The Open Programmable Interrupt Controller (PIC) Register Interface Specification Revision 1.2

Issue Date: October 1995

Issued Jointly by
Advanced Micro Devices and Cyrix Corporation

TERMS AND CONDITIONS
THE OPEN PIC SPECIFICATION

THESE TERMS AND CONDITIONS (these "Terms and Conditions") are extended individually by: (i) Advanced Micro Devices, Inc. ("AMD"); and (ii) Cyrix Corporation ("Cyrix") (collectively, the "Parties"), to any individual or entity ("Recipient") wishing to implement the Open PIC Specification, Version 1.2, (the "Specification").

1. Background and Purpose. The Parties have jointly developed the Specification for the implementation of an interrupt controller used in personal computers. The Parties desire to facilitate industry acceptance of the Specification by creating an open standard and making available certain patented technology necessary for a commercially viable implementation.

2. Definitions. The following terms, as used herein, shall have the following meanings:

2.1. "Affiliate" shall mean any corporation, partnership or other entity (an "Entity") that, directly or indirectly controls, is under common control with, or is controlled by, any Entity; provided that such Entity shall be deemed to be an Affiliate only for so long as such control exists;

2.2. "Implementing Product(s)" shall mean those products (including parts or elements thereof) that: (i) implement the Specification; (ii) are directly related to the Specification; and (iii) are manufactured and sold for use in the field of programmable interrupt controllers solely in connection with devices incorporating industry-standard personal computer architectures (e.g., x86, PowerPC) and extensions thereof; provided, however, that any parts or elements of products manufactured and sold for use in such field that are either not required in order to implement, or are not directly related to, such Specification shall be excluded;

2.3. "Necessary Patent Claim(s)" shall mean the claim(s) of any issued patent, patent application, invention certificate or registration in the Untied States or any other country throughout the world entitled to an effective filing date or priority date on or before one (1) year after the release date of the Specification that: (i) relates to the specific subject matter of the Specification; and (ii) is necessary in order to implement the Technical Design of the Specification. A Necessary Patent Claim shall be deemed to exist only if, and at such time as, a patent actually issues with respect to the relevant applications;

2.4. "Technical Design" shall mean the engineering design as reflected in the schematics, diagrams and/or narrative descriptions of Section 2 through Section 5 of the Specification; and

2.5. "Use" or "Using" as the context requires, shall mean employment or exploitation of any kind (including testing and integration) in connection with making, having made, using or selling Implementing Products.

3. Grant of Immunity.

3.1. Immunity from Suit. Subject to the limitations contained in these Terms and Conditions, the Parties and their Affiliates hereby agree to grant to Recipient a limited immunity from suit for infringement of any Necessary Patent Claim(s) that they or an Affiliate of the Party control, solely with respect to Recipient making, Using, selling or importing, or having made, Used, sold or imported for Recipient, Implementing Products (the "Immunity").

3.2. Limitations.

(a) Enforcement of Necessary Patent Claim. The Immunity shall not apply with respect to Recipient if Recipient enforces or attempts to enforce, or Recipient's Affiliate enforces or attempts to enforce, any Necessary Patent Claim(s) against any person entitled to the Immunity in connection with such person's making, Using, selling or importing any Implementing Products or having Implementing Products made, Used, sold or imported for such person under the Specification.

(b) Combination of Implementing Products. Under no circumstances shall the Immunity be construed as granting an Immunity with respect to Recipient or its Affiliate making, Using, selling or importing, or having made, Used, sold or imported for it combinations of Implementing Products with any other product (including parts or elements thereof) that is not an Implementing Product.

3.3. Termination of Grant of Immunity. The Immunity shall terminate when there are no Necessary Patent Claims issued or otherwise pending in an applicable jurisdiction.

3.4. No Obligation to Enforce. Nothing contained in these Terms and Conditions shall obligate the parties hereto to enforce Necessary Patent Claims against any person, including any person that has enforced its own Necessary Patent Claims against a person entitled to the Immunity in connection with the making, Using, selling or importing of Implementing Products.

3.5. No Further Grant. Except as expressly set forth herein, no licenses or covenants are granted by implication, estoppel, patent exhaustion or otherwise with respect to any other intellectual property or for any other reason.

4. Copyright License. For any copyrighted or copyrightable information that is included in the Technical Design of the Specification, the Parties and their Affiliates grant to Recipient solely in connection with making, Using, selling or importing Implementing Products a nonexclusive, non-royalty bearing, perpetual, worldwide copyright license with respect to such information to the full extent permitted under 17 U.S.C. § 106.

5. Distribution of Products. Recipient shall be solely responsible for making, Using, selling or importing Implementing Products, in whole or in part. Recipient shall not make any representation or warranty, or assume or create any obligation, whether express or implied, on behalf of the Parties, and shall take all reasonable steps to assure the same.

6. Disclaimer and Limitation of Liability.

6.1. Disclaimer. THE PARTIES ARE PROVIDING THE SPECIFICATION "AS IS," AND MAKE NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, RELATING TO THE SPECIFICATION. WITHOUT LIMITING THE FOREGOING, THE PARTIES EXPRESSLY DISCLAIM THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND ANY EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION THAT MIGHT ARISE FROM THE CONTENTS OF THE SPECIFICATION OR RECIPIENT'S IMPLEMENTATION OR ATTEMPTED IMPLEMENTATION OF THE SPECIFICATION. THE PARTIES FURTHER DISCLAIM ANY WARRANTY THAT THE CONTENTS OF THE SPECIFICATION, OR ANY PRODUCT IMPLEMENTING THE SPECIFICATION, IN WHOLE OR IN PART, WILL BE FREE FROM INFRINGEMENT OF ANY PERSON'S INTELLECTUAL PROPERTY.

6.2. Limitation of Liability. IN NO EVENT SHALL THE PARTIES BE LIABLE FOR ANY INCIDENTAL, CONSEQUENTIAL, SPECIAL OR PUNITIVE DAMAGES ARISING OUT OF A CAUSE OF ACTION BASED ON THESE TERMS AND CONDITIONS, OR BASED ON MAKING, USING, SELLING OR IMPORTING PRODUCTS THAT IMPLEMENT THE SPECIFICATION. NOTHING CONTAINED HEREIN SHALL BE CONSTRUED AS AN INDEMNIFICATION OF ANY KIND BY THE PARTIES WITH REGARD TO THE CONTENT OR IMPLEMENTATION OF THE SPECIFICATION.

6.3. No Joint Liability. ANY LIABILITY ARISING OUT OF A BREACH OF THIS AGREEMENT INCLUDING BUT NOT LIMITED TO ANY IMPROPER ASSERTION OF A NECESSARY PATENT CLAIM AGAINST ANY PERSON WITH RESPECT TO AN IMPLEMENTING PRODUCT SHALL BE THE SOLE RESPONSIBILITY OF THE BREACHING PARTY. IN ALL EVENTS, LIABILITY WITH RESPECT TO ACTIVITIES UNDER THIS AGREEMENT SHALL BE SEVERAL AND NOT JOINT.

7. Modifications. The Parties retain the right to modify the Specification at any time without notice.

8. Miscellaneous Provisions.

8.1. Governing Law; Jurisdiction. These Terms and Conditions shall be governed by and interpreted in accordance with the laws of the State of California, excluding that body of law related to choice of laws. Any disputeinvolving any Party and any Recipient arising out of, related to or including in any manner these Terms and Conditions shall be resolved in State or Federal Court in the State of California.

