CA82C59A

PROGRAMMABLE INTERRUPT CONTROLLER

- **Pin and functional compatibility with the industry standard 8259/8259A**
- **Fully static, *high speed* design (10 & 8 MHz)**
- **Compatible with 8080/85, 8086/88, 80286/386 and 68000 family micro-processor systems**
- **TTL input/output compatibility**
- **Low power CMOS implementation**
- **Eight level priority controller**
- **Expandable to 64 levels**
- **Programmable interrupt modes, with each interrupt maskable**
- **Edge- or level-triggered interrupt request inputs**
- **Polling operation**

The CA82C59A is a high performance, completely programmable interrupt controller. It can process eight interrupt request inputs, assigning a priority level to each one, and is cascadable up to 64 interrupt requests. Individual interrupting sources are maskable. Its two modes of operation (Call and Vector) allow it to be used with virtually all 8000 and 80000 type processors, as well as with the 68000 family of microprocessors.

Featuring fully static, very high speed operation, the CA82C59A is designed to relieve the system CPU from polling in a multi-level priority interrupt system. Its high performance makes it ideally suited for aerospace and defense applications. Its very low power consumption makes it useful in portable systems and systems with low power standby modes.

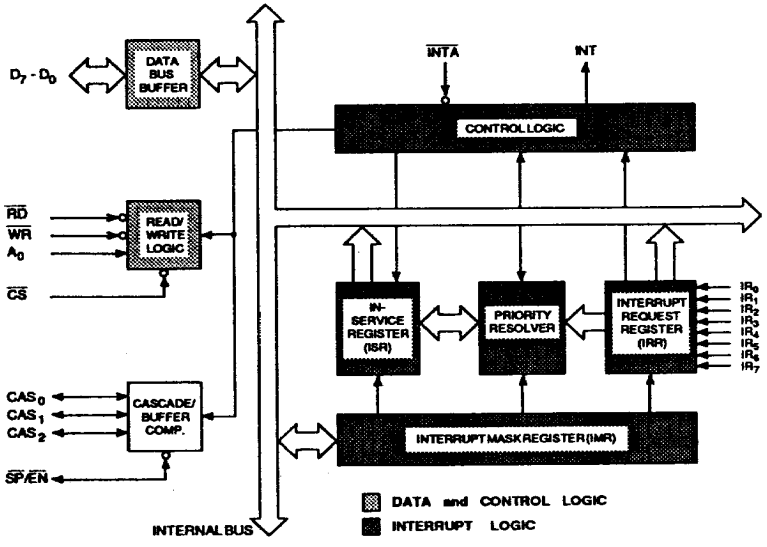


Figure 1 : CA82C59A BLOCK DIAGRAM

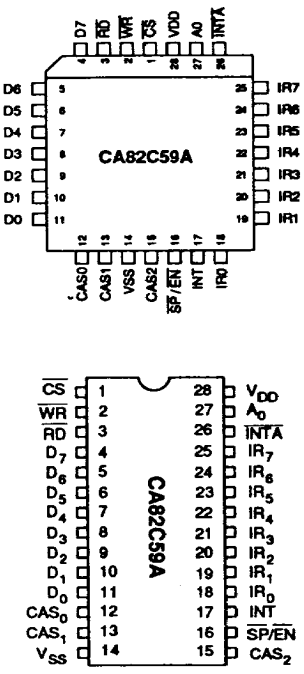


Figure 2 : PLCC and DIP PIN CONFIGURATIONS

Document: 808259.MD500.01

Table 1 : PIN DESCRIPTIONS

Symbol	Pins		Type	Name and Function
	PLCC	PDIP		
A_0	27	27	I	**A_0 Address Line:** This signal acts in conjunction with the $\overline{CS}$, $\overline{WR}$ and $\overline{RD}$ signals. It is used by the CA82C59A to decipher various command words written by the CPU, and Status information read by the CPU. It is typically connected to the CPU - A_0 address line.
CAS_{0-2}	12, 13, 15	12, 13, 15	I/O	**Cascade Line:** These signals are outputs for the master CA82C59A, and inputs for slaved CA82C59As. The CAS lines are used as a private bus by a CA82C59A master to control a multiple CA82C59A system structure.
$\overline{CS}$	1	1	I	**Chip Select:** An active LOW signal used to enable $\overline{RD}$ and $\overline{WR}$ communication between the CPU and the CA82C59A. Note that $\overline{INTA}$ functions are independent of $\overline{CS}$.
$D_7 - D_0$	4 - 11	4 - 11	I/O	**Data Bus:** Bi-directional, 3-state, 8-bit data bus for the transfer of control, status and interrupt vector information.
INT	17	17	O	**Interrupt:** This signal goes HIGH when a valid interrupt request is asserted. It is used to interrupt the CPU, and is thus connected to the CPU interrupt pin.
$\overline{INTA}$	26	26	I	**Interrupt Acknowledge:** Signal used to enable the CA82C59A interrupt vector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the CPU.
IR_{0-7}	18 - 25	18 - 25	I	**Interrupt Requests:** Asynchronous input signals, an interrupt request is executed by raising an IR input (LOW to HIGH), and holding it HIGH until it is acknowledged (Edge Triggered Mode), or just by a HIGH level on an IR input (Level Triggered Mode).
$\overline{RD}$	3	3	I	**Read:** Active LOW signal used to enable the CA82C59A to output status information onto the data bus for the CPU, when CS is LOW.
$\overline{SP}/\overline{EN}$	16	16	I/O	**Slave Program/Enable Buffer:** Active LOW, dual function control signal. When in the *Buffered Mode*, it can be used as an output to control buffer transceivers ($\overline{EN}$). When not in the buffered mode it can be used as an input to designate a master (SP = 1) or a slave (SP = 0).
V_{DD}	28	28	-	**Power:** 5 v ± 10% DC Supply
V_{SS}	14	14	-	**Ground:** 0 v
$\overline{WR}$	2	2	I	**Write:** Active LOW signal used to enable the CA82C59A to accept command words from the CPU, when $\overline{CS}$ is LOW.

FUNCTIONAL DESCRIPTION

The CA82C59A Programmable Interrupt Controller is designed for use in interrupt driven micro-computer systems. Acting as an overall peripherals manager, its functions include:

- Accepting interrupt requests from assorted peripheral devices
- Determining which is the highest priority
- Establishing whether or not the new interrupt is of a higher priority than any interrupts which might be currently being serviced, and if so,
- Issuing an interrupt to the CPU
- Then providing the CPU with the *interrupt service routine* address of the interrupting peripheral

Each peripheral device usually has a specific interrupt service routine which is particular to its operational or functional requirements within the system. The CA82C59A can be programmed to hold a pointer to the service routine addresses associated with each of the peripheral devices under its control. Thus when a peripheral interrupt is passed through to the CPU, the CA82C59A can set the CPU Program Counter to the interrupt service routine required. These *pointers* (or vectors) are addresses in a vector table.

The CA82C59A is intended to run in one of two major operational modes, according the type of CPU being used in the system. The *CALL Mode* is used for 8085 type microprocessor systems, while the *VECTOR Mode* is reserved for those systems using more sophisticated processors such as the 8088/86, 80286/386 or 68000 family.

In either mode, the CA82C59A can manage up to eight interrupt request levels individually, with a maximum capability of up to 64 interrupt request levels when cascaded with other CA82C59As. A selection of priority modes is also available such that interrupt requests can be processed in a number of different ways to meet the requirements of a variety of system configurations.

Priority modes can be changed or reconfigured dynamically at any time during system operation using the operation command words (OCWs), allowing the overall interrupt structure to be defined for a complete system. Note that the CA82C59A is programmed by the system software *as an I/O peripheral.*

The major functional components of the CA82C59A are laid out in the block diagram of Figure 1. Vector data and device programming information are transferred from the system bus to the CA82C59A via the 3-state, bi-directional Data Bus Buffer which is connected to the internal bus of the controller. Control data between the CA82C59A and the CPU, and between master and slave CA82C59A devices, is managed by one of three functional blocks:

- The Read/Write Control block processes CPU initiated reads and writes to the CA82C59A registers
- The Control Logic block receives and generates the signals that control the sequence of events during an interrupt
- The Cascade Control block is used to operate a private bus (CAS_0 - CAS_2) connecting master and slave CA82C59As in those systems having cascaded CA82C59As.

Programming data passed over the system bus is saved in the Initialization and Command Word Registers. Note that the contents of these registers cannot be read back by the CPU.

Peripheral interrupt requests (IR_0 - IR_7) are handled by the functional blocks comprising the Interrupt Request Register (IRR), the Interrupt Mask Register (IMR), the In-Service Register (ISR) and the Priority Decision Logic block. Interrupt requests are received at the IRR, the IMR masks those interrupts which cannot be accepted by the CA82C59A, and the ISR shows those interrupt requests which are currently being processed. These three registers can all be read by the CPU under software control. The Priority Decision Logic block determines which interrupt will be processed next according to a variety of indicators which include the current priority, mode status, current interrupt mask and interrupt service status.

The actual operation of the CA82C59A and its many modes are described in the section following device specifications and characteristics.

