AUSTRALIA TELESCOPE NATIONAL FACILITY PC ISA BUS EXTERNAL BUS INTERFACE

11-Mar-1999

General Description.

The AT correlator "External bus interface" is designed to interface to an 80x86 mother board's 8/16 bit ISA bus.

The external bus provides addressing for 32K by 16-bit words of memory and six specific purpose 16-bit registers. The external bus consists of a non-multiplexed, 50 pin header, containing...

- 16 data lines (D15..0)
- 15 address lines (A14..0)
- Write/Read control line (-WR)
- A command acknowledge line (-ACK)
- A general purpose strobe line (-STB)
- Six specific purpose strobe lines (-R5STB..-R0STB)
- Eight ground lines.

One "External bus interface" board may optimally interface upto three external buses (channels) at once in one ISA slot. For proper software operation the hardware should be configured before use. This is described in the section Hardware Configuration.

Figure 1 shows a simplified board layout, showing the relative positions of the XILINX chips, the dip switches and the links on the board.

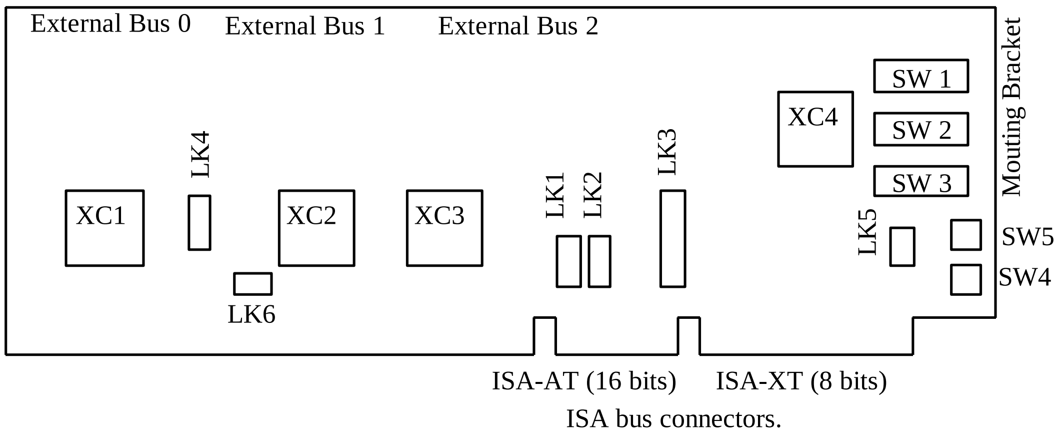


Figure 1 - Simplified board layout, Component Side View.

Hardware Set-up

XILINX Configuration

There are two functions served by the Xilinx chips on the interface.

1. Device selection and contention resolution. This function is wholly carried out by XC4. Which is normally an XC3020PC84-70. This chip must always be present, even if only one external interface is to be used and no device contention is possible.
2. External bus interfacing. This function is provided by XC3, XC2 and XC1 independently of each other. This device is normally an XC3064PC84-70. Each one of these chips handles all of the interfacing with an external bus. They require XC4 for contention resolution and to provide the XC*n*-ENABLE signal.

Configuration	**XC4**	**XC3**	**XC2**	**XC1**
Minimum	4	4	7	7
Partial	4	4	4	7
Maximum	4	4	4	4

Table 1 - Possible Interface Configurations

The interface may be a Minimum, Partial or Maximum configuration as shown in Table 1.

The associated IC's required for each configuration are described in the Hardware Assembly section.

XILINX Configuration PROMS

Outline of XILINX configuration methods...

Using One PROM: The most efficient method of configuring the XILINX chips is to use one single configuration PROM in socket U15. This would be an XC1765S or compatible ATT1765F serial PROM. This would involve having XC4 in MASTER-SERIAL mode, and the rest in SLAVE-SERIAL mode. XC4 would program itself first, then serially pass on data to the other chips, which would receive the data simultaneously. This method would see XC4 configured first, however, it will not enter the programmed state until the PROM has reached it's terminal count.

