

### Features

- Optimized for FLEX® and MAX® architectures
- Offers eight levels of individually maskable interrupts
- Expandable to 64 interrupts
- Offers a flexible priority resolution scheme
- Provides programmable interrupt modes and vectors
- Uses approximately 399 logic elements (LEs) in FLEX devices
- Functionally based on the Intel 8259 device, except as noted in the “Variations & Clarifications” section on page 79

### General Description

The Altera® a8259 MegaCore™ function is a programmable interrupt controller. The a8259 can be initialized by the microprocessor through eight data bus lines (din[7..0] and dout[7..0]), and the ncs, nrd, nwr, int, and ninta control signals. **Figure 1** shows the symbol for the a8259.

Figure 1. a8259 Symbol

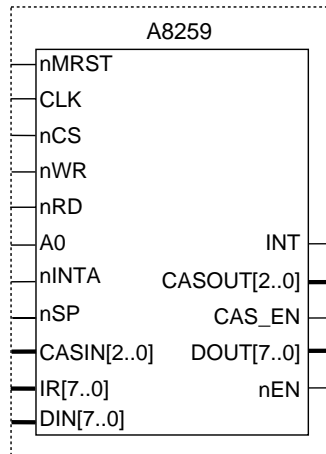


Table 1 describes the input and output ports of the a8259.

Name	Type	Polarity	Description
nmrst	Input	Low	Master reset. When nmrst is asserted, all internal registers assume their default state. The a8259 is idle, awaiting initialization.
clk	Input	–	Clock. All registers are clocked on the positive edge of the clock.
ncs	Input	Low	Chip select. When low, this signal enables the nwr and nrd signals and register access to and from the a8259.
nwr	Input	Low	Write control. When this signal is low (and ncs signal is also low), it enables write transactions to the a8259.
nrd	Input	Low	Read control. When this signal is low (and ncs signal is also low), it enables read transactions from the a8259.
a0	Input	High	Address. This signal serves as a register selector when writing to and reading from internal a8259 registers.
ninta	Input	Low	Interrupt acknowledge. This signal serves as the primary handshake between the a8259 and microprocessor during an interrupt service cycle.
nsp	Input	Low	Slave processor. This signal indicates that the a8259 should be configured as a slave. However, this signal is ignored when the a8259 is configured as a single device. This signal should also be ignored in buffered mode.
casin[2..0]	Input	High	Cascade data bus. These bus signals act as a cascade mode control to a slave a8259. If the a8259 is configured as a master, the bus should be driven low.
ir[7..0]	Inputs	High (1)	Interrupt request. These are eight maskable, prioritized interrupt service request signals.
din[7..0]	Input	–	Data bus. This bus inputs data when writing to internal a8259 registers.
int	Output	High	Interrupt. This signal indicates that the a8259 has made an unmasked service request.
casout[2..0]	Output	High	Cascade data bus. These bus signals act as cascade mode control, and should be connected to the casin[2..0] bus of a slave a8259. When the a8259 is configured as a master, the casout[2..0] bus is ignored.
cas_en	Output	High	Cascade directional bus enable. This signal is intended as a tri-state enable signal to external bidirectional I/O buffers on the cascade control bus.
dout[7..0]	Output	–	Data bus. The output data when reading from internal a8259 registers.
nen	Output	Low	Data enable. This signal indicates that a read cycle is being performed on an internal a8259 register, and it is intended as a tri-state enable to external bidirectional I/O buffers.

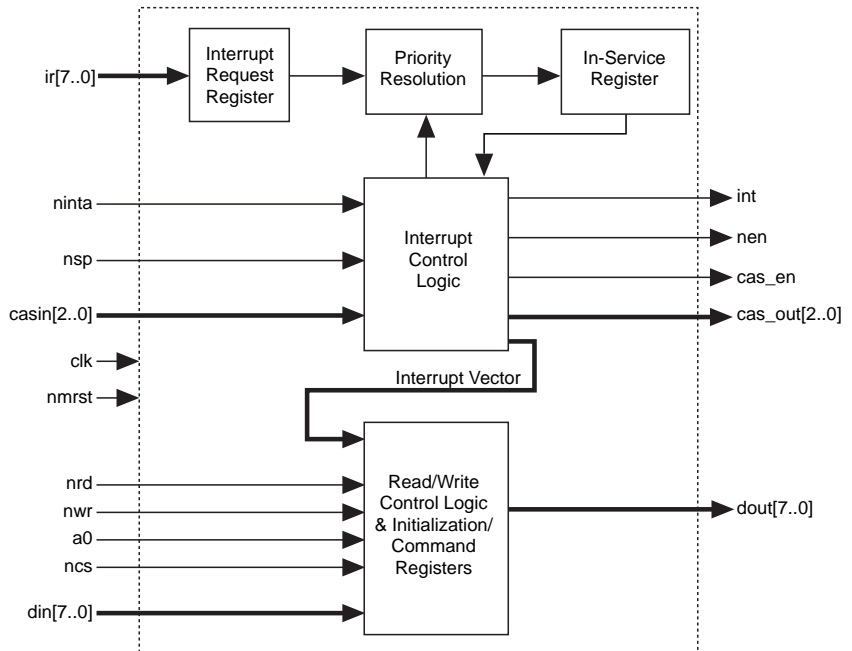
**Note:**

- (1) The interrupt request signals can be set as active high or positive-edge-triggered via bit 3 of Initialization Command Word (ICW) 1 (see “ICW 1” on page 62 for more information).