8.2. Modification. No modification of any provision of these Terms and Conditions shall be effective unless consented to in writing by the Parties.

8.3. Integration. These Terms and Conditions constitute the entire agreement among the parties hereto and supersede all prior or contemporaneous agreements, proposals, understandings, and communications, whether oral or written, with respect to the subject matter hereof.

The Open Programmable Interrupt Controller (PIC) Register Interface Specification Revision 1.2

Note: *Use of this specification is subject to the terms and conditions provided on page 2 of this document.*

1 SCOPE

This document is a register-level, open specification of a programmable interrupt controller architecture. It describes the intended function of a “generic” PIC that supports this architecture. The primary focus of this document is to specify the register level software interface. The secondary purpose is to describe the critical functional features required for proper operation of a controller compliant with this open PIC specification. This is not a device-specific functional specification.

This specification is designed as a baseline, with specific provisions for expansion and addition of features. Individual implementations can go beyond the baseline feature set and capabilities in order to add value to any particular vendor’s silicon or systems.The initial release of this open specification defines a programmable interrupt controller architecture as it applies to integrated motherboard implementations for the interrupt controller. A future revision will address issues related to full register relocation capability as a PCI-compliant device.

1.1 Goals of the Specification

- To provide a complete, open specification of a PIC register level interface
- To specify the functionality required in the interrupt controller to support operation

2 FUNCTIONAL OVERVIEW

2.1 Feature Highlights

- Flexible number of processors supported up to a maximum of 32
- Flexible number of interrupt sources supported up to a maximum of 2048
- Supports the connection of an external 8259 pair for ISA/AT compatibility
- Multicast capable interprocessor interrupts
- Processor initialization control
- Basic interrupt delivery method: directed
- Advanced delivery mode for distributed interrupts
- Four Global high resolution timers
- Feature reporting mechanism includes number of processors and interrupt sources supported
- Flexible feature expansion
- All registers are 32 bits and are aligned on 128 bit boundaries for 32/64/128 bit bus operation
- Register blocks aligned on 4K boundaries to allow the mapping of registers to distinct memory pages

2.2 Implementation Alternatives

2.2.1 Number of Processors Supported

A given implementation can support any number of processors, up to a total of 32. A bit field in the feature reporting register reflects the number of CPUs supported by a specific interrupt controller implementation.

2.2.2 Number of Interrupt Sources Supported

A given implementation may support any number of interrupt inputs up to a maximum of 2048. A bit field in the feature reporting register reflects the number of interrupt inputs that are supported by a specific interrupt controller implementation.

2.2.3 Distributed Delivery Method

For interrupt sources that are configured to be delivered to multiple sources, the interrupt controller is required to provide a distributed delivery mode. This mode must guarantee “exactly once” delivery for each interrupt event and issue events using a fair distribution algorithm. Extension areas are available in the feature reporting registers in order to allow vendors to add more advanced distribution methods as proprietary features. The details of the distribution algorithm will be implementation dependent.

2.2.4 Global Timer Implementation

The open PIC specification requires four programmable timer channels that can generate interrupts to one or more processors. All timers must be clocked at the same implementation dependent frequency as reported in the timer frequency reporting register. Although this specification does not require a specific timer frequency, the clock source must have the following properties:

- Constant, fixed frequency (no stop clock or slow down)
- Minimum frequency of 4 MHz
- Integral number of ticks per second

2.2.5 Reserved Bit Implementation

To ensure compatibility with future versions of this architecture, all unused register bits should be implemented to return zero when read and to be ‘don’t cares’ when written.

2.2.6 Readability of Registers

Unless otherwise specified, all registers must be readable and must return the last value written. Exceptions include the IPI dispatch command ports, the interrupt source ACT bit, and the interrupt acknowledge register. The interrupt acknowledge register is also the only register allowed to exhibit any read side-effects (note that this register is not required in x86-based implementations of the open PIC specification).

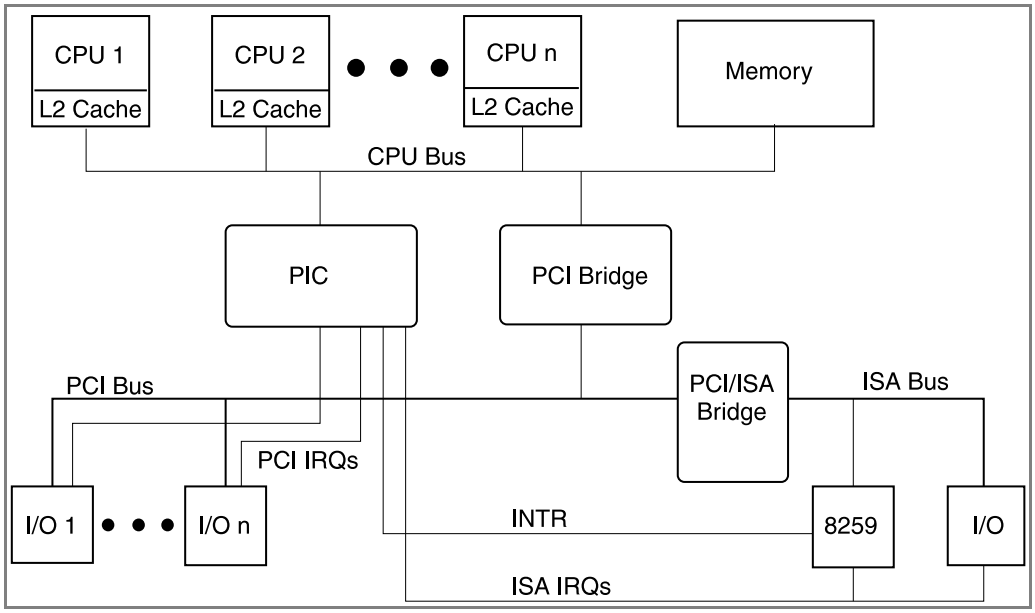
2.2.7 SLiC Backwards Compatibility

Where possible, the PIC software interface in this open PIC specification has been designed to avoid conflicts with the previously defined SLiC dual processor interrupt controller architecture. This approach includes shadowing registers which are common with SLiC (IPI 0 vector and dispatch registers) and avoiding address ranges which would conflict with SLiC unique register locations. The goal is to provide advanced functionality while making the software interface as compatible as possible with the SLiC definition.

