Table of Contents

FIGURES

FIGURES (Con't)

TABLES

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

Intel software products are copyrighted by and shall remain the property of Intel Corporation. Use, duplication or disclosure is subject to restrictions stated in Intel's software license, or as defined in ASPR 7-104.9(a)(9).

No part of this document may be copied or reproduced in any form or by any means without prior written consent of Intel Corporation.

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BITBUS	iLBX	iPDS	Plug-A-Bubble
COMMputer	i_m	iPSB	Prompt
CREDIT	iMMX	iRMX	Promware
Data Pipeline	Insite	iSBC	QUEX
GENIUS	Int_el	iSBX	QUEST
$\overset{\Delta}{i}$	int_el	iSDM	Ripplemode
i	int_elBOS	iSSB	RMX/80
I^2ICE	Intelevision	iSXM	RUPI
ICE	$Int_eligent$ Identifier	Library Manager	Seamless
iCS	$Int_eligent$ Programming	MCS	SOLO
iDBP	Intellec	Megachassis	SYSTEM 2000
iDIS	Intelink	MICROMAINFRAME	UPI
	iOSP	MULTIBUS	

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

Introduction to the Distributed Control Modules

Overview

Intel's Distributed Control Module (iDCM) products provide building blocks for construction of real-time distributed control systems based on the BITBUS™ interconnect. This new serial bus architecture addresses many of the limitations inherent in traditional connection methods. For instance, future system cost reductions are limited in systems based on parallel bus structures due to their electrical and mechanical characteristics. Other traditional connection methods such as current loops and RS 232 C do not provide sufficient performance or flexibility for complex industrial control applications. In addition, although there are numerous industry standards for connecting microprocessors, the MULTIBUS® and the STD-bus for example, there is no standard connection for microcontrollers. The BITBUS interconnect (Table 1) combines existing standards with new standard interfaces to provide the optimal solution for difficult distributed control problems.

The iDCM products combine hardware and software for use in applications that would benefit most from employing distributed architectures. Applications such as robotics, process control, data acquisition and control, and environmental control are a few examples.

Table 1. Standard BITBUS™ Interfaces

Interface	Specification
Electrical	RS485
Cable	10-conductor flat ribbon or 1 to 2 wire twisted pair
Back-plane connector	64-pin Standard DIN
End-cable connector	3M #3446-1302 female
Control-board form-factor	Single-height, Double-depth Eurocard
Data Link control	Synchronous Data-link Control
Data transfer rate	62.5K baud, 375K baud and 2.4M baud
Message formats	Compatible with iRMX 51 format command/response/status
Common command sequences	Integral Remote Access and Control (RAC) function
Operating systems	S/W drivers for iRMX 86, 88, 286R and ISIS (for iPDS only)

Benefits of Distributed Architectures

Distributed architectures are intrinsically more reliable than centralized architectures. In a centralized control system a central controller failure results in a system-wide failure. Distributed systems can be configured to prevent this. Also, distributed systems are more cost effective and more easily modified. For instance, performance improvements in centralized systems are expensive and do not concentrate improvements in the areas where they are needed most. In distributed systems, only the specific parts of the system that require enhancement need be modified. Most importantly, control systems based on distributed architectures have less difficulty responding to the external environment because they have less to manage.

The BITBUS™ Interconnect

The BITBUS interconnect is a serial bus optimized for high speed transfer of short control messages in a hierarchical system. In order to provide an easy to use high performance serial interconnect, transparent to the applications programmer, high-level interfaces are specified. These interfaces include: the message structure and protocol for a multitasking environment, and a set of high-level commands for remote I/O access and application task control. As with traditional bus specifications, the electrical and data protocol levels have been defined.

The BITBUS interconnect supports up to 250 nodes and three bit rates dependent on application performance requirements. Different BITBUS segments may support different bit rates.

A Simple and Reliable Solution

The BITBUS architecture supplies the system designer with a simple and reliable foundation. Some key features of this architecture are: defined high-level interfaces that provide all communication and user program management, the reliable SDLC protocol, power-up diagnostics, standard industrial packaging, compact software and hardware provided in the high performance 12MHz 8044 microcontroller, and a board-level integrated solution. In addition, complex, expensive, and awkward connection problems are no longer a factor because the BITBUS interconnect is a serial bus requiring a simple twisted wire pair.

Open Systems — An Answer to Obsolescence

Intel's Open Systems philosophy requires systems be open to: future VLSI, all levels of integration, third party suppliers, and special requirements. In order to facilitate this design strategy, the BITBUS interconnect was developed as a standard microcontroller interface. The same benefits realized by users of Intel's MULTIBUS architecture will be realized by users of the BITBUS architecture — the ability to exploit VLSI technology without having to pay premiums for new system design, multiple supply sources, wide product selections, and competitive prices.

The Open Systems philosophy characterizes the iDCM product line. Distributed Control Modules are compatible (open) at three levels of integration: components, boards, and systems. This multilevel approach enables OEM's to adapt to new business environments and opportunities as VLSI technology evolves.

Distributed Control Modules

The iDCM product line consists of both software and hardware products: the iRMX™ 51 Executive, iRMX 510 Support Package, iSBX™ 344 BITBUS MULTIMODULE™ Board, and the iRCB 44/10 BITBUS Remote Controller Board. All iDCM hardware products include integral firmware to implement the high-level BITBUS interfaces: message formats, command sequences, and operating system environments.

iRMX™ 51 Executive

The iRMX 51 Executive is a compact, easy to use, software tool for development and implementation of applications built on the high performance 8-bit family of 8051 microcontrollers. A pre-configured version of the Executive is included in firmware of the two iDCM hardware products. During run-time, some of the services provided by this event driven Executive are: task scheduling, interrupt handling, and message passing. Streamlined code, the simple user interface and modular design of the iRMX 51 Executive enhance system reliability.

iRMX™ 510 iDCM Support Package

The iRMX 510 iDCM Support Package provides the software development and run-time support for BITBUS systems. Also included are the software interfaces for other operating systems: iRMX 86, iRMX 286R, iRMX 88, and the iPDS ISIS. These software interfaces ease integration of a BITBUS system into MULTIBUS or iPDS environments.

iSBX™ 344 BITBUS™ MULTIMODULE™ Board

The iSBX 344 board facilitates expansion of MULTIBUS and iPDS systems via the BITBUS interconnect. This board is the iDCM hardware interface. MULTIBUS system capabilities can be expanded to include low-cost remote control using the iSBX 344 MULTIMODULE board and the iRMX 510 iDCM Support Package. Also, BITBUS system capabilities can be expanded using this board with user supplied software. The iSBX 344 board's integral firmware reduces application development time, ensures real time response, eases system integration, and lowers system cost.

iRCB 44/10 BITBUS™ Remote Controller Board

The iRCB 44/10 BITBUS Remote Controller module is a low-end, single-board computer with 24 lines of parallel I/O. The board has a single-high Eurocard form factor with a DIN connector for increased reliablity and integration with standard industrial packaging. One iSBX I/O Expansion connector will accommodate one of many iSBX MULTIMODULE Boards for I/O expansion. Also, sockets for repeaters are provided for extending the BITBUS interconnect beyond the length limits of one BITBUS segment. This board lowers distributed system cost via the BITBUS interconnect support and the same integral firmware provided on the iSBX 344 BITBUS Controller MULTIMODULE board.

Expanding a MULTIBUS® System with Distributed Control Modules

An example of how a MULTIBUS system can be expanded with iDCM Modules is shown in Figures 1 and 2. Figure 1 shows a basic MULTIBUS system: processor board, memory module, and I/O controller. Figure 2 illustrates the expanded system. Some advantages of the expanded system follow: The burden on the central processor has been reduced, thereby increasing overall system performance. System cost reduction is realized because the BITBUS architecture removes the necessity of adding expensive centralized systems to handle increased performance demands. Also, the BITBUS architecture enables implementation of a more efficient and flexible system that is insensitive to the addition of more nodes, or changes in node job functions.

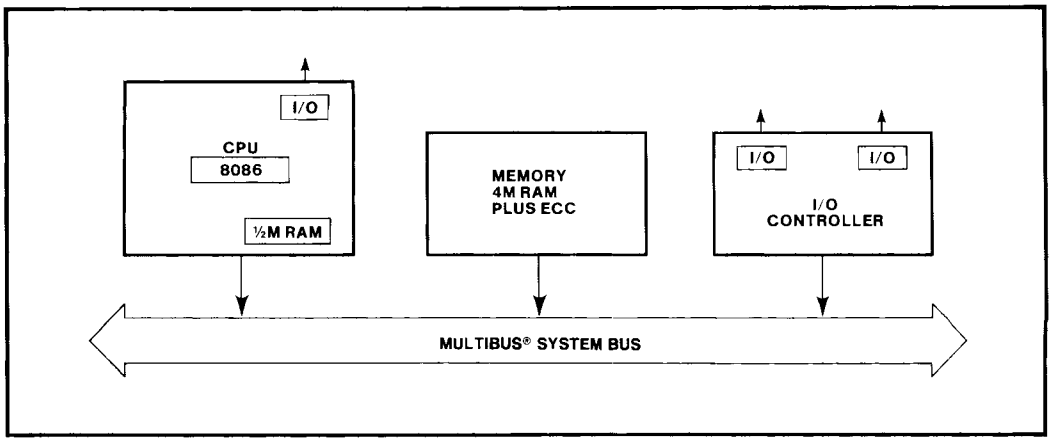


Figure 1. Basic MULTIBUS® System

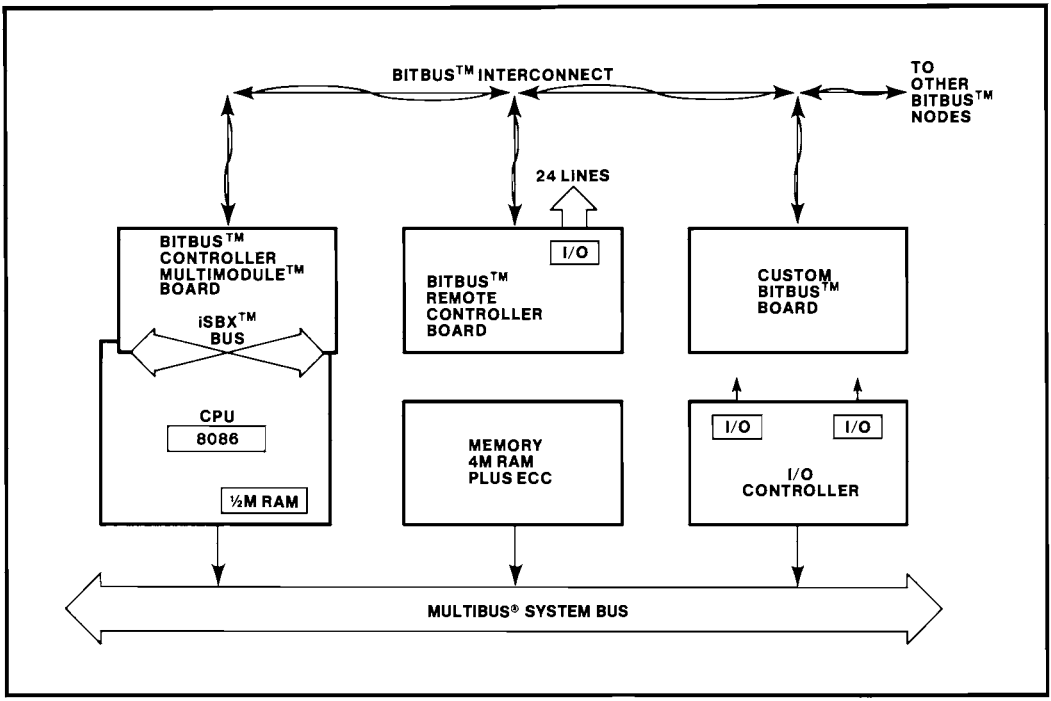


Figure 2. MULTIBUS® System Expanded with BITBUS™ Interconnect and Distributed Control Modules

Summary

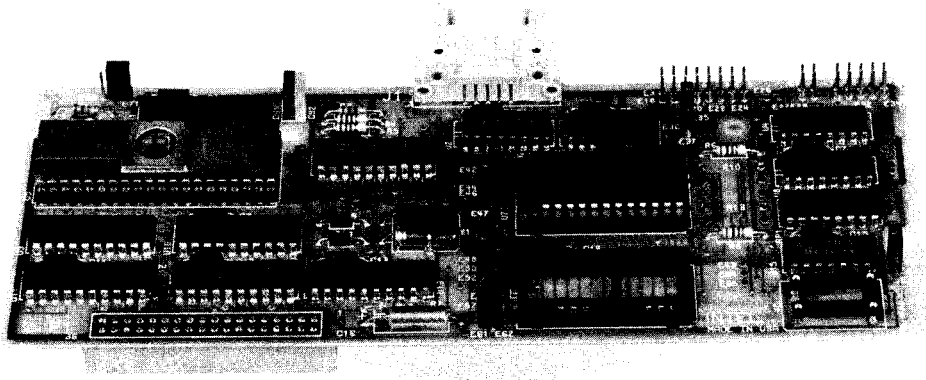
This overview has introduced the attributes and advantages of the BITBUS interconnect and Distributed Control Modules. Initial iDCM products (the iRMX 51 Executive, the iRMX 510 iDCM Support Package, iSBX 344 BITBUS Controller MULTIMODULE and iRCB 44/10 BITBUS Remote Controller boards) are intended to allow rapid assimilation of this new technology. Data sheets describing the individual iDCM products are included in the next section of this document. The final section presents the BITBUS Specification (supported by the iDCM products).

intel

iSBX™ 344
BITBUS™ CONTROLLER MULTIMODULE™ BOARD

- **High performance 12MHz 8044 controller**
- **Integral firmware including the iRMX™ 51 Executive optomized for real-time control applications**
- **Full BITBUS™ support**
- **2 28-pin JEDEC memory sites for user's control functions**
- **Low cost, double-wide iSBX™ BITBUS™ expansion MULTIMODULE™ board**
- **Power up diagnostics increase reliablity**

The iSBX™ 344 BITBUS™ Controller MULTIMODULE™ board is the BITBUS gateway to all Intel products that support the iSBX I/O Expansion Interface. Based on the highly integrated 8044 component (an 8 bit 8051 microcontroller and an SDLC controller on one chip) the iSBX 344 MULTIMODULE board extends the capability of other microprocessors via the BITBUS interconnect. With the other members of Intel's Distributed Control Modules (iDCM) family, the iSBX 344 MULTIMODULE board expands Intel's OEM microcomputer system capabilities to include distributed real-time control. Like all members of the iDCM family, the iSBX 344 MULTIMODULE board includes many features that make it well suited for industrial control applications such as: data acquisition and monitoring, process control, robotics, and machine control.



