POLITECNICO
MILANO 1863





MICROCONTROLLERS

LAB – Interrupt

What is an interrupt?

An interrupt is a **request to the processor** to suspend the current execution, save the current state, and execute an Interrupt Service Routine (ISR) function. When the ISR is exhausted, the processor will resume the regular operation.

Why are interrupts (extremely) useful?

To efficiently manage **internal or external asynchronous events** without affecting the program's normal execution.

- Internal events: ADC end-of-conversion, TIMER overflow, etc…
- External events: input-pin status change, SPI received data, etc…

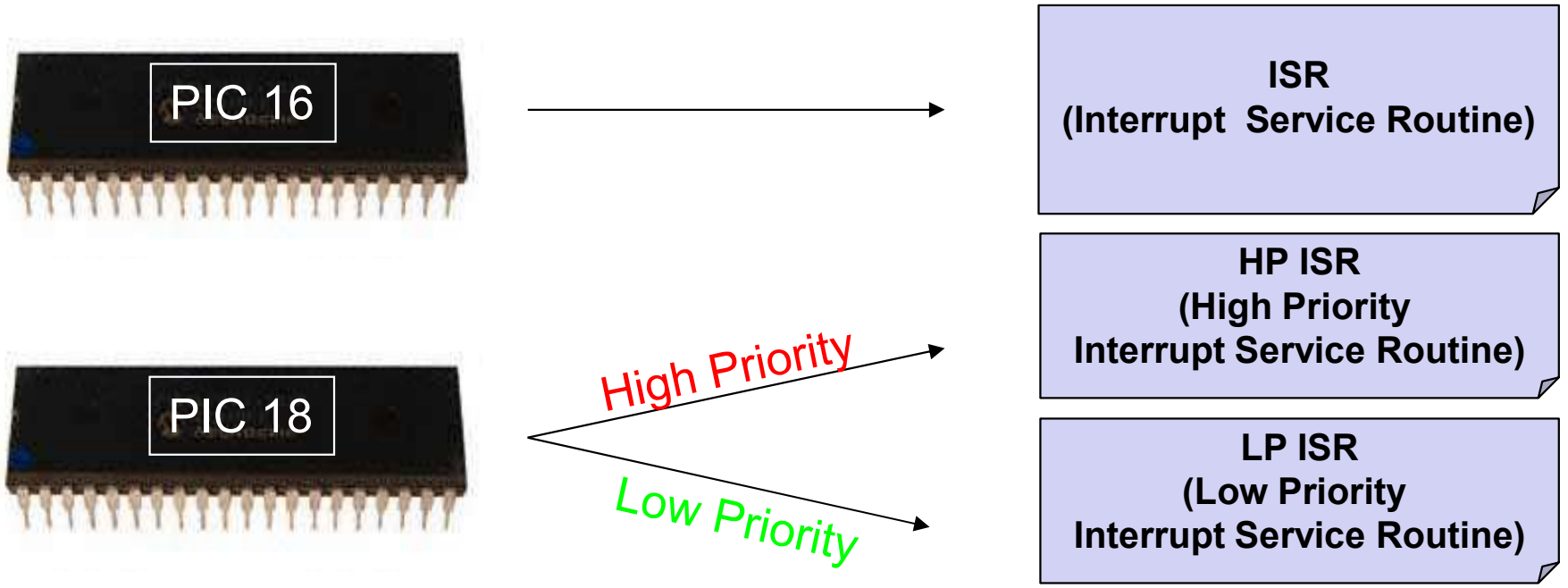
Why are they better than polling?

With interrupts, assigning different priorities to each event that may co-occur, managing the **timing requirements** of specific tasks is possible. Moreover, **no processor elaboration time** is used to check constantly if a condition has happened or not.

