متحكمات STM32

سنتطرق في هذا الكورس إلى المواضيع 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

متحكمات STM32

1

University of Aleppo - Hexabitz - Spring 2023

موضوعات المحاضرة:

- العوامل المؤثرة في تصميم الأنظمة المدمجة.
- ماهو المتحكم المصغر.
- معيارية تصميم بنية المعالجات.
- بنى مسجلات التعليمات في المعالجات.
- مقارنة بين المتحكم المصغر والمعالج المصغر
- المعالجات المبنية بواسطة ARM:
- الهيكلية CORTEX- ARM
- مزايا المتحكم STM32G0
- أنواع اللوحات التطويرية المتوفرة Boards type
- تغذية متحكمات STM32
- مصادر الساعة في متحكمات STM32

العوامل المؤثرة في تصميم الأنظمة المدمجة:

- سعة المعالجة (Processing Power) – MIPS
- 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 بهوية النظام – المعالجة في الزمن الحقيقي

:Microcontroller VS microprocessor

What is Microcontroller ?

المكونات الأساسية للمتحكم المصغر ذو الاغراض العامة

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 الـCPU.

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.

مقارنة بين ذواكر المتحكم المصغر

النوع الذاكرة	الاستخدام	سرعة الكتابة	عدد مرات الكتابة والمسح	ديمومة المعطيات
RAM	مكان معالجة المعطيات	سرعة جداً من مرتبة ns	غير محدود	نفقد المعطيات بانقطاع التغذية
FLASH	مكان تخزين البرنامج	بطيئة من مرتبة ms	قد يتجاوز 100000 مرة	لا نفقد المعطيات بانقطاع التغذية
EEPROM	مكان تخزين معطيات المستخدم	بطيئة من مرتبة ms ولكنها اسرع من FLASH	قد يتجاوز 100000 مرة	لا نفقد المعطيات بانقطاع التغذية

أشهر عوائل المتحكمات المصغرة

ARM Cortex-M based

أشهر عوائل المتحكمات المصغرة

أشهر لغات البرمجة الشائعة لبرمجة المتحكمات المصغرة :

Low level

High level

- Assembly
- C
- C++
- Basic
- Java
- Python
- Matlab
- Visual programming (Labview, Simulink, etc.)

أشهر لغات البرمجة الشائعة لبرمجة المتحكمات المصغرة :

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

معيارية تصميم بنية المعالجات :

معيارية 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.

معيارية تصميم بنية المعالجات :

معيارية Harvard: ناقلين منفصلين أحدهما لنقل التعليمات والآخر لنقل البيانات وتختلف ذاكرة البيانات عن ذاكرة التعليمات حيث أن لكل ذاكرة خطوط عنونة وتحكم وممر معطيات مختلفة، وبالتالي تتم عملية قراءة التعليمات والبيانات في نفس الوقت...

بنى مسجلات التعليمات في المعالجات:

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 (SOC) و system on module (SOM) .
- تمتاز معالجات ARM بكلفتها المنخفضة واستهلاكها المنخفض للطاقة وأيضاً توليد حرارة أقل مقارنةً مع نظيراتها من المعالجات، لذا تعتبر المعالجات المثالية لاستخدامها في الأجهزة المحمولة التي تعتمد على البطاريات في تغذيتها كالهواتف الذكية والكمبيوترات اللوحية computer tap وأيضاً أجهزة الكمبيوترات المحمولة laptop وغيرها العديد من الأنظمة المدمجة.

ARM Architecture

ARM Architecture

الهيكلية ARM- CORTEX

الهيكلية CORTEX - ARM 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 كـ Linux أو Android وغيرها والتي تدعم العديد من التطبيقات، وتستخدم هذه المعمارية في أغلب الهواتف الذكية
- **Cortex-M** : والتي ترمز إلى Embedded وتتميز 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 انترنت 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

أصناف الهيكلية Cortex-M المستخدمة في بناء متحكمات STM32

أصناف الهيكلية Cortex-M المستخدمة في بناء متحكمات STM32

أصناف الهيكلية Cortex-M المستخدمة في بناء متحكمات STM32

STM32G0 المتحكم

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

STM32F3
- Arm Cortex-M4 + FPU at 72 MHz – 90 DMIPS
- Rich analog peripheral set
- High-resolution timer

Instrumentation & Measurement

Digital Power

Motor Control

Mixed-signal MCUs

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 IO 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

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 (±5%) + 16 MHz (±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 (I²S)
- 6x USART (3x with LIN, smartcard, IrDA, modem control)
- 2x LPUART
- 3x I²C 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

STM32G0B1CE المتحكم

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

STM32G0 مزايا المتحكم

- 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 128 Kbytes of Flash memory
  - 36 Kbytes of SRAM
- CRC calculation unit

STM32G0 مزايا المتحكم

- Clock management
  - 4 to 48 MHz crystal oscillator
  - 32 kHz crystal oscillator with calibration
  - Internal 16 MHz RC with PLL option (±1 %)
  - Internal 32 kHz RC oscillator (±5 %)
- Reset and power management
- Voltage range: 1.7 V to 3.6 V
- Low-power modes:
  - Sleep, Stop, Standby, Shutdown

STM32G0 مزايا المتحكم

- Up to 60 fast I/Os
  - All mappable on external interrupt vectors
- Multiple 5 V-tolerant I/Os
- 7-channel DMA controller with flexible mapping
- 12-bit, 0.4 µs ADC (up to 16 ext. channels)
- Two 12-bit DACs, low-power sample-and-hold

STM32G0 مزايا المتحكم

- Two fast low-power analog comparators
- 14 timers (two 128 MHz capable)
- Communication interfaces
- Two I2C-bus
- Four USARTs with master/slave
- One low-power UART
- Two 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 الDatasheet على المواصفات الكهربائية وتوزع الأقطاب

- تحتوي ال User Manual على المسجلات الداخلية وتفاصيل برمجة المتحكم

- المنتديات والمواقع التعليمية المختلفة

كيفية الحصول على معلومات فنية عن المتحكم؟

life.augmented

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 (PVD)

- 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

كيفية الحصول على معلومات فنية عن المتحكم؟

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)]

كيفية استخدام متحكمات STM32

هناك طريقتين أساسيتين لاستخدام متحكمات STM32:

- من خلال بناء وتصميم لوحتك الخاصة
- من خلال استخدام إحدى اللوحات التطويرية المتاحة

Boards type أنواع اللوحات التطويرية المتوفرة

Nucleo board ❑

Boards type أنواع اللوحات التطويرية المتوفرة

Discovery kit ❑

Boards type أنواع اللوحات التطويرية المتوفرة

Eval board ❑

Boards type أنواع اللوحات التطويرية المتوفرة

Blue Pill ❑

Boards type أنواع اللوحات التطويرية المتوفرة

❑ موديولات Hexabitz

تغذية متحكمات STM32

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 stm32 بين 1.7V to 3.6V
- أقل جهد يمكن تغذية المتحكم به هو 1.7V وهو العتبة الدنيا المسموحة VPDR(min)
- أكبر جهد يمكن تغذية المتحكم به هو 3.6V وهو العتبة العليا المسموحة VPoR(max)

تغذية متحكمات STM32

VDD/VDDA:

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

تغذية متحكمات STM32

VDD/VDDA:

- ويقوم بتغذية كلاً من :
- منظم الجهد الداخلي المسؤول عن تغذية المعالج
- المبدلات التشابهية والمقارنات
- Flash memory
- RTC
- المداخل والمخارج 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
    - low-speed external 32.768 kHz oscillator
    - backup registers
  - وذلك عند عدم وجود VDD

تغذية متحكمات STM32

VREF+:

- وهو عبارة عن جهد الدخل المرجعي الخاص بالمبدلات التشابهية
- يتراوح بين 2V والـ VDD عندما يكون الـ VDD>2V
- ويكون مساوٍ للـ VDD عندما يكون الـ VDD<2V

مصادر الساعة في متحكمات STM32

- لمتحكمات STM32 ثلاث مصادر مختلفة للساعة هي:

- HSE
- HSI
- PLL

مصادر الساعة في متحكمات STM32

- لمتحكمات STM32 ثلاث 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

- لمتحكمات 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 ±5% accuracy, also used to clock an independent watchdog

مصادر التصفير Reset في متحكمات STM32

لمتحكمات STM32 ثلاث مصادر للتصفير هي :

- System Reset
- Power Reset
- RTC Domain Reset

1- System Reset

- This first type of reset, resets:
  - all the registers
  - except certain registers for the Reset and Clock Controller.
  - It also does not reset the RTC domain.

1- System Reset

- The System reset sources are:
  - The external reset (generated by a low level on the NRST pin)
  - A window watchdog event
  - An independent watchdog event
  - A software reset request
  - A low-power-mode security reset (which is generated when Stop, Standby or Shutdown mode is entered
  - An option byte loader reset
  - Brown-out or Power-on reset. The reset source flag can be found in the RCC Control and Status register.

1- System Reset

- NRST_MODE selects the operation mode of the NRST pin
  - : input / output reset
  - input only reset or GPIO
  - IRHEN stands for Internal Reset Holder Enable. When this mode is enabled, the NRST pin is driven low until its voltage level goes under the voltage input low threshold.

Mode	Configuration		Behavior
	NRST_MODE	IRHEN	
Input/Output (Legacy)	11	0	20 µs output pulse generated on NRST pin in case of Internal Reset
		1	Output pulse maintained until NRST voltage reaches V_{IL} threshold (~0.3 VDD)
Input only	01	x	Internal resets are not propagated outside of the part (PU always ON)
GPIO	10	x	PF2 only, No Reset Pin

2- Power Reset

- The Brownout reset (BOR) resets all registers except those in the RTC domain powered by VBAT.
- When exiting Standby mode
- When exiting Shutdown mode

3- the RTC domain reset

- which resets the RTC registers, the Backup registers, and the RTC Domain Control Register and occurs when:
  - when the BDRST bit is set in the RTC Domain control register
  - VDD and VBAT are powered on if both supplies have previously been powered off.

Thank you for listening

متحكمات STM32

2

University of Aleppo - Hexabitz - Spring 2023

موضوعات المحاضرة:

- 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 stm32ومكتبة HAL
- برمجة أقطاب الدخل في متحكمات stm32
- التوابع المستخدمة من مكتبة HALللتحكم بالمداخل الرقمية في متحكم STM32.
- بناء تطبيق لإضاءة ليد من خلال مفتاح لحظي باستخدام متحكمات stm32ومكتبة HAL

1. أنماط عمل أقطاب المتحكم GPIO Mode

1. أنماط عمل أقطاب المتحكم GPIO Mode

2. برمجة أقطاب الخرج في متحكمات stm32

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم GPIO:

- **نمط PUSH-PULL:** فبفرض تم وصل مصعد 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 مصعد الليد ويضيء الليد، أما عند تطبيق صفر منطقي على القطب يصبح الترانزستور NMOS بحالة تشغيل on وبالتالي يتم تطبيق جهد الأرضي GND على مصعد الليد ويطفأ الليد كما هو موضح بالشكل التالي:

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم GPIO:

- نمط **PUSH-PULL:**

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم GPIO:

- نمط **PUSH-PULL:**

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم GPIO:

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

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم GPIO:

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

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم GPIO:

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

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**

نميز نمطي عمل لكل قطب خرج من أقطاب المتحكم GPIO:

- نمط **Open-Drain**

Maximum output speed: ويعني 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

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

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) الذي يمكن سحبه أو تصريفه عن طريق المتحكم بشكل كلي هو 150mA وفق المواصفات الكهربائية لعائلة متحكمات STM32.
- وقيمة 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	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink)(1)	150	
I_{IO}	Output current sunk by any I/O and control pin	25	
	Output current source by any I/Os and control pin	-25	

التوابع المستخدمة من مكتبة HAL للتحكم بالمخارج الرقمية في متحكم STM32:

- لإعطاء واحد منطقي set أو صفر منطقي reset لقطب محدد من أي منفذ من منافذ المتحكم نقوم باستخدام التابع التالي من مكتبة HAL:

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

- GPIO_TypeDef *GPIOx: اسم المنفذ المراد التحكم بأحد أقطابه على سبيل المثال GPIOA
- uint16_t GPIO_Pin: رقم القطب المراد التحكم به على سبيل المثال GPIO_PIN_5
- GPIO_PinState PinState: لإعطاء واحد منطقي نكتب GPIO_PIN_SET ولإعطاء صفر منطقي نكتب GPIO_PIN_RESET

