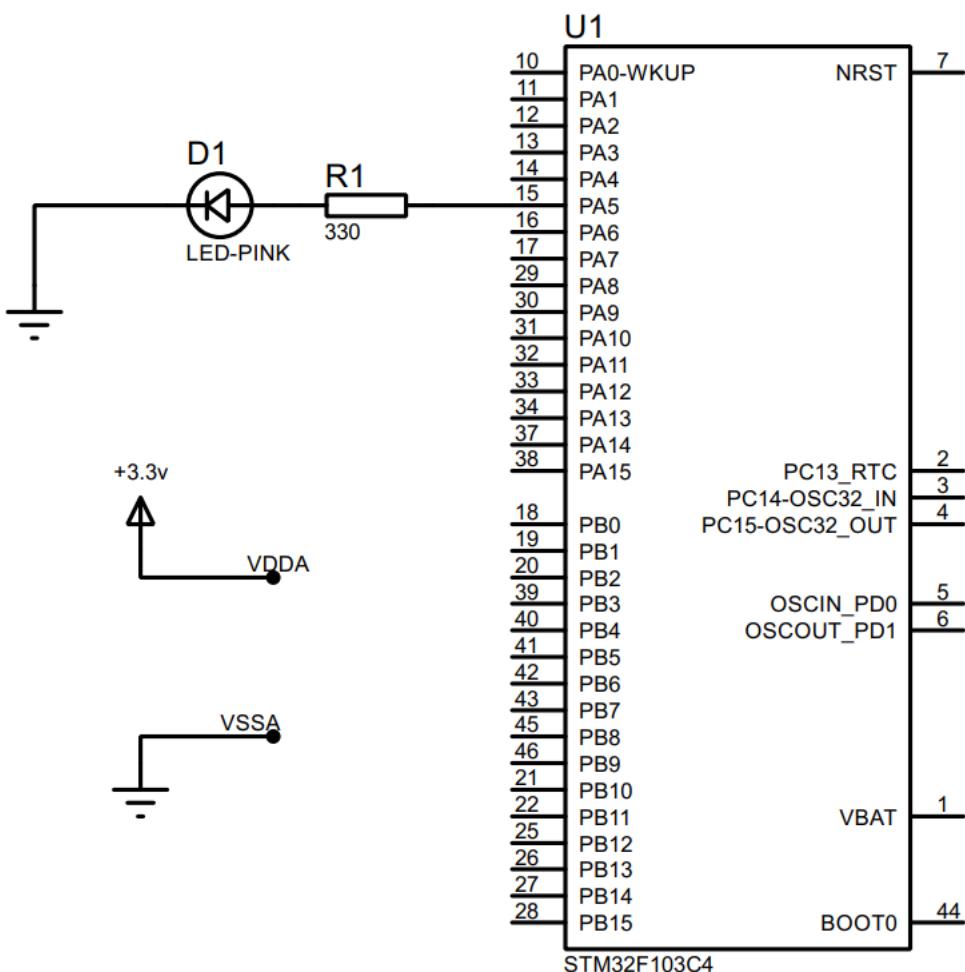
```
HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_5);
```

لإضافة تأخير زمني بالمياللي ثانية نستخدم التابع التالي:

```
HAL_DelayMilliseconds)
```

HAL و مکتبہ STM32

سنقوم بتصميم تطبيق يقوم بعمل toggle لليد المتصل بالقطب PA5



STM32 HAL و مكتبة STM32 و إعدادات القطب PA5 كقطب خرج

نقوم بضبط إعدادات القطب PA5 كقطب خرج



MX s1_ex1_hal.ioc main.c

Pinout & Configuration Clock Configuration Project Manager Tools

GPIO Mode and Configuration

Configuration

Group By Peripherals

Categories A-Z

System Core

DMA

GPIO

IWDG

NVIC

RCC

SYS

WWDG

Analog

Timers

Connectivity

Computing

Middleware

Search Signals

Search (Ctrl+F)

Show only Modified Pins

Pin Name	Signal on	GPIO output	GPIO mode	GPIO Pull	Maximum speed	User Label	Modified
PA5	n/a	Low	Output Push Pull	No pull-up and no pull-down	Low		<input type="checkbox"/>

PA5 Configuration :

GPIO output level : Low

GPIO mode : Output Push Pull

GPIO Pull-up/Pull-down : No pull-up and no pull-down

Maximum output speed : Low

User Label :

Pinout view System view

Pinout view

System view

STM32F103C4Tx LQFP48

Pinout diagram showing the STM32F103C4Tx LQFP48 package with pin numbers and labels. Pin PA5 is highlighted in green.

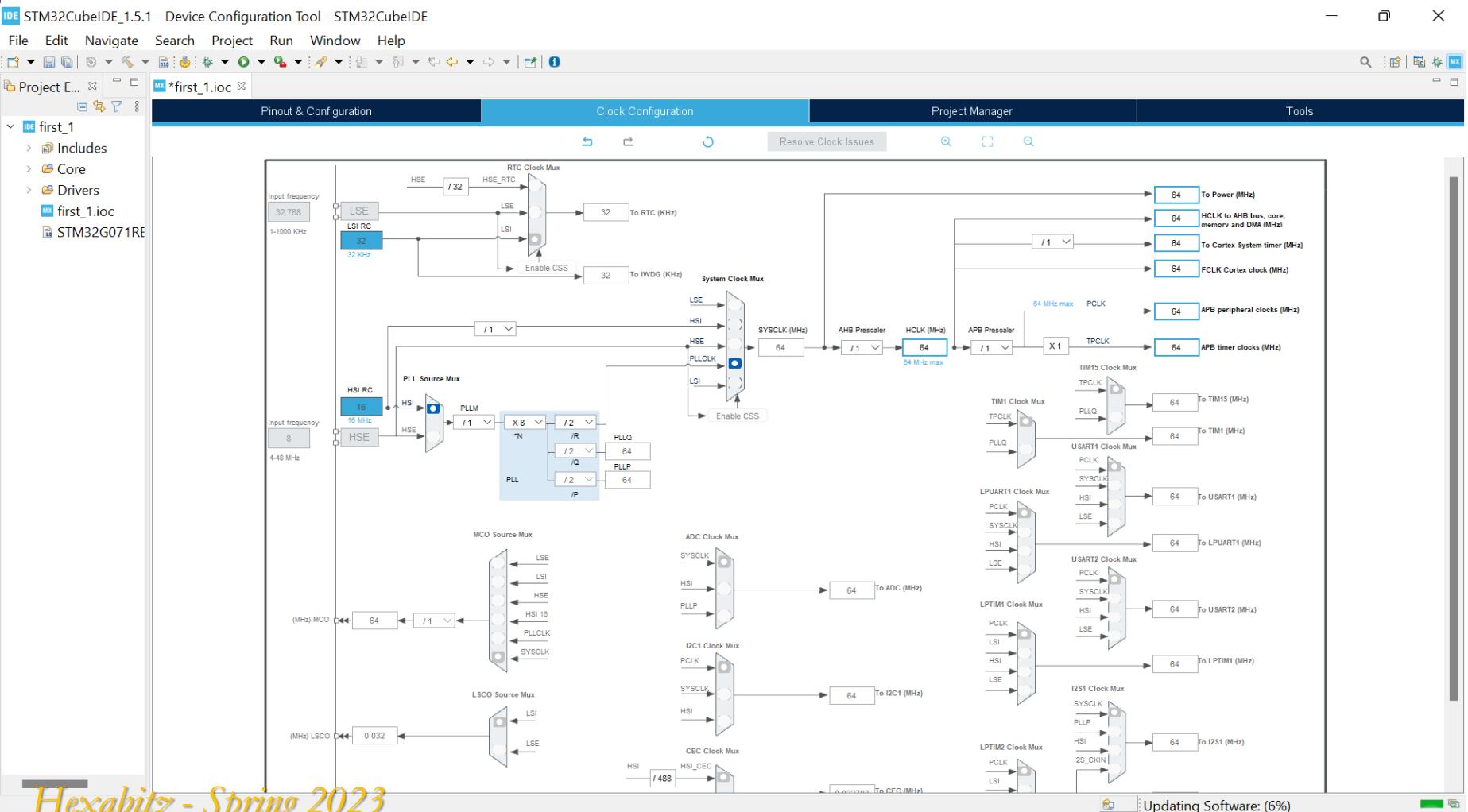
Pinout Labels:

- VBAT
- PC13..
- PC14..
- PC15..
- PD0..
- PD1..
- NRST**
- VSSA
- VDDA
- PA0..
- PA1
- PA2
- PA3
- PA4
- PA5**
- PA6
- PA7
- PB0
- PB1
- PB2
- PB10
- PB11
- VSS
- VDD
- PA15
- PA14
- PA13
- PA12
- PA11
- PA10
- PA9
- PA8
- PB15
- PB14
- PB13
- PB12

Bottom icons: magnifying glass, double arrows, clipboard, file, settings, search, dropdown menu.

التطبيق الأول: إضاعة ليد وإطفاؤه كل sec 0.5 باستخدام متحكمات HAL و مكتبة STM32

نقوم بضبط تردد الساعة للمتحكم



- نقوم بالضغط على **Project...Generate** أو من **Ctrl+s**، ليتم حفظ المشروع وتوليد الكود ثم نقوم بإضافة الجزء التالي:

```
#include "main.h"  
void SystemClock_Config(void);  
static void MX_GPIO_Init(void);  
int main(void) {  
    HAL_Init();  
    SystemClock_Config();  
    MX_GPIO_Init();
```

```
while (1)
{
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
    HAL_Delay(500);
}
```

}

للتحكم بالمخارج الرقمية للمتحكم STM32 دون استخدام مكتبة HAL:

□ هناك مجموعة من المسجلات تستخدم للتحكم بالمخارج الرقمية لمتحكم STM32 سنكتفي فقط بذكر المسجل المسؤول عن عمل المفتاح المنفذ أو لأحد الأقطاب الموجودة فيه set/reset

7.4.6 GPIO port output data register (GPIOx_ODR) (x = A..H)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..H).

للتحكم بالمخارج الرقمية للمتحكم STM32 دون استخدام مكتبة HAL:

- لكتابة واحد منطقي على القطب رقم 5 من المنفذ A نكتب:
- GPIOA->ODR |= 1<<5; // Set the Pin PA5**
- و لكتابة صفر منطقي عليه نكتب:
- GPIOA->ODR &= ~(1<<5); // Reset the Pin PA5**
- كما يمكن جعل المنفذ بالكامل بحالة set من خلال كتابة :
- GPIOA->ODR = 0xFFFF; // Set the PORTA HIGH**
- كما يمكن جعل المنفذ بالكامل بحالة reset من خلال كتابة :
- GPIOA->ODR = 0x00; // Reset the PORTA**

التطبيق الثاني: إضاءة ليد وإطفاؤه كل 0.5 sec دون استخدام مكتبة HAL

```
#include "main.h"
```

```
void SystemClock_Config(void);  
static void MX_GPIO_Init(void);
```

```
int main(void)  
{  
    HAL_Init();  
    SystemClock_Config();  
    MX_GPIO_Init();
```

HAL

```
while (1)
```

```
{
```

```
GPIOA->ODR = 0x0020; // Set the Pin PA5
```

```
HAL_Delay(500);
```

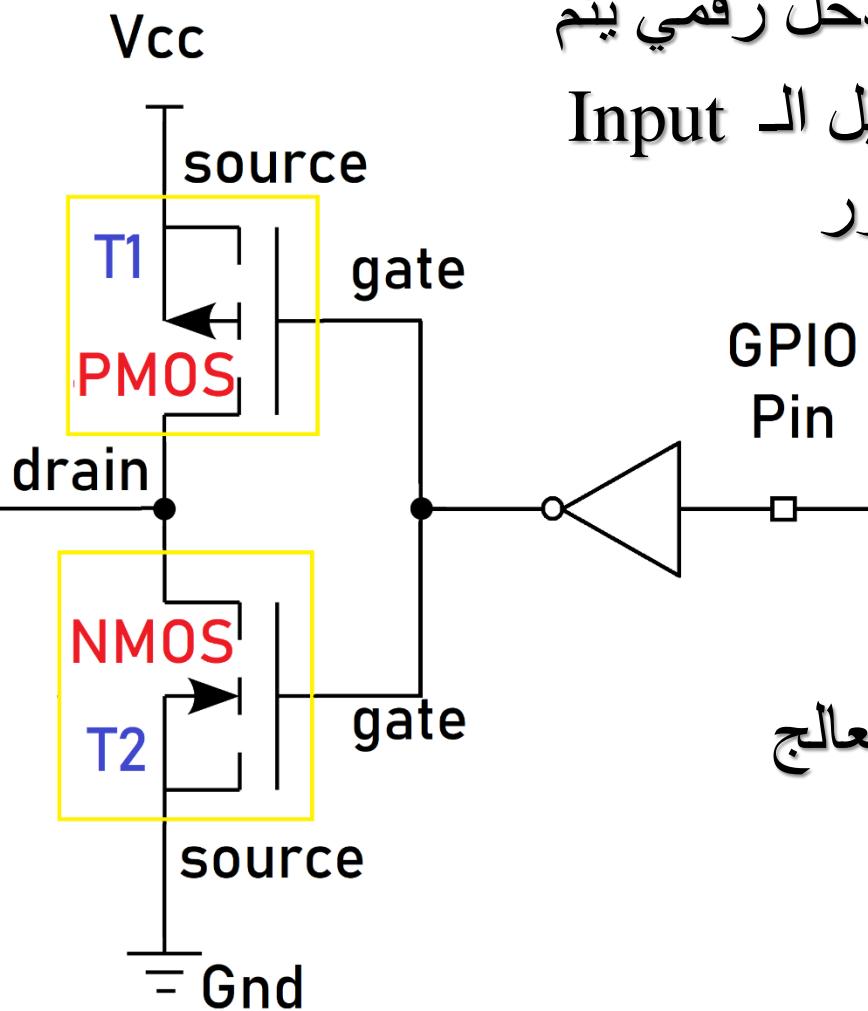
```
GPIOA->ODR = 0x0000; // Reset the PORTA
```

```
HAL_Delay(500);
```

```
}
```

```
}
```

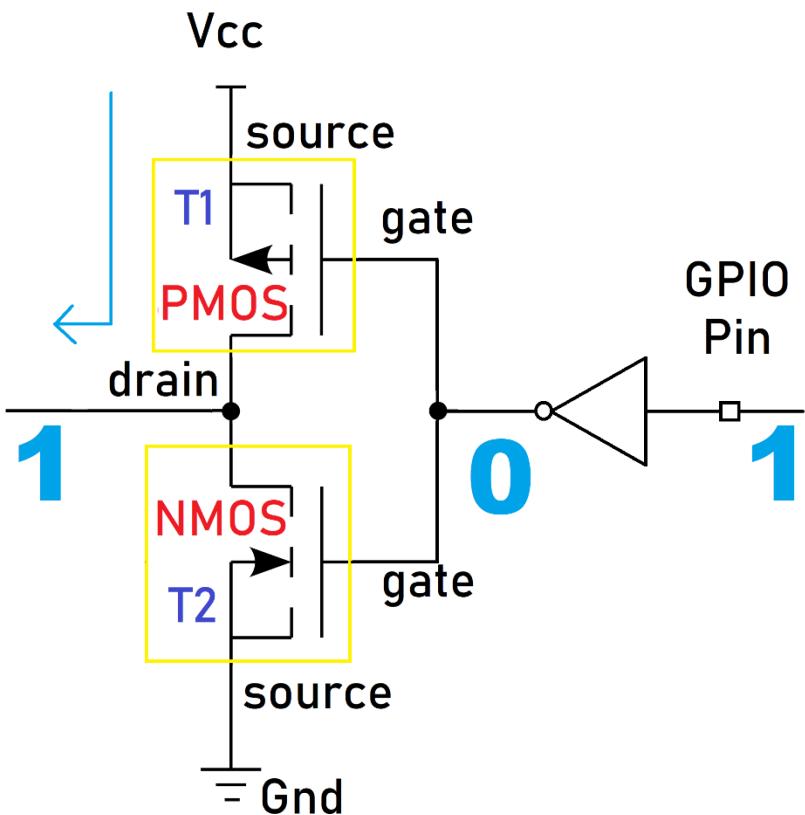
برمجة أقطاب الدخل في متحكمات stm32



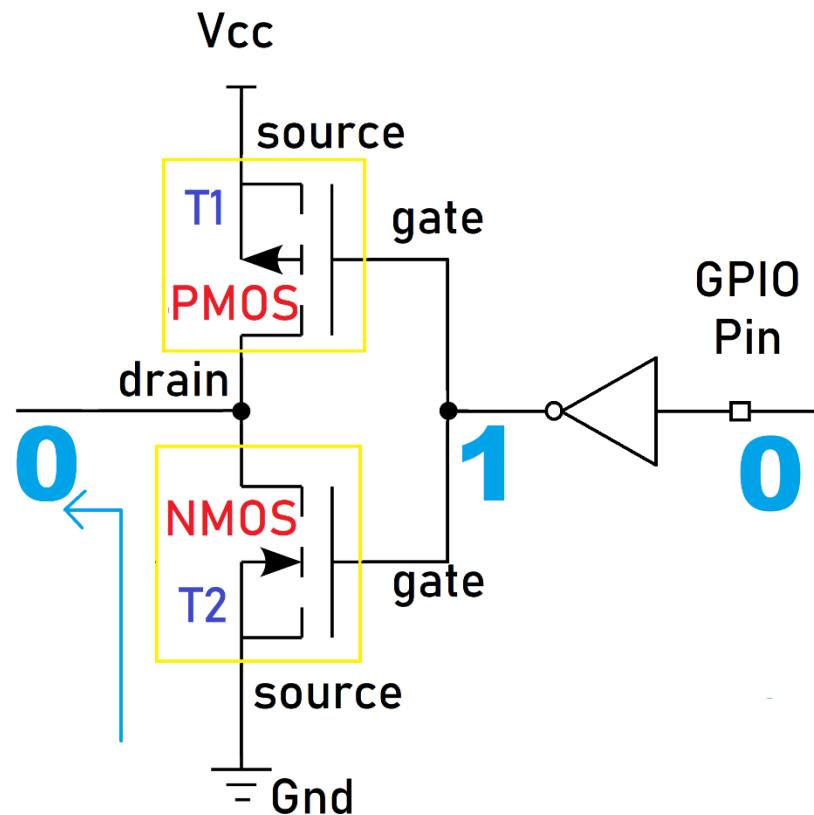
عند استخدام قطب المتحكم كقطب دخل رقمي يتم إلغاء تفعيل Input buffer وتفعيل الـ Output buffer والذي يتكون من ترانزستور من نوع NMOS وترانزستور من نوع PMOS حيث تتصل بوابة الترانزستور مع قطب الـ GPIO و قطب المصرف للترانزستور متصل بالمعالج

Input Buffer

برمجة أقطاب الدخل في متحكمات stm32



Input Buffer reads 1



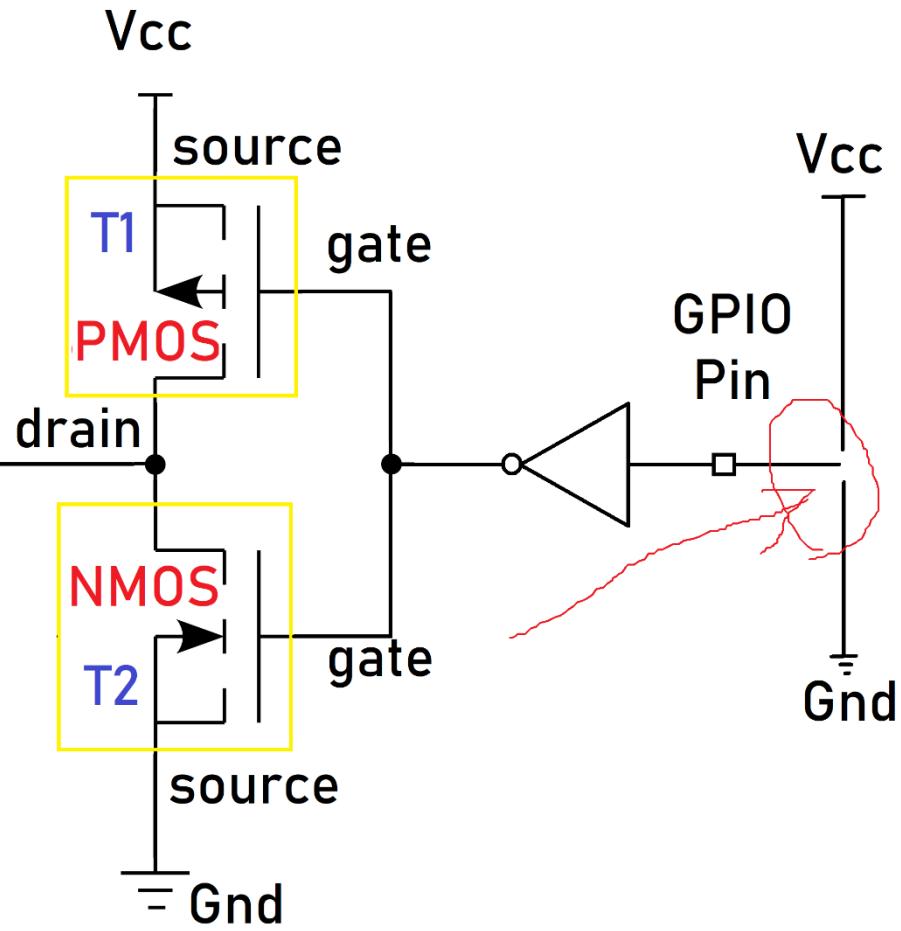
Input Buffer reads 0

الأنماط المختلفة لقطب الدخل الرقمي

نميز ثلاث أنماط مختلفة لقطب الدخل الرقمي للمتحكم :

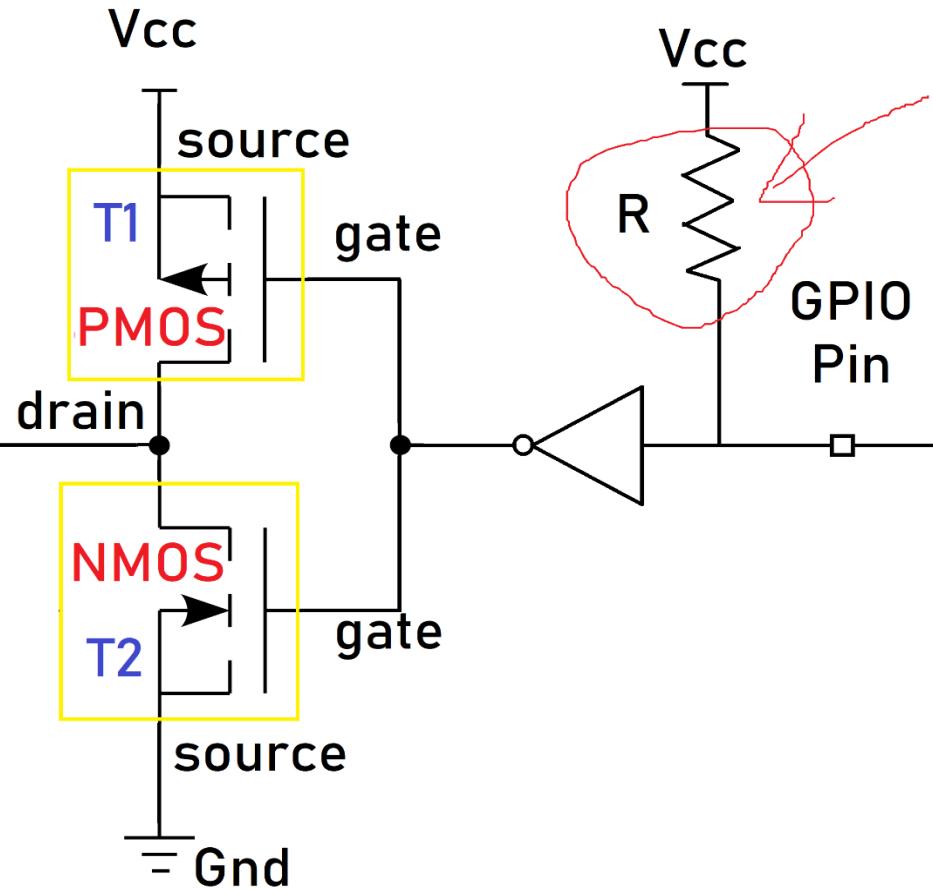
- High impedance of Floating**
- نط ال PULL-Up**
- نط ال Pull-Down**

نمط High impedance of Floating



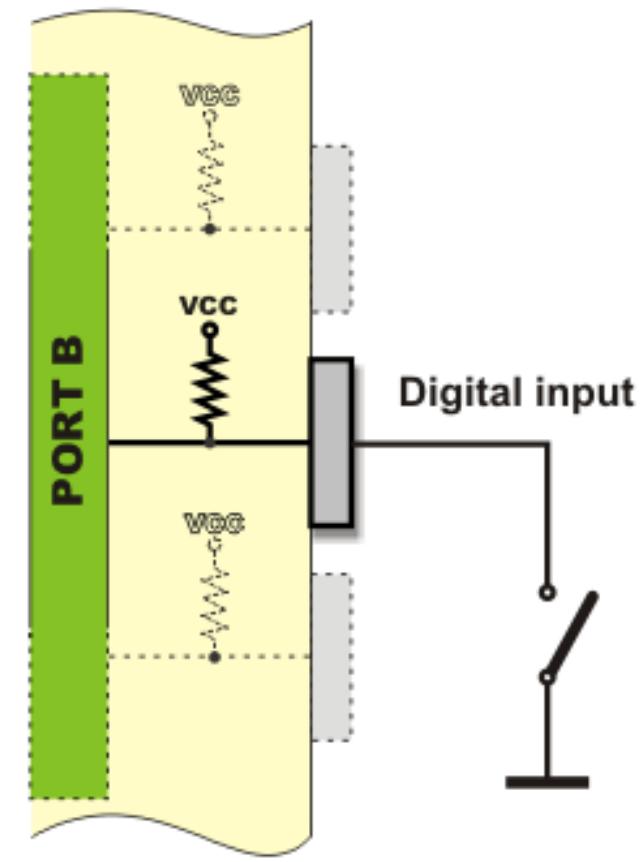
Input Buffer

نط PULL-UP

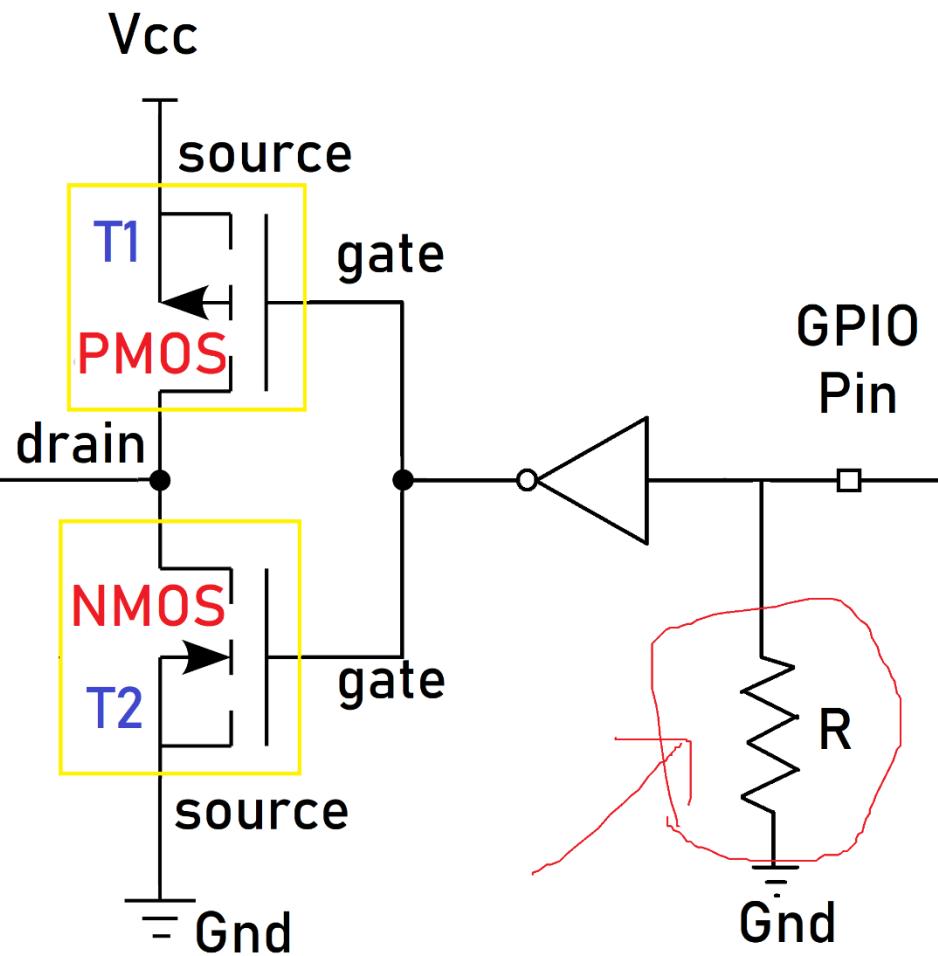


Input Buffer

Pin with pull-up resistor



نط PULL-DOWN □



Input Buffer

التابع المستخدمة من مكتبة HAL للتحكم بالمخارج الرقمية في متحكم STM32

- نستخدم التابع التالي لمعرفة حالة الدخل الرقمي على أحد أقطاب المتحكم

```
HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t  
GPIO_Pin);
```

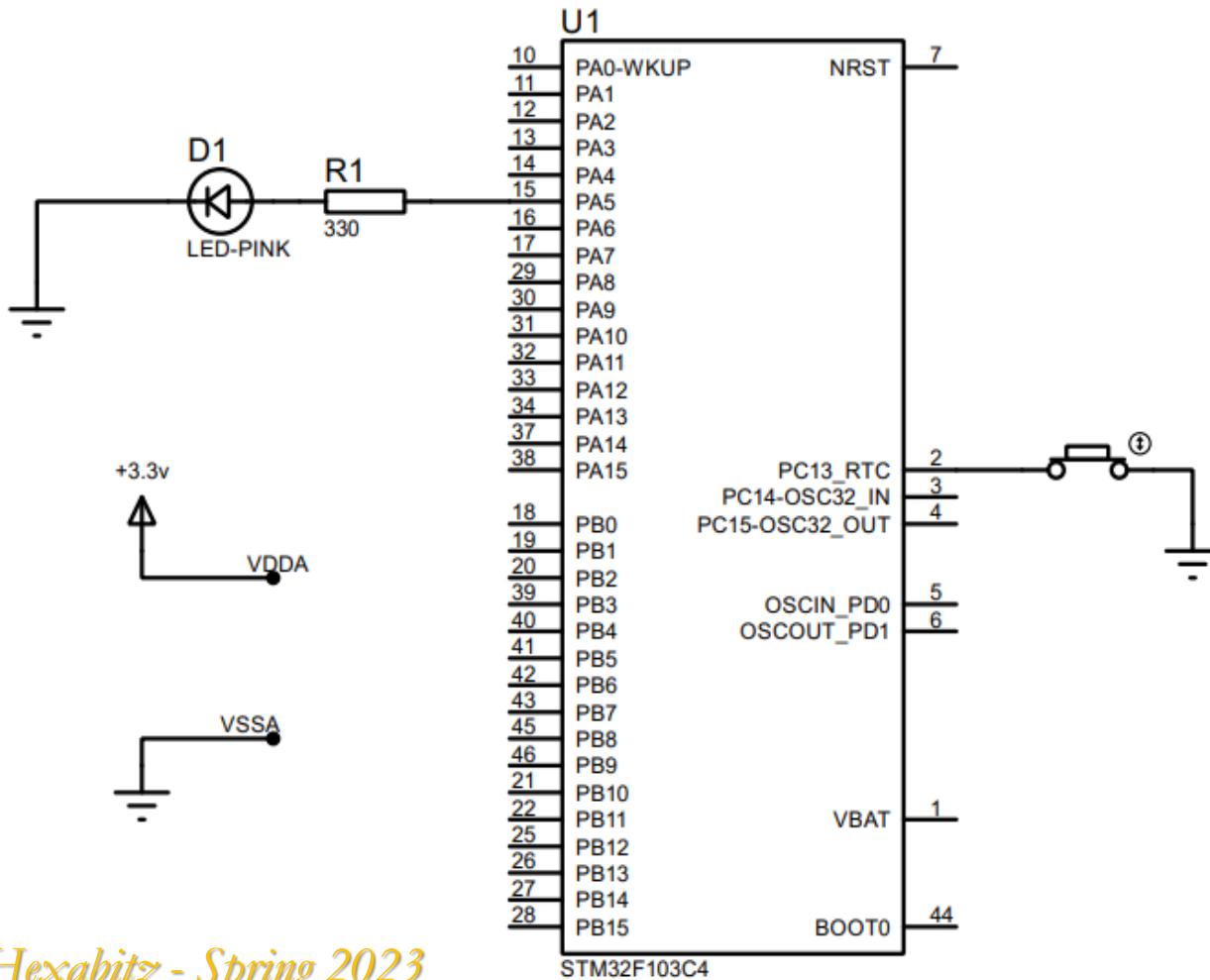
حيث يعيد هذا التابع 0 في حال كانت الحالة المنطقية للقطب في حالة جهد منخفض ، ويعيد 1 في حال كانت الحالة المنطقية للقطب في حالة جهد مرتفع.

مثال: لقراءة حالة الدخل الرقمي على القطب رقم 13 من المنفذ E نستخدم التابع التالي:

```
HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_13);
```

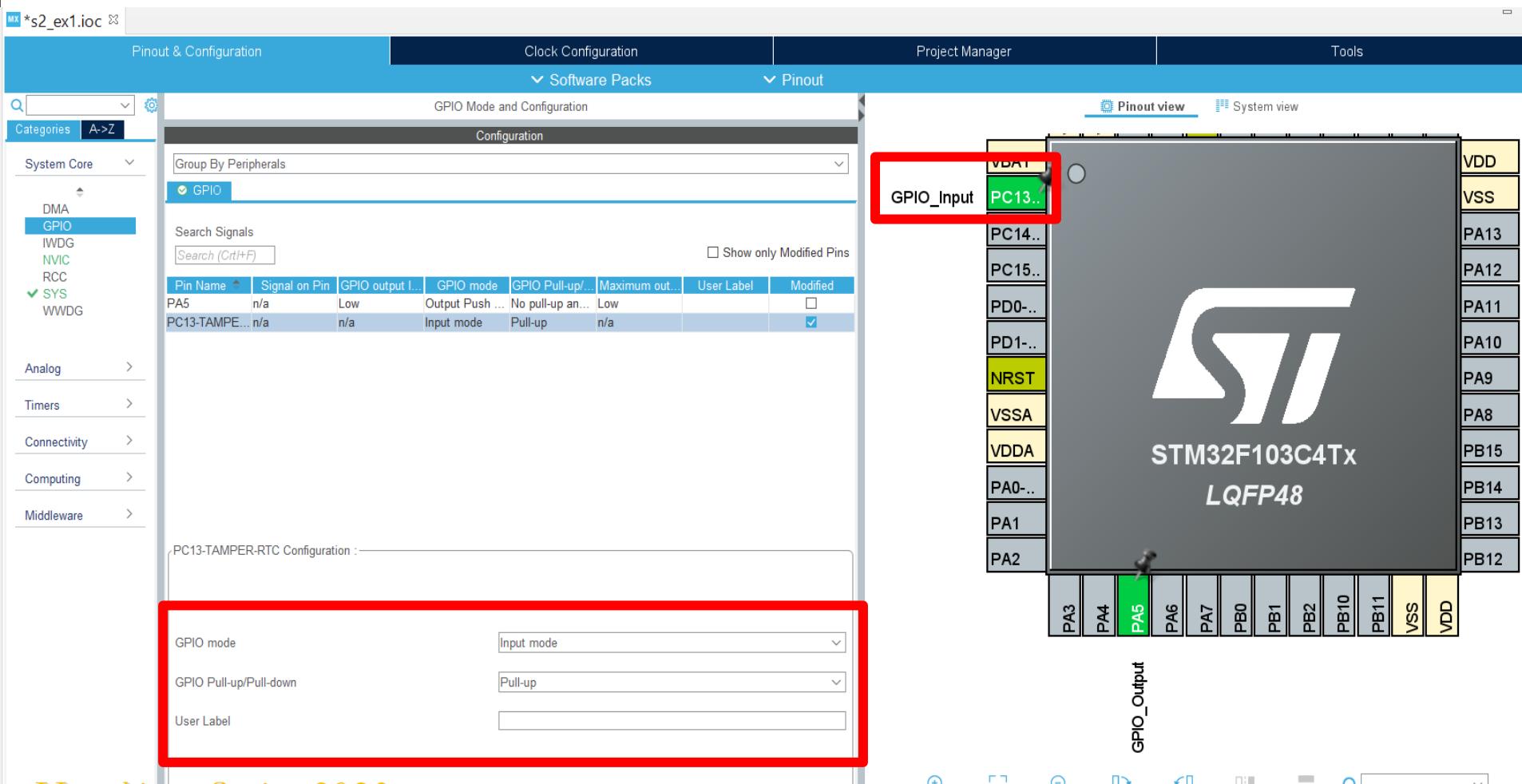
التطبيق الأول: إضاءة ليد من خلال مفتاح لحظي باستخدام متحكمات HAL و مكتبة STM32

□ بناء تطبيق لإضاءة ليد من خلال مفتاح لحظي باستخدام متحكمات HAL و مكتبة stm32



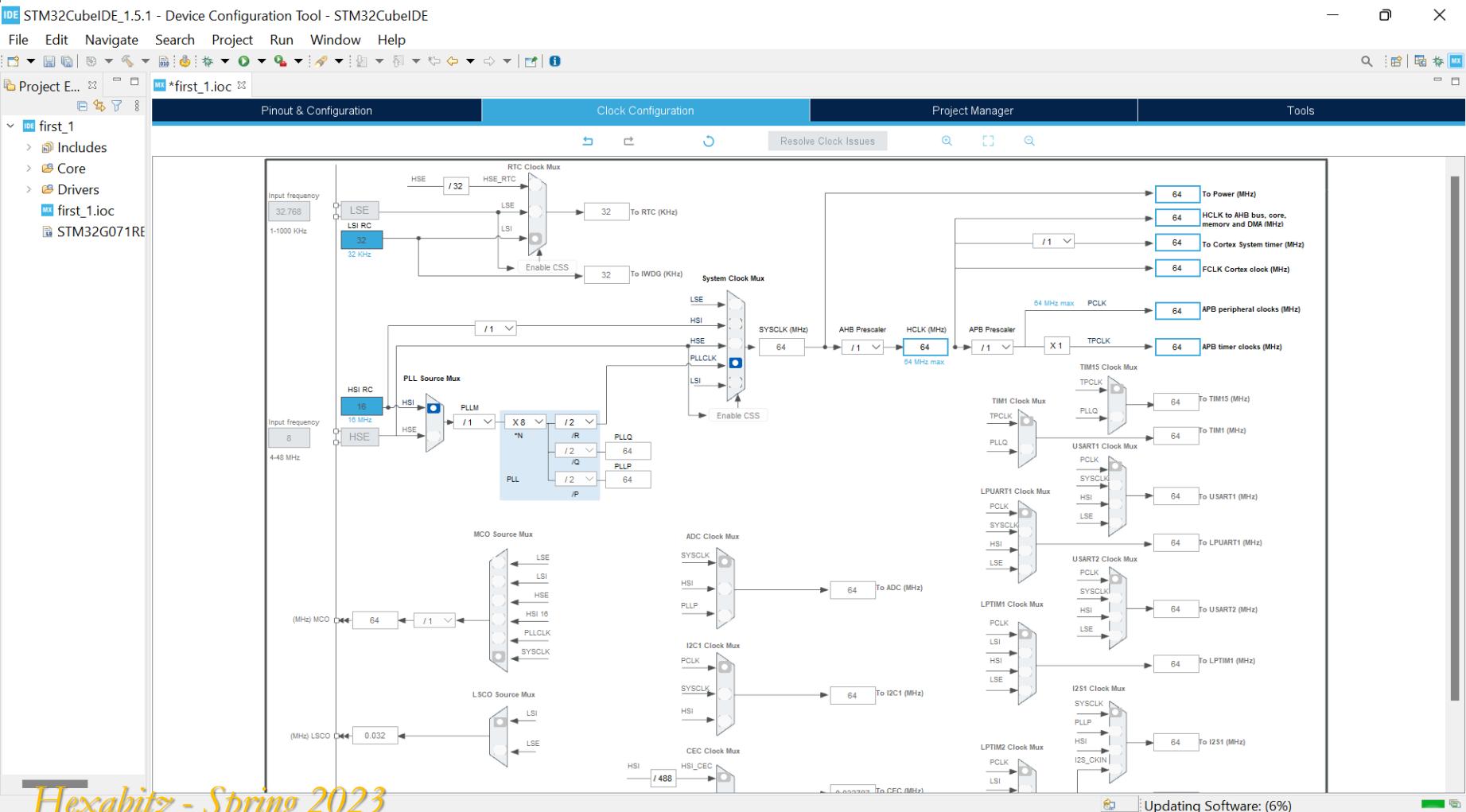
التطبيق الأول: إضاءة لمبة وإطفاؤه كل 0.5 sec باستخدام متحكمات HAL ومكتبة STM32

نقوم بضبط إعدادات القطب خرج PA5 كقطب خرج والقطب PC13 كقطب دخل



التطبيق الأول: إضاءة ليد وإطفاؤه كل sec 0.5 باستخدام متحكمات HAL ومتيبة STM32

نقوم بضبط تردد الساعة للمتحكم



التطبيق الأول: إضاءة لمبة وإطفاؤه كل sec 0.5 باستخدام متحكمات HAL و مكتبة STM32

نقوم بالضغط على **Project...Generate** أو من **Ctrl+s**، ليتم حفظ المشروع وتوليد الكود ثم نقوم بإضافة الجزء التالي:

```
#include "main.h"  
void SystemClock_Config(void);  
static void MX_GPIO_Init(void);  
int main(void)  
{  
    HAL_Init();  
    SystemClock_Config();  
    MX_GPIO_Init();
```

التطبيق الأول: إضاءة ليد وإطفاؤه كل 0.5 sec باستخدام متحكمات HAL وكتبة STM32

```
while (1){  
if(HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13)==0)  
{  
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,1);  
}  
else  
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,0);  
}}}
```

للتحكم بالمداخل الرقمية للمتحكم STM32 دون استخدام مكتبة HAL:

□ هناك مجموعة من المسجلات تستخدم للتحكم بالمدخلات الرقمية لمتحكم STM32 سنكتفي فقط بذكر المسجل المسؤول عن قراءة حالة المنفذ أو أحد الأقطاب الموجودة فيه ويدعى Input data register (IDR) له الشكل التالي:

7.4.5 GPIO port input data register (GPIOx_IDR) (x = A..H)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 IDRy: Port input data (y = 0..15)

These bits are read-only and can be accessed in word mode only. They contain the input value of the corresponding I/O port.

للحكم بالمخارج الرقمية للمتحكم STM32 دون استخدام مكتبة HAL:

□ والمسجل IDR هو مسجل للقراءة فقط ، البات 16:31 غير مستخدمين، أما البات 0:15 فيعبر كل بت عن حالة القطب المقابل له، ففي حال تفعيل مقاومة الرفع الداخلية للقطب عندها سيكون القطب في حالة HIGH بشكل دائم، وعند ضغط المفتاح الموصول معه سيصبح القطب في حالة LOW، لذا يجب مراقبة حالة القطب بشكل مستمر لحين يصبح القطب في حالة LOW عندها يكون المفتاح الموصول معه مضغوط.

□ فلفحص حالة القطب الأول من المنفذ A نستخدم جملة الشرط التالية:

□ if (!(GPIOA->IDR &(1<<1)))

```
#include "main.h"

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void)
{
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
```

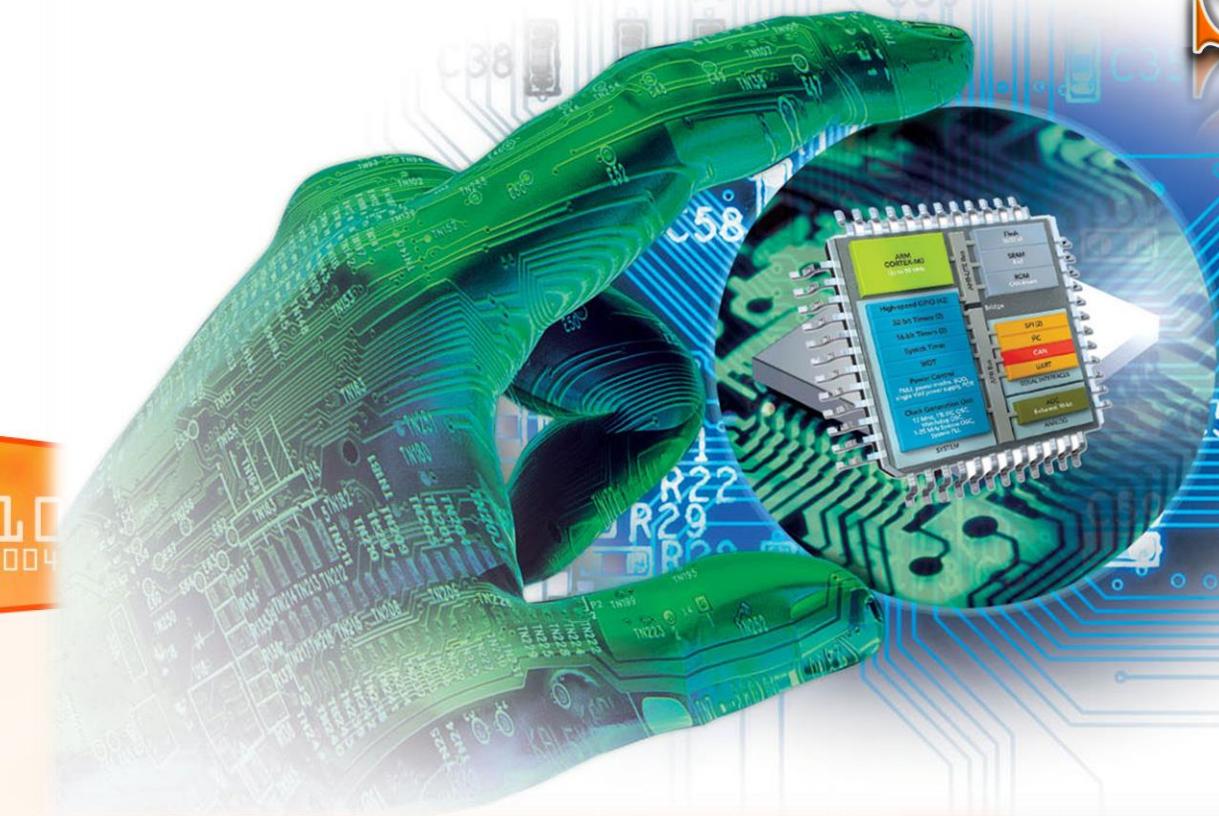
```
while (1)
{
    if (!(GPIOC->IDR &(1<<13)))
    {
        GPIOA->ODR = 1<<5;
    }
    else
        GPIOA->ODR &= ~(1<<5);
}
```

Thank you for listening

مُتَحَكِّمَات

STM32

3



موجو عاٹن المعاڑة:

- Exceptions Overview
- Micro-Coded Interrupts
- Exception Types
- System Exceptions
- متڪم NVIC
- أنماط عمل المعالج
- مصطلاح الـ Preemption
- Interrupt Lifecycle
- Configure Interrupts

نظرة عامة عن مفهوم الاستثناءات: Exceptions Overview

- المقاطعة هي حدث غير متزامن يتسبب في إيقاف تنفيذ الكود الحالى اعتمادا على مبدأ الأولوية
- تنشأ المقاطعات من خلال الـ hardware أو من البرنامج نفسه الـ software
- يتم التحكم بالمقاطعات من خلال الـ NVIC(Nested Vectored Interrupt Controller)
- توفر معالجات ARM هيكيلية خاصة للتعامل مع الاستثناءات/ المقاطعات وتدعى Micro-Coded Architecture حيث يتم تكديس المقاطعات ثم البدء بمعالجتها حسب أولويتها ثم العودة لاستكمال تنفيذ البرنامج الرئيسي وكل ذلك يتم Hardware أي بدون التدخل من قبل المعالج مما يعني سرعة استجابة أعلى للاستثناءات وتخفيض الضغط عن المعالج بالإضافة إلى مرونة أعلى في العمل والقدرة على دعم أنظمة الزمن الحقيقي RTOS

نظرة عامة عن مفهوم الاستثناءات: Exceptions Overview

يتم الدخول إلى المقاطعة والخروج منها عن طريق الـ **Hardware** من خلال الخطوات التالية:

- ❑ حفظ واستعادة (processor context) وتشمل حالة المعالج لحظة حدوث المقاطعة بما في ذلك حالة وقيم المسجلات) عند حدوث المقاطعة وعند العودة منها لاستكمال تنفيذ البرنامج الرئيسي انطلاقاً من التعليمة التي تم التوقف عنها عند حدوث المقاطعة.
- ❑ يُسمح باكتشاف الوصول المتأخر للمقاطعات/ الاستثناءات وخدمتها بناءً على مستوى الأولوية لها.
- ❑ يُسمح بخدمة المقاطعة المعلقة التالية عند الانتهاء من خدمة المقاطعة الحالية دون الحاجة لإعادة مرحلة (حفظ/ استعادة) حالة المعالج (processor context) مما يزيد من سرعة الاستجابة للمقاطعات وتدعي هذه الخاصية بـ **tail-chaining**.

أنواع الاستثناءات: Exception Types

هناك نوعين من الاستثناءات هما:

□ **system exception** يتم توليدها من قبل المعالج نفسه داخلياً

□ **interrupts** المقاطعات وتأتي من العالم الخارجي للمعالج

هناك 15 استثناء من نوع **system exception** تدعمها معالجات CORTEX-M بالإضافة إلى 240 مقاطعة ، لذا فإن معالجات CORTEX-M تدعم 255 استثناء، بحيث:

□ الاستثناء رقم واحد هو لا RESET

□ فقط 9 استثناءات من نوع **system exception** يتم التعامل معها والـ 6 الباقية هي للاستخدامات المستقبلية.

□ الاستثناء رقم 16 هو للمقاطعة رقم 1 (IRQ1).

System Exceptions

RESET هو استثناء من نوع system exception، يتم طلب هذا الاستثناء عند تغذية المعالج أو من خلال الضغط على زر الـ **Reset**

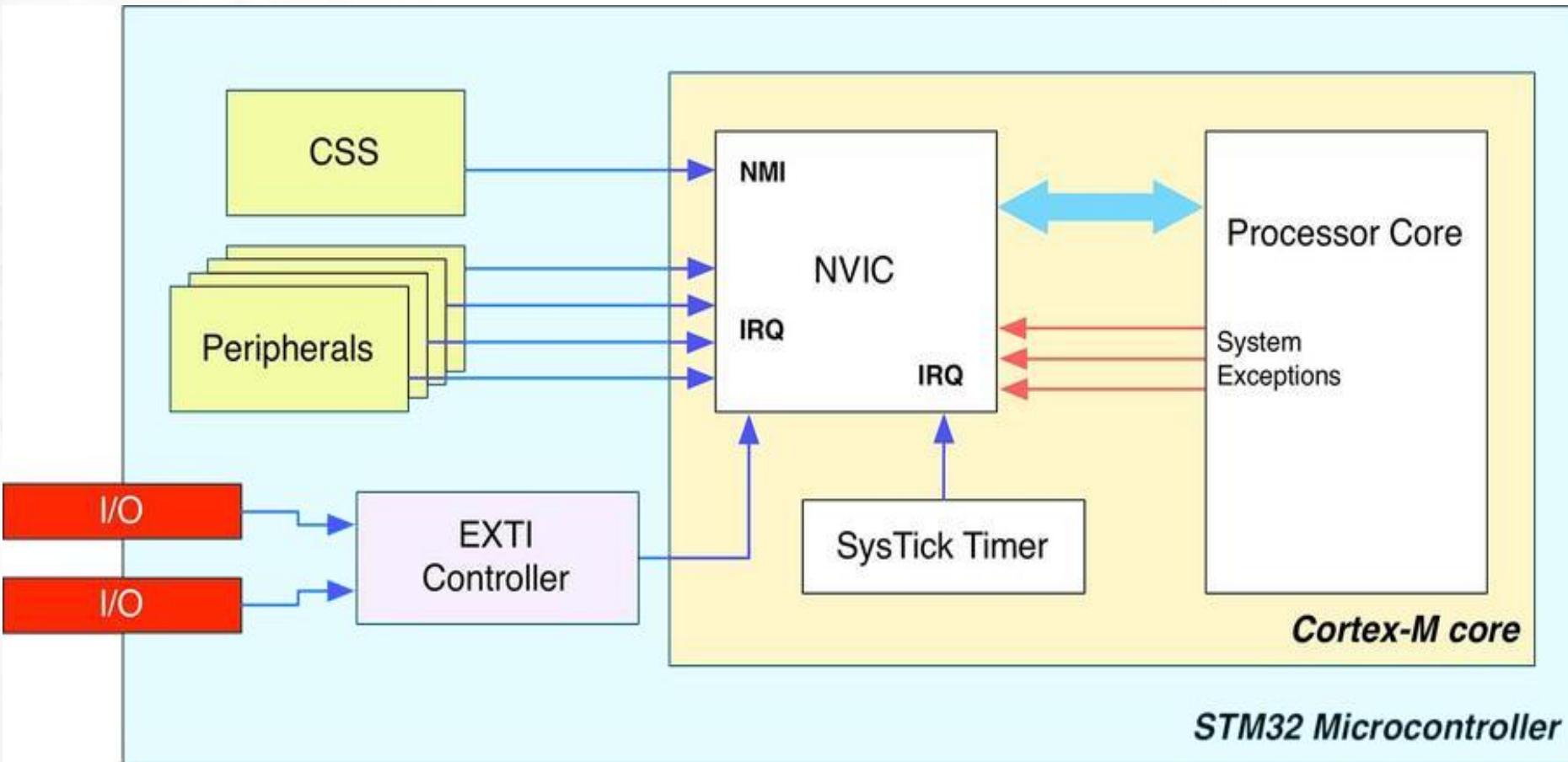
Non-Maskable Interrupt (NMI) : هي الاستثناءات التي لا يمكن إلغاء تفعيلها، يتم قدح هذا الاستثناء عند حدوث خطأ ما عند تنفيذ أحد Exception Handler، ولها أعلى أولوية بعد استثناء الـ Reset، في متحكمات STM32 يتم ربط استثناءات الـ NMI مع نظام حماية الساعة Clock security حيث الـ CSS هي وحدة طرفية تقوم بفحص حالة الساعة الخارجية HSE وفي حال اكتشاف مشكلة ما فيها تقوم بتعطيل HSE وتفعيل الساعة الداخلية HSI.

متحكم NVIC (Nested Vectored Interrupt Controller)

- هو المتحكم المسؤول عن تحديد أولويات المقاولات Interrupt
- هي عبارة عن الاستثناءات التي يتم توليدها من قبل IRQ(الطرفيات أو من قبل البرنامج
- وتحسين أداء المعالج MCU وتقليل زمن استجابة المقاولة
- يوفر NVIC أيضا خطوات محددة للتعامل مع المقاولات التي تحدث أثناء تنفيذ مقاولات أخرى أو عندما يكون المعالج في مرحلة استعادة حاليه السابقة واستئناف عمليته المعلقة التي توقف عندها بسبب حدوث المقاولة

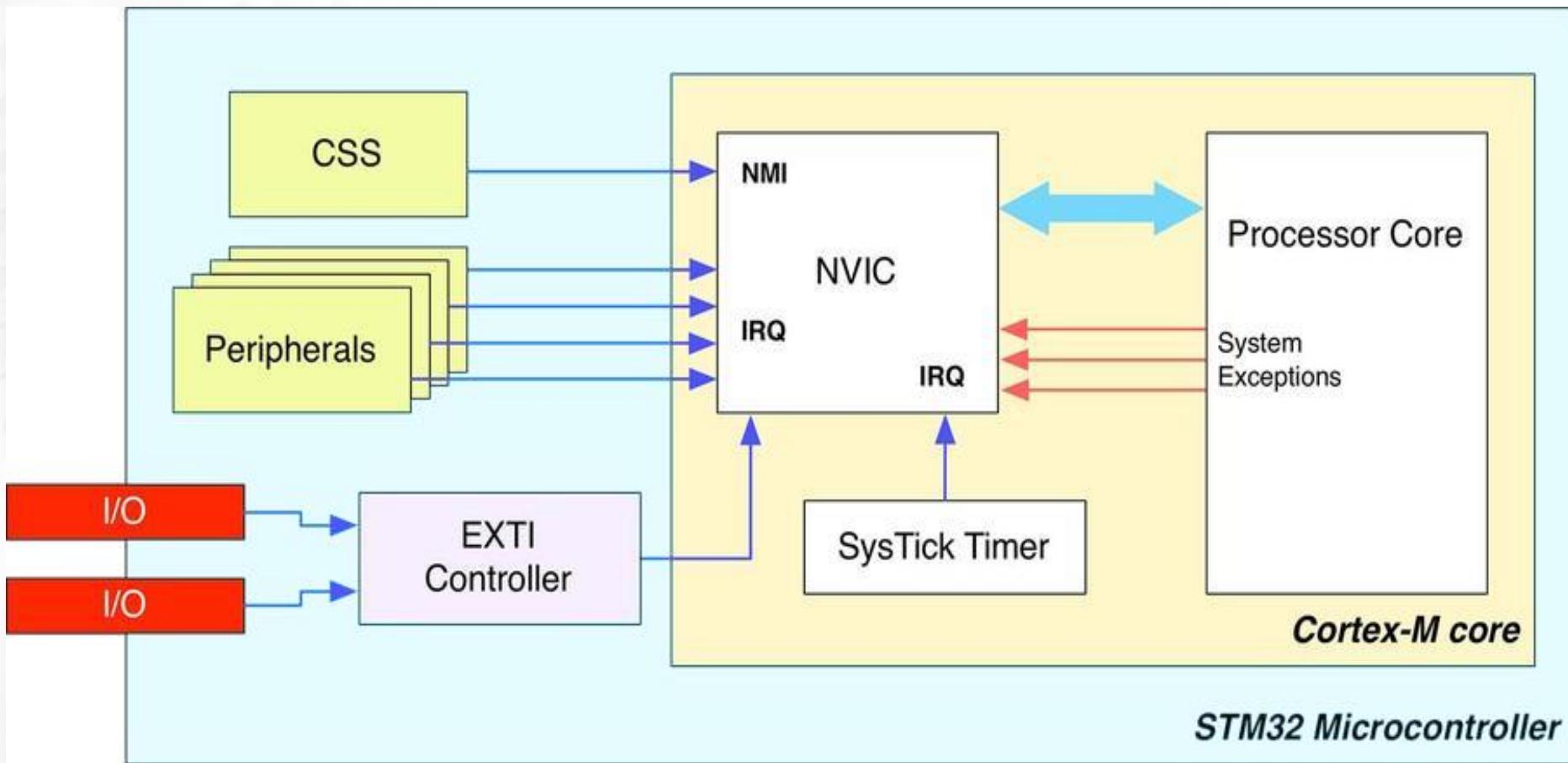
متحكم NVIC (Nested Vectored Interrupt Controller)

يتم ربط مقاطعة مع Clock Security System (CSS) خطوط مقاطعات الـ Non-Maskable Interrupt وهي المقاطعات التي لا يمكن تجاهلها أو إلغاء تفعيلها (NMI)



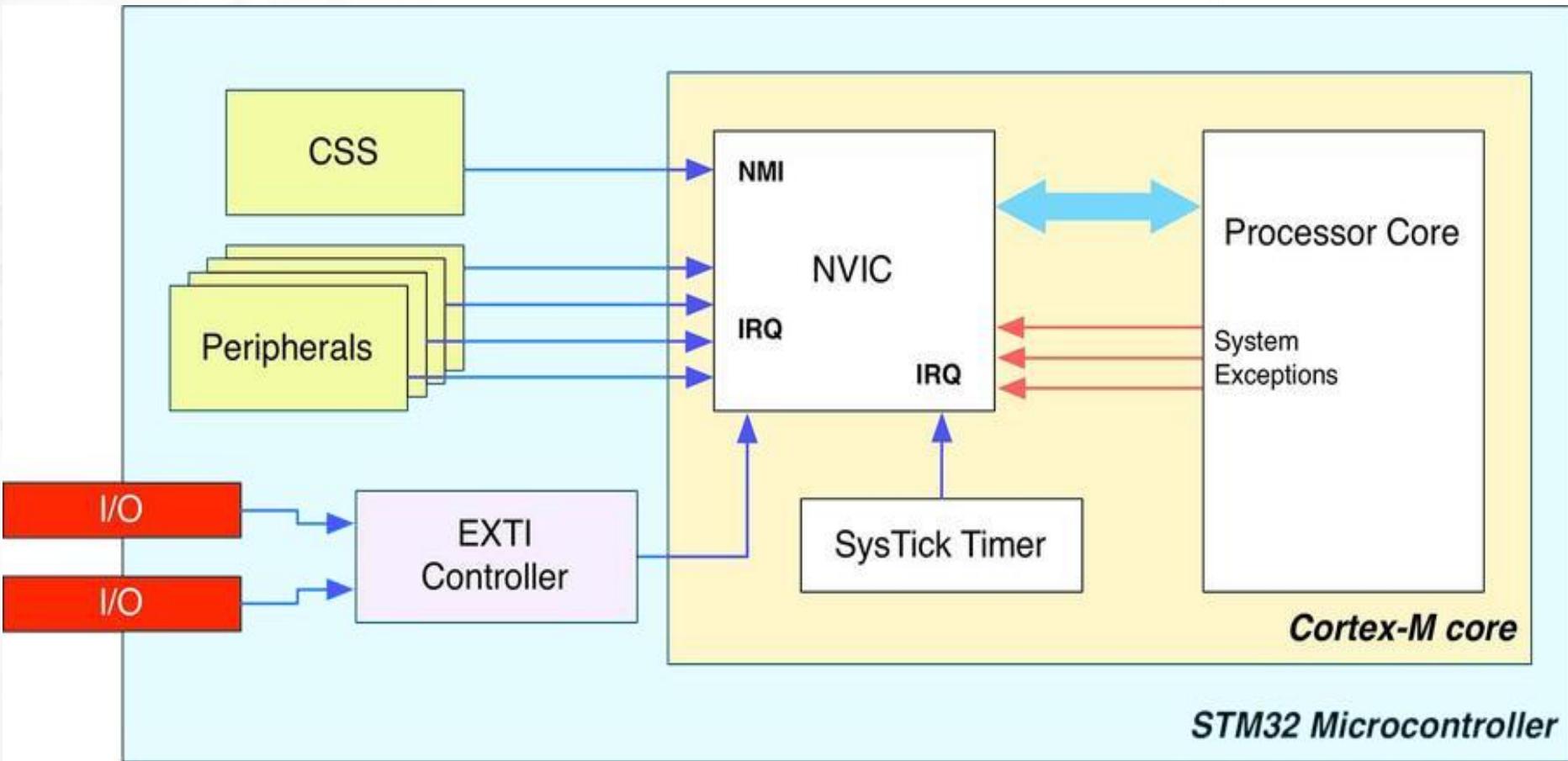
متحكم NVIC (Nested Vectored Interrupt Controller)

□ يتم ربط مقاطعات الطرفيات (مثل مقاطعات المؤقتات، المبدلات التشابهية رقمية وغيرها من الطرفيات) مع خطوط Interrupt Requests (IRQ)



متحكم NVIC (Nested Vectored Interrupt Controller)

المقاطعات الخارجية القادمة من الـ **GPIO** يتم ربطها مع **External Interrupt/Event Controller (EXTI)** ليتم بعد ذلك ربطها مع خطوط **IRQ**، كما هو موضح بالشكل التالي



جدول أشعة المقاطعة interrupts table Vectors

Number	Exception Type	Priority	Function
1	Reset	-3	Reset
2	NMI	-2	Non-Maskable Interrupt
3	Hard Fault	-1	All faults that hang the processor
4	Memory Fault	Configurable	Memory issue
5	Bus Fault	Configurable	Data bus issue
6	Usage Fault	Configurable	Data bus issue
7 ~ 10	Reserved	—	Reserved
11	SVCall	Configurable	System service call (SVC instruction)
12	Debug	Configurable	Debug monitor (via SWD)
13	Reserved	—	Reserved
14	PendSV	Configurable	Pending request for System Service call
15	SysTick	Configurable	System Timer
16 ~ 240	IRQ	Configurable	Interrupt Request

المقاطعات الخارجية External Interrupts

- يحتوي متحكم STM32 على 16 خط للمقاطعات الخارجية هي من الخط 0 حتى الخط 15
- تشير أرقام الخطوط إلى أرقام اطراف ال-GPIOS
- هذا يعني أن الطرف PA0 متصل بالخط LINE0 والطرف PA13 متصل بالطرف LINE13، أيضاً PC0 و PB0 متصلين بالخط LINE0، بمعنى أن جميع الأطراف ذات الأرقام متصلة بالخط (LINE0 حيث يعبر X عن اسم المنفذ) ، أيضاً جميع الأطراف ذات الأرقام PX3 متصلة بالخط LINE3 وهكذا...
- في كل خط من خطوط المقاطعات الخارجية (كل مجموعة) يحق لـ pin واحد أن يقوم بتوليد المقاطعة وعلى البرنامج أن يكتشف أي pin قام بتوليد المقاطعة

المقاطعات الخارجية External Interrupts

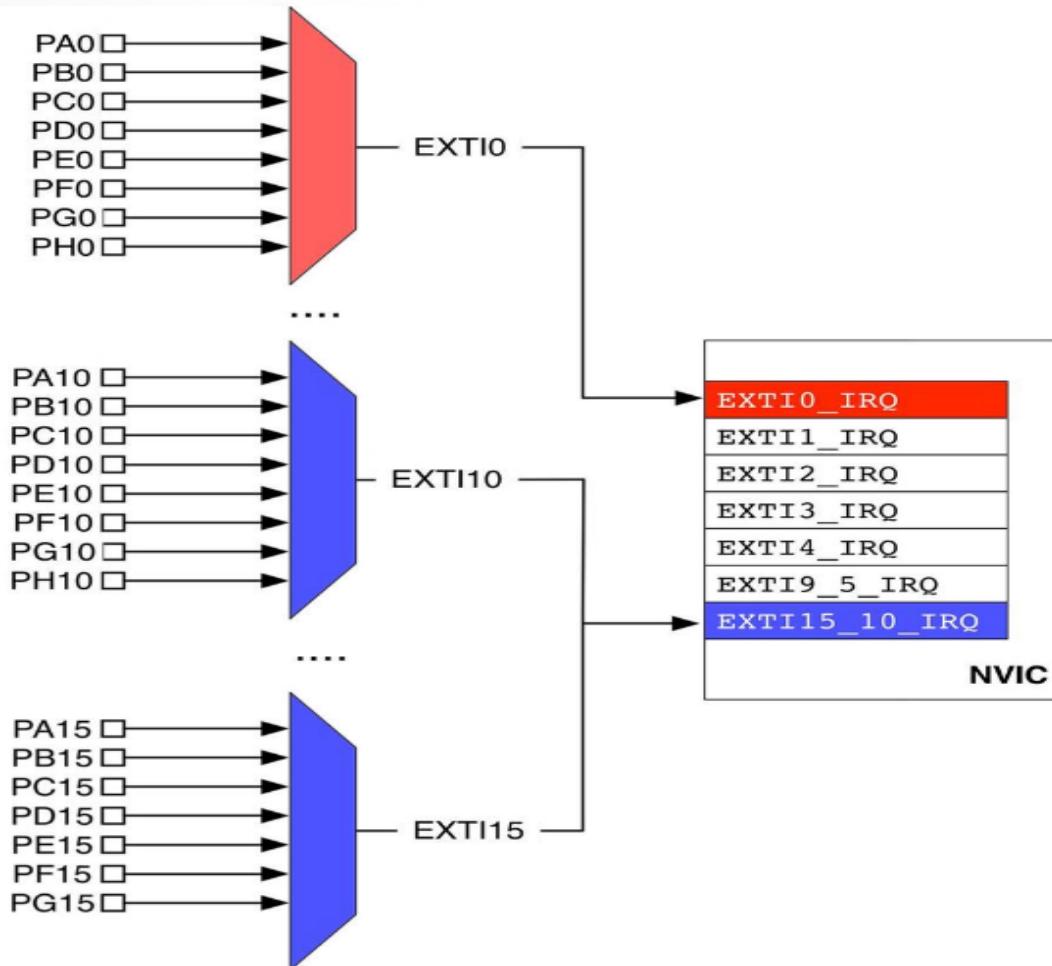
يجب الانتباه للملاحظات التالية:

☐ الأطراف LINE0... PA0, PB0, PC0... جميعها متصلة بالخط PA0 لذا

في اللحظة الواحدة

بإمكانك توليد مقاطعة على طرف واحد فقط من هذه الأطراف.

☐ الأطراف PA5, PA0 متصلين على خطين مختلفين من خطوط المقاطعة لذا يمكنك استخدامهم في نفس اللحظة لتوليد المقاطعة.



المقاطعات الخارجية External Interrupts

يمكن قذح المقاطعة عند الجبهة الصاعدة أو الهابطة أو عند كليهما

GPIO_MODE_IT_RISING



GPIO_MODE_IT_RISING_FALLING

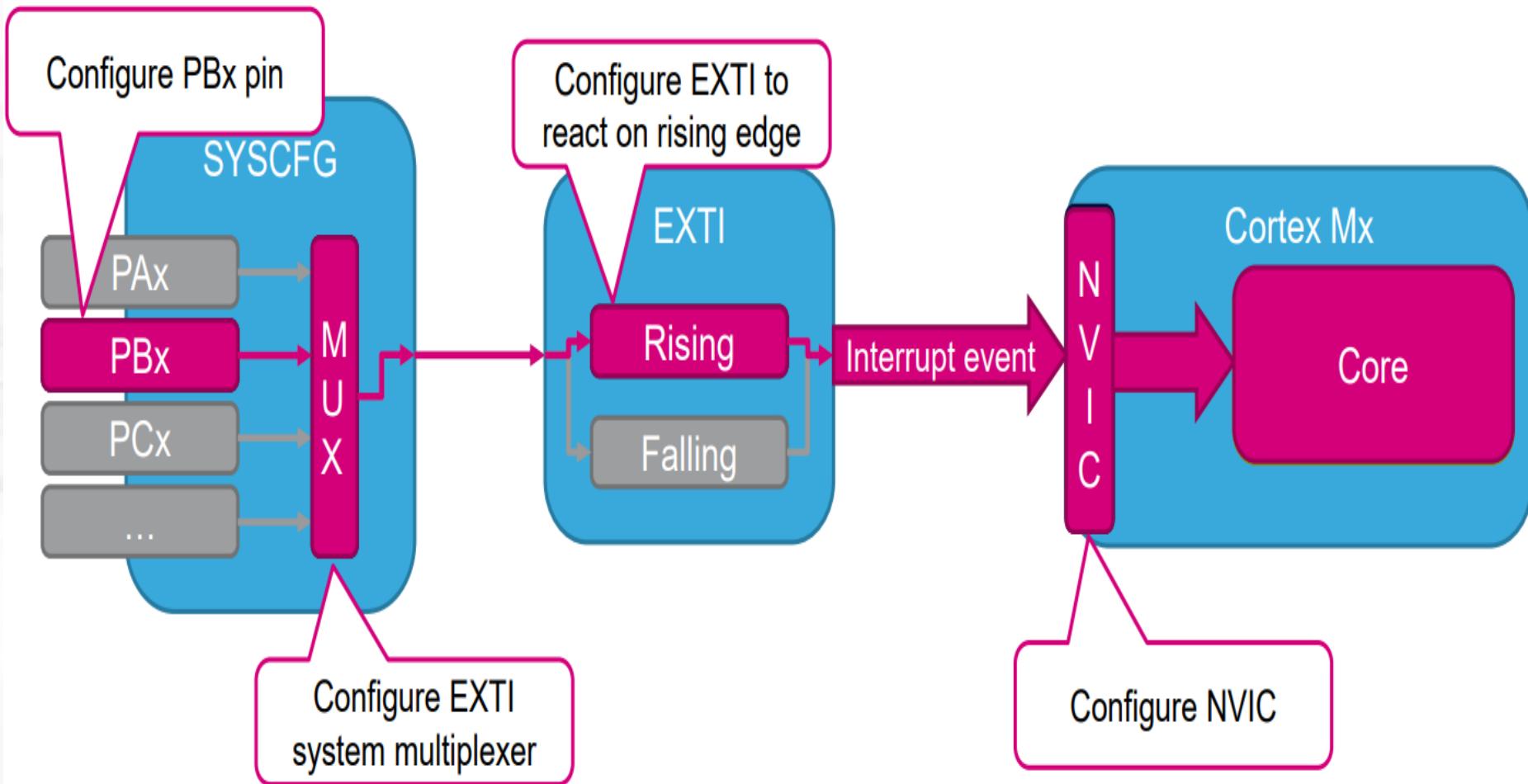


GPIO_MODE_EVT_FALLING



المقاطعات الخارجية External Interrupts

فتحدث المقاطعة الخارجية وفقاً للآلية التالية:



أنماط عمل المعالج Processor mode

للمعالج نمطي عمل أساسين:

في Thread Mode : يكون المعالج في نمط Thread Mode

الوضع الطبيعي له عند تنفيذ الكود أو عند عمل him reset

Handler Mode : يدخل المعالج في نمط Handler Mode

عند حدوث استثناء / مقاطعة حيث يتم حفظ المكان الذي تم توقف

المعالج عنده كما يتم حفظ حالة المسجلات والأعلام flags بشكل

آلي وعن طريق hardware ليضمن استجابة سريعة لحدث

المقاطعة

Exception Behavior

عند حدوث استثناء ما، يتم إيقاف تنفيذ التعليمات الحالية ويتم توجيه المعالج لجدول أشعة المقاطعة حيث:

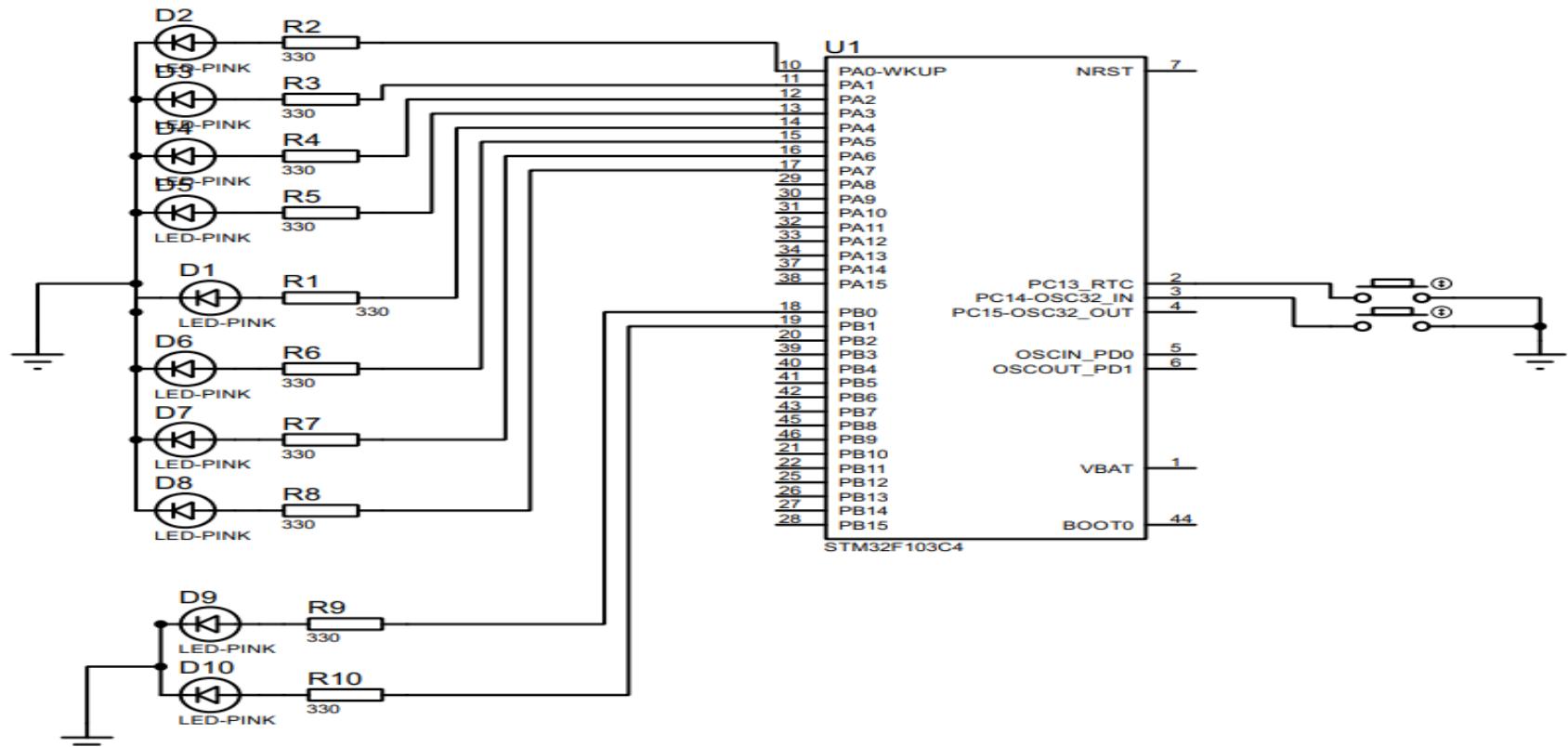
- يقوم المعالج بإنهاe التعليمية الحالية التي يقوم بتنفيذها طالما أنها لا تحتاج لأكثر من دورة تعليمية
- يتم حفظ حالة المعالج (عنوان التعليمية التي وصل لعندتها حالياً) إلى المكدس
- يتم تحميل عنوان برنامج خدمة المقاطعة الحاصلة من جدول أشعة المقاطعة
- يتم البدء بتنفيذ برنامج خدمة المقاطعة ISR، ويستغرق تنفيذ كامل هذه العملية 12 cycles في معالجات
- يقوم برنامج خدمة المقاطعة بتصغير علم المقاطعة التي قامت باستدعائه

Exception Behavior

- في نهاية برنامج خدمة المقاطعة يتم التأكيد من عدم وجود مقاطعة في الانتظار من أجل الانتقال لتنفيذ برنامج خدمة المقاطعة التالية دون استعادة حالة المعالج السابقة بهدف زيادة سرعة الاستجابة للمقاطعة.
- يتم تنفيذ تعليمات **EXC_RETURN** لاستعادة حالة المعالج قبل حدوث المقاطعة
- تستغرق العودة من المقاطعة (استعادة حالة المعالج) في معالجات ARM Cortex-M3/M4 حوالي 10 clock cycles

التطبيق الأول: برمجة زر للطوارئ لاستخدامه في الحالات الحرجية

□ زر طوارئ لتفعيل برنامج الطوارئ حيث يتم تنفيذه بغض النظر عن عمل النظام (إنارة اللدات) و بعد الانتهاء من برنامج خدمة المقاطعة (الطوارئ) يعود للبرنامج الرئيسي ليتابع عمله



التطبيق الأول: برمجة زر للطوارئ لاستخدامه في الحالات الحرجة

نقوم بضبط الأقطاب PB0:PB1، والأقطاب PA0:PA6 كأقطاب
خرج ، والقطب PC13 كقطب دخل مع تفعيل مقاومة الرفع
الداخلية، والقطب PC14 كقطب مقاطعة خارجية

The screenshot shows the STM32CubeMX software interface. On the left, the project navigation bar includes files like *s3_ex1.ioc, main.c, and stm32f1xx_hal_gpio.c. The main window has tabs for Pinout & Configuration, Clock Configuration, Project Manager, and Tools. The Pinout & Configuration tab is active, displaying the GPIO Mode and Configuration table. The table lists pins PA0-WKUP through PA6, PB0, PB1, PC13-TAMPER-..., and PC14-OSC32_IN. PC13 is configured as an input mode with pull-up, and PC14 is configured as an external interrupt input. The right side of the interface shows the physical pinout of the STM32F103C4Tx LQFP48 package, with pins labeled from PA3 to VDD. The ST logo and part number are also visible.