2.3 Example System Architectures

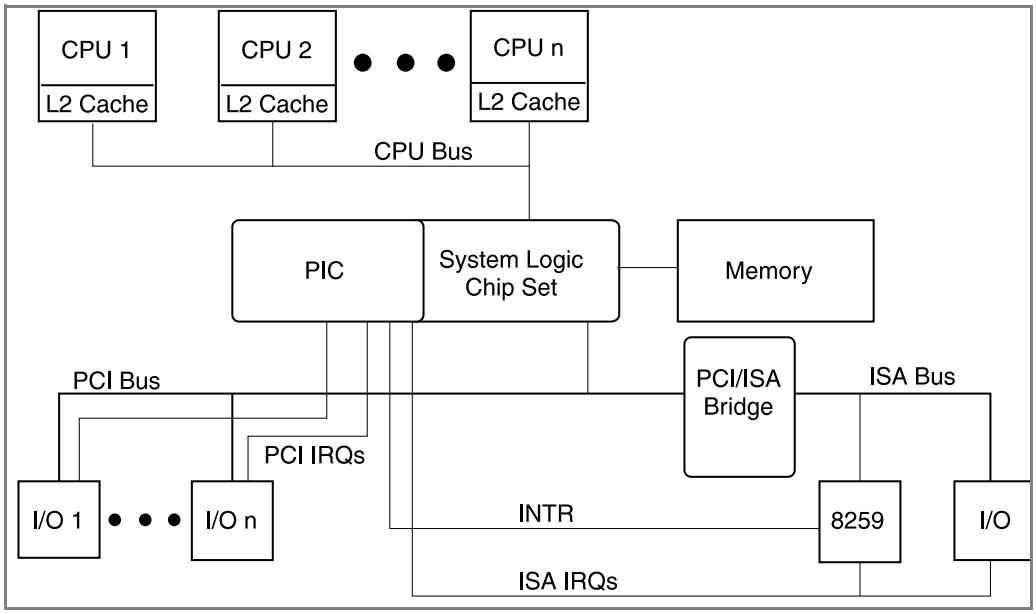
2.3.1 Stand-Alone Controller Example

In this example PCI-based system, a centralized interrupt controller that supports the open PIC specification is implemented as a stand-alone device. It resides on the CPU/ Memory Bus (or Multiprocessing Bus). All interrupt sources are connected to the PIC. The ISA Bus 8259 interrupt controller pair output is also connected to the PIC to enable the pass-through mode for start-up AT compatibility.



2.3.2 Integrated Controller Example

In this example system, a centralized interrupt controller that supports the open PIC specification is integrated into the system logic chipset. As part of the system logic, it has access (internal connection capability) to both the CPU/Memory bus and the PCI bus. All interrupt sources are connected to the PIC. The ISA Bus 8259 interrupt controller pair output is also connected to the PIC to enable the pass through mode for start-up AT compatibility.



3 FEATURE REQUIREMENTS

3.1 Interrupt Handling

3.1.1 I/O Interrupts

The open PIC specification allows as many as 2048 I/O interrupt sources. Each source must be individually programmable to be positive-edge or active-low-level triggered. The vector and priority for each source must also be individually programmable. Each interrupt source will have the option of being directed to a single processor or distributed among a group of processors using the distributed delivery method.

3.1.2 Interprocessor Interrupts/Self Interrupts

The open PIC specification requires a mechanism to allow processors to interrupt each other. Each processor will have four incoming IPI channels that function essentially as edge-triggered interrupt sources. Any processor will have the ability to trigger one of these sources by writing to one of the four IPI command ports. For each command port bit which is written with a one, an IPI event will be generated on the corresponding processor using the vector and priority for the selected channel. Note that a processor will be able to trigger an IPI event on one of its own incoming channels by writing a one to the command port bit corresponding to itself (this is also known as a self interrupt).

Although each processor will appear to have a private copy of the four command ports, the IPI channels will actually function as a shared resource. Multiple IPI events generated on a selected processor's incoming channel will not be queued. The vector and priority for each of the IPI channels will be the same for all processors and be programmable via a set of global registers. There are special restrictions for IPI priority assignments which are outlined in section 5.2.4.

3.2 Time-Slice Timers

The open PIC specification requires four timers that may be programmed to generate repetitive interrupts at fixed periods of time. The timers will provide a shared resource that can generate interrupts to one or more processors using the directed (single destination or multicast) interrupt delivery mode. The interrupt period, vector, and priority must be programmable for each timer. The timers must have a minimum resolution of 0.25 micro-seconds. Each counter must have a sufficient range to provide an interrupt interval of at least 10 milliseconds. All timers must be clocked at the same frequency and are required to run continuously at a constant rate. The timer frequency must be a whole number multiple of 1 Hz so that timers may be used for accurate timekeeping. The actual frequency of the timers will be reported in the timer frequency reporting register.

Timer interrupts will function essentially as edge-triggered events. If the timer period expires while a previous interrupt from the same timer is pending or in service, the subsequent interrupt will be lost. Special restrictions for timer priority assignments are outlined in section 5.2.4.

3.3 Interrupt Delivery Modes and Priority Levels

3.3.1 Delivery Modes

The open PIC specification requires delivery of interrupt events to processors based upon the destination and priority values for each source and the current priority level for each processor. There are two required modes of delivery for interrupt events: directed and distributed.

3.3.1.1 Directed Mode

For directed mode, the interrupt will be delivered to a specific processor (single destination) or to multiple processors (multicast) as specified by the destination field value. Directed mode is the only required mode for IPI and timer interrupt events. For I/O interrupts, directed mode will be used whenever a single destination is specified. Multicast delivery will not be supported for I/O interrupts.

3.3.1.2 Distributed Mode

For distributed mode, interrupt events from a particular source will be distributed among a group of processors specified by the destination field value. Distributed mode is required only for I/O interrupt sources and will be used whenever multiple destinations are selected for these sources. The events will be distributed using a fair, implementation-specific algorithm.

3.3.1.3 Exactly Once Delivery

This open PIC specification specifies “exactly once” delivery for all distributed mode interrupt events. This means that an interrupt event will never be pending or active on more than one processor at a time and will be delivered only once to the selected destination processor. Also, once an interrupt is dispatched[1] to a particular destination, further events from that source will not be dispatched to any other processor until processing of the original event is terminated by an EOI. Table 1 summarizes the types of supported interrupt sources and required delivery modes for each type. It is possible to miss edge triggered interrupt events that arrive while a previous interrupt from the same source is being serviced.

Table 1 PIC Interrupt Sources and Delivery Modes for the Open PIC Specification

Interrupt Source	Delivery Modes Supported
I/O Device	Directed - Single Destination Only Distributed - Used when multiple destinations are selected
Interprocessor interrupts	Directed - Single destination and multicast
Global Timer Interrupts	Directed - Single destination and multicast

3.3.2 Interrupt Source Priority

Each interrupt source will have a programmable priority value in the range from 0 to 15. This value will be used to indicate the priority of this source with respect to the other interrupt sources and will be used in scheduling delivery of interrupt events for that source. An interrupt is considered “delivered” when the CPU executes an interrupt acknowledge cycle and receives the vector for the source. In order for delivery to take place the priority of the source must be greater than the current task priority of the destination processor. Therefore, setting a source priority to zero will prohibit delivery of interrupts from that source.

3.3.3 Processor Current Task Priority

Each processor will have a private current task priority register set that may be set by the system software to indicate the relative importance of the task running on that processor. The value written to this register will be used to control which interrupts will be dispatched to the corresponding processor. The processor must not receive interrupts with a priority level equal to or lower than its current task priority. Therefore, setting the current task priority to 15 will prohibit the delivery of all interrupts to a processor.

1. For the purposes of this specification, “dispatched” indicates that the destination for an interrupt event has been determined and the event is assigned to that processor for delivery. This does not imply that the processor has necessarily received the vector for this interrupt.

3.3.4 Nesting of Interrupt Events

The open PIC specification requires a fully nested interrupt delivery mechanism. A processor must never have an in-service interrupt preempted by an equal or lower priority source. An interrupt is considered to be in service from the time its vector is returned during an interrupt acknowledge cycle until an EOI is received for that interrupt. The open PIC specification currently allows only a non-specific EOI that indicates the end of processing for the highest priority in-service interrupt. Each CPU will have a private EOI register that will control only the in-service state of interrupts delivered to that CPU.