9

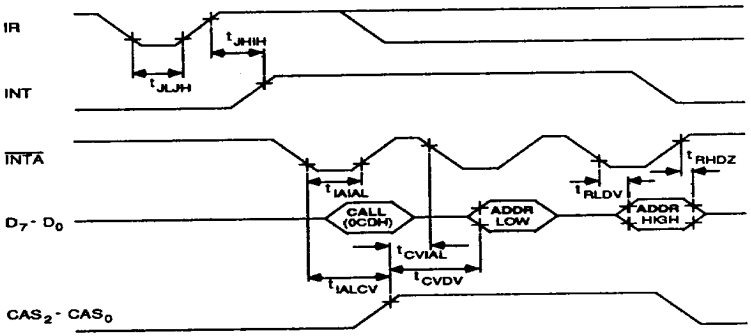
Table 2 : AC CHARACTERISTICS (T_A = –40° to +85°C, V_{DD} = 5V ± 10%)

Symbol	Parameter	Test Conditions	Limits (8 MHz)		Limits (10 MHz)		Units
			Min	Max	Min	Max	
t_{AHDV}	Data valid from stable address	Note 5	-	200	-	160	ns
t_{AHRL}	$A_0/\overline{CS}$ setup to $\overline{RD}/\overline{INTA}\downarrow$		0	-	0	-	ns
t_{AHWL}	$A_0/\overline{CS}$ setup to $\overline{WR}\downarrow$		0	-	0	-	ns
t_{CHCL}	End of *Command* to next *Command* (Not same command type) End of $\overline{INTA}$ sequence to next $\overline{INTA}$ sequence (same as t_{RV2})	Note 1	200	-	160	-	ns
t_{CVDV}	Cascade valid to valid data	Note 5	-	200	-	130	ns
t_{CVIAL}	Cascade setup to second or third $\overline{INTA}\downarrow$ (slave only)	Slave	40	-	30	-	ns
t_{DVWH}	Data setup to $\overline{WR}\uparrow$		160	-	100	-	ns
t_{IAIH}	$\overline{INTA}$ pulse width HIGH	INTA Sequence	160	-	100	-	ns
t_{IAIL}	$\overline{INTA}$ pulse width LOW		160	-	100	-	ns
t_{IALCV}	Cascade valid from first $\overline{INTA}\downarrow$ (master only)	Note 5	-	260	-	160	ns
t_{JHIH}	Interrupt output delay	Note 5	-	200	-	120	ns
t_{JLJH}	Interrupt request width (LOW)[1]	Note 2	100	-	100	-	ns
t_{RHAX}	$A_0/\overline{CS}$ hold after $\overline{RD}/\overline{INTA}\uparrow$		0	-	0	-	ns
t_{RHDZ}	Data float after $\overline{RD}/\overline{INTA}\uparrow$	Note 6	10	85	10	65	ns
t_{RHEH}	Enable inactive from $\overline{RD}\uparrow$ or $\overline{INTA}\uparrow$	Note 5	-	50	-	50	ns
t_{RHRL}	End of $\overline{RD}$ to next $\overline{RD}$ End of $\overline{INTA}$ to next $\overline{INTA}$ within an INTA sequence only		160	-	100	-	ns
t_{RLDV}	Data valid from $\overline{RD}/\overline{INTA}\downarrow$	Note 5	-	120	-	95	ns
t_{RLEL}	Enable active from $\overline{RD}\downarrow$ or $\overline{INTA}\downarrow$	Note 5	-	100	-	70	ns
t_{RLRH}	$\overline{RD}$ pulse width		160	-	100	-	ns
t_{RV1}	Command recovery time	Note 3	200	-	100	-	ns
t_{RV2}	$\overline{INTA}$ recovery time	Note 4	200	-	100	-	ns
t_{WHAX}	$A_0/\overline{CS}$ hold after $\overline{WR}\uparrow$		0	-	0	-	ns
t_{WHDX}	Data hold after $\overline{WR}\uparrow$		0	-	0	-	ns
t_{WHWL}	End of $\overline{WR}$ to next $\overline{WR}$		160	-	100	-	ns
t_{WLWH}	$\overline{WR}$ pulse width		160	-	100	-	ns

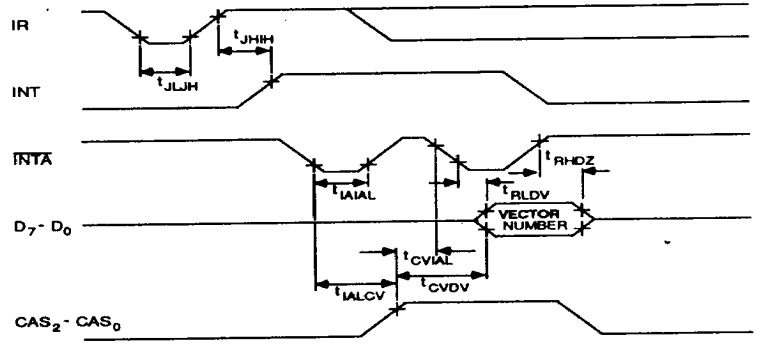
Notes:
1. The time to move $\overline{INTA}$ to/from command (read/write).
2. The time to clear the input latch in edge-triggered mode.
3. The time to move from read to write operation.
4. The time to move to the next $\overline{INTA}$ operation.
5. See Figure 5, *Note 5* for load circuit values.
6. See Figure 5, *Note 6* for load circuit values

Figure 3 : TIMING DIAGRAMS

a) Interrupt Cycle (CALL Mode)

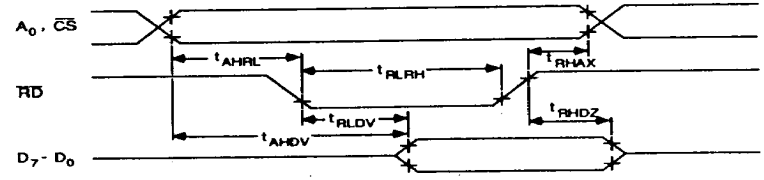


b) Interrupt Cycle (VECTOR Mode)

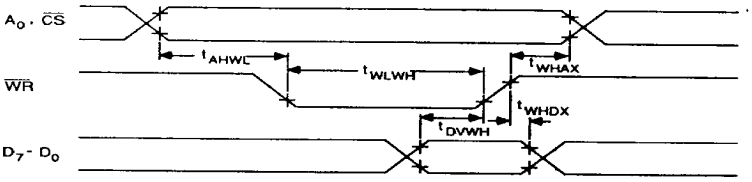


Note that IR input should remain at a high level until the leading edge of the first $\overline{INTA}$ pulse.

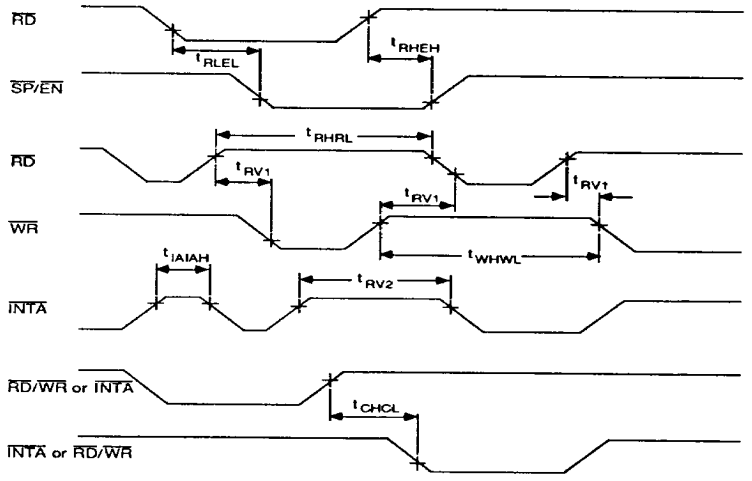
c) Read Cycle

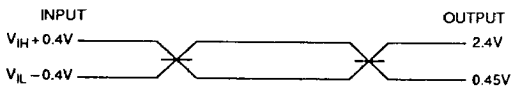


d) Write Cycle



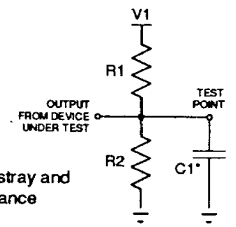
e) Other Timing





AC Testing: All input signals must switch between $V_{IL} - 0.4V$ and $V_{IH} + 0.4V$. Input rise and fall times must be ≤ 15 ns. All timing measurements are made at 2.4V and 0.45V.

Figure 4 : AC TESTING I/O WAVEFORMS



*Includes stray and jig capacitance

Note	V1	R1	R2	C1
5	1.7V	523Ω	open	100pf
6	4.5V	1.8kΩ	1.8kΩ	30pf

Figure 5 : AC TESTING LOAD CIRCUIT

Table 3 : DC CHARACTERISTICS (T_A = –40° to +85°C, V_{DD} = 5V ± 10%)

Symbol	Parameter	Test Conditions	Limit		Units
			Min	Max	
I_{DD}	V_{DD} supply current		-	10	mA
I_{DDSB}	Standby power supply current	V_{DD} = 5.5v, V_{IN} = V_{DD} or Gnd Outputs Open, All IR = V_{DD}	-	10	µA
I_{LI}	Input leakage current	0v ≤ V_{IN} ≤ V_{DD}	–1.0	+1.0	µA
I_{LIR}	IR input load current	V_{IN} = 0v V_{IN} = V_{DD}, All temp ranges	- -	–300 10	µA µA
I_{LOL}	Output leakage current	0v ≤ V_{OUT} ≤ V_{DD}	–10.0	+10.0	µA
V_{IH}	Input HIGH voltage		2.2	V_{DD} + 0.5	V
V_{IL}	Input LOW voltage		–0.5	0.8	V
V_{OH}	Output HIGH voltage	I_{OH} = -2.5 mA I_{OH} = -100 µA	3.0 V_{DD} – 0.4	- -	V V
V_{OL}	Output LOW voltage	I_{OL} = +2.5 mA	-	0.4	V

Table 4 : CAPACITANCE (T_A = 25°C, V_{DD} = V_{SS} = 0V)

Symbol	Parameter	Test Conditions	Limits			Units
			Min	Typ	Max	
C_{IN}	Input capacitance	Freq = 1 MHz	-	7	-	pF
$C_{I/O}$	I/O capacitance	Unmeasured pins are		20	-	pF
C_{OUT}	Output capacitance	returned to Gnd		15	-	pF

Table 5 : RECOMMENDED OPERATING CONDITIONS

Operating Voltage Range		+4.0 to +6.0 Volts
Operating Temperature Range	Commercial	0°C to +70°C
	Industrial	–40°C to +85°C

Table 6 : ABSOLUTE MAXIMUM RATINGS

Power Supply Voltage (V_{DD})	–0.5 to +7.0 Volts
Input (V_{IN}) or I/O Voltage Applied	V_{SS} – 0.5 to V_{DD} + 0.5 Volts
Output (V_{OUT}) Voltage Applied	V_{SS} – 0.5 to V_{DD} + 0.5 Volts
Maximum Power Dissipation	1 Watt
Storage Temperature Range	–65°C to +150°C

Stresses beyond those listed above may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

OPERATIONAL DESCRIPTION

The CA82C59A is designed to operate in one of two mutually exclusive modes, selected according to the type of system processor used; **Call Mode** for 8080/85 type processors, and **Vector Mode** for 8088/86 and 80286/386 type processors. The major difference between these two modes is the way in which interrupt service routine address data is passed to the system CPU. Unless specifically programmed to the contrary, the CA82C59A defaults to the CALL Mode of operation, (see section on Programming).

