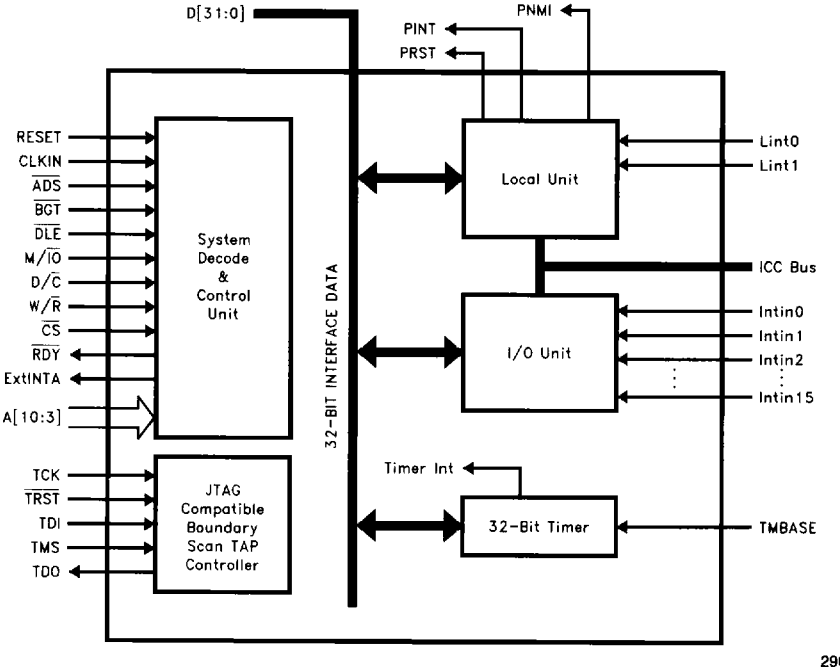
82489DX ADVANCED PROGRAMMABLE INTERRUPT CONTROLLER

82489DX FEATURES OVERVIEW

- Advanced Interrupt Controller for 32-Bit Operating Systems
- Solution for Multiprocessor Interrupt Management
- Dynamic Interrupt Distribution for Load Balancing in MP Systems
- Separate Nibble Bus (Interrupt Controller Communications (ICC) Bus) for Interrupt Messages
- Inter-Processor Interrupts
- Various Addressing Schemes—Broadcast, Fixed, Lowest Priority, etc.
- Compatibility Mode with 8259A
- 32-Bit Internal Registers
- Integrated Timer Support
- 33 MHz Operation
- 132-Lead PQFP Package, Package Type KU
  (See Packaging Specification. Order Number: 240800)

82489DX Block Diagram



290446-1

Refer to Application Note AP-388: 82489DX User's Manual (Order Number 292116) when evaluating your design needs.

October 1993
Order Number: 290446-002

82489DX
Advanced Programmable Interrupt Controller

CONTENTS

CONTENTS

4

CONTENTS

PAGE

CONTENTS

PAGE

CONTENTS

CONTENTS

1.0 INTRODUCTION

The 82489DX Advanced Programmable Interrupt Controller provides multiprocessor interrupt management, providing both static and dynamic symmetrical Interrupt distribution across all processors.

The main function of the 82489DX is to provide interrupt management across all processors. This dynamic interrupt distribution includes routing of the interrupt to the lowest-priority processor. The 82489DX works in systems with multiple I/O subsystems, where each subsystem can have its own set of interrupts. This chip also provides inter-processor interrupts, allowing any processor to interrupt any processor or set of processors. Each 82489DX I/O unit Interrupt Input pin is individually programmable by software as either edge or level triggered. The interrupt vector and interrupt steering information can be specified per pin. A 32-bit wide timer is provided that can be programmed to interrupt the local processor. The timer can be used as a counter to provide a time base to software running on the processor, or to generate time slice interrupts locally to that processor. The 82489DX provides 32-bit software access to its internal registers. Since no 82489DX register reads have any side effects, the 82489DX registers can be aliased to a user read-only page for fast user access (e.g., performance monitoring timers).

The 82489DX supports a generalized naming/addressing scheme that can be tailored by software to fit a variety of system architectures and usage models. It also supports 8259A compatibility by becoming virtually transparent with regard to an externally connected 8259A style controller, making the 8259A visible to software.

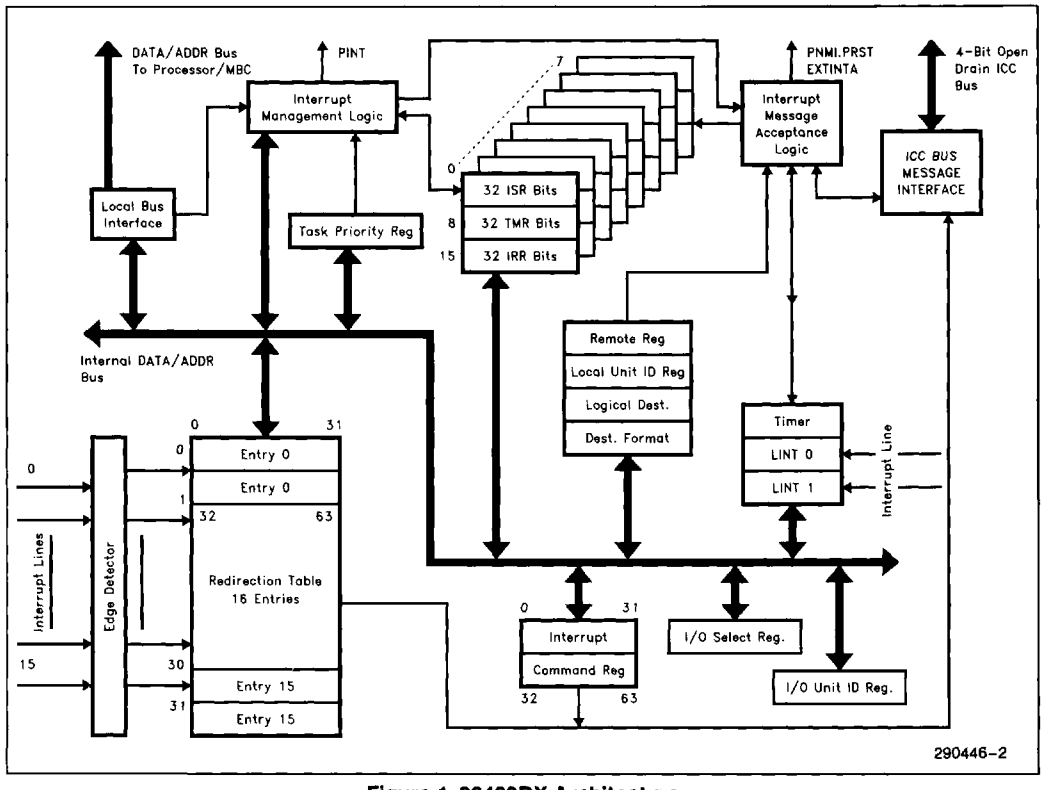


Figure 1. 82489DX Architecture

2.0 FUNCTIONAL OVERVIEW

82489DX Functional Blocks

82489DX contains one Local Unit, one I/O unit and a timer. The ICC bus is used to pass interrupt messages.

ICC BUS

The ICC bus is a 5-wire synchronous bus connecting all 82489DXs (all I/O Untis and all Local Units). The Local Units and I/O Units communicate over this ICC bus. Four of these five wires are used for data transmissions and arbitration, and one wire is a clock.

LOCAL UNIT

The Local Unit contains the necessary intelligence to determine whether or not its processor should accept interrupt messages sent on the ICC bus by other Local Units and I/O Units. The Local Unit also provides local pending of interrupts, nesting and masking of interrupts, and handles all interactions with its local processor such as the INT/INTA/EOI protocol. The Local Unit further provides inter-processor interrupt functionality and a timer to its local processor. The interface of a processor to its 82489DX Local Unit is identical for every processor.

I/O UNIT

The I/O Unit provides the interrupt input pins on which I/O devices inject interrupts into the system in the form of an edge or a level. The I/O unit also contains a Redirection Table for the interrupt input pins. Each entry in the Redirection Table can be individually programmed to indicate whether an interrupt on the pin is recognized as either an edge or a level; what vector and also what priority the interrupt has; and which of all possible processors should service the interrupt and how to select that processor (statically or dynamically). The information in the table is used to send interrupt messages to all 82489DX Units via the ICC bus.

TIMER

The 82489DX provides a 32-bit wide timer that can be programmed to interrupt the local processor. The timer can be used as a counter to provide a time-base to software running on the processor, or to generate time-slice interrupts local to that processor.

3.0 PIN DESCRIPTION

The 82489DX pin description is organized in a small number of functional groups. Pin definitions and protocols have been designed to minimize interface issues. In particular, they support the notion of independently controlled address and data phases. The primary host interface is synchronous in nature.

In the following pin definition table if the signal name has (__) over it, the signal is in its active state when it has a low level. The signal direction column identifies output only signals as a continuous drive (O), tristate (T/S), or open drain (O/D). All bi-directional (BI-D) signals have tri-stating outputs.

4

Pin Definition Table

Symbol	Pin No.	Type	Function
SYSTEM PINS			
RESET	65	I	The **RESET INPUT** forces 82489DX to enter its initial state. The 82489DX Local Unit in turn asserts it PRST (Processor Reset) output. All tri-state outputs remain in high impedance until explicitly enabled.
ExtINTA	41	O	The **EXTERNAL INTERRUPT ACKNOWLEDGE** output is asserted (high) when an external interrupt controller (e.g., 8259) is expected to respond to the current INTA cycle. If deasserted (low), 82489DX will respond, and the INTA cycle must not be delivered to the external controller.
CLKIN	57	I	**CLOCK INPUT** provides reference timing for most of the bus signals.
$\overline{\text{TRST}}$	56	I	**TEST RESET** is the JTAG compatible boundary scan TAP controller reset pin. A weak pull-up keeps the pin high if not driven.
TCK	55	I	**TEST CLOCK** is the clock input for the JTAG compatible boundary scan controller and latches.
TDI	53	I	**TEST DATA INPUT** is the test data input pin for the JTAG compatible boundary scan chain and TAP controller. A weak pull-up keeps this pin high if not driven.
TDO	52	O	**TEST DATA OUTPUT** is the test data output for the JTAG compatible boundary scan chain.
TMS	54	I	**TEST MODE SELECT** is the test mode select pin for the JTAG boundary scan TAP controller. A weak pull-up keeps this pin high if not driven.
TIMER PIN			
TMBASE	59	I	The **TIME BASE** input provides a standard frequency that is only used by the 82489DX timer and that is independent of the system clock.
INTERRUPT PINS			
INTIN[15:0]	82–97	I	These 16 **INTERRUPT INPUT** pins accept edge or level sensitive interrupt requests from I/O or other devices. The pin numbers are specified respectively. INTIN15 corresponds to pin number 82, INTIN14 corresponds to pin number 83 etc., and INTIN0 corresponds to pin number 97. These pins are active high.
LINTIN[1] LINTIN[0]	80 81	I I	Two **LOCAL INTERRUPT INPUT** pins accept edge or level sensitive interrupt requests that can only be delivered to the connected processor. These pins are active high.
REGISTER ACCESS PINS			
$\overline{\text{ADS}}$	64	I	**ADDRESS STROBE** signal indicating the start of a bus cycle. 82489DX does not commit to start the cycle internally until BUS GRANT is detected active.

Pin Definition Table (Continued)

Symbol	Pin No.	Type	Function
REGISTER ACCESS PINS (Continued)			
M/$\overline{IO}$, D/$\overline{C}$, W/$\overline{R}$	63 61 62	I I I	Bus cycle definition signals. Note that since the 82489DX registers can be mapped in either memory or I/O space, the M/$\overline{IO}$ pin is not used for register access cycles; it is only used to decode interrupt acknowledge cycles. 82489DX does not respond to code read cycles.
$\overline{BGT}$	66	I	The **BUS GRANT** input is optional and is used to indicate the address phase of a bus cycle in configurations where address timing cannot be inferred from $\overline{ADS}$. This signal is really used as an address latch enable, but is named as it is to indicate that it can normally be connected to the Intel Cache Controller generated signal of the same name. Must be tied low if not used.
$\overline{CS}$	74	I	The **CHIP SELECT** input indicates that the 82489DX registers are being addressed.
A3 A4 A5 A6 A7 A8 A9 A10	31 29 28 27 26 24 22 21	BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D	The address pins are used as inputs in addressing internal register space. Output function is reserved. They are also used to latch local unit ID on reset.
$\overline{DLE}$	73	I	**DATA LATCH/ENABLE** is optional and is used to indicate committing the data phase of a bus cycle in configurations where data timing cannot be inferred from other cycle timings. Must be tied low if not used.
D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12 D11	105 107 109 110 111 112 114 115 116 118 119 121 122 123 124 125 128 129 130 131 2	BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D	The DATA BUS is for all register accesses and interrupt vectoring.

Pin Definition Table (Continued)

Symbol	Pin No.	Type	Function
REGISTER ACCESS PINS (Continued)			
D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0	3 4 7 8 9 11 12 13 14 16 18	BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D	
DP3 DP2 DP1 DP0	101 102 103 104	BI-D BI-D BI-D BI-D	One Data Parity pin for each byte on the data bus. EVEN parity is generated any time the data bus is driven by the 82489DX.
$\overline{RDY}$	43	O	**READY** output indicates that the current bus cycle is complete. In the case of a read cycle, valid data and the return to inactive state after going active low may be delayed till $\overline{DLE}$ goes active.
PROCESSOR PINS			
PINT	35	T/S	The **PROCESSOR INTERRUPT OUTPUT** indicates to the processor that one or more maskable interrupts are pending. This pin is tri-stated at reset, and has an internal pull-down resistor to prevent false signaling to the processor until the 82489DX Local Unit is enabled and this pin is actively driven.
PRST	38	O	The **PROCESSOR RESET OUTPUT** is asserted/de-asserted upon 82489DX reset, and also in response to ICC bus messages with "RESET" delivery mode. This pin should be used with care.
PNMI	37	T/S	The **NON-MASKABLE INTERRUPT** output is signaled in respone to ICC bus messages with "NMI" delivery mode. This pin is tri-stated at reset, and has an internal pull-down resistor to prevent false signaling to the processor until the Local Unit is enabled and this pin is actively driven.
ICC BUS PINS			
ICLK	60	I	The **ICC BUS CLOCK** input provides synchronous operation of the ICC bus.
MBI[3:0]	76–79	I	The four **ICC BUS IN** inputs are used for incoming ICC bus messages. In smaller configurations the ICC bus input and outputs may be tied directly together at the pins. Pin number for MBI3 is 76, MBI2 is 77, MBI1 is 78 and MBI0 is 79.
MBO3 MBO2 MBO1 MBO0	45 48 49 51	O/D	The four **ICC BUS OUT** outputs are used for outgoing ICC bus messages. The current capacity is only 4 mA. So external bufferes will be needed.

Pin Definition Table (Continued)

Symbol	Pin No.	Type	Function
RESERVED PINS			
Reserved	34, 42	NC	These pins **MUST BE LEFT OPEN.**
Reserved	70, 72, 75		**Reserved** by Intel. **These pins should be strapped to V_{CC}.**
Reserved	71, 19, 20		**Reserved** by Intel. **These pins should be strapped to GND.**
POWER AND GROUND PINS			
V_{CC}	1, 32, 69, 98	POWER	Nominally +5V. These pins along with V_{SS} and V_{SSI} should be separately bypassed.
V_{CCP}	6, 15, 25,100, 108, 117, 126	POWER	Nominally +5V. These pins along with V_{SSP} should be separately bypassed.
V_{CCPO}	39, 46	POWER	Nominally +5V. These pins along with V_{SSPO} should be separately bypassed.
V_{SS}	5, 33, 67, 68, 99	GND	Nominally 0V. These pins along with V_{CC} should be separately bypassed.
V_{SSP}	10, 17, 23, 30, 106, 113, 120, 127, 132,	GND	Nominally 0V. These pins along with V_{CCP} should be separately bypassed.
V_{SSPO}	36, 40, 44, 47, 50	GND	Nominally 0V. These pins along with V_{CCPO} should be separately bypassed.
V_{SSI}	58	GND	Nominally 0V. These pins along with V_{CC} should be separately bypassed.

NOTE:
V_{CC}, V_{CCP} and V_{CCPO} should be of same voltage. V_{SS}, V_{SSP}, V_{SSPO} and V_{SSI} should be 0V.

4

4.0 FUNCTIONAL DESCRIPTION

As far as interrupt management is concerned, the 82489DX's interrupt control function spans over two functional units, the I/O Unit of which there is one per I/O subsystem, and the Local Unit of which there is one per processor. 82489DX has one I/O unit and one Local Unit in a single package. This section takes a detailed look at both local and I/O Units.

I/O Unit

The I/O Unit consists of a set of Interrupt Input pins, an Interrupt Redirection Table, and a message unit for sending and receiving messages from the ICC bus. The I/O Unit is where I/O devices inject their interrupts, the I/O Unit selects the corresponding entry in the Redirection Table and uses the information in that entry to format an interrupt request message. The message unit then broadcasts this message over the ICC bus. The content of the Redirection Table is under software control and is assigned benign defaults upon reset. The masks in the Redirectional Table entries are set to 1 at *hardware reset* to disable the interrupts.

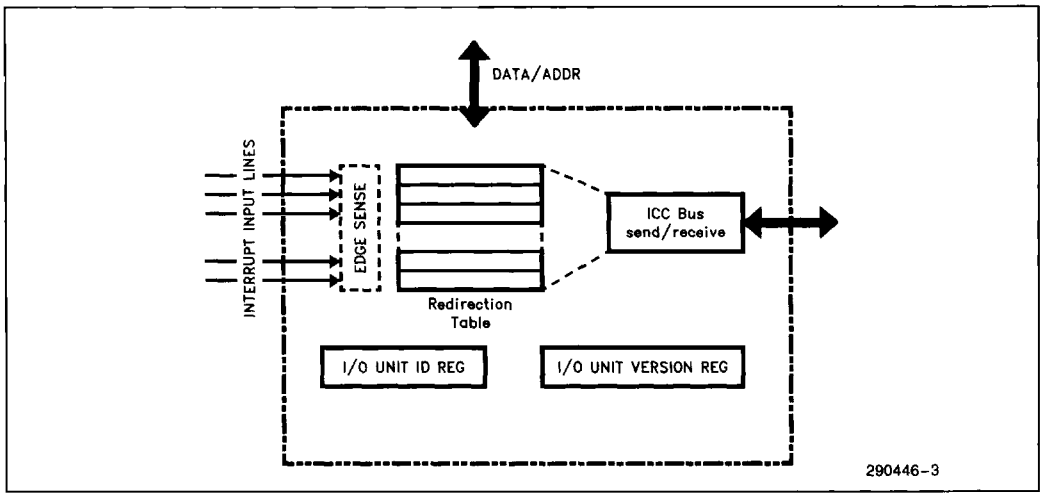


Figure 2. 82489DX I/O Unit Block Diagram

Local Unit

Interrupt Management of the Local Unit is responsible for local interrupt sources, interrupt acceptance, dispensing interrupts to the processor, and sending inter-processor interrupts. Depending on the delivery mode of the interrupt, zero, one or more units can accept an interrupt. A Local Unit accepts an interrupt only if it will deliver the interrupt to its processor. Accepting an interrupt is purely an inter-82489DX matter; dispensing an interrupt to the local processor only involves a 82489DX and its local processor.

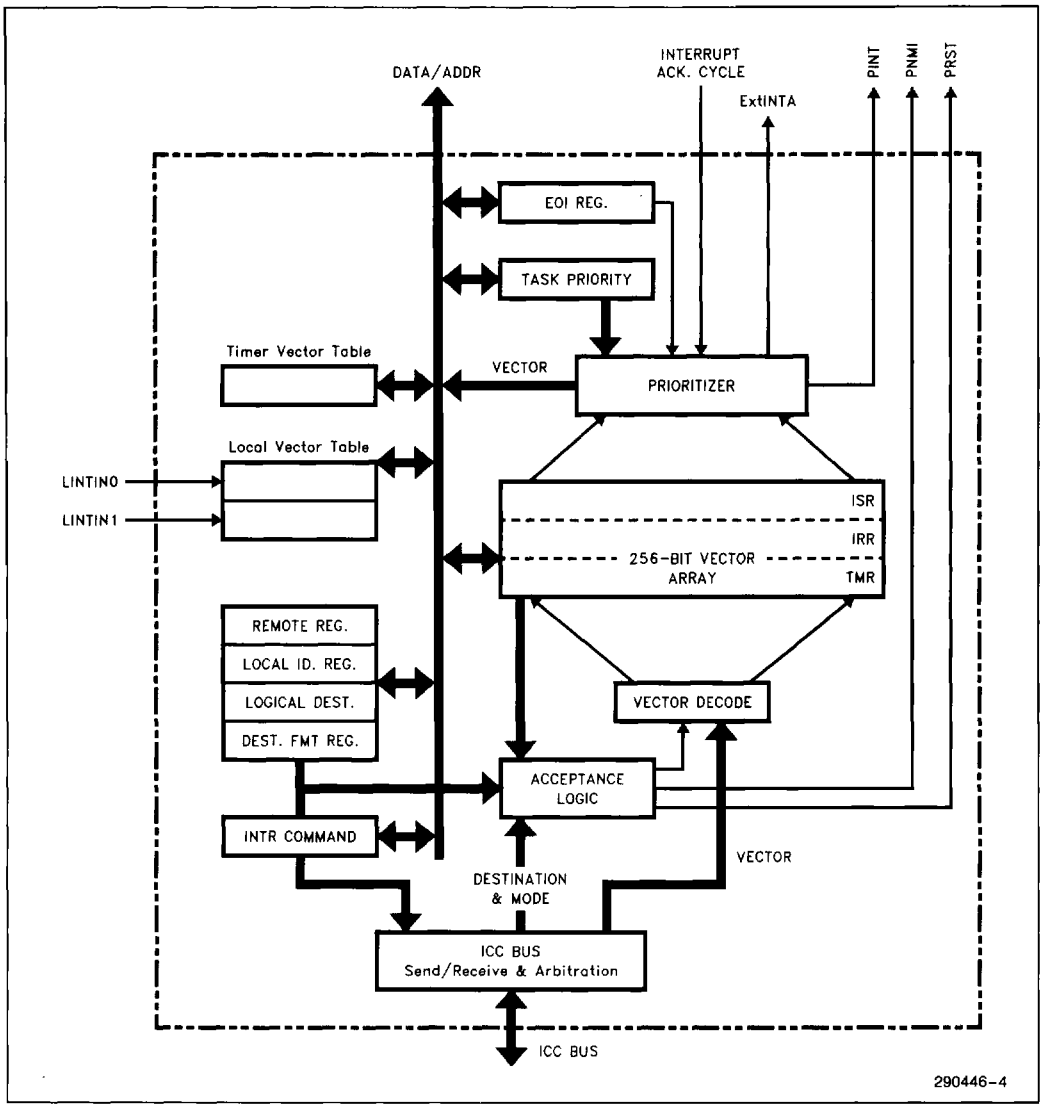


Figure 3. 82489DX Local Unit Block Diagram

4

5.0 INTERRUPT CONTROL MECHANISM

This section describes briefly the interrupt control mechanism in the 82489DX.

5.1 Interrupts

The interrupt control function of all 82489DXs are collectively responsible for delivering interrupts from interrupt sources to interrupt destinations in the multiprocessor system. When a processor accepts an interrupt, it uses the vector to locate the entry point of the handler in its interrupt table. The 82489DX architecture allows for 16 possible interrupt priorities; zero being the lowest priority and 15 being the highest. Priority of interrupt A "is higher than" the priority of interrupt B if servicing A is more urgent than servicing B. An interrupt's priority is implied by its vector; namely priority = vector/16.

With 256 vectors and 16 different priorities, this implies that 16 different interrupt vectors can share a single interrupt priority.

TOTAL ALLOWED INTERRUPT VECTORS

Out of 256 vectors, interrupt vectors 0 to 15 should not be used in the 82489DX. Only 240 interrupt vectors (vectors from 16 to 255) are supported in the 82489DX.

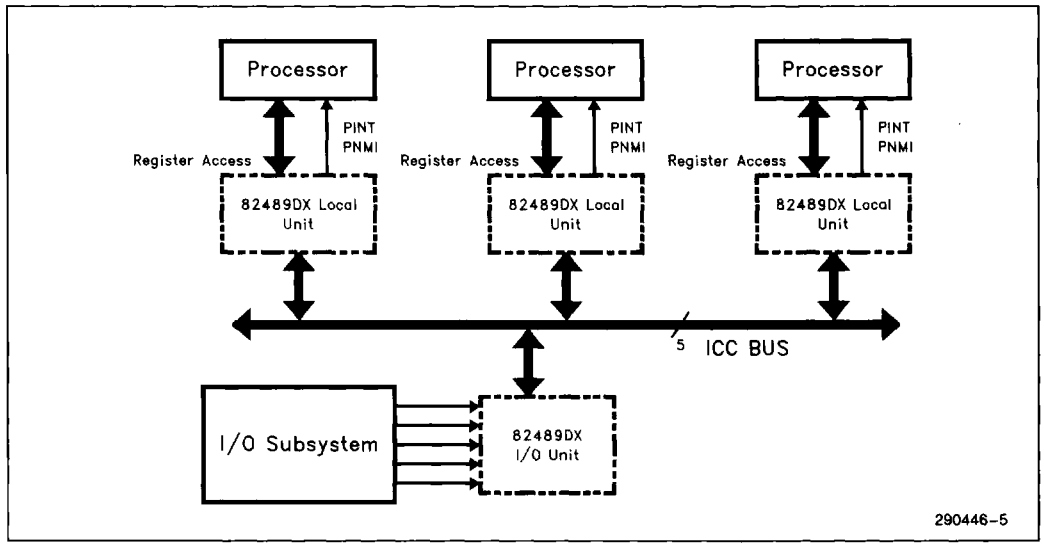


Figure 4. I/O Units and Local Units

INTERRUPT SOURCES

Interrupts are generated by a number of different interrupt sources in the system.

Possible interrupt sources are:

- Externally connected (I/O) devices. Interrupts from these external sources manifest themselves as edges or levels on interrupt input pins and can be redirected to any processor.
- Locally connected devices. These originate as edges or levels on interrupt pins, but they are al ways directed to the local processor only.
- 82489DX timer generated interrupts. Like locally connected devices, 82489DX timer can only interrupt its local processor.
- Processors. A processor can interrupt any individual processor or sets of processors. This supports software self-interrupts, preemptive scheduling, TLB flushing, and interrupt forwarding. A processor generates interrupts by writing to the interrupt command register in its Local Unit.

INTERRUPT DESTINATIONS

I/O Units can only source interrupts whereas Local Units can both source and accept interrupts, so whenever "interrupt destination" is discussed, it is implied that the Local Unit is the destination of the interrupt. In physical mode the destination processor is specified by a unique 8-bit 82489DX local ID. Only a single destination or a broadcast to all (LOCAL ID of all ones) can be specified in physical destination mode.

In logical mode destinations are specified using a 32-bit destination field. All Local Units contain a 32-bit Logical Destination register against which the destination field of the interrupt is matched to determine if the receiver is being targeted by the interrupt. An additional 32-bit Destination Format register in each Local Unit enables the logical mode addressing.

INTERRUPT DELIVERY

The description of interrupt delivery makes frequent use of the following terms:

- Each processor has a processor priority that reflects the relative importance of the code the processor is currently executing. This code can be part of a process or thread, or can be an interrupt handler. A processor's priority fluctuates as a processor switches threads, a thread or handler raises and lowers its priority level to mask out interrupt, and the processor enters an interrupt handler and returns from an interrupt handler to previously interrupted activity.
- A processor is lowest priority within a given group of processors if its processor priority is the lowest of all processors in the group. Note that more than one processor can be the lowest priority in a given group.
- A processor is the focus of an interrupt if it is currently servicing that interrupt, or if it currently has a request pending for the interrupt.

Interrupt delivery begins with an interrupt source injecting its interrupt into the interrupt system at one of the 82489DX. Delivery is complete only when the servicing processor tells its 82489DX Local Unit it is complete by issuing an end-of-interrupt (EOI) command to its 82489DX Local Unit. Only then has all (relevant) internal state regarding that occurrence of the interrupt been erased. The interrupt system guarantees exactly-once delivery semantics of interrupts to the specified destinations. Exactly-once guaranteed delivery implies a number of things:

- The interrupt system never rejects interrupts; it never NAKs interrupt injection, interrupts are never lost, and the same interrupt (occurrence) is never delivered more than once.

Clearly a single edge interrupt or level interrupt counts as a single occurrence of an interrupt. In uniprocessor systems, an occurrence of an interrupt that is already pending (IRR) cannot be distinguished from the previous occurrence. All occurrences are recorded in the same IRR bit. They are therefore treated as "the same" interrupt occurrence.

For lowest-priority delivery mode, by delivering an interrupt first to its focus processor (if it currently has one), the identical behavior can be achieved in a MP (Multiprocessor) system. If an interrupt has a focus processor then the interrupt will be delivered to the interrupt's focus processor independent of priority information. This means that even if there is a lower priority processor compared to the focus processor, the interrupt still gets delivered to the focus processor.

Each edge occurring on an edge triggered interrupt input pin is clearly a one-shot event; each occurrence of an edge is delivered. An active level on a level triggered interrupt input pin represents more of a "continuous event". Repeatedly broadcasting an interrupt message while the level is active would cause flooding of the ICC bus, and in effect transmits very little useful information since the same processor (the focus) would have to be the target.

Instead, for level triggered interrupts the 82489DX merely recreate the state of the interrupt input pin at the destination . The source 82489DX accomplishes this by tracking the state of the appropriate destina-

tion 82489DX's Interrupt Request Register (or pending bit) and only sending inter-82489DX messages when the state of the interrupt input pin and the destination's interrupt request enter a disagreement. Unlike edge triggered interrupts, when a level interrupt goes into service, the interrupt request at the servicing 82489DX is not automatically removed. If the handler of a level sensitive interrupt executes an EOI then that interrupt will immediately be raised to the processor again, unless the processor has explicitly raised its task priority, or the source of that interrupt has been removed.

5.2 Interrupt Redirection

This section specifically talks about how a processor is picked during interrupt delivery. The 82489DX supports two modes for selecting the destination processor: Fixed and Lowest Priority.

- *Fixed Delivery Mode*
  In fixed delivery mode, the interrupt is unconditionally delivered to all local 82489DXs that match in the destination information supplied with the interrupt. Note that for I/O device interrupts typically only a single 82489DX would be listed in the destination. Priority and focus information are ignored. If the priority of a destination processor equal to or higher than the priority of the interrupt, then the interrupt is held pending locally in the destination processor's Local Unit, until the processor priority becomes low enough at which time the interrupt is dispensed to the processor. More than one processor can be the destination in fixed-delivery mode.
- *Lowest Priority Delivery Mode*
  Under the lowest priority delivery mode, the processor to handle the interrupt is the one in the specified destination with the lowest processor priority value. If more than one processor is at the lowest priority, then a unique arbitration ID is used to break ties. For lowest priority dynamic delivery, the interrupt will always be taken by its focus processor if it has one. The lowest priority delivery method assures minimum interruption of high priority tasks. Since each Local Unit only knows its own processor priority, determining the lowest priority processor is done by arbitration on the ICC bus. Only one processor can be the destination in lowest-priority delivery mode.

INTER-82489DX COMMUNICATION

All I/O and Local Units communicate during interrupt delivery. Interrupt information is exchanged between different units on a dedicated five wire ICC bus in the form of broadcast messages. A 82489DX Unit's 8-bit ID is used as its name for the purpose of using the ICC bus, and all 82489DX units using one ICC bus should be assigned a different ID. The Arbitration ID of the Local Units used to resolve tles during lowest priority arbitration is also derived from the Local Unit's ID.

