MXIbus
Multisystem Extension Interface Bus Specification

Version 2.0

April 1997 Edition
Part Number 340007B-01

Internet Support

support@natinst.com
E-mail: info@natinst.com
FTP Site: ftp.natinst.com
Web Address: http://www.natinst.com

Bulletin Board Support

BBS United States: (512) 794-5422
BBS United Kingdom: 01635 551422
BBS France: 01 48 65 15 59

Fax-on-Demand Support

(512) 418-1111

Telephone Support (U.S.)

Tel: (512) 795-8248
Fax: (512) 794-5678

International Offices

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, Denmark 45 76 26 00,
Finland 09 527 2321, France 01 48 14 24 24, Germany 089 741 31 30,
Hong Kong 2645 3186, Israel 03 5734815, Italy 02 413091, Japan 03 5472 2970,
Korea 02 596 7456, Mexico 5 520 2635, Netherlands 0348 433466,
Norway 32 84 84 00, Singapore 2265886, Spain 91 640 0085, Sweden 08 730 49 70,
Switzerland 056 200 51 51, Taiwan 02 377 1200, U.K. 01635 523545

National Instruments Corporate Headquarters

6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: (512) 794-0100

Important Information

Warranty

National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

Product and company names listed are trademarks or trade names of their respective companies.

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

Table of Contents

4. Device-Dependent Considerations

Appendix

Glossary

Index

Figures

Tables

Introduction

Section 1

1.1 Scope

The Multisystem eXtension Interface bus (MXIbus) is a multidrop parallel bus architecture designed for high-speed communication between devices. The MXIbus is a general-purpose gateway that you can use to communicate between two or more devices, such as personal computers, workstation computers, VXIbus mainframes, VMEbus-based computers, stand-alone instruments, or modular instruments.

The MXIbus links together multiple devices by mapping together sections of their address spaces. This address map connection makes remote MXIbus devices appear as if they are local memory resources of other MXIbus devices. For example, an IBM PC AT computer with an AT bus-to-MXIbus interface can control multiple VXIbus mainframes equipped with MXIbus-to-VXIbus interfaces. The VXIbus mainframes with their plug-in instrument modules then appear to the PC AT as if they were plugged directly into the backplane of the PC AT. When the PC AT performs a read or write to a memory location that maps to one of the remote instrument modules, the AT-MXI interface translates the PC AT bus cycle into a MXIbus bus cycle to the remote VXI-MXI interface, which further translates the MXIbus cycle into a VXIbus cycle to reach the particular memory location in the VXIbus mainframe.

The MXIbus architecture is a very high-performance link between devices because it maps actual bus cycles on one device to bus cycles on another device. Because it is a multidrop, multimaster architecture with a full 32-bit multiplexed address and data pathway, multiple 8-, 16-, or 32-bit MXIbus devices can dynamically communicate with each other and control each other's resources at very high speeds. Accessing remote devices is straightforward because all MXIbus device memory is written and/or read through memory mapping.

1.2 Objectives

This specification defines the mechanical, electrical, and functional requirements for the MXIbus. The system objectives of the MXIbus specification are as follows:

- To specify the system characteristics that allow communication between devices that are physically located in different mainframes, enclosures, or chassis
- To specify the electrical and mechanical system characteristics required to design devices that will reliably and unambiguously communicate with other devices interfaced to the MXIbus
- To specify the physical layer protocols that define the interaction between devices and the MXIbus
- To define terminology that describes the MXIbus system and the devices that are interfaced to it

A glossary of terms used in this specification is at the back of this document. Section 2 describes the physical characteristics of MXIbus devices and systems. These include the MXIbus cables, connectors, terminators, and electrical specifications. There are 71 active MXIbus signals that are used to transfer data between MXIbus devices, arbitrate between contending devices, and perform interrupt, trigger, and error-detection functions. Some of these signals have certain timing relationships that ensure signal integrity, described in Section 3. Section 4 contains device-specific information to consider when designing a MXIbus interface.

1.3 Introduction to MXI-2

MXI-2 is the second generation of the MXIbus. MXI-2 expands the number of signals on a standard MXI cable by including all VME interrupts, VXI triggers, CLK10, and all of the VMEbus utility bus signals. Because MXI-2 incorporates all of these signals into a single connector, the triggers, interrupts, and utility signals can extend not only between VXI/VME mainframes, but also to external devices/computers. Thus in a MXI-2 system, CPU interface boards can perform as if they were plugged directly into the VXI/VME backplane.

In addition, MXI-2 boosts data throughput performance past previous-generation MXIbus products by defining new high-performance protocols. MXI-2 is a superset of MXI. All accesses that MXIbus devices

initiate will work with MXI-2 devices. However, MXI-2 defines synchronous MXI block data transfers that surpass previous block data throughput benchmarks. The new synchronous MXI block protocol increases MXI-2 throughput to a maximum of 33 MB/s between two MXI-2 devices.

1.4 Conventions Used in This Specification

The following conventions are used in this specification:

< >	Angle brackets containing numbers separated by an ellipsis represent a range of values associated with a bit or signal name (for example, BDIO<3...0>).
bold italic	Bold italic text denotes a note, caution, or warning.
italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept.
signal names	Signal names appear in all uppercase letters.
numbers	All numbers are in decimal unless otherwise specified in a suffix to the number. An *-h* after a number indicates that the number is in hexadecimal format. A *-b* after a number indicates that the number is in binary format.

Physical Characteristics

Section 2

This section describes the physical characteristics of MXIbus devices and systems, and the cables, connectors, connector signal assignments, cable termination, and electrical specifications needed to implement the MXIbus.

2.1 System Description

A MXIbus system consists of a series of connections between devices via shielded cables. Multiple MXIbus devices are connected together in a linear fashion using cables with common signals. Star configurations are not allowed.

Both ends of the MXIbus are terminated. Terminators may be either stand-alone external devices or embedded inside a device. Embedded terminators must be removable or have the ability to be disabled. All signals except GIN* (grant in), GOUT* (grant out), MXISC* (MXIbus System Controller), ENDDEV (MXIbus End Device), AUXPWR (auxiliary power), and TERMPWR (termination power) are common (that is, bused) between MXIbus devices. Figure 2-1 illustrates an example MXIbus system.

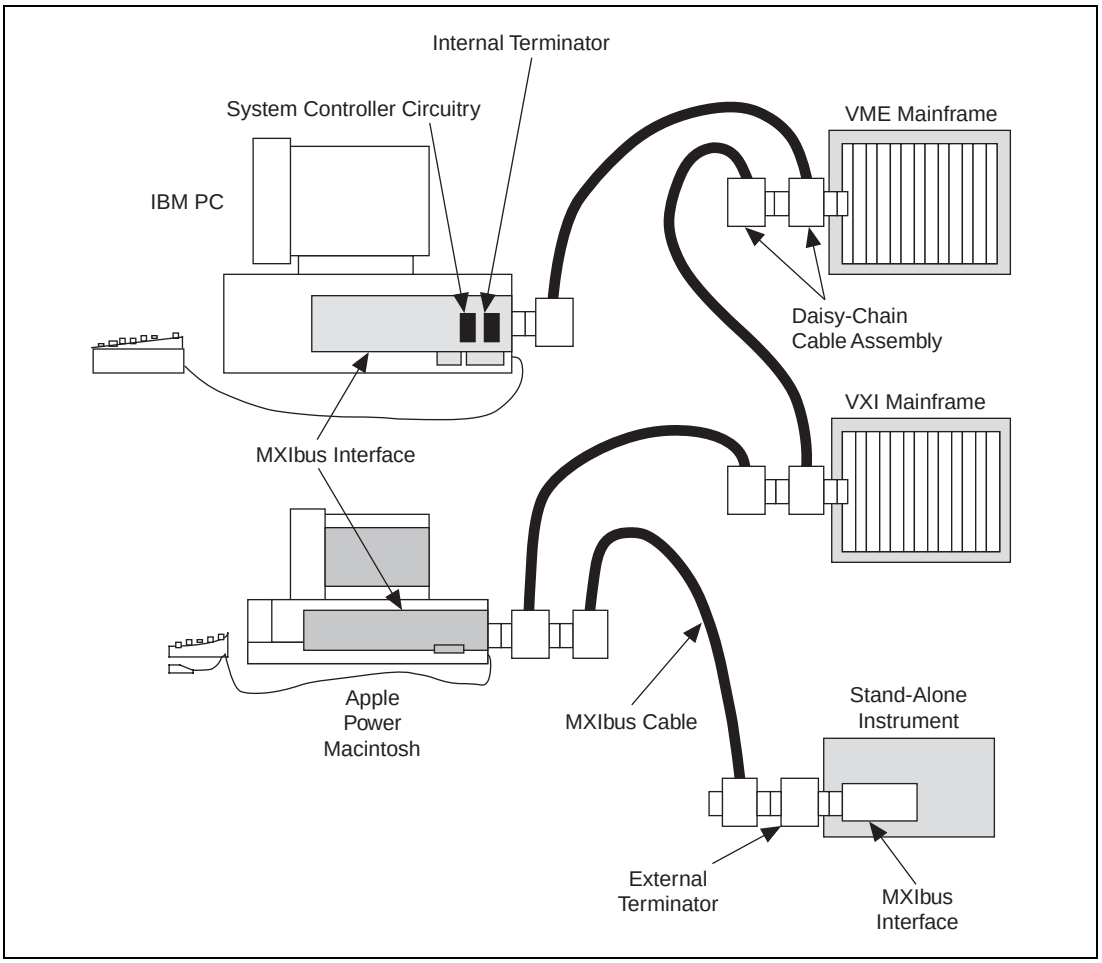


Figure 2-1. MXIbus System Connection Example

The first MXIbus device in the daisy-chain is referred to as the *MXIbus System Controller*. The MXIbus System Controller contains the MXIbus arbiter functional module, the MXIbus priority-select cycle daisy-chain driver, and the MXIbus timeout monitor functional module. Implementation of MXIbus System Controller functionality is optional for a MXIbus device. Each MXIbus device that contains MXIbus System Controller functionality must have a method for defeating the System Controller circuitry. There must be one and only one device in a MXIbus system that functions as the MXIbus System Controller.

2.2 Device Connector

MXIbus specifies the use of shielded, 144-position, high-density connectors for MXIbus devices. MXIbus device connectors consist of four rows of receptacle contacts with adjacent contacts 1.27 mm (0.05 in.) apart, and are physically keyed to ensure proper connection. Device connectors are shielded and connected to the unit's chassis ground to help reduce electromagnetic radiation interference (EMI) and to provide electrostatic discharge (ESD) protection. The metal shell and retaining screw sockets of the MXIbus device connector must make electrical contact with the unit's chassis ground. The device connector has threaded socket inserts that accept the locking screws of the cable assembly connector.

The MXIbus signals must be assigned to the device connector as shown in Figure 2-2. The signal descriptions and timing relationships are given in Section 3.

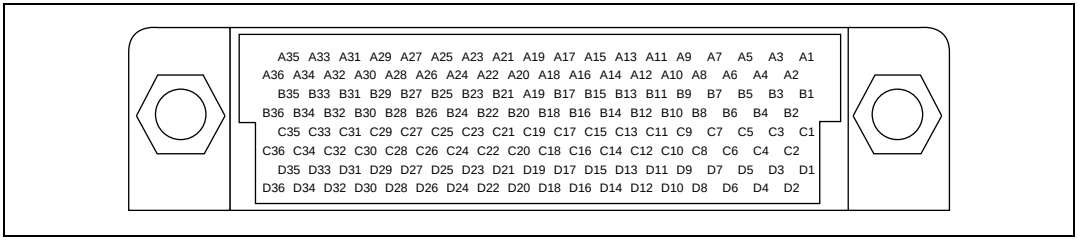


Figure 2-2. Device Connector Pin (Front View)

Table 2-1. Device Connector Pin Assignments

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A1	AD(31)*	B1	AD(14)*	C1	AM(4)*	D1	BUSY*
A2	GND	B2	GND	C2	GND	D2	GND
A3	AD(30)*	B3	AD(13)*	C3	AM(3)*	D3	IRQ(1)*
A4	GND	B4	GND	C4	GND	D4	GND
A5	AD(29)*	B5	AD(12)*	C5	AM(2)*	D5	IRQ(2)*
A6	GND	B6	GND	C6	GND	D6	GND
A7	AD(28)*	B7	AD(11)*	C7	AM(1)*	D7	IRQ(3)*
A8	GND	B8	GND	C8	GND	D8	GND
A9	AD(27)*	B9	AD(10)*	C9	AM(0)*	D9	IRQ(4)*

(Continues)

Table 2-1. Device Connector Pin Assignments (Continued)

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A10	GND	B10	GND	C10	GND	D10	GND
A11	AD(26)*	B11	AD(9)*	C11	WR*	D11	IRQ(5)*
A12	GND	B12	GND	C12	GND	D12	GND
A13	AD(25)*	B13	AD(8)*	C13	SIZE*	D13	IRQ(6)*
A14	GND	B14	GND	C14	GND	D14	GND
A15	AD(24)*	B15	AD(7)*	C15	BTODIS*	D15	IRQ(7)*
A16	GND	B16	GND	C16	GND	D16	GND
A17	AD(23)*	B17	AD(6)*	C17	ACFAIL*	D17	TRG(0)+
A18	GND	B18	GND	C18	GND	D18	TRG(0)-
A19	AD(22)*	B19	AD(5)*	C19	SYSRESET*	D19	TRG(1)+
A20	GND	B20	GND	C20	GND	D20	TRG(1)-
A21	AD(21)*	B21	AD(4)*	C21	SYSFAIL*	D21	TRG(2)+
A22	GND	B22	GND	C22	GND	D22	TRG(2)-
A23	AD(20)*	B23	AD(3)*	C23	BERR*	D23	TRG(3)+
A24	GND	B24	GND	C24	GND	D24	TRG(3)-
A25	AD(19)*	B25	AD(2)*	C25	DTACK*	D25	TRG(4)+
A26	GND	B26	GND	C26	GND	D26	TRG(4)-
A27	AD(18)*	B27	AD(1)*	C27	DS*	D27	TRG(5)+
A28	GND	B28	GND	C28	GND	D28	TRG(5)-
A29	AD(17)*	B29	AD(0)*	C29	AS*	D29	TRG(6)+
A30	GND	B30	GND	C30	GND	D30	TRG(6)-
A31	AD(16)*	B31	CONVERT*	C31	BREQ*	D31	TRG(7)+
A32	GND	B32	GND	C32	GND	D32	TRG(7)-
A33	AD(15)*	B33	PAR*	C33	GIN*	D33	CLK10+
A34	GND	B34	GND	C34	GND	D34	CLK10-
A35	AUXPWR	B35	TERMPWR	C35	GOUT*	D35	MXISC*
A36	AUXPWR	B36	TERMPWR	C36	GND	D36	ENDDEV

2.3 Cable Assemblies

Cable assemblies connect MXIbus devices. MXIbus cable assemblies may be point-to-point, in which two devices are connected, or can be multidrop, whereby two or more devices are connected. Using multidrop cable assemblies, new devices are added to the system in a linear

configuration using the cable's back-to-back connector arrangement, which is similar to the scheme used by the IEEE-488 (GPIB). One cable end contains a single male plug, and the other cable end contains both a male plug and a female receptacle. A diagram of a multidrop MXIbus cable assembly is shown in Figure 2-3.

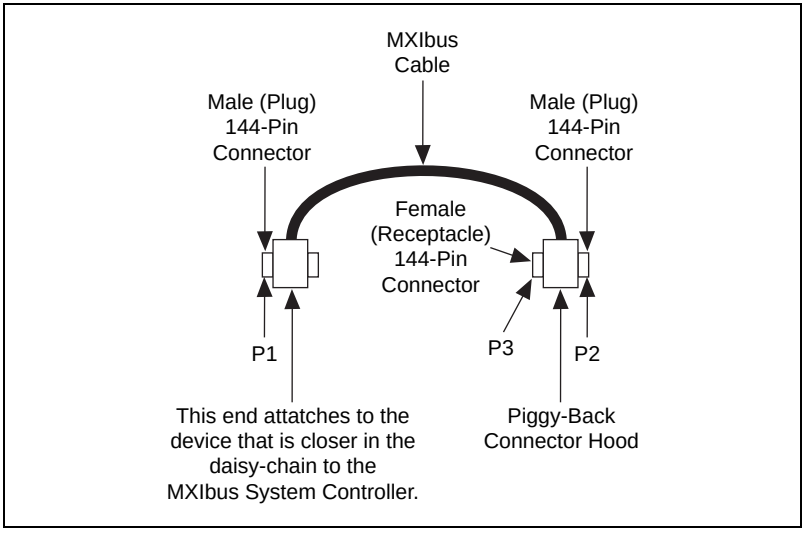


Figure 2-3. Multidrop Cable Assembly

Point-to-point cable assemblies have only two male cable connectors, one on each cable end. All of the signal wires in the point-to-point cable assembly are bused point-to-point except for MXISC*, ENDDEV, AUXPWR, TERMPWR, GIN*, and GOUT*. MXISC* and ENDDEV are discussed later in this section. AUXPWR and TERMPWR are available only on the device connector; there are no connections in the cable. The GIN* and GOUT* signals are wired as follows:

Signal Name	Connected To
P1 GIN*	No connection
P1 GOUT*	P2 GIN*
P2 GOUT*	No connection

All of the signal wires in the multidrop cable assembly are bused between all three connectors except for MXISC*, ENDDEV, AUXPWR, TERMPWR , GIN*, and GOUT*. MXISC* and ENDDEV are discussed later in this section. AUXPWR and TERMPWR are bused between P2

and P3, so that external terminators and other devices that make use of these power sources can be placed either on P3 or between P2 and the MXIbus device. P1 of the cable has no connection to AUXPWR and TERMPWR. The GIN* and GOUT* signals are wired as follows:

Signal Name	Connected To
P1 GIN*	No connection
P1 GOUT*	P2 GIN*
P2 GOUT*	P3 GOUT*
P3 GIN*	No connection

Because MXIbus cables must properly propagate the GIN*-to-GOUT* daisy-chain signals, the cables are naturally polarized and must be properly connected for the system to function. The P1 connector end of the cable assembly must be attached to the device that is closer in the daisy-chain to the MXIbus System Controller. A point-to-point cable can be used as the last cable section in a MXIbus system since P3 of a multidrop cable would not be used in this position.

2.3.1 System Configuration

The two MXIbus signals MXISC* and ENDDEV are used to aid in system configuration. The cable and connector configurations determine the logic level of these signals. A device that receives MXISC* low will automatically assume MXIbus System Controller functions. It will also terminate the MXIbus. A device that receives ENDDEV high recognizes that it is the last device in the MXIbus daisy chain and will terminate the MXIbus.

Figure 2-4 shows the mechanism for implementing these signals. The single-ended connector shown should be the end of the cable electrically closest to the MXIbus System Controller (the upstream end). This is always true for a MXI cable with a single-ended connector on one end and a double-ended connector on the other end. A MXI cable with single connectors on each end is polarized, meaning that the upstream end, while physically similar to the downstream end, is not electrically the same and must be marked so that a user will install the cable with the upstream end closest to the MXIbus System Controller. The double-ended connector is always on the downstream end of the cable (farthest from the MXI System Controller).

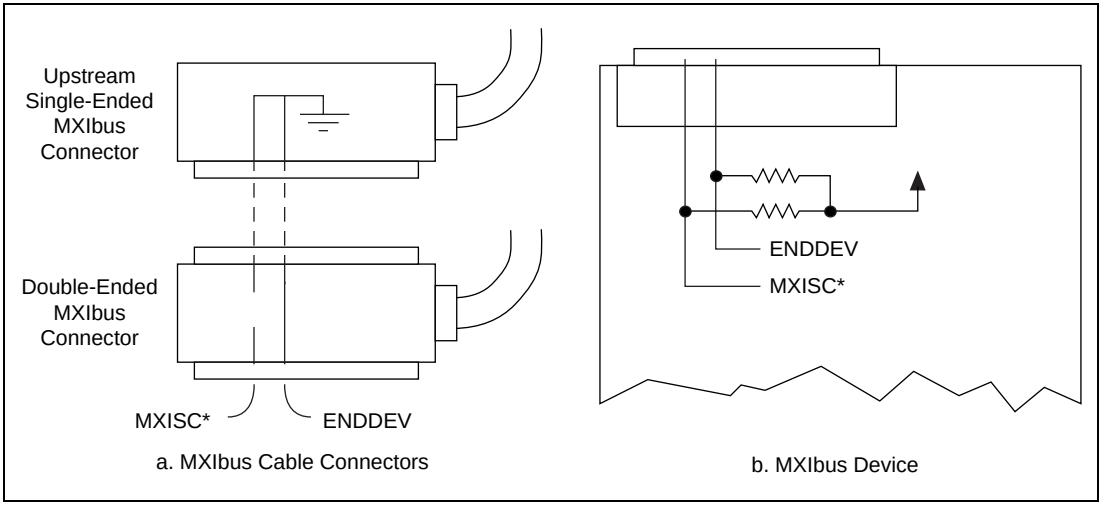


Figure 2-4. MXIbus System Configuration Scheme

Every MXIbus device that uses the MXISC* and/or ENDDEV signals must have pull-up resistors as shown in Figure 2-4.