## Functional Description

Figure 2 shows the a8259 block diagram.

Figure 2. a8259 Block Diagram



The `int` and `ninta` signals provide the handshaking mechanism for the a8259 to signal the microprocessor. The a8259 requests service via the `int` signal and receives an acknowledgment of acceptance from the microprocessor via the `ninta` signal. The `int` signal is applied directly to the microprocessor's interrupt input. Whenever the a8259 receives a valid interrupt request on an `ir` pin (`ir1` through `ir7`), the `int` signal goes high.

The `ninta` input is connected to the microprocessor's interrupt acknowledgment signal. The microprocessor pulses the `ninta` signal twice during the interrupt acknowledgment cycle, which tells the a8259 that the interrupt request has been acknowledged. Then, the a8259 sends the highest priority active interrupt type number onto the `din[7..0]` bus for the microprocessor to acknowledge.

The `ir` inputs are used by external devices to request service, and they can be configured for level-sensitive or edge-sensitive operation.

The `casin[2..0]` and `casout[2..0]` buses, and `nsp` and `cas_en` pins are used to implement the cascade interface. These pins are used when more than one a8259 functions are interconnected in a master/slave configuration, expanding the number of interrupts from 8 up to 64.

## Programming & Initialization

The a8259 operation depends on initial programming. Two types of command words are used for programming the a8259: initialization command words (ICWs) and operation command words (OCWs). ICWs are used to load the a8259 internal control registers, while the OCWs permit the microprocessor to initiate variations in the basic operating modes defined by the ICW registers. [Table 2](#) summarizes how to access the ICW and OCW registers for programming and initialization (for more information on ICW and OCW registers, see [“Register Descriptions” on page 62](#)).

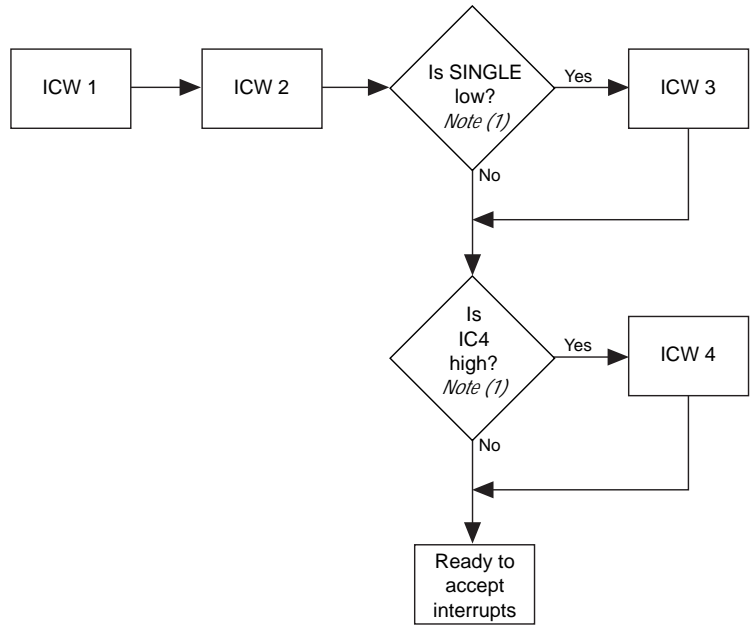
Register	Mnemonics			Description	Access Method
	A0	D4	D3		
ICW 1	0	1	Don't Care	A write with A0 low and D4 high is interpreted as the beginning of an initialization sequence.	Sequential access which starts with ICW 1 and timed by the pulsing <code>nwr</code> signal.
ICW 2	1	Don't Care	Don't Care	This register always follows ICW 1.	
ICW 3	1	Don't Care	Don't Care	The use of this register depends on the value of SINGLE (see <a href="#">Figure 3 on page 61</a> ).	
ICW 4	1	Don't Care	Don't Care	The use of this register depends on the value of IC4 (see <a href="#">Figure 3 on page 61</a> ).	
OCW 1	1	Don't Care	Don't Care	These registers can be accessed randomly (see <a href="#">“Operation Command Word Registers” on page 65</a> for more details).	Random access
OCW 2	0	0	0		
OCW 3	0	0	1		

*Note:*

- (1) “Don't Care” indicates that the bit has no address significance for this register access method. However, the bit will usually have data significance.

To begin an initialization sequence, the `a0` pin must be low, and bit 4 of the `din[7..0]` bus must be high during a valid write cycle. [Figure 3](#) shows the a8259 initialization sequence flow diagram.

Figure 3. a8259 Initialization Sequence Flow Diagram



Note:

(1) For more information on SINGLE and IC4, see Table 3 on page 62.

Figures 4 and 5 show typical write and read cycles, respectively. The *ncs*, *nwr*, and *nrd* signals enable data to be written to and read from the a8259. This data is clocked by the rising edge of *clk*. The *ncs* and *nwr* signals must be held low for an entire clock cycle in order to read or write valid data.

Figure 4. Typical Write Cycle

*X* indicates "don't care." *DV* indicates "data valid."

