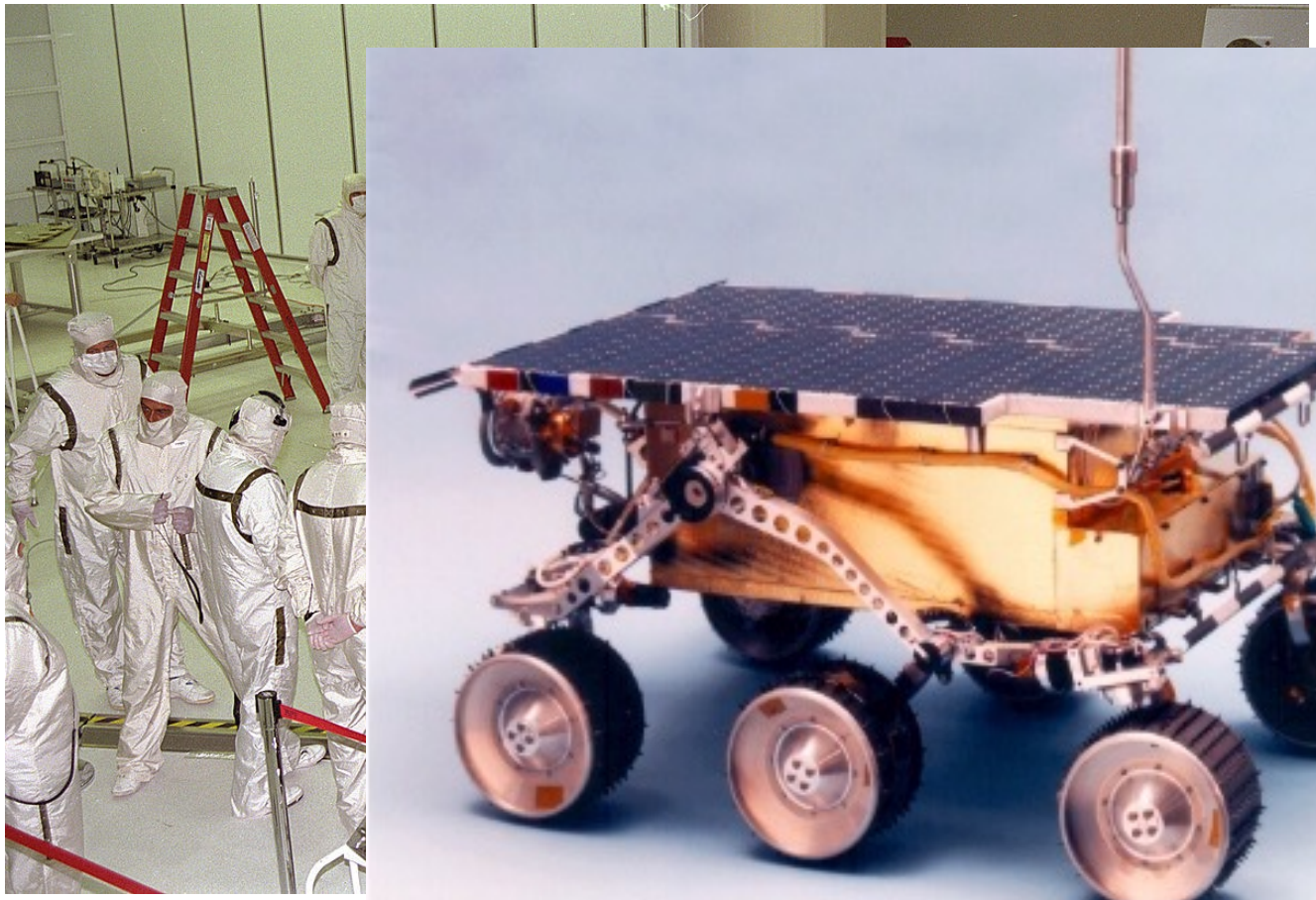
The Mars Pathfinder (1997)

