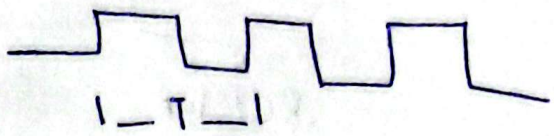
MCu clocks

↳ derive the system



Code → HIL

Hardware in the loop

Ex: clk → Timer

↑ Clock Sources

Clock tree → TRM

Heart

$F\uparrow \propto P_w\uparrow$

clk → 80 MHZ → 40 MHZ → 2. MHZ

→ 2. MHZ

MC

majority

- Power managment unit
- clock

Ex STM32 ↳ RCC

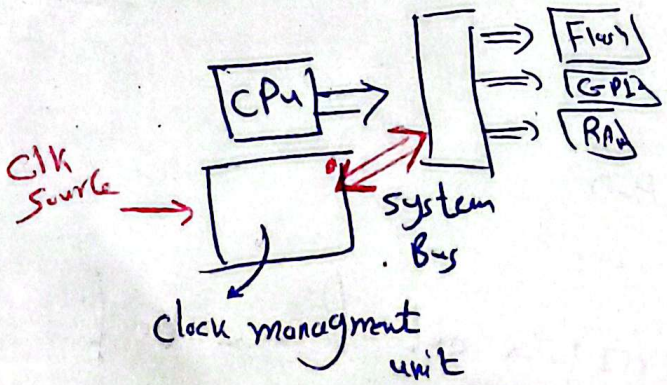
By default: all Periphrals clkX

minority

X No clk managment unit

Ex: Atmega32

⟹ General topology of clock Arch. (sysclk)



Clock

- internal
  - in mp
- external
  - RC/Chip
  - Crystal Ceramic

Mcu interrupts

Sensor
ADC

→ Alarm —X→ [mcu]