OPERATING ENVIRONMENT

Intel's Distributed Control Modules (iDCM) product family contains the building blocks to implement real-time distributed control applications. The iDCM family incorporates the BITBUS interconnect to provide standard high speed serial communication between microcontrollers. The two iDCM hardware products, the iSBX 344 MULTIMODULE board and the iRCB 44/10 BITBUS Remote Controller Board, communicate in an iDCM system via the BITBUS interconnect as shown in Figure 1.

As a member of the iDCM product line the iSBX 344 MULTIMODULE board fully supports the BITBUS microcontroller interconnect. Typically, the iSBX 344 MULTIMODULE board would be part of a node (master or slave) on the BITBUS interconnect in an iDCM system. As shown in Figure 2 the iSBX 344 MULTIMODULE board plugs into any iSBC® board with an iSBX connector.

The iSBX 344 MULTIMODULE board is the hardware interface between Intel's MULTIBUS® and iPDS™ ISIS environment and the BITBUS environment. With this interface the user can harness the capabilities of other Intel microprocessors eg: 80286, 80186, 8086 in a BITBUS/iDCM system or extend an existing MULTIBUS or iPDS ISIS-based system with the iDCM family.

MULTIBUS® and iPDS™ I/O Expansion

Typically, MULTIBUS iSBC boards have a maximum of two iSBX I/O expansion connectors. These connectors facilitate addition of one or two iSBX I/O MULTIMODULE boards with varying numbers of I/O lines. The iSBX 344 MULTIMODULE board increases the number of I/O lines that can be accommodated by a MULTIBUS system by at least an order of magnitude. The iSBX 344 MULTIMODULE board extends the I/O of Intel's Personal Development System (iPDS) or other systems products in a similar manner.

Extending BITBUS™/iDCM System Processing Capability

The iSBX 344 MULTIMODULE board allows utilization of other processors in a BITBUS/iDCM system to accomodate particular application requirements. The MULTIMODULE board is compatible with any iSBX connector so that any board having a compatible connector can potentially enhance system performance. Intel's iRMX 510 iDCM Support Package provides the software interface required for a variety of iSBC boards. The iSBC 186/03, 86/30, 286/10, and 188/48 boards are a few examples. Custom configurations are also possible with user customized software.

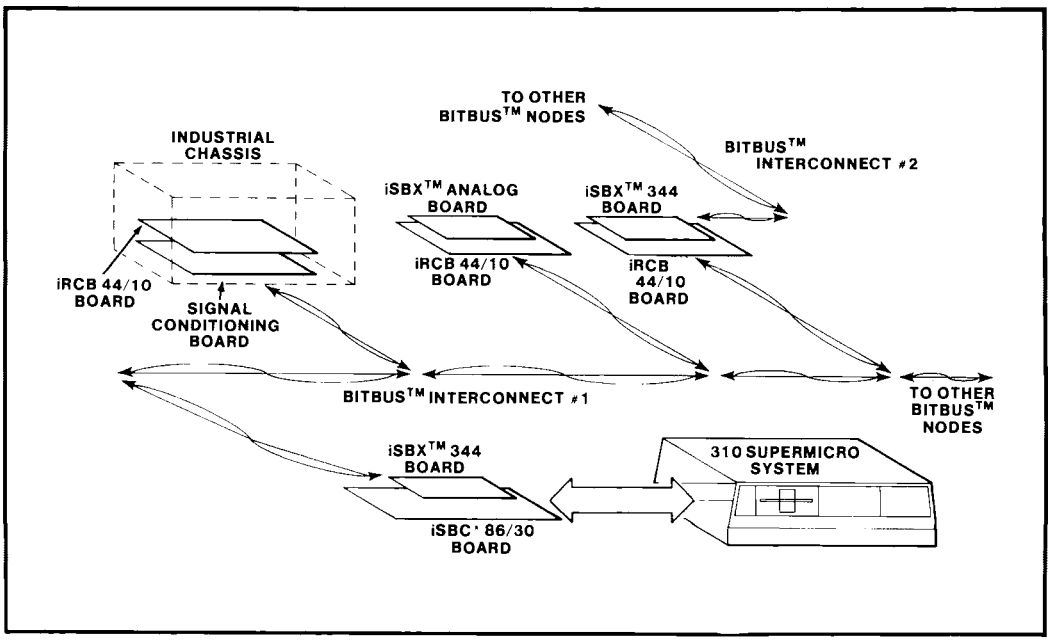


Figure 1. iDCM Operating Environment

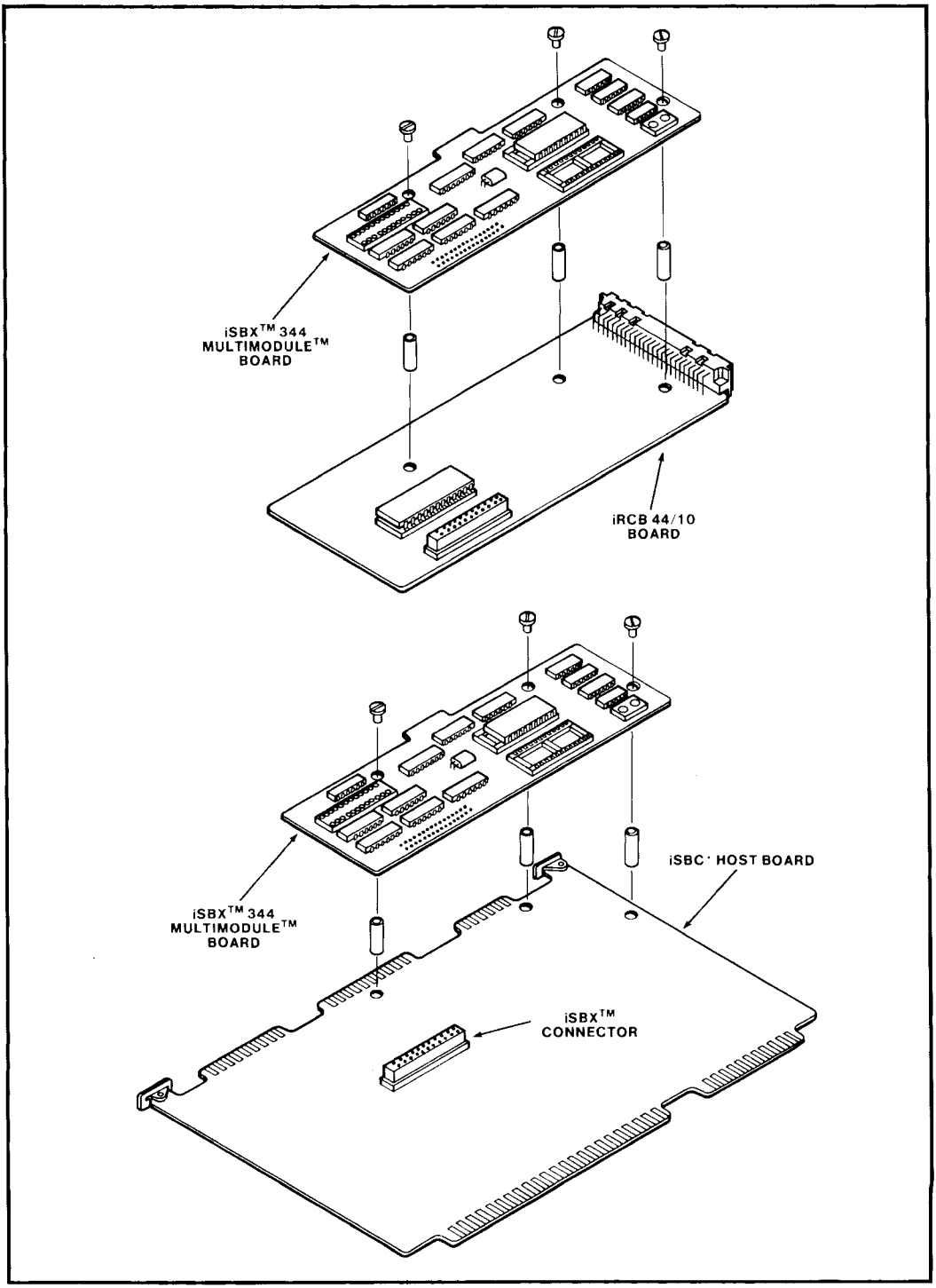


Figure 2. iSBX™ 344 Installation

ARCHITECTURE

Figure 3 illustrates the major functional blocks of the iSBX 344 MULTIMODULE board : iDCM controller, memory, BITBUS microcontroller interconnect, Byte FIFO interface, initialization and diagnostic logic.

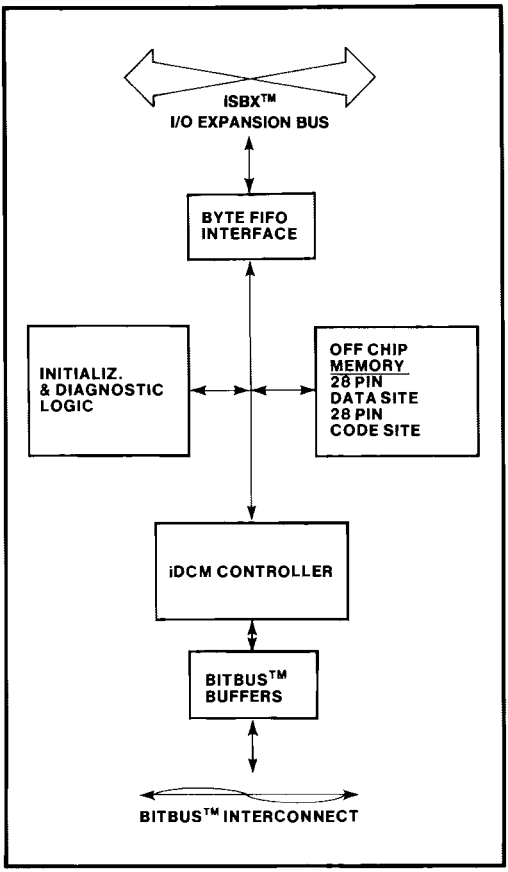


Figure 3. iSBX™ 344 Block Diagram

iDCM Controller

The heart of the iSBX 344 MULTIMODULE board's controlling and communication capability is the highly integrated 12 MHz 8044 microcontroller. The 8044 consists of the advanced 8-bit, 8051 microcontroller and a SDLC controller called the Serial Interface Unit (SIU). This dual processor architecture allows complex control and high speed communication to be realized cost effectively.

Another essential part of the iDCM controller is the integral firmware that resides on-chip to implement the BITBUS interface. In the operating environment of the iSBX 344 MULTIMODULE board, the 8044's SIU acts as a SDLC controller which offloads the on-chip 8051 microcontroller of communication tasks; freeing the 8051 to concentrate on real-time control.

The iDCM controller (8044 microcontroller and on-chip firmware) provides, in one package, a simple user interface, and high performance communications and control capabilities to efficiently and economically build a complex control system.

Memory

The iSBX 344 MULTIMODULE board memory consists of two sections: internal and external. Internal memory is located in the on-chip memory of the iDCM controller. The iRMX 51 Executive and the remaining iDCM controller firmware ration this resource. However, eight bytes of bit addressable internal memory are reserved for the user. Ample space is reserved for user programs and data in the iSBX 344 MULTIMODULE board external memory.

Two 28 pin JEDEC sites comprise the iSBX 344 MULTIMODULE board external memory. One site has been dedicated for data; the other for code. Table 1 lists the supported memory devices for each site. Intel's 2764, 27128, and 2817A are a few examples. The user may choose one of two memory configurations and specify different memory sizes by placing the proper jumpers at system initialization. The most flexible configuration option provides the user with access to the code site for program download or upload. This feature ensures expansion of an existing system is easily accomodated. For example, the addition of another conveyor to a material handling system would require adding another controller or controllers and changes to existing applications code and addition of new code.

