
British Standard Specification for
IEC 821 Bus. Microprocessor system
bus for 1 to 4 byte data

National foreword

This British Standard, prepared under the direction of the Information Systems Technology Standards Policy Committee, is identical with IEC 821 : 1987 'IEC 821 Bus. Microprocessor system bus for 1 to 4 byte data' published by the International Electrotechnical Commission (IEC).

For the purposes of this British Standard, any references to IEC page numbers in the text should be ignored.

Cross-references

International standard	Corresponding British Standard
IEC 297-1 : 1982	BS 5954 : 1980 Specification for dimensions of panels and racks for electronic equipment (Identical, exercising national options)
IEC 297-3 : 1986	BS 5954 : Part 3 : 1985 Specification for subracks and associated plug-in units. (Identical)

The Technical Committee has reviewed the provisions of IEC 603-2 to which reference is made in the text, and has decided that they are acceptable for use in connection with this standard.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

CONTENTS

	Page
National foreword	Inside front cover
Committees responsible	Back cover

CHAPTER 0: SCOPE

Note to the reader	9
--------------------------	---

CHAPTER 1: INTRODUCTION TO THE IEC 821 BUS STANDARD

Section		
1.1	IEC 821 BUS standard objectives	10
1.2	IEC 821 BUS interface system elements	10
1.2.1	Basic definitions	10
1.2.1.1	Terms used to describe the IEC 821 BUS mechanical structure	10
1.2.1.2	Terms used to describe the IEC 821 BUS functional structure	11
1.2.1.3	Types of cycles on the IEC 821 Bus	13
1.2.2	Basic IEC 821 BUS structure	14
1.3	IEC 821 BUS standard diagrams	18
1.4	Standard terminology	18
1.4.1	Signal line states	19
1.4.2	Use of the asterisk (*)	20
1.5	Protocol specification	20
1.5.1	Interlocked bus signals	21
1.5.2	Broadcast bus signal	21
1.6	System examples and explanations	22

CHAPTER 2: IEC 821 BUS DATA TRANSFER BUS

2.1	Introduction	23
2.2	Data Transfer Bus lines	23
2.2.1	Addressing lines	25
2.2.2	Address modifier lines	28
2.2.3	Data lines	30
2.2.4	Data Transfer Bus control lines	33
2.2.4.1	AS*	33
2.2.4.2	DS0* and DS1*	33
2.2.4.3	DTACK*	33
2.2.4.4	BERR*	34
2.2.4.5	WRITE*	35
2.3	DTB modules - Basic description	35
2.3.1	MASTER	35
2.3.2	SLAVE	38
2.3.3	BUS TIMER	40
2.3.4	LOCATION MONITOR	42
2.3.5	Addressing modes	43
2.3.6	Basic data transfer capabilities	45
2.3.7	Block transfer capabilities	47
2.3.8	Read-modify-write capabilities	49
2.3.9	Unaligned transfer capabilities	52

Section	Page
2.3.10 ADDRESS-ONLY capability	55
2.3.11 Interaction between DTB functional modules	55
2.4 Typical operation	57
2.4.1 Typical data transfer cycles	57
2.4.2 Address pipelining	62
2.5 Data Transfer Bus acquisition	63
2.6 DTB timing rules and observations	65

CHAPTER 3: IEC 821 BUS DATA TRANSFER BUS ARBITRATION

3.1 Bus arbitration philosophy	95
3.1.1 Types of arbitration	95
3.2 Arbitration bus lines	97
3.2.1 Bus request and bus grant lines	99
3.2.2 Bus busy line (BBSY*)	99
3.2.3 Bus clear line (BCLR*)	99
3.3 Functional modules	100
3.3.1 ARBITER	100
3.3.2 REQUESTER	103
3.3.3 Data Transfer Bus MASTER	106
3.3.3.1 Release of the DTB	106
3.3.3.2 Acquisition of the DTB	107
3.3.3.3 Other information	107
3.4 Typical operation	107
3.4.1 Arbitration of two different levels of bus request	107
3.4.2 Arbitration of two bus requests on the same bus request line	112
3.5 Race conditions between MASTER requests and ARBITER grants	116

CHAPTER 4: IEC 821 BUS PRIORITY INTERRUPT BUS

4.1 Introduction	117
4.1.1 Single handler systems	117
4.1.2 Distributed systems	117
4.2 Priority Interrupt Bus lines	121
4.2.1 Interrupt request lines	121
4.2.2 Interrupt acknowledge line	121
4.2.3 Interrupt acknowledge daisy-chain - IACKIN*/IACKOUT*	121
4.3 Priority Interrupt Bus modules - Basic description	122
4.3.1 INTERRUPT HANDLERS	123
4.3.2 INTERRUPTER	125
4.3.3 IACK DAISY-CHAIN DRIVER	129
4.3.4 Interrupt request handling capabilities	130
4.3.5 Interrupt request generation capabilities	131
4.3.6 STATUS/ID transfer capabilities	131
4.3.7 Interrupt request release capabilities	132
4.3.8 Interaction between Priority Interrupt Bus modules	134
4.4 Typical operation	138
4.4.1 Single handler interrupt operation	138
4.4.2 Distributed interrupt operation	139
4.4.2.1 Distributed interrupt systems with seven INTERRUPT HANDLERS	139
4.4.2.2 Distributed interrupt systems with two to six INTERRUPT HANDLERS	139

Section	Page
4.4.3 Example: typical single handler interrupt system operation	141
4.4.4 Example: prioritization of two interrupts in a distributed interrupt system ...	145
4.5 Priority Interrupt Bus timing RULES and OBSERVATIONS	147

CHAPTER 5: IEC 821 BUS UTILITY BUS

5.1 Introduction	167
5.2 Utility Bus signal lines	167
5.3 Utility Bus modules	167
5.3.1 The SYSTEM CLOCK DRIVER	167
5.3.2 The SERIAL CLOCK DRIVER	167
5.3.3 The POWER MONITOR	167
5.4 System initialization and diagnostics	172
5.5 Power pins	175
5.6 RESERVED line	175

CHAPTER 6: IEC 821 BUS ELECTRICAL SPECIFICATIONS

6.1 Introduction	177
6.2 Power distribution	177
6.2.1 D.C. voltage specifications	178
6.2.2 Pin and socket connector electrical ratings	179
6.3 Electrical signal characteristics	179
6.4 Bus driving and receiving requirements	180
6.4.1 Bus driver definitions	180
6.4.2 Driving and loading RULES for all IEC 821 BUS lines	181
6.4.2.1 Driving and loading RULES for high current three-state lines (AS*, DS0*, DS1*)	181
6.4.2.2 Driving and loading RULES for standard three-state lines (A01-A31, D00-D31, AM0-AM5, IACK*, LWORD*, WRITE*)	182
6.4.2.3 Driving and loading RULES for high current totem-pole lines (SERCLK, SYSCLK, BCLR*)	183
6.4.2.4 Driving and loading RULES for standard totem-pole lines (BG0OUT*-BG3OUT*/BG0IN*-BG3IN*, IACKOUT*/IACKIN*)	184
6.4.2.5 Driving and loading RULES for open-collector lines (BR0*-BR3*, BBSY*, IRQ1*-IRQ7*, DTACK*, BERR*, SYSFAIL*, SYSRESET*, ACFAIL*, IACK*)	185
6.5 Backplane signal line interconnections	185
6.5.1 Termination networks	186
6.5.2 Characteristic impedance	187
6.5.3 Additional information	190
6.6 User defined signals	191
6.7 Signal line drivers and terminations	191

CHAPTER 7: IEC 821 BUS MECHANICAL SPECIFICATIONS

7.1 Introduction	193
7.2 IEC 821 BUS boards	194
7.2.1 Single height boards	195
7.2.2 Double height boards	195
7.2.3 Board connectors	196

Section	Page
7.2.4 Board assemblies	197
7.2.5 Board widths	197
7.2.6 IEC 821 BUS board warpage, lead length and component height	197
7.3 Front panels	198
7.3.1 Handles	199
7.3.2 Front panel mounting	200
7.3.3 Front panel dimensions	200
7.3.4 Filler panels	201
7.3.5 Board ejectors/injectors	201
7.4 Backplanes	202
7.4.1 Backplane dimensional requirements	203
7.4.2 Signal line termination networks	203
7.5 Assembly of IEC 821 BUS subracks	204
7.5.1 Subracks and slot widths	204
7.5.2 Subrack dimensions	204
7.6 IEC 821 BUS backplane connectors and IEC 821 BUS board connectors	223
7.6.1 Pin assignments for the J_1/P_1 connector	223
7.6.2 Pin assignments for the J_2/P_2 connector	224
APPENDIX A - Glossary of IEC 821 BUS terms	225
APPENDIX B - IEC 821 BUS connector/pin description	230
APPENDIX C - Use of the SERCLK and SERDAT* lines	233

Figures

1-1: System elements defined by this standard	16
1-2: Functional modules and buses defined by this standard	17
1-3: Signal timing notation	22
2-1: Data Transfer Bus functional block diagram	24
2-2: Block diagram: MASTER	36
2-3: Block diagram: SLAVE	38
2-4: Block diagram: BUS TIMER	40
2-5: Block diagram: LOCATION MONITOR	42
2-6: Four ways that 32 bits of data might be stored in memory	52
2-7: Four ways that 16 bits of data might be stored in memory	53
2-8: An example of a single byte read cycle	59
2-9: An example of a double byte write cycle	60
2-10: An example of a quad byte write cycle	61
2-11: Data Transfer Bus MASTER exchange sequence	64
2-12: MASTER, SLAVE and LOCATION MONITOR - Address broadcast timing ALL CYCLES	83
2-13: MASTER, SLAVE and LOCATION MONITOR - Address broadcast timing Single even byte transfers; single odd byte transfers; double byte transfers; quad byte transfers; unaligned transfers	84
2-14: MASTER, SLAVE and LOCATION MONITOR - Address broadcast timing Single byte block transfers; double byte block transfers; quad byte block transfers	85
2-15: MASTER, SLAVE and LOCATION MONITOR - Address broadcast timing Single byte RMW cycles; double byte RMW cycles; quad byte RMW cycles	86

Figures	Page
2-16: MASTER, SLAVE and LOCATION MONITOR - Data transfer timing BYTE(0) READ; BYTE(1) READ; BYTE(2) READ; BYTE(3) READ; BYTE(0-2) READ; BYTE(1-3) READ; SINGLE BYTE BLOCK READ	87
2-17: MASTER, SLAVE and LOCATION MONITOR - Data transfer timing BYTE(0-1) READ; BYTE(2-3) READ; BYTE(0-3) READ; BYTE(1-2) READ; DOUBLE BYTE BLOCK READ; QUAD BYTE BLOCK READ	88
2-18: MASTER, SLAVE and LOCATION MONITOR - Data transfer timing BYTE(0) WRITE; BYTE(1) WRITE; BYTE(2) WRITE; BYTE(3) WRITE; BYTE(0-2) WRITE; BYTE(1-3) WRITE; SINGLE BYTE BLOCK WRITE	89
2-19: MASTER, SLAVE and LOCATION MONITOR - Data transfer timing BYTE(0-1) WRITE; BYTE(2-3) WRITE; BYTE(0-3) WRITE; BYTE(1-2) WRITE; DOUBLE BYTE BLOCK WRITE; QUAD BYTE BLOCK WRITE	90
2-20: MASTER, SLAVE and LOCATION MONITOR - Data transfer timing Single byte RMW cycle	91
2-21: MASTER, SLAVE and LOCATION MONITOR - Data transfer timing Double byte RMW cycles quad byte RMW cycles	91
2-22: Address strobe inter-cycle timing	92
2-23: Data strobe inter-cycle timing A cycle where both data strobes go low followed by a cycle where one or both data strobes go low	92
2-24: Data strobe inter-cycle timing A cycle where one data strobe goes low followed by a cycle where one or both data strobes go low	93
2-25: MASTER, SLAVE and BUS TIMER - Data transfer timing Timed-out cycle	93
2-26: MASTER - DTB control transfer timing	94
3-1: Arbitration bus functional block diagram	96
3-2: Illustration of the daisy-chained bus grant lines	98
3-3: Block diagram: ARBITER	102
3-4: Block diagram: REQUESTER	105
3-5: Arbitration flow diagram: two REQUESTERS, two request levels	109
3-6: Arbitration sequence diagram: two REQUESTERS, two request levels	111
3-7: Arbitration flow diagram: two REQUESTERS, same request level	113
3-8: Arbitration sequence diagram: two REQUESTERS, same request level	115
4-1: Priority Interrupt Bus functional block diagram	118
4-2: Interrupt subsystem structure: single handler system	119
4-3: Interrupt subsystem structure: distributed system	120
4-4: IACKIN*/IACKOUT* DAISY-CHAIN	122
4-5: Block diagram: INTERRUPT HANDLER	124
4-6: Block diagram: INTERRUPTER	128
4-7: Block diagram: IACK DAISY-CHAIN DRIVER	130

