

# Interrupt Controller

Paolo Bernardi

# Input/Output system management

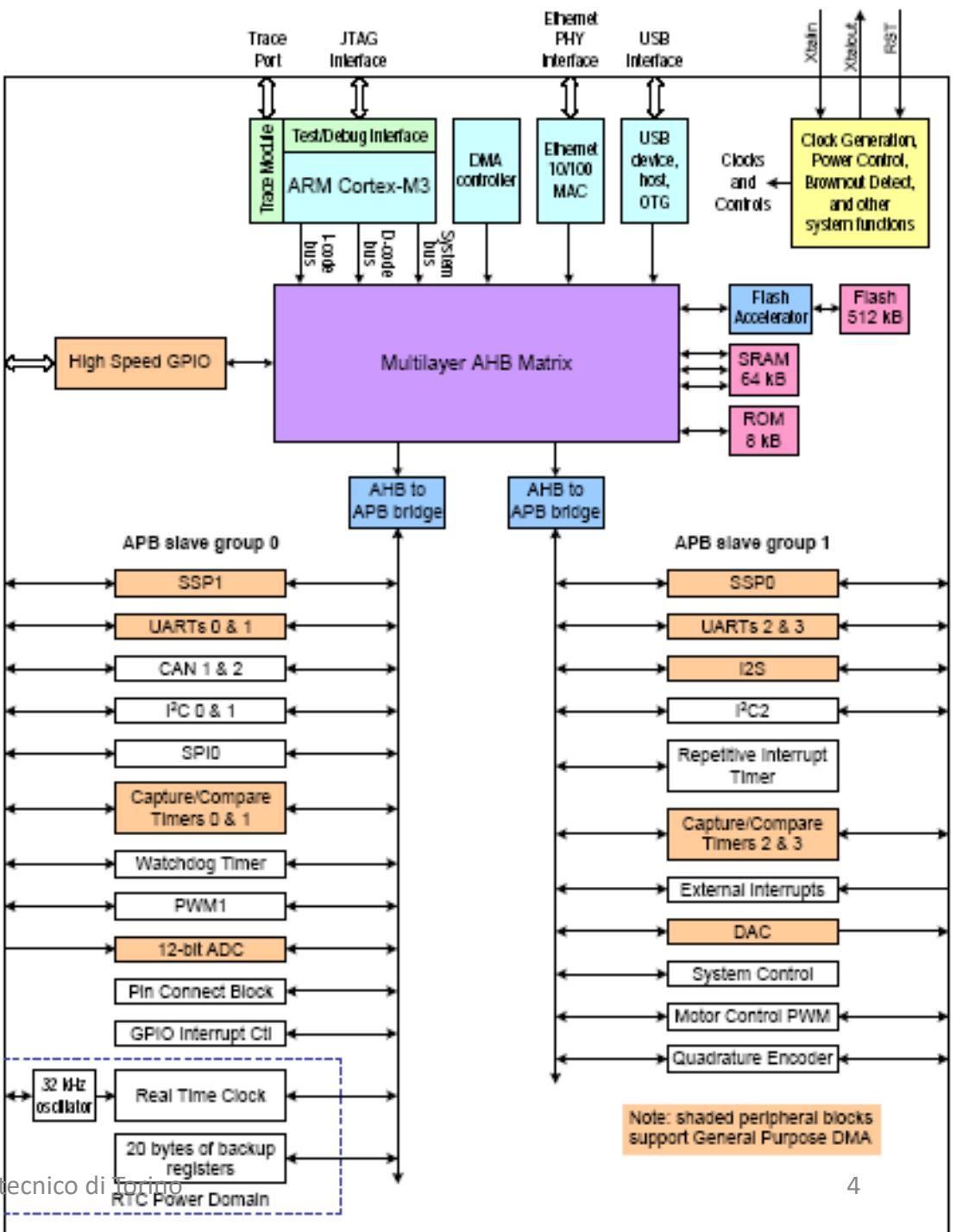
- The main function of a Input/Output (I/O) system is to exchange information with the external world.
- An I/O system needs to be controlled by the CPU which has to intercept service requests
  - For example if a new data is received by a peripheral core
- There are 2 main methods used to manage service requests:
  - Polling
  - Interrupt.