6.0 82489DX LOCAL UNIT REGISTERS DESCRIPTION

6.1 Local Unit ID Register

Each 82489DX Local Unit has a register that contains the Local Unit's 8-bit ID. The Local Unit ID serves as a physical name of the 82489DX Local Unit. It can be used in specifying destination information and is also used for accessing the ICC bus. Eight address lines A[10]–A[3] are sampled on every clock edge while RESET is asserted. The last sample remains in the Local Unit ID register after reset. Alternatively, the ID can be loaded with a register write as part of software initialization. The Local Unit ID is read-write by software.

Bits [31..24]	Bits [23..0]

82489DX Local Unit ID Register

Bits [31..24] Local Unit ID: The Local Unit ID serves as the physical "name" of the Local Unit used for addressing the 82489DX in physical destination mode and for the ICC bus usage. In a system with say four 82489DX, there are 4 Local Units and 4 I/O Units. All the 8 units should be assigned different IDs. For future compatibility use only IDs from 0 to 14.

Bits [23..0] Bits [23..0] are Reserved. They should be written with 0.

6.2 Destination Format Register

Interrupt Destination can be either addressed physically or logically. When the interrupt message addresses the destination physically, each 82489DX in the ICC bus compares the address with its own unit ID. If the message is a broadcast type then every Local Unit accepts the interrupt.

When the message addresses the destination using logical addressing scheme each Local Unit in the ICC bus compares the logical address in the interrupt message with its own Logical Destination Register. If there is a bit match (i.e., if at least one of the corresponding pair of bits match) this local unit is selected for delivery.

All the 32 bits of Destination Format Register of all 82489DX connected in the ICC bus should be written with "1" to enable the addressing scheme.

Destination Format Register

Bits [31:0]

Logical Destination Register

Bits [31:0]

For future compatibility, use only bits 31–24 of Logical Destination Register. For binary compatibility, it is strongly recommended that 82489DX software use only 8 MSB of Logical Destination Register.

6.3 Local Interrupt Vector Table Registers

The Redirection Table serves to steer interrupts that originate in the I/O subsystems to the processors. The Local Vector Table is its equivalent for interrupts that are restricted to only the local processor. The Local Vector Table contains three 32-bit registers. Register 0 corresponds to the timer, registers 1 and 2 correspond to local interrupt input pins, LINTIN0 and LINTIN1.

The format of both the Local 0 and Local 1 interrupt vector tables are identiçal. The following register description talks about both Local Interrupts 0,1 vectors.

Local Interrupts 0, 1 Interrupt Vectors

Vector: [Bits 7–0]

This is the vector to use when generating an interrupt for this entry.

Delivery Mode: [Bits 10–8]

000: Fixed

001: <reserved>

010: <reserved>

011: <reserved>

100: NMI

101: <reserved>

110: <reserved>

111: ExtINT

000: (fixed) means deliver the signal on the INT pin of the local processor. Trigger mode for "fixed" Delivery Mode can be edge or level.

100: (NMI) means deliver the signal on the NMI pin of the local processor. Vector information is ignored. A Delivery Mode equal to "NMI" requires a "level" triggered mode.

111: (ExtINTA) means deliver the signal to the INT pin of the local processor as an interrupt that originated in an externally connected (8259A compatible) interrupt controller. ExtINTA pin is as- serted also. The INTA cycle that corresponds to this ExtINTA delivery, should be routed to the external controller that is expected to supply the vector. A delivery mode of ExtINT requires an edge trigger mode. (See the section on compatiblity for more details.)

Bit 11: Bit 11 is Reserved. It should be written 0.

Delivery Status: [Bit 12]

This field is software-read only. Software writes to this field (as part of a 32-bit word) have no effect on this bit. Delivery status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined.

0: (idle) means that there is currently no activity for this interrupt.

1: (send pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by the recently injected interrupts that are in the process of being delivered.

Local INT0 Vector Table

Bits [31:17]	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]

Local INT1 Vector Table

Bits [31:17]	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]

Figure 5. Local Vector Table

Bit 13: Bit 13 is Reserved. It should be written 0.

Remote IRR: [Bit 14]

This bit is used for level triggered local interrupts; its meaning is undefined for edge triggered interrupts. Remote IRR mirrors the interrupt's IRR bit of this local unit. Remote IRR is software read-only; software writes to this bit do not affect it.

Trigger Mode: [Bit 15]

The Trigger Mode field indicates the type of signal on the local interrupt pin that triggers an interrupt.

0: indicates edge sensitive,

1: indicates level sensitive.

Only the local interrupt pins can be programmed as edge or level triggered. Timer interrupts are always treated as edges.

MASK: [Bit 16]

0: enables injection of the interrupt,

1: masks injection of the interrupt.

Bits [31:17] Bits [31:17] are Reserved. Should be written 0.

6.4 Inter-Processor Interrupt Registers

A processor generates inter-processor interrupts by writing to the Interrupt Command Register in its 82489DX Local Unit. Conceptually, this can be thought of as the processor providing the interrupt's Redirection Table Entry on the fly. Not surprisingly, the layout of the Interrupt Command Register resembles that of an entry in the Redirection Table. Note that the format of this register allows a processor to generate any interrupt. A processor may use this to forward device interrupts originally accepted by it to other processors.

All fields of the Interrupt Command Register are read-write by software with the exception of the Delivery Status field which is read-only. Writing to the 32-bit word that contains the interrupt vector causes the interrupt message to be sent.

Vector: [Bits 7-0]

The vector identifies the interrupt being sent. If the Delivery Mode is "Remote Read", then the Vector field contains the address of the register to be read in the remote 82489DX's Local Unit. The addresses are listed in the section discussing 82489DX Local Unit register summary. For example, for ID register, remote read address of 02 should be specified in vector field.

Delivery Mode: [Bits 10-8]

The Delivery Mode is a 3-bit field that specifies how the 82489DX listed in the destination field (bits 63:32) should act upon reception of this signal. Note that certain Delivery Modes will only operate as intended when used in conjunction with a specific Trigger Mode. These restrictions are indicated for each Delivery Mode.

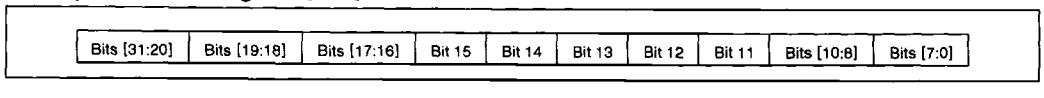
000: (Fixed) means deliver the signal on the INT pin of all processors listed in the destination. Trigger Mode for "fixed" Delivery Mode can be edge or level.

001: (Lowest Priority) means deliver the signal on the INT pin of the processor that is executing at the lower priority among all the processors listed in the specified destination; Trigger Mode for "lowest priority" Delivery Mode can be edge or level.

010: Intel Reserved. Should not be used.

011: (Remote Read) is a request to a remote 82489DX Local Unit to send the value of one of its registers over the ICC bus. The register is selected by providing its address in the Vector field. The register value is latched by the requesting 82489DX and stored in the Remote Register where it can be read by the local processor. A Delivery Mode of "Remote Read" requires an "edge" Trigger Mode.

Interrupt Command Register [31:0]



100: (NMI) means deliver the signal on the NMI pin of all processors listed in the destination, vector information is ignored. A Delivery Mode equal to "NMI" requires a "level" Trigger Mode.

101: (Reset) means deliver the signal to all local units listed in the destination. The destination local unit will assert/deassert its PRST output pin. All addressed 82489DX Local Units will assume their reset state but preserve their ID. One side effect of an ICC bus message with Delivery Mode equal to "Reset" that results in a deassert of reset is that all Local Units (whether listed in the destination or not) will reset their lowest-priority tie breaker arbitration ID to their Local Unit ID (see the section on the ICC bus for details). A Delivery Mode of "Reset" requires a "level" Trigger Mode. "RESET" should not be used with "self" or "all incl self" Shorthand mode since it will leave the system in non-recoverable reset state. If "RESET" is used with "all excl self" mode software should make sure that only one CPU executes this instruction in a MP system.

110: Intel Reserved. Should not be used.

111: Intel Reserved. Should not be used

Destination Mode: [Bit 11]

This field determines the interpretation of the Destination field.

0: (Physical Mode): in Physical Mode, a destination 82489DX is identified by its Local Unit ID. Bits 56 through 63 (8 MSB of the destination field) specify the 8-bit 82489DX Local Unit ID.

1: (Logical Mode): in Logical Mode, destinations are identified by matching on Logical Destination under the control of the Destination Format Register in each Local 82489DX. The 32-bit Destination field is the logical destination.

Delivery Status: [Bit 12]

Delivery Status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined:

0: (Idle) means that there is currently no activity for this interrupt;

1: (Send Pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by other recently injected interrupts that are in the process of being delivered;

Delivery Status is software read-only; software writes to this field (as part of a 32-bit word) do not affect this bit. Software can read to find out if the current interrupt has been sent, and the Interrupt Command Register is available to send the next interrupt. If the Interrupt Command Register is overwritten before the Delivery Status is "Idle", then the destiny of that interrupt is undefined; i.e., the interrupt may have been lost.

Bit 13: Bit 13 is Reserved . Should be written 0.

Level: [Bit 14]

Software can use this bit in conjunction with the Trigger Mode bit when issuing an inter-processor interrupt to simulate assertion/deassertion of level sensitive interrupts.

To assert: Trigger mode = 1 and Level = 1.

To deassert: Trigger mode = 1 and Level = 0.

For example, a message with Delivery Mode of "Reset", a Trigger Mode of "Level", and Level bit of 0 deasserts Reset to the processor of the addressed 82489 DX Local Unit(s). As a side effect, this will also cause all 82489DX to reset their Arbitration ID to their unit ID. (The Arb ID is used for tie breaking in lowest priority arbitration.)

Trigger Mode: [Bit 15]

Software can use this in conjunction with Level Assert/Deassert to generate interrupts that behave as edges or levels.

0: Edge

1: Level

4

Remote Read Status: [Bits 17,16]

This field indicates the status of the data contained in the Remote Read register. This field is read-only to software. Whenever software writes to the Interrupt Command Register using Delivery Mode "Remote Read" the Remote Read status becomes "in-progress" (waiting for the remote data to arrive). The remote 82489DX Local Unit is expected to respond in a fixed amount of ICC bus cycles. If the remote 82489DX Local Unit is unable to do so, then the Remote Read status becomes "Invalid". If successful, the Remote Read status resolves to "Valid". Software should poll this field to determine completion and success of the Remote Read command.

00: (invalid): the content of the Remote Read Register is invalid. This is the case after a Remote Read command issued and the remote 82489DX Local Unit was unable to deliver the Register content in time.

01: (in progress): a Remote Read command has been issued and this 82489DX is waiting for the data to arrive from the remote 82489DX Local Unit.

10: (valid): the most recent Remote Read command has completed and the remote register content in the Remote Read Register is valid.

11: reserved.

Destination Shorthand: [Bits 19,18]

This field indicates whether a shorthand notation is used to specify the destination of the interrupt and if so, which shorthand is used. Destination Shorthands do no use the 32-bit Destination field, and can be sent by software with a single 32-bit write to the 82489DX's Interrupt Command Register. Shorthands are defined for the following common cases: software self interrupt, interrupt to all processors in the system including the sender, interrupts to all processors in the system excluding the sender.

00: (dest field): means that no shorthand is used. The destination is specified in the 32-bit Destination field in the second word (bits 32 to 63) of the Interrupt Command Register.

01: (self): means that the current 82489DX Local Unit is the single destination of the interrupt. This is useful for software interrupts. The Destination field in the Interrupt Command Register is ignored. RESET Delivery mode should not be used with self destination. Only FIXED delivery mode should be used with SELF.

10: (all incl self): means that the interrupt is to be sent to "all" processors in the system including the processor sending the interrupt. The 82489DX will broadcast a message with destination unit ID field set to all ones. RESET assert Delivery mode should not be used with "all incl self" destination.

11: (all excl self): means that the interrupt is to be sent to "all" processors in the system with the exception of the processor sending the interrupt. The 82489DX will broadcast a message with destination unit ID field set to all ones. All-excl-self is useful during selection of a boot processor (init) and also for TLB flush where "self" is flushed using the processor flush instruction. Only one CPU in a MP system should execute "all excl self" destination if used with RESET Delivery mode.

Bits [31:20] Bits [31:20] are Reserved. They should be written 0.

Interrupt Command Register [63:32]

Bits [63:32]

Destination: [Bits 63–32]

This field is only used when the Destination Shorthand is set to "Dest Field". If Destination Mode is Physical Mode, **then the 8 MSB contain a Destination unit ID.** If Logical Mode, the full 32-bit Destination field contains the logical address. The enabling is done by Destination Format Register.

6.5 IRR, ISR, TMR Registers

INTERRUPT ACCEPTANCE

All 82489DX Local Units listen to all messages sent over the ICC bus. For each message, the local unit first checks if it belongs to the destination in the

message. It does this by matching the 32-bit Destination field in the message against its logical Destination Register, if the message addresses in logical mode, and against its physical ID, if the message addresses in physical mode. All 82489DX Local Units that match are said to "belong to the group".

Each 82489DX Local Unit contains three 256-bit registers that play a role in the acceptance of interrupts and in dispensing accepted interrupts to the local processor. Each of these registers is a bit array where bit position i tracks information about the interrupt with vector i. These bits track information about the (PINT) maskable interrupts only. They are not relevant for NMI, RESET or ExtINT type of interrupts. The Interrupt Request Register (IRR) contains the interrupts accepted by this 82489DX Local Unit but not yet dispensed to the processor. The In Service Register (ISR) contains the interrupts that are currently in service by the processor, i.e., the interrupts that have been dispensed to the processor but for which the processor has not yet signaled the End-Of-Interrupt.

Note that the 82489DX's IRR and ISR registers have the same meaning and operation as in the 8259A in fully nested/non-specific EOI mode. Note also that these registers play no role in providing 8259A compatibility. Compatibility is handled by making an external 8259A-style controller directly visible to the processor and having the 82489DX become transparent.

Each interrupt has a vector associated with it, which determines the bit position, and hence the priority for the interrupt. When an interrupt is being serviced, all equal or lower priority interrupts are automatically masked by the 82489DX Local Unit.

The Trigger Mode Register (TMR) indicates for each interrupt whether the interrupt is edge or level. This information is transmitted with each 82489DX interrupt request message and reflects the Trigger Mode bit in the interrupt's Redirection Table entry. If an interrupt goes in service and the TMR bit is 0 (edge), then the interrupt's IRR bit is cleared at the same time the ISR bit is set. If the TMR bit is 1 (level), then the IRR bit is not cleared when the interrupt goes in service. In the latter case, the IRR bit mirrors the state of the interrupt's input pin.

The following diagram shows 82489DX operation with devices A and B sharing a level triggered interrupt input. The diagram illustrates how Remote IRR, and the IRR bit at the destination 82489DX track the state of INTIN. It also illustrates how an EOI is followed immediately be re-raising the interrupt as long as the INTIN is still asserted by some device.

4

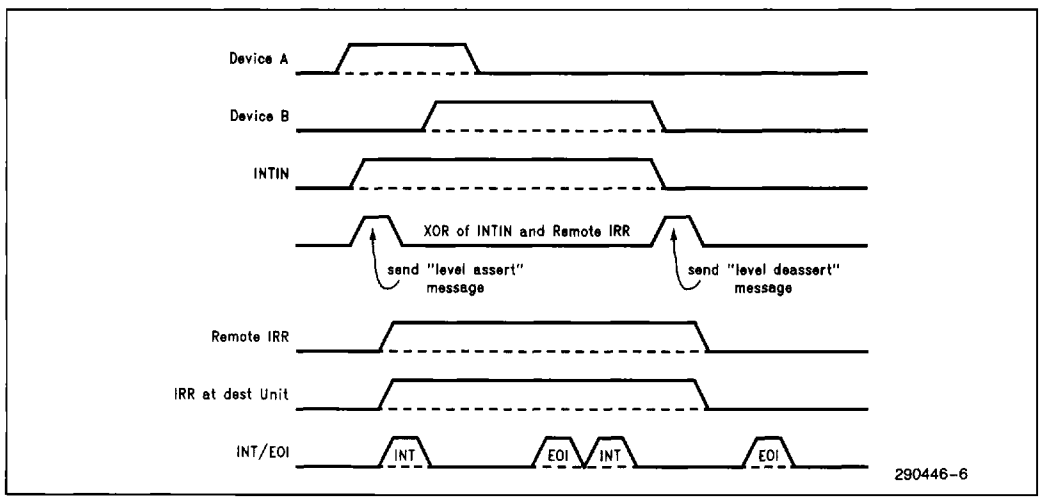


Figure 6. Interrupt Sharing

ISR, IRR, and TMR are read-only by software. Each of these 256-bit registers is accessed as four separate 32-bit registers. Note that there is no general Interrupt Mask Register (IMR) as in the 8259A. The processor masks interrupts temporarily by writing to the Icoal unit's Task Priority Register (described shortly).

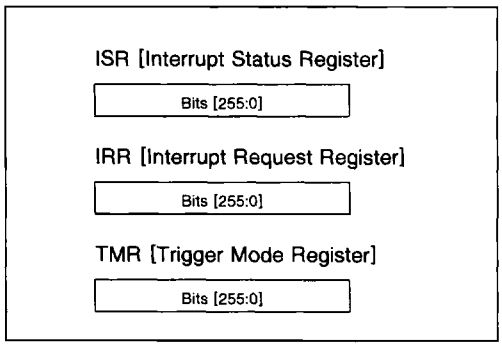


Figure 7. ISR, IRR, and TMR

TMR (Trigger Mode Register):

> If 0 [edge triggered] the corresponding IRR bit is automatically cleared when interrupt service starts. If 1[level triggered] this is not the case; instead, the source 82489DX must explicitly request the IRR bit be cleared (upon deassert of the interrupt input pin or upon sending an appropriate interprocessor interrupt). Upon acceptance of an interrupt, the TMR bit is cleared for edge triggered interrupts and set for level triggered interrupts. This information is carried in the accepted interrupt message. The source 82489DX I/O unit also tracks the state of the destination unit's IRR bit (Remote IRR bit in the Redirection Table). When a level triggered interrupt input is deasserted, the source 82489DX I/O unit detects the discrepancy between the input pin state and the Remote IRR, and automatically sends a message telling the destination 82489DX to clear IRR for the interrupt.

IRR (Interrupt Request Register):

> It contains the active interrupt requests that have been accepted, but not yet dispensed by this 82489DX Local Unit. A bit in IRR is set when the 82489DX Local Unit accepts the interrupt. When TMR is 0, it is cleared when the interrupt is serviced; when TMR is 1, it is cleared when the 82489DX Local Unit receives a message to clear it.

ISR (In Service Register):

> It marks the interrupts that have been delivered to the processor, but that have not been fully serviced in that an End-Of-Interrupt has not yet been received. The ISR register reflects the current state of the processor's interrupt stack.

ACCEPTANCE MECHANISM

Interrupt acceptance proceeds as follows. If the delivery mode is Fixed, then each unit in the destination group unconditionally accepts the interrupt message and sets the interrupt's IRR bit. If the delivery mode is Lowest Priority, then each processor in the group first checks if it is currently the focus of the interrupt by checking its ISR and IRR. If an 82489DX finds one of these bits set for the incoming interrupt, then that 82489DX Local Unit accepts the interrupt independent of priority, and "signals" the other 82489DX Local Units to abort the priority arbitration. This avoids multiple delivery of a same interrupt occurrence to different processors, consistent with interrupt delivery semantics in uniprocessor systems as described above. If a message is to be delivered for NMI or Reset, then all 82489DX Local Units listed in the destination unconditionally assert/deassert the corresponding output pin. ISR, IRR, etc. are bypassed for NMI or reset and vector information is undefined.

The acceptance decision process is illustrated in the flow chart below.

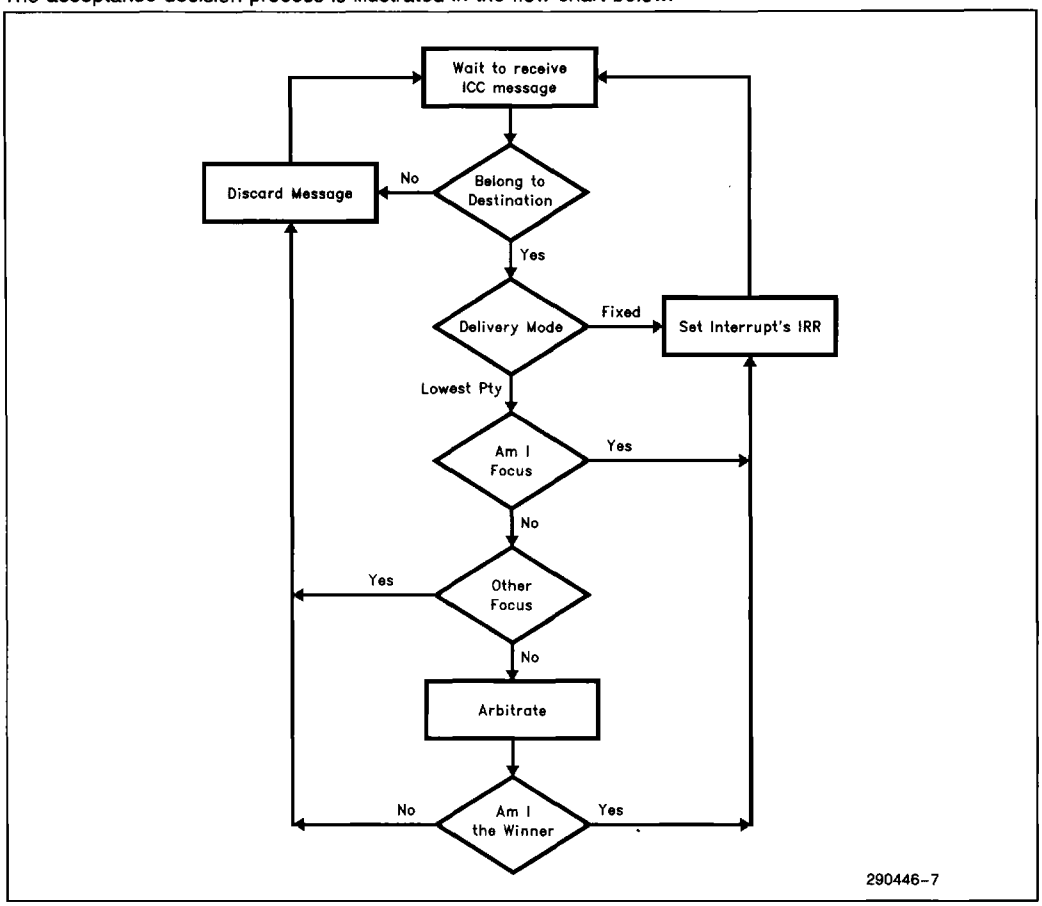


Figure 8. Interrupt Acceptance Flow Chart

6.6 Tracking Processor Priority

Each 82489DX Local Unit should be programmed with task priority so that it can mask interrupts that are less priority than that of the processor temporarily.

Task switching and task priority changes are the result of explicit software action. The operating system may define a number of task scheduling classes. Examples are an idle class, a background class, a foreground class, and a time critical class. Alternatively, different classes can be assigned to user code versus system code. If tasks in different classes are executing when an interrupt comes in, then it may be advantageous to interrupt the processor currently running the task in the least important class. Clearly, if one processor is idle while others are doing work, the idle processor would be the obvious target for servicing the interrupt. This implies that there is use in defining priority levels below all interrupt levels that can participate in lowest priority delivery selection.

At times, the operating system may need to block out interrupts from being serviced. For example, to synchronize access to a shared data structure between a device driver and its interrupt handler the driver raises it priority to equal or higher than the interrupt's priority.

The local 82489DX supports this via its Task Priority Register (the 8259A supports this via the interrupt mask register (IMR).) Software that wants to make use of this is required to inform its 82489DX Local Unit of the prioity change by updating the Task Priority Register. The Task Priority field is 8 bits providing up to 256 distinct priorities. The 4 MSB of this register correspond to the 16 interrupt priorities while the 4 LSB provide more precision. Priorities are best noted as x:y, where x is the value of the 4 MSB and y is the value of the 4 LSB. For example, Task Priority Register values 0:y with $0 < y < 15$ (and 0 in the 4 MSB) can be used to represent the priorities of the task scheduling classes described above ($y = 0$ for idle; $y = 1$ for background; etc.). Except for interrupts with vectors 0 through 15 (which are often predefined by the processor) which all have priority 0:0, the priorities of all other interrupts and their handlers is x:0 with $1 < x < 15$ and is above the base task priorities 0:y.

For example, interrupt vector 123 has priority 7:0 ($123/16 = 7$) and can be masked by any task that raises its priority to a value equal or higher than 7:0.

82489DX uses Task priority register for the purpose of masking the interrupts. The task priority register should be programmed with a priority value to specifiy the priority of task the processor is executing. 82489DX masks any interrupts of lower or equal priority when compared with task priority.

When task priority register is programmed with the priority 15, all the interrupts are masked. When task priority register is programmed with priority level X, by definition, all the interrupts of priority X and below X will be masked. When task priority register is programmed with the priority 0 then all the interrupts above priority 0 are allowed to interrupt the processor. This means that when task priority register is programmed even with the lowest value, i.e., 0, interrupts of priority 0 will be masked. So only 240 interrupt vectors should be used in 82489DX. Interrupt vecotrs from 0 to 15 should not be used.

The first priority value computed is the maximum of:

- Task Priority (4msb : 4lsb) and
- the priority of the highest order ISR bit set ((vector/16) :0).

The value is used to determine whether or not a pending interrupt can be dispensed to the processor.

The second priority value computed is the maximum of:

- Task Priority (4msb : 4lsb), and
- the priority of the highest order ISR bit set ((vector/16) :0), and
- the priority of the highest order IRR bit set ((vector/16) :0).

This value is used during arbitration as part of lowest-priority delivery.

Task Priority Register

Bits [31:8]	Bits [7:0]

From the information in the Task Priority Register and the priority information derived from the ISR and IRR register, the 82489DX Local Unit computes two additional priority values:

Bits [31:8] Bits [31:8] are Reserved. They should be written 0.

Bits [7:0] Task Priority

Bits [7:0] are used to specify the task priority.

6.7 Dispensing Interrupts

DISPENSING INTERRUPTS TO THE LOCAL PROCESSOR

Once a 82489DX Local Unit accepts an interrupt, it guarantees delivery of the interrupt to its local processor. (This part of the 82489DX functions similarly to an 8259A.) Dispensing a maskable interrupt to the local processor begins when the Local Unit asserts the INT pin of its processor. If the processor has interrupts enabled, it will respond by issuing an INTA cycle. This causes the Local Unit to freeze its internal priority state and release the 8-bit vector of the highest priority interrupt on the data bus where it is read by the processor and used to find the handler's entry point. The INT/INTA protocol also causes the interrupt's ISR bit to be set. The corresponding bit in the IRR register is only cleared if the TMR register indicates it should do so (edge triggered interrupts), otherwise (level triggered interrupts), IRR is only cleared when the Interrupt Input Pin is deasserted.

6.8 Spurious Interrupt Vector Register

SPURIOUS INTERRUPT

Note that it can happen that a level-triggered interrupt is deasserted right before its INTA cycle. In that case, all IRR bits may be clear and the prioritizer may not find a vector to give to the processor. To satisfy the processor's demand for a vector, instead, the 82489DX will return a spurious interrupt vector instead.

A similar situation may occur when the processor raises its Task Priority at or above the level of the interrupt for which the Processor INT pin is currently being asserted. When the INTA cycle is issued, the interrupt that was to be dispensed has become masked (masked but remembered).

Dispensing the spurious interrupt vector does not affect the ISR register, so the handler for this vector should just return without EOI. If the vector is shared with a valid interrupt, then the handler can read the vector's bit in the ISR register to check if it is invoked for the valid interrupt (ISR bit set) or not (ISR bit clear). Given the range of 240 vectors, overloading the spurious interrupt with a valid interrupt is not expected to be common practice. The spurious interrupt vector to be used by a Local Unit is programmable via the Spurious Interrupt Vector Register.

UNIT ENABLE

It is possible that Local Units exist in the system that do not have a processor to which to dispense interrupts. The only danger this represents in the system is that if any interrupt is broadcast to all processors using lowest priority delivery mode when all processors are at the lowest priority, there is a chance that a Local Unit without the processor may accept the interrupt if this Local Unit happens to have the lowest Arb ID at the time. To prevent this from happening, all Local Units initialize in the disabled state and must be explicitly enabled before they can either start accepting or transmitting messages from the ICC bus. A disabled 82489DX Local Unit only responds to messages with Delivery Modes set to "Reset". Reset deassert messages should be sent in Physical Destination mode using the target's Local Unit ID since the logical destination information in the lcoal units is undefined (all zeroes) when the 82489DX comes out of Reset.

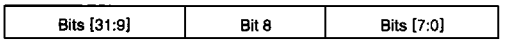


Figure 9. Spurious Interrupt Vector Register

Bits [31 .. 9] Reserved Bits. Should be written 0.

Unit Enable:[Bit 8]

0: When a 0 is written to this bit, this Local Unit gets disabled with regard to responding to messages sent as well as transmitting on the ICC bus. It only responds to messages with Delivery Mode set to "Reset". Reading a 0 at this bit indicates that the unit is disabled.

1: When a 1 is written to this bit, the current Local Unit is enabled for both transmitting and receiving unit messages. Reading a 1 at this bit indicates that the unit is enabled.

Spurious Vector: [Bits 7–0]

For future compatibility, bits [3–0] should be 1111.

6.9 End-of-Interrupt (EOI) Register

Before returning from the interrupt handler, software must issue an End-Of-Interrupt (EOI) command to the 82489DX Local Unit. The data written to EOI register is don't care. This tells the 82489DX to clear the highest priority bit in the ISR register since the interrupt is no longer in service. Upon EOI, 82489DX goes through prioritization returning to the next highest priority activity. This can be a previously interrupted handler (from ISR), a pending interrupt request (from IRR), or an interrupted task (from Task Priority).

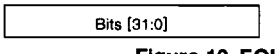


Figure 10. EOI Register

Bits [31:0]: are don't care.

6.10 Remote Read Register

Since all 82489DX Local Units would typically occupy the same address range, an 82489DX local unit's registers can only be accessed by the local processor. From a system debugging point of view, this would mean that a large amount of state would become inaccessible if its corresponding processor hangs for whatever reason. To assist in the debugging of MP systems, the 82489DX support a mechanism that provides read-only access to any register in any other 82489DX lcoal unit in the system.

To read any register in a "remote" 82489DX Local Unit, the processor writes to the Interrupt Command Register specifying a Delivery Mode equal to "Remote Read". The remote 82489DX is specified in the Destination field of the Interrupt Command Register in the usual fashion. Debug software would make sure that this selects a single 82489DX only—for example by using the target's 82489DX Local Unit ID in physical destination mode. Since no vector is associated with remote register access, the Vector field in the Interrupt Command Register is used to select the individual remote 32-bit register to be read. The selector value corresponds to the address (offset) of the register in the local 82489DX's address space. Sending a "Remote Read" command results in sending a message on the ICC bus. The destination 82489DX responds by placing the 32-bit content of the selected register on the ICC bus. This value is read by sending the 82489DX and place it in the Remote Register where software can get at it using regualr register access to its 82489DX Local Unit. The Remote Register is software read-only. The contents of the Remote Register is valid when the Delivery Status in the Interrupt Command Register has become "Idle" again.

