

Advanced Hardware Fundamentals



Simon Chapter 3

Microprocessors - basic system and connections,
busses, address space, interrupt connections, UARTS,
watchdog timer, basic internal elements incl. timers
and I/O

Microprocessors

- Fundamental signals
 - Address lines
 - Data lines
 - READ/ - to get data in
 - WRITE/ - to send data out
 - Clock – heartbeat to pace the system
- “Microprocessor” vs. “Microcontroller”
 - Location/amount of RAM/ROM
 - No difference in programming
 - “microprocessor” becoming generic term for all

Buses

- A sample system has the following components:
 - A microprocessor with 64K address space
 - A ROM with 64K of memory
 - A RAM with 32 K of memory
- What signals do we need?

- Address Space?

Address Space

**Microprocessor
Addresses**

**RAM
Addresses**

**ROM
Addresses**

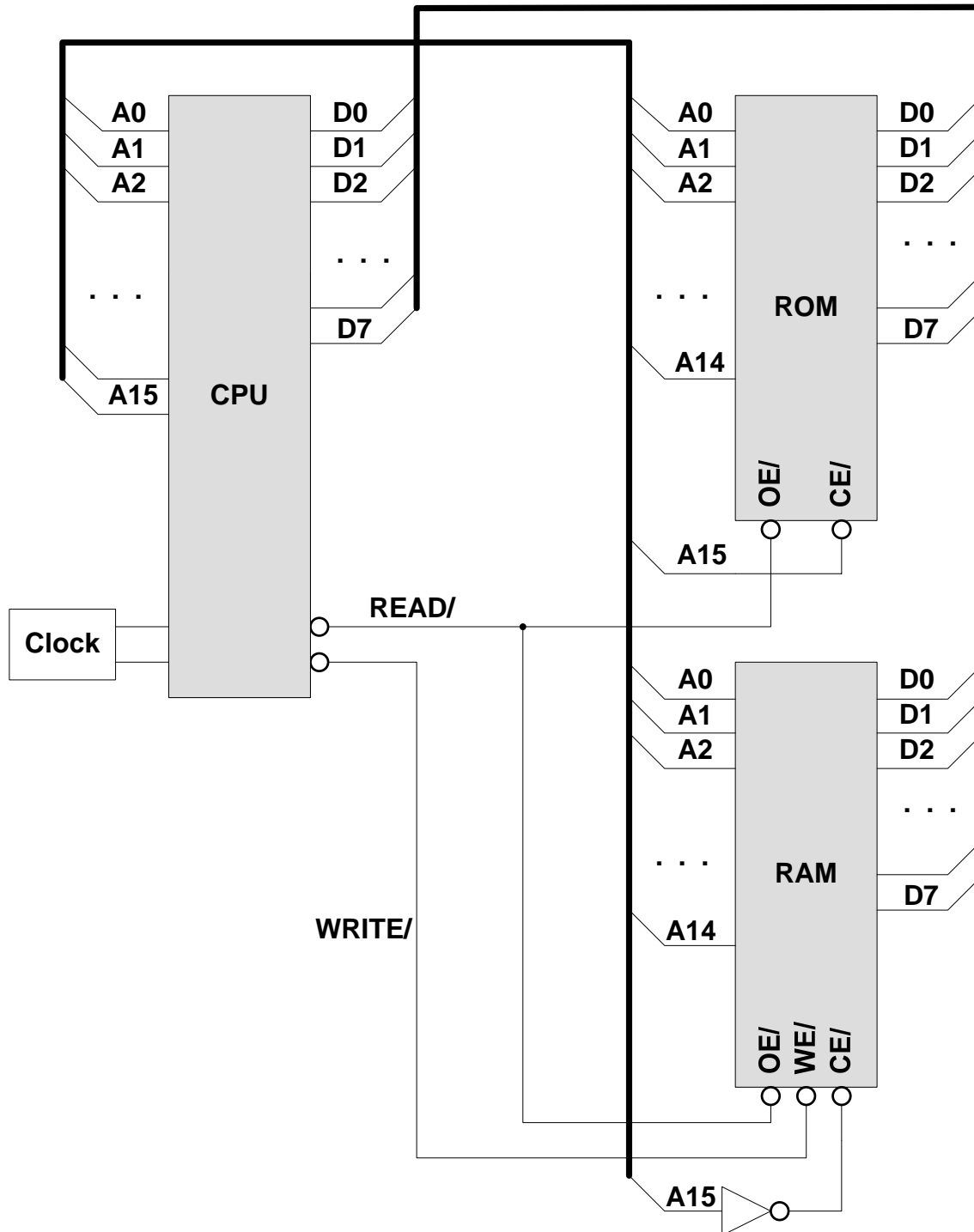


Fig 3.2

Adding Devices on the Bus

- Memory Mapping
 - Cut up the memory “pie”
 - Hardware engineer defines assert lines for Chip Enable
 - Devices “look like” additional memory to the microprocessor