The upstream single-ended connector pulls both MXISC* and ENDDEV low. Because only a single-ended connector can connect to the MXIbus System Controller, the MXIbus System Controller will receive both signals low and thus terminate the MXIbus and assume System Controller functions.

The double-ended connector passes ENDDEV through from one side of the connector to the other. MXISC* is not passed through the connector (neither signal is bused on the cable). A device in the middle of the MXIbus daisy chain has a dual-ended connector attached to it with the upstream end of a cable attached to the back of the dual-ended connector. That device will receive ENDDEV low and MXISC* high, so it will neither terminate the MXIbus nor assume System Controller functions.

The last device in the MXIbus daisy chain will have either a double-ended connector or a downstream single-ended connector attached to it. Neither the downstream single-ended connector nor the double-ended connector pull MXISC* and ENDDEV low. The downstream single-ended connector does not need to connect to the two signals at all. Because the end device, by definition, has only a single cable attached to it (no piggy-backing of cables as with a middle device), no connector can be attached that will pull MXISC* and ENDDEV low. The device will

receive MXISC* and ENDDEV both high and then terminate the MXIbus. It will not assume MXI System Controller functions.

Table 2-2 describes the function of a MXIbus device given the state of the MXISC* and ENDDEV signals at its MXIbus connector.

Table 2-2. System Configuration Truth Table

MXISC*	ENDDEV	MXIbus System Controller?	Terminate MXI?
Low	X	Yes	Yes
High	Low	No	No
High	High	No	Yes

2.3.2 Cable

Raw MXIbus cable requires 69 twisted wire pairs. An ideal impedance match with cable terminators implies a cable characteristic impedance of 120 Ω (single-ended).

Total cable length in a MXIbus system is limited to 20 m. Individual cable length between two MXIbus devices can be any length between 1 and 20 m.

2.3.3 Cable Connector

MXIbus cable connectors must be shielded, 144-position, high-density, plug and receptacle connectors. These connectors mate with the MXIbus device connectors and, optionally, with additional MXIbus cable connectors. The cable shield must be connected to the metal shell and retaining screws of all cable connectors. The metal shell and retaining screws of the cable connectors must provide electrical contact with the device's chassis ground through the device's mating connector shell and retaining screw sockets. Each cable connector must be fitted with a pair of captive locking screws. A retaining ring, or equivalent, must be used to retain the lock screw as a captive element.

All signal wires in the MXIbus cable must be connected to the proper pins of each connector in the cable assembly. The ground wire of each twisted pair must be connected to the ground pin adjacent to the corresponding signal pin.

2.4 Termination

Termination networks must be positioned at the first and last MXIbus devices in the daisy-chain to minimize reflections due to impedance discontinuities at the ends of the cables, and also to bias the signal lines to a false state when they are not driven true. Each signal line must be terminated at both ends of the daisy-chain. The differential signals CLK10± and TRG<7..0>± must be terminated differently than the single-ended MXIbus signals.

Example termination circuits for the single-ended signals are shown in Figure 2-5. The open-circuit termination voltage must be +3.4 V ± 5%, and the impedance of the termination must be 120 Ω ± 5%.

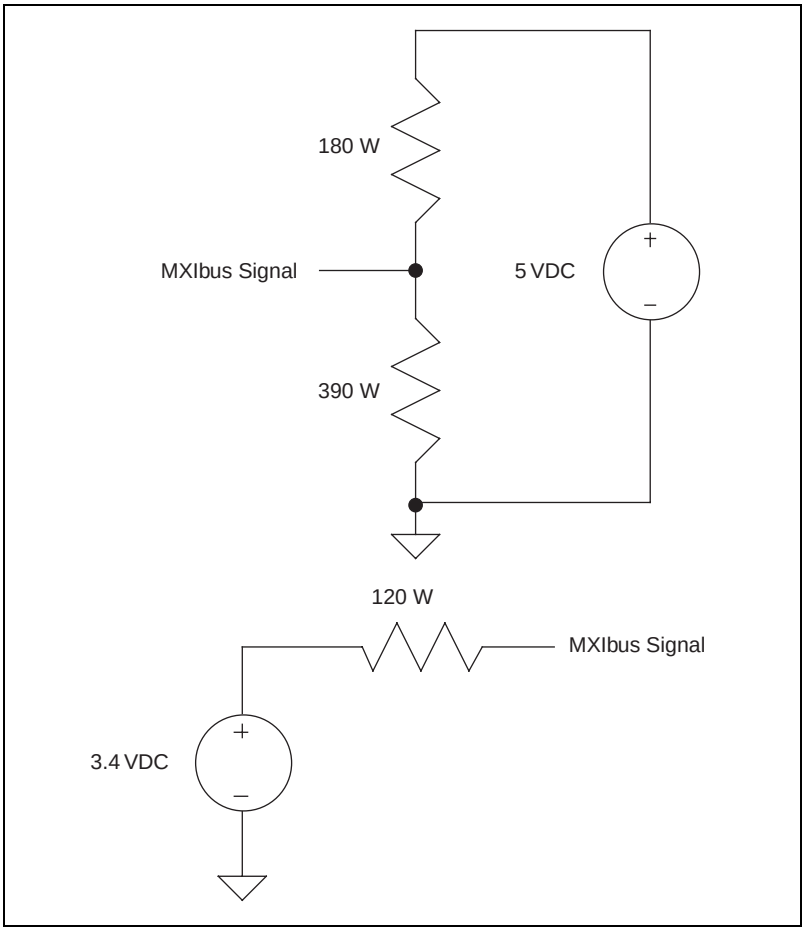


Figure 2-5. Single-Ended Signal Termination Network Examples

An example termination circuit for the differential signals is shown in Figure 2-6. The open-circuit termination voltage for the TRG<7..0>+ signals must be 0 V ± 5%, and the impedance of the termination must be 120 Ω ± 5%. The open-circuit termination voltage for the TRG<7..0>- signals must be +3.4 V ± 5%, and the impedance of the termination must be 120 Ω ± 5%. This termination scheme biases the trigger signals so that they are in the false state when not driven. The CLK10± signal does not need to be biased to a false state when not driven, so the example shows a simpler termination scheme for it.

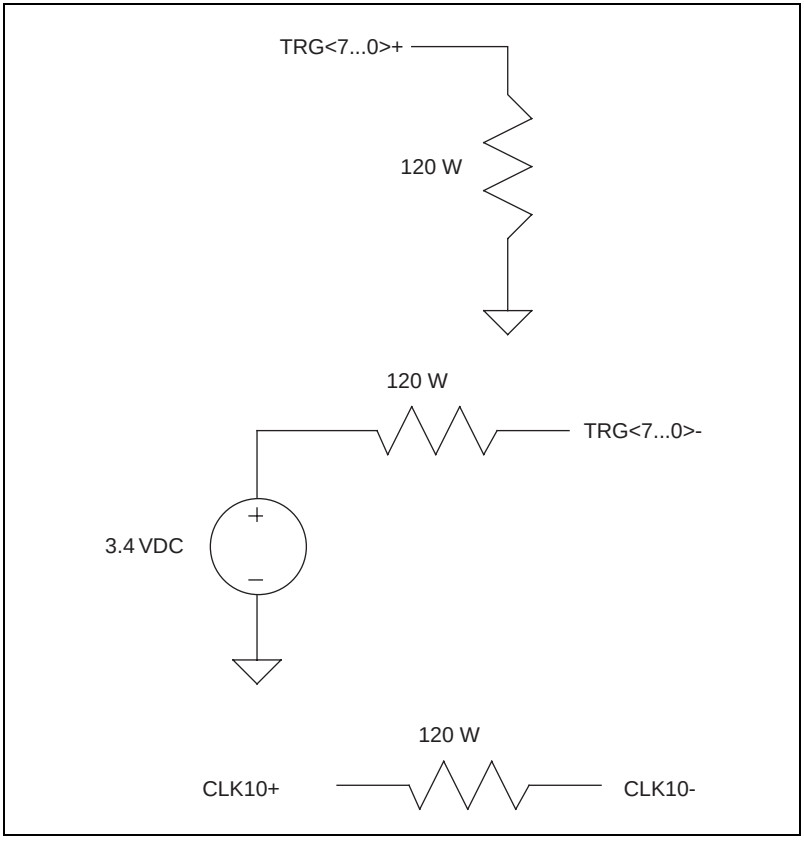


Figure 2-6. Differential Signal Termination Network Examples

MXIbus termination may be provided using terminators that are external to the MXIbus devices in the system. MXIbus devices may also provide for optional internal termination. In addition, a MXIbus device can follow the automatic end-device detection scheme described earlier in this section to determine whether its internal termination circuit should be

enabled. In any event, MXIbus terminators must be located within the maximum stub length dimension given in Section 2.5.

External terminators may provide their own power source, or they may use the termination power supplied by the TERMPWR pin of the device connector. TERMPWR support is optional for a device, but if supported, TERMPWR must provide 3.4 VDC ± 5% at a minimum of 2.5 A. It is recommended that MXIbus devices supply TERMPWR to the connector through protection circuitry that prevents the backflow of power to the MXIbus device, and excessive current draw from the TERMPWR pins.

The termination circuitry for the GIN* and GOUT* signals requires special attention because they are true daisy-chain signals, connected in a point-to-point fashion from device to device. The GIN* signal can be received by a device, processed, and, optionally, retransmitted on the device's GOUT* signal or, in the case of a MXIbus device that does not use the GIN*-to-GOUT* daisy-chain, the GIN* input can be directly connected to the GOUT* output with no logic connection. If the GIN* and GOUT* signals are connected to active logic on a MXIbus device, they must be terminated on that device, regardless of its position within the MXIbus daisy-chain. If the GIN* and GOUT* signals are connected together on a device, the connection must *not* be terminated on that device unless, of course, that device is at one of the ends of the MXIbus daisy-chain.

2.5 Electrical Description

The relationship between the logical states and the electrical voltage level present on the MXIbus signal lines is shown in Table 2-3.

Table 2-3. Logical/Electrical Signal Levels

Logical State	Electrical Level
0 False state	Corresponds to ≥ +2.0 V Called high level (H)
1 True state	Corresponds to ≤ +0.8 V Called low level (L)

In general, the rise and fall times of signals on a bus are the result of a complex set of interactions involving the impedance of the cables, the termination network, the total length of the signal lines, the source impedance of the drivers, and the capacitive loading of the signal lines.

The MXIbus specification requires that all MXIbus devices use specific transceivers to guarantee reliable interoperability.

All MXIbus signals except MXISC*, ENDDEV, and the differential signals CLK10± and TRG<7..0>± must be driven and received by trapezoidal transceivers of a type equivalent to the National Semiconductor DS3662 or DS3862. The differential signals must be driven and received by differential transceivers of a type equivalent to the Texas Instruments SN75ALS176A. MXISC* and ENDDEV can be received with any appropriate receiver for the MXIbus device. Each MXIbus signal must be subjected to no more than one transceiver load per MXIbus device. No more than eight devices (that is, eight transceiver loads) may be connected to the MXIbus at any one time.

Trapezoidal transceivers are used to reduce noise and crosstalk in the MXIbus transmission system. These transceivers have open-collector drivers that generate precise trapezoidal waveforms with typical rise and fall times of approximately 9 ns. The trapezoidal shape, due to the constant rise and fall times, reduces noise coupling (crosstalk) to adjacent lines. The receivers use a lowpass filter to remove noise in conjunction with a high-speed comparator, which recognizes the trapezoidal-shaped signal from noise.

Stub lengths of no more than 4 in. must be allowed off the mainline interconnection within any connected equipment to minimize reflections due to impedance discontinuities. This dimension does not include the length of the circuit path through the device connector.

Signals and Timing

Section **3**

The MXIbus defines 80 active signals, 60 ground lines, two lines for termination power, and two lines for auxiliary power. Table 3-1 shows the signal groupings for the MXIbus. See Section 2 for the MXIbus connector pin assignment.

Table 3-1. Signal Groupings

Category	Description	Signal Name	Lines
Address/Data	Address/Data	AD<31..00>*	32
	Parity	PAR*	1
	Address Modifier	AM<4..0>*	5
	Convert	CONVERT*	1
	Transfer Size	SIZE*	1
	Read/Write	WR*	1
	Address Strobe	AS*	1
	Data Strobe	DS*	1
	Data Transfer Acknowledge	DTACK*	1
	Bus Error	BERR*	1
Arbitration	Bus Busy	BUSY*	1
	Bus Request	BREQ*	1
	Daisy-Chain Grant In	GIN*	1
	Daisy-Chain Grant Out	GOUT*	1
Interrupt	Interrupt Request	IRQ<7..1>*	7
Timing	Clock	CLK10±	2
	Trigger	TRG<7..0>±	16
Utility	System Reset	SYSRESET*	1
	System Fail	SYSFAIL*	1
	AC Fail	ACFAIL*	1
	Bus Timeout Disable	BTODIS*	1
	MXIbus System Controller	MXISC*	1
	MXIbus End Device	ENDDEV	1

Table 3-1. Signal Groupings (Continued)

Category	Description	Signal Name	Lines
Power	Ground Terminator Power Auxiliary Power	GND TERMPWR AUXPWR	60 2 2

When two devices communicate on the MXIbus, one acts as the master and the other acts as the slave. The master originates an operation and the slave terminates the operation. MXIbus devices are able to assume both roles, although a MXIbus device may have master-only capabilities or slave-only capabilities.

3.1 Address/Data

The MXIbus provides support for 32 multiplexed address/data lines (AD<31..00>*). MXIbus devices may use all or only a subset of the available address/data lines to transfer address and data information between devices; however, it is always the responsibility of the MXIbus master to ensure that the address and data width of the transfer matches the capabilities of the slave device. If a transfer is attempted that the addressed slave cannot complete because of data width incompatibilities, the slave may ignore the transfer, or it may terminate the transfer with a bus error acknowledge.

All MXIbus devices shall have even parity generation and detection circuitry. MXIbus devices generate and source parity along with the address and data portions of all MXIbus transfers. The parity signal (PAR*) shall be in such a state so that, when considered together with the AD<31..00>* signal lines, there will be an even number of true (low-level) states.

MXIbus masters and slaves shall generate and check parity by considering the condition of *all* 32 AD<31..00>* lines. This simplifies the implementation of devices with 32-bit MXIbus interfaces because the parity generation and detection circuitry is the same without regard to the size of the address broadcast or data transfer.

Notice that, although masters and slaves are not required to drive and/or monitor the AD<31..00>* lines that do not carry valid addressing/data information, a device is permitted to drive the unused lines to simplify the hardware implementation. If unused lines are driven, they shall be driven in a manner consistent with the timing requirements of the lines

that carry valid addressing/data information. If unused lines are not driven, parity compatibility will be maintained considering that all undriven data lines will float to a high level (logical 0) because of the terminators at the ends of MXIbus.

Although it is possible to implement devices with 16-bit or 24-bit MXIbus interfaces, the system integrator must take care to ensure that both the master and slave are compatible with the desired transfer type. For example, a slave device with a 16-bit MXIbus interface must ensure that a 32-bit master generates a parity level that is consistent with the 16 low-order address/data lines. This is assured if the master either always asserts an even number of the unused address/data lines or does not drive any of the unused lines during the transfer, in which case the lines will float to an inactive state. In either case, the parity level will not be affected because an even number of the unused address/data lines will always be either asserted or unasserted. A data read from a slave device with a 16-bit MXIbus interface will always produce the correct parity because the unused address/data lines will float to their inactive state.

3.1.1 Address Broadcast

All MXIbus transfers are initiated by an address broadcast cycle in which the current MXIbus master broadcasts the necessary addressing information for the upcoming data cycle to the MXIbus slave(s). This information is provided by the address lines AD<31..00>*, the address modifier lines (AM<4..0>*), and the convert (CONVERT*), transfer size (SIZE*), read/write (WR*), and PAR* signals. The MXIbus slaves latch and/or decode these lines when the address strobe (AS*) signal becomes asserted to determine how to respond to the following data cycle(s). Only one slave shall respond at a time to each data cycle.

3.1.1.1 Address Lines

The MXIbus provides 32 address lines (AD<31..00>*). These lines are multiplexed with the MXIbus data lines and are latched and/or decoded by the MXIbus slaves on the assertion edge of AS*. MXIbus masters drive valid address information onto 32, 24, or 16 of the address lines for any given cycle, depending on the address modifier code (see Section 3.1.1.2). Likewise, slaves monitor and decode 32, 24, or 16 of the address lines, depending on the address modifier code. Devices that support 32-bit addresses are called A32 devices. Devices that support 24-bit addresses are called A24 devices, and use address lines AD<23..00>*. Devices that support 16-bit addresses are called A16 devices, and use address lines AD<15..00>*. A24 and A16 devices are

not required to monitor, decode, or drive the unused address lines. MXIbus devices can support one or more of the different address spaces.

The MXIbus master device generates the PAR* signal during the address phase of all transfers. A MXIbus slave that detects invalid parity during any address phase should ignore the transfer and let the bus timeout unit terminate the transfer.

3.1.1.2 Address Modifier Lines

In addition to the 32 AD<31..00>* signal lines, the MXIbus also has five address modifier lines (AM<4..0>*). These lines are similar in function and meaning to the address modifier lines specified in the IEEE-1014 (or VMEbus) specification, and are used by the MXIbus master to pass additional information (that is, the type of bus cycle being performed) to the slave during a transfer. The address modifier codes are shown in Table 3-2. The *Value* column represents the logical value of the address modifier lines in hex rather than the signal levels.

MXIbus slaves shall not respond to reserved address modifier codes. Reserved address modifier codes are for future enhancements. A MXIbus slave may either ignore or terminate (with a BERR* acknowledge signal) any transfers within its address space that contain nonreserved address modifier codes that the MXIbus slave does not support.

Table 3-2. Address Modifier Codes

Address Modifier Line					Value	Function
AM4*	AM3*	AM2*	AM1*	AM0*	(hex)	
L	L	L	L	L	1F	A24, supervisory, block transfer
L	L	L	L	H	1E	A24, supervisory, program access
L	L	L	H	L	1D	A24, supervisory, data access
L	L	L	H	H	1C	A24, supervisory, D64 transfer
L	L	H	L	L	1B	A24, nonprivileged, block transfer
L	L	H	L	H	1A	A24, nonprivileged, program access
L	L	H	H	L	19	A24, nonprivileged, data access
L	L	H	H	H	18	A24, nonprivileged, D64 transfer
L	H	L	L	L	17	A24, synchronous-burst transfer
L	H	L	L	H	16	Reserved
L	H	L	H	L	15	A16, supervisory access
L	H	L	H	H	14	A32, synchronous-burst transfer
L	H	H	L	L	13	Reserved
L	H	H	L	H	12	Priority-selection cycle
L	H	H	H	L	11	A16, nonprivileged access
L	H	H	H	H	10	Reserved
H	L	L	L	L	0F	User-defined
H	L	L	L	H	0E	User-defined
H	L	L	H	L	0D	User-defined
H	L	L	H	H	0C	User-defined
H	L	H	L	L	0B	User-defined
H	L	H	L	H	0A	User-defined
H	L	H	H	L	09	User-defined
H	L	H	H	H	08	User-defined
H	H	L	L	L	07	A32, supervisory, block transfer
H	H	L	L	H	06	A32, supervisory, program access
H	H	L	H	L	05	A32, supervisory, data access
H	H	L	H	H	04	A32, supervisory, D64 transfer
H	H	H	L	L	03	A32, nonprivileged, block transfer
H	H	H	L	H	02	A32, nonprivileged, program access
H	H	H	H	L	01	A32, nonprivileged, data access
H	H	H	H	H	00	A32, nonprivileged, D64 transfer

Block transfer address modifier codes indicate that MXIbus block-mode data is being transferred. For more information about block transfers, refer to Section 3.1.2.2.

