

# Embedded Systems Interfacing

## Interrupts

Embedded Systems Interfacing

1

## Overview

- Interrupt mechanism
- Nesting of interrupt
- Traps vs Hardware Interrupts
- Example 1: Timer 1 Interrupt
- Secondary Oscillator
- Example 2: Timer 1 with Secondary Oscillator

2

## Interrupt Mechanism

- An interrupt is an internal or external event that forces a hardware call to a function called an interrupt service routine.
  - Interrupt enable must be set [initialization]
  - Internal or external event forces interrupt flag to be set [hardware]
  - Event forces routine at interrupt vector to be called, see Table 5-1 [hardware]

3

## Interrupt Mechanism

- Processor state must be preserved [C-30 compiler]
- Interrupt service routine (ISR) must process data [user code]
- Interrupt flag must be cleared [user code]
- Processor state must be restored [C-30 compiler]

4

## C-30 Rules for ISR

- ISR cannot return anything
- No parameters may be passed to ISR
- ISR may not be directly called by other functions
- Ideally, ISR should not call other function

5

## ISR Templates

- Template 1

```
void __attribute__((interrupt,no_auto_psv)) _T1Interrupt(void){
    // interrupt service routine code here ...
}
```
- Template 2

```
void _ISR_T1Interrupt(void){
    // interrupt service routine code here ...
}
```

- Template 1 preferred because it reduces latency.
- Template 2 inserts save/restore of PSVPAG register.

6

# Embedded Systems Interfacing

## Interrupt Sources

- 5 x external pins with level trigger detection
- 22 x external pins connect to Change Notification module
- 5 x input capture modules
- 2 x serial port interfaces (UARTs)
- 4 x synchronous serial interfaces (SPI and I<sup>2</sup>C)

7

## Interrupt Sources

- 1 x Parallel Master Port
- 5 x 16-bit timers (2 x 32-bit timers)
- 1 x Analog-to-digital converter
- 1 x Analog comparator module
- 1 x Real-time clock and calendar
- 1 x Cyclic redundancy check generator

8

## Traps

- Generated by processor fault conditions
- Non-maskable interrupts
- Sources
  - Oscillator failure
  - Address error
  - Stack error
  - Math error
  - Reserved (four)

9

## Real Example with Timer1

```
// 32 MHz fosc or Tcyc = 62.5 ns
volatile int dSec = 0; //decisecond
volatile int Sec = 0; //second
volatile int min = 0; //minute

void _ISR__T1Interrupt(void)
    dSec++;
    if(dSec > 9) {
        dSec = 0;
        Sec++;
        if(Sec > 59) {
            Sec = 0;

```

10

## Real Example with Timer1

```
        Sec = 0;
        Min++;
        if(Min > 59){
            Min = 0;
        }
    }
    _T1IF = 0;
}
```

11

## Real Example with Timer1

```
int main(void) {
    _T1IP = 4;
    TMR1 = 0;
    PR1 = 25000-1;
    TRISA = 0xff00;
    T1CON = 0x8020; // 1:64 prescaler
    _T1IF=0;
    _T1IE=1;
    _IPL = 0;
    while(1) {
        PORTA = Sec;
    }
    return(0);
}
```

Initialization

Endless Loop

12

# Embedded Systems Interfacing

## Secondary Oscillator

- Timer1 has
  - External clock input or
  - Amplifier with feedback to be used with external crystal (resonator)
- Inexpensive 32,768 resonator are available
- Configure Timer1 for external crystal (resonator)
- Set PR1 to 32768-1
- Produces interrupt once per second

15

## Modified for Secondary Osc.

```
//Use similar code to Interrupt.C except
T1CON = 0x8002;
PR1 = (32 x 1024) -1;
```

16

## Real-Time Clock Calendar

- Available in 16-Bit and 32-Bit PICs
- Interrupt once every
  - Second
  - Minute
  - Hour
  - Day
  - Week
  - Month
- Year on specified day of year
- Four years for 29 February
- All counting done in hardware (not software)
- See next set of slides for details

17

## Managing Multiple Interrupt

- Most real applications have two or more interrupt sources
- Keep each ISR as short as possible
- Use priority levels to determine event that will be serviced first in case of coincidental interrupt events
- Nested interrupt require much more care ( \_NSTDIS = 1 to disable nesting)

18

## Additional Info

- C-30 has two interrupt vector tables
  - One for normal program execution
  - Second one for debugging
- The \_ISRFAST macro will allow register swapping for critical registers for faster processor state save and restore

19

## Additional Info

- No global interrupt disable
  - `__asm__ volatile("disi #0x0007");` will disable all interrupts for 7  $T_{CYC}$
  - `__asm__ volatile("disi #0x3FFF");` will disable all interrupts for very long time (16383  $T_{CYC}$ )
  - `DISICNT = 0;` will re-enable all interrupts

20

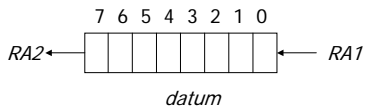
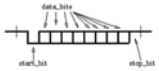
## Additional Info

- To use secondary oscillator you must set SOSCEN bit (see text for necessary code)
- Secondary oscillator cannot be simulated with MPSIM
- To set current time and date you must set RTCWREN bit (see text for necessary code)

21

## Homework #6

- Chapter 5 -- Handout
  - 1 Bit Bang Serial Port
  - 2 Bit Bang Slave Synchronous Peripheral Interface



23