Memory Mapping Example

```
#define NETWORK_CHIP_STATUS ((BYTE *) 0x80000)
...
void vFunction()
{
    BYTE byStatus;
    BYTE *p_byHardware;

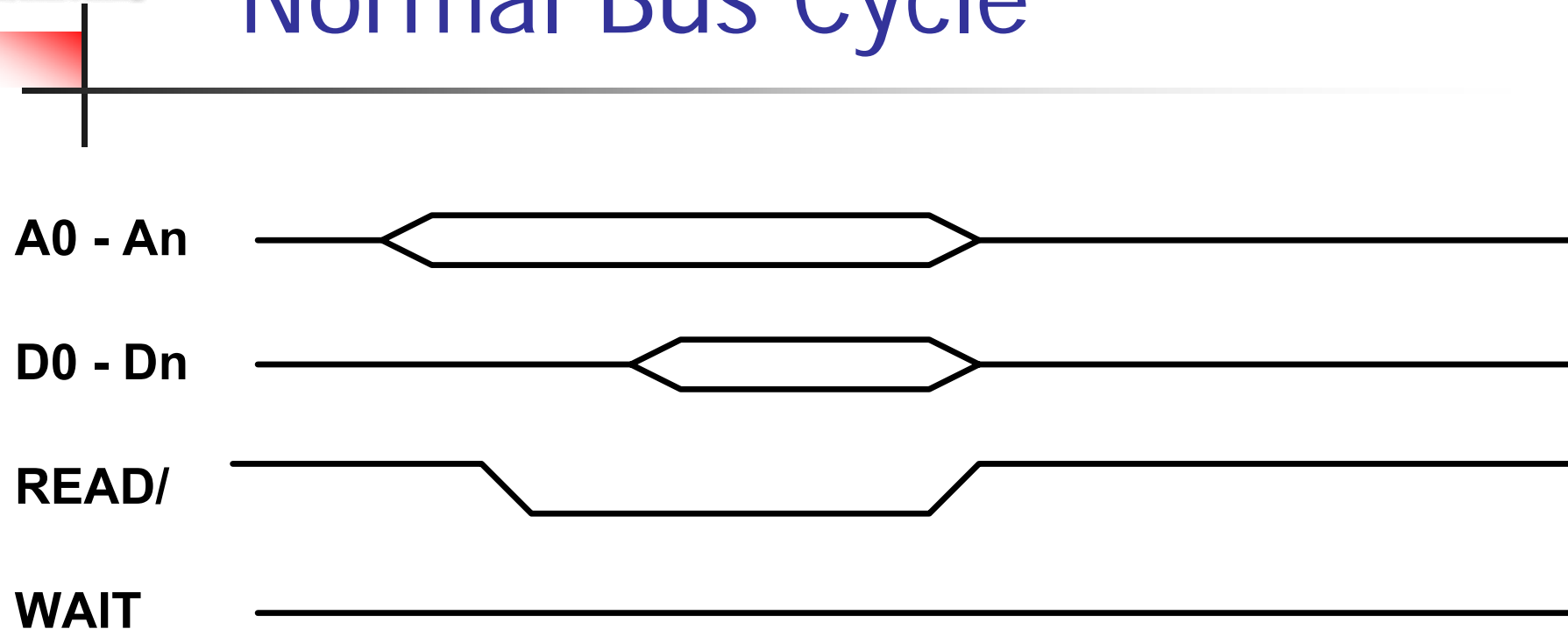
    /* Set up a pointer to the network chip */
    *p_byHardware = NETWORK_CHIP_STATUS;

    /* Read the status from the network chip */
    byStatus = *p_byHardware;
}
```

Bus Handshaking

- Hooking up address and data lines requires proper **timing**
 - Address lines must be stable for used
 - Data on pins must be stable before read/written
- “Bus Cycle”
 - Assuring timing requirements met
 - Bus **handshaking**
- Types of Handshaking
 - No handshake
 - Wait signal
 - Wait states