Call Mode

In CALL mode, the *interrupt service routine address* is passed in two steps, (first the lower byte of the address, followed by the upper byte), in response to three Interrupt Acknowledge ($\overline{INTA}$) signals sent by the CPU to the CA82C59A. In a system containing a single Interrupt Controller, the sequence of steps to respond to a peripheral interrupt request is outlined below, and shown graphically in Figure 6. The interrupt service routine addresses are loaded into the CA82C59A during the initialization procedures.

Step	Event Sequence
1	One or more interrupt request lines (IR_0 - IR_7) are raised HIGH, setting corresponding IRR bits.
2	The requests are evaluated by the CA82C59A, and if their priority is high enough, or if they are not masked, an INT signal is sent to the CPU.
3	The CPU acknowledges the INT with an interrupt acknowledge ($\overline{INTA}$).
4	On receipt of the $\overline{INTA}$, the CA82C59A sets the highest priority ISR bit, and resets the corresponding IRR bit. In addition, the CA82C59A sends a CALL instruction (0CDH) to the CPU via the data bus.
5	The CALL instruction causes the CPU to send two more $\overline{INTA}$ signals to the CA82C59A.
6	On receipt of the first of these two $\overline{INTA}$ signals, the CA82C59A sends the low order 8-bit address byte to the CPU via the data bus. On receipt of the second $\overline{INTA}$, the high order address byte is sent to the CPU.
7	This completes the 3-byte CALL instruction procedure. The ISR bit is reset at the end of the interrupt sequence except in the Automatic EOI mode, where the ISR bit is reset automatically at the end of the third $\overline{INTA}$.

Vector Mode

In VECTOR mode, the interrupt service routine address is calculated by the CPU from a one byte *interrupt vector* supplied by the CA82C59A. The significant bits T_{7-3} of the interrupt vectors are loaded into the CA82C59A during the initialization procedures.

Note that no data is transferred by the CA82C59A to the CPU after the first $\overline{INTA}$ signal (the CA82C59A data bus buffers are disabled). It is similar to the CALL mode in that this cycle is used for internal operations that freeze the state of the interrupts for priority resolution or, in cascaded mode; to issue the interrupt code on the cascade lines (CAS_{0-2}) at the end of this cycle.

The sequence of steps that occur to respond to a peripheral interrupt request in Vector mode are outlined below and illustrated in Figure 9.

Step	Event Sequence
1	One or more interrupt request lines (IR0 - IR7) are raised HIGH, setting corresponding IRR bits.
2	The requests are evaluated by the CA82C59A, and if their priority is high enough, or if they are not masked, an INT signal is sent to the CPU.
3	The CPU acknowledges the INT with an interrupt acknowledge ($\overline{INTA}$).
4	After receipt of the first $\overline{INTA}$ signal from the CPU, the CA82C59A sets the highest priority ISR bit and resets the corresponding IRR bit. The CA82C59A data bus buffer is *not* active during this cycle (high impedance state).
5	Following receipt of the second $\overline{INTA}$ signal generated by the CPU, the CA82C59A sends an 8-bit *interrupt vector* to the CPU via the data bus.
6	This completes the 1-byte VECTOR mode procedure. In the Automatic End-of-Interrupt (AEOI) mode, the ISR bit is reset at the end of the second $\overline{INTA}$. In other EOI modes the ISR bit remains set until an appropriate EOI command is received after the end of the interrupt sequence.

The interrupt sequence procedures when several CA82C59As are cascaded together is shown for both CALL and VECTOR modes in Figures 8 and 11 respectively.

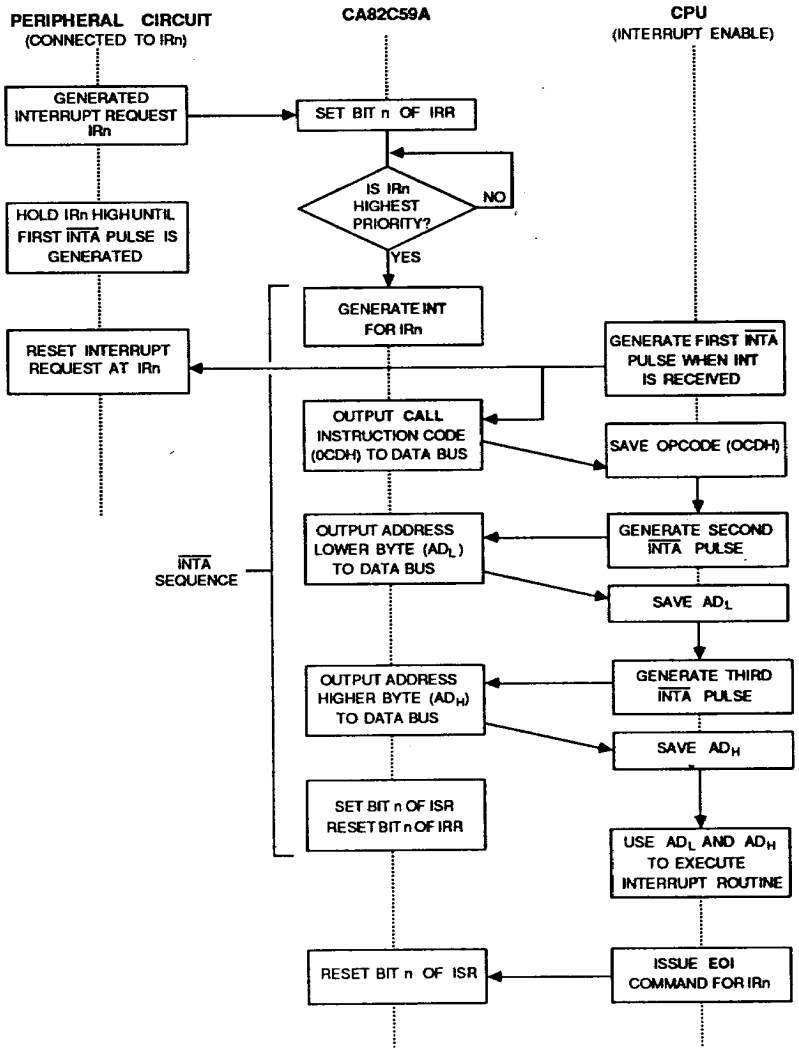


Figure 6 : CALL MODE OPERATION (Single CA82C59A Systems)

9

CONTENTS OF FIRST INTERRUPT VECTOR BYTE

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CALL CODE	1	1	0	0	1	1	0	1

The lower address of the appropriate service routine is enabled onto the data bus during the second INTA pulse.

When the *Interval = 4*, bits A_5 - A_7 are programmed, and A_0 - A_4 are inserted automatically by the CA82C59A.

When the *Interval = 8*, bits A_6 and A_7 only are programmed, with A_0 - A_5 inserted automatically by the CA82C59A.

CONTENTS OF SECOND INTERRUPT VECTOR BYTE

IR	INTERVAL = 4							
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
7	A_7	A_6	A_5	1	1	1	0	0
6	A_7	A_6	A_5	1	1	0	0	0
5	A_7	A_6	A_5	1	0	1	0	0
4	A_7	A_6	A_5	1	0	0	0	0
3	A_7	A_6	A_5	0	1	1	0	0
2	A_7	A_6	A_5	0	1	0	0	0
1	A_7	A_6	A_5	0	0	1	0	0
0	A_7	A_6	A_5	0	0	0	0	0

IR	INTERVAL = 8							
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
7	A_7	A_6	1	1	1	0	0	0
6	A_7	A_6	1	1	0	0	0	0
5	A_7	A_6	1	0	1	0	0	0
4	A_7	A_6	1	0	0	0	0	0
3	A_7	A_6	0	1	1	0	0	0
2	A_7	A_6	0	1	0	0	0	0
1	A_7	A_6	0	0	1	0	0	0
0	A_7	A_6	0	0	0	0	0	0

During the third INTA pulse, the higher address of the appropriate service routine is enabled onto the bus. This address was intially programmed as byte 2 of the initialization sequence (A_8 - A_{15}).

CONTENTS OF THIRD INTERRUPT VECTOR BYTE

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8

Figure 7 : CALL MODE ADDRESS BYTE SEQUENCE

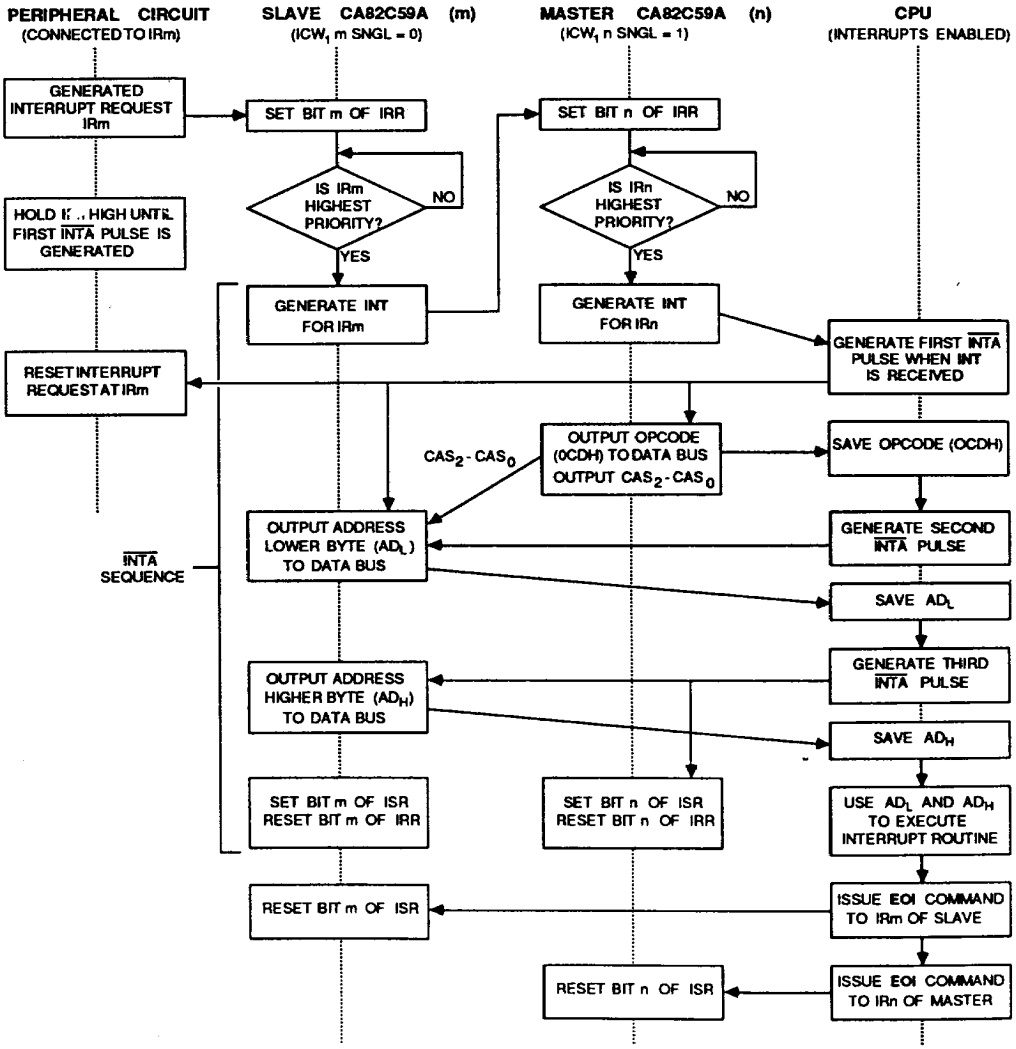


Figure 8 : CALL MODE OPERATION (Cascaded CA82C59A Systems)