التوابع المستخدمة من مكتبة HAL للتحكم بالمخارج الرقمية في متحكم STM32:

- مثال1: لكتابة واحد منطقي على القطب رقم10 من المنفذ Dنستخدم التابع التالي:

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

- مثال2: لكتابة صفر منطقي على القطب رقم5 من المنفذ Aنستخدم التابع التالي:

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

التوابع المستخدمة من مكتبة HAL للتحكم بالمخارج الرقمية في متحكم 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 بالميللي ثانية نستخدم التابع التالي:

```
HAL_Delay(Milliseconds)
```

التطبيق الأول: إضاءة ليد وإطفاؤه كل 0.5 sec باستخدام المكتبات STM32 ومكتبة HAL

- سنقوم بتصميم تطبيق يقوم بعمل toggle لليد المتصل بالقطب PA5

التطبيق الأول: إضاءة ليد وإطفاؤه كل 0.5sec باستخدام المعالج STM32 ومكتبة HAL

- نقوم بضبط إعدادات القطب PA5 كقطب خرج

s1_ex1_hal.ioc | main.c

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
Computing
Middleware

GPIO Mode and Configuration

Configuration

Group By Peripherals

GPIO

Search Signals

Search (Crtl+F)

Show only Modified Pins

Pin Na...	Signal on ...	GPIO out...	GPIO mode	GPIO Pull...	Maximum...	User Label	Modified
PA5	n/a	Low	Output Pu...	No pull-up...	Low		☐

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

VDD VSS PB9 PB8 BOOT0 PB7 PB6 PB5 PB4 PB3 PA15 PA14

VBAT PC13.. PC14.. PC15.. PD0-.. PD1-.. NRST VSSA VDDA PA0-.. PA1 PA2

VDD VSS PA13 PA12 PA11 PA10 PA9 PA8 PB15 PB14 PB13 PB12

STM32F103C4Tx
LQFP48

PA3 PA4 PA5 PA6 PA7 PB0 PB1 PB2 PB10 PB11 VSS VDD

GPIO_Output

التطبيق الأول: إضاءة ليد وإطفاؤه كل 0.5 sec باستخدام المكتبات STM32 ومكتبة HAL

- نقوم بضبط تردد الساعة للمتحكم

التطبيق الأول: إضاءة ليد وإطفاؤه كل 0.5 sec باستخدام المكتبات STM32 ومكتبة HAL

- نقوم بالضغط على Ctrl+s أو من Project...Generate code، 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 مكتبات STM32 ومكتبة HAL

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


}
```

التحكم 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 set/reset للمنفذ أو لأحد الأقطاب الموجودة فيه

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).

التحكم 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 التالي: 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();
```

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

```
while (1)
 {
  GPIOA->ODR = 1<<5;
      HAL_Delay(500);
      GPIOA->ODR &= ~(1<<5);
      HAL_Delay(500);
 }

}
```

برمجة أقطاب الدخل في متحكمات stm32

عند استخدام قطب المتحكم كقطب دخل رقمي يتم إلغاء تفعيل Output buffer وتفعيل الـ Input buffer والذي يتألف 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

ربط المفاتيح وكباسات اللحظية مع مداخل المتحكم

الأنماط المختلفة لقطب الدخل الرقمي

نميز ثلاث أنماط مختلفة لقطب الدخل الرقمي للمتحكم :

- نمط High impedance of Floating
- نمط الـ PULL-Up
- نمط الـ Pull-Down

❑ نمط High impedance of Floating

❑ نمط PULL-UP

Vcc
source
T1
PMOS
gate
drain
NMOS
T2
gate
source
Gnd
Vcc
R
GPIO
Pin
Input Buffer

Pin with pull-up resistor
VCC
VCC
VCC
PORT B
Digital input
MCU

- نمط PULL-DOWN

Input Buffer

برمجة أقطاب الدخل في متحكمات stm32

Table 20. Port bit configuration table

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01		0 or 1
	Open-drain		1	10		0 or 1
Alternate Function output	Push-pull	1	0	11		Don't care
	Open-drain		1	see *Table 21*		Don't care
Input	Analog	0	0	00		Don't care
	Input floating		1			Don't care
	Input pull-down	1	0			0
	Input pull-up					1

التوابع 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 GPIO_PIN_RESET في حال كانت الحالة المنطقية للقطب في حالة جهد منخفض ، ويعيد GPIO_PIN_SET في حال كانت الحالة المنطقية للقطب في حالة جهد مرتفع.

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

HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_13);

التطبيق الأول: إضاءة ليد من خلال مفتاح لحظي باستخدام متحكمات STM32ومكتبة HAL

- بناء تطبيق لإضاءة ليد من خلال مفتاح لحظي باستخدام متحكمات stm32ومكتبة HAL

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

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

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

- نقوم بضبط تردد الساعة للمتحكم

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

- نقوم بالضغط على Ctrl+s أو من Project...Generate code، 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 STM32 ومكتبة HAL

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

للتحكم بالمداخل الرقمية للمتحكم STM32 دون استخدام مكتبة HAL:

- هناك مجموعة من المسجلات تستخدم للتحكم بالمداخل الرقمية لمتحكم STM32سنكتفي فقط بذكر المسجل المسؤول عن قراءة حالة المنفذ أو أحد الأقطاب الموجودة فيه ويدعى Input data register (IDR)وله 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.

للتحكم بالمخارج الرقمية للمتحكم STM32 دون استخدام مكتبة HAL:

- والمسجل IDR هو مسجل للقراءة فقط ، البتات 16:31 غير مستخدمين، أما البتات 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)))

التطبيق الثاني: إضاءة ليد عن طريق مفتاح لحظي 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();
```

التطبيق الثاني: إضاءة ليد عن طريق مفتاح لحظي دون استخدام مكتبة HAL

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


     }}
```

Thank you for listening

متحكمات STM32

3

University of Aleppo - Hexabitz - Spring 2023

موضوعات المحاضرة:

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

نظرة عامة عن مفهوم الاستثناءات: Exceptions Overview

- المقاطعة هي حدث غير متزامن يتسبب في إيقاف تنفيذ الكود الحالي اعتمادا على مبدأ الأولوية
- تنشأ المقاطعات من خلال الـ hardware أومن البرنامج نفسه الـ software
- يتم التحكم بالمقاطعات المتحكم NVIC(Nested Vectored Interrupt Controller)
- توفر معالجات ARM هيكلية 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 وتخفيف 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 عندها 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-Mبالإضافة إلى 240 مقاطعة ، لذا فإن معالجات CORTEX-M تدعم 255 استثناء، بحيث:

- الاستثناء رقم واحد هو للـ 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

- RESETهو 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 STM32يتم ربط استثناءات الـ NMIمع نظام حماية الساعة Clock security system(CSS)، حيث الـ CSSهي وحدة طرفية تقوم بفحص حالة الساعة الخارجية HSEوفي حال اكتشاف مشكلة ما فيها تقوم بتعطيل HSEوتفعيل الساعة الداخلية HSI.

System Exceptions

- Hard Fault يتم توليد هذا الاستثناء عند حدوث استثناء ما وفي حال عدم وجود آلية للتعامل مع هذا الاستثناء، على سبيل المثال عند حدوث استثناء معين وفي حال عدم تفعيل Exception Handlerالخاص به يتم في هذه الحالة توليد استثناء الـ Hard Fault.
- Memory management faultيحدث هذا الاستثناء عند محاولة الوصول لموقع غير مصرح بالوصول له في الذاكرة أو عندما يتم انتهاك قاعدة من قواعد حماية الذاكرة أثناء تنفيذ التعليمات البرمجية.
- BUS Fault يحدث هذا الاستثناء عند حدوث خطأ أثناء الوصول إلى ذاكرة المعطيات أو التعليمات وقد يكون بسبب حدوث خطأ على الناقل Busفي النظام الذاكري.

System Exceptions

- Usage Fault يحدث هذا الاستثناء أثناء تنفيذ التعليمات وقد يكون بسبب استخدام تعليمة غير موجودة أو عند محاولة الوصول لموقع غير مصرح به أو بسبب خطأ في الحالة الناتجة عن تنفيذ التعليمة مثلاً الحصول على قيمة غير صالحة مثل القسمة على صفر.
- System Service Call)SVC Call يتم توليد هذا الاستثناء عند استدعاء تعليمة Supervisor Callوهذه التعليمة تستخدم من قبل نظام الزمن الحقيقي Real Time Operation (RTC).
- Debug Monitor يتم توليد هذا الاستثناء عند استخدام ميزة اكتشاف الأخطاء البرمجية وعندها يكون المعالج في نمط Monitor Debug Mode
- PendSV وهو استثناء آخر يتعلق بالـ RTOS

System Exceptions

- System tick هذا الاستثناء يستخدم دائماً في أنظمة الزمن الحقيقي RTOS، حيث يحتاج كل نظام RTOS إلى مؤقت لجدولة أنشطة النظام بحيث يقوم بتوليد مقاطعة في كل فترة زمنية لترك تنفيذ المهمة الحالية وتنفيذ مهمة أخرى ويكون هذا المؤقت متصل داخلياً بنواة CORTEX-M وليس وحدة خارجية كباقي الطرفيات.
- Interrupt (IRQ) هي عبارة عن الاستثناءات التي يتم توليدها من قبل الطرفيات أو من قبل البرنامج

NVIC (Nested Vectored Interrupt Controller) متحكم

- هو المتحكم المسؤول عن تحديد أولويات المقاطعات Interrupt (IRQ)هي عبارة عن الاستثناءات التي يتم توليدها من قبل الطرفيات أو من قبل البرنامج
- وتحسين أداء المعالج MCUوتقليل زمن استجابة المقاطعة
- يوفر NVIC أيضًا خطوات محددة للتعامل مع المقاطعات التي تحدث أثناء تنفيذ مقاطعات أخرى أو عندما يكون المعالج في مرحلة استعادة حالته السابقة واستئناف عمليته المعلقة التي توقف عندها بسبب حدوث المقاطعة

NVIC (Nested Vectored Interrupt Controller) متحكم

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

NVIC (Nested Vectored Interrupt Controller) متحكم

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

NVIC (Nested Vectored Interrupt Controller) متحكم

- 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

interrupts table Vectors جدول أشعة المقاطعة

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

External Interrupts المقاطعات الخارجية

- يحتوي متحكم STM32 على 16 خط للمقاطعات الخارجية هي من الخط 0 حتى الخط 15
- تشير أرقام الخطوط إلى أرقام اطراف الـ GPIOS
- هذا يعني أن الطرف PA0 متصل بالخط LINE0 والطرف PA13 متصل بالطرف LINE13، أيضاً PB0 و PC0 متصلين بالخط LINE0، بمعنى أن جميع الأطراف ذات الأرقام PX0 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

يجب الانتباه للملاحظات التالية:

- الأطراف PA0،PB0، ...PC0جميعها متصلة بالخط LINE0 لذا في اللحظة الواحدة بإمكانك توليد مقاطعة على طرف واحد فقط من هذه الأطراف.
- الأطراف PA0، PA5 متصلين على خطين مختلفين من خطوط المقاطعة لذا يمكنك استخدامهم في نفس اللحظة لتوليد المقاطعة.

External Interrupts المقاطعات الخارجية

- يمكن قدح المقاطعة عند الجبهة الصاعدة rising edge أو الهابطة falling edge أو عند كليهما

External Interrupts المقاطعات الخارجية

- فتحدث المقاطعة الخارجية وفقاً للآلية التالية:

أنماط عمل المعالج Processor mode

للمعالج نمطي عمل أساسين:

- Thread Mode : يكون المعالج في نمط Thread Modeفي الوضع الطبيعي له عند تنفيذ الكود أو عند عمل resetله
- 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 flagsبشكل آلي وعن طريق الـ hardwareليضمن استجابة سريعة لحدث المقاطعة

مصطلح Preemption

- مصطلح Preemption يعني مقاطعة عمل task أو مهمة معينة بسبب حدوث task أو مهمة أخرى لها أولوية أعلى في التنفيذ وفي هذه الحالة فإن الـ task التي تمت مقاطعتها تسمى preempted
- فعند حدوث عدة مقاطعات أو استثناءات يتم تعليقها جميعاً وتنفيذ الاستثناء أو المقاطعة ذات الرقم الأصغر في جدول أشعة المقاطعات VIT وأثناء تنفيذ هذه الاستثناء / المقاطعة من قبل المعالج وفي حال حدوث استثناء/ مقاطعة ذات أولوية أعلى يتم عمل Preemption للمقاطعة الحالية والذهاب لتنفيذ المقاطعة ذات الأولوية الأعلى، حيث المقاطعة ذات الرقم الأصغر في جدول أشعة المقاطعات لها أولوية أعلى في التنفيذ من قبل المعالج

Preemption مصطلح

مصطلح Preemption

أصبح لدينا مصطلحين :

- Preemption : ويعني مقاطعة استثناء يتم حالياً تنفيذه من قبل المعالج عند حدوث استثناء ذو مستوى أولوية أعلى
- Pending: ويعني تعليق استثناء حدث ولكن لم يتم البدء بتنفيذه ، لحين تنفيذ الاستثناءات ذات الأولوية الأعلى

Preemption مصطلح

- IRQ1 - Group priority 0
- IRQ2 - Group priority 1

Preemption مصطلح

- IRQ1 - Group priority 0, Sub priority 0
- IRQ2 - Group priority 0, Sub priority 1

Interrupt Lifecycle دورة حياة المقاطعة

يمكن للمقاطعة أن تكون:

- مفعلة enabled أو غير مفعلة disabled(وهو الوضع الافتراضي)
- معلقة (على قائمة الانتظار) Pendingأو غير معلقة not pending
- نشطة active أي يتم تنفيذها حالياً أو غير نشطة not active.

