LABORATORIO:
RETI PER AUTOMAZIONE INDUSTRIALE
STM32 NUCLEO

INTRO

antonino.raucea@dieei.unict.it

Ing. Antonino Raucea

Presentazione del corso

- Creare un vocabolario comune
- Permettere di lavorare con micro della STMicroelectronics (www.st.com) a 32 bit della famiglia STM32, la piattaforma usata per gli esempi, in particolare usando la piattaforma Nucleo
- Mostrare tecniche e stili di programmazione e debugging, lontani dalla classica programmazione per programmi PC
- **Un'opportunità!!!**

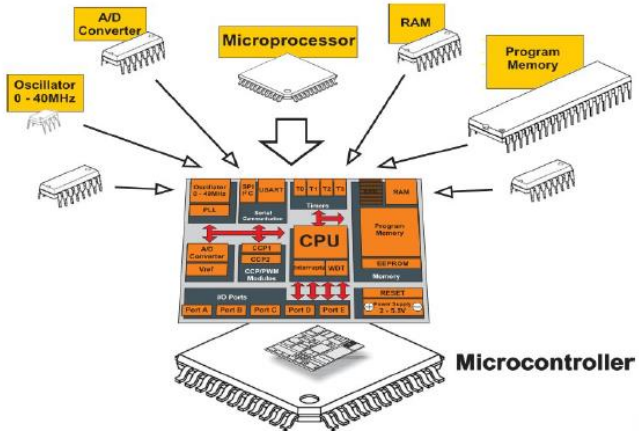
Prerequisiti

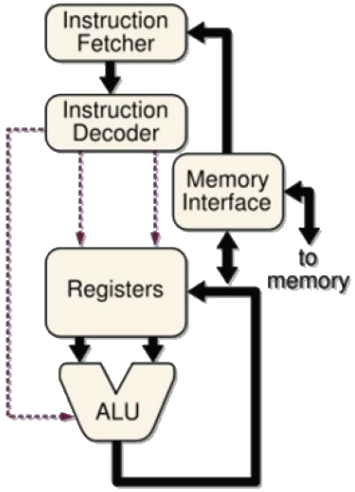
- Conoscenza del C
- Conoscenze «spicciole» di Elettrotecnica e di Elettronica
- **Volontà e pazienza**

Microcontrollore o in breve Micro

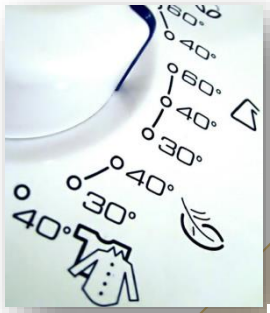
Esempi

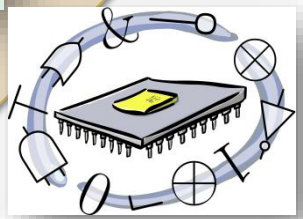
Un microprocessore è destinato unicamente ad effettuare calcoli. Un microcontrollore contiene al suo interno un microprocessore (CPU – sempre destinato a fare calcoli) più numerose altre periferiche che gli permettono di interagire con il mondo esterno: linee di comunicazione, un certo quantitativo di memoria per immagazzinare dati e programmi, convertitori analogico/digitali ecc.



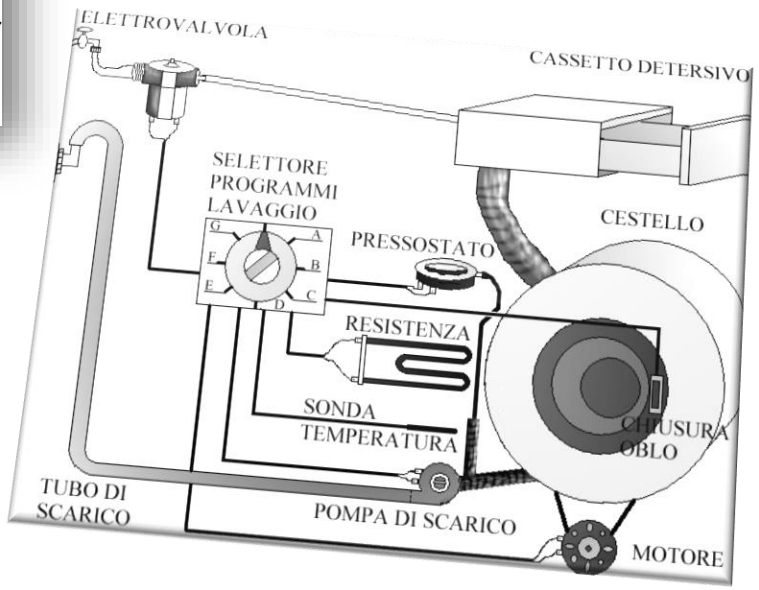


Esempio





FPGA

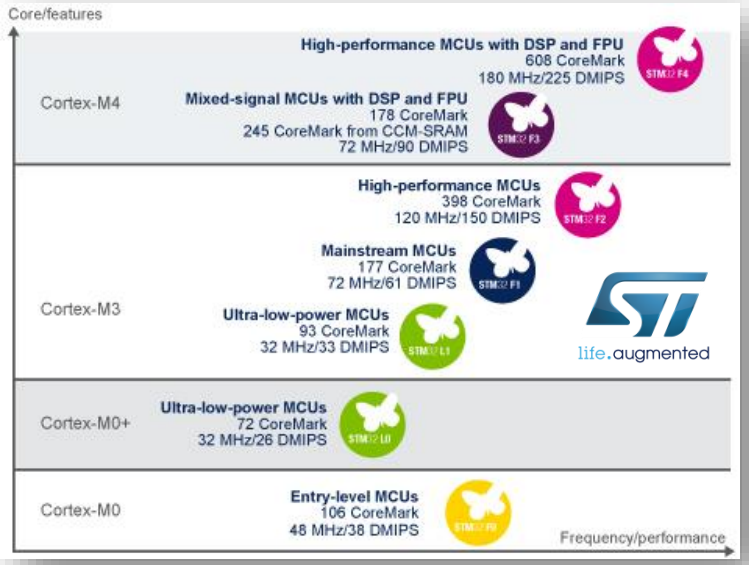


Uso dei micro

Un microcontrollore è il genere di computer in miniatura che potete trovare in ogni genere di oggetti. Se possiede dei tasti ed un display, è probabile che abbia anche un cervello a microcontrollore.