Figures	Page
4-8: Release of the interrupt request lines by ROAK and RORA INTERRUPTERS	134
4-9: An IACK DAISY-CHAIN DRIVER and an INTERRUPTER on the same board	136
4-10: Two INTERRUPTERS on the same board	137
4-11: The three phases of an interrupt sequence	138
4-12: Two INTERRUPT HANDLERS, each monitoring one interrupt request line	140
4-13: Two INTERRUPT HANDLERS, each monitoring several interrupt request lines	141
4-14: Typical single handler interrupt system operation flow diagram	143
4-15: Typical distributed interrupt system with two INTERRUPT HANDLERS, flow diagram	144
4-16: INTERRUPT HANDLER and INTERRUPTER - INTERRUPTER selection timing SINGLE, DOUBLE and QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE	162
4-17: IACK DAISY-CHAIN DRIVER - INTERRUPTER selection timing SINGLE, DOUBLE and QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE	162
4-18: Participating INTERRUPTER - INTERRUPTER selection timing SINGLE, DOUBLE and QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE	163
4-19: Responding INTERRUPTER - INTERRUPTER selection timing SINGLE, DOUBLE and QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE	164
4-20: INTERRUPT HANDLER - STATUS/ID transfer timing SINGLE BYTE INTERRUPT ACKNOWLEDGE CYCLE	164
4-21: INTERRUPT HANDLER - STATUS/ID transfer timing DOUBLE BYTE INTERRUPT ACKNOWLEDGE CYCLE; QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE	165
4-22: Responding INTERRUPTER - STATUS/ID transfer timing SINGLE BYTE INTERRUPT ACKNOWLEDGE CYCLE	165
4-23: Responding INTERRUPTER - STATUS/ID transfer timing DOUBLE BYTE INTERRUPT ACKNOWLEDGE CYCLE; QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE	166
4-24: IACK DAISY-CHAIN DRIVER, responding INTERRUPTER, and participating INTERRUPTER - IACK daisy-chain inter-cycle timing	166
5-1: Utility Bus block diagram	169
5-2: SYSTEM CLOCK DRIVER timing diagram	170
5-3: Block diagram of POWER MONITOR module	170
5-4: POWER MONITOR power failure timing	171
5-5: POWER MONITOR system restart timing	171
5-6: SYSRESET* and SYSFAIL* timing diagram	174
5-7: Current rating for power pins	176
6-1: IEC 821 BUS signal levels	179
6-2: Standard bus termination	187
6-3: Backplane microstrip signal line cross section	188
6-4: Z_0 versus line width	189
6-5: C_0 versus line width	189

Figures	Page
7-1: Subrack with mixed board sizes	205
7-2: Single height board: basic dimensions	206
7-3: Double height board: basic dimensions	207
7-4: Connector position on single and double height boards.....	208
7-5: Cross sectional view of board, connector, backplane, and front panel	209
7-6: Component height, lead length and board warpage	210
7-7: Single height, single width, front panel	211
7-8: Double height, single width, front panel	212
7-9: Front panel mounting brackets and dimensions of single height board	213
7-10: Front panel mounting brackets and dimensions of double height board	214
7-11: Single height filler panel	215
7-12: Double height filler panel	216
7-13: Backplane overall dimensions	217
7-14: Backplane detailed dimensions	218
7-15: "Off board type" backplane termination (view from top of backplane)	219
7-16: "On board type" backplane termination (view from top of backplane)	220
7-17: 21-slot subrack	221
7-18: Board guide detail	222
C-1: SERCLK timing diagram	234

Tables

2-1: The four categories of byte location	25
2-2: Signal levels used to select which byte location(s) are accessed during a data transfer	27
2-3: Address modifier codes	29
2-4: Use of the data lines to move data during each of the 34 cycle types	31
2-5: RULES and PERMISSIONS that specify the use of the dotted lines by the various types of MASTERS	37
2-6: RULES and PERMISSIONS that specify the use of the dotted lines by the various types of SLAVES	39
2-7: Use of the BT0() mnemonic to specify the time-out period of BUS TIMERS	41
2-8: RULES and PERMISSIONS that specify the use of the dotted lines by the various types of LOCATION MONITORS	43
2-9: Mnemonics that specify addressing capabilities	44
2-10: Mnemonics that specify basic data transfer capabilities	47
2-11: Mnemonics that specify block transfer capabilities	49
2-12: Mnemonics that specify read-modify-write capabilities	51
2-13: Transferring 32 bits of data using multiple byte transfer cycles	53
2-14: Transferring 16 bits of data using multiple byte transfer cycles	54
2-15: Mnemonic that specifies unaligned transfer capability	54
2-16: Mnemonic that specifies ADDRESS-ONLY capability	55
2-17: Timing diagrams that define MASTER, SLAVE and LOCATION MONITOR operation	67
2-18: Timing diagrams that define MASTER, SLAVE, LOCATION MONITOR, INTERRUPT HANDLER and INTERRUPTER operation during ALL cycles	68

Tables	Page
2-19: Definitions of mnemonics used in Tables 2-20, 2-21 and 2-22	68
2-20: Use of addressing lines to select a 4-byte group	69
2-21: Use of the DS1*, DS0*, A01, and LWORD* lines to select the byte(s) within a 4-byte group	70
2-22: Use of the data bus lines to transfer data	71
2-23: MASTER, SLAVE, and LOCATION MONITOR timing parameters	73
2-24: BUS TIMER timing parameters	73
2-25: MASTER-Timing RULES and OBSERVATIONS	74
2-26: SLAVE-Timing RULES and OBSERVATIONS	78
2-27: LOCATION MONITOR Timing OBSERVATIONS	81
2-28: BUS TIMER-Timing RULES	82
3-1: RULES and PERMISSIONS that specify the use of the dotted lines by the various types of ARBITERS defined by this standard	103
3-2: RULES and PERMISSIONS that specify the use of the dotted lines by the various types of REQUESTERS defined by this standard	105
4-1: RULES and PERMISSIONS that specify the use of the dotted lines by the various types of INTERRUPT HANDLERS	125
4-2: RULES and PERMISSIONS that specify the use of the dotted lines by the various types of INTERRUPTERS	129
4-3: Use of the IH() mnemonic to specify interrupt handling capabilities	130
4-4: Use of the I() mnemonic to specify interrupt request generation capabilities ..	131
4-5: Mnemonics that specify STATUS/ID transfer capabilities	131
4-6: Mnemonics that specify interrupt request release capabilities	134
4-7: 3-bit interrupt acknowledge code	145
4-8: Timing diagrams that define INTERRUPT HANDLER and INTERRUPTER operation	149
4-9: Timing diagrams that define IACK DAISY-CHAIN DRIVER operation	149
4-10: Timing diagrams that define participating INTERRUPTER operation	150
4-11: Timing diagrams that define responding INTERRUPTER operation	150
4-12: Definitions of mnemonics used in Tables 4-13, 4-14 and 4-15	151
4-13: Use of addressing lines during interrupt acknowledge cycles	151
4-14: Use of the DS1*, DS0*, LWORD*, and WRITE* lines during interrupt acknowledge cycles	152
4-15: Use of the data bus lines to transfer the STATUS/ID	152
4-16: INTERRUPT HANDLER, INTERRUPTER, and IACK DAISY-CHAIN DRIVER timing parameters .	153
4-17: INTERRUPT HANDLER - Timing RULES and OBSERVATIONS	154
4-18: INTERRUPTER - Timing RULES and OBSERVATIONS	157
4-19: IACK DAISY-CHAIN DRIVER - Timing RULES and OBSERVATIONS	160
5-1: Module drive during power-up and power-down sequences	173
6-1: Bus voltage specifications	178
6-2: Bus driving and receiving requirements	180
6-3: Bus driver summary	192
7-1: J ₁ /P ₁ pin assignments	223
7-2: J ₂ /P ₂ pin assignments	224
C-1: SERCLK timing values	234

IEC 821 BUS
MICROPROCESSOR SYSTEM BUS FOR 1 TO 4 BYTE DATA

CHAPTER 0: SCOPE

This standard describes a high performance backplane bus for use in microprocessor based systems. This parallel bus supports single and block transfer cycles on a 32-bit non-multiplexed address and data highway. Transmission is governed by an asynchronous handshaken protocol. The bus allocation provides for multiprocessor architectures. This bus also supports inter-module interrupts for facilitating quick response to internal and external events. The mechanics of the boards and chassis are based on IEC Publication 297: Dimensions of Panels and Racks.

Note to the reader

IEC 822 BUS, named in Section 7.4 represents the bus which is in the process of being standardized by Sub-Committee 47B as a sub-system bus of the IEC 821 BUS which constitutes this standard.

IEC 823 BUS, named in Paragraphs 1.2.1.2 and 5.3.2 and Appendices B and C, represents the bus which is under consideration by Sub-Committee 47B as the serial bus of this IEC 821 BUS.

CHAPTER 1: INTRODUCTION TO THE IEC 821 BUS STANDARD

1.1 IEC 821 BUS standard objectives

This IEC 821 BUS standard defines an interfacing system used to interconnect data processing, data storage and peripheral control devices in a closely coupled hardware configuration. The system has been conceived with the following objectives:

- a) To allow communication between devices on the IEC 821 BUS without disturbing the internal activities of other devices interfaced to the IEC 821 BUS.
- b) To specify the electrical and mechanical system characteristics required to design devices that will reliably and unambiguously communicate with other devices interfaced to the IEC 821 BUS.
- c) To specify protocols that precisely define the interaction between the IEC 821 BUS and devices interfaced to it.
- d) To provide terminology and definitions that describe the system protocol.
- e) To allow a broad range of design latitude so that the designer can optimize cost and/or performance without affecting system compatibility.
- f) To provide a system where performance is primarily device limited, rather than system interface limited.

1.2 IEC 821 BUS interface system elements

1.2.1 Basic definitions

The IEC 821 BUS structure can be described from two points of view: its mechanical structure and its functional structure. The mechanical specification describes the physical dimensions of subracks, backplanes, front panels, plug-in boards, etc. The IEC 821 BUS functional specification describes how the bus works, what functional modules are involved in each transaction, and the rules which govern their behavior. This paragraph provides informal definitions for some basic terms used to describe both the physical and the mechanical structure of the IEC 821 BUS.

1.2.1.1 Terms used to describe the IEC 821 BUS mechanical structure

IEC 821 BUS BACKPLANE

A printed circuit (PC) board with 96-pin connectors and signal paths that bus the connector pins. Some IEC 821 BUS systems have a single PC board, called the J_1 backplane. It provides the signal paths needed for basic operation. Other IEC 821 BUS systems also have an optional second PC board, called a J_2 backplane. It provides the additional 96-pin connectors and signal paths needed for wider data and address transfers. Still others have a single PC board that provides the signal conductors and connectors of both the J_1 and J_2 backplanes.

BOARD

A printed circuit (PC) board, its collection of electronic components, and either one or two 96-pin connectors that can be plugged into IEC 821 BUS backplane connectors.

SLOT

A position where a board can be inserted into an IEC 821 BUS backplane. If the IEC 821 BUS system has both a J_1 and a J_2 backplane (or a combination J_1/J_2 backplane) each slot provides a pair of 96-pin connectors. If the system has only a J_1 backplane, then each slot provides a single 96-pin connector.

SUBRACK

A rigid framework that provides mechanical support for boards inserted into the backplane, ensuring that the connectors mate properly and that adjacent boards do not contact each other. It also guides the cooling airflow through the system, and ensures that inserted boards do not disengage themselves from the backplane due to vibration or shock.

1.2.1.2 Terms used to describe the IEC 821 BUS functional structure

Figure 1-1, page 35, shows a simplified block diagram of the functional structure, including the IEC 821 BUS signal lines, backplane interface logic, and functional modules.

BACKPLANE INTERFACE LOGIC

Special interface logic that takes into account the characteristics of the backplane: its signal line impedance, propagation time, termination values, etc. The IEC 821 BUS standard prescribes certain rules for the design of this logic based on the maximum length of the backplane and its maximum number of board slots.

FUNCTIONAL MODULE

A collection of electronic circuitry that resides on one IEC 821 BUS board and works together to accomplish a task.

DATA TRANSFER BUS

One of the four buses provided by the IEC 821 BUS backplane. The Data Transfer Bus allows MASTERS to direct the transfer of binary data between themselves and SLAVES. (Data Transfer Bus is often abbreviated DTB.)

DATA TRANSFER BUS CYCLE

A sequence of level transitions on the signal lines of the DTB that result in the transfer of an address or an address and data between a MASTER and a SLAVE. The Data Transfer Bus cycle is divided into two portions, the address broadcast and then one or more data transfers. There are 34 types of Data Transfer Bus cycles. They are defined later in this chapter.

MASTER

A functional module that initiates DTB cycles in order to transfer data between itself and a SLAVE module.

SLAVE

A functional module that detects DTB cycles initiated by a MASTER and, when those cycles specify its participation, transfers data between itself and the MASTER.

LOCATION MONITOR

A functional module that monitors data transfers over the DTB in order to detect accesses to the locations it has been assigned to watch. When an access to one of these assigned locations occurs, the LOCATION MONITOR generates an on-board signal.

BUS TIMER

A functional module that measures how long each data transfer takes on the DTB and terminates the DTB cycle if a transfer takes too long. Without this module, if the MASTER tries to transfer data to or from a non-existent SLAVE location it might wait forever. The BUS TIMER prevents this by terminating the cycle.

PRIORITY INTERRUPT BUS

One of the four buses provided by the IEC 821 BUS backplane. The Priority Interrupt Bus allows INTERRUPTER modules to send interrupt requests to INTERRUPT HANDLER modules.

