Section 29. Interrupts (Part II)

HIGHLIGHTS

This section of the manual contains the following major topics:

29.1 Introduction ............................................................................................................... 29-2
29.2 Non-Maskable Traps ............................................................................................... 29-8
29.3 Interrupt Processing Timing ................................................................................. 29-13
29.4 Interrupt Control and Status Registers ............................................................... 29-16
29.5 Interrupt Setup Procedures .................................................................................... 29-43
29.6 Register Map ............................................................................................................ 29-46
29.7 Design Tips .............................................................................................................. 29-47
29.8 Related Application Notes ..................................................................................... 29-48
29.9 Revision History ...................................................................................................... 29-49
29.1  INTRODUCTION

This section contains device-specific information for the following devices:

- dsPIC33FJ12GP201
- dsPIC33FJ12GP202
- dsPIC33FJ12MC201
- dsPIC33FJ12MC202

The dsPIC33F Interrupt Controller module reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the dsPIC33F CPU. The features of this module include the following:

- Up to eight processor exceptions and software traps
- Seven user-selectable priority levels
- Interrupt Vector Table (IVT) with up to 126 vectors
- Unique vector for each interrupt or exception source
- Fixed priority within a specified user priority level
- Alternate Interrupt Vector Table (AIVT) for debugging support
- Fixed interrupt entry and return latencies

29.1.1  Interrupt Vector Table

The Interrupt Vector Table (IVT) shown in Figure 29-1, resides in program memory starting at location 0x000004. The IVT contains 126 vectors that consist of eight non-maskable trap vectors and up to 118 sources of interrupt. In general, each interrupt source has its own vector. Each interrupt vector contains a 24-bit-wide address. The value programmed into each interrupt vector location is the starting address of the associated Interrupt Service Routine (ISR).

29.1.2  Alternate Interrupt Vector Table

The Alternate Interrupt Vector Table (AIVT) is located after the IVT, as shown in Figure 29-1. Access to the AIVT is provided by the Enable Alternate Interrupt Vector Table (ALTIVT) control bit in Interrupt Control Register 2 (INTCON2<15>). If the ALTIVT bit is set, all interrupt and exception processes use the alternate vectors instead of the default vectors. The alternate vectors are organized in the same manner as the default vectors.

The AIVT supports emulation and debugging by providing a means to switch between an application and a support environment without reprogramming the interrupt vectors. This feature also enables switching between applications for evaluation of different software algorithms at run time. If the AIVT is not needed, the AIVT must be programmed with the same addresses used in IVT.

29.1.3  Reset Sequence

A device Reset is not a true exception because the interrupt controller is not involved in the Reset process. The dsPIC33F device clears its registers during a Reset, that forces the Program Counter (PC) to zero. The processor then begins program execution at location 0x000000. The user application programs a GOTO instruction at the Reset address, that redirects program execution to the appropriate start-up routine.

Note: Any unimplemented or unused vector locations in the IVT and AIVT must be programmed with the address of a default interrupt handler routine that contains a RESET instruction.
## Section 29. Interrupts (Part II)

### Figure 29-1: Interrupt Vector Table

<table>
<thead>
<tr>
<th>IVT</th>
<th>AVT</th>
<th>Decreasing Natural Order Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>IVT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AVT</td>
</tr>
<tr>
<td>Reset – GOTO Instruction</td>
<td>0x000000</td>
<td></td>
</tr>
<tr>
<td>Reset – GOTO Address</td>
<td>0x000002</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>0x000004</td>
<td></td>
</tr>
<tr>
<td>Oscillator Fail Trap Vector</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Address Error Trap Vector</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Stack Error Trap Vector</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Math Error Trap Vector</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 0</td>
<td>0x000014</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 1</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 52</td>
<td>0x00007C</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 53</td>
<td>0x00007E</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 54</td>
<td>0x000080</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 116</td>
<td>0x0000FC</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 117</td>
<td>0x0000FE</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>0x000100</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>0x000102</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 0</td>
<td>0x000114</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 1</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 52</td>
<td>0x00017C</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 53</td>
<td>0x00017E</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 54</td>
<td>0x000180</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 116</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Interrupt Vector 117</td>
<td>0x0001FE</td>
<td></td>
</tr>
<tr>
<td>Start of Code</td>
<td>0x000200</td>
<td></td>
</tr>
</tbody>
</table>

See Table 29-1 for Interrupt Vector details.
<table>
<thead>
<tr>
<th>Vector Number</th>
<th>IVT Address</th>
<th>AIVT Address</th>
<th>Interrupt Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000004</td>
<td>0x000104</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>0x000006</td>
<td>0x000106</td>
<td>Oscillator Failure</td>
</tr>
<tr>
<td>2</td>
<td>0x000008</td>
<td>0x000108</td>
<td>Address Error</td>
</tr>
<tr>
<td>3</td>
<td>0x00000A</td>
<td>0x00010A</td>
<td>Stack Error</td>
</tr>
<tr>
<td>4</td>
<td>0x00000C</td>
<td>0x00010C</td>
<td>Math Error</td>
</tr>
<tr>
<td>5</td>
<td>0x00000E</td>
<td>0x00010E</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>0x000010</td>
<td>0x000110</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>0x000012</td>
<td>0x000112</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>0x000014</td>
<td>0x000114</td>
<td>INT0 – External Interrupt 0</td>
</tr>
<tr>
<td>9</td>
<td>0x000016</td>
<td>0x000116</td>
<td>IC1 – Input Compare 1</td>
</tr>
<tr>
<td>10</td>
<td>0x000018</td>
<td>0x000118</td>
<td>OC1 – Output Compare 1</td>
</tr>
<tr>
<td>11</td>
<td>0x00001A</td>
<td>0x00011A</td>
<td>T1 – Timer1</td>
</tr>
<tr>
<td>12</td>
<td>0x00001C</td>
<td>0x00011C</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>0x00001E</td>
<td>0x00011E</td>
<td>IC2 – Input Capture 2</td>
</tr>
<tr>
<td>14</td>
<td>0x000020</td>
<td>0x000120</td>
<td>OC2 – Output Compare 2</td>
</tr>
<tr>
<td>15</td>
<td>0x000022</td>
<td>0x000122</td>
<td>T2 – Timer2</td>
</tr>
<tr>
<td>16</td>
<td>0x000024</td>
<td>0x000124</td>
<td>T3 – Timer3</td>
</tr>
<tr>
<td>17</td>
<td>0x000026</td>
<td>0x000126</td>
<td>SPI1E – SPI1 Error</td>
</tr>
<tr>
<td>18</td>
<td>0x000028</td>
<td>0x000128</td>
<td>SPI1 – SPI1 Transfer Done</td>
</tr>
<tr>
<td>19</td>
<td>0x00002A</td>
<td>0x00012A</td>
<td>U1RX – UART1 Receiver</td>
</tr>
<tr>
<td>20</td>
<td>0x00002C</td>
<td>0x00012C</td>
<td>U1TX – UART1 Transmitter</td>
</tr>
<tr>
<td>21</td>
<td>0x00002E</td>
<td>0x00012E</td>
<td>AD1 – ADC1 Convert Done</td>
</tr>
<tr>
<td>22</td>
<td>0x000030</td>
<td>0x000130</td>
<td>Reserved</td>
</tr>
<tr>
<td>23</td>
<td>0x000032</td>
<td>0x000132</td>
<td>Reserved</td>
</tr>
<tr>
<td>24</td>
<td>0x000034</td>
<td>0x000134</td>
<td>SI2C1 – I2C1 Slave Events</td>
</tr>
<tr>
<td>25</td>
<td>0x000036</td>
<td>0x000136</td>
<td>MI2C1 – I2C1 Master Events</td>
</tr>
<tr>
<td>26</td>
<td>0x000038</td>
<td>0x000138</td>
<td>Reserved</td>
</tr>
<tr>
<td>27</td>
<td>0x00003A</td>
<td>0x00013A</td>
<td>CN – Change Notification Interrupt</td>
</tr>
<tr>
<td>28</td>
<td>0x00003C</td>
<td>0x00013C</td>
<td>INT1 – External Interrupt 1</td>
</tr>
<tr>
<td>29</td>
<td>0x00003E</td>
<td>0x00013E</td>
<td>Reserved</td>
</tr>
<tr>
<td>30</td>
<td>0x000040</td>
<td>0x000140</td>
<td>IC7 – Input Capture 7</td>
</tr>
<tr>
<td>31</td>
<td>0x000042</td>
<td>0x000142</td>
<td>IC8 – Input Capture 8</td>
</tr>
<tr>
<td>32</td>
<td>0x000044</td>
<td>0x000144</td>
<td>Reserved</td>
</tr>
<tr>
<td>33</td>
<td>0x000046</td>
<td>0x000146</td>
<td>Reserved</td>
</tr>
<tr>
<td>34</td>
<td>0x000048</td>
<td>0x000148</td>
<td>Reserved</td>
</tr>
<tr>
<td>35</td>
<td>0x00004A</td>
<td>0x00014A</td>
<td>Reserved</td>
</tr>
<tr>
<td>36</td>
<td>0x00004C</td>
<td>0x00014C</td>
<td>Reserved</td>
</tr>
<tr>
<td>37</td>
<td>0x00004E</td>
<td>0x00014E</td>
<td>INT2 – External Interrupt 2</td>
</tr>
<tr>
<td>38</td>
<td>0x000050</td>
<td>0x000150</td>
<td>Reserved</td>
</tr>
<tr>
<td>39</td>
<td>0x000052</td>
<td>0x000152</td>
<td>Reserved</td>
</tr>
<tr>
<td>40</td>
<td>0x000054</td>
<td>0x000154</td>
<td>Reserved</td>
</tr>
<tr>
<td>41</td>
<td>0x000056</td>
<td>0x000156</td>
<td>Reserved</td>
</tr>
<tr>
<td>42</td>
<td>0x000058</td>
<td>0x000158</td>
<td>Reserved</td>
</tr>
<tr>
<td>43</td>
<td>0x00005A</td>
<td>0x00015A</td>
<td>Reserved</td>
</tr>
<tr>
<td>44</td>
<td>0x00005C</td>
<td>0x00015C</td>
<td>Reserved</td>
</tr>
<tr>
<td>45</td>
<td>0x00005E</td>
<td>0x00015E</td>
<td>Reserved</td>
</tr>
<tr>
<td>46</td>
<td>0x000060</td>
<td>0x000160</td>
<td>Reserved</td>
</tr>
<tr>
<td>47</td>
<td>0x000062</td>
<td>0x000162</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
## Section 29. Interrupts (Part II)