Using Two PROMs: A second method which is simpler to implement would be to use one PROM for the address decoder chip XC4 in U15, and a second PROM for XC3, XC2, and XC1 in U18. The first PROM would be an XC1736S (or compatible) with the XILINX chip XC4 in MASTER-SERIAL mode. The second PROM would be an XC1765S (or compatible) with the XILINX chip XC3 in MASTER-SERIAL mode, and the rest in SLAVE-SERIAL mode. In this case the two PROMS would operate independently of each other. XC3 would also pass on the configuration data to XC2 and XC1 as it simultaneously configured itself. This method would see all four chips configured at start-up.

Using Four PROMs: An alternate method which is less efficient would be to use an individual PROM for each XILINX chip. This would involve using an XC1736S (or compatible) in U15 and a XC1765S (or compatible) in each of U18, U19 and U20 for each of XC3, XC2 and XC1 respectively. Each XILINX chip would be in MASTER-SERIAL mode and would configure independently of each other upon start-up.

Method 1: using only one PROM...

This method as outlined above is the most efficient method, as it only uses one configuration prom in U15 to configure all 4 XILINX chips. The PROM used would have to be an XC1765S or compatible. One limitation of this method, is that XC3, XC2 and XC1 must have the same configuration file.

Programming the PROM would involve using the PROM utility in the Xact Design Environment. The PROM file for XC4 would be loaded at address 0x0000 in the UP direction. Then the PROM file for XC3, which will also be used to load XC2 and XC1, would then be loaded after the first file, also in the UP direction. The file can now be saved, and the PROM programmed.

As described in the outline above, XC4 must be in MASTER-SERIAL mode and the other chips in SLAVE-SERIAL mode. This involves placing three jumpers in link **LK5**. No jumpers are required in **LK6** as XC3, XC2 and XC1 default to SLAVE-SERIAL mode.

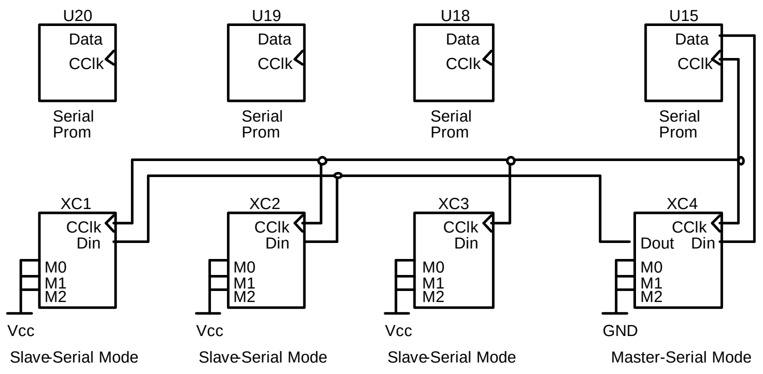


Figure 1 - Shows simplified signal flow from PROM to XILINX chips.

Jumpers need to be installed in link **LK4** to patch the clock and data from XC4. Linking all the jumpers will provide the hookup required for this configuration.

Method 2: using two PROMs...

This method as outlined above is much simpler to implement, as the PROM's only contain one configuration each, they are easier to compile. A PROM would be used in U15 to configure XC4, this would be an XC1736S (or compatible). A separate PROM would be used in U18 to configure XC3, XC2 and XC1 simultaneously. This would be an XC1765S (or compatible). One limitation of this method is that XC3, XC2 and XC1 must have the same configuration file.

Programming the PROMs would involve using the PROM utility in the Xact Design Environment. The PROM file for XC4 would be loaded at address 0x0000 in the UP direction and saved. This would be repeated for XC3 on a new PROM file. The files can now be used to programme each PROM.

As described in the outline above, XC4 and XC3 must be in MASTER-SERIAL mode and the rest in SLAVE-SERIAL mode. This involves placing three jumpers in link **LK5** for XC4. One jumper would be required in **LK6 -** position 3, to switch XC3 into MASTER-SERIAL mode. No jumpers are required for XC2 and XC1 as they default to SLAVE-SERIAL mode.