D64 transfer address modifier codes indicate that 64-bit data is being transferred over the MXIbus. Since the MXIbus is a 32-bit bus, two 32-bit data transfers are used for each 64-bit data to be transferred. The signaling and timing for D64 transfers over the MXIbus is identical to that of 32-bit block cycles described in Section 3.1.2.2, but with two restrictions. The first restriction is that the block must begin on an 8-byte boundary (address lines 2, 1, and 0 all logical 0). The other restriction is that an even number of 32-bit words must be transferred because each pair of 32-bit words gets combined into a single 64-bit data word at the destination.

Synchronous-burst address modifier codes indicate that MXIbus synchronous-burst data is being transferred. For more information about synchronous-burst transfers, refer to Section 3.1.2.3.

A MXIbus master can use the priority-selection cycle address modifier code (a logical value of AM<4..0>* = 12 hex) to qualify multiple MXIbus slaves so that only the highest priority slave is selected during the data cycle. For more information on this technique, see Section 3.1.2.5.

User-defined address modifier codes are available for any purpose that MXIbus vendors or users deem appropriate.

3.1.1.3 CONVERT* Line

The MXIbus defines a data transfer conversion signal (CONVERT*). When this signal is asserted during a block or indivisible transfer (for information regarding block and indivisible transfers, see Sections 3.1.2.2 and 3.1.2.4 respectively), the MXIbus interface on the slave device must convert the block or indivisible cycles into single accesses on the slave's local bus. This conversion can increase performance to devices that do not support block or indivisible transfers on the local bus because block or indivisible transfers can still go across the MXIbus. When a MXIbus slave is converting a block cycle, the address of the single local accesses should be incremented by the data width for each access (4 for 32-bit blocks, 2 for 16-bit blocks, and 1 for 8-bit blocks) to access locations in ascending order. The address of the single cycles on the local bus should remain constant when converting indivisible cycles.

3.1.1.4 WR* Line

The MXIbus read/write signal (WR*) controls the direction of data flow during data transfers. When WR* is low, the MXIbus master performs a write cycle and supplies the data for the slave to store. When WR* is high, the MXIbus master performs a read cycle and the slave provides the data to the master.

3.1.2 Data Transfers

The MXIbus has 32 data lines (AD<31..00>*). These lines are multiplexed with the MXIbus address lines, and data is transferred while the data strobe (DS*) signal is asserted. The MXIbus supports 8-bit, 16-bit and 32-bit data transfers by encoding the transfer size on the AD01*, AD00*, and SIZE* lines (refer to Table 3-4). Data transfers that use all 32 data lines are called D32 cycles. Transfers that use 16 or 8 bits of the data bus are called D16 or D08 cycles, respectively, and use data lines AD<15..00>*. MXIbus devices can support one or more of the different data sizes.

The smallest addressable unit of storage is the byte location. Each byte may be assigned to one of four categories, according to the value of the least significant two bits of its address, as shown in Table 3-3. The *Byte Address* shown in the table represents the logical value rather than the signal levels.

Table 3-3. Byte Addressing

Category	Byte Address (AD<31..00>*)
Byte(0)	*XXX*...*XXX*00b
Byte(1)	*XXX*...*XXX*01b
Byte(2)	*XXX*...*XXX*10b
Byte(3)	*XXX*...*XXX*11b

A set of four bytes whose address differs only in the two least significant bits is referred to as a *longword*. Some, or all, of the bytes in a longword may be accessed simultaneously by a single MXIbus cycle. MXIbus masters use address lines AD<31..02>* and the address modifier lines AM<4..0>* to select which longword will be accessed. Three additional lines, AD01*, AD00*, and SIZE*, select which byte location(s) within the longword are accessed during the data transfer, as shown in Table 3-4.

Table 3-4. Data Transfer Types

Transfer Size	SIZE*	Address Phase		Data Phase			
		AD01*	AD00*	AD<31..24>*	AD<23..16>*	AD<15..08>*	AD<07..00>*
8-bit (Byte)	H	H	H			Byte(0)	
	H	H	L				Byte(1)
	H	L	H			Byte(2)	
	H	L	L				Byte(3)
16-bit (Word)	L	H	H			Byte(0)	Byte(1)
	L	L	H			Byte(2)	Byte(3)
32-bit (Longword)	L	X	L	Byte(0)	Byte(1)	Byte(2)	Byte(3)

Depending on how they are mapped in MXIbus address space, 8-bit-only devices can be designed to always transfer data on *either* AD<15..08>* *or* AD<07..00>*, or they can be designed to transfer data on *both* AD<15..08>* *and* AD<07..00>*. Of course, 16-bit devices shall be able to drive/receive data lines AD<15..00>*, and 32-bit devices shall be able to drive/receive all 32 data lines. All MXIbus devices, regardless of how they represent data on their internal local buses, shall be responsible for ensuring that all data transfers take place over the proper MXIbus data lines.

It is recommended that D16 devices also support D08 transfers if their local bus supports 8-bit transfers. Similarly, it is recommended that D32 devices also support D16 and D08 transfers, as long as their local buses support the smaller width transfers.

During a write cycle, the MXIbus master asserts DS* to indicate that it has placed valid data and valid parity driven on the PAR* signal on the bus. The addressed MXIbus slave asserts data transfer acknowledge (DTACK*) to indicate that it has successfully received the data.

During a read cycle, the addressed MXIbus slave asserts DTACK* to indicate that it has successfully placed valid data and valid parity driven on the PAR* signal on the bus. The MXIbus master unasserts DS* when it receives the data.

The addressed slave may optionally terminate a MXIbus cycle with a RETRY or BERR* response rather than the DTACK* response to DS* from the master. The slave uses the RETRY response if it is currently unable to complete the transfer but would be able to at a later time. If the slave terminates the transfer with a BERR* response, the data transfer is

considered invalid and the cycle is prematurely terminated. The signal timing relationships for the RETRY and BERR* response are slightly different than the DTACK* response. For more information on the RETRY and BERR* response, see Section 3.1.2.6.

The MXIbus supports five fundamental types of data transfers:

- Basic
- Block
- Synchronous burst
- Indivisible
- Priority select

Section 3.1.2.1 describes the basic data transfers, while Figures 3-1 and 3-2 show the timing requirements for basic write and read data transfers, respectively. Section 3.1.2.2 explains block transfers, and the block timing diagrams are presented in Figures 3-3 and 3-4. Synchronous-burst cycle timing is shown in Figures 3-5 through 3-12, and is explained in Section 3.1.2.3. Indivisible cycle timing is shown in Figure 3-13, and is explained in Section 3.1.2.4. Priority-selection cycles are shown in Figure 3-14 and described in Section 3.1.2.5.

Section 3.1.2.6, which describes the RETRY and BERR* acknowledgments, applies to most types of MXIbus data transfer cycles. Figure 3-15 shows the timing of a slave with a RETRY or BERR* acknowledgment in response to any of the allowed MXIbus data transfer cycles.

Notice that in these timing diagrams, *master* refers to the timing as seen on the MXIbus at the master device connector; *slave* refers to the timing as seen on the MXIbus at the slave device connector.

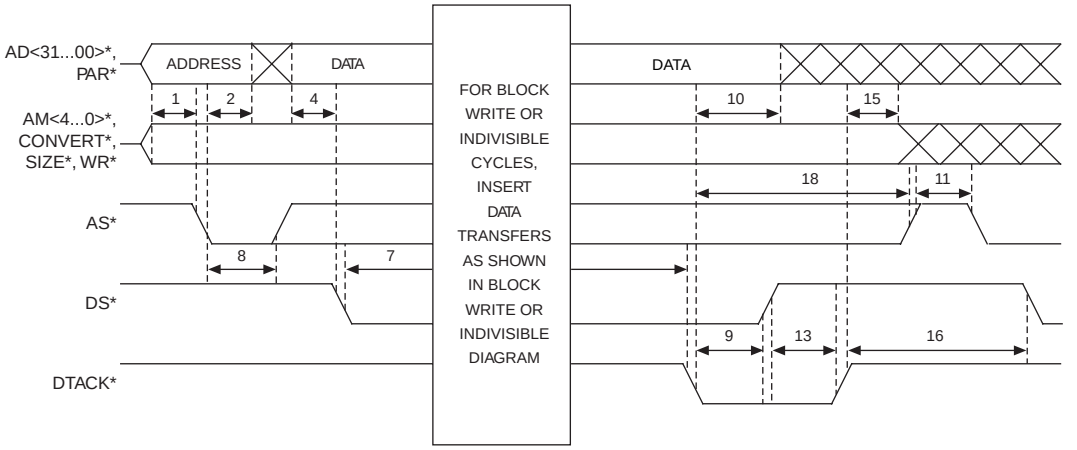
3.1.2.1 Basic Data Transfers

During a basic transfer cycle, the master broadcasts an address and a single data transfer takes place between the master and the addressed slave. This is the most basic type of data transfer on which all other transfer types are based. The basic data transfer cycle address modifier codes are shown in Table 3-5. The table shows the address modifier logical values in hex rather than the signal levels.

Table 3-5. Basic Data Transfer Cycle Address Modifier Codes

Address Modifier AM<4..0>*	Description
1E	A24, supervisory, program access
1D	A24, supervisory, data access
1A	A24, nonprivileged, program access
19	A24, nonprivileged, data access
15	A16, supervisory access
11	A16, nonprivileged access
06	A32, supervisory, program access
05	A32, supervisory, data access
02	A32, nonprivileged, program access
01	A32, nonprivileged, data access

The MXIbus master device generates the PAR* signal during the data phase of basic write transfers. A MXIbus slave that detects invalid parity during the data phase of a basic write transfer should either respond with a BERR* or ignore the transfer and let the bus timeout unit terminate the transfer.

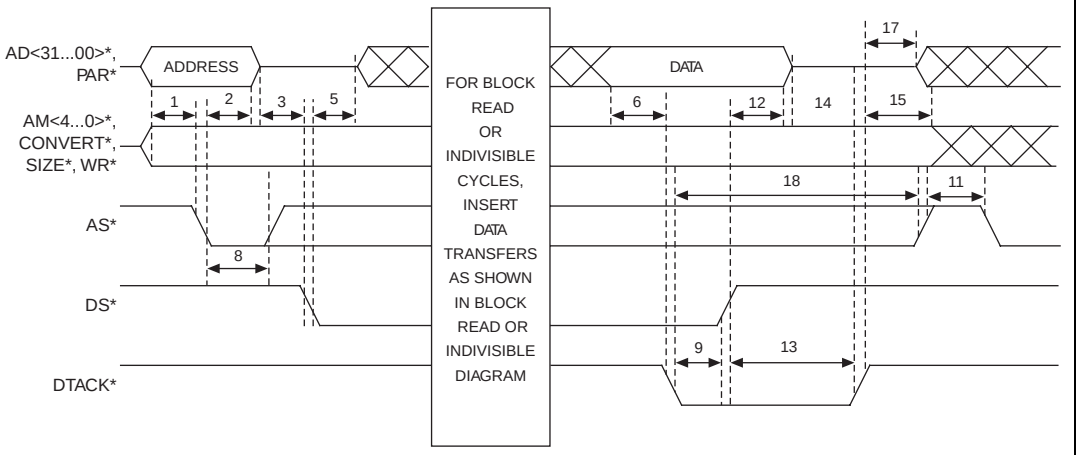


Parameter	Master		Slave		Description
	min	max	min	max	
1	70	–	50	–	AD<31..00>*, PAR*, AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable before AS* low (address setup).
2	50	–	30	–	AD<31..00>* and PAR* valid/stable after AS* low (address hold).
4	70	–	50	–	AD<31..00>* and PAR* valid/stable to DS* low (write data setup).
7	0	–	0	–	DS* low to DTACK* low (slave shall wait for DS* to be asserted before asserting DTACK*).
8	70	–	50	–	Minimum AS* assertion time.
9	0	–	0	–	DTACK* low to DS* high (master shall wait for slave to assert DTACK* before unasserting DS*).
10	0	–	0	–	AD<31..00>* and PAR* valid/stable after DTACK* low (write data hold).
11	70	–	50	–	Minimum AS* unassertion time.
13	0	–	0	–	DS* high to DTACK* high (slave shall wait for master to unassert DS* before unasserting DTACK*).
15	0	–	0	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable after DTACK* high (master shall not change these lines until slave unasserts DTACK*).
16	0	–	0	–	DTACK* high to DS* low (master must wait for slave to unassert DTACK* before asserting DS* for next cycle).
18	0	–	0	–	DTACK* low to AS* high (master must wait for slave to assert the last DTACK* of a block or indivisible transfer before unasserting AS*). Note: this applies only to block and indivisible transfers.

Note: All times are in nanoseconds as measured/observed on the MXIbus.

Figure 3-1. Basic Write Cycle Timing

The MXIbus slave device generates the PAR* signal during the data phase of basic read transfers. A MXIbus master that detects invalid parity during the data phase of a basic read transfer should treat the data received as corrupt and handle the error appropriately.



Parameter	Master		Slave		Description
	min	max	min	max	
1	70	–	50	–	AD<31..00>*, PAR*, AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable before AS* low (address setup).
2	50	–	30	–	AD<31..00>* and PAR* valid/stable after AS* low (address hold).
3	20	–	0	–	AD<31..00>* and PAR* float to DS* low (master shall remove address and PAR* from bus before allowing slave to drive them).
5	0	–	0	–	DS* low before data driven on the bus (slave shall not drive AD<31..00>* and PAR* until it receives DS* low).
6	0	–	20	–	AD<31..00>* and PAR* valid/stable before DTACK* low (read data setup).
8	70	–	50	–	Minimum AS* assertion time.
9	0	–	0	–	DTACK* low to DS* high (master shall wait for slave to assert DTACK* before unasserting DS*).
11	70	–	50	–	Minimum AS* unassertion time.
12	0	–	0	–	AD<31..00>* and PAR* valid/stable after DS* high (read data hold).
13	0	–	0	–	DS* high to DTACK* high (slave shall wait for master to unassert DS* before unasserting DTACK*).

Figure 3-2. Basic Read Cycle Timing

Parameter	Master		Slave		Description
	min	max	min	max	
14	0	–	20	–	AD<31..00>* and PAR* float before DTACK* high (slave shall quit driving AD<31..00>* and PAR* before unasserting DTACK*).
15	0	–	0	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable after DTACK* high (master shall not change these lines until slave unasserts DTACK*).
17	0	–	0	–	DTACK* high to AD<31..00>* and PAR* driven by master (master shall wait for slave to unassert DTACK* before driving AD<31..00>* and PAR*).
18	0	–	0	–	DTACK* low to AS* high (master must wait for slave to assert the last DTACK* of a block or indivisible transfer before unasserting AS*). Note: this applies only to block and indivisible transfers.
Note: All times are in nanoseconds as measured/observed on the MXIbus.					

Figure 3-2. Basic Read Cycle Timing (Continued)

3.1.2.2 Block Transfers

Masters often access memory locations in ascending order. When this is the case, block transfer cycles can be beneficial because they allow the MXIbus master to provide a single address, and then access data in that location and those at higher addresses, without providing additional addressing information. MXIbus devices are not required to implement block-mode functionality, but it is recommended that they do, because block-mode transfers can result in significant improvements in performance. A MXIbus master initiates block-mode operation by sourcing one of the address modifier codes shown in Table 3-6. The table shows the address modifier logical values in hex rather than the signal levels.

Table 3-6. Block Transfer Address Modifier Codes

Address Modifier AM<4..0>*	Description
1F	A24, supervisory, block transfer
1C	A24, supervisory, D64 transfer
1B	A24, nonprivileged, block transfer
18	A24, nonprivileged, D64 transfer
07	A32, supervisory, block transfer
04	A32, supervisory, D64 transfer
03	A32, nonprivileged, block transfer
00	A32, nonprivileged, D64 transfer

When a MXIbus master initiates a block transfer cycle, all of the MXIbus block-mode slaves on the bus latch the MXIbus address into onboard address counters and examine the address (and the address modifier code) to determine if the address is within their address ranges. The block-mode slave that is selected then participates in the data transfer. All MXIbus block-mode slaves shall increment their address counters by the width of the data (4 for 32-bit block, 2 for 16-bit block, and 1 for 8-bit block) at the end of each data transfer within the block and examine the resulting address to determine if the next cycle is intended for them. The MXIbus slave that is selected then participates in the next cycle.

The MXIbus master, upon completing the first data transfer, shall not unassert AS* as it would for a normal cycle. Instead, it shall continue to assert AS* for the duration of the block transfer, and shall repeatedly assert and unassert DS* in response to DTACK* acknowledgments from the addressed MXIbus slave, and thereby transfer data to or from sequential address locations in ascending order. To access the next location(s) on their local buses, the block-mode slaves can either source the new value of the address counter on the local bus, or, if possible, perform block-mode transfers on their local buses.

Notice that the length of a block transfer is not limited to the address space of a single MXIbus device, because each MXIbus block-mode slave shall continually monitor and decode the output of its onboard address counter. The length of a block-mode transfer may range from one

transfer (4, 2, or 1 B, depending on the width of the data path) up to 4 GB (for A32 space).

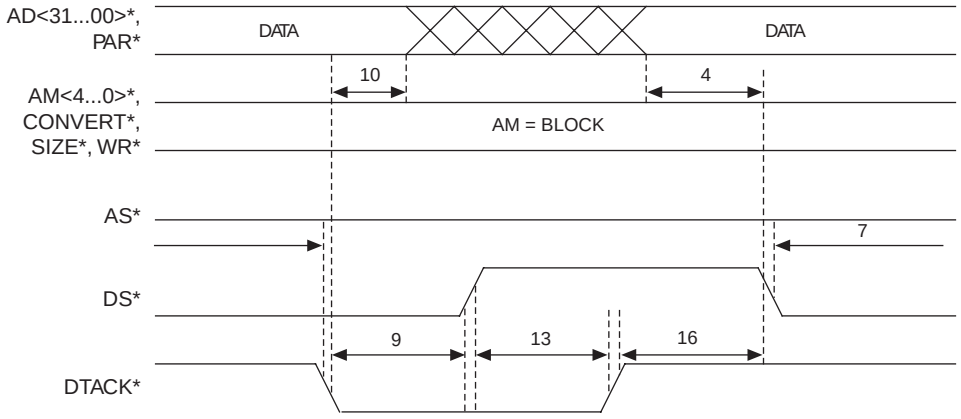
The MXIbus master shall hold AS* asserted until the responding slave asserts DTACK* for the last transfer of the block-mode operation. The unassertion of AS* indicates the end of the block-mode transfer. The slave can use this signal to release control of its local bus, or to perform any other device-dependent function.

During D64 mode (address modifier codes hex 1C, 18, 04, and 00), every two sequential data transfers on the MXIbus are considered a single 64-bit datum by the slave because the MXIbus has only 32 data lines.

MXIbus devices that are transferring data in 8-bit (byte) blocks shall alternate the byte lane used to carry data for each cycle throughout the block-mode operation. The byte lane that is used for any given cycle depends on the implicit state of the two least significant address lines, as shown in Table 3-4.

The MXIbus master is responsible for ensuring that block-mode operations are initiated only with slave devices that support block-mode operation. If a block-mode operation is attempted with a slave that does not support block-mode, the slave may ignore the cycle, or it may immediately terminate the transfer with a BERR* response.

The MXIbus master device generates the PAR* signal during each data phase of a block-write transfer. A MXIbus slave that detects invalid parity during a data phase of a block-write transfer should either respond with a BERR* or ignore the transfer and let the bus timeout unit terminate the transfer.

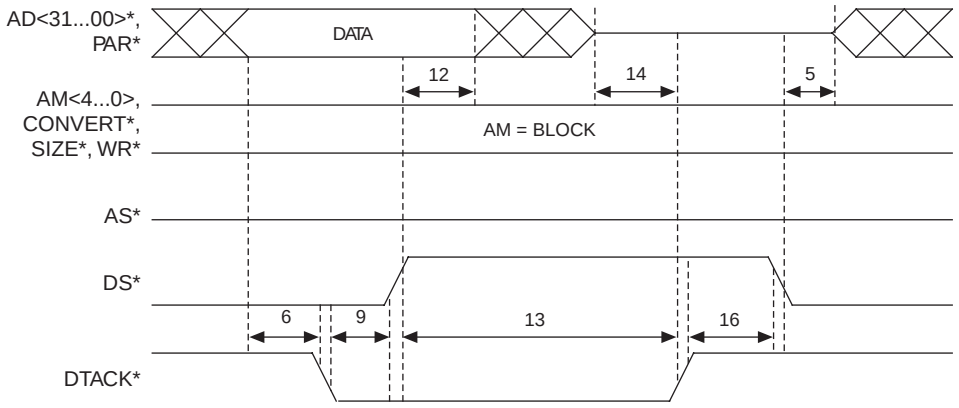


Parameter	Master min	Master max	Slave min	Slave max	Description
4	70	–	50	–	AD<31..00>* and PAR* valid/stable to DS* low (write data setup).
7	0	–	0	–	DS* low to DTACK* low (slave shall wait for DS* to be asserted before asserting DTACK*).
9	0	–	0	–	DTACK* low to DS* high (master shall wait for slave to assert DTACK* before unasserting DS*).
10	0	–	0	–	AD<31..00>* and PAR* valid/stable after DTACK* low (write data hold).
13	0	–	0	–	DS* high to DTACK* high (slave shall wait for master to unassert DS* before unasserting DTACK*).
16	0	–	0	–	DTACK* high to DS* low (master must wait for slave to unassert DTACK* before asserting DS* for next cycle).

