POLLING E INTERRUPT

CON RIFERIMENTO AD UN CORE ARM CORTEX M

FONTI

- PM0214 Programming manual. STM32 Cortex®-M4 MCUs and MPUs programming manual
- Cerri, Ortolani, Venturi. Nuovo Corso di sistemi automatici vol.2. Hoepli
- https://it.emcelettronica.com/corso-embedded-arm-sistemi-operativi-real-time

INTRODUZIONE

Una periferica o un dispositivo facente parte del sistema coordinato dal microcontrollore può lanciare a esso una richiesta di servizio.

Per esempio una porta UART può avvertire che è arrivata una sequenza di caratteri.

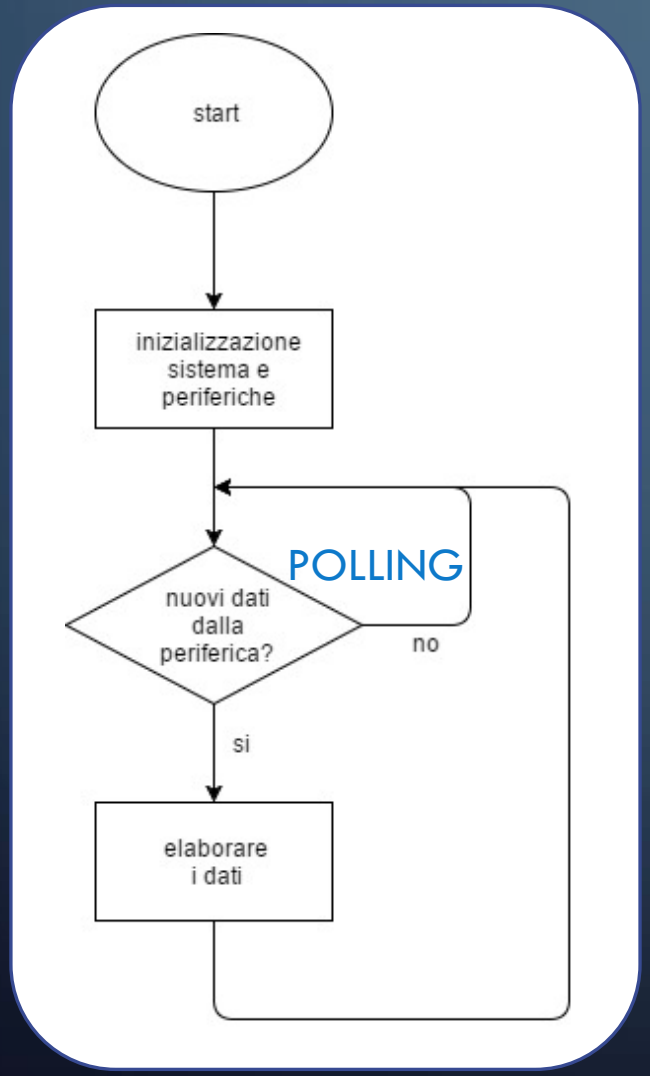
L'istante in cui avviene questa richiesta è del tutto casuale e il microcontrollore deve interrompere la normale sequenza di istruzioni per rispondere.