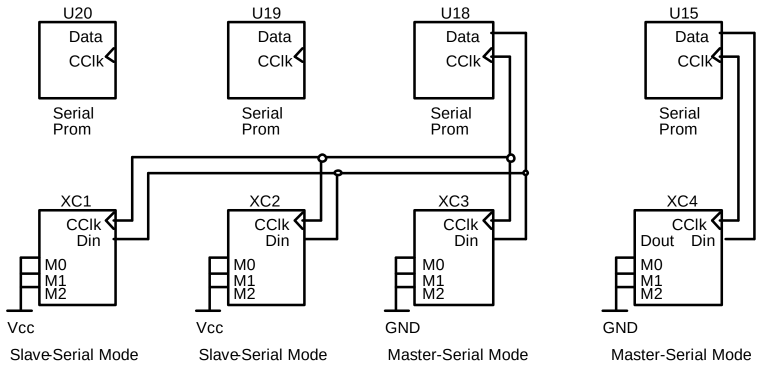


Figure 2 - Shows simplified signal flow from PROMs to XILINX chips.

Jumpers need to be installed in **LK4** to patch the clock and data from XC3. Linking the jumpers at positions 1, 2, 4 and 5 will do this. Links at positions 3 and 6 are left out, as they tie in XC4, which is to be isolated for this configuration.

Method 3 - using 4 PROMs...

This method as outlined above is the most versatile. It uses one PROM for each XILINX chip. This allows every chip to have it's own independent configuration. Unfortunately, this requires the use of four PROMS. The PROM for XC4 would be placed in U15 and would be an XC1736S (or compatible). The PROMs for XC3, XC2 and XC1 would be XC1765S (or compatibles).

Programming the PROMs would involve using the PROM utility in the Xact Design Environment. The PROM files need to be loaded starting at address 0x0000 going UP, and saved. This would be done for all of the PROMs separately. The files are now ready to be programmed into the PROMs.

As described in the outline above, all of the XILINX chips must be in MASTER-SERIAL mode. This involves placing three jumpers in link LK5 for XC4 and jumpers in all three positions of link LK6 for XC3, XC2 and XC1.

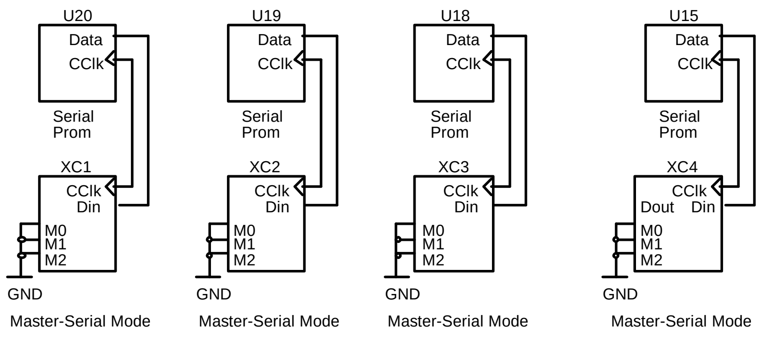


Figure 3 - Shows simplified signal flow from PROMs to XILINX chips.

Jumpers should not be placed in LK4 as each chip is required to configure itself independently.

Device Address Switches

The base addresses for each of the external buses must be selected by setting the 10-way DIP switches near XC4. These switches represent the ISA address lines ISA-A14..5 inclusive. ISA-A15 is always assumed to be "0" by the address decoder chip XC4. This effectively limits the possible addresses to the lower 512KB of the ISA I/O space.

If two or more switches are set to the same address, the device contention logic in XC4 will ensure that only one device will be selected, with priority given in the order XC3, XC2 and finally XC1.

It is advisable, that if more than one external bus is to be run from the same interface, that their relative base addresses be 1KB apart in the I/O space. This is so that minimal I/O space will be used by the interface. This maintains compatibility with any ISA-XT devices that only decode 10 address bits. See references.

A suitable place in the ISA I/O space must be found with at least 32 bytes of I/O free on a 32 byte boundary. This is in case any future changes to the interface will have full access to a 32 byte block starting at the base address.

Interrupt Request Line

The interrupt request line must be installed using a jumper on the interface board. One single jumper must be placed in the link **LK3** marked **IRQ**. Choose the interrupt number required by the system, or by the software. If no interrupt line is selected, no interrupts will reach the platform CPU regardless of the software selection. If the incorrect interrupt line is selected the system may hang.