Provate a fare un elenco e contate quanti dispositivi con microcontrollore usate in una giornata tipo. Qui ci sono alcuni esempi: se al mattino la vostra radio si spenge, e premete più volte il tasto “snooze”, la prima cosa che fate nella vostra giornata è interagire con un microcontrollore. Riscaldare dei cibi nel forno a microonde oppure fare una telefonata con il cellulare, implica usare dei microcontrollori. E questo è solamente l’inizio. Qui ci sono alcuni altri esempi: accendere un televisore con il telecomando, giocare con un videogioco, usare una calcolatrice,ecc. Tutti questi dispositivi contengono dei microcontrollori che interagiscono con voi.

Prodotti sul mercato



The Example: Arduino

The Star Otto is the first Arduino board featuring the STM32F469 processor with WiFi.

The Arduino Star series of boards feature ST Microelectronics processors, and the first of the series is the STAR OTTO. The Otto combines the power of the STM32F469 processor and an Espressif ESP8266 for WiFi, with several on-board peripherals, such as a micro-SD slot, a connector LCD-Audio-Camera, an USB Host, an headphone and speaker output, and an on-board stereo microphone.

ADVANCED

ARM ESP8266 3.3V

Arduino Star OTTO

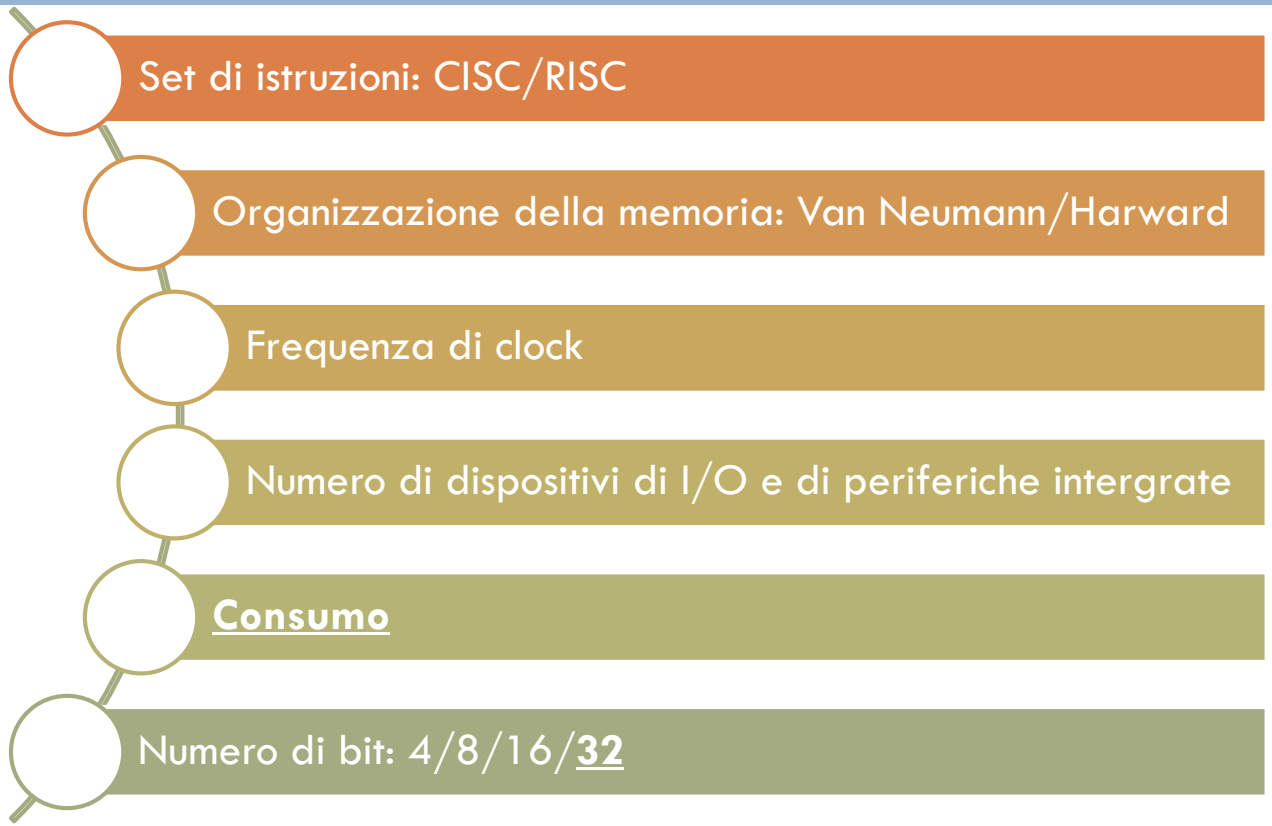
Microcontrollore: vantaggi

- L'integrazione delle periferiche su un singolo chip
- Minor numero di dispositivi discreti per la realizzazione di un sistema
- Dimensioni ridotte del sistema
- **Costi inferiori**
- Sistema nel complesso più affidabile
- Protezione dalle copiature
- **Risparmio energetico**
- Riprogrammabilità del sistema
- Comunicazione diretta con altri sistemi