### Table 29-1: Interrupt Vectors (Continued)

<table>
<thead>
<tr>
<th>Vector Number</th>
<th>IVT Address</th>
<th>AIVT Address</th>
<th>Interrupt Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>0x000064</td>
<td>0x000164</td>
<td>Reserved</td>
</tr>
<tr>
<td>49</td>
<td>0x000066</td>
<td>0x000166</td>
<td>Reserved</td>
</tr>
<tr>
<td>50</td>
<td>0x000068</td>
<td>0x000168</td>
<td>Reserved</td>
</tr>
<tr>
<td>51</td>
<td>0x00006A</td>
<td>0x00016A</td>
<td>Reserved</td>
</tr>
<tr>
<td>52</td>
<td>0x00006C</td>
<td>0x00016C</td>
<td>Reserved</td>
</tr>
<tr>
<td>53</td>
<td>0x00006E</td>
<td>0x00016E</td>
<td>Reserved</td>
</tr>
<tr>
<td>54</td>
<td>0x000070</td>
<td>0x000170</td>
<td>Reserved</td>
</tr>
<tr>
<td>55</td>
<td>0x000072</td>
<td>0x000172</td>
<td>Reserved</td>
</tr>
<tr>
<td>56</td>
<td>0x000074</td>
<td>0x000174</td>
<td>Reserved</td>
</tr>
<tr>
<td>57</td>
<td>0x000076</td>
<td>0x000176</td>
<td>Reserved</td>
</tr>
<tr>
<td>58</td>
<td>0x000078</td>
<td>0x000178</td>
<td>Reserved</td>
</tr>
<tr>
<td>59</td>
<td>0x00007A</td>
<td>0x00017A</td>
<td>Reserved</td>
</tr>
<tr>
<td>60</td>
<td>0x00007C</td>
<td>0x00017C</td>
<td>Reserved</td>
</tr>
<tr>
<td>61</td>
<td>0x00007E</td>
<td>0x00017E</td>
<td>Reserved</td>
</tr>
<tr>
<td>62</td>
<td>0x000080</td>
<td>0x000180</td>
<td>Reserved</td>
</tr>
<tr>
<td>63</td>
<td>0x000082</td>
<td>0x000182</td>
<td>Reserved</td>
</tr>
<tr>
<td>64</td>
<td>0x000084</td>
<td>0x000184</td>
<td>Reserved</td>
</tr>
<tr>
<td>65</td>
<td>0x000086</td>
<td>0x000186</td>
<td>PWM1 – PWM1 Period Match</td>
</tr>
<tr>
<td>66</td>
<td>0x000088</td>
<td>0x000188</td>
<td>QEI – Position Counter Compare</td>
</tr>
<tr>
<td>67</td>
<td>0x00008A</td>
<td>0x00018A</td>
<td>Reserved</td>
</tr>
<tr>
<td>68</td>
<td>0x00008C</td>
<td>0x00018C</td>
<td>Reserved</td>
</tr>
<tr>
<td>69</td>
<td>0x00008E</td>
<td>0x00018E</td>
<td>Reserved</td>
</tr>
<tr>
<td>70</td>
<td>0x000090</td>
<td>0x000190</td>
<td>Reserved</td>
</tr>
<tr>
<td>71</td>
<td>0x000092</td>
<td>0x000192</td>
<td>FLTA1 – PWM1 Fault A</td>
</tr>
<tr>
<td>72</td>
<td>0x000094</td>
<td>0x000194</td>
<td>Reserved</td>
</tr>
<tr>
<td>73</td>
<td>0x000096</td>
<td>0x000196</td>
<td>U1E – UART1 Error</td>
</tr>
<tr>
<td>74</td>
<td>0x000098</td>
<td>0x000198</td>
<td>Reserved</td>
</tr>
<tr>
<td>75</td>
<td>0x00009A</td>
<td>0x00019A</td>
<td>Reserved</td>
</tr>
<tr>
<td>76</td>
<td>0x00009C</td>
<td>0x00019C</td>
<td>Reserved</td>
</tr>
<tr>
<td>77</td>
<td>0x00009E</td>
<td>0x00019E</td>
<td>Reserved</td>
</tr>
<tr>
<td>78</td>
<td>0x0000A0</td>
<td>0x0001A0</td>
<td>Reserved</td>
</tr>
<tr>
<td>79</td>
<td>0x0000A2</td>
<td>0x0001A2</td>
<td>Reserved</td>
</tr>
<tr>
<td>80</td>
<td>0x0000A4</td>
<td>0x0001A4</td>
<td>Reserved</td>
</tr>
<tr>
<td>81</td>
<td>0x0000A6</td>
<td>0x0001A6</td>
<td>PWM2 – PWM2 Period Match</td>
</tr>
<tr>
<td>82</td>
<td>0x0000A8</td>
<td>0x0001A8</td>
<td>FLTA2 – PWM2 Fault A</td>
</tr>
<tr>
<td>83-125</td>
<td>0x0000AA-0x0000FE</td>
<td>0x0001AA-0x0001FE</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
29.1.4 CPU Priority Status

The CPU can operate at one of 16 priority levels that range from 0-15. An interrupt or trap source must have a priority level greater than the current CPU priority to initiate an exception process. The peripheral and external interrupt sources for levels 0-7 can be programmed. CPU priority levels 8-15 are reserved for trap sources.

A trap is a non-maskable interrupt source intended to detect hardware and software problems (see 29.2 “Non-Maskable Traps”). The priority level for each trap source is fixed. Only one trap is assigned to a priority level. An interrupt source programmed to priority level 0 is effectively disabled, since it can never be greater than the CPU priority.

The current CPU priority level is indicated by the following status bits:

- CPU Interrupt Priority Level (IPL<2:0>) status bits in the CPU Status Register (SR<7:5>)
- CPU Interrupt Priority Level 3 (IPL3) status bit in the Core Control (CORCON<3>) register

Because the IPL<2:0> status bits are readable and writable, the user application can modify these bits to disable all sources of interrupts below a given priority level. For example, if IPL<2:0> = 3, the CPU is not interrupted by any source with a programmed priority level of 0, 1, 2, or 3.

Trap events have higher priority than any user interrupt source. When the IPL3 bit is set, a trap event is in progress. The IPL3 bit can be cleared, but not set, by the user application. In some applications, you may need to clear the IPL3 bit when a trap has occurred and branch to an instruction other than the instruction after the one that originally caused the trap to occur.

All user interrupt sources can be disabled by setting IPL<2:0> = 111.

Note: The IPL<2:0> bits become read-only bits when interrupt nesting is disabled. See 29.2.4.2 “Interrupt Nesting” for more information.

29.1.5 Interrupt Priority

Each peripheral interrupt source can be assigned to one of seven priority levels. The user application-assignable interrupt priority control bits for each individual interrupt are located in the Least Significant 3 bits of each nibble within the IPCx registers. Bit 3 of each nibble is not used and is read as a ‘0’. These bits define the priority level assigned to a particular interrupt. The usable priority levels are 1 (lowest priority) through 7 (highest priority). If the IPC bits associated with an interrupt source are all cleared, the interrupt source is effectively disabled.

Note: The application program must disable the interrupts while reconfiguring the interrupt priority levels on the fly. Failure to disable interrupts can produce unexpected results.

More than one interrupt request source can be assigned to a specific priority level. To resolve priority conflicts within a given user application-assigned level, each source of interrupt has a natural order priority based on its location in the IVT. Table 29-1 shows the location of each interrupt source in the IVT. The lower numbered interrupt vectors have higher natural priority, while the higher numbered vectors have lower natural priority. The overall priority level for any pending source of interrupt is first determined by the user application-assigned priority of that source in the IPCx register, and then by the natural order priority within the IVT.

Natural order priority is used only to resolve conflicts between simultaneous pending interrupts with the same user application-assigned priority level. Once the priority conflict is resolved and the exception process begins, the CPU can be interrupted only by a source with a higher user application-assigned priority. Interrupts with the same user application-assigned priority but a higher natural order priority that become pending during the exception process remain pending until the current exception process completes.
Assigning each interrupt source to one of seven priority levels enables the user application to give an interrupt with a low natural order priority, a very high overall priority level. For example, Timer2 can be given a priority of 7 and the External Interrupt 0 (INT0) can be assigned to priority level 1, giving it a very low effective priority.

**Note:** The peripherals and sources of interrupt available in the IVT vary depending on the specific dsPIC33F device. The sources of interrupt shown in this document represent a comprehensive listing of all interrupt sources found on dsPIC33F devices. Refer to the specific device data sheet for further details.
29.2 NON-MASKABLE TRAPS

Traps are non-maskable, nestable interrupts that adhere to a fixed priority structure. Traps provide a means to correct erroneous operation during debugging and operation of the application. If the user application does not intend to correct a trap error condition, these vectors must be loaded with the address of a software routine to reset the device. Otherwise, the user application programs the trap vector with the address of a service routine that corrects the trap condition.

The dsPIC33F consists of four implemented sources of non-maskable traps:

- Oscillator Failure Trap
- Stack Error Trap
- Address Error Trap
- Math Error Trap

For many of the trap conditions, the instruction that caused the trap is allowed to complete before exception processing begins. Therefore, the user application may have to correct the action of the instruction that caused the trap.

Each trap source has a fixed priority as defined by its position in the IVT. An oscillator failure trap has the highest priority, while a math error trap has the lowest priority (see Figure 29-1). In addition, trap sources are classified into two distinct categories: “hard” traps and “soft” traps.

29.2.1 Soft Traps

The math error trap (priority level 11) and stack error trap (priority level 12) are categorized as soft trap sources. Soft traps can be treated like non-maskable sources of interrupt that adhere to the priority assigned by their position in the IVT. Soft traps are processed like interrupts and require two cycles to be sampled and acknowledged prior to exception processing. Therefore, additional instructions may be executed before a soft trap is acknowledged.

29.2.1.1 STACK ERROR TRAP (SOFT TRAP, LEVEL 12)

The stack is initialized to 0x0800 during a Reset. A stack error trap is generated, if the stack pointer address is less than 0x0800.