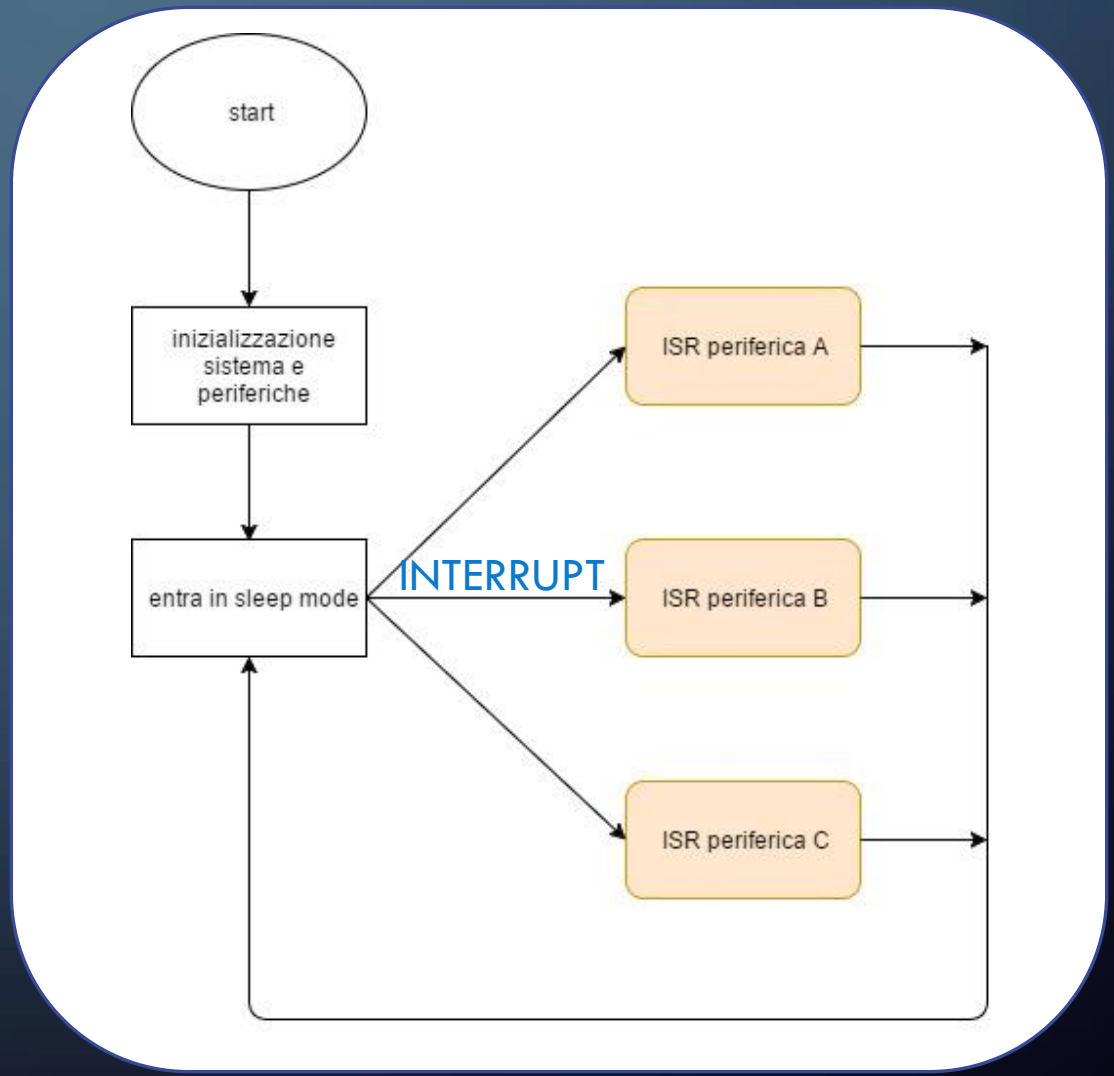
Le richieste di servizio possono essere gestite in due modi

- polling
- interrupt

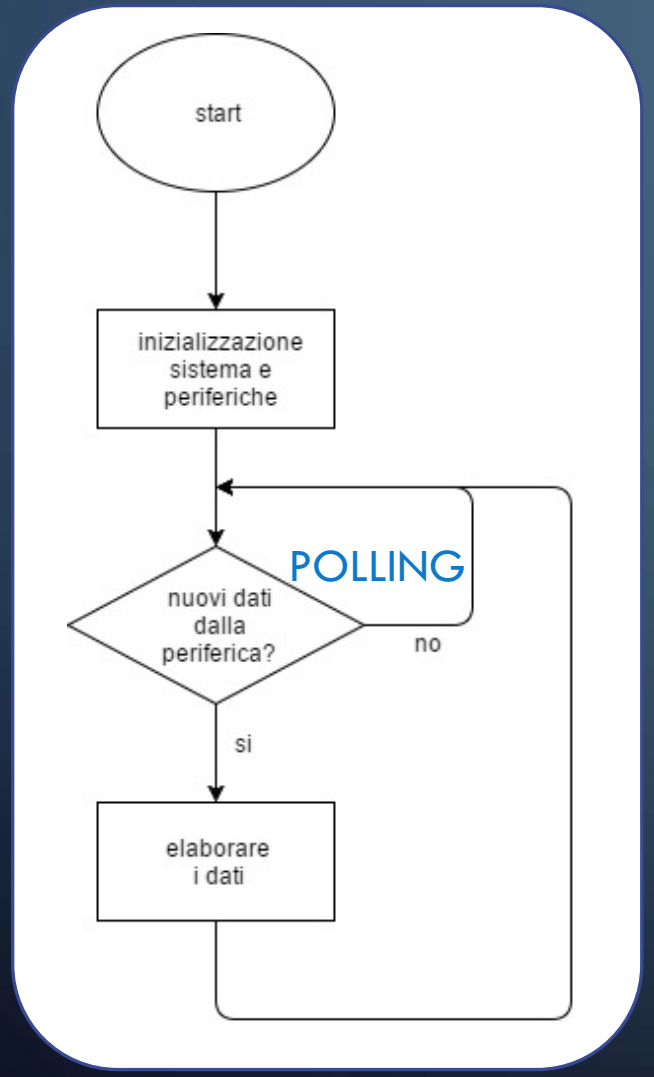
POLLING E INTERRUPT

Esistono differenti modi di gestire le richieste di una periferica:



