405GP

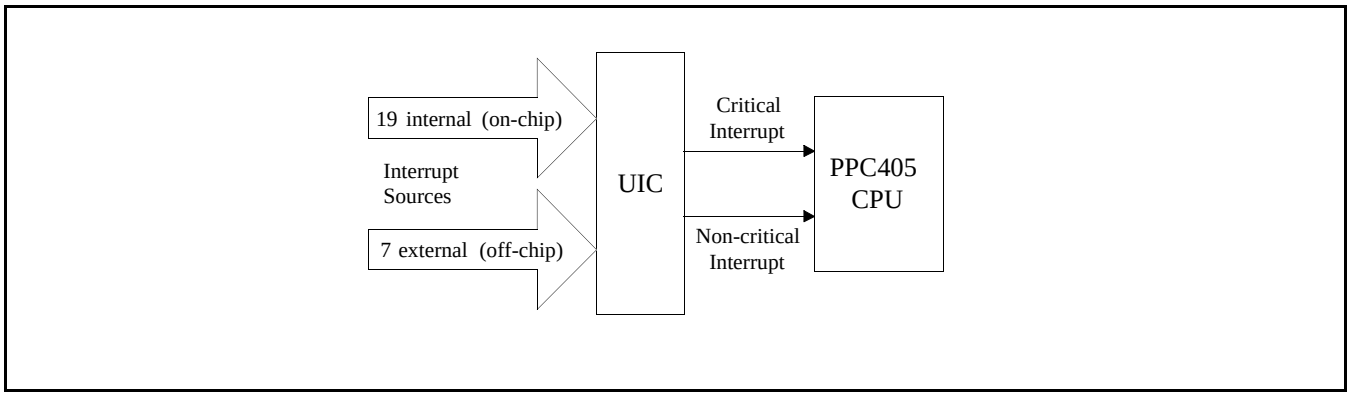
Preliminary Application Note

Using the Interrupt Controller in the PowerPC 405GP

INTRODUCTION

The Universal Interrupt Controller (UIC) in the PowerPC 405GP Embedded Processor handles interrupt signals from all sources external to the CPU. The UIC is an ASIC core that can provide status, configuration, masking and generation of interrupts for up to 32 inputs. The implementation in the PPC405GP provides inputs for 19 internal (on-chip) interrupt sources and 7 external (off-chip) IRQs. There are two outputs from the UIC, critical interrupt and non-critical interrupt. Both are connected to high-level-triggered interrupt inputs at the CPU as shown in Figure 1.

Figure 1. UIC Overview



The interrupt inputs to the UIC are programmable using a set of configuration registers.Each input can be programmed to be high/low-level or rising/falling-edge triggered, and to generate a critical or non-critical output. Status registers are provided to hold the current state of all interrupts and the masked status of all enabled interrupts. For testing and debugging, interrupts can be generated directly from software by setting bits in a status set register.

In order to reduce interrupt servicing latency for critical interrupts, an optional vector generation capability is provided. The resulting vectors can point directly to an interrupt service routine or to a table entry containing the address of the routine.

UIC INTERRUPT ASSIGNMENTS

Table 1 shows the interrupt assignments and configuration for the UIC's inputs. While the external IRQ interrupts are fully programmable, the triggering and polarity of the on-chip internal interrupts must be configured as shown in Table 1.