Table 1. Supported Memory Devices

DEVICE	DATA SITE	CODE SITE
4K x 8-64K x 8 EPROM/ROM	NO	YES
2K x 8-32K x 8 SRAM	YES	YES
2K x 8-16K x 8 NVRAM and E2PROM	NO	YES

BITBUS™ Microcontroller Interconnect

The iSBX 344 MULTIMODULE board fully supports the BITBUS microcontroller interconnect. The BITBUS interconnect is a serial bus optimized for control applications. The interconnect supports both synchronous and self-clocked modes of operation. These modes of operation are selectable dependent on application requirements as are the transmission rates. Table 2 shows different combinations of modes of operations, transmission rates, and distances. The SDLC protocol, BITBUS message format, and compatibility with Intel's other software and hardware products comprise the remainder of this established architecture. These features contribute to BITBUS reliability and usefulness as a microcontroller interconnect.

The BITBUS connection consists of one or two differential pair(s) of wires. The BITBUS interface of the iSBX 344 MULTIMODULE board consists of a half-duplex RS 485 tranceiver and an optional clock source for the synchronous mode of operation.

Byte FIFO Interface

The Byte FIFO Interface on the iSBX 344 MULTIMODULE board implements the required hardware buffering between the iDCM controller and an extension. An extension is defined as a device attached to the iSBX I/O expansion interface on the iSBX 344 MULTIMODULE board. In an iDCM system, an example of an extension is an iSBC 86/30 board which may be considered the host board in a MULTIBUS system. When used with the software handlers in the iRMX 510 iDCM Support Package, implementation of this interface is complete.

For particular applications, the user may wish to develop a custom software interface to the extension or host board. On the iSBX 344 MULTIMODULE board side of the interface the iDCM firmware automatically accepts messages for the FIFO. No user code is required increasing the time available for application system development.

The Byte FIFO supports both byte and message transfer protocol in hardware via three register ports: data, command, and status. The extension side supports polled, interrupt, and limited DMA modes of operation (e.g. 80186 type DMA controllers).

Initalization and Diagnostic Logic

Like the other members of Intel's Distributed Control Modules (iDCM) product line, the iSBX 344 MULTIMODULE board includes many features which make it well suited for industrial control applications. Power up diagnostics is just one of these features. Diagnostics simplify system startup considerably, by immediately indicating an iDCM controller or external bus failure. The LEDs used for power up diagnostics are available for user diagnostics after power up as well as to further contribute to reliable operation of the system.

Initial iSBX 344 MULTIMODULE board parameters are set by positioning jumpers. The jumpers determine the BITBUS mode of operation: synchronous, self-clocked, transmission rate, and address of the iSBX module in the BITBUS system. This minimizes the number of spare boards to be stocked for multiple nodes, decreasing stocking inventory and cost.

INTEGRAL FIRMWARE

The iSBX BITBUS Controller MULTIMODULE board contains resident firmware located in the iDCM Controller. The on-chip firmware consists of: a pre-configured iRMX 51 Executive for user program development; a Remote Access and Control (RAC) function that enables user communication and control of different microcontrollers and I/O points; a communications gateway to

Table 2. BITBUS™ Microcontroller Interconnect Modes of Operation

	Speed Kb/s	Maximum Distance Between Repeaters M/ft	Maximum # Nodes Between Repeaters	Maximum # Repeaters
Synchronous	2400	30/100	28	0
Self Clocked	375	300/1000	28	2
	62.5	1200/4000	28	10

connect the BITBUS interconnect, iSBX bus, and iRMX 51 Executive tasks; and power up diagnostics.

The iRMX 51 Executive is an event-driven software manager that can respond to the needs of multiple tasks. This real-time multitasking executive provides: task management, timing, interrupt handling, and message passing services. Table 3 shows the iRMX 51 user interfaces. Both the executive and the communications gateway allow for the addition of up to seven user tasks at each node while making BITBUS operations transparent.

The Remote Access and Control Function is a special purpose task that allows the user to transfer commands and program variables to remote BITBUS controllers, obtain the status of a remote I/O line(s), or reverse the state of a remote I/O line. Table 4 provides a complete listing of the RAC services. No user code need be written to use this function. Power up tests provide a quick diagnostic service.

The services provided by the iSBX 344 MULTIMODULE board integral firmware simplify the development and implementation of complex real-time control application systems. All iDCM hardware products contain integral firmware thus supplying the user with a total system solution .

DEVELOPMENT ENVIRONMENT

Intel provides a complete development environment for the iSBX 344 MULTIMODULE board. Software development support consists of : the 8051 Software Development Package, and the iRMX 510 iDCM Support Package. The 8051 Software Development Package provides the RL 51 Linker and Relocator Program, and ASM 51. PL/M 51 is also available. The iRMX 510 Support Package includes the iDCM Controller firmware files on diskette as well as iRMX 51 libraries. Hardware tools consist of of the IN-Circuit Emulator (ICE-44), Intel's Portable Development System (iPDS), and Intellec Series II or III Development Systems.

Table 3. iRMX™ 51 Interfaces

COMMAND	DESCRIPTION
RQ SEND MESSAGE	Sends a message (a command from the BITBUS master, a response from a slave, or a simple message between tasks on the same BITBUS component) to another task.
RQ WAIT	Waits for an interrupt, and event time-out, a message, or any combination of the three.
RQ CREATE TASK	Causes a new sequence of code to be run as an iRMX 51 task with a specific function identification code and priority.
RQ DELETE TASK	Stops the specified task and removes it from all execution lists.
RQ ALLOCATE	Allocates a fixed-length buffer from the on-chip, scratch-pad RAM for general use, or, in BITBUS applications, for a BITBUS message buffer.
RQ DEALLOCATE	Returns an on-chip buffer to the system.
RQ SET INTERVAL	Set the time interval to be used as a separate event-timer for the task.
RQ ENABLE INTERRUPT	Allow external interrupts to signal the microcontroller.
RQ DISSABLE INTERRUPT	Stops all external interrupts from signaling the microcontroller.
RQ GET FUNCTION ID	Provides a list of the 8 function identification codes representing the tasks currently operating on the microcontroller.

Table 4. RAC Services

COMMAND	DESCRIPTION
READ I/O	Read external I/O location. Return result in reply message.
WRITE I/O	Write byte to external I/O location.
UPDATE I/O	Write byte to, then read byte from external I/O location. Return result in reply message.
OR I/O	OR data with contents of external I/O location. Return OR'd value.
AND I/O	AND data with contents of external I/O location. Return AND'd value.
XOR I/O	XOR data with contents of external I/O location. Return XOR'd value.
READ INTERNAL MEMORY	Read contents of internal memory location. Return result in reply message.
WRITE INTERNAL MEMORY	Write data to internal memory location.
DOWNLOAD EXTERNAL MEMORY	Write data starting at external memory location.
UPLOAD EXTERNAL MEMORY	Read data starting at external memory location. Return result in reply message.
GET FUNCTIONS	Provides a list of the 8 function identification codes representing the tasks currently operating on the microcontroller.
CREATE TASK	Causes a new sequence of code to be run as in the iRMX™ 51 interface.
DELETE TASK	Stops the specified task and removes it from all execution lists as in the iRMX™ 51 interface.
RAC PROTECT	Suspends or resumes RAC Services.
RESET DEVICE	Returns device software to original state at initialization.

NOTES:
Internal memory locations are included in the 192 bytes of data RAM provided in the microcontroller. External memory refers memory outside the microcontroller — the 28-pin sockets of the iSBX 344 module and the iRCB 44/10 board. Each RAC Access Function may refer to 1, 2, 3, 4, 5, or 6 individual I/O or memory locations in a single command.

SPECIFICATIONS

Word Size

Instruction — 8 bit
Data — 8 bit

Processor Clock 12 MHz

Instruction Execution Times

1 μsec 60% instructions
2 μsec 40% instructions
4 μsec Multiply & Divide

Memory Capacity/Addressing

iDCM Controller — 64 K

Address Range

	Option 1	Option 2
External Memory		
Data	0000H-7FFFH	0000H-7FFFH
Code	1000H-0FFFFH	8000H-0FFEFH
Internal Memory		
Code	0000H-0FFFH	0000H-0FFFH

Terminations

Sockets provided on board for ¼ Watt 5% Carbon type resistors. Resistor value to match characteristic impedance of cable as closely as possible — 120 ohms or greater.

iDCM Controller (8044 + firmware) I/O Addressing as viewed from the 8044

FUNCTION	ADDRESS	READ	WRITE	BIT	COMMENTS
Data	FF00H	✓	✓		
Command	FF01H	✓	✓		Write sets command to extension — Read clears command from extension
Status					
-RFNF*	B3H	✓		✓	Also INT1 Input
-TFNE*	B2H	✓		✓	Also INT0 Input
-TCMD*	92H	✓		✓	
LED #1	90H	✓	✓	✓	
LED #2	91H	✓	✓	✓	
RDY/NE*	B4H	✓	✓	✓	
Node Address	FFFFH	✓			
Configuration	FFFEH	✓			

iSBX™ 344 MULTIMODULE™ board I/O Addressing as viewed from the iSBX™ 344 MULTIMODULE™ board

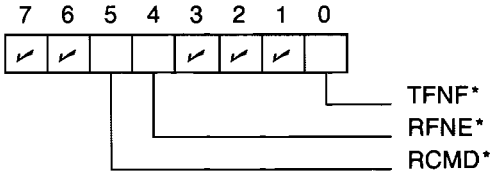
REGISTER FUNCTION	ADDRESS	COMMENTS
Data	Base	Read/Write
Command	Base + 1	Write sets command from extension Read clears command to extension
Status	Base + 2	Read Only

NOTES:
1. Base is determined by MCSO* on extension device

Interrupt/DMA Lines

SIGNAL	LOCATION	INTERFACE OPTION
RINT	MDRQ/MINT0	INT
TINT	MINT1	INT
RCMI	OPT0	INT OR DMA
RDRQ	MDRQ/MINT0	DMA
TDRQ	MINT1	DMA

Status Register Interface



Connector Options

10 Pin Plug

Flat Cable — 3M 3473-6010, TB Ansley 609-1001M, or equal

Discrete Wire — BERG 65846-007, ITT Cannon 121-7326-105, or equal

Pin Out

PIN	SIGNAL
1	+12V
2	+12V
3	GND
4	GND
5	DATA*
6	DATA
7	DCLK*/RTS*
8	DCLK/RTS
9	RGND
10	RGND

Electrical Characteristics

Interfaces

iSBX™ I/O expansion bus — supports the standard I/O Expansion Bus Specification with compliance level D8

Memory Sites — Both code and data sites support the electrical Universal Memory Site specification

BITBUS™ Interconnect — The iSBX 344 MULTIMODULE board supports the BITBUS Specification as follows:

Fully supported synchronous mode at 2.4 Mbits/sec and self clocked mode for 375 kbits/sec and 62.5 kbits/sec

The iSBX 344 MULTIMODULE board presents one standard load to the BITBUS bus

Message length of 18 bytes supported

RAC Function support as shown in Table 4

Power Requirements

.9A at +5V ± 5% iSBX™ 344 MULTIMODULE™ board only — memory NOT included

Physical Characteristics

Double-wide iSBX™ MULTIMODULE™ Form Factor

Dimensions

Height — 10.16 mm (0.4 in) maximum component height

Width — 63.5 mm (2.50 in)

Depth — 190.5 mm (7.50 in)

Weight — 113 gm (4 ounces)

Environmental Characteristics

Operating Temperature — 0°C to 55°C at 200 Linear Feet/Minute Air Velocity

Humidity — 90% non-condensing

Reference Manual (NOT Supplied)

146312 — Guide to Using the Distributed Control Modules

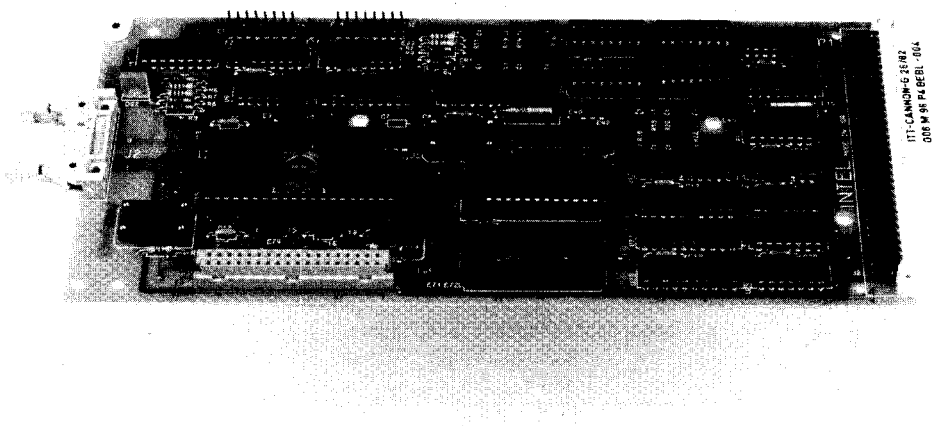
Ordering Information

Part Number	Description
iSBX 344	BITBUS Controller MULTIMODULE board

iRCB 44/10
BITBUS™ Remote Controller Board

- **High performance 12 MHz 8044 controller**
- **Integral firmware including the iRMX™ 51 Executive optomized for real-time control**
- **Full BITBUS™ support**
- **Standard industrial packaging: Eurocard, DIN connector**
- **2 28-pin JEDEC memory sites for user's control functions**
- **Low cost I/O expansion with 8-bit iSBX™ connector**
- **Programmable control/monitoring capability of 24 I/O lines**
- **Power up diagnostics increase reliability**

The iRCB 44/10 BITBUS™ Remote Controller Board, is an intelligent real-time controller and a remote I/O expansion device. Based on the highly integrated 8044 component (an 8 bit 8051 microcontroller and an intelligent SDLC controller on one chip) the iRCB 44/10 board provides high performance control capability at low cost. Incorporating complete BITBUS support, the iRCB 44/10 board and the other members of Intel's Distributed Control Modules (iDCM) family expand Intel's OEM microcomputer system capabilities to include distributed real-time control. Like all members of the iDCM family, the iRCB 44/10 board includes many features that make it well suited for industrial control applications such as: data acquisition and monitoring, process control, robotics, and machine control.