Note: All times are in nanoseconds as measured/observed on the MXIbus.

Figure 3-3. Block-Write Cycle Timing

The MXIbus slave device generates the PAR* signal during each data phase of a block-read transfer. A MXIbus master that detects invalid parity during a data phase of a block-read transfer should treat the data received as corrupt and handle the error appropriately.



Parameter	Master		Slave		Description
	min	max	min	max	
5	0	–	0	–	DS* low to AD<31..00>* and PAR* driven (responding slave shall not drive AD<31..00>* and PAR* until it receives DS* low).
6	0	–	20	–	AD<31..00>* and PAR* valid/stable to DTACK* low (read data setup).
9	0	–	0	–	DTACK* low to DS* high (master shall wait for slave to assert DTACK* before unasserting DS*).
12	0	–	0	–	AD<31..00>* and PAR* valid/stable after DS* high (read data hold).
13	0	–	0	–	DS* high to DTACK* high (slave shall wait for master to unassert DS* before unasserting DTACK*).
14	0	–	20	–	AD<31..00>* and PAR* float to DTACK* high (slave shall quit driving AD<31..00>* and PAR* before unasserting DTACK*). Not necessary if next block-mode transfer will be from same slave. Note: if AS* unasserts (ending the block transfer), the slave must quit driving AD<31..00>* and PAR* no more than 30 ns later.
16	0	–	0	–	DTACK* high to DS* low (master must wait for slave to unassert DTACK* before asserting DS* for next cycle).

Note: All times are in nanoseconds as measured/observed on the MXIbus.

Figure 3-4. Block-Read Cycle Timing

3.1.2.3 Synchronous-Burst Cycles

Some applications may not require an acknowledgment on each data transfer within a block. Instead, an acknowledgment at the end of the last transfer is all that is required to indicate if all the data transferred successfully. For this type of application, synchronous-burst transfer

cycles can be beneficial because they allow the MXIbus master to transfer blocks of data without waiting for a handshake acknowledge signal on each transfer. Additionally, synchronous-burst transfers allow the MXIbus master to send a single address, and then access data either at that location and those at higher addresses (similar to block-mode), or at that location repeatedly (similar to indivisible channel I/O transfers), without providing additional addressing information.

MXIbus devices are not required to implement synchronous-burst functionality, but it is recommended that they do, because synchronous-burst transfers can result in significant improvements in performance (even greater than block-mode). In addition, the performance of synchronous-burst transfers is almost unaffected by the length of MXIbus cable over which the transfer takes place.

A MXIbus master indicates a synchronous-burst cycle by sourcing one of the synchronous-burst address modifier codes shown in Table 3-7. The table shows the address modifier logical values in hex rather than the signal levels.

Table 3-7. Synchronous-Burst Transfer Address Modifier Codes

Address Modifier AM<4..0>*	Description
17	A24, synchronous-burst transfer
14	A32, synchronous-burst transfer

MXIbus synchronous-burst cycles consist of three phases: initiation, data transfer, and termination.

3.1.2.3.1 Initiation

To initiate a synchronous-burst cycle, the MXIbus master conducts a special transfer, a synchronous-burst initiation transfer, to indicate to the selected slave that it is attempting a synchronous-burst cycle. During a synchronous-burst initiation transfer, the master broadcasts an address and the synchronous-burst address modifiers, and a single transfer takes place between the master and the addressed slave, much like a basic write cycle. However, the value the master drives onto the AD<31..00>* lines during the data phase contains the number of bytes the master is going to attempt to transfer following the initiation transfer. The address modifier code to be used for the destination is also included on the AD<31..00>* lines during the data phase of the initiation transfer, because the

AM<4..0>* lines must be used to indicate a synchronous-burst transfer. Notice that the MXIbus master always drives the AD<31..00>* lines during the data phase of the initiation transfer of a synchronous-burst, even when initiating a synchronous-burst read. The bit positions of the transfer count and destination address modifiers during the data phase are as follows:

AD<31..26>*	**AD<25..16>***	**AD<15..00>***
AM<4..3>, '1', AM<2..0>	'0000000000'	CNT<15..00>

CNT<15..00> contains the byte count of the transfer. The CNT<15..00> value must be a multiple of the data size to transfer during the synchronous burst (4 for 32-bit transfers, 2 for 16-bit transfers, and 1 for 8-bit transfers). The bit positions labeled '1' and '0000000000' are reserved and must be driven with the corresponding logical value.

When a MXIbus master initiates a synchronous-burst transfer cycle, all of the MXIbus synchronous burst-mode slaves examine the address to determine if it is within their address ranges and the AM<4..0>* lines match the space the device is mapped into. Slaves that are in any of the MXIbus A24 spaces will be selected by the synchronous-burst A24 address modifier code, logical 17 hex. Slaves that are in any of the MXIbus A32 spaces will be selected by the synchronous-burst A32 address modifier code, logical 14 hex.

The selected slave latches the MXIbus address, destination address modifiers, and transfer count, acknowledges the initiation cycle with DTACK*, and then participates in the data transfer. If no synchronous burst-mode slave acknowledges the transfer, the Bus Timeout Unit (BTO) will assert BERR* after the bus timeout period to terminate the cycle.

The selected synchronous burst-mode slave shall increment its address counters at the end of each transfer if the destination address modifiers indicate block-mode or D64-type transfers. Notice that the address counter will be incremented by 4, 2, or 1 bytes, depending on the data width of the transfer. The slave also uses destination address modifiers to drive the address modifiers on another bus if the slave is a bridge between two buses and not the final destination of the data.

The initiation transfer for a synchronous-burst transfer is the same for synchronous-burst read and write cycles. The timing diagram for the

initiation transfer of a synchronous-burst cycle is shown in Figure 3-5. Notice that although the initiation transfer is essentially a 32-bit write transfer to the selected slave, the WR* and SIZE* lines indicate the transfer direction and data width of the transfers that will take place during the data transfer phase of the synchronous-burst cycle.

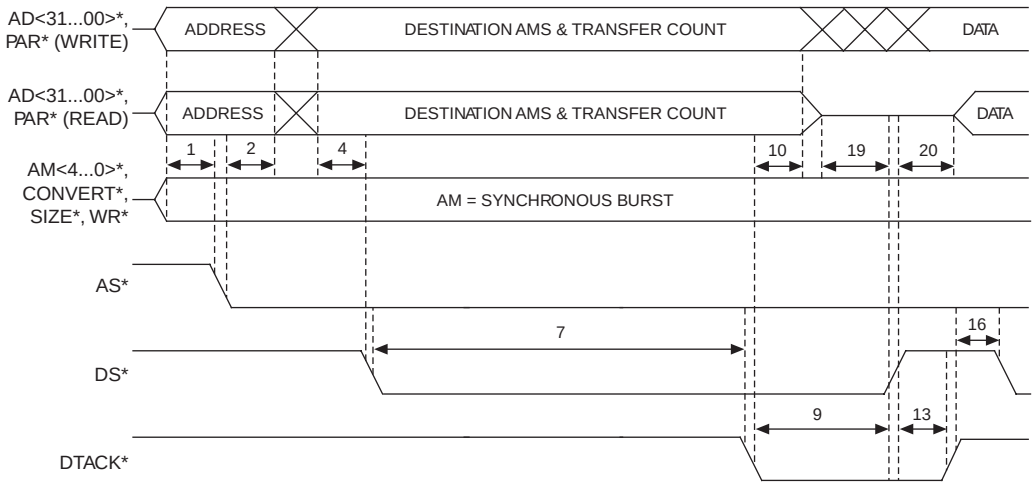
The MXIbus master is responsible for ensuring that synchronous burst-mode operations are initiated only with slave devices that support synchronous-burst operation. If a synchronous-burst operation is attempted with a slave that does not support synchronous burst-mode, the slave may ignore the cycle or may immediately terminate the transfer by asserting the BERR* signal during the initiation cycle.

The MXIbus master, upon completing the initiation transfer, shall not unassert AS* as it would for a normal cycle. Instead, the master continues to assert AS* for the duration of the synchronous-burst cycle.

The destination address modifiers that the synchronous-burst slave latches during the data phase of the initiation transfer are used to determine if the slave should access locations in ascending order or repeatedly access the same location during the synchronous-burst transfer. If the destination address modifiers indicate either a block-mode or D64-type transfer, the slave should access locations in ascending order; otherwise the same location should be accessed repeatedly.

Unlike MXIbus block cycles, the length of a synchronous-burst cycle *is* limited to the address space of a single MXIbus device. Because of the asynchronous nature of this protocol, it is impossible for each MXIbus synchronous-mode slave to continually monitor and decode the output of its onboard address counters during the synchronous-burst transfer. The length of a synchronous burst-mode transfer may range from one transfer (4, 2, or 1 bytes, depending on the width of the data path) up to 65,535 bytes.

The MXIbus master device generates the PAR* signal during the data phase of the initiation transfer. A MXIbus slave that detects invalid parity during the data phase of the initiation transfer should either respond with a BERR* or ignore the transfer and let the bus timeout unit terminate the transfer.



Parameter	Master		Slave		Description
	min	max	min	max	
1	70	–	50	–	AD<31..00>*, PAR*, AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable before AS* low (address setup).
2	50	–	30	–	AD<31..00>* and PAR* valid/stable after AS* low (address hold).
4	70	–	50	–	AD<31..00>* and PAR* valid/stable to DS* low (write data setup).
7	0	–	0	–	DS* low to DTACK* low (slave shall wait for DS* to be asserted before asserting DTACK*).
9	0	–	0	–	DTACK* low to DS* high (master shall wait for slave to assert DTACK* before unasserting DS*).
10	0	–	0	–	AD<31..00>* and PAR* valid/stable after DTACK* low (write data hold).
13	0	–	0	–	DS* high to DTACK* high (slave shall wait for master to unassert DS* before unasserting DTACK*).
16	0	–	0	–	DTACK* high to DS* low (master must wait for slave to unassert DTACK* before asserting DS* for next cycle during synchronous burst-write cycles).
19	20	–	0	–	AD<31..00>* and PAR* float before DS* high on synchronous-burst read cycles (master shall quit driving AD<31..00>* and PAR* before unasserting DS* on synchronous-burst reads).
20	0	–	0	–	AD<31..00>* and PAR* driven after DS* high on synchronous-burst read cycles (slave shall not drive AD<31..00>* and PAR* before the master unasserts DS* on synchronous-burst reads).

Note: All times are in nanoseconds as measured/observed on the MXIbus.

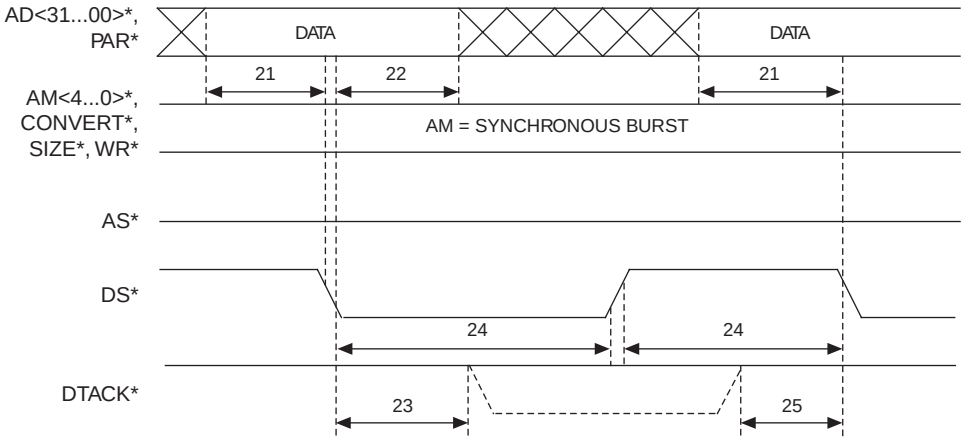
Figure 3-5. Synchronous-Burst Initiation Transfer Timing

3.1.2.3.2 Data Transfer

MXIbus devices that are transferring block-mode data in 8-bit (byte) synchronous-burst transfers shall alternate the byte lane that carries data for each cycle throughout the synchronous burst-mode operation. When transferring nonblock-mode data in 8-bit synchronous-burst transfers, only one byte lane is used. The byte lane for any given cycle depends on the implicit state of the two least significant address lines, as shown in Table 3-4.

The data transfer phase of a synchronous-burst transfer is different for read and write transfers. For synchronous-burst write cycles, the MXIbus master conducts each data transfer by driving the data on the AD<31..00>* lines and pulsing DS*. The slave can throttle the transfer by asserting DTACK*. After DTACK* is asserted, the master will stop sending data until DTACK* is unasserted. The slave must meet timing parameter 23 to guarantee that the master will not send more than four more data transfers after the slave asserts DTACK* due to propagation delays through a MXIbus system. Thus, the slave must be prepared to accept up to four more transfers after it asserts DTACK*. The timing diagram for the data transfer phase of a synchronous burst-write cycle is shown in Figure 4-6.

The MXIbus master device generates the PAR* signal during each data phase of a synchronous burst-write transfer. A MXIbus slave that detects invalid parity during a data phase of a synchronous burst-write transfer should prematurely terminate the transfer with a BERR*.



Parameter	Master min	Master max	Slave min	Slave max	Description
21	40	–	20	–	AD<31..00>* and PAR* valid/stable to DS* low (write data setup).
22	55	–	35	–	AD<31..00>* and PAR* valid/stable after DS* low (write data hold).
23	–	350	–	150	DS* low to DTACK* low (this parameter must be met by the slave to ensure no more than four more transfers will be sent by the master before the transfer is paused. After the master receives DTACK* low, it shall not assert DS* after 50 ns).
24	50	–	30	–	DS* low to DS* high or DS* high to DS* low (minimum assertion or unassertion time for DS*).
25	0	–	0	–	DTACK* high to DS* low (master shall not assert DS* again until slave has unasserted DTACK* once the cycle is paused †).

Notes: All times are in nanoseconds as measured/observed on the MXIbus.

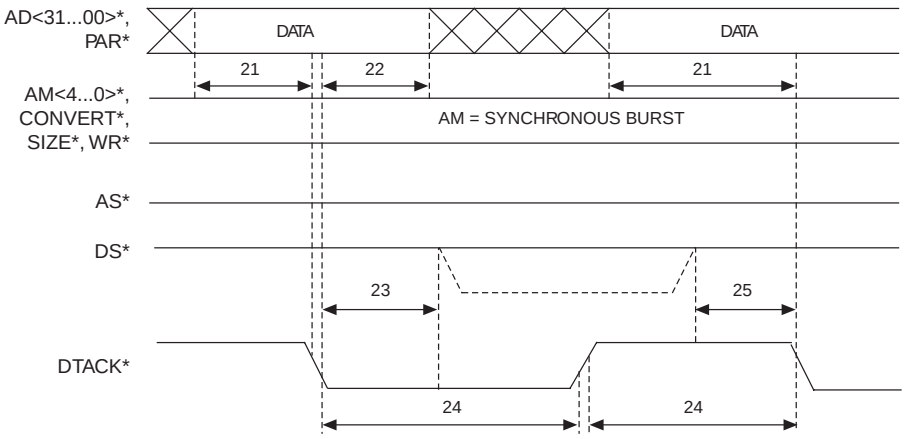
† Paused implies the master has received the DTACK* low and will not be sourcing any more data transfers until DTACK* unasserts.

Figure 3-6. Synchronous Burst-Write Transfer Timing

For synchronous burst-read transfers, the MXIbus slave conducts each data transfer by driving the data on the AD<31..00>* lines and pulsing DTACK*. The master can throttle the transfer by asserting DS*. After DS* is asserted, the slave will stop sending data until DS* is unasserted. The master must meet timing parameter 23 to guarantee that the slave will not send more than four more data transfers after the master asserts DS* due to propagation delays through a MXIbus system. Thus, the master must be prepared to accept up to four more transfers after it

asserts DS*. The timing diagram for the data transfer phase of a synchronous burst-read cycle is shown in Figure 3-7.

The MXIbus slave device generates the PAR* signal during each data phase of a synchronous burst-read transfer. A MXIbus master that detects invalid parity during a data phase of a synchronous burst-read transfer should treat the data received as corrupt and handle the error appropriately.



Parameter	Master min	Master max	Slave min	Slave max	Description
21	20	–	40	–	AD<31..00>* and PAR* valid/stable to DTACK* low (read data setup).
22	35	–	55	–	AD<31..00>* and PAR* valid/stable after DTACK* low (read data hold).
23	–	150	–	350	DTACK* low to DS* low (this parameter must be met by the master to insure no more than four more transfers will be sent by the slave before the transfer is paused. After the slave receives DS* low it shall not assert DTACK* after 50 ns).
24	30	–	50	–	DTACK* low to DTACK* high or DTACK* high to DTACK* low (minimum assertion or unassertion time for DTACK*).
25	0	–	0	–	DS* high to DTACK* low (slave shall not assert DTACK* again until master has unasserted DS* once the cycle is paused †).

Notes: All times are in nanoseconds as measured/observed on the MXIbus.

† Paused implies the slave has received the DS* low and will not be sourcing any more data transfers until DS* unasserts.

Figure 3-7. Synchronous Burst-Read Transfer Timing

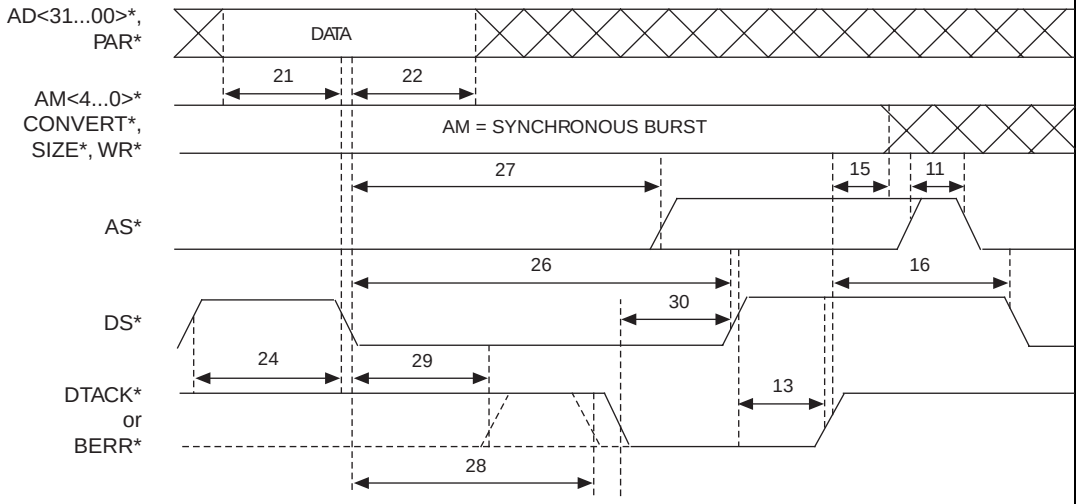
3.1.2.3.3 Termination

A synchronous-burst transfer can be terminated in one of the three ways listed below:

- Normal completion by transferring the exact transfer count
- Premature master-termination (unassert AS*)
- Premature slave-termination (assert BERR*)

Normal completion occurs when the entire transfer count indicated in the initiation transfer has been transferred. While conducting the last transfer, the device supplying the data must wait for a given time period to allow the other device to synchronize with it before terminating the transfer. For synchronous burst-write transfers, the master must assert DS* during the last transfer for a longer time period than usual and then wait for DTACK* (or BERR*) to assert before terminating the cycle by unasserting DS* and AS*. For synchronous burst-read transfers, the slave must assert DTACK* during the last transfer until the master unasserts AS*. The timing diagrams for normal completion of synchronous burst-write and burst-read cycles are shown in Figures 3-8 and 3-9.