INTERRUPTER

A functional module that generates an interrupt request on the Priority Interrupt Bus and then provides STATUS/ID information when the INTERRUPT HANDLER requests it.

INTERRUPT HANDLER

A functional module that detects interrupt requests generated by INTERRUPTERS and responds to those requests by asking for STATUS/ID information.

DAISY-CHAIN

A special type of IEC 821 BUS signal line that is used to propagate a signal level from board to board, starting with the first slot and ending with the last slot. There are four bus grant daisy-chains and one interrupt acknowledge daisy-chain on the IEC 821 BUS.

IACK DAISY-CHAIN DRIVER

A functional module which activates the interrupt acknowledge daisy-chain whenever an INTERRUPT HANDLER acknowledges an interrupt request. This daisy-chain ensures that only one INTERRUPTER will respond with its STATUS/ID when more than one has generated an interrupt request.

ARBITRATION BUS

One of the four buses provided by the IEC 821 BUS backplane. This bus allows an ARBITER module and several REQUESTER modules to coordinate use of the DTB.

REQUESTER

A functional module that resides on the same board as a MASTER or INTERRUPT HANDLER and requests use of the DTB whenever its MASTER or INTERRUPT HANDLER needs it.

ARBITER

A functional module that accepts bus requests from REQUESTER modules and grants control of the DTB to one REQUESTER at a time.

UTILITY BUS

One of the four buses provided by the IEC 821 BUS backplane. This bus includes signals that provide periodic timing and coordinate the power-up and power-down of IEC 821 BUS systems.

SYSTEM CLOCK DRIVER

A functional module that provides a 16 MHz timing signal on the Utility Bus.

SERIAL CLOCK DRIVER

A functional module that provides a periodic timing signal that synchronizes operation of the IEC 823 BUS*. (Although the IEC 821 BUS standard defines a SERIAL CLOCK DRIVER for use with the IEC 823 BUS, and although it reserves two backplane signal lines for use by that bus, the IEC 823 BUS protocol is completely independent of the IEC 821 BUS.) Timing specifications for the SERIAL CLOCK DRIVER are given in Appendix C.

POWER MONITOR MODULE

A functional module that monitors the status of the primary power source to the IEC 821 BUS system, and signals when that power has strayed outside the limits required for reliable system operation. Since most systems are powered by an a.c. source, the POWER MONITOR is typically designed to detect drop-out or brown-out conditions on a.c. lines.

SYSTEM CONTROLLER BOARD

A board which resides in slot 1 of an IEC 821 BUS backplane and has a SYSTEM CLOCK DRIVER, an ARBITER, an IACK DAISY-CHAIN DRIVER and a BUS TIMER. Some also have a SERIAL CLOCK DRIVER, a POWER MONITOR or both.

1.2.1.3 Types of cycles on the IEC 821 BUS**READ CYCLE**

A DTB cycle used to transfer 1, 2, 3, or 4 bytes from a SLAVE to a MASTER. The cycle begins when the MASTER broadcasts an address and an address modifier. Each SLAVE captures the address modifier and address, and checks to see if it is to respond to the cycle. If so, it retrieves the data from its internal storage, places it on the data bus and acknowledges the transfer. The MASTER then terminates the cycle.

WRITE CYCLE

A DTB cycle used to transfer 1, 2, 3, or 4 bytes from a MASTER to a SLAVE. The cycle begins when the MASTER broadcasts an address and address modifier and places data on the DTB. Each SLAVE captures the address and address modifier and checks to see if it is to respond to the cycle. If so, it stores the data and then acknowledges the transfer. The MASTER then terminates the cycle.

* Under consideration.

BLOCK READ CYCLE

A DTB cycle used to transfer a block of 1 to 256 bytes from a SLAVE to a MASTER. This transfer is done using a string of 1, 2, or 4 byte data transfers. Once the block transfer is started, the MASTER does not release the DTB until all of the bytes have been transferred. It differs from a string of read cycles in that the MASTER broadcasts only one address and address modifier (at the beginning of the cycle). Then the SLAVE increments this address on each transfer so that the data for the next transfer is retrieved from the next higher location.

BLOCK WRITE CYCLE

A DTB cycle used to transfer a block of 1 to 256 bytes from a MASTER to a SLAVE. The block write cycle is very similar to the block read cycle. It uses a string of 1, 2, or 4 byte data transfers and the MASTER does not release the DTB until all of the bytes have been transferred. It differs from a string of write cycles in that the MASTER broadcasts only one address and address modifier (at the beginning of the cycle). Then the SLAVE increments this address on each transfer so that the data from the next transfer is stored in the next higher location.

READ-MODIFY-WRITE CYCLE

A DTB cycle that is used to both read from, and write to, a SLAVE location without permitting any other MASTER to access that location. This cycle is most useful in multiprocessing systems where certain memory locations are used to provide semaphore functions.

ADDRESS-ONLY CYCLE

A DTB cycle that consists of an address broadcast, but no data transfer. SLAVES do not acknowledge ADDRESS-ONLY cycles and MASTERS terminate the cycle without waiting for an acknowledgement.

INTERRUPT ACKNOWLEDGE CYCLE

A DTB cycle, initiated by an INTERRUPT HANDLER, that reads a STATUS/ID from an INTERRUPTER. An INTERRUPT HANDLER generates this cycle whenever it detects an interrupt request from an INTERRUPTER and it has control of the DTB.

1.2.2 Basic IEC 821 BUS structure

The IEC 821 BUS interface system consists of backplane interface logic, four groups of signal lines called "buses", and a collection of "functional modules" which can be configured as required. The functional modules communicate with each other using the backplane signal lines.

The "functional modules" defined in this standard are used as vehicles for discussion of the bus protocol, and need not be considered a constraint to logic design. For example, the designer might choose to design logic which interacts with the IEC 821 BUS in the manner described, but uses different on board signals, or monitors other IEC 821 BUS signals. IEC 821 BUS boards might be designed to include any combination of the functional modules defined by this standard.

The IEC 821 BUS functional structure can be divided into four categories. Each consists of a bus and its associated functional modules which work together to perform specific duties. Figure 1-2, page 37, shows the IEC 821 BUS functional modules and buses. Each category is briefly summarized below.

a) Data Transfer

Devices transfer data over the Data Transfer Bus (DTB), which contains data and address pathways and associated control signals. Functional modules called MASTERS, SLAVES, INTERRUPTERS, and INTERRUPT HANDLERS use the DTB to transfer data between each other. Two other modules, called a BUS TIMER and an IACK DAISY-CHAIN DRIVER also assist them in this process.

b) DTB Arbitration

Since an IEC 821 BUS system can be configured with more than one MASTER or INTERRUPT HANDLER, a means is provided to transfer control of the DTB between them in an orderly manner and to guarantee that only one controls the DTB at a given time. The Arbitration Bus modules (REQUESTERS and ARBITER) coordinate the control transfer.

c) Priority Interrupt

The priority interrupt capability of the IEC 821 BUS provides a means by which devices can request service from an INTERRUPT HANDLER. These interrupt requests can be prioritized into a maximum of seven levels. INTERRUPTERS and INTERRUPT HANDLERS use the Priority Interrupt Bus signal lines.

d) Utilities

Periodic clocks, initialization, and failure detection are provided by the Utility Bus. It includes two clock lines, a system reset line, a system fail line, an a.c. fail line, and a serial data line.

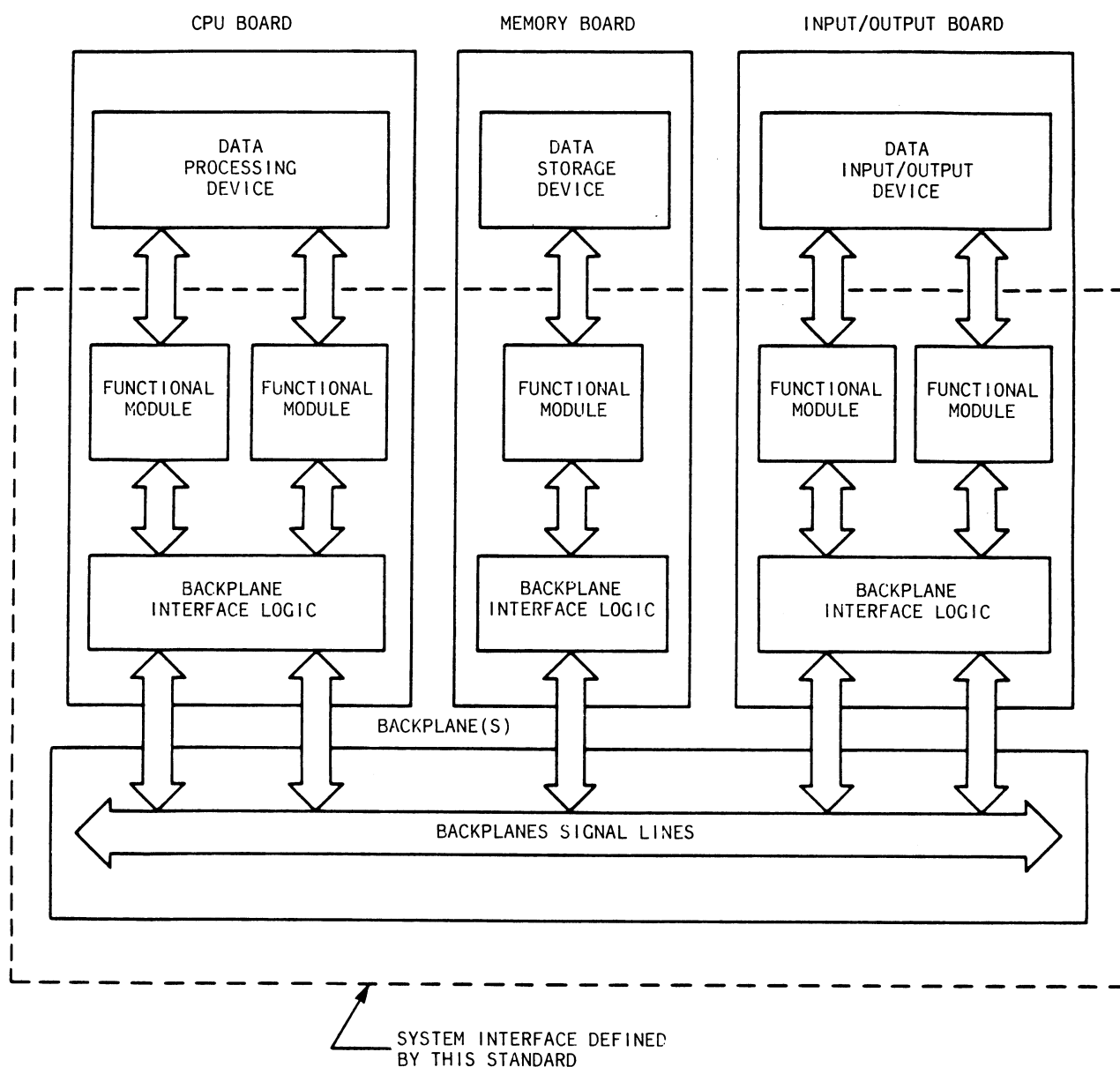


FIG. 1-1. - System elements defined by this standard.

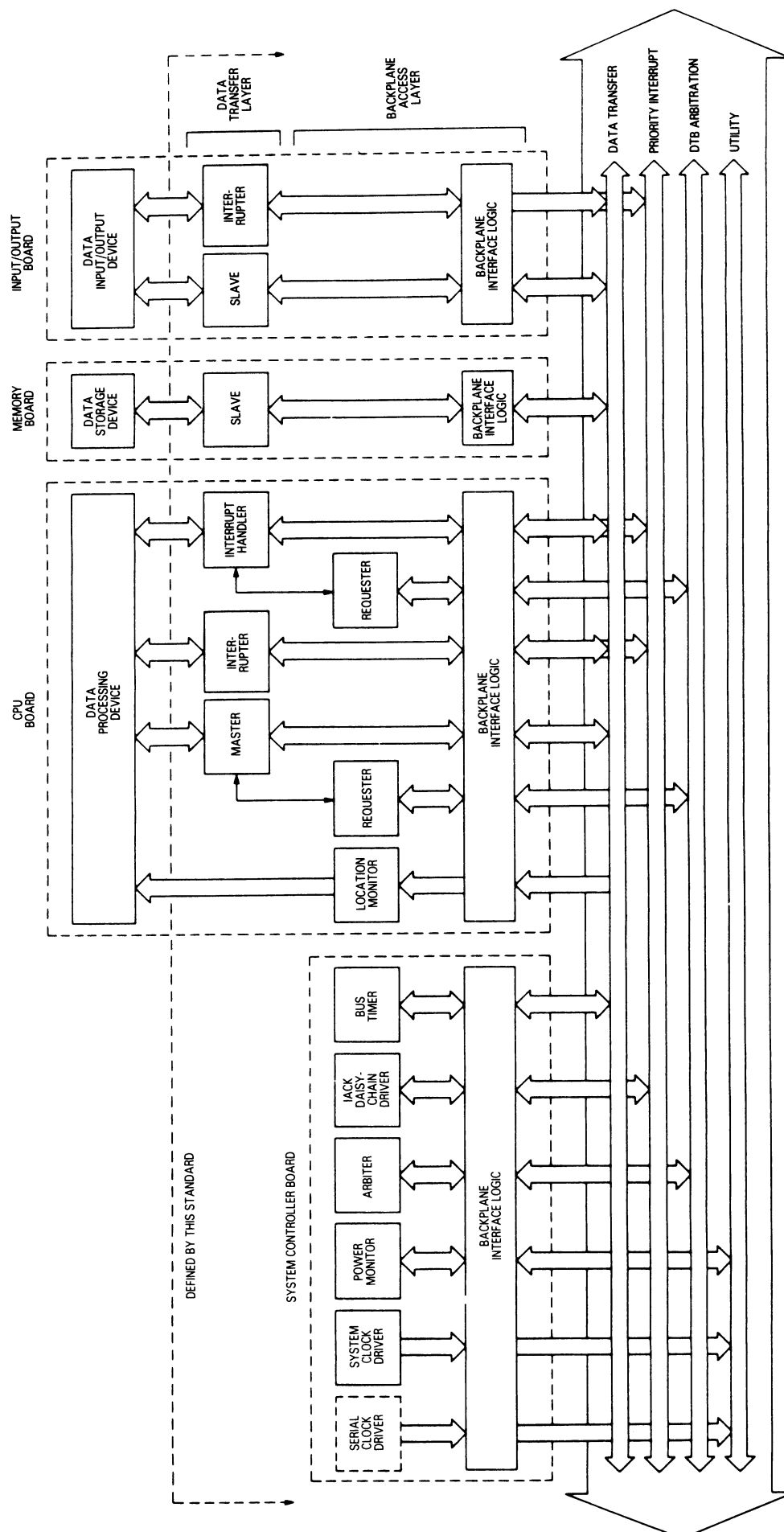


FIG. 1-2. - Functional modules and buses defined by this standard.