Table 1. Interrupt Assignments and Configuration

Interrupt	Triggering	Polarity	Source
0	Level	High	UART0
1	Level	High	UART1
2	Level	High	IIC
3	Edge	Rising	External Bus Master
4	Level	High	PCI External Command Write
5	Level	High	DMA Channel 0
6	Level	High	DMA Channel 1
7	Level	High	DMA Channel 2
8	Level	High	DMA Channel 3
9	Level	High	Ethernet Wake Up
10	Level	High	MAL System Error (SERR)
11	Level	High	MAL TX End of Buffer (TXEOB)
12	Level	High	MAL RX End of Buffer (RXEOB)
13	Level	High	MAL TX Descriptor Error (TXDE)
14	Level	High	MAL RX Descriptor Error (RXDE)
15	Level	High	Ethernet
16	Level	High	External PCI SERR
17	Level	High	ECC Correctable Error
18	Level	High	PCI Power Management
19	N/A	N/A	Reserved
20	N/A	N/A	Reserved
21	N/A	N/A	Reserved
22	N/A	N/A	Reserved
23	N/A	N/A	Reserved
24	N/A	N/A	Reserved
25	Any	Any	External IRQ 0
26	Any	Any	External IRQ 1
27	Any	Any	External IRQ 2
28	Any	Any	External IRQ 3
29	Any	Any	External IRQ 4
30	Any	Any	External IRQ 5
31	Any	Any	External IRQ 6

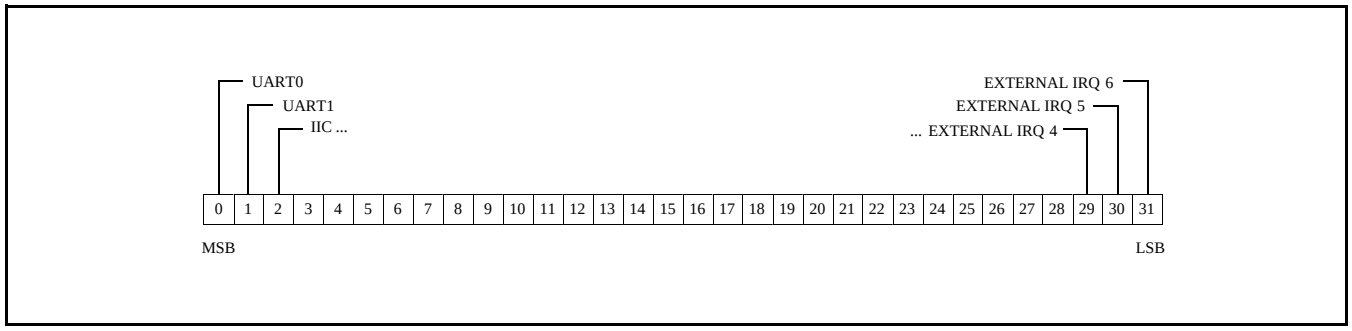
UIC REGISTERS

Table 2 lists the Device Control Registers (DCRs) in the UIC. These registers can only be accessed using the **mfdcr** and **mtdcr** instructions while the processor is in privileged mode (also referred to as the supervisor state). With the exception of the two vector registers, bit assignments for the UIC DCRs correspond to the interrupt assignments in Table 2. These bit assignments are illustrated in Figure 2.

Table 2. UIC Device Control Registers

Mnemonic	Register Name	Address	Access
UIC0_SR	UIC Status Register	0x0C0	Read/Clear
UIC0_SRS	UIC Status Register Set	0x0C1	Read/Set
UIC0_ER	UIC Enable Register	0x0C2	Read/Write
UIC0_CR	UIC Critical Register	0x0C3	Read/Write
UIC0_PR	UIC Polarity Register	0x0C4	Read/Write
UIC0_TR	UIC Trigger Register	0x0C5	Read/Write
UIC0_MSR	UIC Masked Status Register	0x0C6	Read Only
UIC0_VR	UIC Vector Register	0x0C7	Read Only
UIC0_VCR	UIC Vector Configuration Register	0x0C8	Write Only

Figure 2. UIC DCR Bit Assignments



INTERRUPT STATUS

All internal and external interrupts are captured and held in the Status Register (UIC0_SR) until they are intentionally reset. An interrupt has occurred if a “1” is read from the assigned bit field in the register. Interrupts are cleared by writing a “1” to every bit that is to be reset, and bits that are written with a “0” are unaffected. If an attempt is made to reset a level sensitive interrupt while the incoming interrupt signal is still asserted, the status bit will not be cleared. In this case, the interrupt input must be deasserted by the source before the status bit can be successfully reset.

The read-only Masked Status Register (UIC0_MSR) contains the result of masking the Status Register with the Enable Register. Reading status from this register eliminates the need for software to read and apply the enable mask in order to determine which interrupts are both active and enabled.