# System event categories

- Respond to infrequent but important events
  - Alarm conditions like low battery power (i.e., NMI)
  - Error conditions (i.e., USAGE FAULT)
- I/O synchronization
  - Trigger interrupt when signal on a port changes
- Periodic interrupts
  - Generated by the timer at a regular rate
  - Systick timer can generate interrupt when it hits zero
  - Reload value + frequency determine interrupt rate
- Data acquisition samples ADC

# Block diagram NXP LPC1768

- I/O system composed of several peripheral cores
  - Serial ports – UARTS
  - 12-bit ADC / DAC
  - GPIO
  - Timers
  - Other communication protocols like
    - I2C
    - SPI/SSP



# Polling

- Polling is the process where the computer or controlling device frequently interrogates an external device to check for its current state
  - Checking status registers (best practice)
  - Checking data registers.
- The polling is often implemented as a software cycle
  - Performing a predefined sequence of checks at a regular pace
  - A scheduling can be defined to access peripheral cores more or less frequently
- If the polled core needs to be handled, the CPU moves from the polling loop to the handler of the specific event.
- Main characteristics
  - The most of the time is spent in the software cycle (disadvantage – power inefficient)
  - Easy to implement (advantage)
  - High latency in the managing cycle (disadvantage – low performance)
  - Difficult management of nested requests (disadvantage – very low performance).

# Interrupt

- Peripheral devices are directly interacting with the CPU,
  - CPU may perform different tasks having low level priority,
  - Idle mode can be entered,
  - The system wakes up as soon as a peripheral core is requesting a service
- When a request is received, the CPU needs to recognize the source of request in order to execute the proper handler
- Current architectures implement a Vectored Interrupt management method
  - Based on the Interrupt Vector Table (IVT)
  - The CPU collaborates with an external device called Interrupt Controller

# System setup for interrupt mode

- Things you must do when programming a system to use interrupts
- BOOT TIME
  - Initialize data structures
    - Counters, pointers
    - Eventually specify a process variable that may interrupt (i.e., semaphores)
  - Configure Interrupt Controller
    - Enable interrupt sources
    - Set priority of every source

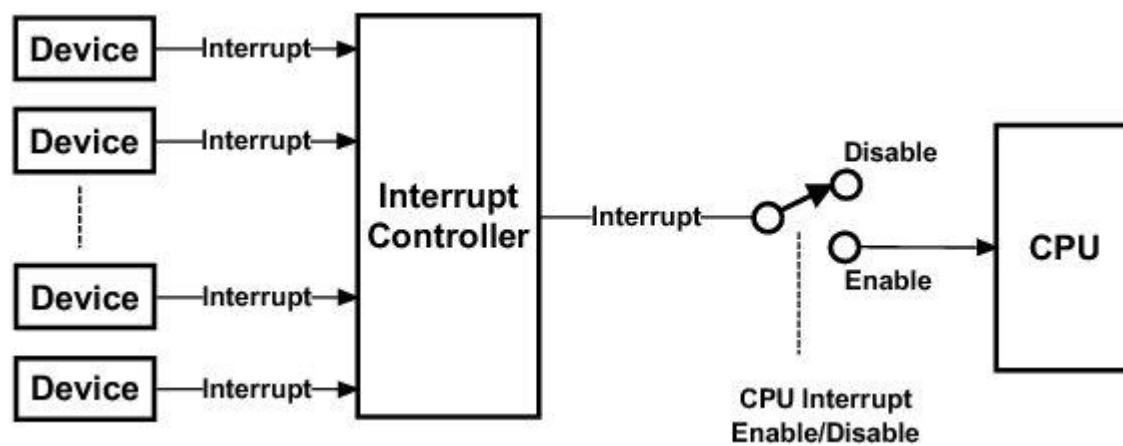
# System setup for interrupt mode (II)

- Things you must do in every Interrupt Service Routine (ISR)
- RUNTIME
  - Acknowledge
    - Clear the flags that indicate the interrupt is active
    - Can be done in different parts of the interrupt service routine
  - Maintain contents of R4-R8,R10-R11 (ABI AAPCS)
  - Communicate via shared global variables.

May be important for  
nesting interruptions

# Interrupt controller

- In computing, an interrupt controller is a device that is used to combine several sources of interrupt into one or more CPU lines, while allowing priority levels to be assigned to its interrupt outputs
- Manages interrupt signals received from devices by combining multiple interrupts into a single interrupt output.