1.3 IEC 821 BUS standard diagrams

As aids to defining or describing IEC 821 BUS operation, several types of diagrams are used, including:

- a) Timing diagrams that show the timing relationships between signal transitions. The times involved will have minimum and/or maximum limits associated with them. Some of the times specified on these diagrams specify the behavior of the backplane interface logic, while other times specify the interlocked behavior of the functional modules.
- b) Sequence diagrams that are similar to a timing diagram but show only the interlocked timing relationships of the functional modules. This diagram is intended to show a sequence of events, rather than to specify the times involved. For example, a sequence diagram might indicate that module A cannot generate signal transition B until it detects module C's generation of signal transition D.
- c) Flow diagrams that show a stream of events as they would occur during an IEC 821 BUS operation. The events are stated in words and result from interaction of two or more functional modules. The flow diagram describes IEC 821 BUS operations in a sequential manner and, at the same time, shows interaction of the functional modules.

1.4 Standard terminology

To avoid confusion, and to make very clear what the requirements for compliance are, many of the paragraphs in this standard are labeled with keywords that indicate the type of information it contains. The keywords are listed below:

RULE
RECOMMENDATION
SUGGESTION
PERMISSION
OBSERVATION

Any text not labeled with one of these keywords describes the IEC 821 BUS structure or operation. It is written in either a descriptive or a narrative style. These keywords are used as follows:

RULE chapter.number:

Rules form the basic framework of the IEC 821 BUS standard. They are sometimes expressed in text form and sometimes in the form of figures, tables, or drawings. All IEC 821 BUS rules **MUST** be followed to ensure compatibility between IEC 821 BUS designs. Rules are characterized by an imperative style. The upper-case words **MUST** and **MUST NOT** are reserved exclusively for stating rules in this standard and are not used for any other purpose.

RECOMMENDATION chapter.number:

Wherever a recommendation appears, designers would be wise to take the advice given. Doing otherwise might result in some awkward problems or poor performance. While IEC 821 BUS has been designed to support high performance systems, it is possible to design an IEC

821 BUS system that complies with all the rules, but has abysmal performance. In many cases, a designer needs a certain level of experience with IEC 821 BUS in order to design boards that deliver top performance. Recommendations found in this standard are based on this kind of experience, and are provided to designers to speed their traversal of the learning curve.

SUGGESTION chapter.number:

In the IEC 821 BUS specification, a suggestion contains advice which is helpful but not vital. The reader is encouraged to consider the advice before discarding it. Some design decisions that need to be made in designing IEC 821 BUS boards are difficult until experience has been gained with the IEC 821 BUS. Suggestions are included to help a designer who has not yet gained this experience. Some suggestions have to do with designing boards that can be easily reconfigured for compatibility with other boards, or with designing the board to make the job of system debugging easier.

PERMISSION chapter.number:

In some cases an IEC 821 BUS rule does not specifically prohibit a certain design approach, but the reader might be left wondering whether that approach might violate the spirit of the rule, or whether it might lead to some subtle problem. Permissions reassure the reader that a certain approach is acceptable, and will cause no problems. The upper-case word MAY is reserved exclusively for stating permissions in this standard and is not used for any other purpose.

OBSERVATION chapter.number:

Observations do not offer any specific advice. They usually follow naturally from what has just been discussed. They spell out the implications of certain IEC 821 BUS rules and bring attention to things that might otherwise be overlooked. They also give the rationale behind certain rules, so that the reader understands why the rule must be followed.

1.4.1 *Signal line states*

The IEC 821 BUS standard describes its protocol in terms of levels and transitions on bus lines.

A signal line is always assumed to be in one of two levels or in transition between these levels. Whenever the term "high" is used, it refers to a high TTL voltage level. The term "low" refers to a low TTL voltage level. A signal line is "in transition" when its voltage is moving between these levels. (See Chapter 6 for voltage thresholds used on the IEC 821 BUS.)

There are two possible transitions which can appear on a signal line, and these are called "edges". A rising edge is the time during which a signal line makes its transition from a low level to a high level. The falling edge is the time during which a signal line makes its transition from a high level to a low level.

Some bus specifications prescribe maximum or minimum rise and fall times for these edges. The problem with doing this is that board designers have very little control over these times. If the backplane is heavily loaded, the rise and fall times will be long. If it is lightly loaded, these times might be short. Even if designers know what the maximum and minimum loading will be, they still need to spend time in

the lab, experimenting to find out which drivers will provide the needed rise and fall times.

In fact, rise and fall times are the result of a complex set of interactions involving the signal line impedances of the backplane, its terminations, the source impedance of the drivers, and the capacitive loading of the signal line. In order to trade off all of these factors, the board designer would have to study transmission line theory, as well as certain specific parameters of drivers and receivers which are not normally found in most manufacturers' data sheets.

Recognizing all of this, the IEC 821 BUS standard doesn't specify rise and fall times. Instead, it specifies the electrical characteristics for drivers and receivers and suggests how to design the backplane. It also tells designers how the worst case bus loading will affect the propagation delay of these drivers so that they can ensure that the IEC 821 BUS timing is met before building a board. If IEC 821 BUS designers follow these propagation delay guidelines, their boards will operate reliably with other IEC 821 BUS compatible boards under worst case conditions.

1.4.2 *Use of the asterisk (*)*

To help define usage, signal mnemonics have an asterisk suffix where required:

- a) An asterisk (*) following the signal name of signals which are level significant denotes that the signal is true or valid when the signal is low.
- b) An asterisk (*) following the signal name of signals which are edge significant denotes that the actions initiated by that signal occur on a high to low transition.

OBSERVATION 1.1:

The asterisk is inappropriate for the asynchronously running clock lines SYCLK and SERCLK. There is no fixed phase relationship between these clock lines and other IEC 821 BUS timing.

1.5 *Protocol specification*

There are two layers of IEC 821 BUS protocol. The lowest IEC 821 BUS layer, called the backplane access layer, is composed of the backplane interface logic, the Utility Bus modules, and the Arbitration Bus modules. The IEC 821 BUS data transfer layer is composed of the Data Transfer Bus and Priority Interrupt Bus modules. Figure 1-2, page 37, shows this layering.

OBSERVATION 1.2:

The signal lines used by the data transfer layer modules form a special class because they are driven by different modules at different times. They are driven with line drivers that can be turned on and off at each board, based upon signals generated in the backplane access layer. It is very important that their turn-on and turn-off times be

carefully controlled to prevent two drivers from attempting to drive the same signal line to different levels. Special timing diagram notation is used in this standard to specify their turn-on and turn-off times. It is shown in Figure 1-3, page 47.

There are two basic kinds of protocol used on the IEC 821 BUS: closed loop protocols and open loop protocols. Closed loop protocols use interlocked bus signals while open loop protocols use broadcast bus signals.

1.5.1 *Interlocked bus signals*

An interlocked bus signal is sent from a specific module to another specific module. The signal is acknowledged by the receiving module. An interlocked relationship exists between the two modules until the signal is acknowledged. For example, an INTERRUPTER can send an interrupt request which is answered later with an interrupt acknowledge signal (no time limit is prescribed by the IEC 821 BUS standard). The INTERRUPTER doesn't remove the interrupt request until the INTERRUPT HANDLER acknowledges it.

Interlocked bus signals coordinate internal functions of the IEC 821 BUS system, as opposed to interacting with external stimuli. Each interlocked signal has a source module and a destination module within the IEC 821 BUS system.

The address strobe and data strobes are especially important interlocking signals. They are interlocked with the data transfer acknowledge and bus error signals and coordinate the transfer of addresses and data which are the basis for all information flow between modules in the data transfer layer.

1.5.2 *Broadcast bus signal*

A module generates a broadcast signal in response to an event. There is no protocol for acknowledging a broadcast signal. Instead, the broadcast is maintained for a minimum specified time, long enough to assure that all appropriate modules detect the signal. Broadcast signals might be activated at any time, irrespective of any other activity taking place on the bus. They are each sent over a dedicated signal line. Some examples are the system reset and a.c. failure lines. These signal lines are not sent to any specific module, but announce special conditions to all modules.

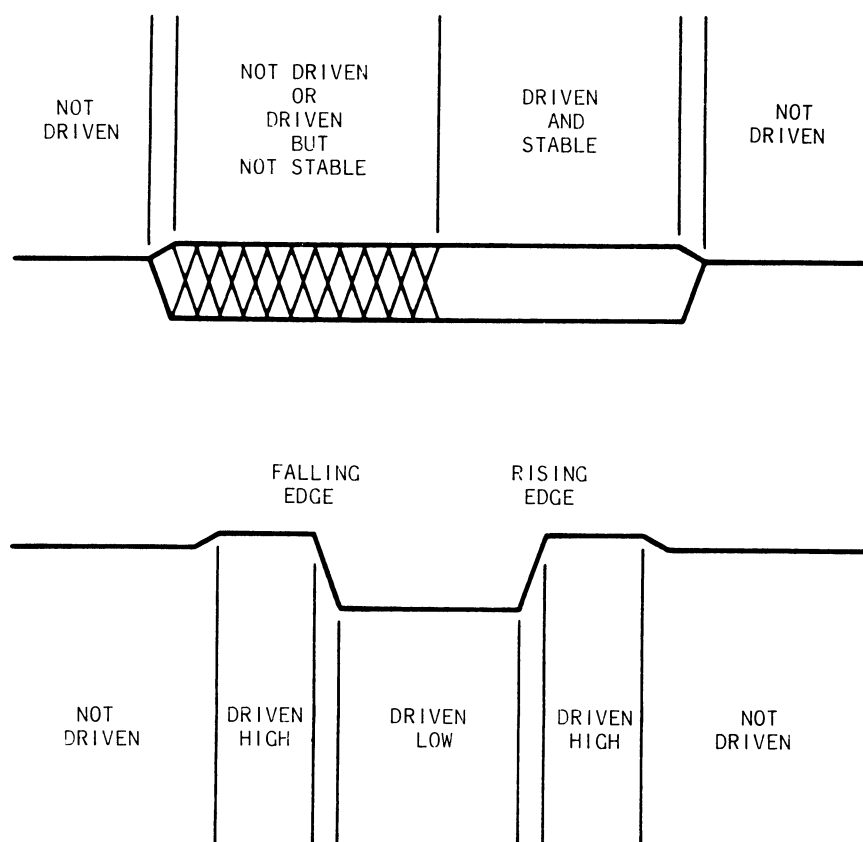


FIG. 1-3. - Signal timing notation.

1.6 System examples and explanations

A protocol specification describes, in detail, the behavior of the various functional modules. It discusses how a module responds to a signal without saying where the signal came from. Because of this, a protocol specification does not give the reader a complete picture of what is going on over the bus. To help the reader, the IEC 821 BUS standard provides examples of typical IEC 821 BUS operations. Each example shows one possible sequence of events: other sequences are also possible. In providing these examples, there is the danger that readers will assume that the sequence shown in the example is the only legal one. To help readers avoid this trap, all examples are given in a narrative style, using the present tense. This is in contrast to the imperative style used when giving rules for compliance with the IEC 821 BUS standard.

CHAPTER 2: IEC 821 BUS DATA TRANSFER BUS

2.1 Introduction

The IEC 821 BUS includes a high speed asynchronous parallel Data Transfer Bus (DTB). Figure 2-1, page 51, shows a typical IEC 821 BUS system, including all of the DTB functional module types. MASTERS use the DTB to select storage locations provided by SLAVES, and to transfer data to or from those locations. Some MASTERS and SLAVES use all of the DTB lines, while others use only a subset.

