Programmazione del Microcontrollore
STM32F3: il nostro primo driver

Luigi Coppolino, Giovanni Mazzeo

Outline

- Come si programma una qualsiasi periferica/interfaccia di un microcontrollore
- Un esempio: I GPIO
  - Cosa sono e a cosa servono i GPIO
  - I GPIO sulla STM32F3-Discovery
  - Come funzionano ed esempi di Registri GPIO
- Organizzazione tipica di un progetto per lo sviluppo di un driver in Eclipse
- Librerie a supporto del programmatore: CMSIS e ST-HAL
- Approfondimento del progetto BlinkLed
- Sviluppo di un estensione al progetto BlinkLed

Programmare una periferica #1

- Ogni periferica o interfaccia di I/O possiede un set dedicato di registri mappati in memoria RAM nel quale scrivere/leggere per programmarne il suo funzionamento o controllarne il suo stato.
- Sono tre le categorie di registri fondamentali che una periferica può avere:
  - **Registri di Stato** – Un insieme di flag da poter leggere per conoscere lo stato della periferica
  - **Registri di Controllo** – Registri da dover scrivere per impostare il funzionamento desiderato della periferica
  - **Registri di Dato** – Un registro dal quale potremmo o leggere o scriver dati di interesse
- La mappatura dei registri è presente nei *reference manual* dei microcontrollori

Programmare una periferica #2

- Esempio semplificato di un timer TIM
- Vogliamo programmare un timer affinché conti in modo decrescente dal valore 100 al valore 0.
- Per fare ciò, dunque, dovremo:
  - Scrivere nel registro di controllo la tipologia di conteggio che si desidera (decrescente), se si vuole che il conteggio ricominci una volta terminato, ecc.
  - Scrivere nel counter register (registro di dato) il valore di partenza (100)
  - Avviare il timer scrivendo nel registro di controllo
  - Periodicamente leggere il registro di stato per controllare se il timer è arrivato al valore 0

i GPIO

- I GPIO sono dei pin generici configurabili dall'utente a *run-time*
- Permettono la comunicazione del microcontrollore con il mondo esterno sia in uscita che in ingresso
- Grazie ai GPIO è possibile avere un numero elevato di unità/periferiche in un microcontrollore mantenendo ridotto il numero di pin
- Possono funzionare a 4 differenti velocità di I/O: 5, 25, 50, 100 MHz

I GPIO sulla STM32F3 - 1/2

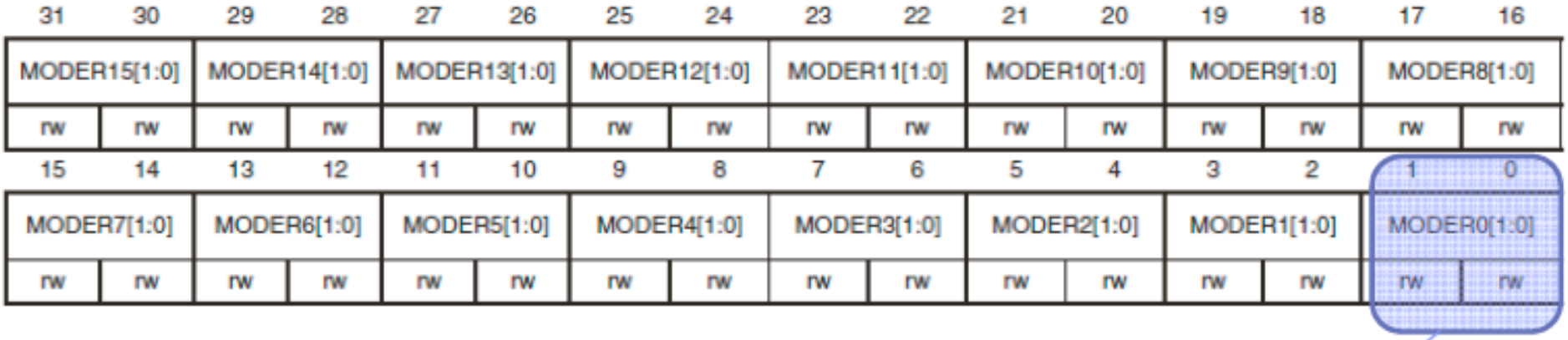
- Il microcontrollore sulla scheda STM32F3-Discovery è provvisto di 8 porti GPIO (definiti dalla A alla H).
- Ogni porto può gestire sino a 16 pins => 16 x 8 = 128 pins totali
- Ogni porto GPIO, inoltre, ha associato un insieme di registri (x = {A…H}) che fanno parte del suo modello di programmazione:
- 4 Registri di Configurazione da 32 bit
  - GPIO port mode register (GPIOx_MODER)
  - GPIO port output type register (GPIOx_OTYPER)
  - GPIO port output speed register (GPIOx_OSPEEDR)
  - GPIO port pull-up/pull-down register (GPIOx_PUPDR)