Remote Read Register

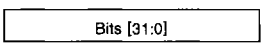


Figure 11. Remote Register

Bits [31:0] Bits [31:0] contain the contents of Remote Read Register.

6.11 82489DX Local Configuration

LOCAL VERSION REGISTER

Each 82489DX Local Unit contains a hardware Version Register that identifies this 82489DX Local Unit version. This register is read only.

Local Version Register

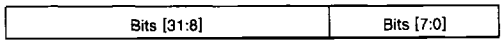


Figure 12. Local Version Register

Version: [Bits 7–0]
This is a version number that identifies this version. This field is hardwired and is read-only. Will be read as "1" for 82489DX.

Bits [31:8] Bits 31:8 are reserved.

6.12 82489DX Timer Registers

Overview

82489DX Local Unit contains one 32-bit wide programmable binary timer for use by the local processor. The timer can select its clock base from one of three possible clock inputs. A timer mode can be programmed to operate in either one-shot mode or periodic mode. The timer can be configured to interrupt the local processor with a vector.

Time Base

The 82489DX has two independent clock input pins:

1. The CLK pin provides the clock signal that drives the 82489DX's internal operation.
2. The TMBASE pin allows an independent clock signal to be connected to the 82489DX for use by the timer functions.

Signals from both CLK and TMBASE can be used as clock inputs that feed the timer. In addition, the 82489DX contains a divider that can be configured to divide either input clock signal. The divider can be programmed to divide the selected input clock by 2, 4, 8, or 16. CLK, TMBASE, and the output of the divider together provide three time bases: Base 0, Base 1, and Base 2. Base 0 is always equal to CLK: Base 1 is always equal to TMBASE; and base 2 is one of; CLK/2, CLK/4, CLK/8, CLK/16, TMBASE/2, TMBASE/4, TMBASE/8, or TMBASE/16. The timer can independently select one of these three time bases as its clock input as depicted in the following diagram.

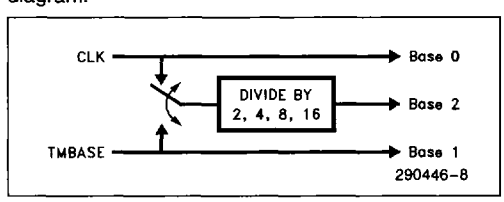


Figure 13. Time Bases

Bits [31:3]	Bit 2	Bits [1:0]

Figure 14. Divider Configuration Register

Bits [31:3] Bits 31 to 3 are reserved. They should be written 0.

Divider Input:

[Bit 2] Selects whether divider's input connects to the 82489DX Local Unit's CLK pin or TMBASE pin.

0: means the divider takes its input signal from CLK,

1: means use TMBASE.

Divide By: [Bits 1,0]

This field selects by how much the divider divides.

00: divide by 2

01: divide by 4

10: divide by 8

11: divide by 16

Timer

Software starts a timer going by programming its Initial Count Register. The timer copies this value into the Current Count Register and starts counting down at the rate of one count for each time base pulse. The time is one of Base 0, Base 1, or Base 2.

The timer has a programmable mode which can be One-Shot or Periodic. After the timer reaches zero in One-Shot mode, the timer simply stays at zero until it is reprogrammed. In Periodic mode, the timer automatically reloads its Current Count from the Initial Count and starts counting down again.

For the timer, interrupt generation can be disabled or enabled, and an arbitrary interrupt vector can be specified. When enabled and the timer reaches zero, an interrupt is generated at the 82489DX Local Unit. Timer generated interrupts are always treated as edges. They can only generate maskable interrupts to the local processor.

A timer set up with its interrupt masked is useful as a time base that can be sampled by the local processor by reading the Current Count Register, for the purpose of measuring the intervals. By mapping the 82489DX's register space into a read-only user page, safe and efficient performance monitoring of user programs can be supported.

If necessary, software may want to ensure that periodic timer interrupts on the different 82489DX Local Units are staggered such that the 82489DXs don't all deliver their interrupt (e.g., a timer slice interrupt) to their local processor at the same time. This staggering avoids bursts of contention for shared resources (bus, cache lines, dispatch queue, locks). Randomness occurring "naturally" may be sufficient to ensure staggering.

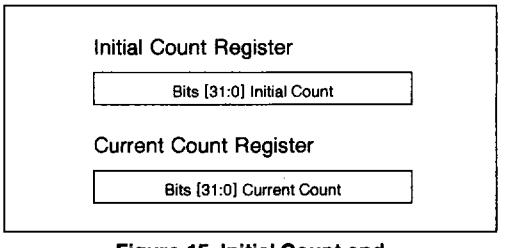


Figure 15. Initial Count and Current Count Registers

Initial Count: Software writes to this register to set the initial count for timer. This register can be written at any time. When written, its value is copied to the Current Count Register and countdown starts or continues from there. The Initial Count Register is read-write by software.

Current Count: This is the current count of timer. It is read-only by software and can be read at any time.

The timer is configured via its Local Vector Table entry shown below (see also Interrupt Control in this section).

Vector: [Bits 7–0]

This is the 8-bit interrupt vector to be used when timer generates an interrupt.

TIMER VECTOR TABLE

Bits [31:20]	Bits [19:18]	Bit 17	Bit 16	Bits [15:13]	Bit 12	Bits [11:8],	Bits [7:0]

Figure 16. Local Vector Table: Timer Entry

Bits 11–8 Reserved. Should be written 0.

Delivery Status: [Bit 12]

Delivery Status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined:

0: (Idle) means that there is currently no activity for this interrupt;

1: (Send Pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by other recently injected interrupts that are in the process of being delivered; Delivery Status is software read-only; software writes to this field (as part of a 32-bit word) do not affect this bit.

Bits 15–13: Reserved. Should be written 0.

MASK: [Bit 16]

This bit serves to mask timer interrupt generation.

0: means not masked, when timer reaches 0, it generates an interrupt with vector at the 82489DX Local Unit

1: means masked, and no interrupt is generated.

Timer Mode: [Bits 17]

This field indicates the operation mode of timer.

0: (One-Shot): the Current Count Register remains at zero after the timer reaches zero, and software needs to reassign the timer's Initial Count Register to rearm the timer.

1: (Periodic): when the timer reaches zero, the Current Count Register is automatically reloaded with the value in the Initial Count Register, and the timer counts down again.

Timer Base: [Bits 19,18]

This field selects the time base input to be used by timer.

00: (Base 0) uses "CLKIN" as input;

01: (Base 1) uses "TMBASE";

10: (Base 2) uses the output of the divider (Base 2).

Bits [31:20] Bits [31:20] are Reserved. Should be written 0.

7.0 82489DX I/O UNIT REGISTERS

REGISTERS ADDRESSING SCHEME

The I/O Unit indirect addressing scheme uses two registers directly mapped into the processor's address space: the I/O Register Select register and the I/O Window register. The I/O register select register selects which I/O unit Register appears in the I/O Window register where it can be manipulated by software.

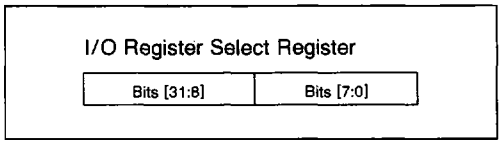


Figure 17. I/O Register Select Register

Bits [31:8]: Reserved. Should be written 0.

Bits [7:0]: I/O REGISTER SELECT: This register selects an 82489DX I/O unit register. The contents of the selected 32-bit register can be manipulated via the I/O Window Register. The I/O Register Select register is read-write by software.

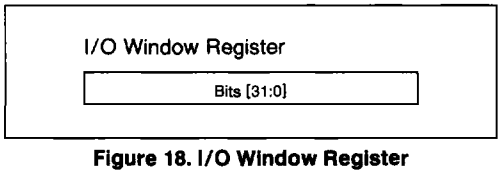


Figure 18. I/O Window Register

Bits [31:0] I/O WINDOW REGISTER: This register is mapped onto the I/O Unit's register selected by the I/O Register Select register. Readability/writability by software is determined by the I/O unit register that is currently selected.

The addresses (offsets to a platform-defined base address) of all registers are listed in the register summary section. Note that register offsets are aligned on 128-bit boundaries; in other words, registers are located only at every fourth 32-bit address. This eliminates the need for lane-steering glue logic when connecting the 82489DX's 32-bit data bus to a wider (64-bit and 128-bit) bus.

82489DX I/O UNIT CONFIGURATION

I/O Unit ID Register

Each 82489DX I/O Unit has a register that contains the I/O Unit's 8-bit ID. The I/O unit ID serves as a physical name of the 82489DX I/O Unit. It is used in arbitrating for ICC bus ownership when the I/O unit wants to access the ICC bus for sending any interrupt message. Unlike the local unit ID, the I/O unit ID is not latched-in from the address bus during hardware reset. The I/O unit ID is set to 0 during reset. The software has to write different ID into the I/O Units before starting interrupt messages on the ICC bus.

I/O Unit ID

Bits [31:24]	Bits [23:0]

Bits [31:24] I/O Unit ID:

The I/O unit ID serves as the physical "name" of the 82489DX unit used for arbitration purposes for the ICC bus usage. In a system with, say, four 82489DX, there are 4 Local Units and 4 I/O Units. All the 8 units should be assigned different ID. The IDs should start with 0 and each unit should have different ID.

Bits [23:0] Bits 23..0 are reserved. Should be written 0.

I/O Unit Version Register

Each 82489DX I/O Unit contains a hardware Version Register that identifies this 82489DX I/O unit version. This register is read only.

I/O Unit Version Register

Bits [31:24]	Bits [23:16]	Bits [15:8]	Bits [7:0]

Version: [Bits 7–0]

This is a version number that identifies this version. This field is hardwired and is read-only. Will be read as "1" for 82489DX.

Bits [15:8] Bits [15:8] are reserved.

Max Redir Entry: [Bits 23–16]

This is the entry number (0 being the lowest entry) of the highest entry in the Redirection Table. It is equal to the number of Interrupt Input Pins minus one of this I/O Unit. This field is hardwired and is read-only.

In the 82489DX I/O unit this is read as 15.

Bits [31:24] Bits [31:24] are reserved.

I/O UNIT INTERRUPT SOURCE REGISTERS

Redirection Tables

The Redirection Table has a dedicated entry for each interrupt input pin. Unlike IRQ pins of the 8259A, the notion of interrupt priority is completely unrelated to the position of the physical interrupt input pin on the 82489DX. Instead, software can decide for each pin individually what it wants the vector (and therefore the priority) of the corresponding interrupt to be. For each individual pin, the operating system can also specify whether the interrupt is signaled as edges or levels, as well as the destination and delivery mode of the interrupt. The information in the Redirection Table is used to translate the interrupt manifestation on the corresponding interrupt pin into an inter-82489DX message.

In order for a signal on an edge-sensitive Interrupt Input pin to be recognized as a valid edge (and not a glitch) the input level on the pin must remain asserted until the time 82489DX I/O Unit sends the corresponding message over the ICC bus. Only then will the source 82489DX be able to recognize a new edge on that Interrupt Input pin. That new edge will only result in a new invocation of the handler if its acceptance by the destination 82489DX causes the Interrupt Request Register bit to go from 0 to 1. (In other words, if the interrupt wasn't already pending at the destination.)

82489DX I/O unit has 16 Redirection Table entries. The layout of an entry in the Redirection Table is as follows:

Redirection Table Entry

Bits [31:17]	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]

Vector (Bits [7:0]

Interrupt vector for this interrupt

Delivery Mode (Bits [10:8])

000: Fixed

001: Lowest Priority

010: <reserved>

011: <reserved>

100: NMI

101: Reset

110: <reserved>

111: ExtINT

Destination Mode (Bit 11)

0: Physical

1: Logical

Delivery Status (Bit 12)

0: Idle

1: Send Pending

Bit 13 Bit 13: Reserved. Should be written 0.

Remote IRR (Bit 14)

Reflects the Remote IRR bit

0: Remote IRR bit is clear.

1: Remote IRR bit is set.

Trigger Mode (Bit 15)

0: Edge

1: Level

Mask (Bit 16)

0: Not Masked

1: Masked

Bits [31:17] Reserved. Should be written 0.

DESCRIPTIONS

Vector: [Bits 7–0]

The vector field is an 8-bit field containing the interrupt for this interrupt.

Delivery Mode: [Bits 10–8]

The Delivery Mode is a 3-bit field that specifies how the 82489DXs listed in the destination field should act upon reception of this signal. Note that remote read is not supported for I/O device interrupts. Note that certain Delivery Modes will only operate as intended when used in conjuction with a specific Trigger Mode. These restrictions are indicated for each Delivery Mode.

000: (Fixed) means deliver the signal on the INT pin of all processors listed in the destination. Trigger Mode for "fixed" Delivery Mode can be edge or level.

001: (Lowest Priority) means deliver the signal on the INT pin of the processor that is executing at the lower priority among all the processors listed in the specified destination; Trigger Mode for "lowest priority" Delivery Mode can be edge or level.

100: (NMI) means deliver the signal on the NMI pin of all processors listed in the destination, vector information is ignored. A Delivery Mode equal to "NMI" requires a "level" Trigger Mode.

101: (Reset) means deliver the signal to all processors listed in the destination by asserting/deasserting the 82489DX's Reset output pin. All addressed 82489DXs' Local Units will assume their reset state but preserve their unit ID. One side effect of a unit message with Delivery Mode equal to "Reset" that results in a deassert of reset is that all 82489DXs' Local Units (whether listed in the destination or not) will reset their lowest-priority tie breaker arbitration ID to their unit ID (see the section on the ICC bus for details). A Delivery Mode of "Reset" requires a "level" Trigger Mode.

111: (ExtINT) means deliver the signal to the INT pin of all processors listed in the destination as an interrupt that originated in an externally connected (8259A-compatible) interrupt controller. The Local Unit receiving this interrupt will activate ExtINTA in response to this interrupt message. A Delivery Mode of "ExtINT" requires an "edge" Trigger Mode. (See the section on Compatibility for details.)

Destination Mode [Bit 11]

This field determines the interpretation of the Destination field.

0: (Physical Mode): in Physical Mode, a destination 82489DX Local Unit is identified by its unit ID. Bits 56 through 63 (8 MSB of the destination field) specify the 8-bit unit ID.

1: (Logical Mode): in Logical Mode, destinations are identified by matching on Logical Destination under the control of the Destination Format Register in each 82489DX Local Unit. The 32-bit Destination field is the logical destination.

Delivery Status: [Bit 12]

Delivery Status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined:

0: (Idle) means that there is currently no activity for this interrupt;

1: (Send Pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by other recently injected interrupts that are in the process of being delivered; Delivery Status is software read-only; software writes to this field (as part of a 32-bit word) do not affect this bit.

Bit 13: Bit 13 is Reserved. Should be written 0.

Remote IRR: [Bit 14]

This bit is used for level triggered interrupts; its meaning is undefined for edge-triggered interrupts. Remote IRR mirrors the interrupt's IRR bit of the destination 82489DX Local Unit. When the value of the bit disagrees with the state of the Interrupt Input line, a unit message is automatically sent to make the destination's IRR both reflect the new state of the Interrupt Input line, and then the Remote IRR bit is updated to track its associated IRR bit. Remote IRR is software read-only; software writes to this bit do not affect it.

Trigger Mode: [Bit 15]

The Trigger Mode field indicates the type of signal on the interrupt pin that triggers an interrupt.

0: indicates edge sensitive,

1: indicates level sensitive.

Mask: [Bit 16]

Use this bit to mask injection of this interrupt.

0: indicates that injection of this interrupt is not masked. An edge or level on an interrupt pin that is not masked results in the delivery of the interrupt to the destination.

1: indicates that injection of this interrupt is masked. Edge-sensitive interrupts signaled on a masked interrupt Input pin are simply ignored (i.e., it is not delivered and is not held pending). Level-asserts or deasserts occurring on a masked level-sensitive pin are also ignored and have no side effects. As expected, changing the mask bit from unmasked to masked while the level remains asserted has the side effect of deasserting the level. It is software's responsibility to deal with the case where the Mask bit is set after the interrupt message has been sent but before the interrupt is dispensed to the processor.

Bits [31:17] Bits [31:17] are reserved. Should be written 0.

Destination

Bits [63:32]

Destination: If the Destination Mode of this entry is "Physical Mode", then the 8 MSB [bits 56 through 63] contain an 82489DX Local Unit ID. If Logical Mode, then the Destination field potentially defines a set of processors. The interpretation of the 32-bit destination field is further enabled by the Destination Format Register in the 82489DX Local Units.

4

8.0 ICC BUS DEFINITION

Physical Characteristics

The ICC bus is a 5-wire synchronous bus connecting all 82489DXs (all I/O Units and all Local Units). Four of these five wires are used for data transmissions and arbitration, and one wire is a clock. The description refers to the logical state of the ICC bus. Electrical levels are just inverse of the logical state described. For example, the section describes that the ICC bus is 0000 when not transmitting any message. This refers to logical state. Electrically, the ICC bus is 1111 when not transmitting any message.

The bus is electrically an open-drain connection providing for both bus use arbitration and arbitration for lowest priority. Being open-drain, the bus is run at a "comfortable" speed such that design-specific termination tuning is not required. Furthermore, each 82489DX receiving a message or participating in an arbitration must be given enough time in a single bus cycle to latch the bus and perform some simple logic operations on the latched information in order to determine whether the next drive cycle must be inhibited.

Note that it is likely in MP systems that additional processors be located on plug-in boards. Since the ICC bus would be part of the connector, the 82489DX to ICC bus connection is defined so that it can be electrically isolated using external drivers. The 82489DX has separate ICC bus input and output pins that can be connected externally to the 82489DX to either provide or not provide isolation.

The isolation can also be used to provide a hierarchical connection of ICC buses electrically supporting large numbers of processors. The number of 82489DXs supported using the hierarchical connection is limited only by ICC bus bandwidth. It should be noted that ICC bus output low current is just 4 mA.

Bus Arbitration

Arbitration (both for use of the bus and for determining the lowest priority 82489DX) depends on all 82489DX message units operating synchronously. To deal with the event where multiple agents start transmitting simultaneously, a distributed arbitration approach is used. Bus arbitration uses a small number of arbitration cycles in the ICC bus. During

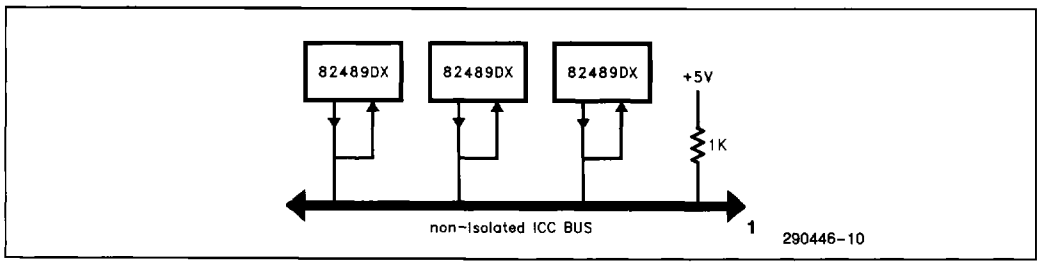


Figure 20. ICC Bus: Simple Direct Connection

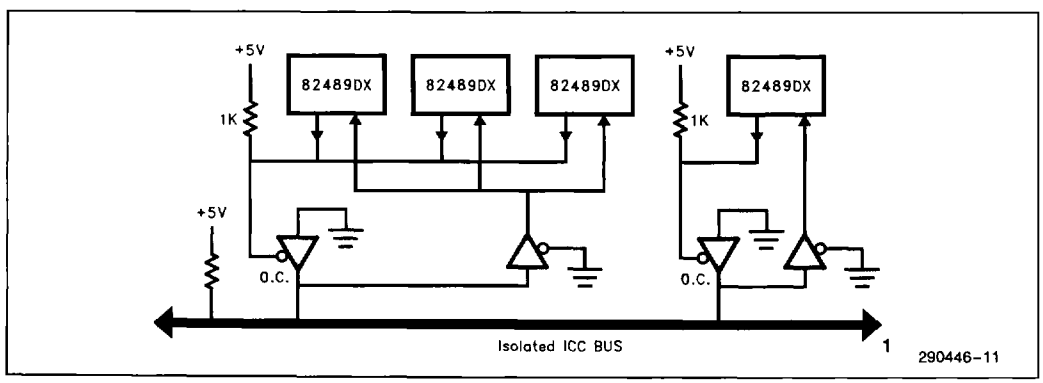


Figure 21. ICC Bits: Hierachical Connection

these cycles, arbitration losers progressively drop off the bus until only the winner remains transmitting. The winner then transmits its actual inter-unit message. Once the sending of a message (including bus arbitration) has started, any possible contender must suppress transmission until enough cycles have elapsed for the message to be fully sent. The number of message cycles depends on the type of message being sent.

A bus arbitration cycle starts by the agent driving its unit ID on the ICC bus. High-order ID bits are driven first, successive cycles proceeding to the low bits of the ID. All losers in a given cycle drop off the bus, using every subsequent cycle as a tie breaker for the previous cycle. By the time all arbitration cycles are completed, there will be only a single agent left driving the bus.

The 8-bit unit ID (I7 I6 I5 I4 I3 I2 I1 I0) is chopped up in successive groups of 2 bits (I7 I6)(I5 I4)(I3 I2)(I1 I0). Each of these tuples is first decoded before driving them on the bus. The 0s and 1 indicate logical levels and not signal levels. The ICC bus is 0000 when not transmitting any message. The decoding used is:

ID Tuple		→	ICC Bus			
(I[i + 1]	I[i])		B3	B2	B1	B0
0	0	→	0	0	0	1
0	1	→	0	0	1	0
1	0	→	0	1	0	0
1	1	→	1	0	0	0

Note that the pattern generated on the ICC bus by tuple (I3 I2) will be represented as i32 i32 i32 i32. The lower case signifies this encoding.

Each tuple of the ID only contributes to a single wire, making it possible for an agent to determine with certainty whether to "drop off" or to continue arbitrating in the next cycle for the following two bits of the unit ID simply by checking whether the bus line the agent is driving is also the highest order 1 on the bus. Each ICC bus cycle therefore arbitrates 2 bits.

1:	i76	i76	i76	i76	ICC bus arbitration
2:	i54	i54	i54	i54	
3:	i32	i32	i32	i32	
4:	i10	i10	i10	i10	
	<message body>				

Lowest-Priority Arbitration

Arbitration is also used to find the 82489DX Local Unit with the lowest processor priority. Lowest-priority arbitration uses the value of the 82489DX's Processor Priority value appended with an 8-bit Arbitration ID (Arb ID) to break ties in case there are multiple units executing at the lowest priority.

Using the constant 8-bit unit ID as the Arb ID has a tendency to skew symmetry since it would favor 82489DXs with low ID values. An 82489DX Local Unit's Arb ID is therefore not the unit ID itself but is derived from it. At reset, an 82489DX Local Unit's Arb ID is equal to its unit ID. Each time a message is broadcast over the ICC bus in lowest priority mode, all 82489DX Local Units increment their Arb ID by one, which gives them a different Arb ID value for the next arbitration. The Arb ID is then endian-reversed (LSB becomes MSB, etc.) to ensure better rotation of which 82489DX gets to have the lowest Arb ID next time around. The reversed Arb ID is then decoded to generate arbitration signals on the ICC bus as described above.

To support hot insertion of processor boards in a running MP system, a mechanism is provided to allow the 82489DX of the added processor to synchronize its Arb ID with the existing 82489DXs. This is accomplished by broadcasting a message with Delivery Mode equal to "Reset", Trigger Mode equal to "Level", and Level equal to 0. This message must be broadcast before the newly added 82489DX is allowed to participate in a lowest-priority arbitration. Depending on the exact sequence under which the newly inserted board is powered-up and initialized, this Arb ID synchronization may occur naturally if a Reset-deassert to the new 82489DX is part of that sequence. If not, the local processor can always send this as an inter processor interrupt (with a null destination), causing only the side effect of resetting all 82489DX Arb IDs.

4

ICC BUS MESSAGE FORMATS

The short message format is described first. Note that the first 19 cycles of both short and long message formats have the same interpretation.

1:	i76	i76	i76	i76	ICC bus arbitration
2:	i54	i54	i54	i54	
3:	i32	i32	i32	i32	
4:	i10	i10	i10	i10	
5:	DM	M2	M1	M0	destination mode and delivery mode
6:	"0"	"0"	L	TM	control bits
7:	V7	V6	V5	V4	vector
8:	V3	V2	V1	V0	

9:	D31	D30	D29	D28	destination
10:	D27	D26	D25	D24	
11:	D23	D22	D21	D20	
12:	D19	D18	D17	D16	
13:	D15	D14	D13	D12	
14:	D11	D10	D09	D08	
15:	D07	D06	D05	D04	
16:	D03	D02	D01	D00	
17:	C	C	C	C	checksum for cycle 5 through 16
18:	"1"	"1"	"1"	"1"	post amble
19:	A	A	A	A	accept (1000 if OK, 1110 if preempt, else error)
20:	"0"	"0"	"0"	"0"	idle 1
21:	"0"	"0"	"0"	"0"	idle 2

Cycles 1 through 4 are bus arbitration as described earlier. Cycle 5 (DM M2 M1 M0) is the Destination Mode which is 0 for Physical mode and 1 for Logical Mode, and the Delivery Mode of the message. The encoding used for the Delivery Mode in the message is identical to the encoding used for the Delivery Mode in the Redirection Table, Local Vector Table, and Interrupt Command Register.

M2	M1	M0	Delivery Mode
0	0	0	Fixed
0	0	1	Lowest Priority
0	1	0	<reserved>
0	1	1	Remote Read
1	0	0	NMI
1	0	1	Reset
1	1	1	ExtINT

Cycle 6 contains the Control Bits of the message. The control bits are:

- TM (Trigger Mode): indicates whether this message corresponds to an edge or level;
- L (Level): indicates whether this is an Assert or a Deassert of a "level" signal. L is undefined when TM is edge.

6: "0" "0" L TM Control Bits
TM = Trigger Mode (0 = edge, 1 = level)
L = Level (0 = deassert, 1 = assert)

The length of the message is derived from the Delivery Mode, the Control Bits, and the Accept cycle of the message.

TM/L (AAAA)

	Edge	Level = Assert	Level = Deassert
Fixed	Short	Short	Short
Lowest Priority	Short (1110)	Short (1110)	Short (1110)
	Long (1000)	Long (1000)	Short
Remote Read	Long	Long	Short
NMI	Short	Short	Short
Reset	Short	Short	Short
ExtINTA	Short	Short	Short

Cycles 7 and 8 are the 8-bit interrupt vector. The vector is only defined for Delivery Modes Fixed, and Lowest-priority. For Delivery Mode of "Remote Read", the vector field contains the address of the register to be read remotely.

If DM is 0 (physical mode), then cycles 9 and 10 are the unit ID and cycles 11 through 16 are zero. If DM is 1 (logical mode), then cycles 9 through 16 are the 32-bit Destination field. The interpretation of the logical mode 32-bit Destination field is performed by the Local Units using the Destination Format Register. The sending 82489DX knows whether it should (incl) or should not (excl) respond to its own message.

Cycle 17 is a checksum over the data in cycles 5 through 16. The checksum is computed by adding all 4-bit quantities of cycles 5 through 16, feeding carry out of the MSB back into the LSB. This protects the data in these cycles against transmission errors. The (single) 82489DX driving the message provides this checksum in cycle 17.

Cycle 18 is a post amble cycle driven as 1111 by the sending 82489DX allowing all 82489DXs to perform various internal computations based on the information contained in the received message. One of the computations takes the computed checksum of the data received in cycles 5 through 16 and compares it against the value in cycle 17. If any 82489DX computes different checksum than the one passed in cycle 17, then that 82489DX will signal an error on the ICC bus in cycle 19 by driving it as 1111. If this happens, all 82489DXs will assume the message was never sent and the sender must try sending the message again, which includes re-arbitrating for the ICC bus. In lowest priority delivery when the interrupt has a focus processor, the focus 82489DX will signal this by driving 1110 during cycle 19. This tells all the other 82489DXs that the interrupt has been accepted, the 82489DXs is preempted, and short message format is used. All (non-focus) 82489DXs will drive 1000 in cycle 19. Under lowest priority mode, 1000 implies that the interrupt currently has no focus processor and that priority arbitration is required to complete the delivery. In that case, long message format is used. If cycle 19 is 1000 for non Lowest Priority mode, then the message has been accepted and is considered sent.

	19:EEEE
1000	OK
1110	preempt
<others>	error (drive error as 1111)

When an 82489DX detects and reports an error during the error cycle, that 82489DX will simply listen to the bus until it encounters two consecutive idle (0000) cycles. These two idle cycles indicate that the message has passed and a new message may be started by anyone. This allows an 82489DX that got itself out of cycle on the ICC bus to get back in sync with the other 82489DXs.

Long Message Format

Cycles 1 through 19 of the long message format are identical to cycles 1 through 19 of the short message format. As mentioned, long message format is used in two cases:

(1) Lowest Priority delivery when the interrupt does not have a focus. Cycles 20 through 27 are eight arbitration cycles where the destination 82489DXs determine the one 82489DX with lowest processor priority/ARB ID value.

(2) Remote Read messages. Cycles 20 through 27 are the 32-bit content of the remotely read register. This information is driven on the bus by the remote 82489DX.

Cycle 28 is an Accept cycle. In lowest priority delivery, all 82489DXs that did not win the arbitration (including those that did not participate in the arbitration) drive cycle 28 with 1000 (co accept), while the winner 82489DX drives 1111. If cycle 28 reads 1111, then all 82489DXs know that the interrupt has been accepted and the message is considered delivered. If cycle 28 reads 1100 (or anything but 1111 for that matter), then all 82489DXs assume the message was unaccepted or an error occurred during arbitration. The message is considered undelivered, and the sending 82489DX will try delivering the message again.

For Remote Read messages, cycle 28 is driven as 1100 by all 8 2489DXs except the responding remote 82489DX, who drives the bus as 1111 in case it was able to successfully supply the requested data in cycles 20 through 27. If cycle 28 reads 1111 the data in cycles 20 through 27 is considered valid; otherwise, the data is considered invalid. The source 82489DX that issued the Remote Read uses cycle 28 to determine the state of the Remote Read Status field in the Interrupt Command Register (valid or invalid). In any case, a Remote Read request is always successful (although the data may be valid or invalid) in that a Remote Read is never retried. The reason for this is that Remote Read is a debug feature, and a "hung" remote 82489DX that is unable to respond should not cause the debugger to hang.

Cycles 29 and 30 are two idle cycles. The ICC bus is available for sending the next message at cycle 31. The two idle cycles at the end of both short and long messages, together with non zero (i.e., non idle) encoding for certain other bus cycles allow an ICC bus agent that happens to be out of phase by one cycle to sync back up in one message simply by waiting for two consecutive idle cycles after reporting its checksum error. This makes use of the fact that valid arbitration cycles are never 0000.