A Stack Limit (SPLIM) register associated with the stack pointer is uninitialized at Reset. The stack overflow check is not enabled until a word is written to the SPLIM register.

All Effective Addresses (EA) generated using W15 as a source or destination pointer are compared against the value in the SPLIM register. If the EA is greater than the contents of the SPLIM register, a stack error trap is generated. In addition, a stack error trap is generated if the EA calculation wraps over the end of data space (0xFFFF).

A stack error can be detected in software by polling the Stack Error Trap (STKERR) status bit (INTCON1<2>). To avoid re-entering the trap service routine, the STKERR status flag must be cleared in software with a RETFIE (Return from Interrupt) instruction before the program returns from the trap.

29.2.1.2 MATH ERROR TRAP (SOFT TRAP, LEVEL 11)

Any of the following events will generate a math error trap:

- Accumulator A overflow
- Accumulator B overflow
- Catastrophic accumulator overflow
- Divide by zero
- Shift Accumulator (SFTAC) operation that exceeds ±16 bits

The following three bits in the INTCON1 register enable the three types of accumulator overflow traps:

- The Accumulator A Overflow Trap Flag (OVATE) control bit (INTCON1<10>) enables traps for an Accumulator A overflow event.
- The Accumulator B Overflow Trap Flag (OVBTE) control bit (INTCON1<9>) enables traps for an Accumulator B overflow event.
• The Catastrophic Overflow Trap Enable (COVTE) control bit (INTCON1<8>) enables traps for a catastrophic overflow of either accumulator. When this trap is detected, these corresponding ERROR bits are set in the INTCON1 register:
  - Accumulator A Overflow Trap Flag (OVAERR)
  - Accumulator B Overflow Trap Flag (OVBERR)
  - Accumulator A Catastrophic Overflow Trap Enable (COVAERR)
  - Accumulator B Catastrophic Overflow Trap Enable (COVBERR)

An Accumulator A or Accumulator B overflow event is defined as a carry-out from bit 31. The accumulator overflow cannot occur, if the 31-bit Saturation mode is enabled for the accumulator. A catastrophic accumulator overflow is defined as a carry-out from bit 39 of either accumulator. The catastrophic overflow cannot occur, if accumulator saturation (31-bit or 39-bit) is enabled.

Divide-by-zero traps cannot be disabled. The divide-by-zero check is performed during the first iteration of the REPEAT loop that executes the divide instruction. The Math Error Status (DIV0ERR) bit (INTCON1<6>) is set when this trap is detected.

Accumulator shift traps cannot be disabled. The SFTAC instruction can be used to shift the accumulator by a literal value or a value in one of the W registers. If the shift value exceeds ±16 bits, an arithmetic trap is generated and the Shift Accumulator Error Status (SFTAERR) bit (INTCON1<7>) is set. The SFTAC instruction executes, but the results of the shift are not written to the target accumulator.

A math error trap can be detected in software by polling the Math Error Status (MATHERR) bit (INTCON1<4>). To avoid re-entering the trap service routine, the MATHERR status flag must be cleared in software with a RETFIE instruction before the program returns from the trap. Before the MATHERR status bit can be cleared, all conditions that caused the trap to occur must also be cleared. If the trap was due to an accumulator overflow, the Accumulator Overflow (OA and OB) status bits (SR<15:14>) must be cleared. The OA and OB status bits are read-only, so the user application must perform a dummy operation on the overflowed accumulator (such as adding '0'), which will cause the hardware to clear the OA or OB status bit.

29.2.2 Hard Traps

Hard traps include exceptions of priority level 13 through level 15, inclusive. The address error (level 13) and oscillator error (level 14) traps fall into this category.

Like soft traps, hard traps are non-maskable sources of interrupt. The difference between hard traps and soft traps is that hard traps force the CPU to stop code execution after the instruction causing the trap has completed. Normal program execution flow does not resume until the trap has been acknowledged and processed.

29.2.2.1 TRAP PRIORITY AND HARD TRAP CONFLICTS

If a higher-priority trap occurs while any lower-priority trap is in progress, processing of the lower-priority trap is suspended. The higher-priority trap is acknowledged and processed. The lower-priority trap remains pending until processing of the higher-priority trap completes.

Each hard trap that occurs must be acknowledged before code execution of any type can continue. If a lower-priority hard trap occurs while a higher-priority trap is pending, acknowledged or being processed, a hard-trap conflict occurs because the lower-priority trap cannot be acknowledged until processing for the higher-priority trap completes.

The device is automatically reset in a hard-trap conflict condition. The Trap Reset Flag (TRAPR) status bit in the RESET Control Register (RCON<15> in the Reset module) is set when the Reset occurs so that the condition can be detected in software.

29.2.2.2 OSCILLATOR FAILURE TRAP (HARD TRAP, LEVEL 14)

An oscillator failure trap event is generated for any of the following reasons:

• The Fail-Safe Clock Monitor (FSCM) is enabled and has detected a loss of the system clock source.
• A loss of PLL lock has been detected during normal operation using the PLL.
• The FSCM is enabled and the PLL fails to achieve lock at a Power-on Reset (POR).
An oscillator failure trap event can be detected in software by polling the Oscillator Failure Trap (OSCFAIL) status bit (INTCON1<1>) or the Clock Fail (CF) status bit (OSCCON<3>) in the Oscillator module. To avoid re-entering the Trap Service Routine (TSR), the OSCFAIL status flag must be cleared in software with a RETFIE instruction before the program returns from the trap.

29.2.2.3 ADDRESS ERROR TRAP (HARD TRAP, LEVEL 13)

Operating conditions that can generate an address error trap include the following:

- A misaligned data word fetch is attempted. This condition occurs when an instruction performs a word access with the Least Significant bit (LSb) of the effective address set to ‘1’. The dsPIC33F CPU requires all word accesses to be aligned to an even address boundary.
- A bit manipulation instruction uses the Indirect Addressing mode with the LSb of the effective address set to ‘1’.
- A data fetch is attempted from unimplemented data address space.
- Execution of a BRA #literal instruction or a GOTO #literal instruction, where literal is an unimplemented program memory address.
- Execution of instructions after the Program Counter has been modified to point to unimplemented program memory addresses. The Program Counter can be modified by loading a value into the stack and executing a RETURN instruction.

When an address error trap occurs, data space writes are inhibited so that data is not destroyed. An address error can be detected in software by polling the ADDRERR status bit (INTCON1<3>). To avoid re-entering the Trap Service Routine, the ADDRERR status flag must be cleared in software with a RETFIE instruction before the program returns from the trap.

| Note: | In the MAC class of instructions, the data space is split into X and Y spaces. In these instructions, unimplemented X space includes all of Y space, and unimplemented Y space includes all of X space. |

29.2.3 Disable Interrupts Instruction

The DISI (disable interrupts) instruction can disable interrupts for up to 16384 instruction cycles. This instruction is useful for executing time-critical code segments. The DISI instruction only disables interrupts with priority levels 1-6. Priority level 7 interrupts and all trap events can still interrupt the CPU when the DISI instruction is active.

The DISI instruction works in conjunction with the Disable Interrupts Count (DISICNT) register in the CPU. When the DISICNT register is non-zero, priority level 1-6 interrupts are disabled. The DISICNT register is decremented on each subsequent instruction cycle. When the DISICNT register counts down to zero, priority level 1-6 interrupts are re-enabled. The value specified in the DISI instruction includes all cycles due to PSV accesses, instruction stalls, etc.

The DISICNT register is both readable and writable. The user application can terminate the effect of a previous DISI instruction early by clearing the DISICNT register. The time that interrupts are disabled can also be increased by writing to or adding to the DISICNT register.

If the DISICNT register is zero, interrupts cannot be disabled by simply writing a non-zero value to the register. Interrupts must first be disabled by using the DISI instruction. Once the DISI instruction has executed and DISICNT holds a non-zero value, the application can extend the interrupt disable time by modifying the contents of DISICNT.

The DISI Instruction (DISI) status bit (INTCON2<14>) is set whenever interrupts are disabled as a result of the DISI instruction.

| Note: | The DISI instruction can be used to quickly disable all user interrupt sources if no source is assigned to CPU priority level 7. |

29.2.4 Interrupt Operation

All interrupt event flags are sampled during each instruction cycle. A pending Interrupt Request (IRQ) is indicated by the flag bit = 1 in an IFSx register. The IRQ causes an interrupt, if the corresponding bit in the Interrupt Enable (IECx) registers is set. For the rest of the instruction cycle in which the IRQ is sampled, the priorities of all pending interrupt requests are evaluated.
No instruction is aborted when the CPU responds to the IRQ. The instruction in progress when the IRQ is sampled is completed before the ISR is executed.

If there is a pending IRQ with a user-assigned priority level greater than the current processor priority level that is indicated by the IPL<2:0> status bits (SR<7:5>), an interrupt is presented to the processor. The processor then saves the following information on the software stack:

- Current PC value
- Low byte of the Processor Status register (SRL)
- IPL3 status bit (CORCON<3>)

These three values allow the return Program Counter address value, MCU status bits, and the current processor priority level to be automatically saved.

After this information is saved on the stack, the CPU writes the priority level of the pending interrupt into the IPL<2:0> bit locations. This action disables all interrupts of lower or equal priority until the ISR is terminated using the RETFIE instruction.

**Figure 29-2: Stack Operation for Interrupt Event**

<table>
<thead>
<tr>
<th>Stack Grows Toward Higher Address</th>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC&lt;15:0&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SR&lt;7:0&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC&lt;22:16&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;Free Word&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This stack stores the IPL3 status bit (CORCON<3>)

W15 (before IRQ)

W15 (after IRQ)

29.2.4.1 RETURN FROM INTERRUPT

The **RETFIE** instruction unstacks the PC return address, IPL3 status bit, and SRL register to return the processor to the state and priority level that existed before the interrupt sequence.

29.2.4.2 INTERRUPT NESTING

Interrupts, by default, are nestable. Any ISR in progress can be interrupted by another source of interrupt with a higher user application-assigned priority level. Interrupt nesting can be disabled by setting the Interrupt Nesting Disable (NSTDIS) control bit (INTCON1<15>). When the NSTDIS control bit is set, all interrupts in progress force the CPU priority to level 7 by setting IPL<2:0> = 111. This action effectively masks all other sources of interrupt until a **RETFIE** instruction is executed. When interrupt nesting is disabled, the user application-assigned interrupt priority levels have no effect except to resolve conflicts between simultaneous pending interrupts.