Before



After

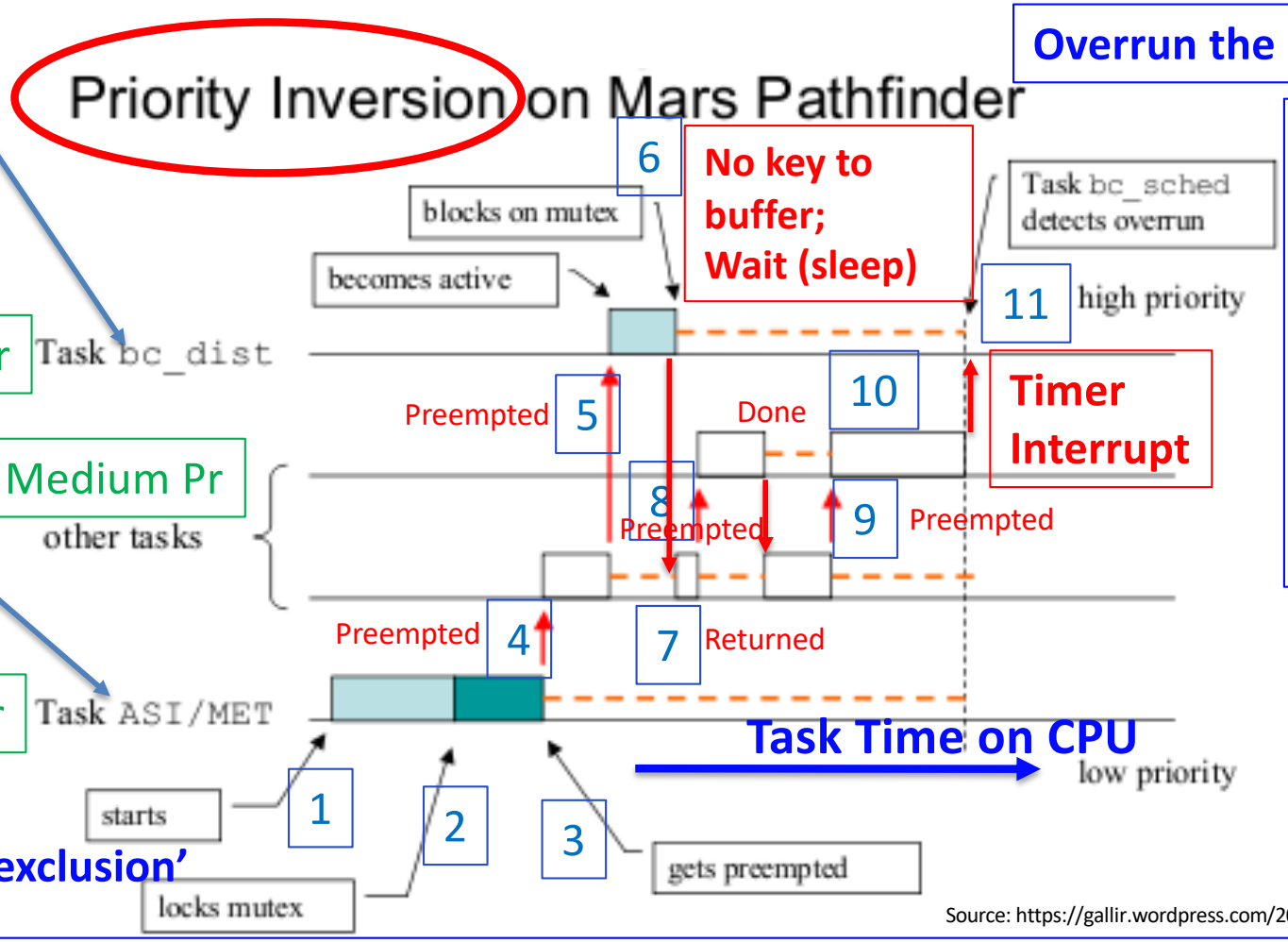
Source: mars.jpl.nasa.gov

Landed on Mars:

- **System kept resetting; why?**
- **Spent one week to debug, remotely**

CPU Sharing with Prioritized Tasks

Sensors collect environmental data; both tasks use the same input buffer



No key to buffer; Wait (sleep)

Overrun the deadline

Timer Interrupt

- Use Timer to detect overrun
- If Timer expired → Task not active/done → problems!

RESET SYSTEM

mutex = 'mutual exclusion'

Source: https://gallir.wordpress.com/2009/07/23/el-bug-marciano/

buffer access by one task at a time; entry by a single key to the buffer lock; must get key first

Mission Critical Embedded System

- Small footprint, light weight → require resource sharing
  - Processor: shared among tasks with priority interrupts and pre-emption
  - Memory: partially implemented, memory-mapped, sharing buffers, etc.
- Single resource: only one task can use at a time
  - Need mutual exclusion mechanism: use a single key to unlock
- No response from task → something is wrong
  - Use timer: reset when count down expired

455 Real Time System Design Project

Computer-Controlled Foosball Player

Need to Know:

1. **Where is the Ball ?**
2. **Who should Defend ?**
3. **Who should Attack ?**
4. **Where to Move ?**