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pu...	Maximum output...	User Label	Modified
PA0-WKUP	n/a	Low	Output Push Pull	No pull-up and n...	Low		
PA1	n/a	Low	Output Push Pull	No pull-up and n...	Low		
PA2	n/a	Low	Output Push Pull	No pull-up and n...	Low		
PA3	n/a	Low	Output Push Pull	No pull-up and n...	Low		
PA4	n/a	Low	Output Push Pull	No pull-up and n...	Low		
PA5	n/a	Low	Output Push Pull	No pull-up and n...	Low		
PA6	n/a	Low	Output Push Pull	No pull-up and n...	Low		
PB0	n/a	Low	Output Push Pull	No pull-up and n...	Low		
PB1	n/a	Low	Output Push Pull	No pull-up and n...	Low		
PC13-TAMPER-...	n/a	n/a	Input mode	Pull-up	n/a		<input checked="" type="checkbox"/>
PC14-OSC32_IN	n/a	n/a	External Interrupt...	Pull-up	n/a	int_button	<input checked="" type="checkbox"/>

PC13-TAMPER-RTC Configuration :

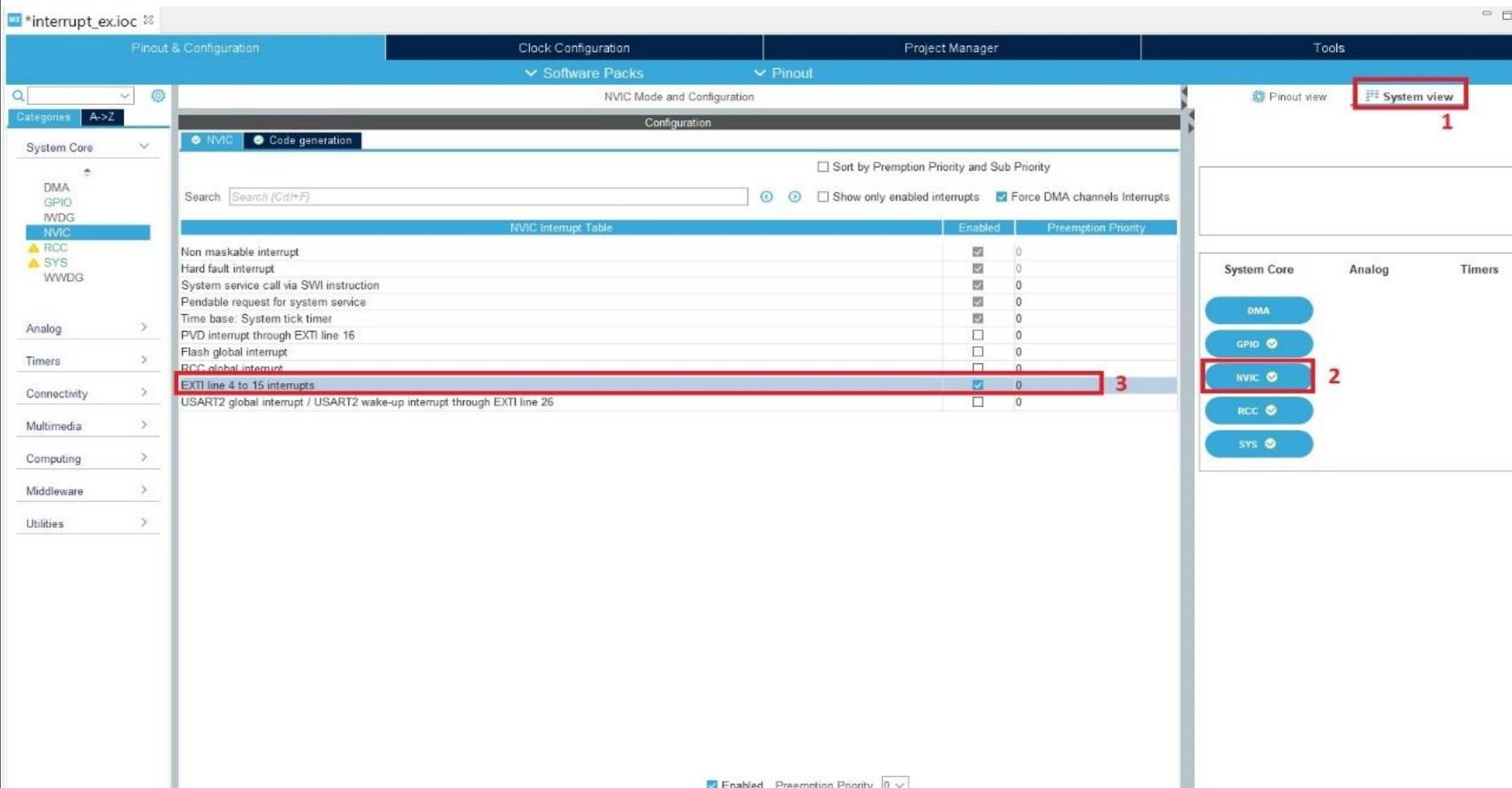
GPIO mode: Input mode

GPIO Pull-up/Pull-down: Pull-up

User Label: int_button

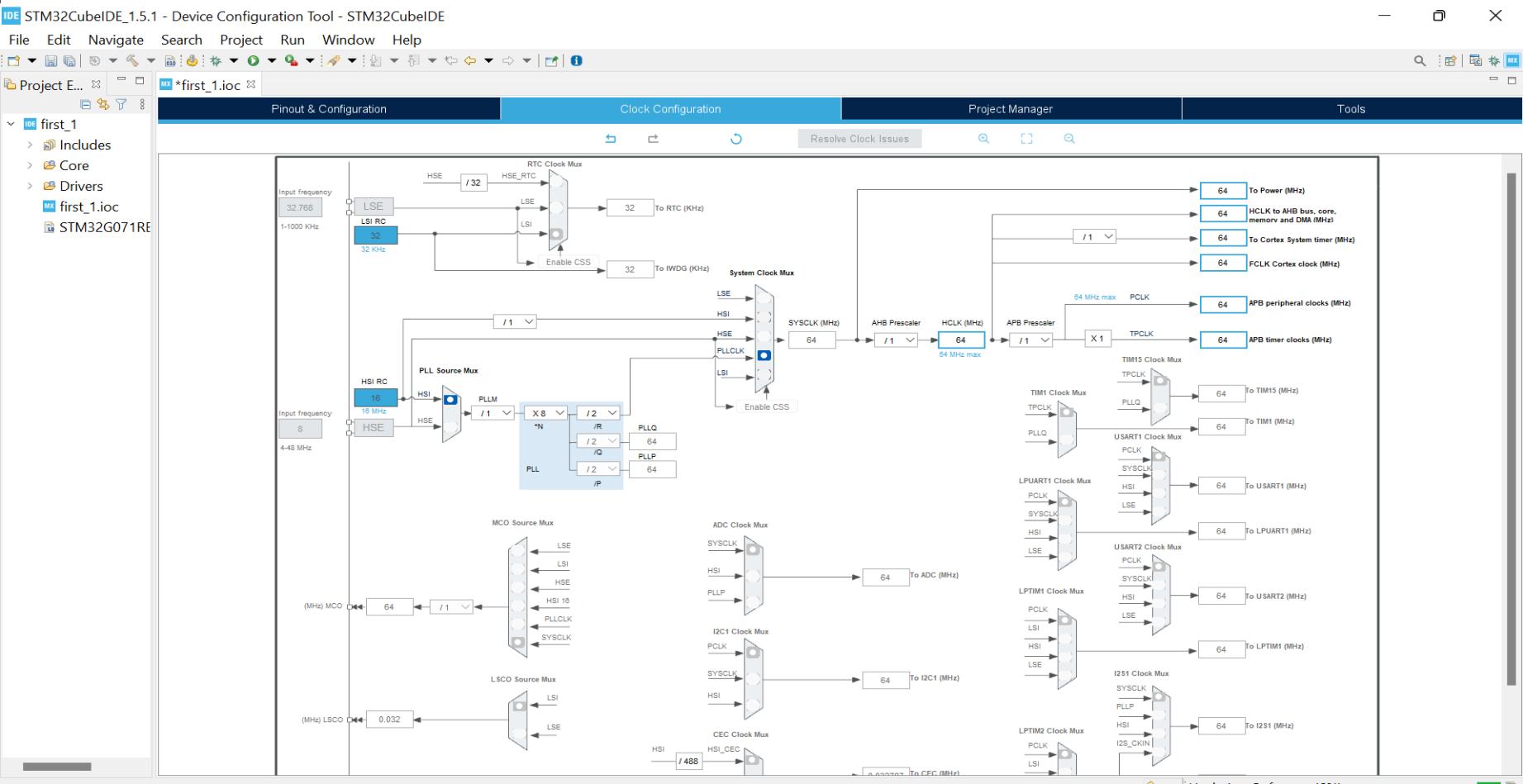
التطبيق الأول: برمجة زر للطوارئ لاستخدامه في الحالات الحرجية

قم بفتح NVIC Tab ثم قم بتفعيل المقاولات الخارجية، يمكنك أيضاً إعادة ضبط مستويات الأولوية للمقاولات:



التطبيق الأول: برمجة زر للطوارئ لاستخدامه في الحالات الحرجة

نقوم بضبط تردد الساعة للمتحكم



التطبيق الأول: برمجة زر للطوارئ لاستخدامه في الحالات الحرجة

□ نقوم بالضغط على Project...Generate أو من Ctrl+s ، ليتم حفظ المشروع وتوليد الكود ثم نقوم بإضافة الجزء التالي:

```
#include "main.h"

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin ==GPIO_PIN_14)
        {HAL_GPIO_WritePin(GPIOB,
GPIO_PIN_0|GPIO_PIN_1 , GPIO_PIN_SET);
}
```

برنامج خدمة المقاطعة

التطبيق الأول: برمجة زر للطوارئ لاستخدامه في الحالات الحرجة

```
int main(void)
{
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    while (1)
    {
        if(HAL_GPIO_ReadPin(GPIOC,
GPIO_PIN_13)==0)
        {
            GPIOA->ODR = 0X0001;
            HAL_Delay(50);
        }
        //*****
    }
}
```

التطبيق الأول: برمجة زر للطوارئ لاستخدامه في الحالات الحرجة

```
GPIOA->ODR = 0X0002;
```

```
HAL_Delay(50);
```

```
//*****
```

```
GPIOA->ODR = 0X0004;
```

```
HAL_Delay(50);
```

```
//*****
```

```
GPIOA->ODR = 0X0008;
```

```
HAL_Delay(50);
```

```
//*****
```

```
GPIOA->ODR = 0X0010;
```

```
HAL_Delay(50);
```

```
//*****
```

التطبيق الأول: برمجة زر للطوارئ لاستخدامه في الحالات الحرجة

```
GPIOA->ODR = 0X0020;  
HAL_Delay(50);  
//*****  
GPIOA->ODR = 0X0040;  
HAL_Delay(50);  
//*****  
GPIOA->ODR = 0X0080;  
HAL_Delay(100);  
{}
```

التطبيق الأول: برمجة زر للطوارئ لاستخدامه في الحالات الحرجة

else

{

GPIOA->ODR = 0X0000;

}

}

}

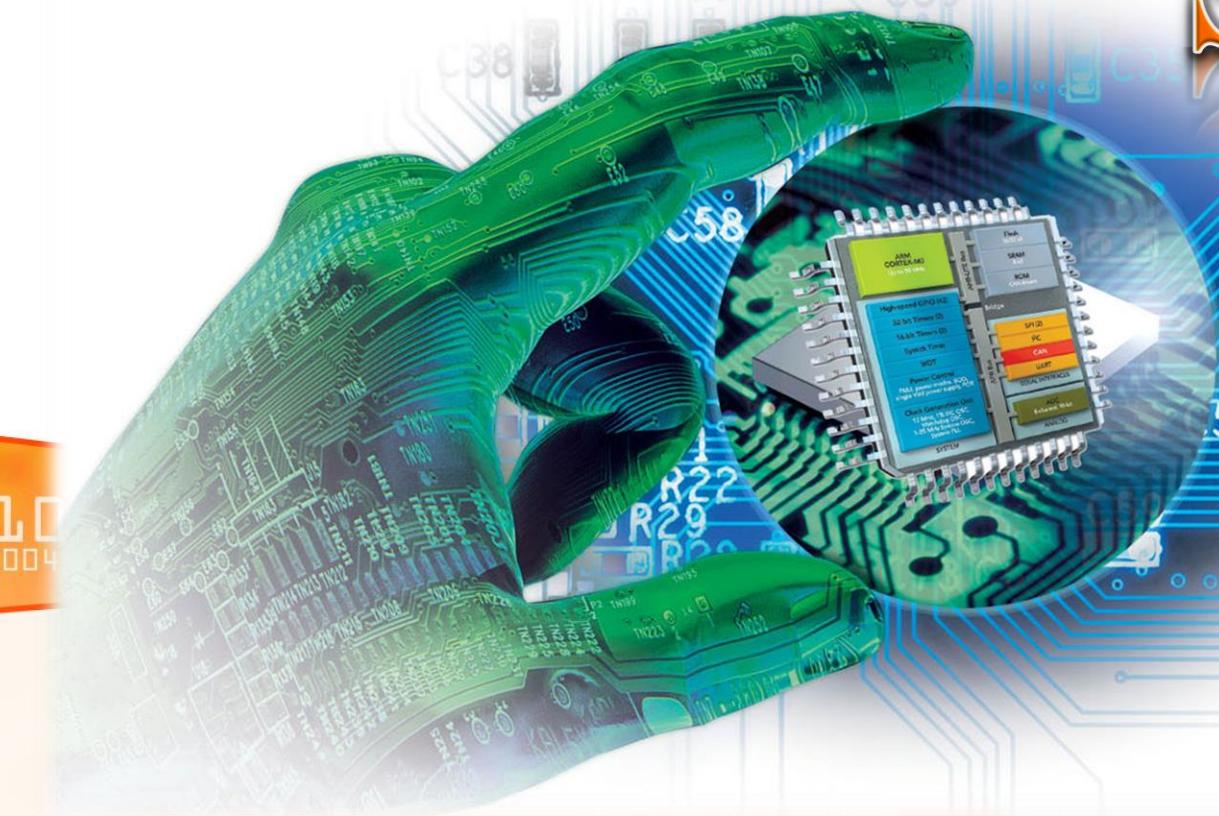
Thank you for listening

مُتَحَكِّمَات

STM32

4

5



موضو عات المعا ضر ظ :

- الأنواع المختلفة للمؤقتات
- مؤقتات الأغراض العامة
- أنماط العمل المختلفة للمؤقتات في متحكمات STM32
- ساعة الزمن الحقيقي (RTC)
- تطبيقات عملية

المؤقتات Timers

- يوجد في متحكمات STM32 العديد من المؤقتات المختلفة كل منها بإمكانها العمل بأنماط مختلفة
- المؤقت عبارة عن عداد يبدأ بالعد من الصفر ويزداد بمقدار عدة واحدة مع كل نبضة ساعة للمتحكم
- بإمكانه العد التصاعدي و التنازلي على حد سواء
- تسمح خاصية ال prescaler أو المقسم الترددی بتقسيم تردد الساعة على عدد معين يتم اختياره بين ال 0 و 65535

الأنواع المختلفة للمؤقتات

الأنواع المختلفة للمؤقتات

المؤقتات
عالية
الدقة

المؤقتات
المتقدمة

مؤقتات
الأغراض
العامة

مؤقتات
الطاقة
المنخفضة

المؤقتات
الأساسية

مؤقتات الأغراض العامة General Purpose Timers:

- المؤقتات في هذه المجموعة تكون إما 16 أو 32 بت (بناءً على عائلة STM32)
- يمتلك عداد تصاعدي / تنازلي بطول 16 بت قابل لإعادة التحميل **auto-reload counter**
- مقسم جهد بطول 16 بت يستخدم لتقسيم تردد الساعة للمتحكم على أي عدد يتراوح بين 1 و 65535

أنماط العمل المختلفة للمؤقتات في متحكمات STM32

Input
Capture

Output
Compare

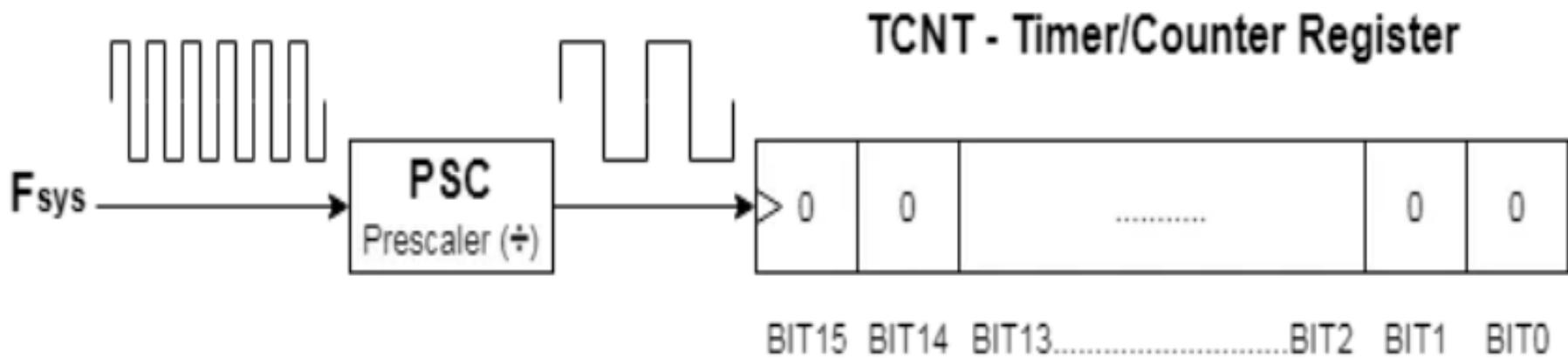
نط
 PWM

نط
 العداد

نط
 المؤقت

نُمْطُ الْمُؤْقَتِ Timer

- عند عمل المؤقت بنمط Timer، فإن المسجل TCNT تزداد قيمة بمقدار واحد مع كل نبضة ساعة للمؤقت
- يكون مصدر الساعة للمؤقت في هذه الحالة داخلي قادم من ساعة المتحكم



- حيث يقوم بالعد من الصفر إلى القيمة المحددة في حقل الـ **Period (preload)** أثناء تهيئة المؤقت ، وأكبر قيمة يمكن أن يصل إليها تحدد حسب طول المؤقت، حيث المؤقت 16 بت يمكنه العد إلى 0xfffff ffff 154

يعتمد تردد (سرعة العد) على المقسم الترددـي Prescaler حيث يتم تقسيم تردد ساعة المؤقت على واحدة من القيم المتاحة وهي من 1 حتى 65535 (حيث أن مسجل الـ Prescaler بطول 16 بت)

عندما يصل العداد إلى القيمة المحددة (preload) يحدث ما يسمى بالـ overflow أي يقوم بالتصغير والعد مرة أخرى من الصفر و يتم رفع العلم الخاص بالـ overflow Update Event(UEV)

نُمط المُؤقت Timer

ولحساب الزمن الذي سيطفح عنده المؤقت نستخدم المعادلة التالية: □

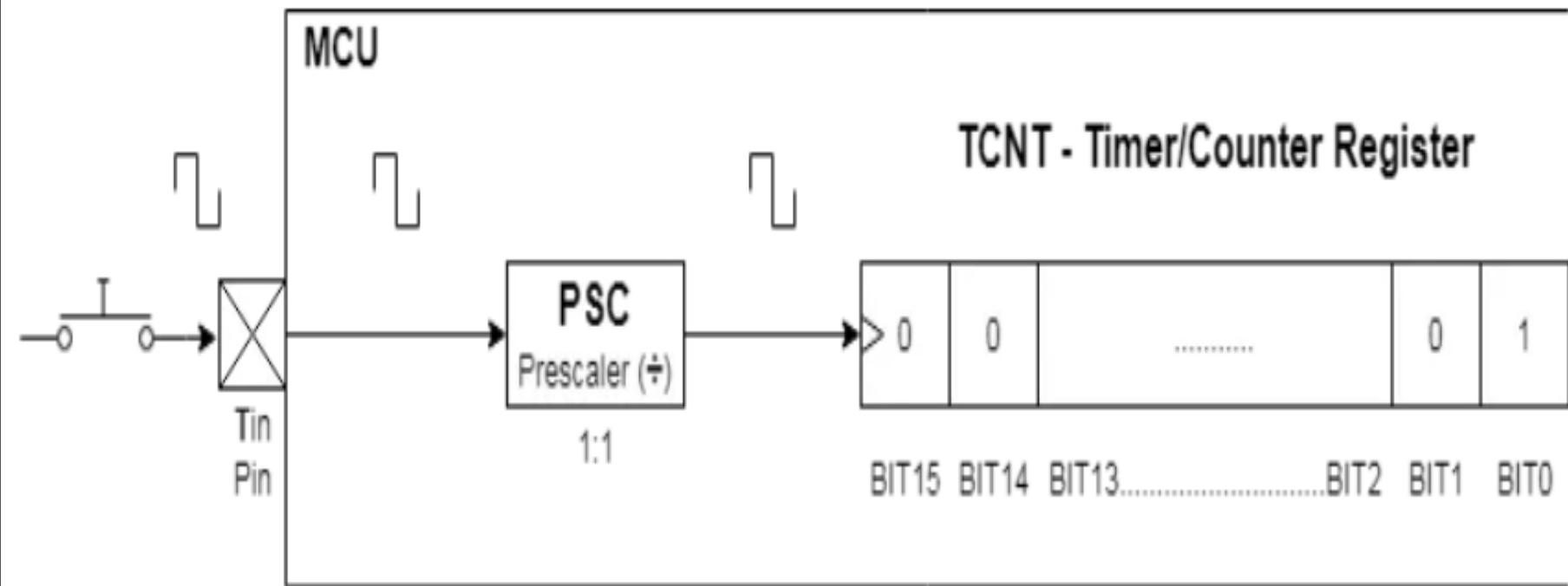
$$T_{out} = \frac{\text{Prescaler} \times \text{Preload}}{F_{CLK}}$$

على سبيل المثال ، لنفترض أن تردد الساعة للمتحكم مضبوط على MHz 48 وقيمة الـ Prescaler تساوي 48000 والـ preload تساوي 500 سيحدث overflow للمؤقت كل: □

$$T_{out} = \frac{48000 \times 500}{48000000} = 0.5sec$$

نُمطُ العَدَاد Counter

يمكن للمؤقت أن يعمل بنمط Counter وفي هذه الحالة سيكون مصدراً لـ الساعة للمؤقت عبارة عن إشارة خارجية ممكن أن تكون قادمة من مفتاح لحظي، عندما ستزداد قيمة المؤقت مع كل جبهة صاعدة/هابطة عند ضغط المفتاح اللحظي وبالتالي سيعد المؤقت عدد المرات التي تم فيها ضغط المفتاح اللحظي



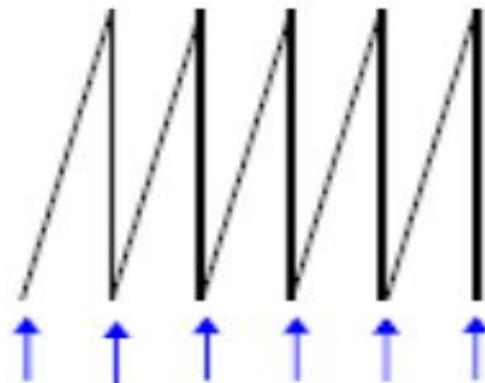
نُمطُ العَدَاد Counter

ويمكنه أن يعمل بثلاث أنماط مختلفة هي:

□ نُمطُ العَدِ التصاعدي Up-counting Mode

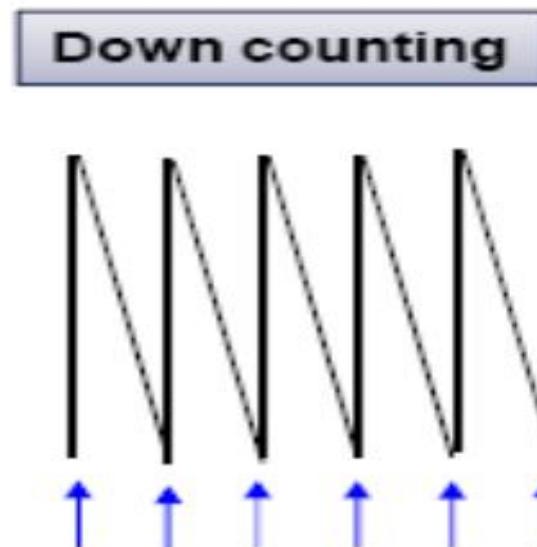
في هذا النُمط فإن العَدَاد يبدأ بالعَدِ من الصفر مع كل نبضة قادمة على قطب الدخُل ويستمر حتى يصل إلى القيمة المخزنة مسبقاً والموجودة في المسجل (TIMx_ARR)، ثم يعود للقيمة صفر ويولد حدث الطفاحان (Overflow) كلَّ مِنْتَهِيَةِ عَدِ الْعَدَادِ مع كل طفحان للعَدَاد.

Up counting



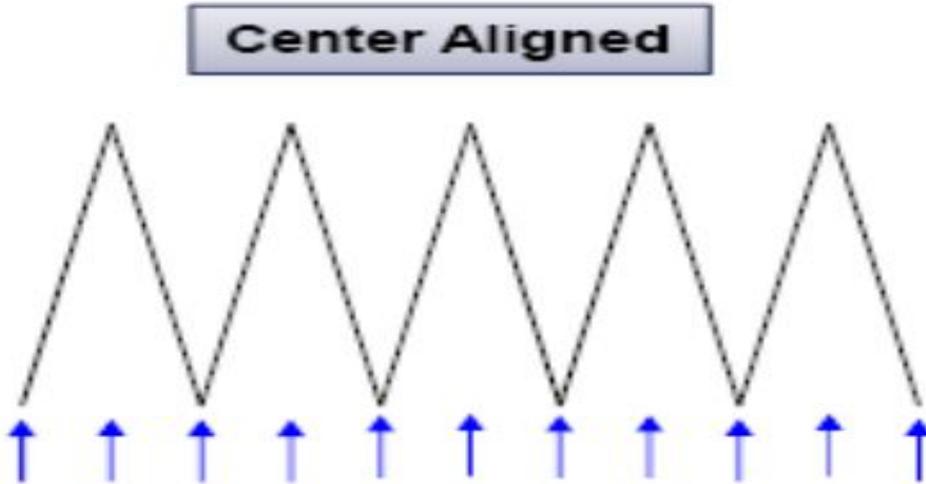
نُمطُ العَدِ التَّنَازُلِي Down-counting Mode □

في هذا النُمط فإن العَدَاد يبدأ بالعَدِ من القيمة المخزنة **auto-reload** في المسجل **TIMx-ARR** مع كل نبضة قادمة على قطب الدخول **value** ويستمر ليصل إلى الصفر ثم يعود ليبدأ من القيمة المخزنة سابقاً ويولد حدث **Underflow event** أيضاً يتم توليد حدث **Underflow** مع كل **Update**



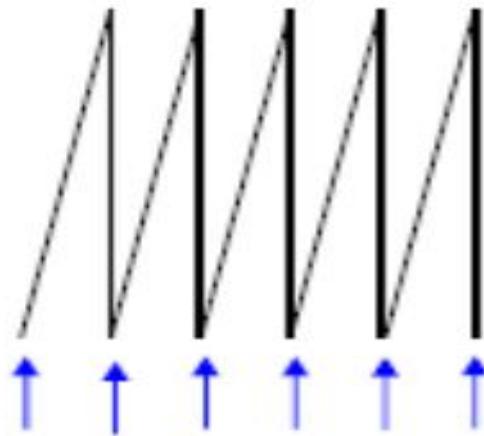
□ نُمطُ العَدَاد التصاعدي تنازلي Center-Aligned Mode:

في هذا النمط فإن العداد يبدأ بالعد التصاعدي من الصفر ويستمر بالعد مع كل نبضة قادمة على قطب الدخل حتى يصل إلى القيمة المخزنة سابقاً-
auto-reload value في المسجل TIMx-ARR ناقص واحد ، ثم يتم توليد حدث الطفhan ثم يبدأ بالعد التنازلي من القيمة المخزنة سابقاً auto-reload value مع كل نبضة قادمة على قطب الدخل وحتى يصل إلى الصفر عندها يتم توليد حدث Underflow ثم يعود للعد التصاعدي من الصفر وهكذا...،

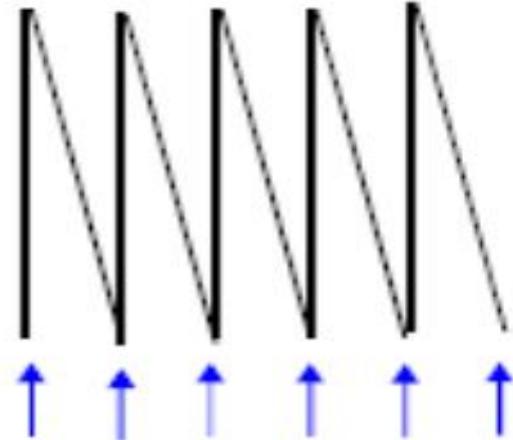


نُمْطُ الْعَدَاد Counter

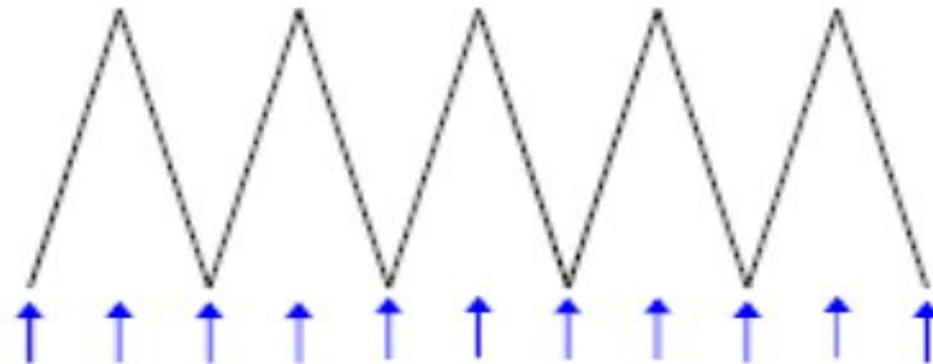
Up counting



Down counting

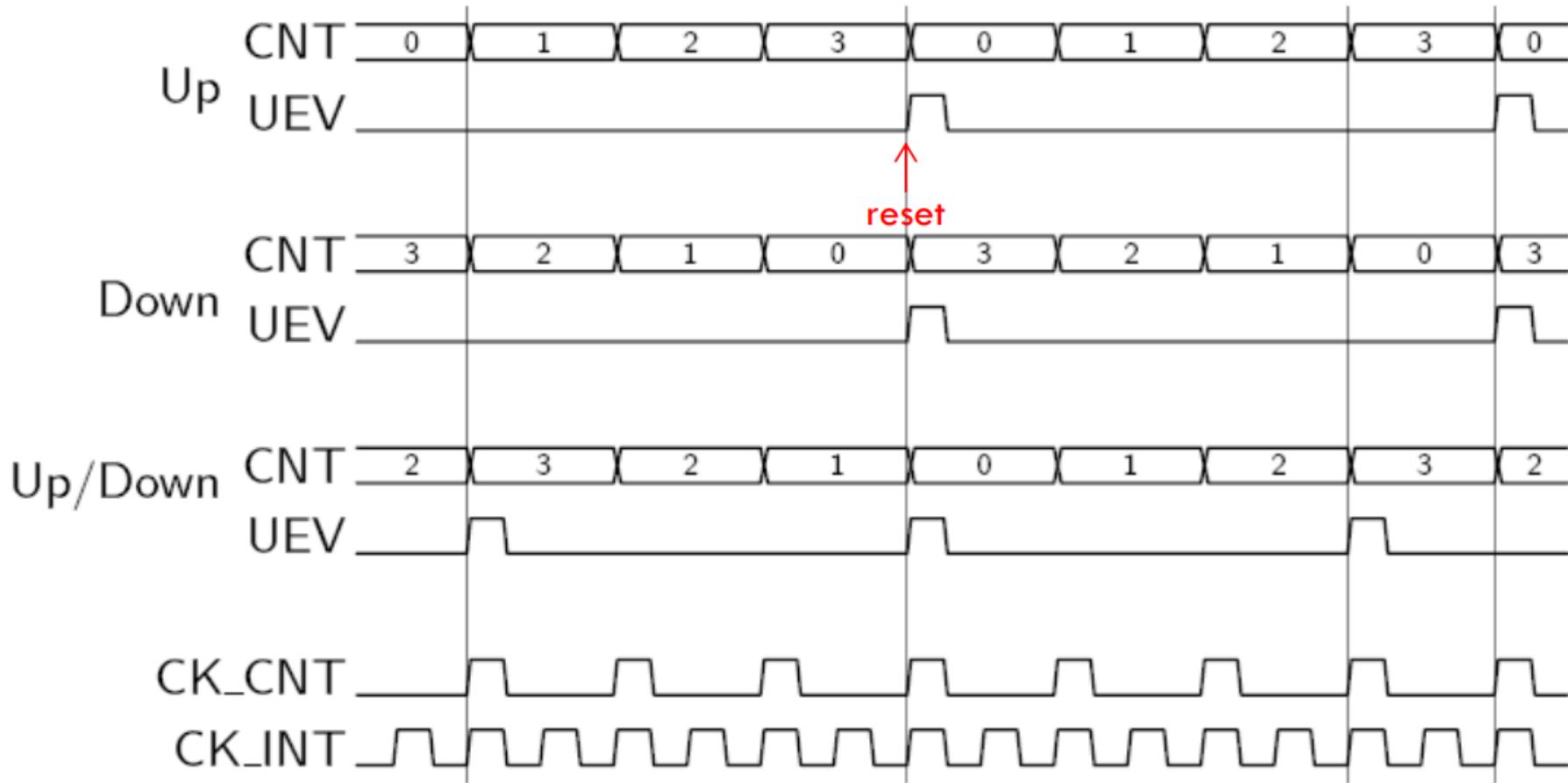


Center Aligned



نُمْطُ الْعَدَاد Counter

وَيَكُونُ الْمُخْطَطُ الزَّمْنِيُّ لِأَنْمَاطِ الْعَدِ الْثَّلَاثَ :

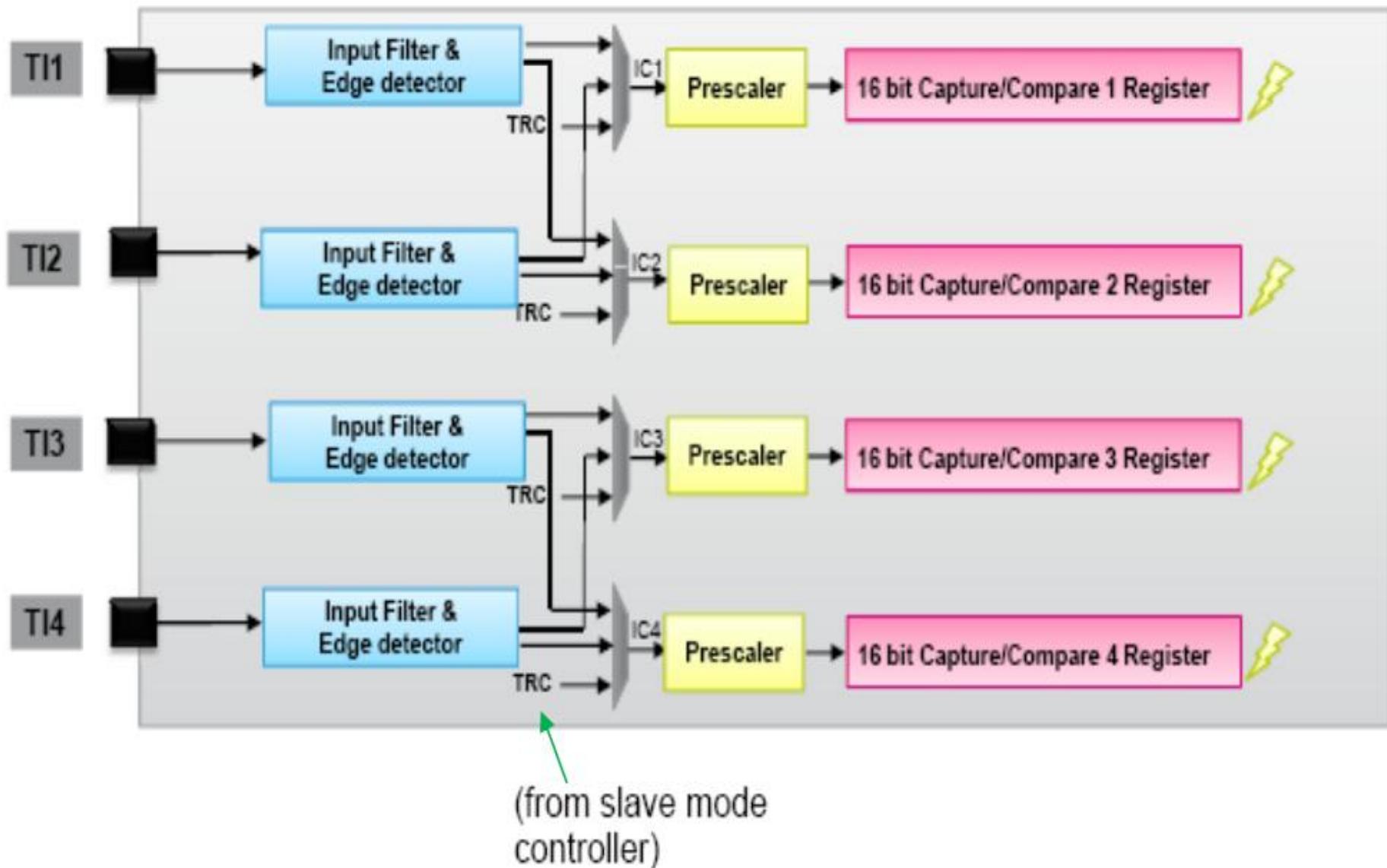


Counter Modes (ARR=3, PSC=1)

نط^ط Input Capture mode

- في نط^ط Input Capture يستقبل المؤقت نبضات الساعة الخاصة به من مصدر داخلي (ساعة المتحكم بعد استخدام المقسم التردد^y)
- يستمر بالعد إلى أن يحدث حدث معين (جبهة صاعدة/ جبهة هابطة) على قطب المتحكم الخاص بقناة الـ Input Capture عند^{ها} يتم حفظ القيمة التي وصل إليها المؤقت إلى Channel مسجل input capture register
- لكل مؤقت في متحكمات STM32 عدة قنوات (input capture/compare output) channels مرتبطة بأقطاب المتحكم يمكن معرفتها من خلال الـ datasheet الخاصة بالمتحكم

Input Capture mode نمط



نُمط PWM mode

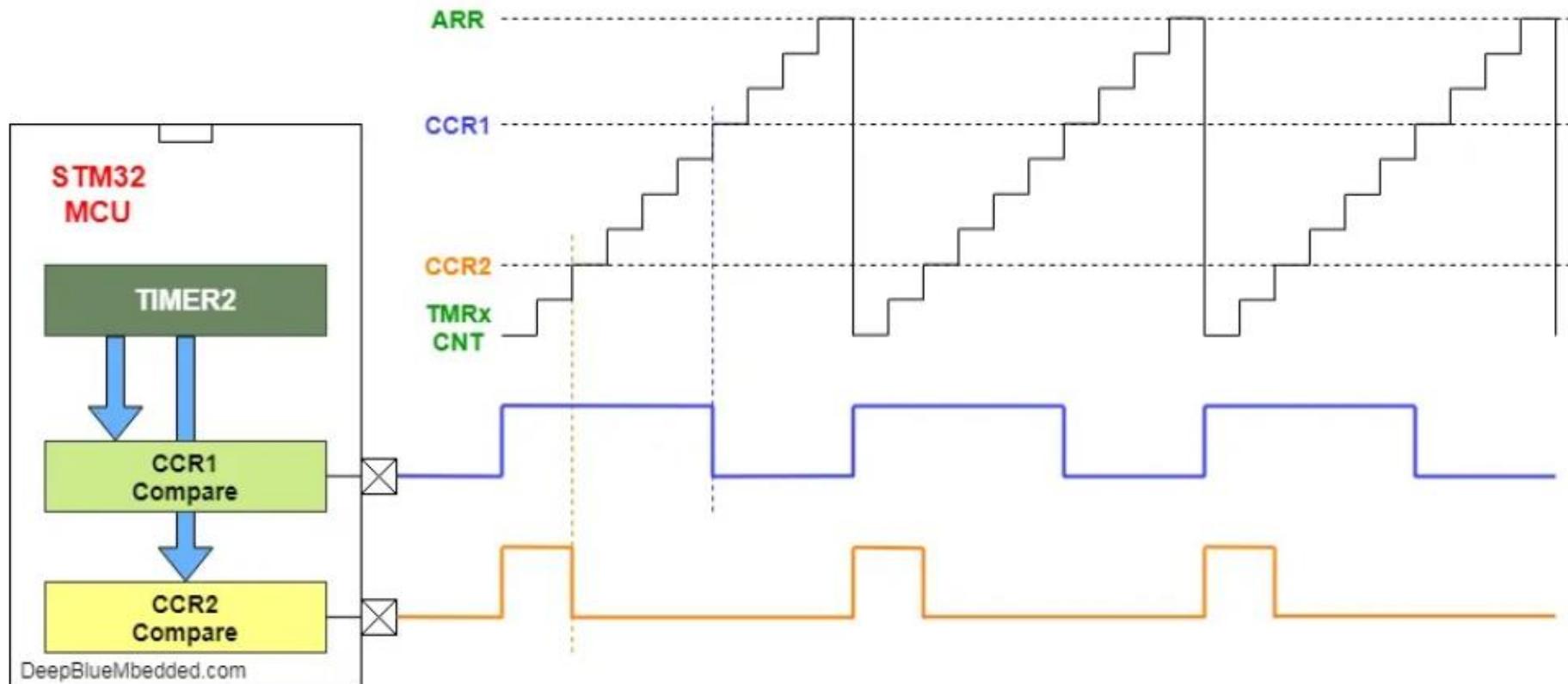
□ في نُمط PWM mode يستقبل المؤقت نبضات الساعة الخاصة به من الساعة الداخلية للمتحكم حيث يبدأ بالعد من الصفر ويزداد مع كل نبضة ساعة للمتحكم (طبعاً مع مراعاة إعدادات المقسم الترددية للمؤقت)

□ يتم وضع قطب الخرج الخاص بالـ PWM في وضع HIGH ويبقى كذلك إلى أن يصل العداد إلى القيمة المخزنة في المسجل CCRx، عدّها يصبح قطب الخرج في وضع LOW إلى أن يصل العداد إلى القيمة المخزنة في المسجل ARRx، وهذا

□ يدعى شكل الإشارة الناتجة بالـ Pulse Width Modulation (Modulation)، حيث يتم التحكم بالتردد من خلال تردد الساعة الداخلية للنظام، والمقسم الترددية Prescaler بالإضافة إلى قيمة المسجل (Auto Reload register) ARRx، كما يتم تحديد قيمة دورة التشغيل الـ duty cycle من خلال قيمة المسجل الـ CCR1

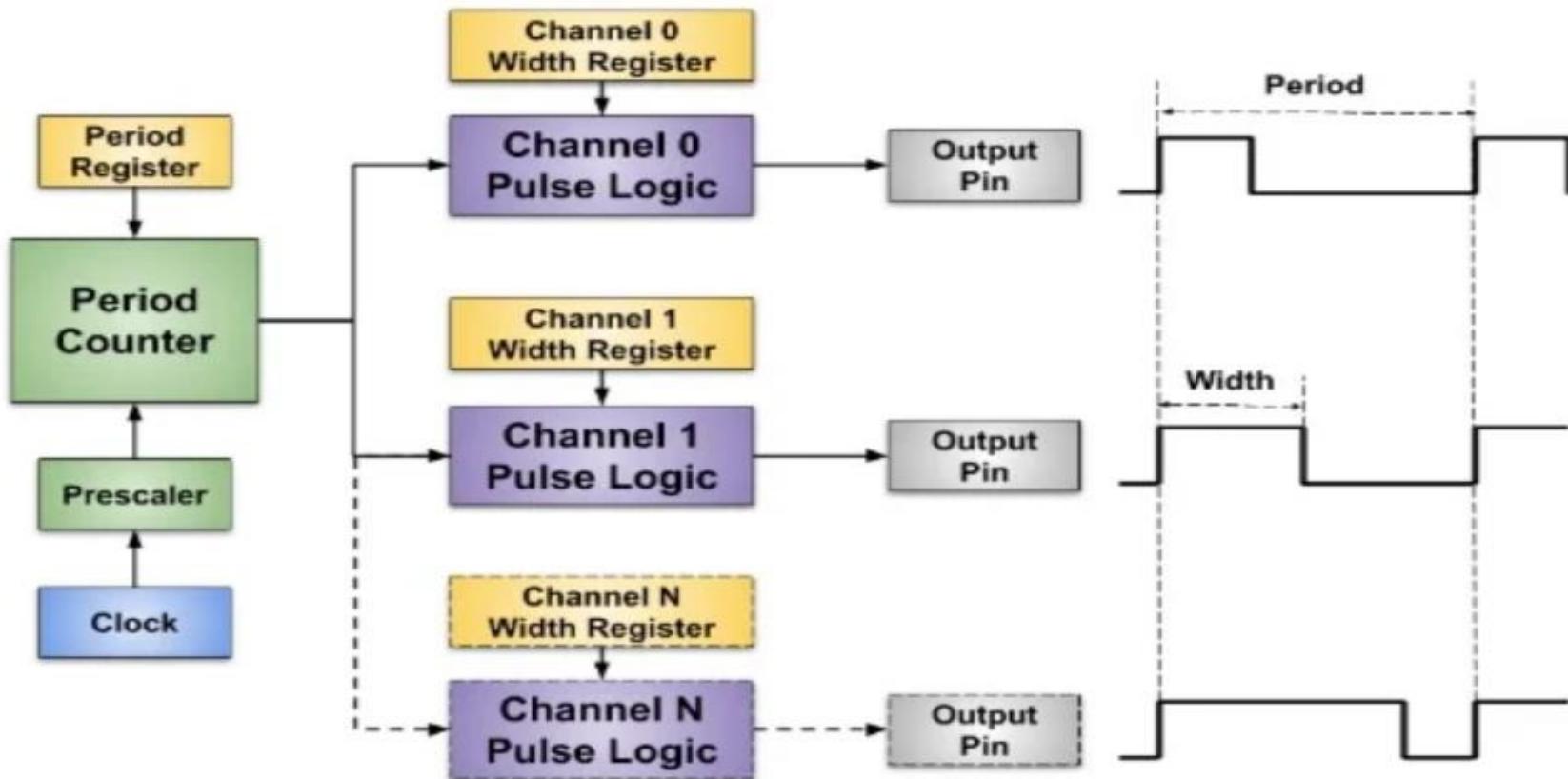
نُمط PWM mode

□ يوضح المخطط التالي كيفية تأثير قيمة المسجل ARR في دور (تردد) إشارة الـ PWM، وكيف تؤثر قيمة المسجل CCR1 في قيمة دورة التشغيل duty cycle



نُمَطْ PWM mode

لكل مؤقت من مؤقتات المتحكم STM32 عدة قنوات ، لذا فإن كل مؤقت بإمكانه توليد عدة إشارات PWM لكل منها دورة تشغيل مختلفة ولكن لها نفس التردد وتعمل بالتزامن مع بعضها



نُمط PWM mode

تردد إشارة الـ :PWM

من PWM (1/FPWM) خلال

يتم التحكم بدور إشارة الـ البارامترات التالية:

قيمة المسجل ARR

قيمة المقسم الترددی Prescaler

تردد الساعة الداخلية internal clock

عدد مرات التكرار

وذلك من خلال العلاقة التالية:

$$F_{PWM} = \frac{F_{CLK}}{(ARR + 1) \times (PSC + 1) \times (RCR + 1)}$$

نُمَط PWM mode

مثال:

• ARR=65535 • Prescaler=1 • MHZ F_{CLK} 72= 72
PWM:، احسب تردد نبضات الـ RCR =0

$$F_{PWM} = \frac{72 \times (10^6)}{(65535 + 1) \times (1 + 1) \times 1} = 549.3 HZ$$

Duty Cycle: دورة التشغيل

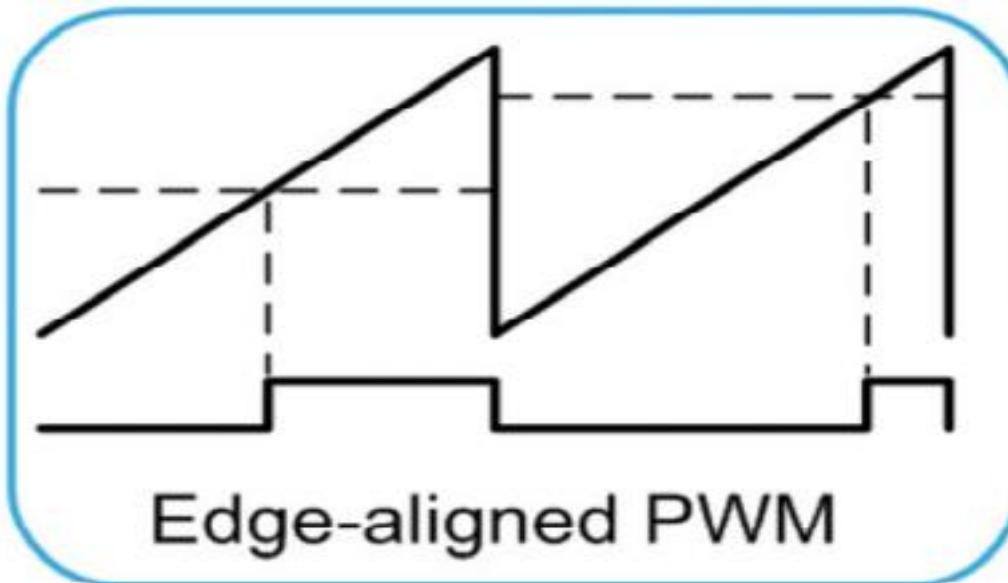
عند عمل المؤقت بنمط PWM وتوليد النبضات في وضع الـ edge-aligned mode up-counting، فإن دورة التشغيل يتم حسابها من خلال العلاقة التالية:

$$DutyCycle_{PWM}[\%] = \frac{CCRx}{ARRx} [\%]$$

نُمَط PWM mode

أنماط الـ PWM المختلفة:

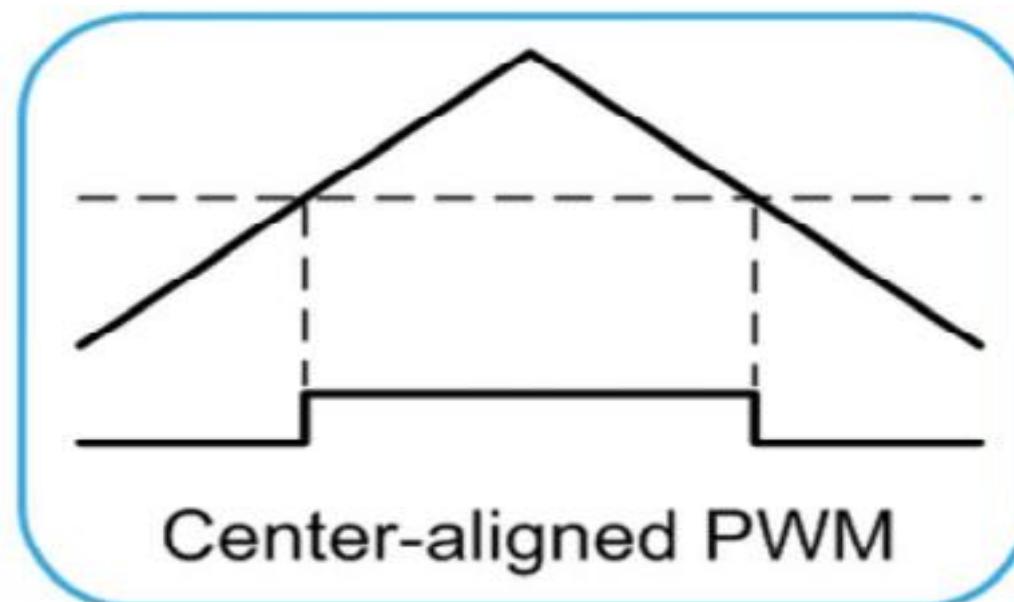
□ **Edge-aligned mode**: في هذا النمط فإن العداد يقوم بالعد بشكل تصاعدي فقط أو تنازلي فقط، وبإمكان المؤقت الواحد أن يولد حتى الـ 6 إشارات PWM لها نفس التردد ولكن بدورات تشغيل مختلفة، وهذه الإشارات جميعها متزامنة باعتبار أن الجبهة الهاابطة هي نفسها لجميع الإشارات.



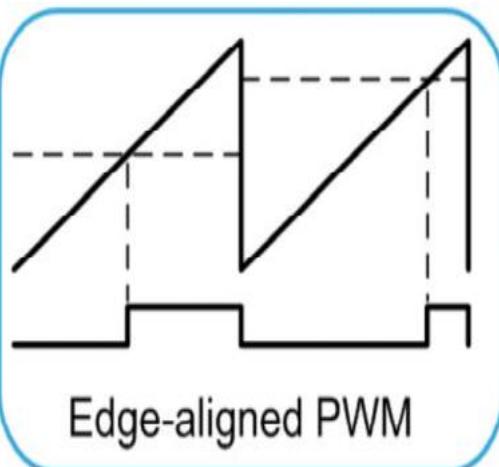
نُمط PWM mode

أنماط الـ PWM المختلفة:

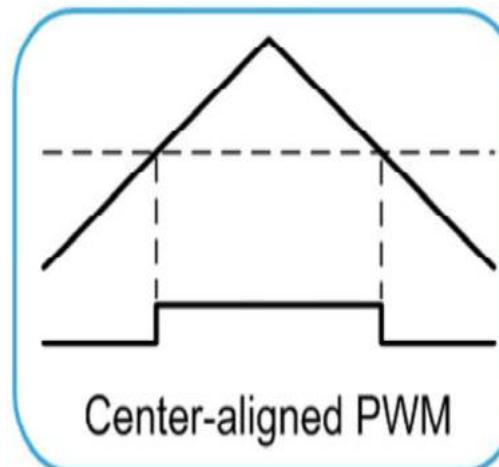
□ **Center-aligned mode** في هذا النمط فإن العداد يقوم بالعد بشكل تصاعدي/تنازلي، وتكون إشارات الـ PWM الناتجة من مؤقت واحد غير متزامنة لأن الجبهة الهاابطة لكل منها مختلفة، لذا فإن أزمنة التبديل لكل إشارة PWM تكون مختلفة عن الإشارة الأخرى



PWM mode نمط

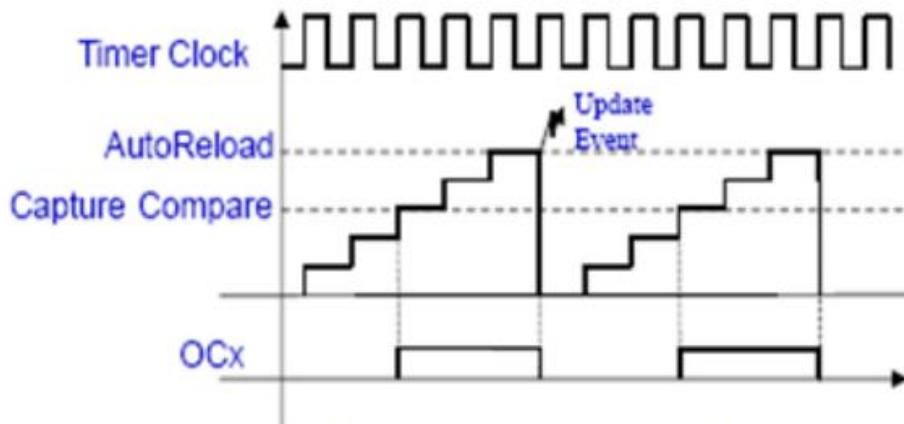


Edge-aligned PWM



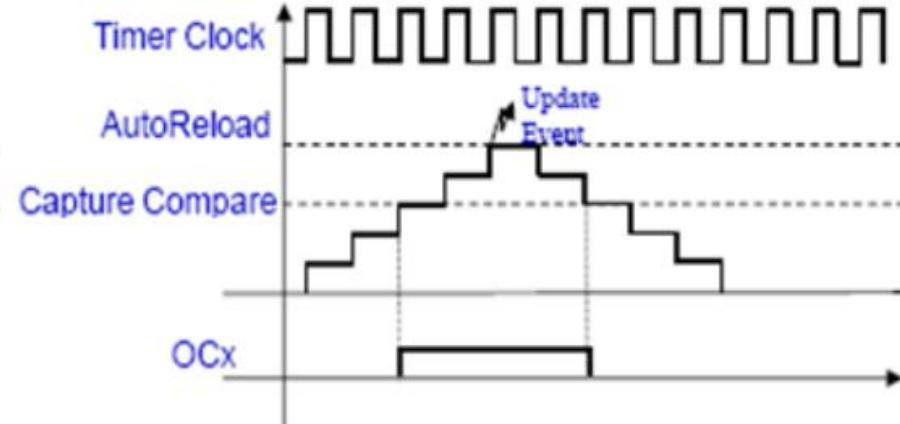
Center-aligned PWM

Edge-aligned Mode



PWM mode 2

Center-aligned Mode



هناك ثلات أوضاع مختلفة لاستخدام المؤقتات هي:

وضع Polling : أي استخدام المؤقت بدون مقاطعة وفي هذه الحالة يجب فحص القيمة التي وصل إليها العداد بشكل يدوي داخل الكود بشكل مستمر أو يمكن بدلاً من ذلك فحص حالة العلم Flag أيضاً بشكل مستمر داخل الكود مما يؤدي إلى تعطيل العديد من وظائف المتحكم أو قد تسبب في عدم الوصول إلى القيمة المحددة بالضبط، لذا فإننا لن نستخدم هذا الوضع ضمن تطبيقاتنا.

توفر مكتبة HAL الدالة التالية لبدء المؤقت:

HAL_TIM_Base_Start();

أوضاع الاستخدام المختلفة للمؤقتات في متحكمات STM32

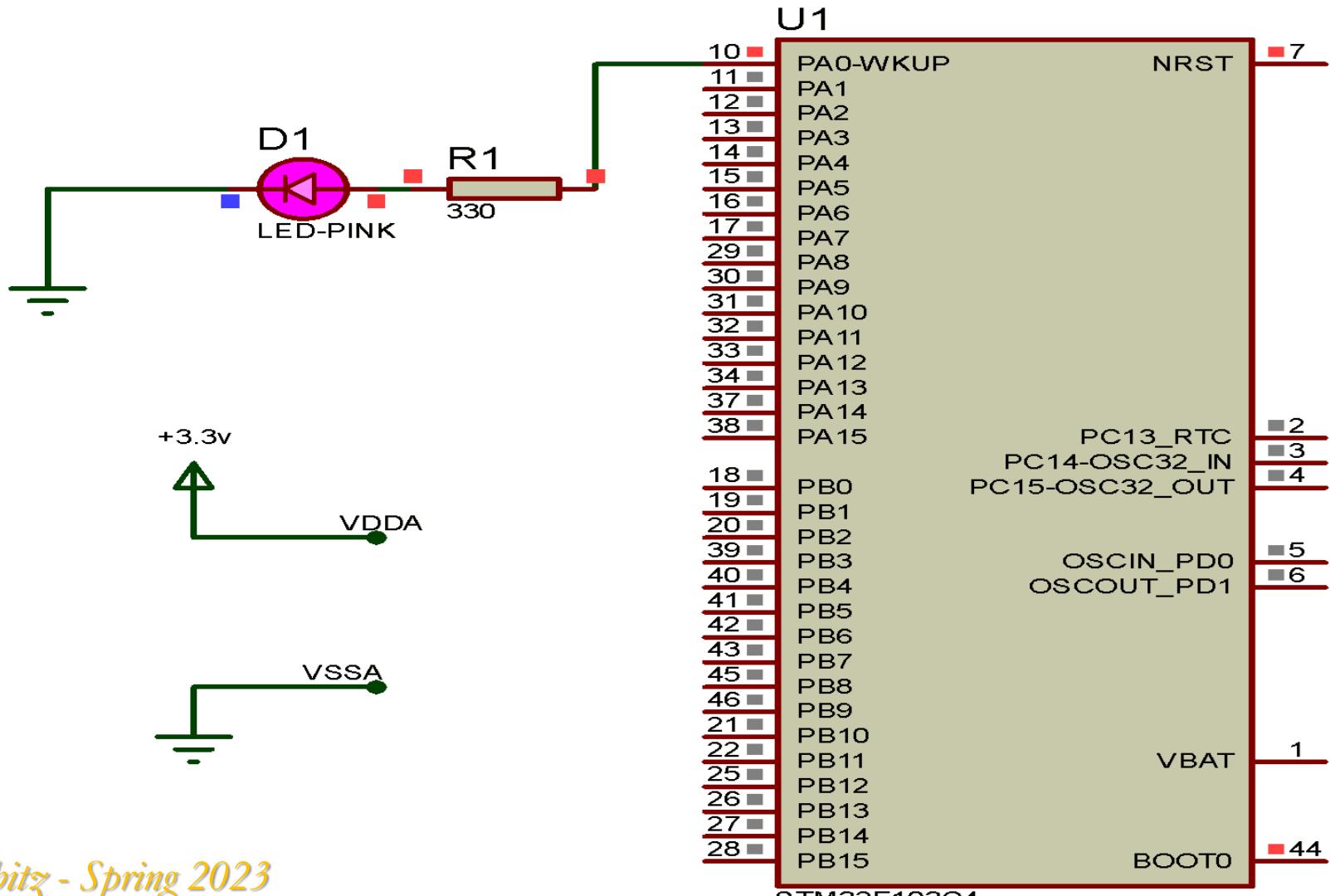
وضع Interrupt  : في هذا الوضع عند الوصول إلى Overflow/underflow أو أي من أحداث المقاطة سيتم التوجّه آلياً لتنفيذ برنامج خدمة المقاطة، وهذا الوضع الذي سستخدمه في جميع التطبيقات القادمة.

توفر مكتبة HAL الدالة التالية لبدء المؤقت في وضع المقاطة:

`HAL_TIM_Base_Start();`

وضع DMA 

التطبيق العملي الأول : استخدام المؤقت في نمط Timer mode وبوضع المقاطعة لتوليد زمن بدلاً من استخدام دالة delay() واستخدامه في عملToggle للپيد الموصول على القطب PA5



التطبيق العملي : استخدام المؤقت في نمط Timer mode وبوضع المقاطعة لتوليد زمن بدلًا من استخدام دالة delay() واستخدامه في عمل Toggle لـ ليد الموصل على القطب PA5

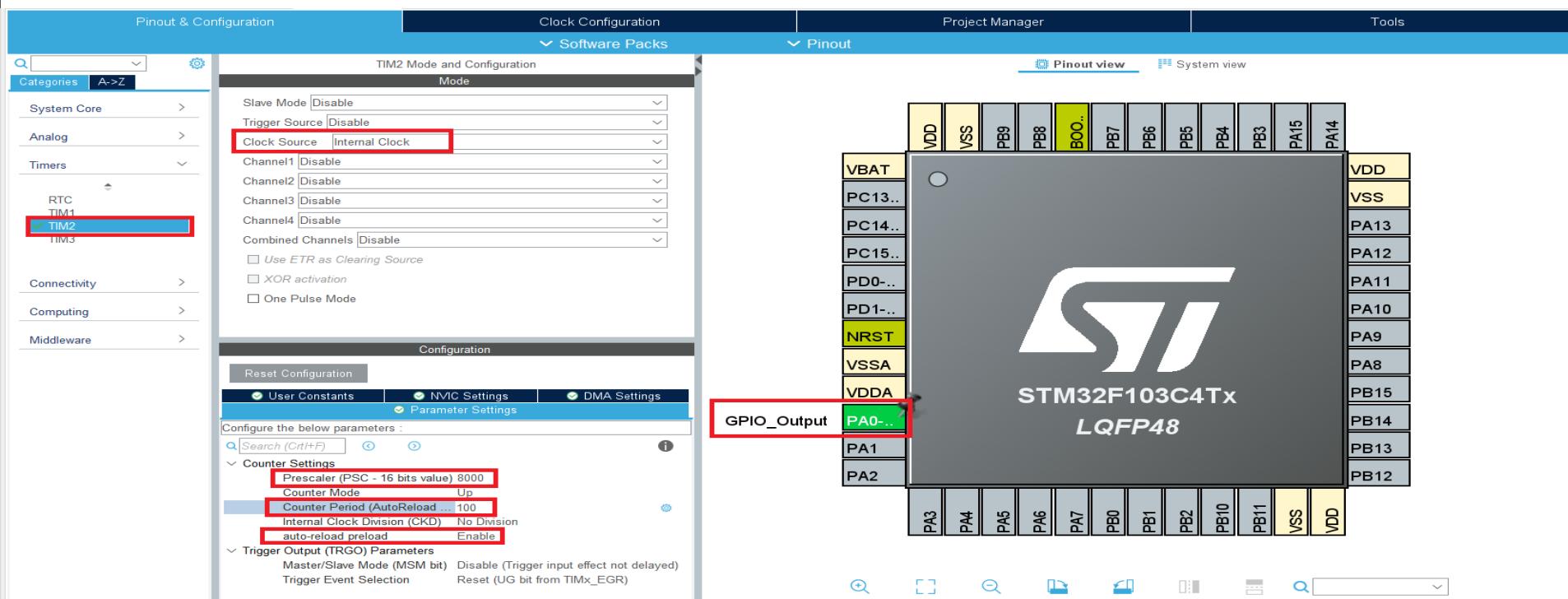
- ختار القطب PA5 لضبطه كقطب خرج
- نقوم بضبط إعدادات المؤقت كي نحصل على زمن 100 msec
لعكس حالة الـ لـيد الموصل على القطب رقم 5 من المنفذ A، من
المعادلة السابقة سنفترض أن تردد ساعة المتحكم هي 8 MHz
والمقسم التردد 8000 بقى فقط حساب (Preload) Period
بتغيير القيم في المعادلة :

$$T_{out} = \frac{Prescaler \times Preload}{F_{CLK}} = \frac{8000 \times Preload}{8000000}$$

$$Preload = 100$$

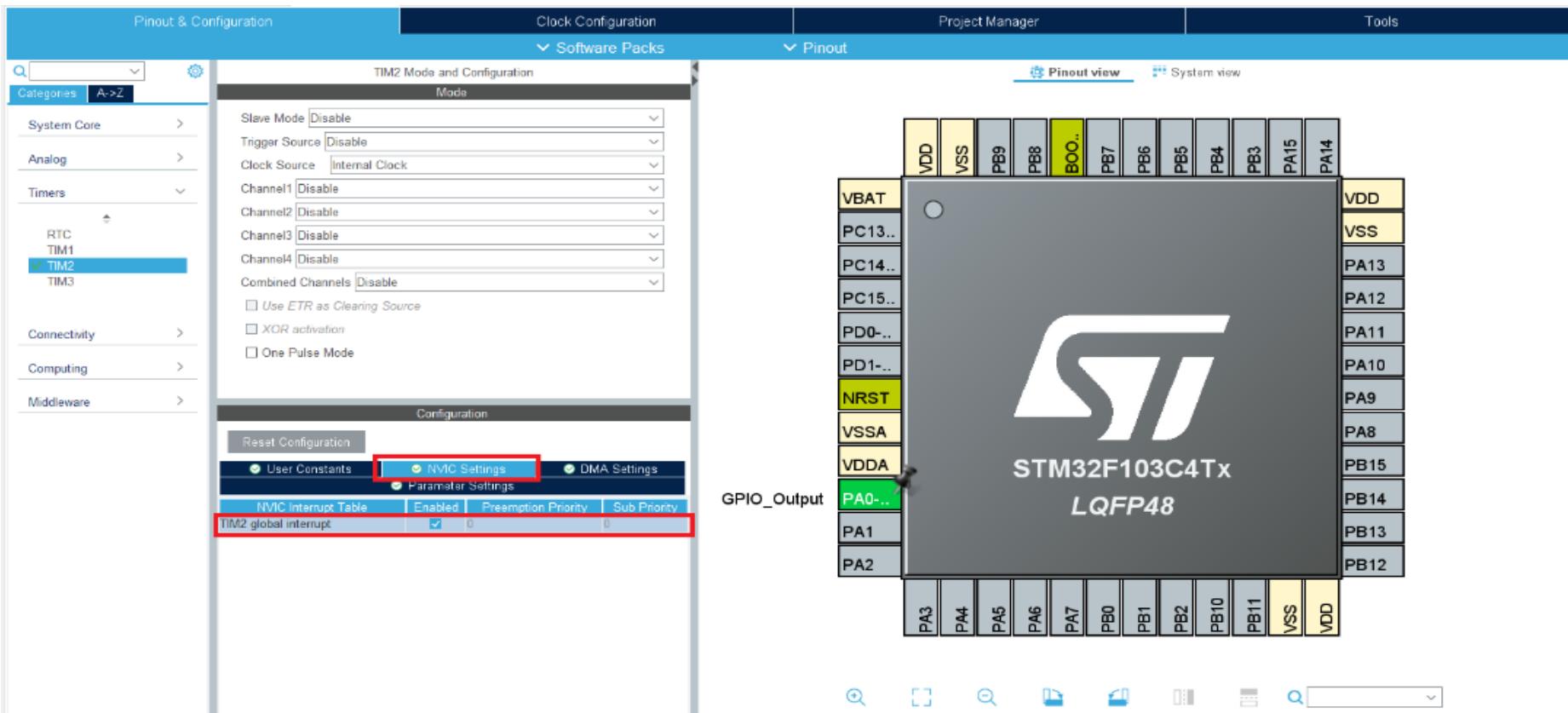
التطبيق العملي : استخدام المؤقت في نمط Timer mode وبوضع المقاطعة لتوليد زمن بدلاً من استخدام دالة delay() واستخدامه في عمل Toggle لليد الموصول على القطب PA5

سنقوم باختيار مصدر الساعة للمؤقت داخلي، المقسم الترددية 8000 ، الـ $\text{Preload} = 100$ ، أيضاً سنقوم بتفعيل إعادة التحميل التلقائي، كما في الشكل التالي:

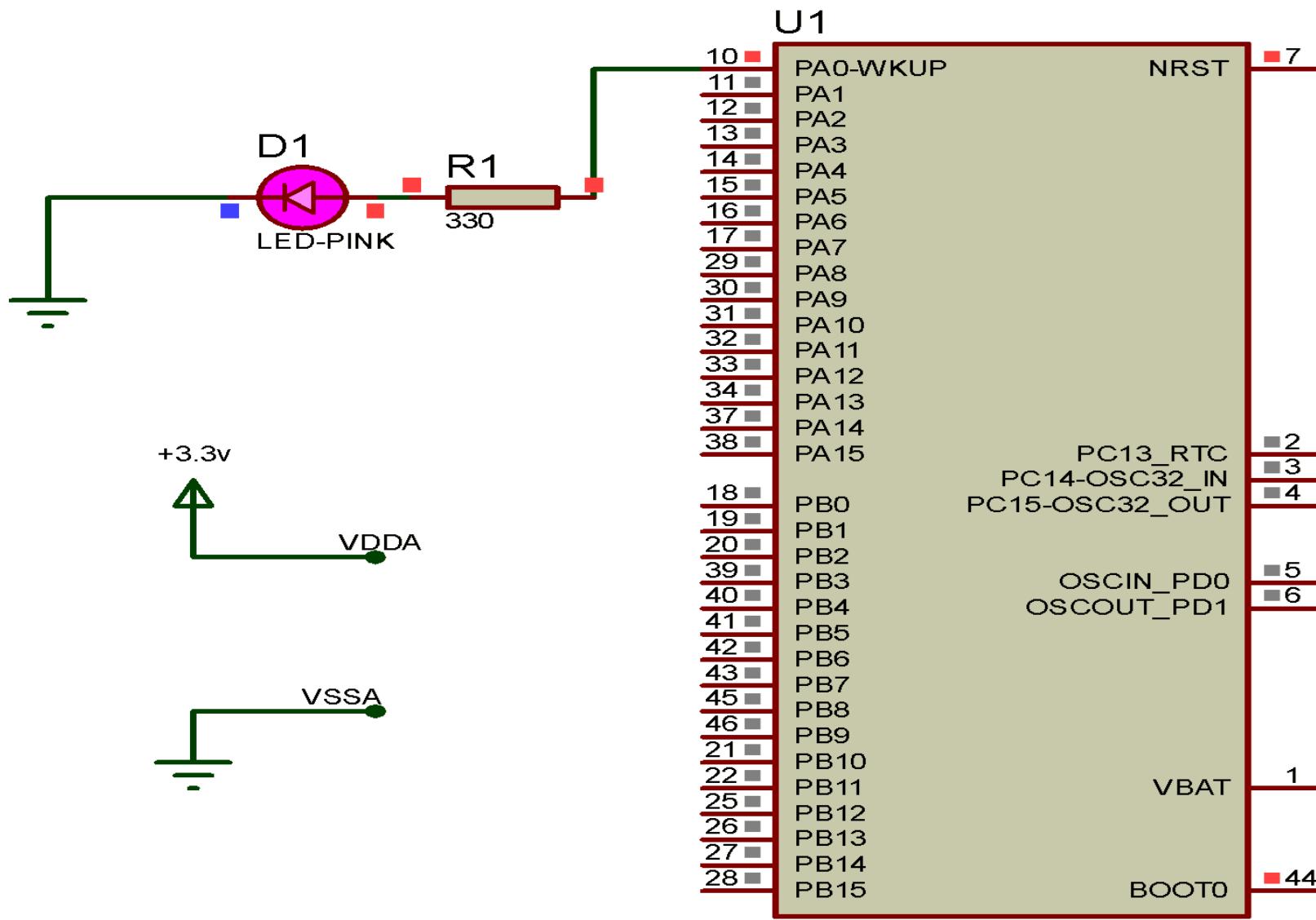


التطبيق العملي : استخدام المؤقت في نمط Timer mode وبوضع المقاطةة لتوليد زمن بدلاً من استخدام دالة delay() واستخدامه في عمل Toggle لـ ليد الموصل على القطب PA5

نقوم بتفعيل مقاطعة المؤقت من شريط الـ NVIC tab



التطبيق العملي الثاني : استخدام المؤقت في نمط PWM mode للتحكم في شدة إضاءة اليد



التطبيق العملي الثاني : استخدام المؤقت في نمط PWM mode للتحكم في شدة إضاءة اليد

سنتبع في هذا التطبيق الخطوات التالية للتحكم بشدة إضاءة اليد:

- ضبط باراترات المؤقت TIM2 ليعمل في نمط الـ PWM وباستخدام الساعة الداخلية للمتحكم internal clock، ثم تفعيل القناة الأولى CH1 لاستخدامها كقناة الخرج لإشارة الـ PWM
- ضبط قيمة المسجل ARR على القيمة العظمى وهي 65535 ، والقسم الترددی prescaler على 1، فيصبح التردد 61HZ خلال العلاقة:

$$F_{\text{PWM}} = (8 \times (10^6)) / ((65535 + 1) \times (1 + 1) \times 1) = 61 \text{HZ}$$

- التحكم بدورة التشغيل duty cycle من خلال كتابة القيمة المناسبة على المسجل CCMR1

جعل دورة التشغيل تتغير من 0% حتى 100% وتعيد الكرة باستمرار

التطبيق العملي الثاني : استخدام المؤقت في نمط PWM mode للتحكم في شدة إضاءة اليد

□ ضبط إعدادات المؤقت ليعمل في نمط PWM يقوم بضبط مصدر الساعة للمؤقت على الساعة الداخلية للنظام internal clock، يقوم بتشغيل القناة CH1 لتكون القناة التي سيتم إخراج إشارة الـ PWM عليها، ضبط قيمة المسجل ARR على القيمة العظمى وهي 65535 ، والمقسم الترددى prescaler على 1، فيصبح التردد 61HZ، نفعل خاصية Auto Reload preload PWM ونختار نمط إشارة الـ

التطبيق العملي الثاني : استخدام المؤقت في نمط PWM mode للتحكم في شدة إضاءة الـ LED

Pinout & Configuration Clock Configuration Project Manager Tools

Categories A-Z

System Core

- Analog
- Timers
 - RTC
 - TIM1
 - TIM2**
 - TIM3
- Connectivity
- Computing
- Middleware

Clock Configuration

Software Packs

Pinout

Pinout view **System view**

VDD	VSS	PB9	PB8	BO0	PB7	PB6	PB5	PB4	PB3	PA15	PA14	VDD	VSS	PA13	PA12	PA11	PA10	PA9	PA8	PB15	PB14	PB13	PB12
VBAT																							
PC13..																							
PC14..																							
PC15..																							
PD0-..																							
PD1-..																							
NRST																							
VSSA																							
VDDA																							
TIM2_CH1	PA0...																						
PA1																							
PA2																							
PA3																							
PA4																							
PA5																							
PA6																							
PA7																							
PB0																							
PB1																							
PB2																							
PB10																							
PB11																							
VSS																							
VDD																							

Search (Ctrl+F) **+** **□** **Q** **File** **Print** **Help**

ساعة الزمن الحقيقي Real Time Clock (RTC)

- ساعة الزمن الحقيقي عبارة عن أداة لحفظ الوقت، تستخدم مع التطبيقات التي يتم تنفيذها عند أزمنة محددة، كساعة التوقيت الموجودة ضمن الغسالات الآلية ، تطبيق إعطاء الأدوية للمرضى بأزمنة محددة وغيرها...
- فهي عبارة عن عدد زمن لكنها تعطي دقة أكبر من المؤقتات الموجودة في المتحكم، فالمؤقتات مناسبة لتوليد الأزمنة المختلفة و إشارات الـ PWM على سبيل المثال..
- معظم متحكمات 8bit لا تحتوي على RTC داخلية وإنما يتم استخدام إحدى شرائح الـ RTC الخارجية كـ DS1302 أو PCF8563 بالإضافة إلى بعض العناصر الالكترونية الازمة كي تعمل بشكل أمثل كما تحتاج إلى مساحة إضافية على الدارة المطبوعة

ساعة الزمن الحقيقي Real Time Clock (RTC)

- تحتوي متحكمات stm32 على موديول RTC مدمج بداخل المتحكم وهي لا تحتاج لأية عناصر إضافية أو دارات ملائمة كي تعمل لوحة الـ RTC مصدرى ساعة هما:
 - RTC Timer/counter: ويستخدم كمصدر ساعة للـ RTCCCLK
 - APB clock: ويستخدم كمصدر ساعة للـ RTC register من أجل عمليات القراءة والكتابة على المسجلات

ساعة الزمن الحقيقي Real Time Clock (RTC)

نبضات الساعة لوحدة الـ RTC (RTCCCLK) يمكن أن تكون قادمة من:

(HSE) High Speed External clock

مقسمة على 32

(LSE) Low Speed External clock

(LSI) Low Speed Internal Clock

لكن عندما يكون المتحكم يعمل في نمط VBAT أو يكون في حالة إيقاف

تشغيل shutdown ، في هذه الحالة يجب أن يكون RTC clock هي

LSI أو LSE

ساعة الزمن الحقيقي RTC

- يتم تقسيم تردد clock RTC باستخدام مقسم تردد قابل للضبط وبطول 7bit ، ومن أجل تخفيض استهلاك الطاقة ينصح باستخدام نسبة تقسيم مرتفعة حيث القيمة الافتراضية هي 128
- ثم يتم تقسيم التردد الناتج عن المقسم باستخدام مقسم تردد آخر قابل للضبط وبطول 15bit ، ويجب أن يكون التردد الناتج عنه 1HZ كي يتم تحديث الزمن والتاريخ في كل 1sec كل BCD registers