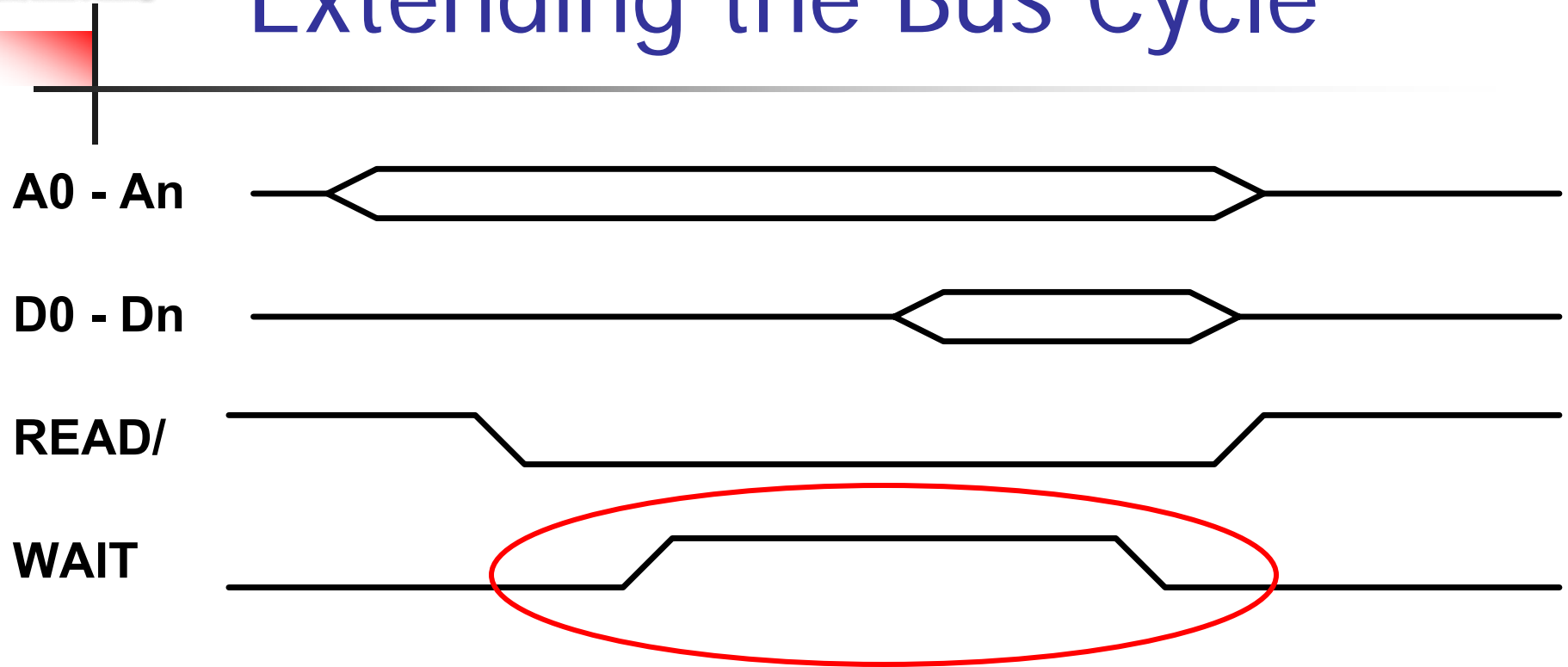
Normal Bus Cycle



Order of Events

- 1) uP puts address on the address bus
- 2) uP asserts READ/ line to tell RAM to get the requested data
- 3) RAM puts data on the data bus
- 4) uP reads data, then disables READ/ to end the bus cycle