Interrupt Lifecycle دورة حياة المقاطعة

- عند حدوث مقاطعة يتم تعليقها لحين يصبح المعالج قادر على خدمتها وفي حال عدم وجود مقاطعات أخرى يتم حالياً تنفيذها، يتم تصفير حالة التعليق pending state تلقائياً من قبل المعالج ومن ثم يبدأ بخدمتها

Interrupt Lifecycle دورة حياة المقاطعة

- المقاطعات ذات الأولوية الأدنى عليها الانتظار لحين عدم وجود مقاطعات بأولوية أعلى منها ، وعند البدء بتنفيذها يتم وضعها في حالة (preempted) inactive عند حدوث مقاطعة لها أولوية أعلى

Interrupt Lifecycle دورة حياة المقاطعة

- يمكن قدح المقاطعة مجدداً أثناء تنفيذها من خلال تفعيل بت الانتظار pending bit الخاص بها ، كما يمكن إلغاؤها من خلال مسح بت الانتظار الخاص بها كما هو بالأشكال التالية

Exception Behavior

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

- يقوم المعالج بإنهاء التعليمة الحالية التي يقوم بتنفيذها طالما أنها لا تحتاج لأكثر من دورة تعليمة
- يتم حفظ حالة المعالج (عنوان التعليمة التي وصل لعندها 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 cyclesفي معالجات
- يقوم برنامج خدمة المقاطعة بتصفير علم المقاطعة التي قامت باستدعائه

Exception Behavior

- في نهاية برنامج خدمة المقاطعة يتم التأكد 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 تعليمة 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 الحرجة

- زر طوارئ لتفعيل برنامج للطوارئ 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 PA0:PA6، والأقطاب PB0:PB1 كأقطاب خرج ، والقطب PC13كقطب دخل مع تفعيل مقاومة الرفع الداخلية، والقطب PC14كقطب مقاطعة خارجية

*s3_ex1.ioc | main.c | stm32f1xx_hal_gpio.c

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

GPIO Mode and Configuration

Configuration

Group By Peripherals

GPIO | NVIC

Search Signals

Search (Crtl+F)

Show only Modified Pins

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		☑
PC14-OSC32_IN	n/a	n/a	External Interrup...	Pull-up	n/a	int_button	☑

PC13-TAMPER-RTC Configuration :

GPIO mode: Input mode

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

User Label:

Pinout view | System view

STM32F103C4Tx LQFP48

VBAT, PC13.. (GPIO_Input), PC14.. (int_button), PC15.., PD0-.., PD1-.., NRST, VSSA, VDDA, PA0-.. (GPIO_Output), PA1 (GPIO_Output), PA2 (GPIO_Output), PA3, PA4, PA5, PA6, PA7, PB0, PB1, PB2, PB10, PB11, VSS, VDD

التطبيق الأول: برمجة زر للطوارئ لاستخدامه في الحالات الحرجة

- قم بفتح NVIC Tab ثم قم بتفعيل المقاطعات الخارجية، يمكنك أيضاً إعادة ضبط مستويات الأولوية للمقاطعات:

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

برنامج خدمة المقاطعة

```
#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);
    }}
```

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

متحكمات STM32 4

University of Aleppo - Hexabitz - Spring 2023

موضوعات المحاضرة:

- الأنواع المختلفة للمؤقتات
- مؤقتات الأغراض العامة
- أنماط العمل المختلفة للمؤقتات في متحكمات STM32
- ساعة الزمن الحقيقي Real Time Clock (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 حدٍ سواء
- تسمح خاصية الـ prescaler أو المقسم الترددي بتقسيم تردد الساعة على عدد معين يتم اختياره بين الـ 0 و 65535

الأنواع المختلفة للمؤقتات

الأنواع المختلفة للمؤقتات

- المؤقتات الأساسية
- مؤقتات الطاقة المنخفضة
- مؤقتات الأغراض العامة
- المؤقتات المتقدمة
- المؤقتات عالية الدقة

General Purpose Timers: مؤقتات الأغراض العامة

- المؤقتات في هذه المجموعة تكون إما 16 أو 32 بت(بناءً على عائلة STM32)،
- يمتلك عداد تصاعدي/ تنازلي بطول 16 بت قابل لإعادة التحميل تلقائياً auto-reload counter
- مقسم جهد بطول 16 بت يستخدم لتقسيم تردد الساعة للمتحكم على أي عدد يتراوح بين 1 و 65535
- له أربع مداخل/ مخارج منطقية قابلة للبرمجة بأحد الوظائف التالية:

  - Input Capture
  - Output Compare
  - توليد نبضات PWM
  - One-Pulse mode Output

أنماط العمل المختلفة للمؤقتات في متحكمات STM32

نمط المؤقت Timer

- عند عمل المؤقت بنمط Timer، فإن المسجل TCNT تزداد قيمته 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 0xffff والمؤقت ذو 32 بت يمكنه العد إلى 0xffff ffff

نمط المؤقت Timer

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

- عندما يصل العداد إلى القيمة المحددة Period (preload) يحدث ما يسمى بالطفحان overflowأي يقوم بالتصفير والعد مرة أخرى من الصفر و يتم رفع العلم الخاص بالطفحان Update Event(UEV)

نمط المؤقت Timer

- ولحساب الزمن الذي سيطفح عنده المؤقت نستخدم المعادلة التالية:

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

- على سبيل المثال ، لنفترض أن تردد الساعة للمتحكم مضبوط على 48 MHzوقيمة الـ Prescalerتساوي 48000 والـ period تساوي 500 سيحدث overflowللمؤقت كل:

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

نمط العداد Counter

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

نمط العداد Counter

ويمكنه أن يعمل بثلاث أنماط مختلفة هي:

- نمط العد التصاعدي Up-counting Mode

في هذا النمط فإن العداد يبدأ بالعد من الصفر مع كل نبضة قادمة على قطب الدخل ويستمر حتى يصل إلى القيمة المخزنة مسبقاً والموجودة في المسجل (TIMx_ARR) ، ثم يعود للقيمة صفر ويولد حدث الطفحان Overflow كمل يتم توليد حدث Update eventمع كل طفحان للعداد

Counter نمط العداد

- نمط العد التنازلي Down-counting Mode

في هذا النمط فإن العداد يبدأ بالعد من القيمة المخزنة auto-reload value في المسجل TIMx-ARR مع كل نبضة قادمة على قطب الدخل ويستمر ليصل إلى الصفر ثم يعو ليبدأ من القيمة المخزنة سابقاً ويولد حدث الـ Underflow event أيضاً يتم توليد حدث الـ Update مع كل Underflow

نمط العداد Counter

- **نمط العد 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 الطفحان Overflow event 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 الـ Underflow ثم يعود للعد التصاعدي من الصفر وهكذا...،

Counter نمط العداد

Up counting

Down counting

Center Aligned

نمط العداد Counter

- ويكون المخطط الزمني لأنماط العد الثلاث :

Counter Modes (ARR=3, PSC=1)

Input Capture mode نمط

- في نمط Input Capture يستقبل المؤقت نبضات الساعة الخاصة به من مصدر داخلي (ساعة المتحكم بعد استخدام المقسم الترددي)
- يستمر 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 المتحكم يمكن معرفتها من خلال الـ datasheet الخاصة بالمتحكم

Input Capture mode نمط

PWM mode نمط

- في نمط 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 الترددي للمؤقت)
- يتم وضع قطب الخرج الخاص بالـ PWMفي وضع HIGHويبقى كذلك إلى أن يصل العداد إلى القيمة المخزنة في المسجل CCRx عندها يصبح قطب الخرج في وضع LOWإلى أن يصل العداد إلى القيمة المخزنة في المسجل ARRx، وهكذا ....
- يدعى شكل الإشارة الناتجة بالـ PWM(Pulse Width Modulation)، حيث يتم التحكم بالتردد من خلال تردد الساعة الداخلية للنظام، والمقسم الترددي Prescalerبالإضافة إلى قيمة المسجل ARRx(Auto Reload register)، كما يتم تحديد قيمة دورة التشغيل الـ duty cycleمن خلال قيمة المسجل الـ CCR1

PWM mode نمط

- يوضح المخطط التالي كيفية تأثير قيمة المسجل ARRفي دور(تردد) إشارة الـ PWM، وكيف تؤثر قيمة المسجل CCR1في قيمة دورة التشغيل duty cycle

PWM mode نمط

- 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

PWM mode نمط

تردد إشارة الـ PWM:

- يتم التحكم بدور إشارة الـ PWM (1/FPWM) من خلال البارامترات التالية:
- قيمة المسجل ARR
- قيمة المقسم الترددي Prescaler
- تردد الساعة الداخلية internal clock
- عدد مرات التكرار

وذلك من خلال العلاقة التالية:

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

PWM mode نمط

مثال:

بفرض أن=72 F_CLK MHZ، Prescaler=1، ARR=65535، RCR =0، احسب تردد نبضات الـ PWM:

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

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:

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

PWM mode نمط

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 PWMلها نفس التردد ولكن بدورات تشغيل مختلفة، وهذه الإشارات جميعها متزامنة باعتبار أن الجبهة الهابطة هي نفسها لجميع الإشارات.

PWM mode نمط

أنماط الـ PWM المختلفة:

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

PWM mode نمط

Edge-aligned Mode

Center-aligned Mode

PWM mode 2

أوضاع الاستخدام المختلفة للمؤقتات في متحكمات STM32

هناك ثلاث 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 تتسبب 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 الـInterrupt : في هذا الوضع عند الوصول إلى Overflow/underflow أو أي من أحداث المقاطعة سيتم التوجه آلياً لتنفيذ برنامج خدمة المقاطعة، وهذا الوضع الذي سنستخدمه في جميع التطبيقات القادمة.