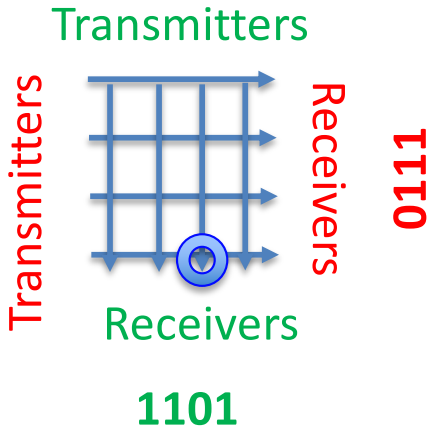
Design Objectives:

1. Track the movement of the ball and move the players to prevent a goal being scored
2. When possible, the players attempt to kick the ball to score a goal

Autonomous Foosball Table

Input sensing:
- **Players positions**
- **Ball position**

Output moving:
- **Position**
- **Speed & Strength**
- **Block or Kick**



How does the grid work ?

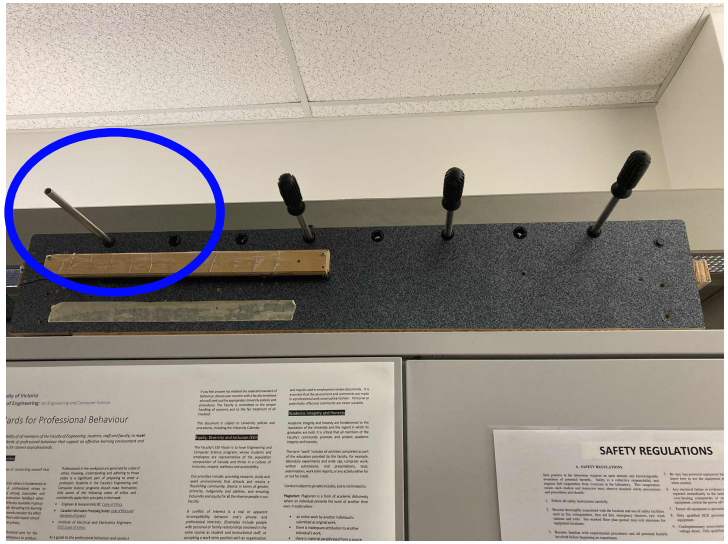
Implementation:

1. Input: A grid of lasers are used to determine the position of the ball
2. Output: Motor drivers are used to control the movement of the goalie stick

Project Hardware

Laser grid to detect ball position

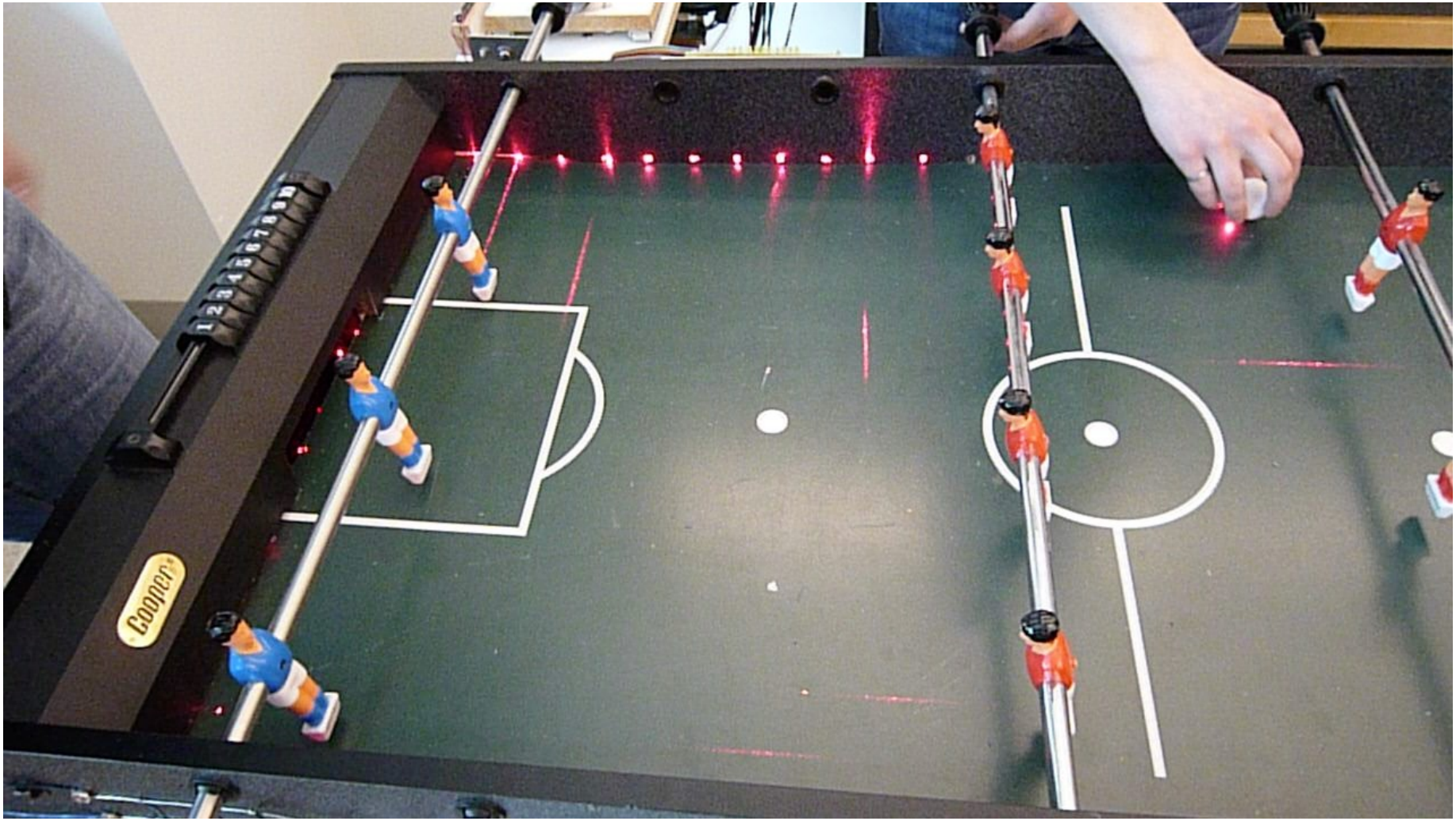
Computer controls one stick with two motors