Microcontrollore: svantaggi

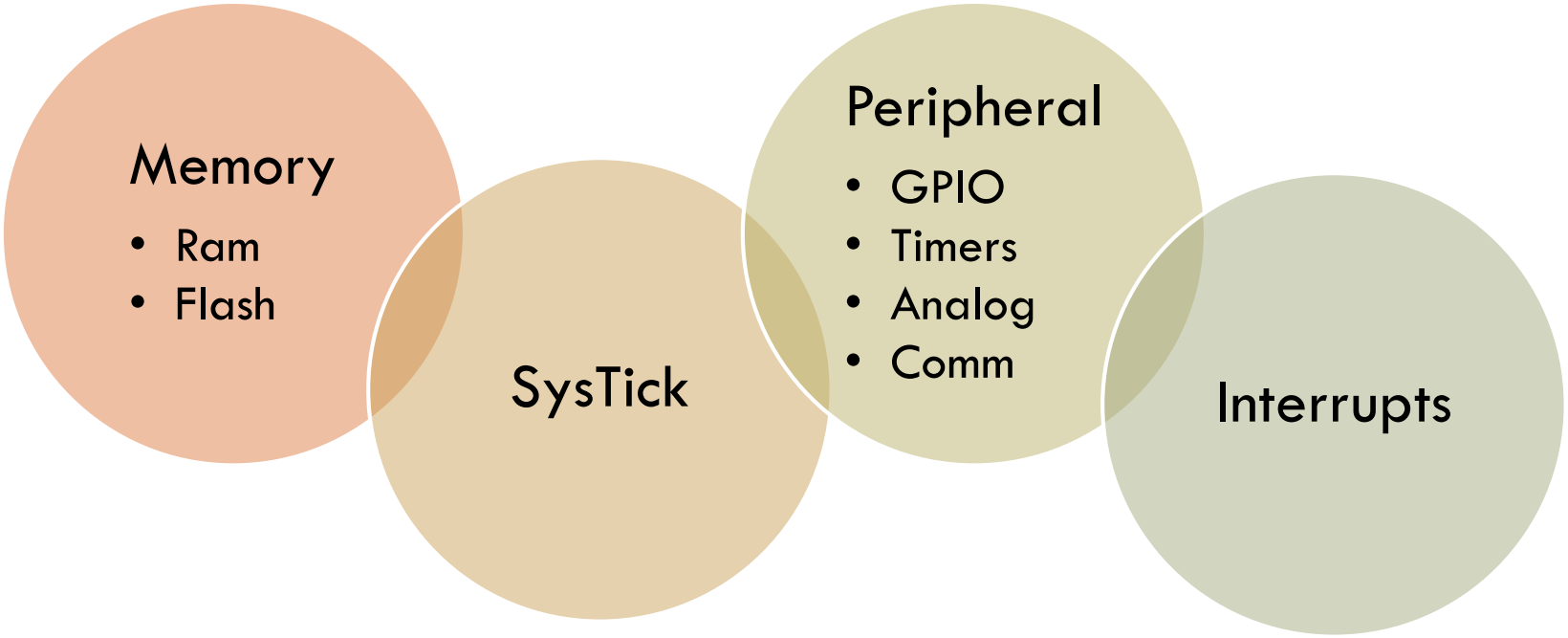
- Potenza di calcolo limitata (??????)
- Assenza di FPU (Floating Point Unit) in quelli low cost
- Gestione dati limitata
- RAM minima
- Cooperazione tra diversi MCU

Architettura



Il numero di bit rappresenta la dimensione massima del dato gestibile in un unico ciclo macchina. Rappresenta inoltre solitamente la dimensione dei singoli registri e del bus dati

Componenti


Memory
• Ram
• Flash
SysTick
Peripheral
• GPIO
• Timers
• Analog
• Comm
Interrupts

Memorie

Un microcontrollore è dotato di memoria interna. La memoria di un microcontrollore ha il compito principale di immagazzinare il programma che deve eseguire e quindi anche di immagazzinare i dati (le variabili) provenienti dall'esterno o elaborate dal programma. Generalmente vengono utilizzati microcontrollori che hanno una memoria di tipo Flash (perchè più pratica e riutilizzabile). Le memorie Flash sono una varietà di memoria EEPROM (o E2PROM) (Electrically Erasable and Programmable Read Only Memory): i dati vengono immagazzinati all'interno della memoria e lì rimangono anche in assenza di alimentazione elettrica (come le memorie delle macchine fotografiche o le pendrive insomma), quando si ha bisogno possono anche essere cancellati elettricamente.

I micro STM32 hanno un'architettura a memoria sezionata: vengono mantenute separate la memorizzazione dati e le istruzioni. Programmando nei linguaggi di alto livello non è necessario specificare il banco di memoria dal momento che queste istruzioni vengono aggiunte direttamente dal compilatore senza preoccupazione (questo è uno dei tanti vantaggi dei linguaggi ad alto livello).

Periferiche

GPIO: sono le linee di Input/Output, il microcontrollore è in grado di rilevare "stimoli" digitali provenienti dall'ambiente esterno (Input: pressione di un tasto, ricezione di un segnale digitale, chiusura di un contatto, ecc) oppure trasmetterli (Output: accensione di un LED, scritte su un display, ecc). Difatti tramite il programma che andremo ad inserire nel nostro microcontrollore, saremo in grado di dire ad esso quali pin (piedini) dovranno funzionare come ingressi (input) e quali come uscite (output).

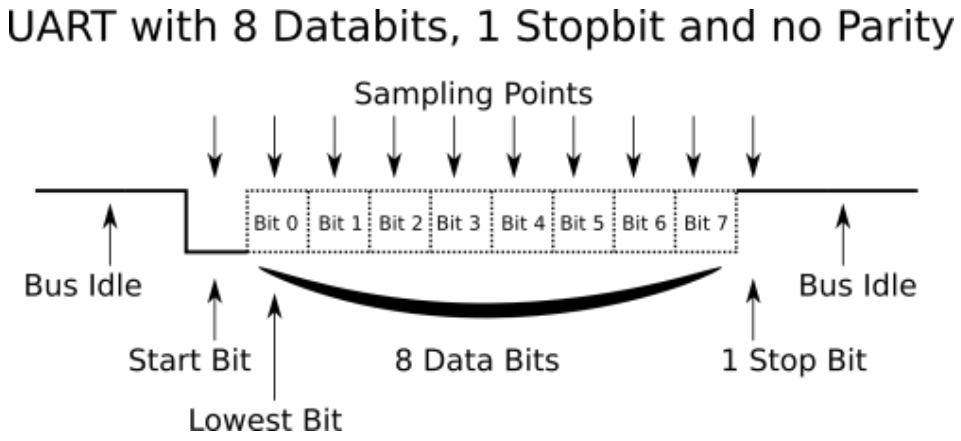
ADC o **DAC**: tramite questo tipo di periferiche siamo in grado di acquisire dall'esterno o generare verso l'esterno un segnale analogico (es: sonde di temperatura, sonde analogiche in generale ecc).