توفر مكتبة HALالدالة التالية لبدء المؤقت في وضع المقاطعة:

```
HAL_TIM_Base_Start();
```

- وضع الـ DMA:

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

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

- نختار القطب PA5 لضبطه كقطب خرج
- نقوم بضبط إعدادات المؤقت كي نحصل على زمن 100 msec لعكس حالة اليد الموصول على القطب رقم 5 من المنفذ A، من المعادلة السابقة سنفترض أن تردد ساعة المتحكم هي 8 MHz والمقسم الترددي 8000 بقي فقط حساب(Period) Preload، بتعويض 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 الترددي 8000 ، الـ Preload=100، أيضاً سنقوم بتفعيل إعادة التحميل التلقائي، كما في الشكل التالي:

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

- نقوم بتفعيل مقاطعة المؤقت من شريط الـ NVIC tab

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

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

سنتبع في هذا التطبيق الخطوات التالية للتحكم بشدة إضاءة الليد:

- ضبط بارامترات 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 ، والمقسم الترددي prescaler على 1، فيصبح التردد 61HZ من خلال العلاقة:

$$F_PWM=(8\times(10^6))/((65535+1)\times(1+1)\times1)=61HZ$$

- التحكم بدورة التشغيل dutycycle 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 الليد

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

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

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core
Analog
Timers
RTC
TIM1
TIM2
TIM3
Connectivity
Computing
Middleware

TIM2 Mode and Configuration

Mode

Slave Mode: Disable
Trigger Source: Disable
Clock Source: Internal Clock
Channel1: PWM Generation CH1
Channel2: Disable
Channel3: Disable
Channel4: Disable
Combined Channels: Disable
Use ETR as Clearing Source
XOR activation
One Pulse Mode

Configuration

Reset Configuration

NVIC Settings | DMA Settings | GPIO Settings
Parameter Settings | User Constants

Configure the below parameters :

Search (Crtl+F)

Counter Settings
- Prescaler (PSC - 16 bits val... 1
- Counter Mode: Up
- Counter Period (AutoReload... 65535
- Internal Clock Division (CKD): No Division
- auto-reload preload: Enable

Trigger Output (TRGO) Parameters
- Master/Slave Mode (MSM bit): Disable (Trigger input effect not delayed)
- Trigger Event Selection: Reset (UG bit from TIMx_EGR)

PWM Generation Channel 1

Pinout view | System view

VDD VSS PB9 PB8 BOO PB7 PB6 PB5 PB4 PB3 PA15 PA14

VBAT PC13.. PC14.. PC15.. PD0-.. PD1-.. NRST VSSA VDDA TIM2_CH1 PA0-.. PA1 PA2

VDD VSS PA13 PA12 PA11 PA10 PA9 PA8 PB15 PB14 PB13 PB12

PA3 PA4 PA5 PA6 PA7 PB0 PB1 PB2 PB10 PB11 VSS VDD

STM32F103C4Tx
LQFP48

Real Time Clock (RTC) ساعة الزمن الحقيقي

- 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 عداد 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 لاتحتوي على RTC داخلية وإنما يتم استخدام إحدى شرائح الـ RTC الخارجية كـ DS1302 أو PCF8563 بالإضافة إلى بعض العناصر الالكترونية اللازمة كي تعمل بشكل أمثلي كما تحتاج إلى مساحة إضافية على الدارة المطبوعة

Real Time Clock (RTC) ساعة الزمن الحقيقي

- تحتوي متحكمات stm32 على موديول RTC مدمج بداخل المتحكم وهي لا تحتاج لأية عناصر إضافية أو دارات ملائمة كي تعمل

لوحدة الـ RTC مصدري ساعة هما:

- RTCCLK:ويستخدم كمصدر ساعة للـ RTC Timer/counter
- APB clock:ويستخدم كمصدر ساعة للـ RTC register من أجل عمليات القراءة والكتابة على المسجلات

Real Time Clock (RTC) ساعة الزمن الحقيقي

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

- High Speed External clock (HSE): وفي هذه الحالة تكون مقسمة على 32
- Low Speed External clock (LSE)
- Low Speed Internal Clock (LSI)
- لكن عندما يكون المتحكم يعمل في نمط VBAT أو يكون في حالة إيقاف تشغيل shutdown ، في هذه الحالة يجب أن يكون RTC clock هي LSE أو LSI

Real Time Clock (RTC) ساعة الزمن الحقيقي

- يتم تقسيم تردد RTC clock باستخدام مقسم ترددي 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 ترددي آخر قابل للضبط وبطول 15bit، ويجب أن يكون التردد الناتج عنه 1HZ كي يتم تحديث الزمن والتاريخ في BCD registers كل 1sec

Real Time Clock المخطط الصندوقي لساعة الزمن الحقيقي (RTC)

Real Time Clock (RTC) ساعة الزمن الحقيقي

- يمكن لوحدة الـ 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 ستستمر وحدة الـ RTC بالعمل حتى عند فصل التغذية الأساسية عن المتحكم ، عندما يكون قطب VBAT موصول ببطارية احتياطية

Real Time Clock (RTC) ساعة الزمن الحقيقي

- استجرار وحدة الـ RTC للتيار فقط 300nA عند تغذيتها بجهد 1.8v وهذا التيار يتضمن استجرار الـ LSI للتيار
- يكون التقويم الناتج عن وحدة الـ RTC مشفر بشفرة الـ 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 إنقاص ساعة من التقويم لإدارة التوقيت الصيفي

Real Time Clock (RTC) ساعة الزمن الحقيقي

- لوحدة الـ RTC مخرجان لهما القدرة على توليد تنبيهان قابلان للبرمجة ولهما 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 و ضبط المنبه على وقت محدد لتشغيل ليد

ضبط إعدادات وحدة الـ RTC وتفعيل المقاطعة الخاصة بها

- حيث سنقوم بضبط التاريخ والساعة والتنبيه من خلال الكود

*rtc_simulation.ioc | main.c | main.c | lcd_txt.h | stm32f1xx_hal_rtc.c

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core: DMA, GPIO, IWDG, NVIC, RCC, SYS, WWDG

Analog

Timers: RTC, TIM1, TIM2, TIM3

Connectivity

Computing

Middleware

RTC Mode and Configuration

Mode

- [x] Activate Clock Source
- [x] Activate Calendar

RTC OUT: Disable

- [ ] Tamper

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
RTC global interrupt	☑	0	0
RTC alarm interrupt through EXTI line 17	☑	0	0

Pinout view | System view

STM32F103C4Tx
LQFP48

PC14.., PC15.., PD0-.., PD1-.., NRST, VSSA, VDDA, PA0-.., PA1, PA2

PA13, PA12, PA11, PA10, PA9, PA8, PB15 (GPIO_Output), PB14 (GPIO_Output), PB13 (GPIO_Output), PB12 (GPIO_Output)

PA3, PA4, PA5 (GPIO_Output), PA6, PA7, PB0, PB1, PB2, PB10 (GPIO_Output), PB11 (GPIO_Output), VSS, VDD

Thank you for listening

متحكمات STM32

5

University of Aleppo - Hexabitz - Spring 2023

موضوعات المحاضرة:

- مقارنة بين بروتوكولات الاتصال التسلسلي الشائعة
- مفهوم UART وUSART
- أنماط العمل المختلفة للـ UARTفي متحكمات STM32
- دوال مكتبة HALالمستخدمة للتعامل مع منفذ الاتصال التسلسلي في نمط الـ Polling
- تطبيق عملي لاستخدام المنفذ التسلسلي USARTمن خلال نمط الـ polling
- إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt
- دوال مكتبة HALالمستخدمة للتعامل مع منفذ الاتصال التسلسلي في نمط الـ interrupt
- تطبيق عملي لاستخدام المنفذ التسلسلي USARTمن خلال نمط الـ interrupt

مقارنة بين بروتوكولات الاتصال التسلسلي 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 والتفرعي:

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

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

النافذة التسلسلية USART/UART

- كل متحكم STM32 يحتوي على الأقل على وحدة طرفية UART واحدة، وأغلب متحكمات STM32توفر على الأقل اثنتين من UART/USART، وأخرى توفر لحد 8 وحدات طرفية UART/USART
- Synchronous:هو الإرسال والاستقبال المتزامن المبني على وجود clockبين المرسل والمستقبل .
- Asynchronous: لا يعتمد على clock وإنما يتم الاكتفاء بإرسال البيانات على خط الإرسال ويتم استقبالها على خط الاستقبال.

Unleash your Creativity!

USART/UART النافذة التسلسلية

Unleash your Creativity!

USART

Universal Synchronous and Asynchronous serial Receiver and Transmitter

النافذة التسلسلية USART

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

Unleash your Creativity!

(Universal Asynchronous Receiver and Transmitter Interface)

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

بنية إطار البيانات (UART Frame Format)

النافذة التسلسلية UART

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

النافذة التسلسلية UART

- تمثل الحالة الخاملة 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 مايسمى 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 إنهاء الإرسال بـ Stop Bit

أنماط العمل المختلفة للـ UARTفي متحكمات STM32

- نمطPolling Mode: يسمى أيضاً Blocking 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 Hardwareويستخدم عندما تكون كمية البيانات المتبادلة ليست كبيرة نسبياً ولا تمثل أهمية عالية من ناحية المعالجة
- نمط المقاطعةInterrupt Mode: ويسمى أيضاً non-Blocking Mode، في هذا النمط لا يتم الانتظار وتفقد البيانات من حين لآخر للتأكد من عملية الارسال 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

أنماط العمل المختلفة للـ UART في متحكمات STM32

- نمطDMA : 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 DMAيقوم بالوصول إلى الذاكرة RAMبدون احتياج أي جهد من المعالج لعمل ذلك، وبدون نمط الـ DMAلا يمكن التعامل مع السرعات العالية في الـ UART

مفاهيم أساسية في الاتصالات التسلسلية:

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

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

Unleash your Creativity!

مفاهيم أساسية في الاتصالات التسلسلية:

Unleash your Creativity!

هناك نمطين للبيانات في الاتصالات التسلسلية وهما:

1) نمط الآسكي (Ascii Mode):

123 > 3-byte

2) النمط الثنائي (BIN Mode):

123 > 1-Byte

مفاهيم أساسية في الاتصالات التسلسلية:

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 التسلسلية:

- **WordLength:** وتعني عدد البتات التي يتم إرسالها أو استقبالها في Frame في المرة الواحدة، وتوفر 3 قيم يمكن الاختيار بينها 7bit,8bit,9bit حيث لا يتضمن هذا الرقم البتات الخاصة بـ Startو الـ Stopوغيرها
- **StopBits:** يحدد عدد البتات الخاصة بالـ Stopالتي سيتم إرسالها، ويمكن الاختيار بين 1 و 2 أي بت واحد أو 2 bitفي نهاية الإشارة.

مفاهيم أساسية في الاتصالات التسلسلية:

- 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 MSBبحيث لو تم استخدام Word Length بـ 8- bitيكون مكانه في البت الثامن، أما لو تم استخدام 9- bitيكون مكانه هو في البت التاسع، ولها نمطين:
- فردي Odd: وتكون قيمة بت الـ Parity مساوٍ للواحد المنطقي عندما يكون عدد الواحدات الموجودة في الكلمة المراد إرسالها زوجي، وصفر منطقي في حال كان عدد الواحدات الموجودة في الكلمة المراد إرسالها فردي.
- زوجي Even: وتكون قيمة بت الـ Parityمساوٍ للواحد المنطقي عندما يكون عدد الواحدات الموجودة في الكلمة المراد إرسالها فردي، وصفر منطقي في حال كان عدد الواحدات الموجودة في الكلمة المراد إرسالها زوجي.

مفاهيم أساسية في الاتصالات التسلسلية:

على سبيل المثال:

- عندما تريد إرسال أي بيانات يتم تحويلها للـ Binary فمثلاً إذا كنا نريد إرسال الكلمة التالية 0b01101110 فمن خلال الـ Parityيتم حساب عدد الواحدات الموجودة ضمن هذه الكلمة المراد إرسالها وهي في هذه الحالة 5، ففي حال كنت تستخدم نمط الفردي ستكون قيمة الـ Parityصفر، أما في حال كنت تستخدم نمط الزوجي ستكون قيمة الـ Parityواحد.
- يتم إرسال قيمة البت الخاص بالـ Parityمن 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 للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

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

- دالة الإرسال:

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

حيث:

- huart : وهو مؤشر يشير إلى Struct_UART_HandleTypeDef أي المنفذ التسلسلي المستخدم للاتصال مثلاً قد يكون huart1 & أو huart2 & أو huart3 & أو huart4&

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

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

- 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 وليكن اسمها Data وتكون معرفة بالشكل التالي:

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

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

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

- 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 للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما للإرسال 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 بالميللي ثانية حتى يتم اكتمال عملية الإرسال، فإذا تم انتهاء هذا الزمن ولم تتم عملية الإرسال سيتم قطع عملية الإرسال وتقوم الدالة بإرجاع HAL_TIMOUT ماعدا ذلك يتم ارجاع HAL_OK، ويمكن استخدام الدالة HAL_MAX_DELAY مكان Timeout وهي ووظيفتها انتظار أقصى زمن ممكن لعملية الإرسال

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

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

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

حيث:

- huart : وهو مؤشر يشير إلى Struct_UART_HandleTypeDef أي المنفذ التسلسلي المستخدم للاتصال مثلاً قد يكون huart1 & أوhuart2 & أوhuart3 & أوhuart4&

دوال مكتبة HAL المستخدمة للتعامل 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:

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

Uint8_t Data[10];

دوال مكتبة HAL المستخدمة للتعامل 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 pDataوهي في المثال التالي 10.
- Timeout:أقصى زمن يتم انتظاره بالميللي ثانية حتى يتم اكتمال عملية الاستقبال، فإذا تم انتهاء هذا الزمن ولم تتم عملية الاستقبال سيتم قطع عملية الاستقبال وتقوم الدالة بإرجاع HAL_TIMOUTماعدا ذلك يتم ارجاع HAL_OK، ويمكن استخدام الدالة HAL_MAX_DELAYووظيفتها انتظار أقصى زمن ممكن لعملية الاستقبال.