OPERATING ENVIRONMENT

Intel's Distributed Control Modules (iDCM) product family contains the building blocks to implement real-time distributed control applications. The iDCM family incorporates the BITBUS interconnect to provide standard high speed serial communication between microcontrollers. The two iDCM hardware products, the iSBX 344 BITBUS controller MULTIMODULE™ board and the iRCB 44/10 BITBUS Remote Controller Board, communicate in an iDCM system via the BITBUS interconnect as shown in Figure 1.

The iRCB 44/10 board can be used as an intelligent remote controller or an I/O expansion device. When performing as an intelligent controller the iRCB 44/10 board not only monitors the status of multiple process points, but it can execute varied user supplied control algorithms. When functioning as an I/O expansion device, the iRCB 44/10 board simply collects data from multiple I/O ports and transmits this information via the BITBUS or iSBX bus interface to the system controller for analysis or updating purposes.

As a member of the iDCM product line the iRCB 44/10 board fully supports the BITBUS microcontroller interconnect. Typically, the iRCB 44/10 board would be a node in a BITBUS system. The iRCB 44/10 board could be part of a master or slave node. (The BITBUS system supports a multidrop configuration: one master, many slaves.)

ARCHITECTURE

Figure 2 illustrates the major functional blocks of the iRCB 44/10 board: iDCM controller, memory, BITBUS microcontroller interconnect, parallel I/O, iSBX expansion, initialization and diagnostic logic.

iDCM Controller

The heart of the iRCB 44/10 board's controlling and communication capability is the highly integrated 12 MHz 8044 microcontroller. The 8044 consists of the advanced 8-bit 8051 microcontroller and a SDLC controller called the Serial Interface Unit (SIU). This dual processor architecture allows complex control and high speed communication functions to be realized cost effectively.

Another essential part of the iDCM controller is the integral firmware that resides on-chip to implement the BITBUS interface. In the operating environment of the iRCB 44/10 board, the 8044's SIU acts as a SDLC controller which offloads the on-chip 8051 microcontroller of communication tasks; freeing the 8051 to concentrate on real-time control.

The iDCM controller (8044 microcontroller and on-chip firmware) provides, in one package, a simple user interface, and high performance communications and control capabilities to efficiently and economically build a complex control system.

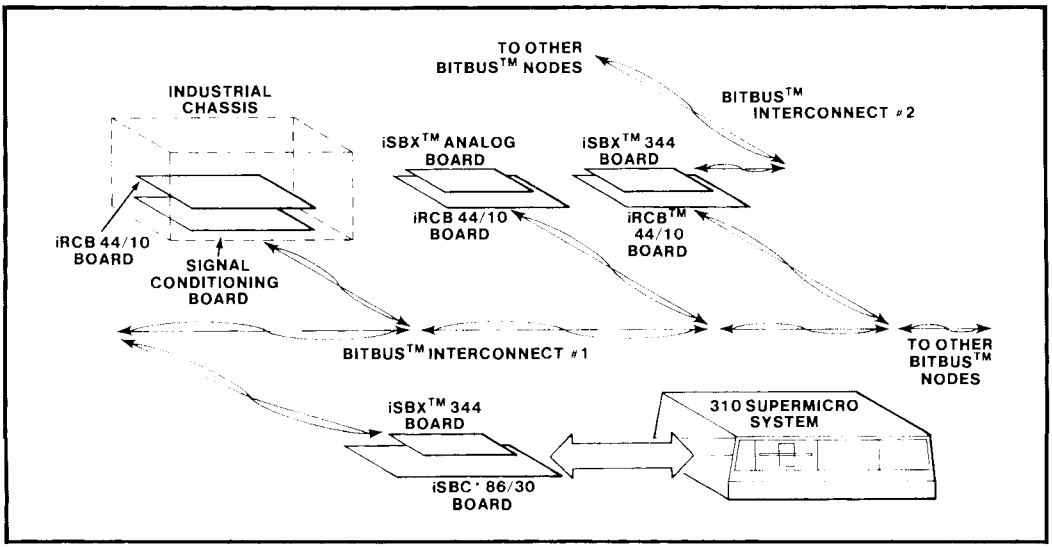


Figure 1. iDCM Operating Environment

Memory

The iRCB 44/10 board memory consists of two sections: internal and external. Internal memory is located in the on-chip memory of the iDCM controller. The iRMX 51 Executive and the remaining iDCM controller firmware ration this resource. However, eight bytes of bit addressable internal memory are reserved for the user. Ample space is reserved for user programs and data in the iRCB 44/10 board external memory.

Two 28 pin JEDEC sites comprise the iRCB 44/10 board external memory. One site has been dedicated for data, the other for code. Table 1 lists the supported memory devices for each site. Intel's 2764, 27128, and 2817A are a few examples. The user may choose one of two memory configurations and specify different memory sizes by placing the proper jumpers at system initialization. The most flexible configuration option provides the user with access to the code site for program download or upload. This feature ensures expansion of an existing system is easily accomodated.

Table 1. Supported Memory Devices

DEVICE	DATA SITE	CODE SITE
4K x 8-64K x 8 EPROM/ROM	NO	YES
2K x 8-32K x 8 SRAM	YES	YES
2K x 8-16K x 8 NVRAM and E2PROM	NO	YES

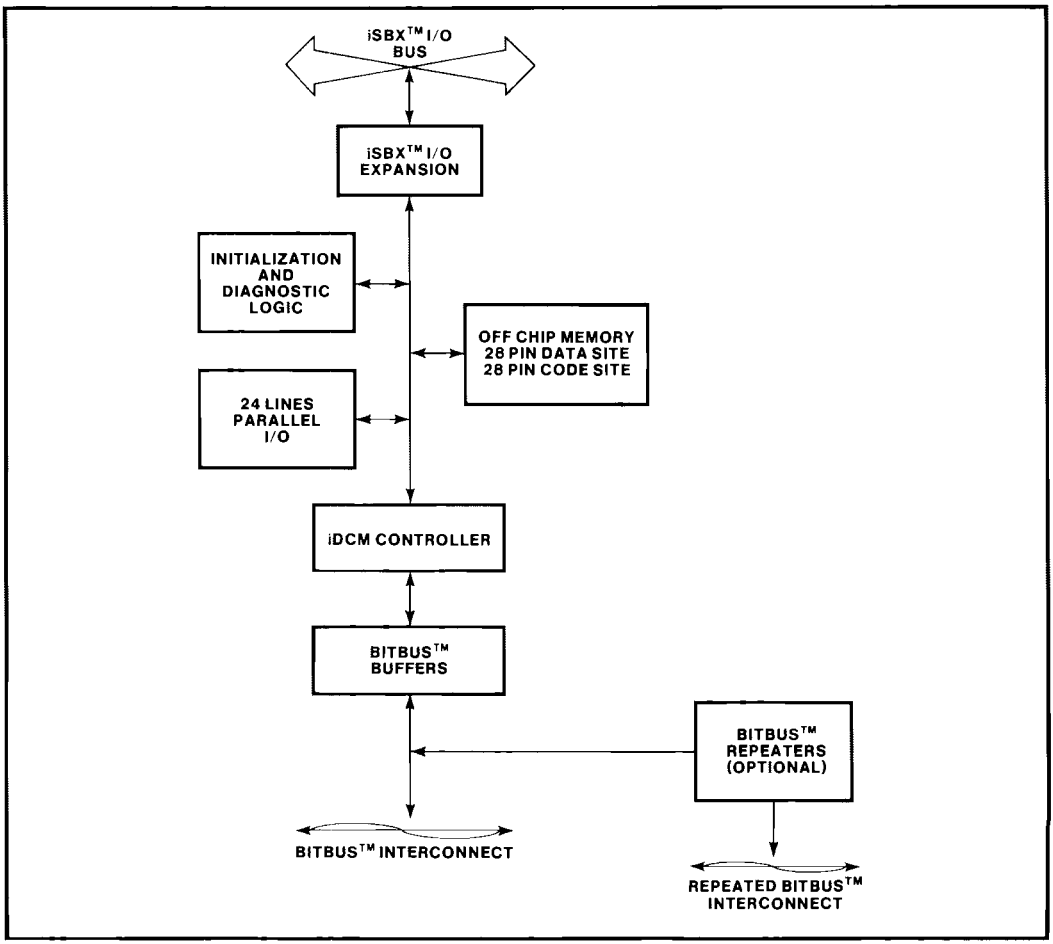


Figure 2. iRCB™ 44/10 Block Diagram

BITBUS™ Microcontroller Interconnect

The iRCB 44/10 board serial interface fully supports the BITBUS microcontroller interconnect. The BITBUS interconnect is a serial bus optimized for control applications. The bus supports both synchronous and self-clocked modes of operation. These modes of operation are selectable dependent on application requirements as are the transmission speeds. Table 2 shows the different combinations of modes of operation, transmission speeds, and distances. The SDLC protocol, BITBUS message format, and compatibility with Intel's other software and hardware products comprise the remainder of the BITBUS architecture. These features contribute to BITBUS system reliability and usefulness as a microcontroller interconnect.

The BITBUS connection consists of one or two differential pair(s) of wires. The serial (BITBUS) interface of the iRCB 44/10 board consists of : a half-duplex RS 485 tranceiver, an optional BITBUS repeater and an optional clock source for the synchronous mode of operation.

Parallel I/O

In order to provide an optimal parallel I/O interface for control applications, the iRCB 44/10 board supports 24 software programmable parallel I/O lines. This feature supplies the flexibility and simplicity required for control and data acquisition systems. Sixteen of these lines are fully programmable as inputs or outputs, with loopback, on a bit by bit basis so that bit set, reset, and toggle operations are streamlined. The remaining eight lines are dedicated as inputs. Figure 3 depicts the general I/O port structure.

The parallel I/O lines can be manipulated by using the Remote Access and Control (RAC) function (in iDCM controller firmware) from a supervisory node or locally by a user program. The user program can also access the RAC function or directly operate the I/O lines. Input, output, mixed — input and output, and bit operations are possible simply by reading or writing a particular port.

iSBX™ Expansion

One iSBX I/O expansion connector is provided on the iRCB 44/10 board. This connector can be used to extend the I/O capability of the board. In addition to specialized and custom designed iSBX boards, a full line of compatible high speed, 8-bit expansion MULTIMODULE boards, both single and double wide, are available from Intel. The only incompatible modules are those that require the MWAIT* signal or DMA operation. A few of Intel's iRCB 44/10 board compatible iSBX MULTIMODULE boards include: parallel I/O, serial I/O, BITBUS expansion, IEEE 488 GPIB, analog input, analog output, and magnetic bubble.

With the iSBX 344 BITBUS Controller MULTIMODULE board and user supplied software, the iRCB 44/10 board can act as an intelligent BITBUS repeater facilitating the transition between two BITBUS segments operating at different speeds.

Initalization and Diagnostic Logic

Like the other members of Intel's Distributed Control Modules (iDCM) product line, the iRCB 44/10 board includes many features which make it well suited for industrial control applications. Power up diagnostics is just one of these features. Diagnostics simplify system startup considerably, by immediately indicating an iDCM controller or external bus failure. The LEDs used for power up diagnostics are available for user diagnostics after power up as well to further contribute to reliable operation of the system.

Initial iRCB 44/10 board parameters are set by positioning jumpers. The jumpers determine the

Table 2. Modes of Operation

	Speed Kb/s	Maximum Distance Between Repeaters M/ft	Maximum # Nodes Between Repeaters	Maximum # Repeaters
Synchronous	2400	30/100	28	0
Self Clocked	375	300/1000	28	2
	62.5	1200/4000	28	10

BITBUS mode of operation: synchronous, self clocked, transmission speed, and address of the iRCB 44/10 board in the BITBUS system. This minimizes the number of spare boards to be stocked for multiple nodes, decreasing stocking inventory and cost.

INTEGRAL FIRMWARE

The iRCB 44/10 board contains resident firmware located in the iDCM Controller. The on-chip firmware consists of: a pre-configured iRMX 51 Executive for user program development; a Remote Access and Control (RAC) function that enables user communication and control of different microcontrollers and I/O points; a communications gateway to connect the BITBUS interconnect, iSBX bus, and iRMX 51 tasks; and power up diagnostics.