I GPIO sulla STM32F3 - 2/2

- 2 Registri di Dato da 32 bit
  - GPIO port input data register (GPIOx _IDR)
  - GPIO port output data register (GPIOx_ODR)

- 4 Registri di Controllo da 32 bit
  - GPIO port bit set/reset register (GPIOx _BSRR)
  - GPIO port configuration lock register (GPIOx_LCKR)
  - GPIO alternate function low register (GPI Ox_AFRL)
  - GPIO alternate function high register (GPIOx_AFRH)

GPIO port Mode Register



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O direction mode.

00: Input (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

2 bits in the register define the properties of 1 Port pin

GPIO port Mode Register

- Il registro GPIOx_MODER è un registro di 32 bits dove ogni insieme di 2 bit consecutivi rappresenta il modo di un singolo pin
- Ad esempio i bits con posizione 0 e 1 del GPIOD_MODER rappresentano il modo del pin GPIO PC0, I bits di posizione 26 e 27 dello stesso registro rappresentano il modo del pin GPIO PC13
- I due bit possono essere definiti come segue:
  - "00" – Input Mode : definisce l'utilizzo del pin in modalità di input
  - "01" – Output Mode: definisce l'utilizzo del pin in modalità di output
  - "10" – Analog Mode: definisce l'utilizzo del pin in modalità analogica
  - "11" – Alternate Functions

Bit Set/Reset Register (BSRR)

- E‘ un registro a 32bit utile al set o reset di specifici pin

- I 16 bit superiori sono mappati su ciascun pin del porto GPIO. Scrivere un 1 in queste locazioni porterà a 0 (o farà il “reset“) del pin associato.

- Allo stesso modo scrivere un 1 nei 16bit inferiori servirà a portare a 1 (o fare il “set“) il pin associato

Gli altri Registri

- Altri esempi di registri sono:
  - GPIO port output speed register (GPIOx_OSPEEDR) – Registro di configurazione della velocità di un singolo pin di output (2MHz, 10MHz, 50MHz)
  - GPIO port input data register (GPIOx _IDR) – Registro utile alla trasmissione di dati in input
  - GPIO port output data register (GPIOx_ODR) – Registro utile alla trasmissione di dati in output

Modo di Funzionamento

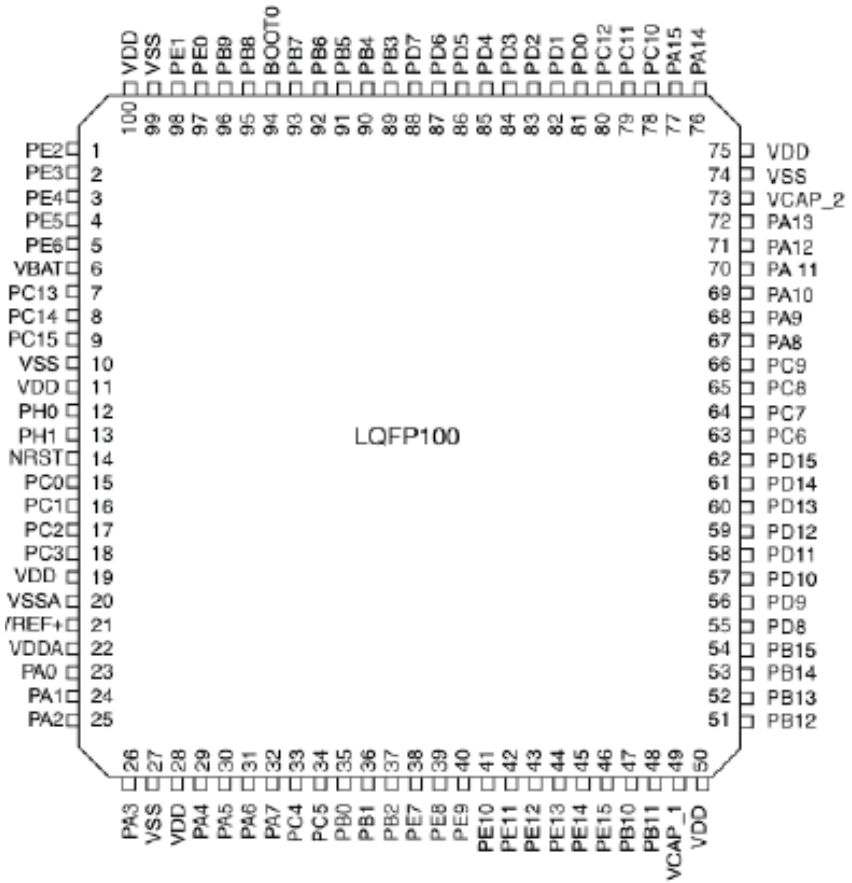
- Come già detto il *mode register* serve al programmatore per configurare il modo di funzionamento del GPIO
- In pratica, il programmatore può decidere se:
  - Abilitare la comunicazione (input/output) del microcontrollore con una specifica unità presente sulla scheda di sviluppo STM32F3-Discovery.