دوال مكتبة HAL المستخدمة للتعامل 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 STM32 مقاطعات لـ UART 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 in multi buffer communication	NF or ORE or FE	EIE

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

يمكن تقسيم المقاطعات IRQs الخاصة بالمنفذ التسلسلي UART لمجموعتين:

- IRQs : التي يتم استدعائها أثناء الإرسال:
  - اكتمال الارسال Transmission complete
  - Clear to send(CTS)
  - مسجل البيانات فارغ transmission Data Register Empty
  - Noise Flag
  - Framing error

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

- IRQs : التي يتم استدعائها أثناء الاستقبال:
  - Idle line detection
  - Overrun error
  - مسجل الاستقبال غير فارغ 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 المقاطعة Interrupt event لكل نوع من خلال Enable Control Bit الخاص به كما في الجدول السابق، حيث كل هذه الـ IRQs لها فقط خط مقاطعة وحيد لكل USART Peripheral

- باعتبار أن لكل وحدة USART في متحكمات STM32 خط مقاطعة وحيد لذا يتوجب على المستخدم تحليل علم المقاطعة Flag Event الذي تم رفعه لمعرفة المقاطعة 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

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

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

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

- 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 مقارنةً بالدالة المستخدمة في نمط الـ Polling، لأنه لم يعد هناك زمن انتظار في نمط المقاطعة.

دوال مكتبة HALالمستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ interrupt

- دالة الاستقبال: إذا أردنا استقبال بيانات على UARTباستخدام وضع 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 متحول على النافذة التسلسلية

- نقوم بضبط إعدادات المنفذ التسلسلي:

*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

NVIC Settings | DMA Settings | GPIO Settings | Parameter Settings | User Constants

Configure the below parameters :

Search (Crtl+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
Rxfifo 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

Pinout view | System view

STM32G071RBTx
LQFP64

SYS_WKUP2, RCC_OSC32_IN, RCC_OSC32_OUT, MCO, TCK, TMS, USART2_TX[STLK_TX], USART2_RX[STLK_RX], LED_GREEN

التطبيق العملي1 : استخدام المنفذ التسلسلي UART2 في نمط الـ Polling لطباعة قيمة متحول على النافذة التسلسلية

- 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 Polling لطباعة قيمة متحول على النافذة التسلسلية

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

التطبيق العملي1 : استخدام المنفذ التسلسلي UART2 في نمط الـ Polling لطباعة قيمة متحول على النافذة التسلسلية

```
  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 العملي2 : استخدام المنفذ التسلسلي UART2 في نمط الـ interrupt لطباعة قيمة متحول على النافذة التسلسلية

- نقوم بضبط إعدادات المنفذ التسلسلي:

*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

NVIC Settings | DMA Settings | GPIO Settings | Parameter Settings | User Constants

Configure the below parameters :

Search (Crtl+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
Rxfifo 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

Pinout view | System view

SYS_WKUP2, RCC_OSC32_IN, RCC_OSC32_OUT, MCO, TCK, TMS

STM32G071RBTx

LQFP64

USART2_TX[STLK_TX], USART2_RX[STLK_RX], LED_GREEN

التطبيق العملي2 : استخدام المنفذ التسلسلي UART2 في نمط الـ interrupt لطباعة قيمة متحول على النافذة التسلسلية

- نقوم بضبط إعدادات مقاطعة المنفذ التسلسلي:

*UART_interrupt.ioc

Pinout & Configuration | Clock Configuration | Software Packs

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 >

USART2 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

NVIC Interrupt Table	Enabled	Preemption Priority
USART2 global interrupt / USART2 wake-up interrupt thr...	☑	0

التطبيق العملي1 : استخدام المنفذ التسلسلي UART2 في نمط الـ interrupt لطباعة قيمة متحول على النافذة التسلسلية

- 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 العملي1 : استخدام المنفذ التسلسلي UART2 في نمط الـ interrupt لطباعة قيمة متحول على النافذة التسلسلية

```
 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 العملي1 : استخدام المنفذ التسلسلي UART2 في نمط الـ interrupt لطباعة قيمة متحول على النافذة التسلسلية

```
 while (1)
{
} }
 Void
 HAL_UART_RxCpltCallback(UART_HandleTypeDe
 *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

متحكمات STM32

6

University of Aleppo - Hexabitz - Spring 2023

موضوعات المحاضرة:

- المبدلات التشابهية الرقمية في متحكمات STM32
- أنماط عمليات التحويل الرقمي ADC conversion 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 قدح 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) لـه المواصفات التالي:

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

المبدلات التشابهية الرقمية في متحكمات STM32
ADC in STM32

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

- لكل مبدل تشابهي رقمي ثلاث مراقبات تشابهية analog watchdogs لمراقبة حالات الـ thresholds Oversampler
- القدرة على توليد طلبات DMA
- القدرة على توليد مقاطعة interrupt
- سحب منخفض للتيار uA@ Msamples/s1181

المبدلات التشابهية الرقمية في متحكمات STM32
ADC in STM32

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

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

المبدلات التشابهية الرقمية في متحكمات STM32
ADC in STM32

يوضح الجدول التالي أهم الخصائص للـ 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 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:

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 قدحه عبر الكود أو من خلال الهاردوير ، وذلك عند تفعيل نمط Auto-off mode، ويتم إضافة زمن البدء startup time تلقائياً بين لحظة قدح المبدل وبدء عملية التحويل

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:

$$Result = \frac{1}{M} * \sum_{n=0}^{n=N-1} Conversion(t_n)$$

- 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 القيمتين

المبدلات التشابهية الرقمية في متحكمات STM32
ADC in STM32

- **المعايرة الذاتية: Self-calibration**

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 التشابهي The ADC Analog Watchdog (AWD):**

عندما يكون جهد الدخل التشابهي القادم على إحدى القنوات التشابهية أقل من العتبة الدنيا أو أكبر من العتبة العليا المسموح بها عندها يتم تفعيل بت الحالة الخاص بالمراقب التشابهي AWD status bit

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 1.5 cycles، 3.5cycles، 7.5cycles، 12.5cycles، 19.5cycles، 39.5cycles، 79.5cycles، 160.5cycles، ويمكن للمستخدم اختيار قيمة من هذه القيم من خلال البتات SMP[2:0] bits ويتم تخزينها في المسجل ADC_SMPR واستخدامها 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

Categories | A->Z

System Core
Analog
ADC1
ADC2
Timers
Connectivity
Computing
Middleware

ADC1 Mode and Configuration

Mode

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

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Search (Crtl+F)

Discontinuous Conversion Mode	Disabled
ADC_Regular_ConversionMode	
Enable Regular Conversions	Enable
Number Of Conversion	1
External Trigger Conversion Source	Regular Conversion launched by software
Rank	1
Channel	Channel 1
Sampling Time	1.5 Cycles
ADC_Injected_ConversionMode	
Enable Injected Conversions	
WatchDog	
Enable Analog WatchDog Mode	

1.5 Cycles
7.5 Cycles
13.5 Cycles
28.5 Cycles
41.5 Cycles
55.5 Cycles
71.5 Cycles
239.5 Cycles

Pinout view | System view

STM32F103C4Tx
LQFP48

TIM2_CH1 PA0-..
ADC1_IN1 PA1

Console | Error Log | Outline
Workspace Log

Activate Windows
Go to Settings to activate Windows.

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، وزمن أخذ عينات 1.5 cycles يكون الزمن الكلي للتحويل:

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

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 البتات ENVR , HIZ bits الموجودين في المسجل VREFBUF_CSR وعلى المستخدم أن ينتظر حتى يتم تفعيل البت VRR والذي يعني أن الجهد المرجعي الخارجي قد وصل إلى القيمة المطلوبة

ENVR	HIZ	Configuration
0	0	VREF buffer OFF VREF+ pin pulled-down to VSSA
0	**1**	**External voltage reference mode (default):** • **VREF buffer OFF** • **VREF+ pin floating**
1	0	Internal voltage reference mode: • VREF buffer ON • VREF+ pin connected to the VREF buffer output
1	1	Hold mode: • 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 Vref+ وفي هذه الحالة لن تحتاج لوصل دارة خارجية لتوليد الجهد المرجعي، كما يدعم مولد الجهد المرجعي الداخلي قيمتي جهد هي:

- 2.5 :V وهذا يتطلب أن يكون الجهد على الطرف VDDA>= 2.4
- 2.48V: وهذا يتطلب أن يكون الجهد على الطرف VDDA>= 2.8

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 الجهود كما هو موضح بالجدول التالي:

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC, $2.4\ V \leq V_{REF+} \leq V_{DDA}$
V_{DDA}(1)	Input, analog supply	Analog power supply equal to V_{DD} and $2.4\ V \leq V_{DDA} \leq 3.6\ V$
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 التالي:

Vin=ADC_Res*(reference voltage/4096)

حيث :

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

أنماط عمليات التحويل ADC conversion modes

يمكن للمبدل التشابهي الرقمي أن يعمل بعدة أنماط عمل هي :

- **Single-channel, single conversion mode:** وهو نمط العمل الأسهل والأبسط مقارنةً مع بقية الأنماط، حيث يقوم الـ ADC بعملية تحويل وحيدة (وعملية أخذ عينات وحيدة) ولقناة واحدة فقط ويتوقف عن العمل بمجرد انتهائه من عملية التحويل

ADC conversion modes أنماط عمليات التحويل

يمكن للمبدل التشابهي 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 ADC sequencer وباستخدام زمن أخذ عينات مختلف لكل قناة من القنوات

ADC conversion modes أنماط عمليات التحويل

يمكن للمبدل التشابهي الرقمي أن يعمل بعدة أنماط عمل هي :

- **Single-channel continuous conversion mode:** في هذا النمط من العمل يتم تحويل القيمة التشابهية القادمة على قناة واحدة إلى قيمة رقمية وتعاد عملية التحويل بشكل مستمر

ADC conversion modes أنماط عمليات التحويل

يمكن للمبدل التشابهي 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 ADCهي:

- 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 CPUعن تنفيذ الكود وتنتظر لحين الانتهاء من عملية التحويل حينها يمكن للـ CPUاستكمال تنفيذ الكود الرئيسي
- The interrupt Method:تعتبر هذه الطريقة طريقة فعالة لاستخدام المبدل التشابهي الرقمي ، حيث نقوم بقدح المبدل فقط و يمكن للـ CPU أن تستكمل تنفيذ الكود والمهام المطلوبة منها لحين انتهاء عملية التحويل عندها سيقوم المبدل بطلب مقاطعة وستتوجه الـ CPUلبرنامج خدمة المقاطعة
- DMA Method:وهي الطريقة الأكثر فعالية في استخدام المبدل التشابهي الرقمي

الأخطاء الناتجة عن التحويل

قد تنتج عن عملية التحويل التشابهي عدة أخطاء سنذكر نوعين منها هما الأكثر شيوعاً:

- **ADC offset error**: وهو الخطأ الناتج بين أول عملية تحويل فعلية وأول عملية تحويل متوقعة ، حيث تتم أول عملية تحويل عندما يتغير خرج المبدل من 0 إلى 1 ، يتم ذلك بالحالة المثالية عندما يتغير الدخل التشابهي بين الـ 0.5 LSB والـ 1.5 LSB عندها يكون الخرج الرقمي 1 منطقي، ويتم الإشارة إلى خطأ الإزاحة بـ Eo ويمكن معايرته من خلال الكود

الأخطاء الناتجة عن التحويل

قد تنتج عن عملية التحويل التشابهي عدة أخطاء سنذكر نوعين منها هما الأكثر شيوعاً:

- خطأ الربح Gain error : وهو الانحراف بين آخر تحويل فعلي وآخر تحويل مثالي ويعبر عنه بـ E_G حيث يتم آخر تحويل فعلي عند الانتقال بين 0xFFE والـ 0xFFF

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);
```

التطبيق1: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن 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 conversion modeحيث سنربط مقاومة متغيرة معه.
- ضبط الـ timer2 في نمط PWMعلى القناة CH1 وسنربط معه ليد.

التطبيق1: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط Polling

- ضبط تردد ساعة المتحكم

التطبيق1: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط Polling

- -ضبط قطب الدخل التشابهي CH7 في نمط التحويل لمرة واحدة Single conversion modeحيث سنربط مقاومة متغيرة معه.

التطبيق1: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط Polling

- ضبط الـ timer2 في نمط PWMعلى القناة CH1 وسنربط معه ليد.