The iRMX 51 Executive is an event-driven software manager that can respond to the needs of multiple tasks. This real-time multitasking executive provides: task management, timing, interrupt handling, and message passing services. Table 3 shows the iRMX 51 user interfaces. Both the Executive and the communications gateway allow for the addition of up to seven user tasks at each node while making BITBUS operations transparent.

The Remote Access and Control Function is a special purpose task that allows the user to trans-

Table 3. iRMX™ 51 Interfaces

COMMAND	DESCRIPTION
RQ SEND MESSAGE	Sends a message (a command from the BITBUS master, a response from a slave, or a simple message between tasks on the same BITBUS component) to another task.
RQ WAIT	Waits for an interrupt, and event time-out, a message, or any combination of the three.
RQ CREATE TASK	Causes a new sequence of code to be run as an iRMX 51 task with a specific function identification code and priority.
RQ DELETE TASK	Stops the specified task and removes it from all execution lists.
RQ ALLOCATE	Allocates a fixed-length buffer from the on-chip, scratch-pad RAM for general use, or, in BITBUS applications, for a BITBUS message buffer.
RQ DEALLOCATE	Returns an on-chip buffer to the system.
RQ SET INTERVAL	Set the time interval to be used as a separate event-timer for the task.
RQ ENABLE INTERRUPT	Allow external interrupts to signal the microcontroller.
RQ DISSABLE INTERRUPT	Stops all external interrupts from signalling the microcontroller.
RQ GET FUNCTION ID	Provides a list of the 8 function identification codes representing the tasks currently operating on the microcontroller.

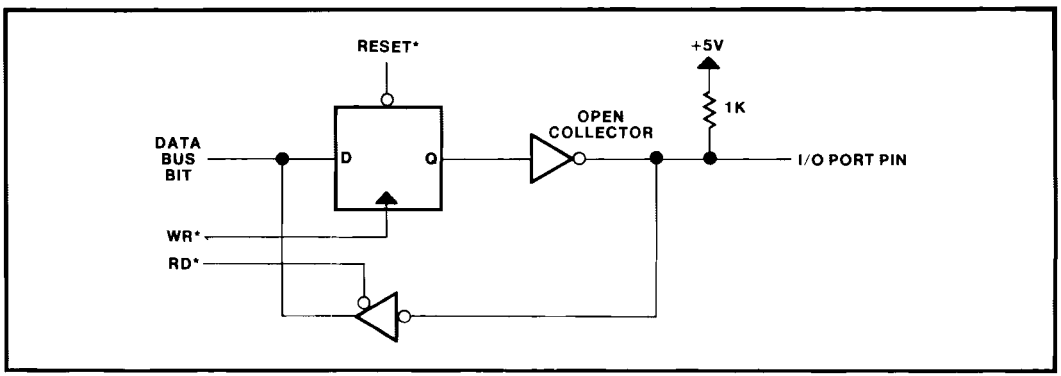


Figure 3. I/O Port Structure

fer commands and program variables to remote BITBUS controllers, obtain the status of a remote I/O line(s), or reverse the state of a remote I/O line. Table 4 provides a complete listing of the RAC services. No user code need be written to use this function. Power up tests provide a quick diagnostic service.

The services provided by the iRCB 44/10 board integral firmware simplify the development and implementation of complex real-time control application systems. All iDCM hardware products contain integral firmware thus supplying the user with a total system solution.

INDUSTRIAL PACKAGING

The iRCB 44/10 form factor is a single high, 220mm deep Eurocard as shown in Figure 4. The Eurocard form factor supports most standard industrial packaging schemes as well as Intel's MULTIBUS® II packaging scheme. The Eurocard form factor specifies reliable DIN connectors. A standard 64 pin connector is included on the iRCB 44/10 board.

DEVELOPMENT ENVIRONMENT

Intel provides a complete development environment for the iRCB 44/10 board. Software development support consists of: the 8051 Software Development Package, and the iRMX 510 iDCM Support Package. The 8051 Software Development Package provides the RL 51 Linker and Relocator Program, and ASM 51. PL/M 51 is also available. The iRMX 510 Support Package includes the iDCM Controller firmware files on diskette, as well as iRMX 51 libraries. Hardware tools consist of of the In-Circuit Emulator (ICE-44), Intel's Portable Development System (iPDS), and Intellec Series II or III Development Systems.

Table 4. RAC Services

COMMAND	DESCRIPTION
READ I/O	Read external I/O location. Return result in reply message.
WRITE I/O	Write byte to external I/O location.
UPDATE I/O	Write byte to, then read byte from external I/O location. Return result in reply message.
OR I/O	OR data with contents of external I/O location. Return OR'd value.
AND I/O	AND data with contents of external I/O location. Return AND'd value.
XOR I/O	XOR data with contents of external I/O location. Return XOR'd value.
READ INTERNAL MEMORY	Read contents of internal memory location. Return result in reply message.
WRITE INTERNAL MEMORY	Write data to internal memory location.
DOWNLOAD EXTERNAL MEMORY	Write data starting at external memory location.
UPLOAD EXTERNAL MEMORY	Read data starting at external memory location. Return result in reply message.
GET FUNCTIONS	Provides a list of the 8 function identification codes representing the tasks currently operating on the microcontroller.
CREATE TASK	Causes a new sequence of code to be run as in the iRMX 51 interface.
DELETE TASK	Stops the specified task and removes it from all execution lists as in the iRMX 51 interface.
RAC PROTECT	Suspends or resumes RAC Services.
RESET DEVICE	Returns device software to original state at initialization.

NOTES:
Internal memory locations are included in the 192 bytes of data RAM provided in the microcontroller. External memory refers memory outside the microcontroller — the 28-pin sockets of the iSBX 344 module and the iRCB 44/10 board. Each RAC Access Function may refer to 1, 2, 3, 4, 5, or 6 individual I/O or memory locations in a single command.

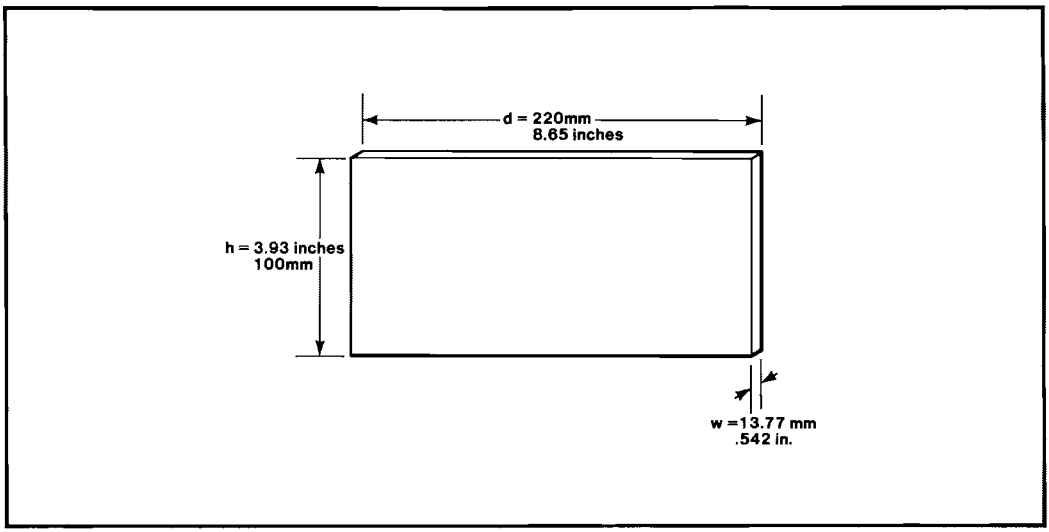


Figure 4. Eurocard Single High Form Factor

SPECIFICATIONS

Word Size

Instruction — 8 bit
Data — 8 bit

Processor Clock 12 MHz

Instruction Execution Times

1 μsec 60% instructions
2 μsec 40% instructions
4 μsec Multiply & Divide

Memory Capacity/Addressing

iDCM Controller — 64 K

Address Range

	Option 1	Option 2
External Memory		
Data	0000H-7FFFH	0000H-7FFFH
Code	1000H-0FFFFH	8000H-0FFEFH
Internal Memory		
Code	0000H-0FFFH	0000H-0FFFH

I/O Capacity

iSBX MULTIMODULE™ board — one single or doublewide not requiring MWAIT* or DMA
24 Digital Lines Programmable Parallel I/O

Interrupt Sources

Two external — iSBX I/O Expansion bus sources or other sources.
BITBUS Microcontroller Interconnect.

Terminations

Sockets provided on board for ¼ Watt 5% Carbon type resistors. Resistor value to match characteristic impedance of cable as closely as possible — 120 ohms or greater.

Repeaters

Sockets provided on board — Devices 75174 and 75175

Connector Options

10 Pin Plug

Flat Cable — 3M 3473-6010, TB Ansley 609-1001M, or equal
Discrete Wire — BERG 65846-007, ITT Cannon 121-7326-105, or equal

DIN Connector Plug

Flat Cable — GW Elco 00-8259-096-84-124, Robinson Nugent RNE-IDC64C-TG30, or equal
Discrete Wire — ITT Cannon G06 M96 P3 BDBL-004 GW Elco 60 8257 3017, or equal

iDCM Controller I/O Addressing

FUNCTION	ADDRESS	READ	WRITE	BIT
PORT A	FFCOH	✓	✓	
PORT B	FFC1H	✓		
PORT C	FFC2H	✓	✓	
MCSO	FF80H-FF87H FF00, FF01	✓	✓	
MSC1	FF88H-FF8F	✓	✓	
LED #1	90H	✓	✓	✓
LED #2	91H	✓	✓	✓
RDY/NE*	B4H	✓	✓	✓
NODE ADDRESS	FFFFH	✓		
CONFIGURATION	FFFEH	✓		
OPT0	92H	✓	✓	✓
OPT1	93H	✓	✓	✓
INT0	B2H	✓		✓
INT1	B3H	✓		✓

10 Pin Repeater Connector Pin Out

PIN	SIGNAL
1	+12V
2	+12V
3	GND
4	GND
5	DATA*
6	DATA
7	DCLK*/RTS*
8	DCLK/RTS
9	RGND
10	RGND

iRCB 44/10 Pin Out

DIN Connector

DIN PIN #	PIN & SOCKET PIN #	FUNCTION	DIN PIN #	PIN & SOCKET PIN #	FUNCTION
1a		GND	1c		GND
2a		+5V	2c		+5V
3a		DATA	3c		DATA*
4a		DLCK/RT	4c		DLCK*/RTS*
5a	1	EXTINT	5c		RGND
6a	3	PB7	6c	2	GND
7a	5	PB6	7c	4	GND
8a	7	PB5	8c	6	GND
9a	9	PB4	9c	8	GND
10a	11	PB3	10c	10	GND
11a	13	PB2	11c	12	GND
12a	15	PB1	12c	14	GND
13a	17	PB0	13c	16	GND
14a	19	PC3	14c	18	GND
15a	21	PC2	15c	20	GND
16a	23	PC1	16c	22	GND
17a	25	PC0	17c	24	GND
18a	27	PC4	18c	26	GND
19a	29	PC5	19c	28	GND
20a	31	PC6	20c	30	GND
21a	33	PC7	21c	32	GND
22a	35	PA7	22c	34	GND
23a	37	PA6	23c	36	GND
24a	39	PA5	24c	38	GND
25a	41	PA4	25c	40	GND
26a	43	PA3	26c	42	GND
27a	45	PA2	27c	44	GND
28a	47	PA1	28c	46	GND
29a	49	PA0	29c	48	GND
30a		+12V	30c		−12V
31a		+5V	31c		+5V
32a		GND	32c		GND

Electrical Characteristics

Interfaces

iSBX I/O expansion bus — supports the standard I/O Expansion Bus Specification with compliance level D8/8F

Memory Sites — Both code and data sites support the electrical Universal Memory Site specification

BITBUS™ Interconnect — The iRCB 44/10 Remote Controller Board supports the BITBUS Specification as follows:
Fully supported synchronous mode at 2.4 Mbits/second and self clocked mode for 375 kbits/second and 62.5 kbits/second.

The iRCB 44/10 Remote Controller Board presents one standard load to the BITBUS bus without repeaters, with repeaters two standard loads

Message length of 18 bytes supported

RAC Function support as shown in Table 4

Parallel I/O — See the Table 5 for Electrical Specifications of the interface.