Actual registers

Date

Day

Month

Date

Year

Time

HH

MM

SS

SSR

Synchronization

DR

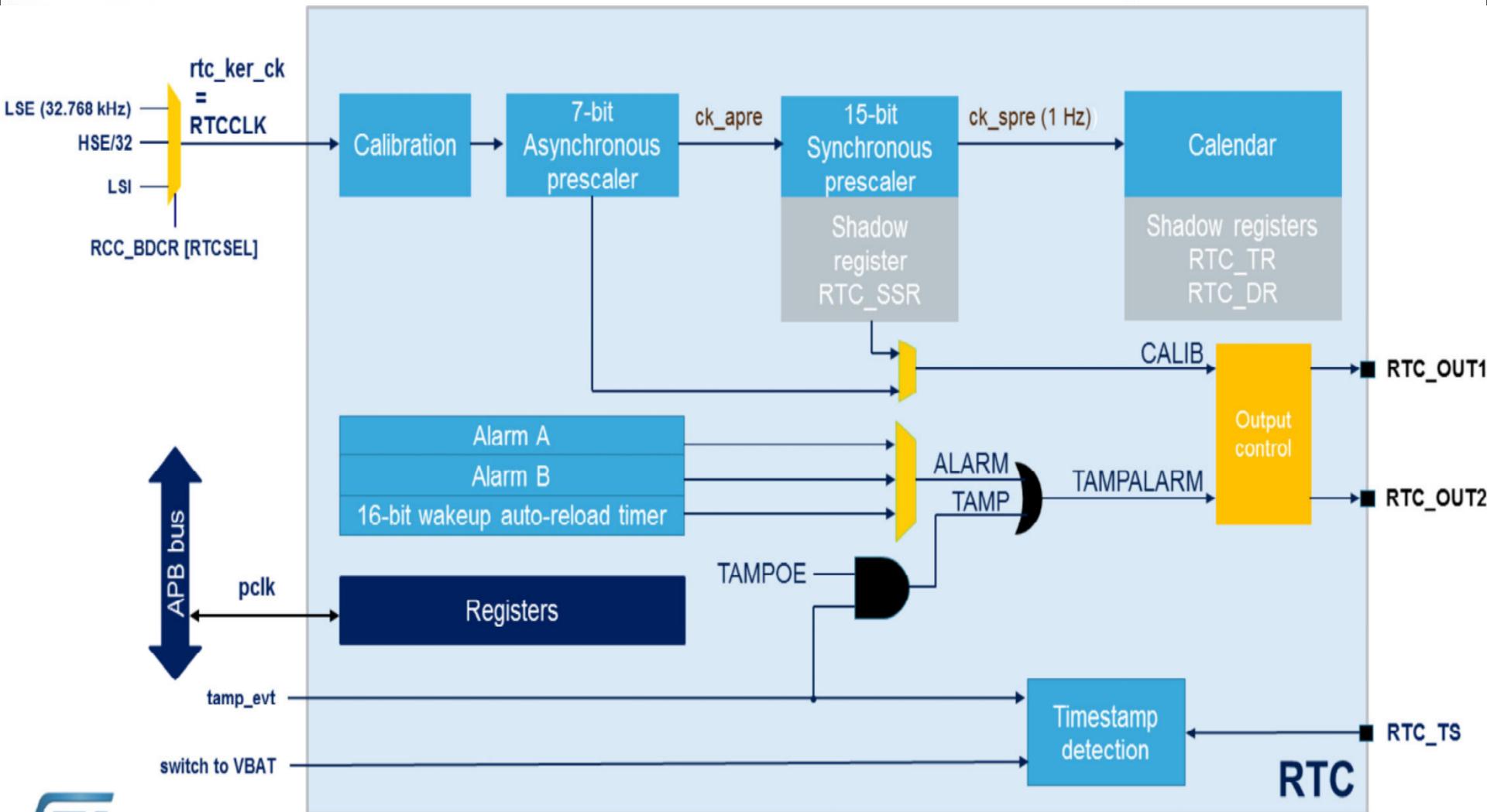
TR

SSR

Shadow registers

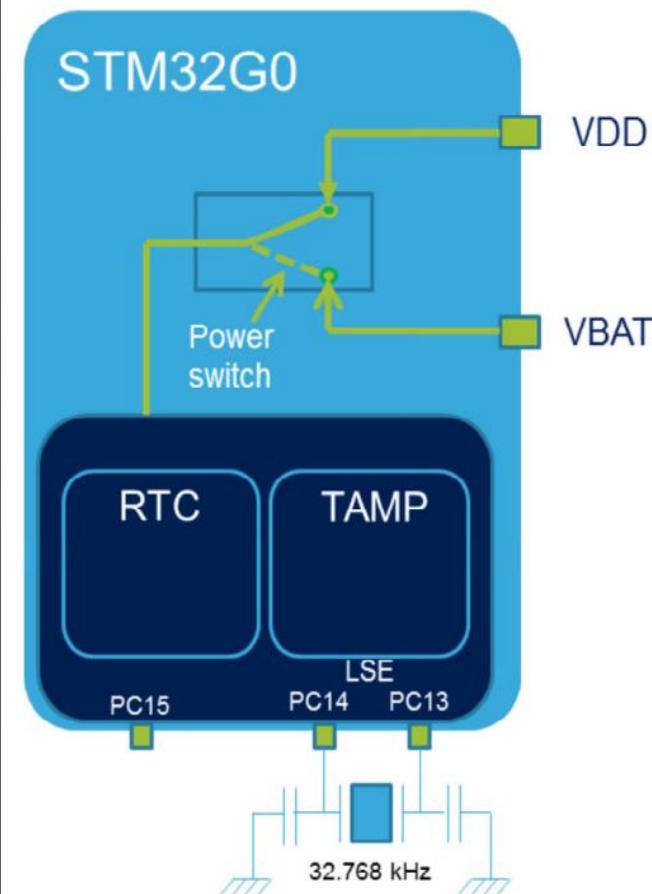
المخطط الصندوقي لساعة الزمن الحقيقي (RTC)

Real Time Clock (RTC)



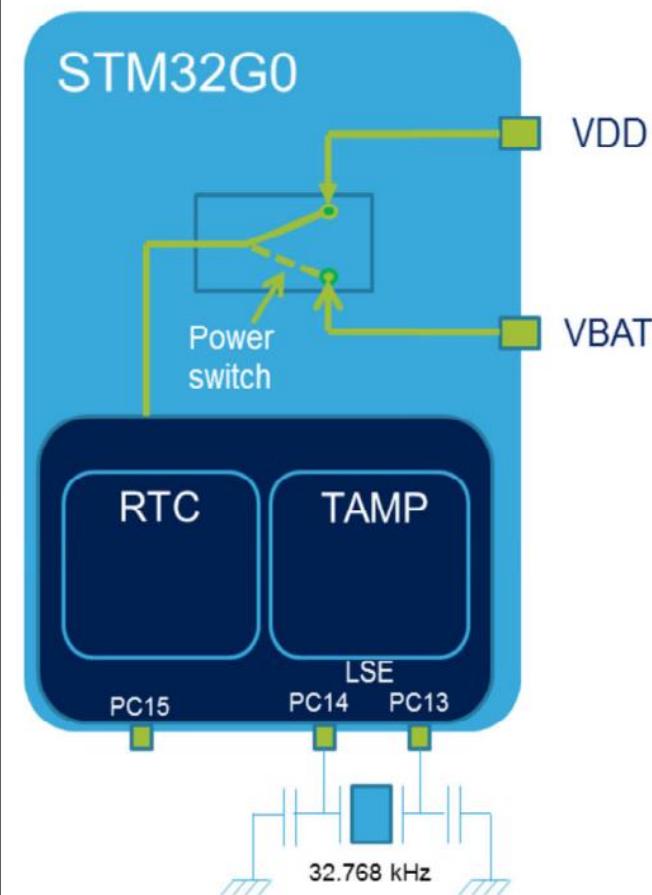
ساعة الزمن الحقيقي RTC

- يمكن لوحدة RTC أن تعمل في جميع أنماط الطاقة المنخفضة للمتحكم
- فعندما يتم تزويدها بنبضات الساعة من خلال Low speed external oscillator (LSE) بتردد 32.768KHz ، عندها مستمرة وحدة RTC بالعمل حتى عند فصل التغذية الأساسية عن المتحكم ، عندما يكون قطب VBAT موصول بطارية احتياطية



ساعة الزمن الحقيقي RTC

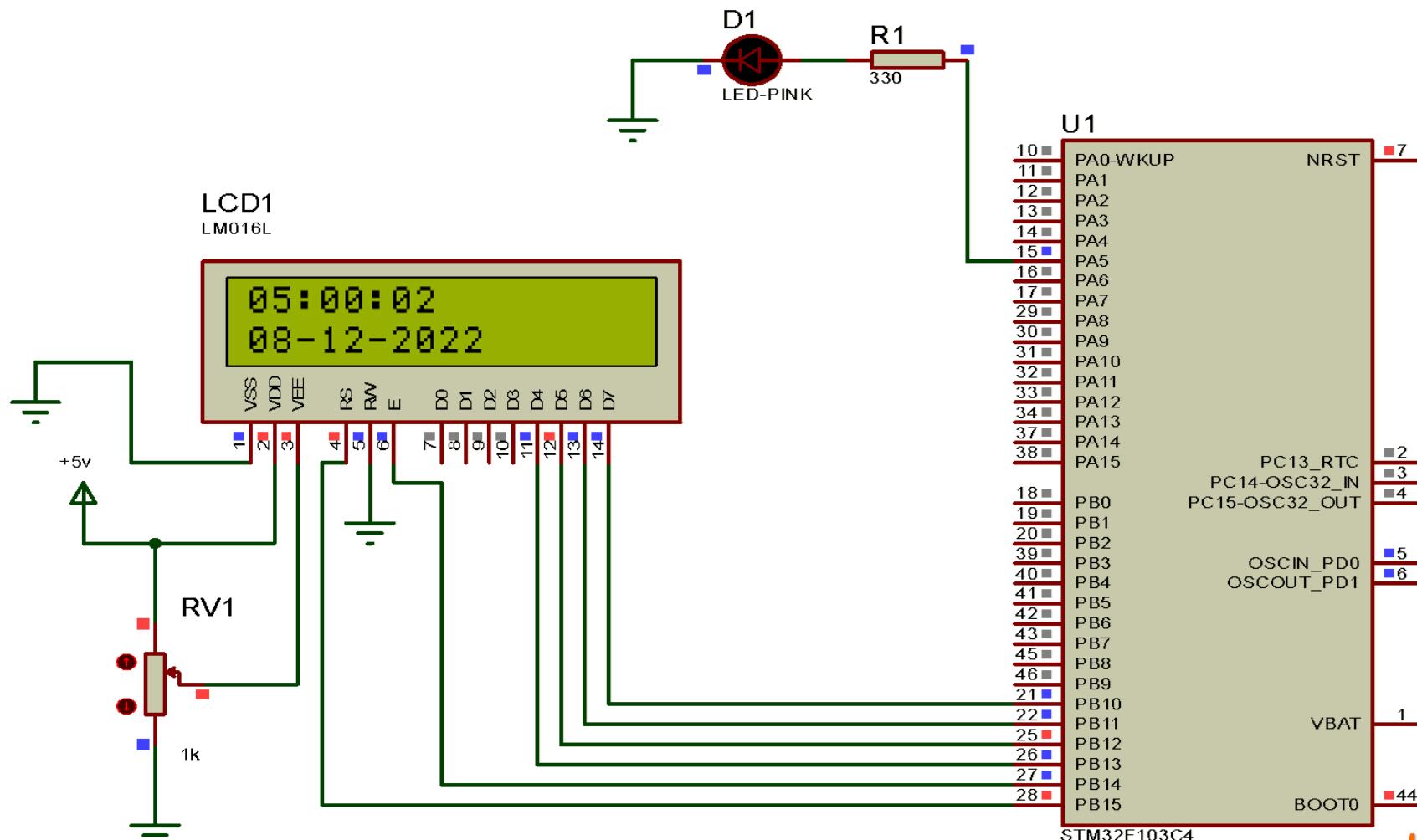
- استهلاك وحدة RTC للتيار فقط 300nA عند تغذيتها بجهد 1.8v
- وهذا التيار يتضمن استهلاك LSI للتيار
- يكون التقويم الناتج عن وحدة RTC مشفر بشفرة BCD لتقليل التعقيد في الكود البرمجي، بما في ذلك الثواني ، الدقائق ، الساعات، الأيام ، الشهور والسنين، بينما تكون أجزاء الثانية مشفرة بالشفرة الثنائية يمكن بسهولة إضافة أو إنفصال ساعة من التقويم لإدارة التوقيت الصيفي



ساعة الزمن الحقيقي Real Time Clock (RTC)

- لوحدة الـ RTC مخرجان لهما القدرة على توليد تببيهان قابلان للبرمجة ولهم القدرة على إيقاظ المعالج من كافة أنماط توفير الطاقة
- تحتوي وحدة الـ RTC على مؤقت مدمج قابل للضبط وإعادة تحميل القيمة آلياً والذي يستخدم لتوليد مقاطعات دورية لها القدرة على إيقاظ المعالج ، كما يمكن ضبط دقة هذا المؤقت

التطبيق العملي الثالث : استخدام وحدة الـ RTC لإظهار التاريخ و الوقت على شاشة lcd او ضبط المنبه على وقت محدد لتشغيل ليد



ضبط إعدادات وحدة RTC وتفعيل المقاطعة الخاصة بها

حيث سنقوم بضبط التاريخ والساعة والتزبيه من خلال الكود



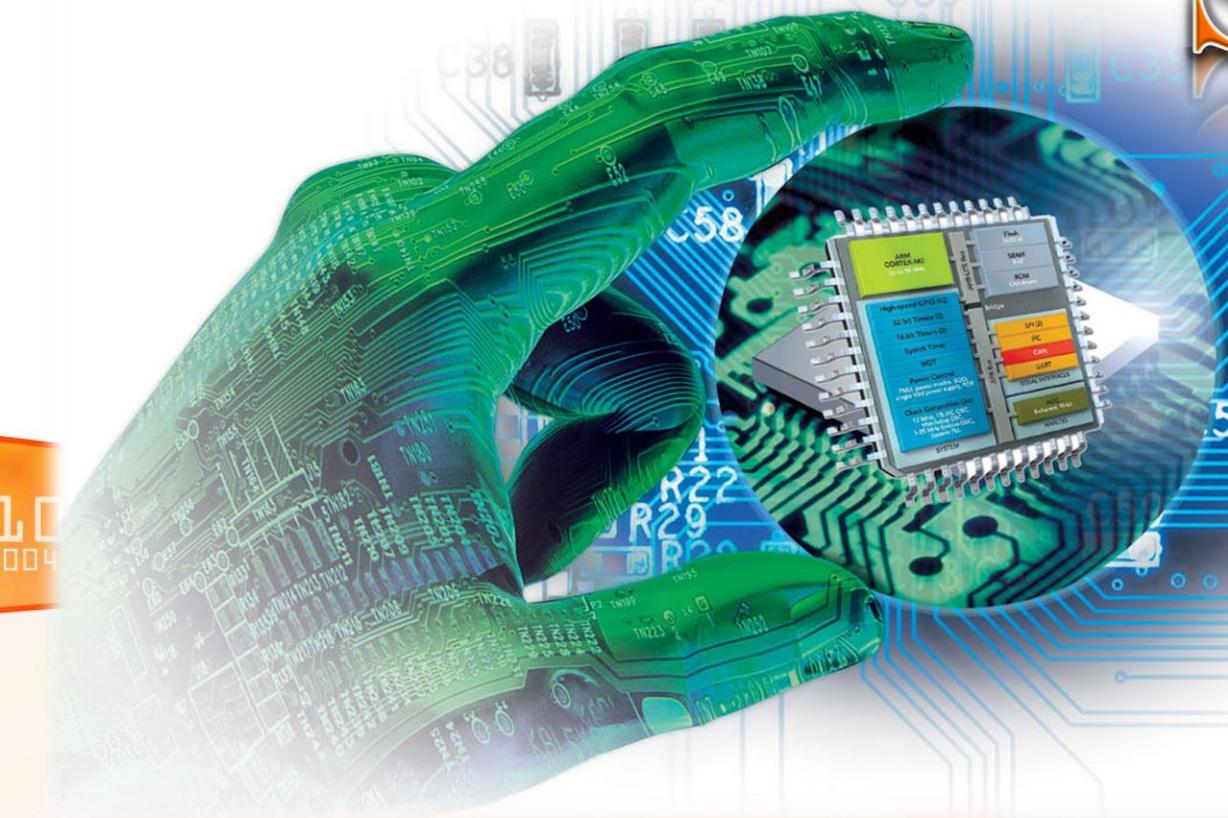
The screenshot shows the STM32CubeMX software interface. On the left, the project navigation bar includes files like *rtc_simulation.ioc, main.c, lcd_txt.h, and stm32f1xx_hal_rtc.c. The main window has tabs for Pinout & Configuration, Clock Configuration, Project Manager, and Tools. The Pinout & Configuration tab is active, displaying the STM32F103C4Tx LQFP48 pinout. The pinout diagram shows pins PA14.., PC15.., PD0-.., PD1-.., NRST, VSSA, VDDA, PA0-.., PA1, PA2, PA3, PA4, PA5, PA6, PA7, PB0, PB1, PB2, PB10, PB11, VSS, and VDD. Several pins are labeled as GPIO_Output. In the configuration area, under the RTC Mode and Configuration tab, the 'Activate Clock Source' and 'Activate Calendar' checkboxes are checked. Under the RTC tab in the configuration, the 'RTC global interrupt' and 'RTC alarm interrupt through EXTI line 17' entries in the NVIC Interrupt Table are selected. The right side of the interface shows the STM32 logo and the part number STM32F103C4Tx LQFP48.

Thank you for listening

مُتَحَكِّمَات

STM32

5



موضع عاًن المعاشرة:

- مقارنة بين بروتوكولات الاتصال التسلسلي الشائعة
- مفهوم USART و UART
- أنماط العمل المختلفة لـ USART في متحكمات STM32
- دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في نمط الـ Polling
- تطبيق عملي لاستخدام المنفذ التسلسلي USART من خلال نمط الـ polling
- إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt
- دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في نمط الـ interrupt
- تطبيق عملي لاستخدام المنفذ التسلسلي USART من خلال نمط الـ interrupt

مقارنة بين بروتوكولات الاتصال الشائعة:

	RS232	RS485	I2C	SPI	M-wire	1-wire	USB	CAN
Sync/Async	Async	Async.	Sync.	Sync.	Sync.	Async.	Async.	Async.
Type	peer-peer	master/slaves	multi-master	multi-master	master/slaves	master/slaves	host/device	multi-master
Duplex	full	half	half	full	full	half	half	half
Signaling	single-ended	Differential	single-ended	single-ended	single-ended	single-ended	Differential	Differential
Max Devices No.	2	32, 128, 256	40 (cap=400pf)	8 (cap, circuit)	8 (cap, circuit)	20 (cap, power)	127 per controller	2048
Data Rate	Up to 115Kbps	Up to 35Mbps	Std.: 100kbps Fast: 400kbps Hi: 3.4Mbps	Up to 10Mbps	Up to 1Mbps	Std.: 16.3Kbps Overdrive: 142kbps	Low: 1.5Mbps Full: 12Mbps Hi : 480Mbps	Up to 1Mbps
Max. Length	15m	1200m (at 100kbps)	6m	3m	3m	300m	5m	1000m (at 62kbps)
Pin Count	2* (Tx, Rx)	2 (A, B)	2 (SDA, SCL)	3 + SS* (SI, SO, SCK)	3 + SS* (DI, DO, SK)	1 (IO)	2 (A+, A-)	2 (CAN_H, _L)
Interfacing	HW	HW	SW HW	HW SW	HW SW	HW & SW	protocol stack	HW & SW
Flow Control	HW or SW handshake	HW or SW handshake	Acknowledge from slave	None	None	CRC, Pulling	Polling by controller	CSMA / CDAMP

مقارنة بين بروتوكول الاتصال التسلسلي والنفري:

المُرسِل

المُستَقِبِل



في الاتصال التفرعي يمكن إرسال عدة برات رقمية بنفس اللحظة الزمنية، ما يمنح سرعة كبيرة في نقل البيانات. بنفس الوقت، تتطلب عملية الاتصال التفرعي ضمان توافق ساعة المُرسِل والمُستَقِبِل والتأكد من عدم تشويش نواقل الاتصال على بعضها البعض.

المُرسِل

البت الأَقْلَ أَهْمِيَّةْ

MSB

المُستَقِبِل

LSB

0 1 1 0 0 0 1 1
D0 D1 D2 D3 D4 D5 D6 D7

في الاتصال التسلسلي يتم إرسال البوتات الرقمية الواحد تلو الآخر مع كل نبضة ساعة، ما يعني معدل أبطأ لنقل البيانات، ولكنها أقل تطلبًا فيما يتعلق بضرورات توافق الساعة وتشويش نواقل الاتصال.

النافذة التسلسليّة USART/UART

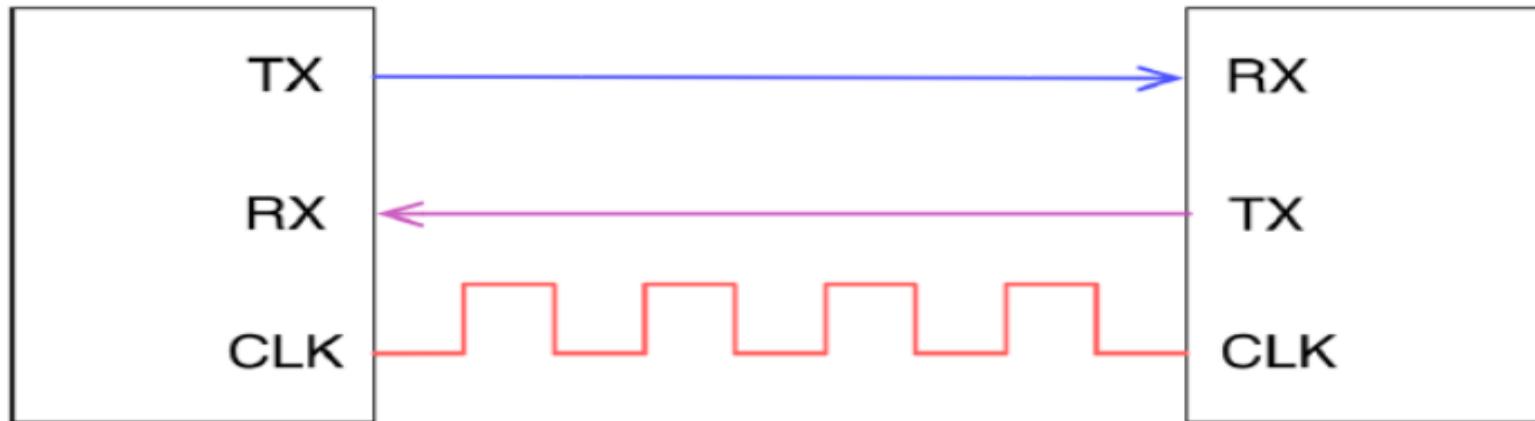
كل متحكم STM32 يحتوي على الأقل على وحدة طرفية UART واحدة، وأغلب متحكمات STM32 توفر على الأقل اثنتين من آخرى توفر لحد 8 وحدات طرفية USART/USART

USART

Synchronous: هو الإرسال والاستقبال المتزامن المبني على وجود clock بين المرسل والمستقبل.

Asynchronous: لا يعتمد على clock وإنما يتم الاكتفاء بإرسال البيانات على خط الإرسال ويتم استقبالها على خط الاستقبال.

النافذة التسلسليّة USART/UART



Device A

USART

Device B



Device A

UART

Device B

200

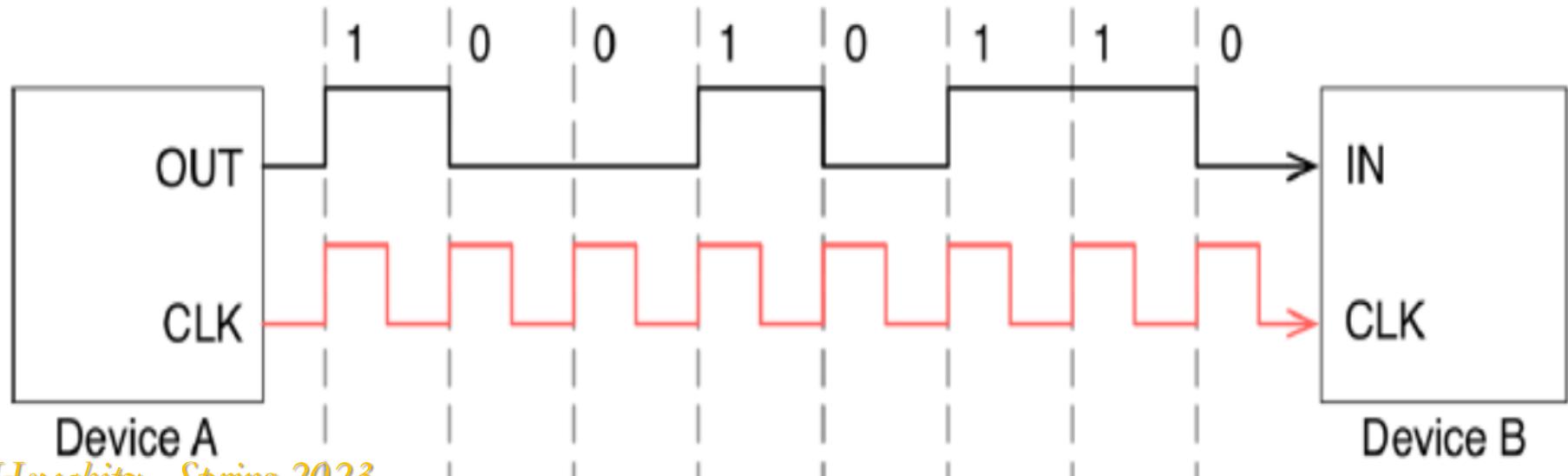
USART

Universal Synchronous and Asynchronous
serial Receiver and Transmitter



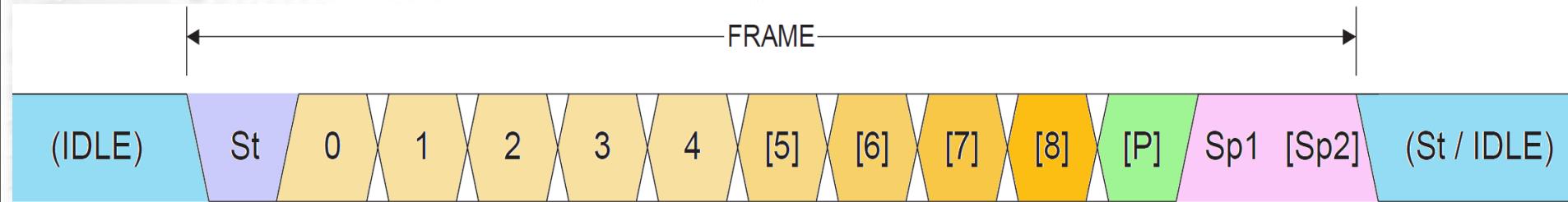
النافذة التسلسليّة USART

المخطط الزمني لعملية الإرسال التسلسلي المتزامن □
Synchronous لبait واحد 0b01101001 من الجهاز
Clock إلى Device B حيث تم استخدام Device A
لضبط توقيت إرسال البيانات، حيث يتم إرسال Bit واحد مع كل
جهة صاعدة لـ Clock، حيث تتعلق سرعة نقل البيانات بتردد الـ
Clock، فكلما زاد تردد الـ Clock كلما زادت سرعة نقل البيانات.



(Universal Asynchronous Receiver and Transmitter Interface)

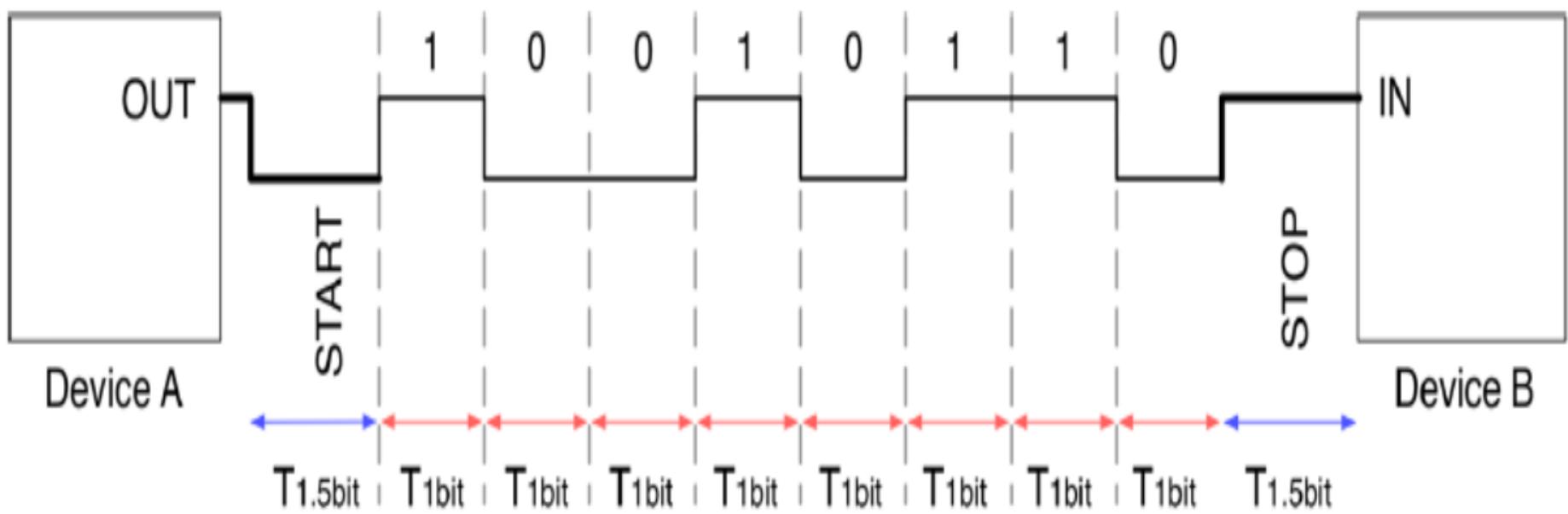
تعتبر هذه النافذة من أكثر نوافذ الاتصال التسلسلي استخداماً في الأنظمة الرقمية والمتحكمات المصغرة والطيفيات التي يمكن أن تربط معها...



بنية إطار البيانات (UART Frame Format)

النافذة التسلسليّة UART

□ في حالة الإرسال الغير متزامن Asynchronous يتم الاستغناء عن الـ **clock** حيث يتم استخدام بت عند بداية الإرسال **Start Bit** و بت عن انتهاء الإرسال **Stop Bit**



النافذة التسلسليّة UART

- تمثل الحالة الخامّلة Idle state بإشارة HIGH وهي حالة عدم الإرسال
- بداية الإرسال تتم من خلال Start Bit وتمثل بإشارة LOW
- هذه الإشارة يتم اكتشافها من قبل المستقبل وتستغرق وقت $T = 1.5$ حيث أن T هو الدور وهو عبارة عن مقلوب التردد أو ما يسمى بـ Baud Rate أي معدل نقل البيانات بين المرسل والمستقبل
- بعد ذلك يتم إرسال الـ 8 bit والتي تمثل البيانات المراد إرسالها حيث يتم إرسال البت الأقل أهمية أولاً LSB، وأحياناً يتم استخدام Parity Bit للتأكد من خلو البيانات من الأخطاء ويتم إنتهاء الإرسال بـ Stop Bit

أنماط العمل المختلفة لـ **STM32** في متحكمات **UART**

نقطة **Blocking Mode**: يسمى أيضاً **Polling Mode**، في هذا النمط يتم تفحص عملية إرسال واستقبال البيانات بشكل مستمر ، حيث ينتظر المعالج لحين انتهاء عملية الإرسال مما يؤدي إلى تأخير معالجة باقي التعليمات وتنفيذ المهام ، وهو نمط العمل الأبسط من ناحية الكود ومن ناحية الـ **Hardware** ويستخدم عندما تكون كمية البيانات المتبادلة ليست كبيرة نسبياً ولا تمثل أهمية عالية من ناحية المعالجة

نقطة **non-Interrupt Mode**: ويسمى أيضاً **Blocking Mode**، في هذا النمط لا يتم الانتظار وتفقد البيانات من حين لآخر للتأكد من عملية الإرسال والاستقبال، حيث عند الانتهاء من إرسال البيانات يتم تفعيل مقاطعة تفيد بانتهاء عملية الإرسال ، وهذا النمط من العمل أفضل من ناحية المعالجة ملائم عندما يكون معدل نقل البيانات صغير نسبياً (أقل من 38400 Bps)

□ **DMA** : وهو النمط الأفضل من ناحية إنتاجية نقل البيانات ومن ناحية سرعة نقل البيانات وعندما نريد تحرير المتحكم من الحمل الإضافي الذي ينتج عن من إحضار البيانات من **RAM** ومعالجتها، فالـ **DMA** يقوم بالوصول إلى الذاكرة **RAM** بدون احتياج أي جهد من المعالج لعمل ذلك، وبدون نمط الـ **DMA** يمكن التعامل مع السرعات العالية في الـ **UART**

هناك بارامترات يجب تحديدها بين المرسل والمستقبل قبل إرسال البيانات في الاتصالات غير المتوافقة وهي:

- ✓ معدل سرعة الإرسال ...1200, 2400, 9600 :**(Baud Rate)**
- ✓ خانة فحص الإيجابية .Even, Odd, or None :**(Parity Bit)**
- ✓ عدد البتات .6, 7, 8, or 9-bit :**(Data Bits)**
- ✓ عدد برات التوقف .1 or 2 :**(Stop Bit)**

هناك **نمطين** للبيانات في الاتصالات التسلسليّة وهما:

(1) **نمط الأسكى** :(Ascii Mode)

123 > 3-byte

(2) **النمط الثنائي** :(BIN Mode)

123 > 1-Byte

مفاهيم أساسية في الاتصالات التسلسليّة:

Baud rate		Oversampling by 16		Oversampling by 8	
S.No	Desired (Bps)	Actual	%Error	Actual	%Error
2	2400	2400	0	2400	0
3	9600	9600	0	9600	0
4	19200	19200	0	19200	0
5	38400	38400	0	38400	0
6	57600	57620	0.03	57590	0.02
7	115200	115110	0.08	115250	0.04
8	230400	230760	0.16	230210	0.8
9	460800	461540	0.16	461540	0.16
10	921600	923070	0.16	923070	0.16
11	2000000	2000000	0	2000000	0
12	3000000	3000000	0	3000000	0
13	4000000	N.A.	N.A.	4000000	0
14	5000000	N.A.	N.A.	5052630	1.05
15	6000000	N.A.	N.A.	6000000	0

مفاهيم أساسية في الاتصالات التسلسليّة:

: وتعني عدد البتات التي يتم إرسالها أو WordLength استقبالها في Frame في المرة الواحدة، وتتوفر 3 قيم يمكن الاختيار بينها 7bit,8bit,9bit حيث لا يتضمن هذا الرقم البتات الخاصة بـ Start وـ Stop وغيرها

: يحدد عدد البتات الخاصة بـ Stop التي سيتم إرسالها، ويمكن الاختيار بين 1 و 2 أي بت واحد أو 2 bit في نهاية الإشارة.

مفاهيم أساسية في الاتصالات التسلسليّة:

□ **Parity**: هو عبارة عن اختبار يستخدم لاكتشاف الأخطاء أثناء عملية الإرسال والاستقبال للبيانات من خلال الـ USART، وهو عبارة عن بت يكون مكانه عند البت الأكثر أهمية MSB حيث لو تم استخدام Word Length بـ 8-bit يكون مكانه في البت الثامن، أما لو تم استخدام 9-bit يكون مكانه هو في البت التاسع ولها نمطين:

□ **فردي Odd**: تكون قيمة بت الـ Parity مساوٍ للواحد المنطقي عندما يكون عدد الواحات الموجودة في الكلمة المراد إرسالها زوجي، وصفر منطقي في حال كان عدد الواحات الموجودة في الكلمة المراد إرسالها فردي.

□ **زوجي Even**: تكون قيمة بت الـ Parity مساوٍ للواحد المنطقي عندما يكون عدد الواحات الموجودة في الكلمة المراد إرسالها فردي، وصفر منطقي في حال كان عدد الواحات الموجودة في الكلمة المراد إرسالها زوجي.

على سبيل المثال:

- عندما تريد إرسال أي بيانات يتم تحويلها لل Binary فمثلاً إذا كنا نريد إرسال الكلمة التالية 0b01101110 فمن خلال الـ Parity يتم حساب عدد الوحدات الموجودة ضمن هذه الكلمة المراد إرسالها وهي في هذه الحالة 5، ففي حال كنت تستخدم نمط الفردي ستكون قيمة الـ Parity صفر، أما في حال كنت تستخدم نمط الزوجي ستكون قيمة الـ Parity واحد.
- يتم إرسال قيمة البت الخاص بالـ Parity من المرسل إلى المستقبل، فإن لم يحصل تطابق بين قيمته عند المرسل مع قيمته عند المستقبل فهذا يعني وجود خطأ ما في الإرسال حيث يتم طلب إعادة الإرسال.

دوال مكتبة HAL المستخدمة للتتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

سنستخدم دالتين رئيسيتين للتتعامل مع المنفذ التسلسلي إحداهما للارسال والأخرى للاستقبال:

دالة الإرسال: □

```
HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t  
Timeout);
```

حيث:

هو مؤشر يشير إلى **huart** □ :

أي المنفذ التسلسلي المستخدم للاتصال مثلاً قد يكون **huart1 & huart2** أو **huart3 & huart4**

دوال مكتبة HAL المستخدمة ل التعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما ل لإرسال والأخرى ل الاستقبال:

□ دالة الإرسال:

```
HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);
```

حيث:

□ Data : وهو مؤشر أيضاً يشير إلى البيانات التي سيتم إرسالها عبر UART وكما نرى نوعه uint8_t أي يقبل إرسال بيانات من نوع Unsigned int وبطول bit8، مثال: قد تكون pData مصفوفة ول يكن اسمها Data وتكون معرفة بالشكل التالي:

```
Uint8_t Data[] = {0,1,2,3,4,5,6,7,8,9};
```

دوال مكتبة HAL المستخدمة ل التعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما ل لإرسال والأخرى ل الاستقبال:

□ دالة الإرسال:

```
HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t  
Timeout);
```

حيث:

Size: وهو متغير يعبر عن حجم البيانات التي سيتم إرسالها أي
pData وهي في المثال السابق 10.

دوال مكتبة HAL المستخدمة ل التعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما للارسال والأخرى للاستقبال:

□ دالة الإرسال:

```
HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);
```

حيث:

□ **Timeout**: أقصى زمن يتم انتظاره بالميلي ثانية حتى يتم اكتمال عملية الإرسال، فإذا تم انتهاء هذا الزمن ولم تتم عملية الإرسال سيتم قطع عملية الإرسال وتقوم الدالة برجوع **HAL_TIMOUT** ماعدا ذلك يتم ارجاع **HAL_OK**، ويمكن استخدام الدالة **HAL_MAX_DELAY** وهي وظيفتها انتظار **Timeout** مكان **Timeout**

دوال مكتبة HAL المستخدمة ل التعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

مثال: إذا أردنا إرسال المصفوفة التالية عبر المنفذ التسلسلي الأول : UART1

```
/* USER CODE BEGIN 0 */  
uint8_t data[]={0,1,2,3,4,5,6,7,8,9};  
/* USER CODE END 0 */
```

نستخدم الدالة التالية:

```
/* USER CODE BEGIN 3 */  
/* Infinite loop */  
while (1)  
{  
    HAL_UART_Transmit(&huart1,data,10,1000);  
}  
/* USER CODE END 3 */
```

دوال مكتبة HAL المستخدمة للتتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

دالة الاستقبال: إذا أردنا استقبال بيانات على UART باستخدام وضع Polling ومكتبات HAL نقوم باستدعاء الدالة التالية:

```
HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t  
Timeout);
```

حيث:

uart : هو مؤشر يشير إلى أي المنفذ التسلسلي Struct_UART_HandleTypeDef المستخدم للاتصال مثلاً قد يكون huart1 & huart2 أو huart3 & huart4

دوال مكتبة HAL المستخدمة ل التعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

دالة الاستقبال:

```
HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);
```

حيث:

: وهو مؤشر أيضاً يشير إلى البيانات التي سيتم استقبالها عبر Data
وكما نرى نوعه uint8_t أي يقبل استقبال بيانات من نوع UART
وبطول bit8، مثال: قد تكون pData مصفوفة ولتكن Unsigned int
اسمها Data و تكون معرفة بالشكل التالي:

```
Uint8_t Data[10];
```

دوال مكتبة HAL المستخدمة ل التعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

دالة الاستقبال:

```
HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t  
Timeout);
```

حيث:

size: وهو متغير يعبر عن حجم البيانات التي سيتم استقبالها أي
وهي في المثال التالي 10.

Timeout: أقصى زمن يتم انتظاره بالميلي ثانية حتى يتم اكتمال
عملية الاستقبال، فإذا تم انتهاء هذا الزمن ولم تتم عملية الاستقبال سيتم
قطع عملية الاستقبال وتقوم الدالة بإرجاع HAL_TIMOUT ماعدا
ذلك يتم إرجاع HAL_OK، ويمكن استخدام الدالة
ووظيفتها انتظار أقصى زمن ممكن لعملية
استقبال HAL_MAX_DELAY.

دوال مكتبة HAL المستخدمة ل التعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

دالة الاستقبال:

```
HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);
```

مثال:

```
/* USER CODE BEGIN 0 */
uint8_t data[10];
/* USER CODE END 0 */
```

```
/* USER CODE BEGIN 3 */
/* Infinite loop */
while (1)
{
    HAL_UART_Receive(&huart1,data,10,1000);
}
/* USER CODE END 3 */
```

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ **interrupt**

توفر جميع متحكمات **UART** مقاطعات لـ **STM32** كما في الجدول التالي:

Interrupt Event	Event Flag	Enable Control Bit
Transmit Data Register Empty	TXE	TXEIE
Clear To Send (CTS) flag	CTS	CTSIE
Transmission Complete	TC	TCIE
Received Data Ready to be Read	RXNE	RXNEIE
Overrun Error Detected	ORE	RXNEIE
Idle Line Detected	IDLE	IDLEIE
Parity Error	PE	PEIE
Break Flag	LBD	LBDIE
Noise Flag, Overrun error and Framing Error	NF or ORE or FE	EIE
Error in multi buffer communication		

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ **interrupt**

يمكن تقسيم المقاولات **IRQs** الخاصة بالمنفذ التسلسلي **UART** لمجموعتين:

: **IRQs** التي يتم استدعائهما أثناء الإرسال:

- **Transmission complete**
- **Clear to send(CTS)**
- **transmission Data Register** مسجل البيانات فارغ
- **Empty**
- **Noise Flag**
- **Framing error**

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

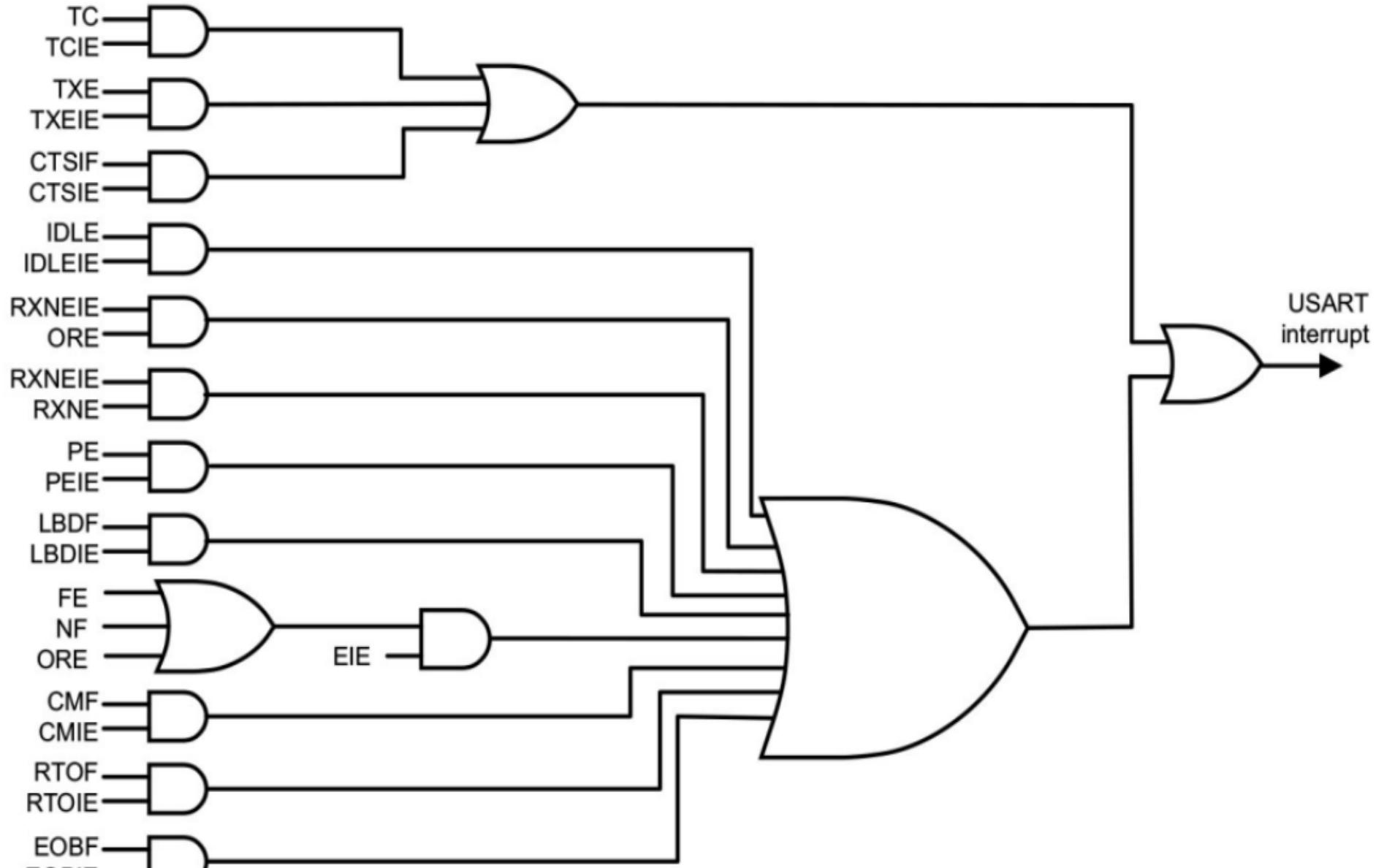
: التي يتم استدعائها أثناء الاستقبال: IRQs

- **Idle line detection**
- **Overrun error**
- **Receive data register not empty**
- **Parity error**
- **Lin break detection**
- **Noise Flag**
- **Framing error**

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ **interrupt**

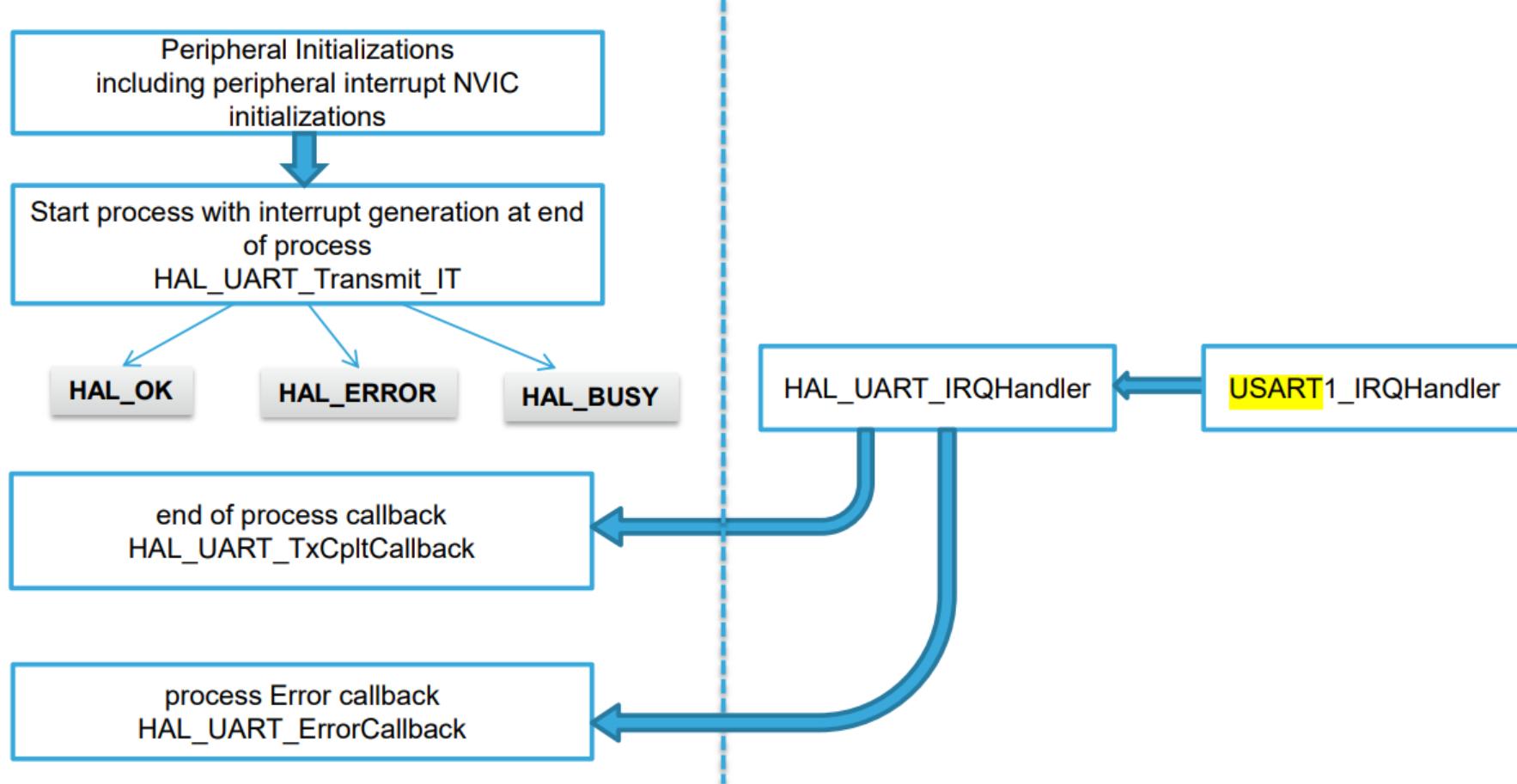
- يتم تفعيل حدث المقاطةة Interrupt event لكل نوع من خلال Enable Control Bit الخاص به كما في الجدول السابق، حيث كل هذه الـ USART Peripheral لها فقط خط مقاطعة وحيد لكل IRQs
- باعتبار أن لكل وحدة USART في متحكمات STM32 خط مقاطعة وحيد لذا يتوجب على المستخدم تحليل علم المقاطةة Flag Event الذي تم رفعه لمعرفة المقاطةة التي حدثت

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt



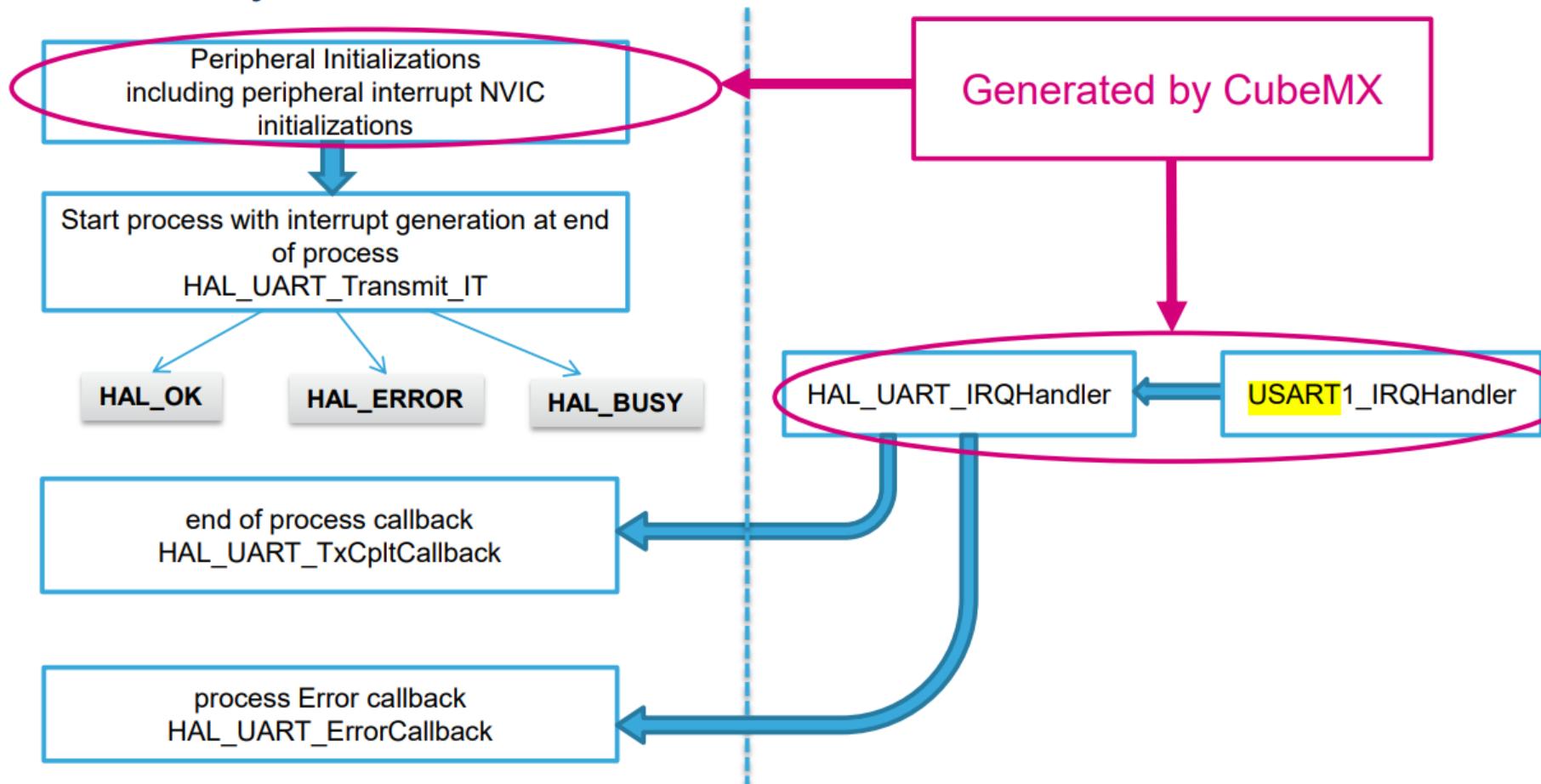
إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

HAL Library UART with IT transmit flow



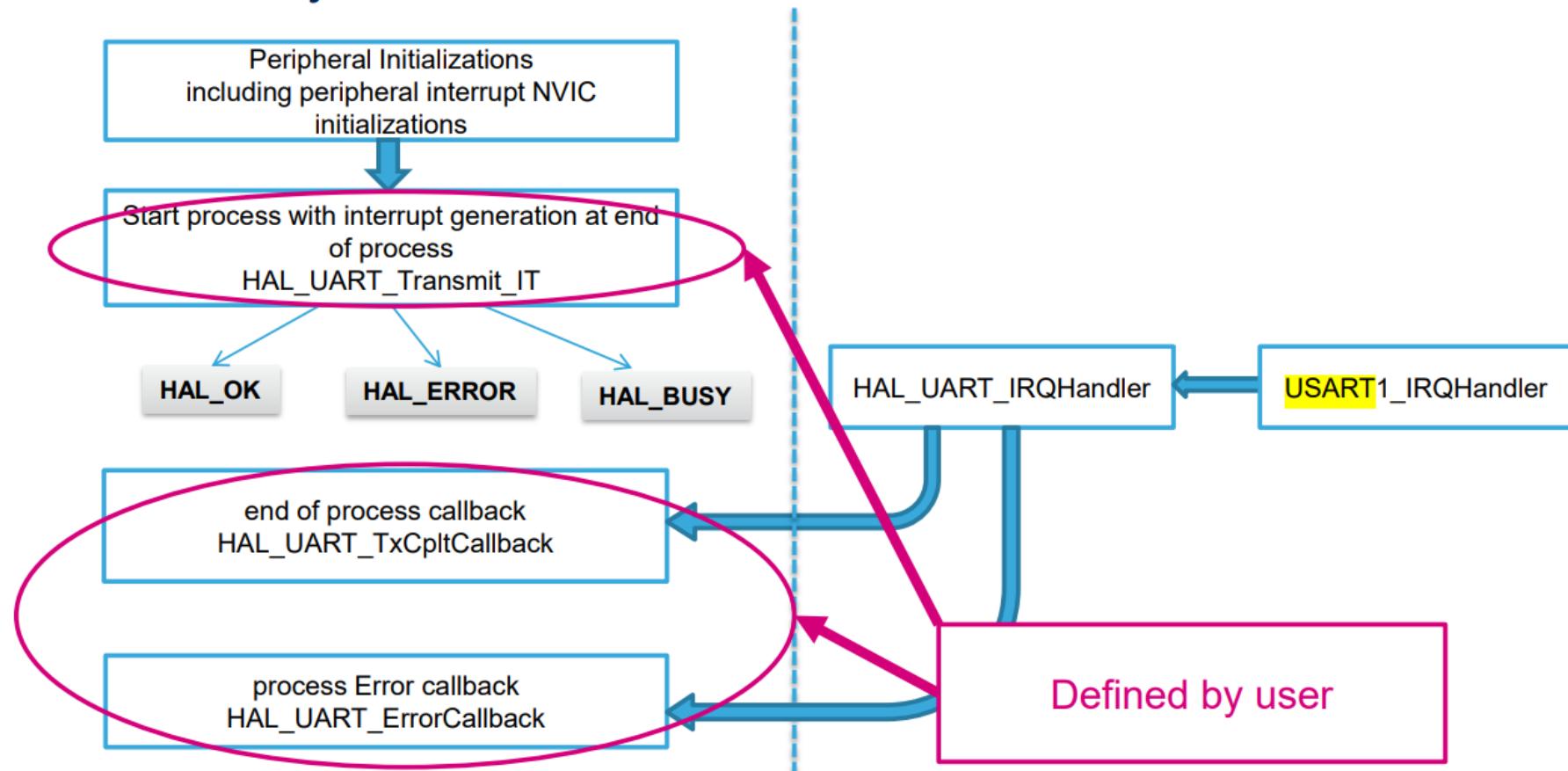
إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

HAL Library UART with IT transmit flow



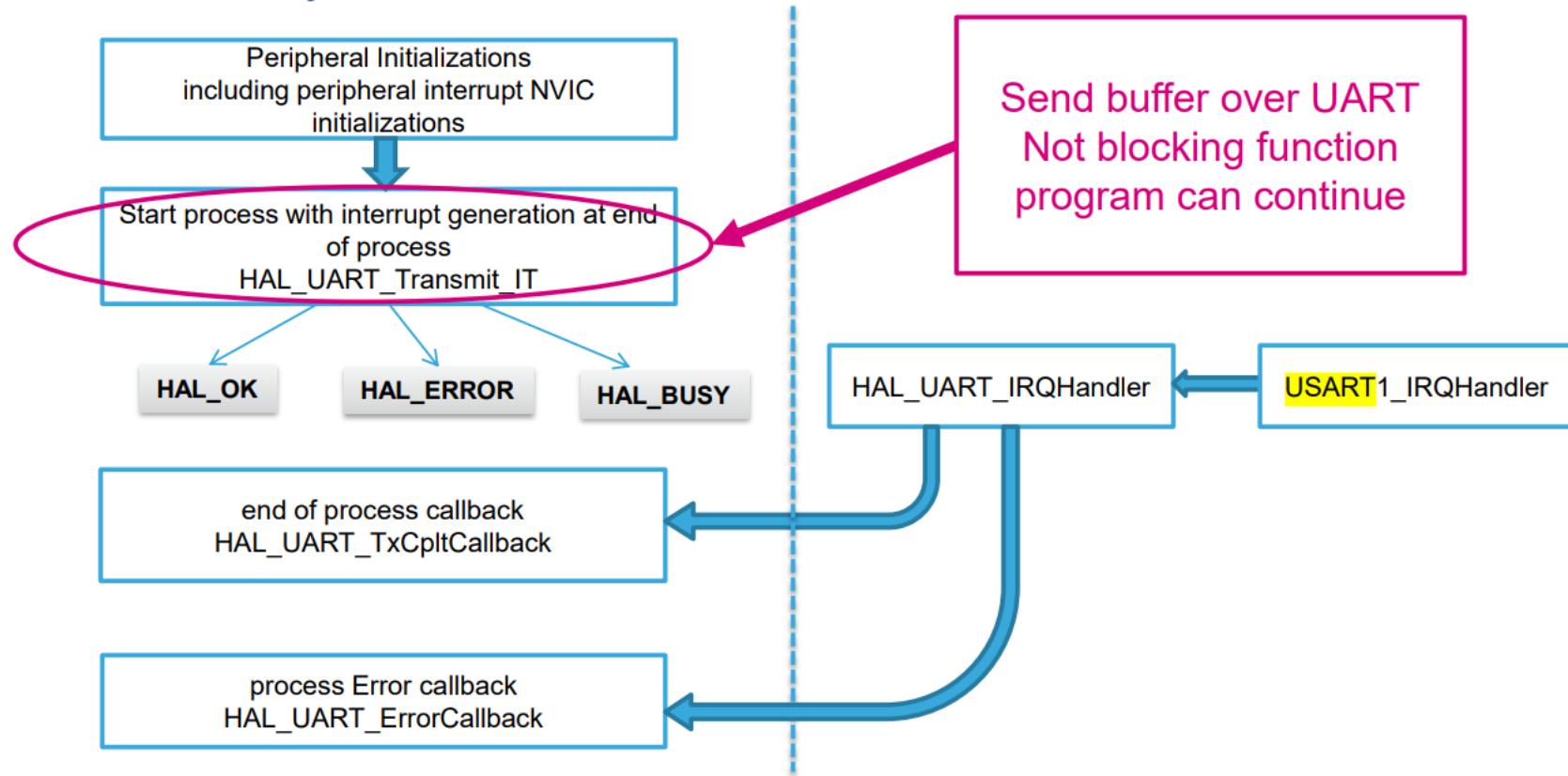
إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

HAL Library UART with IT receive flow



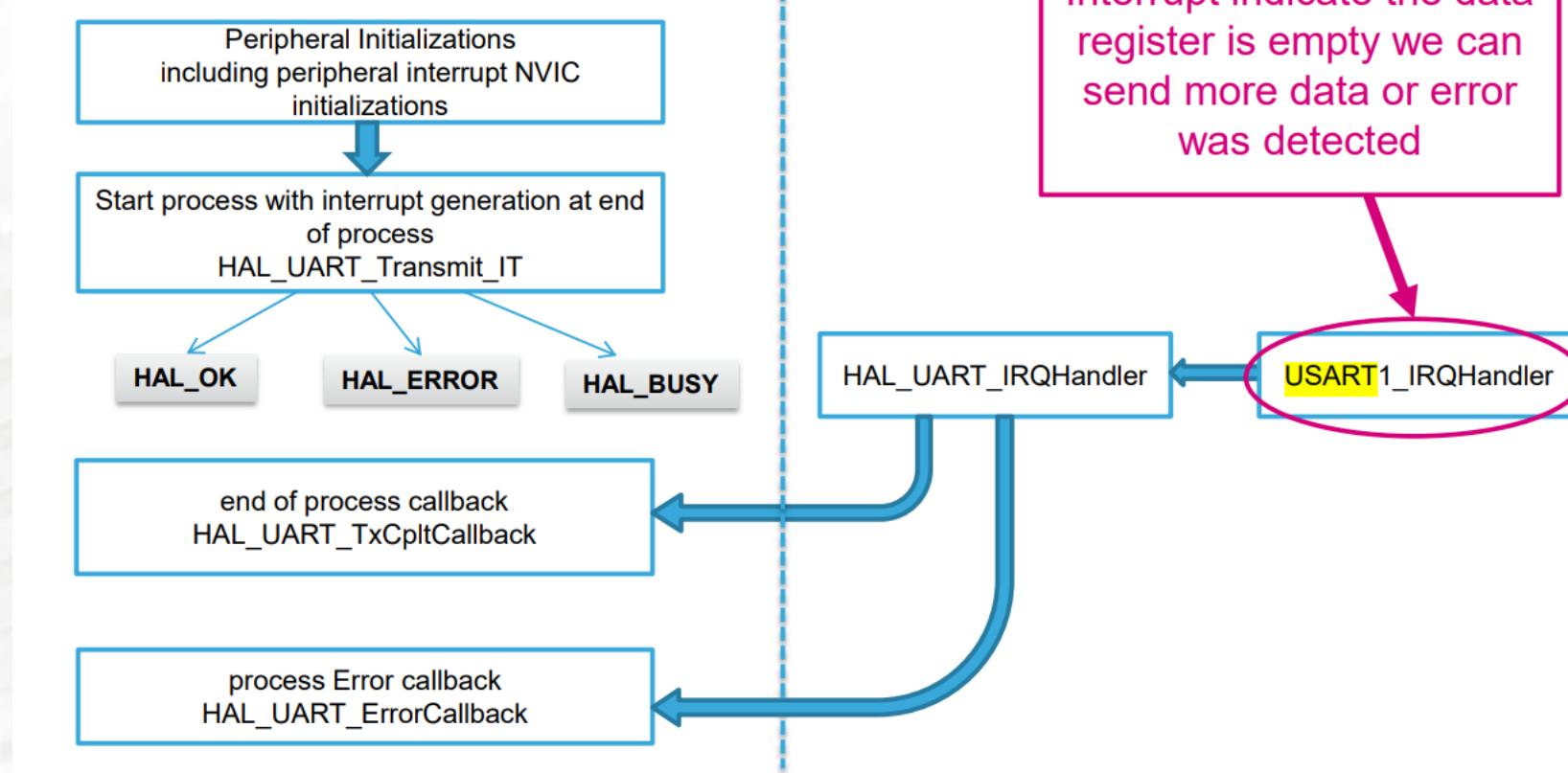
إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

HAL Library UART with IT receive flow



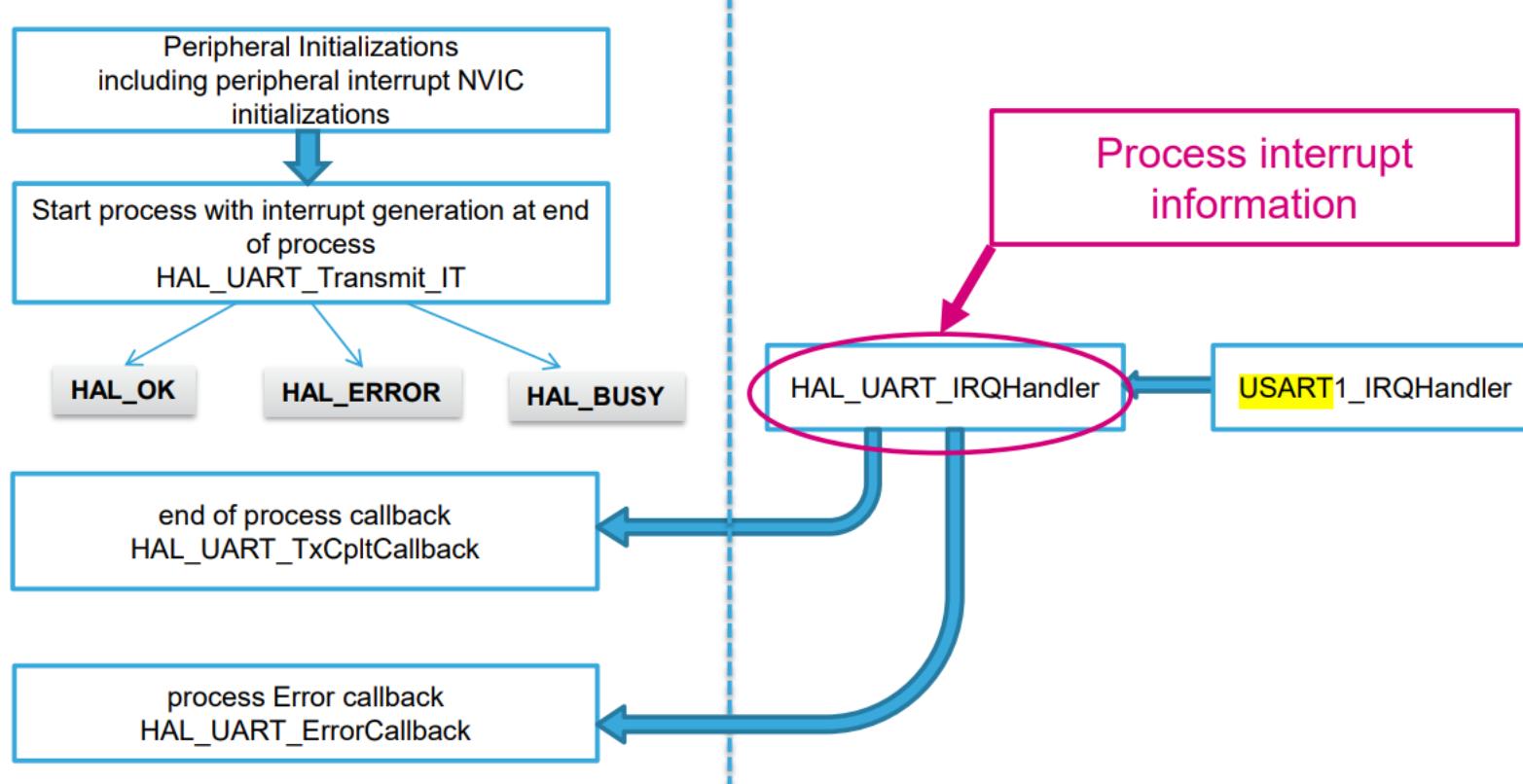
إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

HAL Library UART with IT receive flow



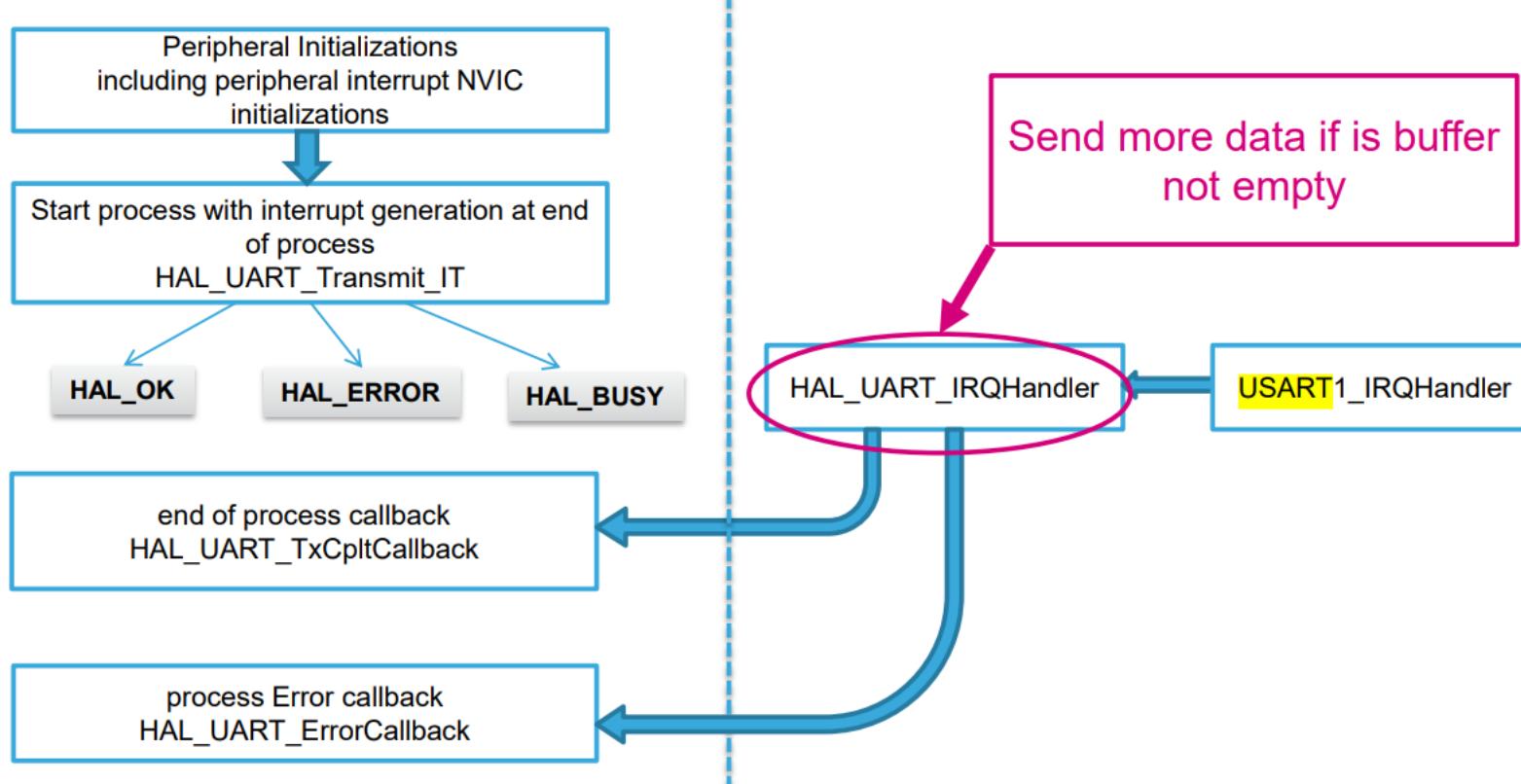
إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

HAL Library UART with IT receive flow



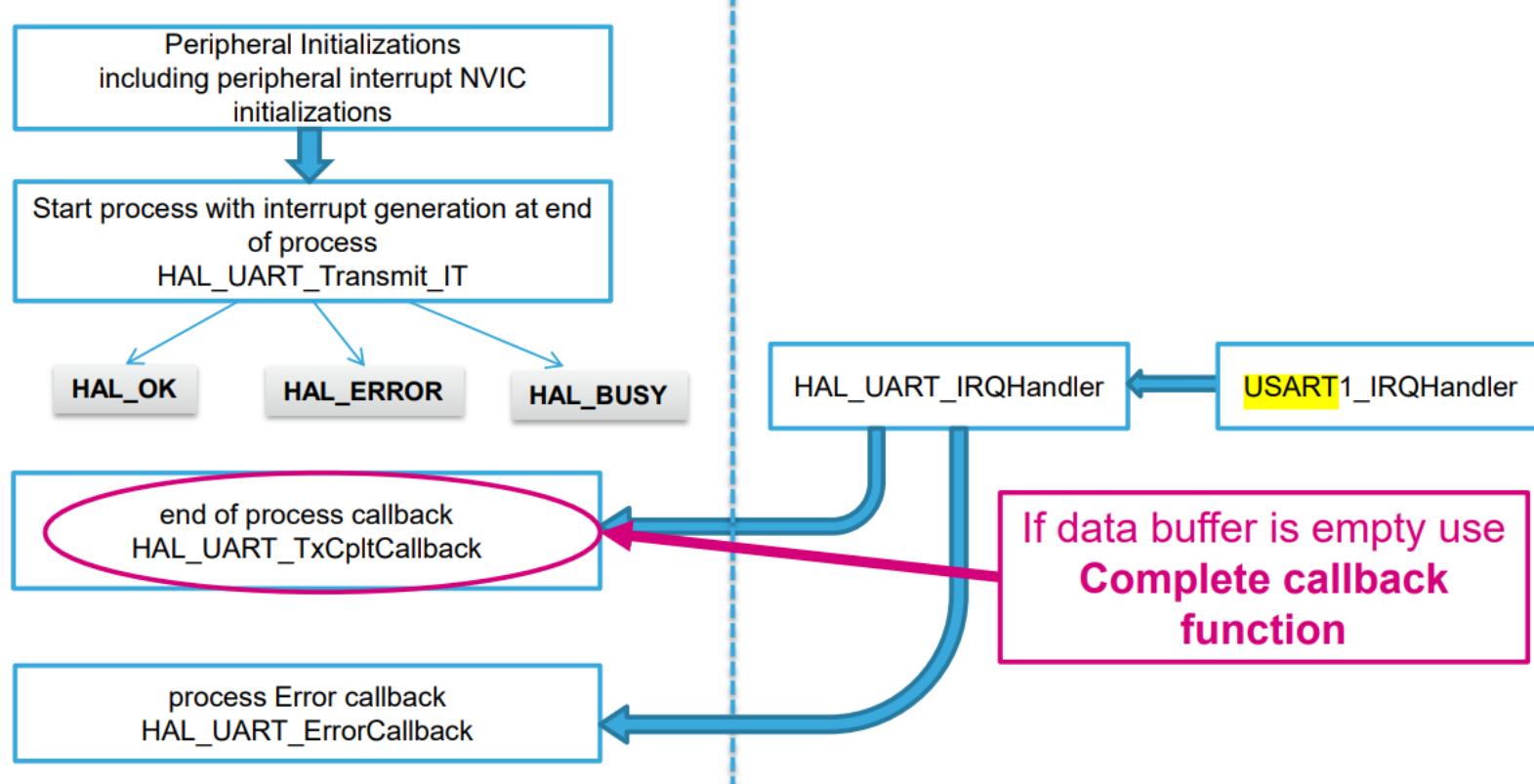
إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

HAL Library UART with IT receive flow



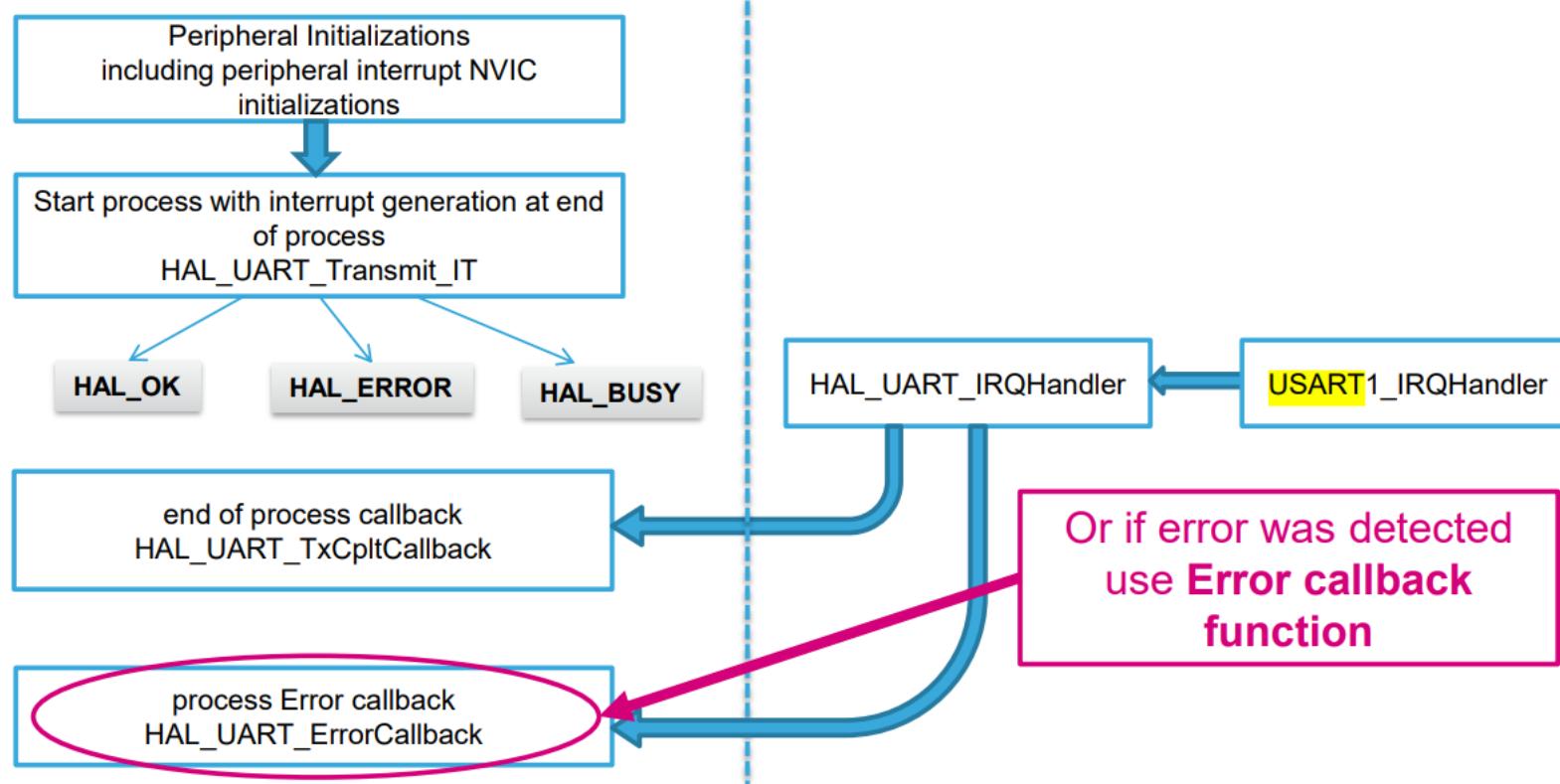
إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

HAL Library UART with IT receive flow



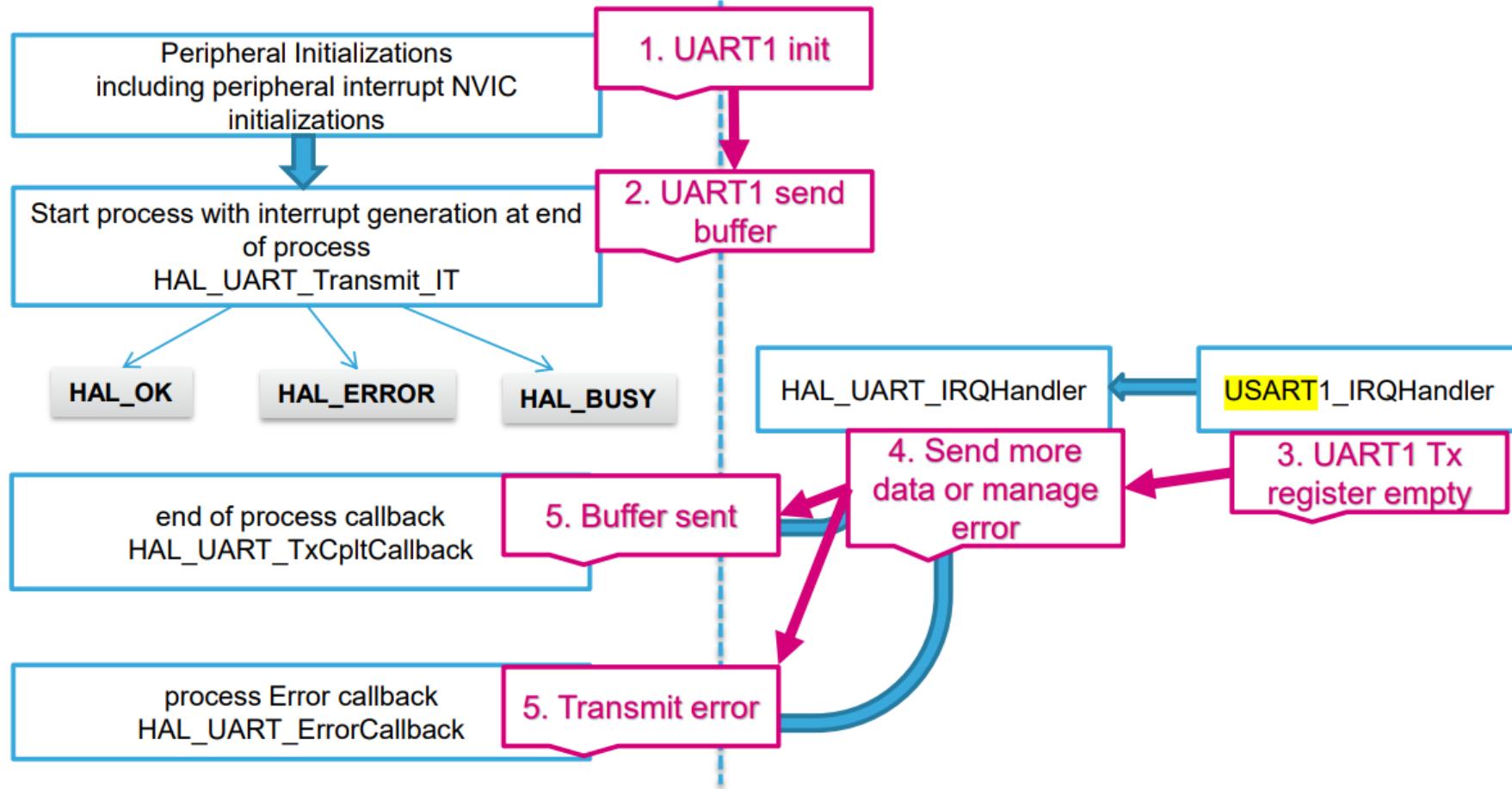
إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

HAL Library UART with IT receive flow



إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

HAL Library UART with IT receive flow



دوال مكتبة HAL المستخدمة ل التعامل مع منفذ الاتصال التسلسلي في وضع الـ Interrupt

سنستخدم دالتين رئيسيتين ل التعامل مع المنفذ التسلسلي إحداهما لإرسال والأخرى ل الاستقبال:

- دالة الإرسال:

`HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);`

حيث:

- يتم إدخال بارامترات هذه الدالة كما قمنا بالشرح سابقاً ، وكما تلاحظ فقد تم إضافة IT في اسم الدالة وأيضاً تم إزالة Timeout من بارامترات هذه الدالة مقارنة بالدالة المستخدمة في نمط الـ Polling، لأنه لم يعد هناك زمن انتظار في نمط المقاطعة.

دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ interrupt

□ دالة الاستقبال: إذا أردنا استقبال بيانات على UART باستخدام وضيع interrupt ومكتبات HAL نقوم باستدعاء الدالة التالية:

```
HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
```

حيث:

□ تعيد دالة الإرسال أو الاستقبال إما HAL_OK في حال تمت عملية الإرسال/الاستقبال بنجاح، أو HAL_error في حال حدوث خطأ أثناء عملية الإرسال/الاستقبال أو HAL_Busy

التطبيق العملي 1: استخدام المنفذ التسلسلي UART2 في نمط الـ Polling لطباعة قيمة متحوال على النافذة التسلسليّة

نقوم بضبط إعدادات المنفذ التسلسلي:

SX *uart.ioc

Pinout & Configuration Clock Configuration Project Manager Tools

Software Packs Pinout

Categories A-Z

System Core

- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog

Timers

Connectivity

- I2C1
- I2C2
- IRTIM
- LPUART1
- SPI1
- SPI2
- UCPD1
- UCPD2
- USART1
- USART2**
- USART3
- USART4

Multimedia

Computing

Middleware

Utilities

USART2 Mode and Configuration

Mode

- Mode: Asynchronous
- Hardware Flow Control (RS232): Disable
- Hardware Flow Control (RS485)
- Slave Select(NSS) Management: Disable

Configuration

Reset Configuration

Parameter Settings User Constants

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

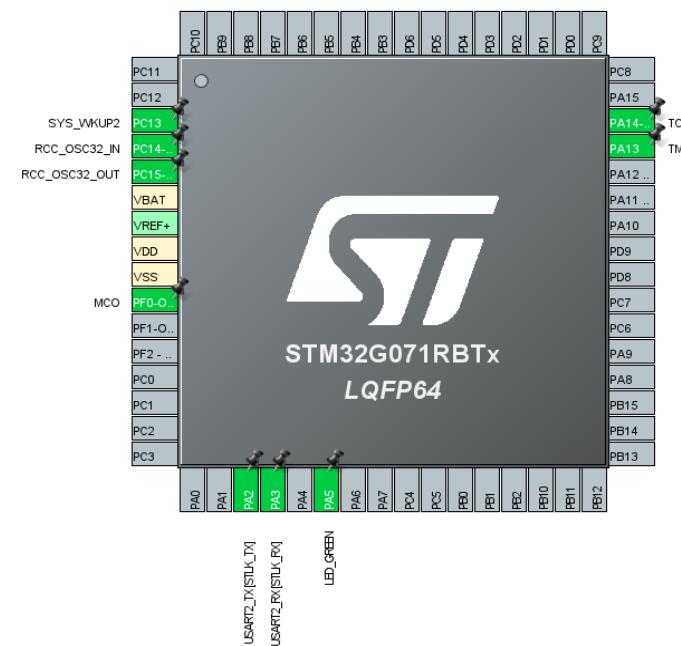
- Baud Rate: 115200
- Word Length: 8 Bits (including Parity)
- Parity: None
- Stop Bits: 1

Advanced Parameters

- Data Direction: Receive and Transmit
- Over Sampling: 16 Samples
- Single Sample: Disable
- ClockPrescaler: 1
- Fifo Mode: Disable
- Txfifo Threshold: 1 eighth full configuration
- Rx fifo Threshold: 1 eighth full configuration

Advanced Features

- Auto Baudrate: Disable
- TX Pin Active Level Inversion: Disable
- RX Pin Active Level Inversion: Disable
- Data Inversion: Disable
- TX and RX Pins Swapping: Disable
- Overrun: Enable
- DMA on RX Error: Enable
- MSB First: Disable



التطبيق العملي 1 : استخدام المنفذ التسلسلي UART2 في نمط الـ Polling لطباعة قيمة متتحول على النافذة التسلسليّة

□ يصبح الكود بالشكل التالي:

```
#include "main.h"

UART_HandleTypeDef huart2;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
int main(void)
{
```

التطبيق العملي 1: استخدام المنفذ التسلسلي UART2 في نمط الـ طباعة قيمة متحوال على النافذة التسلسليّة Polling

```
uint8_t MSG[35] = {'\0'};  
uint8_t X = 0;  
HAL_Init();  
SystemClock_Config();  
MX_GPIO_Init();  
MX_USART2_UART_Init();
```

التطبيق العملي 1 : استخدام المنفذ التسلسلي UART2 في نمط الـ Polling لطباعة قيمة متتحول على النافذة التسلسليّة

```
while (1)
```

```
{
```

```
    sprintf(MSG, "Hello Dudes! Tracing X =  
%d\r\n", X);
```

```
    HAL_UART_Transmit(&huart2, MSG,  
sizeof(MSG), 100);
```

```
    HAL_Delay(500);
```

```
    X++;
```

```
}
```

التطبيق العملي 2 : استخدام المنفذ التسلسلي UART2 في نمط الـ interrupt لطباعة قيمة متتحول على النافذة التسلسليّة

نقوم بضبط إعدادات المنفذ التسلسلي:

SX *uart.ioc

Pinout & Configuration Clock Configuration Project Manager Tools

Categories A-Z

System Core

- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog

Timers

Connectivity

- I2C1
- I2C2
- RTIM
- LPUART1
- SPI1
- SPI2
- UCPD1
- UCPD2
- USART1
- USART2**
- USART3
- USART4

Multimedia

Computing

Middleware

Utilities

UART2 Mode and Configuration

Mode

- Mode: Asynchronous
- Hardware Flow Control (RS232): Disable
- Hardware Flow Control (RS485)
- Slave Select(NSS) Management: Disable

Configuration

Reset Configuration

NVIC Settings DMA Settings GPIO Settings

Parameter Settings User Constants

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

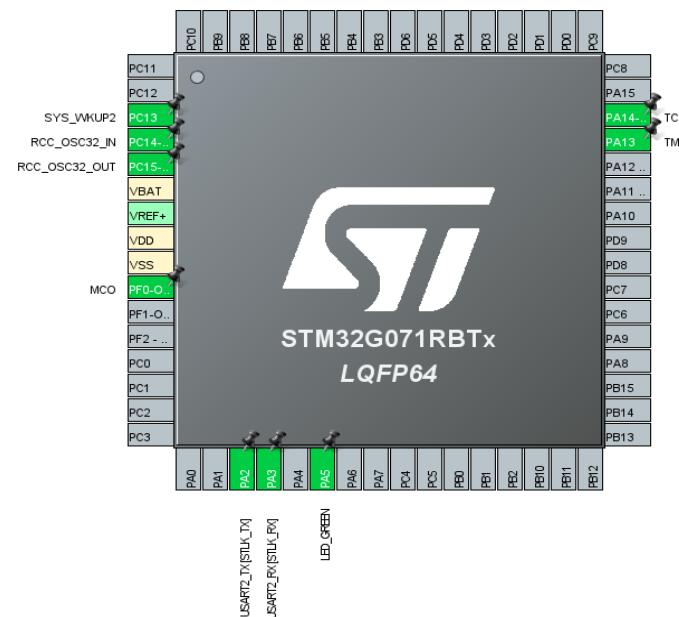
- Baud Rate: 115200
- Word Length: 8 Bits (including Parity)
- Parity: None
- Stop Bits: 1

Advanced Parameters

- Data Direction: Receive and Transmit
- Over Sampling: 16 Samples
- Single Sample: Disable
- ClockPrescaler: 1
- Fifo Mode: Disable
- Txiffo Threshold: 1 eighth full configuration
- Rxiffo Threshold: 1 eighth full configuration

Advanced Features

- Auto Baudrate: Disable
- TX Pin Active Level Inversion: Disable
- RX Pin Active Level Inversion: Disable
- Data Inversion: Disable
- TX and RX Pins Swapping: Disable
- Overrun: Enable
- DMA on RX Error: Enable
- MSB First: Disable



التطبيق العملي 2 : استخدام المنفذ التسلسلي UART2 في نمط الـ interrupt لطباعة قيمة متتحول على النافذة التسلسليّة

نقوم بضبط إعدادات مقاطعة المنفذ التسلسلي:



The screenshot shows the MX Configuration tool interface for a project named *UART_interrupt.ioc. The left sidebar lists various peripheral categories like System Core, Analog, Timers, Connectivity, I2C, SPI, USART, Multimedia, Computing, and Middleware. The USART2 category is currently selected and highlighted in blue. The main workspace contains two tabs: 'Pinout & Configuration' and 'Clock Configuration'. The 'Pinout & Configuration' tab is active, displaying the 'USART2 Mode and Configuration' section under 'Mode'. It shows the mode set to 'Asynchronous', with options for 'Hardware Flow Control (RS232)' (disabled), 'Hardware Flow Control (RS485)' (unchecked), and 'Slave Select(NSS) Management' (disabled). Below this is the 'Configuration' tab, which includes sections for 'Reset Configuration', 'NVIC Settings' (checked), 'Parameter Settings' (checked), 'DMA Settings' (checked), 'GPIO Settings' (checked), and 'User Constants'. A table titled 'NVIC Interrupt Table' shows an entry for 'USART2 global interrupt / USART2 wake-up interrupt thr...' with 'Enabled' checked and 'Preemption Priority' set to 0.

التطبيق العملي 1 : استخدام المنفذ التسلسلي UART2 في نمط الـ interrupt لطباعة قيمة متتحول على النافذة التسلسليّة

□ يصبح الكود بالشكل التالي:

```
#include "main.h"

UART_HandleTypeDef huart2;
uint8_t rx_buffer[4];
uint8_t tx[]='hello';
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
```

التطبيق العملي 1 : استخدام المنفذ التسلسلي UART2 في نمط الـ interrupt لطباعة قيمة متتحول على النافذة التسلسليّة

```
int main(void)
{
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    MX_USART2_UART_Init();
    HAL_UART_Transmit_IT(&huart2, tx, 5);
    HAL_UART_Receive_IT(&huart2, rx_buffer, 4);
```

التطبيق العملي 1 : استخدام المنفذ التسلسلي UART2 في نمط الـ interrupt لطباعة قيمة متتحول على النافذة التسلسليّة

```
while (1)
```

```
{  
}
```

```
Void
```

```
HAL_UART_RxCpltCallback(UART_HandleTypeDef
```

```
*huart)
```

```
{
```

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

```
HAL_UART_Receive_IT(&huart2, rx_buffer, 4);
```

```
HAL_UART_Transmit_IT(&huart2, rx_buffer, 4);
```

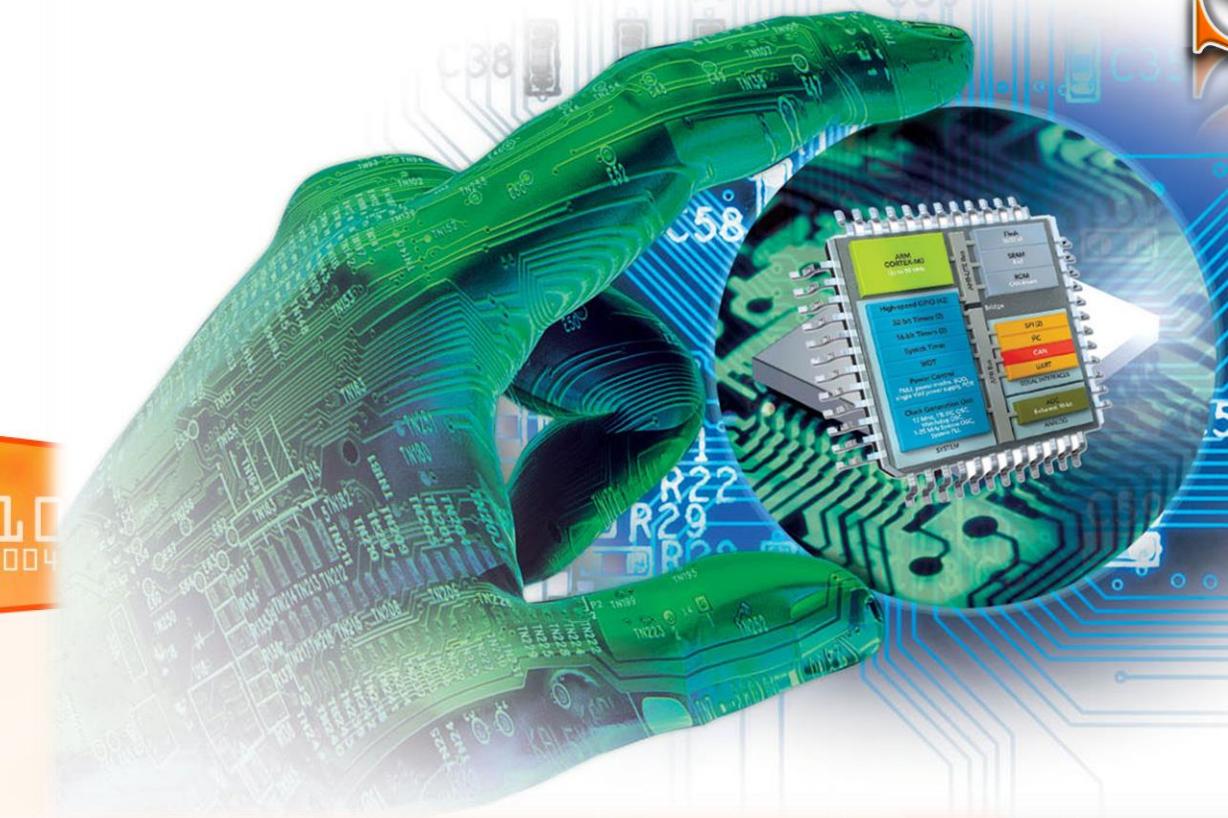
```
}
```

Thank you for listening

مُتَحَكِّمَات

STM32

6



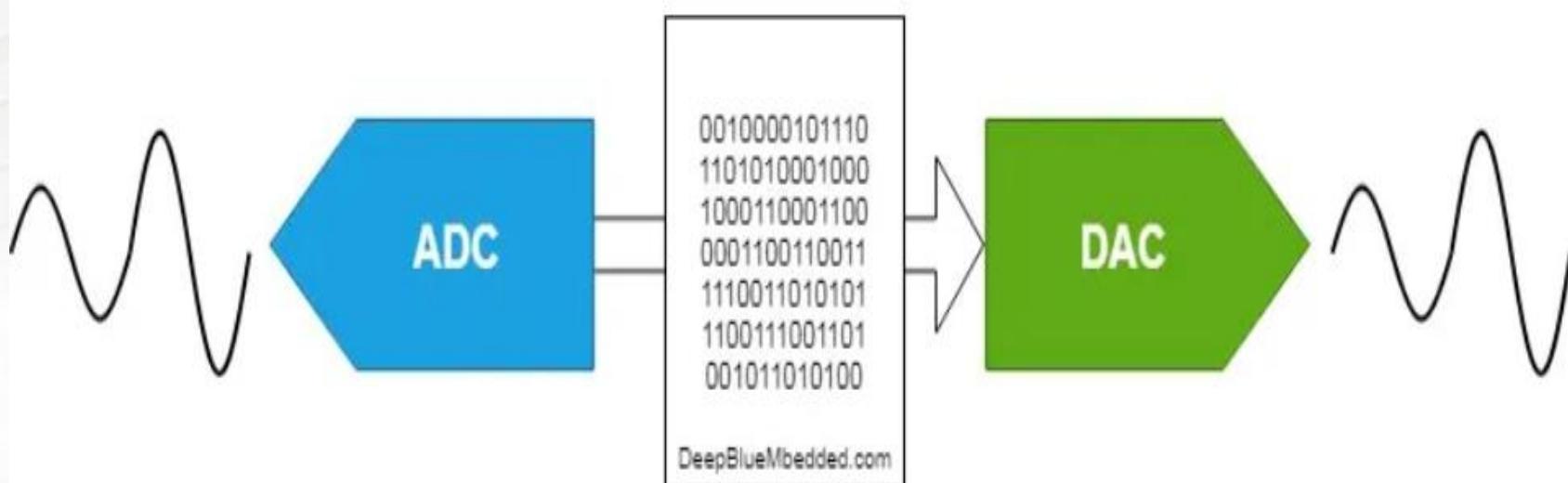
موضع عاًن المعاشرة:

- المبدلات التشابهية الرقمية في متحكمات STM32
- أنماط عمليات التحويل الرقمي modes
- طرق قراءة المبدل التشابهي الرقمي
- الأخطاء الناتجة عن التحويل
- التطبيق العملي

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

المبدلات التشابهية الرقمية عبارة عن دارات الكترونية تقوم بتحويل الجهد التشابهي على دخلها إلى قيمة رقمية بالنظام الثنائي مقابلة لمستوى الجهد، فبمجرد قذح المبدل التشابهي الرقمي يبدأ بأخذ العينات samples ويقوم بعملية تدعى التكميم ليقابل كل مستوى من الجهد بما يناسبه من القيم الرقمية

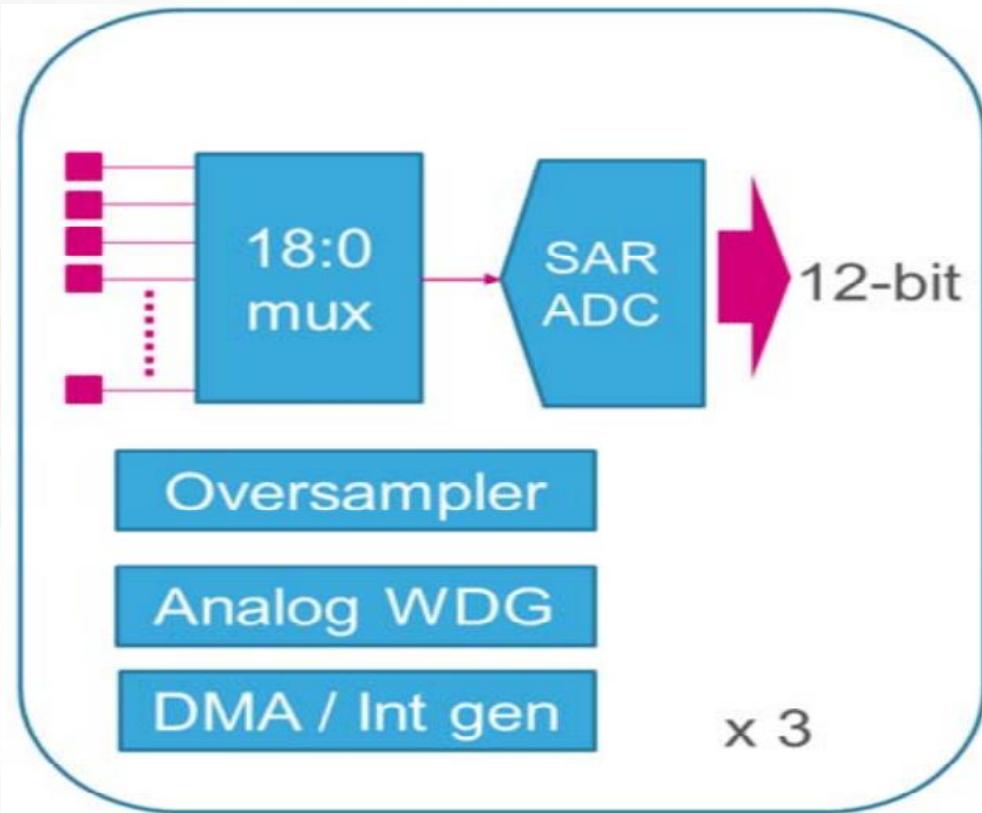


المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

تحتوي متحكمات STM32G0 على مبدل تشابهي رقمي وحيد من نوع Successive approximation ADC(SAR)

التالي:



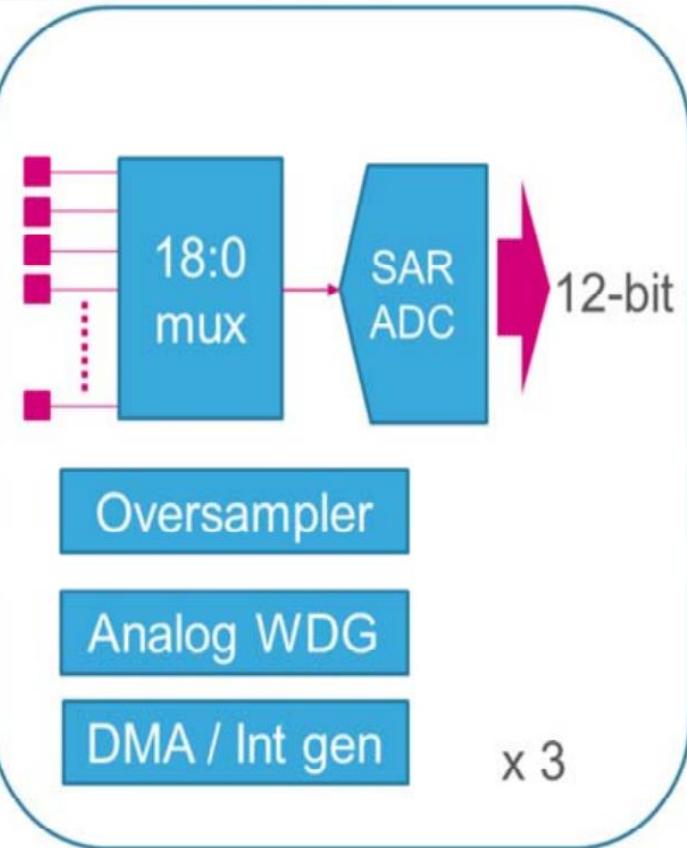
- مبدل ADC وحيد بدقة 12 بت وما يقارب الـ 19 قناة للمبدل
- Oversampler
- بحد أقصى لعملية أخذ العينات يصل إلى Msamples/s 2.5

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

تحتوي متحكمات STM32G0 على مبدل تشابهي رقمي وحيد من نوع Successive approximation ADC(SAR)

التالي:



لكل مبدل تشابهي رقمي ثلاثة مراقبات تشابهية
analog watchdogs
لمراقبة حالات الـ

thresholds Oversampler

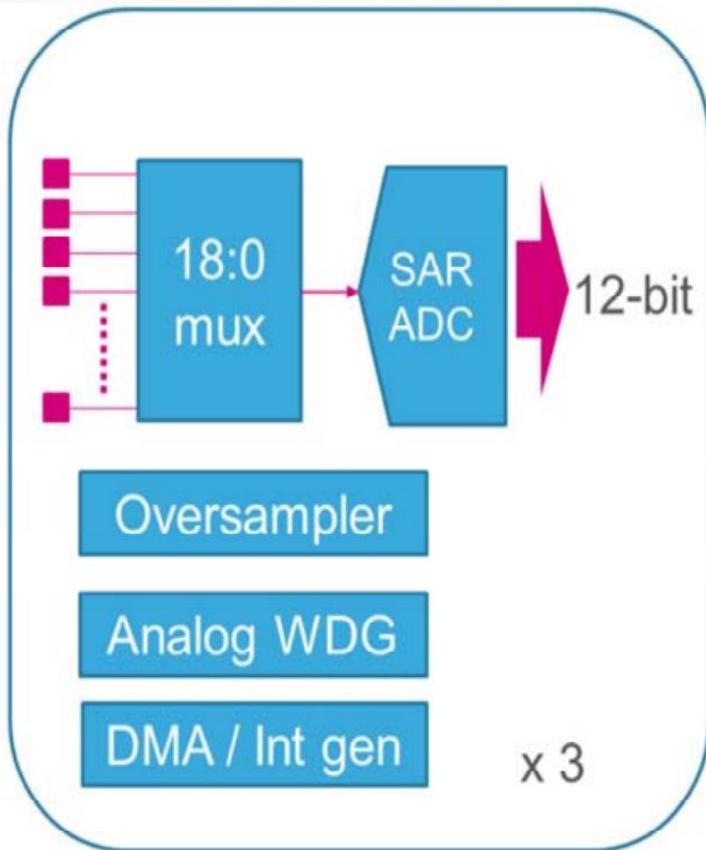
القدرة على توليد طلبات DMA
 القدرة على توليد مقاطعة interrupt
 سحب منخفض للتيار

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

تحتوي متحكمات STM32G0 على مبدل تشابهي رقمي وحيد من نوع Successive approximation ADC(SAR)

التالي:



- قابل للقبح بعدة طرق
- القدرة على إدارة البيانات القادمة ومن ثم تحويلها للأ. CPU.
- تسمح المبدلات التشابهية الرقمية للمتحكم STM32G0 باستقبال القيم التشابهية القادمة من الحساسات، حيث تقوم بتحويلها إلى القيم الرقمية المقابلة لها

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

يوضح الجدول التالي أهم الخصائص لـ ADC الموجود في متحكمات STM32G0

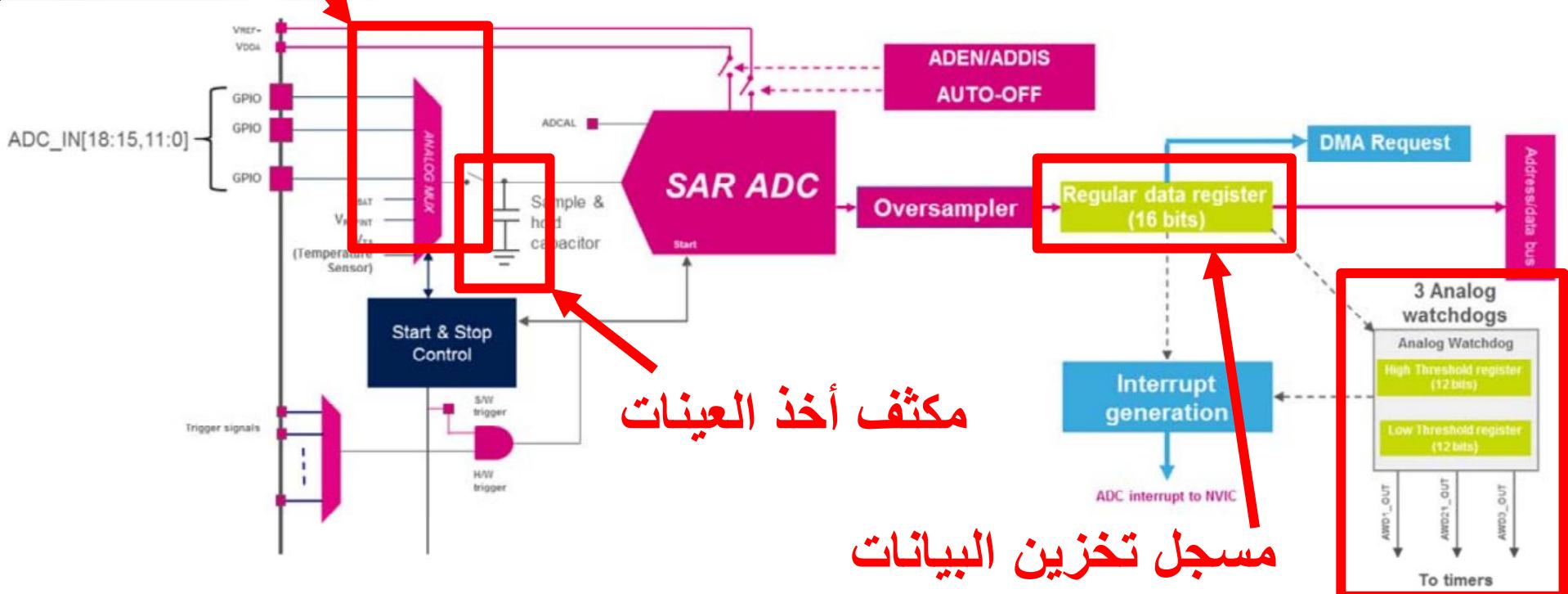
Features	Description
Input channel	Up to 16 external (GPIOs) and 3 internal channels
Type of conversion	12-bit successive approximation
Conversion time	400 ns, 2.5 Msamples/s (when $f_{ADC_CLK} = 35\text{ MHz}$, 12 bits)
Functional mode	Single, Continuous, Scan, and Discontinuous
Triggers	Software or external trigger (Timers & IOs)
Special functions	Analog watchdogs, Hardware oversampling, and Self-calibration
Data processing	Interrupt generation and DMA requests
Low-power modes	Wait, Auto-off, and Power-down

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

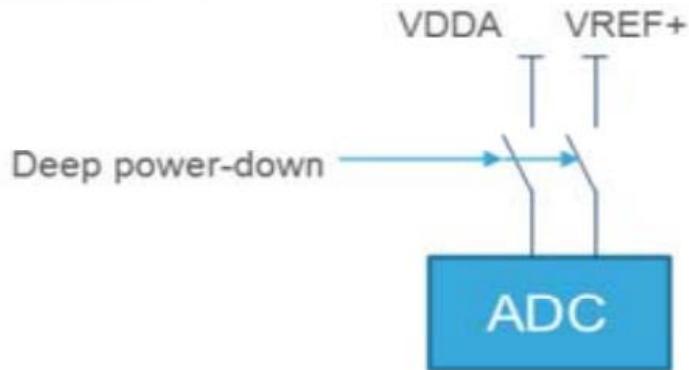
للمبدل التشابهي الرقمي في متحكمات STM32G0 المخطط الصندوقي التالي:

multiplexer

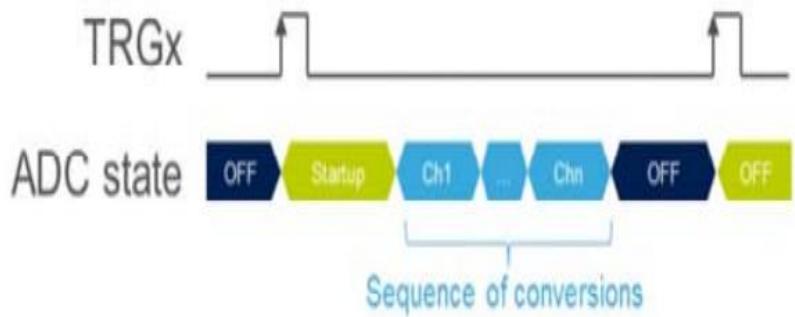


المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32



لاحظ في الجزء الأمامي لعملية التحويل يوجد مفتاح فصل/وصل الطاقة عن المبدل حيث يمكن فصل المبدل عن الطاقة عند عدم استخدامه



للمبدل ميزة إدارة الطاقة والتي تقوم بفصل الطاقة آلياً عن المبدل عند عدم وجود عملية تحويل ويتم إيقاظه أيضاً آلياً عند بدء عملية التحويل من خلال قدره عبر الكود أو من خلال الهايدروير ، وذلك عند تفعيل نمط **Auto-off mode**، ويتم إضافة زمن البدء **startup time** تلقائياً بين لحظة قدر المبدل وبدء عملية التحويل **258**

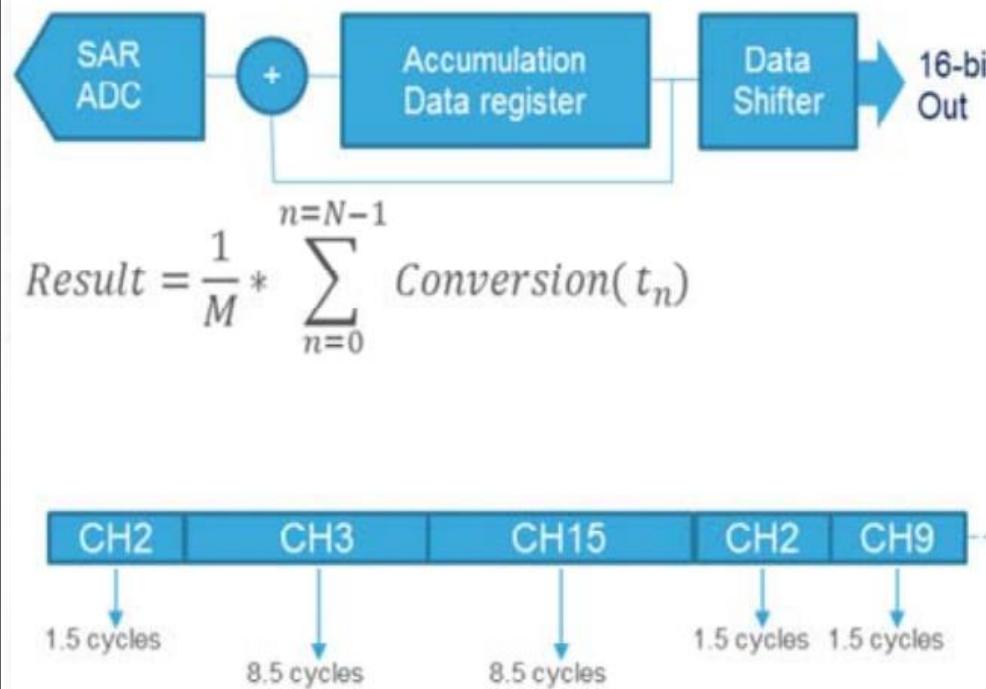
المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

يحتوي المبدل على وحدة أخذ عينات oversampling hardware والتي تقوم بتجميع العينات ثم تقوم بإعادة تقسيمها دون مساعدة الـ CPU، حيث بإمكانها تجميع من 2 حتى 256 عينة ثم إزاحتها نحو اليمين، مما يسمح للمستخدم

باستخدام:

- 19 قناة تحويل وبترتيب تصاعدي أو تنازلي
- 8 قنوات للتحويل بالترتيب الذي يقوم المستخدم بتعريفه
- يمكن ضبط زمن أخذ العينات على قيمتين و تخصيص القيمة المناسبة لكل قناة من هاتين



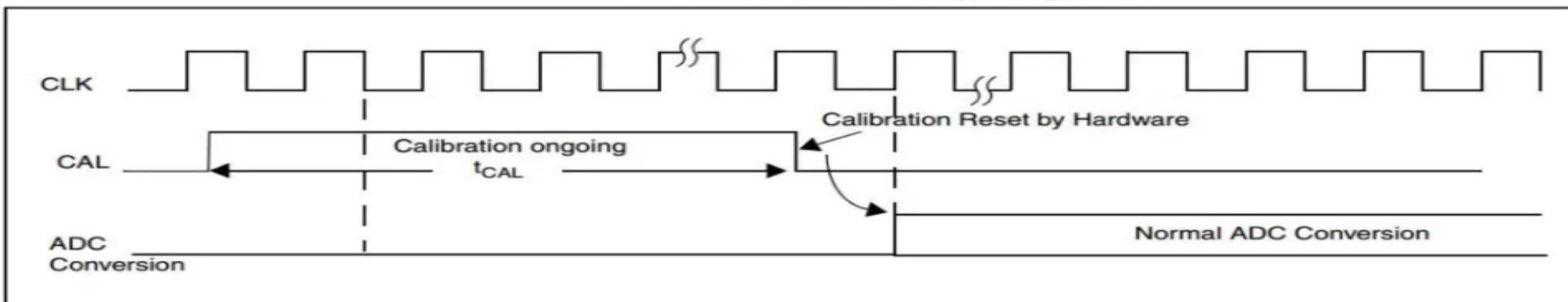
العنات على قيمتين و تخصيص القيمة المناسبة لكل قناة من هاتين

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

المعايرة الذاتية:

يوفر المبدل خاصية المعايرة الذاتية والتي تقلل بشكل كبير الأخطاء الناتجة عن تغيرات مكثف الشحن الداخلي internal capacitor يوضح المخطط الزمني التالي المعايرة الذاتية:



توفر مكتبة HAL دالة ضمن الـ ADC APIs تقوم ببدء عملية المعايرة والذي يوصى بعملها في بداية الكود بعد تهيئة الـ ADC عند تشغيل النظام

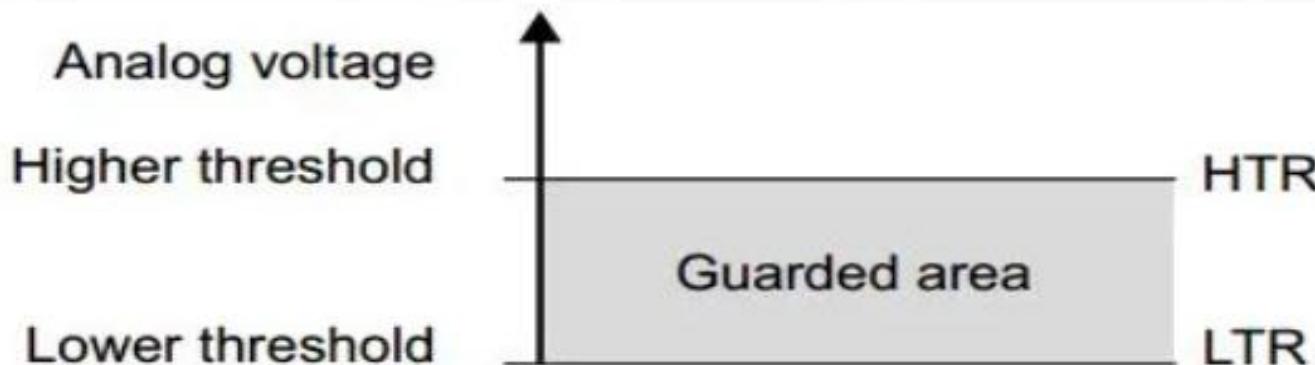
المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

المراقب الشابهي (AWD):

عندما يكون جهد الدخل التشابهي القادر على إحدى القنوات التشابهية أقل من العتبة الدنيا أو أكبر من العتبة العليا المسموح بها عندها يتم تفعيل بit الحالة الخاصة بالمراقب الشابهي AWD status bit

Analog watchdog guarded area



المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

:Sampling times

يتم أخذ عينات الجهد التشابهي خلال عدد معين من دورات الساعة، حيث أزمنة أخذ العينات المتوفرة 1.5cycles ، 3.5cycles ، 7.5cycles ، 12.5cycles ، 19.5cycles ، 39.5cycles ، 79.5cycles ، 160.5cycles و يتم تخزينها في المسجل SMP[2:0] bits و استخدامها لكل قناة من القنوات التشابهية، حيث يمكن أن يكون لكل قناة تشابهية معدل أخذ عينات مختلف.

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

:Sampling times 

Pinout & Configuration Clock Configuration Project Manager Tools

Software Packs Pinout

ADC1 Mode and Configuration

Mode

- IN0
- IN1
- IN2
- IN3
- IN4
- IN5
- IN6
- IN7
- IN8
- IN9
- Temperature Sensor Channel
- Vrefint Channel

EXTI Configuration

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

Search (Ctrl+F)

Discontinuous Conversion Mode

ADC_Regular_ConversionMode

- Enable Regular Conversions
- Number Of Conversion
- External Trigger Conversion Source

Rank

Channel

Sampling Time

ADC_Injected_ConversionMode

- Enable Injected Conversions

WatchDog

- Enable Analog WatchDog Mode

VBAT PC13... PC14... PC15... PDO-O... PD1-O... NRST VSSA VDDA PA0... PA1 PA2 TIM2_CH1 ADC1_IN1

LQFP48

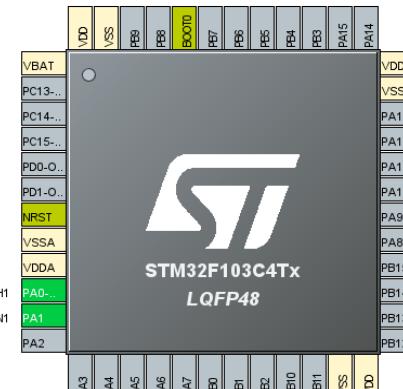
Pinout view System view

Activate Windows

Go to Settings to activate Windows.

Console Error Log Outline

Workspace Log



263

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

:Sampling times

يمكن حساب الزمن الكلي لعملية التحويل من خلال العلاقة التالية:

$$T_{conv} = Sampling\ time + 12.5\ cycles$$

مثال: من أجل ADCCLK=14MHZ، و زمن أخذ عينات يكون الزمن الكلي للتحويل: 1.5 cycles

$$T_{conv} = 1.5 + 12.5\ cycles = 14cycles = 1\mu sec$$

وبالتالي يكون تردد أخذ العينات:

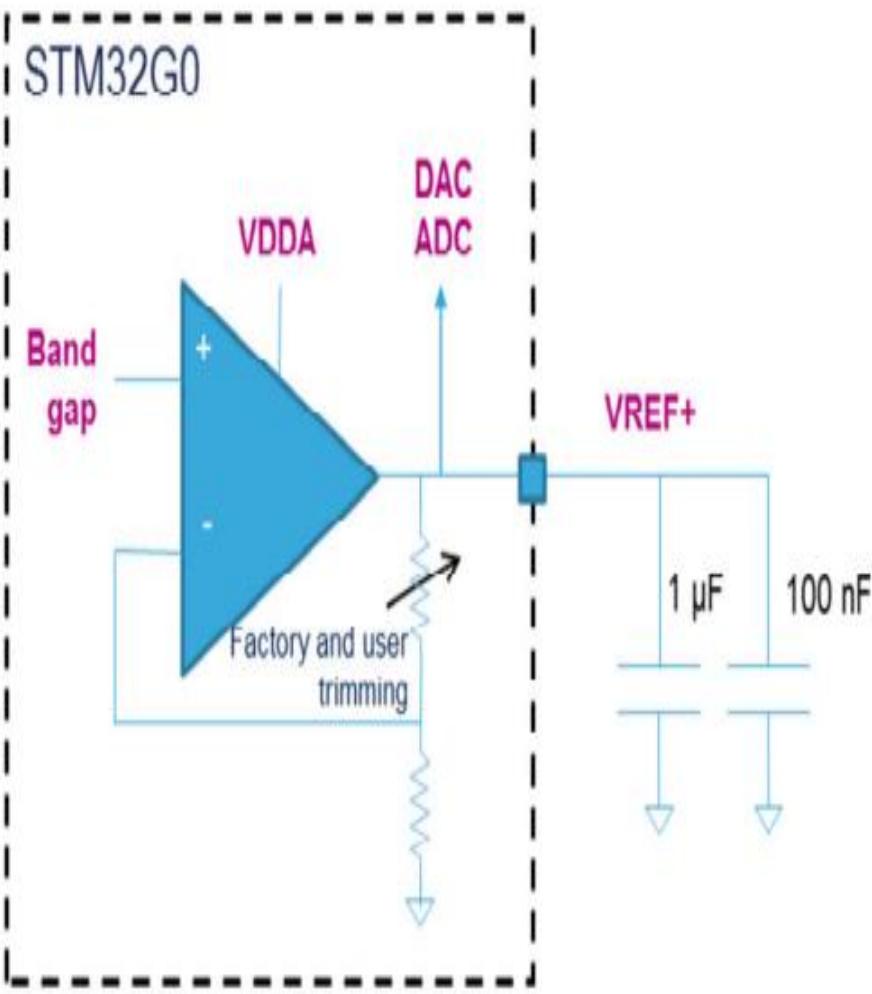
$$Sampling\ rate = 1/T_{conv}$$

ومن أجل المثال السابق يكون تردد أخذ العينات:

$$Sampling\ rate = 1000000 = 1Ms/ses$$

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32



□ **الجهد المرجعي للمبدل التشابهي:**
يوجد داخل متحكمات STM32 مولد جهد مرجعي مدمج بداخلها يقوم بتوليد جهد مرجعي ثابت ومستقر حتى عند تغذيته من بطارية و يمكن استخدامه مع المبدل التشابهي الرقمي ADC والمبدل الرقمي التشابهي DAC، و يعطي في خرجه قيمتين إما 2.5 V أو 2.048 V، كما يمكنه أن يغذي أحمال خارجية باستجرار تيار لا يتجاوز الـ 4 mA

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

□ الجهد المرجعي للمبدل التشابهي:

يمكن ضبط الجهد المرجعي للمبدل التشابهي الرقمي ليكون خارجي أو داخلي من خلال البات الموجودين في المسجل ENVR ، HIZ bits و على المستخدم أن ينتظر حتى يتم تفعيل البت VREFBUF_CSR والذي يعني أن الجهد المرجعي الخارجي قد وصل إلى القيمة المطلوبة VRR

ENVR	HIZ	Configuration
0	0	VREF buffer OFF VREF+ pin pulled-down to VSSA
0	1	External voltage reference mode (default): <ul style="list-style-type: none">• VREF buffer OFF• VREF+ pin floating
1	0	Internal voltage reference mode: <ul style="list-style-type: none">• VREF buffer ON• VREF+ pin connected to the VREF buffer output
1	1	Hold mode: <ul style="list-style-type: none">• VREF buffer ON• VREF+ pin floating. The voltage is held with an external capacitor

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

الجهد المرجعي للمبدل التشابهي:

عند استخدام مولد الجهد المرجعي الداخلي يجب وصل مكثفات على القطب V_{ref+} وفي هذه الحالة لن تحتاج لوصل دارة خارجية لتوليد الجهد المرجعي، كما يدعم مولد الجهد المرجعي الداخلي قيمتي جهد هي:

2.5V -

2.048V -

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

الجهد المرجعي للمبدل التشابهي:

يتم تعريف أقطاب الجهد المرجعي للمتحكم من خلال الـ Datasheet ويفترض أن يتم توصيلها بمجالات معينة من الجهد كما هو موضح بالجدول التالي:

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC, $2.4 \text{ V} \leq V_{REF+} \leq V_{DDA}$
$V_{DDA}^{(1)}$	Input, analog supply	Analog power supply equal to V_{DD} and $2.4 \text{ V} \leq V_{DDA} \leq 3.6 \text{ V}$
V_{REF-}	Input, analog reference negative	The lower/negative reference voltage for the ADC, $V_{REF-} = V_{SSA}$
$V_{SSA}^{(1)}$	Input, analog supply ground	Ground for analog power supply equal to V_{SS}

المبدلات التشابهية الرقمية في متحكمات STM32

ADC in STM32

الجهد المرجعي للمبدل التشابهي:

يتم حساب جهد الدخل التشابهي الناتج عن عملية التحويل من خلال العلاقة التالية:

$$V_{in} = \text{ADC_Res} * (\text{reference voltage} / 4096)$$

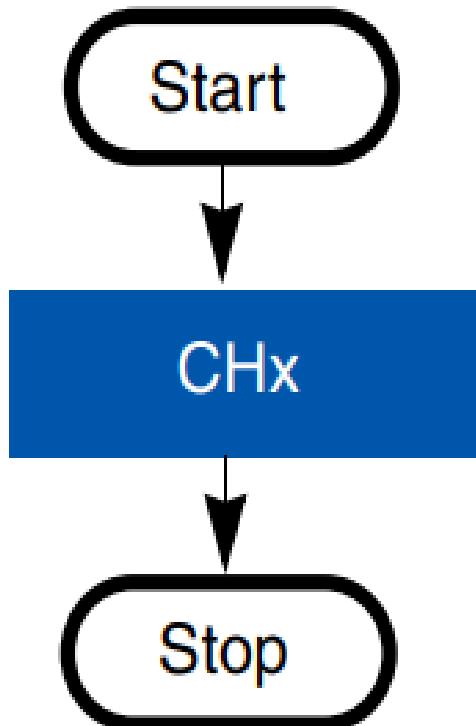
حيث :

$$\text{reference voltage} = (V_{ref+}) - (V_{ref-})$$

أنماط عمليات التحويل ADC conversion modes

يمكن للمبدل التشابهـي الرقمـي أن يعـمل بعـدة أنـماط عـمل هي :

□ **Single-channel, single conversion mode**: وهو نـمط العـمل الأـسـهل والأـبـسـط مـقارـنةً مع بـقـية الـأـنـماـط، حيث يـقـوم الـADC بـعـملـيـة تحـوـيل وـحـيدـة (وـعـملـيـة أـخـذ عـيـنـات وـحـيدـة) ولـقـاتـة وـاحـدـة فـقـط ويـتـوقف عن الـعـمل بـمـجـرـد اـنـتـهـائـه من عـملـيـة التـحـوـيل

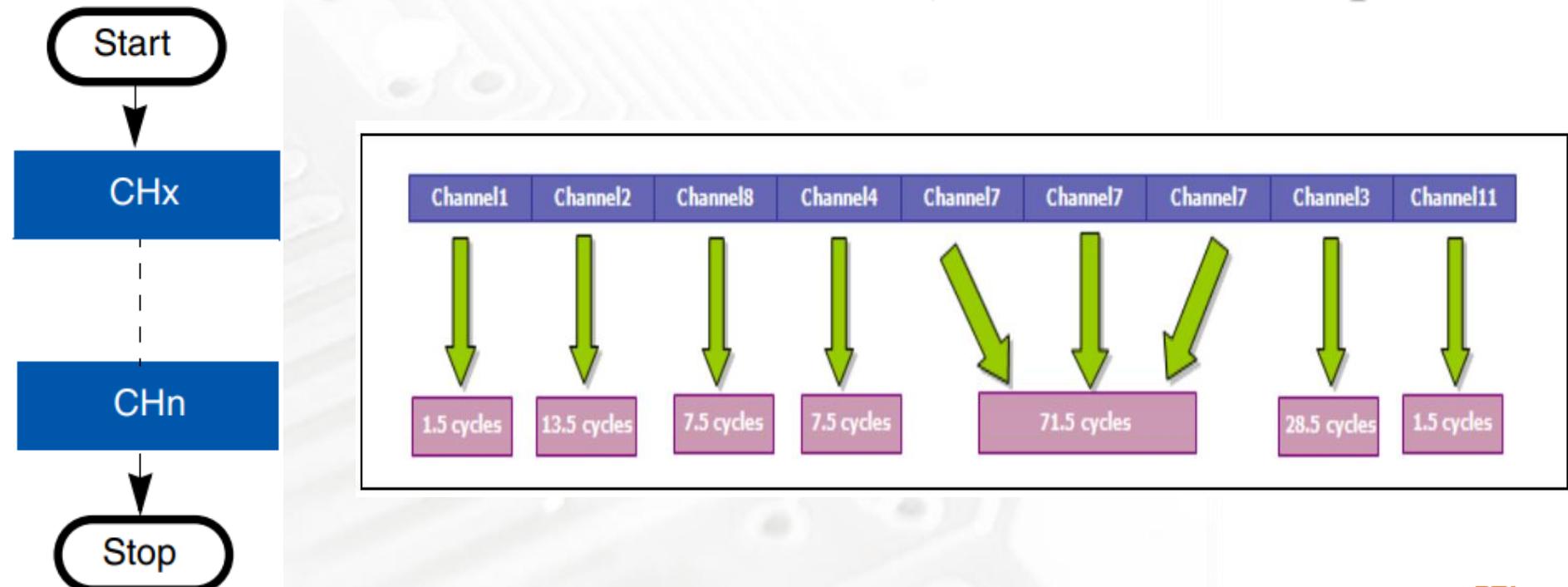


أنماط عمليات التحويل ADC conversion modes

يمكن للمبدل التشابهى الرقمي أن يعمل بعدة أنماط عمل هي :

:Multichannel (scan) single conversion mode

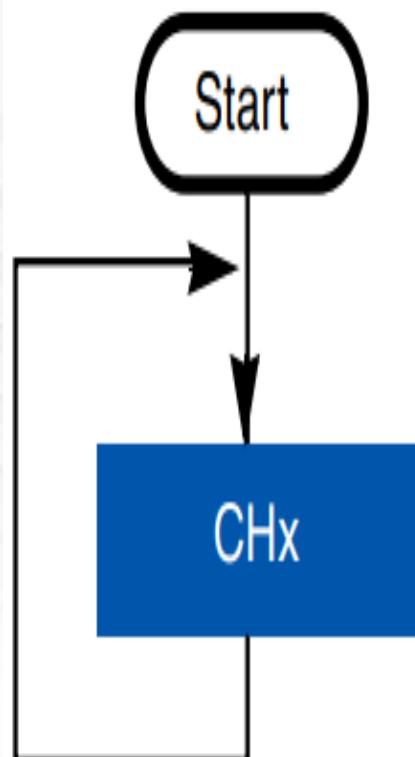
يستخدم هذا النمط من العمل لتحويل الجهود القادمة من عدة قنوات تشابهية ولمرة واحدة فقط، ويمكن اختيار ترتيب القنوات الذي ترغب به من خلال و باستخدام زمن أخذ عينات مختلف لكل قناة من القنوات **ADC sequencer**



أنماط عمليات التحويل ADC conversion modes

يمكن للمبدل التشابهـي الرقمـي أن يعـمل بعـدة أنـماط عـمل هـي :

□ **Single-channel continuous conversion mode**: في هذا النـمـط من العـمل يـتم تحـويل الـقيـمة التـشاـبـهـيـة الـقادـمـة عـلـى قـنـاة وـاحـدة إـلـى قـيـمة رـقـمـيـة وـتـعـاد عـمـلـيـة التـحـولـيـل بـشـكـل مـسـتـمر

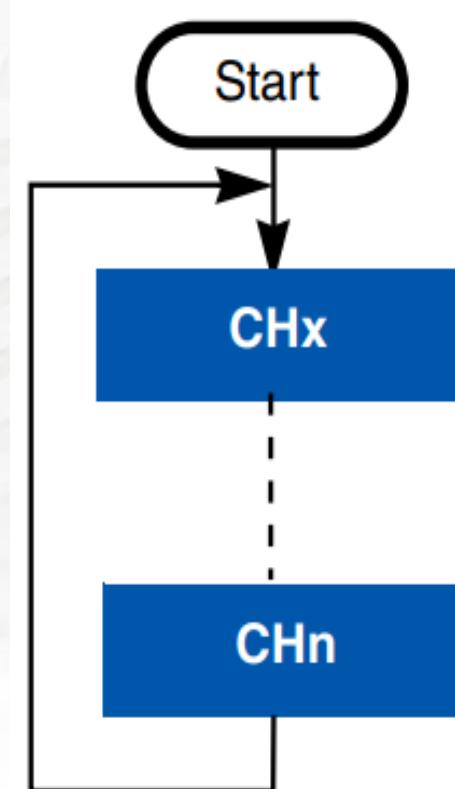


أنماط عمليات التحويل ADC conversion modes

يمكن للمبدل التشابهى الرقمي أن يعمل بعدة أنماط عمل هي :

:**Multichannel (scan) , continuous conversion mode**

يستخدم هذا النمط من العمل لتحويل الجهود القادمة من عدة قنوات تشابهية إلى قيم رقمية ويعيد العملية بشكل مستمر وذلك تبعاً لترتيب معين للقنوات



طرق قراءة المبدل التشابهي الرقمي ADC conversion modes

يوجد ثلاث طرق رئيسية لقراءة الـ ADC هي:

Polling Method: تعتبر الطريقة الأسهل في كتابة الكود لقراءة القيمة القادمة من إحدى القنوات التشابهية، ولكنها ليست الأكثر فعالية ، حيث علينا أن نبدأ بعملية التحويل وتتوقف الـ CPU عن تنفيذ الكود وتنظر لحين الانتهاء من عملية التحويل حينها يمكن للـ CPU استكمال تنفيذ الكود الرئيسي

The interrupt Method: تعتبر هذه الطريقة طريقة فعالة لاستخدام المبدل التشابهي الرقمي ، حيث يقوم بقدر المبدل فقط و يمكن للـ CPU أن تستكمل تنفيذ الكود والمهام المطلوبة منها لحين انتهاء عملية التحويل عنها سيقوم المبدل بطلب مقاطعة وستتوجه الـ CPU لبرنامج خدمة المقاطعة

DMA Method: وهي الطريقة الأكثر فعالية في استخدام المبدل التشابهي الرقمي

دوال مكتبة HAL المستخدمة للتعامل مع المبدل التشابهي الرقمي في وضع الـ Polling

- لبدء المعايرة الذاتية للمبدل التشابهي الرقمي :adc1

HAL_ADCEx_Calibration_Start(&hadc1);

- إعطاء إشارة البدء للمبدل التشابهي الرقمي ليكون جاهز فيما بعد لإجراء عمليات التحويل

HAL_ADC_Start(&hadc1);

- لطلب عملية تحويل من المبدل التشابهي الرقمي

HAL_ADC_PollForConversion(&hadc1, 1);

- إسناد ناتج عملية التحويل إلى المتغير **AD_RES**

AD_RES = HAL_ADC_GetValue(&hadc1);

دوال مكتبة HAL المستخدمة للتعامل مع المبدل التشابهي الرقمي في وضع الـ Interrupt

- لبدء المعايرة الذاتية للمبدل التشابهي الرقمي :adc1

HAL_ADCEx_Calibration_Start(&hadc1);

- إعطاء إشارة البدء للمبدل التشابهي الرقمي ليكون جاهز فيما بعد لإجراء عمليات التحويل

HAL_ADC_Start_IT(&hadc1);

- إسناد ناتج عملية التحويل إلى المتغير AD_RES

AD_RES = HAL_ADC_GetValue(&hadc1);

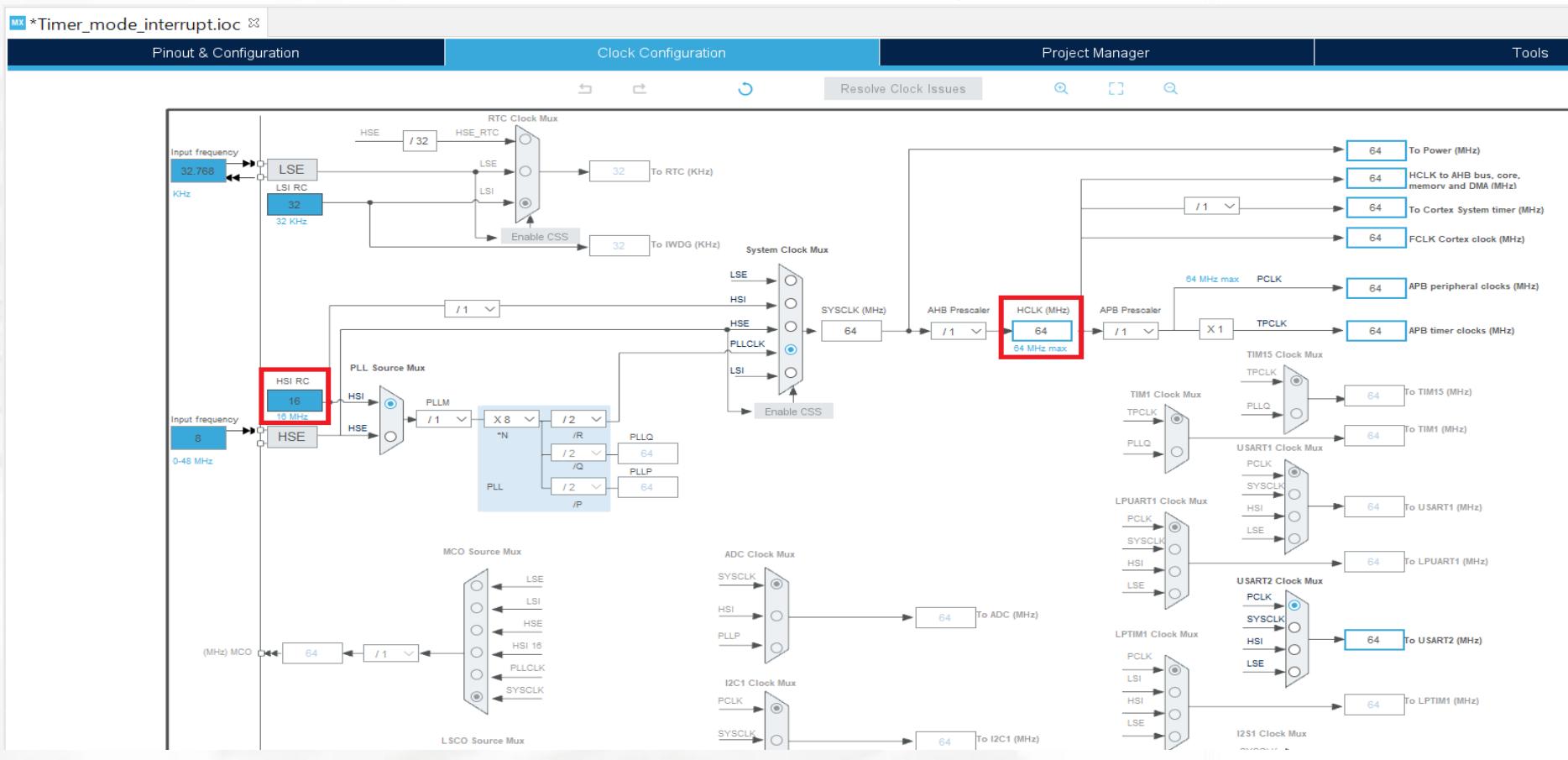
التطبيق 1: التحكم بشده إصاءه ليد موصول على أحد أقطاب الـ PWM من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط Polling

سنقوم بتنفيذ المشروع وفقاً للتسلسل التالي:

- ضبط تردد ساعة المتحكم
- ضبط قطب الدخل التشابهي CH7 في نمط التحويل لمرة واحدة Single حيث سرّبط مقاومة متغيرة معه.
- ضبط الـ timer2 في نمط PWM على القناة CH1 وسرّبط معه ليد.

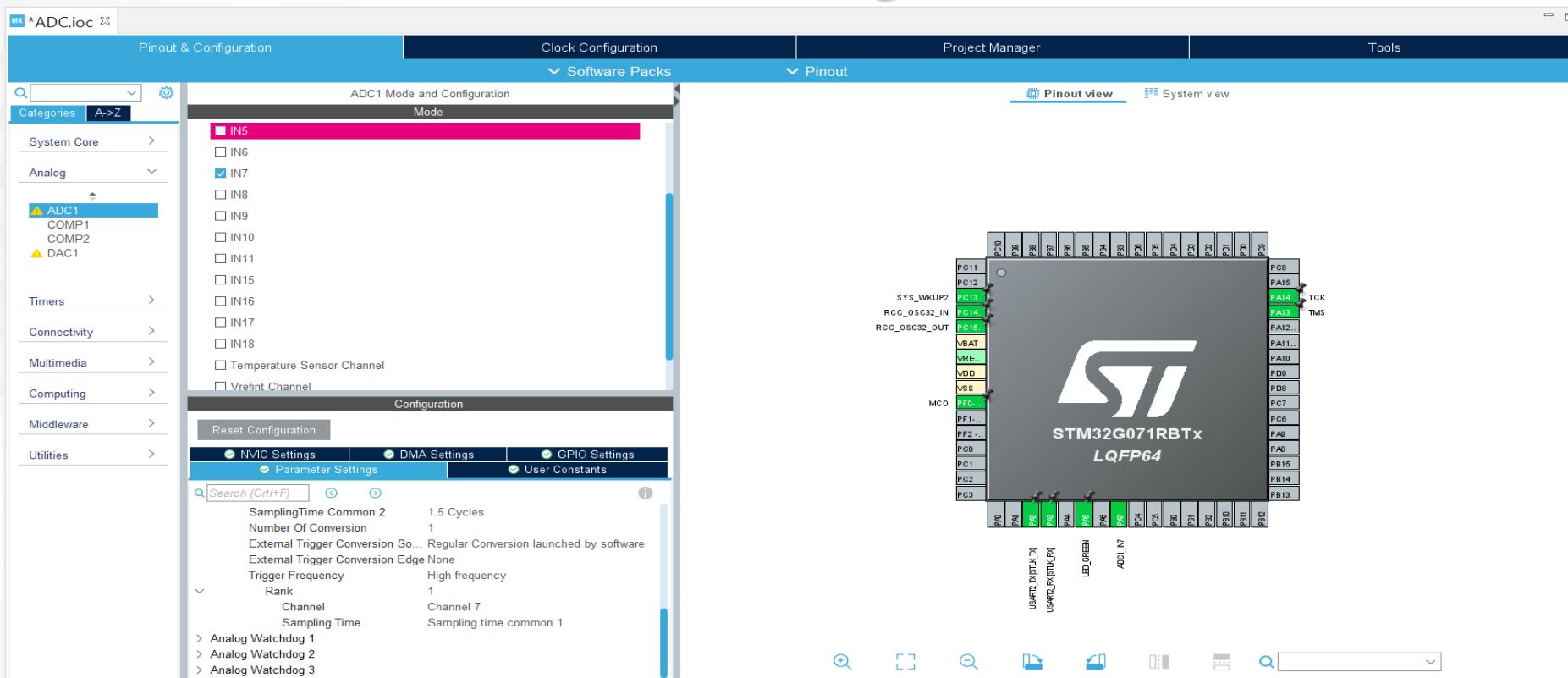
التطبيق 1: التحكم بشدة إصاءه ليد موصول على أحد أقطاب الـ PWM من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل الشابهي باستخدام نمط Polling

ضبط تردد ساعة المتحكم



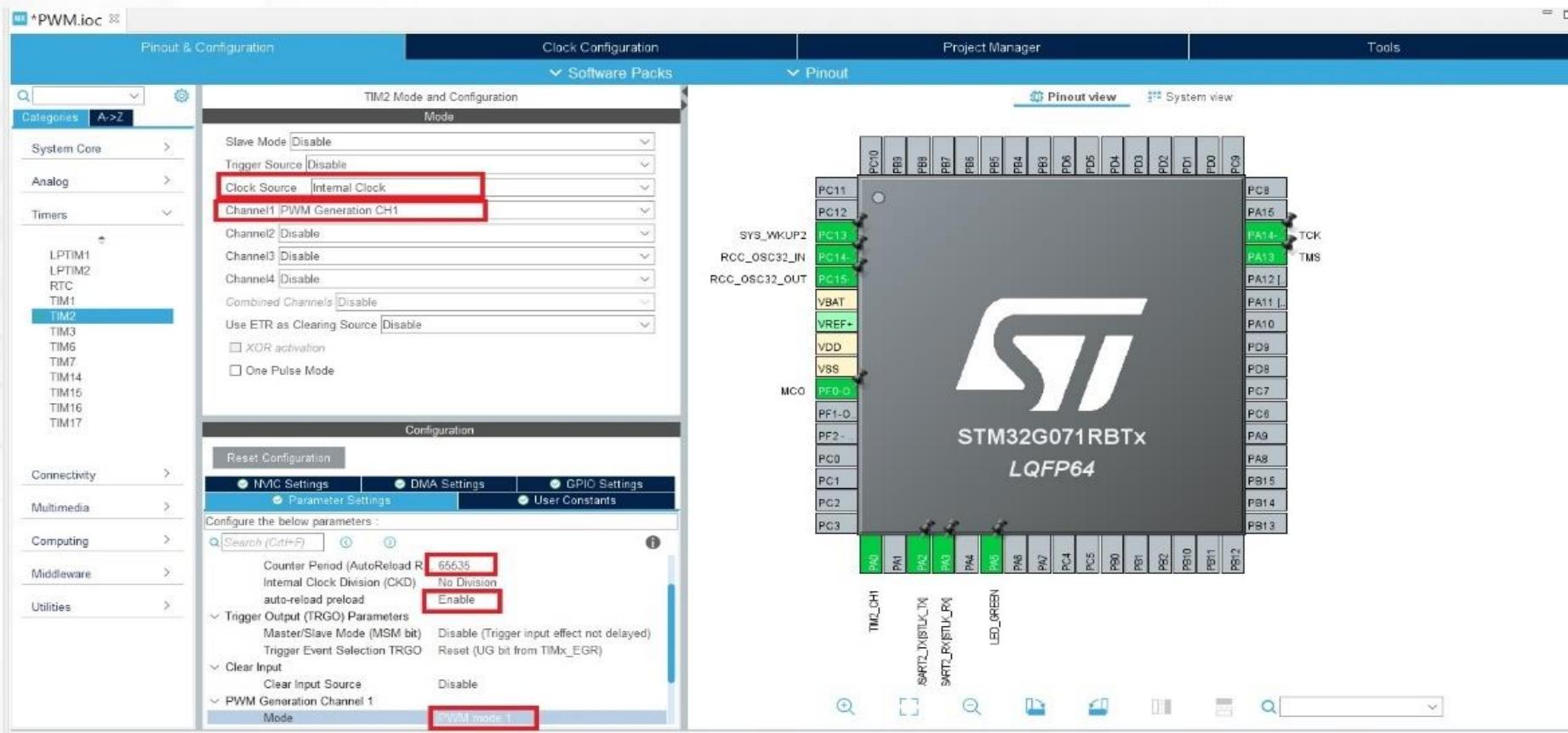
التطبيق 1: التحكم بشده إصاءه ليد موصول على أحد أقطاب الـ PWM من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط Polling

- ضبط قطب الدخل التشابهي CH7 في نمط التحويل لمرة واحدة حيث سرّبط مقاومة متغيرة معه.



التطبيق 1: التحكم بشدة إصاءه ليد موصول على أحد أقطاب الـ PWM من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل الشابهي باستخدام نمط Polling

- ضبط الـ timer2 في نمط PWM على القناة CH1 وسربط معه ليد.