The MXIbus master device generates the PAR* signal during the last data phase of a synchronous burst-write transfer. A MXIbus slave which detects invalid parity during the last data phase of a synchronous burst-write transfer should either terminate the transfer with a BERR* or ignore the transfer and let the bus timeout unit terminate the transfer.



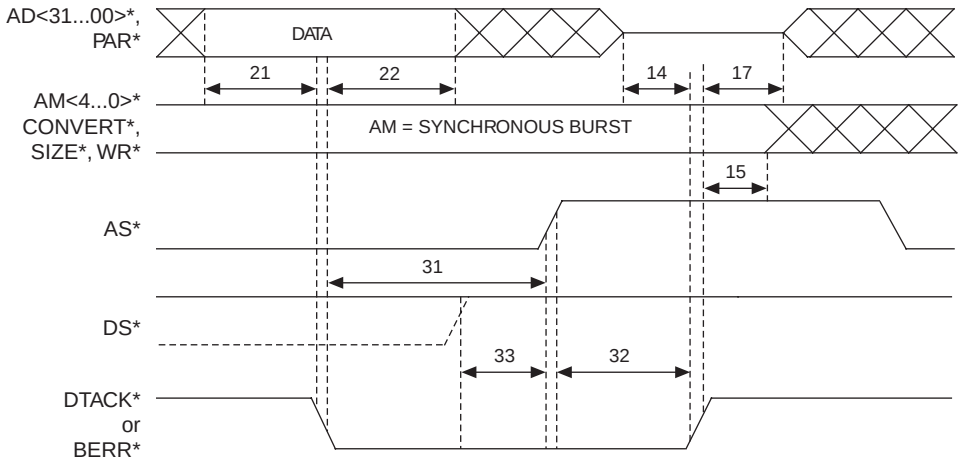
Parameter	Master min	Master max	Slave min	Slave max	Description
11	70	–	50	–	Minimum AS* unassertion time.
13	0	–	0	–	DS* high to DTACK* or BERR* high (slave shall wait for master to unassert DS* before unasserting the last DTACK* or BERR*).
15	0	–	0	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable after DTACK* or BERR* high (master shall not change these lines until the slave unasserts DTACK* or BERR* for the last time).
16	0	–	0	–	DTACK* or BERR* high to DS* low (master shall not assert DS* for next cycle until slave has unasserted the last DTACK* or BERR*).
21	40	–	20	–	AD<31..00>* and PAR* valid/stable to DS* low (write data setup).
22	55	–	35	–	AD<31..00>* and PAR* valid/stable after DS* low (write data hold).
24	50	–	30	–	DS* high to DS* low (minimum unassertion time for DS*).
26	350	–	330	–	DS* low to DS* high (minimum assertion time for DS* on the last transfer).
27	350	–	330	–	DS* low to AS* high—the master must wait this time period before unasserting AS* so the slave can distinguish between a normal termination and a premature termination (Figure 3-10).

Figure 3-8. Synchronous Burst-Write Cycle Termination Timing

Parameter	Master		Slave		Description
	min	max	min	max	
28	350	–	N/A	N/A	DS* low to master samples for DTACK* or BERR* low—the master must wait for this time period to give the slave a chance to respond to the last transfer and give a DTACK* or BERR* for the entire transfer.
29	0	350	–	150	DS* low to slave unasserts DTACK* due to a paused transfer—since this is the last transfer, no more transfers will be sourced by the master; the slave will unassert DTACK* if it was asserted and keep DTACK* and BERR* unasserted until it has completed all the transfers in its internal buffer, then assert DTACK* or BERR* to indicate the acknowledge for the entire transfer.
30	0	–	0	–	DTACK* low or BERR* low after waiting timing parameter 28 to DS* high (master shall not unassert DS* until either DTACK* or BERR* is asserted after timing parameter 28).
Note: All times are in nanoseconds as measured/observed on the MXIbus.					

Figure 3-8. Synchronous Burst-Write Cycle Termination Timing (Continued)

The MXIbus slave device generates the PAR* signal during the last data phase of a synchronous burst-read transfer. A MXIbus master that detects invalid parity during the last data phase of a synchronous burst-read transfer should treat the data received as corrupt and handle the error appropriately.



Parameter	Master min	Master max	Slave min	Slave max	Description
14	0	–	20	–	AD<31..00>* and PAR* float to DTACK* or BERR* high (slave shall quit driving AD<31..00>* and PAR* before unasserting the last DTACK* or BERR*).
15	0	–	0	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable after DTACK* or BERR* high (master shall not change these lines until slave unasserts the last DTACK* or BERR*).
17	0	–	0	–	DTACK* or BERR* high to AD<31..00>* and PAR* driven by master (master shall wait for slave to unassert the last DTACK* or BERR* before driving AD<31..00>* and PAR*).
21	20	–	40	–	AD<31..00>* and PAR* valid/stable to DTACK* low (read data setup). Not necessary for a BERR* acknowledge.
22	35	–	55	–	AD<31..00>* and PAR* valid/stable after DTACK* low (read data hold). Not necessary for a BERR* acknowledge.
31	0	–	0	–	Last DTACK* or BERR* low to AS* high (master shall not unassert AS* until DTACK* or BERR* is low on the last transfer).
32	0	–	0	–	AS* high to DTACK* or BERR* high (slave shall not unassert DTACK* or BERR* until AS* is high).
33	20	–	0	–	DS* high to AS* high (master must remove DS* for the last time before unasserting AS*). Note: DS* might not assert.

Note: All times are in nanoseconds as measured/observed on the MXIbus.

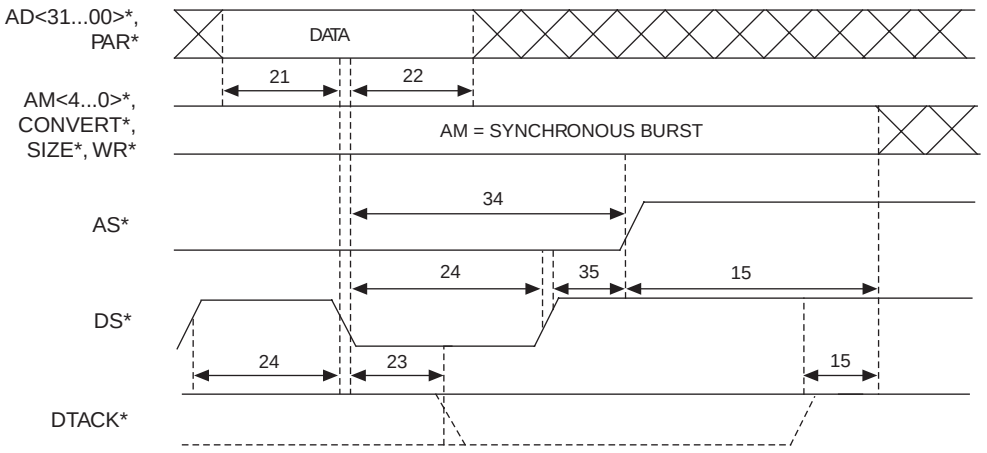
Figure 3-9. Synchronous Burst-Read Cycle Termination Timing

The master of the transfer can prematurely abort synchronous-burst cycles. This should occur only when there is an error in the system, and provides a means of recovering without locking up the MXIbus. When cycles are terminated in this manner, there is no guarantee that any of the data transferred to the destination successfully because there are no acknowledges to provide feedback.

For synchronous burst-write transfers, the master must wait for a defined time period after asserting DS* for the last time then unassert AS*. The waiting period is required to give the slave a chance to assert DTACK* before AS* is unasserted, which prevents DTACK* from asserting after AS* is high, possibly interfering with other cycles.

For synchronous burst-read transfers, the master must assert DS* for a defined time period, then unassert AS* and DS*. The waiting period is required to give the slave the chance to quit sending data (pulsing DTACK*) before AS* is unasserted.

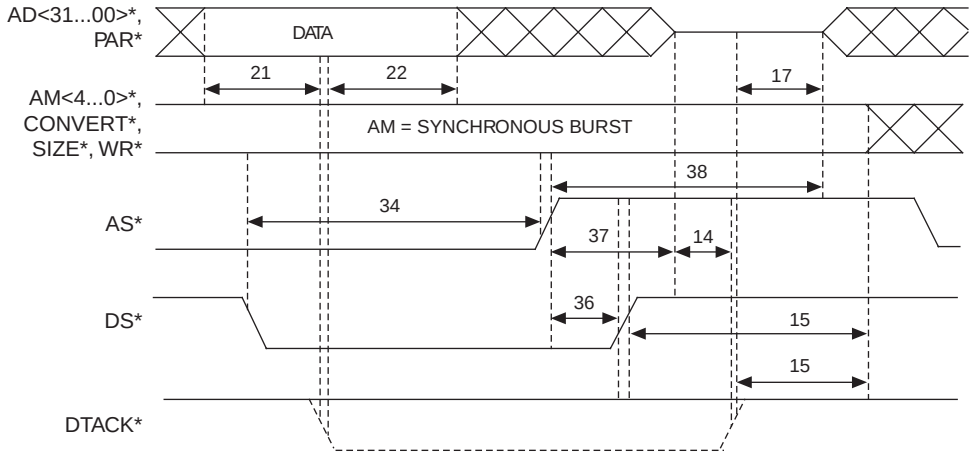
The timing diagrams for prematurely master-terminated synchronous burst-write and burst-read cycles are shown in Figures 3-10 and 3-11.



Parameter	Master min	Master max	Slave min	Slave max	Description
15	0	–	0	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable after DTACK* high and AS* high whichever comes later (master shall not change these lines until it has unasserted AS* and the slave unasserts DTACK*). Note: DTACK* may not assert.
21	40	–	20	–	AD<31..00>* and PAR* valid/stable to DS* low (write data setup).
22	55	–	35	–	AD<31..00>* and PAR* valid/stable after DS* low (write data hold).
23	–	350	–	150	DS* low to DTACK* low (this parameter must be met by the slave to ensure that DTACK* will not be asserted after the transfer is aborted).
24	50	–	30	–	DS* low to DS* high or DS* high to DS* low (minimum assertion or unassertion time for DS*).
34	350	–	330	–	Last DS* low to AS* high (master shall not unassert AS* until at least this time period after the last assertion edge of DS*).
35	20	–	0	–	DS* high to AS* high (master shall unassert DS* before unasserting AS*).

Note: All times are in nanoseconds as measured/observed on the MXIbus.

Figure 3-10. Prematurely Master-Terminated Synchronous Burst-Write Cycle Timing



Parameter	Master		Slave		Description
	min	max	min	max	
14	0	–	20	–	AD<31..00>* and PAR* float to DTACK* high (slave shall quit driving AD<31..00>* and PAR* before unasserting the last DTACK*).
15	0	–	0	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable after DTACK* high and DS* high whichever comes later (master shall not change these lines until it has unasserted DS* and the slave unasserts DTACK*). Note: DTACK* may not assert.
17	0	–	0	–	DTACK* high to AD<31..00>* and PAR* driven by master (master shall wait for slave to unassert the last DTACK* before driving AD<31..00>* and PAR*). Note: DTACK* might not assert.
21	20	–	40	–	AD<31..00>* and PAR* valid/stable to DTACK* low (read data setup).
22	35	–	55	–	AD<31..00>* and PAR* valid/stable after DTACK* low (read data hold).
34	350	–	330	–	Last DS* low to AS* high (master shall not unassert AS* until at least this time period after the assertion edge of DS*).
36	50	–	30	–	AS* high to DS* high (master shall unassert AS* before unasserting DS*; this is how the slave can determine an abort from a paused release).
37	–	50	–	30	AD<31..00>* and PAR* float after AS* high (slave must release AD<31..00>* and PAR* after AS* is unasserted unless the slave is asserting DTACK*).

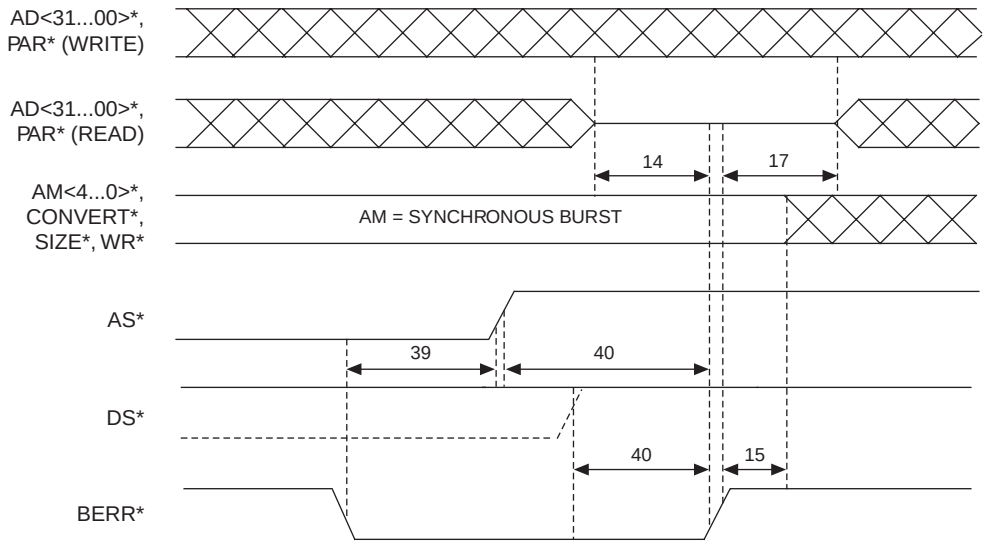
Figure 3-11. Prematurely Master-Terminated Synchronous Burst-Read Cycle Timing

Parameter	Master		Slave		Description
	min	max	min	max	
38	50	–	–	–	AD<31..00>* and PAR* driven after AS* high (master must not drive AD<31..00>* and PAR* until this time after AS* is unasserted). Note: if DTACK* is asserted the master must wait timing parameter 17 before driving these lines.
Note: All times are in nanoseconds as measured/observed on the MXIbus.					

Figure 3-11. Prematurely Master-Terminated Synchronous Burst-Read Cycle Timing (Continued)

The slave of the transfer can prematurely abort synchronous-burst cycles. This should occur only when there is an error in the system, and provides a means of recovering without locking up the MXIbus. When cycles are terminated in this manner, there is no guarantee that any of the data transferred to the destination successfully because there are no acknowledges to provide feedback.

For synchronous burst-write or burst-read transfers, the slave must assert BERR* until AS* and DS* unassert. The synchronous-burst protocol does not support RETRY responses at any time except during the initiation phase. Therefore, the slave should not, under any condition during a synchronous-burst data transfer or termination, assert DTACK* and BERR* at the same time. The timing diagrams for prematurely slave-terminated synchronous burst-write and burst-read cycles are shown in Figure 3-12.



Parameter	Master min	Master max	Slave min	Slave max	Description
14	0	–	20	–	AD<31..00>* and PAR* float to BERR* high (slave shall quit driving AD<31..00>* and PAR* before unasserting BERR*).
15	0	–	0	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable after BERR* high (master shall not change these lines until the slave unasserts BERR*).
17	0	–	0	–	BERR* high to AD<31..00>* and PAR* driven by master (master shall wait for slave to unassert BERR* before driving AD<31..00>* and PAR*).
39	0	–	0	–	BERR* low to AS* high (master shall not unassert AS* until slave has asserted BERR* unless the master is in the process of terminating the transfer for some other reason).
40	0	–	0	–	BERR* high after AS* high and DS* high whichever comes later (slave shall wait for master to unassert AS* and DS* before unasserting BERR*). Note: DS* may not be asserted.

Note: All times are in nanoseconds as measured/observed on the MXIbus.

Figure 3-12. Prematurely Slave-Terminated Synchronous-Burst Cycle Timing

3.1.2.4 Indivisible Transfers

Indivisible transfers, like block-mode transfers, are cycles that cannot be interrupted by other MXIbus masters because the current master keeps

the AS* signal asserted throughout the operation. However, because the address modifier codes for indivisible transfers are the same as for the basic data transfers, and because block-mode address modifier codes are not specified during indivisible operations, the addressed MXIbus slave will not increment the address in its address counter at the end of each data transfer cycle as it normally would during a MXIbus block-mode transfer. Therefore, all subsequent cycles will be made to the same address in MXIbus memory space. Although any number of read and write transfers can be made in any order during an indivisible operation, the most useful cycle combinations are read-modify-write operations and channel I/O operations.

During read-modify-write operations, a master reads a memory location, modifies the contents, and writes the result back to the same location in memory. Read-modify-write operations can establish semaphores in global system memory. Semaphores allow multiple processes to access shared resources in a consistent way.

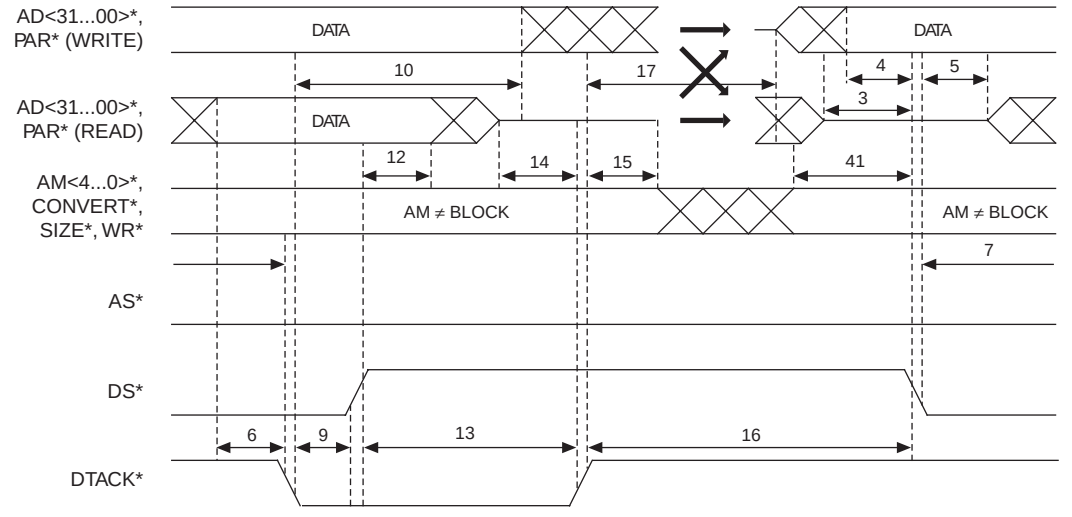
Channel I/O accesses are transfers of the same type (either read or write) to the same location in system memory. Transfers of this type are useful when a FIFO or I/O device in system memory needs to be filled or emptied quickly without the overhead normally associated with broadcasting addressing information for each MXIbus cycle.

Because all indivisible transfers are made to the same location in MXIbus memory space, 8-bit (byte) transfers use the same byte lane throughout the indivisible transfer operation. The byte lane that is used depends on the state of the two least significant address lines that were latched during the address broadcast (see Table 3-4).

Support for indivisible transfer cycles is optional for MXIbus devices. The MXIbus master is responsible for ensuring that indivisible transfers are initiated only with slave devices that support indivisible operations. If an indivisible operation is attempted with a slave that does not support indivisible transfers, the slave may ignore the cycle, or it may immediately terminate the transfer by asserting the BERR* signal.

The MXIbus master device generates the PAR* signal during any write data phase of an indivisible transfer. A MXIbus slave that detects invalid parity during any write data phase of an indivisible transfer should either respond with a BERR* or ignore the transfer and let the bus timeout unit terminate the transfer. The MXIbus slave device generates the PAR* signal during any read data phase of an indivisible transfer. A MXIbus master that detects invalid parity during a read data phase of an

indivisible transfer should treat the data received as corrupt and handle the error appropriately.



Parameter	Master		Slave		Description
	min	max	min	max	
3	20	–	0	–	AD<31..00>* and PAR* float to DS* low (master shall remove address and PAR* from bus before allowing slave to drive them). Note: this applies only when next cycle is a read.
4	70	–	50	–	AD<31..00>* and PAR* valid/stable to DS* low (write data setup).
5	0	–	0	–	DS* low before data driven on the bus (slave shall not drive AD<31..00>* and PAR* until it receives DS* low).
6	0	–	20	–	AD<31..00>* and PAR* valid/stable before DTACK* low (read data setup).
7	0	–	0	–	DS* low to DTACK* low (slave shall wait for DS* to be asserted before asserting DTACK*).
9	0	–	0	–	DTACK* low to DS* high (master shall wait for slave to assert DTACK* before unasserting DS*).
10	0	–	0	–	AD<31..00>* and PAR* valid/stable after DTACK* low (write data hold).
12	0	–	0	–	AD<31..00>* and PAR* valid/stable after DS* high (read data hold).
13	0	–	0	–	DS* high to DTACK* high (slave shall wait for master to unassert DS* before unasserting DTACK*).