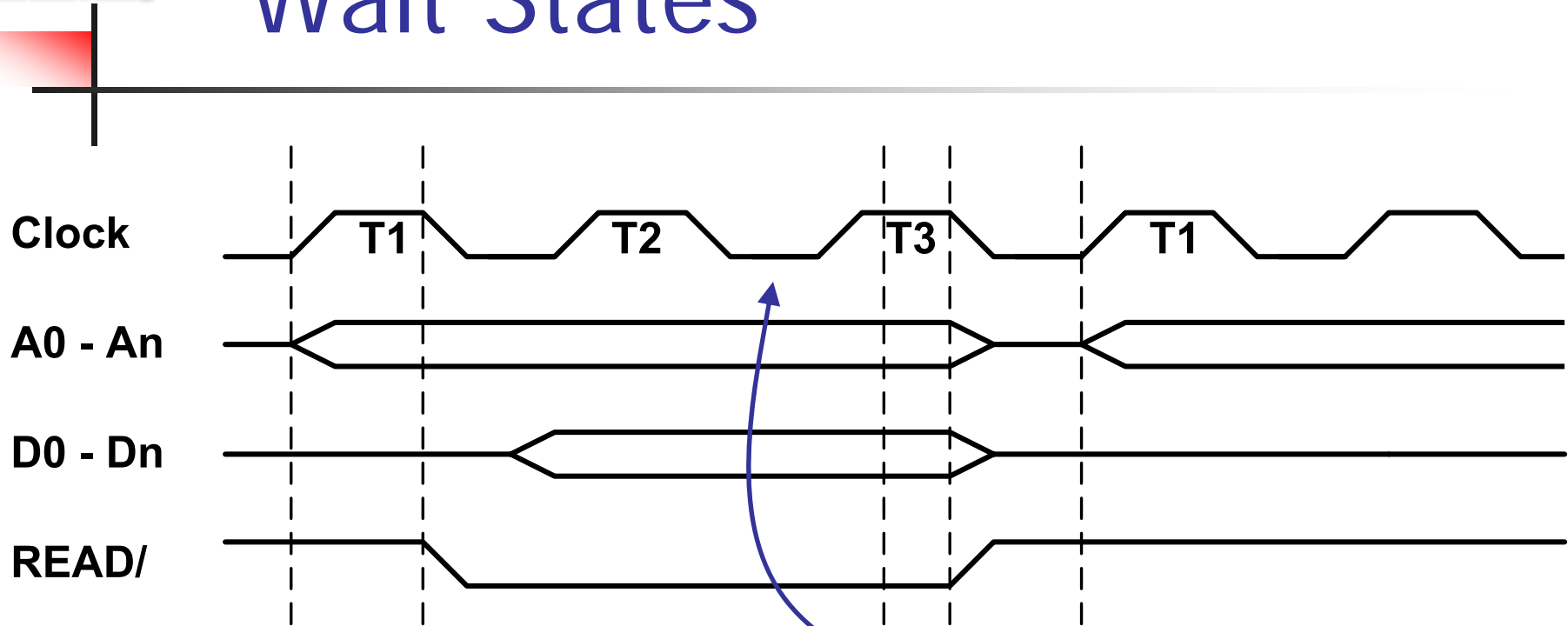
Extending the Bus Cycle



Same Order of Events....except device can request that uP wait as long as it needs to. The uP will wait.

Disadvantage – WAIT must be custom designed. Not a part of a device chip.