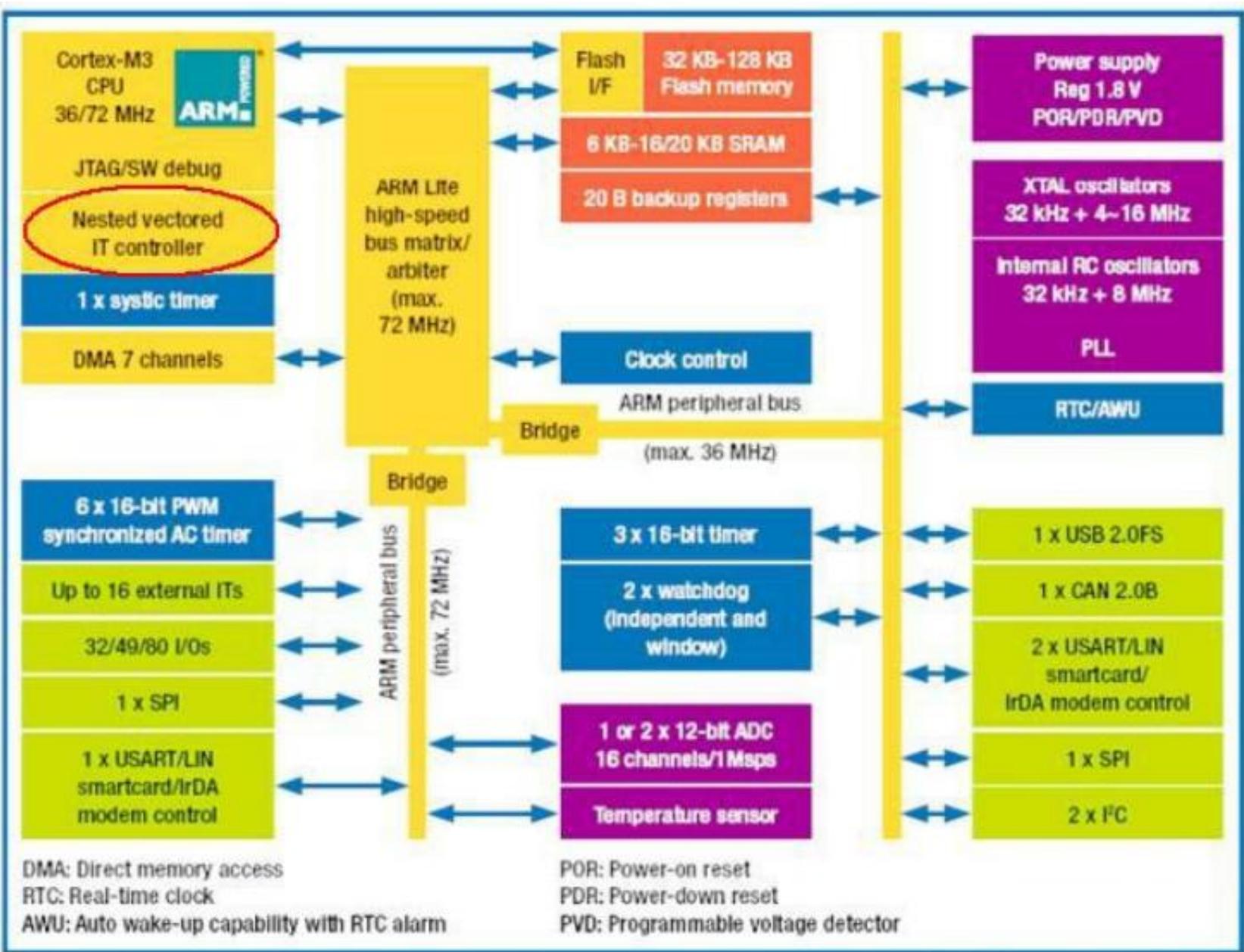
# Nested Vectored Interrupt Controller (NVIC)

## (UM pg. 78)

- The Nested Vectored Interrupt Controller (NVIC) is an integral part of the Cortex-M3.
- The tight coupling to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts
- It manages 35 possible external interrupt.

