BITWISE OPERATIONS in C.   toddi se 17 bit è azero
&, |, ^, ~, <<, >>   . & (1<<27) == 0
AND OR XOR NOT LS RS   (1(1<<17) =1 No||
& = FFFF0000 setta 16 MSB dell'operando a 1 vero puedité...
               resetta 16 LSB dell'operando a 0
| = 0000 00FF setta 8 LSB dell'operando a 1   e setta 24MSB
| = 1 << 20 setta il 20° LSB a 1   dell'operando a
| = ~ (1 << 20) setta il 20° LSB a 0   valore presedente

PROGETTO   NXP LPC 1768

1) STARTUP_LPC 17xx.S → Clock config
   (Boot del chip)   • Heap config
                        • Interrupt Vector Table + Interrupt Handlers [WEAK]
                                                                       EXPORTED
                        • User Initial Stack & Heap
        ▽   RESET HANDLER BX   __main

2) main C   #• include .h (stdio, LPC17xx.h, led.h, button.h)
            #• definizione variabili globali / Scale Flag e button nature

• System Init ()
        ↓
     → 3) init_SoC_board)
              System Init()   • Cortex M3 Exceptions numbers
         System_LPC 17xx.C   • LPC17xx Interrupt numbers   → IRQn type
   • Gestione dei clock + misura: • include Core_cm3.h, system_LPC17xx.h
   • SCS System Control and Status Register • strutture dei registri delle periferiche
   • CLKSRCSEL · Clock Source Select Register (LPC_SC_TypeDef, LPC_PINCON_TypeDef,
     (MAIN OSC-CLK | IRC-OSC | RTC-CLK   LPC_GPIO_TypeDef, LPC_GPIOINT_TypeDef,
      OSCILLATOR   Internal RC RTC   LPC_TIM_TypeDef, LPC_PWM_TypeDef,
        12MHz   oscillator oscillator   UART, UART1, SPI, SSP, I2C, I2S, RIT, RTC
                     4MHz   WDT, ADC, DAC, MCPWM, QEI, CAN1, CAN2,
   • PLL0,1 config
      MAIN USB   → SYSCLK → PLL0 → PLLCLK • MEMORY MAP dei periferici
   • CPU CLOCK SPEED   USB CLK DIVIDER CPU CLOCK • INDIRIZZI BASE (LPC_AHB (RAM) 0_BASE
   • USB   USB CLK DIVIDER   LPC_APB0_BASE 0x4000
   • PERIPHERALS REG0, 1 CCLK PCLK0   ADVANCED PERIPHERAL
   • POWER CONTROL A PERIPHERALS REGISTER (PCONP) LPC_GPIO_BASE)) 0x2009C
   • CLOCK OUTPUT CONFIG REGISTER

   • Misura quanto impiega il sistema   • INDIRIZZI APB0 (LPC_PINCON_BASE = ... + 0x2C00)
   ○ eseguire 1000 NOP   PIN CONNECT BLOCK
   • sole configurazioni dei timer   • INDIRIZZI APB1
   per adattarli alla temporizzazione del   • INDIRIZZI AHB
   mumerizzatore: TIMER0 + init (0, 25000000) • INDIRIZZI GPIO
                   25MS
                    PCLK=25MHz #25   • DICHIARAZIONE delle variabili da usare in C per
   ...   25M → 455992   referenziare fra periferici

   ▽ APSR type   control, NVIC   #define LPC_PINCON ((LPC_PINCON_TypeDef *) LPC_PINCON_BASE
     IPSR   System   casta quella struttura in quell'indirizzo
     XPSR   Per accedere ai campi della struttura basterà...
   • NVIC Enable/Disable IRQ (IRQn_type)
     (Enable/Disable   set/set priority   LPC_PINCON → PINSEL2 4
      set priority   set enable IRQ
      system Reset   set / get clear pending IRQ
                       get active
                       set/get priority

· LED_init()

→ 4) lib_led.c   · unsigned char led_value;

· LED_init() ·

· LPC_PINCON → PINSEL4 &= 0xFFFF0000   ⟶ 1) Imposto selezione periferie
                                              LD12 — LD4
· PC_GPIO2 → FIO DIR |= 0x000000FF   P2.0   P2.7
· PC_GPIO2 → FIOCLR |= 0x000000FF   PORTA1 PINØ
                         SET   2) Imposto i pin della porta 2 per essere usati
· led_value = 0   come GPIO scrivendo "00" nei

· LED_deinit()   bit 0÷15 di PINSEL4

· PC_GPIO2 → FIODIR &= 0xFFFFFF00   3) Imposto la direzione dei PIN0-7
                                           della porta 2, impostato bit GPIO, in output

· BUTTON_init()   4) Setto gli i PIN 0-7 di valore 0.