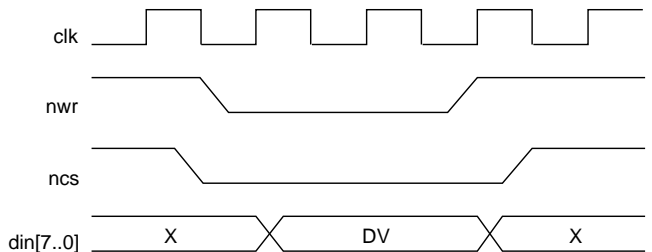
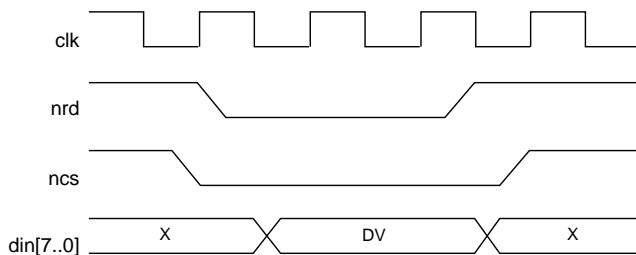


Figure 5. Typical Read Cycle

*X indicates "don't care." DV indicates "data valid."*



## Register Descriptions

The a8259 contains three type of registers:

- Initialization command word (ICW) registers
- Operation command word (OCW) registers
- Interrupt registers

### Initialization Command Word Registers

There are four ICW command registers: ICW 1, ICW 2, ICW 3, and ICW 4.

#### ICW 1

Input data for ICW 1 is sent via the  $\text{din}[7..0]$  bus (a0 must be low and bit 4 of  $\text{din}[7..0]$  must be held high). ICW 1 is deselected with the rising edge of the  $\text{nwr}$  signal. [Table 3](#) describes the ICW 1 register format.

Table 3. ICW 1 Register Format (Part 1 of 2)

Bit	Mnemonic	Description
0	IC4	When low, this bit causes ICW 4 to be reset (i.e., non-buffered mode, no automatic EOI, and a 3-byte interrupt sequence), and the initialization cycle to skip ICW 4. When high, ICW 4 is accessed normally.
1	SINGLE	Single mode. When high, this bit indicates that the a8259 is not cascaded with other a8259 functions. When low, this bit causes the a8259 to operate in cascade mode.
2	ADI	Address interval. When using a 3-byte interrupt sequence, this bit selects the address interval. When low, the address interval is set to eight; otherwise, it is set to four.

Table 3. ICW 1 Register Format (Part 2 of 2)

Bit	Mnemonic	Description
3	LTIM	Level-sensitive or edge-triggered input mode. When high, the $ir[7..0]$ pins are level-sensitive inputs; otherwise, they are positive-edge-triggered.
4	1	This bit is used in conjunction with the $a_0$ signal to select other command registers (see “Interrupt Registers” on page 69).
5	A5	These bits set the interrupt vector address (bits 5 through 7) in a 3-byte interrupt sequence (see “Interrupt Sequencing” on page 70).
6	A6	
7	A7	

*ICW 2*

ICW 2 is selected after the  $a_0$  signal has been high. Input data for ICW 2 is sent via the  $din[7..0]$  bus, and data is clocked by the rising edge of  $clk$ . ICW 2 is deselected with the next falling edge of the  $nwr$  signal.

Table 4 describes the ICW 2 register format.

Table 4. ICW 2 Register Format

Bit	Mnemonic	Description
0	A8	These bits set the interrupt vector address. For bits 8 through 15, the interrupt vector address is set in a single-byte interrupt sequence mode. For bits 3 through 7, the interrupt vector address is set in the same mode. See “Operating Modes & Sequence of Events” on page 76 for more information.
1	A9	
2	A10	
3	A11 / T3	
4	A12 / T4	
5	A13 / T4	
6	A14 / T6	
7	A15 / T7	

If SINGLE (bit 1 of ICW 1) is low, ICW 3 is the next register selected (see “ICW 3” on page 64). If SINGLE is high, ICW 3 is skipped.

The next register considered is ICW 4 (see Figure 3 on page 61). If bit 0 of ICW 1 is high, then ICW 4 is the next register selected; if it is low, ICW 4 is skipped. When a write transaction is completed for ICW 4—or if it is skipped—the initialization sequence is finished, and the a8259 is now ready to accept interrupts.

*ICW 3*

If SINGLE is low, ICW 3 must be initialized. Input data for ICW 3 is sent via the `din[7..0]` bus, and data is clocked by the rising edge of `clk`. ICW 3 is deselected with the next falling edge of the `nwr` signal.

The meaning of the ICW 3 contents depends on whether the a8259 is configured as a master or slave. Table 5 describes the ICW 3 register format for the a8259 configured as a master.

Bit	Mnemonic	Description
0	S0	These bits are slave inputs. When high, each bit indicates that the corresponding interrupt request line is a cascaded slave input. For instance, if S2 is high, the <code>ir2</code> pin is treated as a slave input and receives data from the <code>int</code> signal of another a8259.
1	S1	
2	S2	
3	S3	
4	S4	
5	S5	
6	S6	
7	S7	

Table 6 describes the ICW 3 register format when the a8259 is configured as a slave.

Bit	Mnemonic	Description
0	ID0	Slave identification. These bits set the slave ID for the a8259.
1	ID1	
2	ID2	
3	0	These bits are not used when the a8259 is configured as a slave, and they should be low.
4	0	
5	0	
6	0	
7	0	

At this point in the initialization process, the next register selected depends on whether bit 0 of ICW 1 is high. If bit 0 of ICW 1 is high, ICW 4 is selected (see “ICW 4” on page 65). If bit 0 is low, ICW 4 is skipped and the a8259 is ready to accept interrupts.