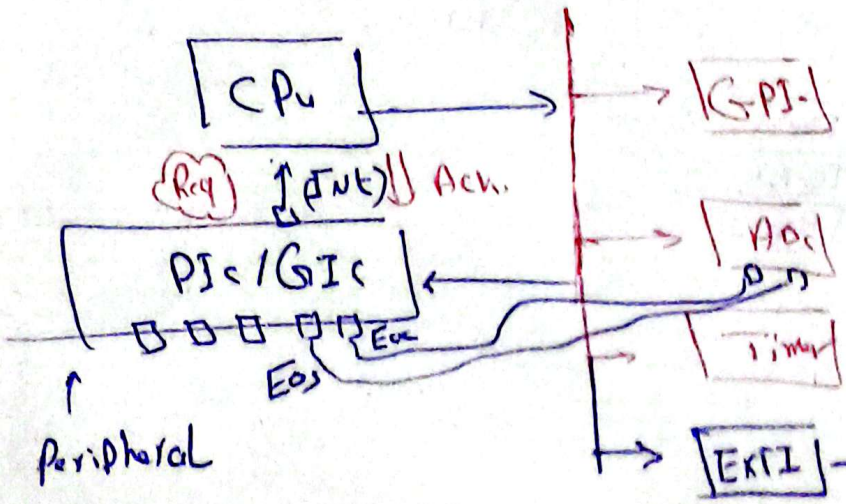
action: ISR

Signal → [cpu] → ISR → Var: Volatile

interrupt Vector table

startup Code

ADC [index | SRC | Pc] → [ ] → Function Pointer

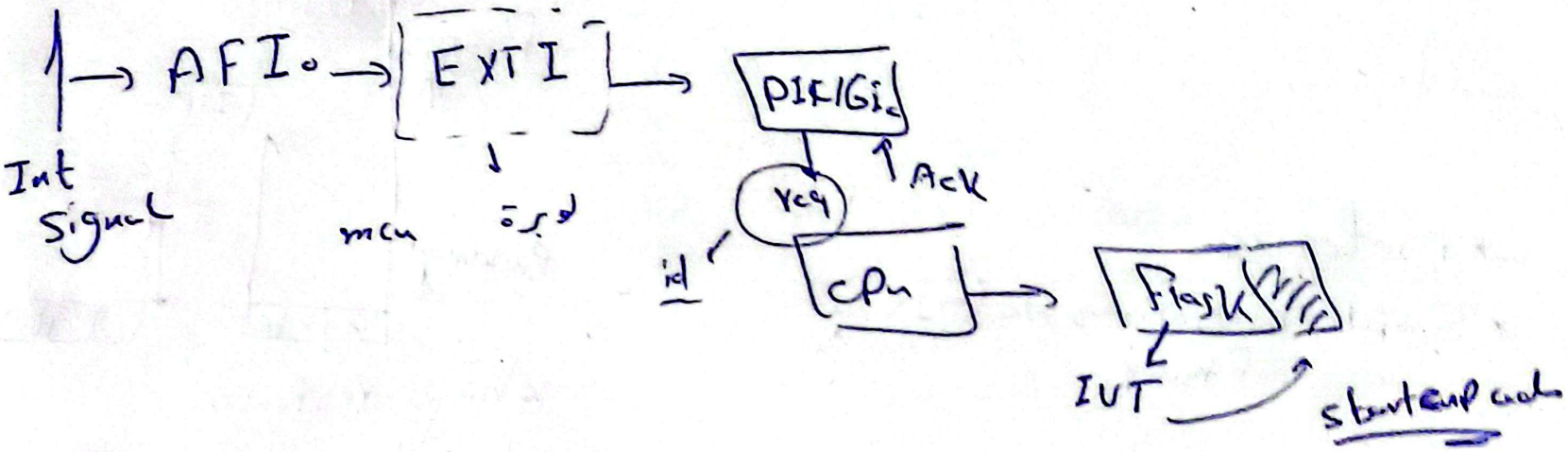


NVIC
Code Peripheral

INT

- Software
  - Faults
  - trap
  - Assembly inst
- Hardware
  - out map

AFIO → multi routing



---

① Determinism
↳ what happens at any point in time line

② Responsiveness
↳ How fast is the response to external event

System

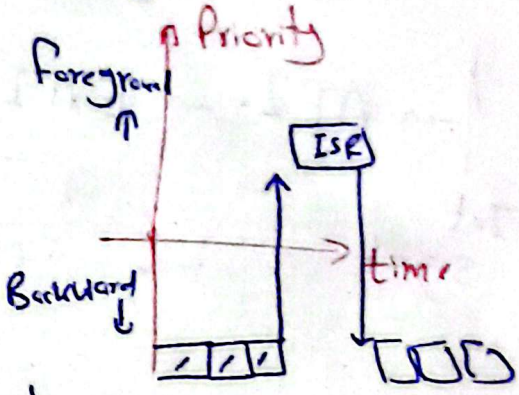
Super loop

while(1) { → } ifc

*Determinism
*Simple hardware → No GJC
" Software

x Low responsive
x high Power Consumption

Foreground / Backward



*high responsive
x Low determinism
x Extra hardware
x Complex Software

Sw mechanism
Event handling

Polling

while (1)
{
check
} Flag Register

INT

En → PIE
Flag (1)
↳ execute ISR

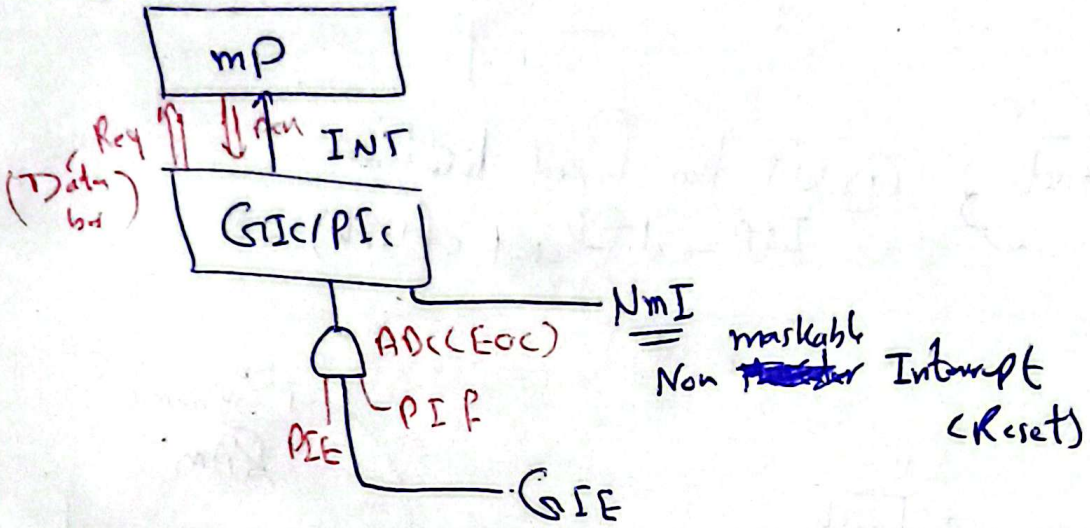
Polling

1. Waste time
2. Bad design

INT

1. No waste time
2. No Bugs