التطبيق 1: التحكم بشده إصاءه ليد موصول على أحد أقطاب آلة PWM من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط Polling

```
#include "main.h"
```

```
ADC_HandleTypeDef hadc1;
```

```
TIM_HandleTypeDef htim2;
```

```
void SystemClock_Config(void);
```

```
static void MX_GPIO_Init(void);
```

```
static void MX_ADC1_Init(void);
```

```
static void MX_TIM2_Init(void);
```

التطبيق 1: التحكم بشده إصاءه ليد موصول على أحد أقطاب آلة PWM من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط Polling

```
int main(void)
{
    uint16_t AD_RES = 0;

    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    MX_ADC1_Init();
    MX_TIM2_Init();
```

التطبيق 1: التحكم بـشدة إصاءه لـيد موصول على أحد أقطاب آلة PWM من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط Polling

```
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);  
// Calibrate The ADC On Power-Up For Better  
Accuracy
```

```
HAL_ADCEx_Calibration_Start(&hadc1);  
while (1)  
{  
    // Start ADC Conversion  
    HAL_ADC_Start(&hadc1);  
    // Poll ADC1 Peripheral & TimeOut = 1mSec  
    HAL_ADC_PollForConversion(&hadc1, 1);
```

التطبيق 1: التحكم بـشدة إصاءه لـيد موصول على أحد أقطاب آلة PWM من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط Polling

```
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);  
// Read The ADC Conversion Result & Map It To PWM  
DutyCycle
```

```
AD_RES = HAL_ADC_GetValue(&hadc1);
```

```
TIM2->CCR1 = (AD_RES<<4);
```

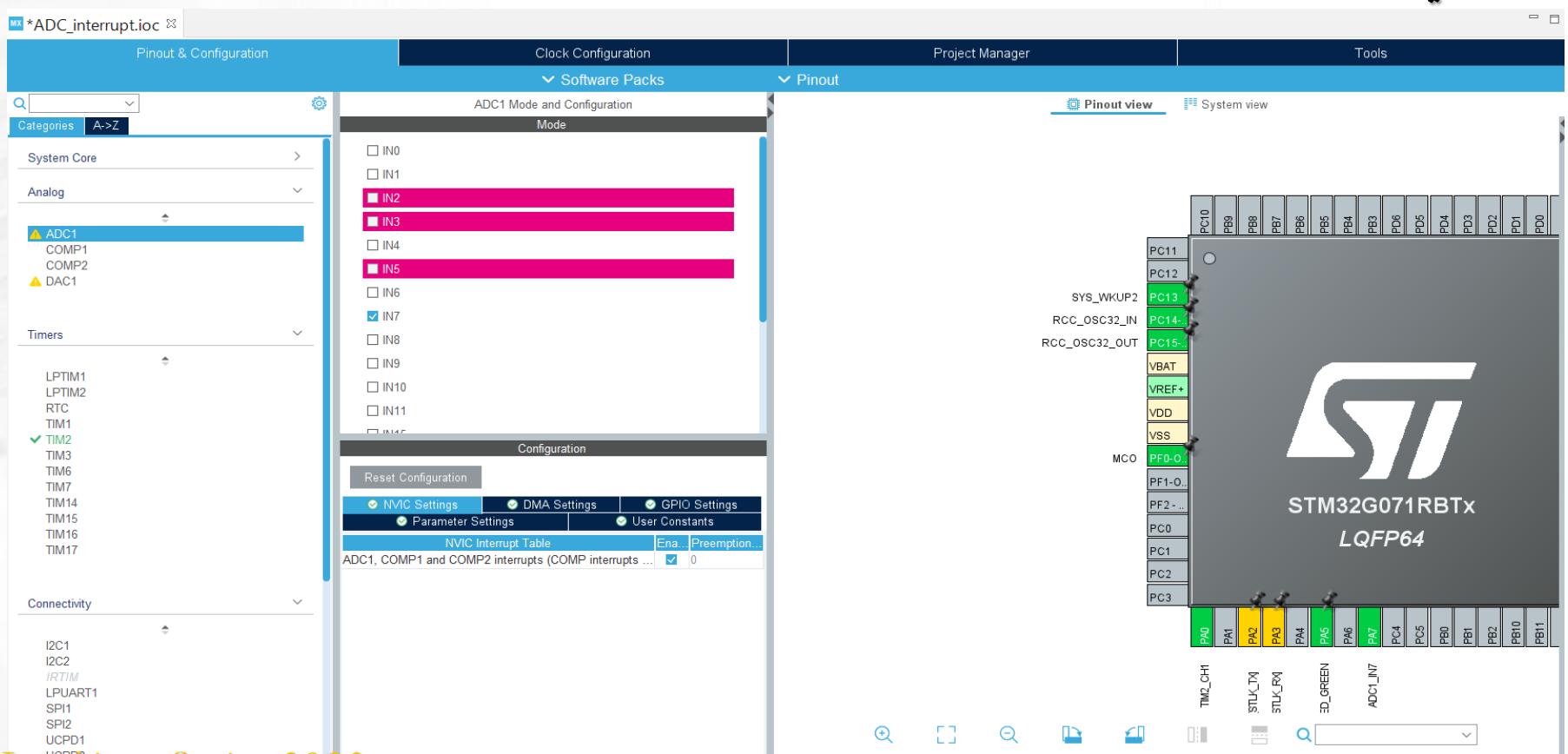
```
HAL_Delay(1);
```

```
}
```

```
}
```

التطبيق 2: التحكم بشده إصاءه ليد موصول على أحد اقطاب الـ PWM من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل الشابهي باستخدام نمط Interrupt

سنقوم بإعادة الإعدادات السابقة بالإضافة إلى تفعيل مقاطعة المبدل الشابهي الرقمي



التطبيق 2: التحكم بـشدة إصاءه لـPin موصول على أحد أقطاب AD من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل PWM التشابهي باستخدام نمط Interrupt

```
#include "main.h"

uint16_t AD_RES = 0;
ADC_HandleTypeDef hadc1;
TIM_HandleTypeDef htim2;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_TIM2_Init(void);
```

التطبيق 2: التحكم بـشدة إصاءه لـLED موصول على أحد أقطاب AD من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل PWM التشابهي باستخدام نمط Interrupt

```
int main(void)
{
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    MX_ADC1_Init();
    MX_TIM2_Init();
    HAL_TIM_PWM_Start(&htim2,
TIM_CHANNEL_1);
```

التطبيق 2: التحكم بـشدة إصاءه لـLED موصول على أحد أقطاب AD من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل

التشابهي باستخدام نمط Interrupt

```
HAL_ADCEx_Calibration_Start(&hadc1);  
while (1)  
{  
    // Start ADC Conversion  
    HAL_ADC_Start_IT(&hadc1);  
    // Update The PWM Duty Cycle With Latest ADC  
    // Conversion Result  
    TIM2->CCR1 = (AD_RES<<4);  
    HAL_Delay(1);  
}  
}
```

التطبيق 2: التحكم بـشدة إصاءه لـيد موصول على أحد أقطاب آلة PWM من خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل الشابهي باستخدام نمط Interrupt

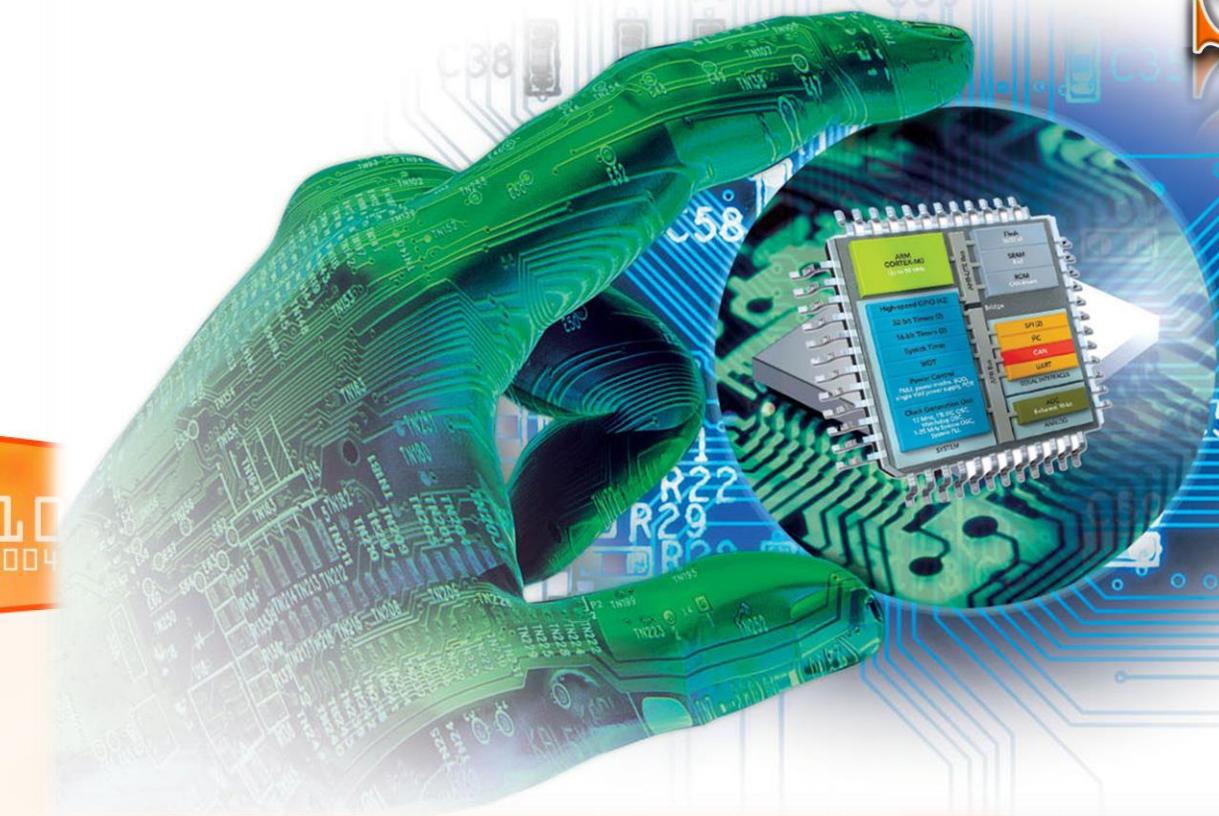
```
Void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef*  
hadc)  
{  
    // Read & Update The ADC Result  
    AD_RES = HAL_ADC_GetValue(&hadc1);  
}
```

Thank you for listening

مُتَحَكِّمَات

STM32

7



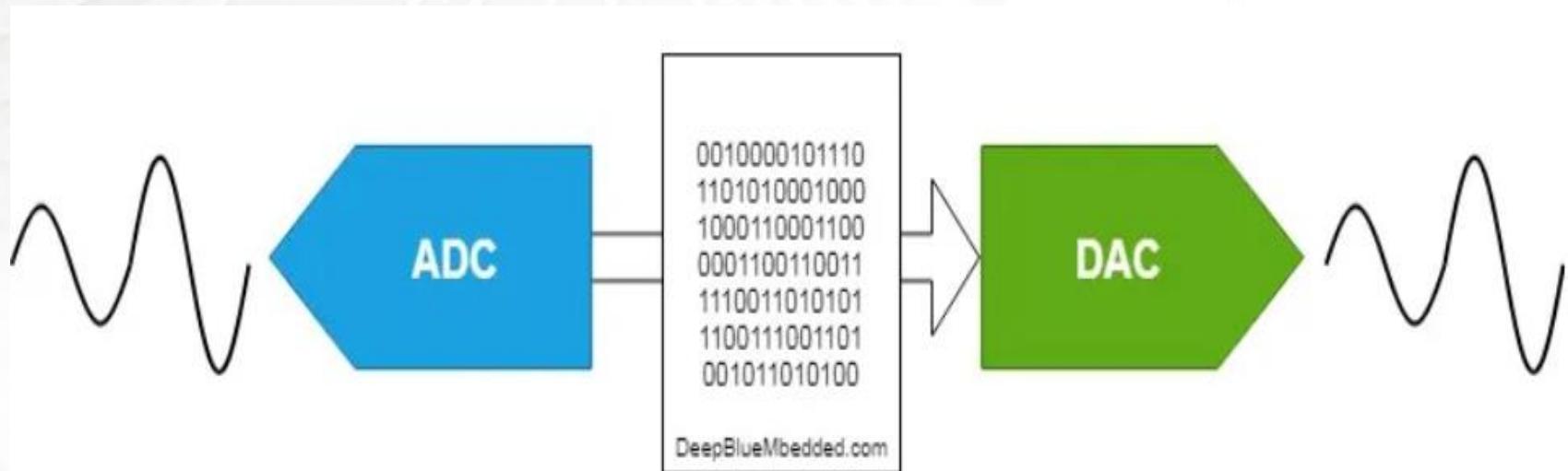
موضو عات المعاصرة:

- المبدلات الرقمية التشابهية في متحكمات STM32
- DAC with output buffer
- DAC data formats – For single channel mode
- DAC data formats – For dual channel mode
- طرق قذح المبدل DAC conversion triggers
- توليد الإشارات المختلفة باستخدام الـ DAC
- ماهي وحدة إدارة الوصول المباشر للذاكرة DMA
- مواصفات وحدة الـ DMA في متحكمات stm32G0

المبدلات الرقمية التشابهية في متحكمات STM32

DAC in STM32

- المبدلات الرقمية التشابهية عبارة عن دارات الكترونية تقوم بتحويل القيم الرقمية على دخلها إلى جهد تشابهي مقابل لها
- فبينما يقوم المبدل التشابهي الرقمي ADC بتحويل الجهد التشابهي إلى بيانات رقمية ، يقوم المبدل الرقمي التشابهي DAC بالعملية العكسية فيقوم بتحويل القيم الرقمية إلى الجهد التشابهي مقابل لها



المبدلات الرقمية التشابهية في متحكمات STM32

DAC in STM32

- لا تحتوي جميع متحكمات stm32 على مبدل رقمي تشابهي بداخلها ، وفي هذه الحالة يتم وصل مبدل رقمي تشابهي خارجي مع المتحكم أو يتم استخدام تقنية PWM-To-DAC conversion
- تحتوي متحكمات stm32G0 على مبدل رقمي تشابهي وحيد بدقة 12bit ، ويمكن ضبطه ليعمل في نمط 8bit أو في نمط 12bit للمبدل التشابهي الرقمي قناتي خرج يمكن أن تعملان بالتزامن أو بشكل مستقل.
- يمكن استخدام قناتي المبدل بنمط buffered/unbuffered modes
- يمكن استخدام قطب خرج المبدل DAC_OUTx كقطب GPIO عند عدم استخدام المبدل

المبدلات الرقمية التشابهية في متحكمات STM32

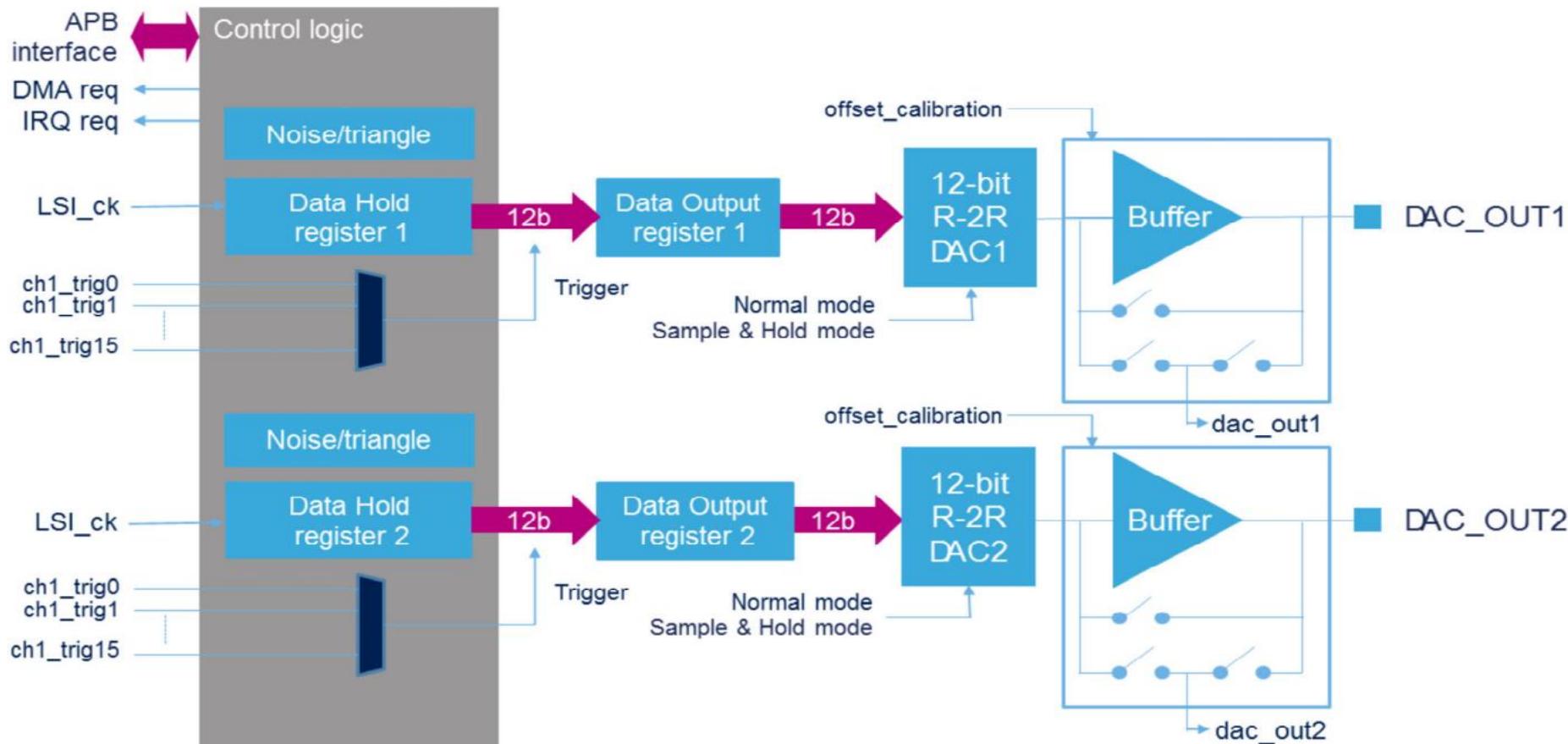
DAC in STM32

- يمكن استخدام قطب خرج المبدل DAC لقيادة الأحمال الخارجية
- بإمكان المبدل التشابهي الرقمي توليد إشارة ضجيج أو إشارة مثلثية
- جهد الدخل المرجعي للمبدل هو V_{ref+}
- يمكن قذح عملية التبديل بعدة طرق إما من خلال قذح خارجي أو داخلي من خلال مؤقت على سبيل المثال
- يمكن استخدام المبدل الرقمي التشابهي بنمط ال DMA
- يمكن استخدام المبدل الرقمي التشابهي بنمط ال sample and hold لتوفير استهلاك الطاقة عند عمل المتحكم في نمط stop mode

المبدلات الرقمية التشابهية في متحكمات STM32

DAC in STM32

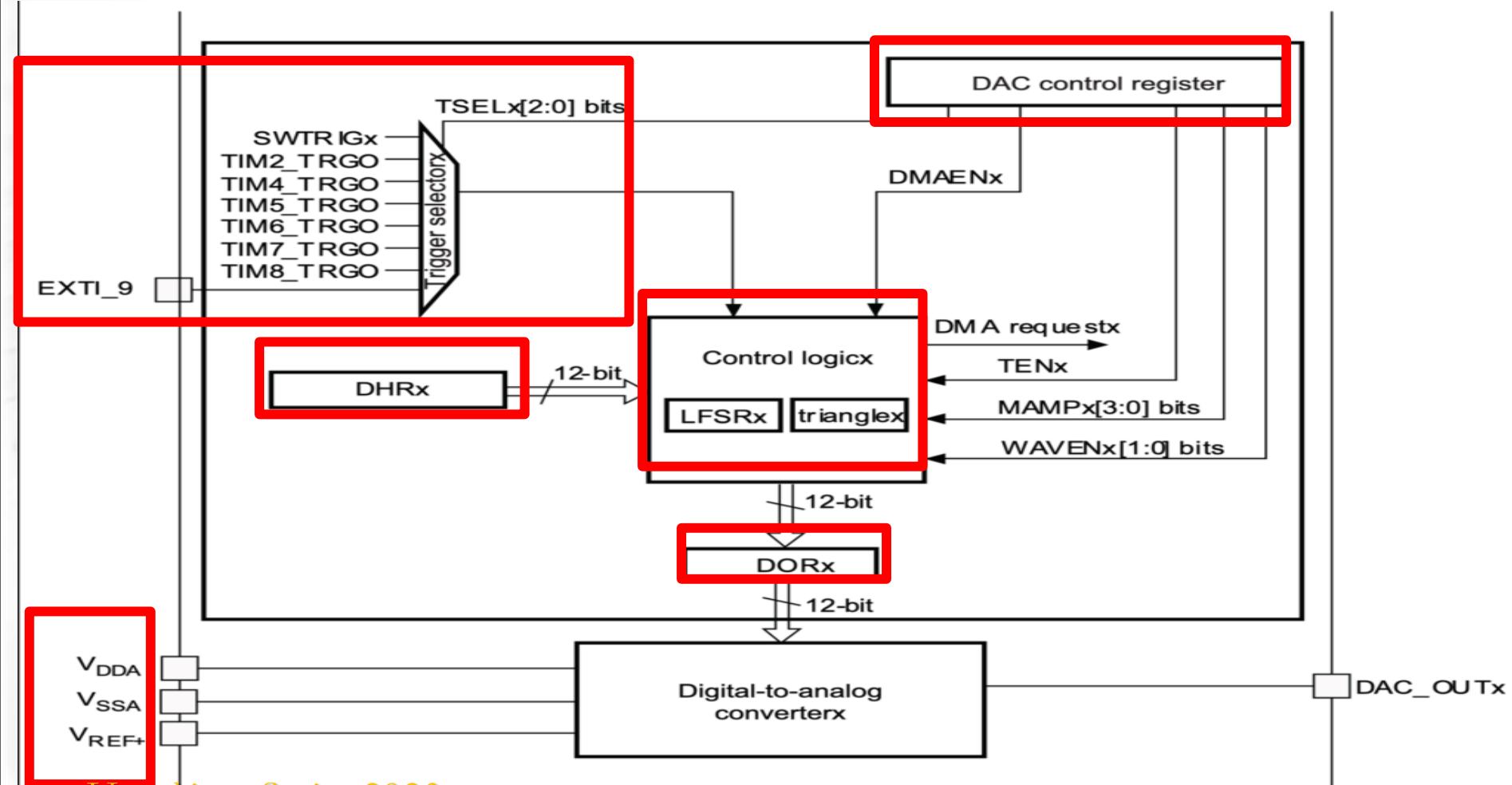
للمبدل الرقمي التشابهي المخطط الصندوقي التالي:



المبدلات الرقمية التشابهية في متحكمات STM32

DAC in STM32

كل قناة تشابهية لها المخطط الصندوقي التالي:



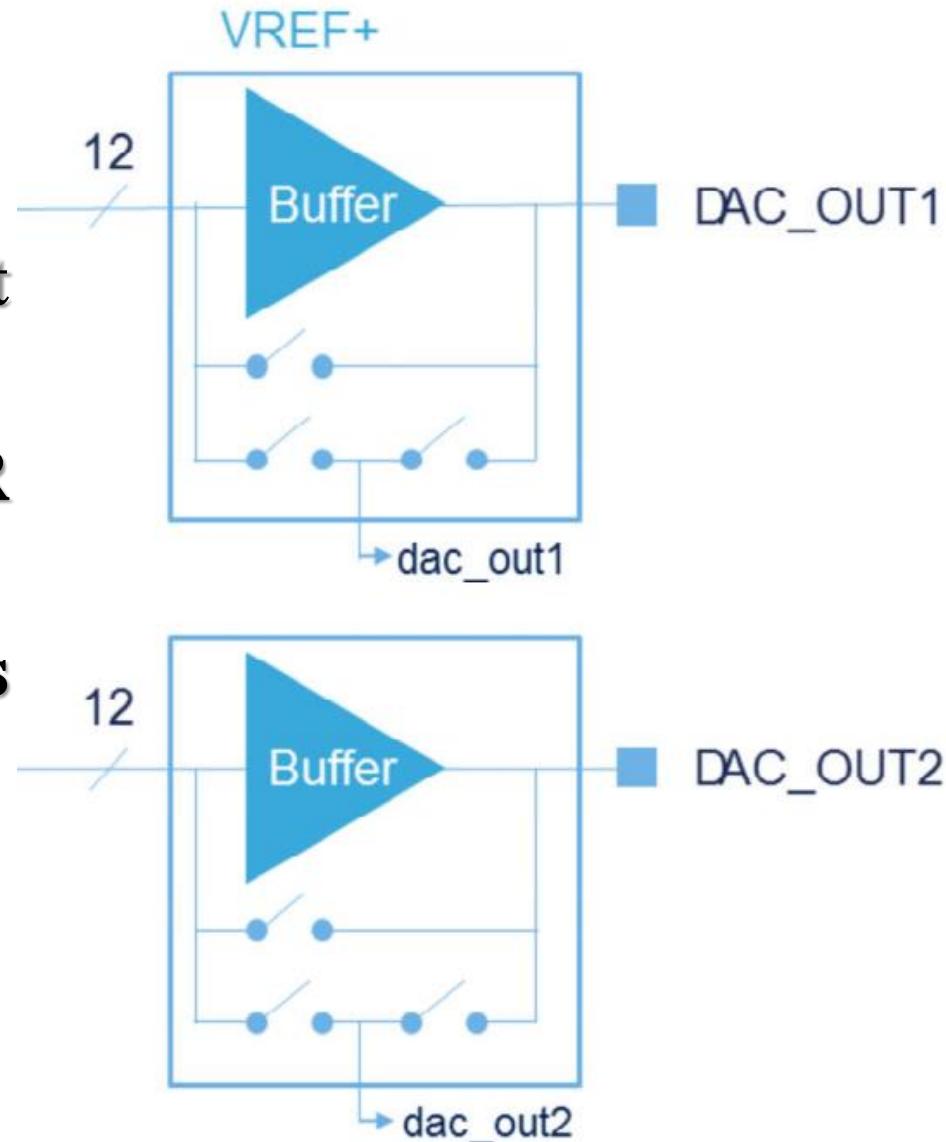
المبدلات الرقمية تشابهية في متحكمات STM32

DAC in STM32

- يمكن تفعيل كل قناة من قنوات خرج المبدل بشكل منفصل من خلال بت التفعيل الخاص بها الموجود في المسجل **DAC_CR**
- يمكن وصل قناة خرج المبدل مع إحدى طرفيات المتحكم مثل المقارن أو المضخم العملياتي أو الـ **ADC**
- يجب معايرة جهد الانزياح للمبدل ، ويتم ذلك من قبل المصنع عند كل Reset للمتحكم ، كما يمكن معايرته برمجياً أثناء العمل.

DAC with output buffer

- ❑ Low impedance output using Buffered mode
- ❑ Row output from R-2R type resistor ladder DAC
- ❑ Output impedance is ~12Kohm



DAC data formats – For single channel mode

:8-bit نمط

:Right aligned data input



:12-bit نمط

:Right aligned data input



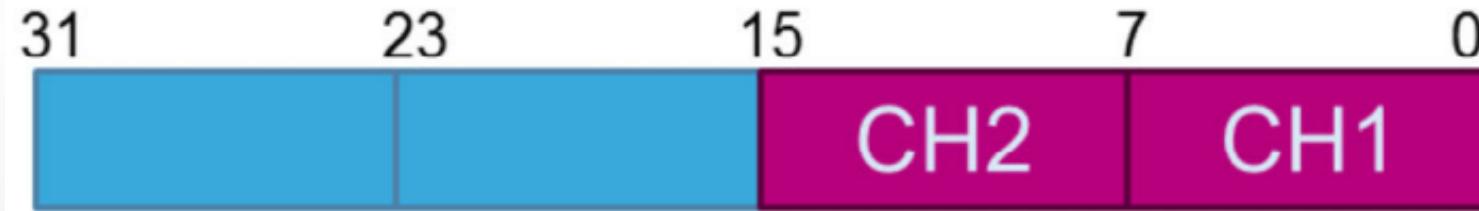
:Left aligned data input



DAC data formats – For dual channel mode

:8-bit نمط

:Right aligned data input



:12-bit نمط

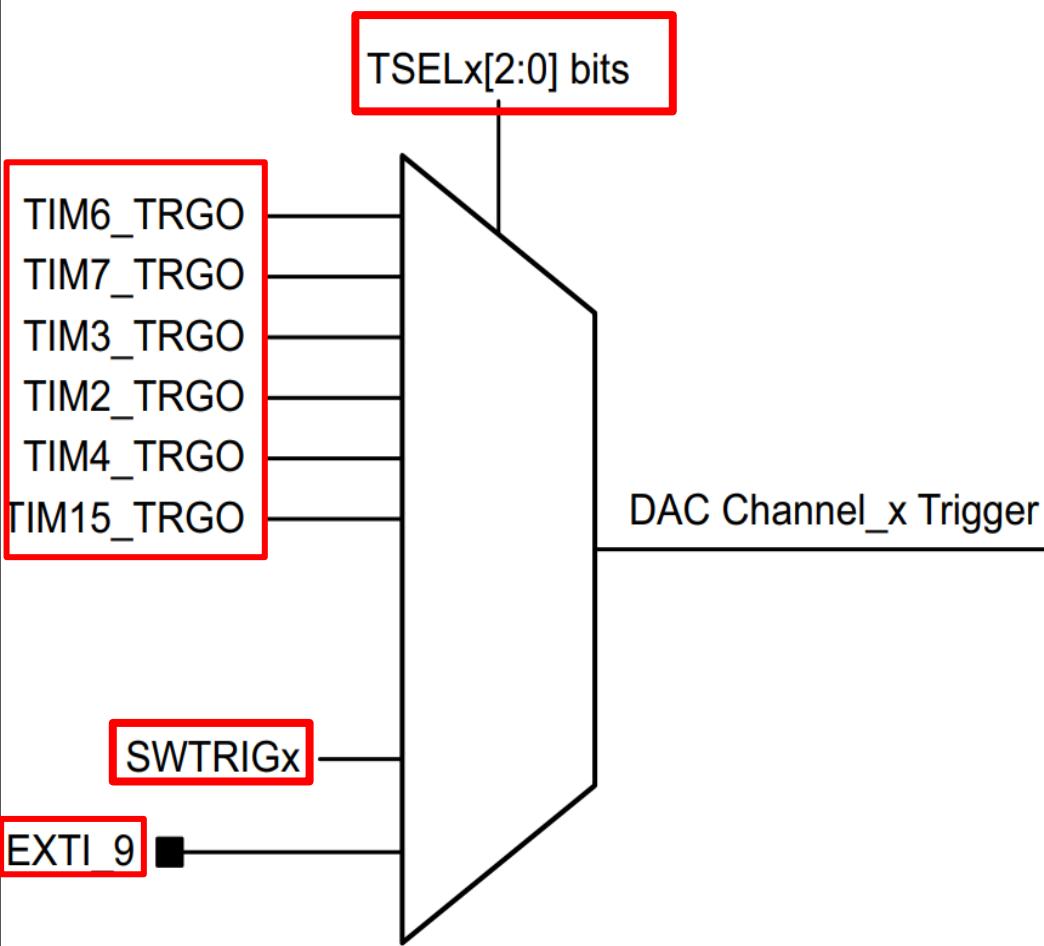
:Right aligned data input



:Left aligned data input



طرق قدح المبدل DAC conversion triggers طرق قدح المبدل



هناك عدة طرق لقدح المبدل
الرقمي التشابهي : DAC :

برمجياً من خلال كتابة

البيانات إلى المسجل

DAC_DHRx

باستخدام أحد مخارج

المؤقتات

خارجياً من خلال أحد

I/O أقطاب

من خلال ضبط بت القدح

برمجياً SWTR

طرق قدح المبدل DAC conversion triggers

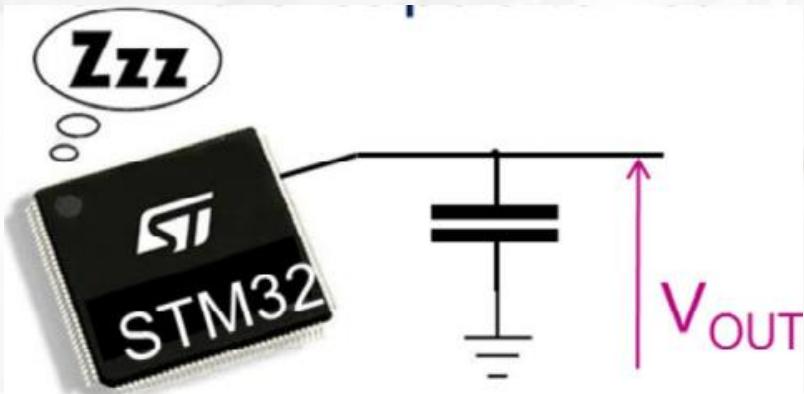
Source	Type	TSEL[2:0]
Timer 6 TRGO event		000
Timer 3 TRGO event		001
Timer 7 TRGO event		010
Timer 5 or Timer 15 TRGO event	Internal signal from on-chip timers	011
Timer 2 TRGO event		100
Timer 4 TRGO event		101
EXTI line9	External pin	110
SWTRIG	Software control bit	111

Sample and hold mode

الغاية من نمط العمل sample and hold mode هو القدرة على الحصول على جهد خرج المبدل الذي تم تحويله ، وذلك عندما يكون المتحكم في نمط الطاقة المنخفضة على سبيل المثال في وضع التوقف stop mode

عندما يتم ضبط المبدل ليعمل في نمط ال sample and hold mode ، يكون المبدل قادر على إخراج جهد الخرج الذي تم تحويله حتى عندما تكون جميع دارات المبدل الرقمية والتشابهية في حالة فصل .

حيث يتم وصل مكثف داخلي أو خارجي على خرج المبدل



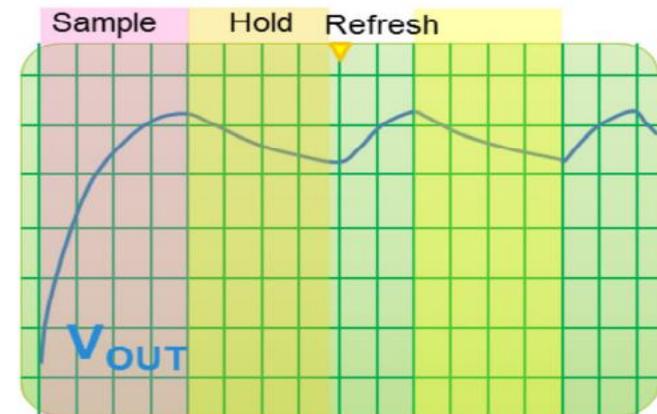
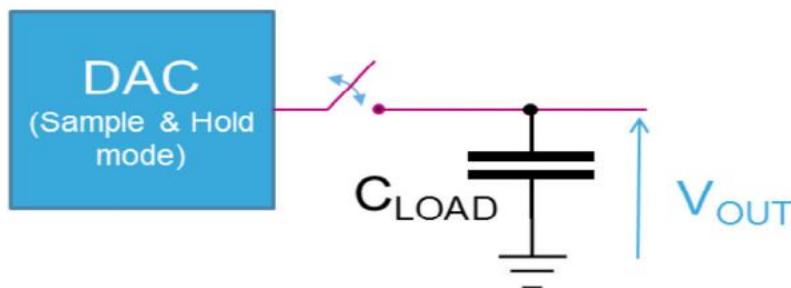
Sample and hold mode

تمر عملية التحويل خلال نمط sample and hold mode بثلاث مراحل:

Sampling phase: مرحلة أخذ العينات في هذه المرحلة يتم شحن المكثف بقيمة الجهد المرغوبة

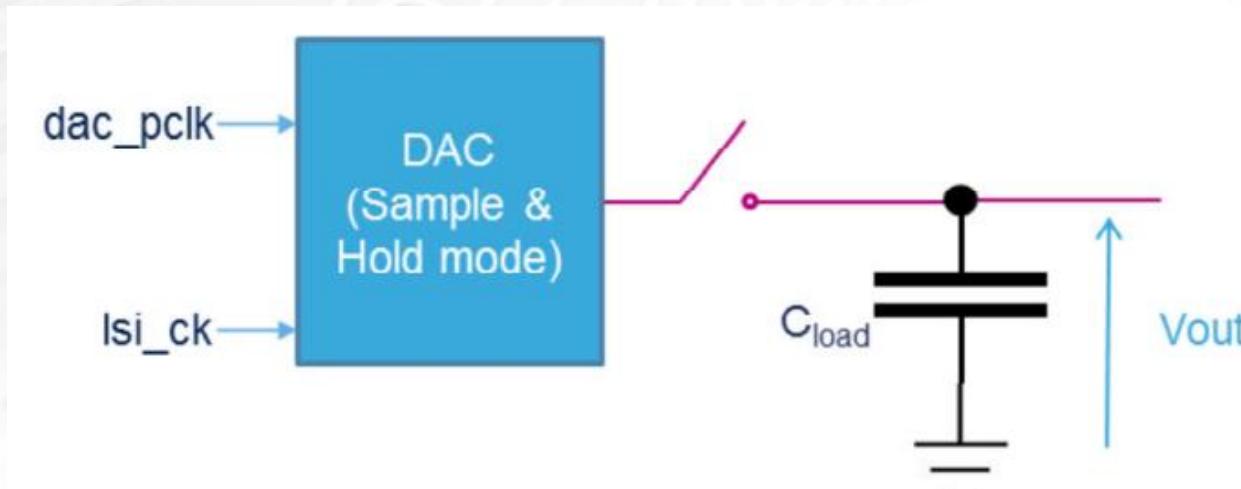
Holding phase: مرحلة المسك في هذه المرحلة يتم الاحتفاظ بقيمة جهد خرج المبدل

Refresh phase: مرحلة التنشيط في هذه المرحلة وبسبب انخفاض جهد المكثف عن القيمة المرغوبة يتم إعادة شحن المكثف



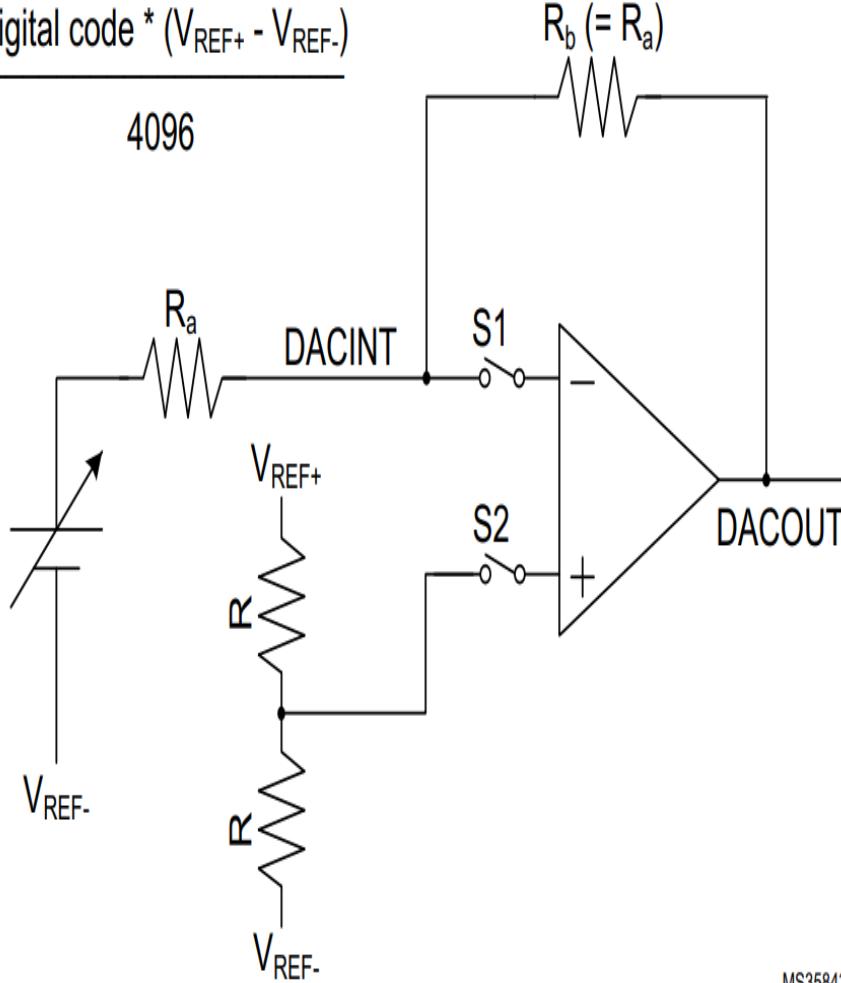
Sample and hold mode

- نطـ الـ sample and hold مـ صـمـ خـصـيـصـاً منـ أـجـلـ أـوـضـاعـ الطـاقـةـ المـنـخـفـضـةـ
- يمـكـنـ ضـبـطـ أـزـمـنـةـ sample hold and refresh timingsـ مـنـ خـلـالـ إـلـاـعـادـاتـ.
- فيـ هـذـاـ النـمـطـ مـنـ الـعـلـمـ يـتـمـ قـيـادـةـ المـبـدـلـ وـجـمـيـعـ الـمـسـجـلـاتـ وـالـدـارـاتـ المـنـطـقـيـةـ مـتـعـلـقـةـ بـهـ مـنـ خـلـالـ الـهـزاـزـ الـكـرـيـسـتـالـيـ الدـاخـلـيـ مـنـخـفـضـ السـرـعـةـ low speed internal oscillator(Isi_clk)



جهد خرج المبدل الرقمي التشابهی

$$V_{DAC} = \frac{\text{Digital code} * (V_{REF+} - V_{REF-})}{4096}$$



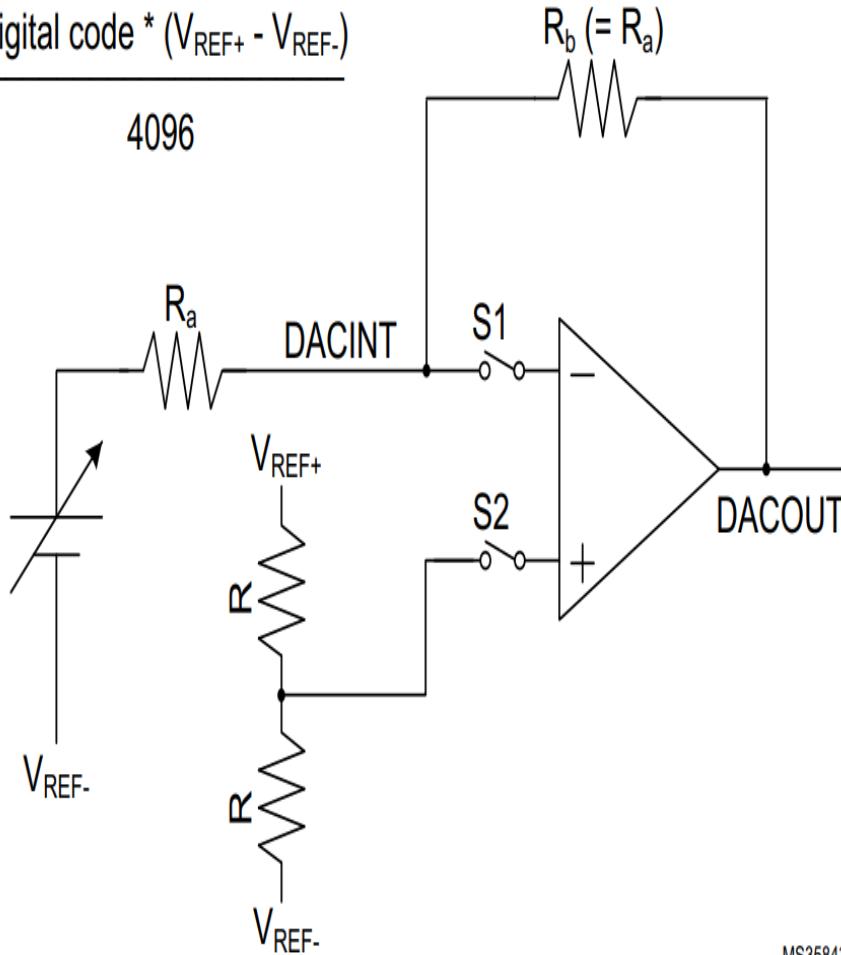
□ يتم حساب جهد خرج المبدل من خلال العلاقة التالية:

$$DACOutput = (V_{REF+}) \times \left(DOR / 4096 \right)$$

ويتراوح جهد خرج المبدل بين 0 وال V_{REF+}

جهد خرج المبدل الرقمي التشابهی

$$V_{DAC} = \frac{\text{Digital code} * (V_{REF+} - V_{REF-})}{4096}$$

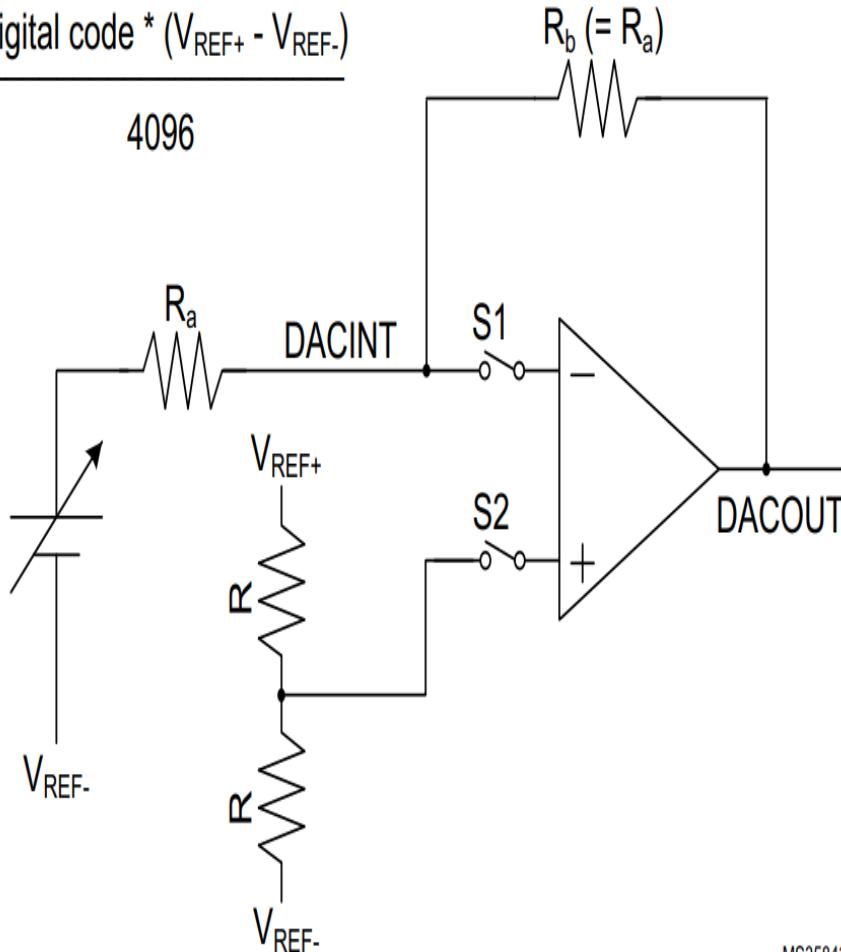


عندما يكون **output** في حالة فصل **buffer** ، عندها يكون جهد دخل المبدل متصل بجهد خرجه عن طريق المقاومتين **Ra** و **Rb** وتصبح ممانعة المبدل هو عبارة عن مجموع هاتين المقاومتين **Ra+Rb** وباعتبار المقاومتين متساويتين تصبح ممانعة المبدل **2*Ra** (وهنا يكون المفتاحين **S1** و **S2** في حالة فصل **open**)



جهد خرج المبدل الرقمي التشابهی

$$V_{DAC} = \frac{\text{Digital code} * (V_{REF+} - V_{REF-})}{4096}$$



MS35843V2

□ عندما يكون **output** في حالة وصل **on** ، تكون المضخم العملياتي يعمل كمضخم عاكس وبعامل ربح 1- و تكون ممانعة الخرج صفرية

سرعة التبديل للمبدل الرقمي التشابهـي

- عندما يكون **output buffer** في حالة وصل **on** ، عندها تتعلق سرعة المبدل بمواصفات المضخم العملياتي المستخدم والذي يحدد من الـ **datasheet**
- أما عندما يكون **output buffer** في حالة فصل **off** ، عندها تتعلق سرعة المبدل بالثابت الزمني **RC** والذي يمكن حسابه من خلال معرفة ممانعة المبدل $RDAC=2*Ra$ ومن خلال قيمة المكثف على خرج المبدل

استخدام مضخم عملياتي خارجي

في هذه الحالة يتم فصل المضخم الداخلي للمبدل ووصل مضخم عملياتي خارجي وهذا تتحدد سرعة المبدل من خلال الثابت الزمني للمبدل RC بالإضافة إلى سرعة المضخم الخارجي (gain bandwidth and slew rate) كما يجب أن تكون قيمة مقاومة R_a مساوية إلى قيمة مقاومة المبدل وإلا سيتتج خطأ في الربح.

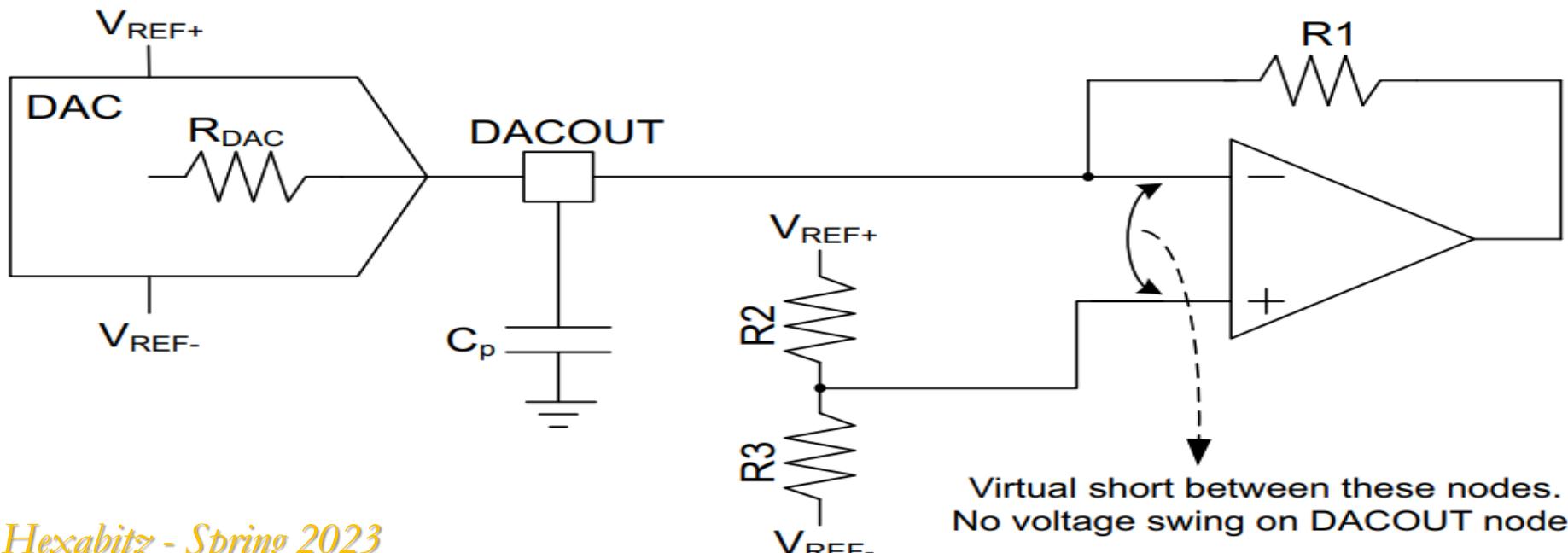
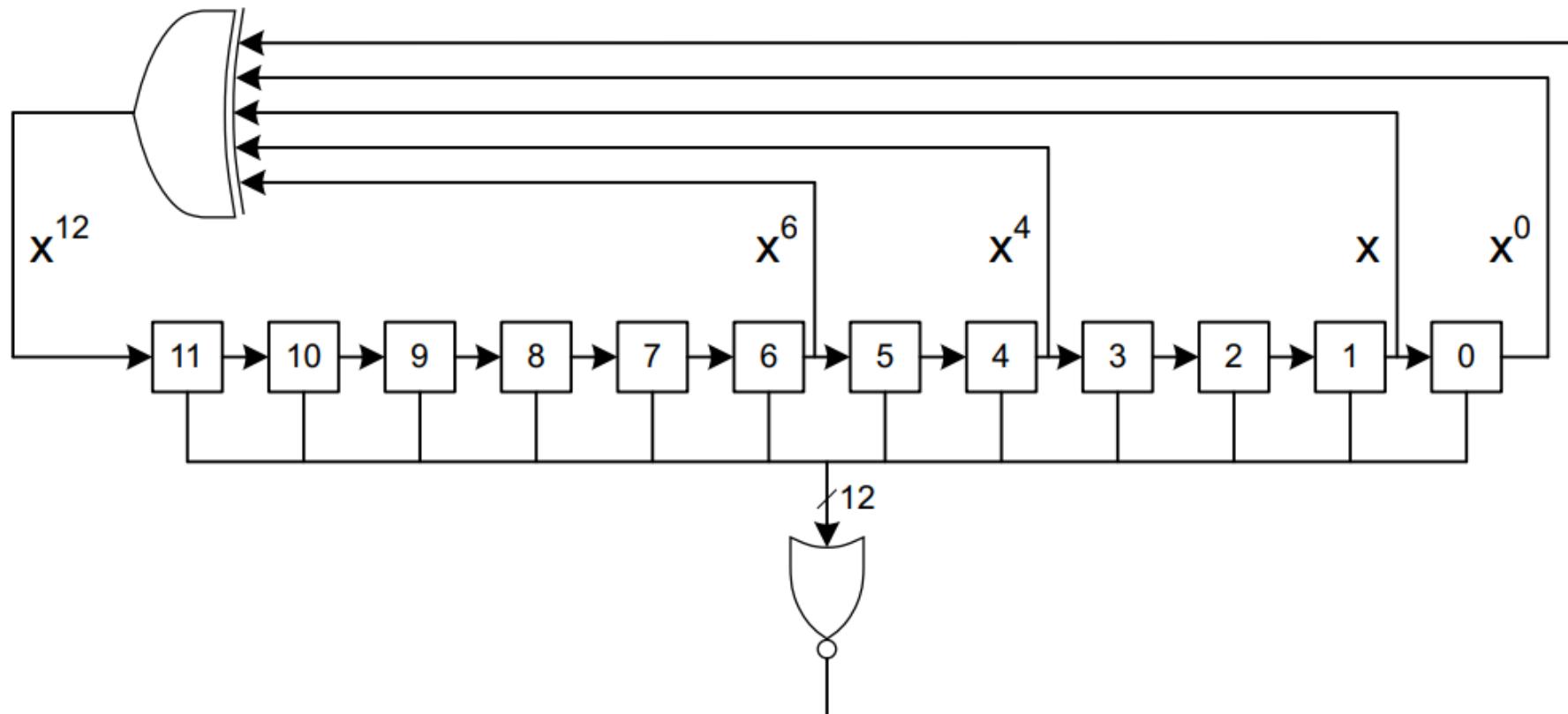


Table 2. Maximum sampling time

Product	Max bus speed	DAC max sampling rate
STM32F0 Series	48 MHz	4.8 Msps
STM32F100xx	24 MHz	2.4 Msps
STM32F101xx STM32F103xx STM32F105xx STM32F107xx	36 MHz	4.5 Msps
STM32F2 Series	30 MHz	7.5 Msps
STM32F3 Series	36 MHz	4.5 Msps
STM32F40x STM32F41x	42 MHz	10.5 Msps
STM32F42x	45 MHz	11.25 Msps
STM32F7 Series	54 MHz	13.5 Msps
STM32G0 Series	64 MHz	8.0 Msps

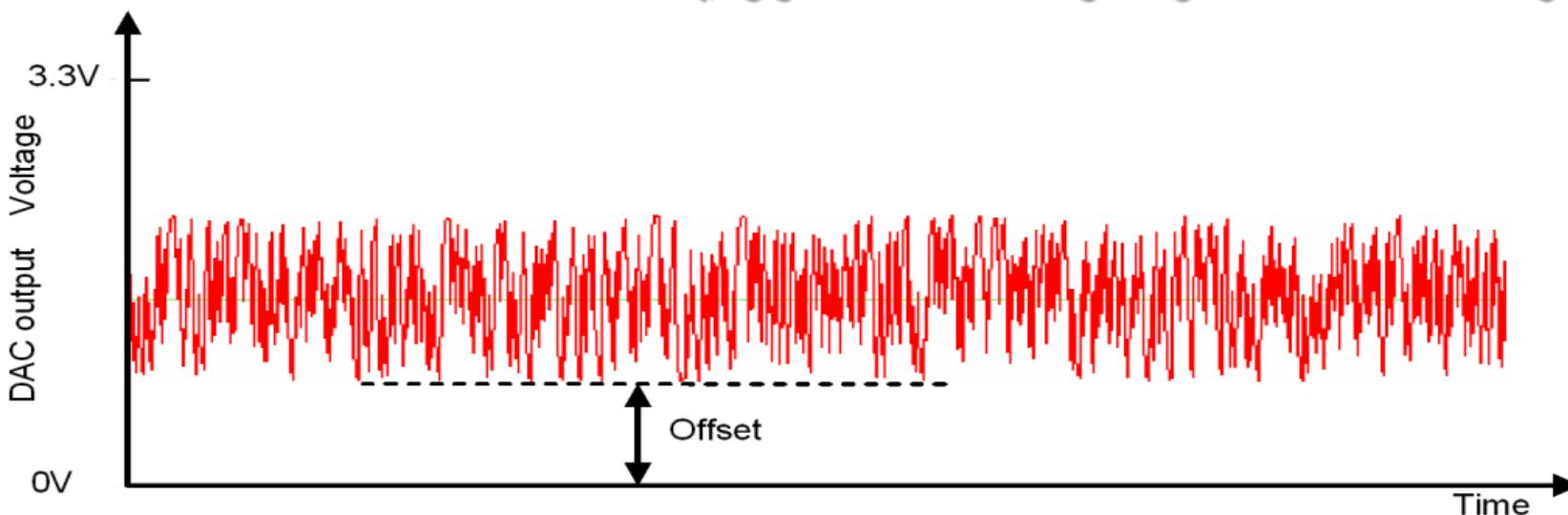
توليد إشارة الضجيج باستخدام المبدل White noise generator

يمكن أن يعمل المبدل الرقمي التشابهـي في متحكمـات stm32 كمولد إشارة ضجيج عشوائي باستخدام مسجل إزاحة وعدد من نقاط التفرع مساوٍ لـ $2^n - 1$ وهو العدد الذي تحتاجـه الإشارة قبل أن تكرـر نفسها



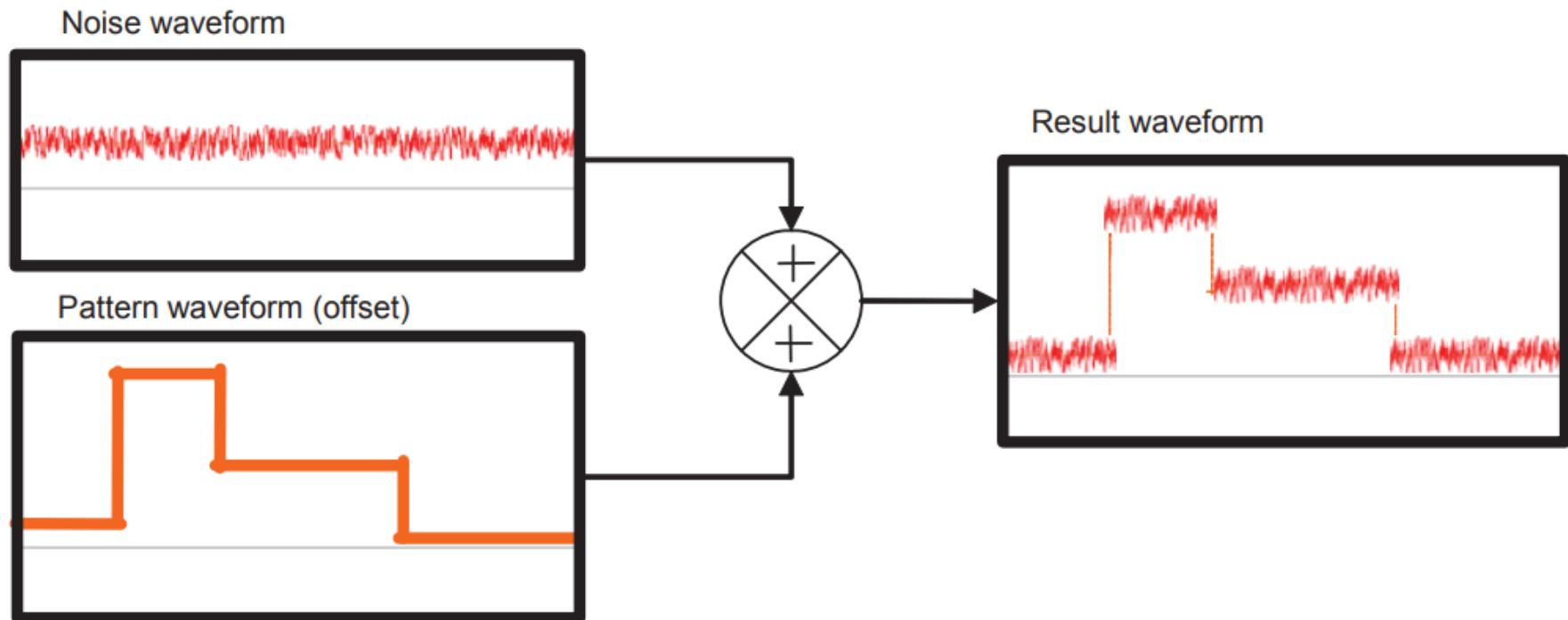
توليد إشارة الضجيج باستخدام المبدل White noise generator

- يمكن اعتبار إشارة الضجيج الناتجة عن المبدل كإشارة ضجيج أبيض باعتبار أن لها توزع طيفي مسطح
- يمكن استخدام هذه الإشارة في توليد الموسيقى الإلكترونية بشكل مباشر أو بشكل غير مباشر عن طريق استخدامها كدخل لمرشح من أجل توليد أشكال أخرى للإشارة الضجيج
- كما يمكن استخدامها لأغراض التحكم على سبيل المثال لاختبار استجابة التردد للمضخمات والمرشحات الإلكترونية



توليد إشارة الضجيج باستخدام المبدل White noise generator

□ كما يمكن الحصول على إشارة ضجيج مزاحمة بمقدار معين من خلال جمعها مع إشارة إزاحة (signal pattern) معرفة مسبقاً والتي يتم حفظها ضمن جدول



توليد إشارة مثلثية باستخدام المبدل

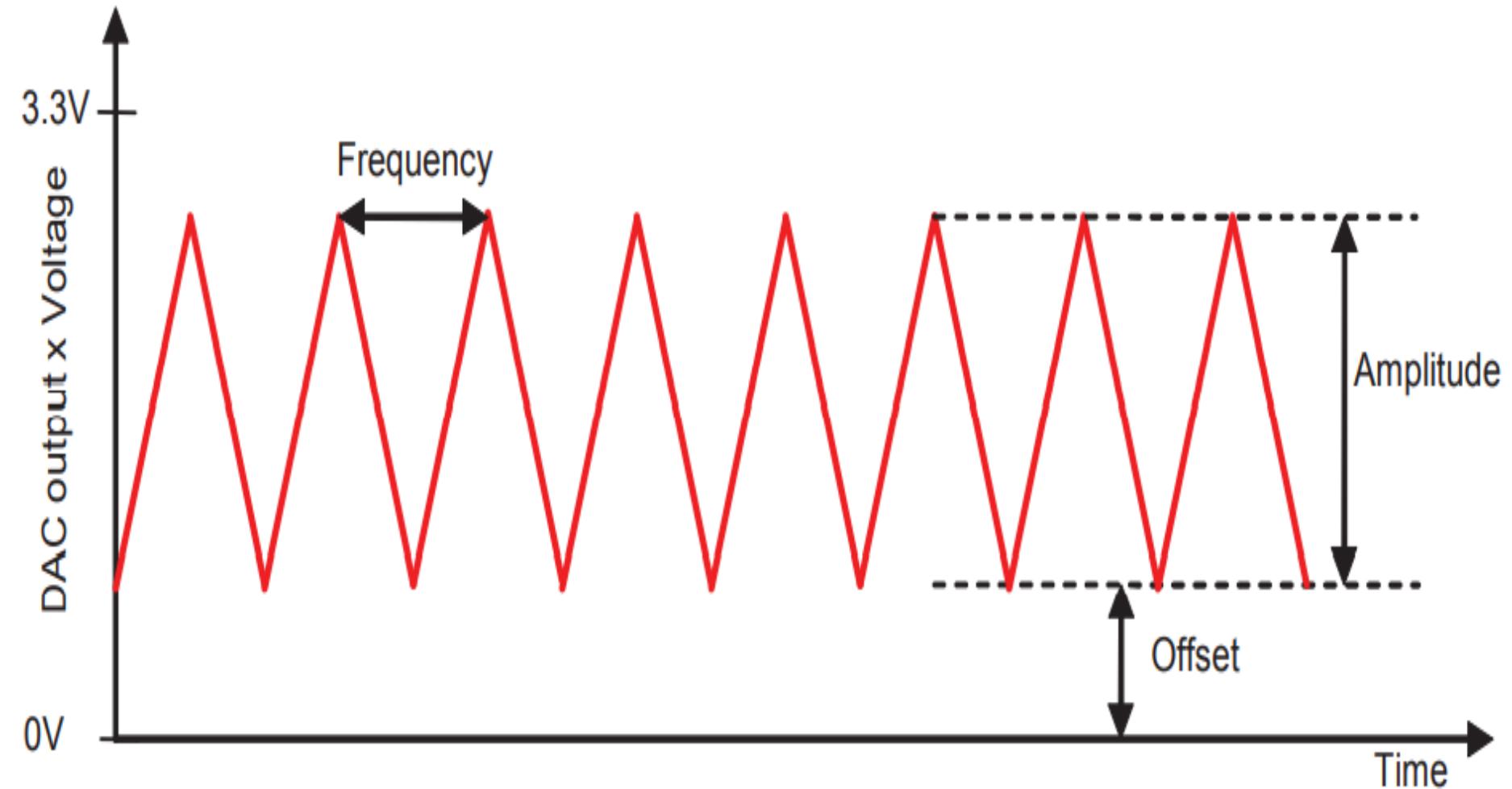
يمكن للمبدل الرقمي التشابهـي في متحكمـات stm32 توليد إشارة مثلثـية مع قابلـية التحكم بمطالـها وترددـها وأيضاً مقدار إزاحة الإشـارة من خلال **DAC_CR**

Table 2. Preprogrammable triangular waveform amplitude values

MAMPx[3:0] bits	Digital amplitude	Analog amplitude (Volt) (with $V_{REF+} = 3.3$ V)
0	1	0.0008
1	3	0.0024
2	7	0.0056
3	15	0.0121
4	31	0.0250
5	63	0.0508
6	127	0.1023
7	255	0.2054
8	511	0.4117
9	1023	0.8242
10	2047	1.6492
≥ 11	4095	3.2992

توليد إشارة مثلثية باستخدام المبدل

Triangular wave generator

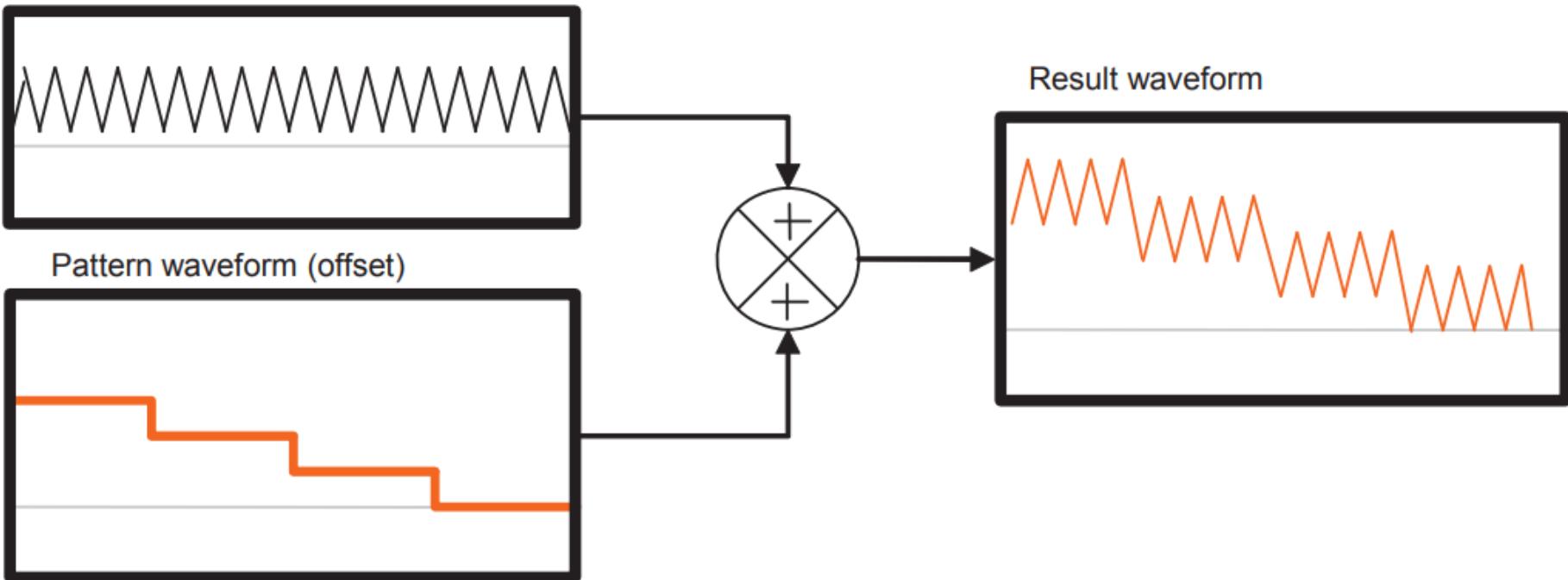


توليد إشارة مثلثية باستخدام المبدل

□ كما يمكن الحصول على إشارة مثلثية مزاحمة بمقدار معين من خلال جمعها مع إشارة إزاحة (signal pattern) معرفة مسبقاً والتي يتم حفظها ضمن جدول، على ألا يتجاوز مطال الإشارة الناتجة عن الـ

4095

Triangular waveform



320

من أجل توليد أي إشارة من خلال المبدل الرقمي التشابهي نتبع الخطوتين التاليتين :

- الخطوة الأولى: توليد جدول Lookup Table وهو عبارة عن العينات الرقمية للإشارة
- الخطوة الثانية: إرسال هذه البيانات إلى المبدل ليقوم بتحويلها إلى إشارة تشابهية

توليد جدول Lookup Table وهو عبارة عن العينات الرقمية للإشارة

يمكن توليد جدول Lookup Table للإشارة من خلال أحد المواقع التي تقوم بحساب العينات الرقمية تبعاً لعدد العينات المراده والمطال الأعظمي للإشارة بالشكل التالي:

Sine Look Up Table Generator Calculator

This calculator generates a single cycle sine wave look up table. It's useful for digital synthesis of sine waves.

Sine Look Up Table Generator Input

Number of points	<input type="text" value="1023"/>
Max Amplitude	<input type="text" value="65535"/>
Numbers Per Row	<input type="text" value="8"/>
<input checked="" type="radio"/> Hex	<input type="radio"/> Decimal
<input type="button" value="Submit"/>	

توليد جدول Lookup Table و هو عبارة عن العينات الرقمية للإشارة

يمكن توليد جدول Lookup Table للإشارة من خلال أحد المواقع التي تقوم بحساب العينات الرقمية تبعاً لعدد العينات المراده والمطال الأعظمي للإشارة بالشكل التالي:

Sine Look Up Table Generator Calculator

```
0x8000, 0x80c9, 0x8192, 0x825b, 0x8324, 0x83ee, 0x84b7, 0x8580,  
0x8649, 0x8712, 0x87db, 0x88a4, 0x896c, 0x8a35, 0x8afe, 0x8bc6,  
0x8c8e, 0x8d57, 0x8e1f, 0x8ee7, 0x8fae, 0x9076, 0x913e, 0x9205,  
0x92cc, 0x9393, 0x945a, 0x9521, 0x95e7, 0x96ad, 0x9773, 0x9839,  
0x98fe, 0x99c4, 0x9a89, 0x9b4d, 0x9c12, 0x9cd6, 0x9d9a, 0x9e5e,  
0x9f21, 0x9fe4, 0xa0a7, 0xa169, 0xa22b, 0xa2ed, 0xa3af, 0xa470,  
0xa530, 0xa5f1, 0xa6b1, 0xa770, 0xa830, 0xa8ef, 0xa9ad, 0xaa6b,  
0xab29, 0xabe6, 0xacaa3, 0xad5f, 0xae1b, 0xaeed7, 0xaf92, 0xb04d,  
0xb107, 0xb1c0, 0xb27a, 0xb332, 0xb3ea, 0xb4a2, 0xb559, 0xb610,  
0xb6c6, 0xb77c, 0xb831, 0xb8e5, 0xb999, 0xba4d, 0xbb00, 0xbbb2,  
0xbc64, 0xbd15, 0xbdc6, 0xbe76, 0xbf25, 0bfd4, 0xc082, 0xc12f,  
0xc1dc, 0xc288, 0xc334, 0xc3df, 0xc489, 0xc533, 0xc5dc, 0xc684,  
0xc72c, 0xc7d3, 0xc879, 0xc91f, 0xc9c3, 0xca67, 0xcb0b, 0xcbae,  
0xcc4f, 0xccf1, 0xcd91, 0xce31, 0xed0, 0xcf6e, 0xd00b, 0xd0a8,  
0xd144, 0xd1df, 0xd279, 0xd313, 0xd3ac, 0xd443, 0xd4db, 0xd571,
```

□ كما يمكن توليد جدول **Lookup Table** للإشارة من خلال برنامج الماتلاب بالشكل التالي:

```
1 clear; clc; % Clear The Previous Points
2 Ns      = 128; % Set The Number of Sample Points
3 RES     = 12;  % Set The DAC Resolution
4 OFFSET  = 0;   % Set An Offset Value For The DAC Output
5 %-----[ Calculate The Sample Points ]-----
6
7 T = 0:((2*pi/(Ns-1))):(2*pi);
8 Y = sin(T);
9 Y = Y + 1;
10 Y = Y*((2^RES-1)-2*OFFSET)/(2+OFFSET);
11 Y = round(Y);
12 plot(T, Y);
13 grid
14 %-----[ Print The Sample Points ]-----
15
16 fprintf('%d, %d, \n', Y);
17
18 %-----[ Other Examples To Try ]-----
19
20 % Y = diric(T, 13);      % Periodic Sinc
21
22 % Y = sawtooth(T)       % Sawtooth
23
24 % Y = sawtooth(T, 0.5); % Triangular
```

DMA capabilities

تحتوي متحكمات على الأقل على stm32 موديل واحد DMA بعة قنوات

ترتبط كل قناة من قنوات DMA مع قناة من قنوات الـ DAC

في حال عدم استخدام نمط الـ DMA ، يقوم المعالج بتزويد المبدل الرقمي التشابهي

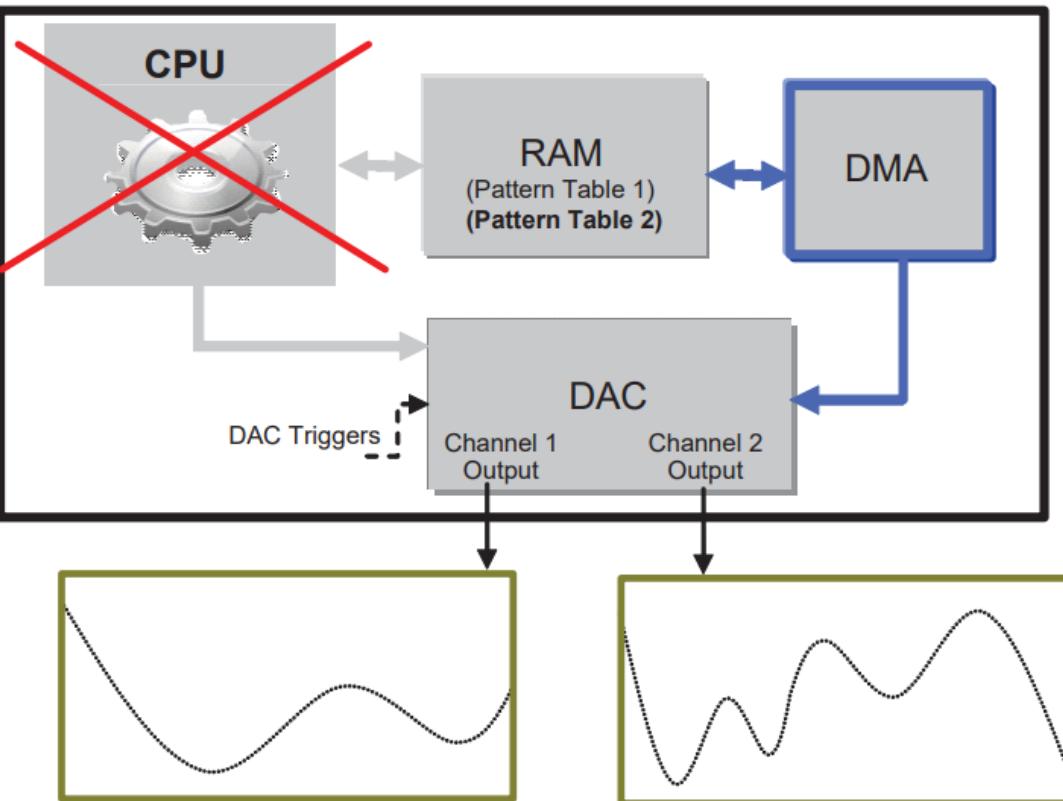
بالبيانات الرقمية لتوليد الإشارة المطلوبة من المبدل، حيث يتم حفظ هذه البيانات في الذاكرة RAM، هنا يكون المعالج هو الوسيط بين الذاكرة

والمبدل ³²⁵

DMA capabilities



أما عند استخدام نمط الـ **DMA** مع المبدل الرقمي التشابهـي ، فإن المودـiolـ الـ **DMA** يقوم بدور الوسيط بين الـ **ذاكرة RAM** والمبدل ، بدون تدخل من المعالج مما يؤدي إلى زيادة كفاءة وفعالية النـظام ويترك المعالج ليقوم بالعمليـات الأخرى



هناك أربع خيارات لكل قناة من قنوات المبدل:

- Disabled**: في هذه الحالة تكون القناة غير مفعّلة، ويمكن استخدام القطب المرتبط بالمبدل كقطب GPIO
- Connected to external pin only**: في هذه الحالة يكون خرج المبدل متصل فقط بالقطب الخارجي المحجوز للمبدل
- Connected to peripherals only**: في هذه الحالة يكون خرج المبدل متصل فقط بأحد طرفيات المتحكم داخلياً
- Connected to external pin and peripherals**: في هذه الحالة يكون خرج المبدل متصل بالقطب الخارجي وبأحد طرفيات المتحكم داخلياً

ضبط إعدادات المبدل في بيئة stm32CubeIDE

هناك أربع خيارات لكل قناة من قنوات المبدل:

Screenshot of the STM32CubeIDE Device Configuration Tool interface. The central panel shows the 'Pinout & Configuration' tab for the 'test.ioc' project. A red box highlights the 'DAC1 mode and Configuration' dropdown, which lists four options: 'OUT1 mode' (selected), 'OUT2 mode', 'External', and 'Connected to external pin only'. The right side of the interface features a detailed pinout diagram for the STM32G071RBTx LQFP64 package, showing all 64 pins labeled PC0 through PE15. The bottom right corner displays a watermark for 'Activate Windows'.

ضبط إعدادات المبدل في بيئة stm32CubeIDE

هنا يتم تفعيل أو إلغاء تفعيل Output buffer، أيضاً يتم اختيار نوع القدر، كما يتم تفعيل أو إلغاء تفعيل نمط الـ sample and hold :mode

Configuration

Reset Configuration

NVIC Settings DMA Settings GPIO Settings

Parameter Settings User Constants

Configure the below parameters :

Search (Ctrl+F) ⏪ ⏩ ⓘ

DAC Out1 Settings

Output Buffer	Enable
Trigger	None
User Trimming	Factory trimming
Sample And Hold	Sampleandhold Disable

ضبط إعدادات المبدل في بيئة stm32CubeIDE

هنا يتم ضبط إعدادات الـ :DMA



Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

DMA Request	Channel	Direction	Priority
DAC1_CH1	DMA1 Channel 1	Memory To Peripheral	Low

Add

Delete

DMA Request Settings

Mode

Peripheral
Increment Address

Memory

Data Width

DMA Request Synchronization Settings

Enable synchronization

Synchronization signal

Signal polarity

Enable event

Request number

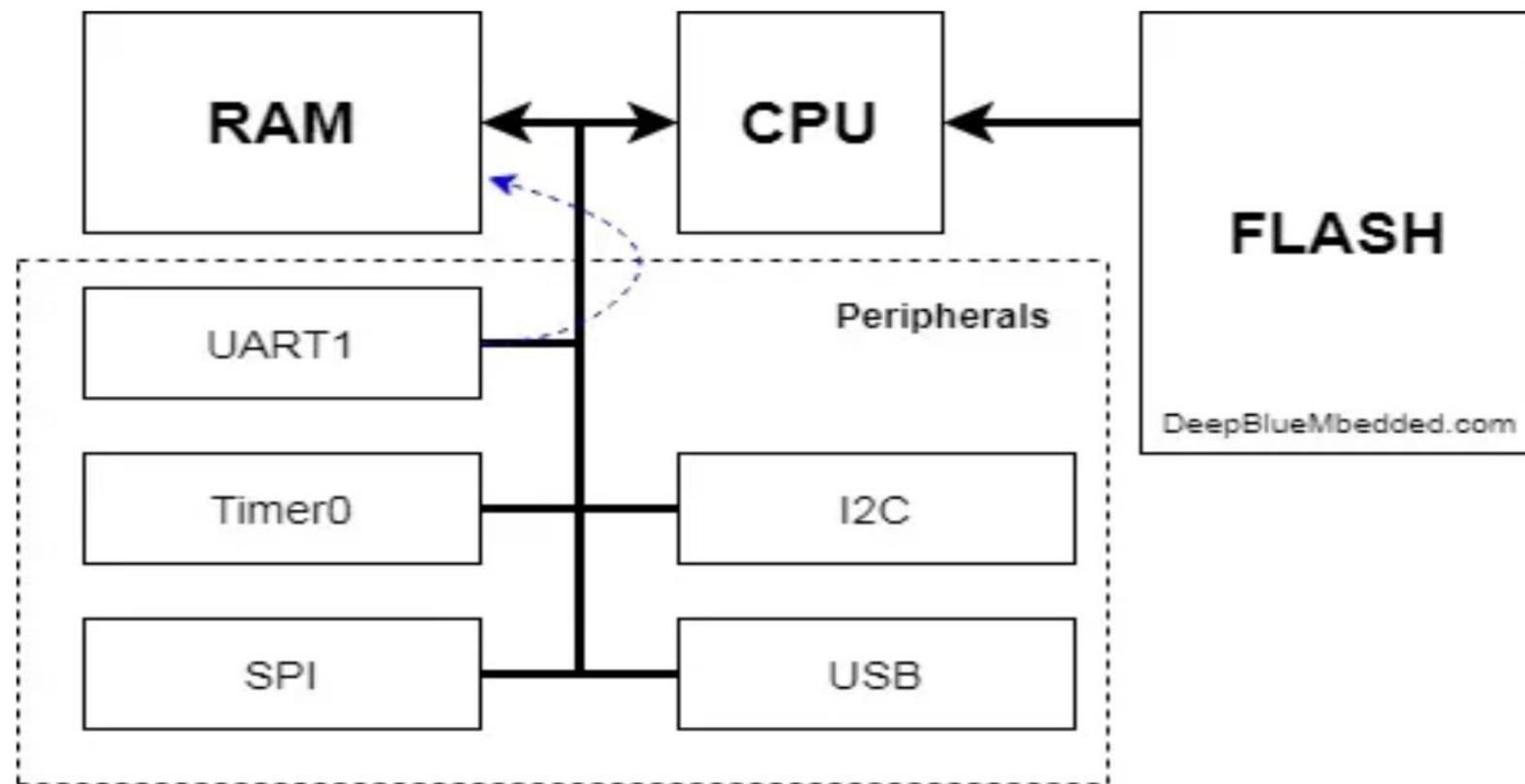
إدارة الوصول المباشر للذاكرة Direct Access Memory (DMA)

□ إدارة الوصول المباشر للذاكرة DMA هي عبارة عن وحدة رقمية منطقية مدمجة داخل متحكمات stm32 وتستخدم لإدارة نقل البيانات بين الذاكرة والطيفيات، وبين الذاواكر نفسها ، حيث يتم تبادل البيانات من خلال وحدة الـ DMA بدون أي تدخل من المعالج ، مما يتيح للمعالج التفرغ لمعالجة باقي الأمور المهمة.