التطبيق1: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن 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);
```

التطبيق1: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن 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();
```

التطبيق1: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن 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 Perihperal & TimeOut = 1mSec
     HAL_ADC_PollForConversion(&hadc1, 1);
```

التطبيق1: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن 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);
  }
}
```

التطبيق2: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط Interrupt

سنقوم بإعادة الإعدادات السابقة بالإضافة إلى تفعيل مقاطعة المبدل التشابهي الرقمي

التطبيق2: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل 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);
```

التطبيق2: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل 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);
```

التطبيق2: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن 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);
  }}
```

التطبيق2: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWMمن خلال مقاومة متغيرة موصولة على أحد أقطاب الدخل التشابهي 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

متحكمات STM32

7

University of Aleppo - Hexabitz - Spring 2023

موضوعات المحاضرة:

- 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 قدح 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 المقابل لها

المبدلات الرقمية تشابهية 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 نمط12bit
- 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 تشابهية 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 Vref+
- يمكن قدح عملية التبديل بعدة طرق إما من خلال قدح خارجي أو داخلي من خلال مؤقت على سبيل المثال
- يمكن استخدام المبدل الرقمي التشابهي بنمط الـ DMA
- يمكن استخدام المبدل الرقمي التشابهي بنمط الـ 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

نمط 8-bit:

- Right aligned data input:

نمط 12-bit:

- Right aligned data input:

- Left aligned data input:

DAC data formats – For dual channel mode

نمط 8-bit:

- Right aligned data input:

نمط 12-bit:

- Right aligned data input:

- Left aligned data input:

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 أقطاب I/O
- من خلال ضبط بت القدح برمجياً SWTR

DAC conversion triggers طرق قدح المبدل

Source	Type	TSEL[2:0]
Timer 6 TRGO event	Internal signal from on-chip timers	000
Timer 3 TRGO event		001
Timer 7 TRGO event		010
Timer 5 or Timer 15 TRGO event		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 الـ 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

- نمط الـ sample and hold مصمم خصيصاً من أجل أوضاع الطاقة المنخفضة
- يمكن ضبط أزمنة sample hold and refresh timings من خلال الإعدادات.
- في هذا النمط من العمل يتم قيادة المبدل وجميع المسجلات والدارات المنطقية المتعلقة به من خلال الهزاز الكريستالي الداخلي منخفض السرعة low speed internal oscillator(lsi_clk)

جهد خرج المبدل الرقمي التشابهي

- يتم حساب جهد خرج المبدل من خلال العلاقة التالية:

$$DACoutput = (V_{REF}+) \times (DOR/4096)$$

ويتراوح جهد خرج المبدل بين 0 والـ VREF+

جهد خرج المبدل الرقمي التشابهي

- عنـدما يكـون output buffer فـي حالـة فصـل off ، عندها يكون جهد دخل المبـدل متصـل بجهـد خرجـه عن طريـق المقـاومتين Ra، Rb وتصـبح ممانعـة المبـدل هو عبارة عن مجموع هـاتين المقـــــــاومتين Ra+Rb وباعتبـــــار المقـــــاومتين متســاويتين تصـبح ممانعـة المبـدل 2*Ra (وهنـا يكـون المفتـــاحين S1 و S2 فـــي حالـة فصل open)

جهد خرج المبدل الرقمي التشابهي

- عندما يكون output buffer في حالة وصل on ، عندها يكون المضخم العملياتي يعمل كمضخم عاكس وبعامل ربح 1- وتكون ممانعة الخرج صفرية

سرعة التبديل للمبدل الرقمي التشابهي

- 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

University of Aleppo - Hexabitz - Spring 2023
University of Aleppo - Hexabitz - Spring 2023

استخدام مضخم عملياتي خارجي

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

زمن أخذ العينات الأعظمي للمبدل

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 التشابهي في متحكمات stm32 كمولد إشارة ضجيج عشوائي باستخدام مسجل إزاحة وعدد من نقاط التفرع مساوٍ لـ 2n-1 وهو العدد الذي تحتاجه الإشارة قبل أن تكرر 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 الالكترونية 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 الالكترونية

توليد إشارة الضجيج باستخدام المبدل White noise generator

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

Triangular wave generator توليد إشارة مثلثية باستخدام المبدل

- يمكن للمبدل الرقمي التشابهي في متحكمات stm32 توليد إشارة مثلثية مع قابلية التحكم بمطالها وترددها وأيضاً مقدار إزاحة الإشارة من خلال المسجل 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

Triangular wave generator توليد إشارة مثلثية باستخدام المبدل

Triangular wave generator توليد إشارة مثلثية باستخدام المبدل

- كما يمكن الحصول على إشارة مثلثية مزاحة 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

استخدام المبدل الرقمي التشابهي لتوليد أشكال مختلفة من الإشارات

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 المرادة 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	1023
Max Amplitude	65535
Numbers Per Row	8
◉ Hex	○ Decimal
	Submit

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 المرادة 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,
0xab29,0xabe6,0xaca3,0xad5f,0xae1b,0xaed7,0xaf92,0xb04d,
0xb107,0xb1c0,0xb27a,0xb332,0xb3ea,0xb4a2,0xb559,0xb610,
0xb6c6,0xb77c,0xb831,0xb8e5,0xb999,0xba4d,0xbb00,0xbbb2,
0xbc64,0xbd15,0xbdc6,0xbe76,0xbf25,0xbfd4,0xc082,0xc12f,
0xc1dc,0xc288,0xc334,0xc3df,0xc489,0xc533,0xc5dc,0xc684,
0xc72c,0xc7d3,0xc879,0xc91f,0xc9c3,0xca67,0xcb0b,0xcbae,
0xcc4f,0xccf1,0xcd91,0xce31,0xced0,0xcf6e,0xd00b,0xd0a8,
0xd144,0xd1df,0xd279,0xd313,0xd3ac,0xd443,0xd4db,0xd571,
```

استخدام المبدل الرقمي التشابهي لتوليد أشكال مختلفة من الإشارات

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

```
clear; clc;    % Clear The Previous Points
Ns     = 128; % Set The Number of Sample Points
RES    = 12;  % Set The DAC Resolution
OFFSET = 0;   % Set An Offset Value For The DAC Output
%------------[ Calculate The Sample Points ]-------------

T = 0:((2*pi/(Ns-1))):(2*pi);
Y = sin(T);
Y = Y + 1;
Y = Y*((2^RES-1)-2*OFFSET)/(2+OFFSET);
Y = round(Y);
plot(T, Y);
grid
%--------------[ Print The Sample Points ]---------------

fprintf('%d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, \n', Y);

%-------------------[ Other Examples To Try ]-------------------

%  Y = diric(T, 13);      % Periodic Sinc

%  Y = sawtooth(T)        % Sawtooth

%  Y = sawtooth(T, 0.5); % Triangular
```

DMA capabilities

- تحتـــــوي متحكمـــــات stm32على الأقل على موديـول DMAواحـد بعدة قنوات
- ترتبط كل قناة من قنوات DAC مـــع قنـــاة مـــن قنوات الـ DMA
- فـي حـال عـدم اسـتخدام نمـط الــ DMA ، يقـوم المعـالج بتزويـد المبـدل الرقمي التشابهي

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

DMA capabilities

- أما عند استخدام نمط الـ DMA مـــع المبـــدل الرقمـي التشـابهي ، فـإن الموديـــول الـــ DMA يقـوم بـدور الوسـيط بـين الـذاكرة RAM والمبـدل ، بدون تدخل من المعالج ممـــا يـــؤدي إلـــى زيـــادة كفـــاءة وفعاليـــة النظـــام ويتـــرك المعـــالج ليقـــوم بالعمليات الأخرى

ضبط إعدادات المبدل في بيئة stm32CubeIDE

هناك أربع خيارات لكل قناة من قنوات المبدل:

- Disabled: في هذه الحالة تكون القناة غير مفعلة، 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:

STM32CubeIDE_1.5.1 - Device Configuration Tool - STM32CubeIDE

File Edit Navigate Search Project Run Window Help

Project E... | *test.ioc

test
- Includes
- Core
- Drivers
- STM32G071RE
- test.ioc

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core
Analog
- ADC1
- COMP1
- COMP2
- DAC1

Timers
Connectivity
Multimedia
Computing
Middleware
Utilities

DAC1 Mode and Configuration

OUT1 mode: Connected to on chip-peripherals only
OUT2 mode: Disable
External: Connected to external pin only
Connected to on chip-peripherals only
Connected to external pin and to on chip-peripherals

Configuration

Reset Configuration

NVIC Settings | DMA Settings
Parameter Settings | User Constants

Configure the below parameters :

Search (Crtl+F)

DAC Out1 Settings

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

Pinout view | System view

STM32G071RBTx
LQFP64

Console | Error Log | Outline

Workspace Log

type filter text

Activate Windows
Go to Settings to activate Windows.

ضبط إعدادات المبدل في بيئة stm32CubeIDE

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

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

		Peripheral	Memory
Mode	Normal	Increment Address	☐	☑
		Data Width	Half Word	Half Word

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 والطرفيات، وبين الذواكر 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:
  - AHB peripherals: على سبيل المثال CRC generator
  - AHB memories: على سبيل المثال SRAM
  - APB peripherals: على سبيل المثال USART

إدارة الوصول المباشر للذاكرة Direct Access Memory (DMA)

- سابقاً وقبل وجود وحدة الـ 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 Direct Access Memory (DMA)

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

إدارة الوصول المباشر للذاكرة Direct Access Memory (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 ، ومما يعني انشغال المعالج عن تنفذ الكود ومما يؤدي في بعض الأحيان إلى تجاوز القيود الخاصة بالزمن المحدد لتنفيذ الكود، وهنا تأتي الحاجة لوحدة الـ DMA

إدارة الوصول المباشر للذاكرة Direct Access Memory (DMA)

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

إدارة الوصول المباشر للذاكرة Direct Access Memory (DMA)

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

إدارة الوصول المباشر للذاكرة Direct Access Memory (DMA)

مواصفات وحدة الـ DMA في متحكمات stm32G0

- **AHB master bus:**

يدعم متحكم DMA منفذي AHB-Lite أحدهما master يستخدم من قبل قنوات الـ DMA للوصول إلى الذاكرة أو إلى مسجلات الطرفيات، والآخر slave يستخدم للوصول إلى متحكم وحدة الـ DMA ومسجلات الحالة والتحكم الوحدة.

Direct Access Memory إدارة الوصول المباشر للذاكرة (DMA)

مواصفات وحدة الـ DMA في متحكمات stm32G0

- المرونة في ضبط الإعدادات:

تدعم معظم الطرفيات في متحكم stm32 نمط الـ DMA ، حيث يعتبر هذا النمط هو الأفضل خصوصاً مع طرفيات الاتصال communication peripherals والمبدلات ADC/DAC

إدارة الوصول المباشر للذاكرة Direct Access Memory (DMA)

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

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

إدارة الوصول المباشر للذاكرة Direct Access Memory (DMA)

مواصفات وحدة الـ DMA في متحكمات stm32G0

- إمكانية تعديل الأولويات برمجياً أو عن طريق الـ hardware:

يدعم متحكم الـ DMA ثلاث مخارج للمقاطعة متصلة بمتحكم المقاطعات NVIC

يمكن تعديل الأولويات لها من خلال البرنامج أو من خلال الهاردوير

إدارة الوصول المباشر للذاكرة Direct Access Memory (DMA)

مواصفات وحدة الـ DMA في متحكمات stm32G0

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

إدارة الوصول المباشر للذاكرة Direct Access Memory (DMA)

مواصفات وحدة الـ DMA في متحكمات stm32G0

- New DMA request Multiplexer (DMAMUX): وهي ميزة موجودة في عائلة stm32G0 وغير موجودة في عائلة stm32f0 ، ومهمتها تنظيم وتخطيط مسار الطلبات القادمة من الطرفيات إلى قنوات الـ 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-Memory transfer**

Stream transfer management

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

Address	Data[7:0]
0xXXXX_XXX0	B0
0xXXXX_XXX1	B1
0xXXXX_XXX2	B2
0xXXXX_XXX3	B3

Address	Data[
0xXXXX_XXX0	00000
0xXXXX_XXX4	00000
0xXXXX_XXX8	00000
0xXXXX_XXXC	00000

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

Address	Data[31:0]
0xXXXX_XXX0	B3B2B1B0
0xXXXX_XXX1	B7B6B5B4
0xXXXX_XXX2	BBBAB9B8
0xXXXX_XXX3	BFBEBDBC

Address	Data[15:0]
0xXXXX_XXX0	B1B0
0xXXXX_XXX2	B5B4
0xXXXX_XXX4	B9B8
0xXXXX_XXX6	BDBC

Circular mode

- في هذا النمط يمكن استخدام الأحداث التي تزودنا بها وحدة الـ 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 وعند الانتهاء منها يتم البدء بالعملية التالية بشكل آلي
- هذا النمط لا يدعم تبادل البيانات memory-to-memory

ضبط الأولويات لقنوات الـ DMA

- هناك أربع خيارات متاحة لضبط الأولويات بين القنوات:

Thank you for listening

متحكمات STM32

8

University of Aleppo - Hexabitz - Spring 2023

موضوعات المحاضرة:

- مبدأ عمل البرتوكول I2C
- مزايا بروتوكول الاتصال التسلسلي I2C في متحكمات STM32
- المخطط الصندوقي لوحدة I2Cفي متحكمات STM32
- أنماط العمل المختلفة للوحدة I2Cفي متحكماتSTM32
- دوال مكتبة HAL المستخدمة للتعامل مع وحدة I2C
- ضبط إعدادات وحدة I2Cفي متحكماتSTM32

مقارنة بين بروتوكولات الاتصال التسلسلي 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 I^2C
(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 ثنائي الاتجاه.
- خط التزامن SCL تتولد إشارته من عنصر Master وله حالتين:

1) أحادي الاتجاه في الأنظمة Single-master.
2) ثنائي الاتجاه في الأنظمة Multi-master.