LOCATION MONITORS monitor data transfers between MASTERS and SLAVES. When an access is done to one of the byte location(s) that it monitors, a LOCATION MONITOR generates an on-board signal. For example, it might signal its on-board processor by means of an interrupt request. In such a configuration, if processor board A writes into a location of the global IEC 821 BUS memory that is monitored by processor B's LOCATION MONITOR, processor B will be interrupted.

After a MASTER initiates a data transfer cycle it waits for the responding SLAVE to respond before finishing the cycle. The asynchronous definition of the IEC 821 BUS allows a SLAVE to take as long as it needs to respond. If a SLAVE fails to respond because of some malfunction, or if the MASTER accidentally addresses a location where there is no SLAVE, the BUS TIMER intervenes, allowing the cycle to be terminated.

2.2 Data Transfer Bus lines

The Data Transfer Bus lines can be grouped into three categories:

Addressing lines	Data lines	Control lines
A01-A31	D00-D31	AS*
AM0-AM5		DS0*
DS0*		DS1*
DS1*		BERR*
LWORD*		DTACK*
		WRITE*

OBSERVATION 2.1:

The two data strobes (DS0* and DS1*) serve a dual function:

- 1) The levels of these two data strobe lines are used to select which byte(s) are accessed.
- 2) The edges of the data strobes are also used as timing signals which coordinate the transfer of the data between the MASTER and SLAVE.

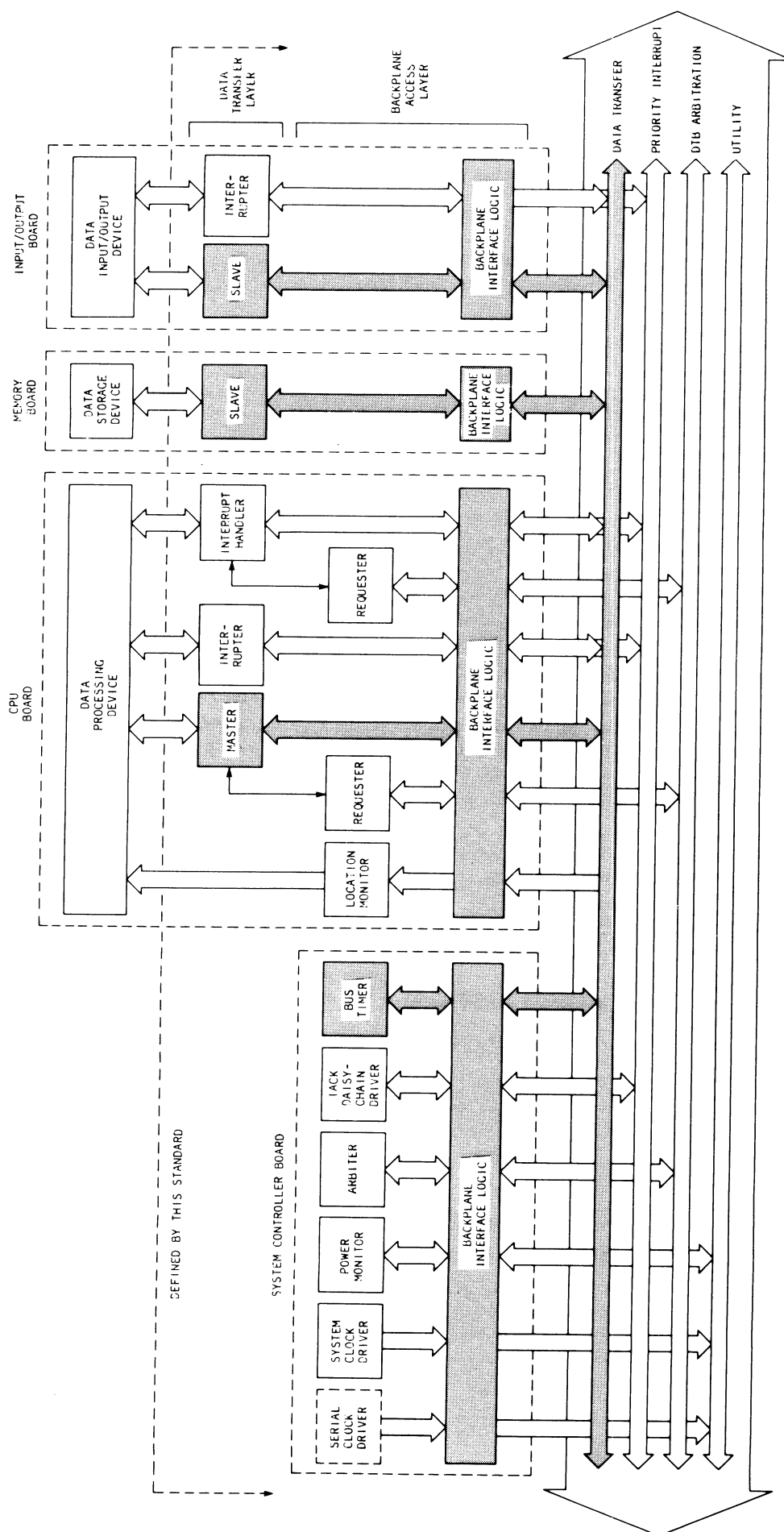


FIG. 2-1. - Data Transfer Bus functional block diagram.

2.2.1 Addressing lines

The smallest addressable unit of storage is the byte location. Each byte location is assigned a unique binary address. Each byte location can be assigned to one of four categories, according to the least significant two bits of its address (see Table 2-1).

Table 2-1

The four categories of byte location

Category	Byte address
BYTE(0)	XXXXXX XXXXXX00
BYTE(1)	XXXXXX XXXXXX01
BYTE(2)	XXXXXX XXXXXX10
BYTE(3)	XXXXXX XXXXXX11

A set of byte locations, whose addresses differs only in the two least significant bits, is called a 4-byte group or a BYTE(0-3) group. Some, or all, of the bytes in a 4-byte group can be accessed simultaneously by a single DTB cycle.

MASTERS use address lines A02-A31 to select which 4-byte group will be accessed. Four additional lines (DS1*, DS0*, A01 and LWORD*) are then used to select which byte location(s) within the 4-byte group are accessed during the data transfer. Using these four lines, a MASTER can access 1, 2, 3, or 4 byte locations simultaneously, depending upon the type of cycle. The 34 possible cycle types are listed in Table 2-2 with the corresponding levels of these four lines.

OBSERVATION 2.2:

In cycles where both data strobes are driven low, one data strobe might go low slightly after the other. In this case, the signal levels indicated in Table 2-2 are the final levels.

OBSERVATION 2.3:

Given the four signal line levels shown in Table 2-2, there are 16 possible combinations of levels. Of these 16, there are two illegal combinations that are not used (see table in RULE 2.1).

RULE 2.1:

MASTERS MUST NOT generate DTB cycles where the final levels of DS0*, DS1*, A01, and LWORD* are either of the following illegal combinations:

DS1*	DS0*	A01	LWORD*
high low	low high	high high	low low

PERMISSION 2.1:

MASTERS that generate BYTE(1-2) READ or BYTE(1-2) WRITE cycles MAY generate either of the two combinations described in RULE 2.1 briefly as transition states (i.e. while one data strobe has fallen, but the other has not).

OBSERVATION 2.4:

Whenever a MASTER drives LWORD* low and A01 high it drives both data strobes low. (Any other combination is illegal.) IEC 821 BUS board designers can take advantage of this to simplify the logic on SLAVES.

PERMISSION 2.2:

To simplify the required logic, SLAVES which respond to BYTE(1-2) READ and BYTE(1-2) WRITE cycles MAY be designed without logic to distinguish between these cycles and the two illegal cycles described in RULE 2.1.

Table 2-2

*Signal levels used to select which byte location(s)
are accessed during a data transfer*

Type of cycle	DS1*	DS0*	A01	LWORD*
ADDRESS-ONLY	high	high	Note 1	
Single even byte transfers BYTE(0) READ or WRITE BYTE(2) READ or WRITE	low low	high high	low high	high high
Single odd byte transfers BYTE(1) READ or WRITE BYTE(3) READ or WRITE	high high	low low	low high	high high
Double byte transfers BYTE(0-1) READ or WRITE BYTE(2-3) READ or WRITE	low low	low low	low high	high high
Quad byte transfers BYTE(0-3) READ or WRITE	low	low	low	low
Single byte block transfers SINGLE BYTE BLOCK READ or WRITE	Note 2			high
Double byte block transfers DOUBLE BYTE BLOCK READ or WRITE	low	low	Note 3	high
Quad byte block transfers QUAD BYTE BLOCK READ or WRITE	low	low	low	low
Single byte RMW transfers BYTE(0) READ-MODIFY-WRITE BYTE(1) READ-MODIFY-WRITE BYTE(2) READ-MODIFY-WRITE BYTE(3) READ-MODIFY-WRITE	low high low high	high low high low	low low high high	high high high high
Double byte RMW transfers BYTE(0-1) READ-MODIFY-WRITE BYTE(2-3) READ-MODIFY-WRITE	low low	low low	low high	high high
Quad byte RMW transfers BYTE(0-3) READ-MODIFY-WRITE	low	low	low	low
Unaligned transfers BYTE(0-2) READ or WRITE BYTE(1-3) READ or WRITE BYTE(1-2) READ or WRITE	low high low	high low low	low low high	low low low

Notes:

- 1.- During ADDRESS-ONLY cycles, both data strobes are maintained high, but the A01 and LWORD* lines might be either high or low.

Table 2-2 (continued)

- 2.- During single byte block transfers, the two data strobes are alternately driven low. Either data strobe might be driven low on the first transfer. If the first accessed byte location is BYTE(0) or BYTE(2), then DS1* is driven low first. If the first accessed byte location is BYTE(1) or BYTE(3), then DS0* is driven low first. A01 is valid only on the first data transfer (i.e. until the SLAVE drives DTACK* or BERR* low the first time) and might be either high or low depending upon which byte the single byte block transfer begins with. If the first byte location is BYTE(0) or BYTE(1), then A01 is low. If the first byte location is BYTE(2) or BYTE(3), then A01 is high.

An example of a single byte block transfer cycle which starts with BYTE(2) is given below:

	DS1*	DS0*	A01	LWORD*
First data transfer	low	high	high	high
BYTE(2)	high	low	X	X
BYTE(3)	low	high	X	X
BYTE(0)	high	low	X	X
Last data transfer	low	high	X	X
BYTE(1)	high	low	X	X
BYTE(2)	low	high	X	X

X = high or low.

- 3.- During a double byte block transfer, the two data strobes are both driven low on each data transfer. A01 is valid only on the first data transfer (i.e. until the SLAVE drives DTACK* or BERR* low the first time) and might be either high or low depending upon what double byte group the double byte block transfer begins with. If the first double byte group is BYTE(0-1), then A01 is low. If the first double byte group is BYTE(2-3), then A01 is high.

An example of a double byte block transfer cycle which starts with BYTE(2-3) is given below:

	DS1*	DS0*	A01	LWORD*
First data transfer	low	low	high	high
BYTE(2-3)	low	low	X	X
BYTE(0-1)	low	low	X	X
Last data transfer	low	low	X	X
BYTE(2-3)	low	low	X	X
BYTE(0-1)	low	low	X	X

X = high or low.

2.2.2 Address modifier lines

There are six address modifier lines. They allow the MASTER to pass additional binary information to the SLAVE during data transfers. Table 2-3 lists all of the 64 possible address modifier (AM) codes and classifies each into one of three categories:

Defined
Reserved
User defined

The defined address modifier codes can be further classified into three categories:

Short addressing AM codes indicate that address lines A02-A15 are being used to select a BYTE(0-3) group.

Standard addressing AM codes indicate that address lines A02-A23 are being used to select a BYTE(0-3) group.

Extended addressing AM codes indicate that address lines A02-A31 are being used to select a BYTE(0-3) group.