3.3.5 Spurious Vector Generation

Under certain circumstances, the PIC may not have a valid vector to return to the processor during an interrupt acknowledge cycle. In these cases the controller will return the spurious vector that will be a programmable value. Following are the cases that will cause the spurious vector to be returned:

- Interrupt request is asserted in response to a level triggered interrupt source which is de-asserted before interrupt acknowledge.
- Interrupt request is asserted for an interrupt source which is masked by an increase in current task priority level for the interrupted processor before interrupt acknowledge.
- Interrupt request is asserted for an interrupt source which is masked using the mask bit in the source configuration register before interrupt acknowledge.

In all cases the spurious vector will not be returned if there is another pending interrupt that has sufficient priority to interrupt that processor. If such an interrupt is available, the vector for that source will be returned. For correct operation the EOI register should not be written in response to the spurious vector.

3.3.6 Processor Initialization

This open PIC specification requires a common processor initialization register that may be used by any processor to assert the initialization input of one or more other processors. This feature will provide a way to start up multiple processors and to provide some diagnostic capabilities.

3.3.7 Processor Identification

The PIC includes a *Who Am I* register to be used for individual processor identification. This register must be located at the same physical address for each processor but will return a unique value based upon which processor is performing the read. This register is intended to be used to identify the physical connection of the processors to the PIC so that the proper destination bit fields can be constructed for I/O and interprocessor interrupts. Most multiprocessing operating systems assign each processor some type of ID for use in interprocessor communications. The *Who Am I* value will not necessarily be the same as this logical processor ID.

3.3.8 ISA/AT 8259 Compatibility Modes

The open PIC specification specifies a mechanism to provide PC-AT power-on compatibility for chipsets using the 8259 interrupt controller architecture. After power-on RESET, the open PIC interrupt controller will default to 8259 pass-through mode. In this mode, the 8259 interrupt request output will be passed directly through the PIC to a single processor's interrupt request input line and the PIC will be essentially disabled. During SMP operation, pass-through mode will be disabled, and the PIC will distribute all system interrupt events.

4 REGISTER INTERFACE REQUIREMENTS

Note: *The paragraphs in the section 4 use a specific format to describe the register structure. Each section describes a register or set of registers. The subsection title defines the register or register set. This is followed by a register description in the following format:*

xFFFFFFFh is Register Name (Read/Write Capability)

where:

x = the configurable first four digits of the Register Memory Address in hexadecimal
FFFFFFFh = the fixed portion of the address in hexadecimal
Register Name = the Register Name specified in this document
(Read/Write Capability) = Read Only, Write Only, or Read/Write

The register list is followed by a pictorial description of the register layout and a verbal description of the register or register set function.

4.1 Per CPU Registers

A copy of each of these registers will be available to each CPU at the same physical address.

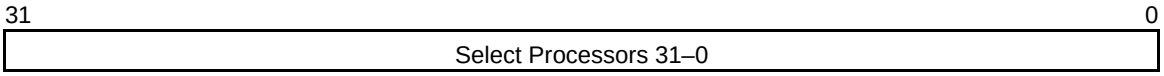
4.1.1 Interprocessor Interrupt Command Ports

xFC00040h is IPI 0 Dispatch Command Port (Write Only—shadowed at xFC00000)

xFC00050h is IPI 1 Dispatch Command Port (Write Only)

xFC00060h is IPI 2 Dispatch Command Port (Write Only)

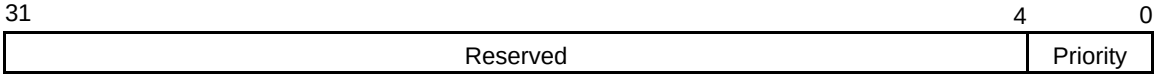
xFC00070h is IPI 3 Dispatch Command Port (Write Only)



Writing to this port will cause an IPI to be sent to one or more processors. The data written to this register will be used as a bit vector to indicate which processors should receive this IPI. This destination may be any single processor or group of processors. The vector and priority used for this IPI must be programmed in the corresponding Vector/Priority Register in the global register space. The vector and priority values for each channel will be the same for all processors and are assumed to remain static after initialization. This is specified as a write-only command port.

4.1.2 Current Task Priority Register

xFC00080h is Current Task Priority Register (Read/Write)

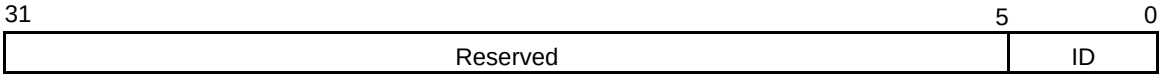


Priority: Current priority level (0–15). Power up default will be priority 15 (all Interrupts masked). Bits 31–4 are reserved. Values written to these bits will be ignored.

This register will be used to set the current task priority for each CPU. The task priority will indicate the relative importance of the currently executing task. Priority levels from 0 to 15 shall be supported. In order for an interrupt to be delivered to a CPU the interrupt priority must be higher than the current task priority. The operating system may use this mechanism to mask lower priority events from interrupting higher priority functions. Setting the priority level to 15 will mask all interrupts because no source can have a priority greater than 15.

4.1.3 Who Am I Register

xFC00090h is Who Am I Register (Read Only)



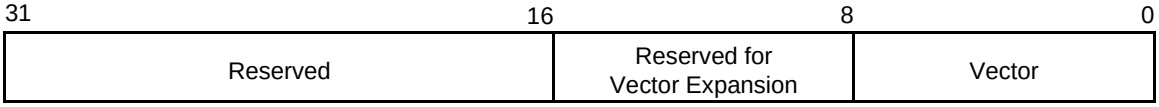
ID: Returns the ID of the processor reading this register. This value will be in the range 0–31 and will indicate which bit in the destination registers corresponds to the processor performing the read. Bits 31–5 are reserved. Values written to these bits will be ignored.

Note: *This processor ID is only intended to indicate the physical connection to the interrupt controller and may not directly correspond to any logical processor numbering provided by the operating system.*

The Who Am I register will provide a mechanism for each processor to determine its physical connection to the PIC. This value may be used to determine the value for the destination masks used for dispatching interrupts.

4.1.4 Interrupt Acknowledge Register

xFC000A0h is Interrupt Acknowledge Register (Read Only)



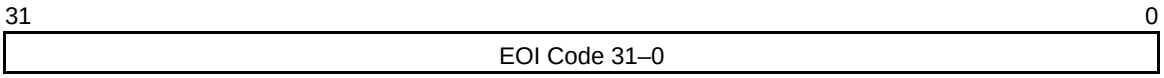
Vector: Returns the vector of the highest priority interrupt pending for this CPU.
Bits 31–8 are reserved.

This register will be required for system implementations that do not support an interrupt acknowledge cycle type. Reading this register will have the same effect as an interrupt acknowledge cycle.

Note: *This register is not required in x86 implementations.*

4.1.5 End of Interrupt (EOI) Register

xFC000B0h is EOI Register (Read/Write)



EOI Code: Must be written as zero for non-specific EOI. Values other than zero are reserved for future enhancements.

