Interrupts

If hardware requests an interrupt, while this instruction is being executed, then this instruction is completed first. Then execution starts from the ISR.

Interrupt Service Routine

"Function" called by the hardware at unpredictable times

mov.w $x3, RS

Return from interrupt; execution starts from here.

Interrupts are used to handle:

Urgent tasks
Slow human input
Waking the CPU from sleep

etc.

Main code or subroutine code

Label → your ISR:

Your code related to the interrupt
Interrupts can be requested by peripherals:

*e.g. ADC10 (Analog to Digital Converter)*

When an input analog voltage has been sampled and is ready for your program to read

**Timer A**

Every time the counter register reaches its maximum value

**GPIO**

Whenever the input signal on a port pin has a rising edge or a falling edge (provided that pin has been configured as an input pin with interrupt enabled)
Every interrupt type has a flag which is set when the conditions for that interrupt occur.

Interrupt flags for port 1 GPIO interrupts for example:

flags

P1IFG0 $\leftrightarrow$ P1.0
P1IFG1 $\leftrightarrow$ P1.1
P1IFG2 $\leftrightarrow$ P1.2
\vdots
P1IFG7 $\leftrightarrow$ P1.7

Each flag has a corresponding interrupt enable bit which must be set to enable that interrupt, e.g.

P1IE $\rightarrow$

For hardware to request an interrupt the corresponding interrupt bit must be set.
Two types of interrupts

Maskable interrupts:

- Status Register (SR)

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>V</td>
<td>SCG1</td>
<td>SCG0</td>
<td>OSC OFF</td>
<td>CPU OFF</td>
<td>GIE</td>
<td>N</td>
<td>Z</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

General Interrupt Enable

Maskable interrupts are ignored if the GIE bit is not set.

To enable maskable interrupts:

Set the interrupt enable bit for that interrupt

+ Enable GIE in the SR

\( \text{bis} \cdot W \times \text{GIE, SR} \)

Non-Maskable Interrupts

GIE bit has no effect on non-maskable interrupts

(Their interrupt bit must still be enabled, which is not set by default)
When an interrupt is requested then:

* Any currently executing instruction is completed
* The PC, which points to the next instruction is pushed on to the stack
* The status register (SR) is pushed on to the stack
* The interrupt with the highest priority is selected if multiple interrupts are waiting for service
* The SR is cleared, further maskable interrupts are disabled since GIE bit is cleared. All low power modes are terminated.
* Address of the corresponding ISR is loaded into the PC.
* ISR starts execution
Stack picture

Interrupt requested during this instruction

\[ \text{mov.w } \times 3, \text{RS} \]
\[ \text{mov.w } \text{R10, \&Result} \]

\[ \text{PC} \]
\[ \times \text{myISR} \]

\[ \text{SP} \]
\[ \text{0x03FC} \]

\[ \text{SR} \]
\[ \text{Cleaned} \]

\[ \text{myISR:} \]
\[ \text{mov.w } \text{R11, R10} \]
\[ \text{reti} \]

\[ \text{RAM} \]
\[ \text{0x0200} \]
\[ \text{0x0202} \]
\[ \text{0x03F8} \]
\[ \text{0x03FA} \]
\[ \text{0x03FC} \]
\[ \text{0x03FF} \]
\[ \text{0x0400} \]

\[ \text{0xDDDD} \]
\[ \text{0xCCCC} \]
\[ \text{SR} \]

Address of

\[ \text{Stack contains two words} \]

\[ \text{picture of the stack just before the ISR starts execution} \]
When reti is executed

* The SP pops from the stack. All previous settings of GIE and low power mode control bits are in effect again, i.e., maskable interrupts are enabled again.

* PC pops from the stack and execution resumes from the point where it was interrupted. If the CPU was asleep then it goes back to sleep.
MEMORY

{  
  \textbf{SFR} : \textit{origin} = 0x0000, \textit{length} = 0x0010  
  \textbf{PERIPHERALS\_8BIT} : \textit{origin} = 0x0010, \textit{length} = 0x00F0  
  \textbf{PERIPHERALS\_16BIT} : \textit{origin} = 0x0100, \textit{length} = 0x0100  
  \textbf{RAM} : \textit{origin} = 0x0200, \textit{length} = 0x0200  
}

\textbf{INFOD} : \textit{origin} = 0x1000, \textit{length} = 0x0040  
\textbf{INFOC} : \textit{origin} = 0x1040, \textit{length} = 0x0040  
\textbf{INFOB} : \textit{origin} = 0x1080, \textit{length} = 0x0040  
\textbf{INFOA} : \textit{origin} = 0x10C0, \textit{length} = 0x0040  

