Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

07 – Interrupts

CS1022 – Introduction to Computing II

Dr Adam Taylor / adam.taylor@tcd.ie
School of Computer Science and
Statistics

Polling or Interrupts?

Want to write a program to take some action when the button is pressed

Approach 1

suppose a memory location (e.g. 0xE1000000) is set to 1 when the button is pressed

keep reading 0xE1000000 until we read 0

do nothing between tests

Approach 2

keep reading 0xE1000000 until we read 0

do useful things between tests

Polling

Approach 3

do useful things

break out of F-D-E cycle when event occurs

Interrupts

Interrupts

$$\left.\begin{array}{l}\text{Interrupt ReQuest (IRQs)}\\ \text{Fast Interrupt reQuest (FIQs)}\end{array}\right\}\begin{array}{l}\text{Exception}\\ \text{Classes}\end{array}$$

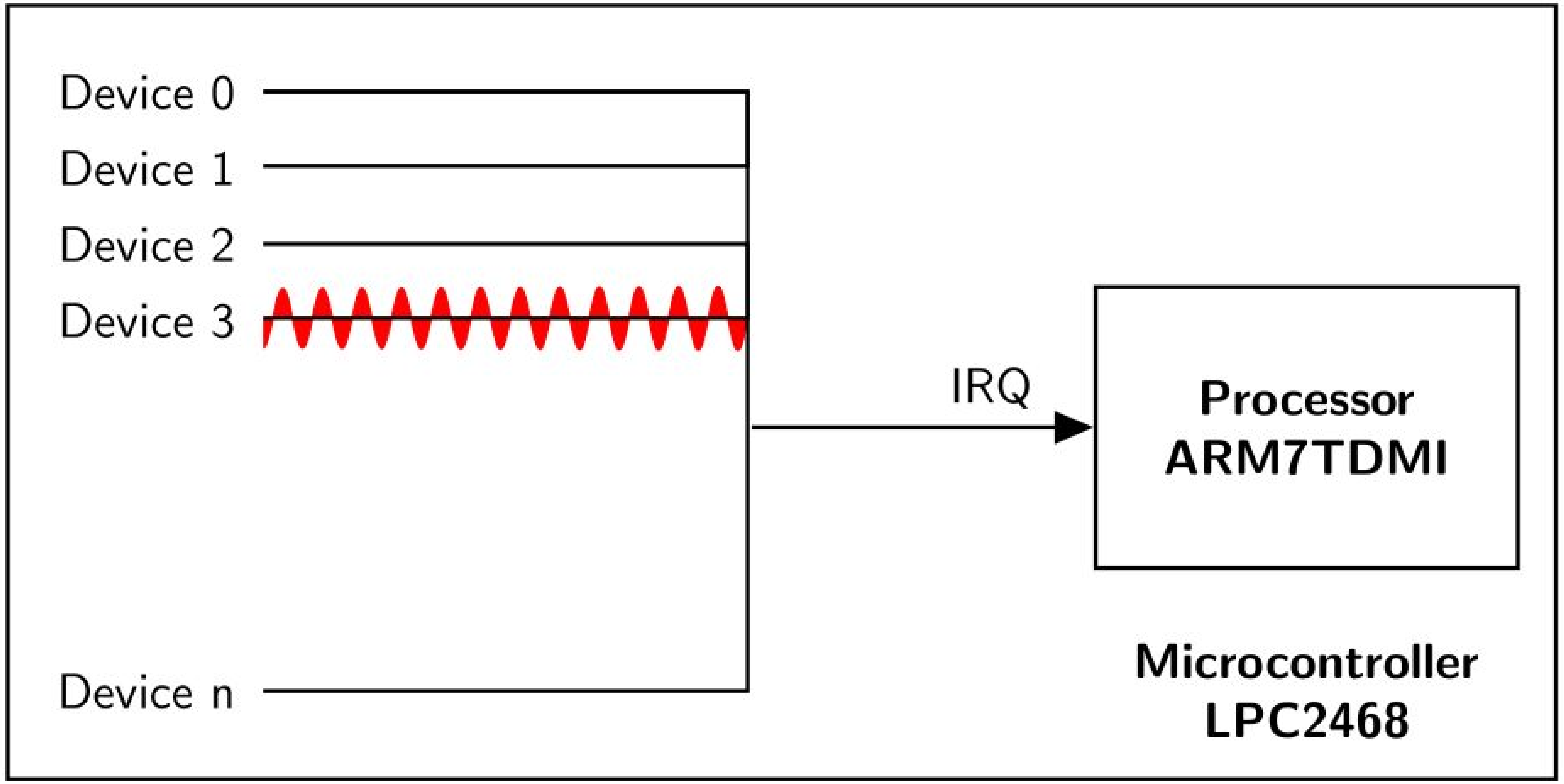
Interrupts are “raised” by devices external to the CPU when events of interest occur