Figure 3-13. Indivisible Cycle Timing

Parameter	Master		Slave		Description
	min	max	min	max	
14	0	–	20	–	AD<31..00>* and PAR* float before DTACK* high (slave shall quit driving AD<31..00>* and PAR* before unasserting DTACK*).
15	0	–	0	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable after DTACK* high (master shall not change these lines until slave unasserts DTACK*).
16	0	–	0	–	DTACK* high to DS* low (master shall not assert DS* again until slave has unasserted DTACK*).
17	0	–	0	–	DTACK* high to AD<31..00>* and PAR* driven by master (master shall wait for slave to unassert DTACK* before driving AD<31..00>* and PAR*). Note: this applies only when previous cycle was a read.
41	70	–	50	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable before DS* asserted.
Note: All times are in nanoseconds as measured/observed on the MXIbus.					

Figure 3-13. Indivisible Cycle Timing (Continued)

3.1.2.5 Priority-Selection Cycles

Priority-selection cycles are transfers in which the current MXIbus master can transparently select the highest priority responding MXIbus slave for a read or write cycle access without prior knowledge of that slave's address. Priority-selection cycles are useful for implementing cycles in which multiple slaves may want to participate, such as interrupt acknowledge and dynamic configuration cycles. Support for priority-selection cycles is optional for MXIbus devices, but must be provided for by the MXIbus System Controller.

Interrupt acknowledge cycles can allow a MXIbus master to determine the source and/or cause of a pending interrupt request. During an interrupt acknowledge cycle, the MXIbus master that is configured as the interrupt handler can use priority-selection cycles to cause the highest priority interrupter to source its vector onto the data bus in response to a read request. Through dynamic configuration, a MXIbus master can configure the MXIbus system resources dynamically, so that switches and/or jumpers are not required to establish shared system resources prior to an application's initialization. Priority-selection cycles ensure that only one device at a time accepts configuration information during access cycles from a master during dynamic configuration.

A priority-selection cycle is denoted by a special code on the address modifier lines. A master starts a priority-selection cycle by driving the AM<4..0>* lines to logical 12 hex and performing a read or write transfer of the desired size. The master determines the size of the access by controlling the SIZE* and AD0* lines, as in a normal cycle. The remaining address lines (AD<15..1>*) can be used to further qualify (or pass additional information to) the responding slaves.

Because any number of slaves may want to respond to a particular priority-selection cycle, an automatic selection mechanism is defined, which is a means of selecting only one active slave at a time. The MXIbus System controller provides this selection mechanism, which is invoked whenever the System Controller detects the priority-selection cycle address modifier code and a data cycle access (DS* asserted).

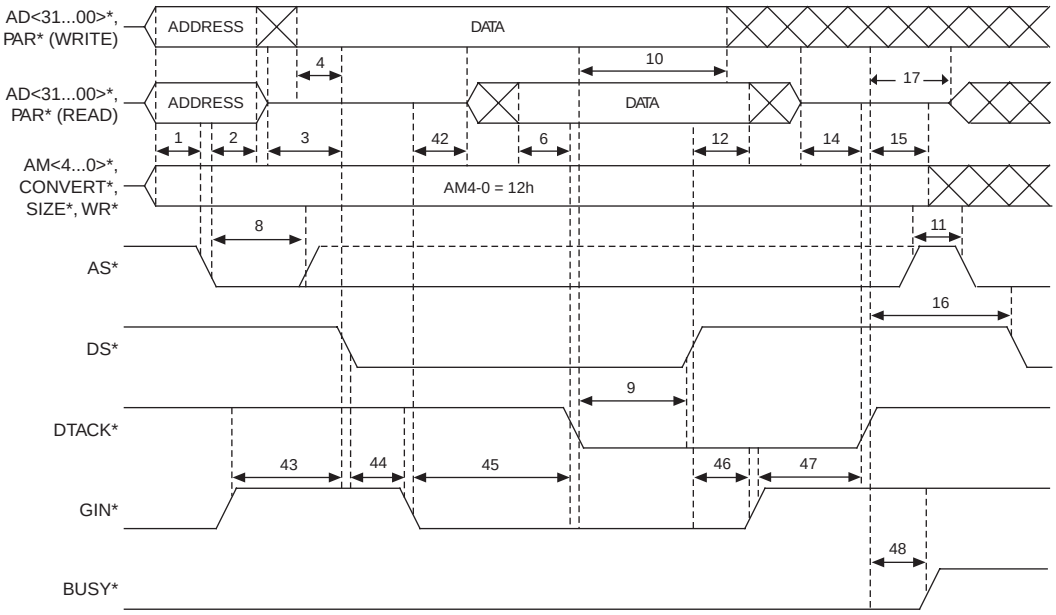
During a priority-selection cycle, the System Controller will start a serial daisy-chain signal by asserting its GOUT* signal. GOUT* propagates down the MXIbus daisy-chain to the next device's GIN* input. If that device is not a potential slave of the cycle, it passes the daisy-chain signal to the next device via its GOUT* signal. When the daisy-chain signal is received by a slave that is a potential responder to the cycle request, the slave can accept the cycle by asserting its DTACK* signal and not pass the daisy-chain grant signal to the next device.

Because of the nature of the daisy-chain signal, devices are prioritized by their relative position within the daisy-chain. The MXIbus device closest to the System Controller has the highest priority and, if participating, will be selected before a device farther from the System Controller that has a lower priority.

Notice that the GOUT* and GIN* signals serve a dual purpose. During priority-selection cycles, the signals select the highest priority responding slave. During arbitration, GOUT* and GIN* select the highest priority requesting master. It is necessary to determine the intended function of the daisy-chain before determining the proper action for a device to take. For example, a master requesting the MXIbus should not assume ownership of the bus if it receives a GIN* due to a priority-selection cycle. Priority-selection cycles can be distinguished from arbitration cycles by the state of the BUSY* signal or by the code on the address modifier lines with DS* is asserted. During a priority-selection cycle, the master always asserts the BUSY* signal and drives a logical 12 hex onto the address modifier lines AM<4..0>*.

The priority-selection cycle daisy-chain circuit must be carefully designed. When a device receives GIN* active during a priority-selection cycle, the device shall determine whether to respond to the cycle request, or to pass the grant down the daisy-chain by driving GOUT* active. Because slave participation in a priority-selection cycle may be decided asynchronously with respect to the GIN* input, it is possible for a slave to receive a GIN* signal at the same time that the slave recognizes that it should respond to the access. The circuitry on the slave must determine whether to pass the daisy-chain or to acknowledge the cycle request. In either case, the daisy-chain circuitry shall be designed so that the slave does not momentarily glitch the GOUT* line.

The MXIbus master device generates the PAR* signal during any write data phase of a priority-selection cycle. A MXIbus slave that detects invalid parity during a write data phase of a priority-selection cycle should either respond with a BERR* or ignore the transfer and let the bus timeout unit terminate the transfer. The MXIbus slave device generates the PAR* signal during any read data phase of a priority-selection cycle. A MXIbus master that detects invalid parity during a read data phase of a priority-selection cycle should treat the data received as corrupt and handle the error appropriately.



Parameter	Master max	Master min	Slave max	Slave min	Description
1	70	–	50	–	AD<31..00>*, PAR*, AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable before AS* low (address setup).
2	50	–	30	–	AD<31..00>* and PAR* valid/stable after AS* low (address hold).
3	20	–	0	–	AD<31..00>* and PAR* float to DS* low (master shall remove address and PAR* from bus before allowing slave to drive them). Note: this applies to read cycles only.
4	70	–	50	–	AD<31..00>* and PAR* valid/stable to DS* low (write data setup).
6	0	–	20	–	AD<31..00>* and PAR* valid/stable before DTACK* low (read data setup).
8	70	–	50	–	Minimum AS* assertion time.
9	0	–	0	–	DTACK* low to DS* high (master shall wait for slave to assert DTACK* before unasserting DS*).
10	0	–	0	–	AD<31..00>* and PAR* valid/stable after DTACK* low (write data hold).
11	70	–	50	–	Minimum AS* unassertion time.

Figure 3-14. Priority-Selection Cycle Timing

Parameter	Master		Slave		Description
	max	min	max	min	
12	0	–	0	–	AD<31..00>* and PAR* valid/stable after DS* high (read data hold).
14	0	–	20	–	AD<31..00>* and PAR* float before DTACK* high (slave shall quit driving AD<31..00>* and PAR* before unasserting DTACK*).
15	0	–	0	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid after DTACK* high (master shall not change these lines until slave unasserts DTACK*).
16	0	–	0	–	DTACK* high to DS* low (master must wait for slave to unassert DTACK* before asserting DS* for next cycle).
17	0	–	0	–	DTACK* high to AD<31..00>* and PAR* driven by master (master shall wait for slave to unassert DTACK* before driving AD<31..00>* and PAR*).
42	–	–	0	–	GIN* low to AD<31..00>* and PAR* driven. Responding slave must not drive AD<31..00>* and PAR* until it receives GIN* daisy-chain.
43	0	–	–	–	GIN* high to DS* low (master must not assert DS* until GIN* is unasserted).
44	0	–	–	–	System Controller must wait for master to assert DS* before starting new daisy-chain.
45	–	–	0	–	Slave must not drive DTACK* until it receives GIN* daisy-chain.
46	0	–	–	–	System Controller will unassert daisy-chain in response to master unasserting DS*.
47	–	–	0	–	Slave must not unassert DTACK* until its GIN* daisy-chain is unasserted.
48	–	–	0	–	Master must not release the bus until slave unasserts DTACK*.
Note: All times are in nanoseconds as measured/observed on the MXIbus.					

Figure 3-14. Priority-Selection Cycle Timing (Continued)

3.1.2.6 RETRY and BERR* Acknowledgment

The selected slave can acknowledge any MXIbus data transfer with a RETRY or BERR* response rather than the normal DTACK* response. One exception is synchronous-burst cycles. A RETRY acknowledge is allowed only for synchronous-burst cycles in response to the initiation phase. The synchronous-burst data transfer and termination phases cannot be acknowledged with RETRY responses. Also, BERR* acknowledgments during the data transfer and termination phases of synchronous-burst cycles have different timing relationships than all

other MXIbus cycles, as described in Section 3.1.2.3.3. This section applies to basic, block, indivisible, and priority-selection cycles as well as the initiation phase of synchronous-burst cycles.

No data is transferred during a data cycle that is acknowledged with a RETRY or BERR* response. The MXIbus master need not check parity on read cycles that were acknowledged with a RETRY or BERR* response. MXIbus slaves need not store any data when issuing a RETRY or BERR* acknowledgment in response to a write cycle, nor generate any data when issuing a RETRY or BERR* acknowledgment in response to a read cycle.

A RETRY acknowledgment signals the MXIbus master that the selected slave cannot currently complete the requested data transfer but will possibly be able to at a later time. After receiving a RETRY acknowledgment, the MXIbus master should repeat the data transfer. If the BREQ* signal is asserted when a master receives a RETRY acknowledgment, the master must release ownership of the MXIbus before repeating the data transfer, which alleviates deadlock situations.

A BERR* acknowledgment indicates that an error condition exists on the current data cycle. A BERR* acknowledgment indicates that an error has occurred only on the current MXIbus data cycle. It cannot be used as a general-purpose asynchronous error indication because BERR* can be asserted only in response to a DS* from the master. Either the MXIbus utility signals SYSFAIL* and ACFAIL* or the MXIbus IRQ<7..1>* signals can report asynchronous error conditions, such as a power failure indication.

Notice that a BERR* acknowledgment indicates only that an error occurred, and fails to indicate the source of the problem. Further device interrogation can determine the source of the problem, but is not specified or required by this document. A master that receives a BERR* acknowledgment may optionally retry the bus cycle, terminate normal operation, attempt to determine the cause of the BERR*, or perform any other appropriate action. Although BERR* conditions may be due to any number of different situations, only one of two devices may assert the BERR* signal in response to a data cycle from the master: the MXIbus System Controller, or the addressed slave.

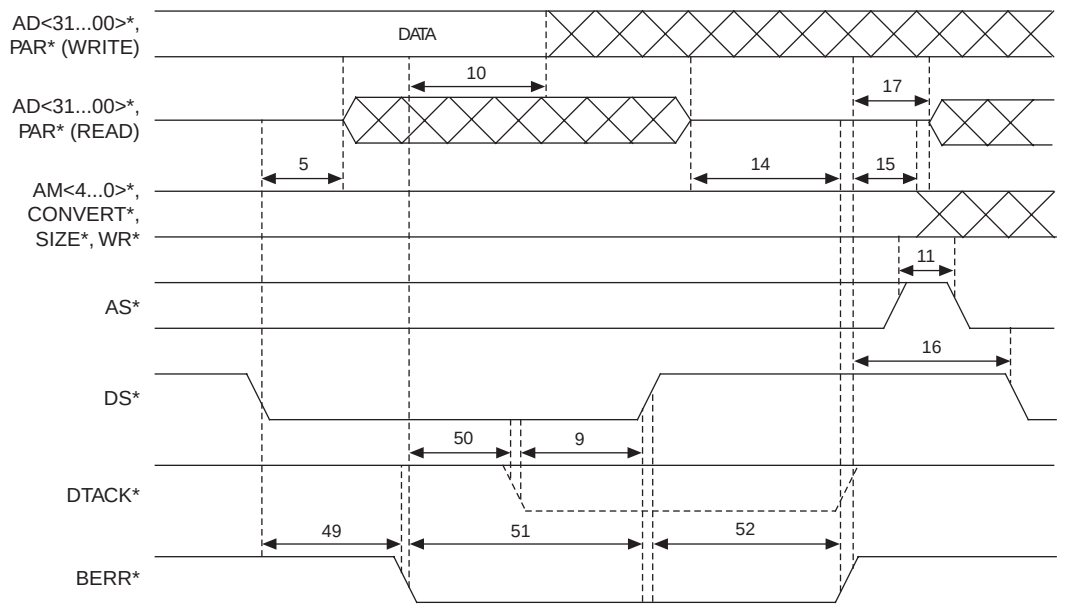
The MXIbus System Controller acts as a bus monitor for timeout conditions in addition to its bus arbitration and priority-selection cycle daisy-chain duties. If a MXIbus master begins a transaction and no device responds to complete the transaction, the bus monitor circuitry

will time out (after a specified delay) and the MXIbus System Controller shall assert the BERR* line to allow the current cycle to complete (that is, the System Controller monitors the length of time that DS* is asserted). The timeout delay is system dependent. For normal operation, a delay of 1 ms is recommended. It is also recommended that devices that implement MXIbus System Controller functionality allow the timeout to be modified to a value that best suits the requirements of the system. The System Controller timeout monitor is required and should never be disabled.

MXIbus slaves may respond with a BERR* acknowledgment if a local bus error is detected, if a parity error is detected on write data, or if the transfer requested by the MXIbus master is not possible (for example, a 32-bit transfer request to a 16-bit slave, or a write request to read-only memory). MXIbus slaves may also assert BERR* if a master attempts to perform a block, synchronous-burst, or indivisible transfer, and the slave is incapable of doing so. MXIbus slaves may also respond with a BERR* for device-dependent reasons (for example, a write to a full FIFO). BERR* support on MXIbus slaves for any condition is optional.

A RETRY acknowledgment consists of the slave asserting the BERR* signal shortly before asserting the DTACK* signal. Masters detect RETRY acknowledgments by sensing both BERR* and DTACK* asserted simultaneously. To prevent loss of performance during successful data transfers, if a master senses DTACK* asserted without BERR* asserted, the signal is immediately considered a DTACK* acknowledgment and the data transfer is successful. If a master senses the BERR* signal asserted, it must wait a prescribed amount of time and sample the DTACK* signal. If DTACK* is then also asserted, it is a RETRY response. If only BERR* was asserted, it was a BERR* acknowledgment.

During a RETRY response, the slave must keep both the BERR* and DTACK* signals asserted until the master unasserts DS*. During a BERR* response, the slave must keep the BERR* signal asserted until the master unasserts DS*. A master device that does not support the MXIbus RETRY response will interpret both RETRY and BERR* responses as BERR* responses.



Parameter	Master min	Master max	Slave min	Slave max	Description
5	0	–	0	–	DS* low before data driven on the bus (slave shall not drive AD<31..00>* and PAR* until it receives DS* low during a read cycle).
9	0	–	0	–	DTACK* low to DS* high (master shall wait for slave to assert DTACK* before unasserting DS*). Note: DTACK* is not asserted for a BERR* acknowledgment.
10	0	–	0	–	AD<31..00>* and PAR* valid/stable after BERR* low (write data hold).
11	70	–	50	–	Minimum AS* unassertion time.
14	0	–	20	–	AD<31..00>* and PAR* float before BERR* and DTACK* high (slave shall quit driving AD<31..00>* and PAR* before unasserting both BERR* and DTACK*).
15	0	–	0	–	AM<4..0>*, CONVERT*, SIZE*, and WR* valid/stable after BERR* and DTACK* high (master shall not change these lines until slave unasserts both BERR* and DTACK*).
16	0	–	0	–	BERR* and DTACK* high to DS* low (master must wait for slave to unassert both BERR* and DTACK* before asserting DS* for next cycle).

Figure 3-15. RETRY and BERR* Acknowledgment

Parameter	Master		Slave		Description
	min	max	min	max	
17	0	–	0	–	BERR* and DTACK* high to AD<31..00>* and PAR* driven by master (master shall wait for slave to unassert BERR* and DTACK* before driving AD<31..00>* and PAR*).
49	0	–	0	–	DS* low to BERR* low (slave must wait for master to assert DS* before asserting BERR*).
50	50	220	70	200	BERR* low to DTACK* low (slave must wait this amount of time after asserting BERR* before asserting DTACK*). Note: DTACK* is not asserted for a BERR* acknowledgment.
51	220	–	200	–	BERR* low to DS* high (master must wait this amount of time after detecting BERR* low before unasserting DS*). This is to allow the slave time to assert DTACK* for a RETRY response.
52	0	–	0	–	DS* high to BERR* and DTACK* high (slave must wait for master to unassert DS* before unasserting BERR* and DTACK*). Note: DTACK* is not asserted for a BERR* acknowledgment.
Note: All times are in nanoseconds as measured/observed on the MXIbus.					

Figure 3-15. RETRY and BERR* Acknowledgment (Continued)

3.2 Arbitration

All MXIbus masters shall have bus request logic to be granted use of the MXIbus. MXIbus devices that have MXIbus System Controller circuitry shall have bus arbiter logic to grant use of the bus to requesting masters. The MXIbus arbitration signals are bus busy (BUSY*), bus request (BREQ*), grant in (GIN*), and grant out (GOUT*). MXIbus supports serial, release-on-request bus arbitration with both fairness and bus lock options.

In a serial arbitration scheme, requesting devices signal their need for the bus by asserting the bus request signal BREQ*. BREQ* is a wired-OR signal that indicates that one or more MXIbus devices are requesting use of the bus. The MXIbus System Controller contains the bus arbiter, which detects when BREQ* is active (and BUSY* is not active) and grants the bus to requesters by asserting its GOUT* signal. GOUT* propagates down the MXIbus daisy-chain to the next device's GIN* signal. If a device is not currently requesting use of the bus, it passes the daisy-chain signal to the next device via its GOUT* signal. When the daisy-chain signal is received by the highest priority requesting device, that device is granted the bus and it will not pass the daisy-chain grant signal to the next device.

Once a requester has been granted control of the bus by way of its GIN* signal, it drives BUSY* active, unasserts BREQ*, and signals to the onboard logic that it has gained control of the bus. Driving BUSY* active indicates to all other MXIbus devices that the bus is now busy. The release-on-request feature of the MXIbus means that the current bus master does not release the bus (by unasserting BUSY*) until another MXIbus device requests the bus by driving BREQ* low. This scheme reduces the number of arbitrations required if a single MXIbus master uses a large percentage of the bus bandwidth. No arbitration may occur on the MXIbus as long as BUSY* is held active by the current master. When the current owner is finished using the bus and detects another device driving the BREQ* line active, it unasserts BUSY*, enabling the arbiter to grant the bus to another pending requester.