Wait States



extra waits
states can be
added here

Order of Events

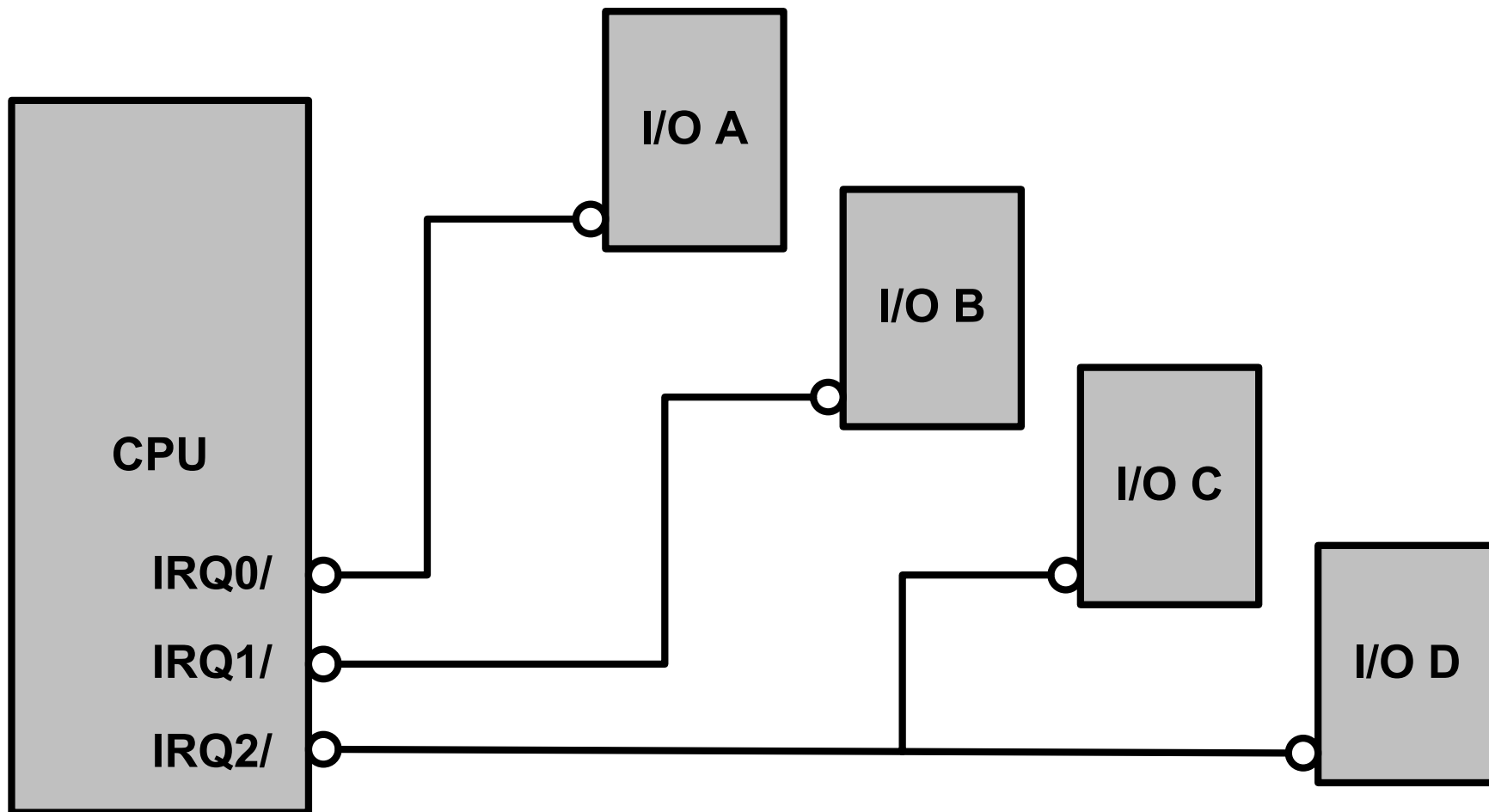
- 1) T1 rising edge - address on the address bus
- 2) T1 falling edge - READ/ line asserted
- 3) uP expects valid data during T3 HIGH
- 4) T3 falling edge - De-asserts READ/

Interrupts

- Interrupt Request – IRQ
 - To interrupt processing
 - uP may have several IRQs
 - Asserted LOW
 - May be level or edge triggered
 - Device IRQs are open-collector – share the line

- Interrupt Service Routine
 - Software to service interrupt
 - Must execute efficiently