9

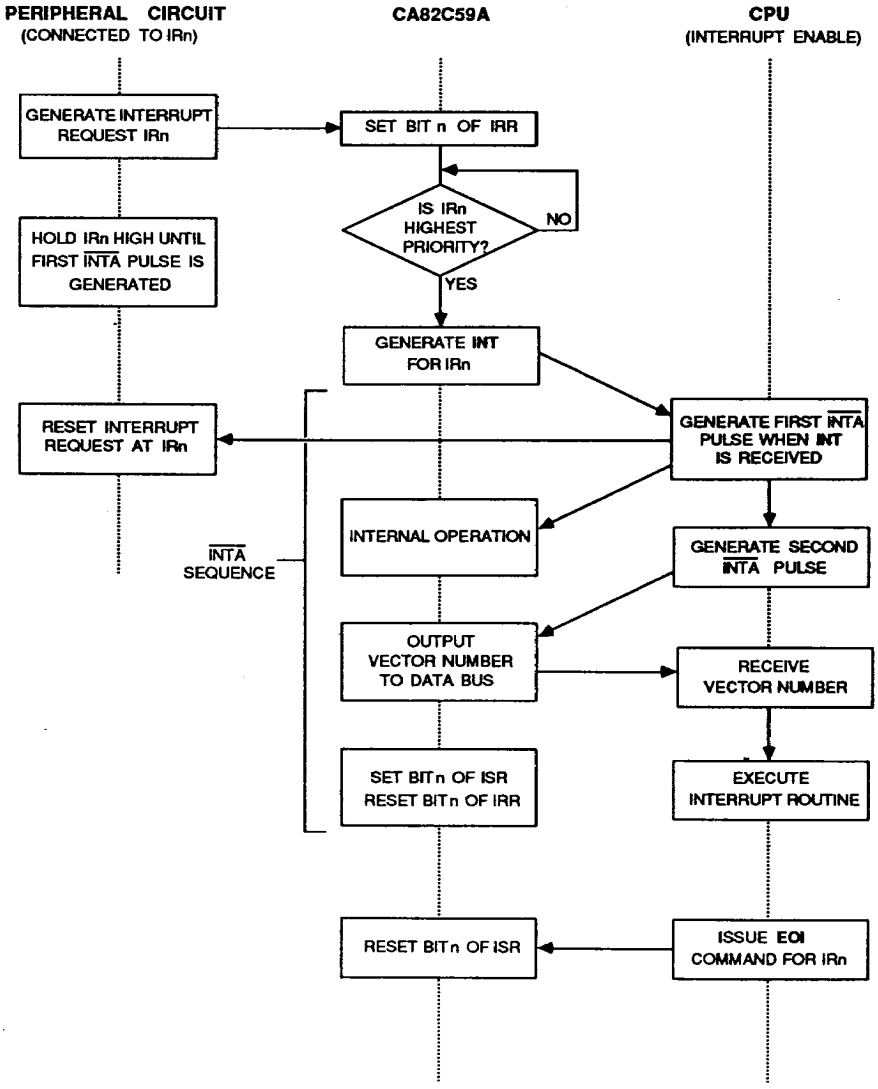


Figure 9 : VECTOR MODE OPERATION (Single CA82C59A Systems)

CONTENTS OF FIRST INTERRUPT VECTOR BYTE

IR	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
7	T_7	T_6	T_5	T_4	T_3	1	1	1
6	T_7	T_6	T_5	T_4	T_3	1	1	0
5	T_7	T_6	T_5	T_4	T_3	1	0	1
4	T_7	T_6	T_5	T_4	T_3	1	0	0
3	T_7	T_6	T_5	T_4	T_3	0	1	1
2	T_7	T_6	T_5	T_4	T_3	0	1	0
1	T_7	T_6	T_5	T_4	T_3	0	0	1
0	T_7	T_6	T_5	T_4	T_3	0	0	0

The value T_7 to T_3 is programmed during byte 2 of the initialization (ICW_2). During the second INTA pulse, the interrupt vector of the appropriate service routine is enabled onto the bus. The low order three bits are supplied by the CA82C59A according to the IR input causing the interrupt.

Figure 10 : VECTOR MODE ADDRESS BYTE

9

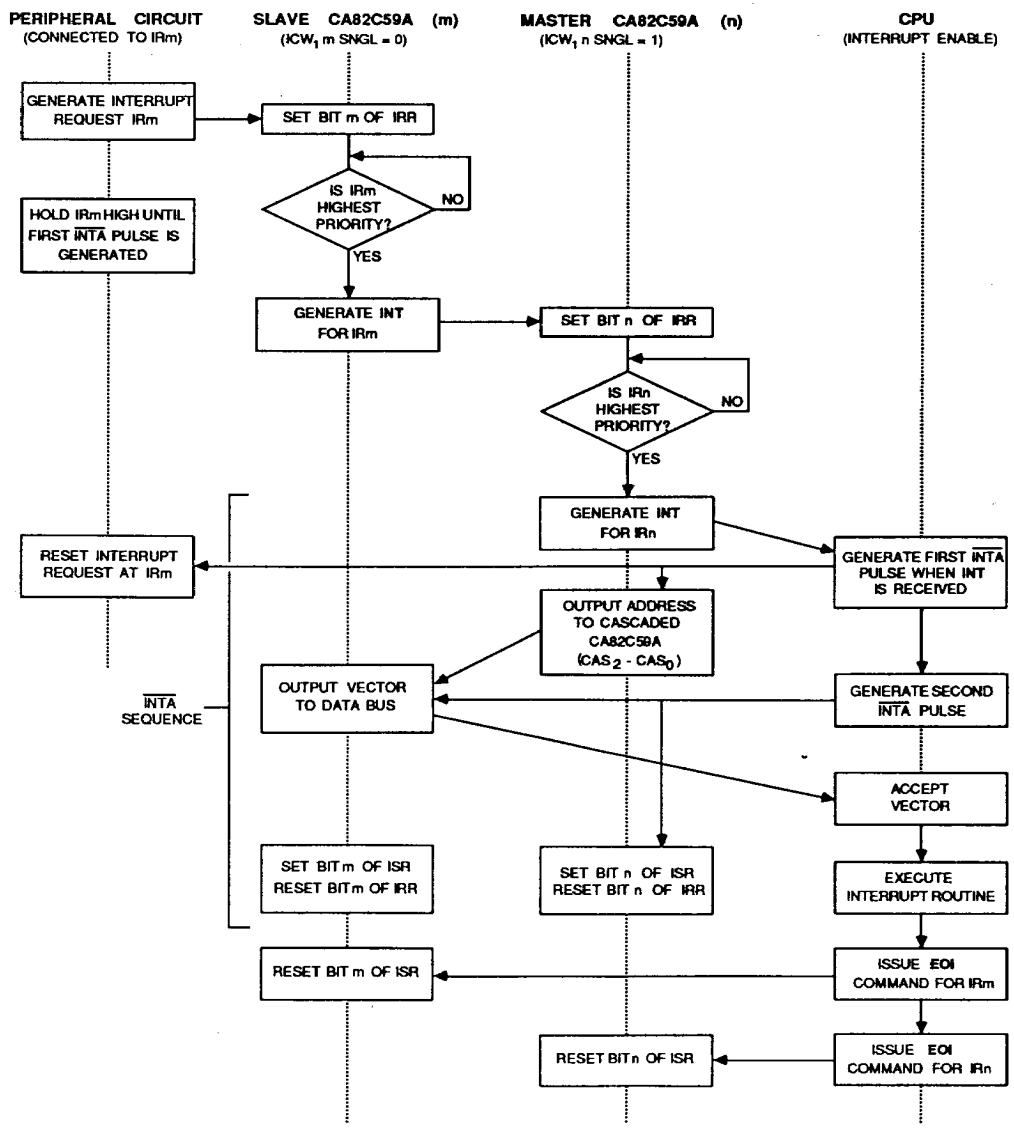


Figure 11 : VECTOR MODE OPERATION (Cascaded CA82C59A Systems)

REGISTERS

The CA82C59A contains a number of registers, used to keep track of interrupts which are being serviced, or pending, as well as those which are masked. These registers are described in Table 7. They can be written to using the *command word* structure, or in the case of IRR, are set by external peripheral devices requesting interrupt service. The contents of all registers can be read by the CPU for status updates (see Table 8).

Table 7 : CA82C59A REGISTERS

Symbol	Name	Function
IMR	Interrupt Mask Register	An 8 - bit wide register that contains the interrupt request lines which are masked.
IRR	Interrupt Request Register	An 8 - bit wide register that contains the levels requesting an interrupt to be acknowledged. The highest request level is RESET from the IRR when an interrupt is acknowledged, (not affected by IMR).
ISR	In-Service Register	An 8 - bit wide register that contains the priority levels which are currently being serviced. The ISR is updated when an *End of Interrupt* Command (EOI) is received.