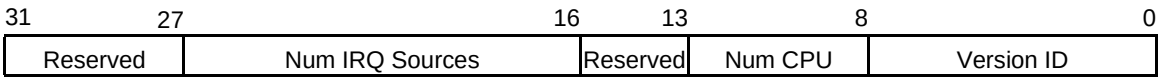
Writing a zero to this register will signal the end of processing for the highest priority interrupt currently in progress for the associated processor. EOI values other than zero are currently undefined and should not be used. Reading this register will return the last value written.

4.2 Global Registers

Only one instance of each of these registers will be provided. These registers will be accessible by each CPU at the same physical address.

4.2.1 Feature Reporting Registers

xFC01000h is Feature Reporting Register 0 (Read Only)



Version ID: Version ID for this interrupt controller. This value will report the level of specification supported by this implementation. Some manufacturers used an early version of the specification to design a PIC. The current document has therefore been assigned a new revision level to distinguish controllers designed to conform to the earlier document. The values for Version ID are defined as follows:

Specification Revision	Version ID
1.0	1
1.2	2

Num CPU: The number of the highest physical CPU supported. For instance, in a 4-processor controller this value will be 3. This allows support of up to 32 processors.

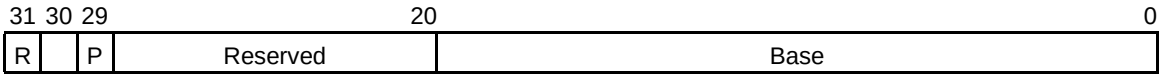
Num IRQ Sources: The number of highest IRQ source supported. For instance, in a 32 source implementation, this value will be 31. This allows support of a maximum of 2048 interrupt sources.

Note: *Reserved bits should always be implemented to return zero to allow use in future revisions which include more features.*

Note: *Two registers shall be provided to report implementation dependent parameters. These registers are read only. This specification defines only feature reporting register 0. Feature reporting register 1 at xFC01010h is reserved for future expansion.*

4.2.2 Global Configuration Registers

xFC01020h is Global Configuration Register 0 (Read/Write)



Base: Base Address Relocation Bits. These bits will determine address bits 28-47 of the register map defined by the open PIC specification. Following reset, this value will be set to 0000Fh which places the register map at a starting address of FFC00000h for 32 bit address systems and FFFF0000FFC00000h for 64 bit address systems. For 32 bit address systems, bits 4 through 19 are not used and will always return zero when read, and should always be written as zero.

P: 8259 Pass Through Enable. Allows the use of an external 8259 pair to provide AT compatibility.

0 = 8259 Pass Through Enable. This mode will connect the 8259 output directly to one of the processor interrupt request pins. The advanced interrupt controller functions will be disabled. This will be the default mode following reset.
1 = 8259 Pass Through Disable. In this mode the PIC will be responsible for all interrupt routing. No 8259 generated inputs will be supported.

R: Reset Controller. Writing a one to this bit will force the controller logic to be reset. All register values will return to their default values. This bit will remain set until the reset sequence is completed, at which time it will be cleared automatically. While this bit is set, the values of all other registers will be undefined.

Two configuration registers will be provided to allow control and configuration of the interrupt controller. This specification defines only Global Configuration Register 0. This register will provide control of several features including the 8259 pass through, register map base address, and interrupt controller soft reset.

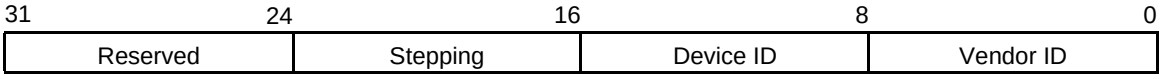
Note: *Global Configuration Register 1 at address xFC01030h is reserved for future expansion.*

4.2.3 Vendor Specific Registers

The address range from xFC01040h–xFC01070h is reserved for vendor specific purposes. These could include diagnostic and testability features which are outside the scope of this specification.

4.2.4 Vendor Identification Register

xFC01080h is Vendor Identification Register (Read Only)



Vendor ID: Specifies the manufacturer of this part. If zero then this register is not supported and the Device ID and Stepping information are not valid. Vendor ID values will be assigned as needed.

Device ID: Vendor specified identifier for this device.

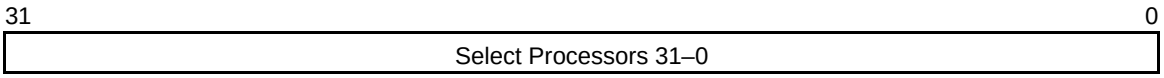
Stepping: Stepping (silicon revision) for this device.

Reserved: Reserved bits, may be defined by vendor.

This register may be used to report the vendor and stepping for a particular part. If this register is not supported, the vendor ID field must return zero when read. If software reads a Vendor ID of zero, then a “generic” PIC compliant with the open PIC specification should be assumed.

4.2.5 Processor Initialization Register

xFC01090h is Processor Initialization Register (Read/Write)



Writing to this register will control the initialization line(s) to each of the processors. Writing a 1 to a bit will cause the corresponding initialization line to be activated and writing a 0 to a bit will cause the initialization line to be deactivated. When a CPU is restarted using this register, the current task priority for that processor will be set to 15 automatically to disable all other interrupts to that processor.

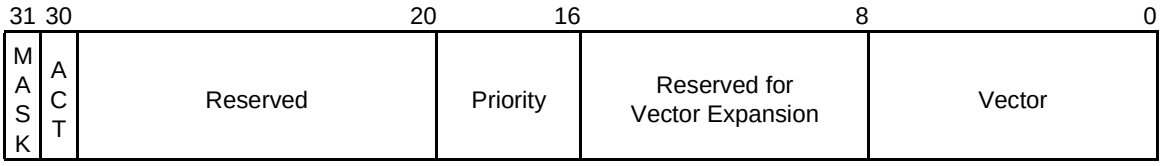
4.2.6 IPI Vector/Priority Registers

xFC010A0h is IPI 0 Vector/Priority Register (Read/Write—shadowed at xFC00008h)

xFC010B0h is IPI 1 Vector/Priority Register (Read/Write)

xFC010C0h is IPI 2 Vector/Priority Register (Read/Write)

xFC010D0h is IPI 3 Vector/Priority Register (Read/Write)



Vector: Vector value to return for this IPI.

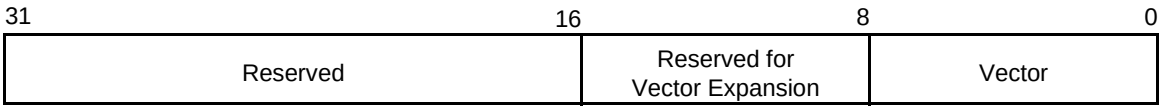
Priority: Priority for this source. See section 5.2.4.

ACT: Activity Bit (Read Only). When set indicates that the vector, priority, and destination information for this source is currently in use by the interrupt controller and should not be changed.

MASK: Mask bit. Setting this bit will disable any further interrupts from this source. Following RESET, this bit will always be set.

4.2.7 Spurious Vector Register

xFC010E0h is Spurious Vector Register (Read/Write)



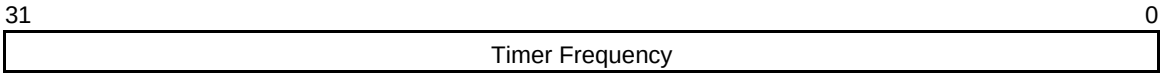
Vector: Spurious interrupt vector value (default is 0xFFh).
Bits 31–8 are reserved. Values written to these bits are ignored.