The IPL<2:0> bits (SR<7:5>) become read-only when interrupt nesting is disabled. This prevents the user software from setting IPL<2:0> to a lower value, and that effectively re-enables interrupt nesting.
29.2.5 Wake-up from Sleep and Idle

Any source of interrupt that is individually enabled using its corresponding control bit in the IECx registers can wake-up the processor from Sleep or Idle mode. When the interrupt status flag for a source is set and the interrupt source is enabled by the corresponding bit in the IEC Control registers, a wake-up signal is sent to the dsPIC33F CPU. When the device wakes from Sleep or Idle mode, one of two actions will occur:

- If the interrupt priority level for that source is greater than the current CPU priority level, the processor will process the interrupt and branch to the ISR for the interrupt source.
- If the user application-assigned interrupt priority level for the source is less than or equal to the current CPU priority level, the processor will continue execution, starting with the instruction immediately following the PWRSAV instruction that previously put the CPU in Sleep or Idle mode.

| Note: | User interrupt sources that are assigned to CPU priority level 0 cannot wake the CPU from Sleep or Idle mode, because the interrupt source is effectively disabled. To use an interrupt as a wake-up source, the user application must assign the CPU priority level for the interrupt to level 1 or greater. |

29.2.6 A/D Converter External Conversion Request

The INT0 external interrupt request pin is shared with the A/D converter as an external conversion request signal. The INT0 interrupt source has programmable edge polarity, that is also available to the A/D converter external conversion request feature.

29.2.7 External Interrupt Support

The dsPIC33F supports up to three external interrupt pin sources (INT0-INT2). Each external interrupt pin has edge detection circuitry to detect the interrupt event. The INTCON2 register has three control bits (INT0EP-INT2EP) that select the polarity of the edge detection circuitry. Each external interrupt pin can be programmed to interrupt the CPU on a rising edge or falling edge event. See Register 29-4 for further details.
29.3 INTERRUPT PROCESSING TIMING

29.3.1 Interrupt Latency for One-Cycle Instructions

Figure 29-3 shows the sequence of events when a peripheral interrupt is asserted during a one-cycle instruction. The interrupt process takes four instruction cycles. Each cycle is numbered in Figure 29-3 for reference.

The interrupt flag status bit is set during the instruction cycle after the peripheral interrupt occurs. The current instruction completes during this instruction cycle. In the second instruction cycle after the interrupt event, the contents of the Program Counter (PC) and Lower-Byte Status (SRL) registers are saved into a temporary buffer register. The second cycle of the interrupt process is executed as a NOP instruction to maintain consistency with the sequence taken during a two-cycle instruction (see 29.3.2 “Interrupt Latency for Two-Cycle Instructions”). In the third cycle, the PC is loaded with the vector table address for the interrupt source and the starting address of the ISR is fetched. In the fourth cycle, the PC is loaded with the ISR address. The fourth cycle is executed as a NOP instruction, while the first instruction in the ISR is fetched.

Figure 29-3: Interrupt Timing During a One-Cycle Instruction

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PC</td>
<td>INST-2</td>
<td>INST(PC)</td>
<td>FNOP</td>
<td>FNOP</td>
<td>ISR</td>
<td>ISR+2</td>
</tr>
<tr>
<td>2</td>
<td>INST(PC)</td>
<td>FNOP</td>
<td>Vector</td>
<td>ISR</td>
<td>ISR+2</td>
<td>ISR+4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CPU Priority</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Peripheral interrupt event occurs at or before midpoint of this cycle</td>
<td>Save PC in temporary buffer</td>
<td>PUSH Low 16 bits of PC (from temporary buffer)</td>
<td>PUSH SRL and High 8 bits of PC (from temporary buffer)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
29.3.2 Interrupt Latency for Two-Cycle Instructions

The interrupt latency during a two-cycle instruction is the same as during a one-cycle instruction. The first and second cycle of the interrupt process allow the two-cycle instruction to complete execution. The timing diagram in Figure 29-4 shows the peripheral interrupt event occurring in the instruction cycle prior to execution of the two-cycle instruction.

Figure 29-5 shows the timing when a peripheral interrupt coincides with the first cycle of a two-cycle instruction. In this case, the interrupt process completes as for a one-cycle instruction (see 29.3.1 “Interrupt Latency for One-Cycle Instructions”).

Figure 29-4: Interrupt Timing During a Two-Cycle Instruction

Figure 29-5: Interrupt Timing, Interrupt Occurs During 1st Cycle of a 2-Cycle Instruction
29.3.3 Returning from Interrupt

To return from an interrupt, the program must call the RETFIE instruction. During the first two cycles of a RETFIE instruction, the contents of the PC and the SRL register are popped from the stack. The third instruction cycle is used to fetch the instruction addressed by the updated program counter. This cycle executes as a NOP instruction. On the fourth cycle, program execution resumes at the point where the interrupt occurred.

![Return from Interrupt Timing Diagram]

29.3.4 Special Conditions for Interrupt Latency

The dsPIC33F allows the current instruction to complete when a peripheral interrupt source becomes pending. The interrupt latency is the same for both one- and two-cycle instructions. However, certain conditions can increase interrupt latency by one cycle, depending on when the interrupt occurs. If a fixed latency is critical to the application, you must avoid the following conditions:

- Executing a MOV.D instruction that uses PSV to access a value in program memory space
- Appending an instruction stall cycle to any two-cycle instruction
- Appending an instruction stall cycle to any one-cycle instruction that performs a PSV access
- A bit test and skip instruction (BTSC, BTSS) that uses PSV to access a value in the program memory space
29.4 INTERRUPT CONTROL AND STATUS REGISTERS

The following registers are associated with the interrupt controller:

- **INTCON1, INTCON2 Registers**

  The following registers control global interrupt functions:
  - INTCON1 contains the Interrupt Nesting Disable (NSTDIS) bit, as well as the control and status flags for the processor trap sources
  - INTCON2 controls external interrupt request signal behavior and use of the alternate vector table

- **IFSx: Interrupt Flag Status Registers**

  All interrupt request flags are maintained in the IFSx registers, where ‘x’ denotes the register number. Each source of interrupt has a status bit which is set by the respective peripherals or external signal and cleared by software.

- **IECx: Interrupt Enable Control Registers**

  All Interrupt Enable Control bits are maintained in the IECx registers, where ‘x’ denotes the register number. These control bits are used to individually enable interrupts from the peripherals or external signals.

- **IPCx: Interrupt Priority Control Registers**

  Each user interrupt source can be assigned to one of eight priority levels. The IPC registers set the interrupt priority level for each source of interrupt.

- **SR: CPU STATUS Register**

  The SR is not specifically part of the interrupt controller hardware, but it contains the IPL<2:0> Status bits (SR<7:5>), which indicate the current CPU priority level. The user application can change the current CPU priority level by writing to the IPL bits.

- **CORCON: Core Control Register**

  The CORCON is not specifically part of the interrupt controller hardware, but it contains the IPL3 Status bit, which indicates the current CPU priority level. IPL3 is a read-only bit so that trap events cannot be masked by the user application.

- **INTTREG: Interrupt Control and Status Register**

  The INTTREG register contains the associated interrupt vector number and the new CPU interrupt priority level, which are latched into Vector Number (VECNUM<6:0>) and Interrupt level (ILR<3:0>) bit fields in the INTTREG register. The new interrupt priority level is the priority of the pending interrupt.

Each register is described in detail in the following sections.

---

**Note:** The total number and type of interrupt sources depend on the device variant. Refer to the specific device data sheet for further details.

### 29.4.1 Assignment of Interrupts to Control Registers

The interrupt sources are assigned to the IFSx, IECx,0 and IPCx registers in the same sequence that they are listed in Table 29-1. For example, the INT0 (External Interrupt 0) source has vector number and natural order priority 0. Therefore, the External Interrupt 0 Flag Status (INT0IF) bit exists in IFS0<0>. The INT0 interrupt uses bit 0 of the IEC0 register as its Enable bit. The IPC0<2:0> bits assign the interrupt priority level for the INT0 interrupt.
Register 29-1: SR: CPU STATUS Register

<table>
<thead>
<tr>
<th>R-0</th>
<th>R-0</th>
<th>R/C-0</th>
<th>R/C-0</th>
<th>R-0</th>
<th>R/C-0</th>
<th>R -0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OA</td>
<td>OB</td>
<td>SA</td>
<td>SB</td>
<td>OAB</td>
<td>SAB</td>
<td>DA</td>
<td>DC</td>
</tr>
</tbody>
</table>

bit 15

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPL&lt;2:0&gt;</td>
<td>RA</td>
<td>N</td>
<td>OV</td>
<td>Z</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

bit 7

Legend:

C = Clear only bit  R = Readable bit  U = Unimplemented bit, read as ‘0’
S = Set only bit   W = Writable bit   -n = Value at POR
‘1’ = Bit is set  ‘0’ = Bit is cleared   x = Bit is unknown

bit 15-8  Not used by the Interrupt Controller
(See the “dsPIC30F/33F Programmer’s Reference Manual” (DS70157) for descriptions of the SR bits.)

bit 7-5  IPL<2:0>: CPU Interrupt Priority Level Status bits(1,2)

111 = CPU Interrupt Priority Level is 7 (15); user interrupts disabled
110 = CPU Interrupt Priority Level is 6 (14)
101 = CPU Interrupt Priority Level is 5 (13)
100 = CPU Interrupt Priority Level is 4 (12)
011 = CPU Interrupt Priority Level is 3 (11)
010 = CPU Interrupt Priority Level is 2 (10)
001 = CPU Interrupt Priority Level is 1 (9)
000 = CPU Interrupt Priority Level is 0 (8)

bit 4-0  Not used by the Interrupt Controller
(See the “dsPIC30F/33F Programmer’s Reference Manual” (DS70157) for descriptions of the SR bits.)

Note 1: The IPL<2:0> bits are concatenated with the IPL<3> bit (CORCON<3>) to form the CPU Interrupt Priority Level. The value in parentheses indicates the IPL if IPL<3> = 1. User interrupts are disabled when IPL<3> = 1.