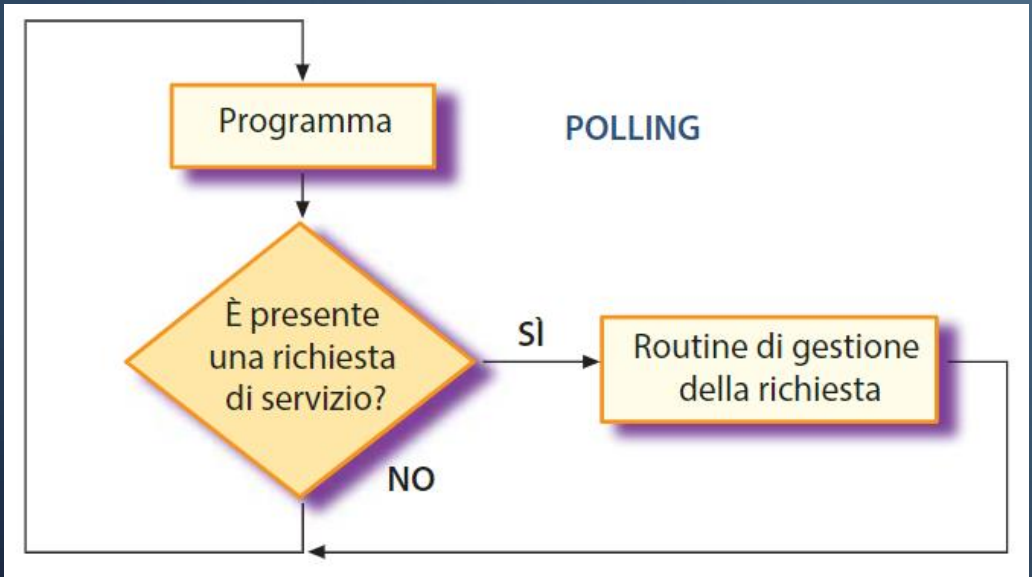


POLLING (1/3)



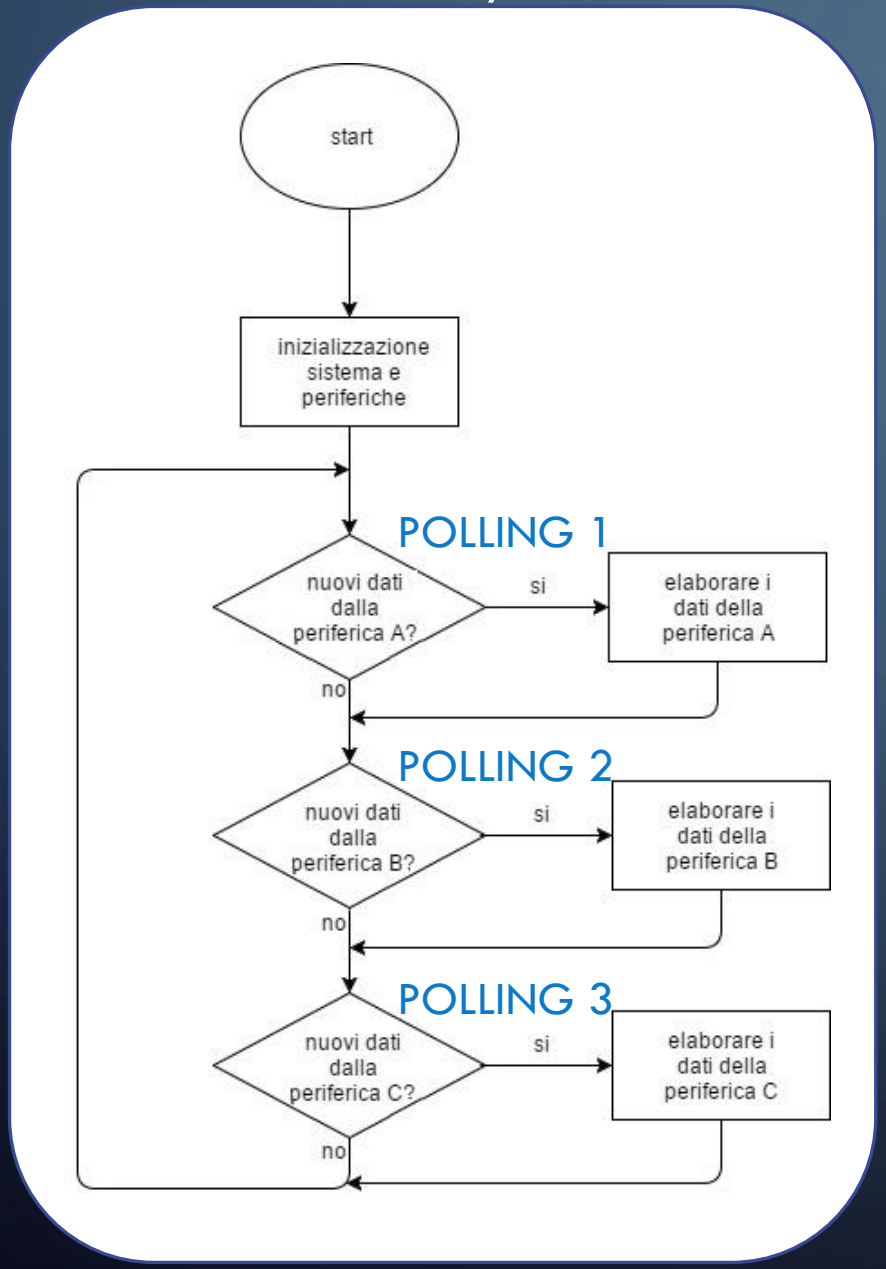
- Con il polling (dall'inglese “to poll”, sondare, monitorare), la CPU controlla periodicamente se la periferica ha bisogno di essere servita.
- Qualora vi sia una richiesta di servizio pendente, questa viene soddisfatta, in caso contrario il programma prosegue senza interruzioni