\textbf{FLASH} : \textit{origin} = 0xC000, \textit{length} = 0x3FE0  
\textbf{INT00} : \textit{origin} = 0xFFE0, \textit{length} = 0x0002  
\textbf{INT01} : \textit{origin} = 0xFFE2, \textit{length} = 0x0002  
\textbf{INT02} : \textit{origin} = 0xFFE4, \textit{length} = 0x0002  
\textbf{INT03} : \textit{origin} = 0xFFE6, \textit{length} = 0x0002  
\textbf{INT04} : \textit{origin} = 0xFFE8, \textit{length} = 0x0002  
\textbf{INT05} : \textit{origin} = 0xFFEA, \textit{length} = 0x0002  
\textbf{INT06} : \textit{origin} = 0xFFFFC, \textit{length} = 0x0002  
\textbf{INT07} : \textit{origin} = 0xFFFFE, \textit{length} = 0x0002  
\textbf{INT08} : \textit{origin} = 0xFFFF0, \textit{length} = 0x0002  
\textbf{INT09} : \textit{origin} = 0xFFFF2, \textit{length} = 0x0002  
\textbf{INT10} : \textit{origin} = 0xFFFF4, \textit{length} = 0x0002  
\textbf{INT11} : \textit{origin} = 0xFFFF6, \textit{length} = 0x0002  
\textbf{INT12} : \textit{origin} = 0xFFFF8, \textit{length} = 0x0002  
\textbf{INT13} : \textit{origin} = 0xFFFFA, \textit{length} = 0x0002  
\textbf{INT14} : \textit{origin} = 0xFFFFC, \textit{length} = 0x0002  
\textbf{RESET} : \textit{origin} = 0xFFFFE, \textit{length} = 0x0002  

Total Address Space 64KB

8Bit Peripherals 240B

16Bit Peripherals 256B

RAM 512B

ROM (FLASH information memory) 256B

ROM (FLASH Code memory) 16352B

ROM (FLASH main memory) 16KB

ROM Interrupt Vectors 32B

ROM (FLASH) 16KB+256B

Not Mapped
Interrupt Vectors

int. type address int. vector table

Peripheral

int0  →  FFFE0
int1  →  FFFE2
int2  →  FFFE4
int3  →  FFE6
int4  →  FFE8
int5  →  FFEA

non-maskable

higher priority

Bottom of memory addressable space

Note:

For single sourced interrupts the interrupt flag is cleared automatically after the interrupt is serviced by the ISR.

For multiple sourced interrupts your ISR must clear the interrupt flag.
### Interrupt Vector Table

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Interrupt Flag</th>
<th>Section Name</th>
<th>Priority</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power-up</td>
<td>PORIFG</td>
<td>RESET</td>
<td>31</td>
<td>(highest)</td>
</tr>
<tr>
<td></td>
<td>RSTIFG</td>
<td></td>
<td></td>
<td>Reset</td>
</tr>
<tr>
<td></td>
<td>WDTIFG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>KEYV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External Reset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Watchdog Timer+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash key violation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC out of-range</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMI</td>
<td>NMIFG</td>
<td>INT14</td>
<td>30</td>
<td>non-maskable</td>
</tr>
<tr>
<td>Oscillator fault</td>
<td>OFIFG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash memory access violation</td>
<td>ACCVIFG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timer1_A3</td>
<td>CCIFG (TACCR0)</td>
<td>INT13</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>Timer1_A3</td>
<td>CCIFG (TACCR2, TACCR1)</td>
<td>INT12</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>Comparator_A+</td>
<td>CAIFG</td>
<td>INT11</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Watchdog Timer+</td>
<td>WDTIFG</td>
<td>INT10</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>Timer0_A3</td>
<td>CCIFG (TACCR0)</td>
<td>INT09</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Timer0_AE</td>
<td>CCIFG (TACCR2, TACCR1) TAIFG</td>
<td>INT08</td>
<td>24</td>
<td>maskable</td>
</tr>
<tr>
<td>USCI_A0/USCI_B0 receive</td>
<td>UCA0RFIG, UCB0RXIFG</td>
<td>INT07</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>USCI_B0 I2C status</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USCI_A0/USCI_B0 receive</td>
<td>UCA0TXIFG, UCB0TXIFG</td>
<td>INT06</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>USCI_B0 I2C receive/transmit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC10</td>
<td>ADC10IFG</td>
<td>INT05</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>I/O Port P2</td>
<td>P2IFG.0 to P2IFG.7</td>
<td>INT03</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>I/O Port P1</td>
<td>P1IFG.0 to P1IFG.7</td>
<td>INT02</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>
main program and subroutines

PORT1_ISR:

.clr.b &P1IFG

; multi sourced interrupt, therefor you must
; clear the interrupt flag

.reti

; return from interrupt

;----------------------------------------------------------------------------------------------
; Interrupt Vectors
;----------------------------------------------------------------------------------------------

.sect ".int02"

.short PORT1_ISR

.sect ".reset"

.short RESET