- بإمكان وحدة الـ DMA الوصول لأي موقع ذاكري بما في ذلك:
- CRC generator: على سبيل المثال AHB peripherals
 - SRAM: على سبيل المثال AHB memories
 - USART: على سبيل المثال APB peripherals

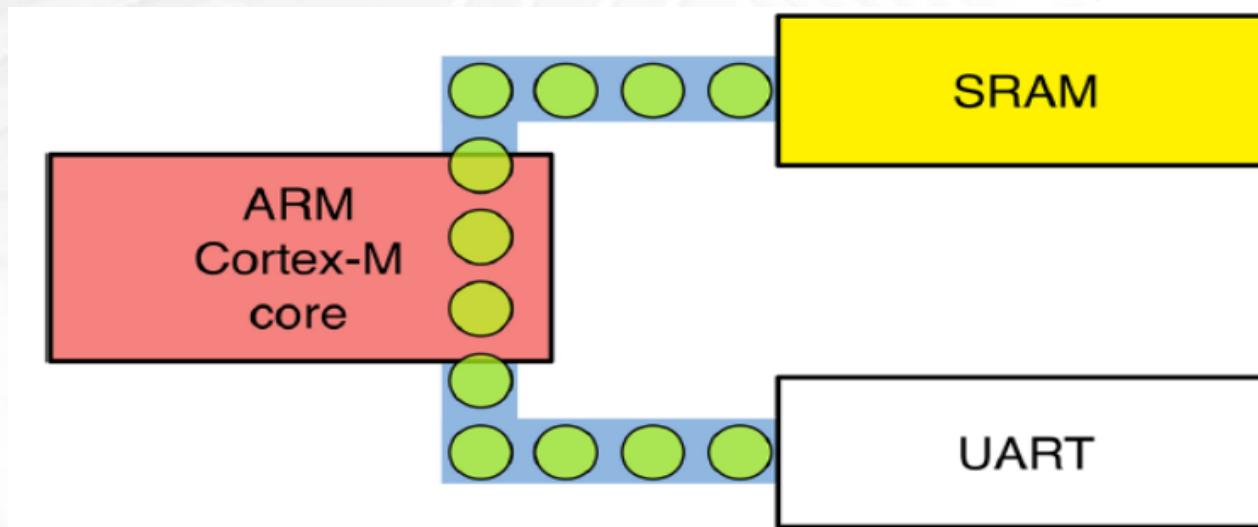
إدارة الوصول المباشر للذاكرة (DMA)

سابقاً وقبل وجود وحدة DMA ، كان على المعالج القيام بكافة الأعمال بما في ذلك جلب التعليمات (الكود) من الذاكرة flash ثم تنفيذ هذه التعليمات ومن ثم نقل البيانات بين الذاكرة والطرفيات



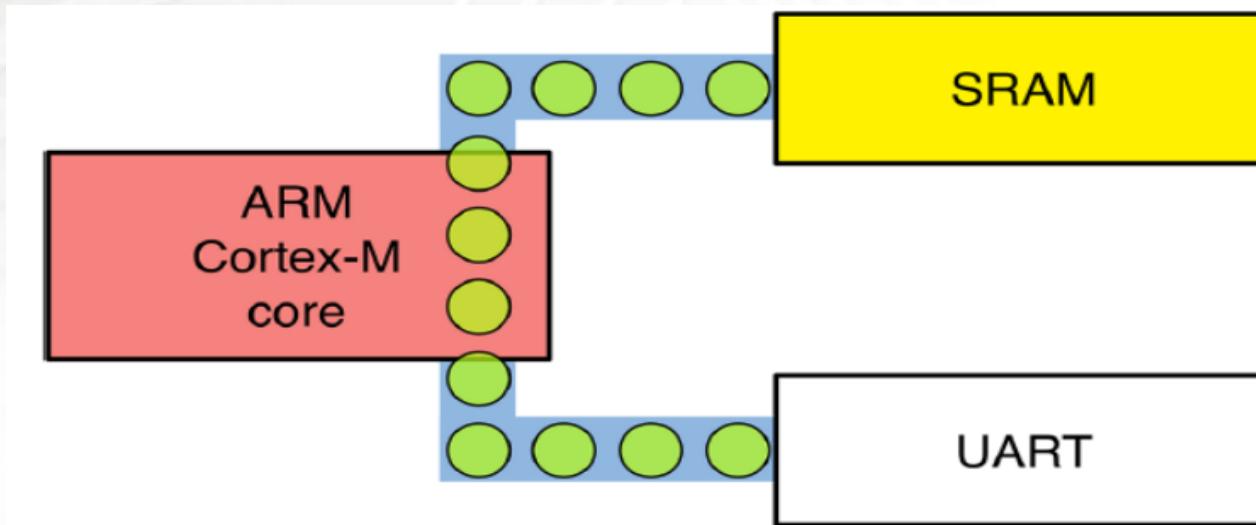
إدارة الوصول المباشر للذاكرة (DMA)

فبفرض أن هناك بيانات قادمة من المنفذ التسلسلي UART1 بطول buffer 20byte ، وعلى المعالج استقبال هذه البيانات ثم نقلها إلى محدد ضمن الذاكرة مع الانتباه إلى عدم ضياع أي من البيانات القادمة عمليات نقل البيانات هذه القادمة من الطرفيات مثل ,SPI, UART و غيرها ، تولد عدد لا نهائي من المقااطعات خلال الثانية الواحدة وعلى المعالج أن يقوم بمعالجتها جميعها



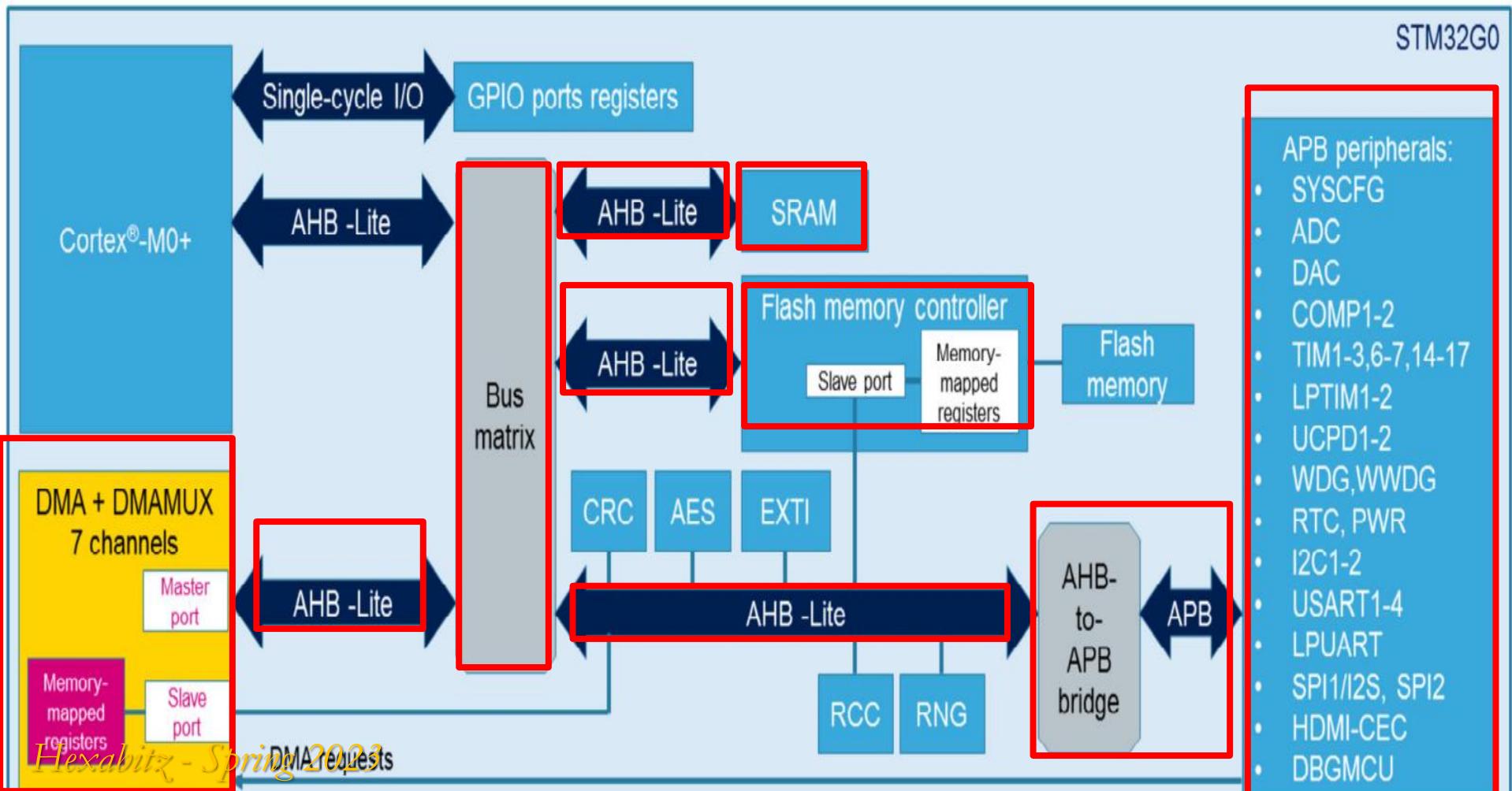
إدارة الوصول المباشر للذاكرة (DMA)

تستغرق عملية الاستجابة للمقاطعة من خلال الذهاب إلى برامج خدمة المقاطعة والعودة منها عدة دورات ساعة، مما يعني ضياع الكثير من الوقت خصوصاً مع المقاطعات التي تحدث باستمرار وبشكل دوري ، وما يعني انشغال المعالج عن تنفذ الكود و مما يؤدي في بعض الأحيان إلى تجاوز القيود الخاصة بالزمن المحدد لتنفيذ الكود، وهنا تأتي الحاجة لوحدة DMA



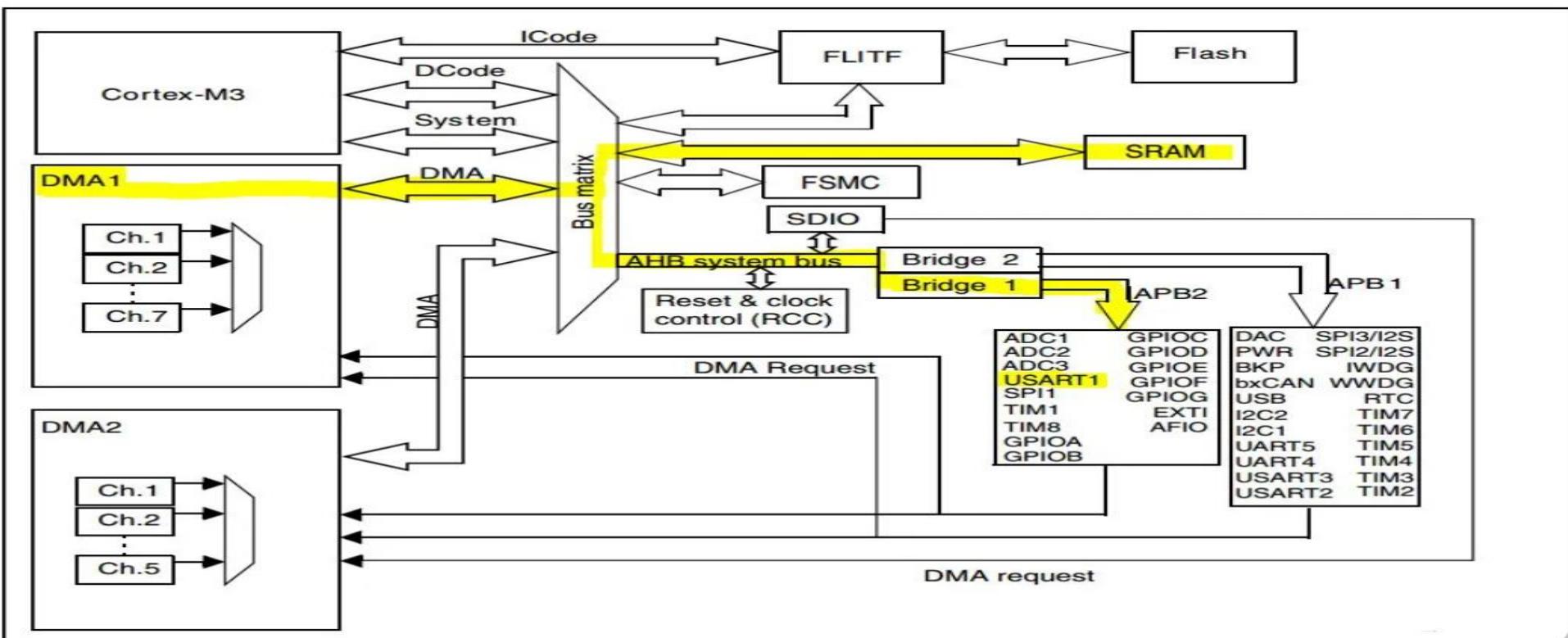
إدارة الوصول المباشر للذاكرة (DMA)

نلاحظ من المخطط الصندوقي أن بإمكان وحدة DMA توجيه البيانات القادمة من الطرفيات إلى الذاكرة مباشرة دون أي تدخل من المعالج



إدارة الوصول المباشر للذاكرة (DMA)

على سبيل المثال نلاحظ أن وحدة DMA هنا قامت بتوجيه البيانات القادمة من المنفذ التسلسلي إلى الذاكرة مباشرةً دون تدخل المعالج



إدارة الوصول المباشر للذاكرة (DMA)

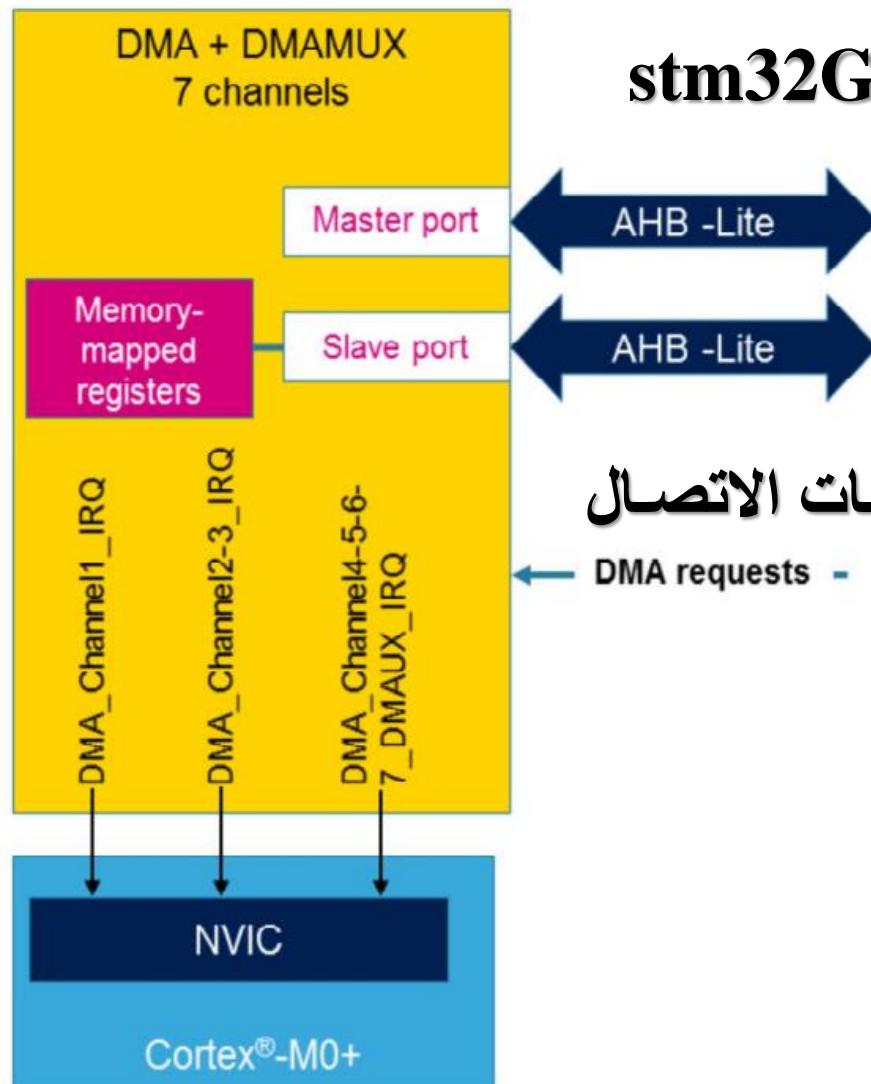


مواصفات وحدة الـ DMA في متحكمات stm32G0

:AHB master bus

يدعم متحكم DMA منفذين
أحد هما master يستخدم من قبل قنوات
الـ DMA للوصول إلى الذاكرة أو إلى
مسجلات الطرفيات، والأخر slave
يستخدم للوصول إلى متحكم
وحدة الـ DMA ومسجلات الحالة والتحكم الوحدة.

إدارة الوصول المباشر للذاكرة (DMA)



مواصفات وحدة الـ DMA في متحكمات stm32G0

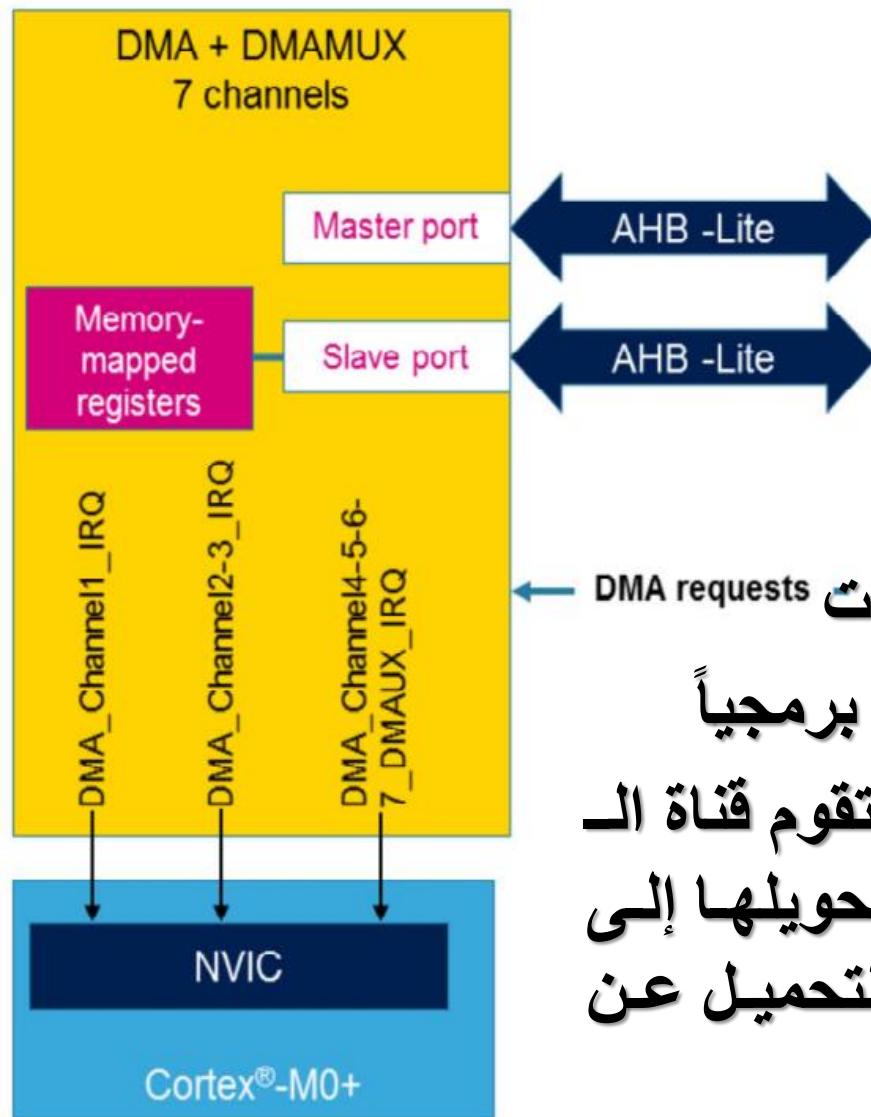
المرؤنة في ضبط الإعدادات:

تدعم معظم الطرفيات في متحكم
stm32G0 نمط الـ DMA ، حيث يعتبر

هذا النمط هو الأفضل خصوصاً مع طرفيات الاتصال
communication peripherals

والمبلات
ADC/DAC

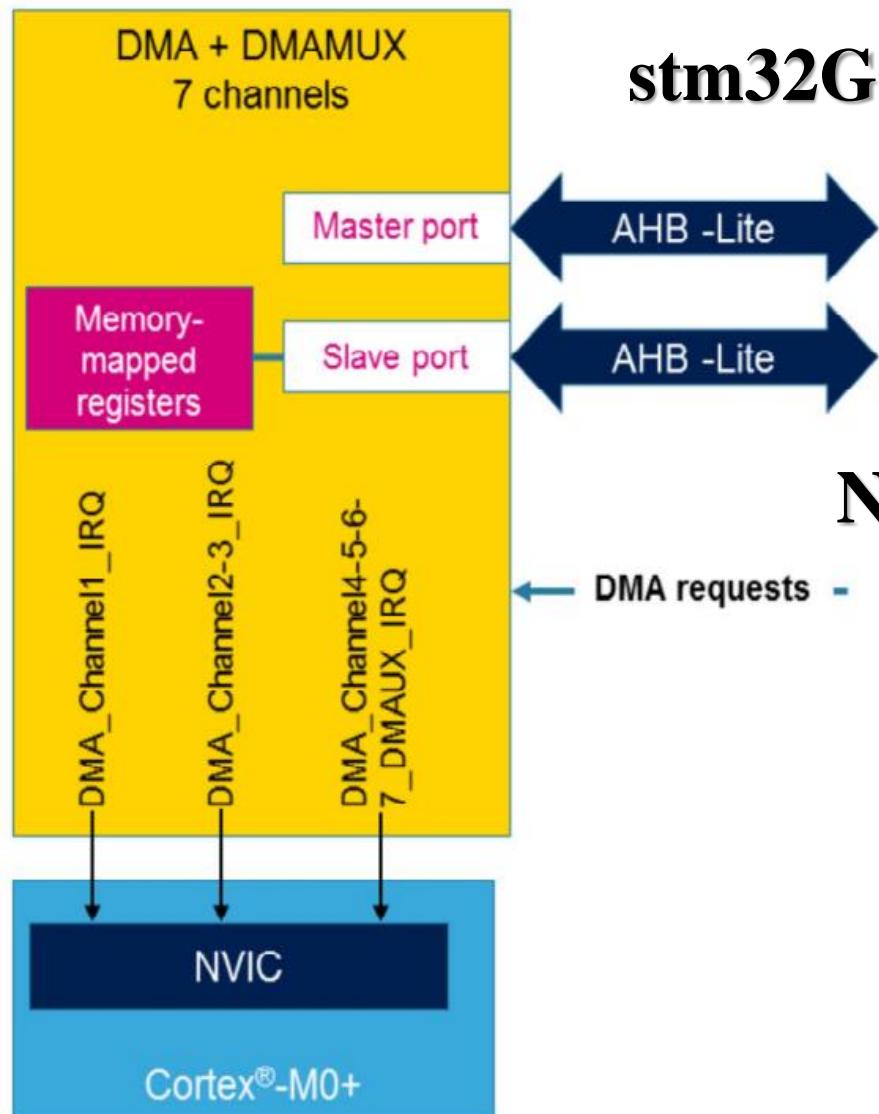
إدارة الوصول المباشر للذاكرة (DMA)



على سبيل المثال المبدل التشابهي الرقمي ADC يتطلب أخذ عينات بشكل دوري وتخزينها ضمن الذاكرة، حيث توفر من متحكمات stm32G0 طريقتين للتعامل مع المبدل :

- استخدام المقاطعات: حيث يتم نقل العينات التي تم تحويلها إلى الذاكرة من قبل المعالج برمجياً
- استخدام نمط ال DMA: في هذا النمط تقوم قناة ال DMA بمهمة نقل العينات التي تم تحويلها إلى الذاكرة ، بهذه الطريقة يتم تخفيف التحميل عن المعالج

إدارة الوصول المباشر للذاكرة (DMA)



مواصفات وحدة الـ DMA في متحكمات stm32G0

إمكانية تعديل الأولويات برمجياً

أو عن طريق الـ hardware

يدعم متحكم الـ DMA ثلاثة مخارج

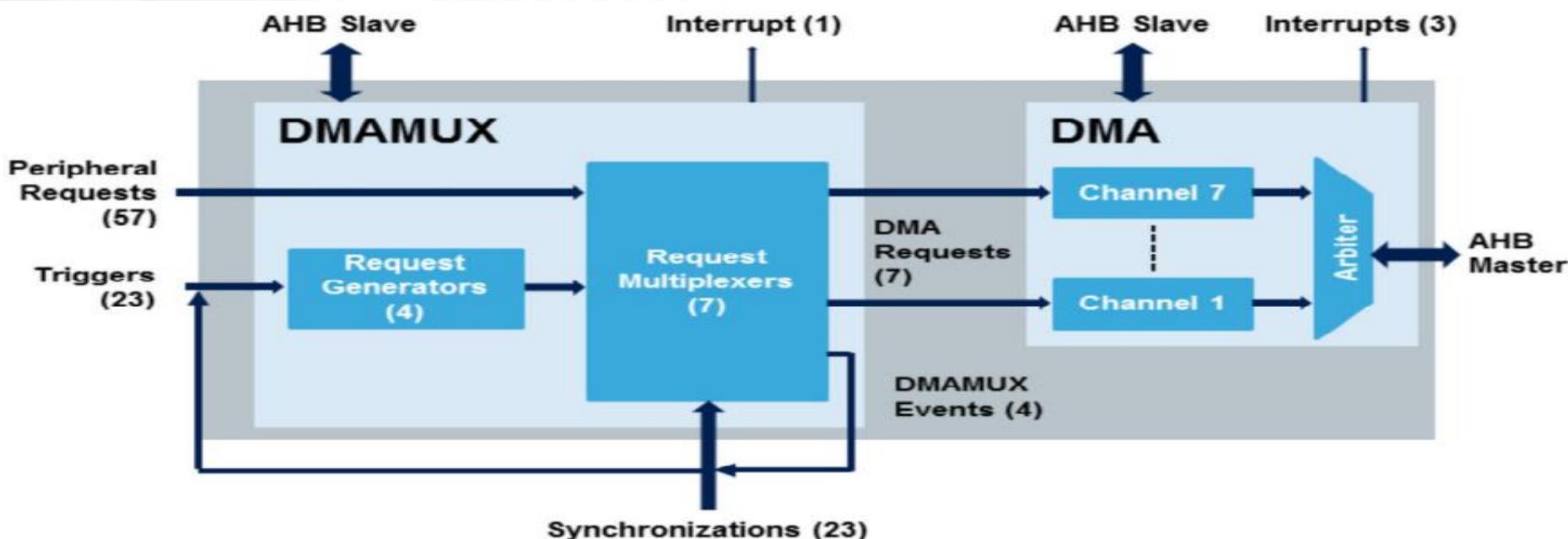
للمقاطعة متصلة بمتحكم المقاطعات NVIC

يمكن تعديل الأولويات لها من خلال البرنامج أو من خلال الهاردوير

إِدَارَةُ الْوَصْولِ الْمُبَاشِرِ لِلذَّاكرَةِ (DMA)

مواصفات وحدة DMA في متحكمات stm32G0

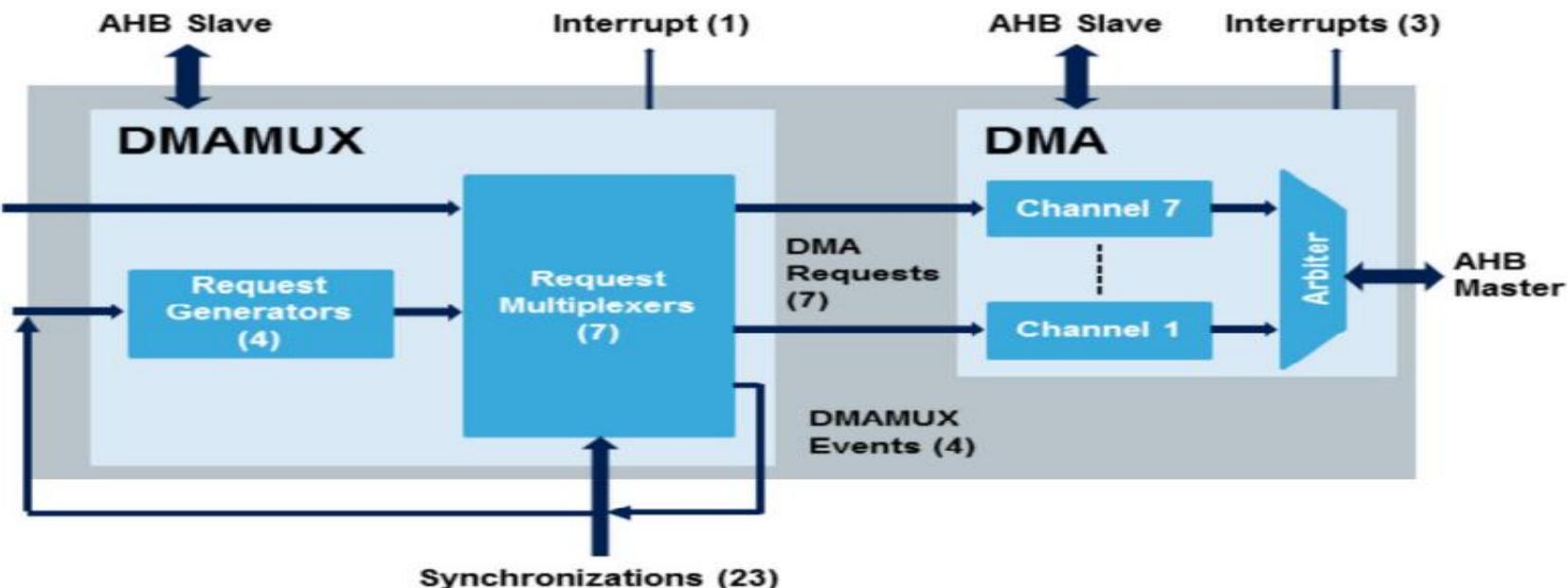
□ متحكم DMA واحد يحتوي على 7 قنوات مستقلة قابلة للضبط، لكل منها أولوية قابلة أيضاً لإعادة ضبطها، ويتم نقل البيانات من خلالها عبر ناقل الـ Bus matrix وصولاً للـ AHB master



إدارة الوصول المباشر للذاكرة (DMA)

مواصفات وحدة DMA في متحكمات stm32G0

New DMA request Multiplexer (DMAMUX) مميزة موجودة في عائلة stm32G0 وغير موجودة في عائلة stm32f0 ، ومهماها تنظيم وتحفيظ مسار الطلبات القادمة من الطرفيات إلى قنوات DMA ، أيضاً لها دور في معالجة الأحداث والتزامن



Individual DMA stream flexibility

من أجل كل قناة من قنوات الـ DMA يمكن ضبط برمجياً كل من:

حجم البيانات لكل من المصدر source والوجهة destination

عنوان كل من الوجهة والمصدر، والمؤشرات الخاصة بها أيضاً يمكن ضبطها لتكون قابلة للزيادة الآلية أو بإعطائها عناوين محددة

يمكن أن يصل حجم البيانات المتبادلة إلى 65535 بايت كحد أعلى

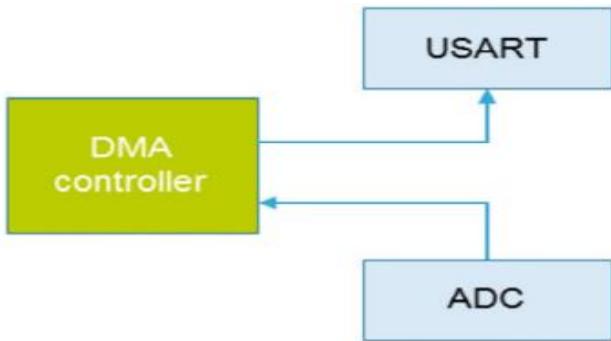
يمكن تفعيل نمط الـ Circular mode حيث يتم إعادة تحميل عنوان كل من المصدر والوجهة وحجم البيانات آلياً عند الانتهاء من عملية تبادل البيانات

Stream transfer management

يمكن أن يتم نقل البيانات بين:

Peripheral-to-peripheral transfer

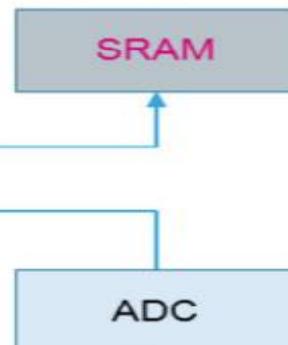
Peripheral-to-Peripheral
transfer example



Peripheral-to-Memory
transfer example

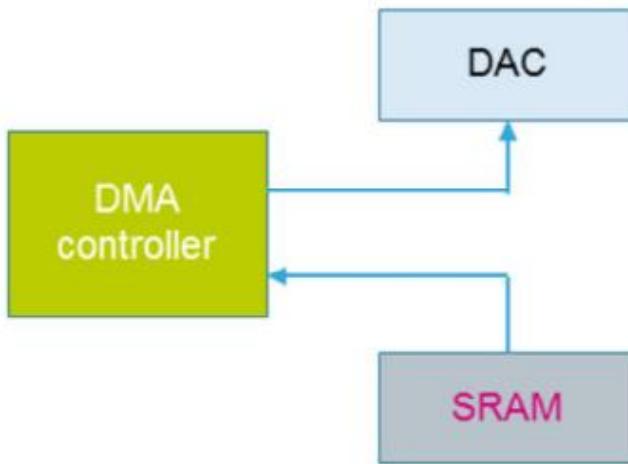
Peripheral-to-Memory transfer

DMA
controller

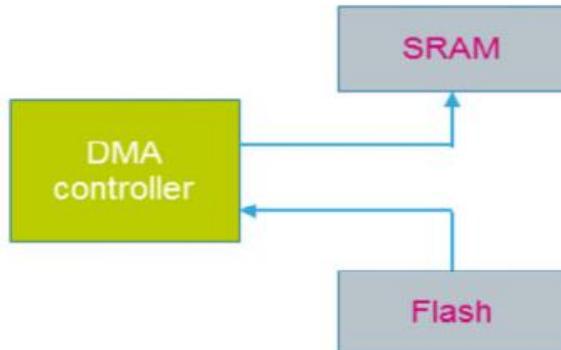


Stream transfer management

Memory-to-Peripheral
transfer example



Memory-to-Memory
transfer example
▪ No hardware request



يمكن أن يتم نقل البيانات بين:

Memory-to-peripheral transfer

Memory-to-Memory transfer

Stream transfer management

عند عدم تساوي حجم البيانات بين الوجهة والمصدر يتم استخدام تقنية **data alignment**

Source port width = 8-bit
Destination port width = 32-bit
Number of data transfers = 4

Source port width = 32-bit
Destination port width = 16-bit
Number of data transfers = 4



Address	Data[7:0]
0xFFFF_FFF0	B0
0xFFFF_FFF1	B1
0xFFFF_FFF2	B2
0xFFFF_FFF3	B3

Address	Data[31:0]
0xFFFF_FFF0	00000000
0xFFFF_FFF4	00000000
0xFFFF_FFF8	00000000
0xFFFF_FFFC	00000000

Address	Data[15:0]
0xFFFF_FFF0	B3B2B1B0
0xFFFF_FFF1	B7B6B5B4
0xFFFF_FFF2	BBBAB9B8
0xFFFF_FFF3	BFBEBDBC

Address	Data[15:0]
0xFFFF_FFF0	B1B0
0xFFFF_FFF2	B5B4
0xFFFF_FFF4	B9B8
0xFFFF_FFF6	BDBC

Circular mode

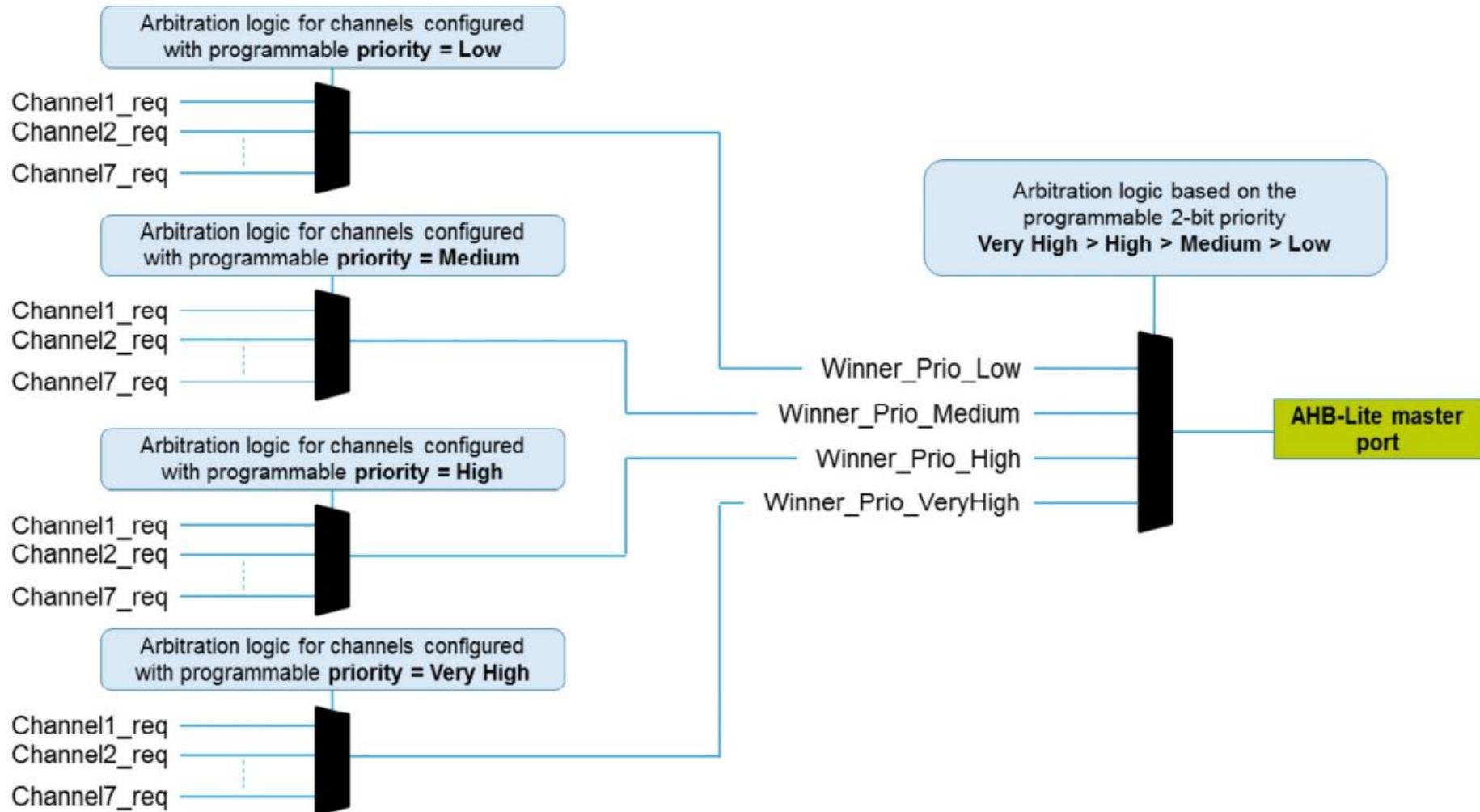
في هذا النمط يمكن استخدام الأحداث التي تزودنا بها وحدة DMA وهي:

Interrupt event	Description
Half transfer	Set when half of the data transfer size has completed
Transfer complete	Set when the full data transfer size has completed
Transfer error	Set when a bus error occurs during the data transfer

كما يمكن ضبط عدد عمليات تبادل البيانات المطلوبة، بحيث يتم البدء بأول عملية تبادل بيانات و عند الانتهاء منها يتم البدء بالعملية التالية بشكل آلي
هذا النمط لا يدعم تبادل البيانات

ضبط الأولويات لقنوات الـ DMA

هناك أربع خيارات متحدة لضبط الأولويات بين القنوات:

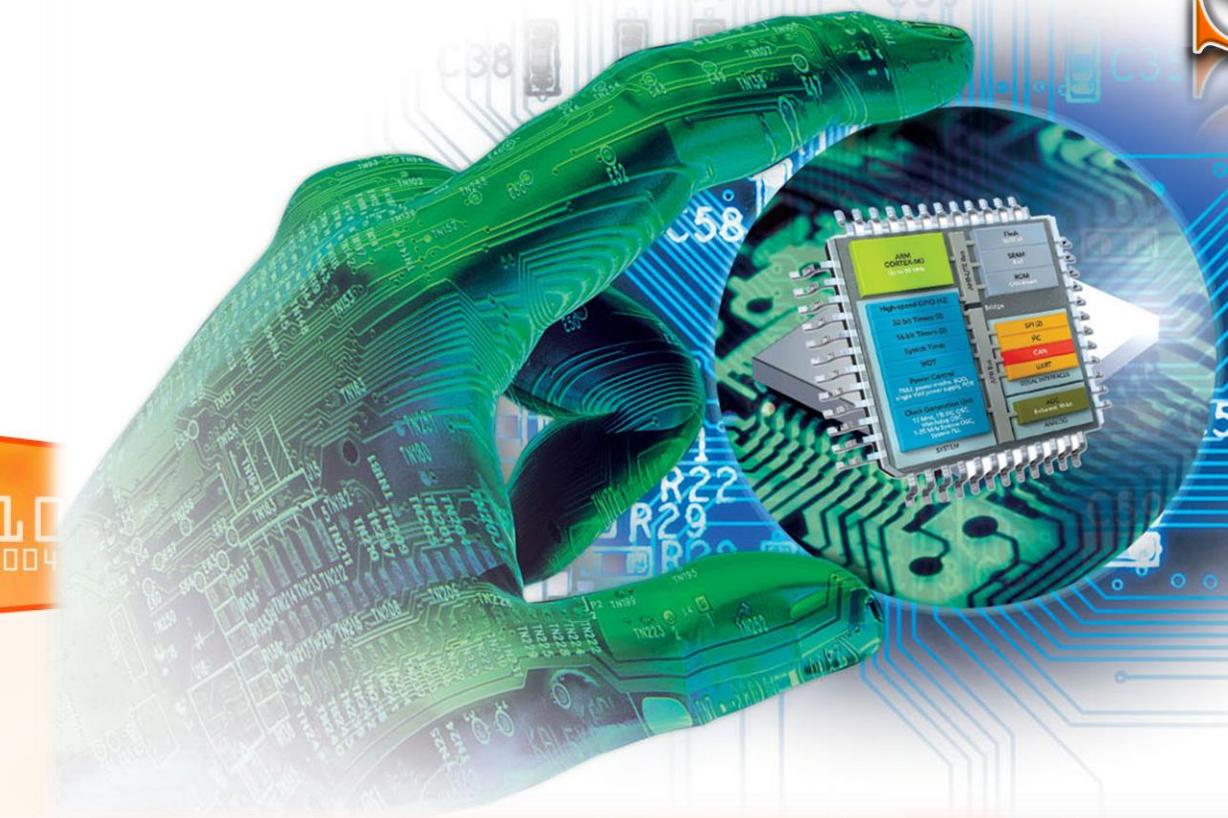


Thank you for listening

مُتَحَكِّمَات

STM32

8



موضو عات المعا ضر ظ :

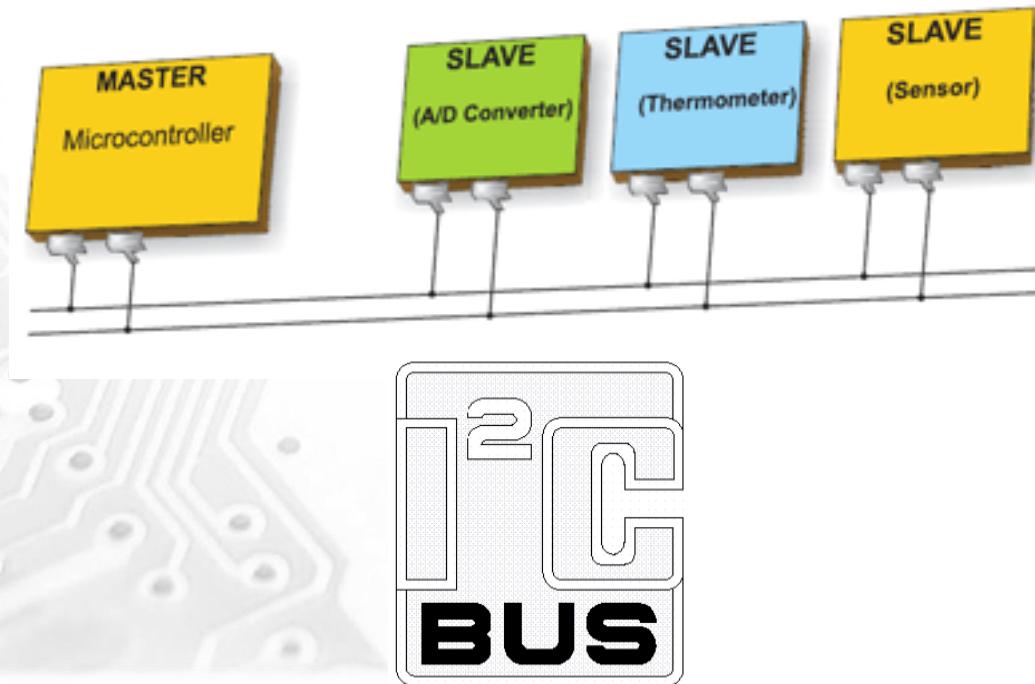
- مبدأ عمل البرتوكول I2C
- مزايا بروتوكول الاتصال التسلسلي I2C في متحكمات STM32
- المخطط الصندوقي لوحدة I2C في متحكمات STM32
- أنماط العمل المختلفة لوحدة I2C في متحكمات STM32
- دوال مكتبة HAL المستخدمة للتعامل مع وحدة I2C
- ضبط إعدادات وحدة I2C في متحكمات STM32

مقارنة بين بروتوكولات الاتصال الشائعة:

	RS232	RS485	I2C	SPI	M-wire	1-wire	USB	CAN
Sync/Async	Async	Async.	Sync.	Sync.	Sync.	Async.	Async.	Async.
Type	peer-peer	master/slaves	multi-master	multi-master	master/slaves	master/slaves	host/device	multi-master
Duplex	full	half	half	full	full	half	half	half
Signaling	single-ended	Differential	single-ended	single-ended	single-ended	single-ended	Differential	Differential
Max Devices No.	2	32, 128, 256	40 (cap=400pf)	8 (cap, circuit)	8 (cap, circuit)	20 (cap, power)	127 per controller	2048
Data Rate	Up to 115Kbps	Up to 35Mbps	Std.: 100kbps Fast: 400kbps Hi: 3.4Mbps	Up to 10Mbps	Up to 1Mbps	Std.: 16.3Kbps Overdrive: 142kbps	Low: 1.5Mbps Full: 12Mbps Hi : 480Mbps	Up to 1Mbps
Max. Length	15m	1200m (at 100kbps)	6m	3m	3m	300m	5m	1000m (at 62kbps)
Pin Count	2* (Tx, Rx)	2 (A, B)	2 (SDA, SCL)	3 + SS* (SI, SO, SCK)	3 + SS* (DI, DO, SK)	1 (IO)	2 (A+, A-)	2 (CAN_H, _L)
Interfacing	HW	HW	SW HW	HW SW	HW SW	HW & SW	protocol stack	HW & SW
Flow Control	HW or SW handshake	HW or SW handshake	Acknowledge from slave	None	None	CRC, Pulling	Polling by controller	CSMA / CDAMP

I2C

Inter Integrated Circuit



بروتوكول الاتصال التسلسلي I²C (Inter Integrated Circuit)

- تم تطويره من قبل شركة Philips في أوائل 1980.
- وضع أساساً لربط متحكم مع بعض المحيطيات في أجهزة التلفاز.
- هو بروتوكول متزامن h-duplex يعتمد على خطين (SDA, SCL).
- خط البيانات **SDA** هو خط ثبائي الاتجاه.
- خط التزامن **SCL** تتولد إشارته من عنصر Master وله حالتين:
 - أحادي الاتجاه في الأنظمة **Single-master** (1)
 - ثبائي الاتجاه في الأنظمة **Multi-master** (2)

بروتوكول الاتصال التسلسلي I²C (Inter Integrated Circuit)

- خرج الأجهزة في البروتوكول I²C هو من نوع .Open collector
- يجب ربط مقاومات رفع خارجية لكلا SDA, SCK.
- في حال البطالة تكون القيمة المنطقية على كلا الخطين "1".
- كل جهاز يملك عنوان فريد (Unique Address) يستخدم لتحديد جهاز الـ "Master" المراد التخاطب معه من الجهاز "Slave".
- العنوان الخاص بكل جهاز يكون مؤلف من 7-Bit (112 nodes).
- يمكن أيضاً أن يكون العنوان الخاص بطول 10-bit (1008-node).

بروتوكول الاتصال التسلسلي I²C (Inter Integrated Circuit)

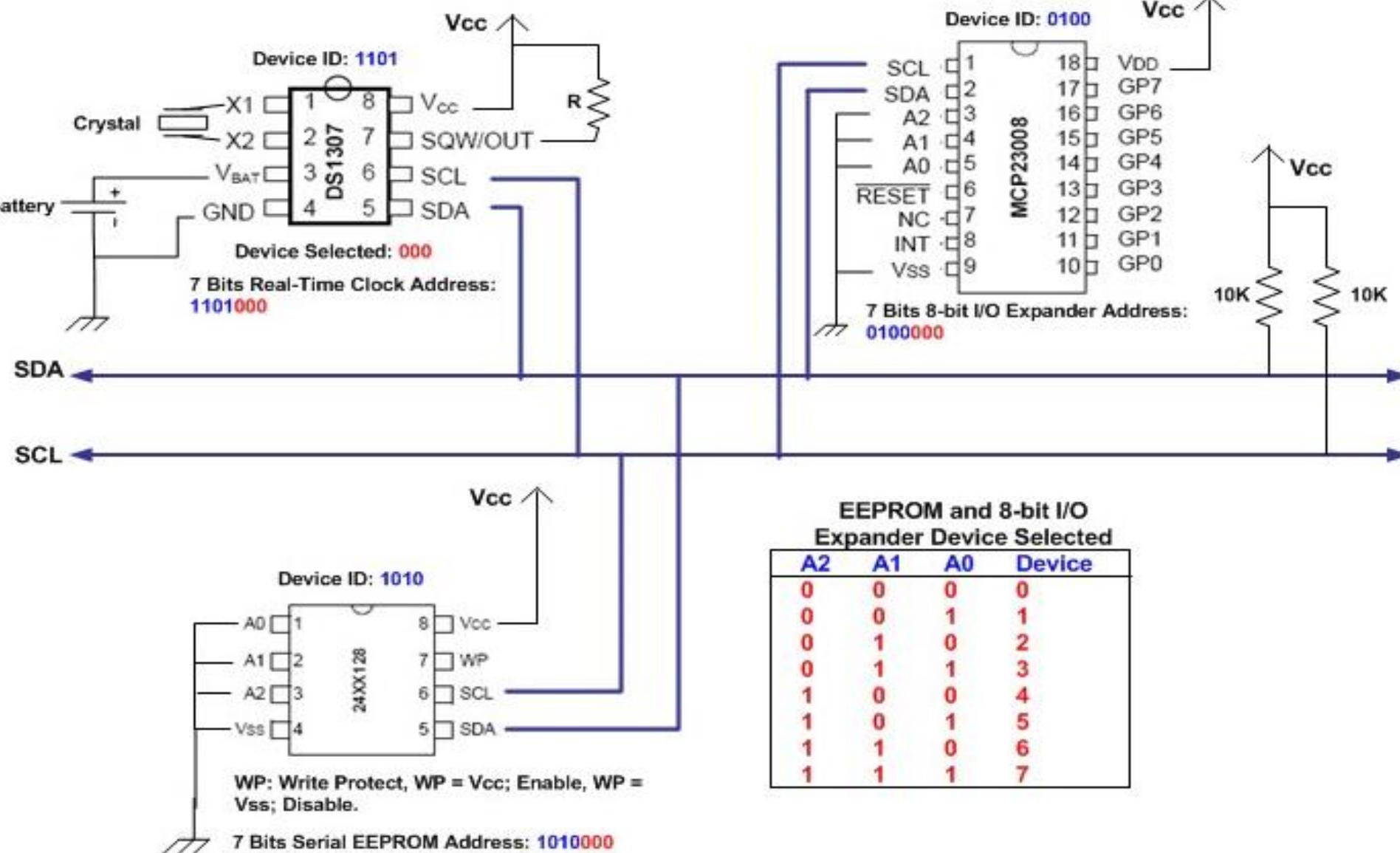
- إن عدد الأجهزة في البروتوكول I2C على خط النقل يعتمد مباشرة على سعة الخط، حيث أن القيمة الأعظمية للسعة يجب أن لا تتجاوز 400، وغالباً تكون سعة كل جهاز بحدود 10pF (40 جهاز).
- الطول الأقصى على ناقل I2C يمكن أن يصل إلى 6 أمتار.
- تختلف سرعة نقل البيانات على ناقل I2C حسب المعيار الشريحة:

Std.: 100kbps (1)

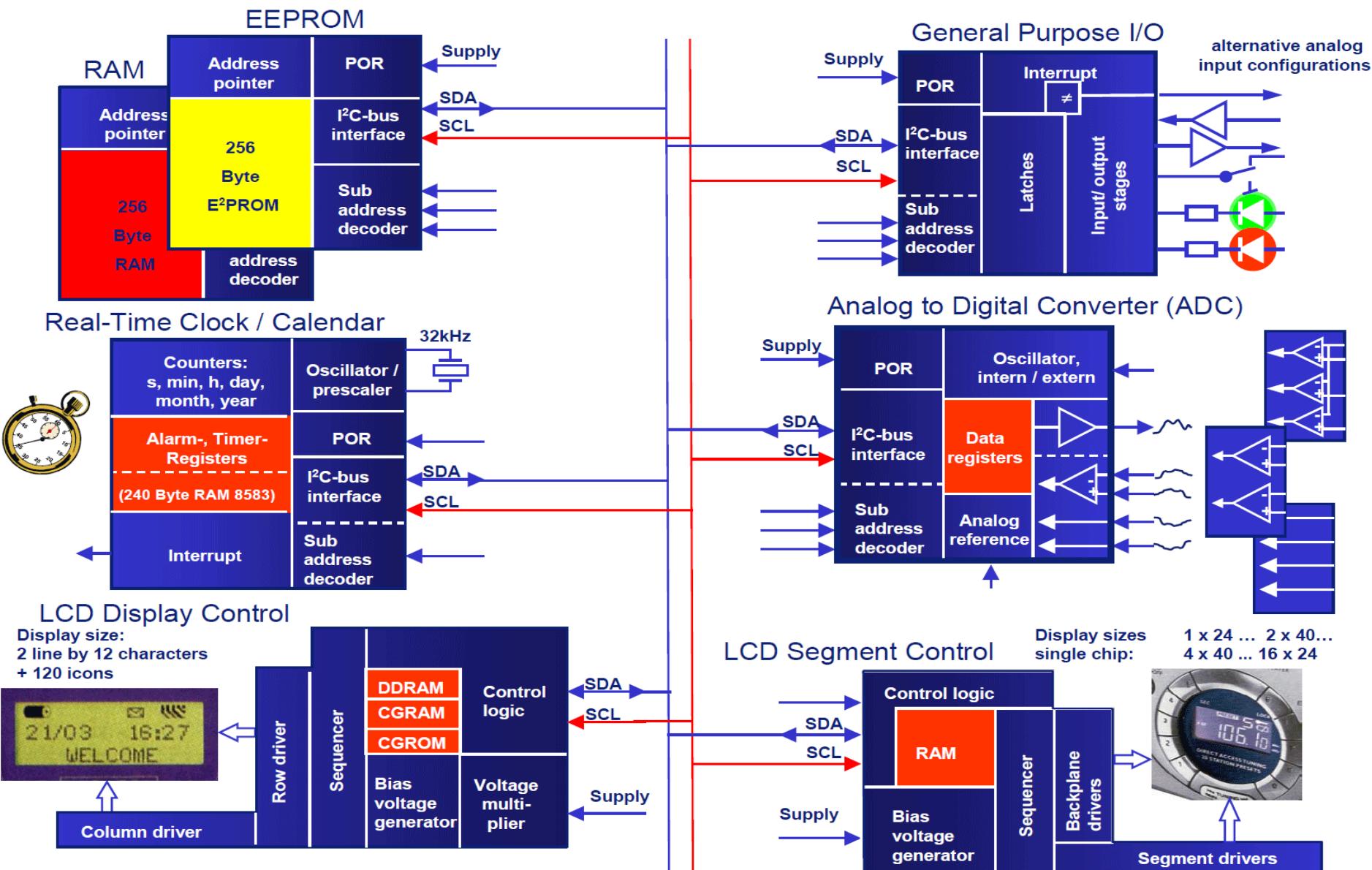
Fast: 400kbps (2)

Hi: 3.4Mbps (3)

بروتوكول الاتصال التسلسلي I2C في متحكمات STM32



بروتوكول الاتصال التسلسلي I2C في متحكمات STM32



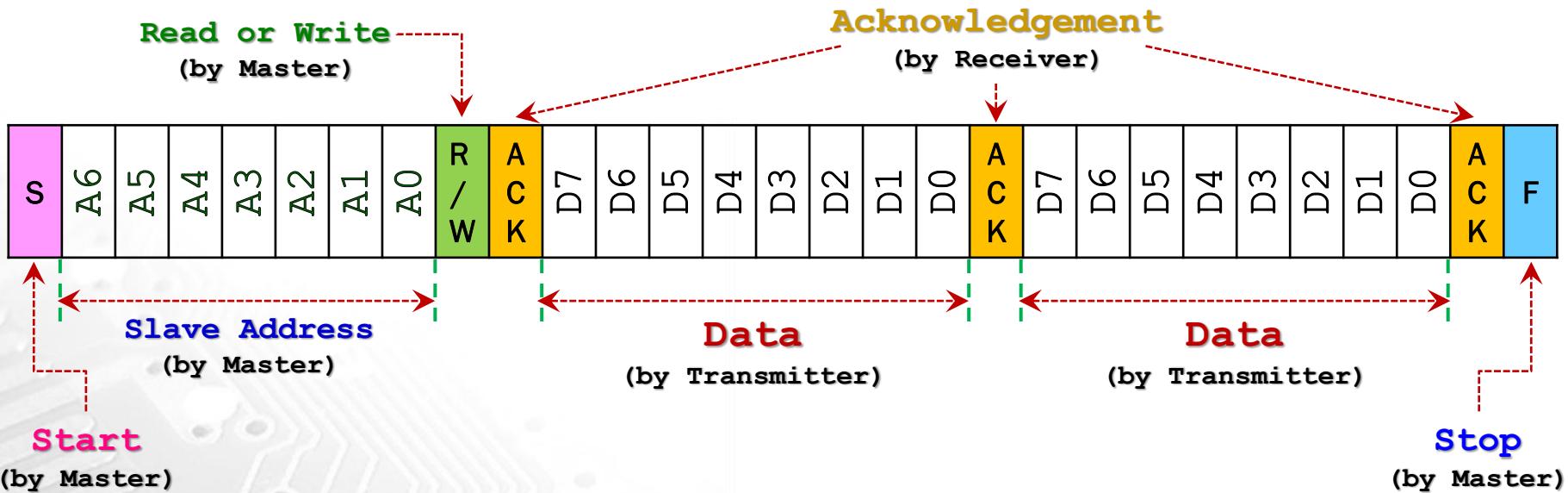
(مبدأ العمل في بروتوكول الاتصال التسلسلي I²C)

- إن مبدأ العمل يعتمد على بروتوكول Master-Slave حيث أن جهاز الـ Master يبتدئ عملية الاتصال مع جهاز Slave ذو عنوان محدد وفقاً للتسلسل التالي:
- (1) يقوم Master بإرسال حالة "Start" (بت بداية الإرسال) يقوم بإعلام جميع أجهزة Slaves الموجودة على Bus للاستعداد للبيانات القادمة على SDA.
 - (2) يقوم Master بإرسال عنوان (7-bit) الجهاز المعني "Slave Address" إضافة إلى بت تحديد العملية (قراءة "1" | كتابة "0").
 - (3) يقوم Slave المعني بالعنوان المرسل من الـ Master بالاستجابة بإشارة مصادقة "ACK" يعلم من خلالها الـ Master بوجوده على الناقل.

(مبدأ العمل في بروتوكول الاتصال التسلسلي I²C)

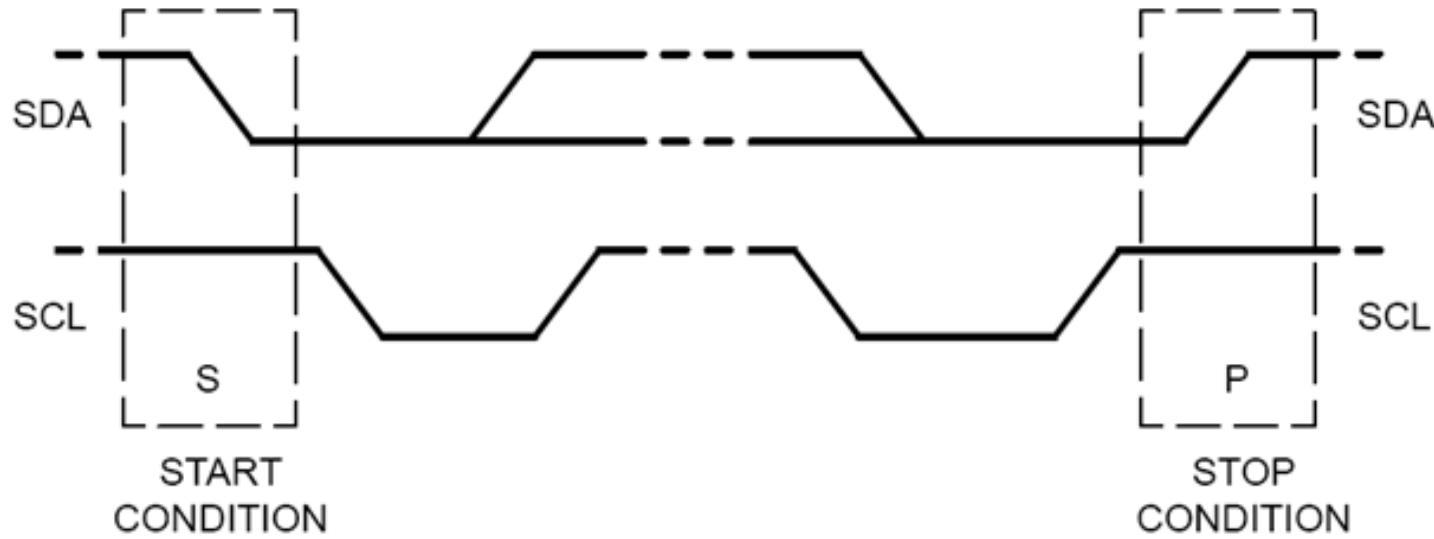
- (4) بعد أن تم تحديد طرف في التخاطب، تبدأ عملية تراسل البيانات بين الطرفين ويتم تحديد المرسل والمستقبل من خلال بت تحديد العملية (قراءة "1" \ كتابة "0") في بait العنوان.
- (5) كلما تم إرسال 8-bit من المرسل يقوم المستقبل (أياً يكن S or M) بالتأكيد بإرسال إشارة مصادقة "ACK" (1-bit). وتستمر التراسل هكذا...
- (6) عندما تنتهي عملية التراسل بين الطرفين يقوم **Master** بإرسال حالة "Stop" (بت نهاية الإرسال) مشيراً إلى انتهاء العملية الجارية.

(مبدأ العمل في بروتوكول الاتصال التسلسلي I²C)



Master controls the clock signal

(مبدأ العمل في بروتوكول الاتصال التسلسلي I²C)

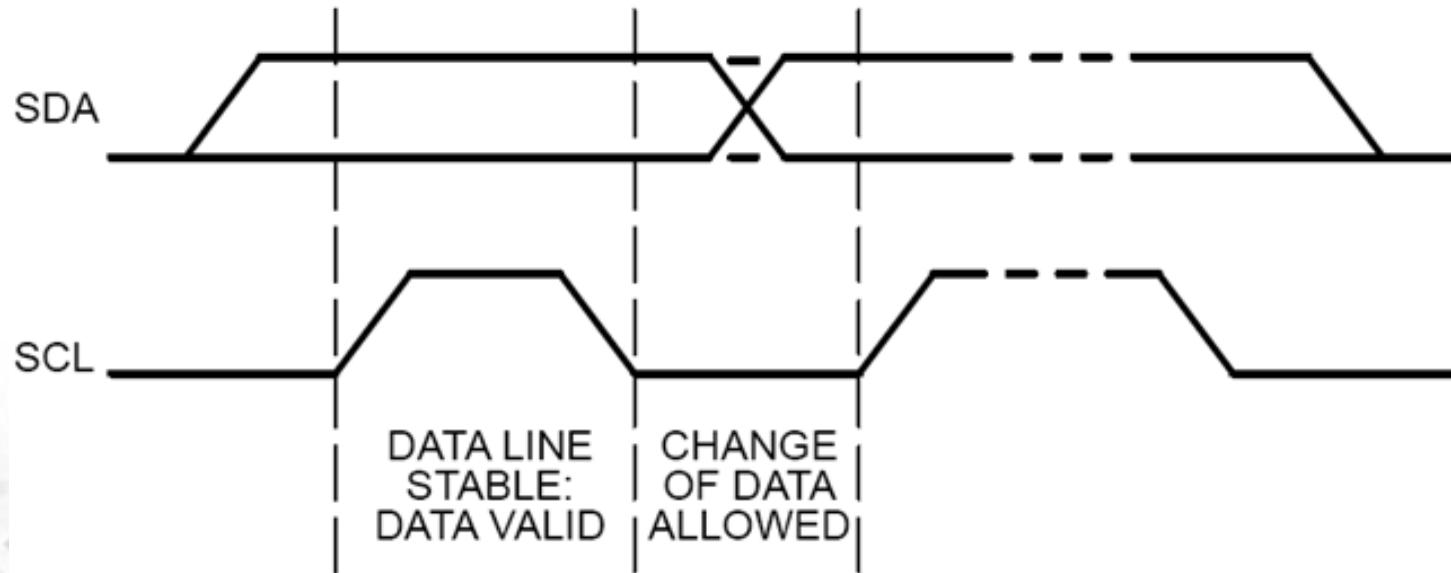


high-to-low transition
of the **SDA** line while
SCL line is high...

low-to-high transition
of the **SDA** line while
SCL line is high...

Ack: Receiver pulls SDA **Low** while
Transmitter allows it to float **High**.

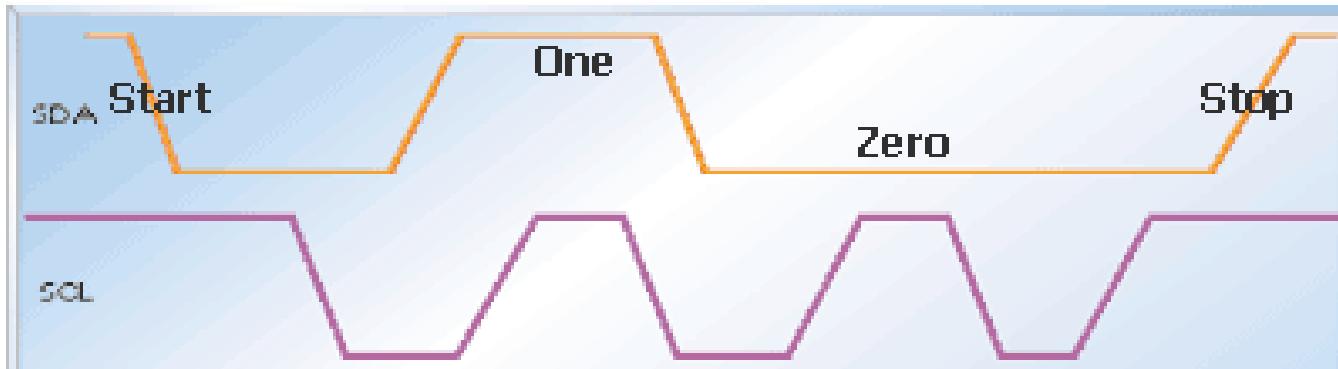
(مبدأ العمل في بروتوكول الاتصال التسلسلي I²C)



Data transition while **SCL** is **low**...

Data validation while **SCL** is **high**...

(مبدأ العمل في بروتوكول الاتصال التسلسلي I²C)



- ❑ **Write Transaction (Master Transmitting):** Master issues **Stop condition (P)** after last byte of data.
- ❑ **Read Transaction (Master Receiving):** Master does not acknowledge final byte, just issues **Stop condition (P)** to tell the slave the transmission is done.

مزايا بروتوكول الاتصال التسلسلي I2C في متحكمات STM32

يمكن أن تعمل الوحدة الطرفية Slave أو Master كـ I2C

بإمكانها توليد واكتشاف عناوين بطول 7bit أو 10bit

تحتوي على مرشحات ضجيج تشابهية ورقمية ويمكن ضبط إعداداتها من خلال الكود

يجب ضبط إعدادات أقطاب الـ Open Drain كـ I2C ووصل مقاومات رفع خارجية معها

مصدر الساعة للوحدة الطرفية قد يكون :

system clock (1)

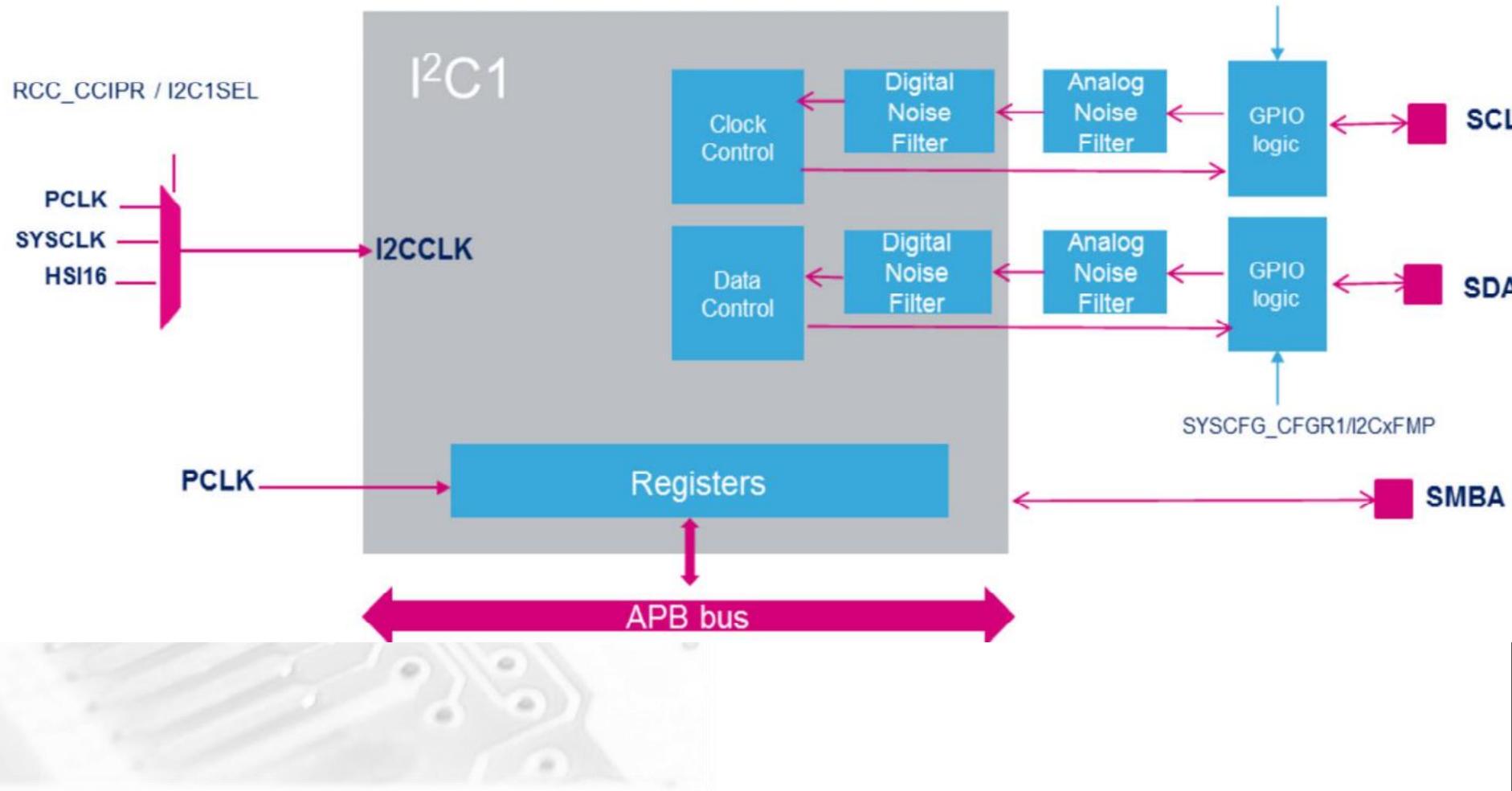
high-speed internal 16 MHz RC oscillator (2)

تدعم سرعتي نقل للبيانات على ناقل I2C :

Std.: 100kbps (1)

Fast: 400kbps (2)

المخطط الصندوق في لوحة STM32 I2C في منحكمات



أنماط العمل المختلفة لوحدة I2C في متغير STM32

- Slave transmitter**
- Slave receiver**
- Master transmitter**
- Master receiver**

حيث يكون الوضع الافتراضي لوحدة I2C هو Slave mode ، كما يمكن

أن تعمل بنمط :

- Polling**
- Interrupt**
- DMA**

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة I2C في وضع الـ Polling

Master Transmission

```
HAL_I2C_Master_Transmit (I2C_HandleTypeDef * hi2c,  
                           uint16_t DevAddress, uint8_t* pData, uint16_t Size,  
                           uint32_t Timeout);
```

حيث: **DevAddress**: عنوان الجهاز وهو مكون من 7 بت ويجب إزاحتة نحو اليسار قبل استدعاء الطرفية (لتحويل من 8 بت ل 7 بت)

pData: مؤشر الى **data buffer** للبيانات المراد إرسالها، **Size** : حجم البيانات، **Timeout**: مدة المهلة قبل فشل الإرسال

Master Reception

```
HAL_I2C_Master_Receive (I2C_HandleTypeDef * hi2c,  
                           uint16_t DevAddress, uint8_t* pData, uint16_t Size,  
                           uint32_t Timeout);
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة I2C في وضع الـ Polling

Slave Transmission

```
HAL_I2C_Slave_Transmit (I2C_HandleTypeDef * hi2c, uint8_t  
* pData, uint16_t Size, uint32_t Timeout);
```

Slave Reception

```
HAL_I2C_Slave_Receive (I2C_HandleTypeDef * hi2c, uint8_t  
* pData, uint16_t Size, uint32_t Timeout);
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة I2C في وضع الـ Interrupt

Master Transmission

```
HAL_I2C_Master_Transmit_IT (I2C_HandleTypeDef * hi2c,  
uint16_t DevAddress, uint8_t * pData, uint16_t Size);
```

- بعد استدعاء هذه الدالة تقوم وحدة I2C ببدء عملية الإرسال لـ كامل البایتات الموجودة ضمن الـ buffer وعند الانتهاء يتم استدعاء التابع التالي:

```
void HAL_I2C_MasterTxCpltCallback (I2C_HandleTypeDef *  
hi2c)  
{  
    // TX Done .. Do Something!  
}
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة I2C في وضع الـ Interrupt

Master Reception

```
HAL_I2C_Master_Receive_IT (I2C_HandleTypeDef * hi2c,  
                           uint16_t DevAddress, uint8_t * pData, uint16_t Size);
```

- بعد استدعاء هذه الدالة تقوم وحدة I2C ببدء عملية الاستقبال لكامل البيانات الموجودة ضمن الـ buffer وعند الانتهاء يتم استدعاء التابع التالي:

```
void HAL_I2C_MasterRxCpltCallback (I2C_HandleTypeDef *  
hi2c)  
{  
    // RX Done .. Do Something!  
}
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة I2C في وضع ال DMA

Master Transmission

```
HAL_I2C_Master_Transmit_DMA (I2C_HandleTypeDef *  
hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size);
```

- بعد استدعاء هذه الدالة تقوم وحدة I2C ببدء عملية الإرسال لـ كامل البایتات الموجودة ضمن الـ buffer وعند الانتهاء يتم استدعاء التابع التالي:

```
void HAL_I2C_MasterTxCpltCallback (I2C_HandleTypeDef *  
hi2c)  
{  
    // TX Done .. Do Something!  
}
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة I2C في وضع ال DMA

Master Reception

```
HAL_I2C_Master_Receive_DMA (I2C_HandleTypeDef * hi2c,  
uint16_t DevAddress, uint8_t * pData, uint16_t Size);
```

- بعد استدعاء هذه الدالة تقوم وحدة I2C ببدء عملية الاستقبال لكامل البايتات الموجودة ضمن ال buffer وعند الانتهاء يتم استدعاء التابع التالي:

```
void HAL_I2C_MasterRxCpltCallback (I2C_HandleTypeDef *  
hi2c)  
{  
    // RX Done .. Do Something!  
}
```

دوال مكتبة HAL المستخدمة للتتعامل مع وحدة I2C

- إذا أردنا إرسال أو استقبال بيانات من مسجلات محددة ضمن الجهاز المرسل له ، على سبيل المثال إذا أردنا إرسال بيانات إلى مسجلات محددة من ذاكرة eeprom متصلة مع المتحكم عبر الناقل I2C نستخدم الدوال التالية:

□ لاستقبال البيانات (نقط polling)

```
HAL_I2C_Mem_Read(I2C_HandleTypeDef * hi2c, uint16_t DevAddress,uint16_t MemAddress,uint16_t MemAddSize,uint8_t * pData,uint16_t Size, uint32_t Timeout )
```

: حيث

: عنوان الذاكرة الداخلية ضمن الجهاز المرسل له :MemAddress

: حجم الذاكرة الداخلية :MemAddSize

ومن أجل نمط المقاطعة نستخدم الدالة : HAL_I2C_Mem_Read_IT

ومن أجل نمط ال DMA نستخدم الدالة HAL_I2C_Mem_Read_DMA

دوال مكتبة HAL المستخدمة للتتعامل مع وحدة I2C

لارسال البيانات (نط polling

```
HAL_I2C_Mem_Write (I2C_HandleTypeDef * hi2c, uint16_t  
DevAddress,uint16_t MemAddress,uint16_t  
MemAddSize,uint8_t * pData,uint16_t Size, uint32_t  
Timeout )
```

: حيث

ومن أجل نمط المقاطعة نستخدم الدالة : **HAL_I2C_Mem_Write_IT** :
ومن أجل نمط ال DMA نستخدم الدالة **HAL_I2C_Mem_Write_DMA**

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة I2C في وضع ال DMA

□ فحص حالة جهاز مرتبط على الناقل :I2C