POLLING (2/3)



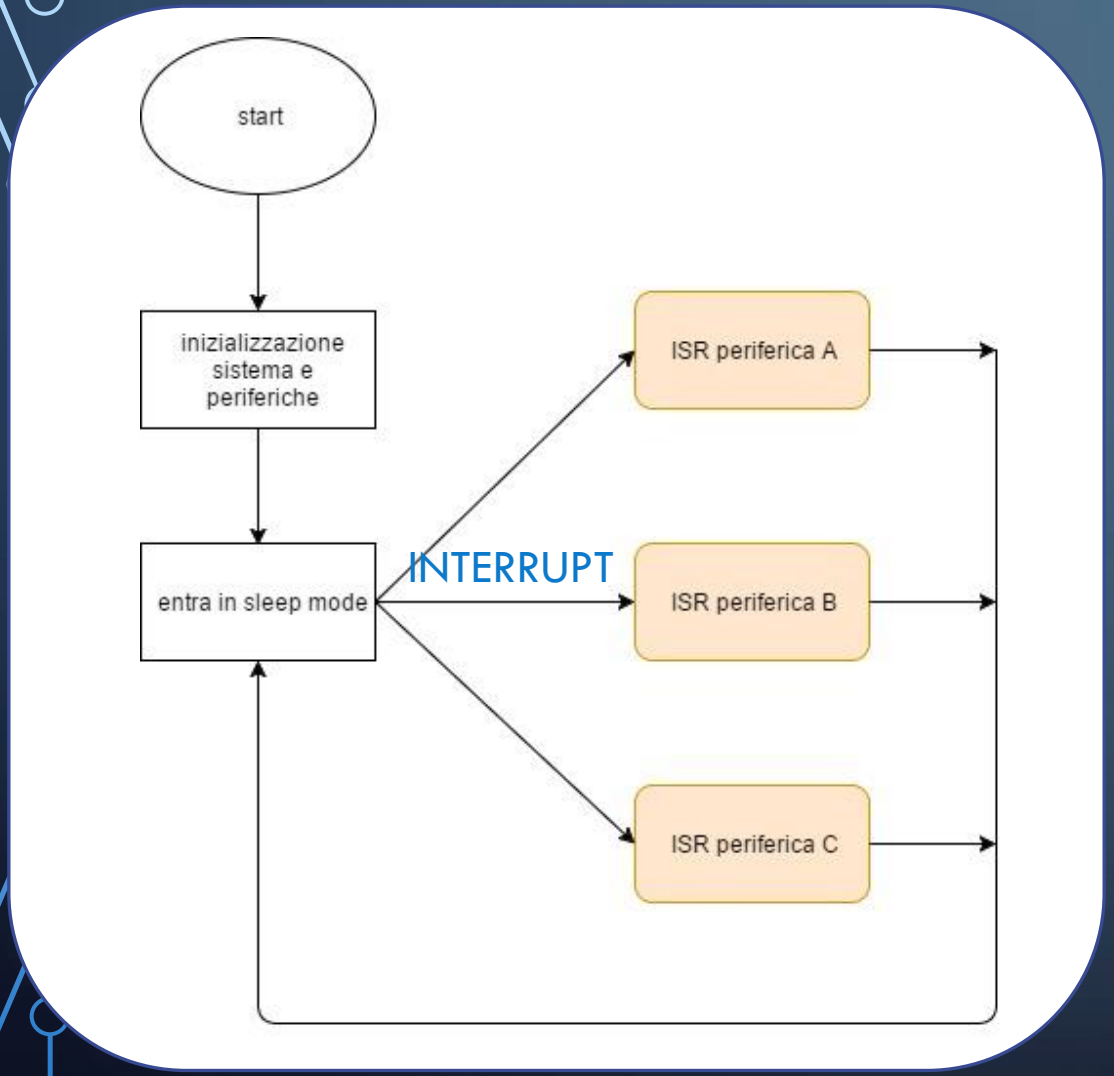
- La conseguenza diretta del polling è che la sospensione della normale esecuzione del programma per soddisfare la richiesta della periferica causa una dilatazione dei tempi di esecuzione
- L'allungamento dei tempi dipende dalla complessità della routine di gestione e da quanto spesso questa viene eseguita