1:	i76	i76	i76	i76	ICC bus arbitration
2:	i54	i54	i54	i54	
3:	i32	i32	i32	i32	
4:	i10	i10	i10	i10	
5:	DM	M2	M1	M0	delivery mode
6:	"0"	"0"	L	TM	control bits
7:	V7	V6	V5	V4	vector
8:	V3	V2	V1	V0	
9:	D31	D30	D29	D28	destination
10:	D27	D26	D25	D24	
11:	D23	D22	D21	D20	
12:	D19	D18	D17	D16	
13:	D15	D14	D13	D12	
14:	D11	D10	D09	D08	
15:	D07	D06	D05	D04	
16:	D03	D02	D01	D00	
17:	C	C	C	C	checksum for cycles 5 through 16
18:	"1"	"1"	"1"	"1"	postamble
19:	A	A	A	A	accept (1000 if OK, 1110 if preempt, else error)
20:	p76	p76	p76	p76	lowest priority arbitration or 32 bits of remote register
21:	p54	p54	p54	p54	processor priority
22:	p32	p32	p32	p32	
23:	p10	p10	p10	p10	
24:	a76	a76	a76	a76	
25:	a54	a54	a54	a54	arbitration ID
26:	a32	a32	a32	a32	
27:	a10	a10	a10	a10	
28:	A	A	A	A	accept
29:	"00"	"0"	"0"	"0"	idle1
30:	"0"	"0"	"0"	"0"	idle2

9.0 HARDWARE TIMINGS

This section covers the following:

— Timing Diagram Notation

— 82489DX Register Access Timing Diagrams with Descriptions

A block diagram of the configuration of the CPU module of a MP system is shown. This in no way is intended to be a complete representation of 486/Intel Cache/Intel Cache Controller connections. It is intended to show all the 82489DX connections, and how they connect to other components on and off the module. This module has arbitrarily been drawn with a 64-bit data bus to show how the expanded address space architecture fits. The unit can be similarly attached to either a 32-bit or 128-bit data bus, with total transparency to shrink-wrap software.

In this configuration, the 82489DX uses the same clock source as the processor and cache. However, it is quite possible to consider 82489DX as a memory bus device and hence supply 82489DX with the memory bus clock, which can be slower than the CPU module clock frequency.

In the configuration shown, the processor's INT and NMI pins could be supplied by other source to allow for the possibility that 82489DX can be totally bypassed if desired, by allowing those signals to be driven from off the module while the 82489DX is disabled. The reset signal generated by the 82489DX goes to the MBC (memory bus controller) which is required to drive configuration lines at reset time. This would probably be configured as a "warm" reset by the MBC.

A future version of cache controller may generate the chip select for 82489DX at a fixed memory location of hexFEE00000. By having the cache controller to provide the chip select signal, it would encourage a standard mapping for 82489DX address space. In some MBC designs, this signal should be connected to the MBC since 82489DX cycles limit bus pipelining by constraining how soon the next bus cycle can come. The 82489DX chip select can be generated by the MBC completely.

The address, data and most of the bus control signals share the respective bus with cache and cache controller. The block diagram shows attachment for only 6 address lines: A4–A9. A10 should be 0. This is all the 82489DX needs for operation, however, if the address lines are used to initialize 82489DX local ID at reset time, 8 address lines are required, A3–A10.

INTERFACING TO THE ICC BUS

The 82489DX has separate ICC bus input and output pins to facilitate using external drivers. The ICC bus input pins (MBI0-3) are TTL-level compatible CMOS inputs. The output pins (MBO0-3) are open-drain pins which required external pull-ups. The open-drain output buffers are small buffers with:

Sink current of < 4 mA. Special consideration must be exercised when driving large capacitive loads or long transmission lines. The pull-up resistor and the capacitive load constitute RC time constant that will affect the output transition times. This in turn will limit the operating frequency of the ICC bus.

When designing in the ICC bus, one needs to consider the loads that each 82489DX will be driving and whether external drivers should be used. In most situations, the ICC bus driven high (MBO pins pulled high by the external pull-up resistors) poses the most challenge. Simulating the target design on an electrical simulator (such as SPICE) will help greatly. as shown in the following examples.

First Order Buffer Models

Figure 21a and 21b are first order input buffer and output buffer models of the MBI and MBO pins. The open-drain of the MBO is modeled as a switch as the primary interest here is the MBO pins going high. These models can be used on SPICE simulations to obtain first order behaviors. The parameters for these models are as follows:

Cp (package capacitance) = 3 pF
Lp (package inductance) = 15 nH
Rb (bond wire resistance) = 0.08Ω
Ci (input buffer capacitance) = 3 pF
Co (output buffer capacitance) = 6 pF
Ro (output buffer impedance) = 30Ω-80Ω

MBO Pull-up Resistor

To minimize the RC time constant, one would like to use the smallest pull-up resistor value possible. The MBO pins has a worst case Iol-spec of 4 mA and V_{CC} = 4.75V. This translates to a minimum pull-up of about 1 KΩ. Where stronger drive is needed (smaller pull-up resistors), external drivers must be used.

Driving Lumped Capacitance

In systems where external drivers are not used, the MBI pins will be tied to the MBO pins. Figure 21d is a SPICE simulation of the MBO output with a 1 KΩ pull-up driving lumped capacitive loads from 10 pF to 150 pF.

At a load of 50 pF, it takes about 30 ns to charge up to 2V. At 100 pF, it takes an additional 25 ns. Figure 21d can be used to estimate the loading delay at different lumped capacitive loads.

4

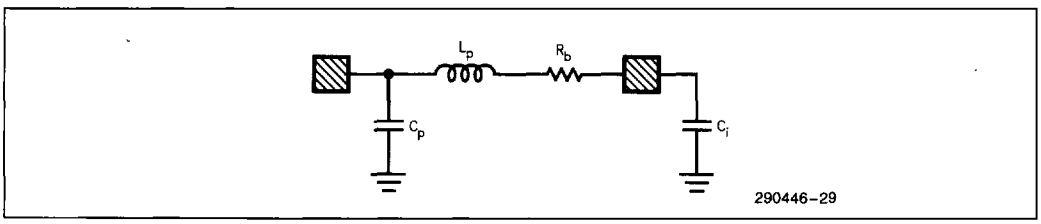


Figure 21a. First Order Input Buffer for MBI Pins

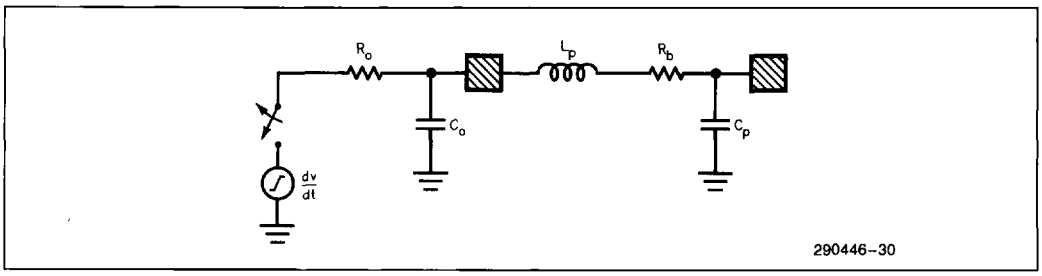


Figure 21b. First Order Open-Drain Output Buffer for MBO Pins

In real systems, the loads are made up of lumped capacitance and transmission lines. More accurate results can be obtained using transmission line models.

Driving Transmission Lines

Two device model

In this example the ICC bus is a signal line on an FR-4 printed circuit board. The line width is 6 mils. Line length of 12 inches and 18 inches are modeled. The FR-4 PC board has the following characteristics:

resistivity =	0.6 mΩ/sq. (0.1Ω/inch for 6 mil width)
inductance =	60 pH/sq. (10 nH/inch for 6 mil width)
capacitance =	0.55 nF/sq. in. (3.3 pF/inch for 6 mil width)

The ICC bus is shared by two 82489DXs, one at each end. The ICC bus is modeled as a transmission line. For the simulation, only one of the 82489DX is driving. A pull-up resistor of 2 KΩ is used at each end (1 KΩ equivalent value) as shown in Figure 21e. Figure 21f shows the signals at each end of the 12 inch transmission line. Trace 1 is the wave form at the driven end and trace 2 is the signal at the receiving end of the line. The 2 ns delay between the two signals is the propagation delay (or flight time) through the 12 inch transmission line. It takes about 35 ns for the voltage to charge up to 2V.

Figure 21g shows the received signal with different line length and with additional lumped capacitance. Trace 1 is for 12 inch only. Trace 2 is for 12 inch with additional 20 pF lumped capacitance to represent interconnect socket capacitance. Trace 3 is for 18 inch plus 20 pF. The presence of the 20 pF at each end of the 12-inch transmission line increases the delay time by 20 ns at 2V.

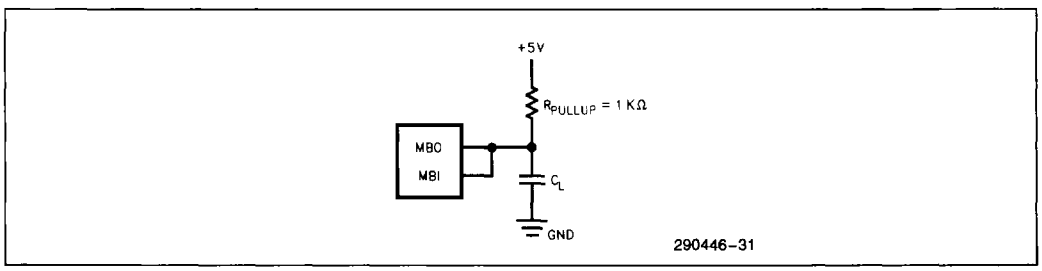


Figure 21c. ICC Bus Driving Lumped Capacitance

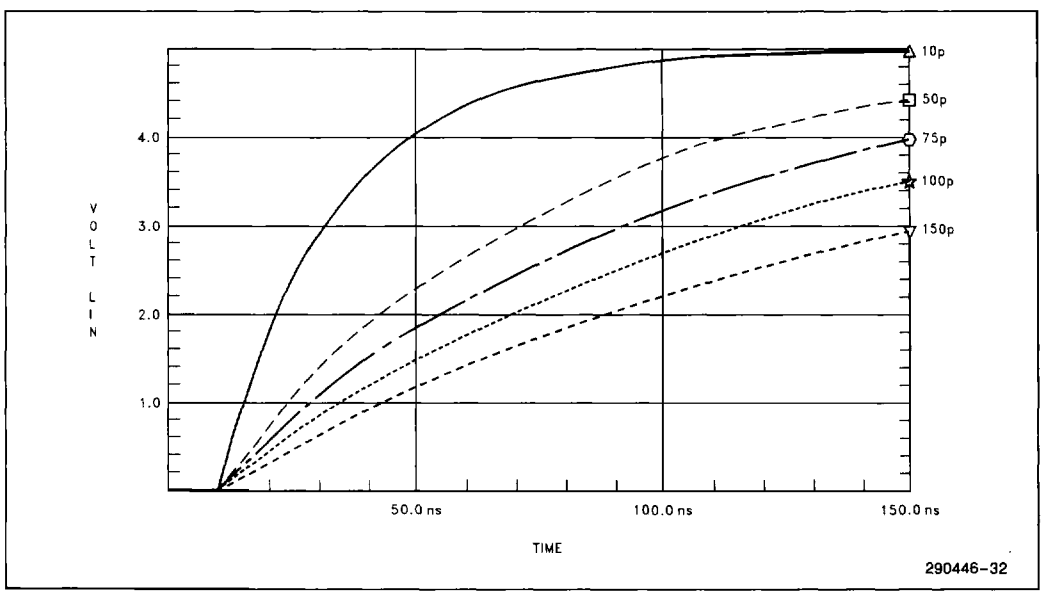


Figure 21d. 1 KΩ Pull-Up Driving Lumped Capacitance

Four Device Model

In this example (Figure 21h), the ICC bus is a 12-inch transmission line with four 82489DXs connected at 4 inch intervals. The loading at each junction consisted of the MBI and MBO buffers and a 20 pF lumped capacitance. 2 KΩ pull-ups are at each end of the transmission line.

As shown in Figure 21i, it takes more than 90 ns for the signal level at both ends to reach 2V.

One way to improve the low to high transition time is to use a stronger pull-up (smaller resistor value) which is possible using external line drivers with their larger current drive capabilities.

Figure 21j shows the difference in output when the model is used with 300Ω pull-ups at each end of the transmission line.

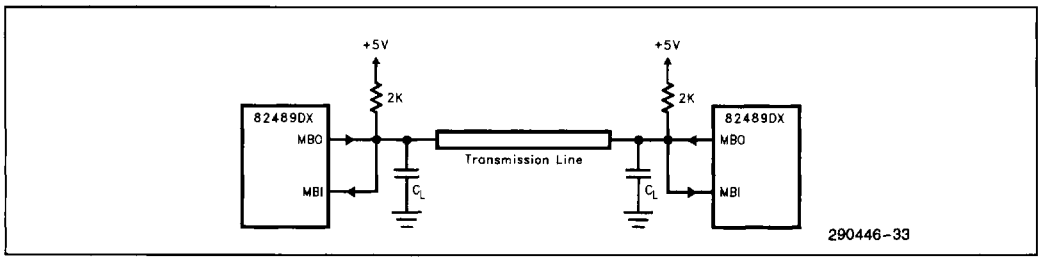


Figure 21e. Unbuffered ICC Bus with Two 82489DX

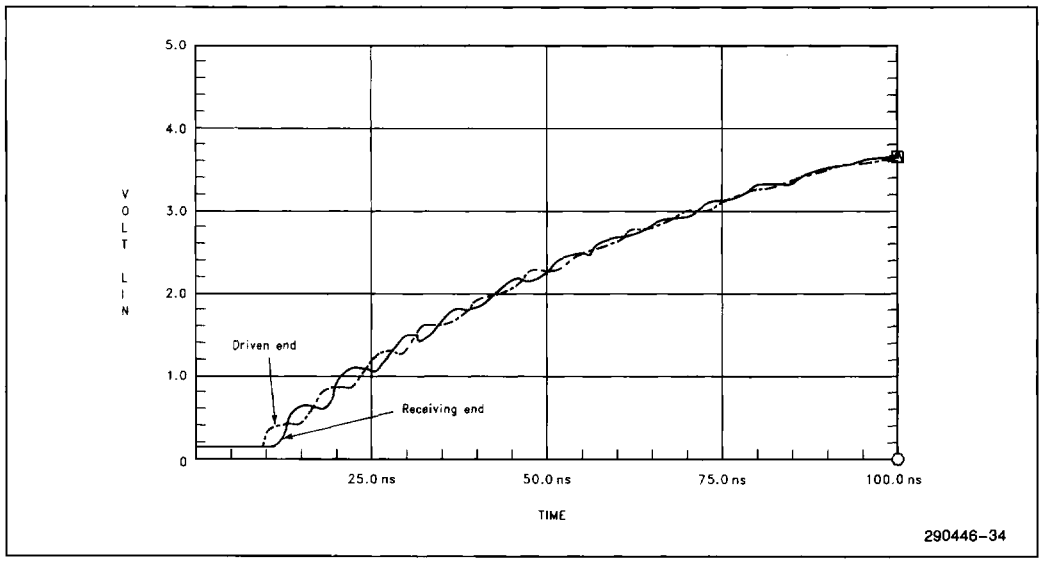


Figure 21f. Waveform at Both Ends of 12" T-line

4

External Drivers/Buffered ICC bus

The 82489DX has separate ICC Bus input (MBI) and output (MBO) pins that can be connected to external line drivers in systems that has appreciable loading on the ICC Bus or where modularity of the bus is needed.

Figure 21k is a typical implementation using external drivers with tri-state outputs. Drivers such as 74F125 or its equivalent can be used. The drivers should be placed as close to the MBO pins as possible. The input buffer on MBI is optional depending on the users ICC Bus scheme. The total delay through the drivers, buffers, transmission line, clock skews etc. must be calculated to ensure that all the ICC bus timing requirements are met.

A hierarchical bus connection can also be used in applications that cannot afford driver/buffer per unit and where bus loading are localized in cluster groups. Figure 21l shows such a connection where each cluster group is connected directly and drivers are used to connect to other clusters. Each cluster group is assumed to be close together physically with small loading on the local ICC bus.

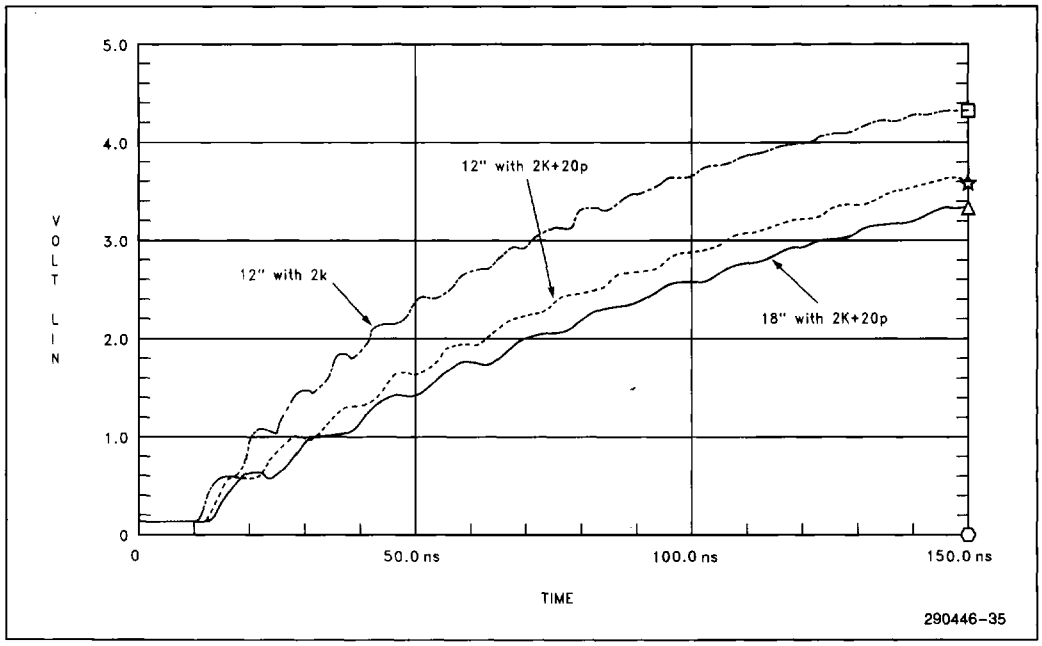


Figure 21g. Waveform at End of T-line with Load

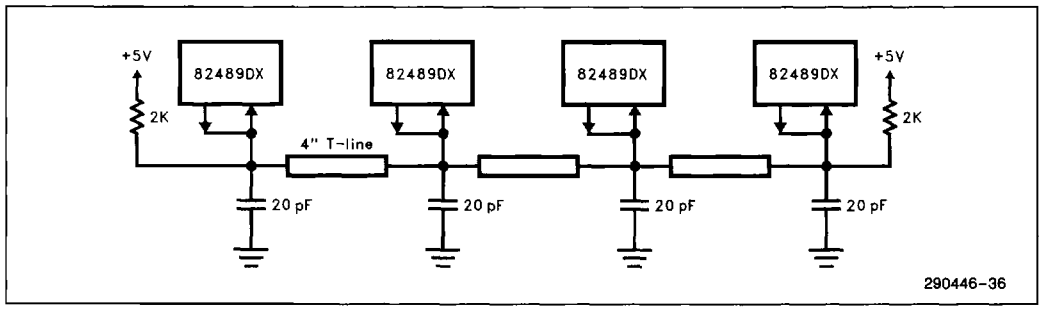


Figure 21h. Four 82489DX Configuration

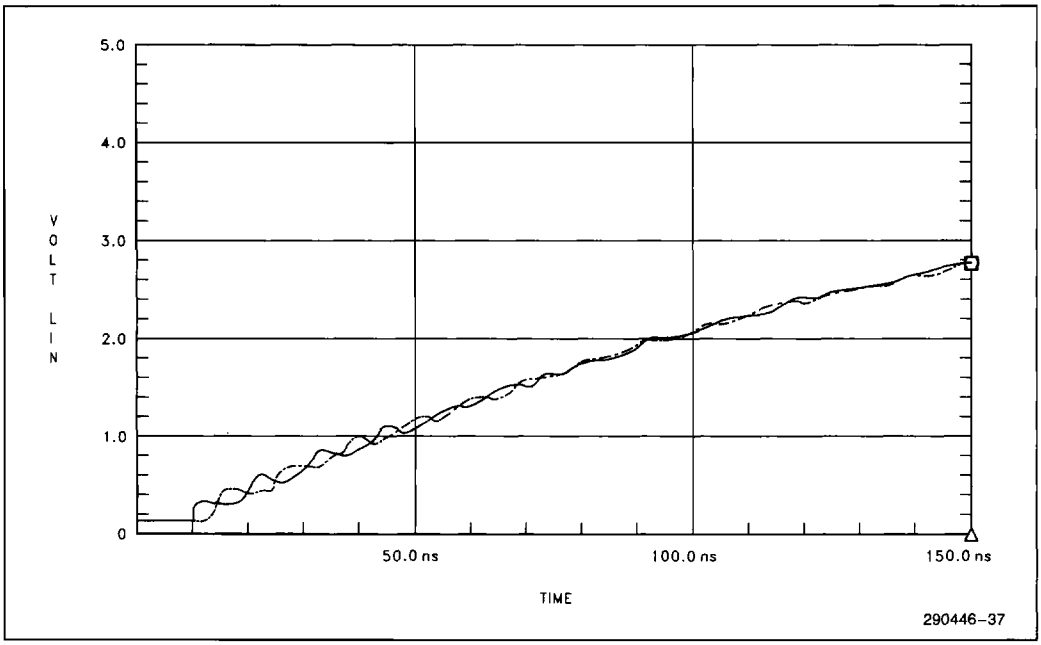


Figure 21i. Waveform for Four Devices on 12" T-line

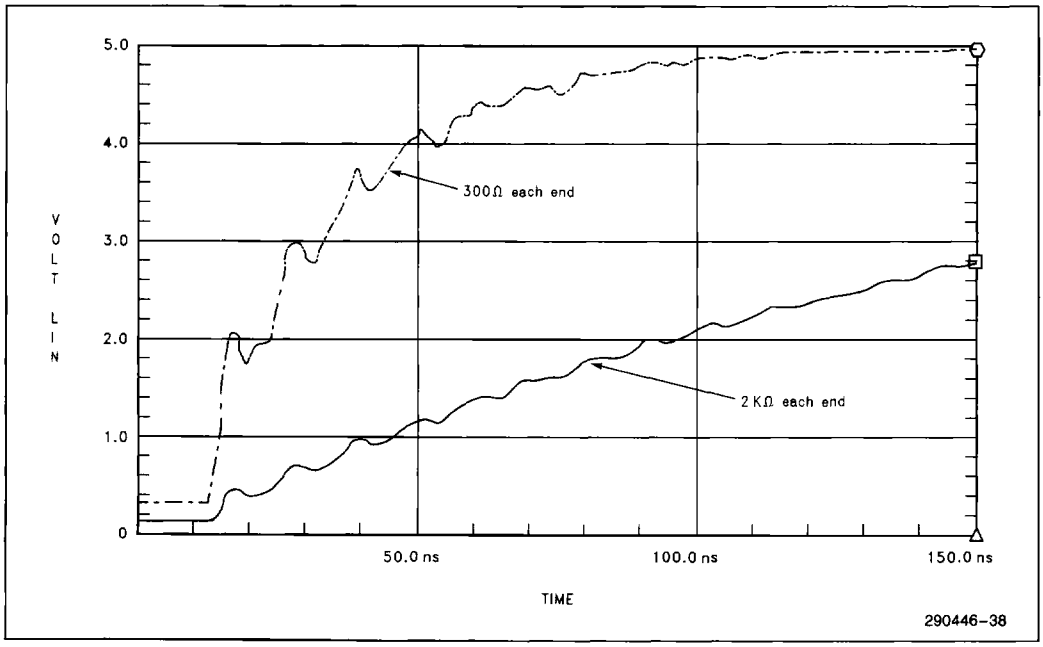


Figure 21j. Waveform with Different Pull-Ups

Transmission Line Termination

As with all high speed designs, one has to consider transmission line effects on signals, especially clock signals. Even though the ICC bus clock, ICLK is usually operated in the 10 MHz range, one has to consider proper transmission line termination also for short rise times. Figure 21m shows the ICLK wave form at the end of a 12 inch T-line when driven by a clock generator with and without series matched termination.

Series termination should not be used for the ICC bus data lines (MBO). The combination of the pull-up resistor and series resistor would degrade the output low voltage, Vol. For example, with a pull-up of 300Ω and a series termination of 50Ω at each end, the Vol voltage at the receiving end would be at 1.55V if the driving end is at 0.4V (see Figure 21n).

ICC BUS Operating Frequency

The 82489DX ICC BUS has a design target of operating up to 16 MHz (62 ns period). As shown in the examples above, the MBO low-to-high transition times are strongly dictated by the loads and the pull-ups used. This will in turn affect the maximum operating frequency of ICLK.

In general, the minimum period is the larger of 62 ns or MBO-to-MBI low data time or MBO-to-MBI high data time.

MBO-to-MBI low time =
(ICLK skew + MBO valid low delay + T-line prop.delay + ext. buffer delay + MBI setup time)

MBO-to-MBI high time =
(ICLK skew + MBO Hi-Z delay + pull-high time + T-line prop.delay + ext. buffer delay + MBI setup time)

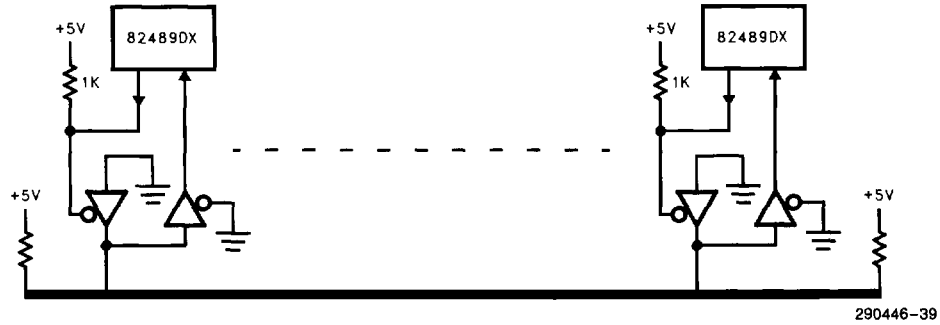
Maximum MBO valid low delay = 50 ns
Maximum MBO H-Z delay = 15 ns
MBI minimum setup time = 8 ns

In the example shown earlier where two 82489DXs are at each end of a 12-inch T-line with no other loads, the pull-high time to 2V is 35 ns (trace 1 in Figure 21g). If the ICLK skew is 2 ns, then this configuration can operate to 62 ns period or 16 MHz.

If the same configuration has additional 20 pF loads at each end, then the pull-high time is 55 ns (trace 2 in Figure 21g). The maximum frequency decreases to 12 MHz (82 ns period).

In the four device model discussed earlier, where the ICC Bus is unbuffered, the pull-high time is 90 ns (Figure 21j). The operating frequency will be less than 8 MHz (117 ns period). If external buffers are used (whereby allowing use of 300Ω pull-up) and assuming the external buffers have delays of 10 ns, the operating frequency is limited by the MBO-to-MBI low time of 72 ns or 14 MHz.

NOTE:
Each application is unique in its configuration and loading on the ICC Bus. The above examples highlighted some of the factors that need to be considered. It is important to do electrical simulation to ascertain if the propose implementation is viable before committing to the printed circuit board.



Note:
Input buffer is optional

Figure 21k. External Driver/Buffer Implementation

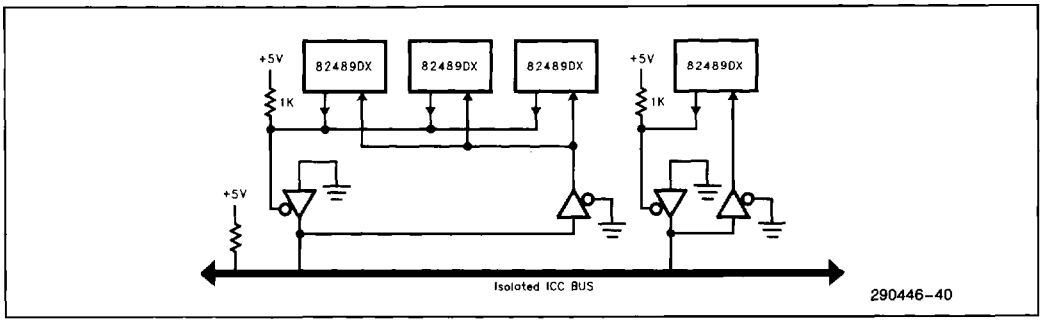


Figure 21l. ICC Bus: Hierarchical Implementation

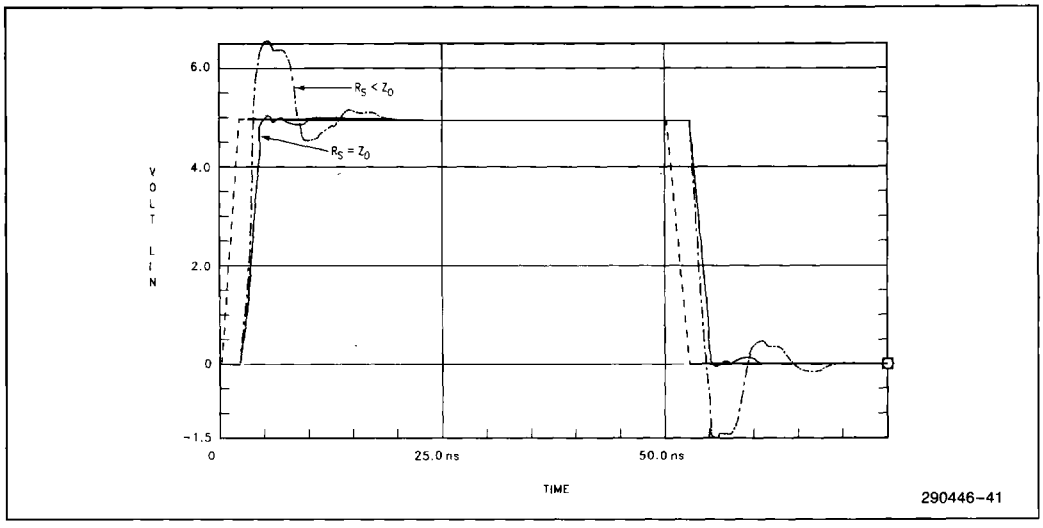


Figure 21m. ICLK Waveform on 12" T-line

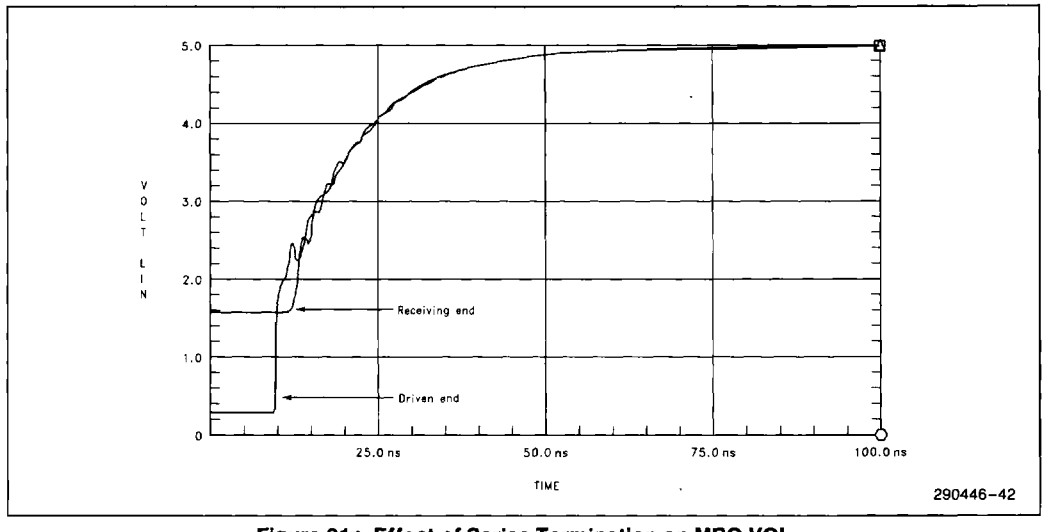


Figure 21n. Effect of Series Termination on MBO VOL

4

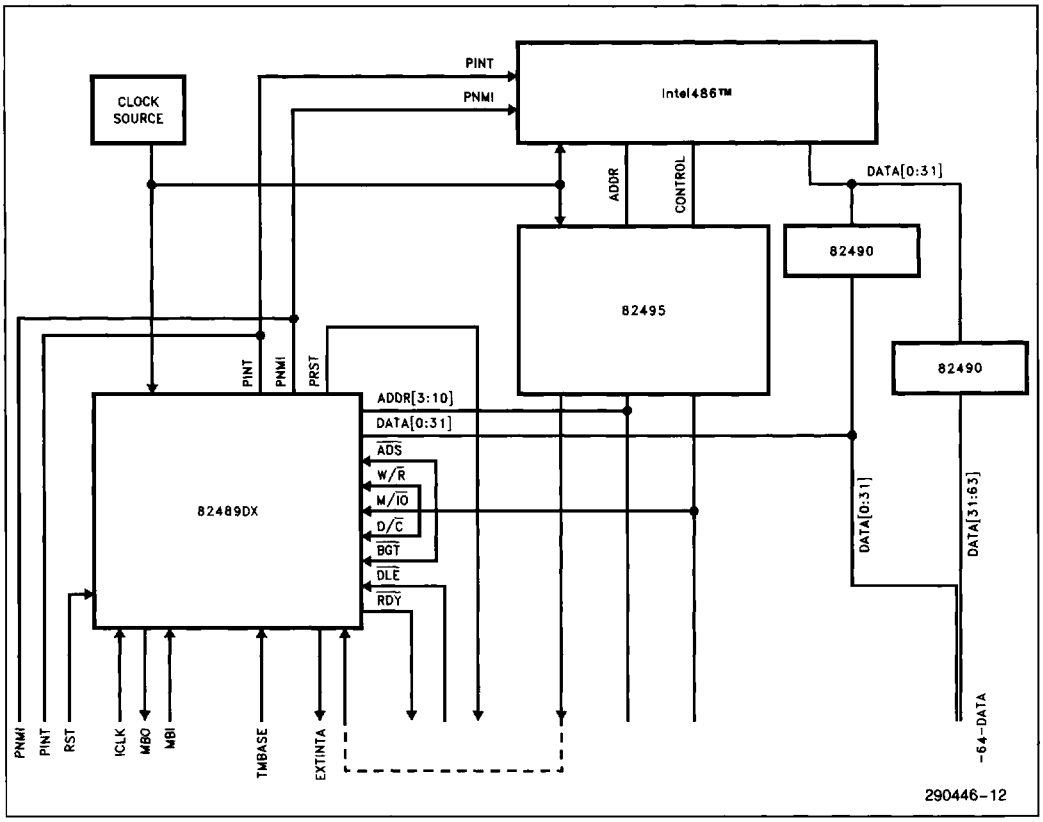


Figure 22. Possible Configuration of the CPU Module

9.1 82489DX Register Access Timing

This section provides descriptions of the four basic cycle timings for the 82489DX, which are:

— Register Write
— Register Read
— Interrupt Acknowledge
— Reset

In addition, Inter-Unit (nibble) bus timings are presented.