Note also that all three external buses on the interface share the same IRQ line. Hence the software must poll all three devices to find out which one is interrupting.

DMA Request Line

The DMA request line must be installed by using a jumper on the interface board. One single jumper must be placed in the link **LK1** marked **DMA REQ.** Choose the DMA request line required by the system, or by the software. This must be the same as the DMA acknowledge line. If no DMA request line is selected, no DMA will be possible regardless of the software selection. If the incorrect DMA request line is selected the system may hang.

Note that all three external buses on the interface share the same DMA REQ line. Hence the software must be very careful only to enable one device for DMA at any one time.

DMA Acknowledge Line

The DMA acknowledge line must be installed by using a jumper on the interface board. One single jumper must be placed in the link **LK2** marked **DMA ACK**. Choose the DMA acknowledge line required by the system, or by the software. This must be the same as the DMA request line. If no DMA acknowledge line is selected, no DMA will be possible regardless on the software selection. If the incorrect DMA acknowledge line is selected the system may hang.

Programming

Introduction

The "External bus interface" looks like three sets of registers in the ISA bus' I/O space.

Offset address	Register	Domain
0x14	MODE	Local
0x10	STATUS	Local
0x0E	R5	External
0x0C	R4	External
0x0A	R3	External
0x08	R2	External
0x06	R1	External
0x04	R0	External
0x02	DATA	External
0x00	ADDRESS	Local - Special

Table 1. -Summary of registers and offset addresses.

Table 1 shows the registers available for each channel of the interface.

The offset address should be added to the base address, which is switch selectable on the board. Each channel has it's independent base address.

The name of each register describes it's basic function, and the right column describes the Read and Write capability and the domain of the register.

The registers marked external pass through the interface without modification. The local registers are stored on the board and are not accessible through the external bus.

The register ADDRESS is an exception to the above paragraph. It is stored on the board, however, it is sent to the external bus via the A14..0 dedicated address lines. This register also possesses other special functions relating to DMA which are explained later.

Software Set-up

The options that may be controlled by software are...

- 8 or 16 bit ISA bus cycle instruction modes.
- The CPU interrupt mask.
- External bus timeout delay.
- DMA facilities.

The following local registers contain the configuration data for each external bus on the interface board.

Mode Register

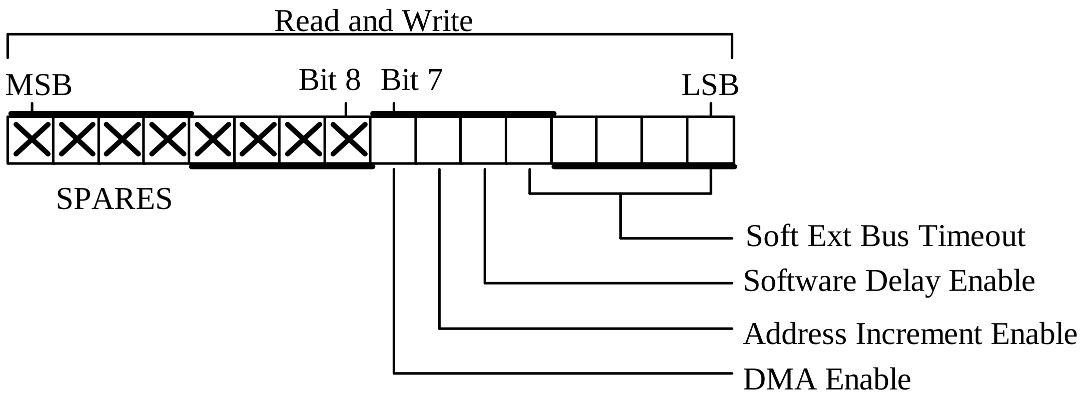


Figure 1 - Mode Register breakdown.

Soft Ext Bus Timeout, is the maximum time in BCLK's that the interface will wait for an -ACK signal from the Block before aborting a transfer. This delay is valid for all cycles that are external to the interface. The default value in this register is zero, however, the software delay must be enabled by using the Software Delay Enable bit in the register. The hardware default for the timeout delay is 15 BCLK's. The maximum achievable delay is 2048 BCLK's, which is the recommended maximum to operate safely without memory brown-outs occurring.