IL POLLING (3/3)



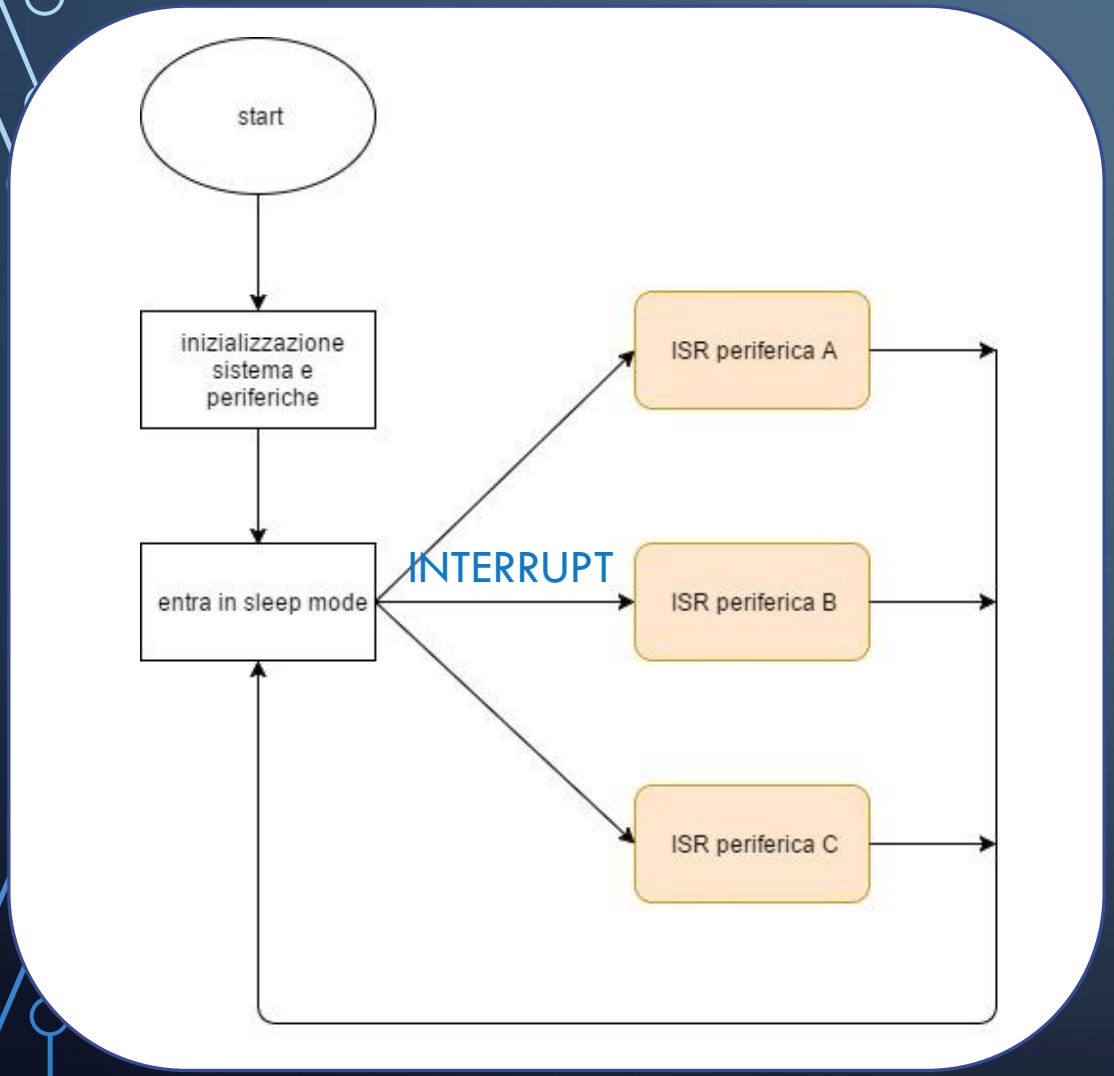
- In molti casi il microcontrollore deve gestire più periferiche
- Il polling può facilmente essere esteso per gestire più processi, in un approccio che viene definito super-loop, in cui ogni periferica viene interrogata in sequenza
- Se le periferiche aumentano l'allungamento dei tempi diventa sempre più imprevedibile
- Inoltre non è facile stabilire un ordine di priorità tra periferiche; quelle più veloci finiscono per attendere lunghi periodi di tempo mentre il processore ne sta servendo altre e questo può essere inaccettabile