Because of the nature of the daisy-chain arbitration scheme, requesters are prioritized by their relative position within the daisy-chain. The requester closest to the MXIbus System Controller has the highest priority, whereas the requester farthest from the System Controller has the lowest priority. If a single device is continuously requesting the bus, it is possible that a device with a lower priority will never be serviced. It is suggested that requesters be designed with a jumper-selectable or programmable arbitration-fairness option. If fairness is enabled, the current master shall refrain from driving BREQ* active after releasing it until BREQ* is detected as inactive for a minimum amount of time. This scheme ensures that all requesting devices will be granted use of the bus. If the fairness option is disabled, the MXIbus will have a fixed priority (set by the bus grant daisy-chain cabling order) and devices closest to the System Controller will always have higher priority than devices farther away from the System Controller.

Notice that a situation exists in which a glitch can occur on the BREQ* line that could occasionally defeat the fairness option if not taken into consideration. Because two or more devices can assert the BREQ* signal at the same time, it may appear that if one device released BREQ*, the other device(s) would continue to keep the signal asserted properly. However, due to differences in driver saturation voltages and ground potentials, one driver may be sinking most or all of the current required to keep the BREQ* signal at a low level. When this driver releases the signal, a high-going “wire-OR glitch” pulse may race down the cable until the other driver(s) can sink the additional current load. At this time, a low signal can again be established, which travels back down the cable to all the devices receivers. This could cause a false glitch of up to 200 ns (5 ns/m * 20 m * 2) to appear on the BREQ* line, which could

occasionally defeat the fairness option. If a device supports the fairness option, it shall be designed to ignore or filter out any wire-OR glitch condition that could occur, so that the fairness algorithm functions properly.

MXIbus masters may lock the MXIbus for an indefinite amount of time by simply continuing to assert BUSY*. This ensures indivisible operations on the MXIbus because it prohibits further arbitration and eliminates possible interruptions from other MXIbus masters. Notice that block, synchronous-burst, and indivisible transfers automatically lock the MXIbus, because AS*, as well as BUSY*, remains asserted throughout the transfer. Because no other master may use the bus when it is locked, it is suggested that the MXIbus be locked only when absolutely required, and only for as short a tenure as is necessary.

If a MXIbus device does not have master capabilities, it is not required to have logic to drive the BREQ* or BUSY* signals; however, all devices shall propagate the daisy-chain signal from the GIN* line to the GOUT* line properly during bus arbitration.

Notice that the GIN* and GOUT* signals serve a dual purpose. During priority-selection cycles, GIN* and GOUT* select the highest priority responding slave. During arbitration, they select the highest priority requesting master. It is necessary to determine the intended function of the daisy-chain before determining the proper action for a device to take. For example, a master requesting the MXIbus should not assume ownership of the bus if it receives a GIN* due to a priority-selection cycle. Priority-selection cycles can be distinguished from arbitration cycles by the state of the BUSY* signal or by the code on the address modifier lines when DS* is asserted. During a priority-selection cycle, the master always asserts the BUSY* signal and drives a logical 12 hex onto the address modifier lines AM<4..0>*.

The MXIbus arbitration steps are as follows:

1. A MXIbus master requests the MXIbus by asserting the BREQ* line.
2. The arbiter on the MXIbus System Controller detects BREQ* asserted and BUSY* unasserted and asserts its GIN* pin (or its GOUT* pin, if the System Controller device does not have master circuitry).
3. The first requester in the daisy-chain that receives GIN* active on the bus grant daisy-chain due to an arbitration cycle and is asserting BREQ* wins the bus. This device asserts BUSY* and unasserts BREQ*. Devices that are not asserting BREQ* and receive an active

GIN* signal will continue the bus grant daisy-chain by driving GIN* onto their GOUT* line.

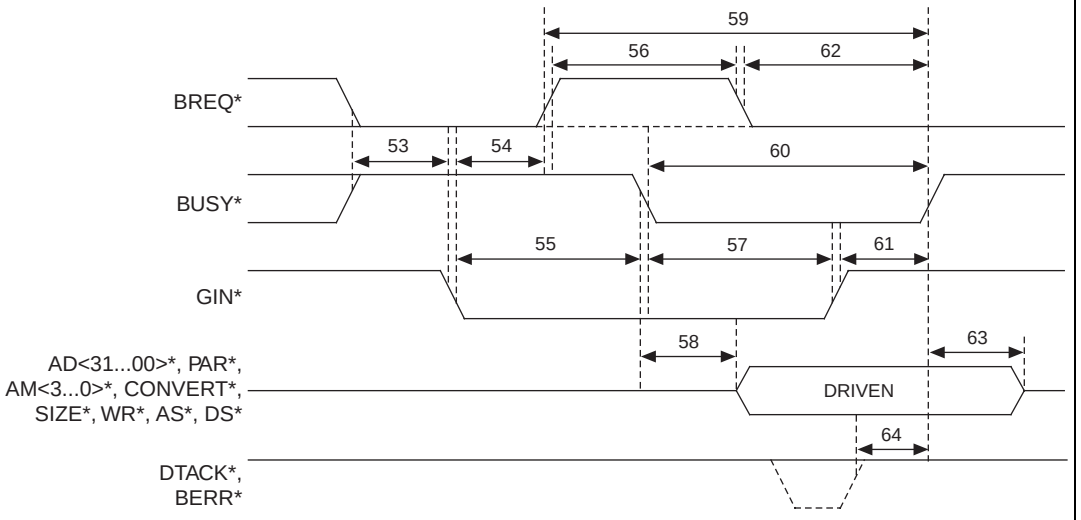
4. When the arbiter detects BUSY* asserted, it unasserts its GIN* pin.
5. The requester continues to assert BUSY* until it finishes its bus transactions and detects that the MXIbus BREQ* line is asserted (that is, until another requester wants control of the bus). If fairness is enabled, the requester shall not assert BREQ* again until it detects BREQ* unasserted on the MXIbus.
6. The arbiter interprets the unassertion of BUSY* as the signal to start the arbitration process again.

The arbitration daisy-chain circuit design must be handled carefully. When a requester receives GIN* active during arbitration, it must determine whether to drive BUSY* active and assume control of the bus, or to pass the bus grant down the daisy-chain by driving GOUT* active. Because MXIbus bus requests are asynchronous with respect to the GIN* input, it is possible for a MXIbus device to receive a GIN* at the same time that the device recognizes that it needs the MXIbus. The circuitry on the MXIbus device must determine whether to pass the bus grant down the daisy-chain or to assume ownership of the MXIbus. In either case, the bus grant daisy-chain circuitry shall be designed so that the requester does not momentarily glitch the GOUT* line.

In the event that a bus grant is issued by the arbiter and no device assumes ownership (that is, BUSY* is never asserted) and BREQ* remains asserted, the MXIbus will hang indefinitely. This condition is a fatal system error, and means that one or more of the MXIbus devices or cables is inoperative and must be replaced, repaired, or removed before system operation can continue.

If a device requires bus ownership and asserts BREQ*, but later (and before it is granted ownership) no longer requires the bus, the device shall continue to assert BREQ* until it receives a GIN* signal. Once it is granted ownership, the circuitry asserts BUSY*, unasserts BREQ*, then unasserts BUSY* (that is, it assumes ownership but performs no bus transactions).

Notice that in the following timing diagrams, *arbiter* refers to the timing as seen on the MXIbus at the arbiter's device connector; *requester* refers to the timing as seen on the MXIbus at the requester's device connector.



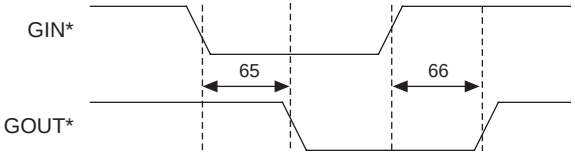
Parameter	Arbiter min	Arbiter max	Requester min	Requester max	Description
53	0	–	–	–	BREQ* low and BUSY* high to GIN* low (arbiter asserts GIN* after receiving BREQ* low and BUSY* high).
54	0	–	0	200	GIN* low to BREQ* high (requester unasserts BREQ* after receiving GIN* low). Note: BREQ* could remain low on the MXIbus if another device is asserting it.
55	0	–	0	–	GIN* low to BUSY* low (requester asserts BUSY* after receiving GIN* low).
56	–	–	200	–	BREQ* high to fair requester reasserting BREQ* (required only for fair masters).
57	0	200	0	–	BUSY* low to GIN* high (arbiter unasserts GIN* after receiving BUSY* low).
58	–	–	0	–	BUSY* low to winning requester driving MXIbus.
59	0	–	20	–	BREQ* high to BUSY* high (current bus owner shall unassert BREQ* before unasserting BUSY*).
60	80	–	100	–	BUSY* assertion time.
61	0	–	0	–	GIN* high to BUSY* high (current bus owner shall not unassert BUSY* until it detects that GIN* has been unasserted).
62	0	–	0	–	BREQ* low to BUSY* high (current bus owner shall not release the bus until detecting BREQ* asserted and it no longer needs the bus).

Figure 3-16. Arbitration Requester Timing

Parameter	Arbiter min	Arbiter max	Requester min	Requester max	Description
63	–	–	–	80	BUSY* high to current bus owner off MXIbus.
64	0	–	0	–	DTACK* and BERR* high to BUSY* high (current bus owner shall not release the bus until both DTACK* and BERR* are unasserted).

Note: All times are in nanoseconds as measured/observed on the MXIbus.

Figure 3-16. Arbitration Requester Timing (Continued)



Parameter	Arbiter min	Arbiter max	Requester min	Requester max	Description
65	–	–	0	–	GIN* low to GOUT* low (device is not requesting bus and is simply passing the grant on to the next device).
66	–	–	0	–	GIN* high to GOUT* high.

Note: All times are in nanoseconds as measured/observed on the MXIbus.

Figure 3-17. Grant In/Grant Out Daisy-Chain Timing

3.3 Interrupt

The MXIbus interrupt architecture allows MXIbus devices to interrupt each other by asserting one of the open-collector IRQ<7..1>* (interrupt request) lines. Any MXIbus device can assert and/or monitor any of the IRQ<7..1>* lines for any device-specific reason. A MXIbus device that is programmed to monitor an IRQ<7..1>* line is called an interrupt handler and can determine the source and/or cause of the pending interrupt request by performing an interrupt acknowledge cycle (see Section 3.1.2.5). The interrupt handler, upon reading the status vector, can perform any desired operation to service the pending interrupt. The IRQ<7..1>* signals have no fixed timing relationship to any other MXIbus signal.

3.4 Timing

The MXIbus defines one clock (CLK10±) and eight trigger (TRG<7..0>±) differential signals for timing purposes. These signals can be used for any device-specific purpose. Because these signals are differential, each signal can be driven from only one MXIbus device at a time. The TRG<7..0>± signals can be used synchronously with the CLK10± signal for the most precise timing requirements. The CLK10± and TRG<7..0>± signals have no fixed timing relationship to any other MXIbus signal.

3.5 Utility

The MXIbus defines three types of utility signals: system status, bus timeout disable, and cabling detection.

The MXIbus system status signals consist of SYSRESET*, SYSFAIL*, and ACFAIL*. The SYSRESET* signal can be used as a system-wide reset. A MXIbus device can use the SYSFAIL* signal to indicate a functional failure. A MXIbus device can use the ACFAIL* signal to indicate a power failure. Any MXIbus device can react or drive these signals for device-specific purposes. The system status signals have no fixed timing relationship to any other MXIbus signal.

The MXIbus defines a single bus timeout disable (BTODIS*) signal. When this signal is driven low, the MXIbus System Controller should disable the bus timeout unit so that any pending data cycle will not be timed. A MXIbus slave device could assert this signal to disable the MXIbus timer when it needs to pass bus timeout responsibilities to another bus. Only the MXIbus slave that was selected by the current data cycle should assert this signal. A slave that asserts this signal is responsible for terminating the cycle with an acknowledgment (DTACK*, BERR*, or RETRY). The BTODIS* signal has no fixed timing relationship to any other MXIbus signal.

The MXIbus defines two cabling-detection signals. MXIbus devices use these signals to detect where the device is relative to other devices in the MXIbus system. Any device that uses these signals should terminate them onboard to ensure proper operation.

The first signal is MXIbus System Controller (MXISC*). When a device receives this signal low, it indicates that that device is the first in the MXIbus system and should take on the System Controller responsibilities.

The second signal is End Device (ENDDEV). When a device receives ENDDEV high, it indicates that the device is at the end of the MXIbus system and should terminate the bus signals as described in Section 2.

A device that detects MXISC* low should also activate its MXIbus termination, because the ENDDEV signal is invalid when MXISC* is low due to the scheme used. In other words, a MXIbus device should terminate the MXIbus if it receives MXISC* low *or* ENDEV high. These two signals allow MXIbus devices to have automatic System Controller and MXIbus termination circuitry to ease system configuration. The cabling-detection signals have no fixed timing relationship to any other MXIbus signal. For more information on how to use these signals, refer to the *System Configuration* part of Section 2.

Device-Dependent Considerations

Section 4

There are several device-dependent issues related to how devices interact with each other to consider when designing a MXIbus device interface. Careful consideration can greatly aid in the system integration task. Some of the more important device-dependent considerations are described in the following sections.

4.1 Deadlock

Many MXIbus devices will be computers or intelligent devices that have their own internal independent bus structures, as well as the ability to become a bus master through local arbitration. MXIbus is itself an independent bus structure with its own independent arbiter. This can be very beneficial because multiple systems can operate in parallel and are required to synchronize their activity only when they access shared resources. However, this arbitration independence does bring up the potential for an interesting problem known as deadlock. *Deadlock* is a condition that exists when multiple masters attempt to access each other's resources simultaneously across an independent bus structure. Because neither master can complete its operation until it is able to access the other's bus, the system, in effect, becomes hung.

For example, consider the system shown in Figure 4-1. Assume that the local master is attempting to access the remote slave device across the MXIbus. It must first arbitrate and win control of the local bus. The local MXIbus link, upon determining that the cycle is intended for an external MXIbus device, will arbitrate for and win control of the MXIbus before starting the MXIbus cycle.

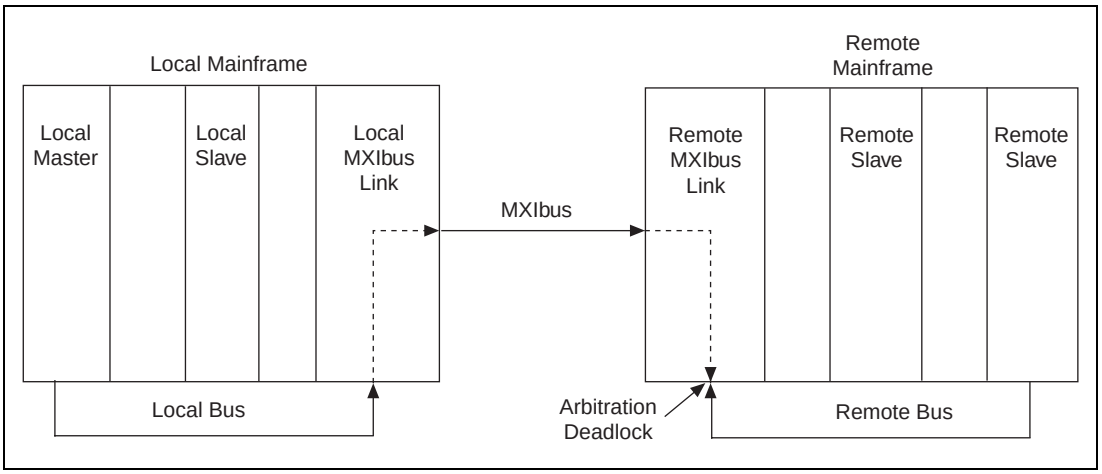


Figure 4-1. Data Transfer Bus Arbitration Deadlock

In response to the MXIbus cycle that maps into its frame, the remote MXIbus link generates a request for the remote bus via the remote bus arbiter. Now assume that at the same time the remote MXIbus link requests the remote bus, the remote bus master also requests the remote bus for a different cycle that is to map from the remote bus master to the local slave via the MXIbus. If the remote master wins control of the remote bus before the remote MXIbus link, the system will lock up because neither of the outstanding cycles can complete until the other is satisfied.

Unfortunately, deadlock situations are unavoidable in systems where multiple masters are allowed to asynchronously access each other's resources without a global arbitration mechanism. If a deadlock condition is detected, the remote MXIbus link shall prematurely terminate the remote master's remote bus cycle, resolving the deadlock. Then the local MXIbus link that owns the MXIbus can complete the cycle access. The terminated cycle can then be re-attempted.

The method of terminating an ongoing transfer differs from system to system but usually involves asserting some sort of bus error or retry signal, both of which MXI supports. Ideally, when the active master receives the bus error/retry indication, it latches the current state of the machine, terminates the cycle, and immediately re-attempts the failed cycle using the stored information. Unfortunately, not all processors and/or bus structures support automatic exception-handling techniques or have automatic retry support. Without automatic support, it may be necessary to include an exception-handling routine as an integral part of

the system software. This routine would then be responsible for re-executing the prior instruction after a deadlock condition.

Because software exception-handling techniques can affect the transparency of the system, the MXIbus arbiter can, in most cases, be designed to optionally function as the global system-wide arbiter. In this mode, the MXIbus arbiter ensures that only one system-wide master is active at a time, eliminating deadlock conditions. Although this prohibits parallel processing, it allows previously developed software application programs to run without modification in an extended system that does not support automatic exception-handling.

The MXIbus arbiter can function as the global system-wide arbiter by *interlocking* the arbitration mechanisms of all bus structures in the system. Therefore, when a master wins arbitration, it has won control not only of its immediate bus, but also of all the bus structures in the entire expanded system. For this technique to work, all participating arbiters must be able to interrupt the normal arbitration sequence of their respective bus structures to pass the arbitration indicator throughout the rest of the system. This generally requires some sort of serial arbitration mechanism.

4.2 Byte Swapping

MXIbus, like the VMEbus, specifies the active byte lane(s) for all data widths supported, but foregoes defining the significance of the data on these byte lanes. By concerning itself only with which byte lanes are used with the proper control lines, and by ignoring the significance of the data, MXIbus remains processor-independent and uncommitted to choosing a data format. As long as all the microprocessors and bus structures in the system obey the same rules regarding data significance and byte-lane ordering, nothing needs to be done. If, however, different processors are interconnected or the byte-lane ordering of the bus structure is inconsistent with the way MXIbus defines it, then data bytes may need to be moved around or swapped at some level so that data is represented correctly between systems.

If data bytes need to be swapped due to data significance incompatibilities, the system software can perform this function before it uses or stores the values to memory. Swapping data in software can be time consuming and is generally not a good solution for high-performance systems.

A much faster approach is to swap the data bytes using hardware. Although this may at first appear to be a relatively straightforward task, it can become quite involved due to the number of possible combinations of byte-lane crossings that can exist in an expanded system. The byte-swapping problem cannot be solved by simply reversing the order of the byte lanes, because there may be situations when you do not want to swap data. For example, although dissimilar processors may disagree on the way multibyte values are stored, they generally agree on the way character strings are represented in memory.

Unfortunately, because it is not always evident when data needs to be swapped, there are no software-transparent solutions to this problem. If required, byte-swapping hardware can be designed so that the system software can control the swapping function. In this way, the system programmer has control over the way data will be represented between the dissimilar systems.

4.3 Memory Management

MXIbus devices communicate by mapping portions of their address space into global system memory. Locations within global system memory are specified by the MXIbus address bus. If the location of the local address space of a MXIbus device conflicts with that of other MXIbus devices in global system memory, then one of the device's memory space must be moved, or mapped, to an available location within the MXIbus address space to avoid conflicts.

MXIbus devices can map their memory through onboard registers or switches into MXIbus address space in any device-dependent manner. The mapping function need only involve the proper selection of the most significant address lines that uniquely specify the selection of a device, but can involve more complex memory mapping functions. A flexible mapping function allows the shared memory space of a device to be moved around in global MXIbus memory to match the specific requirements of a system.

4.4 Resource Locking

In some situations, a MXIbus master may need exclusive use of the MXIbus for more than one cycle, and will need to lock the MXIbus. MXIbus locking is accomplished with no additional mechanisms or protocols. To lock the MXIbus, a MXIbus master simply continues to assert BUSY* or AS* throughout the operation.

An additional form of locking is required if other local bus masters can also access the selected MXIbus slave resource via a separate, independent local bus on the slave device. This additional locking system is called *resource locking*, and may involve additional protocol over simple MXIbus bus locking.

If the system is operating using the interlocked arbitration scheme, then resource locking is ensured because there can be only one system-wide master active at a time. The MXIbus slave hardware can also automatically handle resource locking when it detects that the MXIbus has been locked (AS* remains asserted for multiple cycles).

