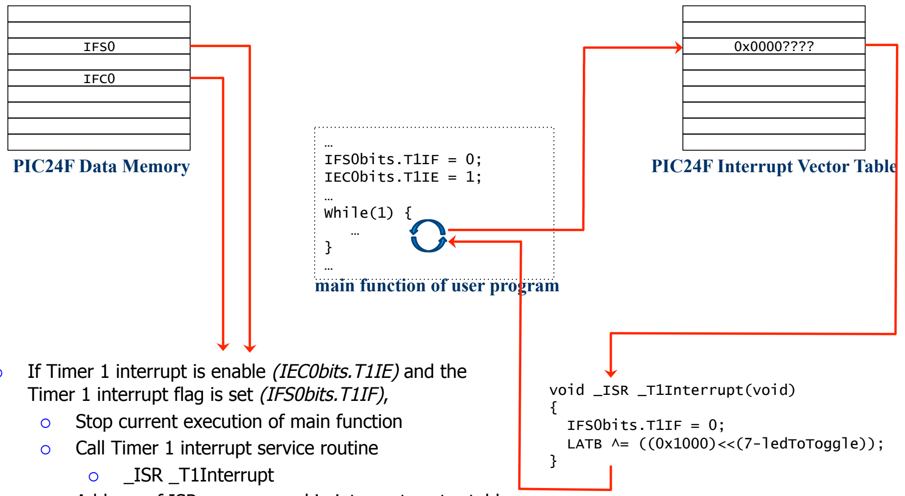
ECE 372 – Microcontroller Design

Interrupts

ECE 372 – Microcontroller Design

Interrupts



```
…
IFS0bits.T1IF = 0;
IEC0bits.T1IE = 1;
…
while(1) {
   …
}
…
```

```
void _ISR _T1Interrupt(void)
{
  IFS0bits.T1IF = 0;
  LATB ^= ((0x1000)<<(7-ledToToggle));
}
```

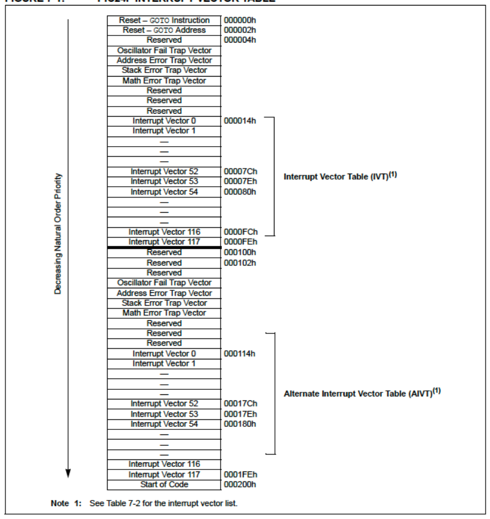
- If Timer 1 interrupt is enable *(IEC0bits.T1IE)* and the Timer 1 interrupt flag is set *(IFS0bits.T1IF)*,
  - Stop current execution of main function
  - Call Timer 1 interrupt service routine
    - _ISR _T1Interrupt
  - Address of ISR programmed in interrupt vector table

ECE 372 – Microcontroller Design

Interrupts

- PIC24F
  - 118 interrupts vectors
  - Unique vector for each possible interrupt
  - Compiler support for defining ISR for all possible interrupt using interrupt name