Table 2-3
Address modifier codes

HEX CODE	ADDRESS MODIFIER						FUNCTION
	5	4	3	2	1	0	
3F	H	H	H	H	H	H	Standard supervisory block transfer
3E	H	H	H	H	H	L	Standard supervisory program access
3D	H	H	H	H	L	H	Standard supervisory data access
3C	H	H	H	H	L	L	Reserved
3B	H	H	H	L	H	H	Standard non-privileged block transfer
3A	H	H	H	L	H	L	Standard non-privileged program access
39	H	H	H	L	L	H	Standard non-privileged data access
38	H	H	H	L	L	L	Reserved
37	H	H	L	H	H	H	Reserved
36	H	H	L	H	H	L	Reserved
35	H	H	L	H	L	H	Reserved
34	H	H	L	H	L	L	Reserved
33	H	H	L	L	H	H	Reserved
32	H	H	L	L	H	L	Reserved
31	H	H	L	L	L	H	Reserved
30	H	H	L	L	L	L	Reserved
2F	H	L	H	H	H	H	Reserved
2E	H	L	H	H	H	L	Reserved
2D	H	L	H	H	L	H	Short supervisory access
2C	H	L	H	H	L	L	Reserved
2B	H	L	H	L	H	H	Reserved
2A	H	L	H	L	H	L	Reserved
29	H	L	H	L	L	H	Short non-privileged access
28	H	L	H	L	L	L	Reserved
27	H	L	L	H	H	H	Reserved
26	H	L	L	H	H	L	Reserved
25	H	L	L	H	L	H	Reserved
24	H	L	L	H	L	L	Reserved
23	H	L	L	L	H	H	Reserved
22	H	L	L	L	H	L	Reserved
21	H	L	L	L	L	H	Reserved
20	H	L	L	L	L	L	Reserved
1F	L	H	H	H	H	H	User defined
1E	L	H	H	H	H	L	User defined
1D	L	H	H	H	L	H	User defined
1C	L	H	H	H	L	L	User defined
1B	L	H	H	L	H	H	User defined
1A	L	H	H	L	H	L	User defined
19	L	H	H	L	L	H	User defined
18	L	H	H	L	L	L	User defined
17	L	H	L	H	H	H	User defined
16	L	H	L	H	H	L	User defined
15	L	H	L	H	L	H	User defined
14	L	H	L	H	L	L	User defined
13	L	H	L	L	H	H	User defined
12	L	H	L	L	H	L	User defined
11	L	H	L	L	L	H	User defined
10	L	H	L	L	L	L	User defined
0F	L	L	H	H	H	H	Extended supervisory block transfer
0E	L	L	H	H	H	L	Extended supervisory program access
0D	L	L	H	H	L	H	Extended supervisory data access
0C	L	L	H	H	L	L	Reserved
0B	L	L	H	L	H	H	Extended non-privileged block transfer
0A	L	L	H	L	H	L	Extended non-privileged program access
09	L	L	H	L	L	H	Extended non-privileged data access
08	L	L	H	L	L	L	Reserved
07	L	L	L	H	H	H	Reserved
06	L	L	L	H	H	L	Reserved
05	L	L	L	H	L	H	Reserved
04	L	L	L	H	L	L	Reserved
03	L	L	L	L	H	H	Reserved
02	L	L	L	L	H	L	Reserved
01	L	L	L	L	L	H	Reserved
00	L	L	L	L	L	L	Reserved

L = Low signal level.
H = High signal level.

RULE 2.2:

Except for the user defined codes, the codes defined in Table 2-3 **MUST NOT** be used for purposes other than those specified.

RULE 2.3:

IEC 821 BUS SLAVE boards **MUST NOT** respond to reserved address modifier codes.

OBSERVATION 2.5:

Reserved address modifier codes are for future enhancements. If SLAVE boards respond to these codes, incompatibilities might result at some future date, when usage of these codes is defined.

PERMISSION 2.3:

User defined codes **MAY** be used for any purpose which board suppliers or board users deem appropriate (page switching, memory protection, MASTER or task identification, privileged access to resources, etc).

RECOMMENDATION 2.1:

To allow IEC 821 BUS users to tailor the use of the user defined address modifier codes to their own needs, decode them in a flexible way on SLAVE boards. Users can then configure the board to give any decoding required by their system.

OBSERVATION 2.6:

Socketed PROMS and FPLAS provide a flexible way for decoding the address modifier codes.

SUGGESTION 2.1:

Where SLAVES are manufactured with a programmed device (e.g. a PROM or FPLA) installed in the socket, program the device so that the SLAVE responds to the following AM codes:

A16 SLAVES with D08(O), D08(EO),	
	D16, or D32 capability: 29, 2D
A24 SLAVES with D08(O), D08(EO),	
	D16, or D32 capability: 39, 3A, 3D, and 3E
A32 SLAVES with D08(O), D08(EO),	
	D16, or D32 capability: 09, 0A, 0D, and 0E
A24 SLAVES with BLT capability:	3B, 3F
A32 SLAVES with BLT capability:	0B, 0F

The mnemonics A16, A24 and A32 are defined in Table 2-9.

The mnemonics D08(O), D08(EO), D16, D32, and BLT are defined in Tables 2-10 and 2-11.

2.2.3 Data lines

IEC 821 BUS systems can be built with a backplane configuration that provides either 16 data lines (D00-D15), or 32 data lines (D00-D31). Backplane configurations that provide 16 data lines allow a MASTER to access only two byte locations simultaneously, while those with 32 data lines allow up to four byte locations to be accessed. When the MASTER has selected 1, 2, 3, or 4 byte locations, using the method described in

Paragraph 2.2.1, it can transfer binary data between itself and those locations over the data bus. Table 2-4 shows how the data lines are used to move data during each of the 34 cycle types.

PERMISSION 2.4:

The data sender (MASTER for a write cycle; SLAVE for a read cycle) MAY drive data lines which are not used to transfer data.

Table 2-4

Use of the data lines to move data during each of the 34 cycle types

During the following types of cycles	The data lines are used to transfer data as shown below:			
	D24-D31	D16-D23	D08-D15	D00-D07
ADDRESS ONLY	No bytes transferred			
Single even byte transfers BYTE(0) READ or WRITE BYTE(2) READ or WRITE			BYTE(0) BYTE(2)	
Single odd byte transfers BYTE(1) READ or WRITE BYTE(3) READ or WRITE				BYTE(1) BYTE(3)
Double byte transfers BYTE(0-1) READ or WRITE BYTE(2-3) READ or WRITE			BYTE(0) BYTE(2)	BYTE(1) BYTE(3)
Quad byte transfers BYTE(0-3) READ or WRITE	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)
Single byte block transfers SINGLE BYTE BLOCK READ or WRITE			Note 1	
Double byte block transfers DOUBLE BYTE BLOCK READ or WRITE			Note 2	
Quad byte block transfers QUAD BYTE BLOCK READ or WRITE	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)

Table 2-4 (continued)

During the following types of cycles	The data lines are used to transfer data as shown below:			
	D24-D31	D16-D23	D08-D15	D00-D07
Single byte RMW transfers BYTE(0) READ-MODIFY-WRITE BYTE(1) READ-MODIFY-WRITE BYTE(2) READ-MODIFY-WRITE BYTE(3) READ-MODIFY-WRITE			BYTE(0) BYTE(2)	BYTE(1) BYTE(3)
Double byte RMW transfers BYTE(0-1) READ-MODIFY-WRITE BYTE(2-3) READ-MODIFY-WRITE			BYTE(0) BYTE(2)	BYTE(1) BYTE(3)
Quad byte RMW transfers BYTE(0-3) READ-MODIFY-WRITE	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)
Unaligned transfers BYTE(0-2) READ or WRITE BYTE(1-3) READ or WRITE BYTE(1-2) READ or WRITE	BYTE(0)	BYTE(1) BYTE(1) BYTE(1)	BYTE(2) BYTE(2) BYTE(2)	BYTE(3)

Notes:

- 1.- During single byte block transfers, data is transferred 8 bits at a time over D00-D07 or D08-D15. One example of this is given below:

	D24-D31	D16-D23	D08-D15	D00-D07
First data transfer			BYTE(2)	BYTE(1)
			BYTE(0)	BYTE(3)
			BYTE(2)	BYTE(1)
Last data transfer			BYTE(2)	BYTE(3)

- 2.- During a double byte block transfer, data is transferred 16 bits at a time over D00-D15. One example of this is given below:

	D24-D31	D16-D23	D08-D15	D00-D07
First data transfer			BYTE(2) BYTE(0)	BYTE(3) BYTE(1)
			BYTE(2) BYTE(0)	BYTE(3) BYTE(1)
			BYTE(2) BYTE(0)	BYTE(3) BYTE(1)
Last data transfer			BYTE(2) BYTE(0)	BYTE(3) BYTE(1)

2.2.4 Data Transfer Bus control lines

The following signal lines are used to control the movement of data over the data transfer lines:

AS* Address Strobe
DS0* Data Strobe Zero
DS1* Data Strobe One
BERR* Bus Error
DTACK* Data Transfer Acknowledge
WRITE* Read/Write

2.2.4.1 AS*

A falling edge on this line informs all SLAVE modules that the address is stable and can be captured.

2.2.4.2 DS0* and DS1*

In addition to their function in selecting byte locations for data transfer, as described in Paragraph 2.2.1, the data strobes also serve additional functions. On write cycles, the first data strobe falling edge indicates when the MASTER has placed valid data on the data bus. On read cycles, the first rising edge tells the SLAVE when it can remove valid data from the data bus.

OBSERVATION 2.7:

IEC 821 BUS MASTERS are not permitted to drive either of the data strobes low before driving AS* low. However, due to the fact that AS* might be more heavily loaded on the backplane than the data strobes, SLAVES and LOCATION MONITORS might detect a falling edge on a data strobe, before they detect the falling edge on AS*.

PERMISSION 2.5:

IEC 821 BUS SLAVES and LOCATION MONITORS MAY be designed to capture the address when they detect a falling edge on a data strobe instead of on the falling edge of AS*.

OBSERVATION 2.8:

IEC 821 BUS SLAVES and LOCATION MONITORS that capture the address on the falling edge of the data strobe(s) need not monitor AS*.

OBSERVATION 2.9:

In order to take full advantage of address pipelining as described in Paragraph 2.4.2, or to perform block read and write cycles, a SLAVE should capture the address on the falling edge of AS*.

2.2.4.3 DTACK*

The SLAVE drives DTACK* low to indicate that it has successfully received the data on a write cycle. On a read cycle, the SLAVE drives DTACK* low to indicate that it has placed data on the data bus.

2.2.4.4 BERR*

BERR* is driven low by the SLAVE or by the BUS TIMER to indicate to the MASTER that the data transfer was unsuccessful. For example, if a MASTER tries to write to a location which contains Read-Only Memory, the responding SLAVE might drive BERR* low. If the MASTER tries to access a location that is not provided by any SLAVE, the BUS TIMER would drive BERR* low after waiting a specified period.

OBSERVATION 2.10:

The BERR* line is a convenience which is useful when debugging IEC 821 BUS systems. It also allows system failures to be detected quickly during normal operation. Not all IEC 821 BUS systems will need this capability.

PERMISSION 2.6:

IEC 821 BUS SLAVES MAY be designed without a driver for BERR*.

SUGGESTION 2.2:

Design SLAVES to respond with a falling edge on BERR* in the following situations:

When a D08(O), D08(EO), or D16 SLAVE is requested to do a quad byte cycle.

When a D08(O) or D08(EO) SLAVE is requested to do a double byte cycle.

When a non-UAT SLAVE is requested to do an unaligned transfer (i.e. a triple byte transfer or a double byte BYTE(1-2) transfer).

When a SLAVE detects an uncorrectable error in the data it retrieves from its internal storage during a read cycle.

The mnemonics D08(O), D08(EO), D16, and UAT are defined in Tables 2-10 and 2-15.

RULE 2.4:

D08(O), D08(EO), and D16 SLAVES MUST NOT respond with a falling edge on DTACK* during a quad byte cycle if they do not have quad byte capability.

RULE 2.5:

D08(O) and D08(EO) SLAVES MUST NOT respond with a falling edge on DTACK* during a double byte cycle if they do not have double byte capability.

RULE 2.6:

A SLAVE MUST NOT respond with a falling edge on DTACK* during an unaligned transfer cycle, if it does not have UAT capability.

2.2.4.5 WRITE*

WRITE* is a level significant signal line that is strobed by the falling edge of the first data strobe driven low. It is used by the MASTER to indicate the direction of the data transfer operations. When WRITE* is driven low the data transfer direction is from the MASTER to the SLAVE.

2.3 DTB modules - Basic description

In addition to the ADDRESS-ONLY CYCLE, the DTB protocol defines 33 different cycle types that can be used to transfer data. Each of these 34 cycle types can be used in any of three addressing modes: short (16 bits), standard (24 bits), and extended (32 bits). The capabilities of MASTERS, SLAVES and LOCATION MONITORS are described by a list of mnemonics that show what cycle types they can generate, accept, or monitor respectively. (This will be described in more detail later in this chapter.)

Paragraphs 2.3.1 through 2.3.4 provide block diagrams for the four types of DTB functional modules: MASTER, SLAVE, LOCATION MONITOR, and BUS TIMER.

RULE 2.7:

Output signal lines shown with solid lines in Figures 2-2 through 2-5, pages 75 to 87, MUST be driven by the module, unless it would always drive them high.

OBSERVATION 2.11:

IF an output signal line is not driven,
THEN terminators on the backplane ensure that it is high.

RULE 2.8:

Input signal lines shown with solid lines in Figures 2-2 through 2-5 MUST be monitored and responded to in the appropriate fashion.

OBSERVATION 2.12:

RULES and PERMISSIONS for driving and monitoring signal lines shown with dotted lines in Figures 2-2 through 2-5 are given in Tables 2-5, 2-6, and 2-8.

2.3.1 MASTER

The block diagram of the MASTER is shown in Figure 2-2. The dotted lines in the diagram show signals whose use varies among the various types of MASTERS. Table 2-5 specifies how the various types of MASTERS use these lines. Further information about how the various types of MASTERS drive the address lines, the data lines, LWORD*, DS0*, and DS1* is given in Tables 2-20, 2-21, 2-22.