Table 52. Interrupt Set-Enable Register 0 register (ISER0 - 0xE000 E100)

Bit	Name	Function
0	ISE_WDT	Watchdog Timer Interrupt Enable. Write: writing 0 has no effect, writing 1 enables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.
1	ISE_TIMER0	Timer 0 Interrupt Enable. See functional description for bit 0.
2	ISE_TIMER1	Timer 1. Interrupt Enable. See functional description for bit 0.
3	ISE_TIMER2	Timer 2 Interrupt Enable. See functional description for bit 0.
4	ISE_TIMER3	Timer 3 Interrupt Enable. See functional description for bit 0.
5	ISE_UART0	UART0 Interrupt Enable. See functional description for bit 0.
6	ISE_UART1	UART1 Interrupt Enable. See functional description for bit 0.
7	ISE_UART2	UART2 Interrupt Enable. See functional description for bit 0.
8	ISE_UART3	UART3 Interrupt Enable. See functional description for bit 0.
9	ISE_PWM	PWM1 Interrupt Enable. See functional description for bit 0.
10	ISE_I2C0	I <sup>2</sup> C0 Interrupt Enable. See functional description for bit 0.
11	ISE_I2C1	I <sup>2</sup> C1 Interrupt Enable. See functional description for bit 0.
12	ISE_I2C2	I <sup>2</sup> C2 Interrupt Enable. See functional description for bit 0.
13	ISE_SPI	SPI Interrupt Enable. See functional description for bit 0.
14	ISE_SSP0	SSP0 Interrupt Enable. See functional description for bit 0.
15	ISE_SSP1	SSP1 Interrupt Enable. See functional description for bit 0.
16	ISE_PLL0	PLL0 (Main PLL) Interrupt Enable. See functional description for bit 0.
17	ISE_RTC	Real Time Clock (RTC) Interrupt Enable. See functional description for bit 0.
18	ISE_EINT0	External Interrupt 0 Interrupt Enable. See functional description for bit 0.
19	ISE_EINT1	External Interrupt 1 Interrupt Enable. See functional description for bit 0.
20	ISE_EINT2	External Interrupt 2 Interrupt Enable. See functional description for bit 0.
21	ISE_EINT3	External Interrupt 3 Interrupt Enable. See functional description for bit 0.
22	ISE_ADC	ADC Interrupt Enable. See functional description for bit 0.
23	ISE_BOD	BOD Interrupt Enable. See functional description for bit 0.
24	ISE_USB	USB Interrupt Enable. See functional description for bit 0.
25	ISE_CAN	CAN Interrupt Enable. See functional description for bit 0.
26	ISE_DMA	GPDMA Interrupt Enable. See functional description for bit 0.
27	ISE_I2S	I <sup>2</sup> S Interrupt Enable. See functional description for bit 0.
28	ISE_ENET	Ethernet Interrupt Enable. See functional description for bit 0.
29	ISE_RIT	Repetitive Interrupt Timer Interrupt Enable. See functional description for bit 0.
30	ISE_MCPWM	Motor Control PWM Interrupt Enable. See functional description for bit 0.
31	ISE_QEI	Quadrature Encoder Interface Interrupt Enable. See functional description for bit 0.



# Library functions in core\_cm3.h

```
935  /** \brief Enable External Interrupt
936
937     This function enables a device specific interrupt in the NVIC interrupt controller.
938     The interrupt number cannot be a negative value.
939
940     \param [in]      IRQn  Number of the external interrupt to enable
941 */
942 static __INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
943 {
944     NVIC->ISER[((uint32_t)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* enable interrupt */
945 }
946
1015 /** \brief Set Interrupt Priority
1016
1017     This function sets the priority for the specified interrupt. The interrupt
1018     number can be positive to specify an external (device specific)
1019     interrupt, or negative to specify an internal (core) interrupt.
1020
1021     Note: The priority cannot be set for every core interrupt.
1022
1023     \param [in]      IRQn  Number of the interrupt for set priority
1024     \param [in]      priority  Priority to set
1025 */
1026 static __INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
1027 {
1028     if(IRQn < 0) {
1029         SCB->SHP[((uint32_t)(IRQn) & 0xF)-4] = ((priority << (8 - __NVIC_PRIO_BITS)) & 0xff); } /* set Priority for Cortex-M System I
1030     else {
1031         NVIC->IP[(uint32_t)(IRQn)] = ((priority << (8 - __NVIC_PRIO_BITS)) & 0xff); }           /* set Priority for device specific In
1032 }
```