```
AL_I2C_IsDeviceReady (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint32_t Trials, uint32_t Timeout);
```

- وتستخدم هذه الدالة للتأكد من وجود slave device على الناقل I2C وأنه يعمل بشكلٍ جيد أم لا.

- تفعيل وحدة I2C :

I2C1 Mode and Configuration

Mode

I2C

Disable

I2C

SMBus-Alert-mode

SMBus-two-wire-Interface

ضبط بارامترات وحدة I2C وتتضمن حالتين :

(1) عندما تكون الوحدة تعمل بنمط Master عنها يجب ضبط كل من سرعة نقل البيانات إما Fast أو standard كما يجب ضبط تردد ساعة الوحدة

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

Master Features

I2C Speed Mode	Standard Mode
I2C Clock Speed (Hz)	100000

Slave Features

Clock No Stretch Mode	Disabled
Primary Address Length selection	7-bit
Dual Address Acknowledged	Disabled
Primary slave address	0
General Call address detection	Disabled

ضبط بaramترات وحدة I2C وتتضمن حالتين :

- (2) عندما تكون الوحدة تعمل بنمط Slave عذها يمكن ضبط كل من عدد بิตات العنونة وإعطاء الوحدة عنوان محدد بالإضافة إلى تفعيل أو تعطيل خاصية clock stretching والتي عند تفعيلها تقوم بالمحافظة على خط الـ SCL في المستوى المنخفض لمنع أي جهاز من إجراء أي عملية لحين تحرير خط الـ master

Configuration

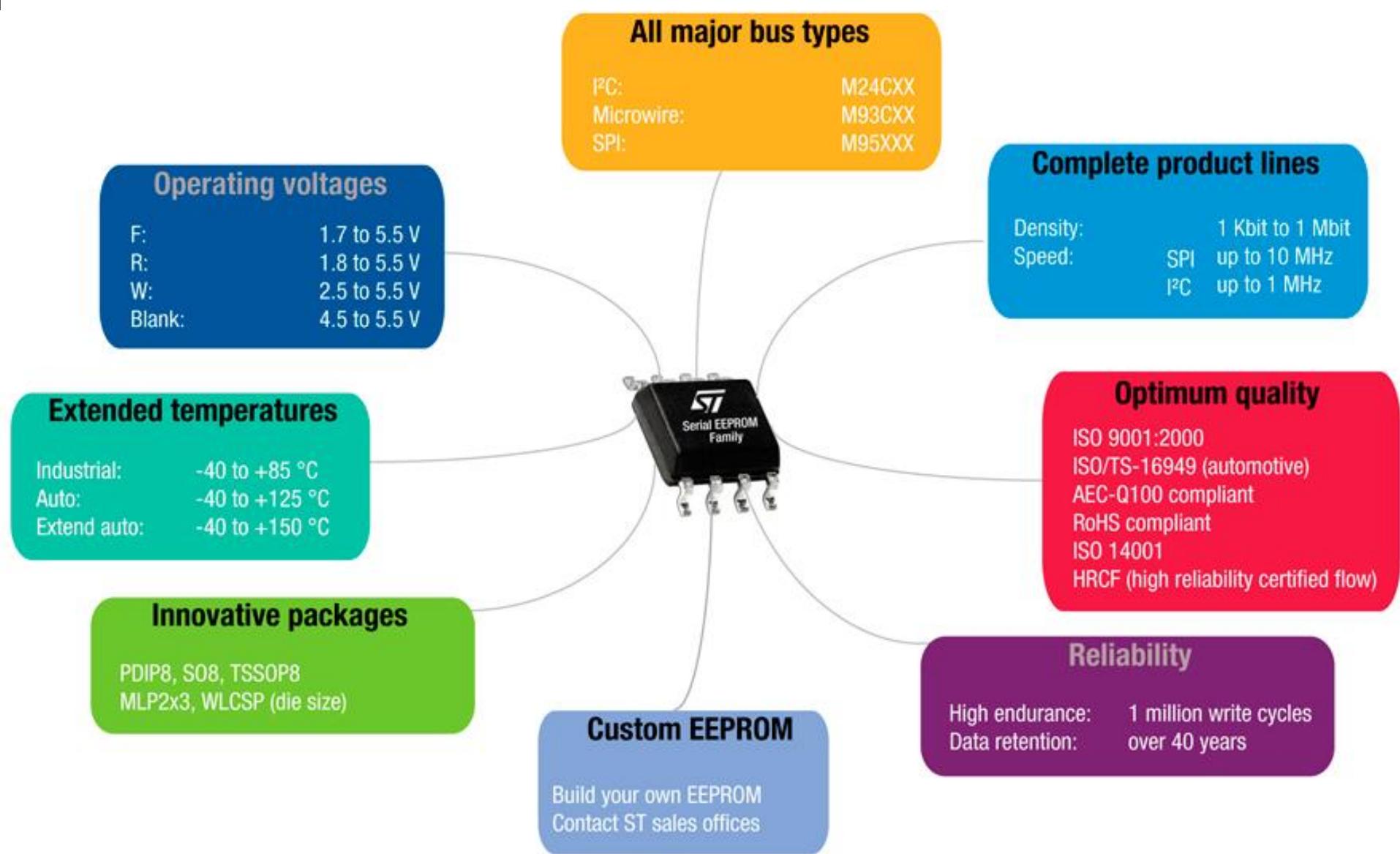
Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

Configure the below parameters :

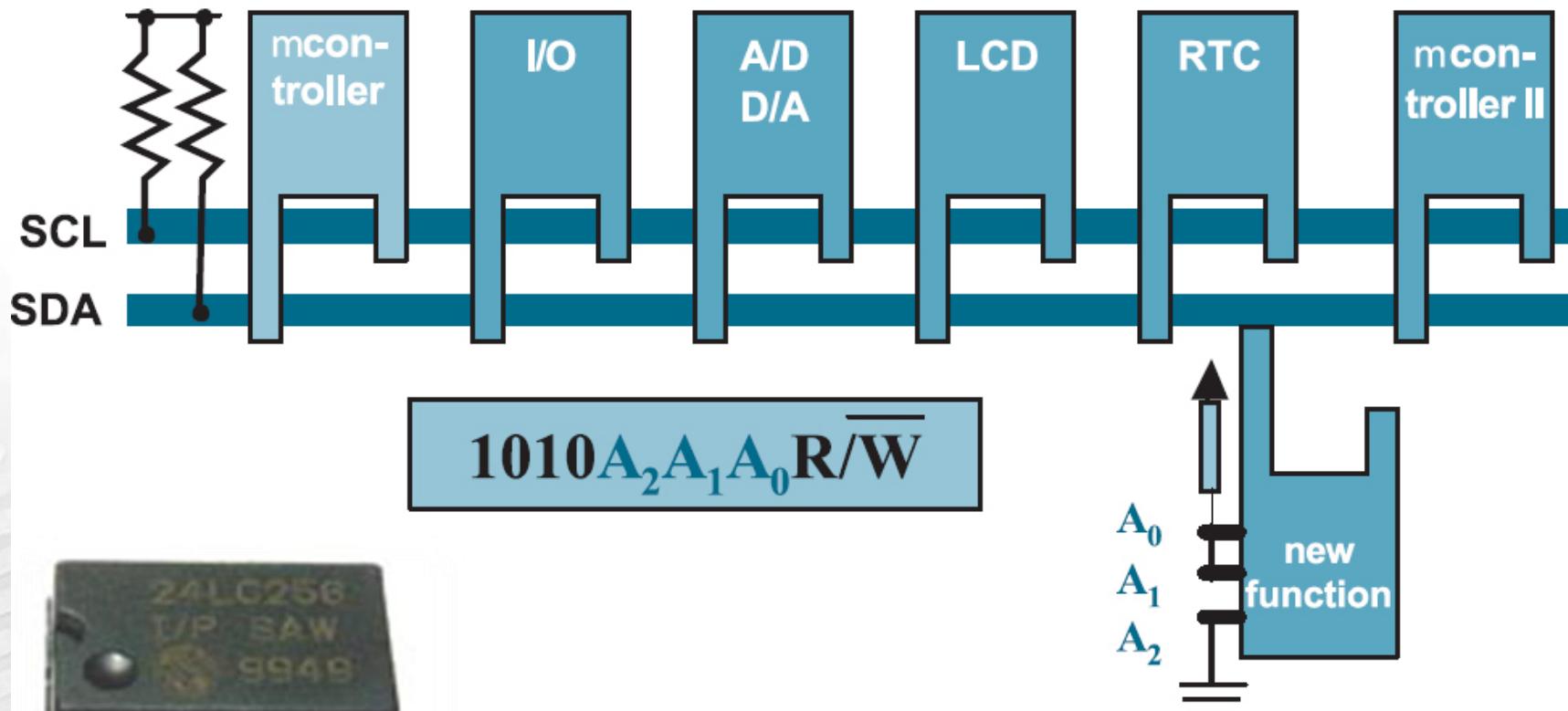
Master Features		Slave Features	
I2C Speed Mode	Standard Mode	Clock No Stretch Mode	Disabled
I2C Clock Speed (Hz)	100000	Primary Address Length selection	7-bit
Dual Address Acknowledged	Disabled	Primary slave address	0
General Call address detection	Disabled		

التطبيق العملي: ربط ذاكرة eeprom مع المتحكم STM32G0B1CE



التطبيق العملي: ربط ذاكرة eeprom مع المتحكم

Unleash your Creativity!



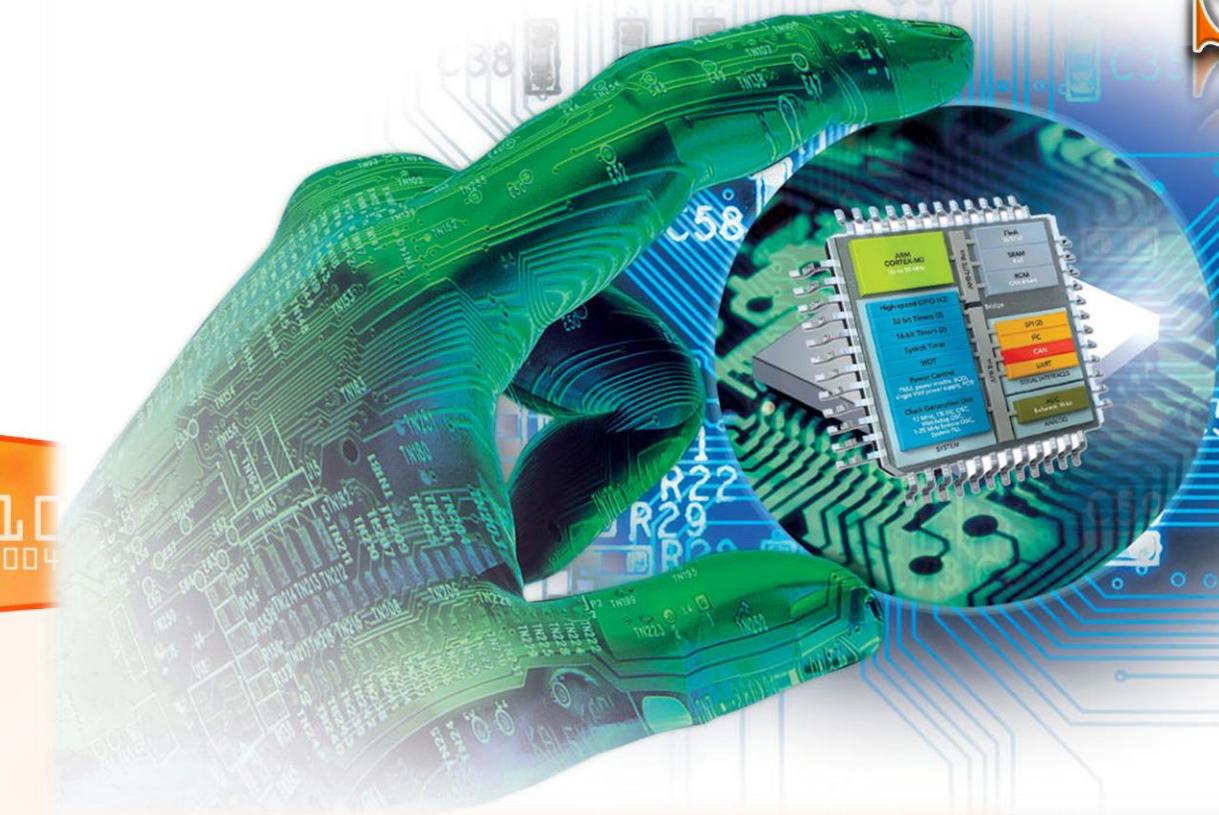
AT24C02 Datasheet

Thank you for listening

مُتَحَكِّمَات

STM32

9



موضع عاًن المعاًضرة:

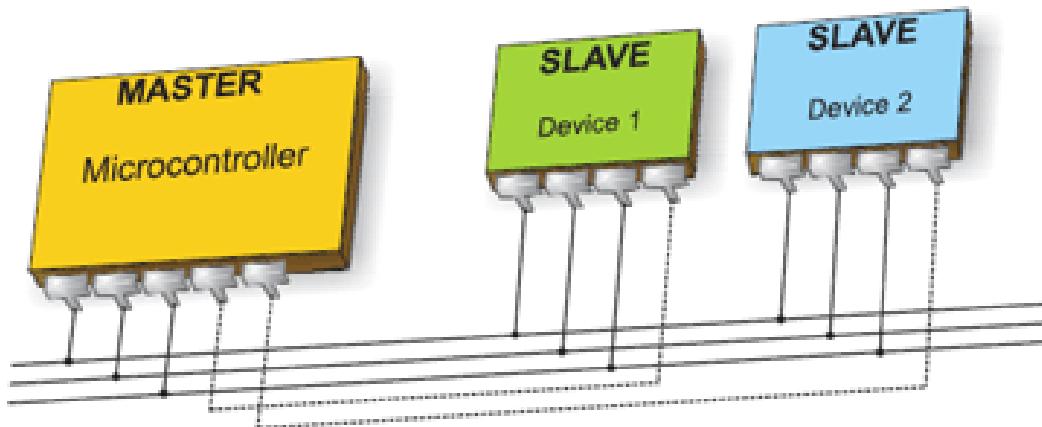
- مبدأ عمل البرتوكول SPI
- مزايا بروتوكول الاتصال التسلسلي SPI في متحكمات STM32
- المخطط الصندوقي لوحدة SPI في متحكمات STM32
- أنماط العمل المختلفة لوحدة SPI في متحكمات STM32
- دوال مكتبة HAL المستخدمة للتعامل مع وحدة SPI
- ضبط إعدادات وحدة SPI في متحكمات STM32

مقارنة بين بروتوكولات الاتصال الشائعة:

	RS232	RS485	I2C	SPI	M-wire	1-wire	USB	CAN
Sync/Async	Async	Async.	Sync.	Sync.	Sync.	Async.	Async.	Async.
Type	peer-peer	master/slaves	multi-master	multi-master	master/slaves	master/slaves	host/device	multi-master
Duplex	full	half	half	full	full	half	half	half
Signaling	single-ended	Differential	single-ended	single-ended	single-ended	single-ended	Differential	Differential
Max Devices No.	2	32, 128, 256	40 (cap=400pf)	8 (cap, circuit)	8 (cap, circuit)	20 (cap, power)	127 per controller	2048
Data Rate	Up to 115Kbps	Up to 35Mbps	Std.: 100kbps Fast: 400kbps Hi: 3.4Mbps	Up to 10Mbps	Up to 1Mbps	Std.: 16.3Kbps Overdrive: 142kbps	Low: 1.5Mbps Full: 12Mbps Hi : 480Mbps	Up to 1Mbps
Max. Length	15m	1200m (at 100kbps)	6m	3m	3m	300m	5m	1000m (at 62kbps)
Pin Count	2* (Tx, Rx)	2 (A, B)	2 (SDA, SCL)	3 + SS* (SI, SO, SCK)	3 + SS* (DI, DO, SK)	1 (IO)	2 (A+, A-)	2 (CAN_H, _L)
Interfacing	HW	HW	SW HW	HW SW	HW SW	HW & SW	protocol stack	HW & SW
Flow Control	HW or SW handshake	HW or SW handshake	Acknowledge from slave	None	None	CRC, Pulling	Polling by controller	CSMA / CDAMP

SPI

Serial Peripheral Interface



بروتوكول الاتصال التسلسلي SPI (Serial Peripheral Interface)

- تم تطويره من قبل شركة Motorola.
- هو بروتوكول متزامن Full-duplex.
- لا يتم الانتخاب من خلال العناوين وإنما من خلال خط اختيار SS.
- مناسب أكثر للتطبيقات ذات Streaming Data مثل ADC.
- إن عدد الأجهزة على خط النقل بحدود 8~12 جهاز.
- الطول الأعظمي على ناقل SPI يمكن أن يصل إلى 3 أمتر.
- سرعة نقل البيانات على ناقل SPI تصل إلى 10Mbit.

بروتوكول الاتصال التسلسلي SPI (Serial Peripheral Interface)

- لا يملك آلية للتأكد فيما إذا كان الجهاز الآخر على الخط أم لا.
- لا يملك آلية مصافحة للتأكد من وصول البيانات.
- التركيب الداخلي للنافذة SPI هو عبارة عن مسجل إزاحة S-to-P.
- يعتبر بروتوكول SPI أكثر استخداماً وانتشاراً من I2C.
- حزمة البيانات يمكن أن تكون بطول أكبر من 8-bit خلافاً لـ I2C.
- من أشهر التطبيقات: ADC, DAC, Data Flash Memory, MMC, Serial EEPROM, RTC, LCD, Sensors.

بروتوكول الاتصال التسلسلي SPI (Serial Peripheral Interface)

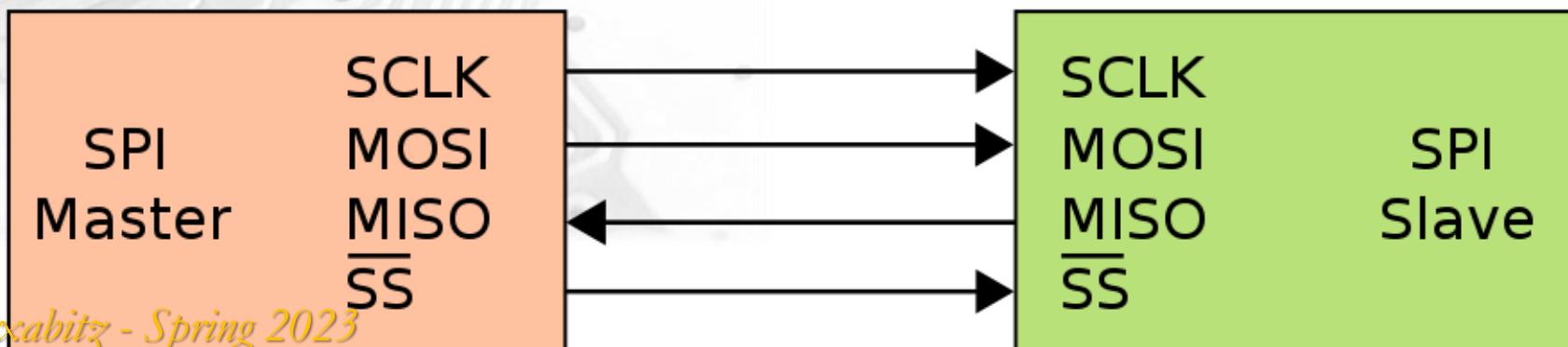
يعتمد على أربع خطوط للتalking مع الوحدات المحيطة على الناقل:

Master Output / Slave Input :**MOSI** ○

Master Input / Slave Output :**MISO** ○

Serial Clock - Synchronization :**SCK** ○

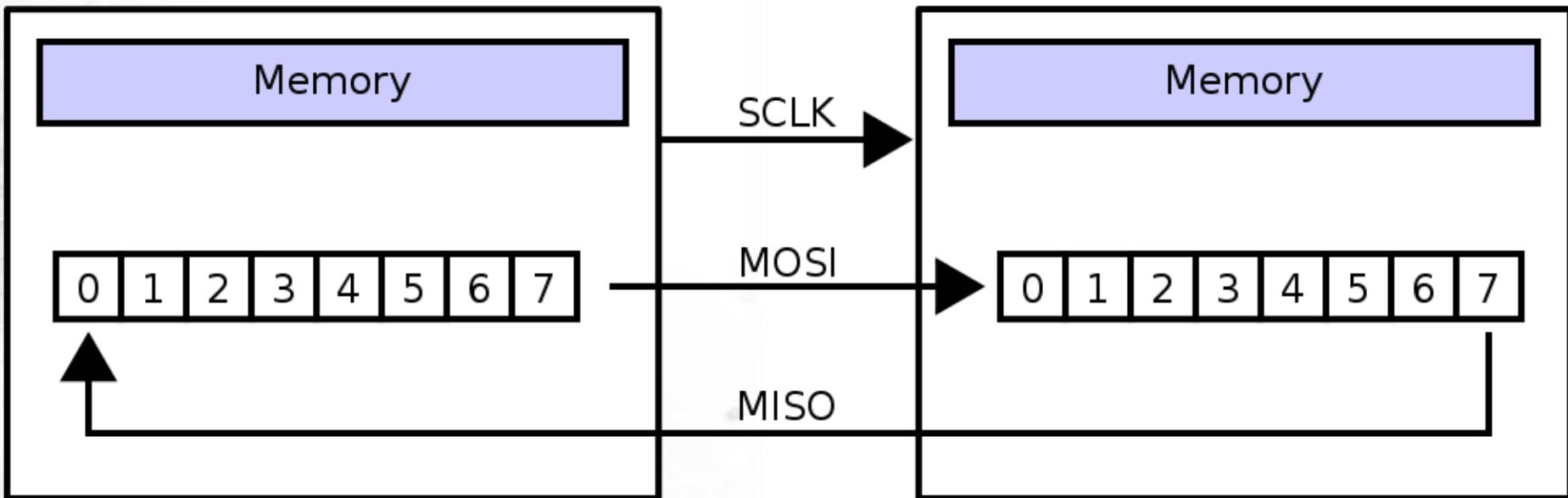
Slave Select (Optional) :**~SS** ○



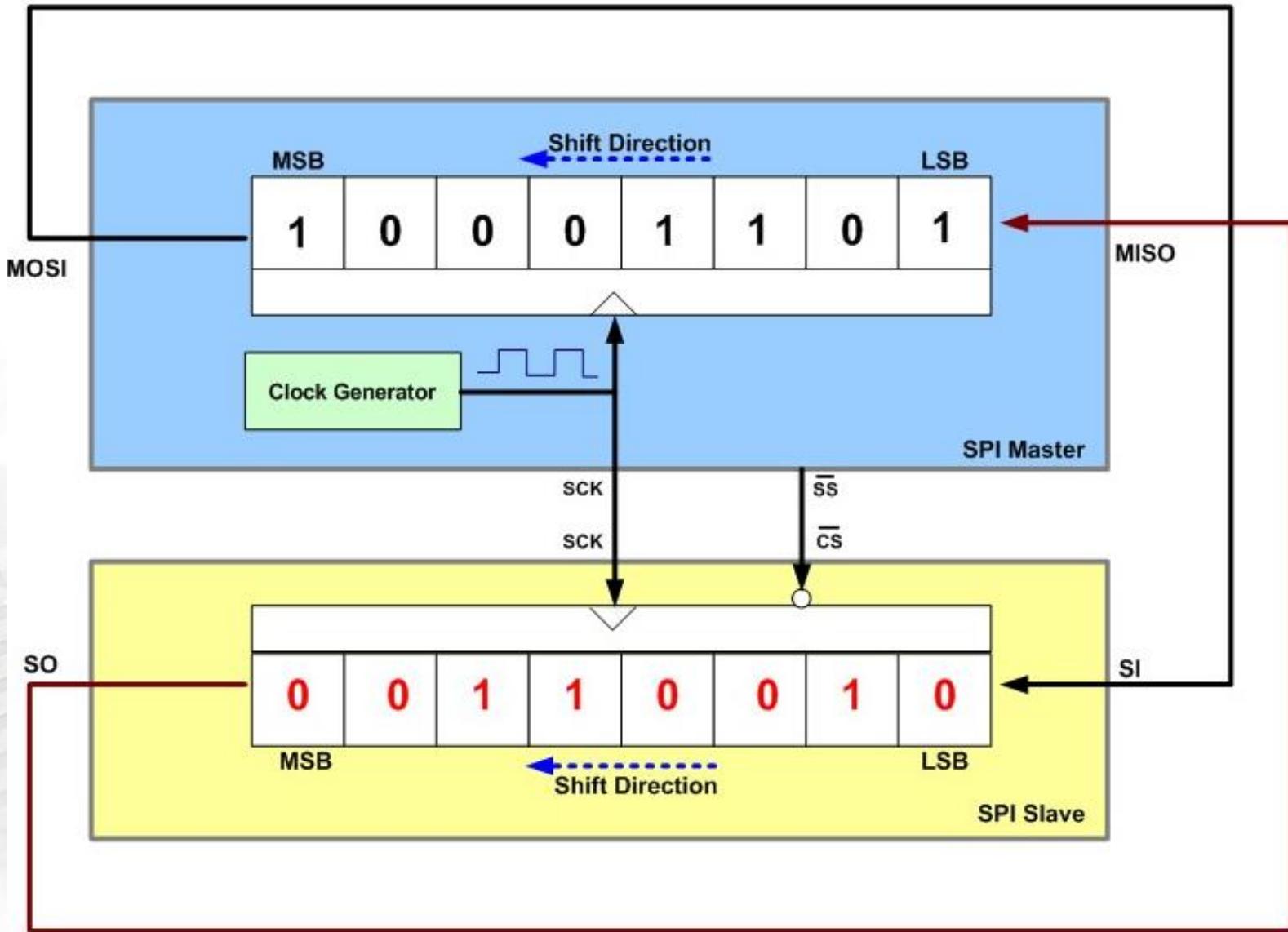
❖ مبدأ تراسل البيانات في ناقل الاتصال التسلسلي SPI

Master

Slave

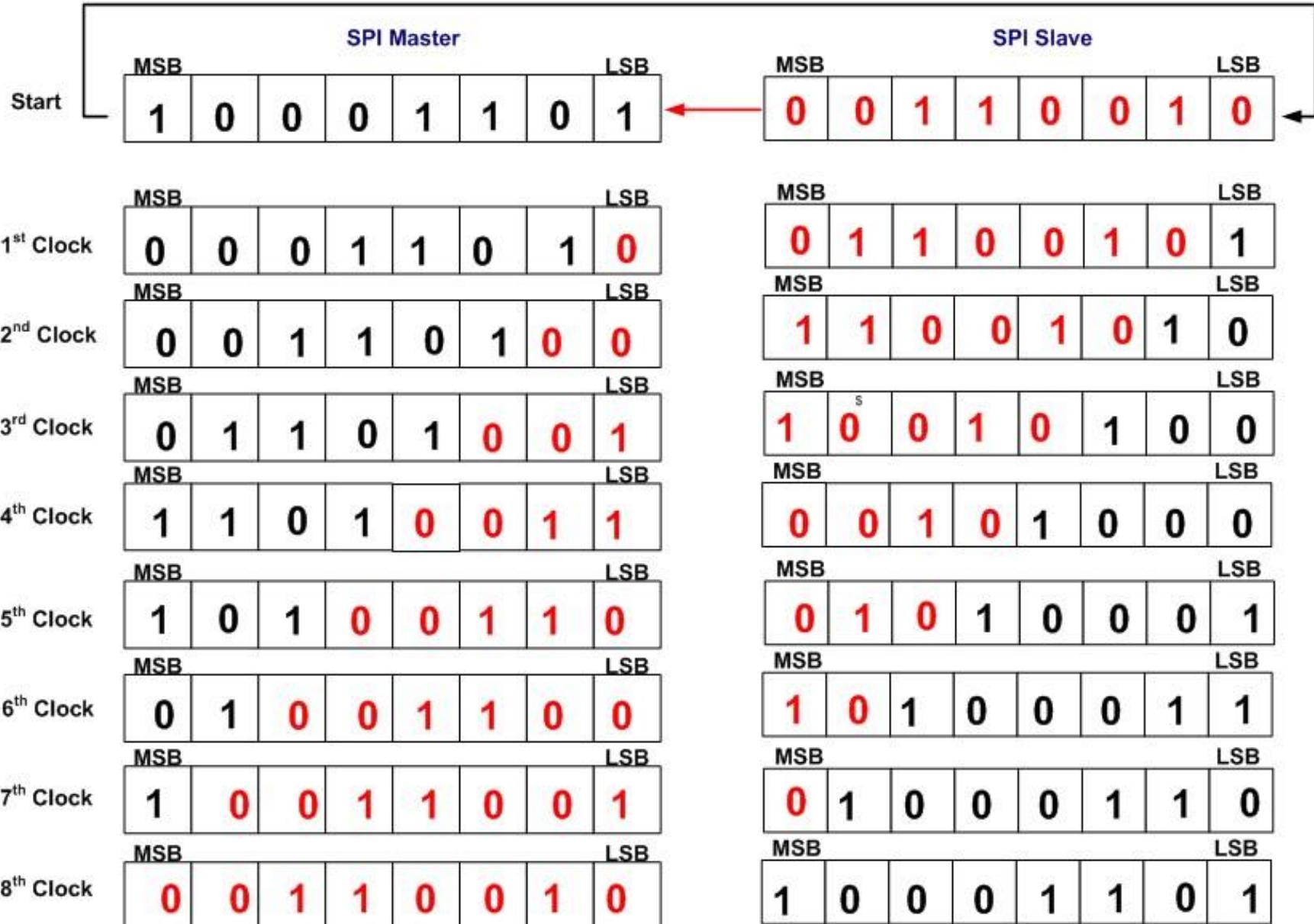


بروتوكول الاتصال التسلسلي SPI في متحكمات STM32



Ref: <http://www.emicro.com/blog/?p=1050>

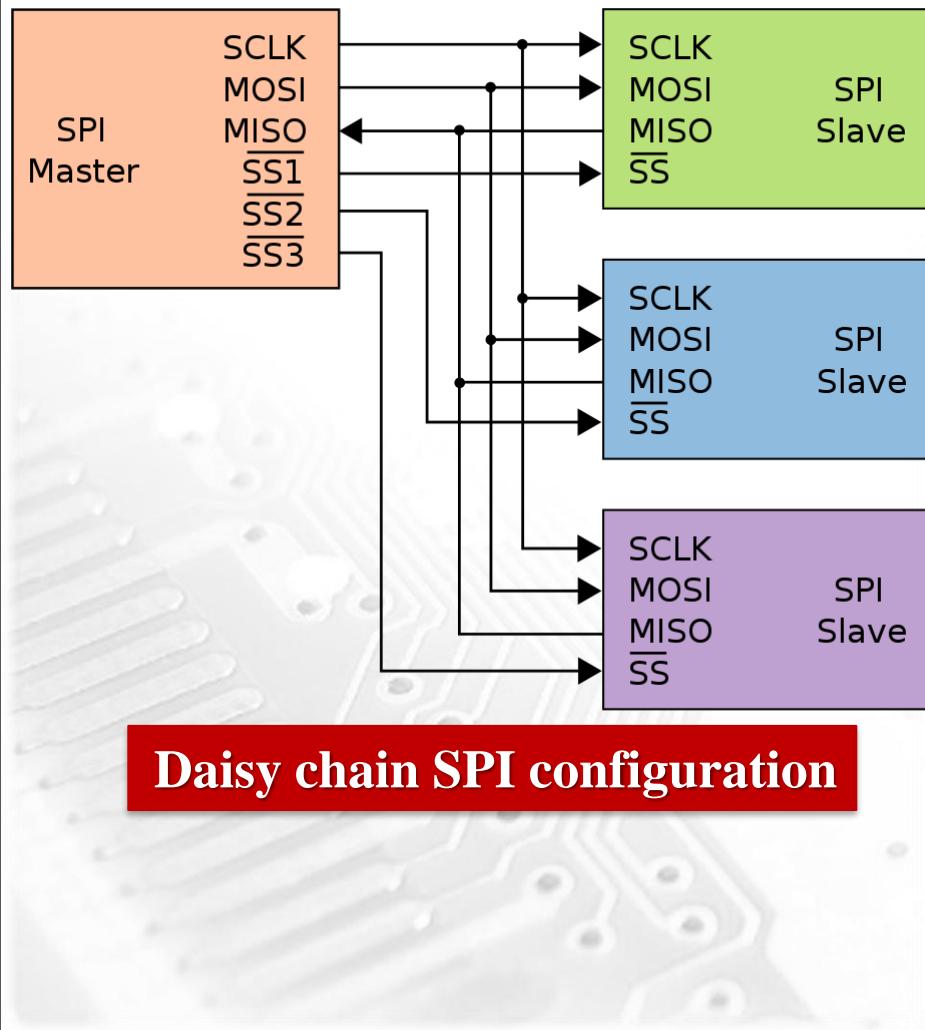
بروتوكول الاتصال التسلسلي SPI في متحكمات STM32



Ref: <http://www.emicro.com/blog/?p=1050>

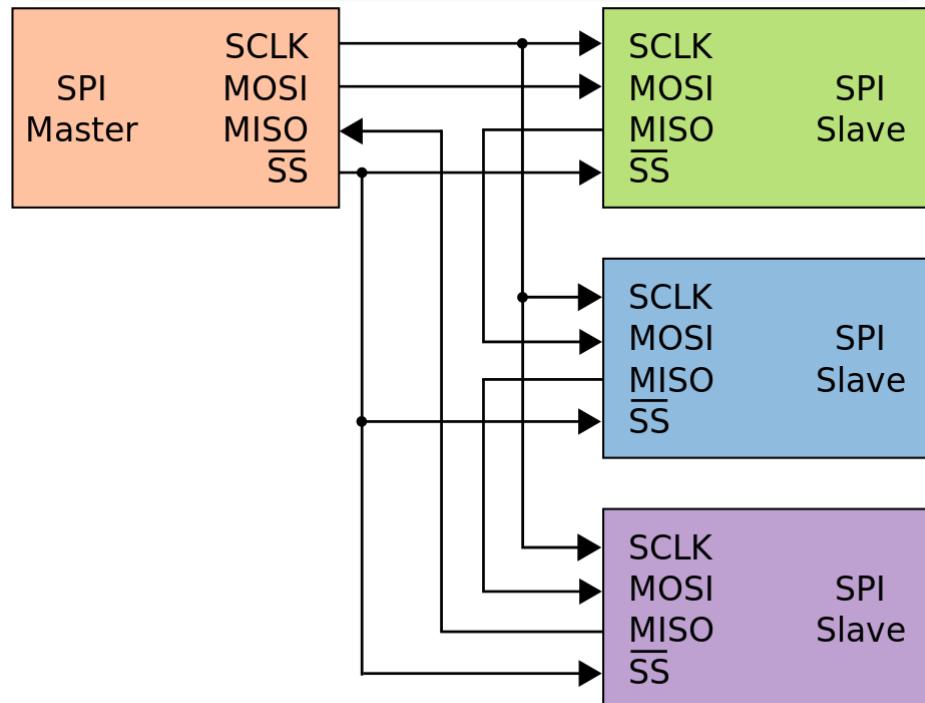
بروتوكول الاتصال التسلسلي SPI في متعدمات STM32

أنماط التوصيل لنقل الاتصال التسلسلي SPI



Daisy chain SPI configuration

Independent slave SPI configuration

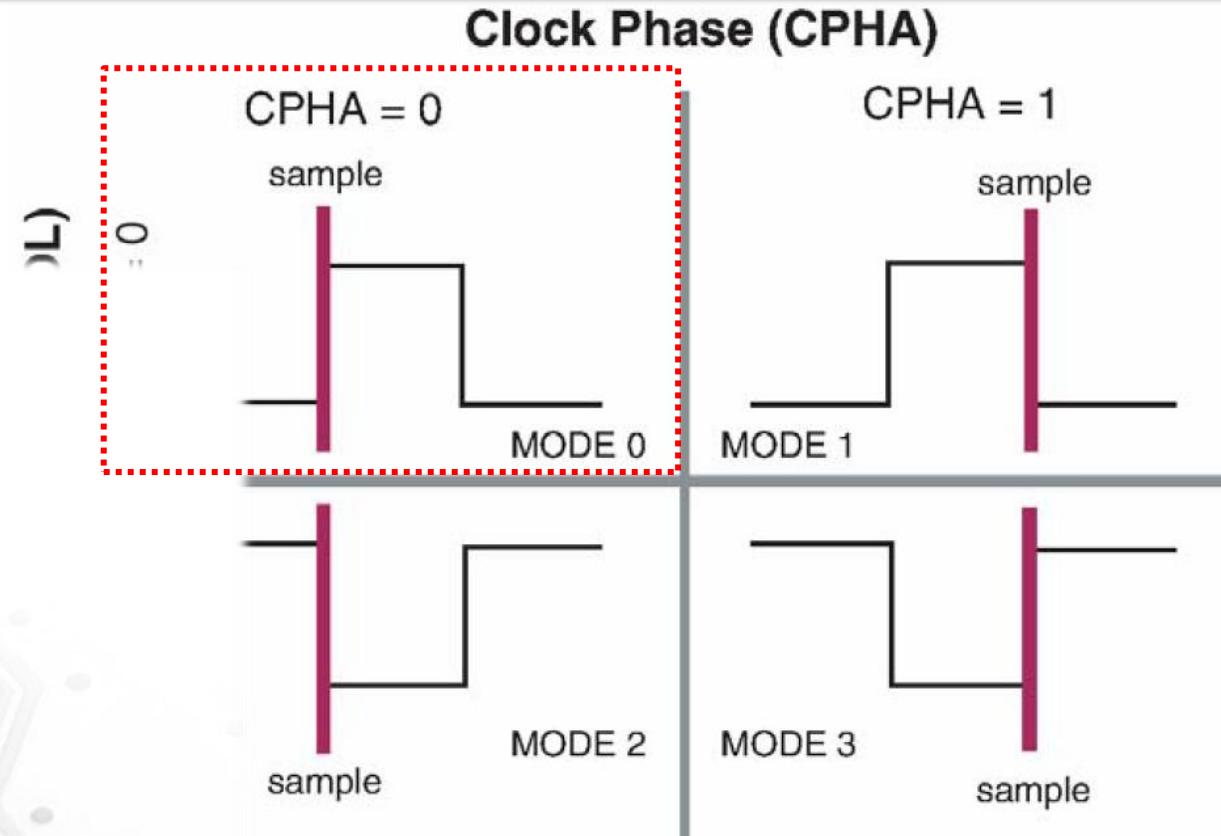


بروتوكول الاتصال التسلسلي SPI في متحكمات STM32

قطبية وطور إشارة التزامن في بروتوكول الاتصال التسلسلي SPI

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

data read on falling edge
data change on rising edge

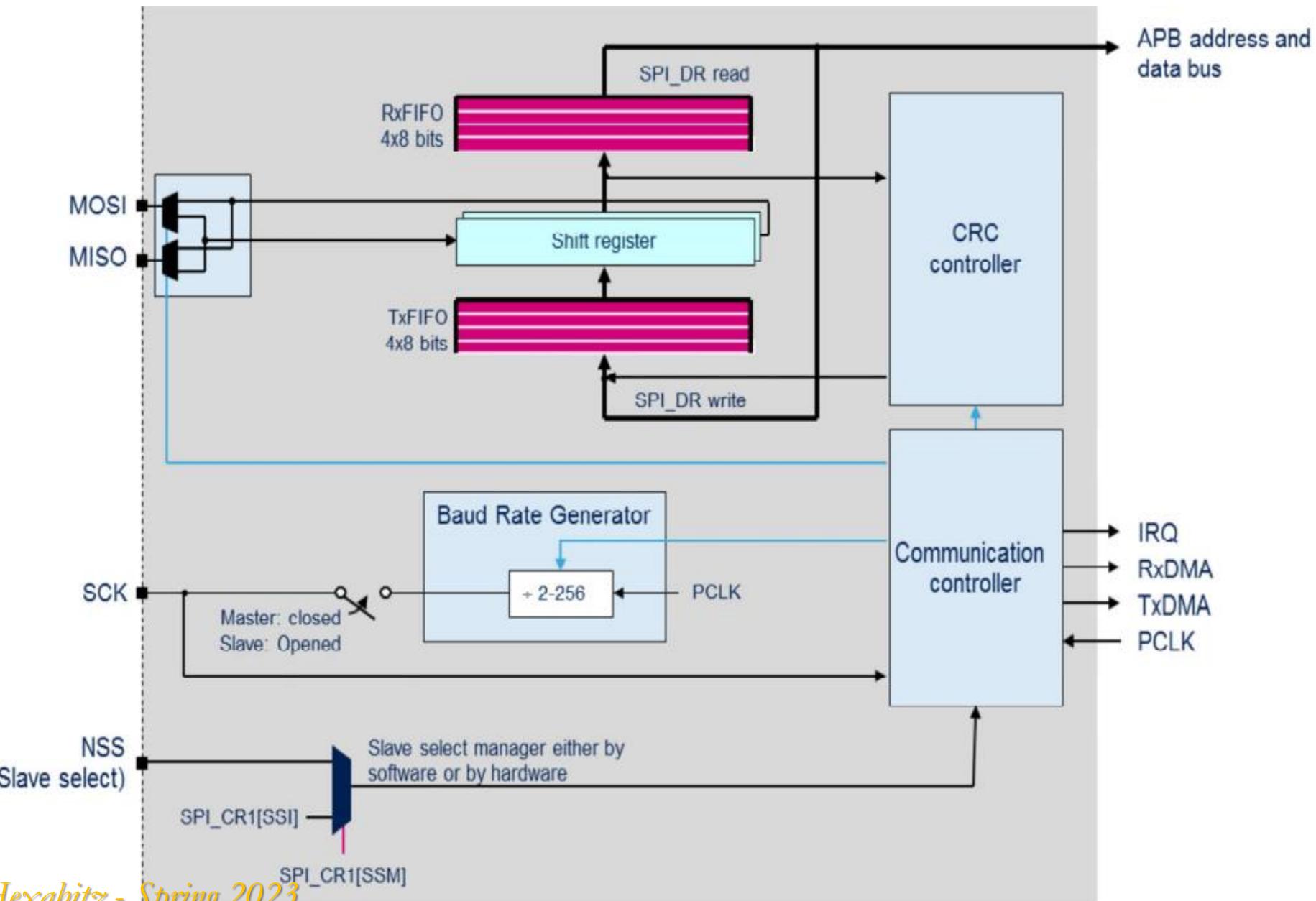


	Leading Edge	Trailing Edge	SPI Mode
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)	0
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)	1
CPOL = 1, CPHA = 0	Sample (Falling)	Setup (Rising)	2
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)	3

- Full-duplex synchronous transfers on three lines**
- Simplex synchronous transfers on two lines with or without a bidirectional data line**
- 8- or 16-bit transfer frame format selection**
- Master or slave operation**
- Multimaster mode capability**
- 8 master mode baud rate prescalers (fPCLK/2 max.)**
- Slave mode frequency (fPCLK/2 max)**

- NSS pin management by hardware or software for both master and slave
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag

المخطط الصندوقى لوحدة SPI في متحكمات STM32



- Master or slave (multi-master & multi slave support)
- Full-duplex, simplex or half-duplex
- Motorola and TI standards supported

وبحيث لا تتجاوز سرعة الاتصال $f_{pclk}/2$ ، وباستخدام سلكين على الأقل

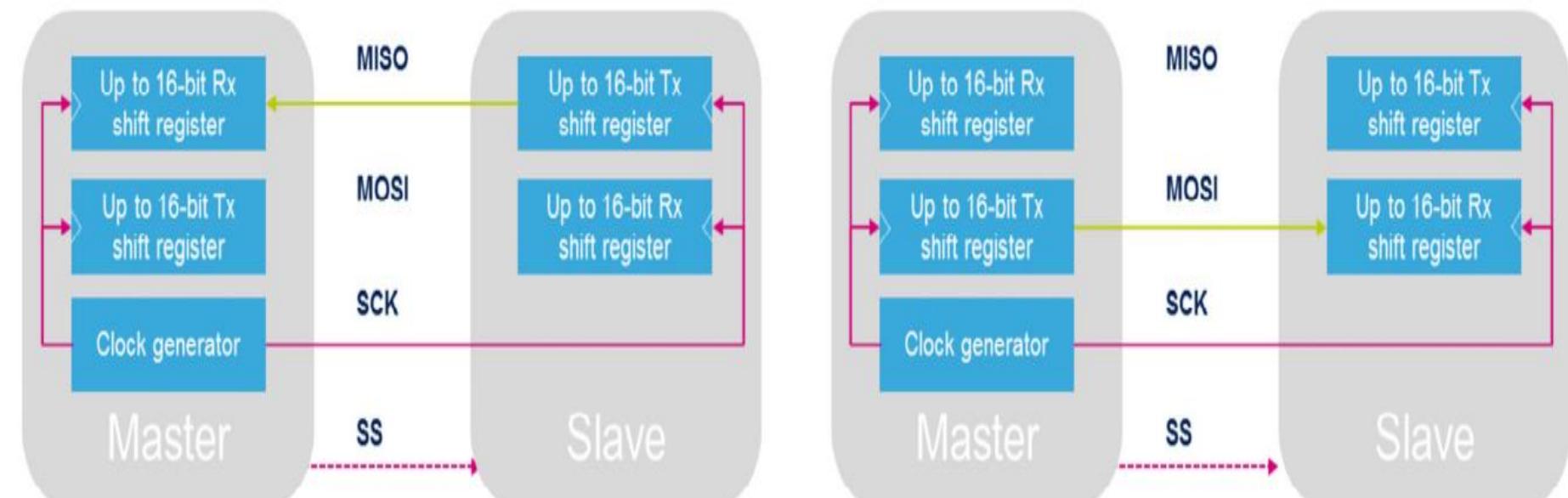
للاتصال التسلسلي وباتجاه واحد، كما يمكن أن تعمل وحدة الـ SPI بنمط :

- Polling
- Interrupt
- DMA

1- Simplex Mode

- في هذا النمط من العمل يكون أحد الطرفين مرسل فقط والآخر مستقبل فقط حيث يمكن للبيانات أن تتدفق باتجاه واحد وبحسب اتجاه البيانات يتم اختيار خط البيانات حيث تحتاج إلى خط بيانات وحيد
- يتم اختيار هذا النمط من العمل عند اختيار نمط transmit-only أو receive-only

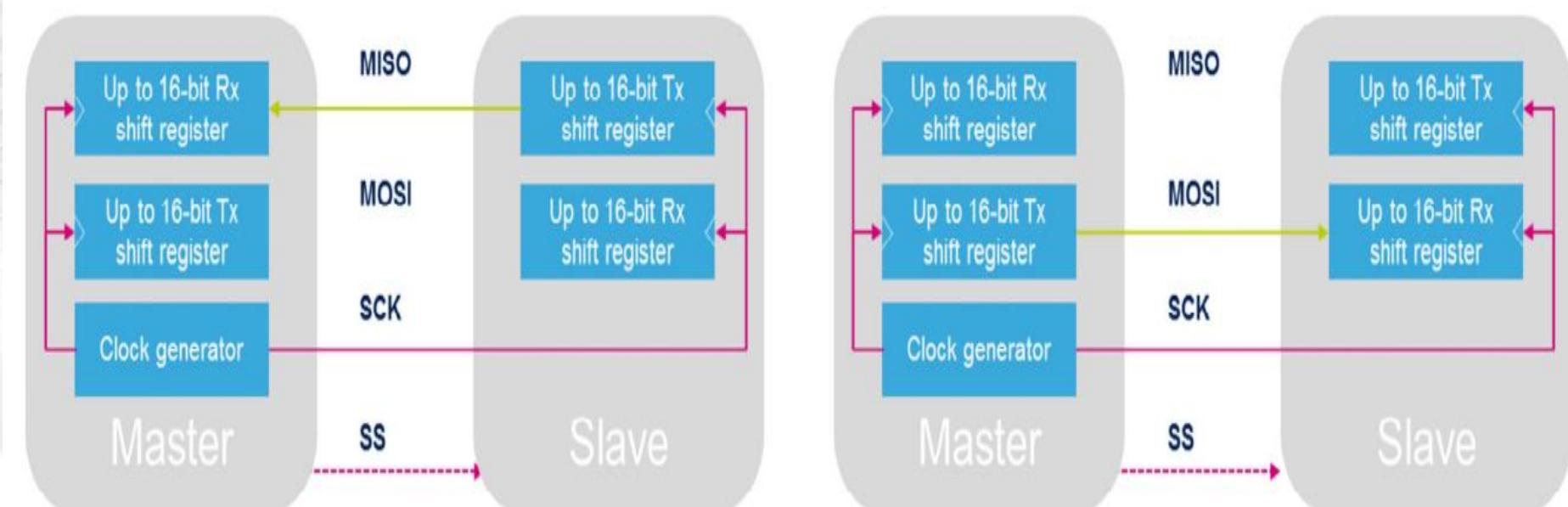
receive-only



1- Simplex Mode

فعلى سبيل المثال عندما يكون الـ master في نمط transmit-mode والـ slave في نمط receive-mode عندها يتم وصل قطب الـ slave من الـ master MOSI مع قطب NSS اختياري لوجود جهاز slave في هذا النمط يكون وصل قطب

واحد فقط



2- Half-duplex Mode

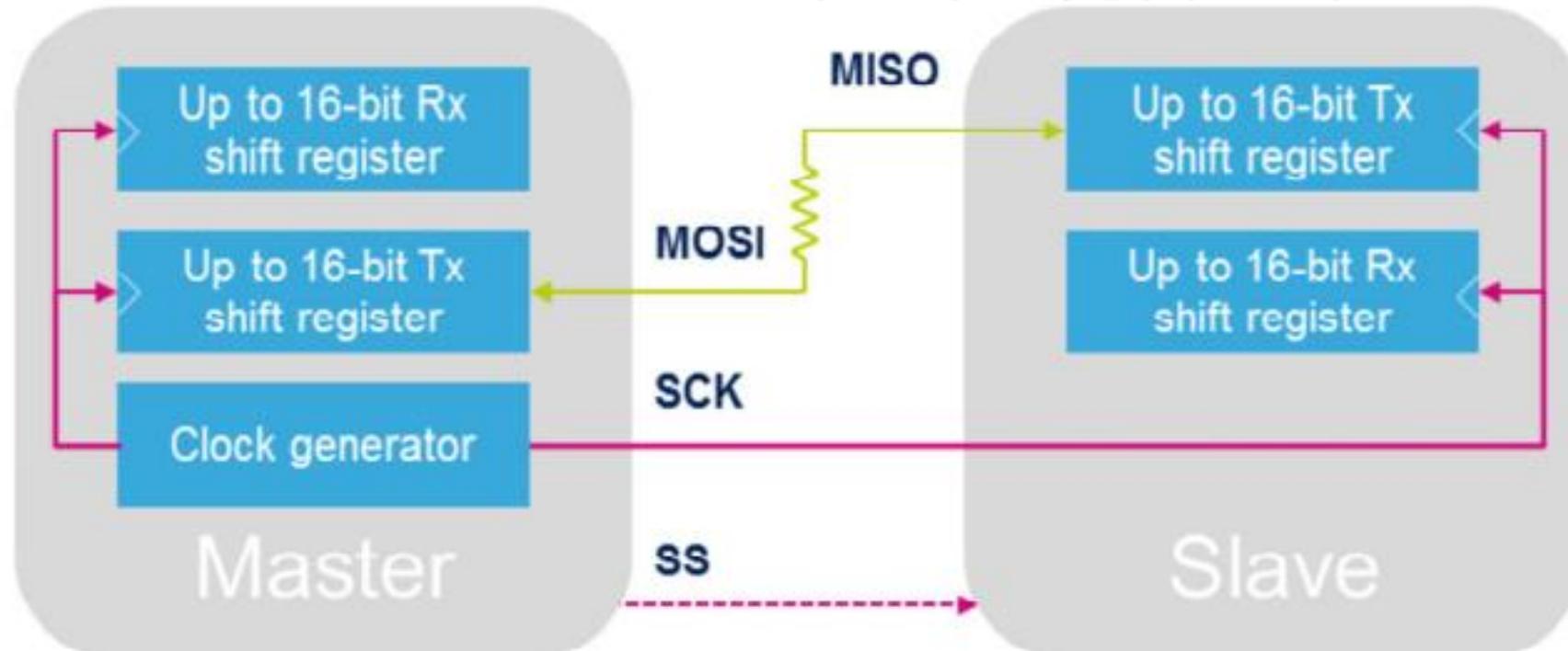
في هذا النمط من العمل يتم استخدام خط بيانات وحيد، حيث يتم وصل

قطب الـ MISO من الـ Master مع الـ Slave من

خلال مقاومة 1Kohm

وبإمكان كل من الـ Master والـ Slave إرسال واستقبال البيانات ولكن

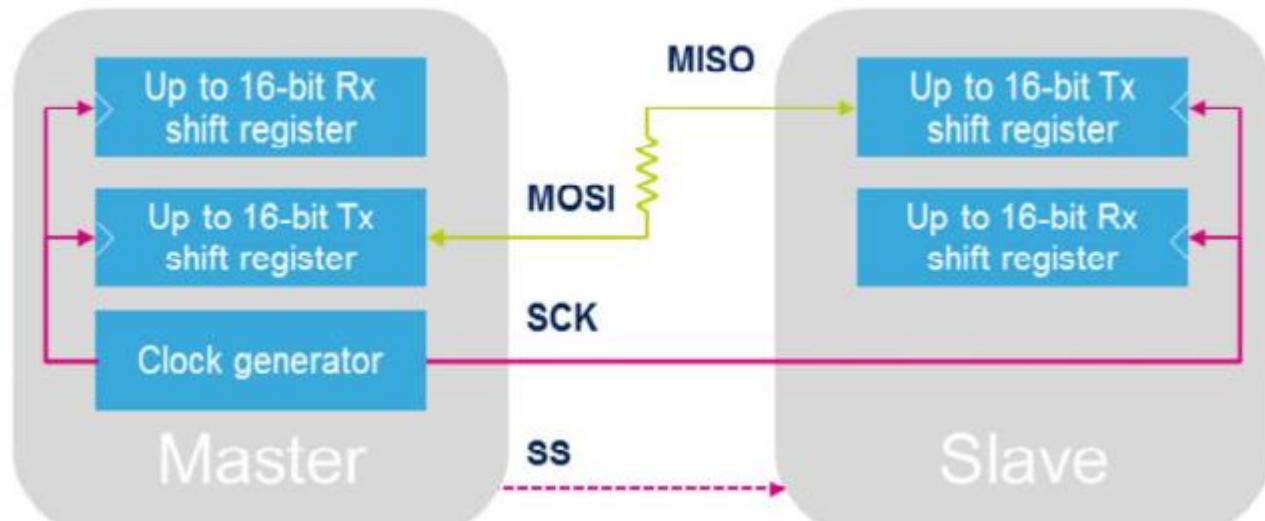
في اللحظة الواحدة يكون إما مرسلاً أو مستقبلاً



2- Half-duplex Mode

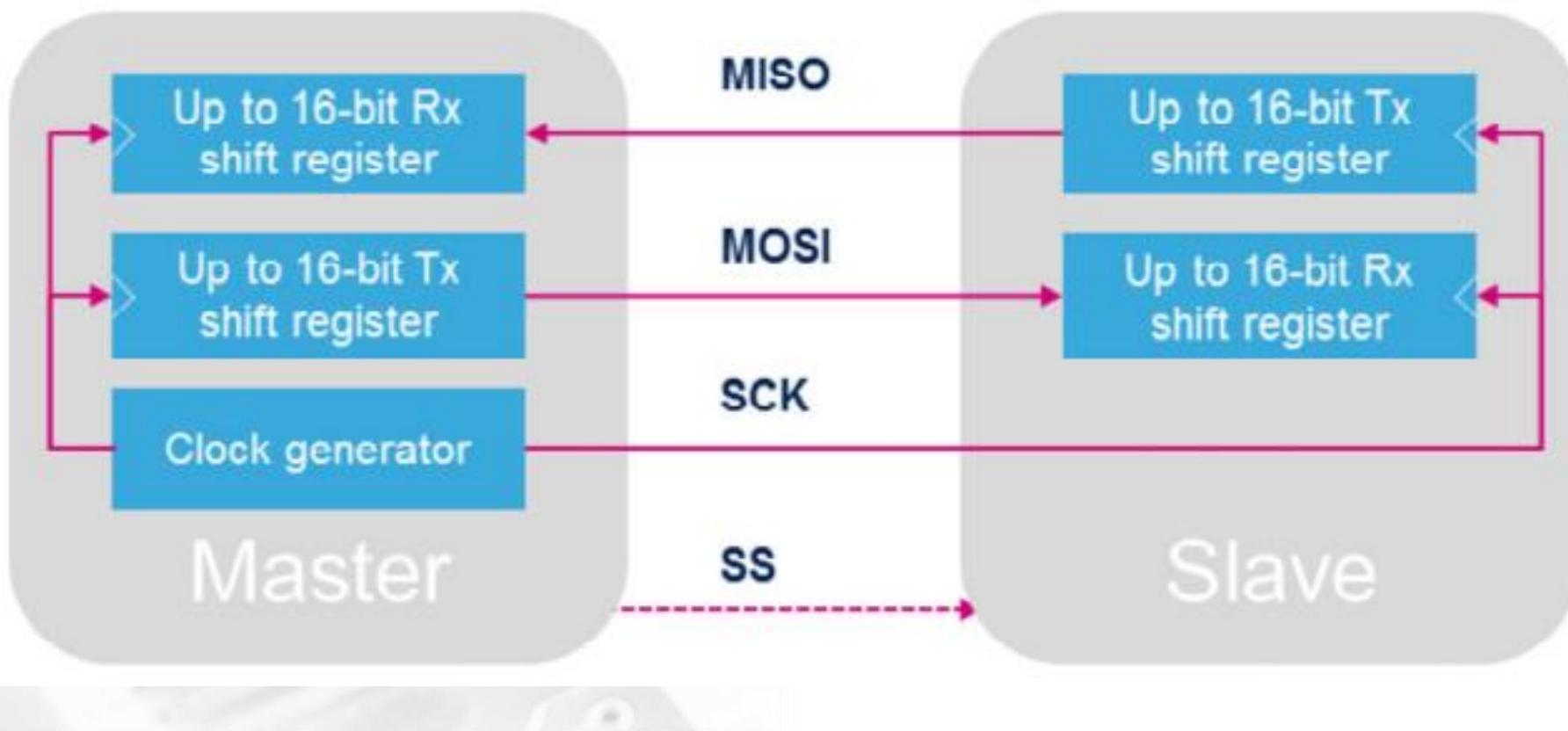
فعدما يقوم الـ slave بإرسال البيانات على خط MISO يقوم الـ Slave باستقبالها ويكون خط الـ MOSI في هذه الحالة للـ Master غير مفعل

أو يقوم الـ Master بإرسال البيانات على خط MOSI ويقوم الـ Slave باستقبالها ويكون خط الـ MISO في هذه الحالة للـ Master غير مفعل



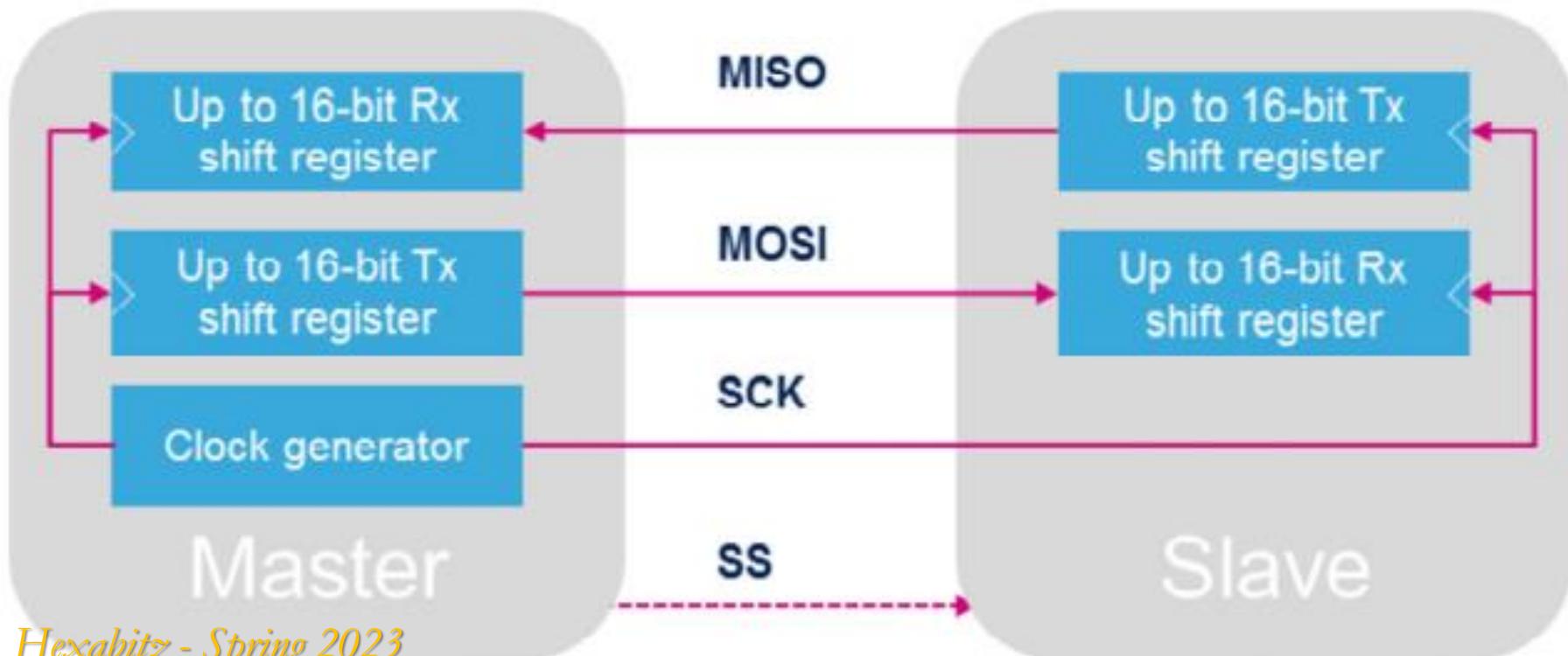
3- Full Duplex Mode

في هذا النمط من العمل يمكن لكلٍ من الـ Master والـ Slave أن يقوم بإرسال واستقبال البيانات في نفس الوقت

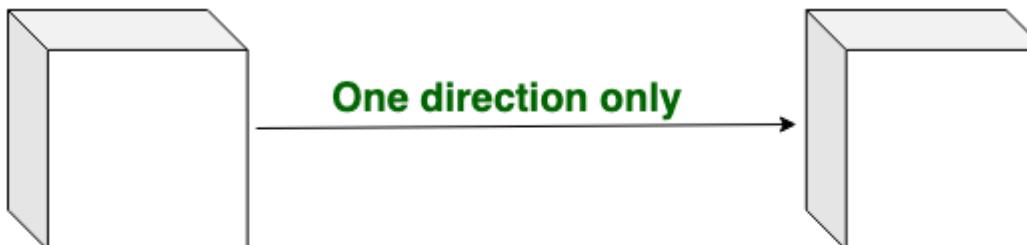


3- Full Duplex Mode

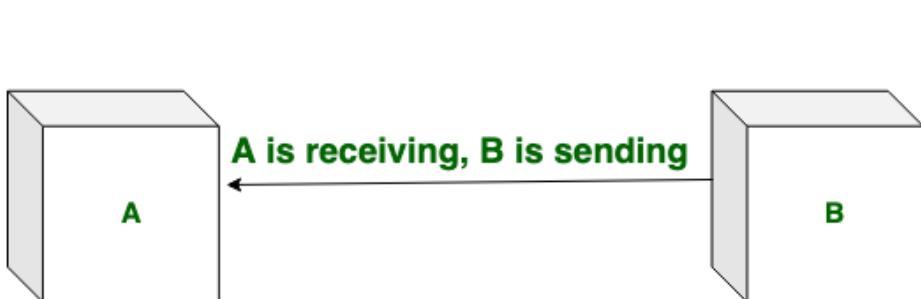
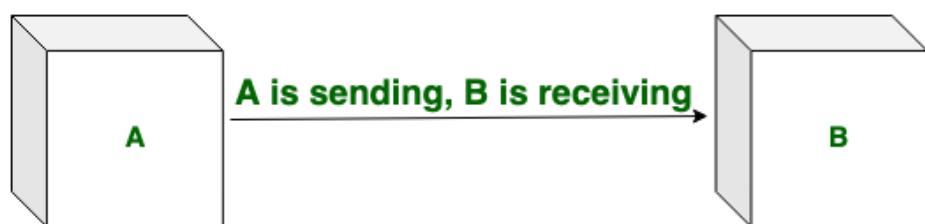
- حيث يتم ارسال البيانات من الـ Master عبر خط الـ MOSI ويقوم الـ Slave باستقبالها
- ويتم ارسال البيانات من الـ Slave عبر خط الـ MISO ويقوم الـ Master باستقبالها



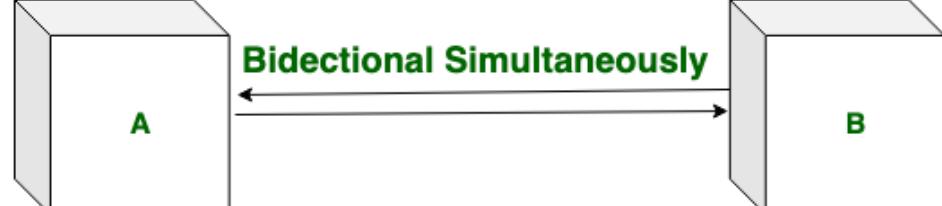
Simplex & Half-duplex & Full duplex Mode



Simplex mode



Half duplex mode



Full duplex mode

NSS pin

نميز حالتين:

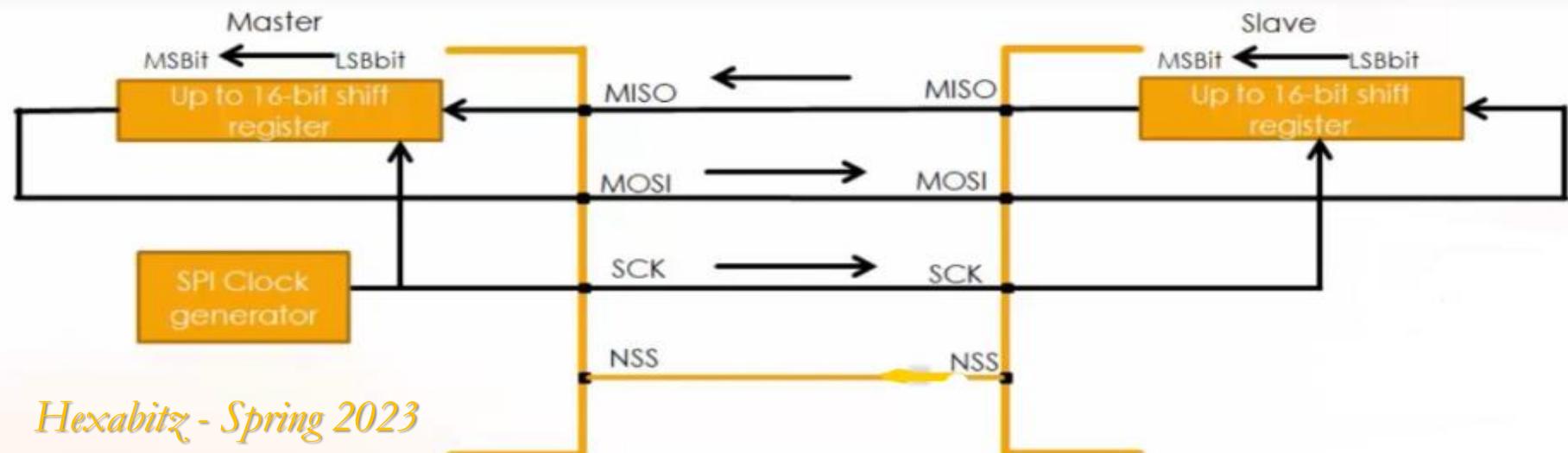
Slave mode : في هذه الحالة يكون قطب ال NSS قطب دخل و تكون

مهمته chip select أي قطب اختيار الشريحة

Master mode : في هذه الحالة وعندما يكون هناك جهاز

واحد ، يكون قطب ال NSS قطب خرج و تكون مهمته اختيار ال slave

المراد الاتصال به



نميز نمطي عمل لوحة SPI :

Motorola mode : في هذا النمط من العمل يكون بإمكانك التحكم

بقطبية وطور إشارة التزامن من خلال التحكم بقيم CPOL , CPOH

TI mode : في هذا النمط من العمل لا يمكنك التحكم بقطبية وطور

إشارة التزامن ، حيث يتم وضع الـ CPOL , CPOH بالمستوى

المنخفض LOW

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة SPI في وضع الـ Polling

Transmission

```
HAL_SPI_Transmit(SPI_HandleTypeDef * hspi, uint8_t *  
pData, uint16_t Size, uint32_t Timeout);
```

حيث: **pData**: مؤشر الى **data buffer** للبيانات المراد ارسالها، **Size**: حجم البيانات، **Timeout**: مدة المهلة قبل فشل الارسال

Reception

```
HAL_SPI_Receive(SPI_HandleTypeDef * hspi, uint8_t *  
pData, uint16_t Size, uint32_t Timeout);
```

Transmit- Receive

```
HAL_SPI_TransmitReceive(SPI_HandleTypeDef * hspi,  
uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t  
Timeout);
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة SPI في وضع الـ Interrupt

Transmission 

```
HAL_SPI_Transmit_IT(SPI_HandleTypeDef * hspi, uint8_t *  
pData, uint16_t Size);
```

- بعد استدعاء هذه الدالة تقوم وحدة SPI ببدء عملية الإرسال لكامل البایتات الموجودة ضمن الـ buffer وعند الانتهاء يتم استدعاء التابع التالي:

```
void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef * hspi)  
{  
    // TX Done .. Do Something ...  
}
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة SPI في وضع الـ Interrupt

Reception

```
HAL_SPI_Receive_IT(SPI_HandleTypeDef * hspi, uint8_t *  
pData, uint16_t Size);
```

- بعد استدعاء هذه الدالة تقوم وحدة SPI ببدء عملية الاستقبال لكامل البأيتات الموجودة ضمن الـ buffer وعند الانتهاء يتم استدعاء التابع التالي:

```
void HAL_SPI_RxCpltCallback(SPI_HandleTypeDef * hspi)  
{  
    // RX Done .. Do Something ...  
}
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة SPI في وضع الـ Interrupt

Transmit- Receive

```
HAL_SPI_TransmitReceive_IT(SPI_HandleTypeDef * hspi,  
uint8_t * pTxData, uint8_t * pRxData, uint16_t Size);
```

- بعد استدعاء هذه الدالة تقوم وحدة SPI ببدء عملية الاستقبال لكامل البايتات الموجودة ضمن الـ buffer وعند الانتهاء يتم استدعاء التابع التالي:

```
void HAL_SPI_TxRxCpltCallback(SPI_HandleTypeDef * hspi)  
{  
    // TX-RX Done .. Do Something ...  
}
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة SPI في وضع DMA

Transmission 

```
HAL_SPI_Transmit_DMA(SPI_HandleTypeDef * hspi, uint8_t  
* pData, uint16_t Size);
```

- بعد استدعاء هذه الدالة تقوم وحدة SPI ببدء عملية الإرسال لـ كامـل الـ باـيـات المـوجـودـة ضـمـن الـ buffer وعـنـد الـ اـنـتـهـاء يـتم اـسـتـدـعـاء التـابـع التـالـي:

```
void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef * hspi)  
{  
    // TX Done .. Do Something ...  
}
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة SPI في وضع الـ DMA

Reception

```
HAL_SPI_Receive_DMA(SPI_HandleTypeDef * hspi, uint8_t *  
pData, uint16_t Size);
```

- بعد استدعاء هذه الدالة تقوم وحدة SPI ببدء عملية الاستقبال لكامل البايتات الموجودة ضمن الـ buffer وعند الانتهاء يتم استدعاء التابع التالي:

```
void HAL_SPI_RxCpltCallback(SPI_HandleTypeDef * hspi)  
{  
    // RX Done .. Do Something ...  
}
```

دوال مكتبة HAL المستخدمة ل التعامل مع وحدة SPI في وضع DMA

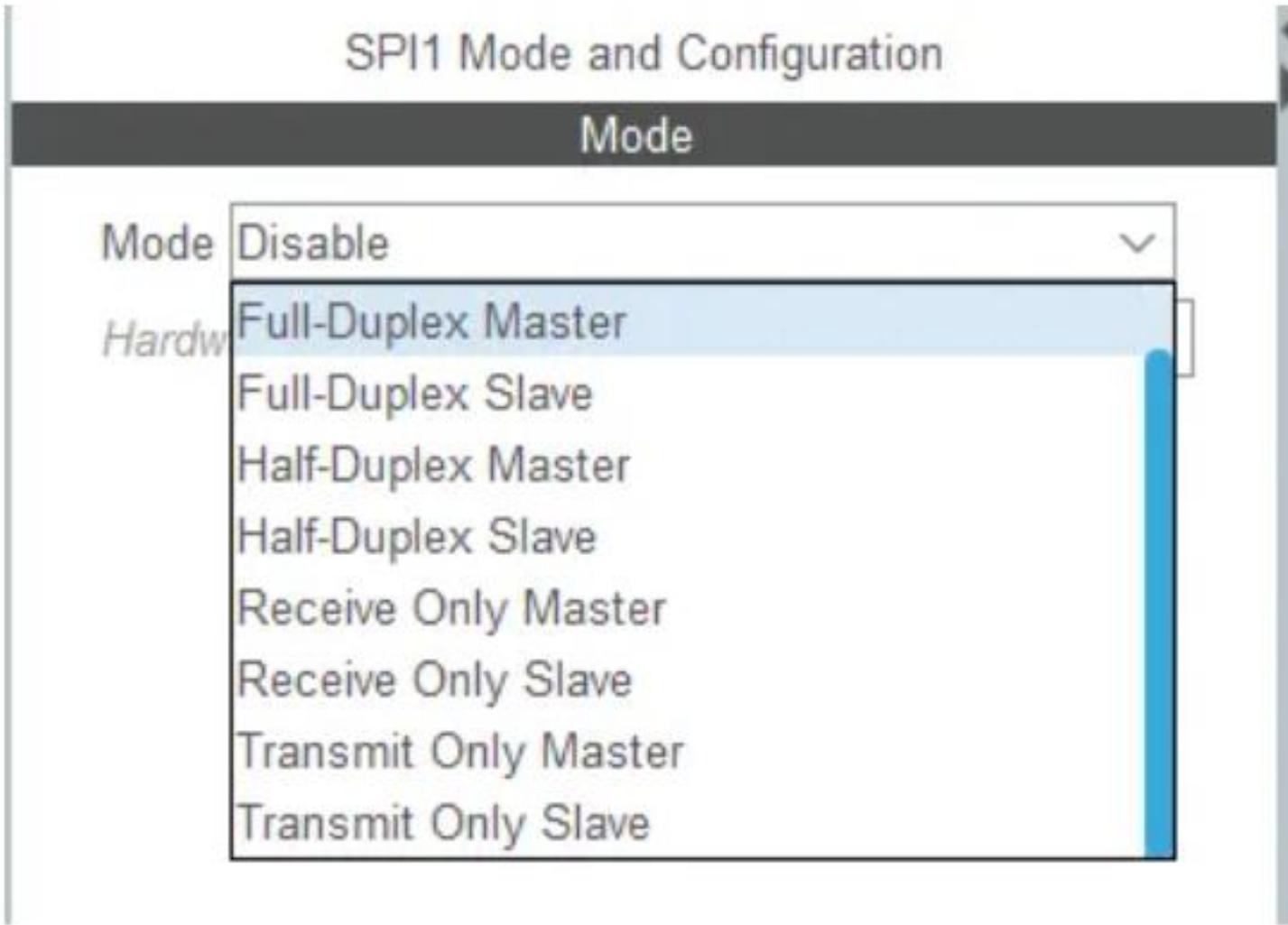
Reception

```
HAL_SPI_TransmitReceive_DMA(SPI_HandleTypeDef * hspi,  
uint8_t * pTxData, uint8_t * pRxData, uint16_t Size);
```

- بعد استدعاء هذه الدالة تقوم وحدة SPI ببدء عملية الاستقبال لـكامل البايتات الموجودة ضمن الـ buffer وعند الانتهاء يتم استدعاء التابع التالي:

```
void HAL_SPI_TxRxCpltCallback(SPI_HandleTypeDef * hspi)  
{  
    // TX-RX Done .. Do Something ...  
}
```

- تحديد نمط عمل وحدة SPI :





ضبط بaramترات وحدة SPI وتتضمن:

ضبط إعدادات الوحدة لتعمل بنمط Motorola وضبط كل من حجم البيانات وتحديد بت البداية بالإضافة إلى تحديد قطبية وطور إشارة التزامن من خلال التحكم بقيم CPOL , CPOH ، بالإضافة إلى ضبط قيمة البت :NSS

Configure the below parameters :

Search (Ctrl+F) (i)

Basic Parameters

- Frame Format
- Data Size
- * First Bit

Motorola
8 Bits
MSB First

Clock Parameters

- Prescaler (for Baud Rate)
- Baud Rate
- * Clock Polarity (CPOL)
- * Clock Phase (CPHA)

2
8.0 MBits/s
Low
1 Edge

Advanced Parameters

- CRC Calculation
- * NSSP Mode
- NSS Signal Type

Disabled
Enabled
Output Hardware

416

ضبط إعدادات وحدة SPI في متكمات STM32



ضبط بارامترات وحدة SPI وتتضمن:

كما يمكن تفعيل المقاطعة الخاصة بوحدة الـ SPI من خلال تاب

NVIC Settings

The screenshot shows the NVIC Settings tab selected in the top navigation bar. Below it, the NVIC Interrupt Table is displayed for the SPI1/I2S1 Interrupt. The 'Enabled' checkbox is checked and highlighted with a red box.

- كما يمكن إضافة طلب DMA لوحدة الـ SPI من خلال تاب **Settings**

The screenshot shows the DMA Settings tab selected in the top navigation bar. Below it, the DMA Request dropdown menu is open, showing options like SPI1_RX and SPI1_TX. The DMA Request dropdown is highlighted with a red box.

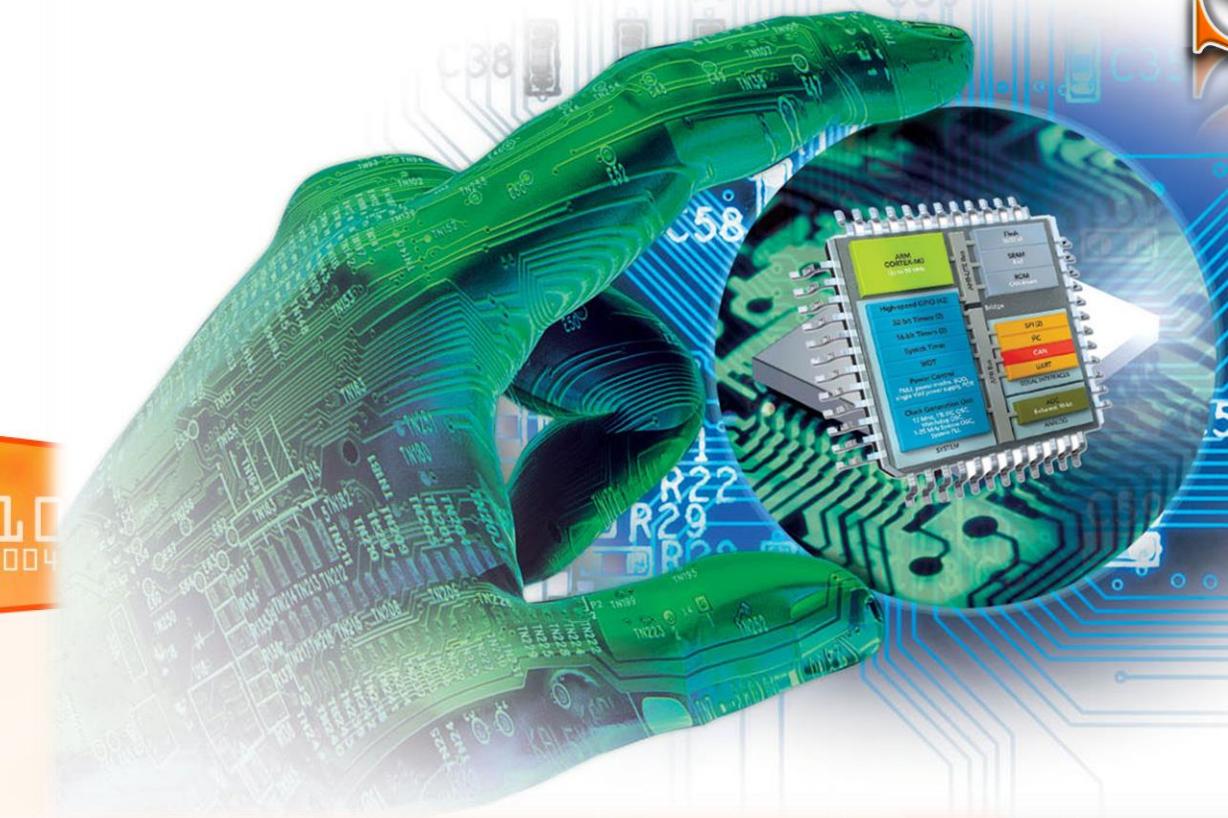
**التطبيق العملي : المطلوب ربط مجموعة متكاملات
من خلال النافذة التسلسليّة SPI أحدّها
.Slave والباقي Master**

Thank you for listening

مُتَحَكِّمَات

STM32

10



Timer Modes in STM32

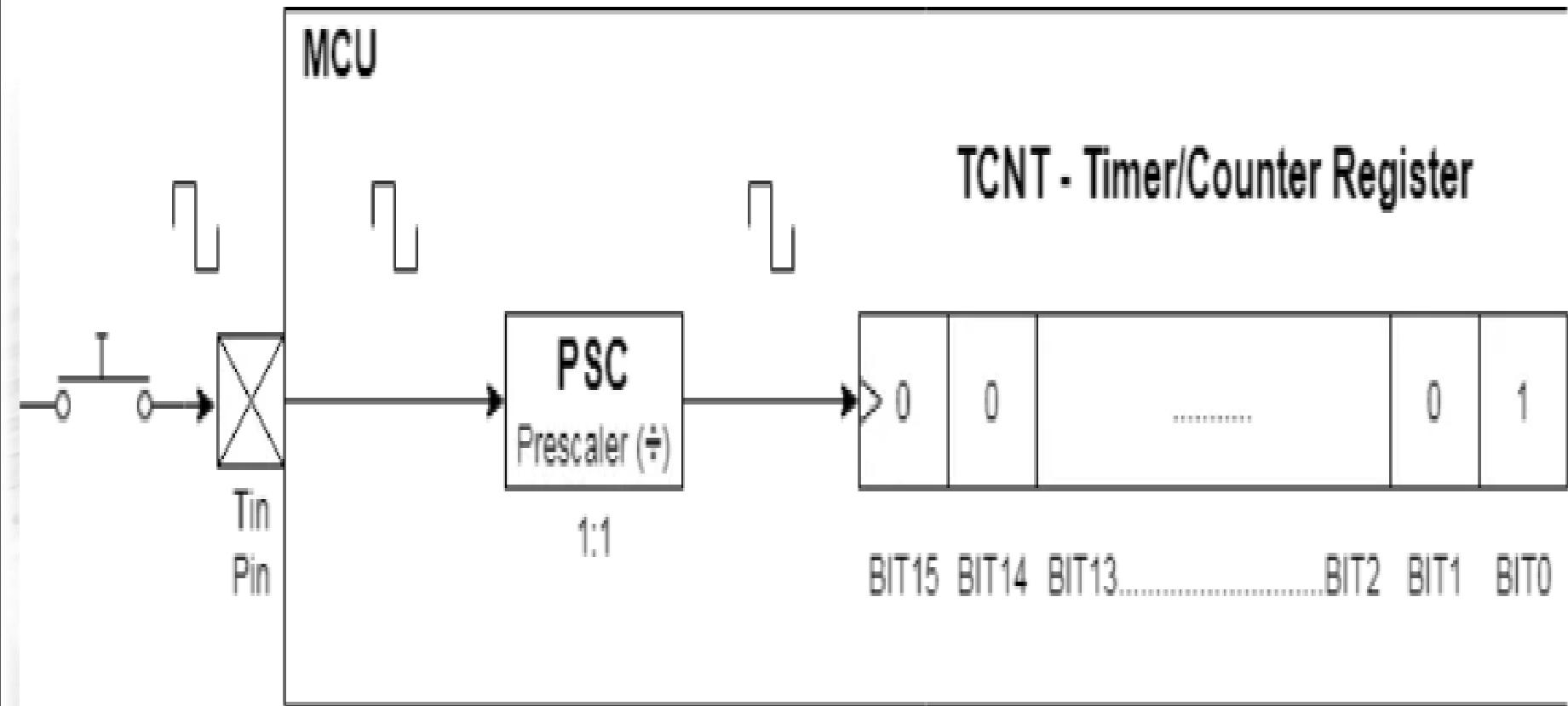
أنماط عمل المؤقتات في متحكمات STM32 :

- Timer Mode**
- PWM mode**
- Counter Mode**
- Input Capture Mode**
- Encoder interface Mode**
- Output Compare**
- One Pulse Mode**

Counter Mode

- في هذا النمط من العمل فإن نبضات الـ Timer تأتي من مصدر خارجي (قطب دخل المؤقت (timer input pin))
- يقوم بالعد التصاعدي أو التنازلي مع كل جبهة صاعدة/هابطة لإشارة الدخول
- هذا النمط من العمل مفيد عندما تحتاج إلى وجود عداد رقمي دون الحاجة للقراءة الدورية لحالة أقطاب الدخول GPIO للمتحكم أو حتى حدوث مقاطعة في كل مرة في حال استخدام أحد أقطاب المقاطعة الخارجية EXTI

Counter Mode



Counter Mode

نميز ثلاثة أنماط مختلفة للمؤقت عندما يعمل ك Counter وهي:

□ نمط العد التصاعدي: في هذا النمط فإن العداد يبدأ بالعد من الصفر مع

كل نبضة قادمة على قطب الدخول ويستمر حتى يصل إلى القيمة المخزنة مسبقاً الموجودة في المسجل (TIMx_ARR)، ثم يعود

للقيمة صفر ويولد مقاطعة الطفحان Overflow

□ نمط العد التنازلي : في هذا النمط فإن العداد يبدأ بالعد من القيمة

المخزنة في المسجل TIMx-ARR مع كل auto-reload value

نبضة قادمة على قطب الدخول ويستمر ليصل إلى الصفر ثم يعود ليبدأ

من القيمة المخزنة سابقاً ويولد حدث Underflow event

Counter Mode

نميز ثلاث أنماط مختلفة للمؤقت عندما يعمل ك Counter وهي:

- نمط العد التصاعدي تنازلي : في هذا النمط فإن العداد يبدأ بالعد التصاعدي من الصفر ويستمر بالعد مع كل نبضة قادمة على قطب الدخл حتى يصل إلى القيمة المخزنة سابقاً auto-reload value ثم يتم توليد حدث الطفحان TIMx-ARR ناقص واحد ، ثم يتم توليد حدث Overflow event ثم يبدأ بالعد التنازلي من القيمة المخزنة سابقاً
- تصاعدية قادمة على قطب الدخл وحتى يصل إلى الصفر عنها يتم توليد حدث Underflow ثم يعود للعد

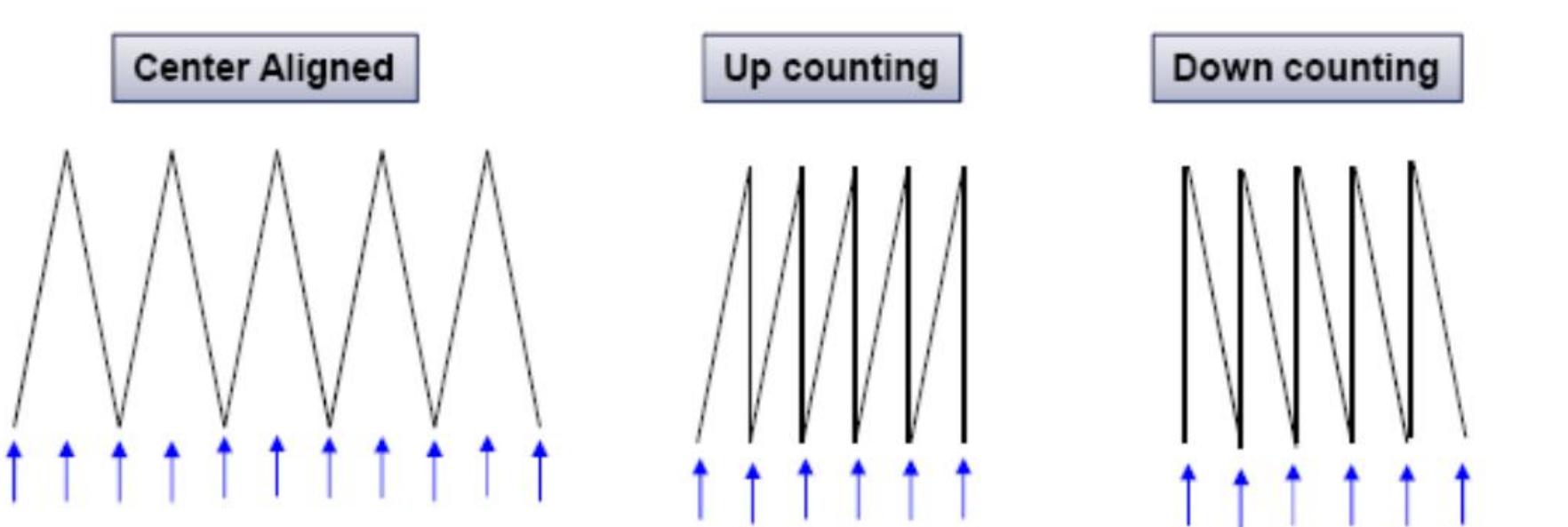
التصاعدي من الصفر وهذا ...