Power Requirements

.9A at +5V ± 5% iRCB 44/10 board only — memory, repeater, or iSBX board NOT included

Physical Characteristics

Single high, 220mm deep Eurocard Form Factor

Dimensions

Width — 13.77 mm (.542 in) maximum component height

Height — 100 mm (3.93 in)

Depth — 220 mm (8.65 in)

Weight — 169 gm (6 ounces)

Environmental Characteristics

Operating Temperature — 0°C to 55°C at 200 Linear Feet/Minute Air Velocity

Humidity — 90% non-condensing

Reference Manual (NOT Supplied)

146312 — Guide to Using the Distributed Control Modules

Table 5. Parallel I/O Electrical Specification

PARAMETER	CONDITION	MIN	MAX	UNITS
V_{OL}	$I_{OL} = 16\,mA$		0.5	V
V_{OH}	$I_{OH} = -2\,mA$	2.4		V
V_{IH}		2.0	7.0	V
V_{IL}		−1.0	0.8	V
I_{IL}	$V_{IL} = 0.5V$		6.0	mA
I_{IH}	V_{IH} = logic high		.0	mA
I_I	$V_{IH} = 7V$		−2.2	mA

Ordering Information

Part Number	Description
iRCB 44/10	BITBUS Remote Controller board

intel

iRMX™ 51
REAL-TIME MULTITASKING EXECUTIVE

- **Software tool for family of 8051 microcontroller based applications**
- **Real-time, multitasking executive**
- **Supports remote task communication**
- **Small — 2.2K Bytes**
- **Reliable**
- **Simple user interface**
- **Compatible with BITBUS™/Distributed Control Modules (iDCM) product line: iSBX™ 344 & iRCB 44/10 boards**

The iRMX™ 51 Executive is a compact, easy to use, software tool for development and implementation of applications built on the high performance 8-bit family of 8051 microcontrollers. A few members of this expansive family are the 8051, 8044, and 8052 microcontrollers. Like the 8051 family, the iRMX 51 Executive incorporates many features that make it exceptionally well suited for real-time control applications requiring manipulation and scheduling of more than one job, and fast response to external stimuli.

The 8051 microcontroller family is the family of choice for applications such as: data acquisition and monitoring, process control, robotics, and machine control. Using the iRMX 51 Executive for a foundation can significantly reduce applications development time. Also, the iRMX 51 Executive fully supports Intel's BITBUS™ microcontroller interconnect expressly designed for reliable high performance real-time control.

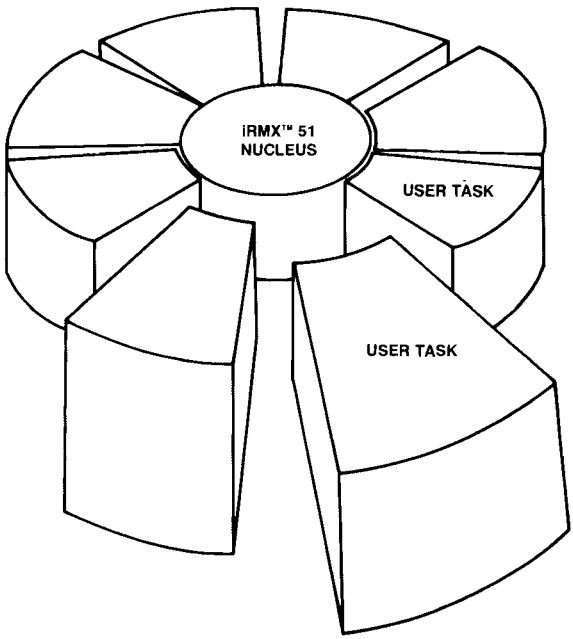


Figure 1. Structure Diagram

ARCHITECTURE

Real-time and Multitasking

Real-time control applications must be responsive to the external environment and typically involve the execution of more than one function (task or set of tasks) in response to different external stimuli. Control of an industrial drying process is an example. This process could require monitoring of multiple temperatures and humidity; control of fans, heaters, and motors that must respond accordingly to a variety of inputs. The iRMX 51 Executive fully supports applications requiring response to stimuli as they occur ie. in real-time. This real-time response is supported for multiple tasks often needed to implement a control application.

Some of the facilities precisely tailored for development and implementation of real-time control application systems provided by the iRMX 51 Executive are: task management, interrupt handling, message passing, and when intergrated with communications support, message passing with different microcontrollers. Also, the iRMX 51 Executive is driven by events: interrupts, timers, and messages ensuring the application system always responds to the environment appropriately.

Task Management

A task is a program defined by the user to execute a particular control function or functions. Multiple programs or tasks may be required to implement a particular function such as 'controlling Heater 1.' The iRMX 51 Executive recognizes three different task states as one of the mechanisms to accomplish scheduling of up to eight tasks. Figure 2 illustrates the different task states and their relationship to one another.

The scheduling of tasks is priority based. The user can prioritize tasks to reflect their relative importance within the overall control scheme. For instance, if Heater 1 must go off line prior to Heater 2 then the task associated with Heater 1 shutdown could be assigned a higher priority ensuring the correct shutdown sequence. The RQ WAIT system call is also a scheduling tool. In this example the task implementing Heater 2 shutdown could include an instruction to wait for completion of the task that implements Heater 1 shutdown.

The iRMX 51 Executive allows for PREEMPTION of a task that is currently being executed. This means that if some external event occurs such as a catastrophic failure of Heater 1, a higher priority task associated with the interrupt, message, or timeout resulting from the failure will preempt the running task. Preemption ensures the emergency will be responded to immediately. This is crucial for real-time control application systems.

Interrupt Handling

The iRMX 51 executive supports sixteen interrupt sources as shown in Table 1. Four of these interrupt sources, excluding timer 0, can be as-

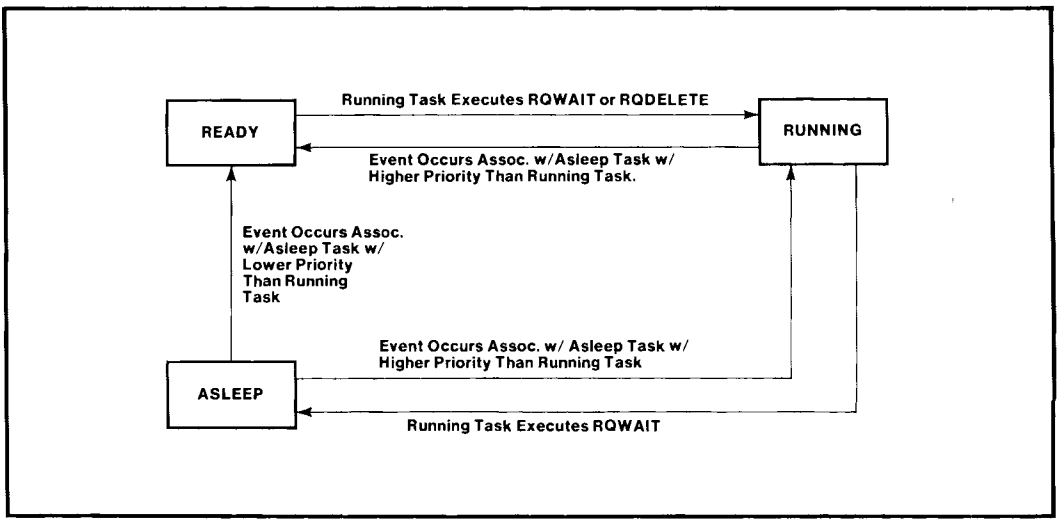


Figure 2. Task State Transition Diagram

signed to a task. When one of the interrupts occurs the task associated with it becomes a running task (if it were the highest priority task in a ready state). In this way, the iRMX 51 Executive responds to a number of internal and external stimuli including time intervals designated by the user.

Table 1. iRMX™ 51 Interrupt Sources

INTERRUPT SOURCE	INTERRUPT NUMBER
External Request 0	00H
Timer 0	01H
External Request 1	02H
Timer 1	03H
Internal Serial Port 1	04H
Reserved	05H
Reserved	06H
Reserved	07H
Reserved	08H
Reserved	09H
Reserved	0AH
Reserved	0BH
Reserved	0CH
Reserved	0DH
Reserved	0EH
Reserved	0FH

Message Passing

The iRMX 51 Executive allows tasks to interface with one another via a simple message passing facility. This message passing facility can be extended to different processors when communications support is integrated within a BITBUS/iDCM system, for example. This facility provides the user with the ability to link different functions or tasks. Linkage between tasks/functions is typically required to support development of complex control applications with multiple sensors (inputs variables) and drivers (output variables). For instance, the industrial drying process might require a dozen temperature inputs, six moisture readings, and control of: three fans, two conveyor motors, a dryer motor, and a pneumatic conveyor. The data gathered from both the temperature and humidity sensors could be processed. Two tasks might be required to gather the data and process it. One task could perform a part of the analysis, then include a pointer to the next task to complete the next part of the analysis . The tasks could continue to move between one another.

REMOTE TASK COMMUNICATION

The iRMX 51 Executive system calls can support communication to tasks on remote controllers. This feature makes the iRMX 51 Executive ideal for applications using distributed architectures. Providing communication support saves significant application development time and allows for more effective use of this time. Intel's iDCM product line combines hardware and software to provide this function.

In an iDCM system, communication between nodes occurs via the BITBUS microcontroller interconnect. The BITBUS microcontroller interconnect is a high performance serial control bus specifically intended for use in applications built on distributed architectures. The iRMX 51 Executive provides BITBUS support.

BITBUS™/iDCM COMPATIBLE

A pre-configured version of the iRMX 51 Executive implements the BITBUS message format and provides all iRMX 51 facilities mentioned previously: task management, interrupt handling, and message passing. This version of the Executive is supplied in firmware on the iDCM Controller with the iDCM hardware products: the iSBX 344 BITBUS Controller MULTIMODULE and the iRCB 44/10 BITBUS Remote Controller boards. It is also supplied on diskette as part of the iRMX 510 iDCM Support Package to ease development of BITBUS systems.

SIMPLE USER INTERFACE

The iRMX 51 Executive's capabilities are utilized through system calls. These interfaces have been defined for ease of use and simplicity. Table 2 includes a listing of these interfaces and their functions. Note tasks may be created at system initialization or run-time using the CREATE TASK call.

Functions such as GET FUNCTION IDS, ALLOCATE/DEALLOCATE BUFFER, and SEND MESSAGE (Messages in the iRMX 51 Executive have a maximum size of 64 bytes.), support communication for distributed architectures. Architectures that define multiple remote stations requiring intelligent and dumb I/O manipulation. The remaining

Table 2. iRMX™ 51 System Interfaces

COMMAND	DESCRIPTION
RQ SEND MESSAGE	Sends a message (a command from the BITBUS master, a response from a slave, or a simple message between tasks on the same BITBUS component) to another task.
RQ WAIT	Waits for an interrupt, an event time-out, a message, or any combination of the three.
RQ CREATE TASK	Causes a new sequence of code to be run as an iRMX 51 task with a specific function identification code and priority.
RQ DELETE TASK	Stops the specified task and removes it from all execution lists.
RQ ALLOCATE	Allocates a fixed-length buffer from the on-chip, scratch-pad RAM for general use, or, in BITBUS applications, for a BITBUS message buffer.
RQ DEALLOCATE	Returns an on-chip buffer to the system.
RQ SET INTERVAL	Set the time interval to be used as a separate event-timer for the task.
RQ ENABLE INTERRUPT	Allow external interrupts to signal the microcontroller.
RQ DISSABLE INTERRUPT	Stops all external interrupts from signaling the microcontroller.
RQ GET FUNCTION ID	Provides a list of the 8 function identification codes representing the tasks currently operating on the microcontroller.

interfaces allow the user to specify the system's response to the external environment — a must for real-time control.

Another feature that eases application development is automatic register bank allocation. The Executive will assign tasks to register banks automatically unless a specific request is made. The iRMX 51 Executive keeps track of the register assignments allowing the user to concentrate on other activities.

The user configures an iRMX 51 system simply by: specifying the initial set of task descriptors and configuration values, and linking the system via the RL 51 Linker and Locator Program with user programs. The nature of the task descriptors allows the user to develop programs, locate them in off-chip ROM, and access them without writing additional code. Programs may be written in ASM 51 or PL/M 51. (Intel's 8051 Software Development Package contains both ASM 51 and RL 51. The iRMX 51 Executive supplies the configuration file and macro defining initial task descriptors.) Figure 3 shows the relationships that exist in the system generation process.

RELIABILE

Real-time control applications require reliability. The nucleus requires about 2K bytes of code space, 40 bytes on-chip RAM, & 218 bytes external RAM. Streamlined code increases performance and reliability, and flexibility is not sacrificed as code may be added to either on-chip or external memory.

The iRMX 51 architecture and simple user interface further enhance reliablity and lower cost. For example, the straightforward structure of the user interfaces, and the transparent nature of the scheduling process contribute to reliability of the overall system by minimizing programming effort. Also, modularity increases reliability of the system and lowers cost by allowing user tasks to be refined independent of the system. In this way, errors are identified earlier and can be easily corrected in each isolated module.

In addition, users can assign tasks a Function ID that allows tracking of the tasks associated with a particular control/monitoring function. This feature reduces maintenance and trouble shooting time thus increasing system run time and decreasing cost.