Interrupts - Intro

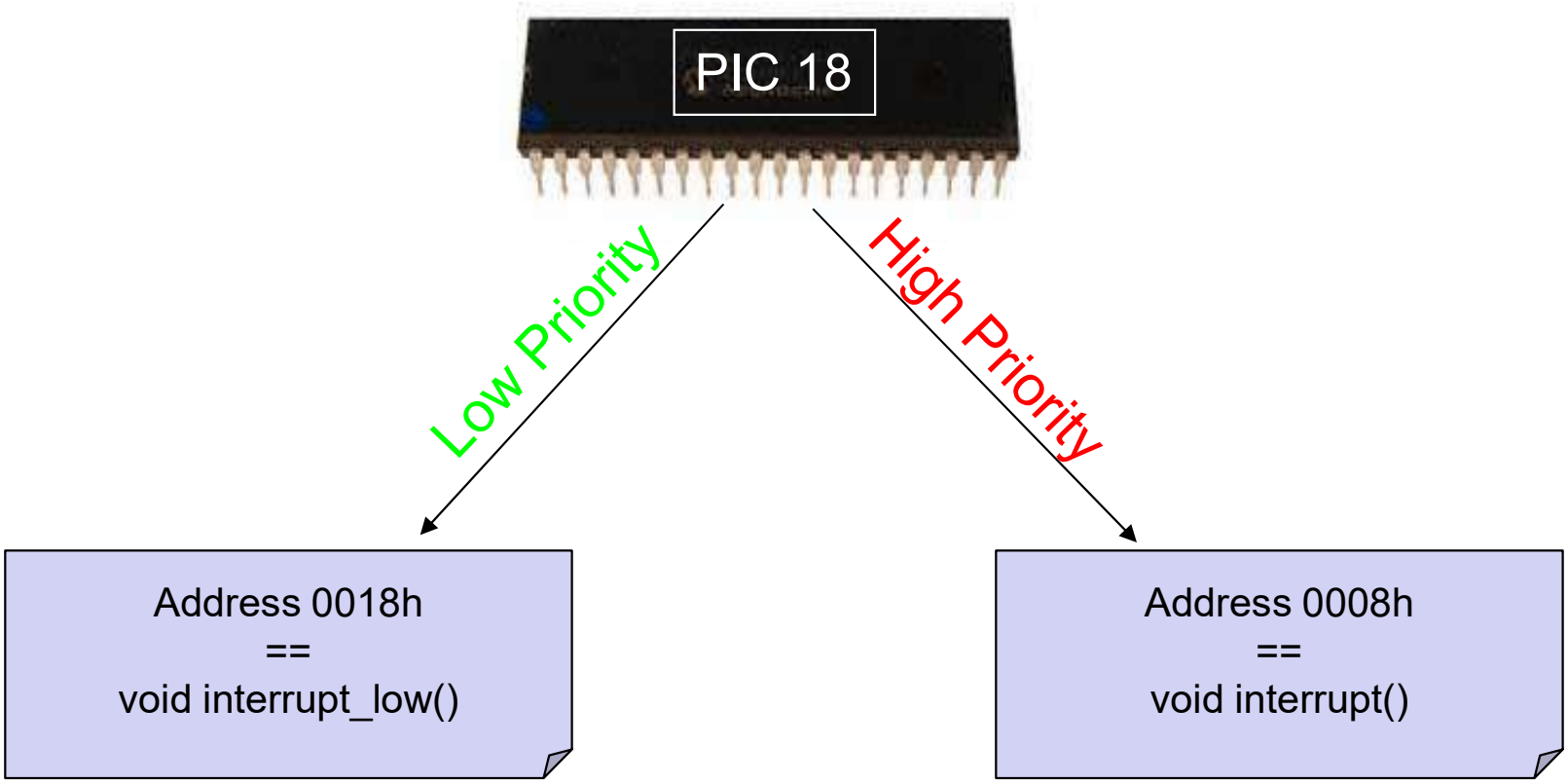
When an interrupt arrives, this will be the flow of events:

- Interrupt event
- Normal execution frozen
- Microcontroller saves current status
- Interrupt routine execution
- Status recovered and normal program flow re-established.