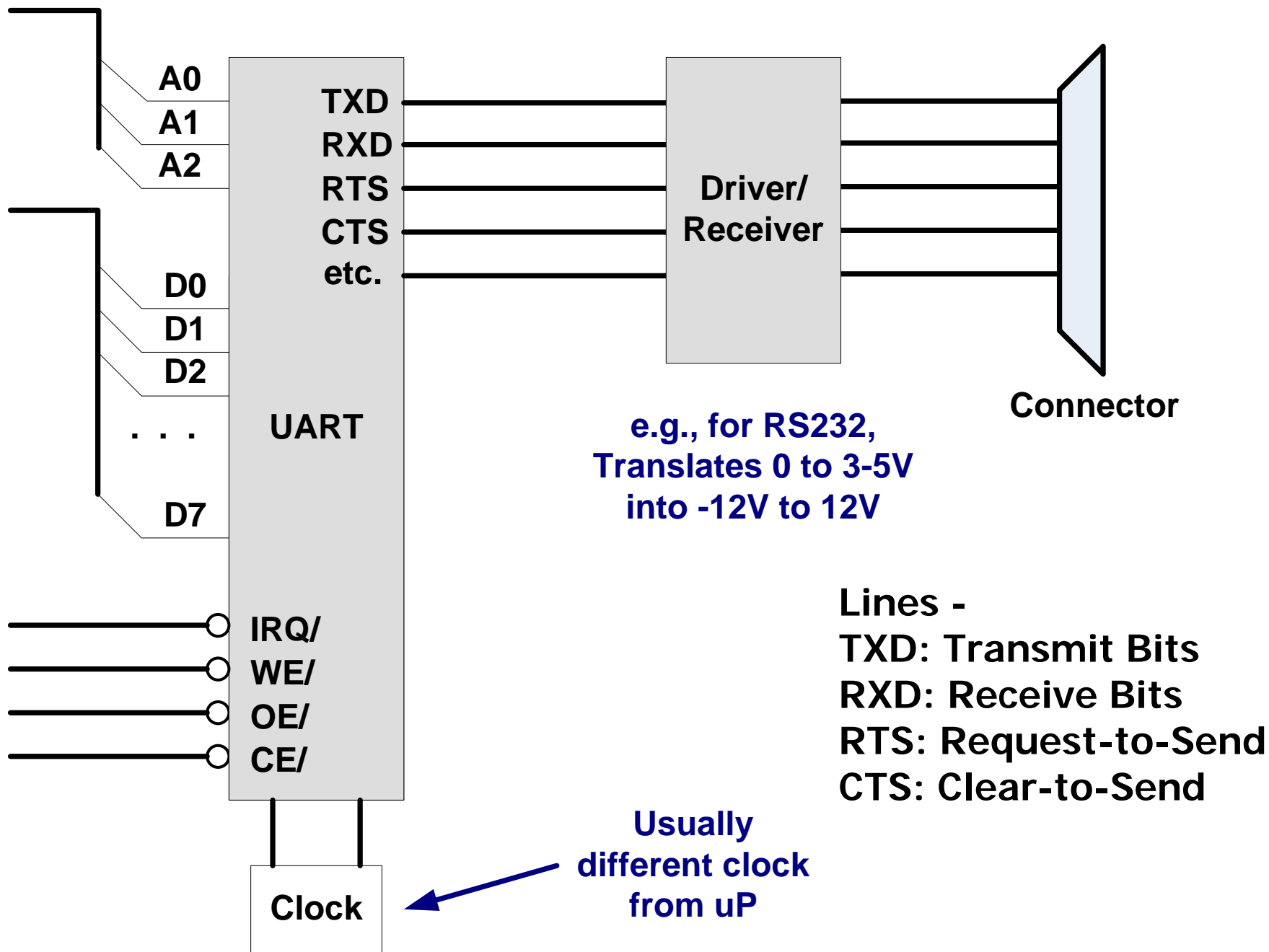
Interrupt Connections



Hardware will include pull-ups

UARTs and RS-232

- Universal Asynchronous Receiver-Transmitter
 - Transmit data via serial interface: byte by byte
 - Can look like memory to the uP
 - Separate clock, a multiple of common bit rates
14.7456 MHz (even multiple of 28,8000)
 - Translates LOW and HIGH levels to appropriate voltages levels
 - uP: 0V to 3V
 - RS-232: -12V to 12V
 - Registers for data storage (FIFO)
 - Error conditions (parity, framing)
 - Configuration registers (data rate, level values)



Glue Circuitry

- Connecting inputs, outputs, changing assertion levels, combining signals, etc.
- Programmable Logic Devices (PLDs)
- Programmable Logic Arrays (PALs)
 - Create custom connections between parts and pins
 - PAL programmer
 - Code defines INPUTS, OUTPUTS, and Equations
- Ultimate Customization – ASICS and FPGAs
 - Application Specific Integrated Circuits (Core-based)
 - Field Programmable Gate Arrays (PAL on steroids)

Watchdog Timer

- External timer that will expire after defined time
- Output: Line will pulse if timer expires
- Inputs: HW/SW logic that is allowed to restart the timer
- If the timer ever expires, the logic failed to restart the timer

WHY?