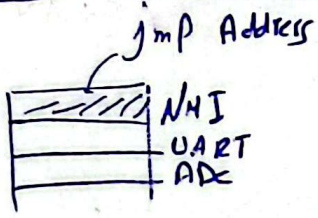
Block diagram



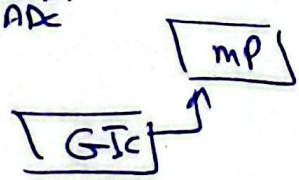
Arm جو Servicing Interrupts

Vector table Based

Fixed Priority

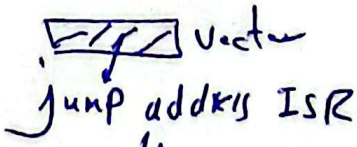


1. Peripheral —Pulse→ GIC
   GIE, PIE, Flag
2. PIC write ID → Register
3. PIC → Pulls mP hold → data bus
4. mP Read ID
5. Jump VT
6. Jump Address Source of INT (ISR)



Non-Vector Priority

Flexible Priority




```
ISR() {
    if (PIF)
}
```

Vector Based	Flexible
Hardware dependent	Software dependent
Time latency ↓	time latency ↑
event → ISR	

Facts
① U.T has fixed location
② ISR → different (flash)

U.T

Static
Flash
No change @ Run time
→ Burner

Faster
No need to disable GIE

Dynamic
RAM
change @ run time
start up Code

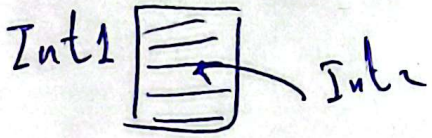
× disable INT
→ ISR [ ] Ram } start up Code
× Enable Int

Int. nesting

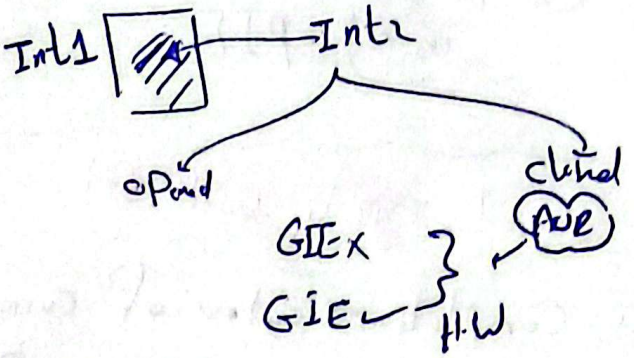
- Not support (x) — sequential
- Support (✓) → if $int_2 > int_1$ in Priority