- e.g. mouse movement, key press, hardware timer expired

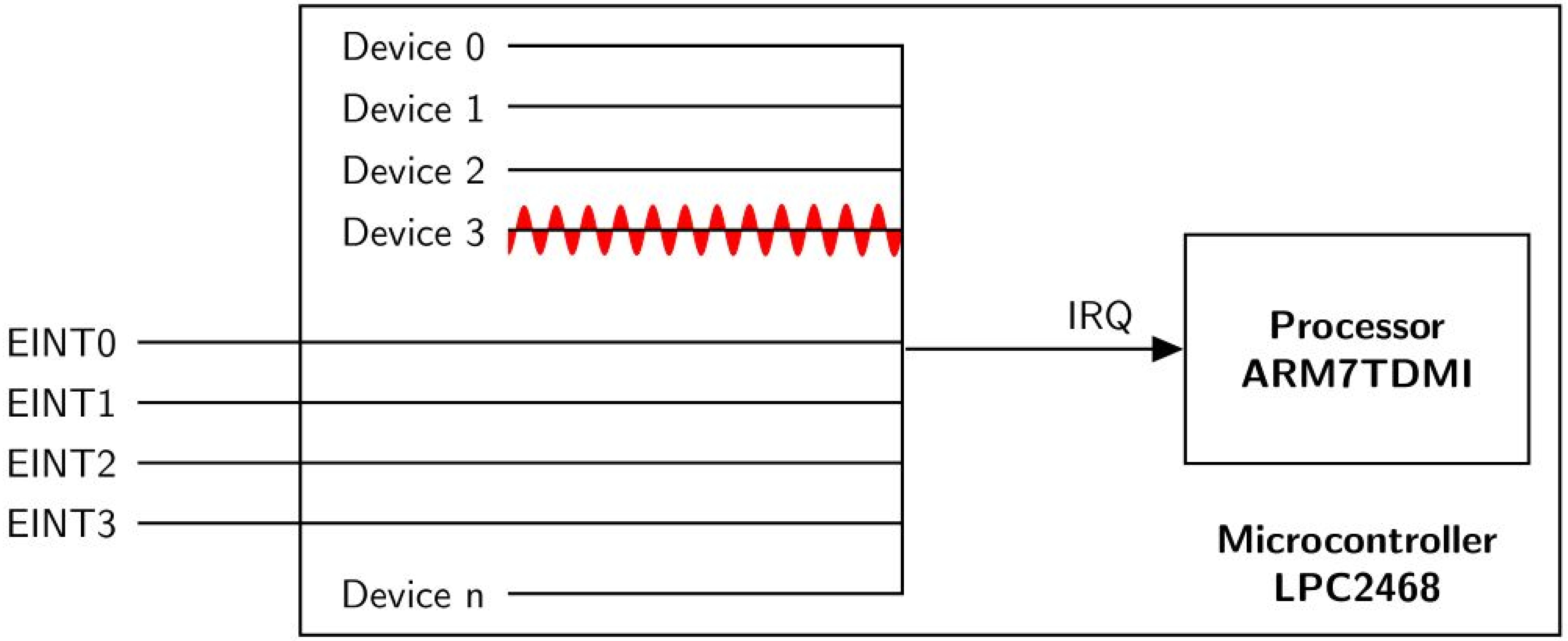
When an IRQ or FIQ occurs, the IRQ or FIQ exception handler is executed

- Interrupt could have been “raised” by any one of the devices connected to the CPU
- Handler must determine which device raised the interrupt and execute a further handler for that device

Raising an Interrupt



Raising an Interrupt



Raising an Interrupt

How can we determine which device raised the interrupt?