2: The IPL<2:0> Status bits are read-only when NSTDIS (INTCON1<15>) = 1.
Register 29-2:  CORCON: Core Control Register

<table>
<thead>
<tr>
<th>bit 15-0</th>
<th>R/W-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/C-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SATA</td>
<td>SATB</td>
<td>SATDW</td>
<td>ACCSAT</td>
<td>IPL3</td>
<td>PSV</td>
<td>RND</td>
<td>IF</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

- Not used by the Interrupt Controller

(bit 15-4)
(See the "dsPIC30F/33F Programmer’s Reference Manual" (DS70157) for descriptions of the CORCON bits.)

(bit 3)
**IPL3**: CPU Interrupt Priority Level Status bit 3\(^{(1)}\)

- 1 = CPU interrupt priority level is greater than 7
- 0 = CPU interrupt priority level is 7 or less

(bit 2-0)
**Not used by the Interrupt Controller**
(See the "dsPIC30F/33F Programmer’s Reference Manual" (DS70157) for descriptions of the CORCON bits.)

**Note 1**: The IPL3 bit is concatenated with the IPL<2:0> bits (SR<7:5>) to form the CPU interrupt priority level.
### Register 29-3: INTCON1: Interrupt Control Register 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>Value at POR</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>NSTDIS</td>
<td>Interrupt Nesting Disable bit</td>
<td>0</td>
<td>Interrupt nesting is enabled</td>
</tr>
<tr>
<td>14</td>
<td>OVAERR</td>
<td>Accumulator A Overflow Trap Flag bit</td>
<td>0</td>
<td>Trap was not caused by overflow of Accumulator A</td>
</tr>
<tr>
<td>13</td>
<td>OVBERR</td>
<td>Accumulator B Overflow Trap Flag bit</td>
<td>0</td>
<td>Trap was not caused by overflow of Accumulator B</td>
</tr>
<tr>
<td>12</td>
<td>COVAERR</td>
<td>Accumulator A Catastrophic Overflow Trap Flag bit</td>
<td>0</td>
<td>Trap was not caused by catastrophic overflow of Accumulator A</td>
</tr>
<tr>
<td>11</td>
<td>COVBERR</td>
<td>Accumulator B Catastrophic Overflow Trap Flag bit</td>
<td>0</td>
<td>Trap was not caused by catastrophic overflow of Accumulator B</td>
</tr>
<tr>
<td>10</td>
<td>OVATE</td>
<td>Accumulator A Overflow Trap Enable bit</td>
<td>0</td>
<td>Trap enabled</td>
</tr>
<tr>
<td>9</td>
<td>OVBTE</td>
<td>Accumulator B Overflow Trap Enable bit</td>
<td>0</td>
<td>Trap disabled</td>
</tr>
<tr>
<td>8</td>
<td>COVTE</td>
<td>Catastrophic Overflow Trap Enable bit</td>
<td>0</td>
<td>Trap disabled</td>
</tr>
<tr>
<td>7</td>
<td>SFTACERR</td>
<td>Shift Accumulator Error Status bit</td>
<td>0</td>
<td>Math error trap was not caused by an invalid accumulator shift</td>
</tr>
<tr>
<td>6</td>
<td>DIV0ERR</td>
<td>Divide-by-zero Error Status bit</td>
<td>0</td>
<td>No Divide-by-zero error trap</td>
</tr>
<tr>
<td>5</td>
<td>ADDRERR</td>
<td>Address Error Trap Status bit</td>
<td>0</td>
<td>Address error trap has not occurred</td>
</tr>
<tr>
<td>4</td>
<td>MATHERR</td>
<td>Math Error Status bit</td>
<td>0</td>
<td>Math error trap has not occurred</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown
Register 29-3: **INTCON1: Interrupt Control Register 1 (Continued)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value 1</th>
<th>Value 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td><strong>STKERR: Stack Error Trap Status bit</strong></td>
<td>Stack error trap has occurred</td>
<td>Stack error trap has not occurred</td>
</tr>
<tr>
<td>1</td>
<td><strong>OSCFAIL: Oscillator Failure Trap Status bit</strong></td>
<td>Oscillator failure trap has occurred</td>
<td>Oscillator failure trap has not occurred</td>
</tr>
<tr>
<td>0</td>
<td><strong>Unimplemented: Read as '0'</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Register 29-4: INTCON2: Interrupt Control Register 2

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTIVT</td>
<td>DISI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 15

Legend:

R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’

-n = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown

bit 15  ALTVT: Enable Alternate Interrupt Vector Table bit
        1 = Use alternate vector table
        0 = Use standard (default) vector table

bit 14  DISI: DISI (disable interrupts) Instruction Status bit
        1 = DISI instruction is active
        0 = DISI instruction is not active

bit 13-3 Unimplemented: Read as ‘0’

bit 2  INT2EP: External Interrupt 2 Edge Detect Polarity Select bit
        1 = Interrupt on negative edge
        0 = Interrupt on positive edge

bit 1  INT1EP: External Interrupt 1 Edge Detect Polarity Select bit
        1 = Interrupt on negative edge
        0 = Interrupt on positive edge

bit 0  INT0EP: External Interrupt 0 Edge Detect Polarity Select bit
        1 = Interrupt on negative edge
        0 = Interrupt on positive edge
### Register 29-5: IFS0: Interrupt Flag Status Register 0

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>AD1IF</td>
<td>U1TXIF</td>
<td>U1RXIF</td>
<td>SPI1IF</td>
<td>SPI1EIF</td>
<td>T3IF</td>
</tr>
</tbody>
</table>

**Legend:**

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’

- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

- **Unimplemented**: Read as ‘0’

- **AD1IF**: ADC1 Conversion Complete Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred

- **U1TXIF**: UART1 Transmitter Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred

- **U1RXIF**: UART1 Receiver Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred

- **SPI1IF**: SPI1 Event Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred

- **SPI1EIF**: SPI1 Fault Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred

- **T3IF**: Timer3 Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred

- **T2IF**: Timer2 Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred

- **OC2IF**: Output Compare Channel 2 Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred

- **IC2IF**: Input Capture Channel 2 Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred

- **T1IF**: Timer1 Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred

- **OC1IF**: Output Compare Channel 1 Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred
### Register 29-5: IFS0: Interrupt Flag Status Register 0 (Continued)

<table>
<thead>
<tr>
<th>bit</th>
<th>Description</th>
<th>Value</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>IC1IF</strong>: Input Capture Channel 1 Interrupt Flag Status bit</td>
<td>1 = Interrupt request has occurred</td>
<td>0 = Interrupt request has not occurred</td>
</tr>
<tr>
<td>0</td>
<td><strong>INT0IF</strong>: External Interrupt 0 Flag Status bit</td>
<td>1 = Interrupt request has occurred</td>
<td>0 = Interrupt request has not occurred</td>
</tr>
</tbody>
</table>
### Register 29-6: IFS1: Interrupt Flag Status Register 1

| Bit 15-14 | **Unimplemented:** Read as '0' |
| Bit 13    | **INT2IF:** External Interrupt 2 Flag Status bit |
| 1         | Interrupt request has occurred |
| 0         | Interrupt request has not occurred |
| Bit 12-8  | **Unimplemented:** Read as '0' |
| Bit 7     | **IC8IF:** Input Capture Channel 8 Interrupt Flag Status bit |
| 1         | Interrupt request has occurred |
| 0         | Interrupt request has not occurred |
| Bit 6     | **IC7IF:** Input Capture Channel 7 Interrupt Flag Status bit |
| 1         | Interrupt request has occurred |
| 0         | Interrupt request has not occurred |
| Bit 5     | **Unimplemented:** Read as '0' |
| Bit 4     | **INT1IF:** External Interrupt 1 Flag Status bit |
| 1         | Interrupt request has occurred |
| 0         | Interrupt request has not occurred |
| Bit 3     | **CNIF:** Input Change Notification Interrupt Flag Status bit |
| 1         | Interrupt request has occurred |
| 0         | Interrupt request has not occurred |
| Bit 2     | **Unimplemented:** Read as '0' |
| Bit 1     | **MI2C1IF:** I2C1 Master Events Interrupt Flag Status bit |
| 1         | Interrupt request has occurred |
| 0         | Interrupt request has not occurred |
| Bit 0     | **SI2C1IF:** I2C1 Slave Events Interrupt Flag Status bit |
| 1         | Interrupt request has occurred |
| 0         | Interrupt request has not occurred |

**Legend:**

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- **x** = Bit is unknown

<table>
<thead>
<tr>
<th>Bit 15-8</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC8IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>IC7IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>INT1IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>CNIF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>MI2C1IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SI2C1IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

| Bit 7     | R/W-0 |
| IC8IF    | —     |
| IC7IF    | —     |
| INT1IF   | —     |
| CNIF     | —     |
| MI2C1IF  | —     |
| SI2C1IF  | —     |

| Bit 0     | R/W-0 |
| SI2C1IF  | —     |
### Register 29-7: IFS3: Interrupt Flag Status Register 3

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>FLTA1IF</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>QEIIF</th>
<th>PWM1IF</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 15</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>bit 14-11</td>
<td>Unimplemented:</td>
<td>Read as '0'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 10</td>
<td>QEIIF: QEI Event Interrupt Flag Status bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Interrupt request has occurred</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt request has not occurred</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 9</td>
<td>PWM1IF: PWM1 Error Interrupt Flag Status bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Interrupt request has occurred</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt request has not occurred</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 8-0</td>
<td>Unimplemented:</td>
<td>Read as '0'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- **x** = Bit is unknown

- **bit 15**  FLTA1IF: PWM1 Fault A Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred
- **bit 14-11**  Unimplemented: Read as '0'
- **bit 10**  QEIIF: QEI Event Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred
- **bit 9**  PWM1IF: PWM1 Error Interrupt Flag Status bit
  - 1 = Interrupt request has occurred
  - 0 = Interrupt request has not occurred
- **bit 8-0**  Unimplemented: Read as '0'
Register 29-8: IFS4: Interrupt Flag Status Register 4