Timer Modes in STM32

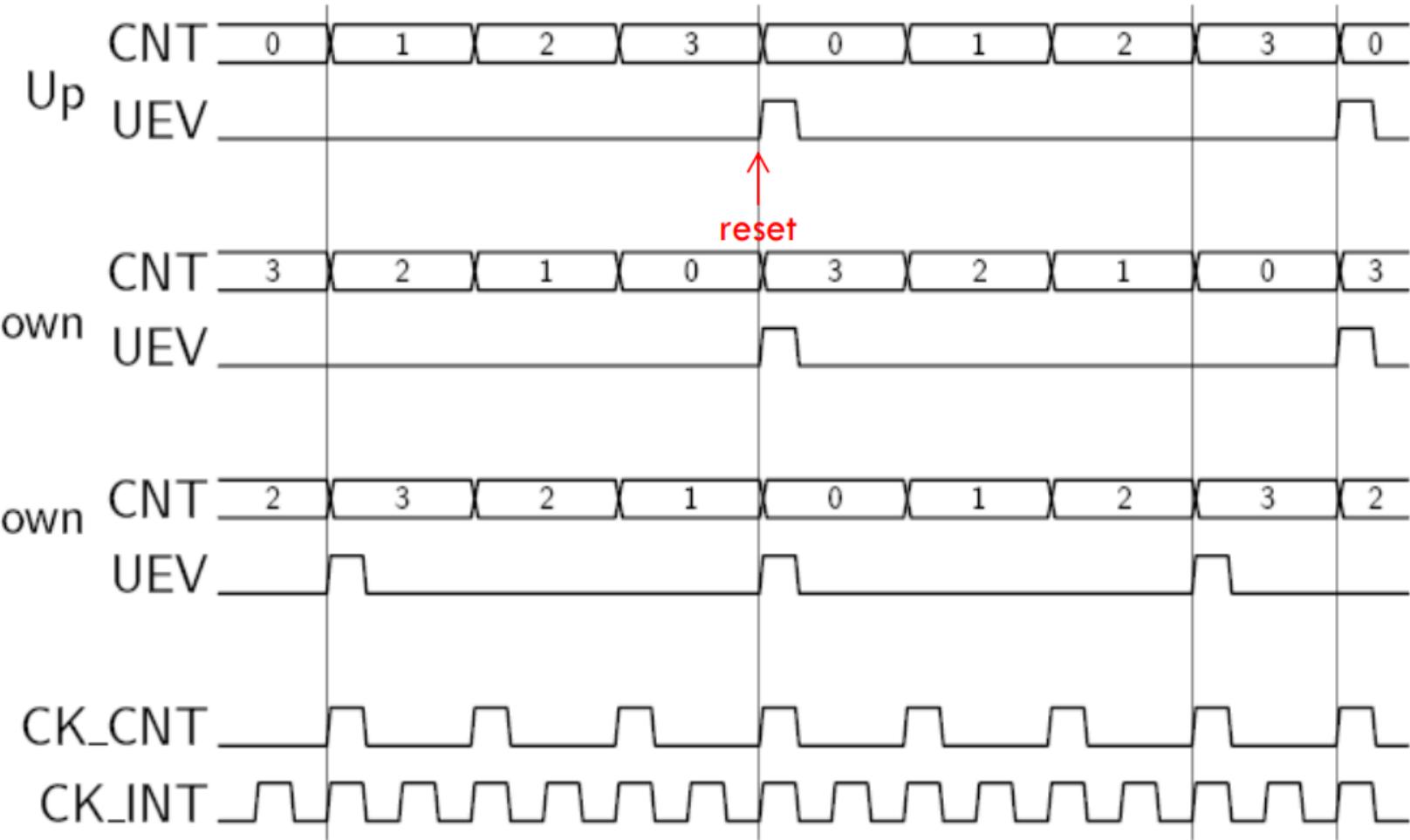
Counter Mode

نميز ثلاث أنماط مختلفة للمؤقت عندما يعمل ك Counter وهي:

- نمط العد التصاعدي تنازلي : في هذا النمط فإن بت الاتجاه DIR في المسجل TIMx_CR1 غير قابل للكتابة أو تغيير قيمته فقط يتم تحديث قيمته من خلال الـ Hardware ويعبر عن الاتجاه الحالي للعد.



Counter Mode



Counter Modes ($\text{ARR}=3$, $\text{PSC}=1$)

Counter Mode

لضبط المؤقت في نمط الـ Counter Mode من خلال CubeMX نختار ما

: يلي :

ضبط مصدر الساعة للمؤقت كي يعمل كعداد رقمي (من خلال قطب الدخول



للمؤقت) على المصدر الخارجي external pin ETR2 وهو PA0 والذي

يوافق القطب PA0

ضبط القيمة المخزنة مسبقاً للعداد counter period على القيمة 20 ،



عندما يصل العداد إلى القيمة 20 ستحدث مقاطعة الطفhan

Counter Mode

لضبط المؤقت في نمط الـ Counter Mode من خلال CubeMX نختار ما

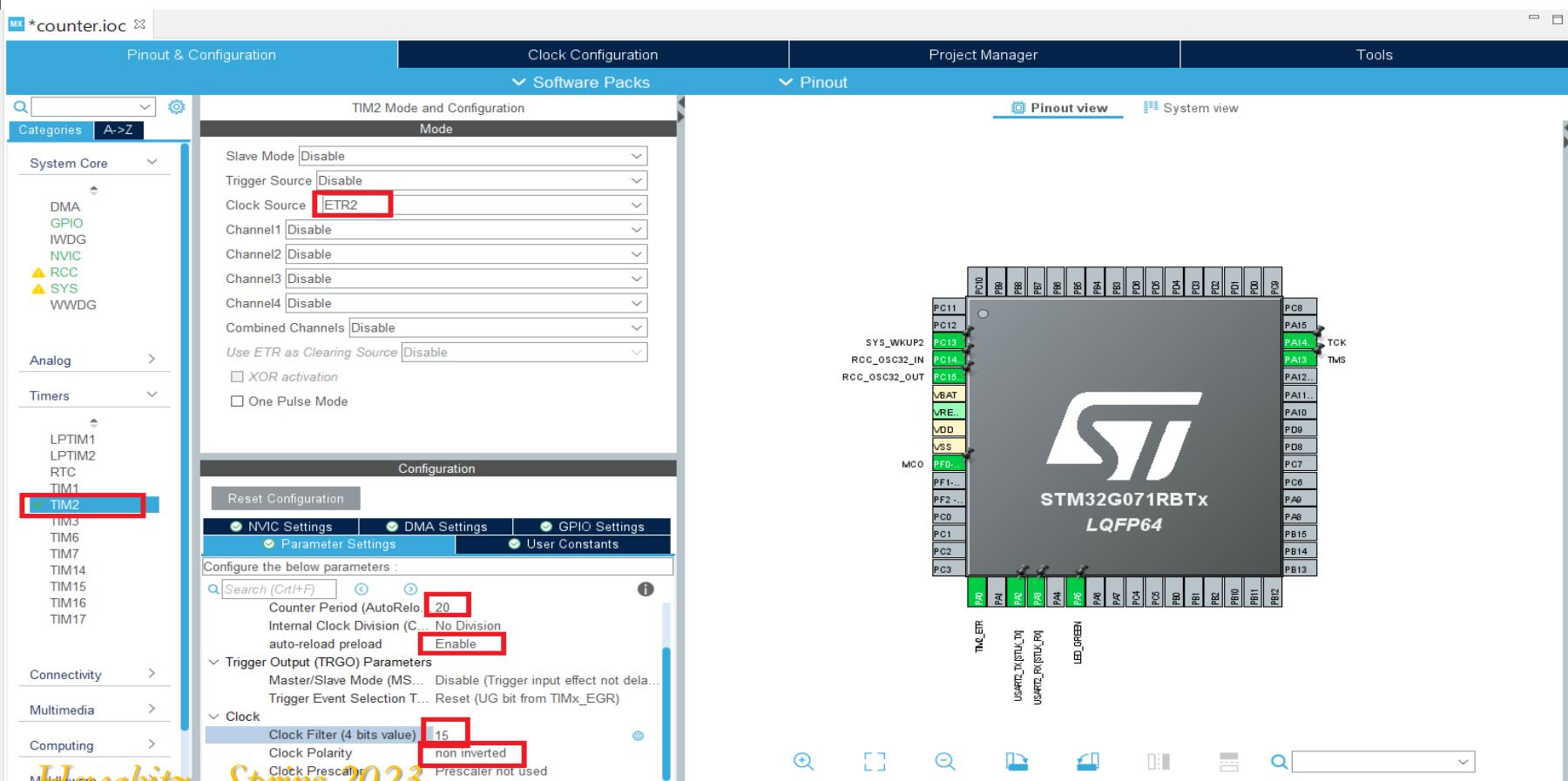
يلي :

ضبط فلتر رقمي لقطب المؤقت لتجنب الضجيج الناتج عن اهتزاز المفتاح، حيث يتراوح مجال القيم للفلتر بين 0 والـ 15، Switch bouncing أخيراً يمكن ضبط الجبهة التي سيعد عندها العداد صاعدة أو هابطة، سنختار الجبهة الصاعدة .

Timer Modes in STM32

Counter Mode

لضبط المؤقت في نمط الـ Counter Mode من خلال نختار ما يلي :



Input Capture Mode

المؤقت في نمط Input Capture يستقبل نبضات الساعة

الخاصة به من مصدر داخلي (ساعة المتحكم بعد استخدام المقسم

الترددية) ويستمر بالعد إلى أن يحدث حدث معين (جبهة صاعدة/

جبهة هابطة) على قطب المتحكم الخاص بقناة الـ Input

Capture Channel عندها يتم حفظ القيمة التي وصل إليها

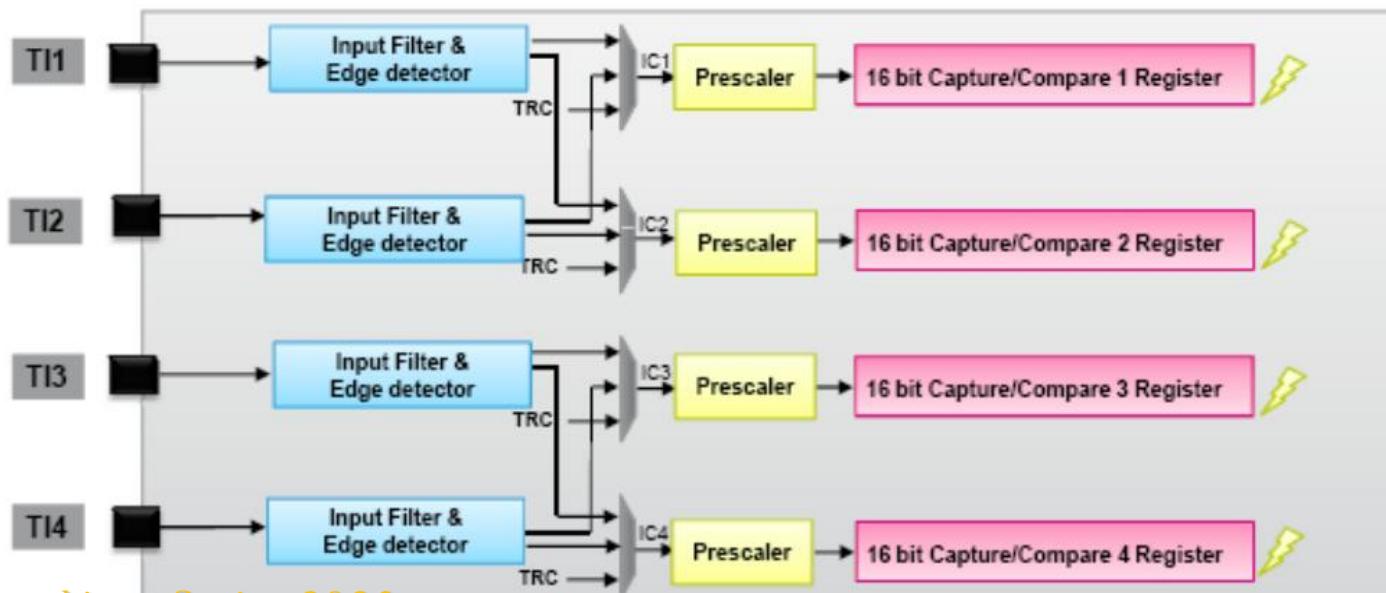
المؤقت إلى مسجل input capture register

لكل مؤقت في متحكمات STM32 عدة قنوات (input)

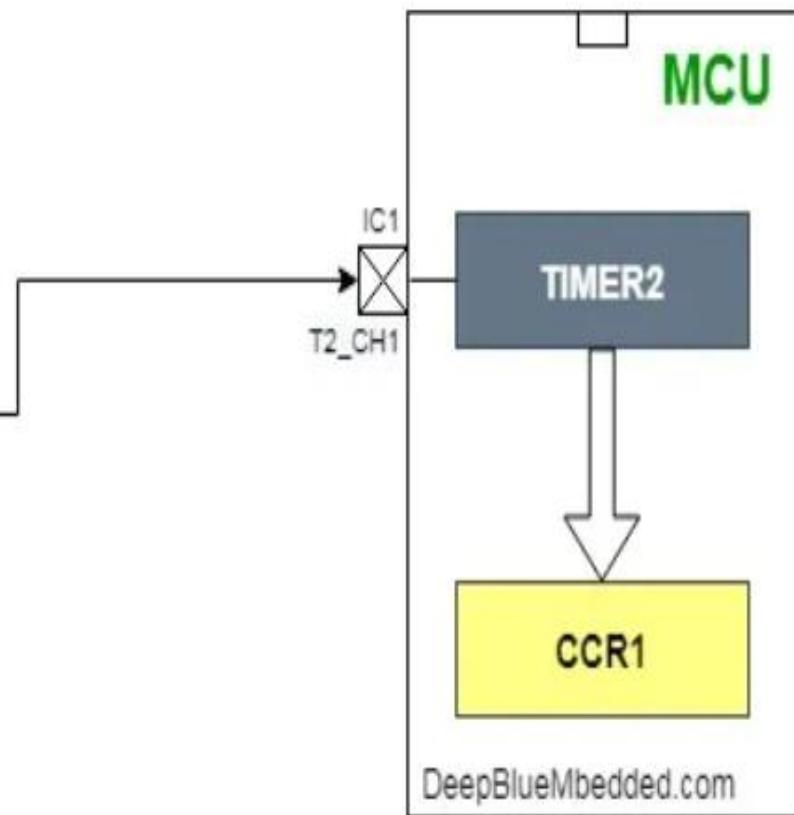
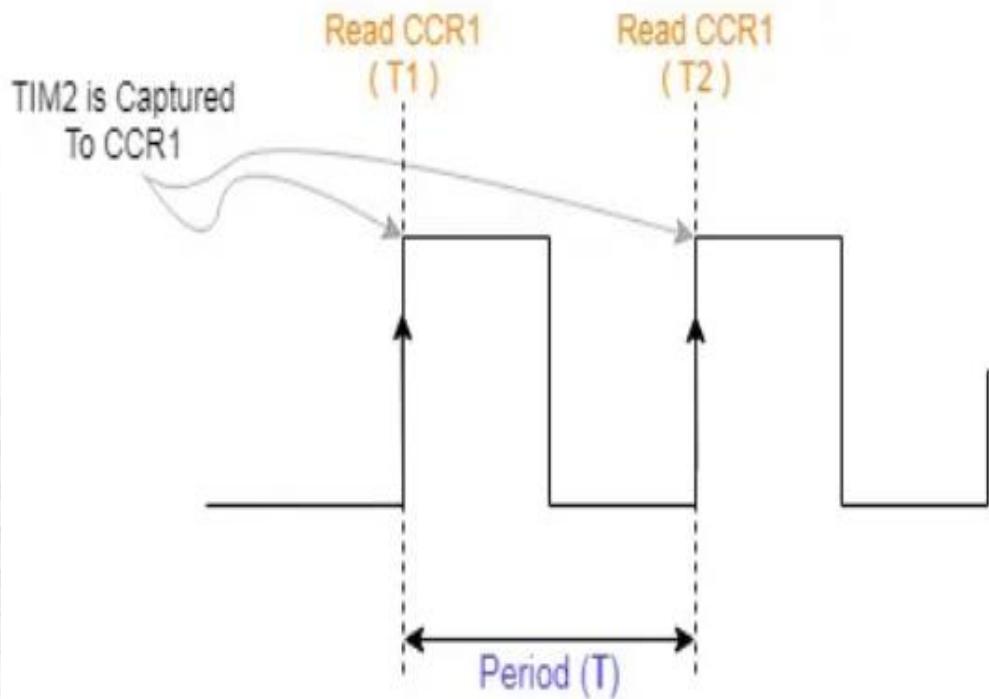
capture/compare output) channels مرتبطة بأقطاب

Input Capture Mode

تستخدم المسجلات (TIMx_CCRx) لمسك القيمة التي وصل إليها العداد بعد الانتهاء من قياس الإشارة القادمة، فعند حدوث حدث عملية الـ Capture يتم رفع العلم CCXIF والموجود ضمن المسجل TIMx_SR ويتم طلب مقاطعة أو DMA في حال كانت إحداهما مفعولة.



Input Capture Mode

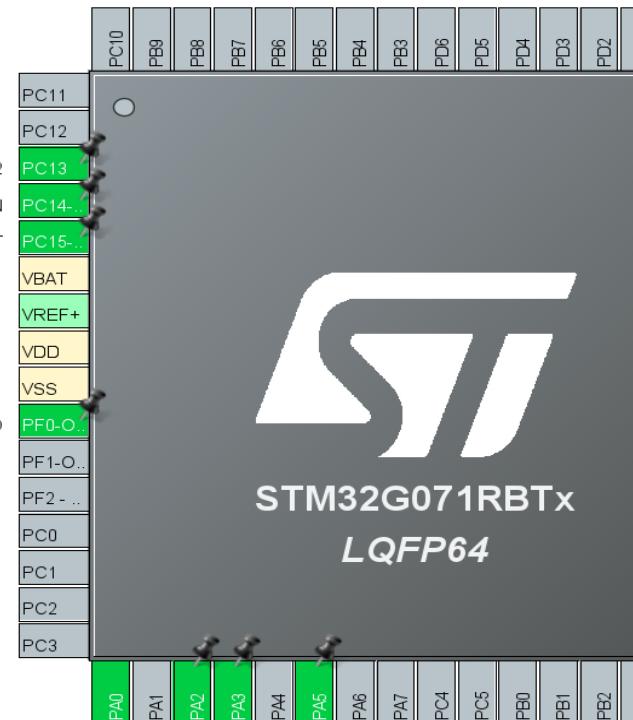


Timer Modes in STM32

Input Capture Mode

لضبط المؤقت في نمط الـ Input Capture من خلال

نختار ما يلي :



The screenshot shows the CubeMX Device Configuration Tool interface for a project named "input_capture.ioc". The main window displays the "Pinout & Configuration" tab, which is further divided into "Clock Configuration", "Project Manager", and "Tools". On the left, a sidebar lists various device components: System Core, Analog, Timers, Connectivity, Multimedia, Computing, Middleware, and Utilities. The "Timers" section is expanded, showing sub-components LPTIM1, LPTIM2, RTC, TIM1 through TIM17, and MCO. "TIM2" is selected and highlighted in blue.

TIM2 Mode and Configuration

- Slave Mode: Disable
- Trigger Source: Disable
- Clock Source: Internal Clock
- Channel1: Input Capture direct mode
- Channel2: Disable
- Channel3: Disable
- Channel4: Disable
- Combined Channels: Disable
- Use ETR as Clearing Source: Disable
- XOR activation
- One Pulse Mode

Configuration

- Reset Configuration:
 - NVIC Settings
 - DMA Settings
 - GPIO Settings
 - Parameter Settings
 - User Constants
- Configure the below parameters:
 - Search (Ctrl+F)
 - Counter Period (AutoRelo): 65535
 - Internal Clock Division (C... No Division)
 - auto-reload preload: Enable
- Trigger Output (TRGO) Parameters
 - Master/Slave Mode (MS...): Disable (Trigger input effect not ...)
 - Trigger Event Selection T... Reset (UG bit from TIMx_EGR)
- Input Capture Channel 1
 - Polarity Selection: Rising Edge
 - IC Selection: Direct
 - Prescaler Division Ratio: No division

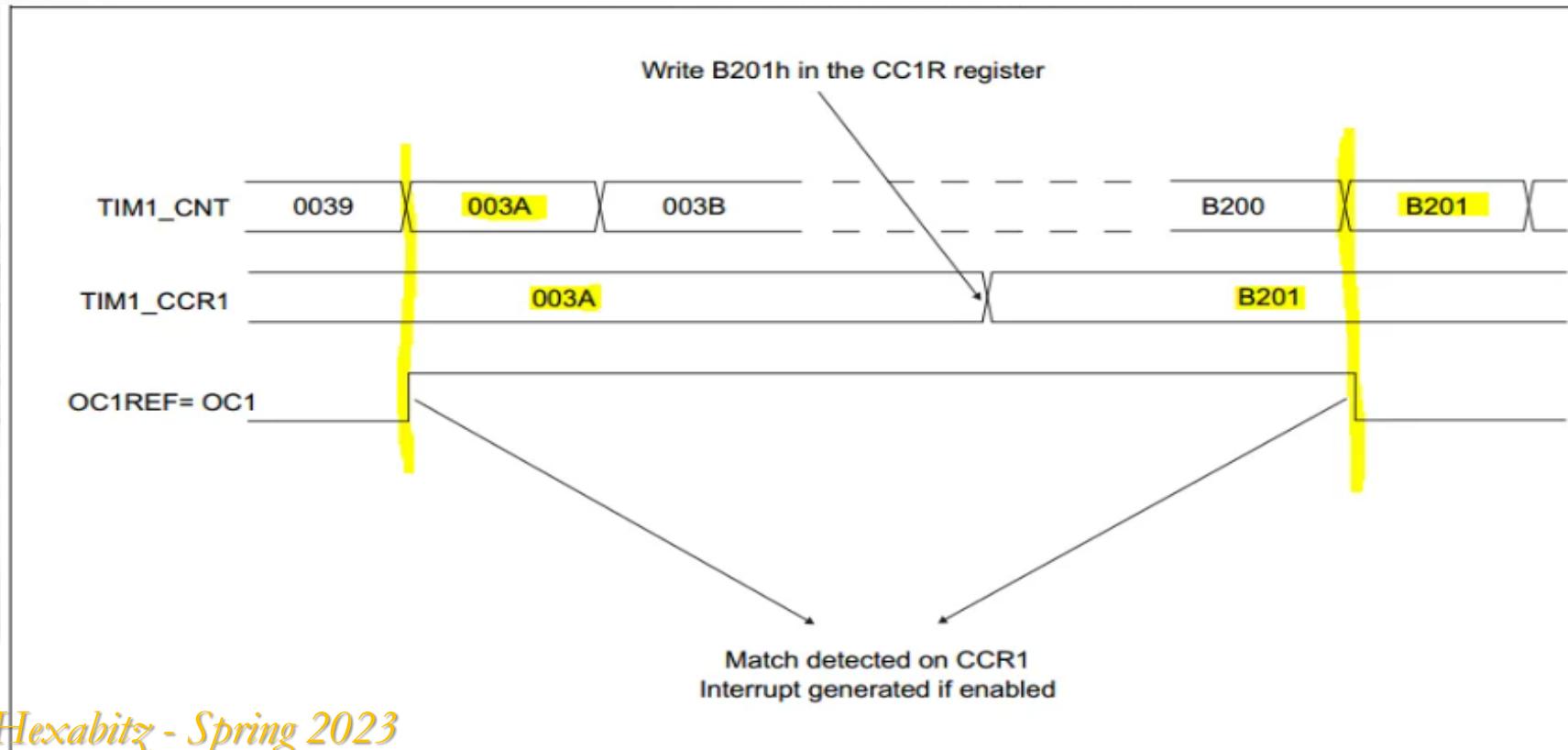
The configuration settings highlighted in orange are: "Clock Source: Internal Clock", "Channel1: Input Capture direct mode", "Counter Period (AutoRelo): 65535", "Internal Clock Division (C... No Division)", "auto-reload preload: Enable", and "Polarity Selection: Rising Edge".

Output Compare Mode

□ هذا النمط من العمل يسمح بالتحكم بخرج معين خلال زمن معين ،
فعد تتساوى القيمة الحالية للعداد مع القيمة المخزنة في مسجل
المقارنة OCR يتم تغيير حالة الخرج الرقمي وفقاً لما تمت برمجته
خلال الكود فمثلاً قد يصبح وضع HIGH أو LOW أو toggles
حتى يبقى بنفس وضعه دون تغيير

Output Compare Mode

يوضح الشكل التالي مثال على نمط Output compare حيث المطلوب عمل القطب الخرج عند زمن معين:

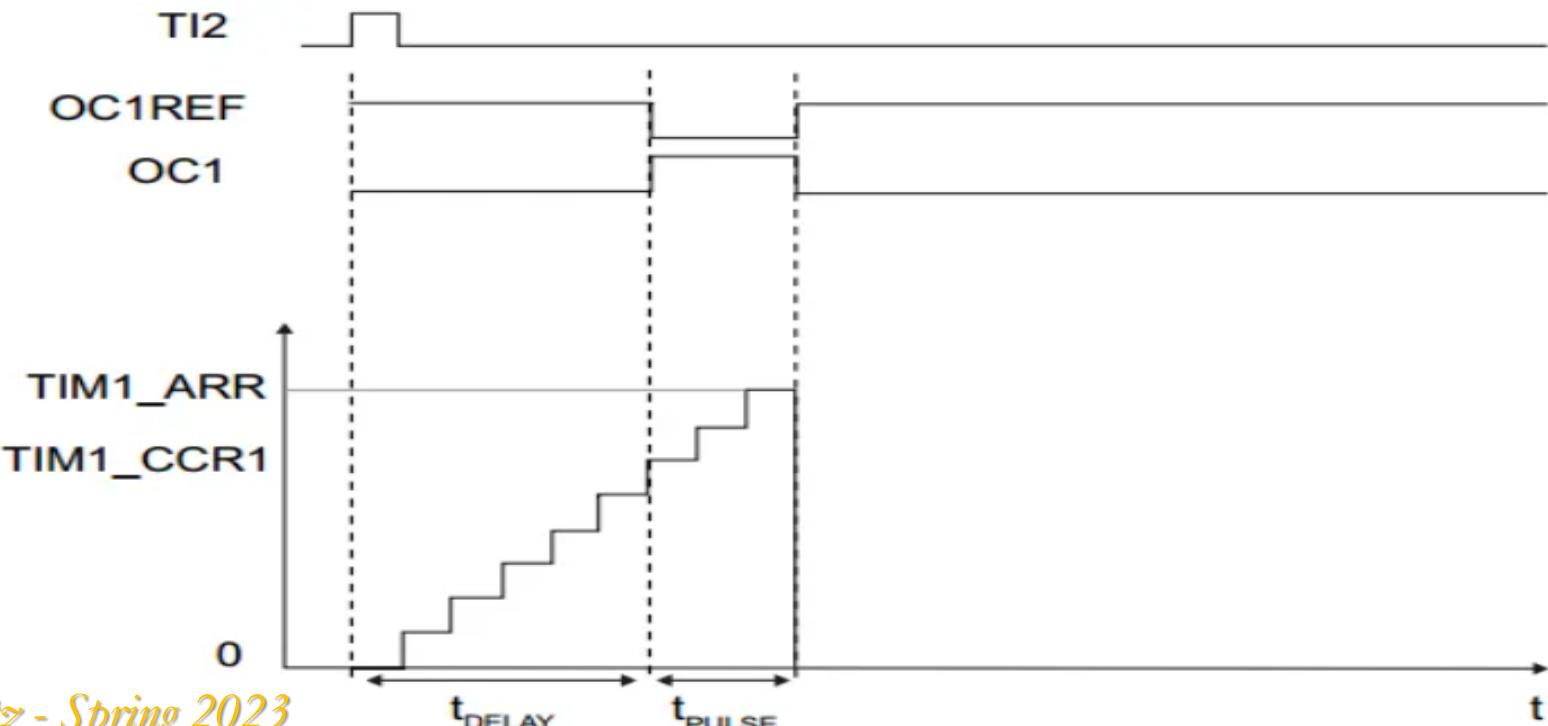


One Pulse Mode

يعتبر هذا النمط من العمل حالة خاصة من الأنماط السابقة، حيث يسمح للعداد بالاستجابة لحدث معين وتوليد نبضة بعرض وتأخير زمني معين يتم تحديدهم من خلال الكود، توليد النبضة يتم بنمط PWM أو Output compare، كما يتم توليدتها فقط عندما تكون قيمة المقارنة مختلفة عن القيمة الابتدائية للعداد، حيث يجب عند ضبط الإعدادات أن تكون $ARR > CNT \geq CCRx$ وبالأخص يجب أن تكون $CCRx > 0$

One Pulse Mode

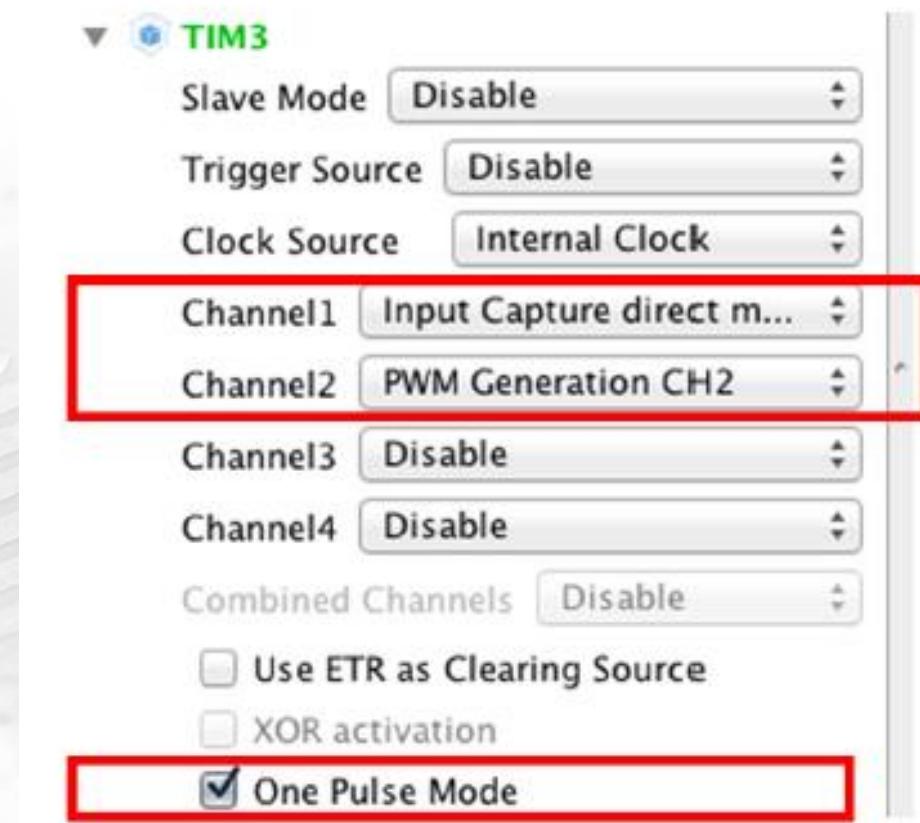
على سبيل المثال إذا أردنا توليد نبضة على الخرج OC1 وعرض tDELAY وبتأخير زمني tPULSE عند الجبهة الصاعدة لإشارة على القطب : □



One Pulse Mode

لضبط المؤقت في نمط الـ One Pulse Mode من خلال

نختار ما يلي :



Encoder Mode

يعمل المؤقت في نمط ال Encoder كعداد لنبضات ساعة خارجية

لمدخلين ، أي أن العداد يبدأ بالعد من الصفر حتى external clock

القيمة المخزنة في المسجل **TIMx_ARR**

حيث من خلال تتبع الإشارات على هذين المدخلين يتم تحديد اتجاه العد

تصاعدي/تنازلي بالإضافة إلى عد النبضات القادمة على هذين الدخلين

أي أن العداد يقوم بلاحقة السرعة واتجاه الدوران للانكودر المتصل مع

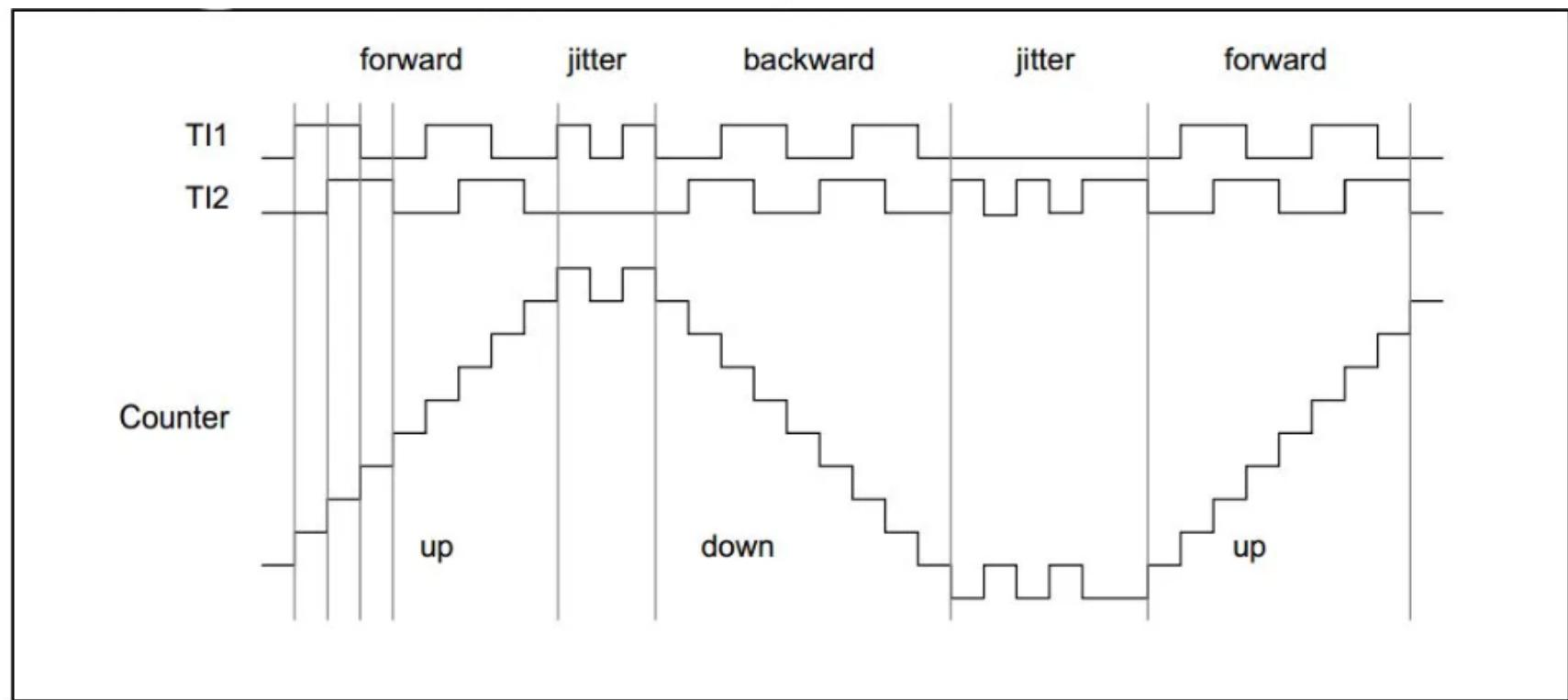
قناتي المؤقت، حيث لا يحتاج الانكودر لأي دارة ملائمة بينه وبين

المتحكم.

Encoder Mode

يمكن للمستخدم الحصول على معلومات ديناميكية كالسرعة والتسارع من

خلال استخدام مؤقت في نمط Encoder

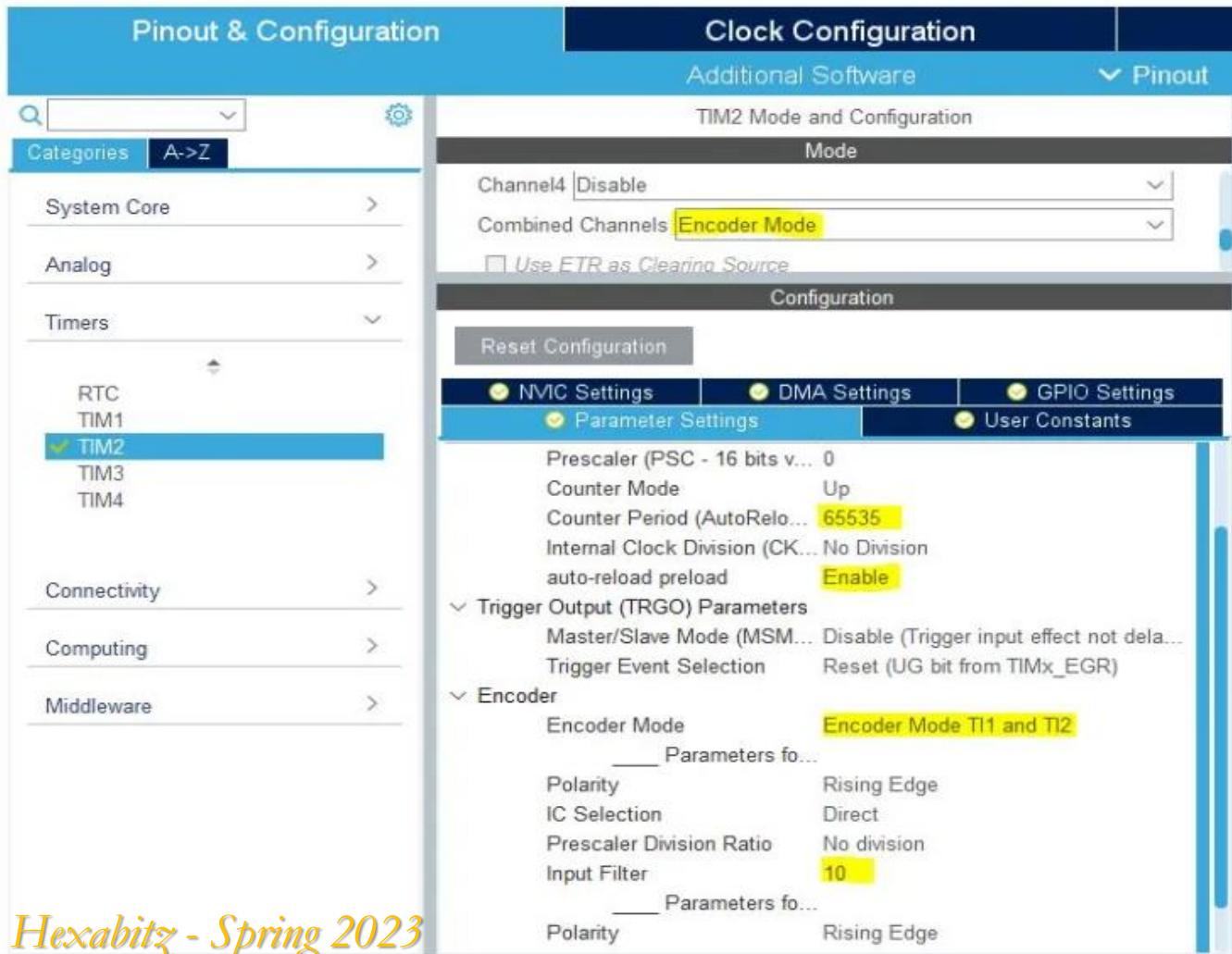


Timer Modes in STM32

Encoder Mode

لضبط المؤقت في نمط الـ Encoder Mode من خلال CubeMX

نختار ما يلي :



أهم مسجلات المؤقتات في متحكمات STM32

TIMx -> CNT : مسجل بطول 16 بت ويمثل القيمة الحالية للمؤقت

TIMx->ARR : مسجل بطول 16 بت وهو مسجل إعادة التحميل التلقائي، ويتم استخدامه للتحكم بتردد إشارة الـ PWM

TIMx->PSC : مسجل بطول 16 بت وهو مسجل المقسم التردددي

TIMx_CCRx ، Capture/Compare Registers، وأيضاً يستخدم للتحكم بدورة التشغيل dutycycle عند عمل المؤقت بنمط الـ PWM

تلخيص لدوال مكتبة HAL المستخدمة للتعامل مع المؤقتات

HAL_TIM_Base_Start(); :Polling

دالة بدء المؤقت بنمط الـ :Interrupt

HAL_TIM_Base_Start_IT();

دالة بدء المؤقت بنمط الـ :DMA

HAL_TIM_Base_Start_DMA();

دالة بدء المؤقت بنمط الـ :Input Capture

HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);

دالة بدء المؤقت بنمط الـ :PWM

HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);

دالة بدء المؤقت بنمط الـ :Encoder mode

HAL_TIM_Encoder_Start(&htim2, TIM_CHANNEL_ALL);

دالة خدمة مقاطعة الطفhan:

HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim)

{

}

دالة خدمة مقاطعة :Input Capture

HAL_TIM_IC_CaptureCallback()

{

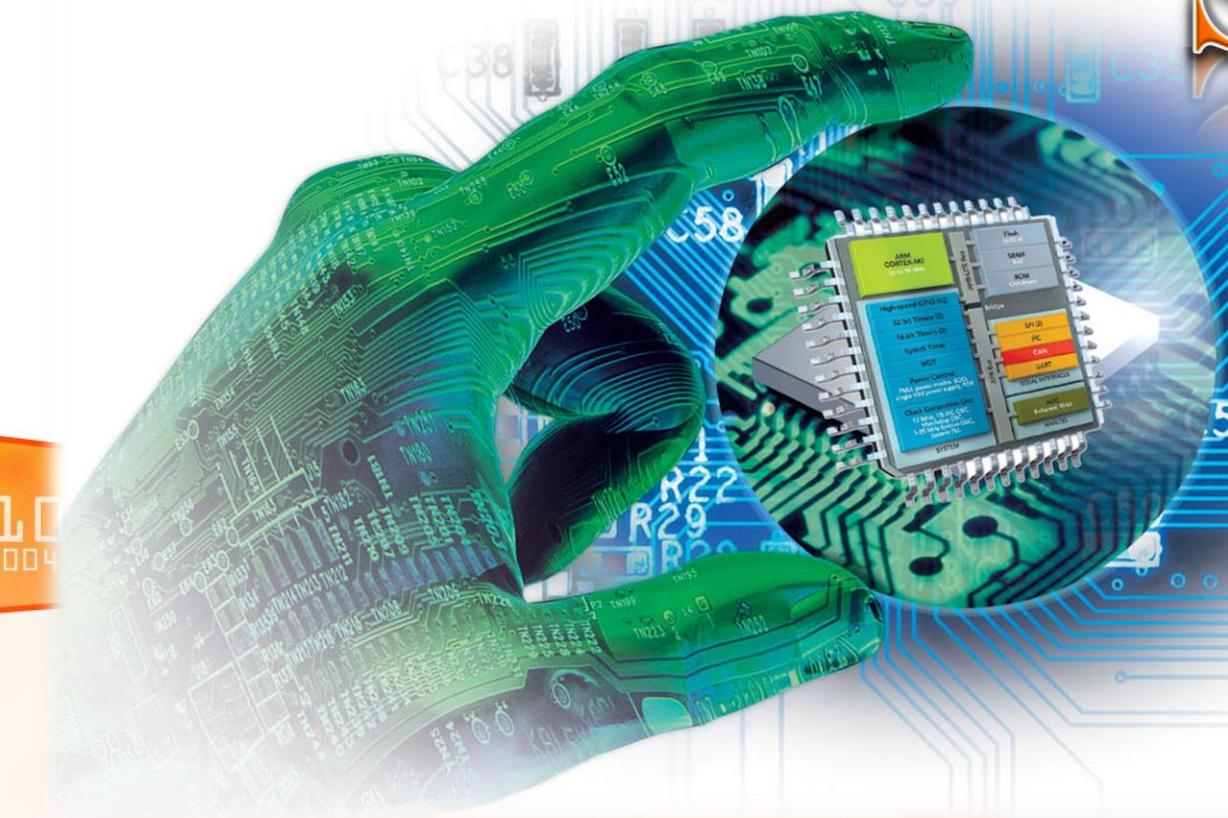
}

Thank you for listening

مُتَحَكِّمَات

STM32

11



- RUN (Range 1) at 64MHZ**
- RUN (Range 2) at 16MHZ**
- LPRUN at 2MHZ**
- SLEEP at 16 MHZ**
- LPSLEEP at 2MHZ**
- STOP 0 (full retention)**
- STOP1(no retention)**
- STANDBY**
- SHUTDOWN**

مزايا متحكمات STM32G0 المتعلقة بأنماط الطاقة

لمتحكمات STM32G0 عدّة مزايا فيما يتعلّق بأنماط الطاقة منها:

تؤمن متحكمات STM32G0 7 أنماط للطاقة المنخفضة وبقدرة عالية

على الاستيقاظ السريع fast Wakeup تتضمّن :

✓ استجرار لا يتّجاوز الـ 40nA مع القدرة على إيقاظ الـ CPU من خلال أحد أقطاب I/O

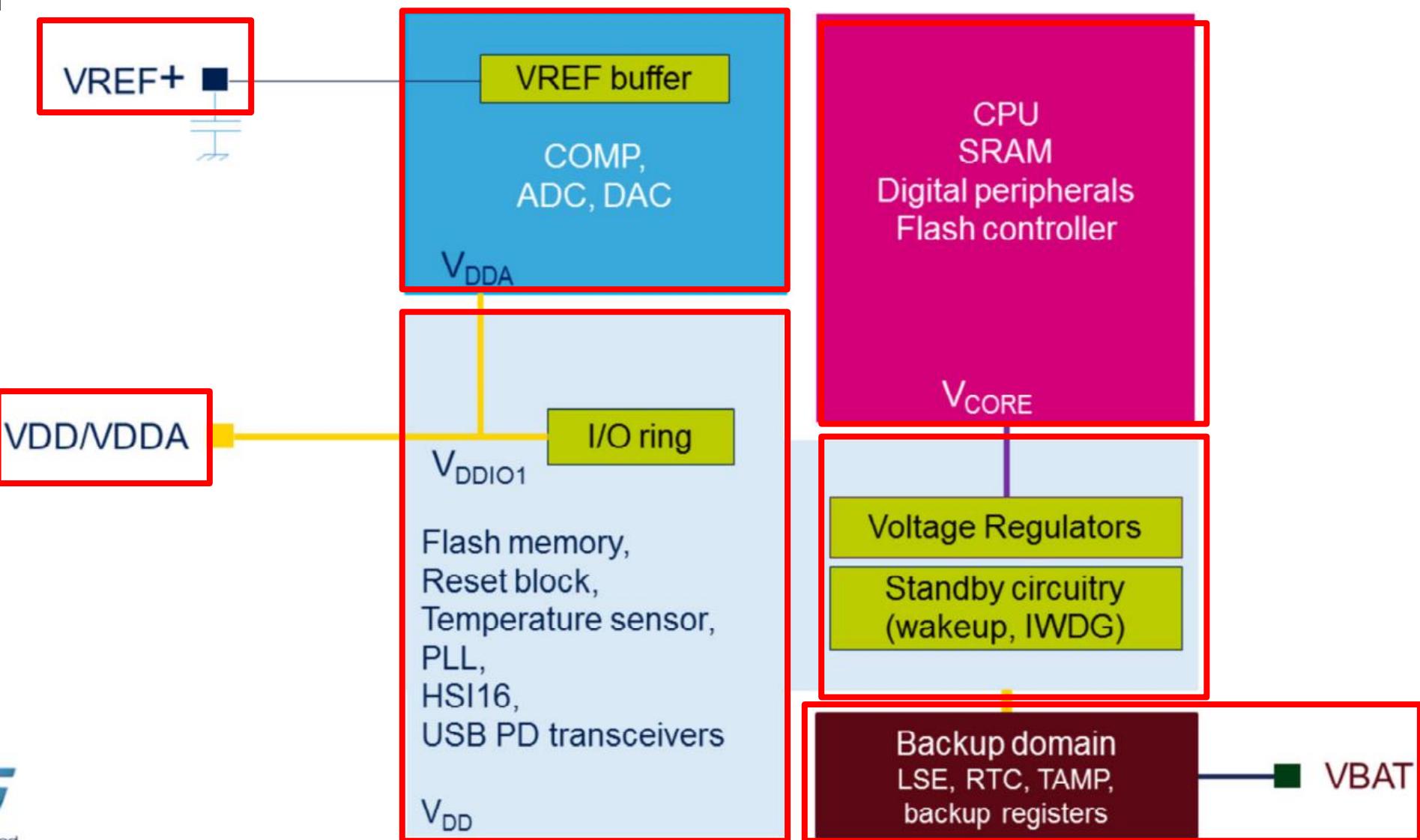
✓ استجرار لا يتّجاوز الـ 320nA لاسترجاع المعلومات من ذاكرة SRAM بحجم 36Kb

✓ كما بإمكان عدد كبير من الطرفيات الموجودة في المتحكم إيقاظ المعالج

استجرار المتحكم لا يتّجاوز الـ 100UA/MHZ عند عمل المتحكم بنمط Run mode وبأعلى تردد عمل ممكّن

كما لمتحكم STM32G0 قطب خاص لتغذية المتحكم من خلال بطارية VBAT

Power schemes



تحتوي متحكمات STM32G0 على منظمي جهد:

منظم الجهد الرئيسي Main regulator : وله مجال عمل يستخدمان عند عمل المتحكم في الأنماط التالية:

Run ✓

Sleep ✓

Stop0 ✓

منظم الطاقة المنخفضة Low-Power regulator : يستخدم عند عمل المتحكم في الأنماط التالية:

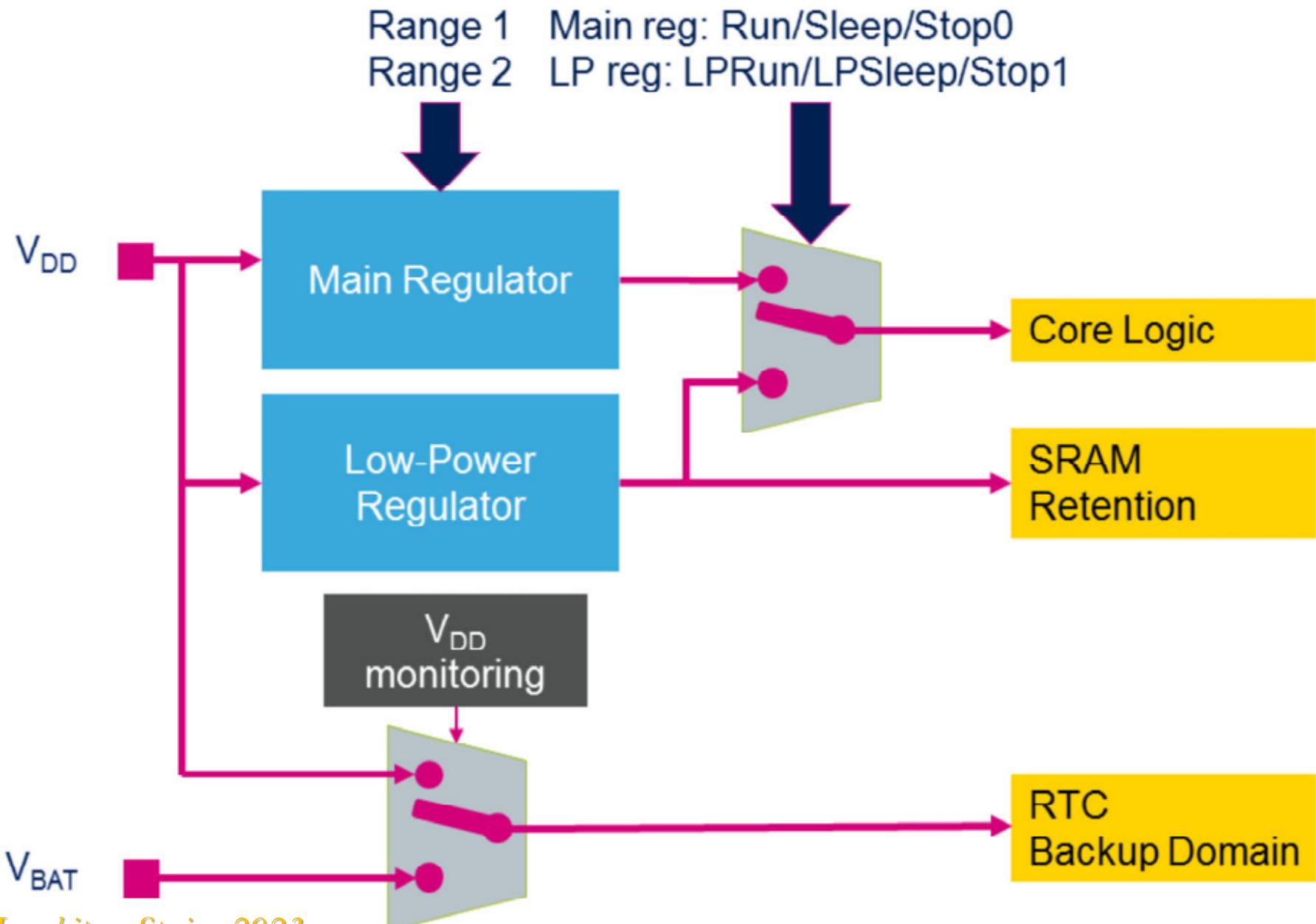
Low-power run ✓

Low-power sleep ✓

Stop1 ✓

Standby ✓

STM32G0 في متحكمات الجهد



لمتحكمات STM32G0 عدة أنماط عمل هي:

- RUN (Range 1) at 64MHZ
- RUN (Range 2) at 16MHZ
- LPRUN at 2MHZ
- SLEEP at 16 MHZ
- LPSLEEP at 2MHZ
- STOP 0 (full retention)
- STOP1(no retention)
- STANDBY + SRAM
- STANDBY
- SHUTDOWN

1. Run Mode

ويشمل 3 حالات:

- استخدام منظم الجهد الرئيسي Main regulator ضمن مجال العمل الأول Range1
- استخدام منظم الجهد الرئيسي Main regulator ضمن مجال العمل الثاني Range2
- استخدام منظم الطاقة المنخفضة Low-Power regulator

1. Run Mode: Rangel

- في هذا النمط من العمل يتم تزويد الـ CPU بنبضات الساعة ويتم قراءة و تنفيذ البرنامج من الذاكرة Flash أو الذاكرة SRAM
- أكبر تردد ساعة يمكن أن يعمل به المتحكم هو 64MHZ
- افتراضياً يتم تزويد الذاكرة SRAM بنبضات الساعة
- أيضاً يمكن تفعيل كافة الطرفيات الموجودة في المتحكم
- معدل استجرار التيار في هذا النمط من العمل هو 100uA/MHZ

1. Run Mode: Range1

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



Cortex M0+

Main regulator (MR)

Flash
memory

SRAM
(36 Kbytes)

Range 1 (up to 64 MHz)

Range 2 (up to 16 MHz)

Low Power regulator (LPR) up to 2 MHz

Range 1
100 μ A/MHz at 64 MHz
(6.4 mA)

Range 2
93 μ A/MHz at 16 MHz
(1.49 mA)

Available
clocks

HSI16

HSE

LSI

LSE

Active cell

Clocked-off
cell

Cell in power-
down

Available
Periph and clock

Ex: Execution from Flash memory

1. Run Mode: Range2

- أكبر تردد ساعة يمكن أن يعمل به المتحكم هو 16MHZ
- لا يمكن مسح أو برمجة الذاكرة Flash في هذا النمط
- يمكن تفعيل كافة الطرفيات الموجودة في المتحكم
- معدل استجرار التيار في هذا النمط من العمل هو 75uA/MHZ عندما تكون الذاكرة Flash في حالة إيقاف Off

1. Run Mode: Range2

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

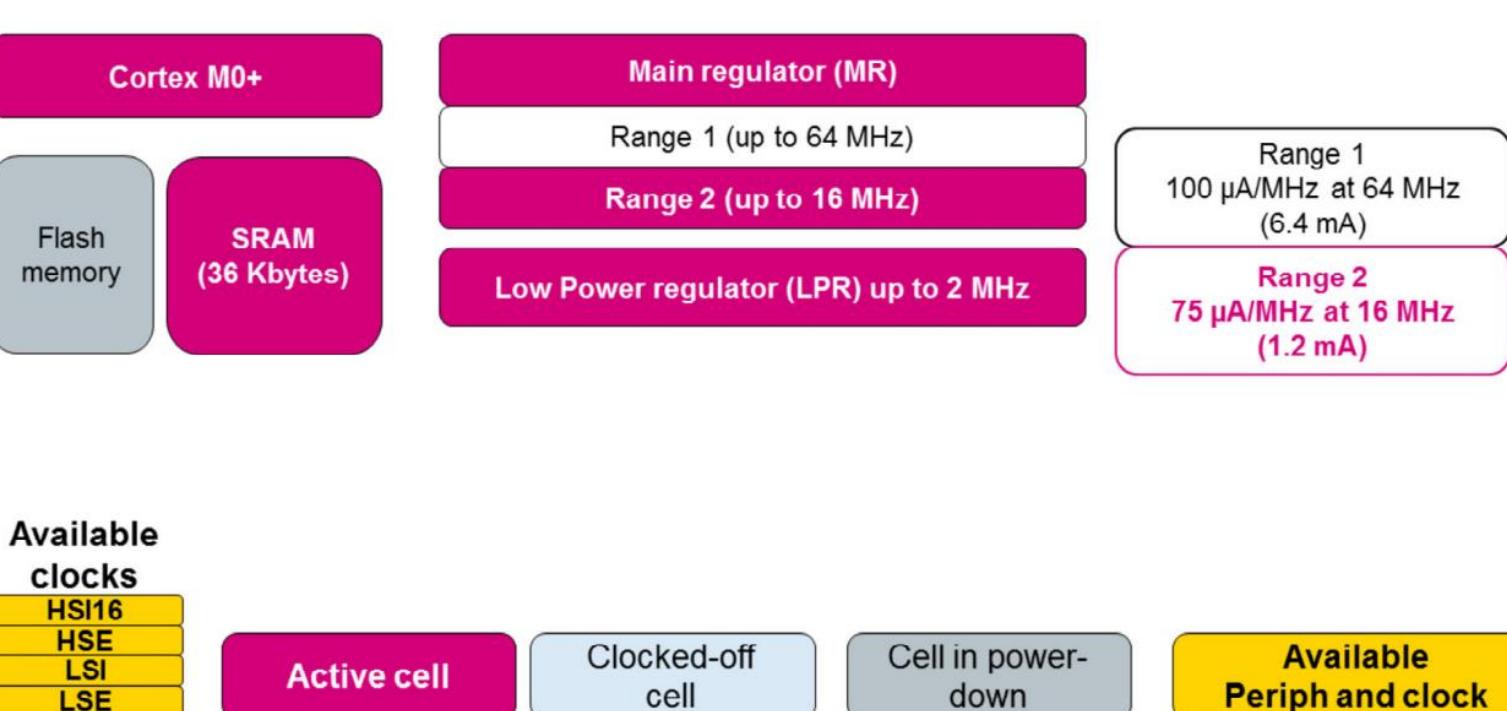
UCPD

RNG

AES

CRC

CEC



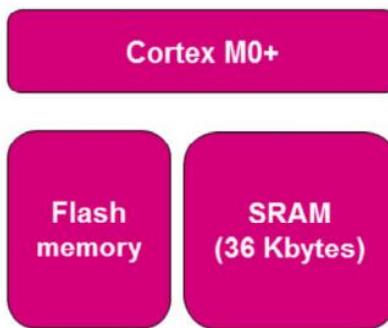
2. Low-power Run Mode

- ❑ في هذا النمط من العمل يتم تزويد الـ CPU بنبضات الساعة ويتم قراءة و تنفيذ البرنامج من الذاكرة Flash أو الذاكرة SRAM
- ❑ أكبر تردد ساعة يمكن أن يعمل به المتحكم هو 2MHZ
- ❑ يمكن فصل التغذية عن الذاكرة Flash لتوفير الطاقة
- ❑ أيضاً يمكن تفعيل كافة الطرفيات الموجودة في المتحكم
- ❑ معدل استجرار التيار في هذا النمط من العمل هو 103uA/MHZ في حالة قراءة الكود من الذاكرة Flash ، و 79uA/MHZ في حالة قراءة الكود من الذاكرة SRAM
- ❑ في هذا النمط يتم فصل المنظم الرئيسي ، ويتم تغذية الوحدات المنطقية من الـ Low power Regulator

2. Low-power Run 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



Ex: Execution from Flash memory

Available clocks

HSI16
HSE
LSI
LSE

Active cell

Clocked-off cell

Cell in power-down

Available Periph and clock

From Flash
103 µA/MHz at 2 MHz
(207 µA)

From SRAM
79 µA/MHz at 2 MHz
(159 µA)

Run and Low-power Run Modes

- أكبر تردد يمكن أن يعمل عنده المتحكم في نمط (Run mode(rang1)) هو 64MHZ ويمكن استخدام الكريستالة الداخلية أو الخارجية
- أكبر تردد يمكن أن يعمل عنده المتحكم في نمط (Run mode(rang2)) هو 16MHZ ويمكن استخدام الكريستالة الداخلية أو الخارجية
- أكبر تردد يمكن أن يعمل عنده المتحكم في نمط Low-power هو 2MHZ

Voltage range	SYSCLK	HSI16	HSE	PLL
Range 1	64 MHz max	16 MHz	48 MHz	128 MHz VCO max = 344 MHz
Range 2	16 MHz max	16 MHz	16 MHz	40 MHz VCO max = 128 MHz
Low-power run	2 MHz max	Allowed with divider	Allowed with divider	Not allowed

Sleep and Low-power Sleep Modes

ويشمل 3 حالات:

- Main : استخدام منظم الجهد الرئيسي Sleep Mode (range1)
ضمن مجال العمل الأول Range1 regulator
- Main : استخدام منظم الجهد الرئيسي Sleep Mode(range2)
ضمن مجال العمل الثاني Range2 regulator
- Low- : استخدام منظم الطاقة المنخفضة Low-power Sleep
Power regulator

Sleep and Low-power Sleep Modes

في هذا النمط من العمل:

يكون المعالج متوقف عن العمل stopped

يمكن فصل أو وصل نبضات الساعة عن أي طرفية من الطرفيات

يتم الدخول بهذا النمط من خلال تنفيذ إحدى التعليمتين WFI(wait و WFE(Wait For Event) أو (for Interrupt)

هناك آليتين للدخول بهذا النمط:

بمجرد تنفيذ إحدى التعليمتين WFI/WFE

عند الانتهاء من تنفيذ برنامج خدمة المقاطة صاحب أدنى أولوية Interrupt Sub Routine

3. Sleep Mode : Range1

- في هذا النمط من العمل يتم فصل نبضات الساعة عن الـ CPU
- أكبر تردد ساعة يمكن أن يعمل به المتحكم هو 64MHZ
- افتراضياً يتم تزويد الذاكرة SRAM بنبضات الساعة كما يمكن فصلها من خلال الكود
- أيضاً يمكن تفعيل كافة الطرفيات الموجودة في المتحكم
- معدل استجرار التيار في هذا النمط من العمل هو 25uA/MHZ

3. Sleep Mode : Range1

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



Cortex M0+

Flash
memory

SRAM
(36 Kbytes)

Main regulator (MR)

Range 1 (up to 64 MHz)

Range 2 (up to 16 MHz)

Low Power regulator (LPR) up to 2 MHz

Range 1
25 μ A/MHz at 64 MHz
(1.6 mA)

Range 2
25 μ A/MHz at 16 MHz
(0.4 mA)

Available clocks

HSI16
HSE
LSI
LSE

Active cell

Clocked-off
cell

Cell in power-
down

Available
Periph and clock

Ex: Flash memory ON

3. Sleep Mode : Range2

- في هذا النمط من العمل لا يمكن مسح أو برمجة الذاكرة Flash
- أكبر تردد ساعة يمكن أن يعمل به المتحكم هو 16MHZ
- معدل استجرار التيار في هذا النمط من العمل هو 25uA/MHZ

3. Sleep Mode : Range2

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



Zzz

Cortex M0+

Flash
memory

SRAM
(36 Kbytes)

Ex: Flash memory ON but cannot be programmed or erased

Main regulator (MR)

Range 1 (up to 64 MHz)

Range 2 (up to 16 MHz)

Low Power regulator (LPR) up to 2 MHz

Range 1
25 μ A/MHz at 64 MHz
(1.6 mA)

Range 2
25 μ A/MHz at 16 MHz
(0.4 mA)

Available
clocks

HSI16
HSE
LSI
LSE

Active cell

Clocked-off
cell

Cell in power-
down

Available
Periph and clock

4. Low-power Sleep Mode

- في هذا النمط من العمل يتم فصل نبضات الساعة عن الـ CPU
- يمكن تزويد الوحدات المنطقية بال питания من خلال المنظم Low-
power regulator
- أكبر تردد ساعة يمكن أن يعمل به المتحكم هو 2MHZ
- يمكن فصل SRAM من خلال الكود
- يمكن فصل الذاكرة flash أو جعلها تعمل بنمط Power down
- أيضاً يمكن تفعيل كافة الطرفيات الموجودة في المتحكم
- معدل استجرار التيار في هذا النمط من العمل هو 46.5uA/MHZ عند فصل الـ flash

4. Low-power Sleep 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



Zzz

Cortex M0+

Flash
memory

SRAM
(36 Kbytes)

Main regulator (MR)

Range 1 (up to 64 MHz)

Range 2 (up to 16 MHz)

Low Power regulator (LPR) up to 2 MHz

Ex: Flash memory OFF

Flash OFF, SRAM OFF
46.5 μ A/MHz at 2 MHz
(93 μ A)

Available clocks

HSI16
HSE
LSI
LSE

Active cell

Clocked-off
cell

Cell in power-
down

Available
Periph and clock

Stop Modes

ويشمل حالتين:

:Stop0 Mode

:Stop1 Mode

5. Stop 0 Mode

- في هذا النمط من العمل يتم فصل نبضات الساعة عن الـ CPU
- منظم الجهد المستخدم في هذا النمط هو المنظم الرئيسي
- يتم إلغاء تفعيل كل من HIS, HSE,PLL
- يتم تزويد وحدة الـ RTC بنبضات الساعة من خلال the internal or external low-speed oscillator
- معظم الطرفيات الموجودة في المتحكم تكون في حالة عدم تفعيل
- بعض الطرفيات تبقى في حالة تفعيل كـ Power voltage detector, ADC, comparators, independent watchdog, low power timers, I2C, UART and low-power UART
- معدل استجرار التيار في هذا النمط من العمل هو 97uA/MHZ في حال عدم تفعيل الهزاز الكريستالي الداخلي HSI
- زمن الـ Wakeup بحدود 2usec

5. Stop 0 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

97 μ A @ 3.0 V

Wakeup time to 16 MHz:

- In SRAM: 2 μ s
- In Flash ON: 2 μ s
- In Flash OFF: 5.5 μ s

Main regulator (MR)

Range 1 (up to 64 MHz)

Range 2 (up to 16 MHz)

Low Power regulator (LPR) up to 2 MHz

Backup domain

Backup Register (5x32 bits)

RTC & TAMPER

Wake-up event

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

Active cell

Clocked-off
cell

Cell in power-
down

Available
Periph and clock

5. Stop 1 Mode

- هذا النمط من العمل مشابه للنمط السابق ماعدا أن منظم الجهد المستخدم هو **Low-power regulator**
- معدل استجرار التيار في هذا النمط من العمل عند عدم تفعيل الـ **RTC** هو **1.3uA/MHZ**
- زمن الـ **Wakeup** بحدود **5usec**

5. 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



Cortex M0+

Main regulator (MR)

Flash
memory

SRAM
(36 Kbytes)

Low Power regulator (LPR) up to 2 MHz

Backup domain

Backup Register (5x32 bits)

RTC & TAMPER

Available
clocks

HSI16
HSE
LSI
LSE

Active cell

Clocked-off
cell

Cell in power-
down

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

Stop Modes comparison

- معدل استجرار التيار في نمط الـ stop0 أعلى منه في نمط الـ stop1
- بينما زمن الـ Wakeup في نمط الـ stop0 أقل منه في نمط الـ stop1
- منظم الجهد المستخدم في نمط الـ stop0 هو المنظم الرئيسي بينما في نمط الـ stop1 المنظم المستخدم هو Low-power regulator

	STOP0	STOP1
Consumption	25 °C, 3 V 97 µA	1.3 µA when RTC is disabled
Wakeup time to 16 MHz	5.5 µs in Flash memory initially powered down 2 µs in RAM	9 µs in Flash memory initially powered down 5 µs in RAM
Wakeup clock		HSI16 at 16 MHz
Regulator	Main regulator	Low power regulator
Peripherals	RTC, I/Os, BOR, PVD, COMPs, IWDG 2 LP TIMERS 1 LP UART (Start, address match or byte reception) 2 U(S)ARTx (Start, address match or byte reception) 1 I2C (address match)	

6. Standby Mode

- يعتبر هذا النمط من العمل هو **lowest power mode** حيث يمكن الاحتفاظ بحوالي الـ **36Kbyte** من الذاكرة **SRAM**
- يمكن الانتقال بشكل آلي من التغذية من خلال قطب الـ **VDD** إلى قطب الـ **VBAT**
- افتراضياً تكون منظمات الجهد في حالة **power down mode** وبالتالي فقد كل البيانات المخزنة ضمن الـ **SRAM** ومسجلات الطرفيات
- الساعة المستخدمة للـ **wakeup** هي الهزاز الكريستالي الداخلي **HSI** وبتردد **16MHZ**
- وضع الـ **low power Brown-Out Reset (BOR)** يكون مفعل بشكل دائم في هذا النمط مما يضمن عمل **Reset** للمتحكم عند انخفاض جهد التغذية

6. Standby 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 can be configured
w/ or w/o pull-up
w/ or w/o pull-down

+

Cortex M0+

Main regulator (MR)

Flash
memory

SRAM
(36 Kbytes)

Low Power regulator (LPR) up to 2 MHz

Backup domain

Backup Register (5x32 bits)

RTC & TAMPER

Available
clocks

HSI16

HSE

LSI

LSE

Active cell

Clocked-off
cell

Cell in power-
down

Wake-up event

NRST

BOR

PVD

RTC + Tamper

USART

LP UART

I2C 1

CEC

COMP

LPTIM 1

LPTIM 2

IWDG

GPIOs

(WKUP pins)

Available
Periph and clock

7. Shutdown Mode

- استجرار التيار في هذا النمط هو 40 nA at 3.0 V
- هو نمط مشابه لنمط الـ **standby** مع وجود اختلاف من ناحية الـ **power down reset** أي أن الـ **power monitoring** عدم تفعيل وبالتالي عدم القدرة على تبديل تغذية المتحكم إلى **VBAT** ، وهذا يعني أن المتحكم سيكون في حالة عدم استقرار في حال انخفض جهد التغذية عن الـ $1.6V$
- الهاز الكريستالي المنخفض السرعة **LSI** يكون في حالة عدم تفعيل وبالتالي مؤقت المراقبة الـ **watchdog** في حالة عدم التفعيل
- هناك 5 مصادر للـ **wakeup** من خلال 5 أقطاب من أقطاب المتحكم وأيضاً من خلال الـ **RTC** و **tamper events**

7. Shutdown Mode

- تكون منظمات الجهد في حالة **power down mode**
- يتم تزويد وحدة الـ RTC بنبضات الساعة من خلال هزاز كريستالي
- خارجي منخفض السرعة **external low-speed clock**
- زمن الـ **wakeup** في هذا النمط بحدود 250usec

7. Shutdown 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 can be configured
w/ or w/o pull-up
w/ or w/o pull-down

But floating when exit from Shutdown

w/o RTC: 40 nA @ 3.0 V
w/ RTC: 350 nA @ 3.0 V

Wakeup time to 16 MHz:
In Flash memory: 250 µs

Cortex M0+

Main regulator (MR)

Flash
memory

SRAM
(36 Kbytes)

Low Power regulator (LPR) up to 2 MHz

Backup domain

Backup Register (5x32 bits)

RTC & TAMPER

Available clocks

HSI16
HSE
LSI
LSE

Active cell

Clocked-off
cell

Cell in power-
down

Wake-up event

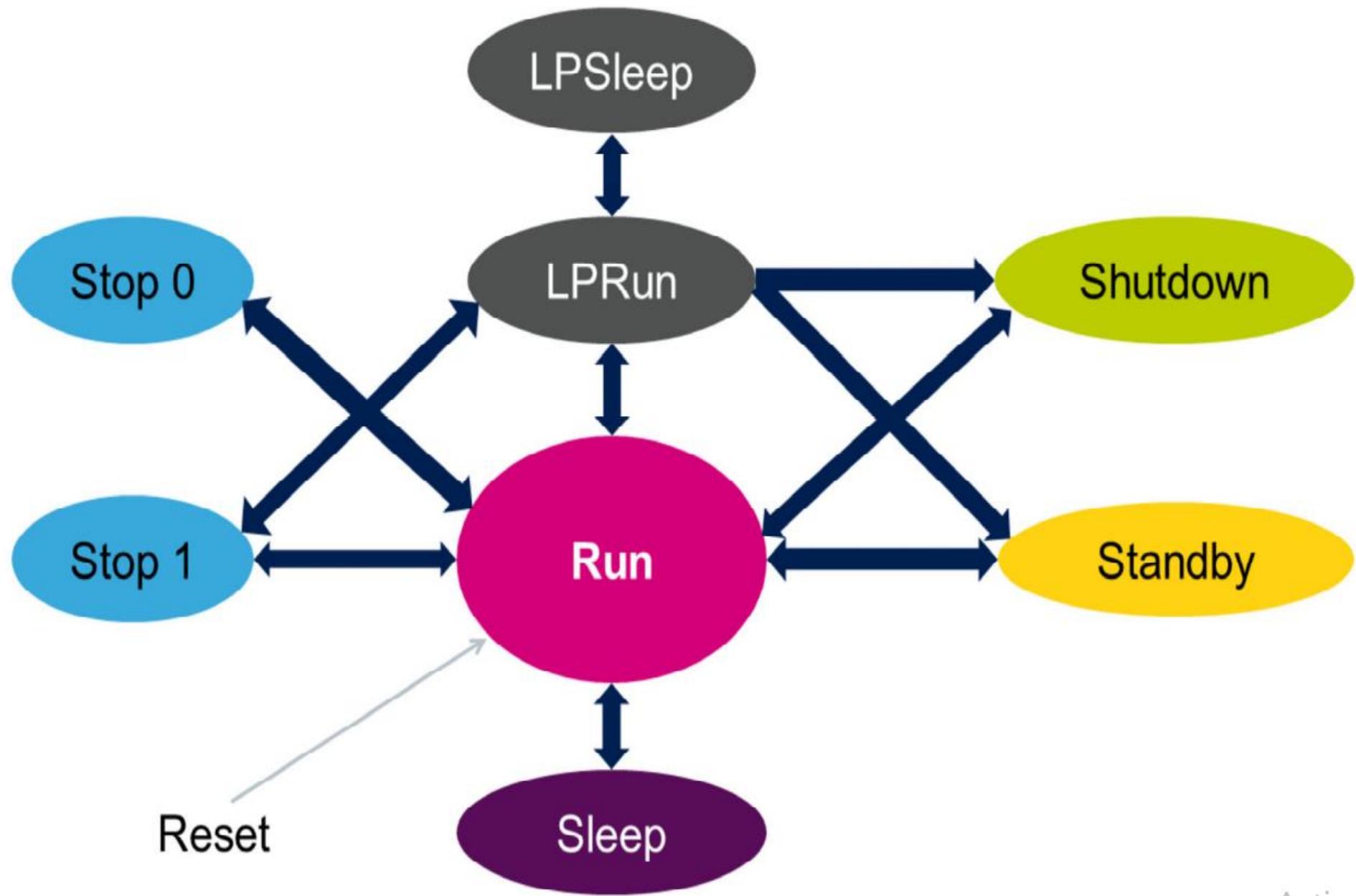
NRST
BOR
PVD
RTC + Tamper
USART
LP UART
I2C 1
CEC
COMP
LPTIM 1
LPTIM 2
IWDG
GPIOs (WKUP pins)

Available
Periph and clock

Low-power Modes Summary

Mode	Regulator	CPU	Flash	SRAM	Clocks	Peripherals	Wakeup time
Run	MR Range 1	Yes	ON ⁽¹⁾	ON	Any	All	N/A
	MR Range 2					All	
LPRun	LPR	Yes	ON ⁽¹⁾	ON	Any except PLL	All	
Sleep	MR Range 1	No	ON ⁽¹⁾	ON ⁽²⁾	Any	All	11 cycles
	MR Range 2					Any interrupt or event	
LPSleep	LPR	No	ON ⁽¹⁾	ON ⁽²⁾	Any except PLL	All Any interrupt or event	11 cycles
Stop 0	MR	No	OFF	ON	LSE/LSI	Reset pin, all I/Os BOR, PVD, RTC, IWDG, COMP, DAC, USARTx, LPUART, I2C, LPTIMx, UCPD, CEC	2 µs RAM 5.5µs Flash memory
Stop 1	LPR						5 µs RAM 9 µs Flash memory
Standby	LPR	DOWN	OFF	SRAM ON	LSE/LSI	Reset pin, 5 WKUPx pins BOR, RTC, IWDG	14 µs
	OFF			DOWN			
Shutdown	OFF	DOWN	OFF	DOWN	LSE	Reset pin, 5 WKUPx pins RTC	258 µs

Low-power Modes Summary



دوال مكتبة HAL المستخدمة لإدخال المتحكم في نمط الـ sleep mode

لإدخال المتحكم في نمط الـ **sleep mode** يجب أولاً إلغاء تفعيل مقاطعة systick interrupt وإلا ستقوم هذه المقاطعة في كل مرة بإيقاظ المتحكم:

HAL_SuspendTick();

يتم إدخال المتحكم بنمط الـ **sleep mode** من خلال التعليمة WFI (Wait For Interrupt) وفي هذه الحالة يمكن إيقاظ المتحكم من خلال إحدى مقاطعات الطرفيات، أو من خلال التعليمة WFE (Wait For Event)

HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFI);

دوال مكتبة HAL المستخدمة لإيقاظ المتحكم من نمط الـ sleep mode

في حال استخدمنا التعليمة WFI فإن المتحكم يمكن إيقاظه من خلال أي من مقاطعات الطرفيات، أي أن المتحكم سيسنّيّقظ بمجرد حدوث أي مقاطعة، وفي هذه الحالة يجب إعادة تفعيل مقاطعة الـ systick بعد أن يتم إيقاظ المتحكم :

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    HAL_ResumeTick();
}
```

دوال مكتبة HAL المستخدمة لإيقاظ المتحكم من نمط ال sleep mode

أيضاً يمكن إيقاظ المتحكم من نمط ال sleep mode عند حدوث مقاطعة يقوم بالذهاب إلى برنامج خدمة المقاطعة ويقوم بتنفيذها وعند انتهاءه يعود مجدداً لنمط ال sleep mode وفي هذه الحالة فإن المتحكم يعمل فقط في نمط المقاطعة interrupt mode ، لاستخدام هذه الميزة نستخدم الدالة التالية قبل دالة تفعيل نمط ال sleep mode :

HAL_PWR_EnableSleepOnExit();

ولإلغاء تفعيل نمط المقاطعة نستخدم:

HAL_PWR_DisableSleepOnExit();

دوال مكتبة HAL المستخدمة لإدخال المتحكم في نمط ال stop mode

لإدخال المتحكم في نمط ال stop mode يجب أولاً إلغاء تفعيل مقاطعة systick interrupt وإلا ستقوم هذه المقاطعة في كل مرة بإيقاظ المتحكم:

HAL_SuspendTick();

يتم إدخال المتحكم بنمط ال WFI من خلال التعليمة stop mode (Wait For Interrupt) وفي هذه الحالة يمكن إيقاظ المتحكم من خلال أحدى مقاطعات الطرفيات، أو من خلال التعليمة WFE (Wait For Event)

HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);

دوال مكتبة HAL المستخدمة لإيقاظ المتحكم من نمط الـ stop mode

في حال استخدمنا التعليمة WFI فإن المتحكم يمكن إيقاظه من خلال أي من مقاطعات الطرفيات، RTC ، watchdog (IWDG) ، أي أن المتحكم سيستيقظ بمجرد حدوث أي مقاطعة، وفي هذه الحالة يجب إعادة تفعيل مقاطعة الـ systick interrupt بعد أن يتم إيقاظ المتحكم :

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        SystemClock_Config();
        HAL_ResumeTick();
    }
}
```

دوال مكتبة HAL المستخدمة لإدخال المتحكم في نمط الـ standby mode

لإدخال المتحكم في نمط الـ standby mode :

يجب أولاً تصفير أعلام الاستيقاظ

```
/* Clear the WU FLAG */
```

```
__HAL_PWR_CLEAR_FLAG(PWR_FLAG_WU);
```

```
/* clear the RTC Wake UP (WU) flag */
```

```
__HAL_RTC_WAKEUPTIMER_CLEAR_FLAG(&hrtc,  
RTC_FLAG_WUTF);
```

دوال مكتبة HAL المستخدمة لإدخال المتحكم في نمط ال standby mode

ثم تفعيل قطب الإيقاظ

HAL_PWR_EnableWakeUpPin(PWR_WAKEUP_PIN1);

يمكن أيضاً استخدام الإيقاظ الدوري باستخدام ال RTC

```
if (HAL_RTCEx_SetWakeUpTimer_IT(&hrtc, 0x2710,  
RTC_WAKEUPCLOCK_RTCCLK_DIV16)!=  
HAL_OK)
```

{

Error_Handler();

}

حيث يمثل الرقم 0x2710 الزمن المرغوب لإيقاظ المتحكم عنده وهو هنا يمثل

5sec

دوال مكتبة HAL المستخدمة لإدخال المتحكم في نمط الـ standby mode

ثم أخيراً للدخول في نمط الـ **standby**

HAL_PWR_EnterSTANDBYMode();

Thank you for listening