'280x Interrupt Hardware Priority Overview

On a '280x DSP with the PIE block enabled, the interrupts are prioritized in hardware by default as follows:

Global Priority (CPU Interrupt level):

CPU Interrupt:	Hardware Priority	
Reset	1	(Highest)
INT1	5	
INT2	6	
INT3	7	
INT4	8	
INT5	9	
INT6	10	
INT7	11	
,,,,	….	
INT12	16	
INT13	17	
INT14	18	
DLOGINT	19	(Lowest)
RTOSINT	20	
reserved	2	
NMI	3	
ILLEGAL	-	
USER1`	-	(Software Interrupts)
USER2	-	
…..		

CPU Interrupts INT1 - INT14, DLOGINT and RTOSINT are maskable interrupts. These interrupts can be enabled or disabled by the CPU Interrupt enable register (IER).

Group Priority (PIE Level):

If the Peripheral Interrupt Expansion (PIE) block is enabled, then CPU interrupts INT1 to INT12 are connected to the PIE. This peripheral expands each of these 12 CPU interrupt into 8 interrupts. Thus the total possible number of available interrupts in the PIE is 96. Note, not all of the 96 are used on a 280x device.

CPU Interrupt	PIE Group	PIE Interrupts							
		Highest ----------		Hardware Priority Within the Group			----------		Lowest
INT1	1	INT1.1	INT1.2	INT1.3	INT1.4	INT1.5	INT1.6	INT1.7	INT1.8
INT2	2	INT2.1	INT2.2	INT2.3	INT2.4	INT2.5	INT2.6	INT2.7	INT2.8
INT3	3	INT3.1	INT3.2	INT3.3	INT3.4	INT3.5	INT3.6	INT3.7	INT3.8
…. etc…									
…. etc…									
INT12	12	INT12.1	INT12.2	INT12.3	INT12.4	INT12.5	INT12.6	INT12.7	INT4.8

Each of the PIE groups has its own interrupt enable register (PIEIERx) to control which of the 8 interrupts (INTx.1 - INTx.8) are enabled and permitted to issue an interrupt.

‘280x Interrupt Priorities

The PIE block is organized such that the interrupts are in a logical order. Interrupts that typically require higher priority, are organized higher up in the table and will thus be serviced with a higher priority by default.

The interrupts in a 280x system can be categorized as follows (ordered highest to lowest priority):

1) Non-Periodic, Fast Response:

These are interrupts that can happen at any time and when they occur, they must be serviced as quickly as possible. Typically these interrupts monitor an external event.

On the 280x, such interrupts are allocated to the first few interrupts within PIE Group 1 and PIE Group 2. This position gives them the highest priority within the PIE group. In addition, Group 1 is multiplexed into the CPU interrupt INT1. CPU INT1 has the highest hardware priority. PIE Group 2 is multiplexed into the CPU INT2 which is the 2nd highest hardware priority.

2) Periodic, Fast Response:

These interrupts occur at a known period, and when they do occur, they must be serviced as quickly as possible to minimize latency. The A/D converter is one good example of this. The A/D sample must be processed with minimum latency.

On the 280x, such interrupts are allocated to the group 1 in the PIE table. Group 1 is multiplexed into the CPU INT1. CPU INT1 has the highest hardware priority.

3) Periodic:

These interrupts occur at a known period and must be serviced before the next interrupt. Some of the PWM interrupts are an example of this. Many of the registers are shadowed, so the user has the full period to update the register values.

In the 280x PIE module, such interrupts are mapped to group 2 - group 5. These groups are multiplexed into CPU INT3 to INT5 (the ePWM, eCAP and eQEP), which are the next lowest hardware priority.

4) Periodic, Buffered:

These interrupts occur at periodic events, but are buffered and hence the processor need only service such interrupts when the buffers are ready to filled/emptied. All of the serial ports (SCI / SPI / I2C / CAN) either have FIFO's or multiple mailboxes such that the CPU has plenty of time to respond to the events without fear of losing data.

In the 280x, such interrupts are mapped to INT6, INT8, and INT9, which are the next lowest hardware priority.

Software Prioritization of Interrupts - The DSP28 Example

The user will probably find that the PIE interrupts are organized where they should be for most applications. However, some software prioritization may still be required for some applications.

Recall that the basic software priority scheme on the C28x works as follows:

- **Global Priority:**
  This priority can be managed by manipulating the CPU IER register. This register controls the 16 maskable CPU interrupts (INT1 - INT16).

- **Group Priority:**
  This can be managed by manipulating the PIE block interrupt enable registers (PIEIERx). There is one PIEIERx per group and each control the 8-interrupts multiplexed within that group.

The DSP28 software prioritization of interrupt example demonstrates how to configure the Global priority (via IER) and group priority (via PIEIERx) within an ISR in order to change the interrupt service priority based on user assigned levels. The steps required to do this are:

1. **Set the global priority.**
   Modify the IER register to allow CPU interrupts with a higher user priority to be serviced.