Software Delay Enable activates the Soft External Bus Timeout Delay when set to "1". The default state is "0", which allows the hardware default to become effective upon start-up.

Address Increment Enable sets up an increment on the ADDRESS register after every cycle (Read or Write) to the external bus. The default state is "0", which disables this function.

DMA Enable instructs the interface to proceed with a DMA access cycle. It will assert the DMA-REQ line for the interface board until the appropriate DMA-ACK signal is received from the ISA bus. The DMA-REQ and DMA-ACK signals must be configured on the interface hardware using jumpers, consult the Hardware Set-up section. The direction of the transfer is set up by the DMA controller chip on the ISA platform through software. The DMA process may be configured to access the same external address or an incrementing external address through the use of the Address Increment Enable bit. The default state for DMA Enable is "0", which disables DMA upon start-up.

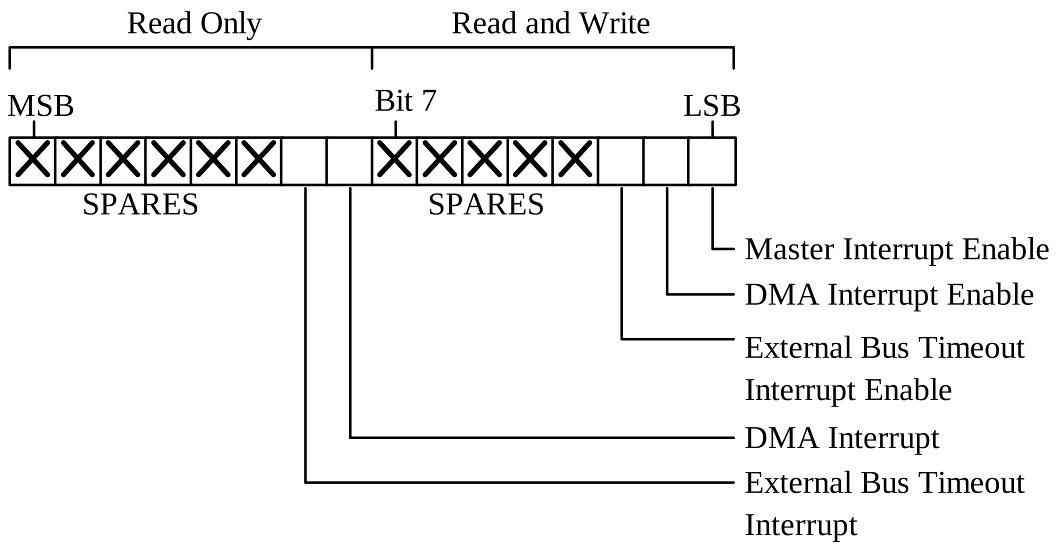


Figure 2 - Status Register breakdown.

The Status Register consists of two parts, as can be seen in Figure 2. The MSB half is a read only byte which contains interrupt information for the CPU. Attempting to write to this byte will only reset its contents. The LSB half is a read and write byte, which contains an interrupt mask. This allows the software to enable/disable specific interrupts, or all at once; using the Master Interrupt Enable bit.

Master Interrupt Enable bit is one bit that enables/disables all other interrupts. This may be done quickly without changing the states of the other individual interrupts. The default state is "0", which disables all interrupts. The specific hardware interrupt that is asserted is determined by the hardware. See Hardware Set-up for more information. If more than one external bus is operating on one board, they all share the same interrupt lines, hence the software must poll the upper byte of this register in conjunction with the lower byte, to see which asserted the interrupt.

DMA Interrupt Enable is the mask bit for the DMA Interrupt for the CPU. This interrupt is asserted when the TC signal is received from the platform CPU. It must be enabled in conjunction with the "Master Interrupt Enable" bit, in order for the software to become aware that a DMA block has been transferred. The default state is "0", which disables this interrupt.

External Bus Timeout Interrupt Enable is the mask bit for the CPU interrupt generated by a Timeout occurring on the external bus. This may be due to either the software selected timeout, or the default timeout. This interrupt indicates that a transfer cannot take place. This bit must be enabled in conjunction with the "Master Interrupt Enable" bit, in order for the interrupt to be generated on the ISA bus. The default state is "0", which disables this interrupt.