Table 8 : REGISTER READ/WRITE OPERATIONS

Operations		Other	Bit Programming			
CA82C59A	CPU	Conditions	CS	RD	WR	A_0
IRR to Data Bus	IRR Read	IRR set by OCW_3	0	0	1	0
ISR to Data Bus	ISR Read	ISR set by OCW_3				
Polling data to Data Bus	Polling	Polling data is read instead of IRR and ISR				
IMR to Data Bus	IMR Read		0	0	1	1
Data Bus to ICW_1 Reg.	ICW_1 Write	Set ICW_1 ($D_4 = 1$)	0	1	0	0
Data Bus to OCW_2 Reg.	OCW_2 Write	Set OCW_2 (D_4, $D_3 = 0$)				
Data Bus to OCW_3 Reg.	OCW_3 Write	Set OCW_3 ($D_4 = 0$, $D_3 = 1$)				
Data Bus to ICW_2 Reg.	ICW_2 Write	Refer to section on Control	0	1	0	1
Data Bus to ICW_3 Reg.	ICW_3 Write	Words for ICW_2 - ICW_4				
Data Bus to ICW_4 Reg.	ICW_4 Write	writing procedure				
Data Bus to IMR	OCW_1 Write	After initialization				
Data Bus set to			0	1	1	x
High Impedance State			1	x	x	x
Illegal State			0	0	0	x

9

PROGRAMMING

The CA82C59A is initialized and programmed with special command words issued by the CPU. These commands fall into two major categories: *Initialization Command Words* (ICW_1 - ICW_4), and *Operational Command Words* (OCW_1 - OCW_3). Initialization commands are used to bring the CA82C59A to a known state when the system is first activated, or after a system restart.

Operational commands are used once the CA82C59A is in operation (and *after* it has been initialized), to set, or alter specific interrupt program modes. The format and use of these two command types is described below.

Initialization Commands

The CA82C59A is initialized by a sequence of 2 to 4 command words (ICWs), where the actual number of commands sent depends on the system configuration, and the initial operating modes to be programmed. Note that *each* CA82C59A in the system *must* be initialized before operations begin in earnest (Figure 13).

The initialization sequence is started when the CPU sends $A_0 = 0$ and ICW_1 with $D_4 = 1$ (Figure 12). During initialization, the events below occur automatically:

- Edge sense circuit is reset. Thus, after initialization, an interrupt request must make a LOW-to-HIGH transition to be recognized.
- IMR is cleared.
- The priority of IR_7 is set to 7 (the lowest priority).
- Special Mask Mode is reset.
- Status read is set to IRR.
- If SNGL bit of $ICW_1 = 1$, then ICW_3 must be programmed.
- If IC_4 bit of $ICW_1 = 0$, then functions selected in ICW_4 are reset: Non-buffered Mode, no Automatic EOI, Call Mode operation.
- If $IC_4 = 1$, then CA82C59A will expect ICW_4.

Bit Definitions (ICW_1, ICW_2)

- IC_4 Set if ICW_4 is to be issued. This bit *must* be set for systems operating in Vector Mode.
- SNGL Set if this CA82C59A is not cascaded to other CA82C59As in the system (ICW_3 not issued). When CA82C59As are cascaded, SNGL is reset and ICW_3 is issued.
- ADI CALL Address Interval. If ADI = 1, then interval = 4. If ADI = 0, then interval = 8.
- LTIM Level Trigger Mode. If LTIM = 1, edge detect logic on the IR inputs is disabled, and the CA82C59A operates in level triggered mode.
- $A_{5\text{-}15}$ Service routine *Page Starting Address* (Call Mode). In a single CA82C59A system, the 8 interrupt request levels generate CALLs to 8 equally spaced locations in memory. These are spaced at intervals of either 4 or 8 memory locations according to the ADI value. Thus, the service routines associated with each CA82C59A in the system occupy pages of 32 or 64 bytes respectively.

  Bits A_0 - A_4 are automatically inserted to give an address length of 2 bytes (A_0 - A_{15}).

  Note that the 8-byte interval is compatible with CA80C85B restart instructions.
- $A_{11\text{-}15}$ Service routine *Vector Address Byte*. In the vector mode, bits A_{11} - A_{15} are inserted in the five most significant places of the vector byte. The three least significant bits are inserted by the CA82C59A according to the interrupt request level. The ADI and A_5 - A_{10} bits are ignored.

Bit Definitions (ICW_3)

This word is used only when SNGL = 0 in ICW_1, at which time the contents then depend on whether it is being sent to a master CA82C59A, or a slave device.

- **Master Mode:** Sent to the master CA82C59A, each bit of ICW_3 represents a potential slave device connected to an IR input. If a slave exists, the corresponding bit in ICW_3 is set. Where a slave is not attached to an IR input of the master, the corresponding bit is reset.

  In operation, the master outputs byte 1 of the interrupt sequence to the bus, then enables the appropriate slave (via the cascade bus $CAS_{0\text{-}2}$) to output bytes 2 and 3 (Call Mode) or byte 2 only (Vector Mode).
- **Slave Mode:** When sent to a slave CA82C59A, bits $ID_{0\text{-}2}$ contain the slave address on the cascade bus. Each slave device in the system *must* be initialized with a unique address. Remaining bits are not used.

  In operation, the slave compares the cascade input to its 3-bit address and if they match, outputs byte 2 and 3 (Call Mode), or byte 2 only (Vector Mode) to the bus.

Bit Definitions (ICW_4)

This word is used only when bit IC_4 in ICW_1 is set. Note that only five bits are used.

- µPM Microprocessor System Mode: µPM = 0 for Call Mode, µPM = 1 for Vector Mode.
- AEOI This bit is set if the *Automatic End of Interrupt Mode* is to be programmed.

- **M/S** When Buffered Mode is selected, the M/S bit is used to determine the master/slave programming. That is, M/S = 1 indicates the device is a master, while M/S = 0 indicates a slave. If the BUF bit is not set, M/S is not used.
- **BUF** Buffered Mode is programmed by setting BUF = 1. In buffered mode, the output pin SP/EN becomes an enable output, and the M/S bit determines whether the device is a master or a slave.
- **SFNM** Special Fully Nested Mode is programmed by setting SFNM = 1.

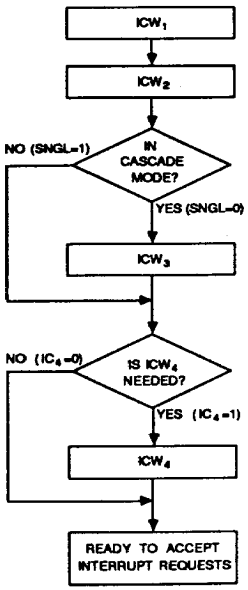


Figure 12 : INITIALIZATION FLOW CHART

Operational Commands

Once the CA82C59A has been initialized, it can accept and process interrupt requests received on its IR input lines. Interrupts are processed according to the modes programmed during the initialization process. A number of commands, sent to the CA82C59A from the CPU, allow the programmed modes or the interrupt request priorities to be changed *on the fly* during operation. These commands are described in the sections following (and in Figure 14).

Bit Definitions (OCW_1)

OCW_1 is used to set and clear mask bits in the IMR, thus enabling or disabling specific IR inputs. In the *Special Mask Mode*, the ISR is also masked.

- M_{0-7} Bits M_{0-7} correspond to the 8 IR inputs. If bit M_n = 1, the IR_n input is disabled. If M_n = 0, the IR_n input is enabled.

Bit Definitions (OCW_2)

OCW_2 is used to program the different End of Interrupt (EOI) Modes, and alter the interrupt request priorities.

- L_{0-2} These bits designate the interrupt level to be acted upon when bit SL = 1 (active).
- EOI The End of Interrupt command is issued by the CPU, rather than by the CA82C59A (in automatic EOI mode). Note that this bit is used in conjunction with bits R and SL to control the interrupt priority assignments and rotations.
- SL Set Interrupt Level bit. This lowest priority interrupt is assigned to the IR input corresponding to the octal value of L_{0-2}.
- R This bit determines if interrupt priority rotation is in effect. R = 1 indicates priorities will be rotated, perhaps combined with other modes.

Bit Definitions (OCW_3)

OCW_3 is used to program the Special Mask Mode, the Polling Mode, and select internal registers to be read by the CPU.

- RIS If RIS = 0, select ISR. If RIS = 1, select IRR.
- RR Read Register bit. If RR = 1, output the contents of the register selected by bit RIS onto the bus. The register selection is retained, so OCW_3 does not have to be reissued in order to read the same register again.
- P If P = 1, the Polling Mode is selected for this CA82C59A. In this mode the CPU will poll for new interrupt requests, rather than having the CA82C59A actively set the CPU *INT* input.
- SMM If *Special Mask Mode* is enabled (ESMM = 1), then SMM = 1 programs the special mask mode, and SMM = 0 clears the special mask mode.
- ESMM If ESMM = 1, then the special mask mode is enabled, and can be set or reset by the SMM bit. If ESMM = 0, the special mask mode is disabled and SMM is ignored.