Register access occurs in three distinct phases:

1. Control Phase,
2. Address Phase and
3. Data Phase.

They always occur in this order, although in some cases address and data phases can occur in the same clock cycle, as will be seen in the diagrams.

NOTE:
As mentioned previously, the clock signal in all these timing diagrams is assumed to be the processor clock.

TIMING DIAGRAM NOTATION

The 82489DX bus (register access) interface is synchronous. Unless otherwise noted, setup and hold times for inputs, and delay times for outputs are measured with respect to a rising clock edge. Therefore the timing diagrams contain very few construction lines to identify timing parameters and ones that exist are explicitly discussed. High and low level for signals are obvious; dashed lines at the middle of the signal range indicate a tristate output buffer in high impedance mode; hashed or cross-hatched lines indicate a "don't care" state for inputs.

Cycle expansion marks—short curved lines breaking the waveform—appear throughout. These appear in sets or groups which are identified by construction arrows and/or vertical column alignment. Conditions

resulting in cycle expansion are listed at the bottom of each diagram, and the associated set of expansion marks indicates which signals must be stretched for that condition. Signals not so indicated are not affected by that condition, and continue without any cycle stretching. For example, the data phase of a transaction may be delayed, stretching all data related signals, while address related signals can continue to the next cycle.

Sample points for input signals are marked with bold, downward pointing arrows. Since sample timing for address, data and cycle definition signals is dependent on the timing of related control signals, a bar is used on top of the arrow to indicate the signal with the independent timing. Each signal group has exactly one independent signal. In general, independent signals are sampled on every clock, and therefore must meet setup and hold times on every clock edge. Signals having dependent timing, (indicated by the arrow with no bar), are only sampled when the associated independent signal is active, and therefore setup and hold times for dependent signals need only be met at the indicated sample points.

REGISTER WRITE TIMING

For discussion of this bus cycle, refer to Figure 23, 82489DX Timing Diagram 1. This shows the relationship between the three phases of the bus cycle.

The control phase is independently timed by the $\overline{ADS}$ signal. The cycle definition signals [M/$\overline{IO}$, D/$\overline{C}$, are dependently sampled with $\overline{ADS}$ as indicated by the bold sample point arrows labeled "C". The cycle definition signals will be sampled in the first clock when $\overline{ADS}$ is active (low). The control signal should remain stable until $\overline{ADS}$ goes inactive. For any valid 82489DX cycle, the memory bus controller should ensure that the $\overline{ADS}$ pulse for a subsequent bus cycle is NOT presented until after the 82489DX asserts its $\overline{RDY}$ pin (low) as shown.

The address phase is independently timed by the ($\overline{BGT}$) signal, as indicated by the bold sample point arrows labeled "A". This signal is actually used an address latch enable, however, its name is intended to imply that in most cases it can be directly driven by the Intel cache controller signal of the same name. The $\overline{BGT}$ pulse may be delayed until the address bus is available, in which case all address and data phase signals will be stretched. Note that $\overline{DLE}$ must not occur before $\overline{BGT}$. 82489DX does not start the internal cycle until $\overline{BGT}$ is recognized with the appropriate chip select signal, If multiple $\overline{ADS}$ has been issued without $\overline{BGT}$ and a valid chip select, the internal cycle starts with the most recent $\overline{ADS}$ cycle definition preceding the $\overline{BGT}$ with valid chip select.

NOTE C:
Address information, including chip select ($\overline{CS}$), is sampled in the first clock when $\overline{BGT}$ is active (low), and they must remain stable until $\overline{BGT}$ goes inactive, OR until $\overline{RDY}$ is asserted (low), whichever occurs first. $\overline{CS}$ should be stable when $\overline{ADS}$ is active.

In some configurations, $\overline{BGT}$ may not be provided, and can be permanently tied low. In this case, the independent address timing will occur exactly one clock after the $\overline{ADS}$ signal is first sampled low, and the dependent address information (address and chip select) will be assumed stable at this time. 82489DX recognize that independent $\overline{BGT}$ timing is not provided by sampling a low state of $\overline{BGT}$ at the time $\overline{ADS}$ is first sampled low.

The data phase is independently timed by the $\overline{DLE}$ signal, as indicated by the bold sample point arrows labeled "D". In the case of register writes, this signal work logically like a synchronous data latch enable. The $\overline{DLE}$ pulse may be delayed until the data bus is available, in which case data and $\overline{RDY}$ will be stretched.

NOTE D:
Write data are sampled the first clock when $\overline{DLE}$ is active (low), and should remain stable until $\overline{DLE}$ goes inactive, OR until $\overline{RDY}$ is asserted (low), whichever comes first.

In some configurations, $\overline{DLE}$ may not be provided, and can be permanently tied low. In this case, the independent data timing will occur exactly one clock after the $\overline{ADS}$ signal is first sampled low, OR on the same clock as $\overline{BGT}$ is first sampled low, whichever occurs later. The data bus will be assumed stable at this time. 82489DX recognize that independent $\overline{DLE}$ timing is not provided by sampling a low state of $\overline{DLE}$ at the time $\overline{ADS}$ is first sampled low.

Cycle completion is signaled by the $\overline{RDY}$ signal. Its relative positioning on any of the timing diagrams does NOT imply the number of clock cycles required for an access. $\overline{RDY}$ is delayed as needed in order for the 82489DX to complete the cycle. It is then asserted (low) for one clock cycle and then deasserted. Again, the next $\overline{ADS}$ cannot start until after $\overline{RDY}$ has been driven low. $\overline{ADS}$ must return to an inactive high state before the next cycle can be issued. It is highly recommended not to have, $\overline{ADS}$ more than one clock wide.

4

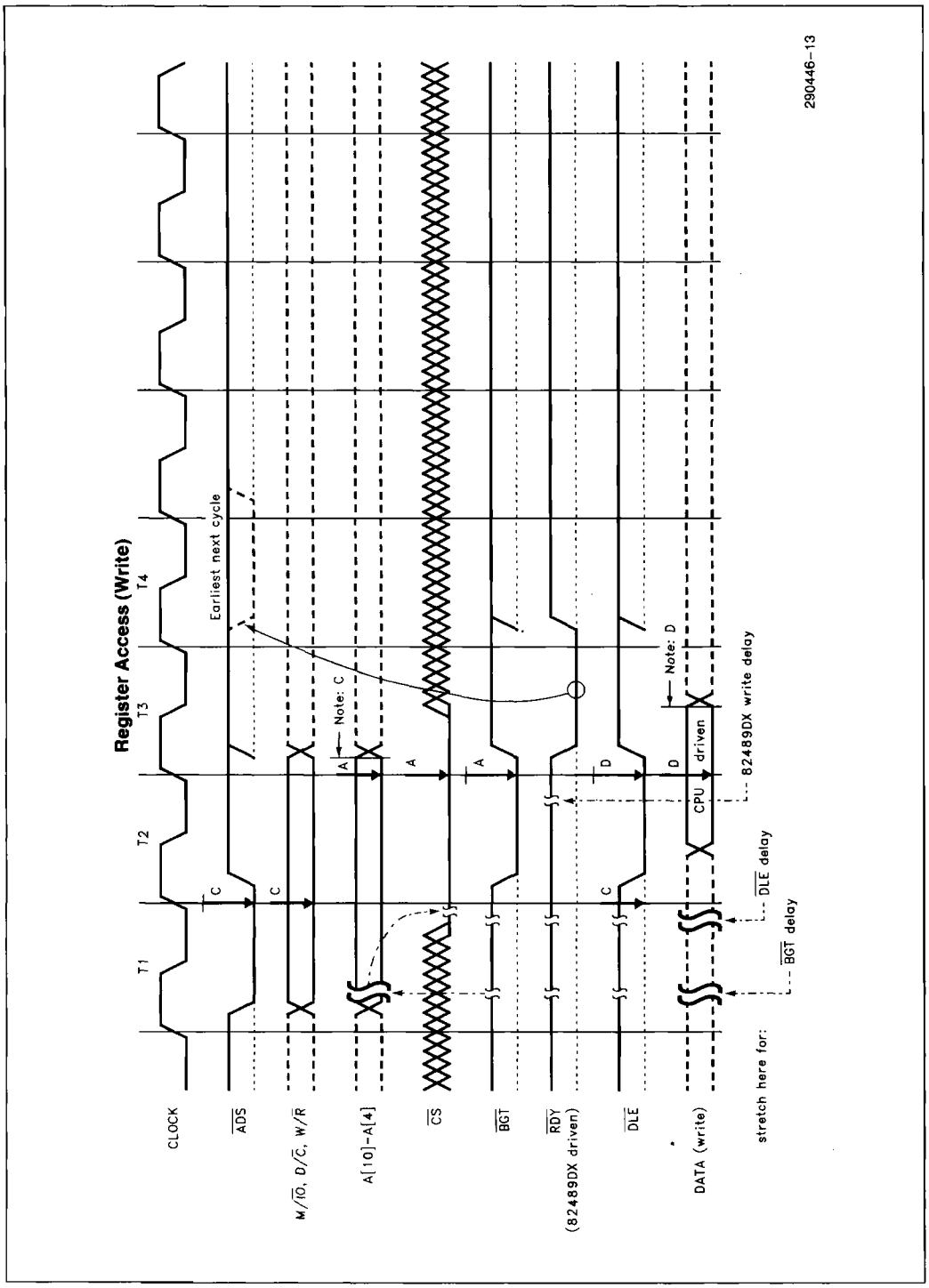


Figure 23. Timing Diagram 1.

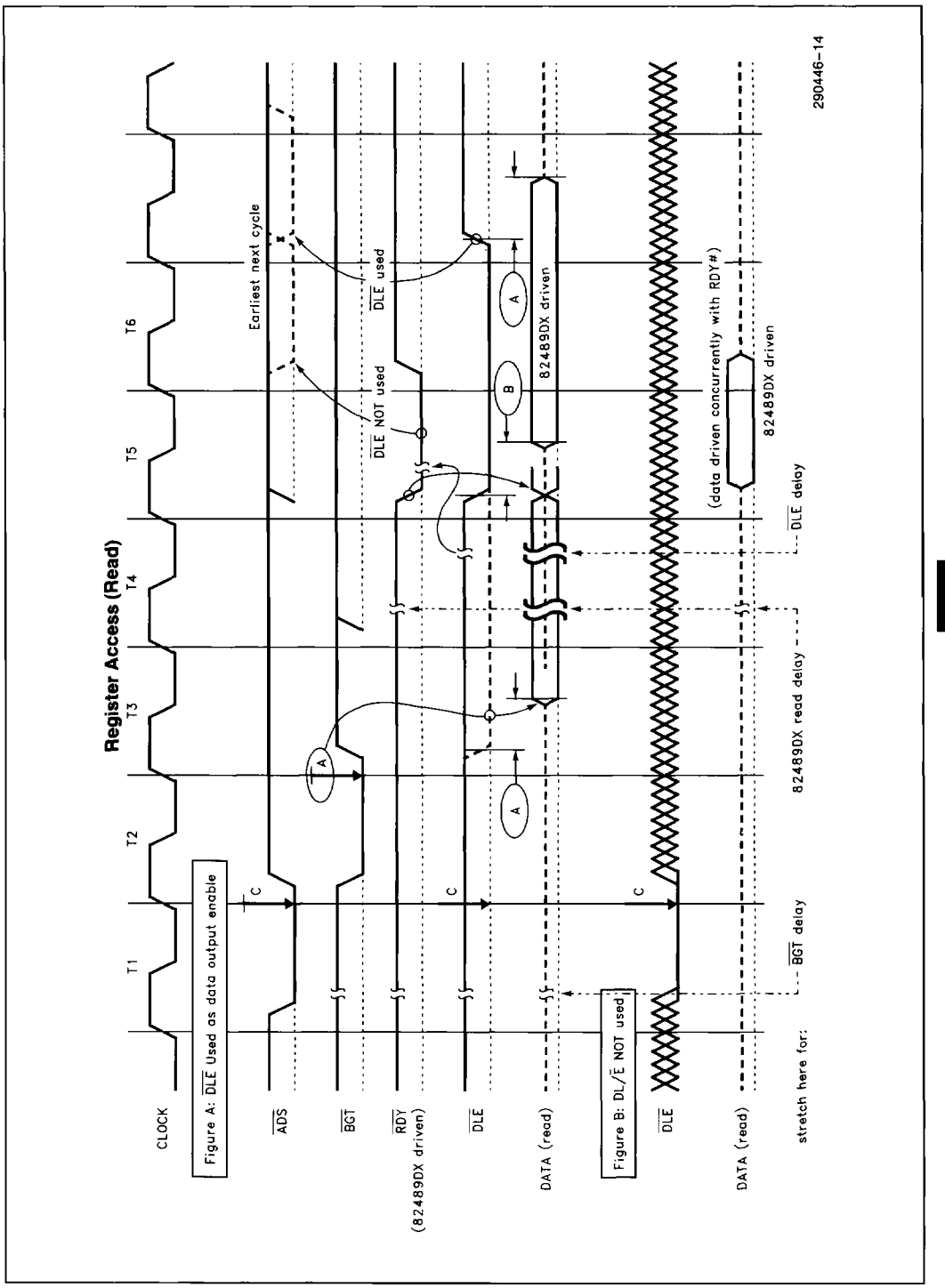


Figure 24. Timing Diagram 2

4

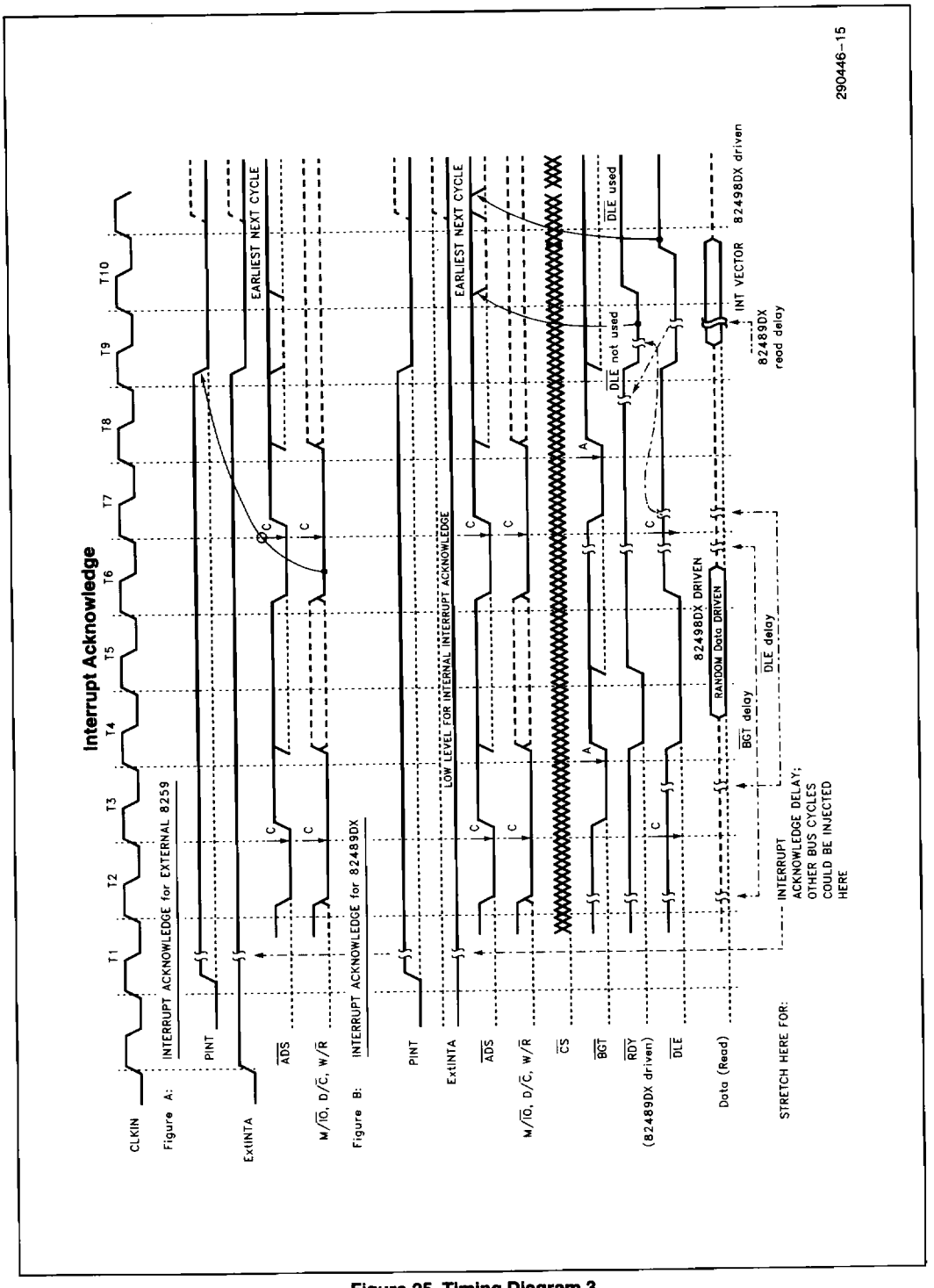


Figure 25. Timing Diagram 3

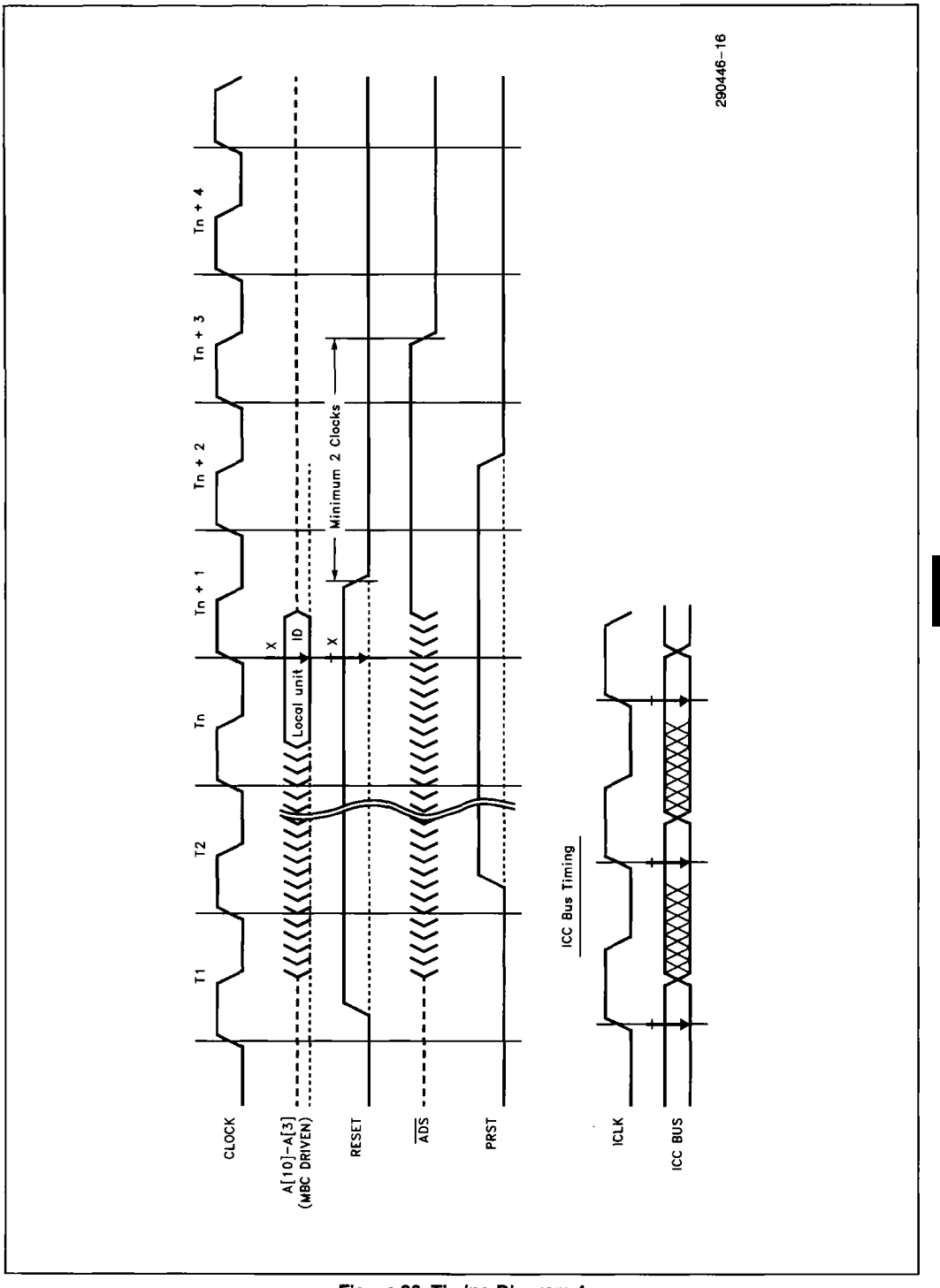


Figure 26. Timing Diagram 4

REGISTER READ TIMING

For discussion of this bus cycle, refer to Figure 24, 82489DX timing Diagram 2. It shows the relationship of the three phases of the bus cycle, however, the dependent control and address signals are not shown here, since they behave exactly as in the case of a register write. See the previous section for the description of control and address phases of the bus cycle.

In the case of a read when $\overline{DLE}$ is used (Figure 24A), it works logically like an asynchronous, output data enable. The 82489DX drives the data bus within time delay "A" after $\overline{DLE}$ is asserted, which must not occur before $\overline{BGT}$. Note that even though the bus is being driven, the data only becomes valid during the clock cycle in which $\overline{RDY}$ is asserted, after that point, valid data is maintained on the bus as long as $\overline{DLE}$ remains asserted, after which the data bus returns to high impedance state within time delay "B" of $\overline{DLE}$ deassertion. If $\overline{DLE}$ is asserted late, 82489DX could complete its internal read cycle and return $\overline{RDY}$ early. In that case, $\overline{RDY}$ active low state will be maintained until $\overline{DLE}$ is asserted.

In the case of a read when $\overline{DLE}$ is NOT used (Figure 24B) the data is driven for exactly one clock cycle, coincident with $\overline{RDY}$ being asserted.

$\overline{DLE}$ is sampled with the control signals to determine whether it is being used. If sampled in the asserted state when $\overline{ADS}$ is active, the $\overline{DLE}$ will be considered not used, and its state during the remainder of the cycle doesn't matter. This is consistent with the notion of permanently tying this signal low when not used, as described in the previous section.

Indication of the end of the bus cycle is dependent on the use of $\overline{DLE}$. When it is not used, (i.e., permanently tied to ground) $\overline{RDY}$ indicates the end of the bus cycle, as it does in the case of write access. When it is used, $\overline{DLE}$ deassertion indicates the end of the cycle, since the 82489DX could be driving the bus well after $\overline{RDY}$ is deasserted. In either case, the 82489DX can accept the next $\overline{ADS}$ pulse anytime after $\overline{RDY}$ has been asserted. However, note that if $\overline{DLE}$ is being used, the next $\overline{ADS}$ should be delayed until $\overline{DLE}$ can be safely sampled inactive. Note these two options for earliest next cycle in the timing diagram.

INTERRUPT ACKNOWLEDGE TIMING

For discussion of this bus cycle, refer to Figure 25 82489DX timing diagram 3, This cycle is the result of the 82489DX posting an interrupt to the processor by asserting PINT. After PINT is asserted, other bus cycles may occur before the interrupt acknowledge cycle.

PINT can be asserted for any external (8259) interrupt as shown in Figure 25A, or for an 82489DX generated interrupt as shown in Figure 25B. ExtINTA indicates whether PINT was asserted in response to an 8259 request or an 82489DX request. This signal is used by external control logic to either allow or preclude the 8259 from responding to the subsequent interrupt acknowledge cycle. It should be noted that ExtINTA pin gets deactivated at third clock after the $\overline{ADS}$ of the second INTA cycle.

When ExtINTA is high, the 82489DX will not respond to the acknowledge cycle other than to deassert PINT signal, and clear the pending external interrupt two full clock cycles after the $\overline{ADS}$ of the second INTA cycle, as shown in Figure 25A. Once PINT is asserted in response to an external interrupt, it can only be deasserted by an INTA cycle. An INTA cycle is recognized by 82489DX as soon as the bus cycle definition is sampled with $\overline{ADS}$ in a low state. $\overline{BGT}$ is not needed in this case.

When ExtINTA is low, the 82489DX will respond to the acknowledge cycle, as shown in Figure 25B. In this case, external logic (e.g., the memory bus controller) is expected to prevent any attached 8259 from seeing the acknowledge cycle. When PINT is asserted in response to an 82489DX internal interrupt, it can be deasserted by an INTA cycle. The PINT signal will be deasserted 5 full clocks after $\overline{BGT}$ of the second INTA cycle.

Note that ExtINTA is stable at all times while PINT is asserted. That means that even if new interrupts arrive between the time an interrupt is posted to the processor, and the acknowledge occurs, the 82489DX will not change its commitment for an external (8259) or internal (82489DX) acknowledge cycle, regardless of priority. This also means that PINT may be raised for a high priority internal interrupt right after responding to the external Interrupt. In any event, PINT will be kept low for a minimum of two clocks before reasserting itself.

The interrupt acknowledge cycle is indicated by the bus cycle definition signals all being low, and looks like two consecutive read cycles, except that there is no explicit address information. The actual content of the address pins during this cycle is processor dependent, and therefore there is no chip select either. Chip select is implied by a combination of the bus cycle definition signals (all low) and $\overline{BGT}$.

Note that there is a "dummy" data phase in the first interrupt acknowledge cycle. This allows parity to be generated on the bus for processors like i860XP. During this cycle, 82489DX drives random data on the bus with the appropriate parity. The interrupt Request Register is "frozen" and the highest priority

pending interrupt vector is returned to the processor in the second acknowledge cycle.

The second acknowledge cycle has a complete data phase with timings identical to those of an ordinary register read. The data returned is the vector of the highest priority internally pending 82489DX interrupt, or the spurious interrupt vector, if there is no interrupt pending higher than the current processor priority.

Note that the timing diagram shows $\overline{DLE}$ being used (sampled high during $\overline{ADS}$). However, just as in a normal read cycle, the option exists not to use $\overline{DLE}$ (i.e., permanently tied to ground).

RESET AND MISCELLANEOUS TIMING

For discussion of this bus cycle, refer to Figure 26 82489DX Timing Diagram 4. It shows the 82489DX reset cycle, the timing of some related signals, and the ICC bus. The RESET input has a setup and hold time to the system clock edge, CLKIN, as do other independently timed signals, The RESET signal will reset the two asynchronous system on the chip, namely the ICC bus unit running synchronously to the ICLK and all the other unit running synchronous to the system clock, CLKIN. RESET must meet the minimum reset time with respect to both clocks and there should be at least one ICLK rising edge during reset. The TAP controller should also be initialized.

During reset, an eight-bit 82489DX Local Unit ID can be optionally initialized. Eight address lines, A10–A3 are sampled on every clock edge while RESET is asserted. The last sample remains in the 82489DX Local Unit ID register after reset. Alternatively, the 82489DX Local Unit ID can be loaded with a register write as part of software initialization, before 82489DX operation is started. In any event, the register must be initialized before the 82489DX can communicate on the ICC bus, including sending/receiving RESET messages. All valid signal to 82489DX should wait at least two full clocks after RESET is deasserted.

The PRST signal (reset output) is asserted both with RESET input, or under software control. Its on-off delay times are relative to the rising clock edge. The duration of PRST under software control is defined by the software itself. Also note that the PNMI pin has the same timing as does PRST when the latter is software controlled.

The ICC bus signals are both input and output on each cycle. Setup, hold and delay times are all measured with respect to the ICC bus Clock ICLK which has no relationship to the Processor clock on which the remainder of the 82489DX runs. This means that the ICC bus is independently sampled on each ICLK edge, as shown. It also implies that largest possible hold time will not exceed the minimum delay time.

After reset, all 82489DX registers are reset to "0" state. The mask bits in the local vector table and the redirection table are reset to "1" state to mask out all interrupts. All reserved bits are all wired to "0" state permanently on chip.

10.0 BOUNDARY SCAN DESCRIPTION

The 82489DX is equipped with the JTAG boundary scan standard. This feature allows the user to test the interconnections between 82489DX and the external hardware once they have been assembled onto a printed circuit board or other substrate. In addition to the JTAG mandatory instruction set, 82489DX also provides the INTEST instruction which allows static testing of the on-chip logic.