*ICW 4*

ICW 4 is initialized when bit 0 of ICW 1 is high. Input data for ICW 4 is sent via the  $\text{din}[7..0]$  bus, and the data is clocked by the rising edge of  $\text{clk}$ . ICW 4 is deselected with the next falling edge of the  $\text{nwr}$  signal. When a write transaction for ICW 4 is finished—or if ICW 4 is skipped—the initialization sequence is complete, and the a8259 is ready to accept interrupts.

Table 7 describes the ICW 4 register formats.

Bit	Mnemonic	Description
0	$\mu\text{PM}$	Microprocessor mode. When this bit is low, the a8259 operates in a 3-byte interrupt sequence mode. If the bit is high, it operates in a single-byte interrupt sequence mode.
1	AEOI	Automatic end of interrupt. When this bit is high, the AEOI is enabled; otherwise, the AEOI is disabled.
2	M/S	Master/slave. When this bit is high in buffered mode, the a8259 is configured as a slave, and when it is low, the a8259 is configured as a master. When the device is not in buffered mode, this bit is in a “don’t care” condition.
3	BUF	Buffered mode. When this bit is high, the a8259 is in buffered mode. See “ <a href="#">Operating Modes &amp; Sequence of Events</a> ” on page 76 for more information.
4	SFNM	Special fully nested mode. When this bit is high, the a8259 is in special fully nested mode.
5	0	These bits are unused and should be set low.
6	0	
7	0	

## Operation Command Word Registers

Once the appropriate OCW registers have been issued to the a8259, they will be ready for operation.

There are three OCW registers: OCW 1, OCW 2, and OCW 3. These command registers control the operation of the a8259, and permit the interrupt interface operation to be further modified—after the a8259 has been initialized. Unlike the initialization sequence, which requires the outputs of an ICW to be in a special sequence, the OCWs can be issued under program control whenever needed and in any order.

*OCW 1*

OCW 1 is selected by setting the `a0` pin high. Input data for OCW 1 is sent via the `din[7..0]` bus, and the data is clocked by the rising edge of `clk`.

**Table 8** describes the OCW 1 register format.

Bit	Mnemonic	Description
0	M0	When more than one of these bits is high, the corresponding interrupt request inputs are masked; otherwise, they are not masked.
1	M1	
2	M2	
3	M3	
4	M4	
5	M5	
6	M6	
7	M7	

*OCW 2*

OCW 2 is selected by setting the `a0` pin and resetting bits 3 and 4 of the `din[7..0]` bus low. Input data for OCW 2 is sent via the `din[7..0]` bus, and the data is clocked by the rising edge of `clk`.

**Table 9** describes the OCW 2 register format.

Bit	Mnemonic	Description
0	L0	Interrupt level. These bits determine the interrupt level that is acted upon when bit 6 (SL) is asserted (see <a href="#">Table 10</a> ).
1	L1	
2	L2	
3	1	These bits are used as address decode and must always be low.
4	1	
5	EOI	These bits control the rotate and end of interrupt (EOI) commands (see <a href="#">Table 11 on page 67</a> ).
6	SL	
7	R	

**Table 10** describes the interrupt levels acted upon when SL (bit 6 of OCW 2) is asserted.

Interrupt Level	Mnemonic		
	L2	L1	L0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

**Table 11** describes the rotate and EOI commands controlled by bits 5 through 7 of the OCW 2 command register.

R	SL	EOI	Command
0	0	1	Non-specific EOI command
0	1	1	Specific EOI command
1	0	1	Rotate on non-specific EOI command
1	0	0	Rotate on automatic EOI mode (set)
0	0	0	Rotate on automatic EOI mode (clear)
1	1	1	Rotate on specific EOI command (L0, L1, and L2 are used)
1	1	0	Specific priority command (L0, L1, and L2 are used)
0	1	0	No operation

### *OCW 3*

OCW 3 is selected by setting the `a0` pin, resetting bit 4 low, and bit 3 high. Input data for OCW 3 is sent via the `din[7..0]` bus, and the data is clocked by the rising edge of `clk`.

Table 12 describes the OCW 3 register format.

Bit	Decode	Description
0	RIS	Read register command. These bits control which status register will be accessed on the next read cycle (see Table 13).
1	RR	
2	P	When this bit is high and RR is high, the a8259 enters the poll mode; the next <code>nrd</code> cycle ends the poll mode (see “Poll Command” on page 75).
3	1	These bits are used as address decode. Bit 4 must be low and bit 3 must be high.
4	0	
5	SMM	Special mask mode. These bits are used to enable, set, and clear the special mask mode function (see Table 14).
6	ESMM (1)	
7	0	This bit is unused and should be tied to GND.

Note:

(1) Enable special mask mode.

Table 13 describes the read register commands for bits 0 and 1 of the OCW 3 command register.

RR	RIS	Command
0	0	No action
0	1	No action
1	0	Read interrupt request register (IRR) on next read cycle
1	1	Read in-service register (ISR) on next read cycle

Table 14 describes the special mask mode commands for bits 5 and 6 of the OCW 3 command register.

