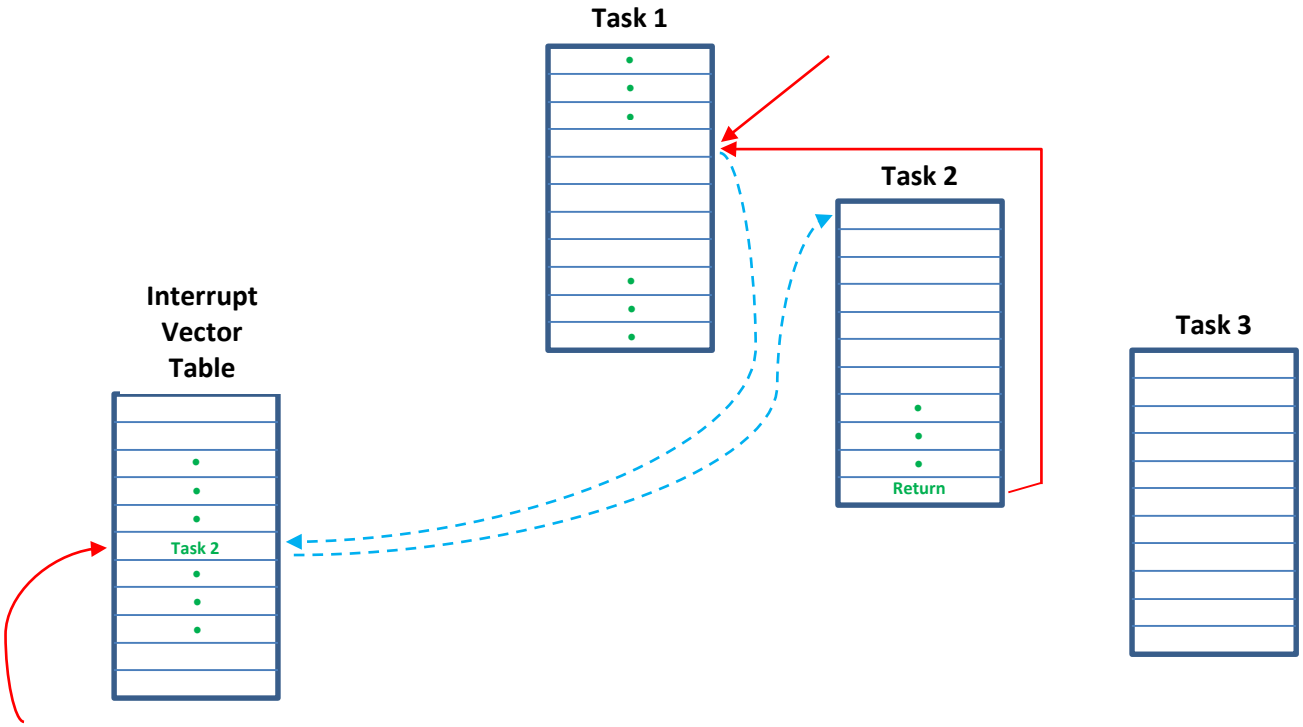
Chapter 4 – Interrupt Processing

What is an interrupt?

➔ An asynchronous event that re-directs the program flow.

- The source of the interrupt (i.e., the cause of the interrupt) can be external to the µC or internal[1] to the µC.
- An external source is often asynchronous to the µC clock and therefore needs to be synchronized to the µC clock.
- Both external and internal sources are, in a sense, "asynchronous" to the code flow.



[1] e.g. CPU timer timed out, A-D conversion done (ready for reading), etc

Why is it useful?

➔ The code does not need to poll (idle in a loop) to detect when an event occurs.

interrupts → facilitate pre-emption → facilitate multi-tasking[2]

fast interrupt response time → facilitates real-time processing

What happens when there is an interrupt?

1. Interrupts are disabled
2. Context is saved (to stack) (14 registers on c2000)
3. Pipeline is flushed
4. Program counter loaded with address stored in interrupt vector table at position that corresponds to the source of interrupt
5. Code branches to ISR (Interrupt Service Routine)
6. ISR code runs
7. Context is restored (from stack)
8. Interrupts are re-enabled
9. ISR returns to previous task

[2] More specifically, hardware-based interrupts. An operating system can implement software task switching.