OPERATING ENVIRONMENT

The iRMX 51 Executive supports applications development based on any member of the high performance 8051 family of microcontrollers. The Executive is available on diskette with user linkable libraries or in the Distributed Control Modules (iDCM) controller preconfigured in on-

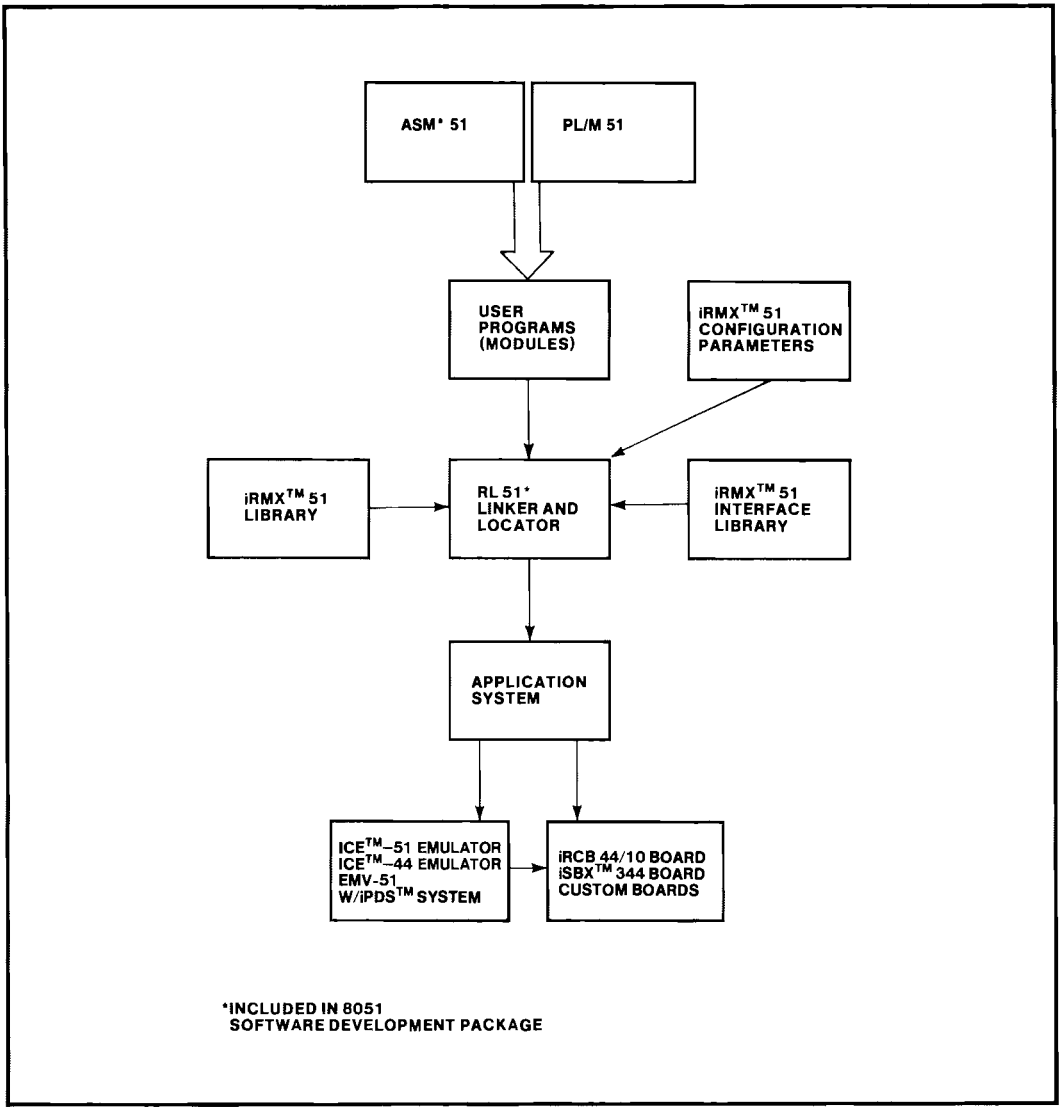


Figure 3. System Generation Process

chip ROM. (The iDCM controller is an 8044 component that consists of an 8051 microcontroller and SDLC controller on one chip with integral firmware.)

When in the iDCM environment (Figure 4), the iRMX 51 Executive can communicate with iRMX based systems like the System 286/310 or ISIS based systems like the Intel Portable Development System (iPDS) by using the iRMX 510 iDCM Support Package.

DEVELOPMENT ENVIRONMENT

Intel provides a complete development environment for the 8051 family of microcontrollers. This environment encompasses iDCM system (BITBUS based) applications also. Software development support consists of: the 8051 *Software Development* Package, and the iRMX 510 iDCM Support Package. Hardware tools consist of a variety of In Circuit Emulators (ICE), Intel's Portable Development System (iPDS) with EMV-51, and Intellec® Series II or III Development Systems.

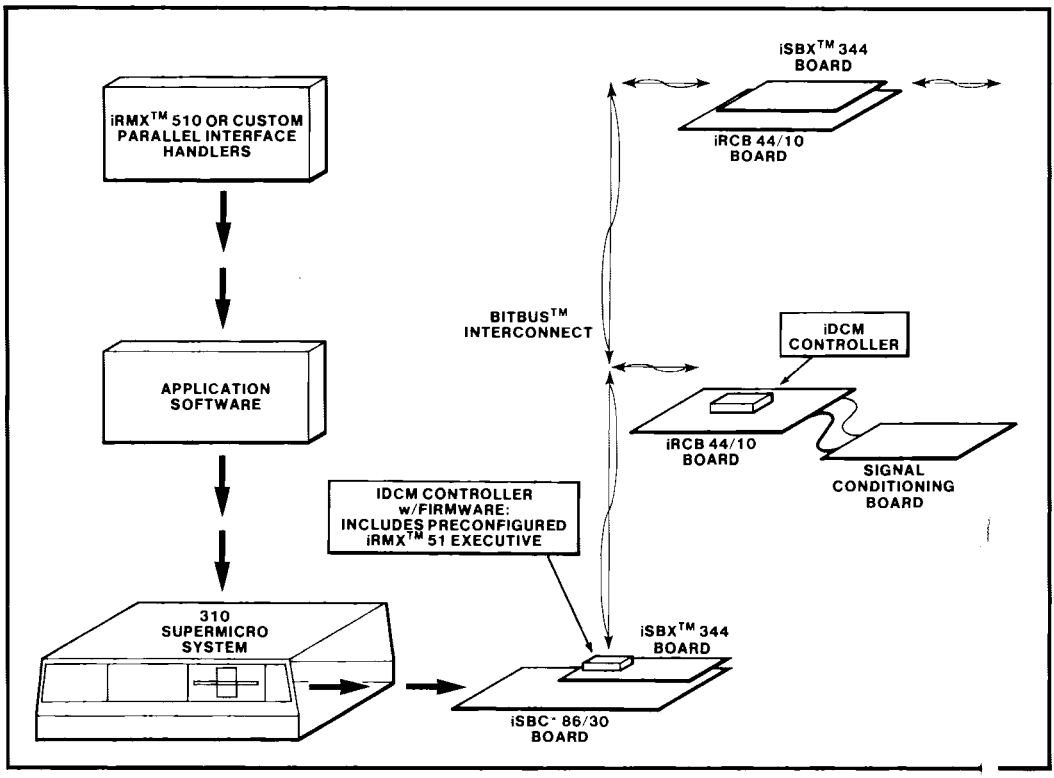


Figure 4. iDCM Operating Environment

SPECIFICATIONS

Supported Hardware

Microcontrollers

8051	80C51
8052	8044
8751	8744
8031	80C31
8032	8344

iDCM Product Line

iSBX 344 MULTIMODULE Board
iRCB 44/10 Remote Controller Board

Compatible Software

iRMX™ 510 iDCM Support Package

Development Tools

ICE™ 51 or ICE 44 Emulators
Intellec Series II or III Development Syst m
iPDS System w/EMV-51
iRMX 510 iDCM Support Package
8051 Software Development Package

Reference Manual (Supplied)

146312-001 — Guide to Using the Distı uted Control Modules

Ordering Information

Part Number	Description
iRMX 51BY	Executive for 8051 Family of Microcontrollers with Reference Manual. A, B, and F Media Formats Supplied

iRMX™ 510
iDCM SUPPORT PACKAGE

- **Low cost remote communication/ control expansion for MULTIBUS® based systems**
- **Extends functionality of BITBUS™/iDCM systems**
- **Software development support for BITBUS™/iDCM products: iSBX™ 344 and iRCB 44/10 boards**
- **Simple software interface for iRMX™ 86, 286, 88, and iPDS™ ISIS operating system compatibility**

The iRMX™ 510 iDCM Support Package contains the necessary software tools to interface MULTIBUS®, and iPDS™ ISIS systems to BITBUS™ systems in both a development environment and during run-time. With other members of the Distributed Control Modules family, the iRMX 510 iDCM Support Package expands Intel's OEM Microcomputer Systems capabilities to include distributed real-time control.

The iRMX 510 Package software interface handlers and the iSBX™ 344 BITBUS Controller MULTIMODULE™ board extend the capabilities of other microprocessors such as the 8086, 80186, or 80286 in iDCM, MULTIBUS, or iPDS systems. Support of iRMX 51 applications is provided via the iRMX 51 libraries incorporated in the iRMX 510 Support Package. Also, the Support Package completes the development environment for BITBUS/iDCM products : iSBX 344 and iRCB 44/10 boards. When used with an ICE-44 Emulator the iDCM controller is accurately simulated resulting in a highly effective product development effort.

MULTIBUS®, iPDS,™ and iDCM SYSTEM EXPANSION

The iRMX 510 Support Package provides the software interface between Intel's MULTIBUS and iPDS environment, and the BITBUS environment. With Intel's Distributed Control Modules hardware interface, the iSBX 344 MULTIMODULE board, this capability enables the user to expand the existing functionality of an iRMX-based SYSTEM 310, for example, to include control and monitoring of a material handling operation. Intel's Personal Development System (iPDS) can be used as a central supervisory station for data acquistion in a laboratory or for program development. The iRMX 510 iDCM Support Package provides a general purpose interface. For custom applications, users may wish to develop a custom interface.

OPERATING ENVIRONMENT

The iRMX 510 Support Package is supplied on diskettes formated for iRMX, Intellec® Series II or III and iPDS ISIS development systems. Application programs or tasks residing on an extension in the iDCM environment may use the iRMX 510 interface. (Application programs or tasks are written in iRMX 88, 86 or ISIS compatible code.) Some examples of extensions in an iDCM system are the iSBC 86/05, 88/25, 186/03 boards and the iPDS system. Figure 2 shows how the iRMX 510 interface is integrated into an iDCM system.

For iRMX 86, 88, or 286R-based systems, configuration of the iRMX 510 interface requires two steps: configuring the interface to the hardware and then the supporting executive. Hardware configuration requires creating a file of configuration parameters, compiling it, and linking the result with the application program. When using the iRMX 510 Package with the iPDS ISIS system, hardware configuration is not required.

ARCHITECTURE

The major functional blocks of the iRMX 510 Support Package are: iRMX 86, 286R, 88 and iPDS ISIS parallel interface handlers, iDCM Controller firmware files, and iRMX 51 include files.

Simple Parallel Interface Handlers

The iRMX 510 Support Package includes parallel interface handlers for systems using the iRMX 86 or 286R Operating System, the iRMX 88 Executive, or Intel's Personal Development System ISIS Operating System. These software handlers pass iRMX 51 messages to and from the iSBX 344 parallel interface (Byte FIFO). In iRMX 86, 286R or 88 — based systems , the interface executes as two tasks: one to transmit, the other to receive the message. In iPDS systems the interface is a procedural call: DCM TRANSMIT, DCM RECIEVE, or DCM STATUS CHECK. In both cases the handlers are straightforward and easy to use. Figure 1 illustrates transmission of a message in an iRMX-based system.