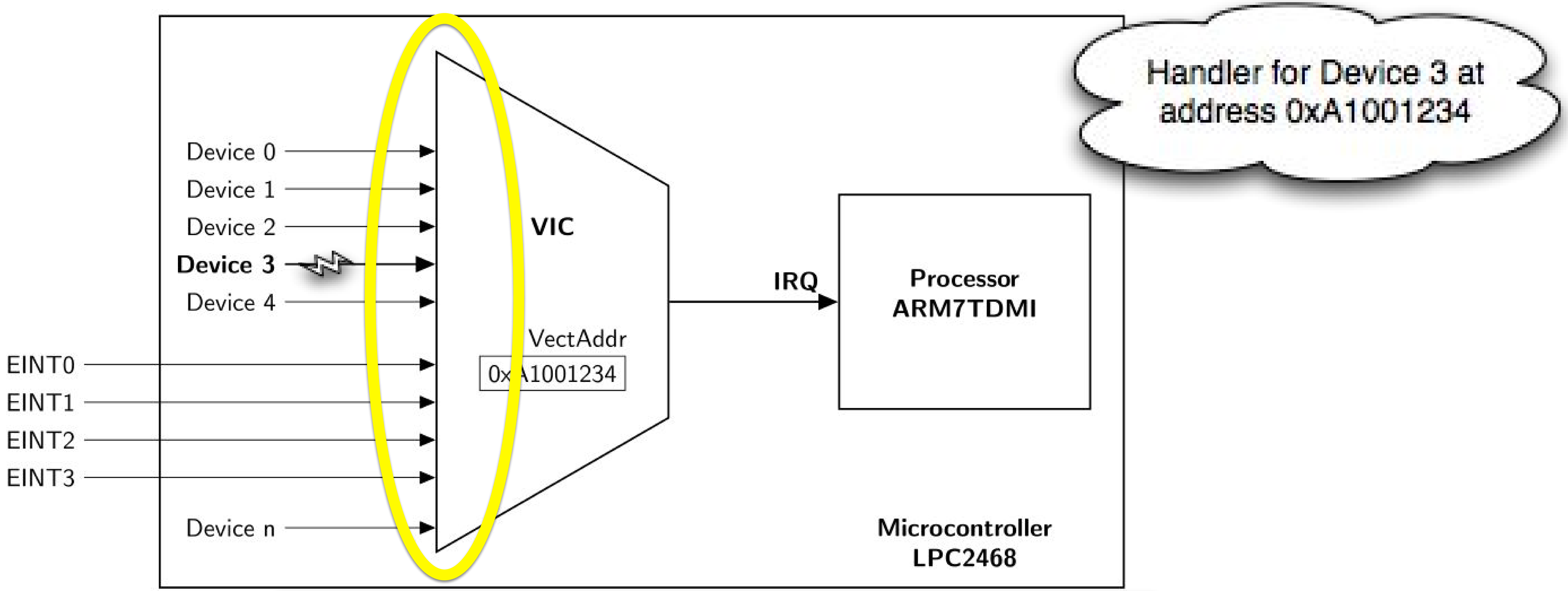
"Ask" each device (e.g. read a memory-mapped status register) whether it has a pending interrupt

- Takes multiple cycles (multiple LDRs) to determine source of interrupt
- Too slow for IRQs
- Way to slow for FIQs

Get assistance from an external "helper" device

- *Vectored Interrupt Controller ("VIC")*

Raising an Interrupt – Vectored Interrupt Controller



```
LDR PC, [address of VIC VectAddr register]
```

```
LDR PC, [PC, #-0x0120]
```

LPC2468 Vectored Interrupt Controller

Up to 32 interrupt sources

- Internal (e.g. TIMER, ADC)
- External (EINT0/1/2/3, e.g. P2.10 push-button)

Interrupt sources configured independently

When one of the sources raises an interrupt, the VIC …

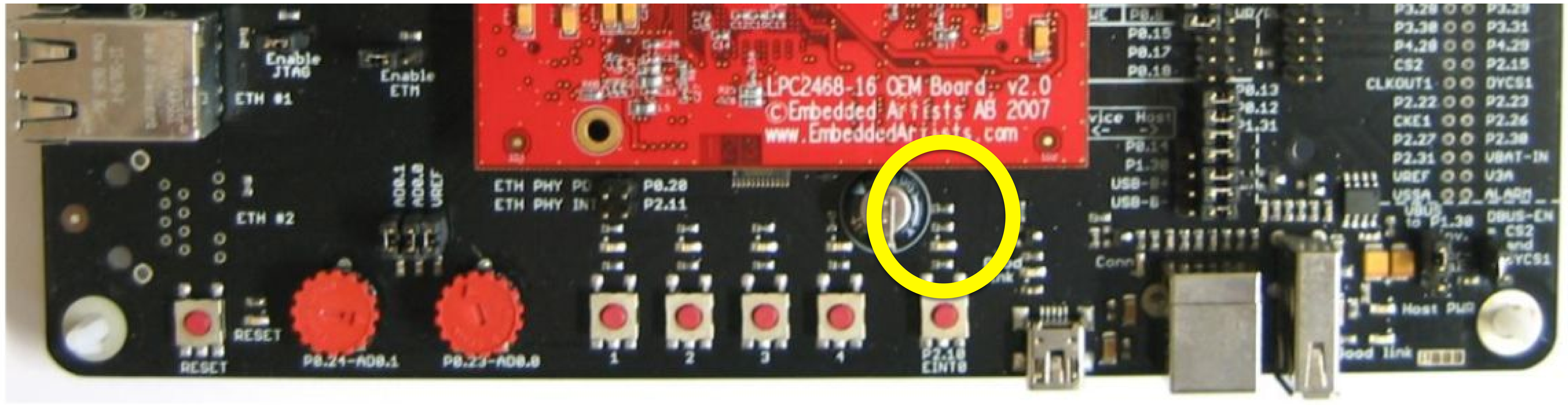
- loads address of a source-specific handler into VectAddr
- raises either the processor’s IRQ line or FIQ line