9

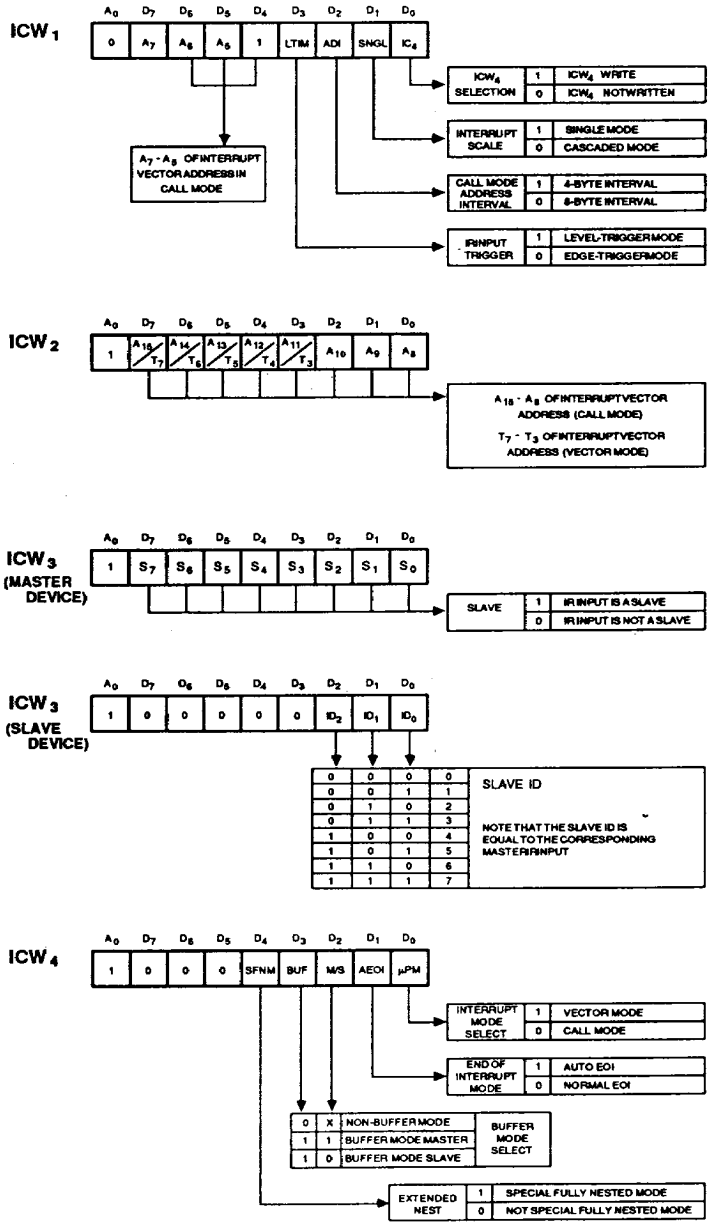


Figure 13 : INITIALIZATION COMMAND WORD FORMAT

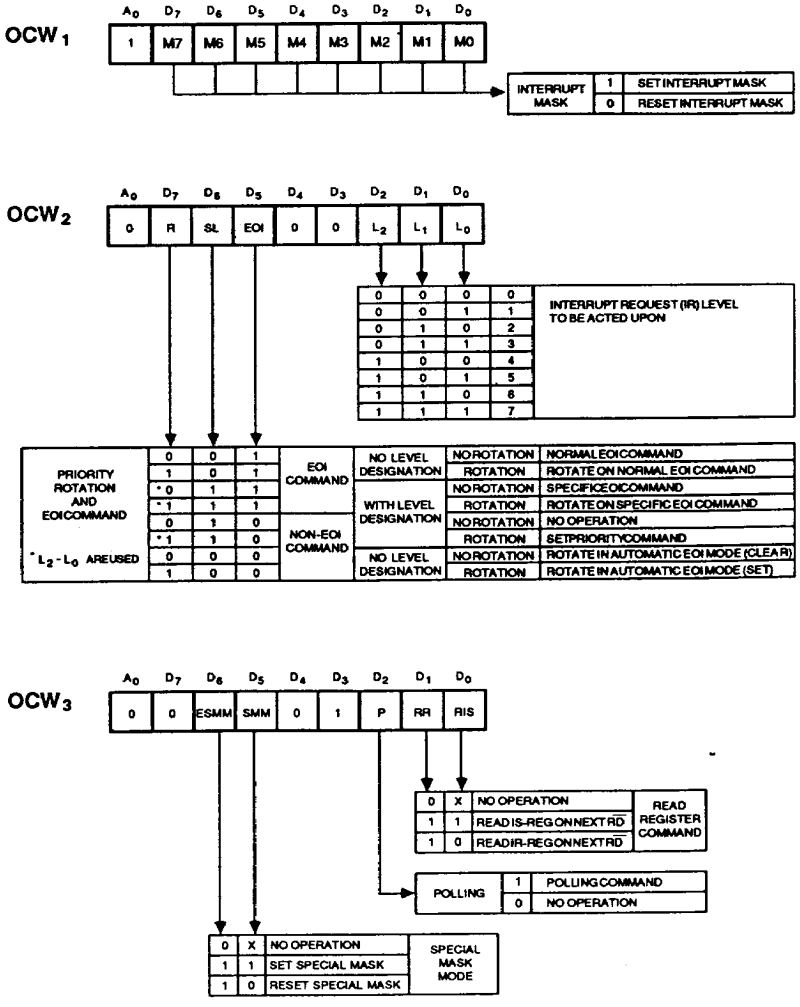


Figure 14 : OPERATIONAL COMMAND WORD FORMAT

9

OPERATIONAL MODES

The CA82C59A can be programmed to operate in a number of different modes which are summarized and described below. Depending on the mode, some are set during initialization, and some during operation (with the *command word* structure).

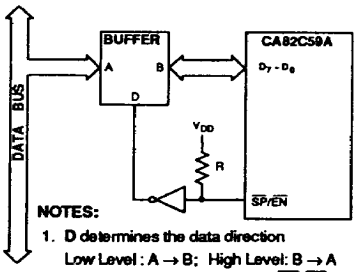
Mode	Location Set
Buffer Mode	BUF, M/S (ICW_4)
Cascaded Mode	SNGL (ICW_1)
• Master Mode	(ICW_3)
• Slave Mode	ID_{0-2} (ICW_3)
End of Interrupt (EOI) Modes	EOI (OCW_2)
• Automatic EOI Mode	AEOI (ICW_4)
• Non-specific EOI	EOI, SL, R (OCW_2)
• Specific EOI	EOI, SL, R (OCW_2)
Nested Modes	
• Fully Nested Mode	default mode
• Special Fully Nested Mode	SFNM (ICW_4)
	AEOI (ICW_4)
Polling Mode	P (OCW_3)
Rotation Modes	
• Automatic Rotation Mode	R, SL, EOI (OCW_2)
• Specific Rotation Mode	R, SL, L_{0-2} (OCW_2)
Special Mask Mode	ESMM, SMM (OCW_3)
System Modes	
• CALL Mode	μPM (ICW_4)
• VECTOR Mode	μPM (ICW_4)
	IC_4 (ICW_1)
Trigger Modes	
• Edge Triggered Mode	LTIM (ICW_1)
• Level Triggered Mode	LTIM (ICW_1)

Buffer Mode

In larger systems the CA82C59A may be required to drive the data bus through a buffer. To handle this situation, the *Buffer Mode* is programmed during initialization (using the BUF and M/S bits in ICW_4).

When in buffer mode, $\overline{SP}/\overline{EN}$ is used to enable the data bus buffers, and determine the direction of data flow through the buffer (Figure 15). This signal is active when the output ports of the CA82C59A are activated.

Note that when *cascaded* CA82C59As are required to be used in the buffer mode, the master/slave selection is done using the M/S bit of ICW_4, (and SNGL bit of ICW_1 is set to 1). M/S is set to 1 for the master mode, 0 for the slave mode.



NOTES:
1. D determines the data direction
   Low Level : A → B; High Level: B → A
2. The CA82C59A is set to input $\overline{SP}/\overline{EN}$ in its initial state, and is pulled up by R to set D to the low level during initialization.

Figure 15 : BUFFER MODE

Cascaded Mode

In systems that contain more than 8 priority interrupt levels, several CA82C59A devices can be easily cascaded together to handle a maximum of 64 interrupt levels. In a cascaded configuration, one CA82C59A serves as a *master*, controlling up to 8 *slaves* (able to handle 8 interrupts each). The master selects the slaves via the cascade local bus (CAS_{0-2}), enabling the corresponding slave to output the interrupt service routine address (or vector) following each of the second and third $\overline{INTA}$ signals (second $\overline{INTA}$ signal only in Vector mode). At the end of the interrupt cycle, the EOI command must be issued *twice*, one for the master, and one for the appropriate slave.

The CAS_{0-2} bus lines are normally held in a LOW state, and activated only for slave inputs (non-slave inputs to the master do not affect the cascade bus). The slave address will be held on the cascade bus from the trailing edge of the first $\overline{INTA}$ signal to the trailing edge of the last $\overline{INTA}$ signal (either second or third depending on the system mode).

Within the system, each slave can be programmed to operate in a different mode (except AEOI), independently of other slave devices. The interrupt output (INT) of each slave is tied to one of the interrupt request lines (IR_{0-7}) of the master device. Unused IR pins on the master device can be connected to other peripheral devices (as in the single standalone mode of operation), or left unconnected.

Note that an address decoder is required to activate the chip select ($\overline{CS}$) input of each CA82C59A in the system.

Master Mode

A CA82C59A operating in the *Master Mode* can be controlling both peripheral interrupts as well as other CA82C59A slave devices. Since both types of inputs are connected to the IR_{0-7} pins, it is necessary to differentiate between the two. This is acccomplished in the initialization control word (ICW_3) sent out to the master. ICW_3 sets the S_n bits corresponding to IR_n pins connected to slaves equal to one (1), while S_m bits corresponding to IR_m pins connected to peripheral inputs are reset to zero.

Peripheral interrupt requests to IR_m pins ($S_m = 0$) are handled by the master as if it were operating singly. That is, the CAS line remain LOW, and the master provides the interrupt or vector as required.

Slave interrupt requests to IR_n pins ($S_n = 1$) are handled as follows; the master sends an interrupt to the CPU if the slave requesting the interrupt has priority. If so, the master outputs the slave address *n* to the CAS bus on the first $\overline{INTA}$ signal, then lets the slave complete the remainder of the interrupt cycle. Note again however, that two EOI commands are required to terminate the sequence, one each for the master and slave.

Slave Mode

When a slave CA82C59A receives a peripheral interrupt request, and it has no higher interrupt requests pending, the slave sends an interrupt request to the master via its INT output. This interrupt request is passed by the master to the CPU, which then initiates the interrupt cycle, in turn causing the master to output the slave's address on the CAS bus. Each slave in the system continuously monitors the CAS bus, comparing the addresses thereon until a match is found with its own address. When the slave initiating the interrupt request finds an address match, it completes the interrupt sequence as though it were a single CA82C59A.

Note: Since the master holds the CAS bus LOW (corresponding to CAS address 0) when processing peripheral interrupt requests, address 0 should not be used as a slave address unless the system contains the full complement of 8 slaves.

End of Interrupt (EOI) Modes

The *EOI Modes* are used to terminate the *request for interrupt service* sequence, update the ISR register and alter the interrupt priorities. The EOI mode selected depends on the *nesting mode* currently programmed. The options are discussed in the sections following.

Automatic EOI (AEOI) Mode

In *AEOI Mode*, the ISR bit corresponding to the interrupt is set and reset automatically during the final $\overline{INTA}$ signal. This means that the CPU does not have to issue an EOI command at the end of the interrupt routine.

Caution is urged in using AEOI however, as the ISR does not save the *routine currently in service* in this mode. Thus, unless the interrupts are disabled by the interrupt service routine, a stack overflow situation can result from newly generated interrupts (which bypass the priority structure), or from level triggered interrupts.