FIGURE 7-1: PIC24F INTERRUPT VECTOR TABLE



Decreasing Natural Order Priority

Reset – GOTO Instruction	000000h
Reset – GOTO Address	000002h
Reserved	000004h
Oscillator Fail Trap Vector	
Address Error Trap Vector	
Stack Error Trap Vector	
Math Error Trap Vector	
Reserved	
Reserved	
Reserved	
Interrupt Vector 0	000014h
Interrupt Vector 1	
—	
—	
—	
Interrupt Vector 52	00007Ch
Interrupt Vector 53	00007Eh
Interrupt Vector 54	000080h
—	
—	
—	
Interrupt Vector 116	0000FCh
Interrupt Vector 117	0000FEh
Reserved	000100h
Reserved	000102h
Reserved	
Oscillator Fail Trap Vector	
Address Error Trap Vector	
Stack Error Trap Vector	
Math Error Trap Vector	
Reserved	
Reserved	
Reserved	
Interrupt Vector 0	000114h
Interrupt Vector 1	
—	
—	
—	
Interrupt Vector 52	00017Ch
Interrupt Vector 53	00017Eh
Interrupt Vector 54	000180h
—	
—	
—	
Interrupt Vector 116	
Interrupt Vector 117	0001FEh
Start of Code	000200h

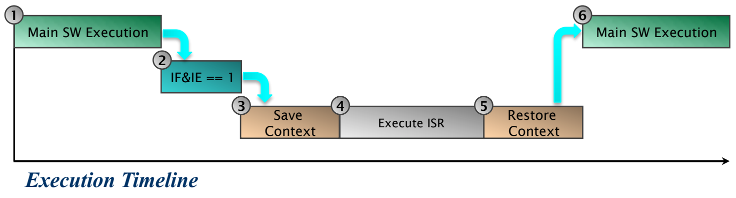
Interrupt Vector Table (IVT)(1)

Alternate Interrupt Vector Table (AIVT)(1)

Note 1: See Table 7-2 for the interrupt vector list.

ECE 372 – Microcontroller Design

Interrupts



- Interrupt Execution Timeline
  - Interrupts are checked every execution cycle
  - Interrupt service will save restore the execution context of the main execution loop at the begin and end of ISR execution
    - Saves all working registers and program counter
    - Automatically handled by compiler

ECE 372 – Microcontroller Design

Defining Interrupts

- Microchip Compiler for PIC24 Support
  - Interrupts can be defined primarily in two ways
  - Option 1 *(direct/verbose method)*

```
void __attribute__((interrupt)) _CNInterrupt(void)
{
   // interrupt code goes here
}
```

  - Option 2 *(using #defines in p24fj64ga002.h)*

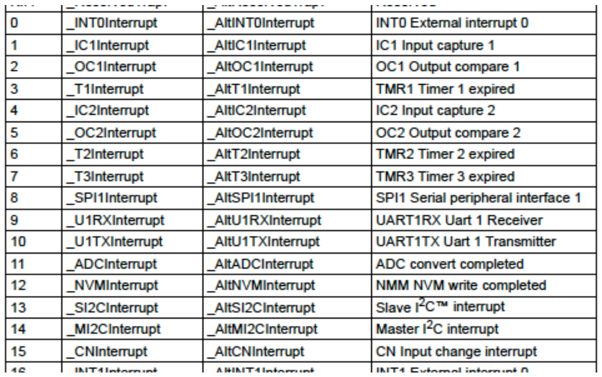
```
void _ISR _T1Interrupt(void)
{
   // interrupt code goes here
}
```