Bit 6 ESMM	Bit 5 SMM	Command
0	0	No action
0	1	No action
1	0	Reset special mask
1	1	Set special mask

## Interrupt Registers

The a8259 contains two interrupt registers:

- Interrupt request register (IRR)
- In-service register (ISR)

### *Interrupt Request Register*

The IRR stores all interrupts that are requesting service. In edge-triggered mode (when bit 3 of ICW 1 is low), each `ir` pin is synchronized to the `clk` signal. Positive-edge detection is performed, and the result is clocked into the IRR.

In level-triggered mode (when bit 3 of ICW 1 is high), each `ir` signal is clocked directly into the IRR. The falling edge of the first `ninta` signal from the microprocessor freezes the IRR so all interrupts can be evaluated. The level of an `ir` signal from the microprocessor must be maintained until after the falling edge of the `ninta` signal. Interrupt handshaking protocol must be completed before the next interrupt can be received. [Table 15](#) shows the IRR format.

Bit	Decode
0	IR0
1	IR1
2	IR2
3	IR3
4	IR4
5	IR5
6	IR6
7	IR7

### *In-Service Register*

The ISR stores the interrupt level currently being serviced. Data is enabled by the first `ninta` signal of the interrupt acknowledge sequence. In AEIO mode, data is reset upon the final rising edge of the `ninta` signal in the interrupt sequence. Otherwise, the microprocessor must issue an EOI command by writing the appropriate value to the OCW 2 command register. [Table 16](#) shows the ISR format.

Table 16. ISR Format

Bit	Decode
0	ISR0
1	ISR1
2	ISR2
3	ISR3
4	ISR4
5	ISR5
6	ISR6
7	ISR7

## Interrupt Sequencing

The a8259 supports two interrupt sequencing modes:

- 3-byte interrupt sequence mode
- Single-byte interrupt sequence mode

### 3-Byte Interrupt Sequence Mode

The 3-byte interrupt sequence mode provides a 24-bit interrupt vector. The interrupt sequence for this mode is as follows:

1. One or more of the interrupt request signals ( $ir[7..0]$ ) are high, which sets the corresponding bit in the IRR.
2. The a8259 checks the priority and masks for the interrupt, and if appropriate, sets the  $int$  signal.
3. The microprocessor responds by asserting  $ninta$  to the a8259.
4. The a8259 latches the interrupt request signals on the falling edge of the  $ninta$  signal (when level-triggered). The a8259 sets the corresponding bit in the IRR on the following rising edge of  $ninta$ . Simultaneously, the bit in the IRR is reset. The a8259 places a fixed vector opcode of binary 11001101 on the  $dout[7..0]$  bus while the  $ninta$  signal is low. The vector opcode indicates that the following two bytes will contain the interrupt vector.
5. The microprocessor responds to the vector opcode by sending two more  $ninta$  pulses. The falling edge of the first  $ninta$  pulse causes the lower eight interrupt vector address bits to be placed on the  $dout[7..0]$  bus. The contents of the lower eight bits depend on the value of the address interval bit (bit 2 of ICW 1). See [Tables 17](#) and [18](#).

**Table 17. Contents of the First Interrupt Vector Bytes** *Notes (1), (2)*

ir	dout7	dout6	dout5	dout4	dout3	dout2	dout1	dout0
7	A7	A6	A5	1	1	1	0	0
6	A7	A6	A5	1	1	0	0	0
5	A7	A6	A5	1	0	1	0	0
4	A7	A6	A5	1	0	0	0	0
3	A7	A6	A5	0	1	1	0	0
2	A7	A6	A5	0	1	0	0	0
1	A7	A6	A5	0	0	1	0	0
0	A7	A6	A5	0	0	0	0	0

**Table 18. Contents of the First Interrupt Vector Bytes** *Notes (2), (3)*

ir	dout7	dout6	dout5	dout4	dout3	dout2	dout1	dout0
7	A7	A6	1	1	1	0	0	0
6	A7	A6	1	1	0	0	0	0
5	A7	A6	1	0	1	0	0	0
4	A7	A6	1	0	0	0	0	0
3	A7	A6	0	1	1	0	0	0
2	A7	A6	0	1	0	0	0	0
1	A7	A6	0	0	1	0	0	0
0	A7	A6	0	0	0	0	0	0

Notes to tables:

- (1) Interval = 4 (bit 2 of ICW 1 is high).
  - (2) A7 through A5 are derived from the contents of bits 5, 6, and 7 of ICW 1.
  - (3) Interval = 8 (bit 2 of ICW 1 is low).
6. The upper eight interrupt vector address bytes are released on the falling edge of the second `ninta` pulse. The contents of the upper eight interrupt vector address bytes are always derived from the contents of ICW 2. See [Table 19](#).