بروتوكول الاتصال التسلسلي I^2C
(Inter Integrated Circuit)

- خرج الأجهزة في البروتوكول I^2C 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 "Slave" المراد التخاطب معه من الجهاز "Master".
- 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 I^2C
(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 400pF، وغالباً تكون سعة كل جهاز بحدود 10pF (40 جهاز).
- الطول الأعظمي على ناقل I2C يمكن ان يصل إلى 6 أمتار.
- تختلف سرعة نقل البيانات على ناقل I2C حسب المعيار الشريحة:

1) Std.: 100kbps
2) Fast: 400kbps
3) Hi: 3.4Mbps

بروتوكول الاتصال التسلسلي I2C في متحكمات STM32:

EEPROM and 8-bit I/O Expander Device Selected

A2	A1	A0	Device
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

بروتوكول الاتصال التسلسلي I2C في متحكمات STM32:

(مبدأ العمل في بروتوكول الاتصال التسلسلي I^2C)

- إن مبدأ العمل يعتمد على برتوكول Master-Salve حيث أن جهاز الــMaster 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 الــMaster بالاستجابة بإشارة مصادقة "ACK" يُعلم من خلالها الــMaster بوجوده على الناقل.

(مبدأ العمل في بروتوكول الاتصال التسلسلي I^2C)

4) بعد أن تم تحديد طرفي التخاطب، تبدأ عملية تراسل البيانات بين الطرفين ويتم تحديد المرسل والمستقبل من خلال بت تحديد العملية (قراءة "1" \ كتابة "0") في بايت العنوان.

5) كلما تم إرسال 8-bit من المرسل يقوم المستقبل (أياً يكن M or S) بالتأكيد بإرسال بإشارة 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 I^2C)

Master controls the clock signal

(مبدأ العمل في بروتوكول الاتصال التسلسلي I^2C)

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 I^2C)

Data transition while SCL is low…

Data validation while SCL is high…

(مبدأ العمل في بروتوكول الاتصال التسلسلي I^2C)

- **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 I2C كـ Master أو Slave
- 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 I2C كـ Open Drain ووصل مقاومات رفع خارجية معها
- مصدر الساعة للوحدة الطرفية قد يكون :
  1) system clock
  2) high-speed internal 16 MHz RC oscillator
- تدعم سرعتي نقل للبيانات على ناقل I2C :
  1) Std.: 100kbps
  2) Fast: 400kbps

المخطط الصندوقي لوحدة I2C في متحكمات STM32:

University of Aleppo - Hexabitz - Spring 2023

أنماط العمل المختلفة للوحدة I2C في متحكمات 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 للتعامل 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 7بت ويجب إزاحته نحو اليسارقبل استدعاء الطرفية(للتحويل من8بت لـ 7بت)

pData: مؤشر الـ 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 للتعامل 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 للتعامل 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 لكامل البايتات 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 للتعامل 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);
```

ـ بعد استدعاء هذه الدالة تقوم وحدة I2C ببدء عملية الاستقبال لكامل البايتات 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 للتعامل مع وحدة I2C في وضع الـ 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 لكامل البايتات 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 للتعامل مع وحدة I2C في وضع الـ 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 الـ buffer وعند الانتهاء يتم استدعاء التابع التالي:

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

دوال مكتبة HAL المستخدمة للتعامل مع وحدة I2C

- إذا أردنا ارسال 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 (نمط 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:

MemAddress: عنوان الذاكرة الداخلية ضمن الجهاز المرسل له

MemAddSize: حجم الذاكرة الداخلية

ومن أجل نمط المقاطعة نستخدم الدالة: HAL_I2C_Mem_Read_IT

ومن أجل نمط الـ DMA نستخدم الدالة HAL_I2C_Mem_Read_DMA

دوال مكتبة HAL المستخدمة للتعامل مع وحدة I2C

- لارسال البيانات (نمط 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 الـ DMA نستخدم الدالة HAL_I2C_Mem_Write_DMA

دوال مكتبة HALالمستخدمة للتعامل مع وحدة I2C في وضع الـ 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 I2C

- Disable
- I2C
- SMBus-Alert-mode
- SMBus-two-wire-Interface

- ضبط بارامترات وحدة I2C وتتضمن حالتين :

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

ضبط إعدادات وحدة I2C في متحكمات STM32

- ضبط بارامترات وحدة I2C وتتضمن حالتين :

2) عندما تكون الوحدة تعمل بنمط Slave عندها يمكن ضبط كل من عدد بتات 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 master من إجراء أي عملية لحين تحرير خط الـ SCL

التطبيق العملي: ربط ذاكرة eeprom مع المتحكم Stm32G0B1CE

التطبيق العملي: ربط ذاكرة eeprom مع المتحكم Stm32G0B1CE

Unleash your Creativity!

AT24C02 Datasheet

نهاية المحاضرة

Thank you for listening

متحكمات STM32

9

موضوعات المحاضرة:

- مبدأ عمل البرتوكول SPI
- مزايا بروتوكول الاتصال التسلسلي SPI في متحكمات STM32
- المخطط الصندوقي لوحدة SPI في متحكمات STM32
- أنماط العمل المختلفة للوحدة SPI في متحكماتSTM32
- دوال مكتبة HAL المستخدمة للتعامل مع وحدة SPI
- ضبط إعدادات وحدة SPI في متحكماتSTM32

مقارنة بين بروتوكولات الاتصال التسلسلي 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 أمتار.
- 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 برتوكول SPI أكثر استخداماً وانتشاراً من I2C.
- حزمة البيانات يمكن أن تكون بطول أكبر من 8-bit خلافاً لـI2C.
- 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 للتخاطب مع الوحدات المحيطية 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

بروتوكول الاتصال التسلسلي SPI في متحكمات STM32:

Ref.: http://www.ermicro.com/blog/?p=1050

بروتوكول الاتصال التسلسلي SPI في متحكمات STM32:

SPI Master and Slave Data Transfer Diagram

Ref.: http://www.ermicro.com/blog/?p=1050

❖ أنماط التوصيل لناقل 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

مزايا بروتوكول الاتصال التسلسلي SPI في متحكمات STM32:

- 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)

مزايا بروتوكول الاتصال التسلسلي SPI في متحكمات STM32:

- 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 الصندوقي لوحدة SPI في متحكمات STM32:

أنماط العمل المختلفة للوحدة SPI في متحكمات STM32

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

وبحيث لاتتجاوز سرعة الاتصال fpclk/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 نحتاج إلى خط بيانات وحيد
- يتم اختيار هذا النمط من العمل عند اختيار نمط **transmit-only** أو **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 MOSI من الـ master مع قطب MOSI من الـ slave
- في هذا النمط يكون وصل قطب NSS اختياري لوجود جهاز slave واحد فقط

2- Half-duplex Mode

- في هذا النمط من العمل يتم استخدام خط بيانات وحيد، حيث يتم وصل قطب الـ MOSI من الـ Master مع الـ MISO من الـ Slave من خلال مقاومة 1Kohm
- وبإمكان كل من الـ Master والـ Slave إرسال واستقبال البيانات ولكن في اللحظة الواحدة يكون إما مرسل أو مستقبل

2- Half-duplex Mode

- فعندما يقوم الـ slave بإرسال البيانات على خط MISO يقوم الـ Master باستقبالها ويكون خط الـ MOSI في هذه الحالة للـ Slave 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 البينات 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 ويقو مالـ 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 الـ Nss قطب دخل وتكون مهمته chip select أي قطب اختيار الشريحة
- Master mode: في هذه الحالة وعندما يكون هناك جهاز Master واحد ،يكون قطب الـ Nss قطب خرج وتكون مهمته اختيار الـ slave المراد الاتصال به

Motorola Mode and Texas Instrument Mode

نميز نمطي عمل لوحدة 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 للتعامل 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 الـ data buffer للبيانات المراد إرسالها، Size : حجم البيانات، Timeout: مدة المهلة قبل فشل الإرسال

- **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 للتعامل 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 البايتات 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 للتعامل 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 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 للتعامل 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 للتعامل مع وحدة SPI في وضع الـ 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 لكامل البايتات 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 للتعامل 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 للتعامل مع وحدة SPI في وضع الـ 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 لكامل 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 إعدادات وحدة SPI في متحكمات STM32

- ضبط بارامترات 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:

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters :

Search (Crtl+F)

Basic Parameters	
Frame Format	Motorola
Data Size	8 Bits
* First Bit	MSB First
Clock Parameters	
Prescaler (for Baud Rate)	2
Baud Rate	8.0 MBits/s
* Clock Polarity (CPOL)	Low
* Clock Phase (CPHA)	1 Edge
Advanced Parameters	
CRC Calculation	Disabled
* NSSP Mode	Enabled
NSS Signal Type	Output Hardware

ضبط إعدادات وحدة SPI في متحكمات STM32

- ضبط بارامترات وحدة SPI وتتضمن:

كما يمكن تفعيل المقاطعة الخاصة بوحدة الـ SPI من خلال تاب NVIC Settings

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

التطبيق العملي : المطلوب ربط مجموعة متحكمات Stm32G0B1CE من خلال النافذة التسلسلية SPI أحدها Master والبقية Slave.

Thank you for listening

متحكمات STM32

10

University of Aleppo - Hexabitz - Spring 2023

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

Counter Mode

Counter Mode

نميز ثلاث أنماط مختلفة للمؤقت عندما يعمل كـ Counter وهي:

- **نمط العد التصاعدي:** في هذا النمط فإن العداد يبدأ بالعد من الصفر مع كل نبضة قادمة على قطب الدخل ويستمر حتى يصل إلى القيمة المخزنة مسبقاً والموجودة 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 auto-reload value في المسجل TIMx-ARR مع كل نبضة قادمة على قطب الدخل ويستمر ليصل إلى الصفر ثم يعو ليبدأ من القيمة المخزنة سابقاً ويولد حدث الـ **Underflow event**

Counter Mode

نميز ثلاث أنماط مختلفة للمؤقت عندما يعمل كـ Counter وهي:

- **نمط العد التصاعدي تنازلي :** 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-ARRناقص واحد ، ثم يتم توليد حدث الطفحان Overflow event ثم يبدأ بالعد التنازلي من القيمة المخزنة سابقاً auto-reload valueمع كل نبضة قادمة على قطب الدخل وحتى يصل إلى الصفر عندها يتم توليد حدث الـ Underflowثم يعود للعد التصاعدي من الصفر وهكذا...

Counter Mode

نميز ثلاث أنماط مختلفة للمؤقت عندما يعمل كـ 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 (ARR=3, 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 الدخل للمؤقت) على المصدر الخارجي external pin وهو ETR2 والذي يوافق القطب PA0
- ضبط القيمة المخزنة مسبقاً للعداد counter period على القيمة 20 ، عندها عندما يصل العداد إلى القيمة 20 ستحدث مقاطعة الطفحان

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 Switch bouncing، حيث يتراوح مجال القيم للفلتر بين 0 والـ 15، 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 CubeMX نختار ما يلي :

*counter.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: LPTIM1, LPTIM2, RTC, TIM1, TIM2, TIM3, TIM6, TIM7, TIM14, TIM15, TIM16, TIM17

Connectivity

Multimedia

Computing

TIM2 Mode and Configuration

Mode

Slave Mode: Disable
Trigger Source: Disable
Clock Source: ETR2
Channel1: Disable
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 (Crtl+F)

Counter Period (AutoRelo... 20
Internal Clock Division (C... No Division
auto-reload preload: Enable
Trigger Output (TRGO) Parameters
Master/Slave Mode (MS... Disable (Trigger input effect not dela...
Trigger Event Selection T... Reset (UG bit from TIMx_EGR)
Clock
Clock Filter (4 bits value): 15
Clock Polarity: non inverted
Clock Prescaler: Prescaler not used

Pinout view | System view

Input Capture Mode

- المؤقت في نمط Input Capture يستقبل نبضات الساعة الخاصة به من مصدر داخلي (ساعة المتحكم بعد استخدام المقسم الترددي) 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 channels output) capture/compare مرتبطة بأقطاب المتحكم يمكن معرفتها من خلال الـ datasheet الخاصة بالمتحكم

Input Capture Mode

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

Input Capture Mode

Input Capture Mode

- لضبط المؤقت في نمط الـ Input Capture من خلال CubeMX

نختار ما يلي :

*input_capture.ioc

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Device Configuration Tool

Software Packs | Pinout

Categories | A->Z

System Core
Analog
Timers
LPTIM1
LPTIM2
RTC
TIM1
TIM2
TIM3
TIM6
TIM7
TIM14
TIM15
TIM16
TIM17
Connectivity
Multimedia
Computing
Middleware
Utilities

TIM2 Mode and Configuration

Mode

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 (Crtl+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

Pinout view | System view

PC10 PB9 PB8 PB7 PB6 PB5 PB4 PB3 PD6 PD5 PD4 PD3 PD2 PD1
PC11
PC12
SYS_WKUP2 PC13
RCC_OSC32_IN PC14-...
RCC_OSC32_OUT PC15-...
VBAT
VREF+
VDD
VSS
MCO PF0-O...
PF1-O...
PF2 - ...
PC0
PC1
PC2
PC3
PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7 PC4 PC5 PB0 PB1 PB2 PB10

STM32G071RBTx
LQFP64

Output Compare Mode

- هذا النمط من العمل يسمح بالتحكم بخرج معين خلال زمن معين ، فعند تتساوى القيمة الحالية للعداد مع القيمة المخزنة في مسجل المقارنة OCRيتم تغيير حالة الخرج الرقمي وفقاً لما تمت برمجته خلال الكود فمثلاً قد يصبح وضع HIGHأو LOWأو togglesأو حتى يبقى بنفس وضعه دون تغيير

Output Compare Mode

- يوضح الشكل التالي مثال على نمط Output compare حيث المطلوب عمل toggle لقطب 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 PWMأو Output compare،كما يتم توليدها فقط عندما تكون قيمة المقارنة مختلفة عن القيمة الابتدائية للعداد، حيث يجب عند ضبط الإعدادات أن تكون $CNT<CCRx \leq ARR$وبالأخص يجب أن تكون $CCRx>0$

One Pulse Mode

- على سبيل المثال إذا أردنا توليد نبضة على الخرج OC1 وبعرض tPULSE وبتأخير زمني tDELAY عند الجبهة الصاعدة للإشارة على القطب : TI2

One Pulse Mode

- لضبط المؤقت في نمط الـ One Pulse Mode من خلال CubeMX

نختار ما يلي :

Encoder Mode

- يعمل المؤقت في نمط الـ Encoder كعداد لنبضات ساعة خارجية external clock لمدخلين ، أي أن العداد يبدأ بالعد من الصفر حتى القيمة المخزنة في المسجل TIMx_ARR
- حيث من خلال تتابع الإشارات على هذين المدخلين يتم تحديد اتجاه العد تصاعدي/تنازلي بالإضافة إلى عد النبضات القادمة على هذين الدخلين
- أي أن العداد يقوم بملاحقة السرعة واتجاه الدوران للانكودر المتصل مع قناتي المؤقت، حيث لا يحتاج الانكودر لأي دارة ملاءمة بينه وبين المتحكم.

Encoder Mode

- يمكن للمستخدم الحصول على معلومات ديناميكية كالسرعة والتسارع من خلال استخدام مؤقت في نمط Encoder

Encoder Mode

- لضبط المؤقت في نمط الـ Encoder Mode من خلال CubeMX

نختار ما يلي :

Pinout & Configuration | Clock Configuration | Additional Software | Pinout

Categories | A->Z

System Core
Analog
Timers
RTC
TIM1
TIM2
TIM3
TIM4
Connectivity
Computing
Middleware

TIM2 Mode and Configuration

Mode

Channel4 Disable
Combined Channels Encoder Mode
Use ETR as Clearing Source

Configuration

Reset Configuration

NVIC Settings | DMA Settings | GPIO Settings
Parameter Settings | User Constants

Prescaler (PSC - 16 bits v...	0
Counter Mode	Up
Counter Period (AutoRelo...	65535
Internal Clock Division (CK...	No Division
auto-reload preload	Enable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM...	Disable (Trigger input effect not dela...
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
Encoder	
Encoder Mode	Encoder Mode TI1 and TI2
____ Parameters fo...	
Polarity	Rising Edge
IC Selection	Direct
Prescaler Division Ratio	No division
Input Filter	10
____ Parameters fo...	
Polarity	Rising Edge

أهم مسجلات المؤقتات في متحكمات STM32

- **TIMx ->CNT:** مسجل بطول 16بت ويمثل القيمة الحالية للمؤقت
- **TIMx->ARR:** مسجل بطول 16بت وهو مسجل إعادة التحميل التلقائي، ويتم استخدامه للتحكم بتردد إشارة الـ PWM
- **TIMx->PSC:** مسجل بطول 16بت وهو مسجل المقسم الترددي
- **TIMx_CCRx:** وهو Capture/Compare Registers، وأيضاً يستخدم للتحكم بدورة التشغيل dutycycle عند عمل المؤقت بنمط الـ PWM

تلخيص لدوال مكتبة HAL المستخدمة للتعامل مع المؤقتات

- دالة بدء المؤقت بنمط الـ Polling: HAL_TIM_Base_Start();
- 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);
```