If neither of these techniques is a practical alternative, implementing a lock register on the MXIbus slave device can accomplish resource locking. Each data sequence that would need to be resource-locked can then be "framed" with command accesses to the slave's onboard lock register, which allows the MXIbus master to manually lock and unlock the remote slave resources in a device-specific manner.

4.5 Interface Registers

The MXIbus specification does not require that MXIbus devices implement any particular register set for device configuration and/or device-to-device communication. Thus, the designer has the flexibility to decide if MXIbus-accessible registers are necessary in a given application and, if so, the freedom to choose a register set that best meets the system's price, performance, and functional requirements.

If a general-purpose set of MXIbus interface registers is desired, one register set that offers a standardized interface approach with a wide range of functionality and is an excellent match with the general requirements of the MXIbus architecture, is the VXIbus specification register set. Although a detailed discussion of the VXIbus register set is beyond the scope of this specification, a short discussion of the basic functionality is given in the following paragraphs.

The VXIbus register set is a standardized set of communication and configuration registers that can accommodate up to 256 devices in a system. Each device has up to 64 bytes of address space for its register needs, and all device registers are located in the top quarter of A16 space. Devices can support a variety of register formats, depending on their functional and cost requirements.

The most basic register format is for Register-Based devices, and defines a simple set of four registers that support device identification and configuration. A more advanced register format is available for Message-Based devices, and adds a full set of communication registers, which allow devices to communicate using high-level protocols (message passing) along an established system hierarchy. Message-Based devices also have a signal register for efficient device-to-device signaling. Another register format, defined for Memory-Based devices, produces a more appropriate interface for memory devices that have blocks of sequential data storage in either A24 or A32 space. The final register format, for Extended devices, defines new subclasses of devices for future applications. The *VXIbus Mainframe Extender Specification* establishes a new subclass of Extended device registers, which allows multiple VXIbus mainframes to be connected together transparently.

The VXIbus register set is an organization through which all of the system resources can be configured automatically by a single software module called the Resource Manager. This technique eliminates the need for configuration switches and jumpers on devices. The Resource Manager identifies the device types and manufacturers, manages the device's self-test and diagnostic sequence, allocates A24 and A32 memory to each device based on the addressing requirements, establishes a system-wide communication channel hierarchy to prevent potential conflicts between multiple processors attempting to manage the same device resource, and initiates normal system operations.

The Resource Manager can also dynamically configure each device's register resources by automatically establishing a unique register set location for each device so that potential conflicts of the configuration and communication registers between devices are eliminated.

4.6 Power Up/Down

The MXIbus does not specify any particular power-up or power-down sequences for system initialization or reinitialization. Devices may be powered up or down in any order as long as a MXIbus master does not attempt to access a slave that is not powered up and fully operational. The MXIbus transceivers are guaranteed not to glitch the MXIbus during power-up or power-down; however, there are still several other considerations that need to be addressed by the system integrator if the system is required to remain viable when MXIbus devices are powered down. These considerations are discussed in the following sections.

4.6.1 System Controller

The MXIbus System Controller must never be powered down if the system is to remain operational. The MXIbus System Controller has the system arbiter, the daisy-chain driver for priority-selection cycles, and bus timeout responsibility, which must always remain operational for the MXIbus to function properly.

4.6.2 Terminators

The MXIbus terminators minimize reflections in the MXIbus cable system and bias the signals to their false state when they are not driven active by a device. The terminators are located at the end devices in the MXIbus daisy-chain and must remain powered at all times for the MXIbus to function. Because the MXIbus System Controller will always host one of the termination networks and cannot be powered down during MXIbus operations, the only termination network that the system integrator needs to be concerned with is the one hosted by the last device in the MXIbus daisy-chain (far-end terminator). If the last device in the daisy-chain supplies power to the far-end terminator, then it, like the System Controller, must remain powered up during all MXIbus operations. Optionally, an independent power source can power the far-end terminator, which allows the last device in the daisy-chain to be powered down without affecting MXIbus operations.

4.6.3 Daisy-Chain Propagation

If it is required that the MXIbus link remains operational when devices are powered-down and are left connected to the MXIbus, then those devices must have a mechanism for connecting the GIN* and GOUT* signals so that the daisy-chain remains functional. An electromechanical relay, switch, or other appropriate device can accomplish this.

It is important to note that, although it is possible to power-up and/or power-down devices during MXIbus data transfers, devices should not be powered up or down during arbitration or priority-selection cycles because these cycles use the GIN*-GOUT* daisy-chain. The GIN*-GOUT* daisy-chain may not be stable during power-up or power-down because of the “bouncing” characteristic associated with electromechanical devices. To ensure continued MXIbus operation when powering devices up or down, it is necessary to ensure that an active daisy-chain operation is not ongoing. Any MXIbus master can guarantee that no daisy-chain operation is ongoing by arbitrating for the MXIbus and holding it throughout the power-up and power-down sequence. The

MXIbus System Controller can also delay the inception of the MXIbus daisy-chain at any time.

Appendix

Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve technical problems and a form you can use to comment on the product documentation. When you contact us, we need the information on the Technical Support Form and the configuration form, if your manual contains one, about your system configuration to answer your questions as quickly as possible.

National Instruments has technical assistance through electronic, fax, and telephone systems to quickly provide the information you need. Our electronic services include a bulletin board service, an FTP site, a Fax-on-Demand system, and e-mail support. If you have a hardware or software problem, first try the electronic support systems. If the information available on these systems does not answer your questions, we offer fax and telephone support through our technical support centers, which are staffed by application engineers.

Electronic Services

Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call (512) 795-6990. You can access these services at:

United States: (512) 794-5422
Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 01 48 65 15 59
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as `anonymous` and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the /`support` directories.

Fax-on-Demand Support

Fax-on-Demand is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access Fax-on-Demand from a touch-tone telephone at (512) 418-1111.

E-Mail Support (currently U.S. only)

You can submit technical support questions to the applications engineering team through e-mail at the Internet address listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

`support@natinst.com`

Telephone and Fax Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

	 Telephone	 Fax
Australia	03 9879 5166	03 9879 6277
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 527 2321	09 502 2930
France	01 48 14 24 24	01 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Israel	03 5734815	03 5734816
Italy	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 520 2635	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
U.K.	01635 523545	01635 523154

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: *MXIbus Specification Version 2.0*

Edition Date: March 1997

Part Number: 340007B-01

Please comment on the completeness, clarity, and organization of the manual.

_____

_____

_____

_____

_____

_____

_____

If you find errors in the manual, please record the page numbers and describe the errors.

_____

_____

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

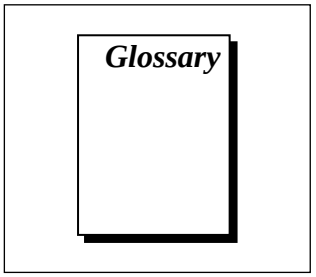
Address _____

_____

Phone (____) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
(512) 794-5678



Prefix	Meaning	Value
n-	nano-	10^{-9}
µ-	micro-	10^{-6}
m-	milli-	10^{-3}

Symbols

*	An asterisk (*) following a signal name designates a low-active signal.
°	degrees
Ω	ohms
%	percent

A

A	amperes
A16 space	VXIbus address space equivalent to the VME 64 KB short address space. In VXI, the upper 16 KB of A16 space is allocated for use by VXI devices configuration registers. This 16 KB region is referred to as VXI configuration space.
A24 space	VXIbus address space equivalent to the VME 16 MB standard address space.

A32 space	VXIbus address space equivalent to the VME 4 GB extended address space.
ACFAIL*	A VMEbus backplane signal that is asserted when a power failure has occurred (either AC line source or power supply malfunction), or if it is necessary to disable the power supply (such as for a high temperature condition).
address	Character code that identifies a specific location (or series of locations) in memory.
address modifier	One of six signals in the VMEbus specification used by VMEbus masters to indicate the address space in which a data transfer is to take place.
address space	A set of 2^n memory locations differentiated from other such sets in VXI/VMEbus systems by six addressing lines known as address modifiers. *n* is the number of address lines required to uniquely specify a byte location in a given space. Valid numbers for *n* are 16, 24, and 32. In VME/VXI, because there are six address modifiers, there are 64 possible address spaces.
address window	A portion of address space that can be accessed from the application program.
ANSI	American National Standards Institute
arbiter	A functional module that accepts bus requests from MXIbus devices requesting to use the data bus and grants control of the bus to one device at a time. The arbiter is located on the MXIbus System Controller.
arbitration	A process in which a potential bus master gains control over a particular bus.
asserted	A signal in its active true state.
asynchronous	Not synchronized; not controlled by time signals.

B

B	bytes
backplane	An assembly, typically a printed circuit board, with 96-pin connectors and signal paths that bus the connector pins. A C-size VXIbus system will have two sets of bused connectors called J1 and J2. A D-size VXIbus system will have three sets of bused connectors called J1, J2, and J3.
BERR*	Bus error signal
binary	A numbering system with a base of 2.
block-mode transfer	An uninterrupted transfer of data elements in which the master sources only the first address at the beginning of the cycle. The slave is then responsible for incrementing the address on subsequent transfers so that the next element is transferred to or from the proper storage location. In VME, the data transfer may have no more than 256 elements; MXI does not have this restriction.
BTO unit	Bus Timeout Unit; a functional module that times the duration of each data transfer and terminates the cycle if the duration is excessive. Without the termination capability of this module, a bus master attempt to access a nonexistent slave could result in an indefinitely long wait for a slave response.
bus master	A device that is capable of requesting the Data Transfer Bus (DTB) for the purpose of accessing a slave device.
byte	Indicates an 8-bit data quantity.

C

C	Celsius
CLK10	A 10 MHz, ± 100 ppm, individually buffered (to each module slot), differential ECL system clock that is sourced from Slot 0 of a VXIbus mainframe and distributed to Slots 1 through 12 on P2. It is distributed to each slot as a single-source, single-destination signal with a matched delay of under 8 ns.

Commander	A message-based device which is also a bus master and can control one or more Servants.
configuration registers	A set of registers through which the system can identify a module device type, model, manufacturer, address space, and memory requirements. In order to support automatic system and memory configuration, the VXIbus specification requires that all VXIbus devices have a set of such registers.
controller	An intelligent device (usually involving a CPU) that is capable of controlling other devices.

D

daisy-chain	A method of propagating signals along a bus, in which the devices are prioritized on the basis of their position on the bus.
Data Transfer Bus	DTB; one of four buses on the VMEbus backplane. The DTB is used by a bus master to transfer binary data between itself and a slave device.
deadlock	Unresolved situation in which two devices are vying for the use of a resource.
DMA	Direct Memory Access; a method by which data is transferred between devices and internal memory without intervention of the central processing unit.
driver	The output portion of a logic device, which drives a signal line to a high or low logic level.
DTACK*	Data Acknowledge signal
DTB	See Data Transfer Bus.
dynamic configuration	A method of automatically assigning logical addresses to VXIbus devices at system startup or other configuration times.
dynamically configured device	A device that has its logical address assigned by the Resource Manager. A VXI device initially responds at Logical Address 255 when its MODID line is asserted. A MXIbus device responds at Logical Address 255 during a priority select cycle. The Resource Manager subsequently assigns it a new logical address, which the device responds to until powered down.

E

embedded controller	An intelligent CPU (controller) interface plugged directly into the VXI backplane, giving it direct access to the VXIbus. It must have all of its required VXI interface capabilities built in.
EMC	Electromechanical Compliance
EMI	Electromagnetic Interference
external controller	In this configuration, a plug-in interface board in a computer is connected to the VXI mainframe via one or more VXIbus extended controllers. The computer then exerts overall control over VXIbus system operations.

F

fair requester	A MXIbus master that will not arbitrate for the MXIbus after releasing it until it detects the bus request signal inactive. This ensures that all requesting devices will be granted use of the bus.
false	A logic value of zero (0).
functional module	Circuitry that performs a specific task.

H

H	High signal level.
hex	Hexadecimal; the numbering system with base 16, using the digits 0 to 9 and letters A to F.
Hz	hertz; cycles per second.

I

I/O	input/output; the techniques, media, and devices used to achieve communication between machines and users.
IEEE	Institute of Electrical and Electronics Engineers
IEEE-1014	The VME specification.

in.	inches
interlocked arbitration scheme	Contrasted with normal operation scheme, an optional scheme of operation in which the system performs as one large VMEbus mainframe with only one master of the entire system (VMEbus and MXIbus) at any given moment. In this scheme there is no chance for a deadlock situation.
interrupt	A means for a device to request service from another device.
interrupt handler	A VMEbus functional module that detects interrupt requests generated by Interrupters and responds to those requests by requesting status and identify information.
interrupt level	The relative priority at which a device can interrupt.
IRQ*	Interrupt signal

K

KB	Kilobytes of memory

L

L	Low signal level.
logical address	An 8-bit number that uniquely identifies each VXIbus device in a system. It defines the A16 register address of a device, and indicates Commander and Servant relationships.
longword	Indicates a 32-bit (4-byte) data quantity.

M

m	meters
mainframe extender	A device such as the VXI-MXI-2 or the VME-MXI-2 that interfaces a VXI/VMEbus mainframe to an interconnect bus. It routes bus transactions from the VXI/VMEbus to the interconnect bus or vice versa. A mainframe extender has a set of registers that defines the routing mechanisms for data transfers, interrupts, and utility bus signals, and has optional VXI/VMEbus first slot capability.

mapping	Establishing a range of address space for a one-to-one correspondence between each address in the window and an access in VXI/VMEbus memory.
master	A functional part of a MXI/VME/VXIbus device that initiates data transfers on the backplane. A transfer can be either a read or a write.
master-mode operation	A device is in master mode if it is performing a bus cycle which it initiated.
MB	Megabytes of memory
MBLT	Eight-byte block transfers in which both the Address bus and the Data bus are used to transfer data.
memory device	A VMEbus device that not only has configuration registers, but also has memory that is accessible through addresses on the VME data transfer bus.
message-based device	An intelligent device that implements the defined VXIbus registers and communication protocols. These devices are able to use Word Serial Protocol to communicate with one another through communication registers.
MITE	A National Instruments custom ASIC, a sophisticated dual-channel DMA controller that incorporates the Synchronous MXI and VME64 protocols to achieve high-performance block transfer rates.
MODID	Module Identification lines
MTBF	Mean Time Between Failure
MXI-2	The second generation of the National Instruments MXIbus product line. MXI-2 expands the number of signals on a standard MXIbus cable by including VXI triggers, all VXI interrupts, CLK10, SYSFAIL*, SYSRESET*, and ACFAIL*.
MXIbus	Multisystem eXtension Interface Bus; a high-performance communication link that interconnects devices using round, flexible cables.
MXIbus device	A host computer adapter, a peripheral controller, or an intelligent peripheral that may be attached to the MXIbus.

MXIbus System Controller — A functional module that has arbiter, daisy-chain driver, and MXIbus cycle timeout responsibility. Always the first device in the MXIbus daisy chain.

N

Non-Slot 0 device — A device configured for installation in any slot in a VXIbus mainframe other than Slot 0. Installing such a device into Slot 0 can damage the device, the VXIbus backplane, or both.

nonprivileged access — One of the defined types of VMEbus data transfers; indicated by certain address modifier codes. Each of the defined VMEbus address spaces has a defined nonprivileged access mode.

O

open-collector — A driver that sinks current in the low state but does not source any significant current when in the high state. Terminating resistors at the ends of the MXIbus ensure that the signal line voltage rises to the high level whenever it is not driven low.

P

P1 — The minimum connector required for a VMEbus system. It includes 24 address lines, 16 data lines and all control, arbitration, and interrupt signals.

P2 — A second VMEbus connector providing 32 bits of address and data. RETRY support is also located on P2. In VXI, the P2 connector adds trigger, MODID, and CLK10 signals.

P3 — A third connector defined by the VXIbus specification that adds a 100 MHz CLK and additional triggering capabilities. The VME-MXI-2 does not have support for P3.

parity — Ensures that there is always either an even number or an odd number of asserted bits in a byte, character, or word, according to the logic of system. If a bit should be lost in data transmission, its loss can be detected by checking the parity.

propagation	The transmission of signal through a computer system.

R

read	To get information from any input device or file storage media.
receiver	The input portion of a logic device, which receives a signal line and detects a high or low logic level.
register-based device	A Servant-only device that supports VXIbus configuration registers. Register-based devices are typically controlled by message-based devices via device-dependent register reads and writes.
requester	A functional module that requests use of the MXIbus for its corresponding master device. Every MXIbus master must have a requester functional module.
Resource Manager	A message-based Commander located at Logical Address 0, which provides configuration management services such as address map configuration, Commander and Servant mappings, and self-test and diagnostic management.
response	A signal or interrupt generated by a device to notify another device of asynchronous event. Responses contain the information in the Response register of a sender.
retry	An acknowledge by a destination that signifies that the cycle did not complete and should be repeated.

S

s	seconds
Servant	A device controlled by a Commander; there are message-based and register-based Servants.
Shared Memory Protocol	A communication protocol that uses a block of memory that is accessible to both a client and a server. The memory block operates as a message buffer for communications.

signal	Any communication between message-based devices consisting of a write to a Signal register. Sending a signal requires that the sending device have VMEbus master capability.
signal assertion	The act of driving a signal to the true state.
signal negation	The act of driving a signal to the false state or causing terminators to bias the signal to the false state (by placing the driver in the high-impedance condition).
slave	A functional module that detects bus cycles the MXIbus master initiates and, when those cycles select it, transfers data between itself and the MXIbus master. MXIbus devices may be master-only, slave-only, or have combined master/slave capabilities.
slave-mode operation	A device is in slave mode it if is responding to a bus cycle.
Slot 0 device	A device configured for installation in Slot 0 of a VXIbus mainframe. This device is unique in the VXIbus system in that it performs the VXI/VMEbus System Controller functions, including clock sourcing and arbitration for data transfers across the backplane. Installing such a device into any other slot can damage the device, the VXIbus backplane, or both.
statically configured device	A device whose logical address cannot be set through software; that is, it is not dynamically configurable.
supervisory access	One of the defined types of VMEbus data transfers; indicated by certain address modifier codes.
Synchronous MXI Block Protocol	A block data transfer protocol defined by the MXI-2 specification for high-performance data transfers.
SYSFAIL	A VMEbus signal that is used by a device to indicate an internal failure. A failed device asserts this line. In VXI, a device that fails also clears its PASSed bit in its Status register.
SYSRESET	A VMEbus signal that is used by a device to indicate a system reset or power-up condition.

System Controller	A functional module on a MXIbus device that has arbiter, daisy-chain driver, and MXIbus cycle timeout responsibility. A MXIbus network has only one active System Controller, which is always the first device in the MXIbus daisy-chain.
system hierarchy	The tree structure of the Commander/Servant relationships of all devices in the system at a given time. In the VXIbus structure, each Servant has a commander. A Commander can in turn be a Servant to another Commander.

T

terminators	Also called *termination networks*; devices located at the ends of a MXIbus daisy-chain that are used to minimize reflections and bias signals to their unasserted states.
TERMPWR	Termination Power; 3.4 VDC for the MXIbus.
trigger	Either TTL or ECL lines used for intermodule communication.
true	A logic value of one (1).

U

unasserted	A signal in its active false state.

V

V	volts
VDC	volts direct current
VME	Versa Module Eurocard or IEEE 1014
VMEbus System Controller	A device configured for installation in Slot 0 of a VXIbus mainframe or Slot 1 of a VMEbus chassis. This device is unique in the VMEbus system in that it performs the VMEbus System Controller functions, including clock sourcing and arbitration for data transfers across the backplane. Installing such a device into any other slot can damage the device, the VMEbus/VXIbus backplane, or both.

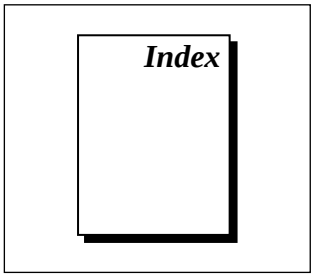
VXIbus	VMEbus Extensions for Instrumentation

W

word	Indicates a 16-bit (2-byte) data quantity.
Word Serial Protocol	The simplest required communication protocol supported by message-based devices in a VXIbus system. It utilizes the A16 communication registers to transfer data using a simple polling handshake method.
write posting	A mechanism that signifies that a device will immediately give a successful acknowledge to a write transfer and place the transfer in a local buffer. The device can then independently complete the write cycle to the destination.

X

X	Don’t care value.



Index

B

C

D

E

F

G

I

L

M

P

R

S

T

U

V

W