2. **Set the Group priority**
   Modify the appropriate PIEIERx register to allow group interrupts with a higher user set priority to be serviced.

3. **Enable interrupts**

The DSP28 software prioritized interrupts example provides a method using mask values that are configured during compile time to allow you to manage this easily.

To setup software prioritization for the DSP28 example, the user must first assign the desired global priority levels and group priority levels.

This is done in the DSP280x_SWPrioritizedIsrLevels.h file as follows:

1. User assigns global priority levels:
   INT1PL - INT16PL

   These values are used to assign a priority level to each of the 16 interrupts controlled by the CPU IER register. A value of 1 is the highest priority while a value of 16 is the lowest. More then one interrupt can be assigned the same priority level. In this case the default hardware priority would determine which would be serviced first. A priority of 0 is used to indicate that the interrupt is not used.

2. User assigns PIE group priority levels:
   GxyPL (where x = PIE group number 1 - 12 and y = interrupt number 1 - 8)

   These values are used to assign a priority level to each of the 8 interrupts within a PIE group. A value of 1 is the highest priority while a value of 8 is the lowest. More then one interrupt can be assigned the same priority level. In this case the default hardware priority would determine which would be serviced first. A priority of 0 is used to indicate that the interrupt is not used.

Once the user has defined the global and group priority levels, the compiler will generate mask values that can be used to change the IER and PIEIERx registers within each ISR. In this manner the interrupt software prioritization will be changed. The masks that are generated at compile time are:

- IER mask values:
  MINT1 - MINT16

  The user assigned INT1PL - INT16PL values are used at compile time to calculate an IER mask for each CPU interrupt. This mask value will be used within an ISR to allow CPU interrupts with a higher priority to interrupt the current ISR and thus be serviced at a higher priority level.

- PIEIERxy mask values:
  MGxy (where x = PIE group number 1 - 12 and y = interrupt number 1 - 8)

  The assigned group priority levels (GxyPL) are used at compile time to calculate PIEIERx masks for each PIE group. This mask value will be used within an ISR to allow interrupts within the same group that have a higher assigned priority to interrupt the current ISR and thus be serviced at a higher priority level.

Using the IER/PIEIER Mask Values

Within an interrupt service routine, the global and group priority can be changed by software to allow other interrupts to be serviced. The procedure for setting an interrupt priority using the mask values created in the DSP28_SWPrioritizedIsrLevels.h is the following:

1. **Set the global priority.**
   - Modify IER to allow CPU interrupts from the same PIE group as the current ISR.
   - Modify IER to allow CPU interrupts with a higher user defined priority to be serviced.

2. **Set the group priority**
   - Save the current PIEIERx value to a temporary register.
   - The PIEIER register is then set to allow interrupts with a higher priority within a PIE group to be serviced.

4. **Enable interrupts**
   - Enable all PIE interrupt groups by writing all 1's to the PIEACK register
   - Enable global interrupts by clearing INTM

5. **Execute ISR**. Interrupts that were enabled in steps 1-3 (those with a higher software priority) will be allowed to interrupt the current ISR and thus be serviced first.

6. **Restore the PIEIERx register**

7. **Exit**

Example Code

The sample C code below shows an EV-A Comparator 1 Interrupt service routine software prioritization written in C. This interrupt is connected to PIE group 2 interrupt 1.

```
// Connected to PIEIER2_1 (use MINT2 and MG21 masks):
#if (G21PL != 0)
interrupt void EPWM1_TZINT_ISR(void)   // EPWM1 Trip Zone
{
    // Set interrupt priority:
    volatile Uint16 TempPIEIER = PieCtrlRegs.PIEIER2.all;
    IER |= M_INT2;
    IER &= MINT2;                      // Set "global" priority
    PieCtrlRegs.PIEIER2.all &= MG21;   // Set "group"  priority
    PieCtrlRegs.PIEACK.all = 0xFFFF;   // Enable PIE interrupts
    EINT;

    // Insert ISR Code here.......
    // for now just insert a delay
    for(i = 1; i <= 10; i++) {}

    // Restore registers saved:
    DINT;
    PieCtrlRegs.PIEIER2.all = TempPIEIER;

    //  Add ISR to Trace
    ISRTrace[ISRTraceIndex] = 0x0021;
    ISRTraceIndex++;
}
#endif


CMP1INT_ISR:
            ASP
            ADDB    SP,#1
            CLRC    OVM,PAGE0
            MOVW    DP,#0x0033
            MOV     AL,@36
            MOV     *-SP[1],AL
            OR      IER,#0x0002
            AND     IER,#0x0002
            AND     @36,#0x000E
            MOV     @33,#0xFFFF
            CLRC    INTM

             User code goes here….

            SETC    INTM
            MOV     AL,*-SP[1]
            MOV     @36,AL
            SUBB    SP,#1
            NASP
            IRET
```

The interrupt latency is approx 22 cycles.