This register should be used to set the value of the vector returned for spurious interrupts. Refer to section 3.3.5 for more information on spurious vector generation.

4.2.8 Global Timer Registers

The open PIC specification requires four global timers that may be used for system timing or to generate periodic interrupts. The count frequency will be the same for all four counters and will be reported in the timer frequency reporting register. Each counter will have four registers used for configuration and control.

xFC010F0h is Timer Frequency Reporting Register (Read/Write)



This register will be used to report the frequency of the clock source for the global timers. This register will report the frequency value in ticks/second (Hz). The timers are required to be clocked an integral number of times per second. In order to meet the timer frequency specifications, the minimum value in this register will be 0x3D0900h.

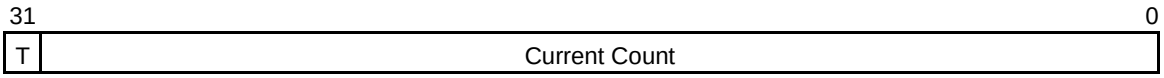
Note: *Depending upon implementation this register may be hard-wired for a predetermined frequency source or it may be written by the system initialization code during startup after the clock frequency has been determined.*

xFC01100h is Global Timer 0 Current Count (Read Only)

xFC01140h is Global Timer 1 Current Count (Read Only)

xFC01180h is Global Timer 2 Current Count (Read Only)

xFC011C0h is Global Timer 3 Current Count (Read Only)



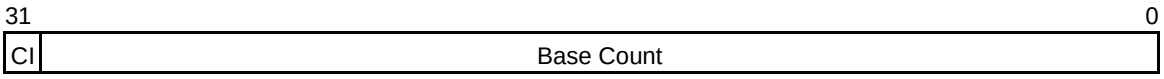
T and Current Count: This register will contain the 31-bit current count value and the toggle bit (T) for the timer. The current count will be loaded initially with the base count, and the toggle bit will be cleared, whenever the count inhibit CI bit in the base count register transitions from a 1 to a 0. When the timer counts to zero, an interrupt will be generated (if not masked), the toggle bit will be inverted, and the count is will be reloaded from the base count register.

xFC01110h is Global Timer 0 Base Count (Read/Write)

xFC01150h is Global Timer 1 Base Count (Read/Write)

xFC01190h is Global Timer 2 Base Count (Read/Write)

xFC011D0h is Global Timer 3 Base Count (Read/Write)



Base Count: This field will contain the 31-bit base count for the timer. This value will be copied to the current count register whenever the CI bit transitions from a 1 to a 0. If the CI is already set to zero (counting in progress), the base count will be copied to the current count at the next zero crossing of the current count.

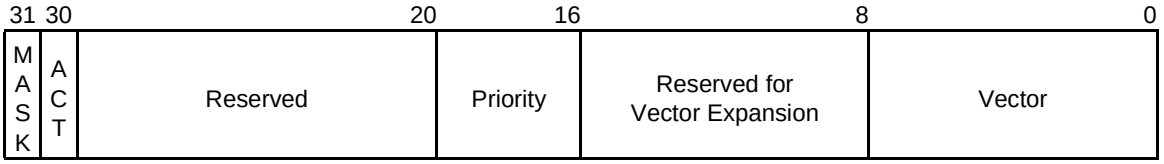
CI: Count Inhibit. Setting this bit to one will inhibit counting for this channel. Setting it to zero will cause the base count to be copied into the current count, the toggle bit to reset (clear), and counting to proceed. Following RESET, this bit will be set (counting will be disabled).

xFC01120h is Global Timer 0 Vector/Priority Register (Read/Write)

xFC01160h is Global Timer 1 Vector/Priority Register (Read/Write)

xFC011A0h is Global Timer 2 Vector/Priority Register (Read/Write)

xFC011E0h is Global Timer 3 Vector/Priority Register (Read/Write)



Vector: vector value to return for this interrupt.

Priority: Priority for this source.

ACT: Activity Bit, Read Only - When set, this bit will indicate that the vector, priority, and destination information for this source are currently in use by the interrupt controller and should not be changed.

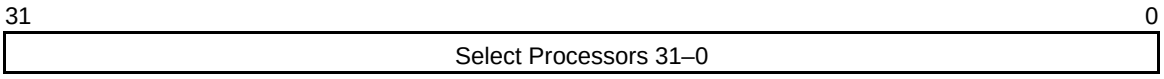
MASK: Mask bit - Setting this bit will disable any further interrupts from this source. Following RESET, this bit will always be set.

xFC01130h is Global Timer 0 Destination Register (Read/Write)

xFC01170h is Global Timer 1 Destination Register (Read/Write)

xFC011B0h is Global Timer 2 Destination Register (Read/Write)

xFC011F0h is Global Timer 3 Destination Register (Read/Write)

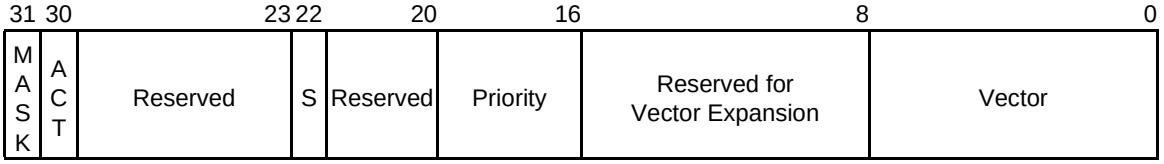


This register will be used to specify the destination(s) for this timer's interrupt events. For timer interrupt events, only the directed delivery mode will be supported. This register may be used to specify multiple destinations (multicast delivery).

4.2.9 Interrupt Source Registers

The address range from xFC10000h to xFC1FFF0h is reserved for interrupt source configuration registers. Each source will have a Vector/Priority Register and Destination Register. The number of interrupt sources (and therefore configuration registers) will be implementation dependent and will be reported in feature reporting register 0. The registers for each source will have the following format:

Vector/Priority Register (Read/Write)



Vector: Vector value to return for this interrupt

Priority: Priority for this source.

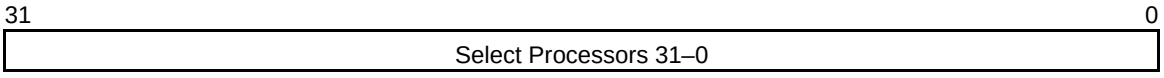
S: Sense - 0 = Positive Edge Triggered, 1 = Active Low Level Triggered

Res. - Reserved

ACT: Activity Bit, Read Only - When set, the bit will indicate that the vector, priority, and destination information for this source are currently in use by the interrupt controller and should not be changed.

MASK: Mask bit - Setting this bit will disable any further interrupts from this source. Following RESET this bit will always be set.