Nesting

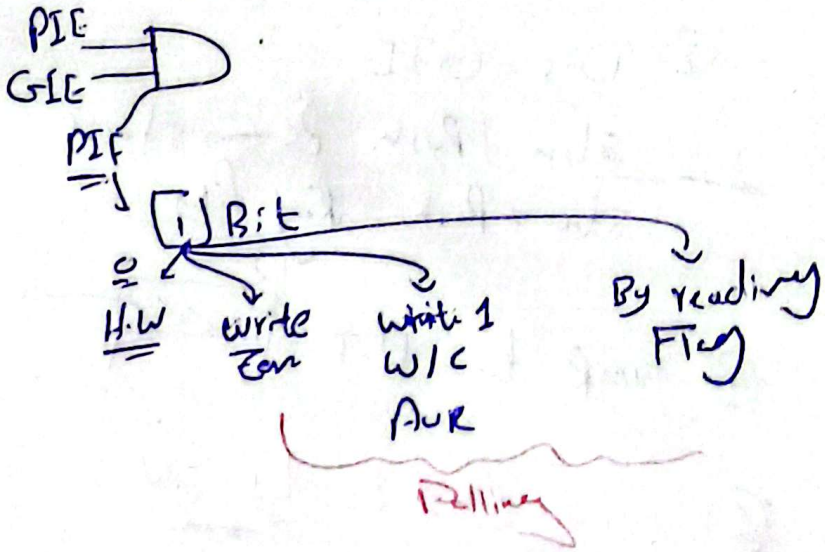
- Self



- Normal



Flag Clearance



Int

- internal
  Software
- external
  Hardware

Int

- maskable
  any Periph
  Can be disabled using PIE
- Non maskable
  Reset
  Not affected neither by PIE nor GIE

Inst-Cycle with Int

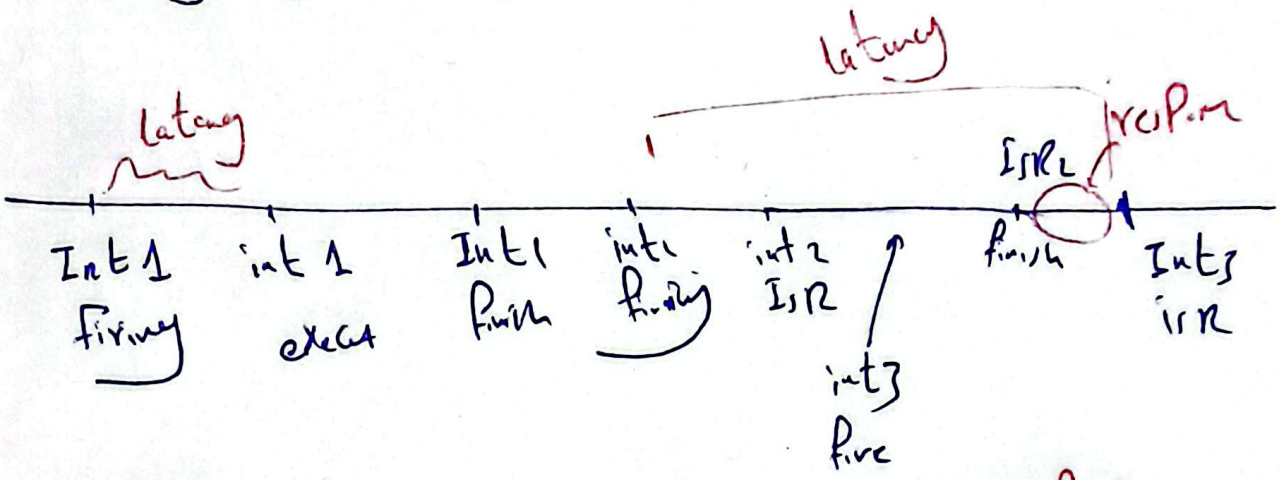
1. Complete execution of current instr.
2. Stop main & clear PIF
3. Dis-GIE

H.W Store / Push PC → stack
SW Store / Push Reg files

4. jump to V.T  5. jump to ISR

6. enable GIE by SW
7. Pop Reg. & PC
8. Continue main Program

Latency & response



Latency = response − waiting time

Latency ≈ 0 → Synchrony × Asynchrony