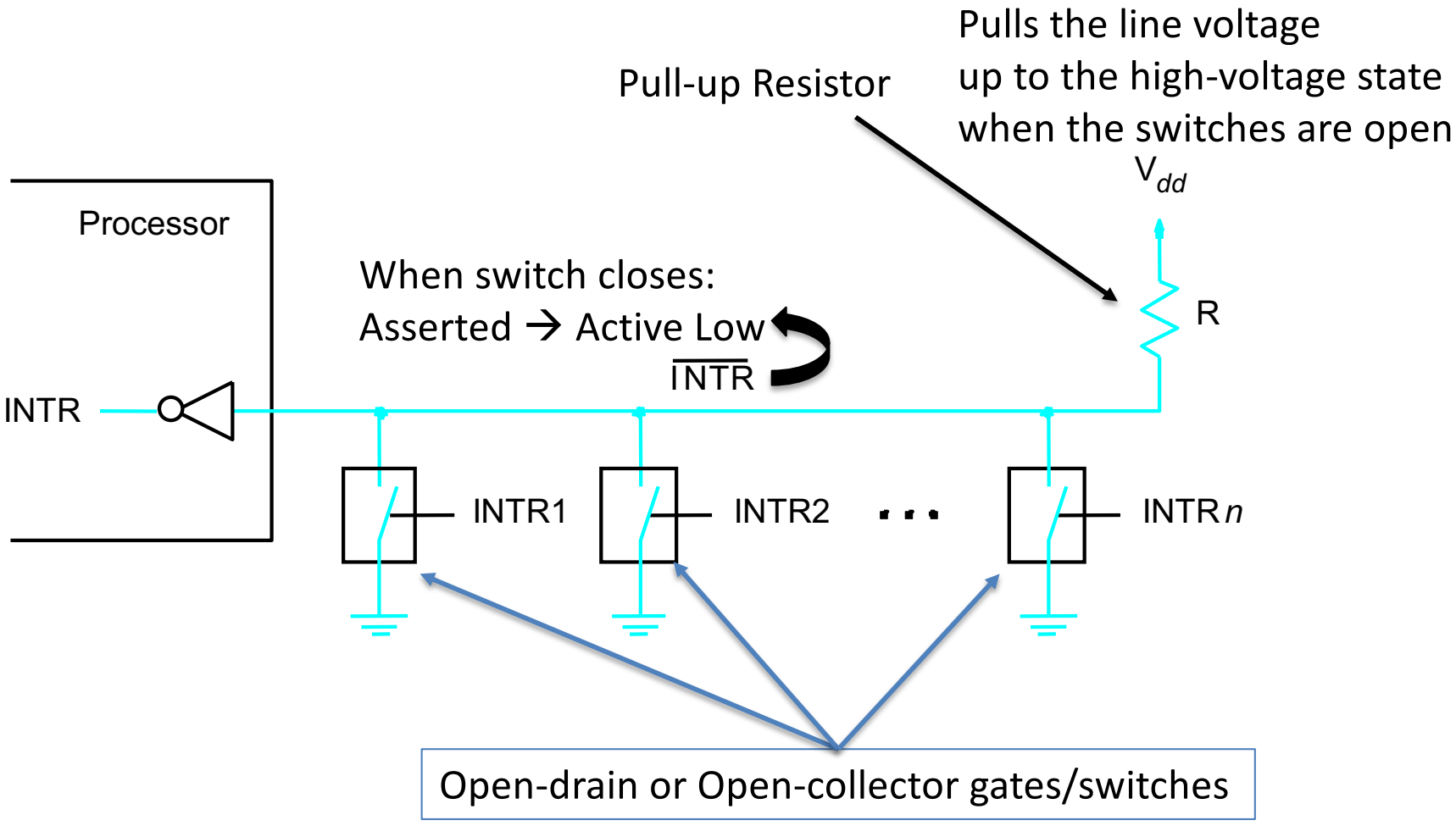
Data:

A. What: Image, speed, coordinates ?
B. How: wired, wireless ?
C. When: polling, interrupts ?

Project Testing & Exam

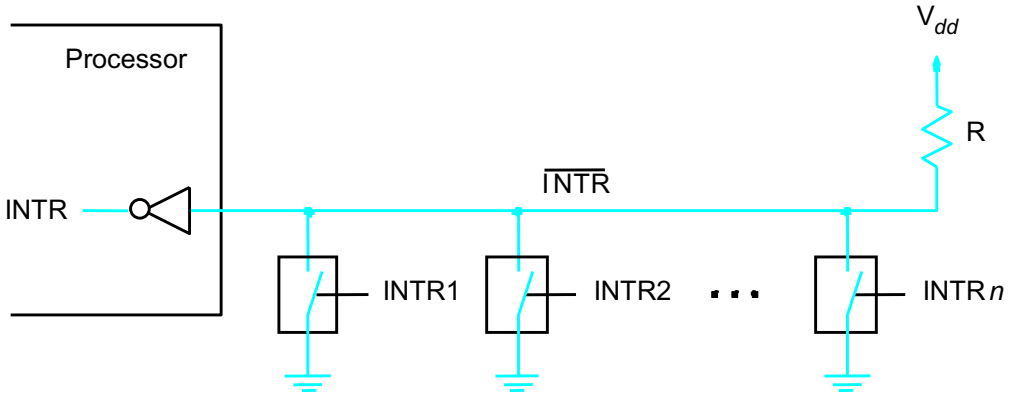


3.2.2 Multiple-Device INTR HW



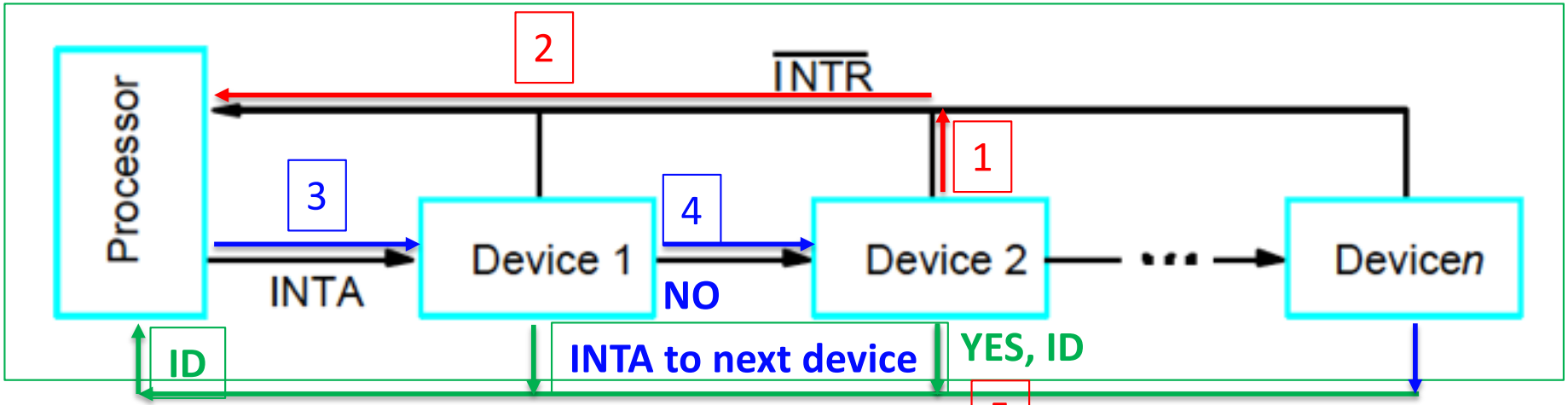
Daisy Chain & Self Identification