<table>
<thead>
<tr>
<th>bit 15-0</th>
<th>Unimplemented: Read as '0'</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 10</td>
<td>FLTA2IF: PWM2 Fault A Interrupt Flag Status bit</td>
</tr>
<tr>
<td></td>
<td>1 = Interrupt request has occurred</td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt request has not occurred</td>
</tr>
<tr>
<td>bit 9</td>
<td>PWM2IF: PWM2 Error Interrupt Enable bit</td>
</tr>
<tr>
<td></td>
<td>1 = Interrupt request has occurred</td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt request has not occurred</td>
</tr>
<tr>
<td>bit 8-2</td>
<td>Unimplemented: Read as '0'</td>
</tr>
<tr>
<td>bit 1</td>
<td>U1EIF: UART1 Error Interrupt Flag Status bit</td>
</tr>
<tr>
<td></td>
<td>1 = Interrupt request has occurred</td>
</tr>
<tr>
<td></td>
<td>0 = Interrupt request has not occurred</td>
</tr>
<tr>
<td>bit 0</td>
<td>Unimplemented: Read as '0'</td>
</tr>
</tbody>
</table>
## Section 29. Interrupts (Part II)

### Register 29-9: IEC0: Interrupt Enable Control Register 0

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AD1IE</td>
<td>U1TXIE</td>
<td>U1RXIE</td>
<td>SPI1IE</td>
<td>SPI1EIE</td>
<td>T3IE</td>
</tr>
<tr>
<td>bit 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2IE</td>
<td>OC2IE</td>
<td>IC2IE</td>
<td></td>
<td>T1IE</td>
<td>OC1IE</td>
<td>IC1IE</td>
<td>INT0IE</td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

- **bit 15-14**: Unimplemented: Read as ‘0’
- **bit 13**: AD1IE: ADC1 Conversion Complete Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 12**: U1TXIE: UART1 Transmitter Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 11**: U1RXIE: UART1 Receiver Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 10**: SPI1IE: SPI1 Event Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 9**: SPI1EIE: SPI1 Event Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 8**: T3IE: Timer3 Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 7**: T2IE: Timer2 Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 6**: OC2IE: Output Compare Channel 2 Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 5**: IC2IE: Input Capture Channel 2 Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 4**: Unimplemented: Read as ‘0’
- **bit 3**: T1IE: Timer1 Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 2**: OC1IE: Output Compare Channel 1 Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
Register 29-9: IEC0: Interrupt Enable Control Register 0 (Continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Setting</th>
<th>Description</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>IC1IE</strong>: Input Capture Channel 1 Interrupt Enable bit</td>
<td>1 = Interrupt request enabled</td>
<td>0 = Interrupt request not enabled</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td><strong>INT0IE</strong>: External Interrupt 0 Enable bit</td>
<td>1 = Interrupt request enabled</td>
<td>0 = Interrupt request not enabled</td>
<td></td>
</tr>
</tbody>
</table>
### Section 29. Interrupts (Part II)

Register 29-10: IEC1: Interrupt Enable Control Register 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>Value at POR</th>
<th>Enabled State</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>INT2IE</td>
<td>External Interrupt 2 Enable bit</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14-12</td>
<td>Unimplemented: Read as '0'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11-8</td>
<td>Unimplemented: Read as '0'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>IC8IE</td>
<td>Input Capture Channel 8 Interrupt Enable bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>IC7IE</td>
<td>Input Capture Channel 7 Interrupt Enable bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>CNIE</td>
<td>Input Change Notification Interrupt Enable bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>INT1IE</td>
<td>External Interrupt 1 Enable bit</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>CNIE</td>
<td>Input Change Notification Interrupt Enable bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Unimplemented: Read as '0'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MI2C1IE</td>
<td>I2C1 Master Events Interrupt Enable bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>SI2C1IE</td>
<td>I2C1 Slave Events Interrupt Enable bit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
- **‘1’** = Bit is set
- **‘0’** = Bit is cleared
- **x** = Bit is unknown
Register 29-11:  IEC3: Interrupt Enable Control Register 3

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLTA1IE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

bit 15  
**FLTA1IE**: PWM Fault A Interrupt Enable bit
  1 = Interrupt request enabled
  0 = Interrupt request not enabled

bit 14-11  
**Unimplemented**: Read as ‘0’

bit 10  
**QEIIIE**: QEI Event Interrupt Enable bit
  1 = Interrupt request enabled
  0 = Interrupt request not enabled

bit 9  
**PWM1IE**: PWM Error Interrupt Enable bit
  1 = Interrupt request enabled
  0 = Interrupt request not enabled

bit 8-0  
**Unimplemented**: Read as ‘0’

Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- ‘-n’ = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown
### Register 29-12: IEC4: Interrupt Enable Control Register 4

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FLTA2IE</td>
<td>PWM2IE</td>
<td></td>
</tr>
</tbody>
</table>

- **bit 15**
  - **FLTA2IE**: PWM2 Fault A Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled

- **bit 8**
  - **PWM2IE**: PWM2 Error Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U1EIE</td>
<td></td>
</tr>
</tbody>
</table>

- **bit 7**
  - **U1EIE**: UART1 Error Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

- **bit 15-11** **Unimplemented**: Read as ‘0’
- **bit 10** **FLTA2IE**: PWM2 Fault A Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 9** **PWM2IE**: PWM2 Error Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 8-2** **Unimplemented**: Read as ‘0’
- **bit 1** **U1EIE**: UART1 Error Interrupt Enable bit
  - 1 = Interrupt request enabled
  - 0 = Interrupt request not enabled
- **bit 0** **Unimplemented**: Read as ‘0’
### Register 29-13: IPC0: Interrupt Priority Control Register 0

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>T1IP&lt;2:0&gt;</td>
<td>—</td>
<td>OC1IP&lt;2:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**bit 15**

**bit 14-12**

- **T1IP<2:0>:** Timer1 Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 110 = Interrupt is priority 6
  - 101 = Interrupt is priority 5
  - 100 = Interrupt is priority 4
  - 011 = Interrupt is priority 3
  - 010 = Interrupt is priority 2
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled

**bit 11**

- **Unimplemented:** Read as '0'

**bit 10-8**

- **OC1IP<2:0>:** Output Compare Channel 1 Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 110 = Interrupt is priority 6
  - 101 = Interrupt is priority 5
  - 100 = Interrupt is priority 4
  - 011 = Interrupt is priority 3
  - 010 = Interrupt is priority 2
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled

**bit 7**

- **Unimplemented:** Read as '0'

**bit 6-4**

- **IC1IP<2:0>:** Input Capture Channel 1 Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 110 = Interrupt is priority 6
  - 101 = Interrupt is priority 5
  - 100 = Interrupt is priority 4
  - 011 = Interrupt is priority 3
  - 010 = Interrupt is priority 2
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled

**bit 3**

- **Unimplemented:** Read as '0'

**bit 2-0**

- **INT0IP<2:0>:** External Interrupt 0 Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 110 = Interrupt is priority 6
  - 101 = Interrupt is priority 5
  - 100 = Interrupt is priority 4
  - 011 = Interrupt is priority 3
  - 010 = Interrupt is priority 2
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled
### Section 29. Interrupts (Part II)

**Register 29-14: IPC1: Interrupt Priority Control Register 1**

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 15</td>
<td>T2IP&lt;2:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td>OC2IP&lt;2:0&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

**Bit Descriptions:**
- **bit 15** Unimplemented: Read as ‘0’
- **bit 14-12** T2IP<2:0>: Timer2 Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled
- **bit 11** Unimplemented: Read as ‘0’
- **bit 10-8** OC2IP<2:0>: Output Compare Channel 2 Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled
- **bit 7** Unimplemented: Read as ‘0’
- **bit 6-4** IC2IP<2:0>: Input Capture Channel 2 Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled
- **bit 3-0** Unimplemented: Read as ‘0’
Register 29-15: IPC2: Interrupt Priority Control Register 2

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>U1RXIP&lt;2:0&gt;</td>
<td>—</td>
<td>SPI1IP&lt;2:0&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 15

bit 14-12

Unimplemented: Read as ‘0’

U1RXIP<2:0>: UART1 Receiver Interrupt Priority bits

111 = Interrupt is priority 7 (highest priority interrupt)

000 = Interrupt source is disabled

bit 11

Unimplemented: Read as ‘0’

bit 10-8

SPI1IP<2:0>: SPI1 Event Interrupt Priority bits

111 = Interrupt is priority 7 (highest priority interrupt)

000 = Interrupt source is disabled

bit 7

Unimplemented: Read as ‘0’

bit 6-4

SPI1EIP<2:0>: SPI1 Error Interrupt Priority bits

111 = Interrupt is priority 7 (highest priority interrupt)

000 = Interrupt source is disabled

bit 3

Unimplemented: Read as ‘0’

bit 2-0

T3IIP<2:0>: Timer3 Interrupt Priority bits

111 = Interrupt is priority 7 (highest priority interrupt)

000 = Interrupt source is disabled

Legend:

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as ‘0’

-n = Value at POR  
‘1’ = Bit is set  
‘0’ = Bit is cleared  
x = Bit is unknown
### Register 29-16: IPC3: Interrupt Priority Control Register 3

<table>
<thead>
<tr>
<th>bit 15-7</th>
<th>bit 6-4</th>
<th>bit 3</th>
<th>bit 2-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented: Read as '0'</td>
<td>AD1IP&lt;2:0&gt;: ADC1 Conversion Complete Interrupt Priority bits</td>
<td>Unimplemented: Read as '0'</td>
<td>U1TXIP&lt;2:0&gt;: UART1 Transmitter Interrupt Priority bits</td>
</tr>
<tr>
<td><strong>Legend:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R = Readable bit</td>
<td>W = Writable bit</td>
<td>U = Unimplemented bit, read as '0'</td>
<td></td>
</tr>
<tr>
<td>-n = Value at POR</td>
<td>‘1’ = Bit is set</td>
<td>‘0’ = Bit is cleared</td>
<td>x = Bit is unknown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>AD1IP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>U1TXIP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

#### bit 15
- Value at POR: '1' = Bit is set, Read as '0'
- '0' = Bit is cleared
- x = Bit is unknown

#### bit 6-4
- AD1IP<2:0>: ADC1 Conversion Complete Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - •
  - •
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled

#### bit 3
- Unimplemented: Read as '0'

#### bit 2-0
- U1TXIP<2:0>: UART1 Transmitter Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - •
  - •
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled
### Register 29-17: IPC4: Interrupt Priority Control Register 4

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNIP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**bit 15**