Timer: Il compito di queste periferiche è appunto quello di effettuare conteggi in maniera regolare e vengono utilizzati un po' per tutto: generare onde quadre, eseguire operazioni allo scadere di un tempo prefissato, realizzare applicazioni multitasking ecc.

Periferiche di comunicazione: tramite i bus di comunicazione siamo in grado di comunicare con altri dispositivi: USB, CAN, SPI, USART, I2C.

L'Usart ci permette di comunicare con la porta seriale RS232 del pc o di altri strumenti (e non solo), e quindi scambiare informazioni o dare istruzioni. Il protocollo di comunicazione CAN ad esempio è molto utilizzato sulle automobili per far comunicare tra loro i vari dispositivi. Il protocollo I2C è usatissimo per comunicare con piccoli dispositivi tipo chip realtime-clock (RTC), misuratori di distanza ad ultrasuoni, chipche permettono di espandere gli I/O ecc ecc.

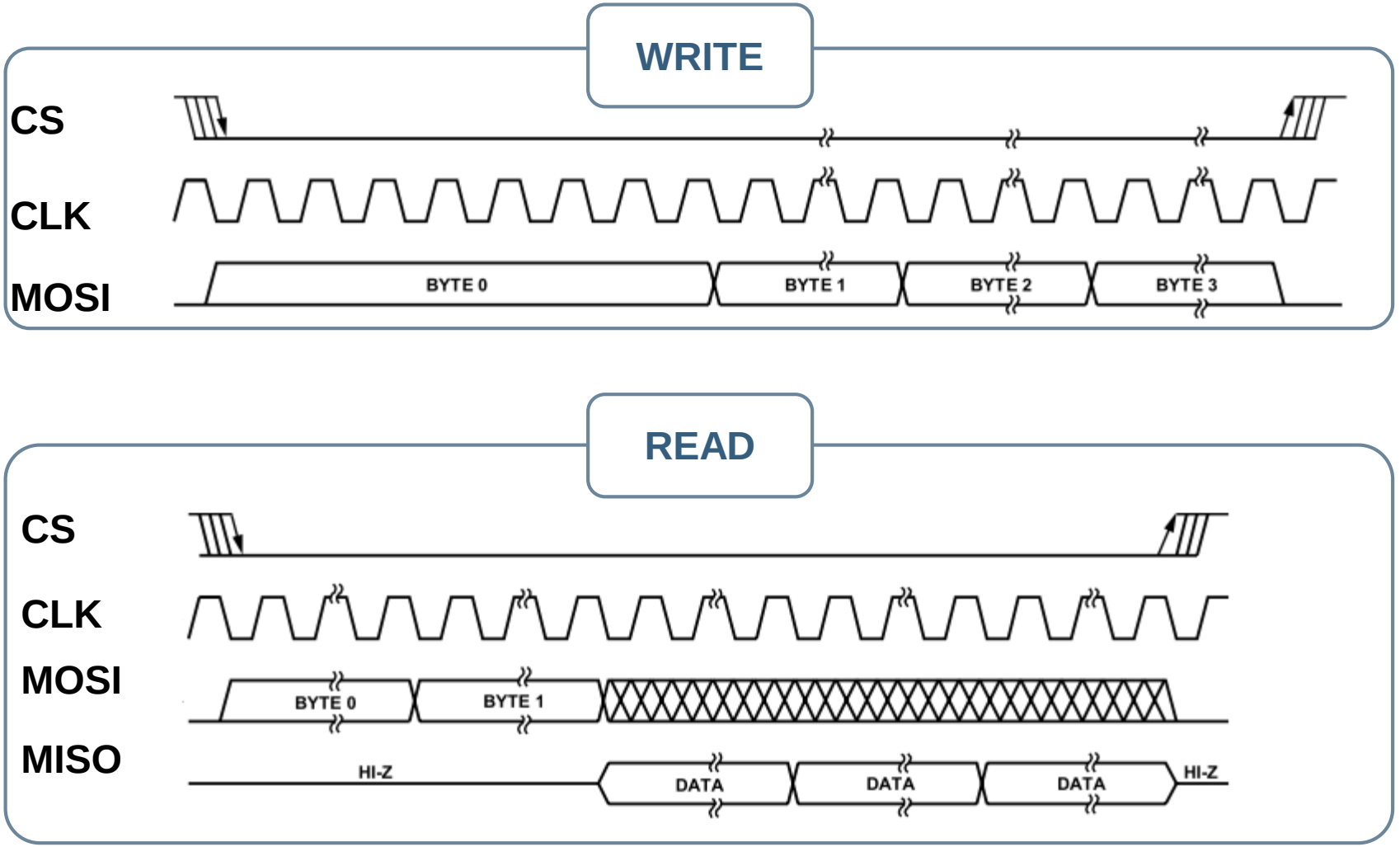
UART



Inviare un dato come testo o come byte

“48”	=>	‘4’, ‘8’	=>	0x34, 0x38
48	=>	0x30	=>	‘0’

SPI Signals


WRITE
CS
CLK
MOSI
BYTE 0
BYTE 1
BYTE 2
BYTE 3
READ
CS
CLK
MOSI
MISO
BYTE 0
BYTE 1
HI-Z
DATA
DATA
DATA
HI-Z

Interrupt

Definizione (Da wikipedia):

un segnale asincrono che indica il bisogno di attenzione oppure un evento sincrono che consente l'interruzione di un processo qualora si verifichino determinate condizioni.

Esempio:

Siamo a casa e abbiamo un telefono: per vedere se qualcuno ci sta chiamando è necessario che ogni tanto alziamo la cornetta? No! Il telefono squilla avvisandoci che qualcuno ci sta chiamando, quindi alzeremo la cornetta solo quando sta squillando.