- Let devices identify themselves

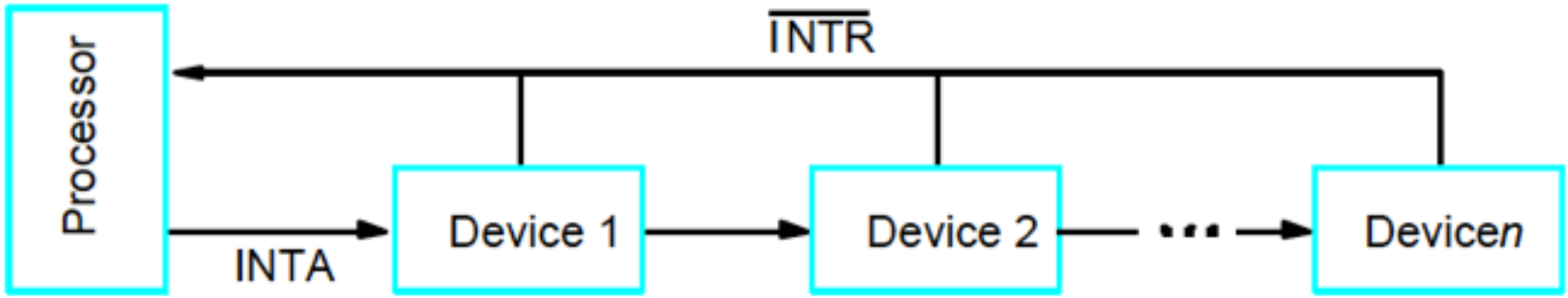


Potential Problem ?

Starvation:
If Device 1 makes INTR continuously



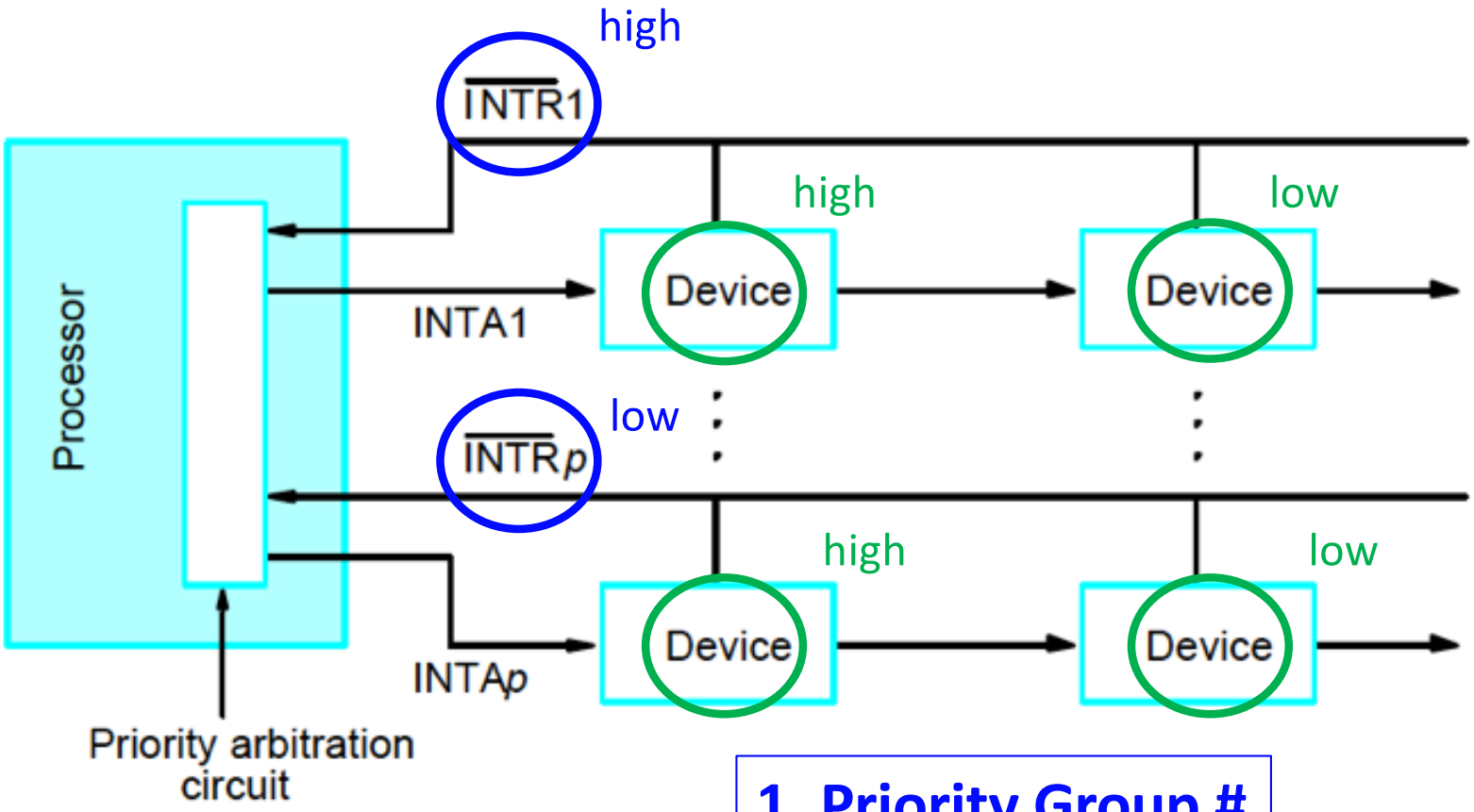
Daisy Chain Priority



Priority of the devices ?