# NVIC Constant and addresses

- core\_cm3.h

```
832 /* Memory mapping of Cortex-M3 Hardware */
833 #define SCS_BASE          (0xE000E000)           /*!< System Control Space Base Address */
834 #define ITM_BASE          (0xE0000000)           /*!< ITM Base Address */
835 #define CoreDebug_BASE    (0xE000EDF0)           /*!< Core Debug Base Address */
836 #define SysTick_BASE      (SCS_BASE + 0x0010)       /*!< SysTick Base Address */
837 #define NVIC_BASE         (SCS_BASE + 0x0100)       /*!< NVIC Base Address */
838 #define SCB_BASE          (SCS_BASE + 0x0D00)       /*!< System Control Block Base Address */
839
840 #define InterruptType    ((InterruptType_Type *) SCS_BASE)  /*!< Interrupt Type Register */
841 #define SCB               ((SCB_Type *) SCB_BASE)        /*!< SCB configuration struct */
842 #define SysTick           ((SysTick_Type *) SysTick_BASE)  /*!< SysTick configuration struct */
843 #define NVIC              ((NVIC_Type *) NVIC_BASE)       /*!< NVIC configuration struct */
844 #define ITM               ((ITM_Type *) ITM_BASE)        /*!< ITM configuration struct */
845 #define CoreDebug         ((CoreDebug_Type *) CoreDebug_BASE) /*!< Core Debug configuration struct */
846
```

# NVIC Constant and addresses

- core\_cm3.h

```
214  /** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).  
215  */  
216  typedef struct  
217  {  
218      __IO uint32_t ISER[8];          /*!< Offset: 0x000 (R/W)  Interrupt Set Enable Register */  
219      uint32_t RESERVED0[24];  
220      __IO uint32_t ICER[8];          /*!< Offset: 0x080 (R/W)  Interrupt Clear Enable Register */  
221      uint32_t RESERVED1[24];  
222      __IO uint32_t ISPR[8];          /*!< Offset: 0x100 (R/W)  Interrupt Set Pending Register */  
223      uint32_t RESERVED2[24];  
224      __IO uint32_t ICPR[8];          /*!< Offset: 0x180 (R/W)  Interrupt Clear Pending Register */  
225      uint32_t RESERVED3[24];  
226      __IO uint32_t IABR[8];          /*!< Offset: 0x200 (R/W)  Interrupt Active bit Register */  
227      uint32_t RESERVED4[56];  
228      __IO uint8_t  IP[240];          /*!< Offset: 0x300 (R/W)  Interrupt Priority Register (8Bit wide) */  
229      uint32_t RESERVED5[644];  
230      __O  uint32_t STIR;             /*!< Offset: 0xE00 ( /W) Software Trigger Interrupt Register */  
231 } NVIC_Type;  
...
```

# Experiment priority and nested interruptions

Screenshot of the µVision4 IDE showing the configuration and monitoring of Nested Vectored Interrupt Controller (NVIC) settings for an ARM Cortex-M3 target.

The Project Explorer shows files like `startup_LPC17xx.s`, `sample.c`, `lib_led.c`, and `button.h`.

The Peripherals tab is selected, and the Nested Vectored Interrupt Controller is highlighted.

The code editor displays the `main.c` file:

```
/* LPC17xx definitions */
#include "button.h"

/* from funct_led */
char value; /* defined in funct_led */

/* System Initialization (i.e., PLL)
 * LED Initialization
 * BUTTON Initialization
 */

/* Loop forever
 */

/* General Purpose Input/Output 2 (GPIO 2) - Fast Interface
 * Registers
 * EXTINT: 0x06 EXTMODE: 0x07
 * EXTPOLAR: 0x00
 */

Computer Architectures - Politecnico di Torino
```

The NVIC configuration window shows the following table:

Idx	Source	Name	E	P	A	Priority
32	PLL0 Lock	PLOCK0	0	1	0	0
33	RTC CIF		0	0	0	0
33	RTC ALF		0	0	0	0
34	External Interrupt 0	EINT0	1	0	0	3
35	External Interrupt 1	EINT1	1	0	1	2
36	External Interrupt 2	EINT2	1	0	1	1
37	External Interrupt 3	EINT3	0	0	0	0
37	GPIO Interrupts		0	0	0	0
38	A/D Converter	ADC	0	0	0	0
39	Brown Out Detect	BOD	0	0	0	0
40	USB		0	0	0	0
41	CAN		0	0	0	0