DMA Interrupt bit is asserted when the platform controller asserts the TC (Terminal Count) signal after a DMA transfer involving the specific external bus. This is the only indication to the software that a DMA block has been transferred. The actual hardware interrupt generated is determined by the hardware configuration on the interface. See the Hardware Set-up section for more information. The software must poll all the external buses connected to the interface and check both the interrupt mask bits and the interrupt status bits to determine if an interrupt has occurred. All the interrupt bits will be cleared after a read or write to the top half on the Status Register.

External Bus Timeout Interrupt bit is asserted when a timeout has occurred while attempting to read or write to the external bus. This is due to a failure to receive a -ACK signal from the block in good time. The timeout time is determined by either the software delay or the default hardware delay, depending which is enabled at the time. The actual hardware interrupt generated is determined by the hardware configuration on the interface. See the Hardware Set-up section for more information. As with all interrupts, the software must poll all the external buses connected to the interface and check both the interrupt mask bits and the interrupt status bits to determine if an interrupt has occurred. All the interrupt bits will be cleared after a read or write to the top half on the Status Register.

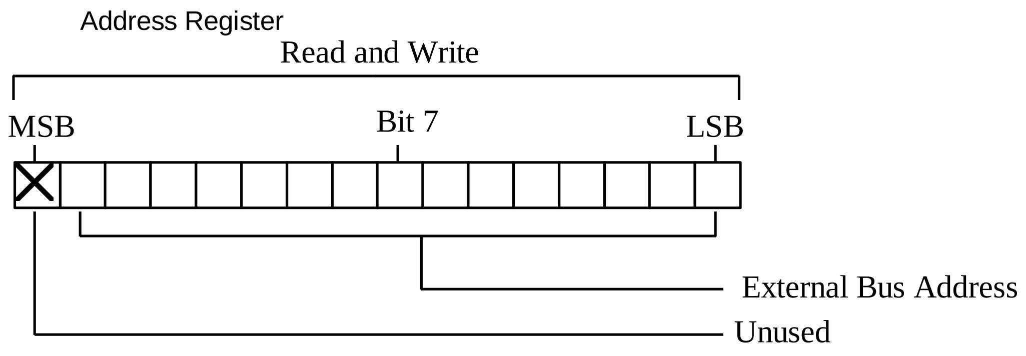


Figure 3 - Address Register breakdown.

The **Address Register** always contains the current address being sent out on the address lines A14..0 on the external bus. It may be read or written to at any time, except during a DMA transfer cycle. During a DMA transfer the "Automatic Address Increment Enable" bit in the Mode register will cause the ADDRESS value in this register to increment by two byte address values, ie: after each word is transferred, the word address will increment by one. It may be checked after a DMA transfer to see how many words were transferred. The default address value upon start-up of this register is "0x0000".

Appendix A - External Bus Description

Signal Descriptions

A14..A0	Address lines.
D15..D0	Data lines.
-WR	Write line. If high when **-STB** goes low indicates that transaction is a read. If low when -STB goes low indicates that the transaction is a write.
-STB	Strobe line. High to low edge indicates valid address on **A15..A0** and valid **-WR** line. For a data write, it also means that the data lines **D15..D0** are also valid.
-ACK	Acknowledge line, High to low transition from the slave means that it as finished with the data placed on the bus by the host.
-R*n*STB	Register *n* strobe line. Same as for the **-STB** line, but indicates a transaction with Register *n*. The address lines are undefined. The data and the **-WR** line have the same meaning as for the **-STB** signal.

External Bus Protocol and Timing. Four types of transactions can be carried out on the external bus, Addressed Read, Addressed Write, Register Read and Register Write. The addressed read and write refer to reading and writing to an addressable location on the external bus using the ADR and DATA register. Register reading and writing refers to reading and writing from one of the six registers on the bus (R0..R5).