أهم برامج خدمة المقاطعة للمؤقتات في متحكمات STM32

- دالة خدمة مقاطعة الطفحان:

```
HAL_TIM_PeriodElapsedCallback( TIM_HandleTypeDef* htim)
{

}
```

- دالة خدمة مقاطعة Input Capture:

```
HAL_TIM_IC_CaptureCallback()
{

}
```

Thank you for listening

متحكمات STM32

11

University of Aleppo - Hexabitz - Spring 2023

موضوعات المحاضرة:

- 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 عدة مزايا فيما يتعلق بأنماط الطاقة منها:

- تؤمن متحكمات stm32G0 7 أنماط للطاقة المنخفضة وبقدرة عالية على الاستيقاظ السريع fast Wakeup تتضمن :
  - ✓ استجرار لايتجاوز الـ 40nA مع القدرة على إيقاظ الـ CPU من خلال أحد أقطاب I/O
  - ✓ استجرار لايتجاوز الـ 320nA لاسترجاع المعلومات من ذاكرة SRAM بحجم36Kb
  - ✓ كما بإمكان عدد كبير من الطرفيات الموجودة في المتحكم إيقاظ المعالج
- استجرار المتحكم لايتجاوز الـ 100UA/MHZ عند عمل المتحكم بنمط Run mode وبأعلى تردد عمل ممكن
- كما لمتحكم stm32G0 قطب خاص لتغذية المتحكم من خلال بطارية VBAT

Power schemes

منظمات الجهد في متحكمات stm32G0

تحتوي متحكمات STM32G0 على منظمي جهد:

- منظم الجهد الرئيسي Main regulator : وله مجالي 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

منظمات الجهد في متحكمات stm32G0

أنماط الطاقة في متحكمات stm32G0

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: Range1

- 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

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

Ex: Execution from SRAM memory

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
100 µA/MHz at 64 MHz
(6.4 mA)

Range 2
75 µA/MHz at 16 MHz
(1.2 mA)

Available clocks

- HSI16
- HSE
- LSI
- LSE

Active cell

Clocked-off cell

Cell in power-down

Available Periph and clock

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

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

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

From SRAM
79 µA/MHz at 2 MHz
(159 µA)

Available clocks

- HSI16
- HSE
- LSI
- LSE

Active cell | Clocked-off cell | Cell in power-down | Available Periph and clock

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:

- Sleep Mode (range1): استخدام منظم الجهد الرئيسي Main regulator ضمن مجال العمل الأول Range1
- Sleep Mode(range2): استخدام منظم الجهد الرئيسي Main regulator ضمن مجال العمل الثاني Range2
- Low-power Sleep: استخدام منظم الطاقة المنخفضة Low-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 for Interrupt) أو WFE(Wait For Event)
- هناك آليتين للدخول بهذا النمط:
  - ✓ بمجرد تنفيذ إحدى التعليمتين WFI/WFE
  - ✓ عند الانتهاء من تنفيذ برنامج خدمة المقاطعة صاحب أدنى أولوية lowest priority 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

3. Sleep Mode : Range2

- في هذا النمط من العمل لايمكن 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

Ex: Flash memory ON but cannot be programed or erased

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

Range 1
25 µA/MHz at 64 MHz
(1.6 mA)

Range 2
25 µA/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

life.augmented

4. Low-power Sleep Mode

- في هذا النمط من العمل يتم فصل نبضات الساعة عن الـ CPU
- يمكن تزويد الوحدات المنطقية بالتغذية 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

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 ها 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، وبالتالي تبقى بحالة تفعيل
- 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 الداخليHSI
- 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 µA @ 3.0 V
Wakeup time to 16 MHz:
In SRAM: 2 µs
In Flash ON: 2 µs
In Flash OFF: 5.5 µs
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
life.augmented
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

Zzz

Cortex M0+

Flash memory

SRAM (36 Kbytes)

Flash memory not powered:
- w/o RTC: 1.3 µA @ 3.0 V
- w/ RTC: 4.1 µA @ 3.0 V

Flash memory powered:
- w/o RTC: 7.0 µA @ 3.0 V

Wakeup time to 16 MHz:
- In SRAM: 5 µs
- In Flash ON: 5 µs
- In Flash OFF: 9 µs

Main regulator (MR)

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

Available clocks

- **HSI16**
- HSE
- **LSI**
- **LSE**

Active cell | Clocked-off cell | Cell in power-down | **Available Periph and clock**

life.augmented

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 سيتم 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

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 monitoring أي أن الـ power down reset في حالة 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
- الهزاز 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 and 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

Wake-up event

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

Available clocks

- HSI16
- HSE
- LSI
- **LSE**

life.augmented

Active cell | Clocked-off cell | Cell in power-down | **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 Any interrupt or event	11 cycles
	MR Range 2						
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 HALالمستخدمة 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 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)
{
  HAL_ResumeTick();
}
```

دوال مكتبة HALالمستخدمة لإيقاظ المتحكم من نمط الـ sleep mode

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

```
HAL_PWR_EnableSleepOnExit ();
```

- ولإلغاء تفعيل نمط المقاطعة نستخدم:

```
HAL_PWR_DisableSleepOnExit ();
```

دوال مكتبة HALالمستخدمة لإدخال المتحكم في نمط الـ stop mode

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

```
HAL_SuspendTick();
```

- يتم إدخال المتحكم بنمط الـ stop 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_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);
```

دوال مكتبة HALالمستخدمة لإيقاظ المتحكم من نمط الـ stop mode

- في حال استخدمنا التعليمة WFI فإن المتحكم يمكن إيقاظه من خلال أي من مقاطعات الطرفيات، watchdog (IWDG), ، RTC 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 HALالمستخدمة 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 HALالمستخدمة لإدخال المتحكم في نمط الـ standby mode

- ثم تفعيل قطب الإيقاظ

```
HAL_PWR_EnableWakeUpPin(PWR_WAKEUP_PIN1);
```

- يمكن أيضاً استخدام الإيقاظ الدوري باستخدام الـ 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 HALالمستخدمة لإدخال المتحكم في نمط الـ standby mode

- ثم أخيراً للدخول في نمط الـ standby

```
HAL_PWR_EnterSTANDBYMode();
```

Thank you for listening