N.B. STM32F3 != STM32F3-Discovery. STM32F3 è il microcontrollore, STM32F3-Discovery è la scheda che comprende poi il microcontrollore

  - Abilitare la comunicazione con l'esterno di una periferica del microcontrollore (e.g. timer, SPI, I2C) (alternate function)
  - Abilitare la comunicazione con l'esterno di una periferica che trasmette segnali analogici (e.g. ADC/DAC, PWM)

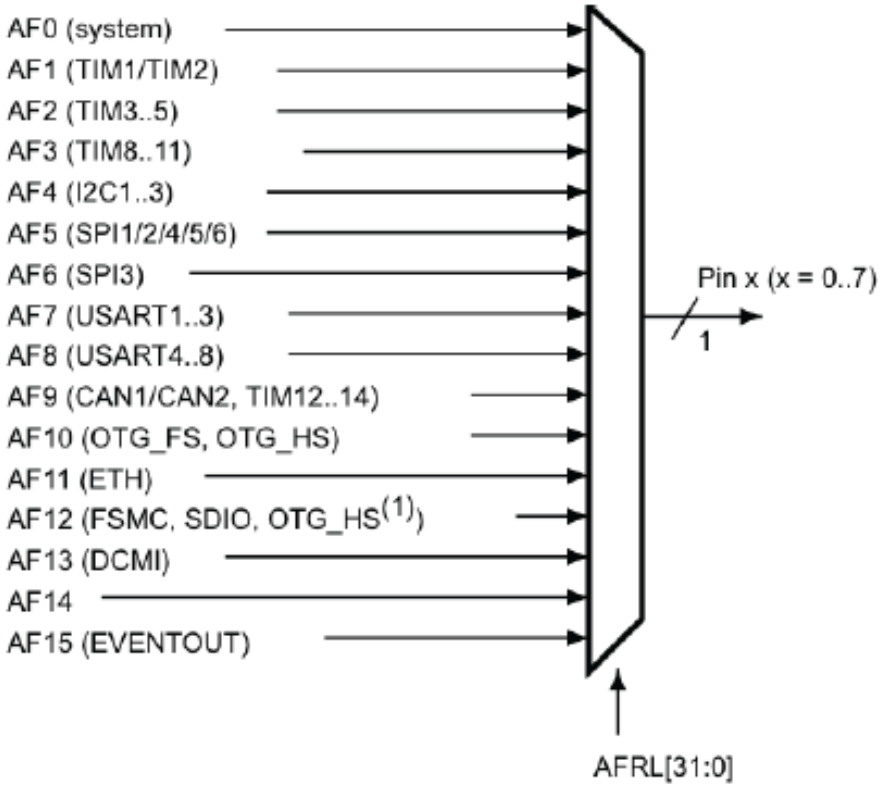
Alternate Functions – 1/3

- Permetto di stabilire la comunicazione del mondo esterno con i dispositivi interni al SoC.
- Consentono di ridurre il numero di pin richiesti dal package
- Le Alternate functions permettono l'utilizzo del GPIO per periferiche come UART, SPI ed altri.
- Si noti che se la modalità è impostata su AF le configurazioni per quel pin degli altri registri saranno sovrascritte con le impostazioni del tipo di periferica adottata.



Alternate Functions – 2/3

- I pin di I/O sono connessi alle periferiche del SoC attraverso un MUX
- Ogni pin di I/O ha un MUX con 16 Alternate Functions



Alternate Functions – 3/3

MCU pin			Board function													
Main function	Alternate functions	LQFP100	CS43L22	MP45DT02	LIS302DL	Pushbutton	LED	SWD	USB	OSC	Free I/O	Power supply	CN5	CN2	P1	P2
PA4	SPI1_NSS/ SPI3_NSS/ USART2_CK/ DCMI_HSYNC/ OTG_HS_SOF/ I2S3_WS/ ADC12_IN4/ DAC1_OUT	29	LRCK/AIN1x												16	