ENABLING INTERRUPTS

When an interrupt occurs, it will always be indicated in the Status Register. However, to have any effect on the UIC's outputs to the processor, the interrupt must be also be enabled. Individual interrupts are enabled by setting the assigned bit fields in the Enable Register (UIC0_ER) to “1” and disabled by clearing them to “0”.

In some cases on-chip interrupts must also be enabled in the control registers of the sourcing subsystem. For example, to enable a terminal count interrupt from the DMA Controller, several bits would need to be set in one of the DMA Channel Control Registers.

The external IRQs use pins that are shared with GPIO[17:23]. The function of these pins is controlled by bits 12:18 in the Chip Control Register (CPC0_CR0). Clearing the assigned bit in this register will disable the GPIO channel and enable the pin as an interrupt input.

At the system level, the critical and non-critical interrupt inputs to the processor core are enabled by setting the Critical Interrupt Enable (CE) and External Interrupt Enable (EE) bits in the Machine State Register (MSR). It should be noted that, in this context, the non-critical interrupts are sometimes referred to as “external” interrupts meaning external to the CPU rather than external to the chip. The MSR can only be accessed using the privileged **mfmsr** and **mtmsr** instructions.The EE bit by itself can be set or cleared using the **wrtee** or **wrteei** instructions which are also privileged.

CONFIGURING INTERRUPTS

The UIC's interrupt inputs are programmed for level or edge sensitivity using the Trigger Register(UIC0_TR). A “1” in the appropriate bit location will cause an interrupt to be triggered by a rising or falling edge. A “0” will cause it to be triggered by a change in logic level. A “1” in the same bit location in the Polarity Register (UIC0_PR) will cause the interrupt to be generated in response to a high level or a rising edge. A “0” will cause it to be generated by a low level or a falling edge. Note that the triggering and polarity of the external IRQ's are programmable while the on-chip internal interrupts are predefined and must be configured as shown in Table 1.

All interrupts can be programmed to cause critical or non-critical outputs from the UIC to the processor. If the assigned bit is set to “1” in the Critical Register (UIC0_CR), an enabled interrupt captured in the Status Register will generate a critical interrupt signal to the processor. A “0” in the same bit location will cause a non-critical interrupt signal to be generated.

Table 3. Interrupt Configuration Summary

Mnemonic	Register Name	Programming	
UIC0_ER	UIC Enable Register	1 = enabled	0 = disabled
UIC0_TR	UIC Trigger Register	1 = edge	0 = level
UIC0_PR	UIC Polarity Register	1 = rising / high	0 = falling / low
UIC0_CR	UIC Critical Register	1 = critical	0 = non-critical

GENERATING INTERRUPTS FOR TESTING AND DEBUGGING

When needed for testing and debugging, interrupts can be generated directly from software by setting bits in the Status Register Set location (UIC0_SRS). Writing a “1” to any bit position will set the corresponding bit in the Status Register and cause an interrupt to be generated if it is enabled. Locations written with a “0” are unaffected. Reading this register returns the contents of the Status Register.

OPTIONAL CRITICAL INTERRUPT VECTORS

The Vector Register (UIC0_VR) provides an optional way to reduce the interrupt service latency for critical interrupts. This register contains an automatically generated 32 bit address that is the sum of a programmed base address and an offset determined from the relative bit assignment of the highest priority critical interrupt that is enabled and active. The offset is computed by finding the difference between the bit position of the highest priority interrupt (programmable to be either 0 or 31) and the bit position of the active interrupt and then multiplying that difference by 512. A general interrupt service routine can use the resulting vector to jump directly to a service routine for a particular interrupt as long as the entire routine can be contained in 512 words orless.Alternatively, the vector can point to a table entry containing the address of a particular service routine. Interrupt vectors are not generated for non-critical interrupts The vector base address is programmed into the highest order 30 bits of the Vector Configuration Register (UIC0_VCR). Because the two least significant bits of this address are assumed to be “00”, the base address will always be on a full word boundary. The least significant bit in this register determines which end of the Status Register is the highest priority interrupt bit. When this field is set to “1” UIC0_SR[0] is the highest priority interrupt bit. When it is set to “0”,UIC0_SR[31] is the highest priority interrupt bit.