Destination Register (Read/Write)

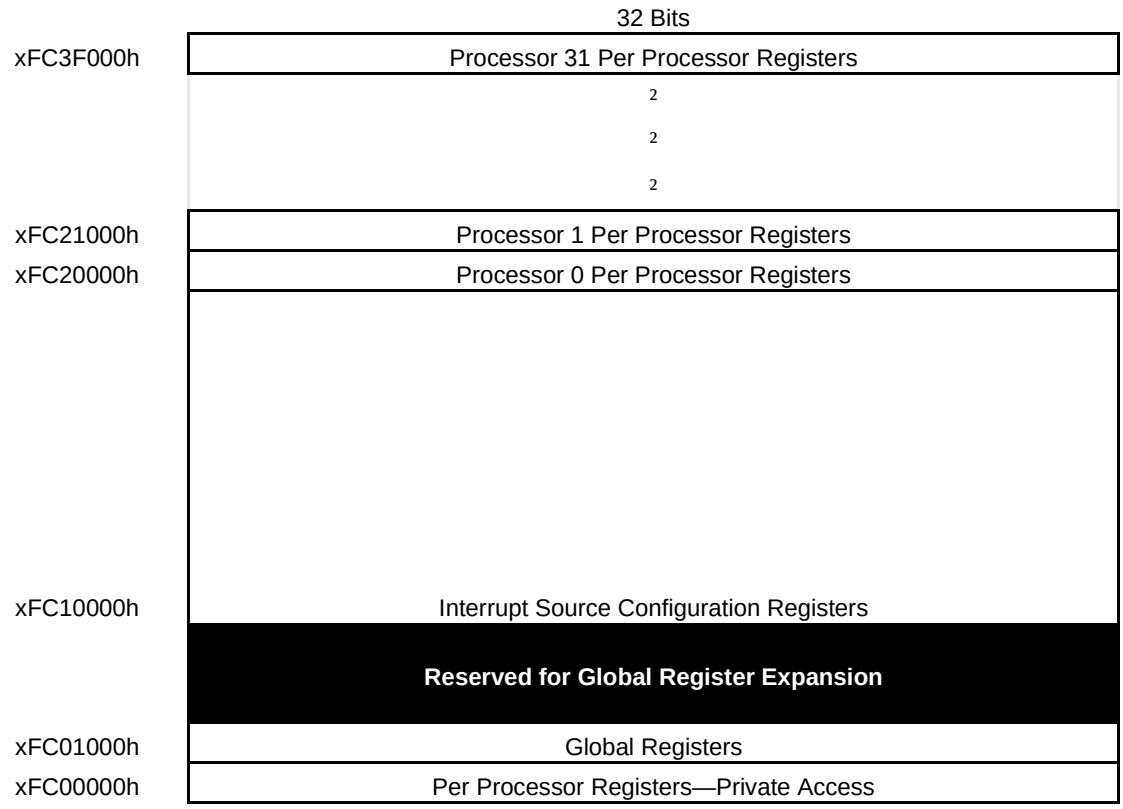


This register will be used to specify the destination(s) for this interrupt source. If a single destination is selected, then events from this source will be directed to that destination. If multiple destinations are selected, then interrupt events from this source will be distributed among the selected destination processors using a fair, implementation-specific distribution algorithm.

4.3 Register Map Summary

4.3.1 Complete Controller Map

The base address of the register map will be determined by the value in the I/O Relocation register. This value will control the most significant four bits of the register addresses. The default base address is FFC00000h. The entire register map occupies a 256-Kbyte address space and will be partitioned as follows:



Note: *All processor registers above Processor 0 are optional. Actual number of processors will be implementation dependent.*

4.3.2 Per Processor Registers

	32 Bits
xFC00FF0h	**Reserved**
	2
	2
	2
xFC000C0h	**Reserved**
xFC000B0h	EOI Register
xFC000A0h[1]	Interrupt Acknowledge Register
xFC00090h	Who Am I Register
xFC00080h	Current Task Priority Register
xFC00070h[2]	IPI 3 Dispatch Command Port
xFC00060h[2]	IPI 2 Dispatch Command Port
xFC00050h[2]	IPI 1 Dispatch Command Port
xFC00040h[2]	IPI 0 Dispatch Command Port
xFC00030h	**Reserved**
xFC00020h	**Reserved**
xFC00010h	**Reserved**
xFC00008h[3,4]	Shadow of IPI Vector Register 0
xFC00000h[3]	Shadow of IPI Dispatch Register 0

Notes:

1. Not required in x86 implementations.

2. Write-only command port.

3. Provided for SLiC register map compatibility.

4. This register is not required in 128-bit data bus implementations.

4.3.3 Global Registers

	32 Bits
xFC0FFF0h	**Reserved**
	2
	2
	2
xFC01200h	**Reserved**
xFC011F0h	Timer 3 Destination Register
xFC011E0h	Timer 3 Vector/Priority Register
xFC011D0h	Timer 3 Base Count Register
xFC011C0h	Timer 3 Current Count Register
xFC011B0h	Timer 2 Destination Register
xFC011A0h	Timer 2 Vector/Priority Register
xFC01190h	Timer 2 Base Count Register
xFC01180h	Timer 2 Current Count Register
xFC01170h	Timer 1 Destination Register
xFC01160h	Timer 1 Vector/Priority Register
xFC01150h	Timer 1 Base Count Register
xFC01140h	Timer 1 Current Count Register
xFC01130h	Timer 0 Destination Register
xFC01120h	Timer 0 Vector/Priority Register
xFC01110h	Timer 0 Base Count Register
xFC01100h	Timer 0 Current Count Register
xFC010F0h	Timer Frequency Reporting Register
xFC010E0h	Spurious Vector Register
xFC010D0h	IPI 3 Vector/Priority Register
xFC010C0h	IPI 2 Vector/Priority Register
xFC010B0h	IPI 1 Vector/Priority Register
xFC010A0h	IPI 0 Vector/Priority Register
xFC01090h	Processor Initialization Register
xFC01080h	Vendor Identification Register
xFC01070h	**Vendor Reserved**
xFC01060h	**Vendor Reserved**
xFC01050h	**Vendor Reserved**
xFC01040h	**Vendor Reserved**
xFC01030h	**Global Configuration Register 1**
xFC01020h	Global Configuration Register 0
xFC01010h	**Feature Reporting Register 1**
xFC01000h	Feature Reporting Register 0

Note: *Global Configuration Register 1 and Feature Reporting Register 1 are reserved for future architecture expansion.*

4.3.4 Interrupt Source Configuration Registers

	32 Bits
xFC1FFF0h	Interrupt Source 2047 Destination
xFC1FFE0h	Interrupt Source 2047 Vector/Priority
	2
	2
	2
xFC10010h	Interrupt Source 0 Destination
xFC10000h	Interrupt Source 0 Vector/Priority

Note: *Actual number of interrupt sources will be implementation dependent.*

5 PROGRAMMING REQUIREMENTS

5.1 General Register View

The open PIC specification specifies two general types of registers: per processor and global. Each processor will have a dedicated set of registers that will provide control for processor specific functions, such as current task priority, EOI, and interprocessor interrupt generation. Each processor will be able to access its own copy of these registers at the same physical location starting at xFC00000h. In addition, these registers will be duplicated in a publicly accessible area starting at xFC20000h.

The global registers will provide control for common interrupt controller resources, such as the interrupt inputs and controller configuration registers. Each processor will be able to access the global registers at the same physical location starting at xFC01000h. The interrupt source configuration registers will be part of the global register set and begin at xFC10000h.

Refer to section 4.3 for a detailed register map.

5.2 General Programming Guidelines

5.2.1 Use of Reserved Bits

In order to guarantee compatibility with future versions of the open PIC specification, the following programming guidelines should be followed:

- Software should make no assumptions about the read state of reserved bits.
- Reserved bits should always be written with the value they return when read. In other words, the registers should be programmed by reading the value, modifying the appropriate fields, and writing the value back.