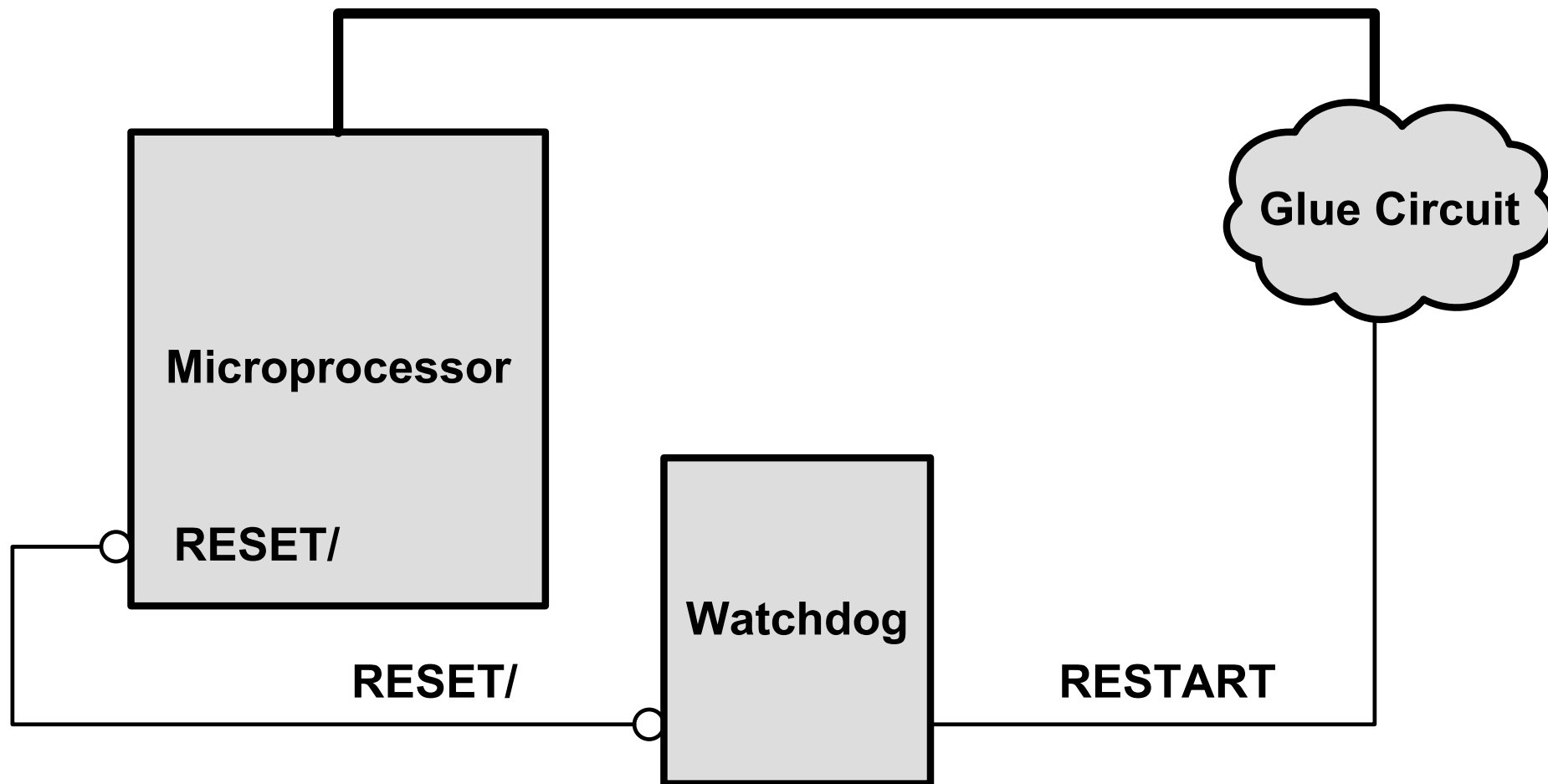
- Because the software probably crashed.

Therefore.....

- Use the WD output line to reset the microprocessor.

Watchdog Timer Typical Use

Data, address, READ/, and WRITE/



Use logic to periodically "Kick the Dog"

Rights and Wrongs

- **WRONG**

- Do not use the ISR to kick the dog...why not?

- **RIGHT**

- Use "regular" code
- Use external logic

Microprocessor Built-Ins

- Auxiliary circuits (peripherals) - same silicon
- Generally appear in uP memory map

- Timers
- DMA – Direct Memory Access
- I/O Pins
- Address Decoding
- Memory Caches and Instruction Pipelines

Timers

- A counter that can be preset
- Prescaler - multiples of the clock frequency
- Generally counts down to zero, the triggers interrupt
- Can be one-shot or periodic
- Can be used to drive an output pin

- **VERY USEFUL!**

Input/Output (I/O) Pins

- Can be configured as Input or Output
- Pin voltage changed by writing to a register
- Purposes
 - Turning LEDs on/off
 - Resetting Watchdog Timer
 - Reading EEPROM
 - Switching between banks of RAM
 - Keypads
 - LCDs, etc.

Sample I/O