Immaginate: siamo a casa ad effettuare le nostre faccende, non abbiamo bisogno di alzare ogni tanto la cornetta, perché il telefono ci avviserà con uno squillo, consentendoci di interrompere ciò che stiamo facendo e dedicarci quindi alla telefonata.

Concetto chiave:

lo squillo del telefono è il nostro interrupt: ci consente di interrompere momentaneamente ciò che stavamo facendo per dedicarci alla situazione che ha generato l'interrupt fino a quando non decidiamo di riprendere le nostre faccende nel punto in cui le avevamo rimaste.

Interrupt : azioni

Il programma principale viene interrotto al verificasi dell'evento che fa scaturire un interrupt

→

Viene salvato il valore numerico del PC (Program Counter, che è un contatore interno del micro che tiene conto di quale riga di programma sta eseguendo) nello Stack (che è un registro apposito per memorizzare tale valore).

→

Il micro salta ad una determinata locazione di memoria (denominata Interrupt Vector) nella quale vengono appunto memorizzate le istruzioni da eseguire in caso di interrupt.

↓

Il micro esegue le istruzioni contenute nell'interrupt vector.

→

Finito di eseguire tali istruzioni, il micro preleva dallo Stack la riga del programma principale in cui si era fermato.

→

Il programma principale riprende dal punto in cui era stato interrotto dall'interrupt.

Nei registri di delle periferiche ci sono dei bit che permettono di attivare o disattivare un determinato evento di interrupt; e dei bit che ci avvisano se si è verificato questo o quel determinato tipo di interrupt. e vengono settati dal micro quando si verifica l'evento di interrupt e devono essere resettati da noi via software.

Piattaforme ARM : Core Cortex-Mx

ST has licensed all Cortex®-M cores

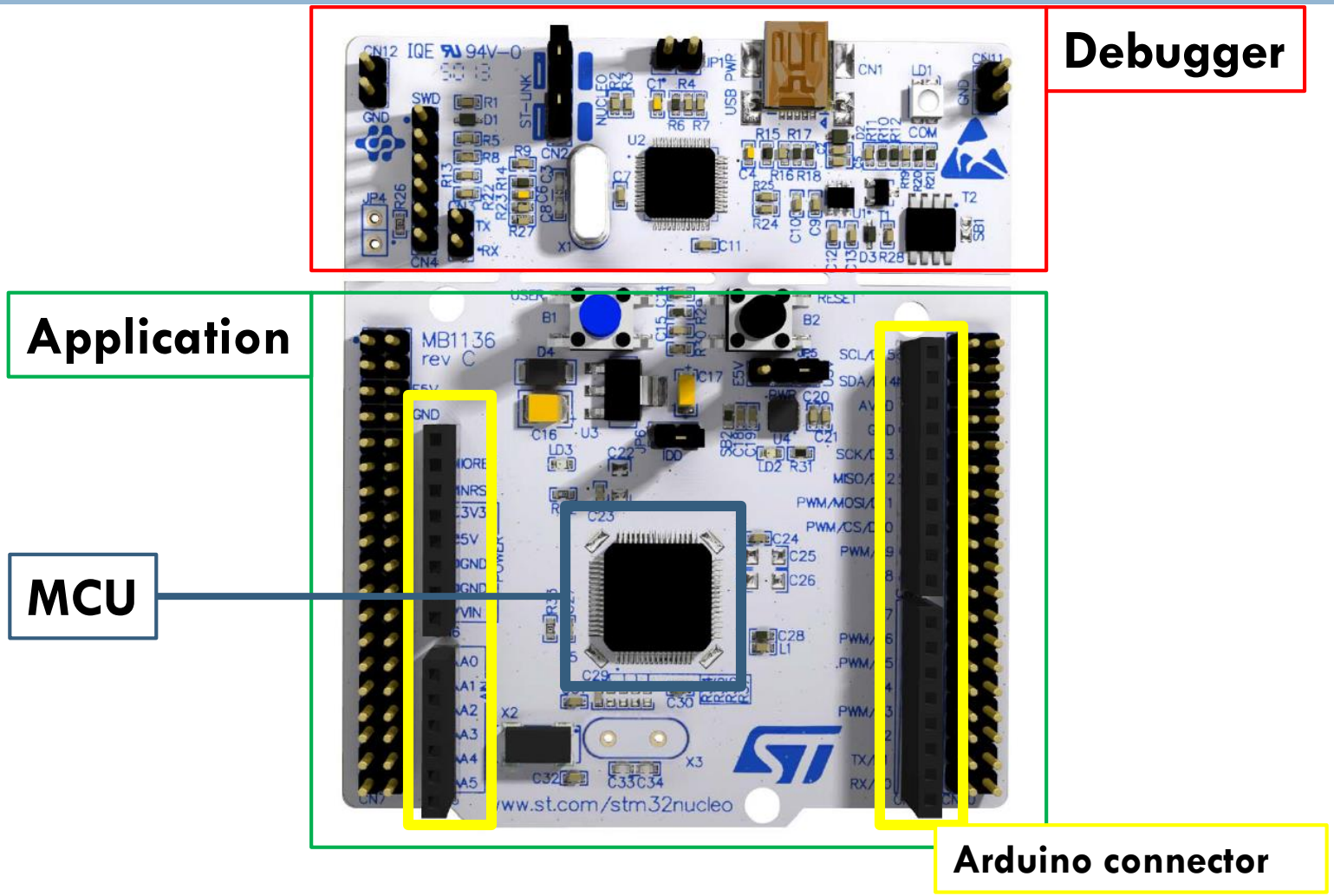
- **Forget traditional 8/16/32-bit classifications and get**
  - Seamless architecture across all applications
  - Every product optimized for ultra-low power and ease of use

Cortex-M0/M0+	Cortex-M3	Cortex-M4	Cortex-M7
8/16-bit applications	16/32-bit applications	16/32-bit DSC applications	16/32-bit DSC applications

Binary and tool compatible

life.augmented

STM32 NUCLEO Platform



Passi

Configuratore: STM32CubeMX

Ambiente di Sviluppo (IDE) : IAR Workbench

Compilatore (Integrato in IAR)