Selected Interrupt: EINT0 (Priority: 3)

Interrupt Control & State:

- INT\_CTRL\_ST: 0x00400024
- VECTACTIVE: 0x24
- RETTBASE
- VECTPENDING: 0x00
- ISRPREEMPT
- ISRENDING

Application Interrupt & Reset Control:

- AIRC: 0xFA050000
- PRIGROUP: 0: 7.1
- VECTRESET
- VECTCLRACTIVE
- SYSRESETREQ
- ENDIANNESS

Vector Table Offset:

- VTO: 0x00000000
- TBLOFF: 0x00000000
- TBLBASE

Software Interrupt Trigger:

- SW\_TRIG\_INT: 0x00000000
- INTID: 0x00

General Purpose Input/Output 2 (GPIO 2) - Fast Interface Registers:

Bit	31	32	33	34	35	36	37	38	39	40	41
FIO2DIR	0x000000FF				0x00000000						
FIO2MASK	0x00000000				0x00000000						
FIO2SET	0x00000000				0x00000000						
FIO2CLR	0x00000000				0x00000000						
FIO2PIN	0x00002300				0x00002300						
Pins	0x00002F00				0x00002F00						

Computer Architectures - Politecnico di Torino

Page 15

# BOOT

```

8 void BUTTON_init(void) {
9
10    LPC_PINCON->PINSEL4 |= (1 << 20);
11    LPC_GPIO2->FIODIR &= ~(1 << 10);
12
13    LPC_PINCON->PINSEL4 |= (1 << 22);
14    LPC_GPIO2->FIODIR &= ~(1 << 11);
15
16    LPC_PINCON->PINSEL4 |= (1 << 24);
17    LPC_GPIO2->FIODIR &= ~(1 << 12);
18
19    LPC_SC->EXTMODE = 0x7;
20
21    NVIC_EnableIRQ(EINT2_IRQn);
22    NVIC_SetPriority(EINT2_IRQn, 1);
23    NVIC_EnableIRQ(EINT1_IRQn);
24    NVIC_SetPriority(EINT1_IRQn, 2);
25    NVIC_EnableIRQ(EINT0_IRQn);
26    NVIC_SetPriority(EINT0_IRQn, 3);
27 }
28

```

Nested Vectored Interrupt Controller (NVIC)

Idx	Source	Name	E	P	A	Priority
31	SSP1		0	0	0	0
32	PLL0 Lock	PLOCK0	0	1	0	0
33	RTC CIF		0	0	0	0
33	RTC ALF		0	0	0	0
34	External Interrupt 0	EINT0	1	0	0	3
35	External Interrupt 1	EINT1	1	0	0	2
36	External Interrupt 2	EINT2	1	0	0	1
37	External Interrupt 3	EINT3	0	0	0	0
37	GPIO Interrupts		0	0	0	0
38	A/D Converter	ADC	0	0	0	0
39	Brown Out Detect	BOD	0	0	0	0
40	USB		0	0	0	0

Selected Interrupt  
 Enable    Pending    Active   Priority: 3

Interrupt Control & State  
INT\_CTRL\_ST: 0x00400000   VECTACTIVE: 0x00  
 RETTOBASE   VECTPENDING: 0x00  
 ISRPREEMPT    ISR PENDING

Application Interrupt & Reset Control  
AIRC: 0xFA050000   PRIGROUP: 0: 7.1  
 VECTRESET    SYSRESETREQ  
 VECTCLRACTIVE    ENDIANESS

Vector Table Offset  
VTO: 0x00000000   TBLOFF: 0x00000000  
 TBLBASE

Software Interrupt Trigger  
SW\_TRIGGER\_INT: 0x00000000   INTID: 0x00

# RUNTIME (1)

- CASE 1)