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MI2C1IP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>SI2C1IP&lt;2:0&gt;</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**bit 7**

**Legend:**

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

**bit 15**

- **Unimplemented**: Read as '0'

**bit 14-12**

- **CNIP<2:0>**: Change Notification Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled

**bit 11-7**

- **Unimplemented**: Read as '0'

**bit 6-4**

- **MI2C1IP<2:0>**: I2C1 Master Events Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled

**bit 3**

- **Unimplemented**: Read as '0'

**bit 2-0**

- **SI2C1IP<2:0>**: I2C1 Slave Events Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled
Register 29-18: IPC5: Interrupt Priority Control Register 5

<table>
<thead>
<tr>
<th>bit 15</th>
<th>bit 14-12</th>
<th>bit 11</th>
<th>bit 10-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>IC8IP&lt;2:0&gt;</td>
<td>U-0</td>
<td>IC7IP&lt;2:0&gt;</td>
</tr>
<tr>
<td>R/W-1</td>
<td></td>
<td>R/W-0</td>
<td></td>
</tr>
<tr>
<td>U-0</td>
<td></td>
<td>R/W-1</td>
<td></td>
</tr>
<tr>
<td>R/W-0</td>
<td></td>
<td>R/W-0</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

Bit 15: Unimplemented: Read as ‘0’
Bit 14-12: IC8IP<2:0>: Input Capture Channel 8 Interrupt Priority bits
- 111 = Interrupt is priority 7 (highest priority interrupt)
- 001 = Interrupt is priority 1
- 000 = Interrupt source is disabled

Bit 11: Unimplemented: Read as ‘0’
Bit 10-8: IC7IP<2:0>: Input Capture Channel 7 Interrupt Priority bits
- 111 = Interrupt is priority 7 (highest priority interrupt)
- 001 = Interrupt is priority 1
- 000 = Interrupt source is disabled

Bit 7-3: Unimplemented: Read as ‘0’
Bit 2-0: INT1IP<2:0>: External Interrupt 1 Priority bits
- 111 = Interrupt is priority 7 (highest priority interrupt)
- 001 = Interrupt is priority 1
- 000 = Interrupt source is disabled
Register 29-19: IPC7: Interrupt Priority Control Register 7

<table>
<thead>
<tr>
<th>bit 15</th>
<th>bit 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
-n = Value at POR
‘1’ = Bit is set
‘0’ = Bit is cleared
x = Bit is unknown

bit 15-7 Unimplemented: Read as ‘0’

bit 6-4 INT2IP<2:0>: External Interrupt 2 Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
*   *
*   *
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 3-0 Unimplemented: Read as ‘0’
### Section 29. Interrupts (Part II)

**Register 29-20: IPC14: Interrupt Priority Control Register 14**

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>( \text{QEIIIP}&lt;2:0&gt; )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>( \text{PWM1IP}&lt;2:0&gt; )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 0</td>
</tr>
</tbody>
</table>

**Legend:**

- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- \(-n\) = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- \(x\) = Bit is unknown

- **bit 15-11** Unimplemented: Read as ‘0’
- **bit 10-8** QEIIIP<2:0>: QEII Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 110
  - 101
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled
- **bit 7** Unimplemented: Read as ‘0’
- **bit 6-4** PWM1IP<2:0>: PWM1 Interrupt Priority bits
  - 111 = Interrupt is priority 7 (highest priority interrupt)
  - 110
  - 101
  - 001 = Interrupt is priority 1
  - 000 = Interrupt source is disabled
- **bit 3-0** Unimplemented: Read as ‘0’
Register 29-21: IPC15: Interrupt Priority Control Register 15

<table>
<thead>
<tr>
<th>bit 15</th>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FLTA1IP&lt;2:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

bit 15 **Unimplemented**: Read as ‘0’

bit 14-12 **FLTA1IP<2:0>**: PWM1 Fault A Interrupt Priority bits

- **111** = Interrupt is priority 7 (highest priority interrupt)
- **110** = Interrupt is priority 6
- **101** = Interrupt is priority 5
- **100** = Interrupt is priority 4
- **011** = Interrupt is priority 3
- **010** = Interrupt is priority 2
- **001** = Interrupt is priority 1
- **000** = Interrupt source is disabled

Register 29-22: IPC16: Interrupt Priority Control Register 16

<table>
<thead>
<tr>
<th>bit 15</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U1EIP&lt;2:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

bit 15-7 **Unimplemented**: Read as ‘0’

bit 6-4 **U1EIP<2:0>**: UART1 Error Interrupt Priority bits

- **111** = Interrupt is priority 7 (highest priority interrupt)
- **110** = Interrupt is priority 6
- **101** = Interrupt is priority 5
- **100** = Interrupt is priority 4
- **011** = Interrupt is priority 3
- **010** = Interrupt is priority 2
- **001** = Interrupt is priority 1
- **000** = Interrupt source is disabled

bit 3-0 **Unimplemented**: Read as ‘0’
### Section 29. Interrupts (Part II)

**Register 29-23: IPC18: Interrupt Priority Control Register 18**

| Bit 15-11 | Unimplemented: Read as '0' |
| Bit 10-8  | FLTA2IP<2:0>: PWM2 Fault A Interrupt Priority bits |
|           | • 111 = Interrupt is priority 7 (highest priority interrupt) |
|           | • • |
|           | 001 = Interrupt is priority 1 |
|           | 000 = Interrupt source is disabled |

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

| Bit 7     | Unimplemented: Read as ‘0’ |
| Bit 6-4   | PWM2IP<2:0>: PWM2 Interrupt Priority bits |
|           | • 111 = Interrupt is priority 7 (highest priority interrupt) |
|           | • • |
|           | 001 = Interrupt is priority 1 |
|           | 000 = Interrupt source is disabled |

| Bit 3-0   | Unimplemented: Read as ‘0’ |

---

© 2008 Microchip Technology Inc. Preliminary DS70189C-page 29-41
Register 29-24: INTTREG: Interrupt Control and Status Register

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>bit 15</td>
<td>bit 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>—</td>
<td>VECNUM&lt;6:0&gt;</td>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

- **Unimplemented:** Read as ‘0’
- **ILR<3:0>:** New CPU Interrupt Priority Level bits
  - 1111 = CPU Interrupt Priority Level is 15
  - 1110 =
  - 1101 =
  - 1100 =
  - 0001 = CPU Interrupt Priority Level is 1
  - 0000 = CPU Interrupt Priority Level is 0

- **Unimplemented:** Read as ‘0’

- **VECNUM<6:0>:** Vector Number of Pending Interrupt bits
  - 1111111 = Interrupt Vector pending is number 135
  - 1111110 =
  - 1111101 =
  - 1111100 =
  - 1111011 =
  - 1111010 =
  - 1111001 =
  - 1111000 =
  - 1110111 =
  - 1110110 =
  - 1110101 =
  - 1110100 =
  - 1110011 =
  - 1110010 =
  - 1110001 =
  - 1110000 =
  - 1101111 =
  - 1101110 =
  - 1101101 =
  - 1101100 =
  - 1101011 =
  - 1101010 =
  - 1101001 =
  - 1101000 =
  - 1100111 =
  - 1100110 =
  - 1100101 =
  - 1100100 =
  - 1100011 =
  - 1100010 =
  - 1100001 =
  - 1100000 =
  - 1011111 =
  - 1011110 =
  - 1011101 =
  - 1011100 =
  - 1011011 =
  - 1011010 =
  - 1011001 =
  - 1011000 =
  - 1010111 =
  - 1010110 =
  - 1010101 =
  - 1010100 =
  - 1010011 =
  - 1010010 =
  - 1010001 =
  - 1010000 =
  - 1001111 =
  - 1001110 =
  - 1001101 =
  - 1001100 =
  - 1001011 =
  - 1001010 =
  - 1001001 =
  - 1001000 =
  - 1000111 =
  - 1000110 =
  - 1000101 =
  - 1000100 =
  - 1000011 =
  - 1000010 =
  - 1000001 =
  - 1000000 =
  - 0111111 =
  - 0111110 =
  - 0111101 =
  - 0111100 =
  - 0111011 =
  - 0111010 =
  - 0111001 =
  - 0111000 =
  - 0110111 =
  - 0110110 =
  - 0110101 =
  - 0110100 =
  - 0110011 =
  - 0110010 =
  - 0110001 =
  - 0110000 =
  - 0101111 =
  - 0101110 =
  - 0101101 =
  - 0101100 =
  - 0101011 =
  - 0101010 =
  - 0101001 =
  - 0101000 =
  - 0100111 =
  - 0100110 =
  - 0100101 =
  - 0100100 =
  - 0100011 =
  - 0100010 =
  - 0100001 =
  - 0100000 =
  - 0011111 =
  - 0011110 =
  - 0011101 =
  - 0011100 =
  - 0011011 =
  - 0011010 =
  - 0011001 =
  - 0011000 =
  - 0010111 =
  - 0010110 =
  - 0010101 =
  - 0010100 =
  - 0010011 =
  - 0010010 =
  - 0010001 =
  - 0010000 =
  - 0001111 =
  - 0001110 =
  - 0001101 =
  - 0001100 =
  - 0001011 =
  - 0001010 =
  - 0001001 =
  - 0001000 =
  - 0000111 =
  - 0000110 =
  - 0000101 =
  - 0000100 =
  - 0000011 =
  - 0000010 =
  - 0000001 =
  - 0000000 =

- **Interrupt Vector pending is number 135**
- **Interrupt Vector pending is number 9**
- **Interrupt Vector pending is number 8**
Section 29. Interrupts (Part II)

29.5 INTERRUPT SETUP PROCEDURES

29.5.1 Initialization

To configure an interrupt source, do the following:

1. Set the NSTDIS control bit (INTCON1<15>), if you do not plan to use nested interrupts.
2. Select a user-assigned priority level for the interrupt source by writing the control bits in
   the appropriate IPCx Control register. The priority level depends on the specific
   application and type of interrupt source. If you do not plan to use multiple priority levels,
   program the IPCx register control bits for all enabled interrupt sources to the same
   non-zero value.

   Note: At a device Reset, the IPC registers are initialized with all user interrupt sources
   assigned to priority level 4.

3. Clear the interrupt flag status bit associated with the peripheral in the associated IFSx
   Status register.
4. Enable the interrupt source by setting the interrupt enable control bit associated with the
   source in the appropriate IECx Control register.