Processor executes source-specific handler loading Program Counter with address in VectAddr register

Interrupts Example

Design and write a program that will cause an LED to blink with a period of 1s

- Use LED connected to pin P2.10 of the LPC2468 (see IO example)
- User TIMER0 integrated into LPC2468 to generate an interrupt every 1 second.



TIMER0/1/2/3

Four integrated timer/counters

Multipurpose

- Count input signal transitions (examples?)
- Capture time on input signal transition (examples?)
- Measure time interval (counting input clock cycles) (examples?)

Optionally generate an interrupt when an event occurs (events?)

See LPC2468 User Manual Chapter 24

Approach

When program starts …

- Configure pin P2.10 for LED output (PINSEL4 – GPIO, FIO2DIR1 – output)
- Stop and Reset TIMER0
- Configure TIMER0 for periodic 1 second interrupts
- Configure VIC channel for TIMER0 (IRQs or FIQs, handler address, enable channel)
- Clear any previous TIMER0 interrupts (e.g. from last time program ran)
- Start TIMER0

When an interrupt occurs …

- Reset TIMER0 interrupt (so the device can raise further interrupts)
- Read and invert LED state (FIO2PIN1)
- Clear VIC interrupt source (so VIC can raise further interrupts, perhaps from other devices)

Configure TIMER0 for Periodic 1 Second Interrupts

Reset and disable TIMER0 (it might already be enabled)

T0TCR [0xE0004004] – Set bit 1 = 1 (reset) and bit 0 = 0 (stop)

Clear any previous TIMER0 interrupt

T0IR [0xE0004000] – Write 0xFF to clear interrupts

Want timer mode, generating interrupts every second

T0CTCR [0xE0004070] – Set bits 1:0 to 00 for timer mode

Want to generate an interrupt (when the count register equals the match register) every 1 seconds

T0MR0 [0xE0004018] = 12,000,000 (assuming TIMER is “fed” with a 12MHz clock)

Want to generate an interrupt, reset and restart the timer when the count register equals the match register

T0MCR [0xE0004014] – Set bits 1:0 to 11

No pre-scaling required (period is sufficiently small)

T0PR [0xE000400C] – Set to 0 for prescale factor of 1

See LPC2468 User Manual!!

Configure VIC for TIMER0 Interrupts

Identify the VIC vector number for TIMER0 … 4

Corresponds to a bit mask 0x10 (0001 0000) – we will use this later …

Set vector for IRQ (not FIQ)

VICIntSelect [0xFFFFF00C] – set bit 4 to 0 – BIC using our mask (0x10)

Set priority for vector 4 (e.g. to 15)

VICVectPriority0 (Base Address of VICVectPriority array) = 0xFFFFF200

VICVectPriority4 [0xFFFFF200 + (4 x 4)] – Set to 15

Set handler address

VICVectAddr0 (Base Address of VICVectAddr array) = 0xFFFFF100

VICVectAddr4 [0xFFFFF100 + (4 x 4)] – Set to address of our TimerHandler routine

Enable vector 4

VICIntEnable [0xFFFFF010] – Write 1 to bit 4 (using our mask (0x10))

Write the Interrupt Handler

Interrupt handler looks like a subroutine, with a few differences

- Note the ^ in the LDMFD instruction to restore the SPSR as the CPSR when we return
- Note the subtraction of 4 from the LINK register
- Note: we save/restore all used registers, including R0-R3
- No parameters passed to the subroutine

Interrupt handler must

- Clear TIMER0 interrupt – Set T0IR to 0xFF to clear TIMER0 interrupt
- Perform action (e.g. inverting LED state)
- Clear VIC (by writing 0 to VICVectAddr)
- Return