**Table 19. Contents of the Second Interrupt Vector Bytes**

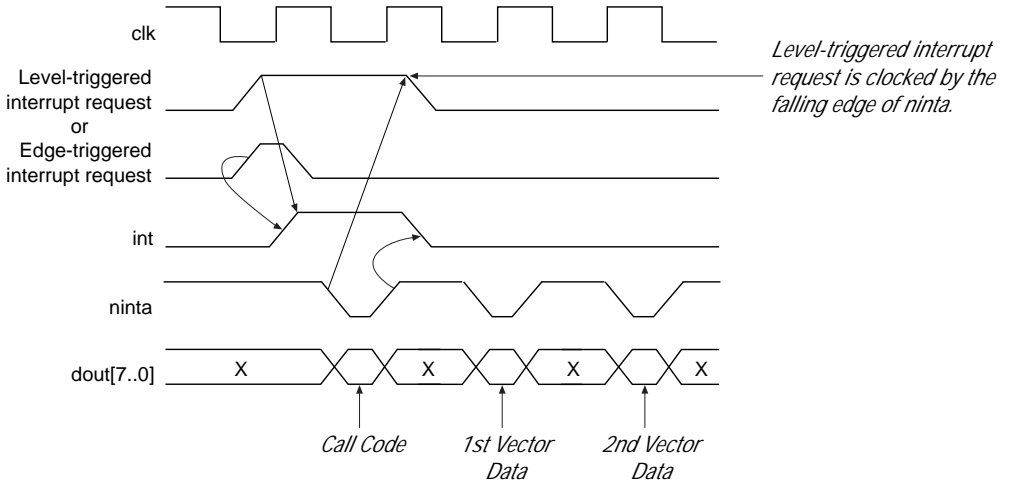
dout7	dout6	dout5	dout4	dout3	dout2	dout1	dout0
A15	A14	A13	A12	A11	A10	A9	A8

7. In AEOI mode, a bit in the ISR is reset on the rising edge of the last `ninta` pulse. When not in AEOI mode, an appropriate EOI command must be issued to end the interrupt sequence.

Figure 6 shows the timing waveforms for the 3-byte interrupt sequence mode.

Figure 6. 3-Byte Interrupt Sequence Mode Timing Waveforms

*X indicates "Don't Care."*



## Single-Byte Interrupt Sequence Mode

The single-byte interrupt sequence mode provides an 8-bit interrupt vector. The interrupt sequence for this mode is as follows:

1. One or more of the `ir[7..0]` signals are high, which sets the corresponding bit in the IRR.
2. The a8259 checks the priority and masks for the interrupt, and if appropriate, sets the `int` signal.
3. The microprocessor responds by asserting `ninta` to the a8259.
4. The a8259 latches the `ir` signal on the falling edge of the `ninta` signal (when level-triggered). The a8259 sets the corresponding ISR bit on the following rising edge of the `ninta` signal. Simultaneously, the interrupt request bit is reset, and no data is driven onto the `dout[7..0]` bus for this cycle.
5. The microprocessor issues a second `ninta` pulse. An 8-bit interrupt vector is driven onto the `dout[7..0]` bus. See [Table 20](#).



*Table 20. Contents of the Interrupt Vector Bytes Note (1)*

ir	dout7	dout6	dout5	dout4	dout3	dout2	dout1	dout0
7	T7	T6	T5	T4	T3	1	1	1
6	T7	T6	T5	T4	T3	1	1	0
5	T7	T6	T5	T4	T3	1	0	1
4	T7	T6	T5	T4	T3	1	0	0
3	T7	T6	T5	T4	T3	0	1	1
2	T7	T6	T5	T4	T3	0	1	0
1	T7	T6	T5	T4	T3	0	0	1
0	T7	T6	T5	T4	T3	0	0	0

Note:

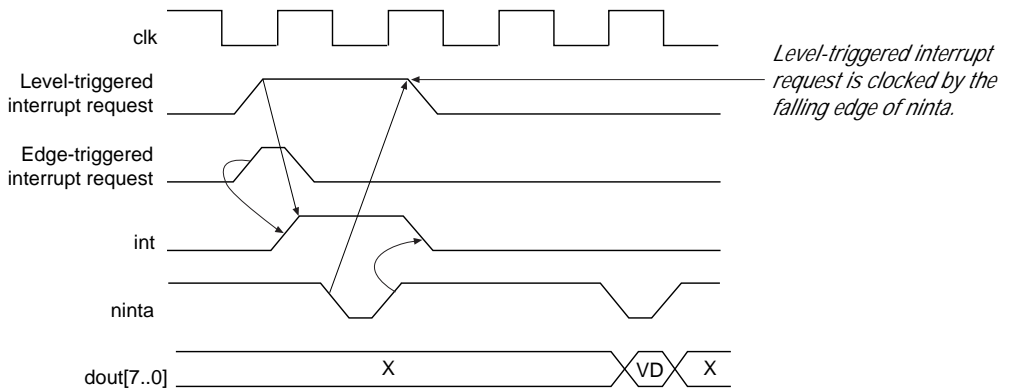
(1) T7 through T3 derive from the contents of ICW 2 (bits 4 through 7).

- In AEOI mode, the ISR bit is reset on the rising edge of the last *ninta* pulse. When not in AEOI mode, an appropriate EOI command is issued to end the interrupt sequence.

Figure 7 shows timing waveforms for the single-byte interrupt sequence mode.

Figure 7. Single-Byte Interrupt Sequence Mode Timing Waveforms

*X indicates "Don't Care." VD indicates "Vector Data."*



## Operational Commands

The a8259 supports several operational commands:

- Priority rotation
- EOI
- Special mask mode (SMM)
- Trigger modes
- Poll command

### Priority Rotation

The priority rotation command can adjust the interrupt request priority. The a8259 supports two types of rotation commands: automatic and specific rotation.