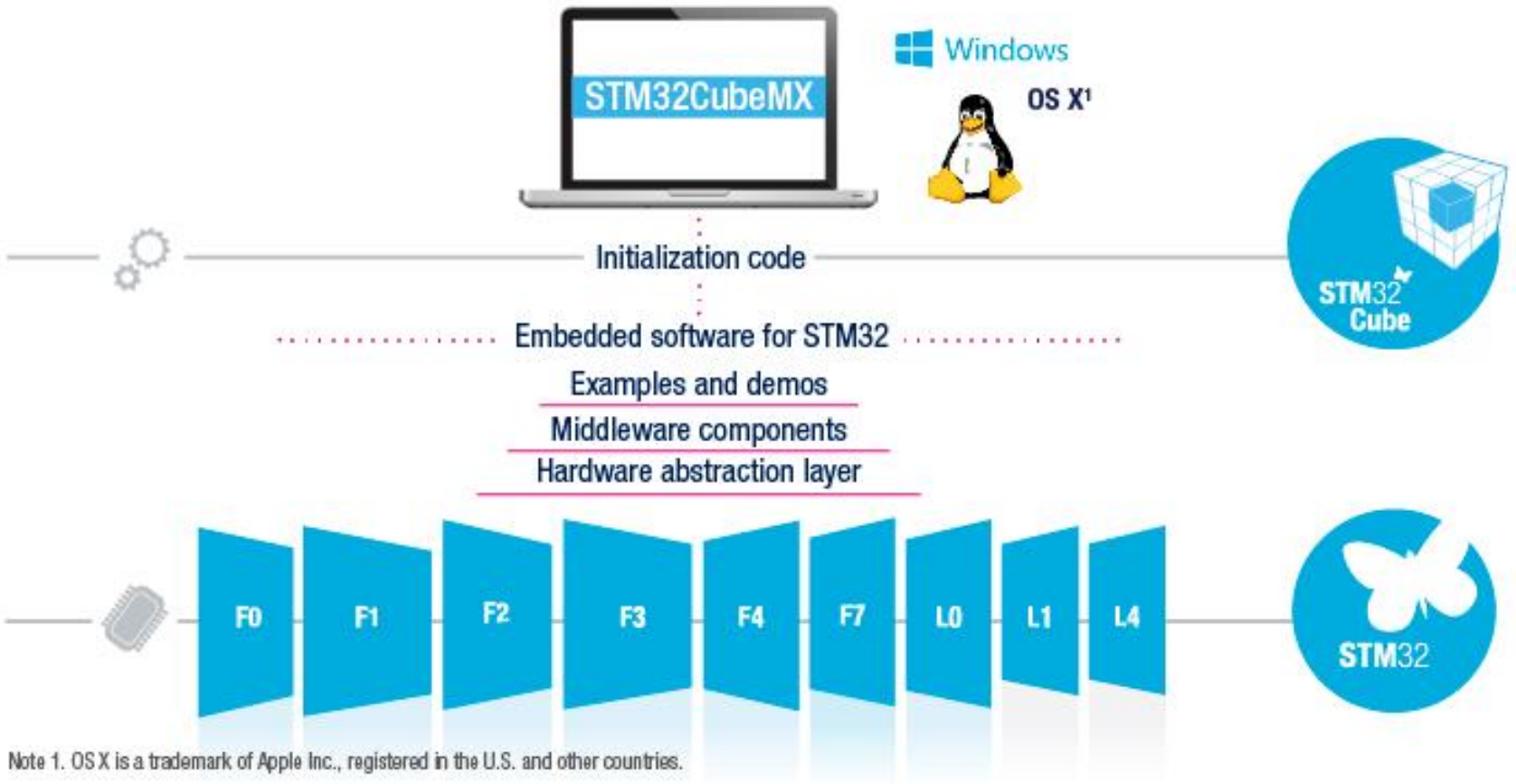
Flash loader (Integrato in IAR WB)

Debugger : ST-Link 2.1 (Nucleo)