INTERRUPT (1/3)



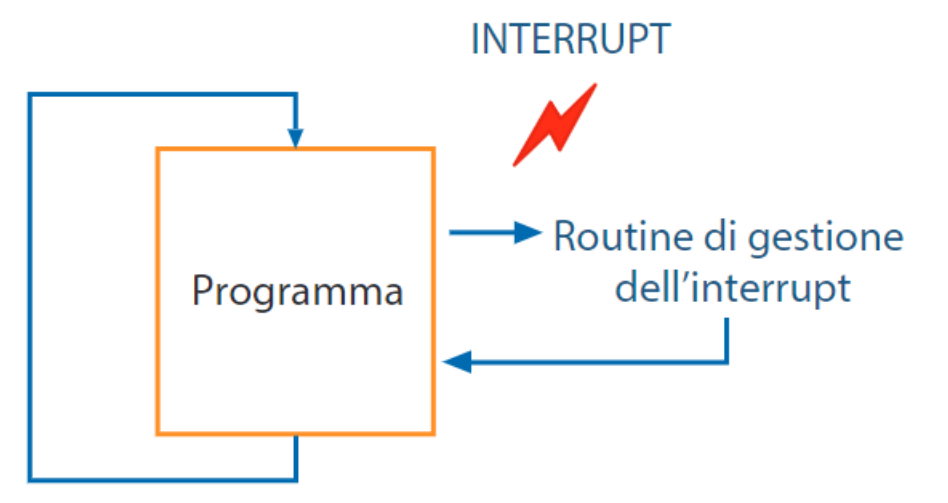
- Gli interrupt sono generati dalle periferiche che richiedono il servizio (le periferiche devono essere in grado di generare una richiesta di interrupt, quindi l'hardware è più complesso)
- Se una periferica genera un interrupt la sua richiesta viene soddisfatta dal microprocessore (interrupt driven) elaborando la relativa ISR (interrupt service routine)
- Finita l'esecuzione della routine il microprocessore ritorna a svolgere il programma principale dal punto in cui lo aveva interrotto

INTERRUPT (2/3)



- In assenza di dati da elaborare il processore può entrare in uno stato di risparmio energetico (sleep mode), dal quale viene risvegliato solo dall'evento di interrupt
- Gli interrupt possono essere inoltre configurati con diversi livelli di priorità in modo che vengano servite prima le periferiche più veloci
- La risposta ad un interrupt NON è come l'esecuzione di una funzione. Nel caso di funzioni il passaggio a queste è predeterminato da una chiamata in un ben definita riga di codice. L'interrupt invece è un evento casuale generato dalle periferiche che quindi può avvenire in qualsiasi punto del programma

INTERRUPT (3/3)



La gestione dell'interrupt comprende 3 fasi:

1. la periferica lancia la richiesta di servizio
2. la richiesta viene ricevuta dalla CPU
3. la CPU attiva la routine di gestione dell'interrupt