The automatic rotation command rotates an interrupt that has just been serviced to the lowest priority. For example, if the `ir5` interrupt has just been serviced, it is assigned the lowest priority, and the `ir6` interrupt is then given the highest priority. For a system with equal priority interrupts, this process ensures that an interrupt waits for no more than seven other devices to be serviced. Automatic priority can be configured to operate on a non-specific EOI, or an automatic EOI using the OCW 2 command register.

The specific rotation command is similar to the automatic rotation command, except the interrupt to be assigned the lowest priority is specified using bits 0 through 2 of the OCW 2 command register. Specific rotation can be accomplished by issuing a set priority command or a rotate-on-specific EOI command.

### End of Interrupt

The EOI command is used to clear the last interrupt request serviced in an ISR bit. There are two methods used to issue an EOI command: automatic or non-specific.

If an automatic EOI command is used (bit 1 of ICW 4 is high), a non-specific EOI command is issued at the rising edge of the last `ninta` pulse in the interrupt sequence. A non-specific EOI clears the ISR bit currently set at the highest priority. As long as a specific rotation is not used, a non-specific EOI will always clear last interrupt request serviced in the ISR bit.

When the interrupt priority scheme is disturbed (usually in specific rotation), a specific EOI command is issued to clear the ISR bit of the interrupt request specified in bits 0 through 2 of the OCW 2 command register.

## Special Mask Mode

In SMM, masking an interrupt does not inhibit the reception of lower priority interrupts. Only the interrupt being serviced is masked. With SMM, any interrupt may be selectively enabled using the mask register.

## Trigger Modes

Interrupt request lines may be configured in an edge- or level-triggered mode. In edge-triggered mode, an interrupt request is clocked on the rising edge of the clock. In level-triggered mode, an interrupt is generated merely by placing a high on an `ir` pin. This level must be maintained until after the falling edge of the first `ninta` pulse of the interrupt sequence. The trigger mode can be programmed in bit 3 of the ICW 1 command register.

## Poll Command

The poll command provides a way to expand a system allowing the microprocessor to service more than 64 interrupts. In poll mode, the `int` signal should be ignored. Each `a8259` is “polled” individually to determine which interrupts are requesting service. After setting the poll (using bit 2 of OCW 3), the microprocessor simply reads from each `a8259` (each interrupt read transaction must be preceded by a write transaction to the poll bit in order to reset the ISR). If an interrupt is pending, the corresponding ISR bit is set on the falling edge of the read cycle, and an interrupt ID byte (bit 7) set high is placed on the `dout[7..0]` bus.

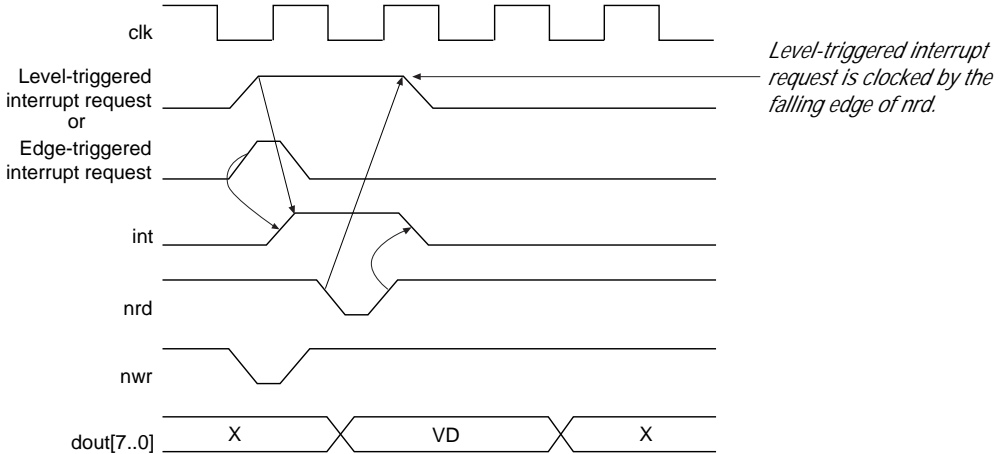
Table 21 shows the interrupt ID word format for the poll command.

Bit	Decode	Description
0	ID0	Interrupt ID. These bits identify the pending interrupt.
1	ID1	
2	ID2	
3	Don't Care	–
4	Don't Care	–
5	Don't Care	–
6	Don't Care	–
7	IP	Interrupt pending. When set, this bit indicates an interrupt is pending. If this bit is cleared, the interrupt ID will be ignored.

Figure 8 shows timing waveforms of the poll mode.

Figure 8. Poll Mode Timing Waveforms for Bit 2 of OCW 3

*X indicates "Don't Care." VD indicates "Vector Data."*



## Operating Modes & Sequence of Events

The a8259 can operate in four different modes:

- Fully nested mode
- Cascade mode
- Special fully nested mode
- Buffered mode

### Fully Nested Mode

The fully nested mode is the default mode, after the master clear. The a8259 will enter this mode when initialization is completed, unless another mode is specifically programmed. When the a8259 is in fully nested mode, the following sequence of events occurs:

1. The interrupt requests are prioritized from 0 (highest priority) to 7 (lowest priority).
2. When an interrupt request is acknowledged, the highest priority, unmasked request is determined, the corresponding bit in the ISR is set, and the IRR bit is reset. While the ISR bit is being set, further interrupts from lower priority sources are ignored. Interrupts from higher priority sources are clocked, causing the *int* signal to remain active until the interrupt is serviced.