→ 5) lib_button.c

· BUTTON_init()   1) Imposto selezione periferie

· LPC_PINCON → PINSEL4 |= 1<<20 | Key1 | Key2 | INT0 | RESET |
· | SW4 | SW3 | SW2 | SW1 |
·   22 | INT1 | INT2 | INT0 | |
·   24 | P2.11| P2.12| P2.10| |
· LPC_GPIO2 → FIODIR &= ~(1<<20) | PINSEL4.22 | PINSEL4.24 | PINSEL4.20 |
·   22
·   24   2) Imposto i PIN 20;25 della porta 2
                                           per essere usati come EINT0, INT1,
· LPC_SC → EXTMODE = 0x7   EINT2

                    111   3) Imposto le direzione di PIN 20-22
              |X X 0 =   della porta 2, impostato il GPIO, in output
              | 0 (LEVEL-SENSITIVE)   EINT INPUT
              |per EINT1,2,0   4) Imposto il segnale
                                        EXTMODE per far partire l'interrupt
· NVIC_EnableIRQ (EINT2_IRQn);   sul fronte di discesa del pulsante
   (... const ↓) (LPC17xx.h)

6) enable ▓▓▓▓▓▓   → FUNC1 LED.C
                                      · LED_NUM   8
   IRQ Button.c   · led_mask = ~ 1UL << 0
7) EINT 0 _ IRQ HANDLER()   · led_value
                                      · LED_on (num) { SCRIVE IN OUTPUT 1
   · LED_ON(0)   · LPC_GPIO2 → FIOPIN |= led_mask [num];
   · LPC_SC → EXTINT = (1<<0)   · led_value = PC_GPIO2 → FIOPIN;
                                       · LED_OFF (num)
8) EINT 1 ...   &= ~ ...1

   · LED_ON(1)
     LPC_SC → EXTINT = (1<<1)

9) EINT 2

   · LED_OFF (2)
     LED OFF (2)
   · LPC_SC → EXTINT &= (2<<2)   LEVEL SENSITIVE → ... EINT1 del PIN
                                    EDGE SENSITIVE → ... interrupt EINT del PIN
                                    → scrivendo per 2 quel bit viene messo a 1

Gestione INTERRUPT

1) NVIC_SetPriority (EINT? _IRQn, N?)
2) NVIC_EnableIRQ(EINT?)
3) Inizializzare strutture dati (vettori, puntatori, flag, sezioni...)
4) Scrivere ISR che conservi R4-R8, R10, R11, logica di risposta, strutture non globali
5) Prima della fine delle ISR pulire registro che indica interrupt attivo:

   LPC_SC -> EXTINT &= (1 << ?)

Gestione POWER MODES

⚠ NO DEEP SLEEP + DEBUG

- SLEEP   LPC -> SC -> PCON &= 0xFFFFFFFC (PM1 PM0)   DEEPSLEEP bit=0
- DEEP SLEEP   " "   1
- POWER DOWN   LPC_SC -> PCON &= 0xFFFFFFFD (0 1)   DEEPSLEEP = 0
- DEEP POWER DOWN LPC_SC -> PCON |= 0x1
                                    |= 0x3   (1 1)   DEEPSLEEP = 1

-- ASM("wfi") e wait for interrupt

Accendere periferici -> PCONP |= 1 (varianti)

Gestione TIMER   1 TIMER -> 1 IR, TCR, TC, PR, PC, MCR, MC0, MC1, MC2, MC3, CCR, CR0, CR1

1) Per TIM2 e TIM3 settare, bit corrispondenti (PCONP 2/3) a 1 per fornire
   alimentazione (TIM0 e TIM1 diretti di default in PCONP

2) Setta valori dei Match Register per ogni timer LPC_TIM0 -> MR0 = ...