FIRMWARE

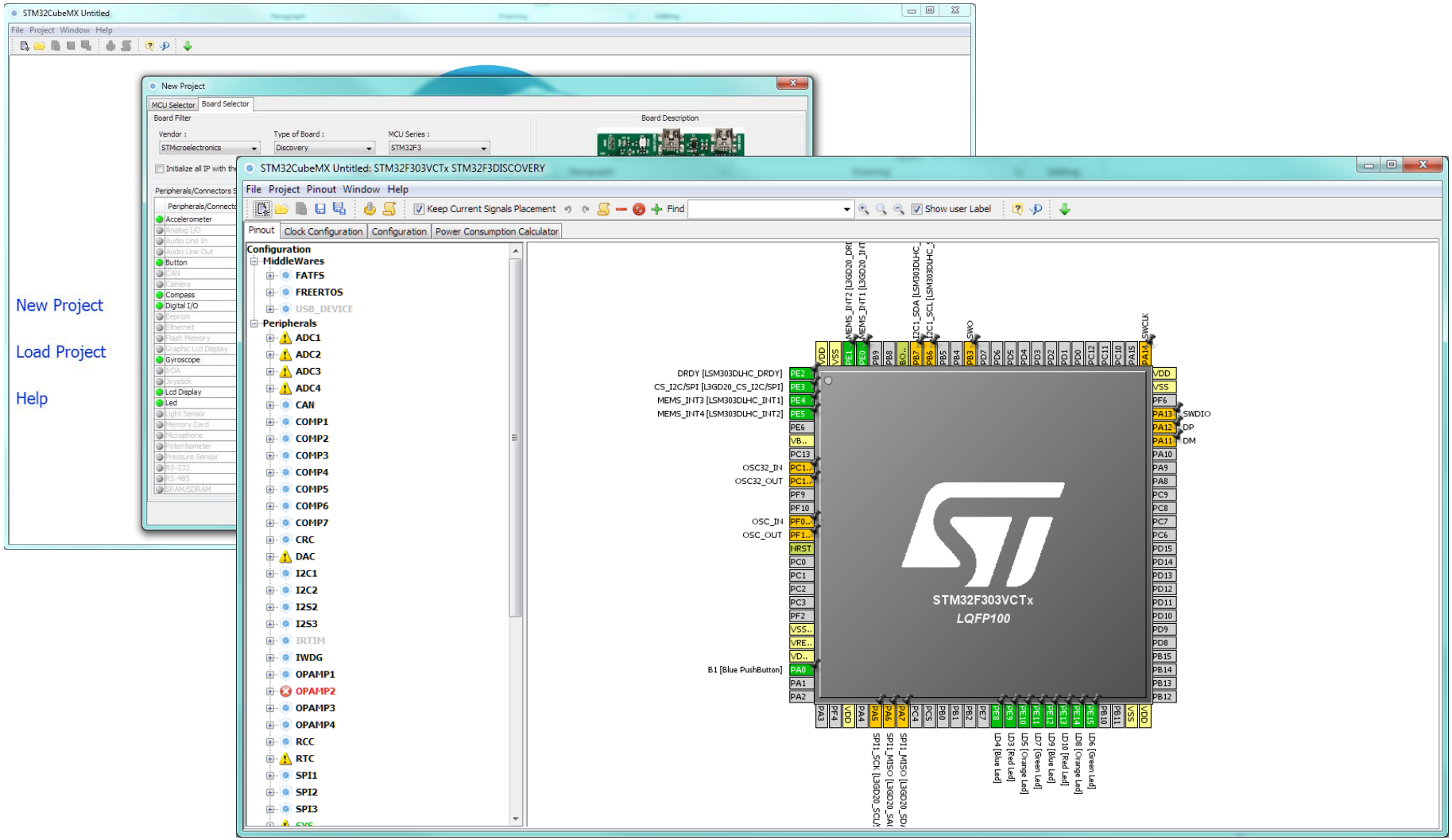
Configuratore : STM32CubeMX

Ver 4.25.0

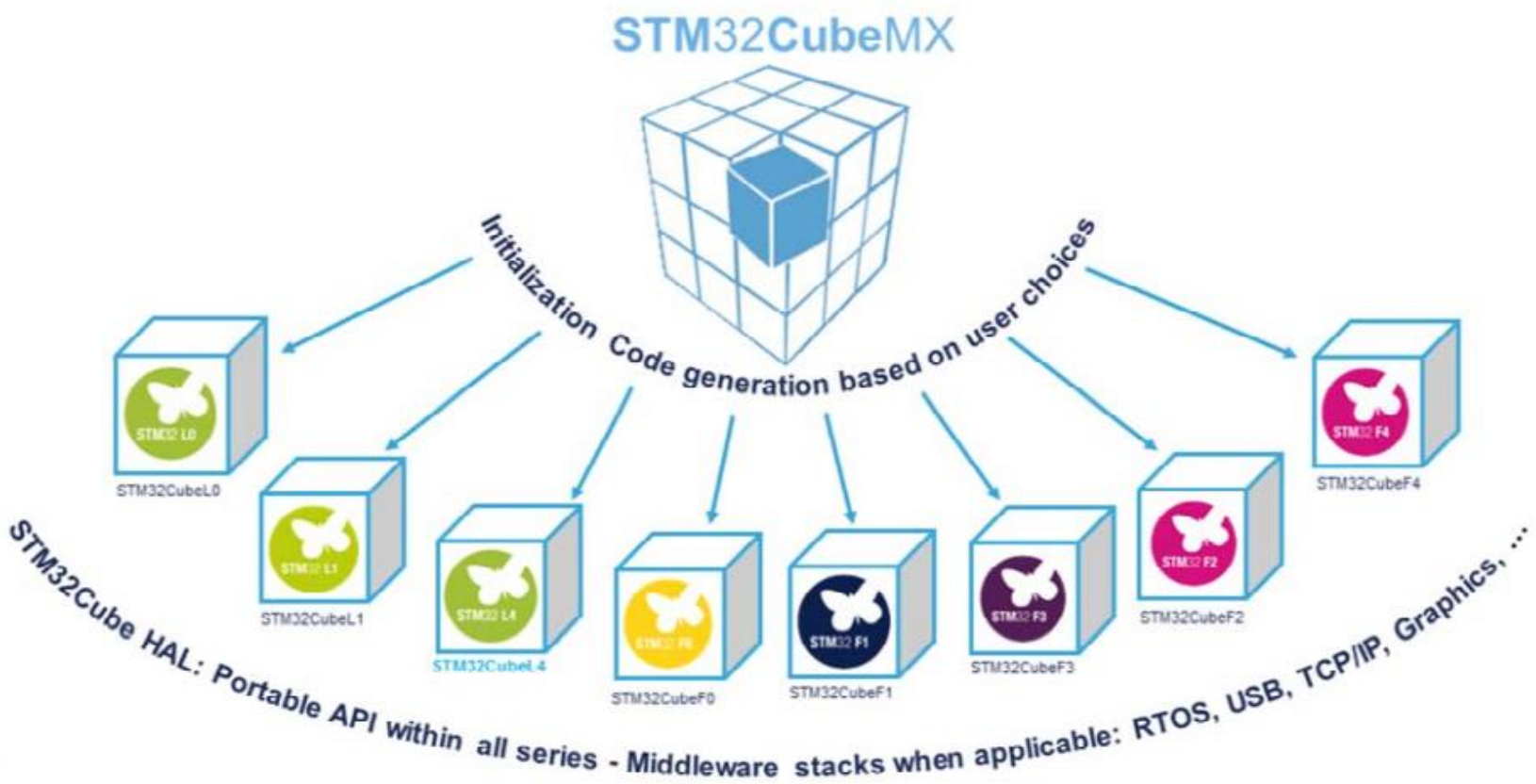


Java based

Configuratore : STM32CubeMX



STM32 HAL

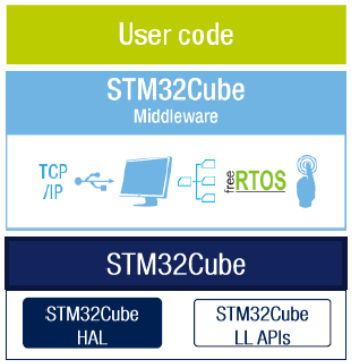

STM32CubeMX
Initialization Code generation based on user choices
STM32CubeL0
STM32CubeL1
STM32CubeL4
STM32CubeF0
STM32CubeF1
STM32CubeF3
STM32CubeF2
STM32CubeF4
STM32Cube HAL: Portable API within all series - Middleware stacks when applicable: RTOS, USB, TCP/IP, Graphics, …