The Automatic EOI mode is programmed by setting the AEOI bit in $ICW_4 = 1$.

Non-specific (Normal) EOI Mode

When the CA82C59A is operated in the *Fully Nested Mode*, it can easily determine which ISR bit is to be reset at the conclusion of an interrupt sequence. In this case, the non-specific EOI command is used to reset the highest priority level selected from the interrupts in service, (the valid assumption being that the last interrupt level acknowleged and serviced necessarily corresponds to the highest priority ISR bit set).

A *Non-specific EOI Mode* is selected via OCW_2, where EOI = 1, SL = 0 and R = 0). Refer to Figure 14a, c.

Specific EOI Mode

The *Specific EOI Mode* is required when the fully nested (normal) mode is not used, and the CA82C59A is unable to determine the last interrupt level acknowledged (such as might be encountered if the *Special Mask Mode* is programmed). The Specific EOI command identifies the ISR bit (interrupt level) to be reset using bits L_{0-2} of OCW_2, which also has the following bit settings: EOI = 1, SL = 1 and R = 0. Refer to Figure 16b, d.

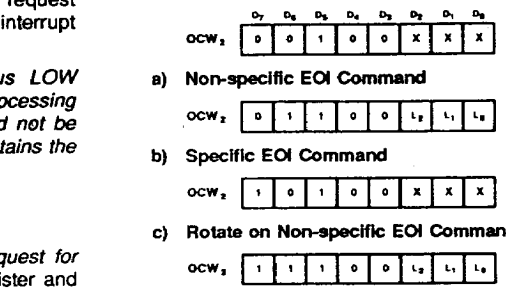


Figure 16 : EOI COMMANDS

9

Mask Modes

The *Mask Modes* are used to selectively enable or disable interrupt requests. This is distinct from the automatic disabling of an interrupt which is in effect while a request from the same interrupt is being serviced.

Normal Mask Mode

Interrupt request lines IR_{0-7} can be individually masked in the Interrupt Mask Register (IMR), using OCW_1. Each bit in IMR masks one interrupt request line if it is set (= 1), with no effect on the other interrupt request lines. Bit 0 masks IR_0, bit 1 masks IR_1 etc.

Special Mask Mode

The *Special Mask Mode* is used to dynamically alter the interrupt priority structure under software control during program execution. In this mode, when a mask bit is set in OCW_1, it disables further interrupts at that level, and enables interrupts from all other levels that are not masked. This includes those interrupts which are lower (as well as higher) in priority.

The *Special Mask Mode* is set by ESMM = 1 and SMM = 1 in OCW_3. To clear this mode, the CPU must issue another OCW_3 with ESMM = 1 and SMM = 0. Setting ESMM = 0 alone has no effect. To correctly enter the *Special Mask Mode*, use the following procedure:

1. CPU reads the ISR
2. CPU writes the ISR data from (1) to the IMR
3. CPU selects *Special Mask Mode* by issuing OCW_3 with ESMM = 1 and SMM = 1.

This procedure ensures that all interrupt requests not currently in service will be enabled.

Note that if IMR is not set equal to ISR when *Special Mask Mode* is selected, bits which may be set in the ISR will be ignored. If a corresponding bit is not set in IMR, that interrupt request may be serviced, causing all interrupts of lower priority to be effectively disabled. This is illustrated in Figure 17.

When the *Special Mask Mode* is selected, the *Specific EOI Mode* must be used to terminate the interrupt sequence, so the CPU can explicitly specify which ISR bit is to be reset.

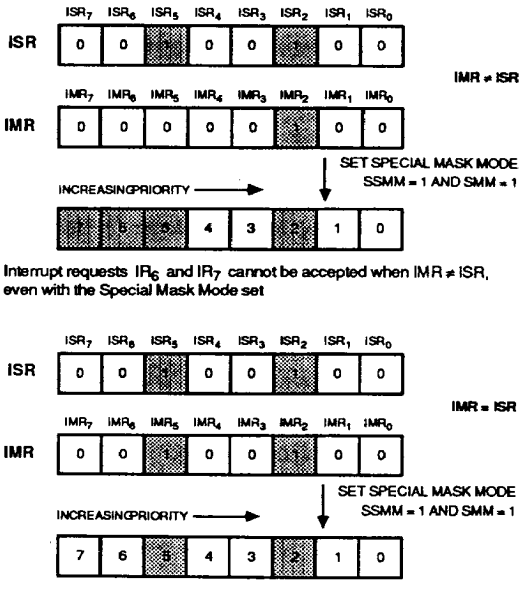


Figure 17 : SPECIAL MASK MODE

Nested Modes

The nesting modes are used to determine, and change, the priority of incoming interrupt requests.

Fully Nested Mode

The default nesting mode, entered automatically after initialization, unless another mode has been programmed. In the *Fully Nested Mode,* the interrupt request priorities are set in descending order from 0 to 7. That is: IR_0 is the highest priority interrupt, IR_7 the lowest.

When an interrupt is acknowledged, the highest priority request is selected, the corresponding ISR bit is set, and the service routine address information is output to the data bus. The ISR bit is reset by the EOI command from the CPU, or automatically if *AEOI Mode* is programmed, at the completion of the interrupt sequence. While the ISR bit is set, all interrupt requests of equal or lower priority are inhibited. Interrupt requests of higher priority will generate an interrupt to the CPU, but whether or not

these will be acknowledged depends on the system software logic. Interrupt priorities can be altered in this mode using one of the *Rotation Modes*.

Note that fully nested interrupt priorities are not necessarily preserved in those systems containing cascaded CA82C59As, as it is possible for interrupts of higher priority than the one being serviced to be ignored. This situation occurs when a slave accepts a peripheral interrupt request (and passes the request to the master). When the master accepts the request, it locks out further interrupts from that slave. Should an interrupt of higher priority come in to the same slave, it will not be recognized until the interrupt being serviced has completed processing. To preserve interrupt priorities in this situation, use the *Special Fully Nested Mode*.

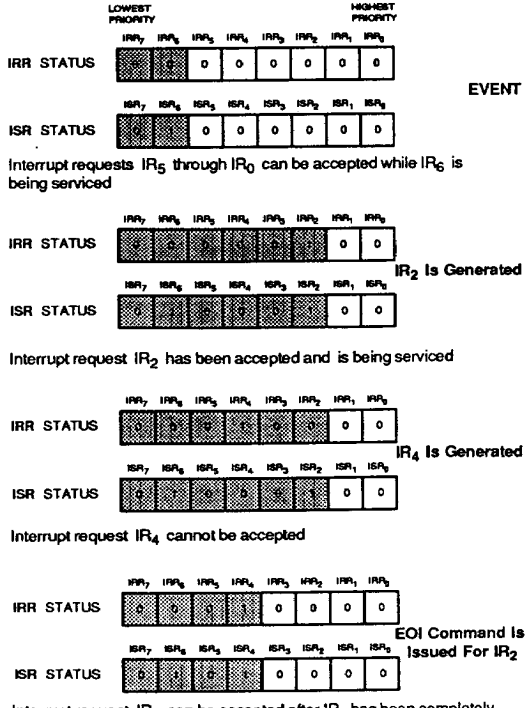


Figure 18 : FULLY NESTED MODE

Special Fully Nested Mode

This mode is very similar to the *Fully Nested Mode*, but is used in systems with cascaded CA82C59s, so as to preserve the interrupt priorities within each slave, as well as within the master. The *Special Fully Nested Mode* is programmed by setting the SFNM bit in the ICW_4 word in both the master and the slave during initialization. This allows interrupt requests of a higher level than the one being serviced to be accepted by the master from the same slave. That is, the slave is not locked out from the priority logic in the master, and higher priority interrupts within the slave will be recognized by the master, which will generate an interrupt to the CPU.

Caution should be exercised in this mode during the End Of Interrupt processing. It is essential that the system software check whether or not the interrupt just serviced was the *only* one from that slave. After the first non-specific EOI has been issued to the slave, the CPU should read the slave's ISR and check that no other bits are set (ISR = 0). Only if the slave ISR is zero, should a non-specific EOI be sent to the master to complete the interrupt sequence. If bits are set in the slave ISR, no EOI should be sent to the master.

Polling Mode

The *Polling Mode* is used to bypass the CA82C59A Interrupt control logic in favour of CPU software control over interrupt request processing. This allows systems to be built up with more than one master CA82C59A, and consequently, the system can contain more than 64 interrupt priority levels (since each master can handle 64 levels individually). In this case, the CPU polls each master looking for the highest priority interrupt request within the realm of each master.

In the *Polling Mode*, the INT outputs of the CA82C59A masters, and the INT input of the CPU are disabled. Interrupt service to individual peripheral devices is accomplished in system software by using the *Poll Command* (Bit 'P' = 1 in OCW_3). When a poll command has been issued, the CA82C59A waits for the CPU to perform a register read. This read is treated by the CA82C59A as an interrupt acknowledge, and it sets the appropriate ISR bit and determines the priority level if there is an interrupt request pending. It then outputs the polling data byte onto the data bus (see Figure 19), including the binary code of the highest priority level requesting service. The CPU then processes the interrupt according to the polling data read, terminating the interrupt sequence with an EOI.

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
OCW_3	INT	0	0	0	0	W_2	W_1	W_0

NOTES:

1. W_0 - W_2 is binary code of highest priority level requesting service.
2. **INT** is equal to one (1) if there is an interrupt.

Figure 19 : POLL COMMAND

9

Rotation Modes

The different *Rotation Modes* allow the interrupt request priorities to be changed either automatically, or under software control. This is particularly useful for situations where there are a number of equal priority devices, or where a particular application may call for a specific priority change.

Automatic Rotation Mode

The *Automatic Rotation Mode* is recommended where there are a number of equal priority devices. In this mode the device is assigned the lowest priority immediately after it has had an interrupt request serviced. Its priority is subsequently increased as other devices have their interrupt requests serviced, and are then rotated to the bottom of the priority list. In the worst case, a device would have to wait for a maximum of seven other device interrupts of equal priority to be serviced *once*, before it was serviced. The effect of rotation on interrupt priority is illustrated in Figure 20.

Automatic Rotation can be activated in one of two ways, both using the command word OCW_2, and both combined with EOI modes:

- Rotation in Non-specific EOI Mode (R = 1, SL = 0 and EOI = 1)
- Rotation in Automatic EOI Mode (R = 1, SL = 0 and EOI = 0). This mode must be cleared by the CPU (accomplished by sending OCW_2 with: R = 0, SL = 0 and EOI = 1.