Come si Configura un Registro

- Per settare i bit di un registro si fa uso di **Maschere**
- Ci sono tre tipologie di maschere che si possono applicare a seconda dell'operazione che si vuole effettuare:
  - Bitwise ANDing: maschera utile a fare il "Clear" di uno o più bit
  - Bitwise ORing: maschera utile a fare il "Set" di uno o più bit
    - Ad esempio: REG1= "001101"

Maschera AND: "001001" Maschera OR: "100000"

"001101"	"001101"
"001001"	"101101"

Configurare un registro in C

- In C per configurare un registro e quindi fare una maschera si fa uso delle seguenti istruzioni:

  - PERIPHERAL->REG |= 0x04; <=> PERIPHERAL->REG = PERIPHERAL->REG | 0x04;

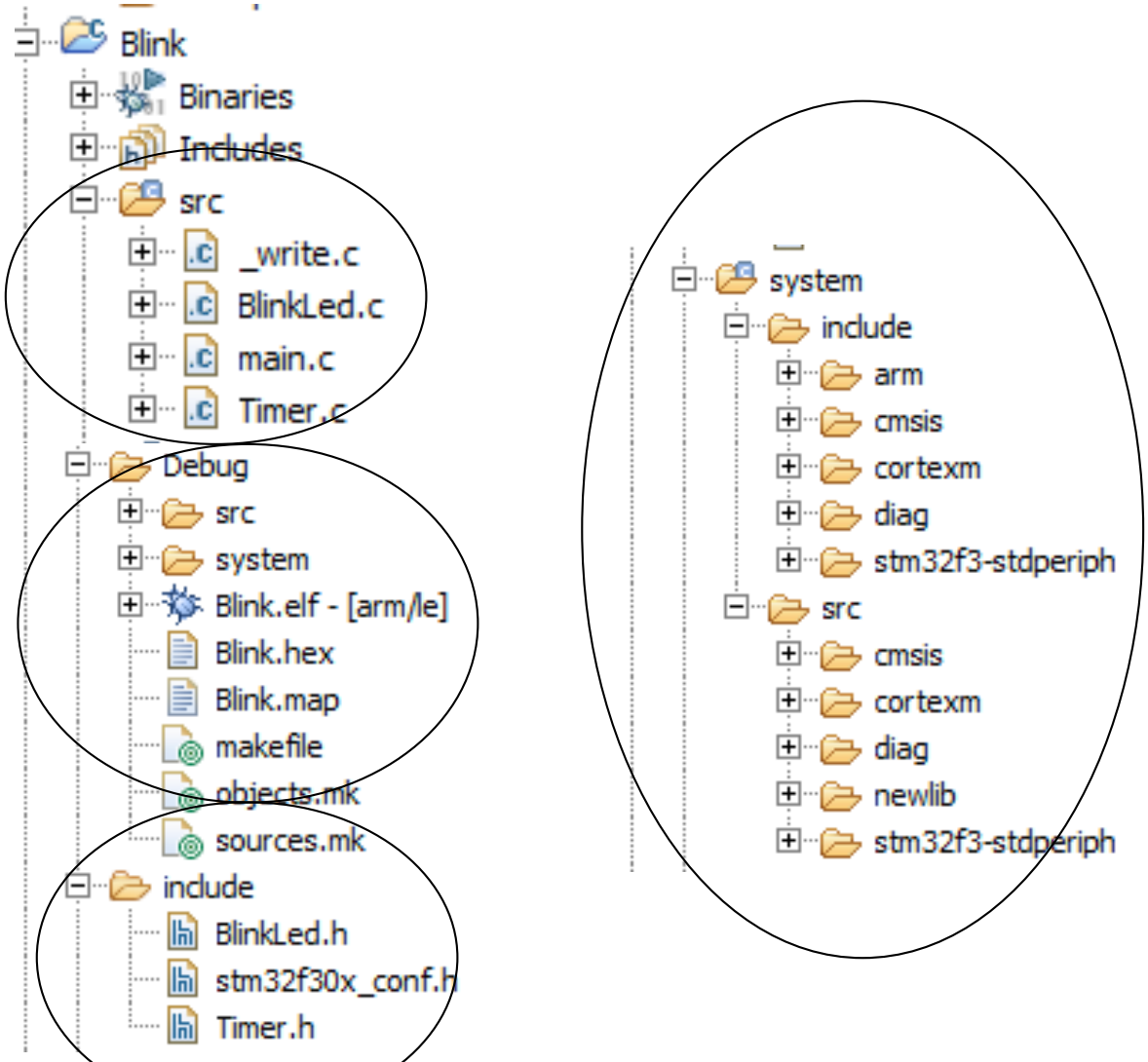
  Tale istruzione serve a utilizzare una maschera OR data da "0100"