3) Setta valori dei 4 MCR -> Interrupt, Reset, Stop   IRS/MR1 IRS/MR2 IRS/MR3
                              MR0
   LPC_TIM0 -> MCR = 0...4096   4 MCR
                                                                     4 TIMER (varianti)

4) Se serve timer > 2^{32} cc settare valore del Prescale Register PR in modo che
   il timer counter TC venga incrementato solo 1 volta ogni PR+1 cc
   LPC_TIM0 -> PR = 3,

5) NVIC_SetPriority (TIMER0 _IRQn)   TCR[0] = counter ENABLE
6) NVIC_EnableIRQ (TIMER0 _IRQn)   (timer counter register
7) enable_timer (...) -> LPC_TIM0 -> TCR = 1   (disable = 0)
8) reset_timer (...) -> reg val = LPC_TIM0 -> TCR   LPC_TIM0 -> TCR = reg val
9) TIMER0 _IRQHandler () { ... LPC_TIM0 -> IR = 1 }   interrupt

20) Ottenere ... RT/TIMER ... emulativo per vedere t ... reale

Gestione Registro interrupt timer

1) Imposto clock di ingresso del RIT : $LPC\_SC \rightarrow PCLKSEL1$ $\&= \sim(3<<26)$
   $= 00 = CCLK/4$

2) Alimento RIT (di default spento) $\qquad$ $LPC\_SC \rightarrow PCLKSEL1$ $|= (1<<26)$
   $= 01 = CCLK$
   $LPC\_SC \rightarrow PCONP$ $|= (1<<16)$
   $|= 2$
   $= 10 = CCLK/2$

3) Match value = intervallo periodico
   del RIT (in cc)
   $|= 3$
   $= 11 = CCLK/8$

   periodo — in parte
   $LPC \rightarrow RICOMPVAL = RIT$ Interval;

4) Imposto Registro di Controllo Repetitive interrupt timer

   RICTRL
   $\begin{cases} RITINT \left[\begin{smallmatrix}bit\\0\end{smallmatrix}\right] < \begin{smallmatrix}1 \to attivo \ interrupt\\ 0 \to NON \ "\ "\end{smallmatrix} \\ RITENCLR \left[\begin{smallmatrix}bit\\1\end{smallmatrix}\right] < \begin{smallmatrix}1 \to quando \ raggiunge \ valore \ match \ si \ azzera\\ 0 \to \ "\ "\ \qquad NON \ "\end{smallmatrix} \qquad LPC\_RIT \rightarrow RICTRL |= (1<<1) \\ RITENBR \left[\begin{smallmatrix}bit\\2\end{smallmatrix}\right] < \begin{smallmatrix}1 \to timer \ si \ ferma \ quando \ il \ processore \ si ferma \ per \ fare \ debug \ \to |= (1<<2);\\ 0 \to \ in \ NON \ "\end{smallmatrix} \end{cases}$

5) Imposto contatore a zero
   $LPC \rightarrow RIT \rightarrow COUNTER = 0$ (reset RIT)

6) $NVIC\_Enable\ IRQ\ (RIT\_IRQn)$

7) enable RIT $\downarrow$

   proprio $RITEN \left[\begin{smallmatrix}bit\\3\end{smallmatrix}\right] < \begin{smallmatrix}1 \ parte \ il \ timer\\ 0 \ timer \ disabilitato\end{smallmatrix}$ $\qquad LPC\_RIT \rightarrow RICTRL |= (1<<3);$
   $(disable\_RIT)$

8) Gestione DEBOUNCING

   EINT0 ira handler $\rightarrow$ enable_RIT(); $\to$ schiaccio pulsante e parte il timer
   · $NVIC\_DISABLE\ IRQ\ (EINT\ ?\ \_IRQn)$ $\to$ pulsante non vengono letti :
     entro debouncing
   · $LPC\_PINCON \rightarrow PINSEL4$ $\&= \sim(1<<24)$ $\to$ bit (13:22) = 00
   · $LPC\_SC \rightarrow EXTINT$ $\&= (1<<1)$ $\to$ pulite flag
     interrupt EINT0
   · FINE! Non si deve + occupare di gestire
     continua cosi!

   dopo 50 ms
   viene eseguito

9) RIT_IRQ handler · deb passa a 1 (non statica!)
   $LPC\_GPIO2 \rightarrow FIOPIN < \begin{smallmatrix}1 \ pulsante \ spento\\ 0 \ pulsante \ acceso\end{smallmatrix}$
   alla fine pulisco il flag di interrupt
   $LPC\_RIT \rightarrow RICTRL |= 0x1$