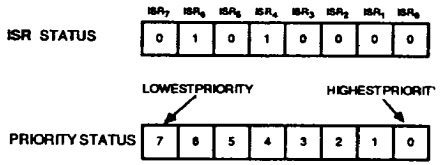
Specific Rotation Mode

The Specific Rotation Mode provides a mechanism to arbitrarily change interrupt priority assignments. This is accomplished by programming the lowest priority interrupt request line (specified by bits L_{0-2} in OCW_2), thereby fixing the other priorities. That is, if IR_4 is programmed as the lowest priority device, then IR_5 will have the highest priority.

Caution: Because this change in priority levels is different from the normal *Fully Nested Mode*, it is *essential* that the user manage the interrupt nesting via the system software.

Specific rotation can be activated in one of two ways, both using the command word OCW_2:

- The *Set Priority* command is issued in OCW_2 with: R = 1, SL = 1 and L_{0-2} equal to the binary code of the lowest priority device.
- As part of the *Specific EOI Mode*, with OCW_2, (Rotate on Specific EOI command) values: R = 1, SL = 1, EOI = 1 and L_{0-2} equal to the binary priority level code of the lowest priority device. When the specific EOI is issued by the CPU, the CA82C59A resets the ISR bit designated by bits L_{0-2} in OCW_2, then rotates the priorities so that the interrupt just reset becomes the lowest priority.



BEFORE ROTATE: IR_4 is the highest priority requiring service

AFTER ROTATE: IR_4 was serviced, all other priorities have been rotated correspondingly

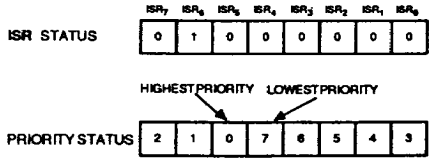


Figure 20 : EFFECT OF ROTATION

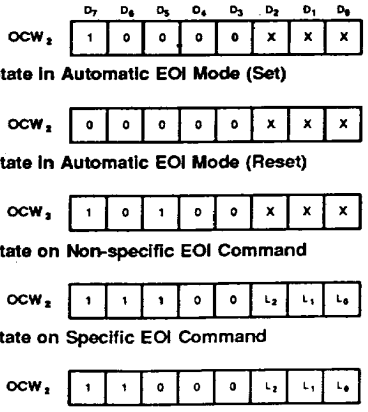


Figure 21 : ROTATION COMMANDS

System Modes

The CA82C59A *must* operate in the system mode that corresponds to the processor type used in the system. *Call Mode* is used for 8085 type systems (and features an interrupt cycle controlled by three $\overline{INTA}$ signals), while *Vector Mode* is used for more sophisticated 8088/86 and 80286/386 type sytems (and features an interrupt cycle controlled by only two $\overline{INTA}$ signals). These modes are described in more detail back in the Operational Description section.

Trigger Modes

In the CA82C59A, the interrupt request lines (IR_{0-7}) can be programmed for either edge or level triggering sensitivity, with the requirement that all IR lines must be in the same mode. That is, all edge triggered, or all level triggered. Figure 23 illustrates the priority cell diagram which shows a conceptual circuit of the level sensitive and edge sensitive input circuitry of the IR lines.

Note that to ensure a valid interrupt request is registered by the CA82C59A, it is essential that the IR input remain HIGH until after the first $\overline{INTA}$ has been received. In both modes, if the IR input goes LOW before this time, the interrupt will be registered as a *default* IR_7 regardless of which IR input initiated the interrupt request. This *default* IR_7 can be used to detect (and subsequently ignore) spurious interrupt signals such as those caused by glitches or noise on the IR input lines. The technique is described below:

- If the IR_7 input is not used, it can be assigned solely to intercepting spurious interrupt requests, invoking a simple service routine that contains a return only, thus effectively ignoring the interrupt.
- If IR_7 is used for a peripheral interrupt, a default IR_7 is detected with the extra step of reading the ISR. A normal IR_7 interrupt causes the corresponding bit to be set in the ISR, while a default IR_7 interrupt does not. It is necessary that the system software keep track of whether or not the IR_7 service routine has been entered. In the event that another IR_7 interrupt occurs before servicing is complete, it will be a default IR_7 interrupt (and should be ignored).

Edge Triggered Mode

Programmed by setting the LTIM bit in ICW_3: LTIM = 0 for *low-to-high-transition* edge triggering. An interrupt request is detected by a rising edge on an IR line. The IR line must remain HIGH until after the falling edge of the first $\overline{INTA}$ signal has been received from the CPU. This is required to latch the corresponding IRR bit. It is recommended that the IR line be kept HIGH to help filter out noise spikes that might cause spurious interrupts. To send the next interrupt request, temporarily lower the IR line, then raise it.

Level Triggered Mode

Programmed by setting the LTIM bit in ICW_3: LTIM = 1 for level triggering. An interrupt request is detected by a HIGH level on an IR line. This HIGH level must be maintained until the falling edge of the first $\overline{INTA}$ signal (as in the edge-triggered mode), to ensure the appropriate IRR bit is set. However, in the level triggered mode, interrupts are requested *as long as* the IR line remains HIGH. Thus, care should be exercised so as to prevent a stack overflow condition in the CPU.

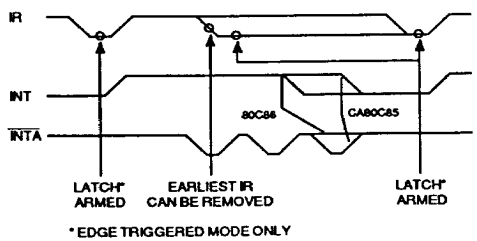


Figure 22 : TRIGGER MODE TIMING

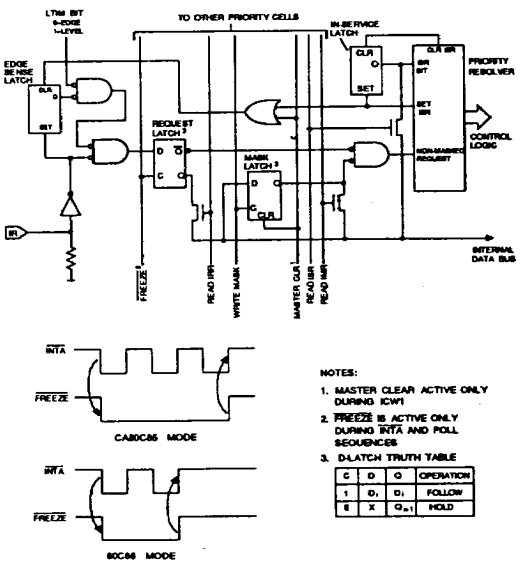


Figure 23 : PRIORITY CELL STRUCTURE

APPLICATION DIAGRAMS

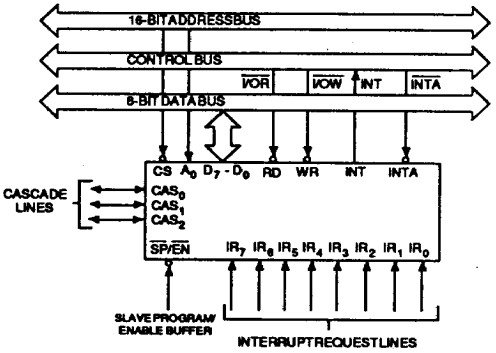
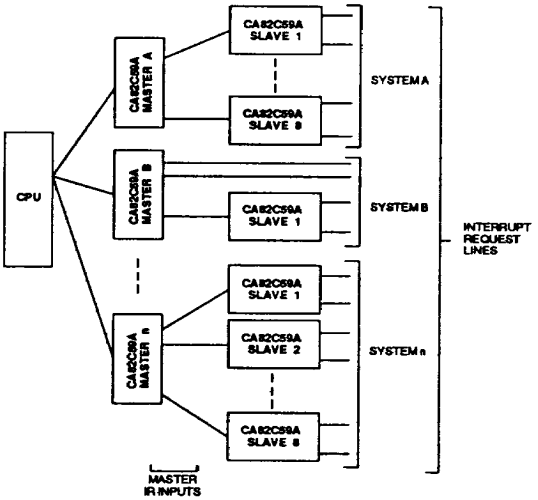


Figure 24 : CA82C59A IN STANDARD SYSTEM CONFIGURATION



NOTES:

1. MASTER CA82C59As IN POLL MODE
2. MAXIMUM OF 64 INPUTS PER SYSTEM
3. TOTAL CAPACITY LIMITED BY CPU

Figure 26 : MULTIPLE CA82C59A MASTERS IN A POLLED SYSTEM

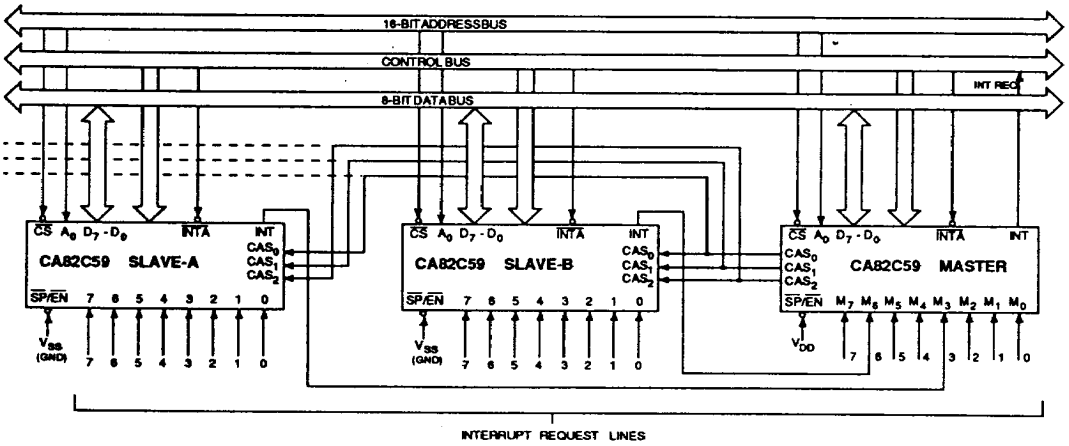


Figure 25 : MULTIPLE CA82C59As IN A CASCADED SYSTEM