Interrupts – LP vs HP

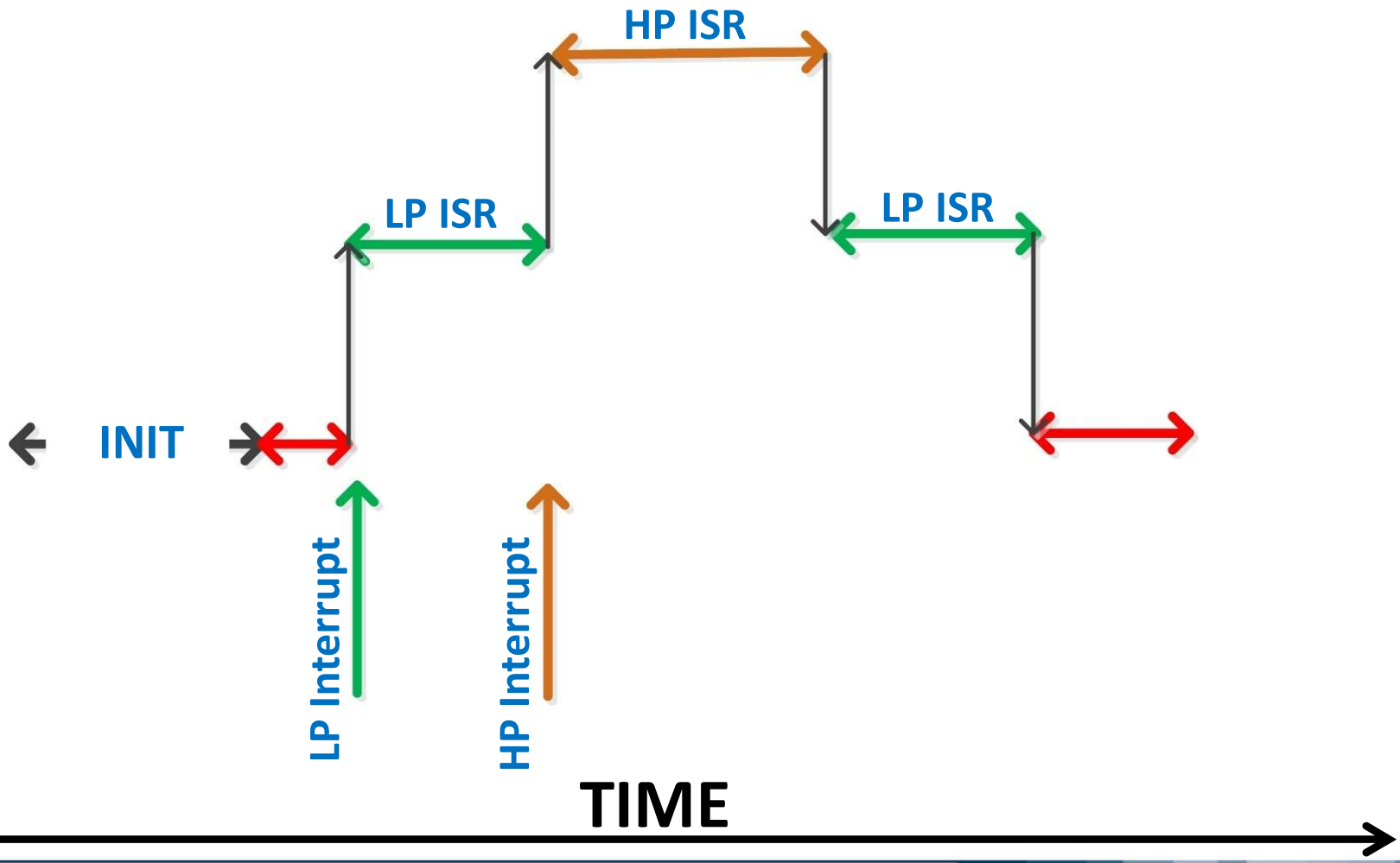
In PIC, the interrupt is handled by saving the state and pointing to a reserve program address. The ISR address is chosen depending on the interrupt priority,



Interrupts – LP vs HP

Main execution can be interrupted by both HP and LP interrupt.

ISR can be interrupted ONLY by a higher priority level interrupt.