Example – *interrupt* keyword

DSP2802x_DefaultISR.h

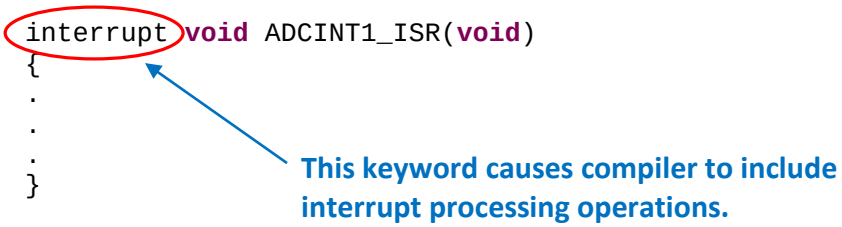
```
//function prototype:

interrupt void  ADCINT1_ISR(void);
```

ELEX7820-Lab3AD-AdcIsr.c

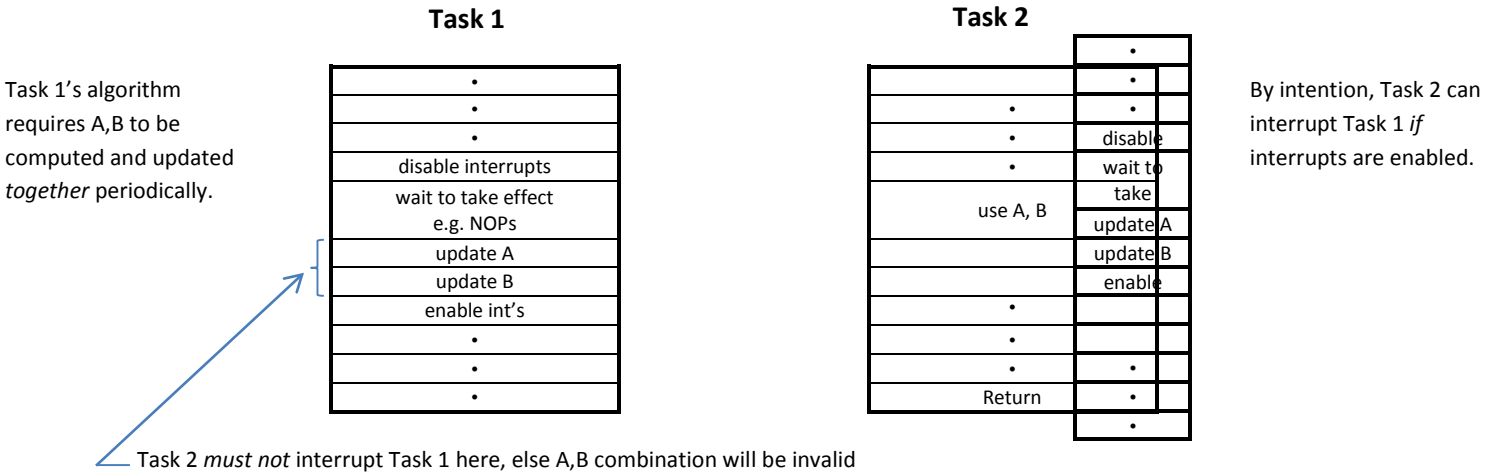
```
//function definition:

interrupt void ADCINT1_ISR(void)
{
.
.
.
}
```



Watch out for this:

➔ Temporary disabling of interrupts to protect a section of code…

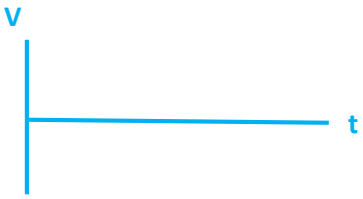


Case Study – High-Voltage Measurement Product Based on DSPs

➔ **Motorola datasheet said "user should wait 7 NOP's for interrupt disable to take effect"**

➔ **Systems installed in field in September, each system had 12 DSP chips**

➔ **We observed problem in field in May:**



- **DC offset "jump"**
- **occurred 1 in 10 billion times**

➔ **Spent 6 weeks debugging in lab using "accelerated testing"**

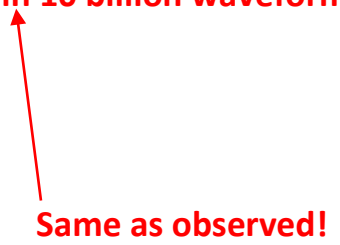
➔ **Motorola published erratum in July:**

- **must wait 12 NOP's, not 7**

➔ **Changed to 12 NOP's (upgraded code in field): problem went away**

➔ **Calculated chance of original problem:**

- **There were 12 – 7 = 5 NOP's of time where code was "exposed"**
- **Chance of problem: 1 in 10 billion waveforms**



Interrupt Logic on c2000 (Refer to "sprufn3d (or newer) - … Interrupts Reference Guide.pdf")