  - Compiler has defined names for

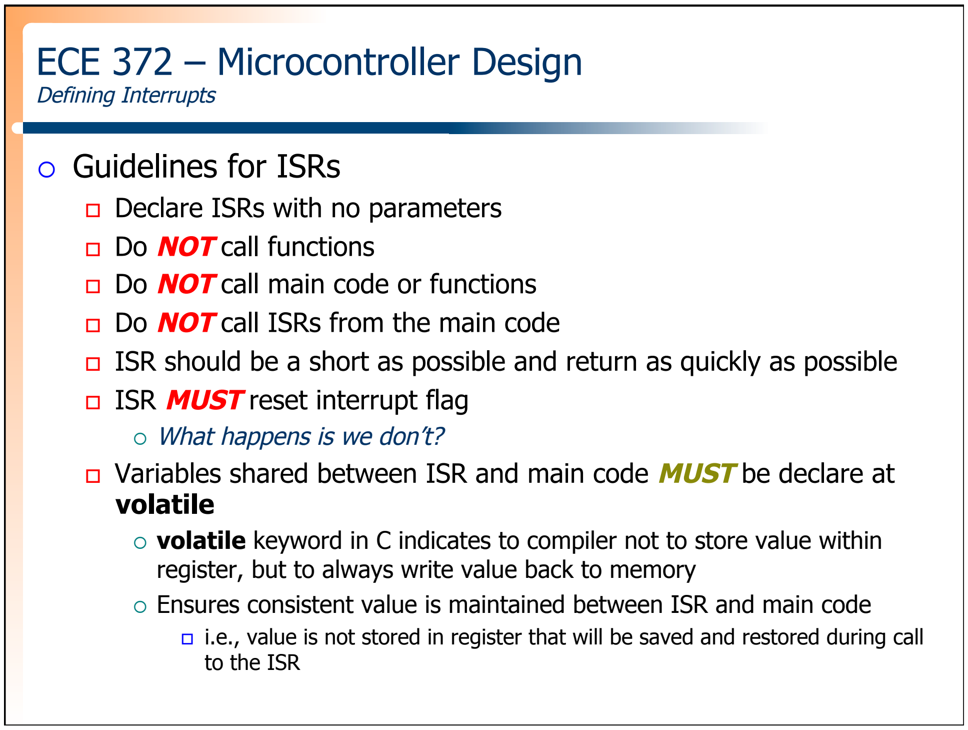
ECE 372 – Microcontroller Design

Defining Interrupts

- Microchip Compiler for PIC24 Support
  - Compiler has defined names for all interrupts
    - Example: _T1Interrupt : Timer 1
    - Example: _CNInterrupt : Change Notification Interrupt



0	_INT0Interrupt	_AltINT0Interrupt	INT0 External interrupt 0
1	_IC1Interrupt	_AltIC1Interrupt	IC1 Input capture 1
2	_OC1Interrupt	_AltOC1Interrupt	OC1 Output compare 1
3	_T1Interrupt	_AltT1Interrupt	TMR1 Timer 1 expired
4	_IC2Interrupt	_AltIC2Interrupt	IC2 Input capture 2
5	_OC2Interrupt	_AltOC2Interrupt	OC2 Output compare 2
6	_T2Interrupt	_AltT2Interrupt	TMR2 Timer 2 expired
7	_T3Interrupt	_AltT3Interrupt	TMR3 Timer 3 expired
8	_SPI1Interrupt	_AltSPI1Interrupt	SPI1 Serial peripheral interface 1
9	_U1RXInterrupt	_AltU1RXInterrupt	UART1RX Uart 1 Receiver
10	_U1TXInterrupt	_AltU1TXInterrupt	UART1TX Uart 1 Transmitter
11	_ADCInterrupt	_AltADCInterrupt	ADC convert completed
12	_NVMInterrupt	_AltNVMInterrupt	NMM NVM write completed
13	_SI2CInterrupt	_AltSI2CInterrupt	Slave I^{2}C™ interrupt
14	_MI2CInterrupt	_AltMI2CInterrupt	Master I^{2}C interrupt
15	_CNInterrupt	_AltCNInterrupt	CN Input change interrupt



ECE 372 – Microcontroller Design

Defining Interrupts

- Guidelines for ISRs
  - Declare ISRs with no parameters
  - Do ***NOT*** call functions
  - Do ***NOT*** call main code or functions
  - Do ***NOT*** call ISRs from the main code
  - ISR should be a short as possible and return as quickly as possible
  - ISR ***MUST*** reset interrupt flag
    - *What happens is we don't?*
  - Variables shared between ISR and main code ***MUST*** be declare at **volatile**
    - **volatile** keyword in C indicates to compiler not to store value within register, but to always write value back to memory
    - Ensures consistent value is maintained between ISR and main code
      - i.e., value is not stored in register that will be saved and restored during call to the ISR