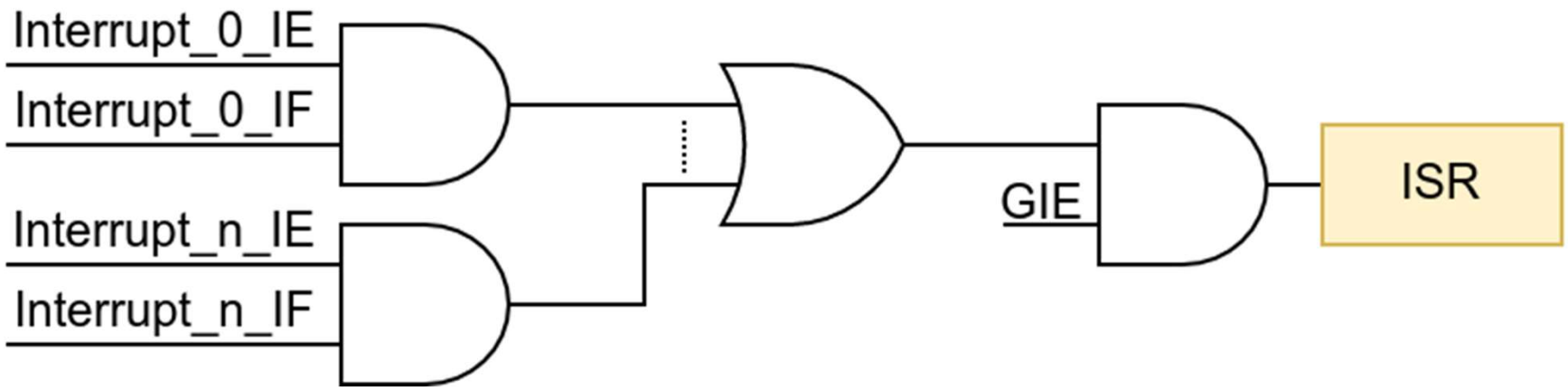
Interrupts – Main concept

There are three main concepts related to PIC interrupt:

- IE (Interrupt Enable): Registers for enabling interrupts
- IF (Interrupt Flag): Registers to see which entity has raised the interrupt
- ISR (Interrupt Service Routine): What will be performed when an interrupt is raised

Interrupts – Main concept

An interrupt is generated when both the IF and all the IE on the path are enabled. The GIE will be automatically reset and set when entering and exiting ISR. If IF is not reset for the end of ISR, a new ISR will be issued.



Remember to enable also PEIE when using peripherals

Interrupts – How to

Initialization part:

- Enable single entity interrupt that you want to be active
- Reset Interrupt Flag
- Enable Global Interrupt (GIE)

Interrupt Service Routine (ISR):

- Find what entity has raised the interrupt by checking the Interrupt Flag
- Reset the relative Interrupt Flag
- Write here your (small) code

Interrupts – Register

Setting the registers in the right way is made easy by reading the device's datasheet.

In total, there are 19+1 registers used to control interrupt:

- *INTCON, INTCON2, INTCON3 (Interrupt Control registers)*
- *PIR1, PIR2, PIR3, PIR4, PIR5 (Peripheral Interrupt registers)*
- *PIE1, PIE2, PIE3, PIE4, PIE5 (Peripheral Interrupt Enable registers)*
- *IPR1, IPR2, IPR3, IPR4, IPR5 (Interrupt Priority registers)*
- *RCON (Reset Control register)*
- *IOCB (Interrupt On Change port B register)*

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

INTCON: the first register that we will use

PORTB Interrupt On Change

Always read the datasheet before using interrupts!

Four of the PORTB pins (RB<7:4>) are individually configurable as interrupt-on-change pins. Control bits in the IOCB register enable (when set) or disable (when clear) the interrupt function for each pin.

When set, the RBIE bit of the INTCON register enables interrupts on all pins which also have their corresponding IOCB bit set.

PORTB Interrupt On Change

Always read the datasheet before using interrupts!

For enabled interrupt-on-change pins, the values are compared with the old value latched on the last read of PORTB. The 'mismatch' outputs of the last read are OR'd together to set the PORTB Change Interrupt flag bit (RBIF) in the INTCON register.

A mismatch condition will continue to set the RBIF bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.