The detailed information related to the IEEE Std 1149.1-1990 (the JTAG standard) can be obtained from the reference document *IEEE Standard Test Access Port and Boundary Scan Architecture (IEEE Std 1149.1–1990)*.

4

10.1 Boundary Scan Architecture

The boundary scan logic contains the following elements:

- **Five Test Access Ports (TAP):** They are labeled as $\overline{trst}$, tck, tdi, tdo and tms. All ports are input pins except tdo, which is a tri-state output pin.
- **A TAP Controller:** The logic is used to control the boundary scan activity.
- **82489DX Device ID Register:** This is a 32-bit read-only register. The DID can be shifted out in ascending order to the tdo pin.
- **JTAG Instruction Register (IR):** This is a 4-bit register which accepts instruction code shifted in from the tdi pin. The opcode stored in the IR register is used to control operation.
- **Boundary Scan Register:** This is a 137 stages scan path which connects almost all 82489DX signal pins for boundary scan purposes.
- **Bypass Register:** This register simply allows the data which goes into tdi pin to be shifted out directly from tdo.

The following block diagram illustrates the implementation of the JTAG architecture in the 82489DX design.

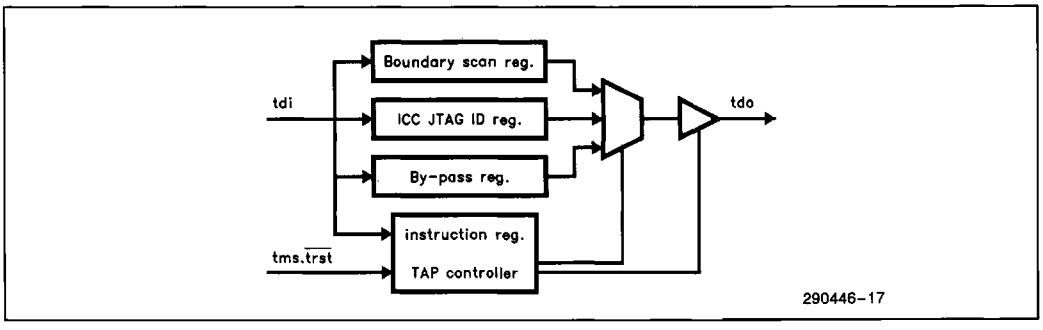


Figure 27. Block Diagram of the JTAG Architecture

Test Access Ports

$\overline{trst}$ TAP controller master reset pin. When $\overline{trst}$ is low, the TAP controller's state machine will be reset to "test-logic-reset" state asynchronously. This pin is tied to a weak internal pull-up for keeping to be a logical 1 when not driven.

tck This is the test logic clock. The test logic will change state on the rising edge of the tck.

tdi Test data input. Data is shifted into the tdi pin on the rising edge of tck. This pin is tied to a weak internal pull-up for keeping it to be a logical 1 when not driven.

tms Test mode select. This pin is used to select the state of the TAP controller. This pin is synchronous to the rising edge of the tck. This pin is tied to a weak internal pull-up for keeping it to be a logical 1 when not driven.

tdo Test data output. This is a tri-state pin which allows the data to be shifted out.

TAP CONTROLLER

The TAP controller in 82489DX is implemented to conform the IEEE1149.1 standard. The TAP controller is a single phase clock, synchronous finite state machine. It controls the sequence of the operation of the test logic.

The value of the test mode state (tms) pin at a rising edge of tck controls the sequence of the state changes. The state diagram for the TAP controller is shown in Figure 28. Test designers must consider the operation of the state machine in order to have the correct sequence of value to drive on tms.

The behavior of the TAP controller and other test logic in each of the controller states is briefly described as follows.

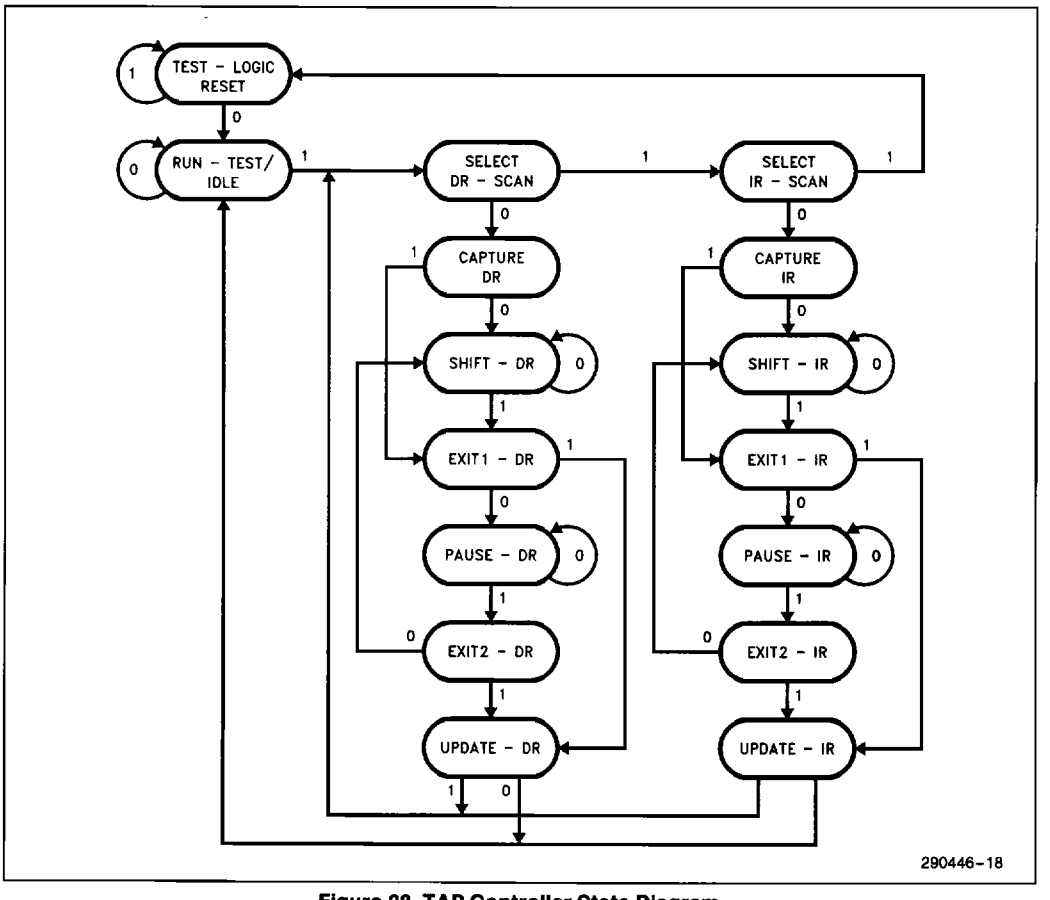


Figure 28. TAP Controller State Diagram

TEST-LOGIC-RESET

The test logic is disabled so that normal operation of the on-chip system logic (i.e., in response to stimuli received through the system pins only) can continue unhindered. This is achieved by initializing the instruction register to contain the IDCODE instruction. No matter what the original state of the controller, it will enter Test-Logic-Reset when tms is held high for at least five rising edges of tck. The controller remains in this state while tms is high.

If the controller should leave the Test-Logic-Reset controller state as a result of an erroneous low signal on the tms line at the time of the rising edge on tck (for example, a glitch due to external interference), it will return to the Test-Logic-Reset state following three rising edges of tck with the tms line at the intended high logic level. The operation of the test logic is such that no disturbance is caused to on-chip system logic operation as the results of such an error. On leaving the Test-Logic-Reset controller state, the controller moves into the Run-Test/Idle controller state where no action will occur because the current instruction has been set to select operation of the device identification register. The test logic is also inactive in the Select-DR-Scan and Select-IR-Scan controller states.

Note that the TAP controller will also be forced to the Test-Logic-Reset controller state asynchronously by applying a low logic level at $\overline{trst}$.

RUN-TEST/IDLE

A controller state between scan operations. Once entered, the controller will remain in the Run-Test/Idle state as long as tms is held low. When tms is high and a rising edge is applied at tck, the controller moves to the Select-DR-Scan state.

In the Run-Test/Idle controller state, activity in selected test logic occurs only when certain instructions are present such as RUNBIST. Since 82489DX does not have RUNBIST instruction, this state is acting like an idle state.

The instruction does not change while the TAP controller is in this state.

SELECT-DR-SCAN

This is a temporary controller state in which all test data register (82489DX has one test data register which is the boundary scan shift registers path) selected by the current instruction retain their previous state.

If tms is held low and a rising edge is applied to tck when the controller is in this state, then the controller moves into the Capture-DR state and a scan sequence for the selected test data register is initiated. If tms is held high and a rising edge is applied to tck, the controller moves on to the Select-IR-Scan state.

The instruction does not change while the TAP controller is in this state.

SELECT-IR-SCAN

This is a temporary controller state in which all test data registers selected by the current instructing retain their previous state.

If tms is held low and a rising edge is applied to tck when the controller is in this state, then the controller moves into the Capture-IR state and a scan sequence for the instruction register is initiated. If tms is held high and a rising edge is applied to tck, the controller returns to the Test-Logic-Reset state. **The instruction does not change while the TAP controller is in this state**

CAPTURE-DR

In this controller state data may be parallel-loaded into test data registers selected by the current instruction on the rising edge of tck. If a test data register selected by the current instruction does not have a parallel input, or if capturing is not required for the selected test, then the register retains its previous state unchanged. ***The instruction does not change while the TAP controller is in this state.***

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Exit 1-DR state if tms is held at 1 or the Shift-DR state if tms is held at 0.

SHIFT-DR

In this controller state, the test data register connected between tdi and tdo as a result of the current instruction shifts data one stage towards its serial output on each rising edge of tck. Test data registers that are selected by the current instruction, but are not placed in the serial path, retain their previous state unchanged. ***The instruction does not change while the TAP controller is in this state.***

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Exit 1-DR state if tms is held at 1 or remains in the Shift-DR state if tms is held at 0.

EXIT 1-DR

This is a temporary controller state, if tms is held high, a rising edge applied to tck while in this state causes the controller to enter the Update-DR state, which terminates the scanning process. If tms is held low and a rising edge is applied to tck, the controller enters the Pause-DR state. **All test date registers selected by the current instructions retain their previous state unchanged.**

The instruction does not change while the TAP controller is in this state.

PAUSE-DR

This controller state allows shifting of the test data register in the serial path between tdi and tdo to be temporarily halted. All test data registers selected by the current instruction retain their previous state unchanged.

The controller remains in this state while tms is low. When tms goes high and a rising edge is applied to tck, the controller moves on to the Exit 2-DR state. The instruction does not change while the TAP controller is in this state.

EXIT 2-DR

This is a temporary controller state. If tms is held high and a rising edge is applied to tck while in this state, the scanning process terminates and the TAP controller enters the Update-DR controller state. If tms is held low and a rising edge is applied to tck, the controller enters the Shift-DR state.

All test data registers selected by the current instruction retain their previous state unchanged. The instruction does not change while the TAP controller is in this state.

UPDATE-DR

Some test date registers may be provided with a latched parallel output to prevent changes at the parallel output while data is shifted in the associated shift-register path in response to certain instructions (e.g., EXTENT, INTEST, and RUNBIST). Data is latched onto the parallel output of these test data registers from the shift-register path on the falling edge of tck in the Update-DR controller state. The data held at the latched parallel output should not change other than in this controller state unless operation during the execution of a self test is required (e.g., during the Run-Test/Idle controller state in response to a design-specific public instruction).

All shift-register stages in test data registers selected by the current instruction retain their previous state unchanged. **The instruction does not change while the TAP controller is in this state.**

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Select-DR-Scan state if tms is held at 1 or the Run-Test/Idle state if tms is held at 0.

CAPTURE-IR

In this controller state the shift-register contained in the instruction register loads a pattern of fixed logic values on the rising edge of tck. In addition, design-specific data may be loaded into shift-register stages that are not required to be set to fixed values.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

When the TAP controller is in this state and rising edge is applied to tck, the controller enters either the Exit 1-IR state tms is held at 1 or the Shift-IR state if tms is held at 0.

SHIFT-IR

In this controller state the shift-register contained in the instruction register is connected between tdi and tdo and shifts data one stage towards its serial output on each rising edge of tck.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state. When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Exit1-IR state if tms is held at 1 or remains in Shift-IR state if tms is held at 0.

EXIT 1-IR

This is a temporary controller state. If tms is held high, a rising edge applied to tck while in this state causes the controller to enter the Update-IR state, which terminates the scanning process. If tms is held low and a rising edge is applied to tck, the controller enters the Pause-IR state.

Test data registers selected by the current instruction retain their previous state. The instruction does not changes while the TAP controller is in this state and the instruction register retains its state.

PAUSE-IR

This controller state allows shifting of the instruction register to be halted temporarily.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

The controller remains in this state while tms is low. When tms goes high and a rising edge is applied to tck, the controller moves on to the Exit 2-IR state.

EXIT 2-IR

This is a temporary controller state. If tms is held high and a rising edge is applied to tck while in this state, termination of the scanning process results, and the TAP controller enters the Update-IR controller state. If tms is held low and a rising edge is applied to tck, the controller enters the Shift-IR state.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

UPDATE-IR

The instruction shifted into the instruction register is latched onto the parallel output from the shift-register path on the falling edge of tck in this controller state. Once the new instruction has been latched, it

becomes the current instruction. **Test data registers selected by the current instruction retain their previous state.**

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters the Select-DR-Scan states if tms is held at 1 or the Run-Test/Idle state if tms is held at 0.

INSTRUCTION REGISTER

The function of the instruction register is to select the operating mode of the test logic. For instance, read the ID register, or capture the 82489DX output signals. 82489DX has implemented 4 instructions.

Instruction	Mandatory/Optional	Opcode
bypass	m	1 1 1 1
extest	m	0 0 0 0
sample/preload	m	0 0 0 1
idcode	m	0 0 1 0
reserved	o	1 0 0 1

Bypass Instruction

The bypass instruction selects the bypass register to be connected to tdi and tdo, effectively bypassing the test logic on the 82489DX boundary scan path and reducing the shift length to be on one bit. Note that an open circuit fault in the board level test data path will cause the bypass register to be selected following an instruction scan cycle due to the internal pull-up on the tdi pin. This has been done to prevent any unwanted interference with the proper operation of the system logic.

Extest Instruction

The extest instruction allows testing of circuitry external to the component package, typically board interconnects. It does so by driving the values loaded into the 82489DX's boundary scan register out on the output pins corresponding to each boundary scan cell and capturing the values on 82489DX's input pins to be loaded into their corresponding boundary scan register locations. I/O pins are selected as input or output depending on the value located into the output control cell. Values shifted into input latch in the boundary scan register are never used by the internal logic of the 82489DX.

NOTE:
82489DX must be reset after extest instruction has been executed.

Sample/Preload Instruction

The sample/preload instruction has two functions that it can perform. When the TAP controller is in the CAPTURE-DR state, the sample/preload instruction allows a snap-shot of the normal operation of the 82489DX without interfering with that normal operation. The instruction causes boundary scan register cells associated with outputs to sample the value being driven into the 82489DX. On both outputs and inputs the sampling occurs on the rising edge of tck. When preloads data into the 82489DX pins to be driven to the board by executing the extest instruction. Data is preloaded to the pins from the boundary scan register on the falling edge of tck.

dcode Instruction

The idcode instruction selects the device identification register to be connected to tdi and tdo, allowing the device ID code to be shifted out of the device on tdo. Note that the bit stream shifted into tdi will appear on tdo after all 32 bits of the DID has been shifted out.

DEVICE IDENTIFICATION REGISTER (DID)

The device identification is a 32 bits number which can be read by the external hardware by using the idcode instruction. The 82489DX device ID is assigned to 1489A013 (hex). This is subject to change. The upper 4 bits of DID may be changed for different version. The 16-bit number (bit 27–bit 12) 489A (hex) is the part ID. The lower 12 bits are the manufacturer ID for Intel which must be 013 (hex).

BOUNDARY SCAN REGISTER

82489DX has only one test data register, i.e., the boundary scan register. The boundary scan register is a single shift register path containing the boundary scan cells that are connected to all signal input and output pins of the 82489DX. There are three generic type of boundary scan cells—input, output, and bi-directional. For each input only cell, one stage of shift register is added to the boundary scan path.

All output pins will become tri-stateable when boundary scan is activated, regardless whether they are tri-stateable or not in the normal operation. To explain further, the user will enable/disable an output driver with a specific tri-state control cell in the scan path. The user must shift in a proper control signal for these tri-state control cells in the scan path.

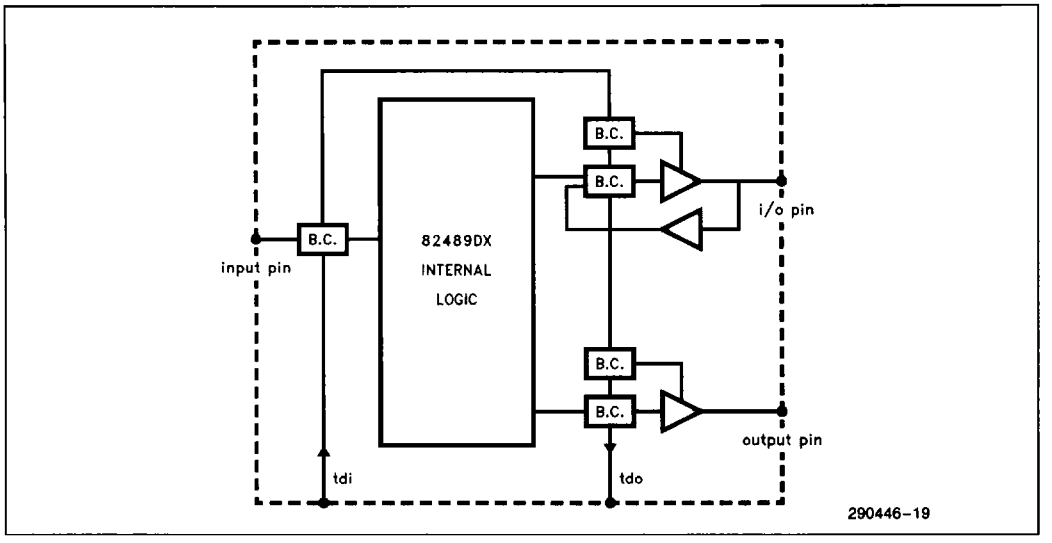


Figure 29. Logical Structure of Boundary Scan Register

4

BOUNDARY SCAN CELL NAMES IN ORDER FROM tdi TO tdo

The following table is a list of the boundary scan cell names in the order from tdi to tdo. The type information indicates the purpose of the cells.

I = input only cell

B = bi-directional cell

T = tri-state output cell

C = tri-state control. Note that the signal name enclosed within the parenthesis is controlled by this cell.

Cell Number	Type	Name	Pin #
1	I	CLKIN	57
2	I	TMBASE	59
3	I	ICLK	60
4	I	$\overline{DC}$	61
5	I	$\overline{WR}$	62
6	I	M/$\overline{IO}$	63
7	I	$\overline{ADS}$	64
8	I	RESET	65
9	I	$\overline{BGT}$	66
10	I	reserved	70
11	I	reserved	71
12	I	reserved	72
13	I	$\overline{DLE}$	73
14	I	$\overline{CS}$	74
15	I	reserved	75
16	I	MBI3	76
17	I	MBI2	77
18	I	MBI1	78
19	I	MBI0	79
20	I	LINTIN1	80
21	I	LINTIN0	81
22	T	reserved	
23	C	(reserved)	
24	I	INTIN15	82
25	I	INTIN14	83
26	T	reserved	
27	C	(reserved)	
28	I	INTIN13	84
29	I	INTIN12	85
30	T	reserved	
31	C	(reserved)	
32	I	INTIN11	86
33	I	INTIN10	87
34	T	reserved	
35	C	(reserved)	
36	I	INTIN9	88
37	I	INTIN8	89
38	T	reserved	
39	C	(reserved)	
40	I	INTIN7	90
41	I	INTIN6	91
42	T	reserved	
43	C	(reserved)	
44	I	INTIN5	92
45	I	INTIN4	93
46	T	reserved	
47	C	(reserved)	
48	I	INTIN3	94
49	I	INTIN2	95
50	T	reserved	
51	C	(reserved)	
52	I	INTIN1	96
53	I	INTIN0	97
54	I	reserved	
55	B	DP3	101
56	C	(DP[3:0])	
57	B	DP2	102
58	B	DP1	103
59	B	DP0	104
60	T	reserved	
61	C	(reserved)	
62	B	D31	105
63	C	(D[31:0])	
64	B	D30	107

Cell Number	Type	Name	Pin #
65	B	D29	109
66	B	D28	110
67	T	reserved	
66	C	(reserved)	
69	B	D27	111
70	B	D26	112
71	T	reserved	
72	C	(reserved)	
73	B	D25	114
74	B	D24	115
75	B	D23	116
76	T	reserved	
77	C	(reserved)	
78	B	D22	118
79	B	D21	119
80	B	D20	121
81	B	D19	122
82	B	D18	123
83	B	D17	124
84	B	D16	125
85	B	D15	128
86	B	D14	129
87	B	D13	130
88	B	D12	131
89	B	D11	2
90	T	reserved	
91	C	(reserved)	
92	B	D10	3
93	B	D9	4
94	T	reserved	
95	C	(reserved)	
96	B	D8	7
97	B	D7	8
98	B	D6	9
99	B	D5	11
100	B	D4	12
101	B	D3	13

Cell Number	Type	Name	Pin #
102	B	D2	14
103	B	D1	16
104	B	D0	18
105	I	reserved	19
106	B	reserved	20
107	C	(cell106, A[10:3])	
108	B	A10	21
109	B	A9	22
110	B	A8	24
111	B	A7	26
112	B	A6	27
113	B	A5	28
114	B	A4	29
115	B	A3	31
116	T	reserved	34
117	C	reserved	
118	T	PINT	35
119	C	(PINT)	
120	T	PNMI	37
121	C	(PNMI)	
122	T	PRST	38
123	C	(PRST)	
124	T	ExtINTA	41
125	C	(reserved)	
126	T	reserved	42
127	C	(reserved)	
128	T	$\overline{\text{RDY}}$	43
129	C	($\overline{\text{RDY}}$)	
130	T	MBO3	45
131	C	(MBO3)	
132	T	MBO2	48
133	C	(MBO2)	
134	T	MBO1	49
135	C	(MBO1)	
136	T	MBO0	51
137	C	(MBO0)	

BYPASS REGISTER

The bypass register is simply a 1-bit shift register which connects between the tdi and tdo. When selected by using the bypass instruction, the data shifted into tdi will be shifted out from tdo one tck clock later.

JTAG TAP Controller Initialization

The TAP controller must be reset to test-logic-reset state when 82489DX is first powered up. There are two ways to reset the TAP controller:

1. Assert $\overline{\text{trst}}$ to be 0, it will reset the TAP controller asynchronously.
2. Assert tms to be 1, and clock the TAP controller at least five times, the TAP controller will be reset after the fifth rising edge of the tck.

After reset, the idcode instruction is loaded into the IR automatically.

Note that the tms and $\overline{\text{trst}}$ pins both have an internal weak pull-up device to keep them to be logic 1 level. Therefore the user can simply apply 5 clocks at the tck input to reset the TAP controller. If the TAP controller is not reset properly, 82489DX may not function because the boundary scan logic might be active which will impact the signals flow in and out to the chip.

11.0 ELECTRICAL CHARACTERISTICS

NOTICE: This data sheet contains information on products in the sampling and initial production phases of development. The specifications are subject to change without notice. Verify with your local Intel Sales office that you have the latest data sheet before finalizing a design.

11.1 D.C. Specifications

ABSOLUTE MAXIMUM RATINGS

Case Temperature Under Bias . . . −65°C to +110°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin
with Respect to Ground . . . . . −0.5 to V_{CC} + 0.5

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

V_{CC} = 5V ±5%; T_C = 0°C to +85°C

Symbol	Parameter	Min (ns)	Max	Units	Notes
V_{IL}	Input LOW Voltage (TTL)	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage (TTL)	2.0	V_{CC} + 0.3	V	
V_{OL}	Output LOW Voltage (TTL)		+0.45	V	(Note 1)
V_{QH}	Output HIGH Voltage (TTL)	2.4		V	(Note 2)
I_{CC}	33 MHz Power Supply Current		200	mA	
I_{LI}	Input Leakage Current		15	µA	
I_{LL}	Input Leakage Current		−600	µA	(Note 5)
I_{LH}	Output Leakage Current		600	µA	(Note 4)
I_{LO}	Output Leakage Current		15	µA	(Note 3)
C_{IN}	Input Capacitance		3	pF	
C_O	I/O or Output Capacitance		6	pF	
C_{CLKIN}	Clock Capacitance		3	pF	
I_{MLO}	ICC Bus Output Low Current		4	mA	(Note 6)
C_{MC}	ICC Bus Total Capacitance		100	pF	
V_{MH}	ICC Bus Input High (TTL)	2.0	V_{CC} + 0.3	V	
V_{ML}	ICC Bus Input Low (TTL)	−0.3	+0.8	V	

NOTES:
1. This parameter is measured with current load of 4 mA.
2. This parameter is measured with current load of 1.0 mA.
3. This parameter is for output without pulldown.
4. This parameter is for tri-state output with pulldown and V_{OH} = 3.0V.
5. This parameter is for input with pullup at V_{IL} = 0V.
6. ICC bus output low current is measured at 0.6V.

11.2 A.C. Specifications

A.C. Parameters Referencing 33 MHz System Clock

V_{CC} = 5V ±5%; T_C = 0°C to +85°C

Symbol	Parameter	Ref. Fig.	Load (pF)	Min (ns)	Max (ns)	Notes
tc	CLKIN Period	30		30	100	(Note 1)
t1	CLKIN High Time	30		5		
t2	CLKIN Low Time	30		5		
t3	CLKIN Rise Time	30			3	(Note 2)
t4	CLKIN Fall Time	30			3	(Note 2)
t5	$\overline{ADS}$, $\overline{BGT}$, $\overline{DLE}$, M/$\overline{IO}$, D/$\overline{C}$, W/$\overline{R}$, $\overline{CS}$ Setup Time	31		8		
t6	D31–D0, DP3–DP0, A9–A3 Setup Time	31		8		
t8	$\overline{ADS}$, $\overline{BGT}$, $\overline{DLE}$, M/$\overline{IO}$, D/$\overline{C}$, W/$\overline{R}$, $\overline{CS}$ Hold Time	31		5		
t10	D31–D0, DP3–DP0, A9–A3 Hold Time	31		5		
t11	D31–D0, DP3–DP0, Valid Delay	30	50		18	
t12	D31–D0, DP3–DP0, Low-Z Delay When $\overline{DLE}$ is Not Used	32	50	3		(Note 7)
t13	D31–D0, DP3–DP0, High-Z Delay When $\overline{DLE}$ is Not Used	32	50		14	(Note 7)
t14	D31–D0, DP3–DP0 Enable Delay When $\overline{DLE}$ is Used	33	50	3	42	
t15	D31–D0, DP3–DP0 Disable Delay When $\overline{DLE}$ is Used	33	50	3	14	
t20	$\overline{RDY}$ Valid Delay	30	50	3	18	
t21	PRST, PNMI, PINT Valid Delay	30	50	3	34	
t22	RESET Setup Time	31		8		(Note 5)
t23	RESET Hold Time	31		5		(Note 5)
	RESET Cycle Time			5 tc		(Note 3)
				1 tic		(Note 3)
t24	INTIN[15:0], LINTIN[1:0] Low Time			10		(Note 6)

All parameters are given in nanoseconds.
TTL Level timing is measured at 1.5V for both "0" and "1" levels.

NOTES:
1. ICC bus clock ICLK period must be at least 5 ns longer than system clock CLKIN for proper synchronization of the internal asynchronous signals.
2. System clock CLKIN measured from 0.8V–2.0V.
3. Minimum Reset cycle is the greater of the two cycle times.
4. Minimum pulse width must be met for valid level to be attained on the DATA or ADDRESS output.
5. Set up and hold time is required for RESET to start at the next rising edge of the clock.
6. INTIN and LINTIN low time is measured from 1.5V of the falling edge to 1.5V of rising edge.
7. Not 100% tested. Guaranteed by design characterization.

Time Base A.C. Parameters

V_{CC} = 5V ±5%; T_C = 0°C to +85°C

Symbol	Parameter	Ref. Fig.	Min (ns)	Max (ns)	Note
tmc	TMBASE Period	35	40	10000	
t30	TMBASE High Time	35	10		
t31	TMBASE Low Time	35	10		
t32	TMBASE Rise Time	35		8	
t33	TMBASE Fall Time	35		8	

TAP Controller A.C. Parameters V_{CC} = 5V ±5%; T_C = 0°C to +85°C

Symbol	Parameter	Ref. Fig.	Min (ns)	Max (ns)	Note
ttc	TCK Period	35	40	1000	
t50	TCK High Time	35	10		
t51	TCK Low Time	35	10		
t52	TCK Rise Time	35		8	
t53	TCK Fall Time	35		8	
t54	TDI, TMS, $\overline{TRST}$ Setup Time	34	10		
t55	TDI, TMS, $\overline{TRST}$ Hold Time	34	5		
t56	TDO VALID Delay	34	5	24	(Note 1)
t57	Output Delay in EXTest in EXTEST Mode	34	5	27	(Note 1)
t58	$\overline{TRST}$ Minimum Low Time		10		(Note 2)

All parameters are given in nanoseconds.
TTL level timing is measured at 1.5V for both "0" and "1" levels.

NOTES:
1. These parameters are specified for 50 pF load.
2. This parameter is measured at 1.5V between the rising and falling edges.

4

A.C. Parameters for ICC Bus

V_{CC} = 5V ±5%; T_C = 0°C to +85°C

Symbol	Parameter	Ref. Fig.	Min (ns)	Max (ns)	Notes
tic	ICLK Period	35	60		(Note 1)
t40	ICLK High Time	35	20		
t41	ICLK Low Time	35	20		
t42	ICLK Rise Time	35		10	
t43	ICLK Fall TIme	35		10	
t44	MBI3–MB10 Setup Time	36	8		
t45	MBI3–MB10 Hold Time	36	5		
t46	MB03–MBO0 VALID Low Delay	36		50	(Note 2)
t47	MB03–MBO0 VALID High-Z Delay	36	5	15	(Note 3)
t48	MB03–MBO0 VALID Low-Z Delay	36	12	25	(Note 3)

All parameters are given in nanoseconds.
TTL level timing is measured at 1.5V for both "0" and "1" levels.

NOTES:
1. MBI3–0 and MBO3–0 timing is tested at 150 ns cycle time.
2. This parameter is specified for 50 pF load.
3. Not 100% tested. Guaranteed by design characterization.

12.0 REGISTER SUMMARY

82489DX registers can be located at any 1 Kbyte boundary in either memory or I/O space for as far as the 82489DX architecture itself is concerned. From a platform standard point of view, it is recommended to locate all 82489DX Local Units in memory space at address 0xFEE0—0000. It is further recommended that all 82489DX I/O Units also be located in memory space; I/O Unit 1 at address 0xFEC0—0000, I/O Unit 2 (if present) at address 0xFEC0—1000, and so on. Chip select for the 82489DX should be based on a full decode of address pins A31–A10.

All directly accessible 82489DX registers are 32 bits wide and are aligned at 128-bit boundaries. The register being accessed is determined by bits 4 through 9 of the address. This is listed in the tables below.

Addresses not listed are reserved by the architecture. The tables also show whether the register is readable and/or writable by software, and what the side effects are of software accessing the register.

After reset, all registers are initialized to all zeroes with the following exceptions, The Local Unit ID field is initialized with data present on the 8 LSB address pins. The Mask bit is initialized to 1 ("masked" state) in all entries in both the local vector table and the redirection table.