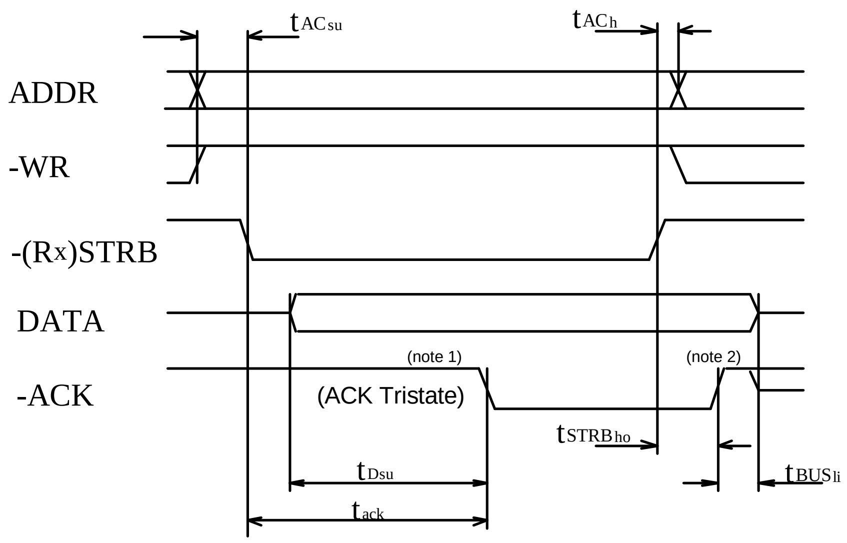
The timing diagram for Addressed Read and Addressed Write appear in Fig(2) and Fig(3) respectively. On a read transaction, the external bus host (the QBUS interface board) places the address on the address lines (A[14..0]) and sets **-WR** High. Some time later the host asserts **-STB**. This is used to indicate that **A[14..0]** and **-WR** are valid. The slave must then place the required data on the data lines (**D[15..0]**) and sets **-ACK** low some time later. In response to this the host deasserts **-STB** and the slave must then deassert **-ACK** and tri-state it and the data lines.

When the host wishes to write to the bus, it sets up the data on **D[15..0]**, address on **A[14..0]** and sets **-WR** low and some time later asserts **-STB** to indicate that valid address and data are on the bus. When the slave has finished with the data, it asserts **-ACK** low. The host then deasserts **-STB**. The slave then deasserts **-ACK**.

Protocol for transactions dealing with the external bus registers is similar to those dealing with the addressable locations on the bus, except that the address lines are no longer necessary, and assume don't care states. Each of the four registers has its own strobe line on the external bus (**-R*n*STB**, where ***n***=0..5). When the slave detects one of the **-R*n*STB** lines active, it performs the read or write on that register. (*Note, the address bus will always reflect the contents of the address register. There is, therefore, no reason why the address lines cannot be used in conjunction with a* ***-RnSTB*** *line for additional functionality*).

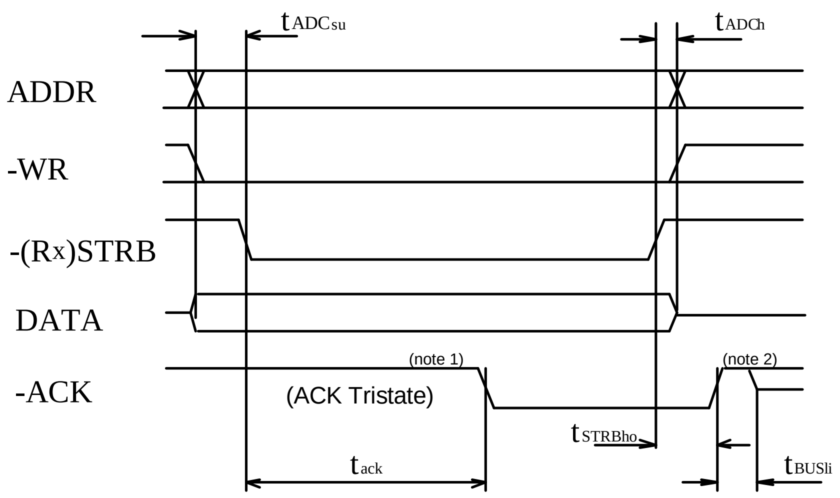
The timing values for the external bus are given in Table(2). There are maximum values on the response times (**-STB** to **-ACK**) so that a timeout mechanism may be employed to detect nonexistent and faulty memory and registers. Such a condition is detected on the QBus by a timeout on the register access.