Following these guidelines will allow future implementations maximum flexibility when defining reserved bits.

5.2.2 Dynamically Changing I/O Interrupt Configuration

The PIC will provide a mechanism for safely changing the I/O interrupt source configuration. This mechanism will be provided to support systems that allow dynamic configuration of I/O devices (e. g., hot swapping). In order to change the vector, priority, sense, or destination of an active (unmasked) interrupt source, the following sequence should be performed:

- Mask the source using the MASK bit in the vector/mode register
- Wait for the activity bit (ACT) for that source to be cleared
- Make the desired changes
- Unmask the source

This sequence ensures that the vector, priority, sense, destination, and mask information will remain valid until all processing of pending interrupts is complete.

5.2.3 EOI Register

Each processor will have a private EOI register which is used to signal the end of processing for a particular interrupt event. If multiple nested interrupts are in service, the EOI command will terminate the interrupt service of the highest priority source. Once an interrupt is acknowledged only sources of higher priority will be allowed to interrupt the processor until the EOI command is received. This register should always be written with a value of zero which is the non-specific EOI command.

5.2.4 IPI and Timer Channel Priority

The priority value programmed for each active IPI and timer channel must be a unique priority on a per CPU basis. This means that for a given CPU, the priority levels for IPI and timer channels targeted to that CPU must be distinct from the priority levels used for all other interrupt sources.

Note: *Multiple timers or IPIs can be programmed for the same priority levels if they specify mutually exclusive CPU targets in their destination registers.*

5.2.5 Interrupt Acknowledge Register

This register will be required only for system implementations that do not utilize interrupt acknowledge bus cycles to retrieve the interrupt vector. Upon receipt of an interrupt signal a processor may read this register to retrieve the vector of the interrupt source that caused the interrupt. This read cycle will have the same effect as an INTA cycle (i. e., further interrupts to the same processor will be masked depending upon priority). The contents of this register must be valid until it is read or an interrupt acknowledge cycle is performed.

5.3 Reset State

After power on or software-initiated RESET, the registers defined by the open PIC specification will be initialized to the following states:

xFC00040h—Write only IPI 0 Dispatch Command Port	**Read Value Undefined**																															
xFC00050h—Write only IPI 1 Dispatch Command Port	**Read Value Undefined**																															
xFC00060h—Write only IPI 2 Dispatch Command Port	**Read Value Undefined**																															
xFC00070h—Write only IPI 3 Dispatch Command Port	**Read Value Undefined**																															
xFC00080h—R/W Current Task Priority Register	**Reserved**																														Priority	
																															Fh	
xFC00090h—Read only Who Am I Register	**Reserved**																														ID	
																															Note 1	
xFC000A0h—Read only Interrupt Acknowledge Register	**Reserved**																														Vector	
	Undefined																															
xFC000B0h—R/W EOI Register	EOI Code																															
	00000000h																															
xFC01000h—Read only Feature Reporting Register 0	**Reserved**																Num IRQ Sources								**Rsrvd.**		Num CPU				Version ID	
																	0–7FFh ($0–2047_{10}$)										0–1Fh				2h	
xFC01020h—R/W Global Configuration Register 0	R		P	**Reserved**																											Base Address	
	0		0																												0000Fh	
xFC01080h—Read only Vendor Identification Register	**Reserved**																Stepping								Device ID						Vendor ID	
	Note 2																Note 2								Note 2						Mfg. ID Number	
xFC01090h—R/W Processor Initialization Register	Select Processors 31–0																															
	00000000h																															
xFC010A0h—R/W IPI 0 Vector/Priority Register	M	A	**Reserved**														Priority				**Reserved**										Vector	
	1																														**Undefined**	
xFC010B0h—R/W IPI 1 Vector/Priority Register	M	A	**Reserved**														Priority				**Reserved**										Vector	
	1																														**Undefined**	
xFC010C0h—R/W IPI 2 Vector/Priority Register	M	A	**Reserved**														Priority				**Reserved**										Vector	
	1																														**Undefined**	
xFC010D0h—R/W IPI 3 Vector/Priority Register	M	A	**Reserved**														Priority				**Reserved**										Vector	
	1																														**Undefined**	
xFC010E0h—R/W Spurious Vector Register	**Reserved**																				**Reserved**										Vector	
																															FFh	
xFC010F0h—R/W Timer Frequency Reporting Register	Timer Frequency																															
	Note 3																															
xFC01100h—R/W Global Timer 0 Current Count	T	Current Count																															
	Undefined																															
xFC01110h—R/W Global Timer 0 Base Count	C	Base Count																															
	1	**Undefined**																															
xFC01120h—R/W Global Timer 0 Vector/Priority Register	M	A	**Reserved**														Priority				**Reserved**										Vector	
	1																														**Undefined**	

xFC01130h—R/W Global Timer 0 Destination Register	Select Processors 31–0																														
	Undefined																														
xFC01140h—R/W Global Timer 1 Current Count	T	Current Count																													
		Undefined																													
xFC01150h—R/W Global Timer 1 Base Count	C	Base Count																													
	1	**Undefined**																													
xFC01160h—R/W Global Timer 1 Vector/Priority Register	M	A	**Reserved**														Priority				**Reserved**										Vector		
	1																														**Undefined**		
xFC01170h—R/W Global Timer 1 Destination Register	Select Processors 31–0																														
	Undefined																														
xFC01180h—R/W Global Timer 2 Current Count	T	Current Count																													
		Undefined																													
xFC01190h—R/W Global Timer 2 Base Count	C	Base Count																													
	1	**Undefined**																													
xFC011A0h—R/W Global Timer 2 Vector/Priority Register	M	A	**Reserved**														Priority				**Reserved**										Vector		
	1																														**Undefined**		
xFC011B0h—R/W Global Timer 2 Destination Register	Select Processors 31–0																														
	Undefined																														
xFC011C0h—R/W Global Timer 3 Current Count	T	Current Count																													
		Undefined																													
xFC011D0h—R/W Global Timer 3 Base Count	C	Base Count																													
	1	**Undefined**																													
xFC011E0h—R/W Global Timer 3 Vector/Priority Register	M	A	**Reserved**														Priority				**Reserved**										Vector		
	1																														**Undefined**		
xFC011F0h—R/W Global Timer 3 Destination Register	Select Processors 31–0																														
	Undefined																														
xFC10000h—R/W Interrupt 0 Vector/Priority Register	M	A	**Reserved**													S	Priority				**Reserved**										Vector		
	1																														**Undefined**		
xFC10010h—R/W Interrupt 0 Destination Register	Select Processors 31–0																														
	Undefined																														
	[2]																														
	[2]																														
	[2]																														
xFC1FFE0h—R/W Interrupt 2047 Vector/Priority Register	M	A	**Reserved**													S	Priority				**Reserved**										Vector		
	1																														**Undefined**		
xFC1FFF0h—R/W Interrupt 2047 Destination Register	Select Processors 31–0																														
	Undefined																														

Notes:

1. ID Value returns a unique value for each processor.

2. Values assigned by manufacturer.

3. Implementation dependent.

6 PowerPC ARCHITECTURE APPENDIX

Implementations of the open PIC specification for PowerPC systems will conform to this specification with the following exceptions:

1. The private access interface to the per processor registers (addresses xFC00000h–xFC00FF0h) is not required.

2. The Who Am I Register in each per processor register space is not required.

3. The interrupt acknowledge register is required.