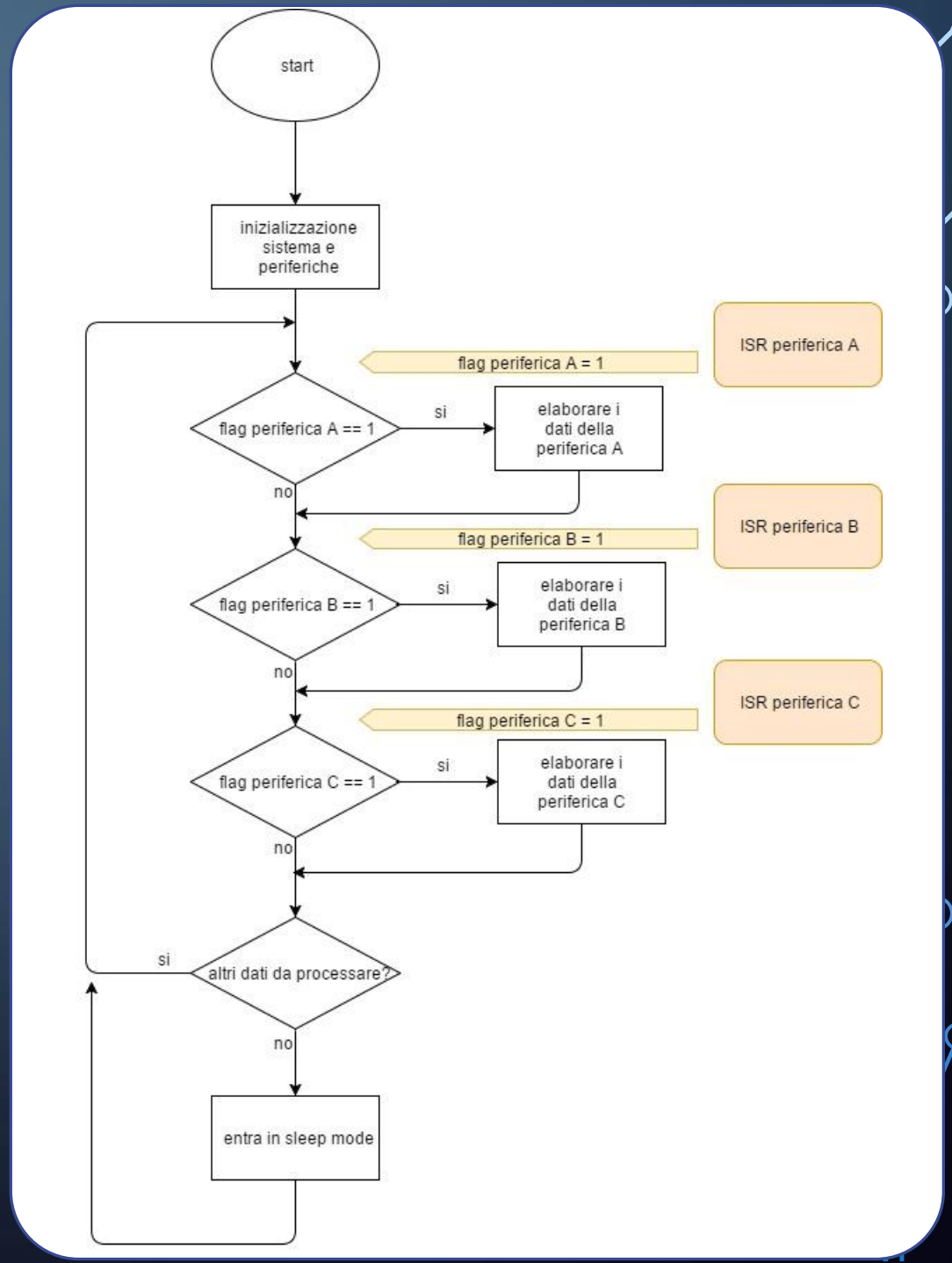
TECNICA MISTA

È possibile rispondere alle interruzioni in due tempi:

- la prima deve essere elaborata rapidamente
- la seconda può essere elaborata in un secondo momento

In queste situazioni si adotta un approccio misto, polling e interrupt driven, per gestire il flusso dell'applicazione: quando una periferica richiede attenzione genera un interrupt come in un'applicazione interrupt driven. L'ISR aggiorna una variabile (globale e volatile) di stato del programma in maniera tale che la seconda parte dell'elaborazione possa essere eseguita nel loop principale in approccio polling

In questo modo si può migliorare la velocità di risposta dell'applicazione, e il processore può comunque entrare in uno stato di basso consumo quando le periferiche non richiedono un servizio

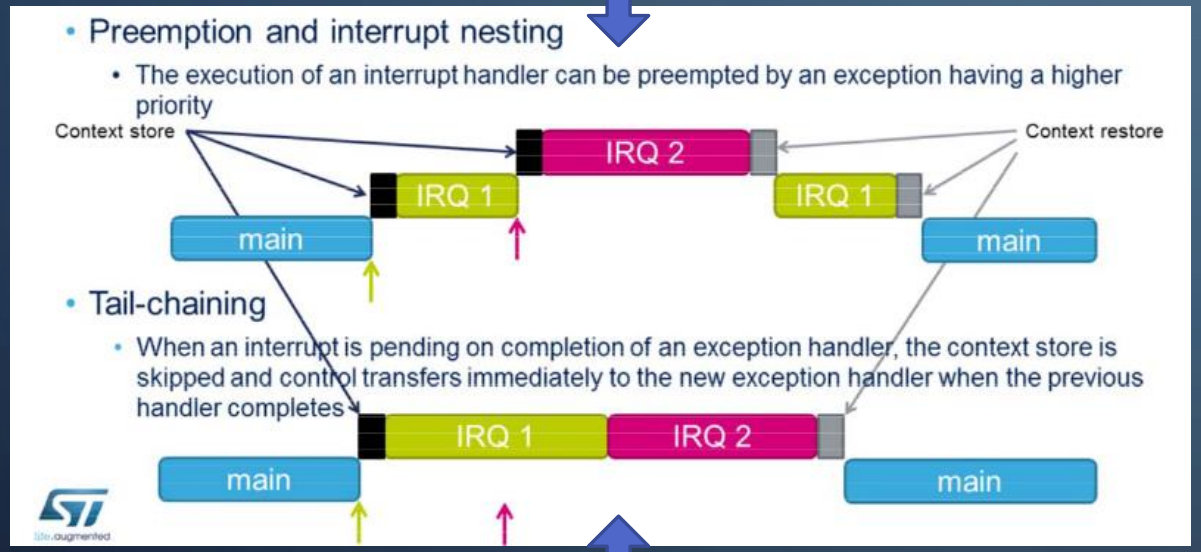


INTERRUZIONI NEL CORE ARM CORTEX M (NIVC)