STM32 HAL

STM32Cube - Embedded software

HAL APIs

- Features:
  - High level and functional abstraction
  - Easy port from one series to another
  - 100% coverage of all peripherals
  - Integrates complex middleware such as USB/TCP-IP/Graphics/Touch Sense/RTOS
  - Can work with STM32CubeMX tool on the PC to generate initialization code



- Limitations:
  - May be challenging to low level C programmers in the embedded space.
  - Higher portability creates bigger software footprints or more time spent executing adaptation code

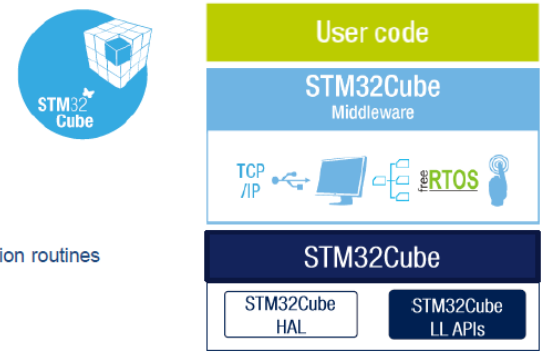
Portability	Optimization (Memory & Mips)	Easy	Readiness	Hardware coverage
+++	+	++	+++	+++

STM32 HAL

STM32Cube - Embedded software

Low-Layer APIs

- Features:
  - Highly Optimized
  - Register Level Access
  - Small code expressions
  - Closely follows the reference manual
  - Debugging close to register level
  - Peripheral block initialization APIs
    - Initialization, de-initialization and default initialization routines
    - SPL-Like functionally speaking
    - More optimized than SPL, fitting lots of situations
    - No need for direct register manipulation
    - Easier debugging of procedural code



- Limitations:
  - Specific to STM32 devices, not portable directly between series
  - Not matching complex peripherals such as USB
  - Lack of abstraction for runtime means developers must understand peripheral operation at register level
  - Available on STM32L4, L0 and F0 series
  - Peripheral block initialization APIs have the same limitations as the SPLs (except availability considerations)

Portability	Optimization (Memory & Mips)	Easy	Readiness	Hardware coverage
+	+++	+	++	++

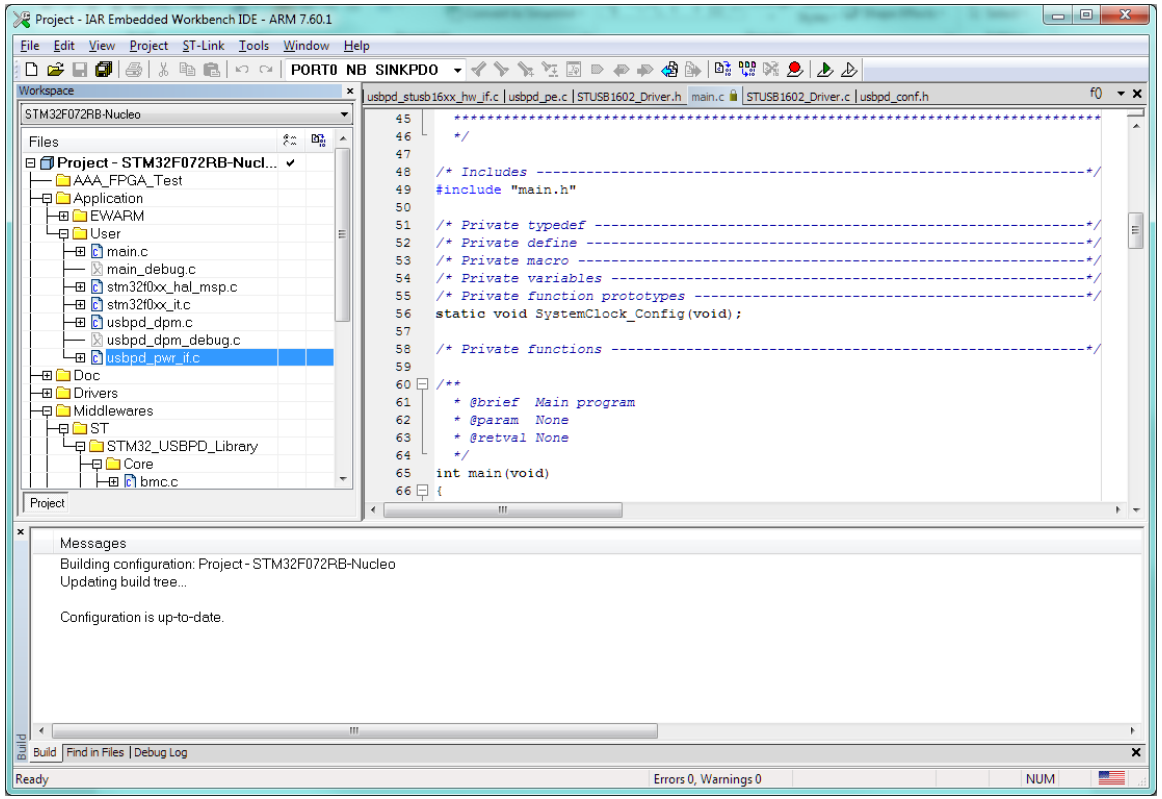
life.augmented

IDE : IAR Workbench

8.10.1

Feature:

- Breakpoints
- Watch / Live watch
- Registers
- Output view
- Memory
- Disassembly



Debugger : Integrated ST-Link

Joint Test Action Group (JTAG)
Serial Wire Debug (SWD)