Proximity to Processor

Daisy Chain Priority Groups

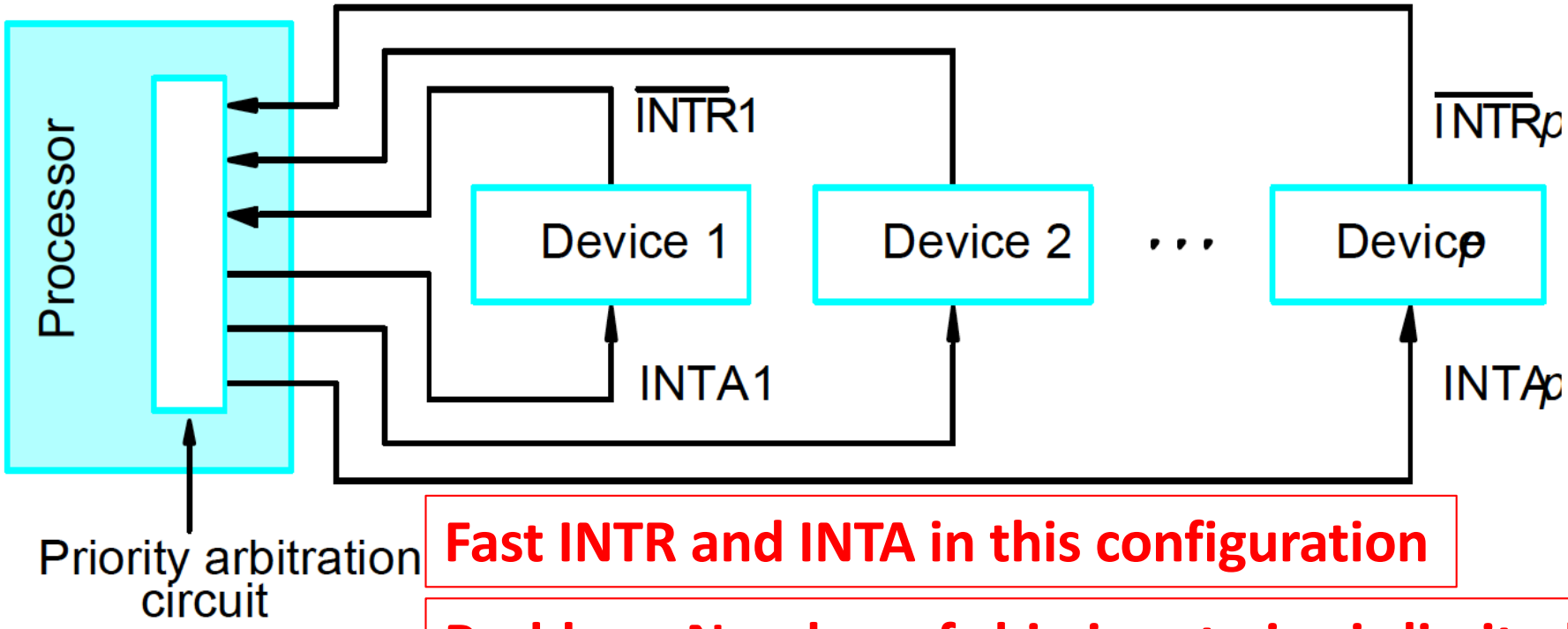


Priority of the devices ?