3. During the appropriate handshaking sequence using the `inta` and `ninta` signals, the interrupt vector information is placed on the `dout[7..0]` bus.
4. In AEOI mode, the ISR bit is reset on the rising edge of the last `ninta` pulse. When not in AEOI mode, an appropriate EOI command is issued to end the interrupt sequence.

## Cascade Mode

The cascade mode provides easy expansion of the a8259. In this mode, a single a8259 is configured as a master, while other a8259 functions (from 1 to 8 a8259 functions) are configured as slaves. The `int` signal of each slave is connected to an `ir` input on the master. The master's `int` signal serves as an interrupt to the microprocessor. The master's `casout[2..0]` bus is connected to a slave's `casin[2..0]` bus. Each a8259 has a unique `ncs` signal and all other inputs to the a8259 are connected in parallel.

When a slave receives an interrupt, the master asserts its `int` signal. The master enables the slave by placing the slave's address on the `casout[2..0]` bus at the rising edge of the first `ninta` pulse. The slave is then responsible for completing the `int` and `ninta` handshaking required by the interrupt sequence. The slave will place its interrupt vector information on the `dout[7..0]` bus as required by the interrupt sequence.

### *3-Byte Interrupt Sequence in Cascade Mode*

For a 3-byte interrupt sequence in cascade mode, the handshaking between the `int` and `ninta` signals is as follows:

1. The master clocks the ISR bit that corresponds to the slave input on the falling edge of the first `ninta` pulse. The master also simultaneously resets the IRR bit and places a fixed vector opcode of binary 11001101 on the `dout[7..0]` bus. The vector opcode indicates that the two bytes that follow will contain the interrupt vector. The master enables the slave by placing the slave's address on the `casout[7..0]` bus at the rising edge of the first `ninta` pulse.

2. The microprocessor responds to the vector opcode by sending two more `ninta` pulses. The slave sets the appropriate ISR bit on the falling edge of the second `ninta` pulse. Simultaneously, the slave's IRR bit is reset. The falling edge of the second `ninta` pulse also causes the slave to place the lower eight interrupt vector address bits on the slave's `dout[7..0]` bus. The upper eight interrupt vector address bits are released on the falling edge of the third `ninta` pulse.
3. Two EOI commands must be issued to end the interrupt sequence: one to the master and the other to the slave.

### *Single-Byte Interrupt Sequence in Cascade Mode*

In a single-byte interrupt sequence in cascade mode, the handshaking between the `int` and `ninta` signals is as follows:

1. The master sets the ISR bit that corresponds to the slave input on the falling edge of the first `ninta` pulse. The master also simultaneously resets the IRR bit and no data is driven onto the `dout[7..0]` bus for this cycle. The master enables the slave by placing the slave's address on the `casout[7..0]` bus at the rising edge of the first `ninta` pulse.
2. The microprocessor issues a second `ninta` pulse. The slave sets the corresponding ISR bit on the falling edge of the second `ninta` pulse. Simultaneously, the slave's IRR bit is reset. The slave drives the eight interrupt vector address bits onto the `dout[7..0]` bus.
3. Two EOI commands must be issued to end the interrupt sequence: one to the master and one to the slave.

The slave's address will remain on the `casout[2..0]` bus until the rising edge of the last `ninta` pulse.

### **Special Fully Nested Mode**

This mode is used in conjunction with the cascade mode to preserve the priority structure within each slave. To operate in this mode, only bit 1 of ICW 4 of the master should be high. The slaves are configured in normal fully nested mode. When a slave is in service, it will not be locked out of the master's priority logic; the master can recognize interrupts from higher priority sources within that slave.

To complete the interrupt service and ensure that all interrupts from the slave have been serviced, the microprocessor sends a non-specific EOI command to the slave and reads the slave's interrupt request register for active low signals. If the interrupt request register is low, a non-specific EOI command is issued to the master. Otherwise, the master services the pending interrupt request.

### Buffered Mode

The buffered mode was originally intended to support board designs where tri-state buffers were needed to drive the data bus. The a8259 has separate `nsp` and `nen` signals, and the `nen` signal is always available. Instead of using the `nsp` signal, the buffered mode can determine the master/slave configuration by using bits 2 and 3 of the ICW 4 command register.

## Variations & Clarifications

The following characteristics distinguish the Altera a8259 function from the Intel 8259A device:

- A master clear is provided with the a8259.
- A clock signal has been added, and synchronous design rules have been incorporated to improve operation and reliability. All input signals except `nmrst` and `ir` should be synchronous to the clock signal. All inputs must be asserted for one clock cycle to ensure reliable operation.
- Bidirectional I/O pins are split into separate inputs, outputs, and corresponding tri-state control lines. This features makes the a8259 compatible with the bus or multiplexer scheme used internally in a design.
- The `dout[7..0]`, `casout[2..0]`, `int`, `cas_en`, `en`, and `nen` outputs are driven by complex logic structures and are prone to glitches. If appropriate, these signals should be registered in a target application.
- Because the a8259 can be used in various Altera architectures, no timing information is included in this data sheet.
- Automatic EOI in slave mode is implemented within the a8259.

Copyright © 1995, 1996, 1997, 1998, 1999 Altera Corporation, 101 Innovation Drive,  
San Jose, CA 95134, USA, all rights reserved.

By accessing this information, you agree to be bound by the terms of Altera's  
Legal Notice.