WHEN AN INTERRUPT OCCURS

Interrupts from the UIC are presented to the processor as either a critical or non-critical interrupt input. Critical interrupts are recognized by the processor only if they are enabled by MSR[CE]. Similarly non-critical interrupts are recognized only if they are enabled by MSR[EE]

When the processor takes a **critical interrupt**, it writes the address of the next instruction to Save/Restore Register 2 (SRR2) and saves the contents of the MSR in Save/Restore Register 3(SRR3). MSR[CE] is reset to “0” in order to prevent another critical interrupt from occurring before SRR2 and SRR3 are saved by the interrupt handling routine. All other fields in the MSR are also reset to “0” except for Machine Check Enable (ME) which is left unchanged. The high order 16 bits of the program counter are then written with the user-programmed contents of the Exception Vector Prefix Register (EVPR) and the low order 16 bits are written with the Critical Interrupt Vector offset, **0x100**.

The processor's response to a **non-critical interrupt** from the UIC is very similar. First, it writes the address of the next instruction to Save/Restore Register 0 (SRR0) and saves the contents of the MSR in Save/Restore Register 1 (SRR1). MSR[EE] is reset to “0” in order to prevent another non-critical interrupt from occurring before SRR0and SRR1 are saved by the interrupt handling routine.All other fields in the MSR are also reset to “0” except for Machine CheckEnable (ME) which is left unchanged. The high order 16 bits of the program counter are then written with the user-programmed contents of the Exception Vector Prefix Register (EVPR) and the low order 16 bits are written with the Non-Critical Interrupt Vector offset, **0x500**.

Inside the interrupt handling routine, SRR2/SRR3 or SRR0/SRR1 should be saved as soon as possible. If desired, interrupts can then be re-enabled by setting MSR[CE] or MSR[EE]. Software must also save any other registers whose contents will be needed to restore the original state of the system.

After servicing the interrupt, the program counter and MSR are restored by executing either a Return From Critical Interrupt (**rfci**) or Return From Interrupt (**rfi**) instruction. Execution then resumes at the address in the program counter.

AN EXAMPLE: DMA TRANSFER WITH AN END OF TRANSFER INTERRUPT

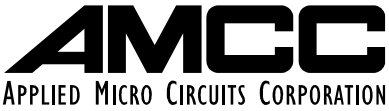
As an example, consider a non-critical interrupt that occurs when DMA Channel 2 reaches terminal count at the end of a DMA transfer.

The following steps would be taken to enable and configure this interrupt:

- Set DMA0_CR2[ETD] and DMA0_CR2[TCE] to enable terminal count.
- Set DMA0_CR2[CIE] to enable the interrupt at the DMA Controller.
- Clear UIC0_TR[D2IT] to make the interrupt input level sensitive.
- Set UIC0_PR[D2IP] to make the interrupt trigger on a high level.
- Clear UIC0_CR[D2IC] to cause the interrupt to be non-critical.
- Set UIC0_ER[D2IE] to enable the interrupt at the UIC.
- Set MSR[EE] to cause the CPU to accept non-critical interrupt inputs.

When the interrupt actually occurs, software would take the following steps:

- Save the current machine state.
- Set MSR[EE] to re-enable non-critical interrupts (if desired).
- Read UIC0_MSR to determine the source of the current interrupt.
- Read DMA0_SR[CS2] to determine that terminal count has been reached.
- Execute the rest of the interrupt service routine.
- Clear DMA0_SR[CS2] to reset terminal count at the DMA controller.
- Set UIC0_SR[D2IS] to clear the interrupt at the UIC.
- Restore the machine state and execute an **rfi** instruction.



Applied Micro Circuits Corporation
6290 Sequence Dr., San Diego, CA 92121
Phone: (858) 450-9333 — (800) 755-2622 — Fax: (858) 450-9885
http://www.amcc.com