1. Priority Group #

2. Proximity to Processor

Multiple Interrupts: $INTR_i/INTA_i$



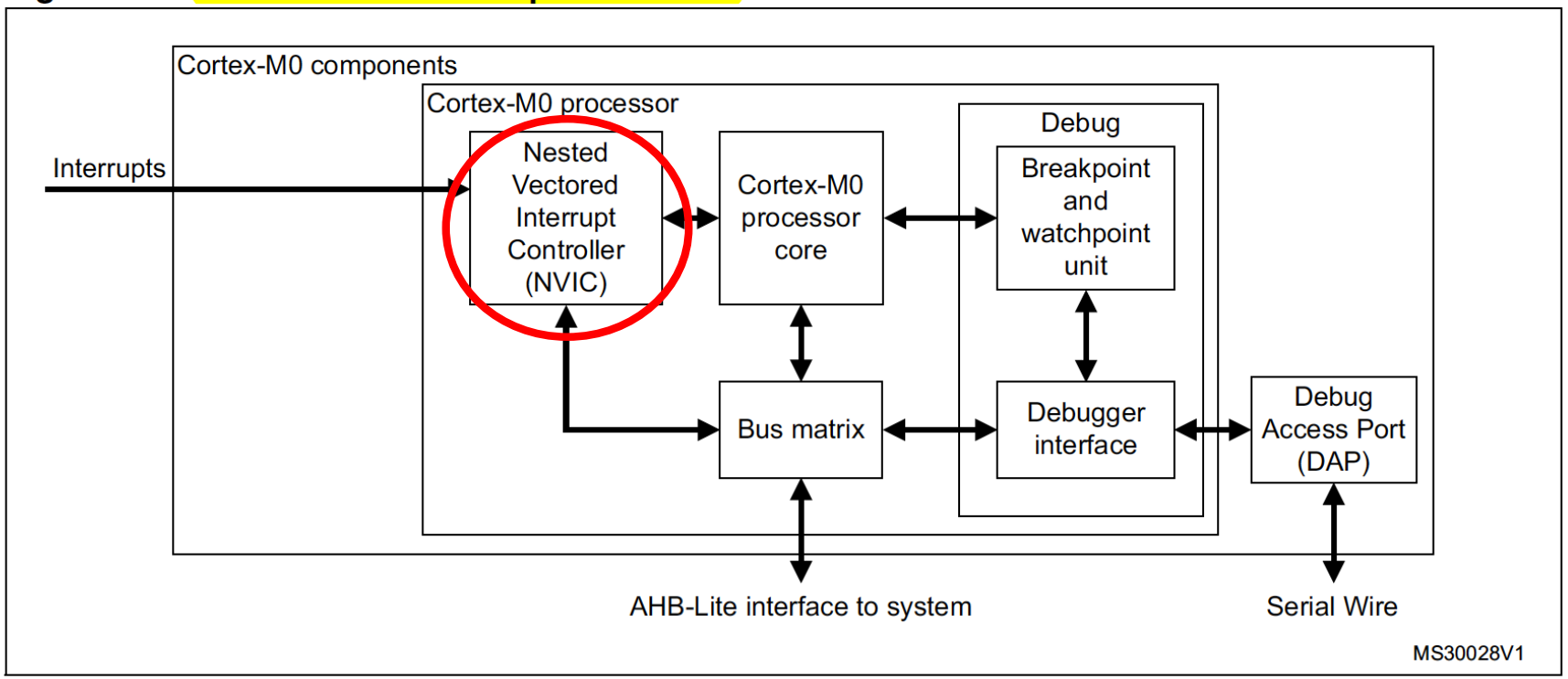
Fast INTR and INTA in this configuration

Problem: Number of chip input pins is limited

Solution: Use dedicated I/O Interface Chip
e.g., Cortex Nested Vector Interrupt Controller

Cortex NVIC (PM-9)

Figure 1. STM32 Cortex-M0 implementation



Cortex NVIC (ARM site)

6.1 NVIC functional description

The NVIC supports up to 240 interrupts, each with up to 256 levels of priority that can be changed dynamically. The processor and NVIC can be put into a very low-power sleep mode, leaving the Wake Up Controller (WIC) to identify and prioritize interrupts. Also, the processor supports both level and pulse interrupts.

Table 6-1 NVIC registers

Address	Name	Type	Reset	Description
0xE000E004	ICTR	RO	-	Interrupt Controller Type Register, ICTR
0xE000E100 - 0xE000E11C	NVIC_ISER0 - NVIC_ISER7	RW	0x00000000	Interrupt Set-Enable Registers
0xE000E180 - 0xE000E19C	NVIC_ICER0 - NVIC_ICER7	RW	0x00000000	Interrupt Clear-Enable Registers
0xE000E200 - 0xE000E21C	NVIC_ISPR0 - NVIC_ISPR7	RW	0x00000000	Interrupt Set-Pending Registers
0xE000E280- 0xE000E29C	NVIC_ICPR0 - NVIC_ICPR7	RW	0x00000000	Interrupt Clear-Pending Registers
0xE000E300 - 0xE000E31C	NVIC_IABR0 - NVIC_IABR7	RO	0x00000000	Interrupt Active Bit Register
0xE000E400- 0xE000E4EC	NVIC_IPR0 - NVIC_IPR59	RW	0x00000000	Interrupt Priority Register