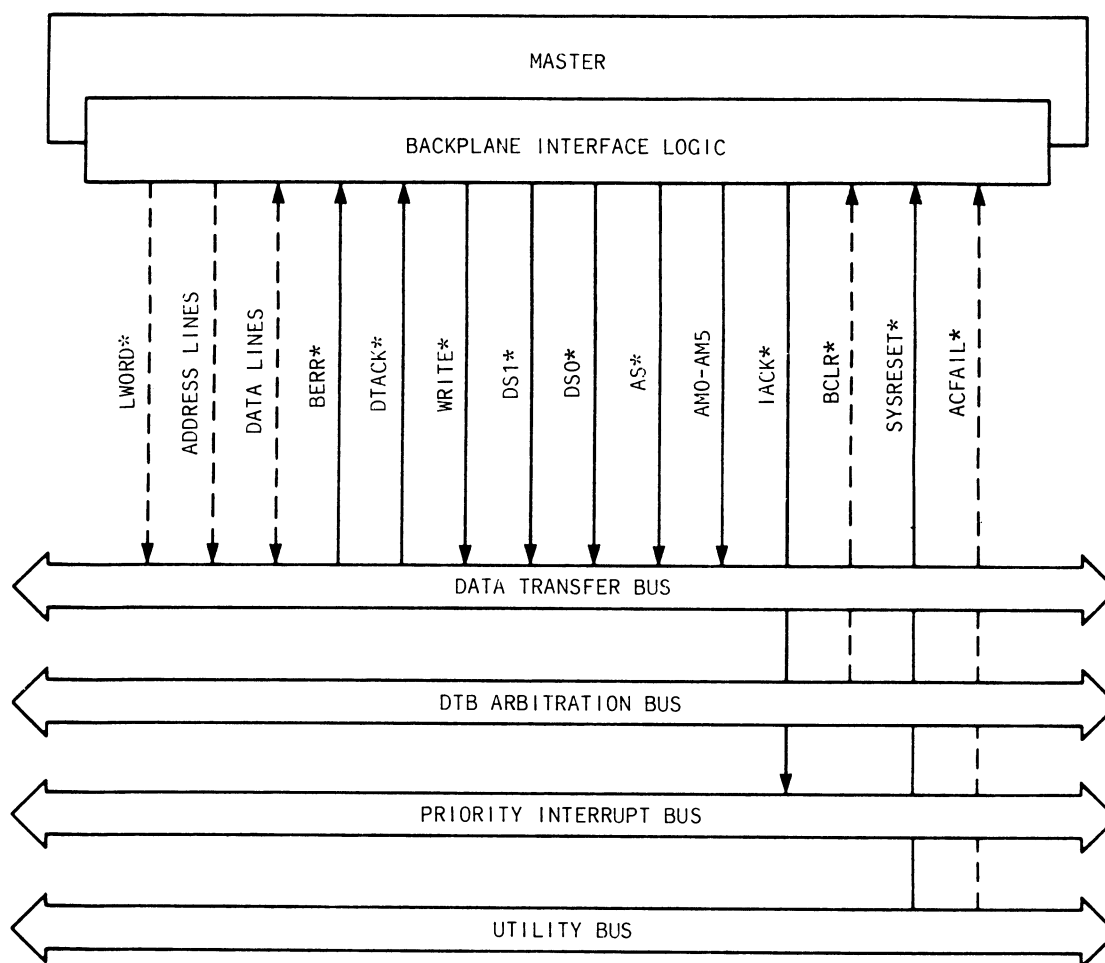


FIG. 2-2. - Block diagram: MASTER.

Table 2-5

RULES and PERMISSIONS that specify the use of the dotted lines by the various types of MASTERS

Type of MASTER	Use of dotted lines
D08(E0)	MUST monitor and drive D00-D15. MAY or MAY NOT drive LWORD*. MAY or MAY NOT drive or monitor D16-D31
D16	MUST monitor and drive D00-D15. MAY or MAY NOT drive LWORD*. MAY or MAY NOT drive or monitor D16-D31
D32	MUST drive LWORD*. MUST monitor and drive D00-D31. MUST NOT drive IACK* low
A16	MUST drive A01-A15. MAY or MAY NOT drive A16-A31
A24	MUST drive A01-A23 MAY or MAY NOT drive A24-A31
A32	MUST drive A01-A31
ALL	MAY or MAY NOT monitor BCLR*, or ACFAIL* (see Chapters 3 and 5)

Notes:

- 1.- The mnemonics D08(E0), D16, and D32 are defined in Table 2-10.
- 2.- The mnemonics A16, A24, and A32 are defined in Table 2-9.

2.3.2 SLAVE

The block diagram of the SLAVE is shown in Figure 2-3. The dotted lines in the diagram show signals whose use varies among the various types of SLAVES. Table 2-6 shows how the various types of SLAVES use these lines. Further information about how the various types of SLAVES drive the data lines is given in Table 2-22.

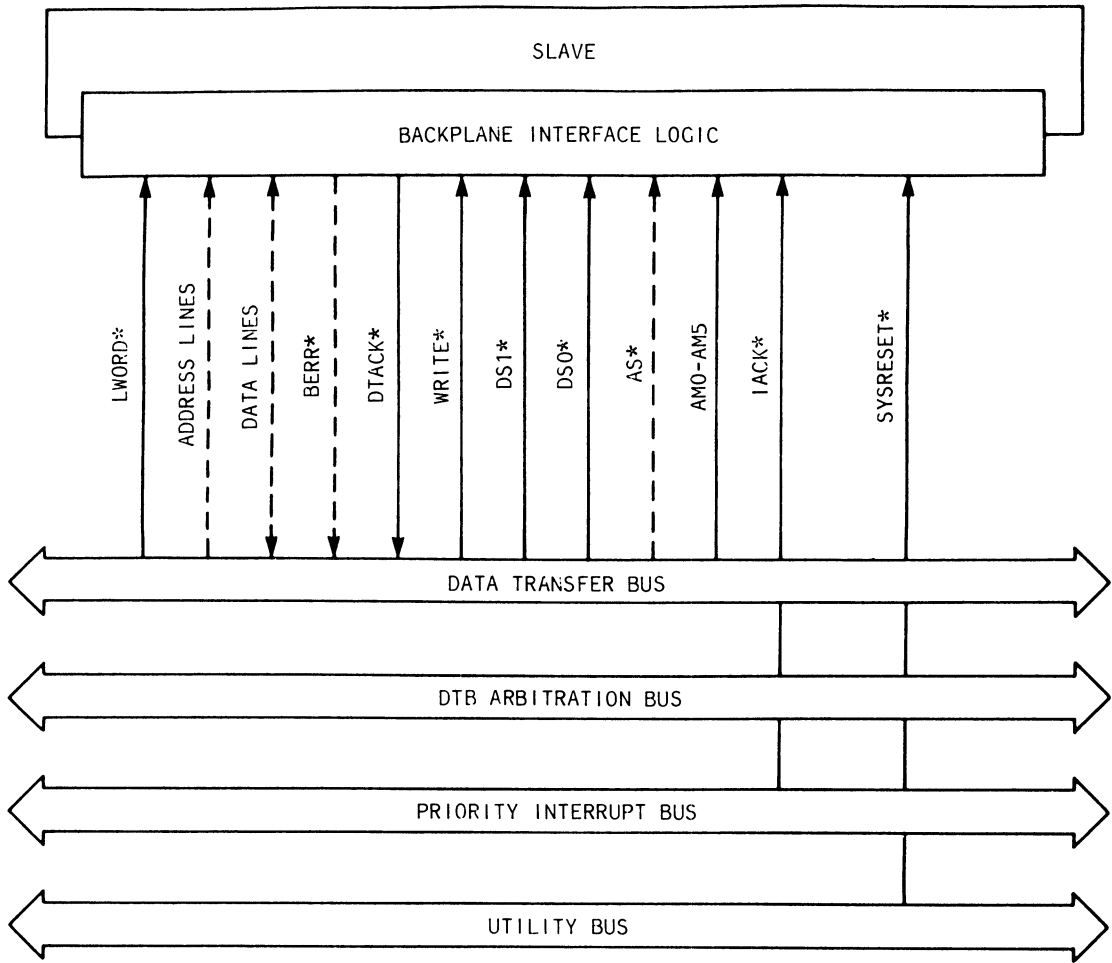


FIG. 2-3. - Block diagram: SLAVE.

Table 2-6

*RULES and PERMISSIONS that specify the use of the dotted lines
by the various types of SLAVES*

Type of SLAVE	Use of dotted lines
D08(0)	MUST monitor and drive D00-D07. MAY or MAY NOT monitor AS*. MAY or MAY NOT monitor or drive D08-D31
D08(E0)	MUST monitor and drive D00-D15. MAY or MAY NOT monitor AS*. MAY or MAY NOT monitor or drive D16-D31
D16	MUST monitor and drive D00-D15. MAY or MAY NOT monitor AS*. MAY or MAY NOT monitor or drive D16-D31
D32	MUST monitor and drive D00-D31. MAY or MAY NOT monitor AS*
A16	MUST monitor A01-A15. MAY or MAY NOT monitor A16-A31
A24	MUST monitor A01-A23. MAY or MAY NOT monitor A24-A31
A32	MUST monitor A01-A31
ALL	MAY or MAY NOT drive BERR*

Notes:

- 1.- The mnemonics D08(0), D08(E0), D16, and D32 are defined in Table 2-10.
- 2.- The mnemonics A16, A24, and A32 are defined in Table 2-9.

2.3.3 BUS TIMER

The block diagram of the BUS TIMER is shown in Figure 2-4. BUS TIMERS can be designed to drive BERR* low after various periods of time. Table 2-7 shows how the BT0() mnemonic is used to describe the various types of BUS TIMERS.

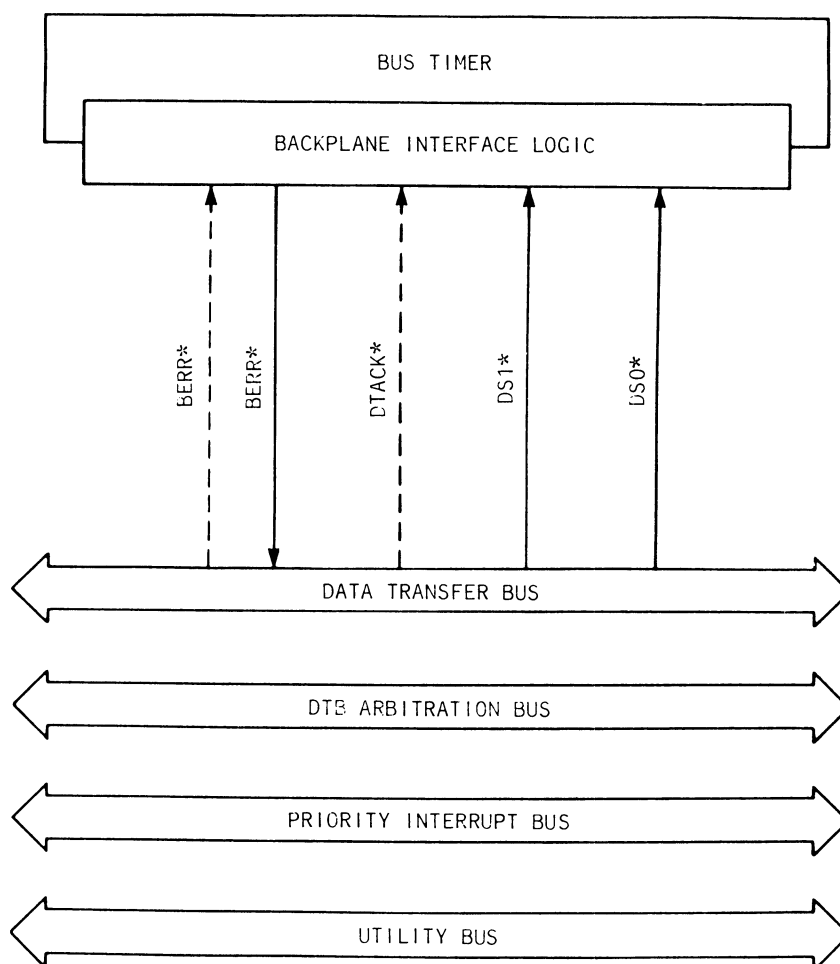


FIG. 2-4. - Block diagram: BUS TIMER.

Table 2-7

*Use of the BT0() mnemonic to specify
the time-out period of BUS TIMERS*

The following mnemonic	When applied to a	Means that it
BT0(x)	BUS TIMER	Drives BERR* low if the first data strobe stays low for longer than x microseconds (μ s)

OBSERVATION 2.13:

The dotted DTACK* and BERR* lines shown in Figure 2-4, page 83, allow to implement a BUS TIMER in one of two ways:

- a) to drive BERR* low when the first data strobe stays low for longer than the bus time-out period, regardless of the levels on the DTACK* and BERR* lines.
- b) to drive BERR* low when the first data strobe stays low for longer than the bus time-out period, but only if both DTACK* and BERR* are high at the point of time-out.

2.3.4 LOCATION MONITOR

The block diagram of the LOCATION MONITOR is shown in Figure 2-5. The dotted lines in the diagram show signals whose use varies among the various types of LOCATION MONITORS. Table 2-8 shows how the various types of LOCATION MONITORS use these lines.