  - PERIPHERAL->REG &= 0x02; <=> PERIPHERAL->REG = PERIPHERAL->REG & 0x02;

  Tale istruzione serve ad utilizzare una maschera AND data da "0010"

Organizzazione tipica di un progetto per lo sviluppo di un driver in Eclipse

- Un progetto tipico comprende:
  - Librerie messe a disposizione dei produttori a supporto della programmazione
  - Codice dell'utente
  - File utili al debug

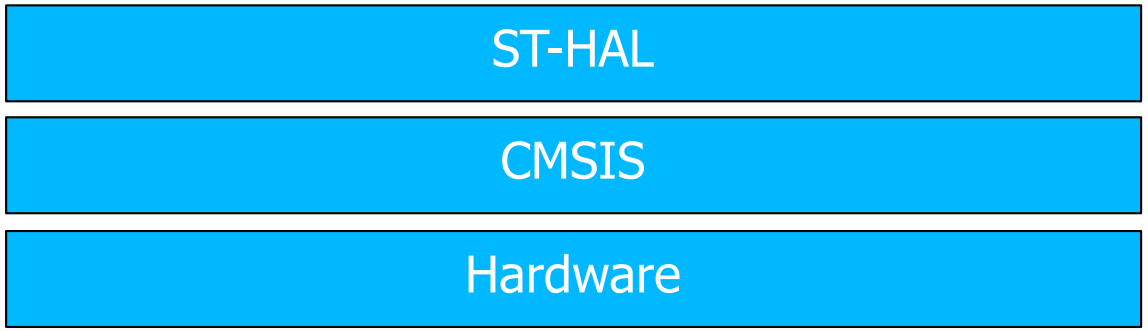


Librerie a Supporto del Programmatore

- I produttori forniscono
  - Header files e Librerie per utilizzare il dispositivo
    - L'header file definisce tutti i tipi e strutture dati necessarie alla programmazione
    - Definisce costanti simboliche per accedere ai registri di controllo delle periferiche
    - Definisce costanti simboliche per i valori di configurazione dei registri di controllo
  - Source files con le effettive implementazioni delle funzioni di libreria

CMSIS e ST-HAL

- Le librerie *Cortex Microcontroller Software Interface Standard (CMSIS)* sono delle librerie *vendor-independent* sviluppate da ARM che permettono di accedere in modo “semplice” all’hardware dei device basati su ARM Cortex
- Le *ST Hardware Abstraction Layer (HAL)* sono un insieme di librerie *vendor-specific* sviluppate da ST per i microcontrollori ST (e.g. STM32F3, STM32F4 STM32F0). Rappresentano un addizionale livello di astrazione.



Materiale per lo sviluppo di Driver

- Consigli prima di iniziare:
  - Fare sempre uso dei manuali quando si sviluppa un driver per una o più unità
  - User Manual:
    - http://www.st.com/content/ccc/resource/technical/document/user_manual/8a/56/97/63/8d/56/41/73/DM00063382.pdf/files/DM00063382.pdf/jcr:content/translations/en.DM00063382.pdf
  - Reference Manual:
    - http://www.st.com/content/ccc/resource/technical/document/reference_manual/4a/19/6e/18/9d/92/43/32/DM00043574.pdf/files/DM00043574.pdf/jcr:content/translations/en.DM00043574.pdf
  - HAL-API-Manual
    - http://www.st.com/content/ccc/resource/technical/document/user_manual/a6/79/73/ae/6e/1c/44/14/DM00122016.pdf/files/DM00122016.pdf/jcr:content/translations/en.DM00122016.pdf

Primo Driver per la gestione dei GPIO

- **Obiettivo**: Accendere un led sulla scheda STM32F4-Discovery
  - Passo 1 – Cercare nel manuale a quale porto e a quali pin GPIO sono associati i led della scheda
  - Passo 2 – Abilitare il clock sul porto GPIO trovato
  - Passo 3 – Inizializzare il GPIO
  - Passo 4 – Fare il "Set" dei pin relativi ai led di interesse
- Solo per questo primo esempio faremo uso delle librerie ST-stdperiph, di un livello di astrazione leggermente minore delle librerie ST-HAL

Esercizio

- Sviluppare un driver che permetta di far cambiare lo stato di un led (Lampeggiante o Fisso) in seguito alla pressione del pulsante "User" posto sulla scheda
  - Consigli:
    - Fare riferimento al manuale per trovare il GPIO del pulsante user.
    - Ragionare sul flusso di dati dei pulsanti