For the I/O Unit, only the I/O register select and I/O window registers are directly accessible in the address space. The other I/O unit registers are accessed indirectly through the select and window register.

I/O Unit Registers

Register	Address (9:4)	SW	Side Effects
I/O Register Select	00 0000	W	
I/O Window Register	00 0001		

Register	I/O Reg Select (7:0)	SW	Side Effects
I/O Unit ID Register	0000 0000	rw	
Version Register	0000 0001	r	
Redirection Table [0] (31:0)	0001 0000	rw	
Redirection Table [0] (63:32)	0001 0001	rw	
Redirection Table [1] [31:0]	0001 0010	rw	
Redirection Table [1] [63:32]	0001 0011	rw	
Redirection Table [2] (31:0)	0001 0100	rw	
Redirection Table [2] (63:32)	0001 0101	rw	
Redirection Table [3] [31:0]	0001 0110	rw	
Redirection Table [3] [63:32]	0001 0111	rw	
Redirection Table [4] (31:0)	0001 1000	rw	
Redirection Table [4] (63:32)	0001 1001	rw	
Redirection Table [5] [31:0]	0001 1010	rw	
Redirection Table [5] [63:32]	0001 1011	rw	
Redirection Table [6] (31:0)	0001 1100	rw	
Redirection Table [6] (63:32)	0001 1101	rw	
Redirection Table [7] [31:0]	0001 1110	rw	
Redirection Table [7] [63:32]	0001 1111	rw	
Redirection Table [8] (31:0)	0010 0000	rw	
Redirection Table [8] (63:32)	0010 0001	rw	
Redirection Table [9] [31:00]	0010 0010	rw	
Redirection Table [9] [63:32]	0010 0011	rw	
Redirection Table [10] (31:00)	0010 0100	rw	
Redirection Table [10] (63:32)	0010 0101	rw	
Redirection Table [11] [31:0]	0010 0110	rw	
Redirection Table [11] [63:32]	0010 0111	rw	
Redirection Table [12] (31:0)	0010 1000	rw	
Redirection Table [12] (63:32)	0010 1001	rw	
Redirection Table [13] [31:0]	0010 1010	rw	
Redirection Table [13] [63:32]	0010 1011	rw	
Redirection Table [14] (31:0)	0010 1100	rw	
Redirection Table [14] (63:32)	0010 1101	rw	
Redirection Table [15] [31:0]	0010 1110	rw	
Redirection Table [15] [63:32]	0010 1111	rw	

LOCAL UNIT REGISTERS

Registers	Address (9:4)	SW	Side Effects
Local Unit ID Register	00 0010	rw	
Version Register	00 0011	r	
Reserved	00 0100		
Reserved	00 0101		
Reserved	00 0110		
Reserved	00 0111		
Task Priority Register	00 1000	rw	mask intr dispense
Reserved	00 1001		
Reserved	00 1010		
EOI Register	00 1011	rw	prioritization cycle
Remote Register	00 1100	r	
Logical Destination Reg.	00 1101	rw	
Destination Format Reg.	00 1110	rw	
Spurious Vector Register	00 1111	rw	
ISR (31:0)	01 0000	r	
ISR (63:32)	01 0001	r	
ISR (95:64)	01 0010	r	
ISR (127:96)	01 0011	r	
ISR (159:128)	01 0100	r	
ISR (191:160)	01 0101	r	
ISR (223:192)	01 0110	r	
ISR (255:224)	01 0111	r	
TMR (31:0)	01 1000	r	
TMR (63:32)	01 1001	r	
TMR (95:64)	01 1010	r	
TMR (127:96)	01 1011	r	
TMR (159:128)	01 1100	r	
TMR (191:160)	01 1101	r	
TMR (223:192)	01 1110	r	
TMR (255:224)	01 1111	r	
IRR (31:0)	10 0000	r	
IRR (63:32)	10 0001	r	
IRR (95:64)	10 0010	r	
IRR (127:96)	10 0011	r	
IRR (159:128)	10 0100	r	
IRR (191:160)	10 0101	r	

LOCAL UNIT REGISTERS (Continued)

Registers	Address (9:4)	SW	Side Effects
IRR (223:192)	10 0110	r	
IRR (255:224)	10 0111	r	
Intrpt Comnd Reg. (31:0)	11 0000	rw	send interrupt
Intrpt Comnd Reg. (63:32)	11 0001	rw	
Local Vector Table [timer]	11 0010	rw	
Reserved	11 0011		
Reserved	11 0100		
Local Vector Table [local int 0]	11 0101	rw	
Local Vector Table [local int 1]	11 0110	rw	
Reserved	11 0111		
Initial Count Register	11 1000	rw	
Current Count Register	11 1001	r	
Reserved	11 1010		
Reserved	11 1011		
Reserved	11 1100		
Reserved	11 1101		
Divider Configuration Reg.	11 1110	rw	
Reserved	11 1111		

NOTE:
Address space 101000 to 101111 and 111111 are reserved

13.0 TIMING DIAGRAMS

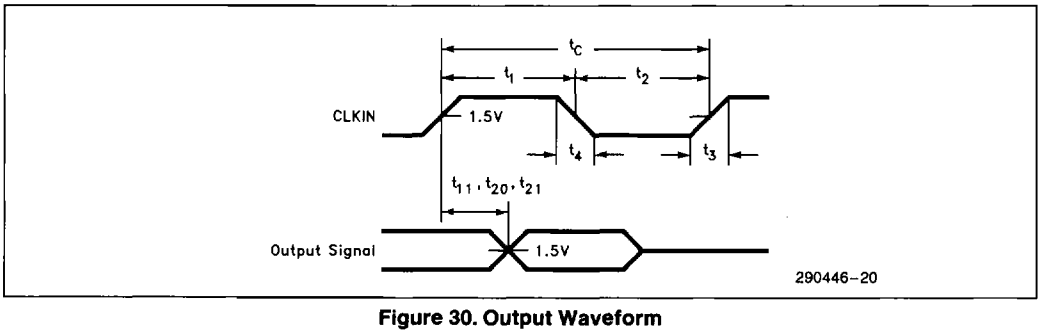


Figure 30. Output Waveform

13.0 TIMING DIAGRAMS (Continued)

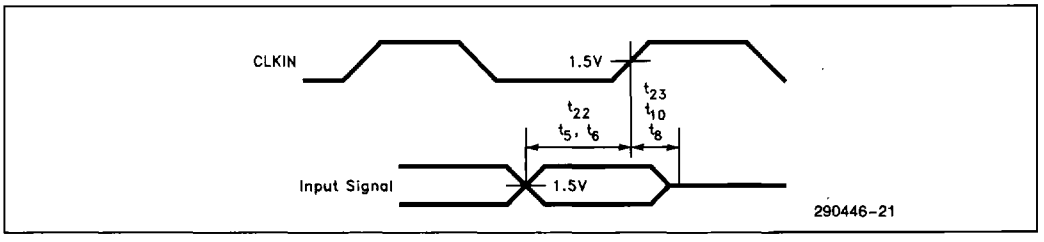


Figure 31. Input Waveforms

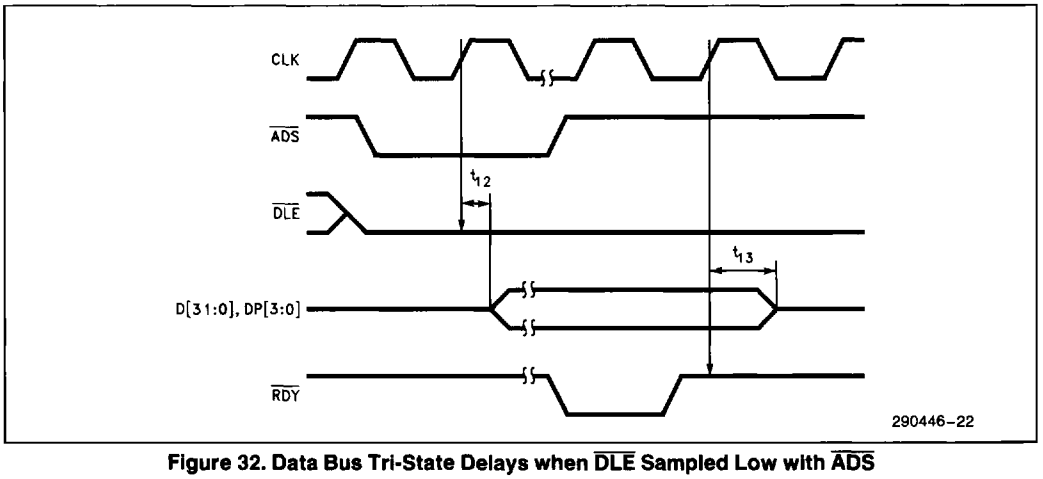


Figure 32. Data Bus Tri-State Delays when DLE Sampled Low with ADS

13.0 TIMING DIAGRAMS (Continued)

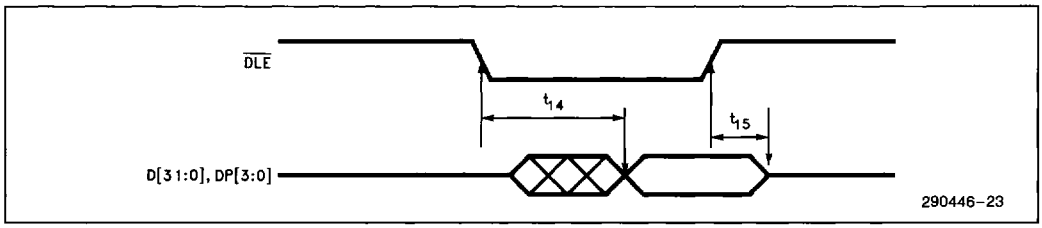


Figure 33. Data Enable/Disable Delay when $\overline{DLE}$ is Sampled High with $\overline{ADS}$

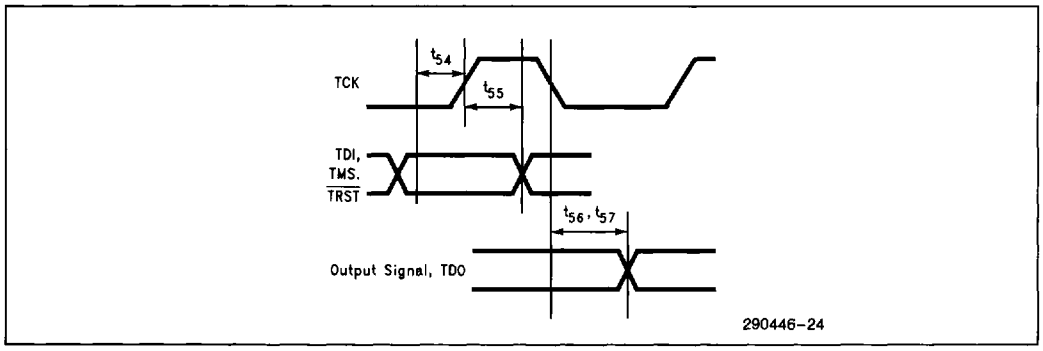


Figure 34. TAP Signal Timings

4

13.0 TIMING DIAGRAMS (Continued)

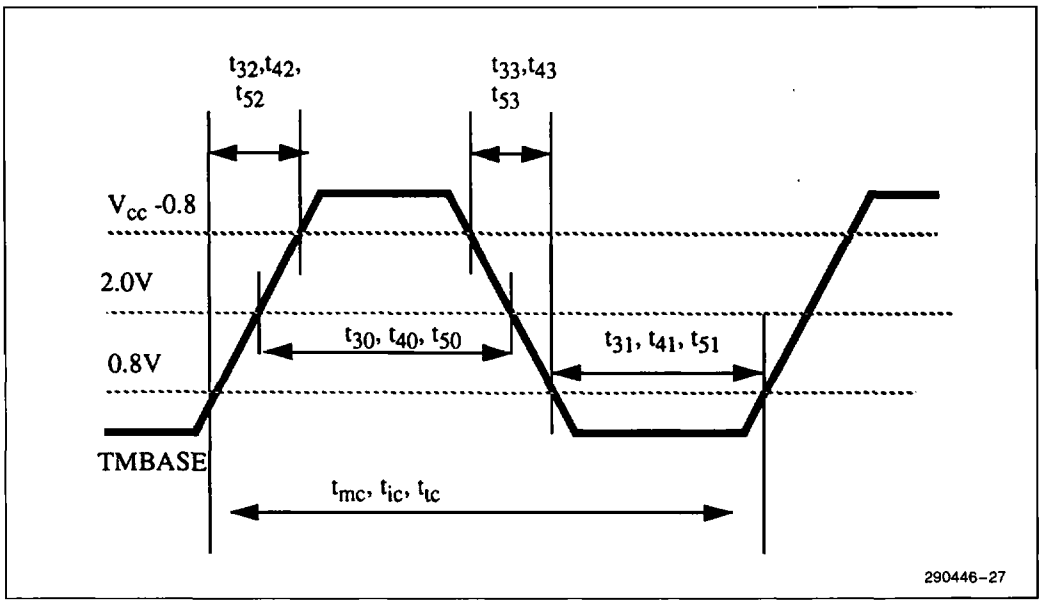


Figure 35. TMBASE, ICLK, TCK Timing

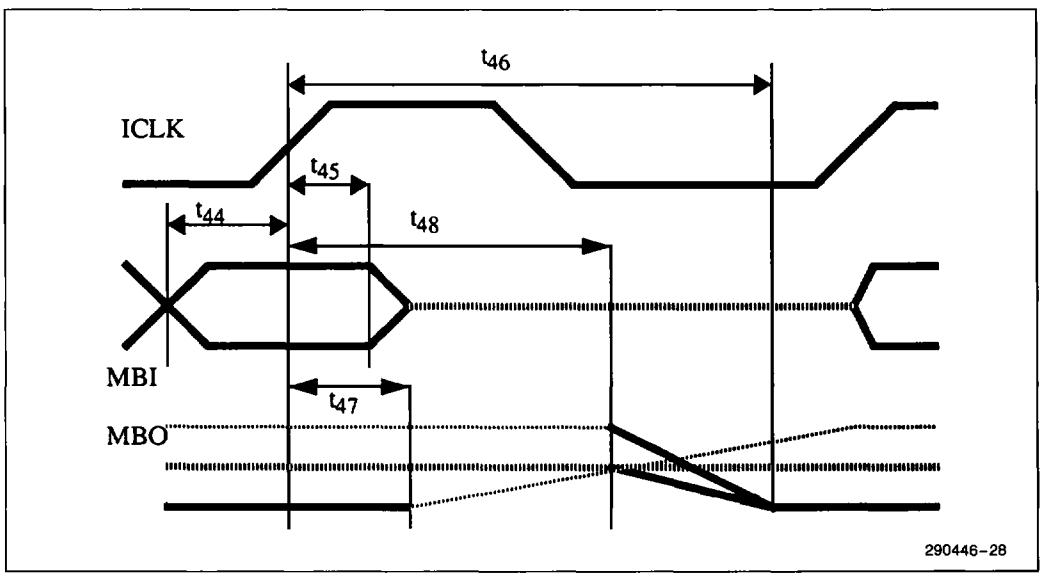
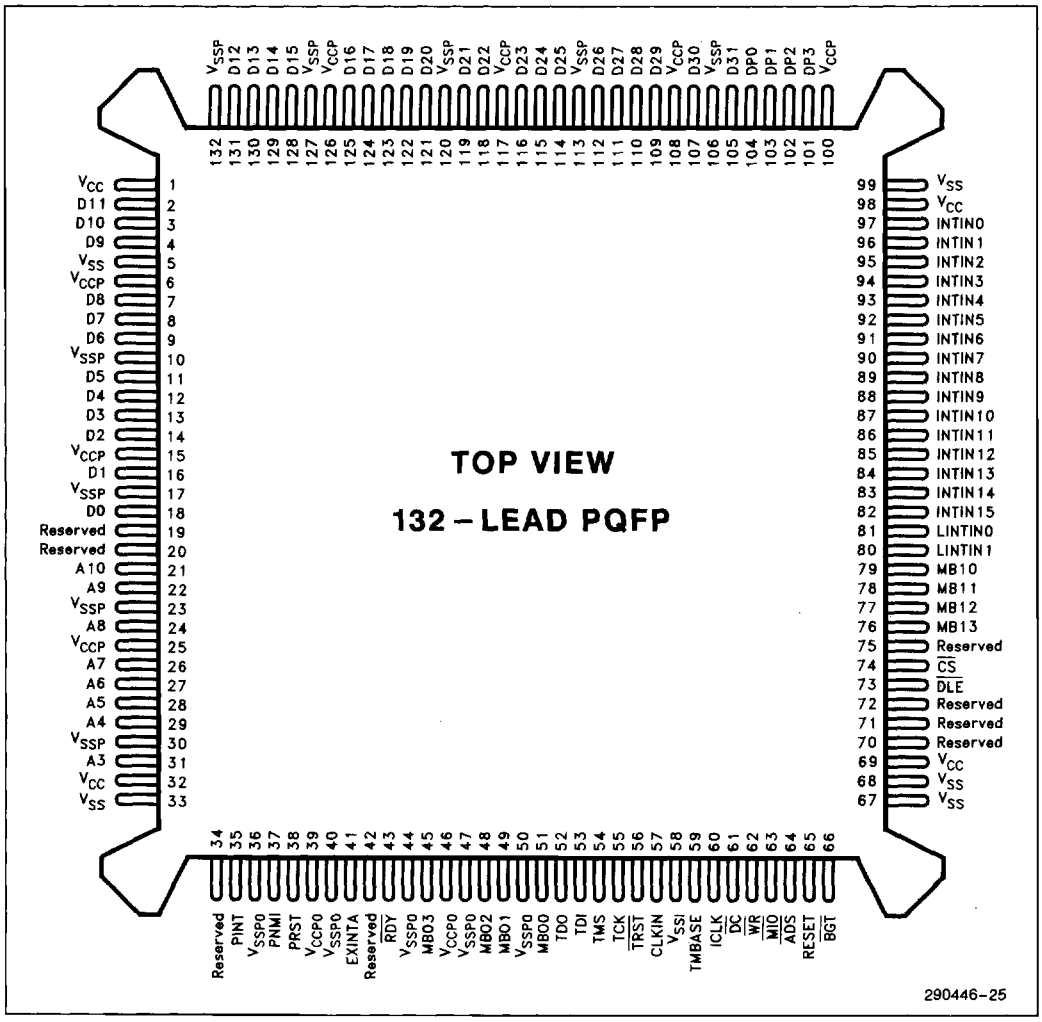


Figure 36. ICC BUS Open-Drain Output Delay

14.0 PACKAGE PIN-OUT

132-Lead PQFP
Package Type KU (See Packaging Specification. Order Number 240800)

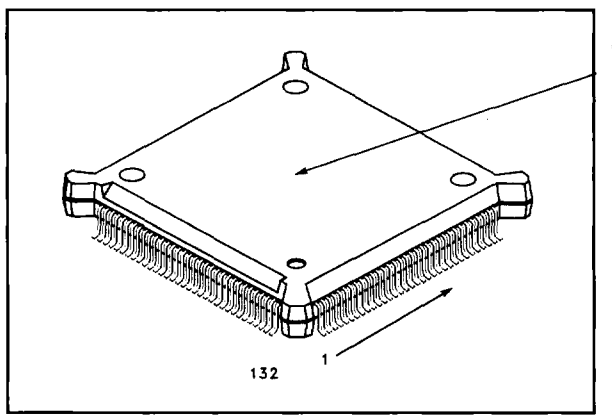


NOTE:
See pin description section for appropriate pin-strapping of the reserved pins.

15.0 PACKAGE THERMAL SPECIFICATION

The 82489DX is specified for operation when the case temperature is within the range of 0°C to +85°C. The case temperature may be measured in any environment, to determine whether the device is within the specified operating range.

The PQFP case temperature should be measured at the center of the top surface opposite the pins, as shown in Figure below.



Plastic Quad Flat Pack (PQFP)

PQFP Package Thermal Characteristics

Parameter	Thermal Resistance—°C/W					
	Air Flow Rate (Ft./Min)					
	0	200	400	600	800	1000
θ Junction to Case	8	8	8	8	8	8
θ Junction to Ambient	32.5	25.5	20	18.5	16	15

NOTES:
1. Table above applies to 82489DX PQFP plugged into a socket or soldered directly into the board.
2. $\theta_{JA} = \theta_{JC} + \theta_{CA}$.

16.0 GUIDELINES FOR 82489DX USERS

16.1 Initialization

This section outlines one possible initialization scenario. Other scenarios are certainly possible, and one would be selected as part of a platform standard initialization scheme. The intent of this section is to illustrate that the initialization support provided by the 82489DX is adequate to support MP (Multiprocessor) system initialization.

Each 82489DX has a RESET input pin connected to a common Reset line. Upon system reset, this common reset line is activated, causing all the 82489DXs to go through reset. All 82489DX local units (note: only local units and not I/O units) latch their ID from their address bus on reset. The ID can be provided by the bus control agent based on slot number.

The local units next assert their processor's Reset pin, holding the processor in reset, and next perform their internal reset, setting all registers to their initial state. The initial state of all 82489DX Units (both local and I/O units) is "all masks set" and all Local Units disabled; registers are otherwise initialized to zero. Note that the PINT and PNMI output pins are in tri-state mode when the local unit is disabled. After this, each 82489DX local unit will deassert its processor's Reset pin, allowing the processors to come out of reset and perform self test and start executing initialization code.

Note that while connecting PRST pin it should be noted that whenever PRST pin is activated by 82489DX either because of software reset message or hardware reset, the 82489DX itself is reset. It should be taken care in the cases of Warm reset where only processors need to be reset and not the interrupt controller. In brief, the usage of PRST depends upon the system requirement on various reset.

Somewhere in this code sequence, the processors that are "alive" will enable their 82489DX local units, and attempt to force all the other processors back into Reset. Forcing the other processors into reset is performed by sending them the inter-processor interrupt with Destination Mode = "Physical", Delivery Mode = "Reset", Trigger Mode = "Level", Level = "1", and Destination Shorthand = "All Excl Self". Only the first processor to get the ICC bus will succeed in sending this signal and reset all other 82489DXs and their processors. The other processors are kept in reset until such time that an MP operating system decides they can become active again. The only running processor next performs the rest of system initialization.

Eventually, an MP operating system will be booted at which time the operating system would send "deassert reset" interprocessor signals to activate the other processors in the system. A mechanism must be provided by the platform that allows the added processors to differentiate the very first reset from a subsequent one.

16.2 Compatibility

COMPATIBILITY LEVELS

The 82489DX can be used in conjunction with standard 8259A-style interrupt controllers to provide a range of compatibility levels.

At the lowest level we have "PC shrink-wrap" compatibility. This level effectively creates a uniprocessor hardware environment within the MP platform capable of booting/running DOS shrinkwrap software. In this mode, only the 8259A generates interrupts and the 82489DX becomes a virtual wire. The interrupt latency can be minimized by connecting the 8259A interrupt to local unit directly.

The next level preserves the software compatible view of an 8259A but it allows more than one processor to be active in the system. This results in an asymmetrical arrangement, with one processor fielding all 8259A interrupts but with added inter-processor interrupt capability. In this mode, 82489DX "merges" 8259A interrupts with inter-processor interrupts. Existing I/O drivers would be bound to the compatible CPU and interface directly with the 8259A.

At the next compatibility level, 8259A compatible drivers can be mixed with native 82489DX drivers. Devices can generate interrupts at either 8259A or an 82489DX. This provides for partial symmetry as individual drivers migrate from the 8259A to native 82489DXs.

Another 8259A compatible point can be defined for MP systems. Each processor could have its own compatible 8259A controllers, allowing multiple processors to run compatible I/O drivers, but statically spreading the load across the available processors.

82489DX/8259A INTERACTION

The principle of compatible operation is very straightforward; the 82489DX(s) become a virtual wire connecting the 8259A's INT output through to the processor, while at the same time making 8259A visible to the processor.

The two connection schemes described only differ in the number of 82489DX(s) (one or two) that are located in the path from the 8259A to the processor. In the one 82489DX example illustrated in Figure 37, the INT output of the 8259A connects to one of the Interrupt Input pins of the 82489DX through an edge generation logic. This could be an interrupt pin on the 82489DX's I/O unit or local unit; assume a local interrupt input is used. The Local Vector Table entry for the interrupt pin that connects to the 8259A is set up with a Delivery Mode of "ExtINT" and edge trigger mode. This indicates that the interrupt is generated by an external controller. The processor's INT pin connects to the 82489DX PINT pin.

This setup enables the 82489DX local unit to detect assertions (up-edges) of the 8259A's INT output pin and pass this on to the processor's INT input. 82489DX asserts ExtINTA pin along with (one clock prior to) PINT pin to indicate "8259" interrupt. When the processor performs its INTA cycle the 82489DX itself does not respond other than deasserting PINT to the processor. At the third clock after $\overline{ADS}$ in the second bus cycle of INTA cycle ExtINTA is deasserted. External logic should make use of the ExtINTA signal to make the INTA cycle visible to the 8259A and the 8259A should provide the vector. At the same time, the local unit considers the external request as delivered, and need not wait for the external 8259A's INT to be deasserted. *A new up-edge must be generated on the 8259A INT pin before the local unit will assert the processor's INT pin on behalf of the 8259A. External edge generation logic should be used for this.* Compatible software interacts directly with the 8259A.

The mechanism is essentially the same in the two-82489DX scheme. The difference is that the 8259A connects to an interrupt input pin of the 82489DX I/O unit in the I/O system. The Redirection Table entry for this pin is again programmed with an "ExtINT" Delivery Mode, and the (single) 82489DX destination local ID corresponding to the compatible DOS processor. Capturing the up-edges of the 8259A's INT pin by the 82489DX local unit now involves sending messages from the 82489DX I/O unit to the 82489DX local unit via the ICC bus. The "virtual wire" now includes messages over the ICC bus.

Adding inter-processor ICC interrupts (or any other 82489DX generated interrupts) to the compatible operation is accomplished by having the 82489DX internally OR the 8259A's INT request with any 82489DX interrupt request.

Before the 82489DX actually sends the interrupt signal to the processor, the 82489DX decides whether it does this for an 82489DX interrupt or whether it does this on behalf of the external controller. When the processor performs the corresponding INTA cycle, only the 82489DX knows whether it should respond with a vector, or whether the external 8259A should.

If the 82489DX needs to respond, then it will enable an externally implemented trap that prevents the 8259A from seeing the INTA cycle. If the 8259A needs to respond, then the 82489DX will not enable the INTA trap, and the INTA will be allowed to reach the 8259A. 82489DX implements this by asserting its EXTINTA pin to indicate external 8259A should respond with the vector. The 82489DX local unit controls the INTA trap via its "ExtINTA" output pin; the 82489DX does not actually provide the trap itself.

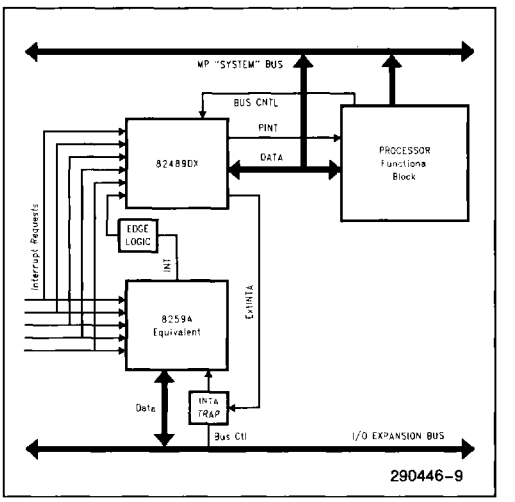


Figure 37. Edge Logic

82489DX/8259A DUAL MODE CONNECTION

In systems that can be booted either as a configuration with compatible 8259A or without, device interrupt lines are connected to both the Interrupt Request pins of the 8259A and Interrupt Input pins of the 82489DX with all interrupts either masked at the 82489DX or at the 8259A. Some EISA and Micro-Channel chip sets that include on-chip 8259As also have internally connected interrupt requests. For example, the 82357 (the ISP of the EISA chipset) generates timer and DMA chaining interrupts internally. These are not available as separate interrupts outside the ISP. In non compatible mode the ISP timers are not used, since each local 82489DX unit provides its own timer. Therefore, the ISP's 8259A is configured to mask out all interrupts except the DMA chaining interrupt which is configured in level-sensitive, auto EOI mode. This causes the 8259A's INT output to track the state of the internal DMA interrupt request. The 8259A's INT output is then connected to one of the 82489DX interrupt input pins programmed to generate a regular (i.e., not

"ExtINT") level-sensitive interrupt. The ISP 8259A then no longer functions as an external interrupt controller; it has been logically disabled, and it needs no interrupt acknowledge or EOI. The INTA and EOI cycles occur only at the 82489DX. It should be noted that 82489DX accepts only active high level/edge interrupt inputs. External programmable logic should take care of polarity reversal that may be needed in EISA system for sharing of interrupts.

16.3 Hardware Guidelines

82489DX HARDWARE STATE ON RESET

The 82489DX goes to reset state either by Hardware Reset state or Software Reset message received on the ICC bus. On reset, 82489DX is disabled. The following is the hardware state of 82489DX after reset.

PRST	Active (HIGH)
PNMI	TRI-STATED (Internal Pull-Down Provided)
PINT	TRI-STATED (Internal Pull-Down Provided)

82489DX is disabled on Reset and unless specifically enabled, it does not start its interrupt mechanism.

The difference between hardware reset and software reset message is that during hardware reset 82489DX samples the address bus and stores the last sample in Local Unit ID whereas for software reset it does not sample and store the unit ID. In addition, during the hardware reset pulse should be wide enough to accommodate for at least one rising and falling edge of ICLK. On hardware reset ExtINTA is held high.

PULL UP AND PULL DOWN RESISTORS

PNMI, PINT are tri-stated at power on and they are maintained in tri-state condition till the unit is enabled. Eventhough internal pull down resistor is provided on PNMI and PINT external additional pull down resistor may be needed depending upon the loading on these pins by external logic. The DC characteristics gives the control specification from which the value of resistor, if needed, can be calculated.

It should be kept in mind that the ICC bus being electrically open drain bus requires pull up resistors at the MBO pins. ICC bus output low current is just 4 mA.

PINT and ExtINTA Timings

It should be noted that for ExtINTA type of interrupts **PINT gets activated one clock after ExtINTA gets activated.** When getting deactivated, both PINT and ExtINTA gets deactivated at the same clock edge.

ExtINTA Timings

In the interrupt acknowledge cycle for External Interrupt control, 82489DX asserts ExtINTA. It decodes the type of cycle from CPU control signals like M/$\overline{IO}$, D/$\overline{C}$ and W/$\overline{R}$. CPU does two bus cycles back to back for interrupt acknowledge cycle. 82489DX maintains ExtINTA active throughout the first cycle. For next cycle (when the vector will be given by external 8259) after 82489DX senses the start of the cycle (by $\overline{ADS}$) 82489DX deactivates ExtINTA. External control logic may be inserting wait states to match the 8259 timings. Since 82489DX has no way of finding out the cycle completion, 82489DX deasserts ExtINTA before the second bus cycle gets completed. This should be kept in mind while using ExtINTA for external interrupt control logic.

82489DX AND MEMORY MAPPING

The 82489DX is a 32-bit high performance interrupt controller. It allows the CPU to do 32-bit read and write to it. By memory mapping 82489DX the system performance can be enhanced. It should be noted that 82489DX does not support pipelining. Eventhough 82489DX can be memory mapped, its functionality as an interrupt controller should be kept in mind while programming the virtual memory management control data structure. The caching policy for the page where 82489DX is mapped should also be done with the functionality of 82489DX in mind. For example, the reads to 82489DX should not be cached and writes should be write-through. Since 82489DX registers are aligned at 128-bit boundaries, memory mapping 82489DX with interleaved memory system should not be a problem.