settable	SPI1	SPI1 global interrupt
settable	SPI2	SPI2 global interrupt
settable	USART1	USART1 global interrupt
settable	USART2	USART2 global interrupt
settable	USART3	USART3 global interrupt
settable	EXTI15_10	EXTI Line[15:10] interrupts
settable	RTC_ALARM	RTC alarms through EXTI line 18 interrupts
settable	DFSDM3	DFSDM3 global interrupt
settable	TIM8_BRK	TIM8 Break interrupt
settable	TIM8_UP	TIM8 Update interrupt
settable	TIM8_TRG_COM	TIM8 trigger and commutation interrupt

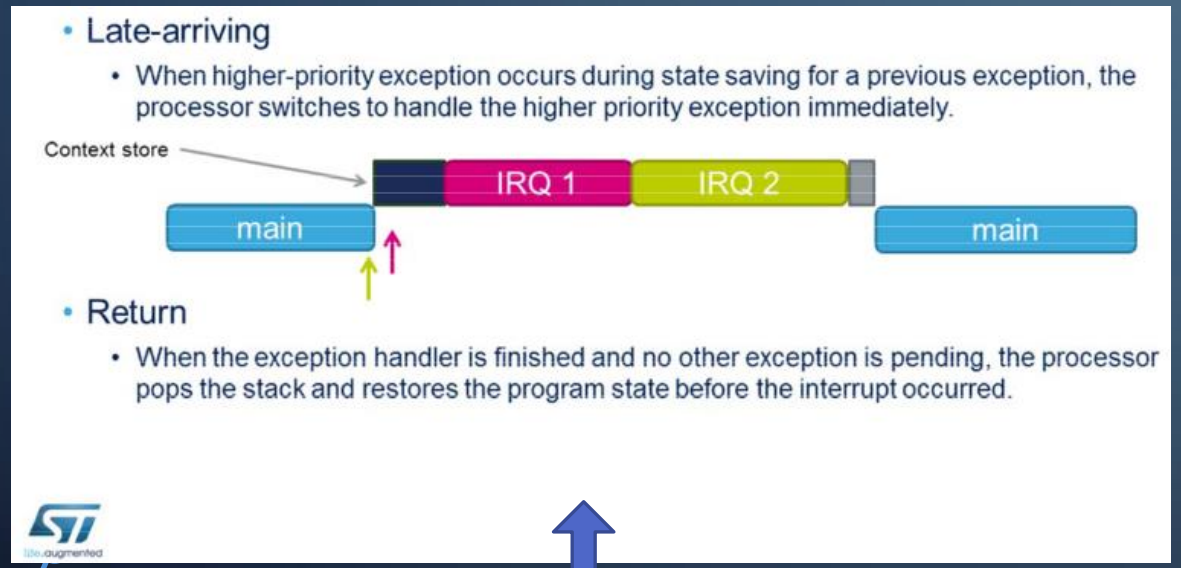
Nelle CPU ARM Cortex M è implementato un meccanismo di gestione degli interrupt chiamato “Nested Vectored Interrupt Control” (NVIC) che gestisce fino a 91sorgenti di interrupt (chiamati channel interrupt) con 16 livelli di priorità impostabili singolarmente per ogni sorgente di interrupt

GESTIONE DELLE PRIORITÀ CON NIVC (1/2)



- Quando un interrupt a priorità più alta arriva mentre si sta servendo un interrupt a priorità più bassa, quest'ultimo viene interrotto e anticipato (preemped) da quello nuovo
- Quando un interrupt a priorità più bassa arriva mentre si sta servendo un interrupt a priorità più alta, questo viene messo in coda (tail-chaned) e servito non appena finisce la ISR dell'interrupt a priorità maggiore

GESTIONE DELLE PRIORITÀ CON NIVC (2/2)



- Quando arriva un interrupt, prima di servire la ISR, la CPU salva il contesto in cui si trova (cioè salva i dati per poter ripartire a svolgere il programma principale dal punto in cui lo ha interrotto). Se durante l'operazione di salvataggio arriva un interrupt a priorità maggiore (late-arriving), questo viene servito immediatamente e quello a priorità minore verrà servito in coda
- Nel caso in cui non vi siano più interrupt pendenti la CPU torna a svolgere il programma principale dal punto in cui lo aveva interrotto