Timing Diagrams



BCC IF Read Transaction

Fig(2): Read Transaction



BCC IF Write Transaction

Fig(3): Write Transaction.

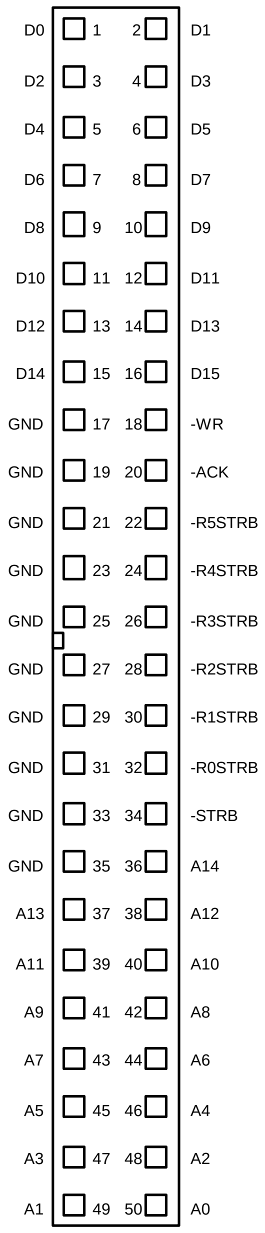
Notes to figs 2 and 3:

1. The ACK line is pulled high by a 4K7 Ω(nom) resistor at the Interface card end The ACK driver may remain tri-stated up until it needs to drive the line low.
2. The ACK line should be driven high before being tri-stated.

Quantity		**Min**	**Max**
$t_{AC_{su}}$	Address/Control setup.	32nS	
$t_{ADC_{su}}$	Address/Data/Control setup.	32nS	
t_{ack}	ACK response.	0nS	2µS
t_{AC_h}	Address/Control hold.		32nS
t_{ADC_h}	Address/Data/Control hold.		32nS
$t_{D_{su}}$	Data setup.	32nS	
$t_{BUS_{li}}$	Bus linger before tri-state.		32nS
$t_{STRB_{ho}}$	Strobe Hold after ACK.	0nS	2µS

Table(2): External Bus timing constraints.

50 pin header



Fig(6): External bus pinouts.

Appendix B - ISA I/O space description

Hex Range	Device
000-01F	DMA controller 1, 8237A-5
020-03F	Interrupt controller 1, 8259A, Master
040-05F	Timer, 8254-2
060-06F	8042 (Keyboard controller)
070-07F	Real time clock, NMI mask
080-09F	DMA page register, 74LS612
0A0-0BF	Interrupt controller 2, 8259A
0C0-0DF	DMA controller 2, 8237A-5
0F0	Clear math coprocessor BUSY
0F1	Reset math coprocessor
0F8-0FF	Math coprocessor
1F0-1F8	Fixed disk
200-207	Game I/O
278-27F	Parallel printer port 2
2F8-2FF	Serial port 2
300-31F	Prototype card
360-36F	Reserved
378-37F	Parallel printer port 1
380-38F	SDLC, Bisynchronous 2
3A0-3AF	Bisynchronous 1
3B0-3BF	Monochrome display & printer adapter
3C0-3CF	Reserved
3D0-3DF	Color/Graphics monitor adaptor
3F0-3F7	Diskette controller
3F8-3FF	Serial port 1

Appendix C - ISA Interrupt Allocations

Level	Function		
NMI	Parity, watchdog timer, arbitration time-out, channel check		
IRQ 0	Timer 0 Output		
IRQ 1	Keyboard interrupt input		
IRQ 2	Interrupt input second controller		
		IRQ 8	Real-time clock
		IRQ 9	Redirct cascade
		IRQ 10	Reserved
		IRQ 11	Reserved
		IRQ 12	Mouse
		IRQ 13	Math coprocessor
		IRQ 14	Fixed disk
		IRQ 15	Reserved
IRQ 3	Serial alternate		
IRQ 4	Serial primary		
IRQ 5	Reserved		
IRQ 6	Diskette		
IRQ 7	Parallel port		

NOTE: IRQ 8 through IRQ 15 are cascaded through IRQ 2