1. The EINT2 interrupt is taken and being served
2. The EINT1 interrupt (with lower priority) is taken
3. EINT1 is pending and will be served only when EINT2 is fully handled

```
6 void EINT0_IRQHandler (void)
7 {
8     LED_On(0);
9     LPC_SC->EXTINT &= (1 << 0); /* clear pending interrupt */
10 }
11
12
13 void EINT1_IRQHandler (void)
14 {
15     LED_On(1);
16     LPC_SC->EXTINT &= (1 << 1); /* clear pending interrupt */
17 }
18
19 void EINT2_IRQHandler (void)
20 {
21     LED_Off(0);
22     LED_Off(1);
23     LPC_SC->EXTINT &= (1 << 2); /* clear pending interrupt */
24 }
```

Nested Vectored Interrupt Controller (NVIC)

Idx	Source	Name	E	P	A	Priority
31	SSP1		0	0	0	0
32	PLL0 Lock	PLOCK0	0	1	0	0
33	RTC CIF		0	0	0	0
33	RTC ALF		0	0	0	0
34	External Interrupt 0	EINT0	1	0	0	3
35	External Interrupt 1	EINT1	1	1	0	2
36	External Interrupt 2	EINT2	1	0	1	1
37	External Interrupt 3	EINT3	0	0	0	0
37	GPIO Interrupts		0	0	0	0
38	A/D Converter	ADC	0	0	0	0
39	Brown Out Detect	BOD	0	0	0	0
40	USB		0	0	0	0

Selected Interrupt  
 Enable  Pending  Active Priority: 3

Interrupt Control & State  
INT\_CTRL\_ST: 0x00423824 VECTACTIVE: 0x24  
 RETTOBASE VECTPENDING: 0x23  
 ISRPREEMPT  ISRPENDING

Application Interrupt & Reset Control  
AIRC: 0xFA050000 PRIGROUP: 0: 7.1 ▾  
 VECTRESET  SYSRESETREQ  
 VECTCLRACTIVE  ENDIANESS

Vector Table Offset  
VTO: 0x00000000 TBLOFF: 0x00000000  
 TBLBASE

Software Interrupt Trigger  
SW\_TRIG\_INT: 0x00000000 INTID: 0x00

Computer Architectures - Politecnico di Torino

# RUNTIME (2)

- CASE 2)
  1. The EINT1 interrupt is taken and being served
  2. The EINT2 interrupt (with higher priority) is taken
  3. EINT1 is suspended and completed only when EINT2 is handled

```
6 void EINT0_IRQHandler (void)
7 {
8     LED_On(0);
9     LPC_SC->EXTINT &= (1 << 0); /* clear pending interrupt */
10 }
11
12 void EINT1_IRQHandler (void)
13 {
14     LED_On(1);
15     LPC_SC->EXTINT &= (1 << 1); /* clear pending interrupt */
16 }
17
18 void EINT2_IRQHandler (void)
19 {
20     LED_Off(0);
21     LED_Off(1);
22     LPC_SC->EXTINT &= (1 << 2); /* clear pending interrupt */
23 }
24 }
```

Nested Vectored Interrupt Controller (NVIC)

Idx	Source	Name	E	P	A	Priority
31	SSP1		0	0	0	0
32	PLL0 Lock	PLOCK0	0	1	0	0
33	RTC CIF		0	0	0	0
33	RTC ALF		0	0	0	0
34	External Interrupt 0	EINT0	1	0	0	3
35	External Interrupt 1	EINT1	1	0	1	2
36	External Interrupt 2	EINT2	1	0	1	1
37	External Interrupt 3	EINT3	0	0	0	0
37	GPIO Interrupts		0	0	0	0
38	A/D Converter	ADC	0	0	0	0
39	Brown Out Detect	BOD	0	0	0	0
40	USB		0	0	0	0

Selected Interrupt  
 Enable  Pending Priority: -2

Interrupt Control & State  
INT\_CTRL\_ST: 0x00400024 VECTACTIVE: 0x24  
 RETTOBASE VECTPENDING: 0x00  
 ISRPREEMPT  ISR PENDING

Application Interrupt & Reset Control  
AIRC: 0xFA050000 PRIGROUP: 0: 7.1 ▾  
 VECTRESET  SYSRESETREQ  
 VECTCLRACTIVE  ENDIANESS

Vector Table Offset  
VTO: 0x00000000 TBLOFF: 0x00000000  
 TBLBASE

Software Interrupt Trigger  
SW\_TRIGGER\_INT: 0x00000000 INTID: 0x00

cnico@laura: ~ 18