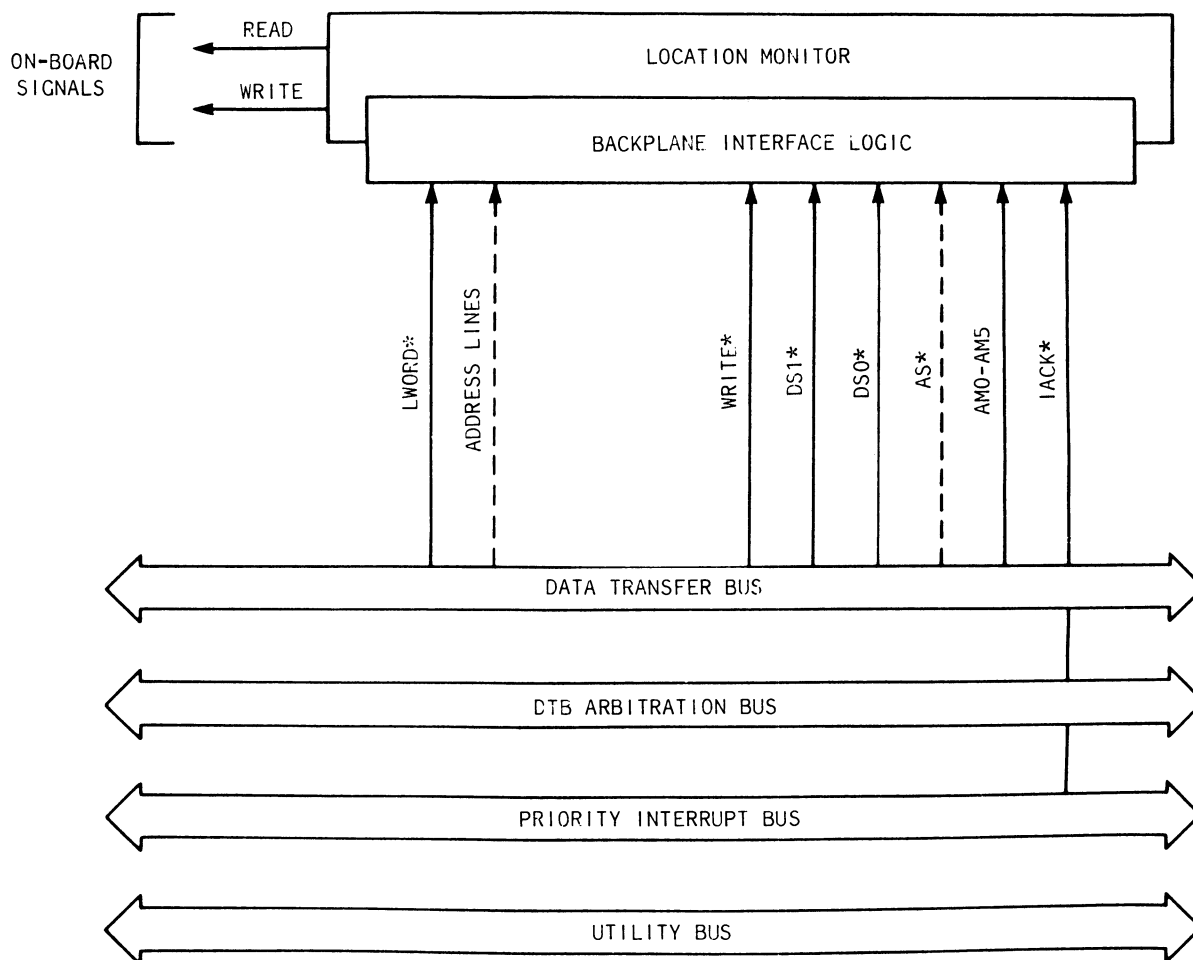


FIG. 2-5. - Block diagram: LOCATION MONITOR.

Table 2-8

*RULES and PERMISSIONS that specify the use of the dotted lines
by the various types of LOCATION MONITORS*

Type of LOCATION MONITOR	Use of dotted lines
A16	MUST monitor A01-A15. MAY or MAY NOT monitor A16-A31
A24	MUST monitor A01-A23. MAY or MAY NOT monitor A24-A31
A32	MUST monitor A01-A31
ALL	MAY or MAY NOT monitor AS*

Note:

The mnemonics A16, A24 and A32 are defined in Table 2-9.

2.3.5 Addressing modes

MASTERS broadcast an address over the DTB at the beginning of each cycle. This broadcasted address might be a 16, 24 or 32-bit address, depending upon the capabilities of the MASTER broadcasting it. The 16-bit addresses are "short addresses", the 24-bit addresses are "standard addresses", and the 32-bit addresses are "extended addresses".

Table 2-9 shows the various mnemonics used to describe the addressing capabilities, and how each is used to describe MASTERS, SLAVES, and LOCATION MONITORS.

Table 2-9

Mnemonics that specify addressing capabilities

The following mnemonic	When applied to a	Means that it
A16	MASTER	Can generate cycles with short (16-bit) addresses.
	SLAVE	Can accept cycles with short (16-bit) addresses.
	LOCATION MONITOR	Can monitor cycles with short (16-bit) addresses
A24	MASTER	Can generate cycles with standard (24-bit) addresses.
	SLAVE	Can accept cycles with standard (24-bit) addresses.
	LOCATION MONITOR	Can monitor cycles with standard (24-bit) addresses
A32	MASTER	Can generate cycles with extended (32-bit) addresses.
	SLAVE	Can accept cycles with extended (32-bit) addresses.
	LOCATION MONITOR	Can monitor cycles with extended (32-bit) addresses

The MASTER broadcasts an Address Modifier (AM) code along with each address to tell SLAVES whether the address is short, standard, or extended.

Short addresses are generated by A16 MASTERS, and accepted by A16 SLAVES. Standard addresses are generated by A24 MASTERS, and accepted by A24 SLAVES. Extended addresses are generated by A32 MASTERS, and accepted by A32 SLAVES.

Short addressing is intended primarily for addressing input/output (I/O) devices. It allows A16 SLAVES to be designed with less logic, since they do not have to decode as many address lines. While I/O boards can be designed to decode standard addresses and extended addresses, short addressing usually makes this unnecessary.

Standard and extended addressing modes are intended primarily for addressing memory, although there is no rule against designing I/O boards that also respond to these addressing modes. Standard and extended addressing modes allow much larger addressing ranges.

RULE 2.9:

SLAVE boards MUST decode all of the address modifier lines.

OBSERVATION 2.14:

RULE 2.9 allows a SLAVE to differentiate short addresses, standard addresses, and extended addresses.

OBSERVATION 2.15:

In addition to the three modes of addressing described here, there is a fourth mode which is used on interrupt acknowledge cycles (see Chapter 4). These interrupt acknowledge cycles can be distinguished from data transfer cycles by the fact the the IACK* signal line is low instead of high.

RULE 2.10:

Whenever a MASTER broadcasts an address over the address bus, it MUST ensure that IACK* is high.

PERMISSION 2.7:

A MASTER MAY either drive IACK* high during the address broadcast or it MAY leave IACK* undriven. (The bus terminators will then hold it high.)

RULE 2.11:

SLAVES MUST NOT respond to DTB cycles when IACK* is low.

RECOMMENDATION 2.2:

Since many systems will include a mixture of A16, A24, and A32 SLAVES, include A16 and A24 capability on 32-bit CPU cards in addition to the A32 capability and include A16 capability on CPU cards with A24 capability.

2.3.6 Basic data transfer capabilities

There are four basic data transfer capabilities associated with the DTB: D08(E0) (Even and Odd byte), D08(O) (Odd byte only), D16, and D32. These capabilities allow flexibility when interfacing different types of processors and peripherals to the bus.

Eight-bit processors can be interfaced to the bus as D08(E0) MASTERS. Sixteen-bit processors can be interfaced to the bus as D16 MASTERS. The D16 SLAVE module is useful for interfacing 16-bit memory devices or 16-bit I/O SLAVES to the DTB.

Many existing peripheral chips have registers that are only 8 bits wide. While these chips often have several of these registers, they cannot provide the contents of two registers simultaneously when a D16 MASTER attempts to access two adjacent locations with a double byte read cycle. These 8-bit peripheral ICs can be interfaced to the DTB as a D08(O) SLAVE. D08(O) SLAVES provide only BYTE(1) or BYTE(3) locations and respond only to single byte accesses. (Since single byte accesses to these odd byte locations always take place over D00-D07, this simplifies the D08(O) SLAVE'S interface logic.)

RECOMMENDATION 2.3:

Since most 16-bit microprocessors can also access memory 8 bits at a time, include D08(E0) MASTER capability on 16-bit CPU boards in addition to the D16 capability. In addition to allowing 8-bit data transfers to and from memory, this also allows them to access D08(O) SLAVES.

RECOMMENDATION 2.4:

Since most 32-bit microprocessors can also transfer data to and from memory 8 and 16 bits at a time, include D08(E0) and D16 capability on 32-bit CPU boards in addition to D32 capability. The D08(E0) capability not only allows 8-bit data transfers to and from memory, but it also allows the MASTER to access D08(O) SLAVES.

OBSERVATION 2.16:

It might seem logical to define "Even byte only" SLAVES which respond to the even byte memory locations adjacent to the D08(O) SLAVES. But, this cannot be done because there is only one data transfer acknowledge line. If a MASTER were to select both an even byte and an odd byte location simultaneously, by doing a double byte transfer, both SLAVES would drive the same data acknowledge (DTACK*) line and the MASTER would not know whether both boards had acknowledged the access.

RECOMMENDATION 2.5:

Since many systems will include MASTERS with various data transfer capabilities, include D08(E0) and D16 capability on 32-bit SLAVE (memory) boards in addition to the D32 capability, and include D08(E0) capability on 16-bit SLAVE boards in addition to the D16 capability.

OBSERVATION 2.17:

Since D08(O) SLAVES respond only to odd byte addresses, they cannot provide contiguous memory. D08(O) SLAVES are useful only for I/O, status, or control registers, while D08(E0), D16 and D32 SLAVES are also useful for memory.

Table 2-10 shows the various mnemonics used to describe the basic data transfer capabilities, and how each is used to describe MASTERS, SLAVES, and LOCATION MONITORS.

Table 2-10

*Mnemonics that specify basic data
transfer capabilities*

The following mnemonic	When applied to a	Means that it
D08(E0)	MASTER SLAVE LOCATION MONITOR	Can generate the following cycles: Can accept the following cycles: Can monitor the following cycles: Single byte read cycles: BYTE(0) READ BYTE(1) READ BYTE(2) READ BYTE(3) READ Single byte write cycles: BYTE(0) WRITE BYTE(1) WRITE BYTE(2) WRITE BYTE(3) WRITE
D08(0)	SLAVE	Can accept the following cycles: Single byte read cycles: BYTE(1) READ BYTE(3) READ Single byte write cycles: BYTE(1) WRITE BYTE(3) WRITE
D16	MASTER SLAVE LOCATION MONITOR	Can generate the following cycles: Can accept the following cycles: Can monitor the following cycles: Double byte read cycles: BYTE(0-1) READ BYTE(2-3) READ Double byte write cycles: BYTE(0-1) WRITE BYTE(2-3) WRITE
D32	MASTER SLAVE LOCATION MONITOR	Can generate the following cycles: Can accept the following cycles: Can monitor the following cycles: Quad byte read cycle: BYTE(0-3) READ Quad byte write cycle: BYTE(0-3) WRITE

Note:

(E0) is Even Odd, and (0) is Odd only.

2.3.7 Block transfer capabilities

MASTERS often access several memory locations in ascending order. When this is the case, block transfer cycles are very useful. They allow the MASTER to provide a single address, and then access data in

that location and those at higher addresses, without providing additional addresses.

When a MASTER initiates a block transfer cycle, the responding SLAVE latches the address into an on-board address counter. The MASTER, upon completing the first data transfer (i.e. driving the data strobes high) does not allow the address strobe to go high. Instead, it repeatedly drives the data strobe(s) low in response to data transfer acknowledgements from the SLAVE, and transfers data to or from sequential memory locations in ascending order.

To access the next location(s), the SLAVE increments an on-board counter that generates the address for each transition of the data strobe(s).

OBSERVATION 2.18:

Block transfer cycles of indefinite length are not allowed, because this complicates the design of memory boards. Specifically, all block transfer SLAVES (the one that responds, as well as those that do not) would need to latch the initial address and then increment the address counter on each bus transfer. All SLAVES would then have to decode the incremented address to see if the block transfer has crossed a board boundary into their address range. While this is certainly possible, such address decoding typically limits access times of the SLAVE. To simplify the design of these SLAVES, and to permit faster access times, RULE 2.12 has been formulated.

RULE 2.12:

Block transfer cycles **MUST NOT** cross any 256-byte boundary.

OBSERVATION 2.19:

RULE 2.12 limits the maximum length of block transfers to 256 bytes. However, knowing that only A01 through A07 will change during the course of the block transfer simplifies the design of block transfer SLAVES. The upper address lines only have to be decoded once, at the beginning of the block transfer cycle, allowing much faster access times on all subsequent data transfers.

OBSERVATION 2.20:

In some cases it might be necessary to transfer a large block of data which crosses one or more 256-byte boundaries; in such a case if the hardware on boards which does the block transfers is designed to recognize the arrival at a 256-byte boundary, it can momentarily drive AS* high and then initiate another block transfer without the intervention of system software.

The block read cycle is very similar to a string of read cycles. Likewise, the block write cycle is very similar to a string of write cycles. The difference is that only the initial address is broadcast by the MASTER and the address strobe is held low during all of the data transfers.