JTAG CIRCUIT CONSIDERATIONS

The JTAG circuit is used for boundary scan test. The JTAG pins has a TCK, (JTAG clock), $\overline{TRST}$, (JTAG Reset), TDI, (Test Data Input), TMS, (Test Mode Select) and TDO, (Test Data Output).

The JTAG circuitry, if not used, should be properly deactivated so that it will not interfere in the normal functional operations. The JTAG can be inactivated in any one of the following ways:

1. JTAG inactivation through $\overline{TRST}$: The TMS, Test mode Select should be either left open (internal pull up is provided) or tied to V_{CC}. The $\overline{TRST}$ can be pulsed low (bring it low and after meeting the pulse width requirement bring it back high again) to keep the JTAG circuitry to idle state. The $\overline{TRST}$ pulse brings the JTAG circuitry to idle state and TMS being kept high maintains the JTAG circuitry in idle state.

2. JTAG inactivation through TCK: The TMS, Test mode select should be either left open (internal pull up is provided) or tied to V_{CC}. The TCK within 5 clocks brings the JTAG circuitry to idle state. The TMS, being held at logic high level, maintains the JTAG circuitry in idle state.

16.4 Programming Guidelines

The 82489DX register data structure contains different fields to specify the mode of operations and the options available within each mode. Since certain options are applicable to specific modes only (for example "Remote Read" mode applies only to interrupt command register, it does not have relevance to I/O unit's redirection tables) the following programming guidelines are provided.

UNIQUE ID REQUIREMENT

All the local units and I/O units hooked in a ICC bus should have unique ID before they can use the bus. This should be ensured by the programmer since for ICC bus arbitration the units (whether it is local unit or I/O unit) arbitrate with their unit ID.

For future compatibility, the Units should be assigned IDs starting with 0, 1, 2 etc. with the highest ID in the system being number of units minus 1. So in a four 82489DX system there are four local units and four I/O units. The ID starts with 0 and the highest ID in the system will be 7. Note that each unit should have different ID in the system.

ATOMIC WRITE READ TO TASK PRIORITY REGISTER

Normally, the task priority register is written with highest priority to mask certain low level interrupts before entering into critical section code. In a system where 82489DX is memory mapped the CPU may buffer this task priority register write to its on chip write buffer. The following scenerio can happen in such situation: CPU posts task priority register write to its on chip write buffer and enters into the critical code. A lower priority interrupt (which should not enter the critical code) interrupts the CPU before the write buffer gets flushed into task priority register). The CPU accepts the lower priority interrupt. To avoid the situation atomic write read to task priority register should be done. The read following write ensures that the write buffer is flushed to task priority register and the atomicity ensures that no interrupt will be accepted by the CPU during its write to task priority.

It should be noted that if the CPU does interrupt acknowledge cycle only after flushing the write buffers then the above situation may not arise.

CRITICAL REGIONS AND MUTUAL EXCLUSION

Each 82489DX has a single Interrupt Command Register that it uses to send interrupts to other processors. The programmer should make sure to synchronize access to this register. Specifically, 1). writing all fields of the register, 2). Sending the interrupt message (by writing the LSB register), and 3). waiting for Delivery State to become Idle again, should occur as a single atomic operation. For example, if interrupt handlers are allowed to send inter-processor interrupts, then interrupt dispensing to the processor must be disabled for the duration of these activities.

INTERRUPT COMMAND REGISTER PROGRAMMING SEQUENCE

The interrupt command register (31:0) has a side effect of sending interrupt once it is written. The destination is provided in the interrupt command register (63:32). So always interrupt command register (63:32) should be programmed before programming interrupt command register(31:0).

Program Interrupt Command Register (63:32)

↓

Program Interrupt Command Register (31:0)

INTERRUPT VECTOR

Two different interrupts should not be programmed with the same interrupt vector.

LOCAL AND I/O UNIT

Only Interrupt command register supports "Remote Read" Delivery mode. Local and I/O unit interrupts do not support "Remote Read".

ICR (INTERRUPT COMMAND REGISTER)

1. ExtINTA delivery mode is not supported for all destination shorthands.
2. "Remote Read" should always be programmed as "Edge" triggered interrupt.
3. "Remote Read" should always be programmed with physical destination mode (and not with Logical Destination mode). Broadcast addressing should not be used for Remote Read.
4. For "all incl Self" and "All exc. self" destination shorthands, "remote read" delivery mode should not be used.
5. For "all incl self" and "self" destination shorthands "Reset" delivery mode should not be used.

ICR (INTERRUPT COMMAND REGISTER)
(Continued)

6. For "all exc self" destination shorthand if "Reset" delivery mode is used, it should be ensured at system level that only one processor executes this instruction at any time.
7. Messages could be sent out in "Logical" or "Physical" mode with destination ID of all 1's depending on the way Destination mode entry is programmed. In brief, "All incl self" and "All exc. Self" support both "Logical" and "Physical" addressing mode.
8. When destination shorthand (i.e., broadcast) is used with lowest priority destination mode, then even though all participates in arbitrating for destination, only the lowest priority gets the message. So even though the addressing is broadcast since the destination mode is lowest priority only one gets the message.
9. When destination shorthand (i.e., broadcast) is used with "Fixed" destination mode, then all the units get the message.

ISR/IRR/TMR

Bits 0–15 of IRR/ISR/TMR do not track interrupt. No interrupt of vector numbers from 0–15 can be posted. The total interrupt supported are 240. When reading the lowest 32 bits of these registers, 0 will be returned for the lower 16 bits.

FOCUS PROCESSOR

Focus processor is applicable only within the addressed units.

ExtINT Interrupt Posting

The external interrupt has no priority relationship with the 82489DX priority. But when posting an interrupt to the processor, if both an external interrupt and a 82489DX interrupt are pending, 82489DX could post either one to the processor. In 82489DX implementation, it would post external interrupt whenever there is no other 82489DX interrupt that can be posted to the processor. It should be also noted that External interrupts can not be masked by raising task priority. However, they can be masked by the mask bit in the table entry for the ("ExtINTA") external interrupt.

The extINT interrupts are specific in their characteristics in that they do not have any priority relationship with the rest of the interrupt structure. ISR and IRR bits in 82489DX are used to do the housekeeping functions for interrupt priority. Since extINT interrupts do not have any priority relationship, ISR and IRR bits are not maintained for external interrupts. As far as interrupt acceptance is concerned, if more than one extINT interrupts are directed towards a local unit, that local unit treats all the extINT interrupts directed to it as only one extINT interrupt. This leads to an important point that in a system not more than one interrupt should be programmed as extINT interrupt type with the same destination. It should be noted that there can be more than one extINT type of interrupt in a system with each having different local unit as destination.

Synchronizing Arb IDs

Initialization of an 82489DX's local unit ID is implementation dependent. In some platforms, power-on reset will latch the right values into the 82489DXs; in other platforms, unique IDs may be assigned by initialization firmware. In both cases the 82489DX I/O unit should be assigned unique ID by initialization firmware. The important point is that the 82489DXs are required to have unique IDs before they can use the bus, and in addition, all their Arb IDs must be "in sync". Synchronizing Arb IDs is accomplished as a side effect of a "deassert reset" interrupt command. This resets the (rotating) Arb ID to the (constant) unit ID; it assumes that all 82489DXs have their unique ID.

4

LOWEST PRIORITY

"Only once delivery" semantics for a group destination is guaranteed only if multiple fixed delivery of the same interrupt vector are not mixed. For lowest priority arbitration to work, all the arbitration ID of local 82489DXs in the system should be in sync. This means after local unit IDs are written in all local units (each ID should be different from other IDs) a RESET DEASSERT message should be sent in ALL INCLUSIVE mode. The RESET DEASSERT message should be sent before system is used for lowest priority arbitration. This ensures that all ARB IDs are also different. (Arb IDs are copied from local unit IDs during RESET DEASSERT message.)

The RESET DEASSERT message, if not sent, only one delivery semantics may not be guaranteed in the cases where lowest arbitration is used in the system.

DISABLING LOCAL UNIT

Once the 82489DX is enabled by setting bit 8 of spurious vector register to 1, the user should not disable the local unit by resetting the bit to 0. The result will put the local unit in an inconsistant state. The local unit can be disabled by sending "reset" interrupt message to the local unit.

ISSUING EOI

EOI, End of Interrupt issuing indicates end of service routine to 82489DX. The ISR bit which is set during INTA cycle gets cleared by EOI. This section discusses the relevence of EOI to the specific types of interrupts and its timing related to interrupt deassertion.

EXTERNAL INTERRUPTS AND EOI

External Interrupts should be programmed as edge type. INTA cycles to external interrupts are taken automatically as EOI by 82489DX. This is similar to AEOI, Automatic End of Interrupt of 8259A. So there is no need to issue EOI to 82489DX for external interrupt servicing. This is done to achieve software transparency in the compatible mode.

SPURIOUS INTERRUPTS AND EOI

Spurious interrupts do not have any priority relationship to other interrupts in the system. So IRR is not set for spurious interrupts. EOI should not be issued for spurious interrupts. It is advisable not to share the spurious interrupt with any vector. If spurious interrupt vector is shared with some other interrupt then while servicing issuing EOI depends on the source of interrupt. If the source is spurious interrupt (for which the corresponding IRR is not set) then EOI should not be used. If the source is a valid interrupt sharing the spurious interrupt vector (for which the IRR is set) then EOI should be issued.

NM AND EOI

For NM type of interrupt no IRR bit is set. So EOI should not be issued while servicing NMI type of interrupts.

TASK PRIORITY REGISTER

Task priority register is used to specify the priority of the task the processor is executing. In 8259 the priority is defined only among the interrupts that it handles. 82489DX goes farther ahead in handling priority. In multitasking system, in addition to device interrupts various tasks have different priority and 82489DX allows consideration of the priority at system level. The processor specifies the priority of the task it executes by writing to task priority register. Now any interrupts at and below the task priority will be masked temporarily till the task priority gets lowered. The masking granularity is at priority level. Out of 256 interrupt vectors 16 priority levels are specified and 16 vectors share one priority level. Since the masking granularity by the task priority register is at priority level, group of 16 vectors get masked when task priority register is increased by one level.

When task priority is at its minimum level of 0, interrupt vectors having level 1 to 16 are passed to CPU. Stated in other words, even when the task priority register is at its minimum (of level 0), interrupt vectors at level 0 will be masked. This means that the interrupt should not be programmed with vectors 0 to 15. So out of 256 interrupt vectors, only 240 interrupt vectors (vector 16 to 255) can be used in 82489DX.

ExtINT INTERRUPT AND TASK PRIORITY

ExtINT interrupt does not have any priority relationship with other interrupts or task priority register. So ExtINT interrupt can not be masked by raising task priority. They can be masked by writing to the vector table entry which corresponds to ExtINT interrupt.

REMOVING MASKS

When enabling units and removing Mask bits in situations where a device may already be injecting interrupts into the 82489DX system, the Mask in the Redirection Table should be removed last to ensure proper initial state (e.g., Remote IRR bit matching IRR in local unit).

DELIVERY MODE AND TRIGGER MODE

It is software's responsibility to make sure that Delivery Mode and Trigger Mode are set to meaningful combinations as listed below.

Delivery Mode	Trigger Mode
Fixed	edge/level
Lowest Priority	edge/level
Remote Read	edge
NMI	level
Reset	level
ExtINT	edge

Software is also responsible for not using meaningless Delivery Modes in Redirection Table entries and local Vector Table entries (e.g., use of Remote Read delivery mode).

ASSIGNING INTERRUPT VECTORS

Software has total control over the assignment of interrupt vectors to interrupt sources. The operating system writer should be aware of a number of things when doing this assignment.

Some processor architectures assign a predefined meaning to some of the vectors (i.e., entries in the interrupt table) as entry points to certain trap and exception handlers (e.g., divide error, invalid opcode, page fault, etc.). The programmer is strongly advised not to reuse these vectors.

The programmer must also be careful when using the same vector number to represent different interrupt sources (sharing vectors). This is especially true for level triggered interrupts. When multiple sources with different Redirection table entries share an interrupt vector, any of the sources deactivating its level signal will remove the interrupt request for all sources. Giving each interrupt source its interrupt vector in any case is the preferred approach.

SENDING INTER-PROCESSOR INTERRUPTS

Each 82489DX has a single Interrupt Command Register that it uses to send interrupts to other processors. It is software's responsibility to synchronize access to this register. Specifically, 1) writing all fields of the register, 2) sending the interrupt message (by writing the low register), and 3) waiting for Delivery State to become Idle again, should occur as a single atomic operation. For example, if interrupt handlers are allowed to send inter-processor interrupts, then interrupt dispensing to the processor must be disabled for the duration of these activities.

DELAY WITH LEVEL TRIGGERED INTERRUPTS

When a level triggered interrupt source deasserts its interrupt input, the destination will clear the interrupt's IRR bit only after receiving the message from the ICC bus. This introduces a small delay between the removal of the interrupt at the source and the removal of the interrupt at the processor. To avoid generating unnecessary interrupts, the interrupt handler should remove the interrupt at the source (at the device) as early as possible in the handler.

In any case, handlers should be able to deal with unnecessary interrupts.

RESET DEASSERT

A side effect of a reset deassertion message broadcast in the ICC bus is that all 82489DX local units reset their Arb ID to their unit ID. Interrupt commands that use the "self" Destination Shorthand do not generate a message on the ICC bus. If software only wants to generate the side effect of resetting Arb IDs, it should use a command with Logical Destination Mode and a Destination field containing all zeroes.

INTERRUPT MASKING

There are a number of levels at which interrupts can be masked, each resulting in a different behavior on interrupt delivery.

- First, interrupt injection (or deliver) can be masked by setting the Mask bit in the interrupt's Redirection Table or local Vector Table entry. These interrupts are ignored, no message is sent for them. Granularity is an individual interrupt.
- Second, each 82489DX can individually mask interrupt dispensing by raising its Task Priority to some level. This 82489DX will not dispense interrupts to its processor of this and lower priority unless it is currently the focus of the interrupt. Note again that the 82489DX is designed to operate as fully nested with non-specific EOI (to use existing 8259A terminology). There is no explicit interrupt mask (such as MR) and there is no notion of specific EOI.
- Third, each processor may provide a mechanism that masks all interrupt dispensing to it using the processor supplied instructions or status bits to do so. This does not interfere with lowest priority arbitration of the processor's 82489DX local unit.

CHANGING REDIRECTION TABLES

Redirection Tables are typically set up at initialization time. When modifying a Redirection Table entry "on the fly" the programmer must be aware of state kept at other 82489DXs relative to the interrupt being modified.

DEVICE DRIVERS WITH 82489DX

It is strongly recommended to read the device status registers before servicing the device. This is because if an edge triggered device deasserts its interrupt before interrupt acknowledge cycle (it should NOT) 82489DX will NOT give spurious vector. It will give genuine interrupt vector corresponding to the device. So, interrupt service routine should validate the interrupt request before servicing the device.

SYSTEM HARDWARE AND SOFTWARE DESIGN CONSIDERATIONS

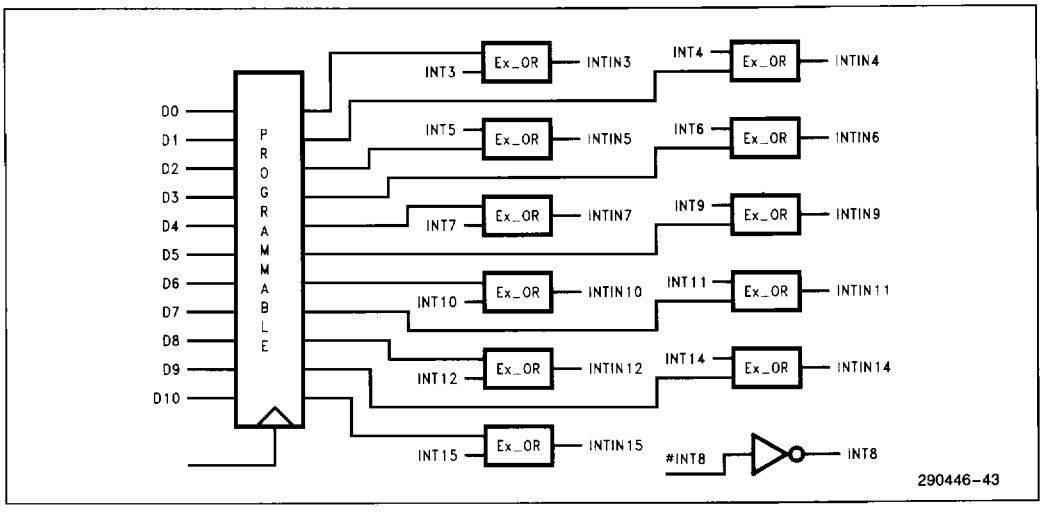
Design Consideration 1

Description: The following design consideration has to be taken care of when using ISP (82357) as external interrupt controller. 82489DX allows connecting external 8259 type interrupt controller at one of its inputs. The mode associated with the interrupt input which has 8259 connected to it is called ExtINTA mode. 82489DX allows only EDGE TRIGGERED programming option for ExtINTA mode. But in the case of 82357, the INT output from ISP stays high in case more than one interrupt is pending at its inputs. It does not always inactivate its INT output after INTA cycle. This will lead to a situation where ISP keeps the interrupt at high level continously and waits for INTA cycle. But since 82489DX expects an edge for interrupt sensing (for ExtINTA interrupts) it does not pass the interrupt to CPU and further interrupts are lost. So External circuitry should monitor the end of SECOND CYCLE of INTA cycle and force an inactive state at 82489DX's input. To avoid glitches at 82489DX input, this external logic should clear its output only at the end of second INTA cycle. It should be set by high going 82357 output. It should never be cleared by low going 82357 output. That is it should not follow 82357 output.

Design Consideration 2

Description: The following design consideration has to be taken care of when using 82489DX in EISA systems. EISA ISP(82357) chip integrates 8259A. It additionally allows sharing of interrupts. To facilitate this sharing it has a programmable register, ELCR (Edge / Level trigger control register) by which certain interrupt inputs can be programmed as edge (low to high except for RTC) or level (the level is active low). The determination of edge or level is done during initial configuration of EISA system by reading EISA add in boards from the interrupt description data structures. The solution is to have programmable logic at the interrupt inputs so that 82489DX is compatible with EISA ISP. This will introduce one more register and logic to support this. This should be an 11-bit programmable register and an array of ExOR logic (12 ExOR gates or equivalent PLD). The ISP allows programmability of the following interrupts.

INT3 INT4 INT5 INT6 INT7 INT9 INT10 INT11 INT12 INT14 INT15. In addition to the above 11 interrupts, it fixes **INT8** to be active low edge triggered interrupt. INT8 is the only case where it is active low edge triggered type. So the following logic can be used to add programmability in 82489DX based EISA system. Before connecting these 11 interrupt lines directly (#INT8 which is from Real Time Clock is always active low edge triggered. #INT8 can be passed through an inverter since there is no need for programmability) to the 82489DX they should pass through an array of 11 Ex__OR gates. One input of Ex__OR gate connects to the corresponding INT pin and other input connects to a bit of programmable register. The output of Ex__OR gate is connected to 82489DX. The idea of Ex__OR is to use as a controlled inverter.

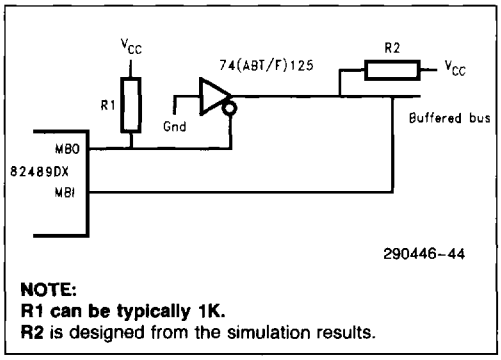


INTIN are the interrupt inputs to the 82489DX and INT are the system interrupt. The Ex_OR gating register is programmed after EISA configuration is found from add in boards as how these interrupt lines are going to be used in that particular configuration. If a particular input is edge triggered, then the corresponding bit in the register is written with 0. If a particular input is level triggered, then the corresponding bit in the register is written with 1.

8259 by itself does not have polarity control whereas 8259 when implemented in EISA chipsets have the polarity control. Similarly APIC does not have by itself polarity register. So polarity register should be programmed as a part of system BIOS and not APIC BIOS.

Design Consideration 3

I_{CC} bus drive is an open drain bus with drive capacity of 4 mA only. Since data is transmitted at each I_{CC} clock, the "charging" of I_{CC} bus should be fast enough to ensure proper logic level at each clock edge. The I_{CC} bus needs pull up resistors since it is open drain bus. Since the drive is only 4 mA, the pull up resistor value can not be less than 5V/4 mA. This being the limit of the resistor value, the length and the characteristics of the I_{CC} trace forces a capacitance value. Both the resistor and capacitance brings a RC time constant to the I_{CC} bus waveform. So, Electrical consideration has to be given to and practice of controlled impedence should be exercised for layout of the I_{CC} bus. The length of the trace should be kept as minimum as possible. If the length of the I_{CC} bus can't be kept less, than say 6 inch, because of mechanical design of the system, the external line drivers should be added to I_{CC} bus and I_{CC} bus should be simulated with the added driver characteristics.



NOTE:
R1 can be typically 1K.
R2 is designed from the simulation results.

Design Consideration 4

This is related to ADS#, BGT# and CS# timings. For bus cycles not intended for 82489DX, (CS# = 1 where 82489DX is supposed to sample it), any change in CS# line while the ADS# is still active, may erroneously cause a RDY# returned from 82489DX. Anomolous behavior may result if for BGT# ties low cases

a) BGT# goes away just one clock after ADS# or

b) ADS# is still active, and CS# changes during this period.

For other cases anomolous behavior results if CS# changes when ADS# is still active. The following considerations are important from timing point of view. Always limit the pulse width of 82489DX ADS# to one CLKIN. Also avoid changing levels on BGT#/CS# line, when ADS# is active for cases being identified as BGT# tied low (BGT# sampled low when ADS# goes active). Also avoid changing levels on CS# line when BGT# is active.

Design Consideration 5

82489DX does not recognize the interrupt when an edge occurs at the interrupt input pin while interrupt is masked. When later it is unmasked there is no further edge and so 82489DX never passes that forgotten edge and that interrupt channel becomes unusable after that.

The recommendation is that first 82489DX should be unmasked and then the device interrupt should be enabled in the device register. By this, software can ensure that always an edge will occur after an interrupt is unmasked.

Design Consideration 6

Description: Edge triggered interrupts should not deassert their output till they are acknowledged by INTA cycle from CPU.

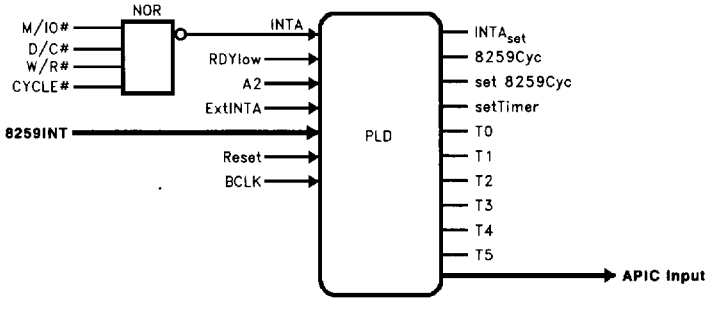
Issue:

82489DX employs glitch detection logic for edge triggered logic. To make sure the detected edge interrupt is not a glitch, 82489DX samples the input again before sending the interrupt message. The time difference between the first sampling of interrupt to be active and second sampling (just before sending the interrupt) is not a constant number. This is because the ICC bus might have been occupied by other messages. So, for example if during first sampling it was detected that INTIN0 and INTIN15 are both active and after sending INTIN0 it samples INTIN15 again before sending message for

INTIN15. But between this time ICC bus might have been occupied by other messages. So even if an edge triggered interrupt is held active high for a really long time and then brought low before INTA cycle, it is considered as a glitch. Because it may happen that the second sampling occurred just when the interrupt line got low.

Once the glitch detection circuitry found this "glitch", it goes back to the state where it will start sampling and waiting for an active edge to occur. This takes more than one clock cycle (CLK) and if the "glitch interrupt" generates an edge before that time after the second sampling of low level is done, then the edge is lost forever.

Since the time when the second sampling is done is unknown, the best way is to make sure the edge triggered interrupts do not deassert their outputs till they are acknowledged by INTA cycle from CPU. It is found that in some cases 8259 can generate brief active low pulses on its output. So the glue logic between 8259 and 82489DX input pin should make sure that 82489DX input pin is clear only after getting second interrupt acknowledge cycle. The glue logic should not just follow the 8259 output. Put in other words, after interrupt acknowledge cycle to 8259, if the 8259 input is seen active high, it should generate an edge at 82489DX input. Moreover, even if 8259 output goes low the glue logic should not lower its output since the only time when the glue logic can deassert its output is when it finds an interrupt acknowledge cycle for 8259. The following PLD equations and schematics serves as an example for the glue logic between 8259 and 82489DX.



290446-45

BCLK is the standard EISA BCLK.
CYCLE# is an external signal to indicate ADS# is asserted earlier and RDY low is not yet returned.

```
APIC input =    /T0  * /Reset * /APIC input * 8259INT  * INTA set  ; Sample 8259 interrupt
             + /T0 * /Reset * APICinput  * INTA set          ; Hold till it is cleared by delayed interrupt acknowledge
             + / INTA set * 8259INT
Set 8259Cyc  = /Reset * INTA * ExtINTA    ; This INTA cycle is for 8259

8259Cyc      = set8259Cyc * /T0 * /Reset   ; Set 8259cyc will set 8259 cycle and T0 will clear it
             + /T0 * /Set8259Cyc * 8259Cyc * /Reset ; Hold 8259 cycle till T0 clears it

INTA set     = /Reset * /INTA set * INTA ; wait for very first INTA cycle after reset
             + /Reset * INTA set ; once first INTA cycle after Reset is found, set the INTA set

Set timer    = 8259cyc * /A2 * /RDYlow    ; Start the timer at end of second INTA cycle

T0           = Set timer * /T5  * /Reset  ; Set timer will set T0 and T5 will clear
               + /Set timer * T0 * /T5 * /Reset ; Till T5 clears it hold T0

T1 := T0 * /Reset * /T5  ; Follow T0 after one clock for setting but clear along with T0

T2 := T1 * /Reset * /T5  ; Follow T0 after two clock for setting but clear along with T0

T3 := T2 * /Reset * /T5  ; Follow T0 after three clock for setting but clear along with T0

T4 := T3 * /Reset * /T5  ; Follow T0 after four clock for setting but clear along with T0

T5 := T4 * /Reset        ; Follow T0 after 5 clock for setting
```

290446-46

NOTES:
T1, T2, T3, T4 and T5 are clocked signals and others are combinatorials.
This circuit and PLD equations are given for concept clarification purpose. They are not tested.
$INTA_{set}$ is needed so that some 8259 logic at power on activates its INT output to 1 and it deactivates its output after only 8259 initialization (which should happen after APIC initialization) and since APIC needs to detect rising edge at 8259, it is essential to follow the 8259 until first interrupt. This is the only occasion 8259 output will be just followed.

DIRECTIONS FOR EASY MIGRATION TO FUTURE INTEGRATED APIC

The following are the software programming directions Intel strongly recommends for easy migration from 82489DX to integrated APIC. The audience to this portion of the document are both hardware designers and firmware developers for APIC based systems. In the following discussions, the APIC BIOS is viewed functionally as two subsections 1) APIC BIOS which are all interrupt vector, priority, interrupt distribution related functions and the remaining portion of BIOS which is referred to part of system BIOS which is responsible for interrupt polarity programming, starting next processor, etc.

Note that the names APIC BIOS and APIC DRIVER are interchangeably used in the following discussion. Different Operating systems refer such functional module differently.

Consideration 1

Question: The logical destination register in future implemented APIC may have only 8 MSBs defined and 82489DX has 32 bits specified. Will this hinder binary level compatibility?

Response: In logical destination (flat addressing mode) 82489DX can go up to 32 CPUs whereas future APIC can go up to 8 CPUs with flat logical addressing mode. ***For binary compatibility, it is strongly recommended that 82489DX software use ONLY 8 MSB of logical destination register.***

Consideration 2

Question: The present day MP systems with external control ports for starting next processor may program those external control ports for starting next processor. APIC DRIVER may use external control ports for starting next processor. In future implementations of APIC, the starting of next processors may use more refined mechanisms which may not use external control ports. Will this introduce compatibility problem?

Response: Again, the starting of next processor is really part of MP system DRIVER and depending of the mechanism used to start next processor it will vary. In future implementation if starting next processor is done using new mechanisms, the starting next processor portion of MP DRIVER will be changed accordingly. Even though this will not result in any change in the APIC DRIVER which deals with interrupt priority, distribution, etc., the corresponding change will be needed in the starting application processors portion of DRIVER.

One possible method of implementing software is using version register. Version register is different in 489DX and future implementations of APIC. Taking care of these differences, such as mechanism for starting next processor, should be possible using version register.

Consideration 3

Question: APIC architecture, by its nature, seems to misinterpret spurious interrupts as genuine interrupts. That is, if an edge triggered interrupt goes inactive before interrupt acknowledge cycle, APIC, instead of giving spurious interrupt vector, gives genuine interrupt vector. Is it true that this is not the case with 8259? If that is the case, drivers which do not check device status registers for servicing the device may work with 8259 but may not work with APIC. Is this a compatibility problem?

Response: No, this is not true. Even with 8259 there is a time window in which a similar thing can happen. For example if interrupt goes inactive just after first INTA cycle but before second INTA cycle 8259 will also signal this spurious interrupt as genuine interrupt. So drivers which do not check device status registers may also fail with 8259.

Our strong recommendation to device drivers is to read device status register before servicing the device. If the device status register indicates that there is no valid source of the interrupt, the service routine should just issue EOI and return. It should not service the device. This should take care of the new drivers that will be written for APIC. To coexist with 8259, the APIC interrupt input connected to 8259 will be programmed for virtual wire mode. In virtual wire mode, the time window of 8259 will apply. So the driver will behave same way as it was behaving with 8259.

Consideration 4

Question: EISA system has active low level polarity. 82489DX itself does not have polarity control register to support this EISA feature. Implementations using external polarity register may implement the polarity register at different address. Will this introduce a problem for achieving the goal of single binary?

Response: 8259 by itself does not have polarity control whereas 8259 when implemented in EISA chipset has the polarity control. Similarly APIC does not have by itself polarity register. When implemented in ESC chipset, it will have polarity control register. So polarity register should be programmed as a part of EISA BIOS and not APIC BIOS. Since system BIOS or EISA BIOS should be able to take charge of changes, if any, to polarity control register. APIC BIOS should not be affected by differences in the address for polarity register.

Consideration 5

Question: 8259 recognizes the interrupt when an edge occurs at the interrupt input pin even if the interrupt was masked. So when the interrupt input is later unmasked, the interrupt is posted to the CPU. 82489DX does not register this edge and if interrupt happens when the interrupt is masked 82489DX just ignores the interrupt. When later it is unmasked there is no further edge and so 82489DX never passes that forgotten edge and that interrupt channel becomes unusable after that.

Response: When the interrupt is masked, logically interrupt controller should ignore whatever happens there. It is strongly recommended that first 82489DX should be unmasked and then the device interrupt should be enabled. By this sequence, software can ensure that always an edge will occur at the APIC input only after the interrupt is unmasked.

Please contact Intel for platform level specification in Multiprocessor system design using APIC.