29.5.2 Interrupt Service Routine

The method used to declare an Interrupt Service Routine (ISR) and initialize the Interrupt
Vector Table with the correct vector address depends on the programming language (C or
Assembler) and the language development tool suite used to develop the application. In
general, the user application must clear the interrupt flag in the appropriate IFSx register for
the source of interrupt that the ISR handles. Otherwise, the application will immediately
re-enter the ISR after it exits the routine. If you code the ISR in Assembler, it must be
terminated using a RETFIE instruction to unstack the saved PC value, SRL value and old CPU
priority level.

29.5.3 Trap Service Routine

A Trap Service Routine (TSR) is coded like an ISR, except that the code must clear the
appropriate trap status flag in the INTCON1 register to avoid re-entry into the TSR.

29.5.4 Interrupt Disable

To disable interrupts, do the following:

1. Push the current SR value onto the software stack using the PUSH instruction.
2. Force the CPU to priority level 7 by inclusive ORing the value 0xE0 with SRL.

To enable interrupts, you can use the POP instruction to restore the previous SR value.

   Note: Only interrupts with a priority level of 7 or less can be disabled. Trap sources
   (level 8-level 15) cannot be disabled.

The DISI instruction disables interrupts of priority levels 1-6 for a fixed period of time. Level 7
interrupt sources are not disabled by the DISI instruction.
29.5.5 Code Example

Example 29-1 provides sample code that enables nested interrupts and sets up Timer1, Timer2, Timer3, and change notice peripherals to priority levels 2, 5, 6 and 4, respectively. It also illustrates how interrupts can be enabled and disabled using the Status Register. Sample ISRs illustrate interrupt clearing.

Example 29-1: Nested Interrupt Code Example

```c
void enableInterrupts(void)
{
    /* Set CPU IPL to 0, enable level 1-7 interrupts */
    /* No restoring of previous CPU IPL state performed here */
    SRbits.IPL = 0;

    return;
}

void disableInterrupts(void)
{
    /* Set CPU IPL to 7, disable level 1-7 interrupts */
    /* No saving of current CPU IPL setting performed here */
    SRbits.IPL = 7;

    return;
}

void initInterrupts(void)
{
    /* Interrupt nesting enabled here */
    INTCON1bits.NSTDIS = 0;

    /* Set Timer3 interrupt priority to 6 (level 7 is highest) */
    IPC2bits.T3IP = 6;

    /* Set Timer2 interrupt priority to 5 */
    IPC1bits.T2IP = 5;

    /* Set Change Notice interrupt priority to 4 */
    IPC4bits.CNIP = 4;

    /* Set Timer1 interrupt priority to 2 */
    IPC0bits.T1IP = 2;

    /* Reset Timer1 interrupt flag */
    IFS0bits.T1IF = 0;

    /* Reset Timer2 interrupt flag */
    IFS0bits.T2IF = 0;

    /* Reset Timer3 interrupt flag */
    IFS0bits.T3IF = 0;

    /* Enable CN interrupts */
    IEC1bits.CNIE = 1;
}
```
Example 29-1: Interrupt Setup Code Example (Continued)

    /* Enable Timer1 interrupt */
    IEC0bits.T1IE = 1;

    /* Enable Timer2 interrupt (PWM time base) */
    IEC0bits.T2IE = 1;

    /* Enable Timer3 interrupt */
    IEC0bits.T3IE = 1;

    /* Reset change notice interrupt flag */
    IFS1bits.CNIF = 0;
    return;
}

void __attribute__((__interrupt__)) _T1Interrupt(void)
{
    /* Insert ISR Code Here*/
    /* Clear Timer1 interrupt */
    IFS0bits.T1IF = 0;
}

void __attribute__((__interrupt__)) _T2Interrupt(void)
{
    /* Insert ISR Code Here*/
    /* Clear Timer2 interrupt */
    IFS0bits.T2IF = 0;
}

void __attribute__((__interrupt__)) _T3Interrupt(void)
{
    /* Insert ISR Code Here*/
    /* Clear Timer3 interrupt */
    IFS0bits.T3IF = 0;
}

void __attribute__((__interrupt__)) _CNInterrupt(void)
{
    /* Insert ISR Code Here*/
    /* Clear CN interrupt */
    IFS1bits.CNIF = 0;
}
### 29.6 REGISTER MAP

A summary of the Special Function Registers associated with the dsPIC33F Interrupts (Part II) module is provided in Table 29-2.

<table>
<thead>
<tr>
<th>SFR Name</th>
<th>SFR Addr</th>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>All Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON1</td>
<td>0080</td>
<td>NSTDIS</td>
<td>OVAERR</td>
<td>OVBERR</td>
<td>COVAERR</td>
<td>COVBERR</td>
<td>OVAE</td>
<td>OVBTE</td>
<td>COVTE</td>
<td>SFTACERR</td>
<td>DIV0ERR</td>
<td>—</td>
<td>MATHERR</td>
<td>ADDRERR</td>
<td>STKERR</td>
<td>OSCFAIL</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>INTCON2</td>
<td>0082</td>
<td>ALTIVT</td>
<td>DISI</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>IFS0</td>
<td>0084</td>
<td>—</td>
<td>—</td>
<td>AD1IF</td>
<td>U1TXIF</td>
<td>U1RXIF</td>
<td>SPI1IF</td>
<td>SPI1EIF</td>
<td>T3IF</td>
<td>T2IF</td>
<td>OC2IF</td>
<td>IC2IF</td>
<td>—</td>
<td>INT1F</td>
<td>OC1IF</td>
<td>IC1IF</td>
<td>INT0IF</td>
<td>—</td>
</tr>
<tr>
<td>IFS1</td>
<td>0086</td>
<td>—</td>
<td>—</td>
<td>INT2IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IC8IF</td>
<td>IC7IF</td>
<td>—</td>
<td>INT1IF</td>
<td>CNIF</td>
<td>—</td>
<td>MI2C1IF</td>
<td>SI2C1IF</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>IFS3</td>
<td>008A</td>
<td>FLTA1IF</td>
<td>—</td>
<td>—</td>
<td>QEIIF</td>
<td>PWM1IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>IFS4</td>
<td>008C</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FLTA2IF</td>
<td>PWM2IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>IEC0</td>
<td>0094</td>
<td>AD1IE</td>
<td>U1TXIE</td>
<td>U1RXIE</td>
<td>SPI1IE</td>
<td>SPI1IE</td>
<td>T3IE</td>
<td>T2IE</td>
<td>OC2IE</td>
<td>IC2IE</td>
<td>—</td>
<td>T1IE</td>
<td>OC1IE</td>
<td>IC1IE</td>
<td>INT0IE</td>
<td>—</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>IEC1</td>
<td>0096</td>
<td>—</td>
<td>—</td>
<td>INT2IE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IC8IE</td>
<td>IC7IE</td>
<td>—</td>
<td>INT1IE</td>
<td>CNIE</td>
<td>—</td>
<td>MI2C1IE</td>
<td>SI2C1IE</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>IEC3</td>
<td>009A</td>
<td>FLTA1IE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>QEIIE</td>
<td>PWM1IE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>IEC4</td>
<td>009C</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FLTA2IE</td>
<td>PWM2IE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>IPC0</td>
<td>00A4</td>
<td>T1IP&lt;2:0&gt;</td>
<td>—</td>
<td>OC1IP&lt;2:0&gt;</td>
<td>—</td>
<td>IC1IP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>U1IE</td>
<td>0000</td>
</tr>
<tr>
<td>IPC1</td>
<td>00A6</td>
<td>T2IP&lt;2:0&gt;</td>
<td>—</td>
<td>OC2IP&lt;2:0&gt;</td>
<td>—</td>
<td>IC2IP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>IPC2</td>
<td>00A8</td>
<td>U1RXIP&lt;2:0&gt;</td>
<td>—</td>
<td>SP11IP&lt;2:0&gt;</td>
<td>—</td>
<td>SP11IEP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>IPC3</td>
<td>00AA</td>
<td>—</td>
<td>—</td>
<td>AD1IP&lt;2:0&gt;</td>
<td>—</td>
<td>AD1IEP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>IPC4</td>
<td>00AC</td>
<td>CNIP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>IPC5</td>
<td>00AE</td>
<td>IC8IP&lt;2:0&gt;</td>
<td>—</td>
<td>IC7IP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>IPC7</td>
<td>00B2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>IPC14</td>
<td>00C0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>IPC15</td>
<td>00C2</td>
<td>FLTA1IP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>IPC16</td>
<td>00C4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>IPC18</td>
<td>00C8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FLTA2IP&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000</td>
</tr>
<tr>
<td>INTTREG</td>
<td>00E0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Legend:  
$x$ = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.
29.7 DESIGN TIPS

Question 1:  *What happens when two sources of interrupt become pending at the same time and have the same user application-assigned priority level?*

Answer:  The interrupt source with the highest natural order priority will take precedence. The natural order priority is determined by the Interrupt Vector Table (IVT) address for that source. Interrupt sources with a lower IVT address have a higher natural order priority.

Question 2:  *Can the DISI instruction be used to disable all sources of interrupt and traps?*

Answer:  The DISI instruction does not disable traps or priority level 7 interrupt sources. However, the DISI instruction can be used as a convenient way to disable all interrupt sources if no priority level 7 interrupt sources are enabled in the user’s application.
This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the dsPIC33F Product Family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Interrupts module include the following:

<table>
<thead>
<tr>
<th>Title</th>
<th>Application Note #</th>
</tr>
</thead>
<tbody>
<tr>
<td>No related application notes at this time.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Please visit the Microchip Web site (www.microchip.com) for additional Application Notes and code examples for the dsPIC33F Family of devices.
29.9 REVISION HISTORY

Revision A (February 2007)
This is the initial release of this document.

Revision B (February 2007)
Minor edits throughout document.

Revision C (August 2008)
This revision incorporates the following updates:

• Registers:
  - The bit description for bit 6 and bit 4 in the Interrupt Control Register 1 (INTCON1) has been corrected (see Register 29-3).

• Notes:
  - Added a note after the first paragraph in 29.1.5 “Interrupt Priority”, which provides information on changing the interrupt priority levels “on-the-fly”.

• Additional minor corrections such as language and formatting updates are incorporated throughout the document.