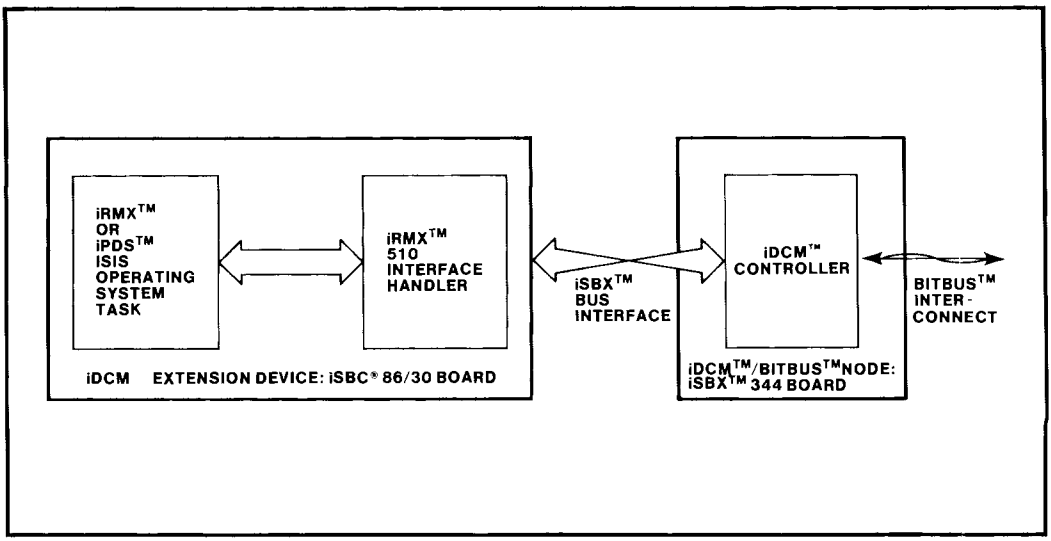


Figure 1. Message Transfer to an iDCM System

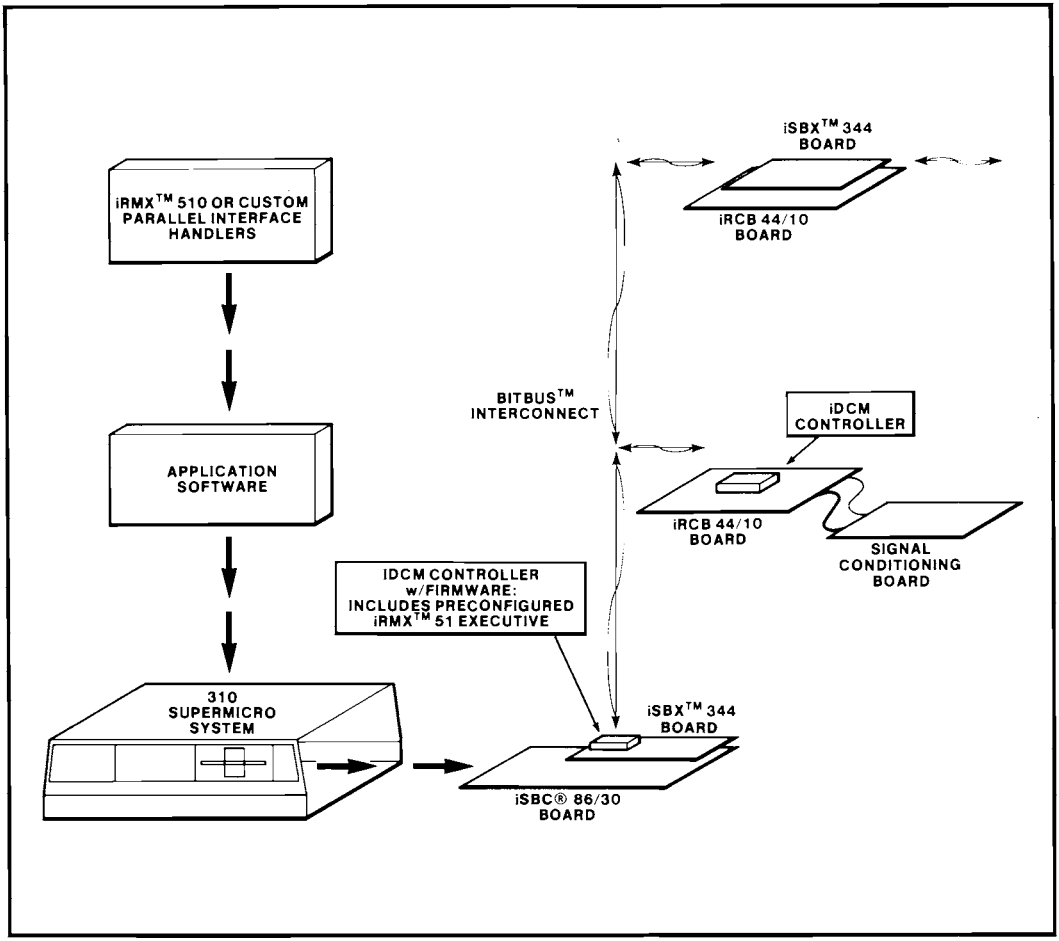


Figure 2. iDCM Operating Environment

The software handlers ease integration of other processors into an iDCM system and provide the tools to quickly expand a MULTIBUS system, or an iPDS ISIS system. Significant reduction in application system software development time results, with more effort concentrated on the overall application.

iDCM Controller Firmware

Also included in the iRMX 510 Support Package is the iDCM Controller firmware in loadable object files, iRMX 51 libraries, and iDCM Controller Include files. An Intellec Development System and ICE-44 Emulator can be used with the loadable object files to accurately simulate the iDCM Controller. This capability significantly decreases development effort by reducing trial and error production of application system software. The iRMX 51 Interface Library and iDCM Controller Include files allow development of user code for iDCM systems.

DEVELOPMENT ENVIRONMENT

The iRMX 510 Support Package completes the development environment for iDCM application system development when used with an Intellec Series II or III Development System, In-Circuit Emulator (ICE-44), and the 8051 Software Development Package. As part of Intel's complete development environment for the 8051 family of microcontrollers, the iRMX 510 Support Package may also be used with an iPDS system and EMV-51 or an Intellec Series II or III Development System and an ICE-51 Emulator.

SPECIFICATIONS

Supported Hardware/Software for iDCM Systems

Operating System	Supported Extension*
iRMX 86 Release 5.0	iSBC 86/05, 86/14, 86/30, 186/03, 186/51, 188/48, 88/25, 88/45 boards
iRMX 88 Release 3.0	iSBC 86/05, 86/14, 86/30, 186/03, 186/51, 188/48, 88/25, 88/45 boards
iRMX 286R	iSBC 286/10 board
ISIS Release 1.0 (PDS)	iPDS System

*Each extension device uses an iSBX 344 BITBUS Controller MULTIMODULE Board

Supported Hardware — 8051 Microcontroller Family

8051	80C51
8052	8044
8751	8744
8031	80C31
8032	8344

Compatible Software

iRMX 86 Release 5.0
iRMX 286R
iRMX 88 Release 3.0
iPDS ISIS Release 1.0
iRMX 51 Release 1.0

Development Tools

ICE-51 or ICE-44 Emulators
iPDS System with EMV-51
Intellec Series II or III Development System
8051 Software Development Package

Reference Manual

146312-001 — Guide to Using the Distributed Control Modules (Supplied)

Ordering Information

Part Number	Description
iRMX 510BY	iDCM Support Package w/ Reference Manual A,B,E, and F Media Formats Supplied.

8044AH/8344AH
HIGH PERFORMANCE 8-BIT MICROCONTROLLER WITH ON-CHIP SERIAL COMMUNICATION CONTROLLER

- **8044AH — CPU/SIU with Factory Mask Programmable ROM**
- **8344AH — An 8044AH used with External Program Memory**
- **8744H — An 8044AH with User Programmable/Erasable EPROM**

8051 MICROCONTROLLER CORE

- **Optimized for Real Time Control 12 MHz Clock, Priority Interrupts, 32 Programmable I/O lines, Two 16-bit Timer/Counters**
- **Boolean Processor**
- **4K x 8 ROM, 192 x 8 RAM**
- **64K Accessible External Program Memory**
- **64K Accessible External Data Memory**
- **4 μs Multiply and Divide**

SERIAL INTERFACE UNIT (SIU)

- **Serial Communication Processor that Operates Concurrently to C.P.U.**
- **2.4 Mbps Maximum Data Rate**
- **375 Kbps using On-Chip Phase Locked Loop**
- **Communication Software in Silicon:**
  - **— Complete Data Link Functions**
  - **— Automatic Station Responses**
- **Operates as an SDLC Primary or Secondary Station**

The RUPI-44 family integrates a high performance 8-bit Microcontroller, the Intel 8051 Core, with an intelligent/high performance HDLC/SDLC serial communication controller, called the Serial Interface Unit (SIU). See Figure 1. This dual architecture allows complex control and high speed data communication functions to be realized cost effectively.

Specifically, the 8044's Microcontroller features: 4K byte On-Chip program memory space; 32 I/O lines; two 16-bit timer/event counters; a 5-source; 2-level interrupt structure; a full duplex serial channel; a Boolean processor; and on-chip oscillator and clock circuitry. Standard TTL and most byte-oriented MCS-80 and MCS-85 peripherals can be used for I/O and memory expansion.

The Serial Interface Unit (SIU) manages the interface to a high speed serial link. The SIU offloads the On-Chip 8051 Microcontroller of communication tasks, thereby freeing the CPU to concentrate on real time control tasks.

The RUPI-44 family consists of the 8044, 8744, and 8344. All three devices are identical except in respect of on-chip program memory. The 8044 contains 4K bytes of mask-programmable ROM. User programmable EPROM replaces ROM in the 8744. The 8344 addresses all program memory externally.

The RUPI-44 devices are fabricated with Intel's reliable +5 volt, silicon-gate HMOSII technology and packaged in a 40-pin DIP.

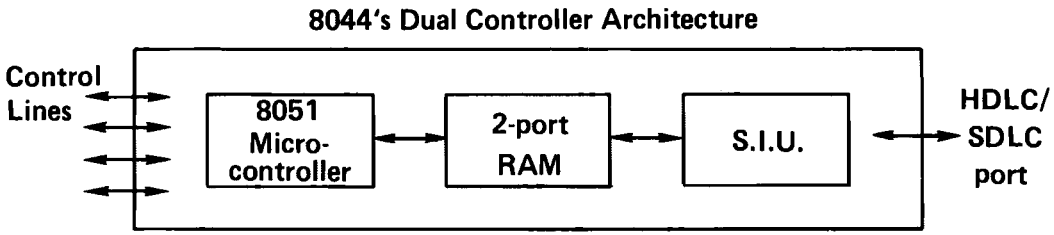


Figure 1. Dual Controller Architecture

Table 1. RUPI™ -44 Family Pin Description

VSS
Circuit ground potential.

VCC
+5V power supply during operation and program verification.

PORT 0
Port 0 is an 8-bit open drain bidirectional I/O port. It is also the multiplexed low-order address and data bus when using external memory. It is used for data output during program verification. Port 0 can sink/source eight LS TTL loads.

PORT 1
Port 1 is an 8-bit quasi-bidirectional I/O port. It is used for the low-order address byte during program verification. Port 1 can sink/source four LS TTL loads.

In non-loop mode two of the I/O lines serve alternate functions:
—$\overline{RTS}$ (P1.6). Request-to-Send output. A low indicates that the RUPI-44 is ready to transmit.
—$\overline{CTS}$ (P1.7) Clear-to-Send input. A low indicates that a receiving station is ready to receive.

PORT 2
Port 2 is an 8-bit quasi-bidirection I/O port. It also emits the high-order address byte when accessing external memory. It is used for the high-order address and the control signals during program verification. Port 2 can sink/source four LS TTL loads.

PORT 3
Port 3 is an 8-bit quasi-bidirectional I/O port. It also contains the interrupt, timer, serial port and RD and WR pins that are used by various options. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. Port 3 can sink/source four LS TTL loads.

In addition to I/O, some of the pins also serve alternate functions as follows:
—I/$\overline{O}$ RxD (P3.0). In point-to-point or multipoint configurations, this pin controls the direction of pin P3.1. Serves as Receive Data input in loop and diagnostic modes.
—DATA TxD (P3.1) In point-to-point or multipoint configurations, this pin functions as data input/output. In loop mode, it serves as transmit pin. A '0' written to this pin enables diagnostic mode.
—$\overline{INT0}$ (P3.2). Interrupt 0 input or gate control input for counter 0.
—$\overline{INT1}$ (P3.3). Interrupt 1 input or gate control input for counter 1.
—TO(P3.4). Input to counter 0.
—SCLK T1 (P3.5). In addition to I/O, this pin provides input to counter 1 or serves as SCLK (serial clock) input.
—$\overline{WR}$ (P3.6). The write control signal latches the data byte from Port 0 into the External Data Memory.
—$\overline{RD}$ (P3.7). The read control signal enables External Data Memory to Port 0.

RST/VPD
A high level on this pin resets the RUPI-44. A small internal pulldown resistor permits power-on reset using only a capacitor connected to VCC. If VPD is held within its spec while VCC drops below spec, VPD will provide standby power to the RAM. When VPD is low, the RAM's current is drawn from VCC.

ALE/$\overline{PROG}$
Provides Address Latch Enable output used for latching the address into external memory during normal operation. It is activitated every six oscillator periods except during an external data memory access. It also receives the program pulse input for programming the EPROM version.

$\overline{PSEN}$
The Program Store Enable output is a control signal that enables the external Program Memory to the bus during external fetch operations. It is activated every six oscillator periods, except during external data memory accesses. Remains high during internal program execution.

$\overline{EA}$/VPP
When held at a TTL high level, the RUPI-44 executes instructions from the internal ROM when the PC is less than 4096. When held at a TTL low level, the RUPI-44 fetches all instructions from external Program Memory. The pin also receives the 21V EPROM programming supply voltage on the 8744.

XTAL 1
Input to the oscillator's high gain amplifier. Required when a crystal is used. Connect to VSS when external source is used on XTAL 2.

XTAL 2
Output from the oscillator's amplifier. Input to the internal timing circuitry. A crystal or external source can be used.

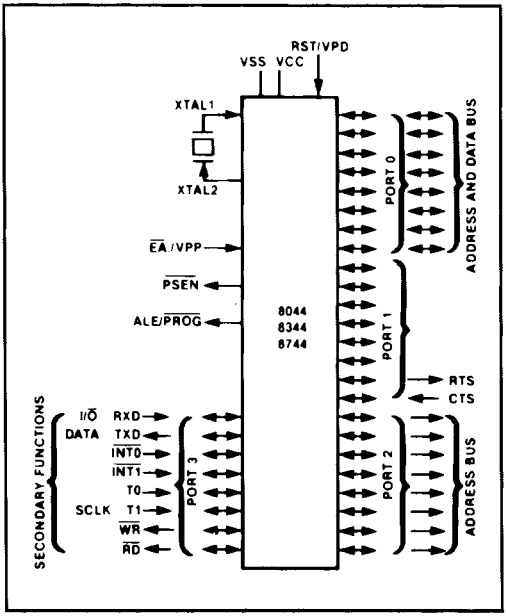


Figure 2. Logic Symbol

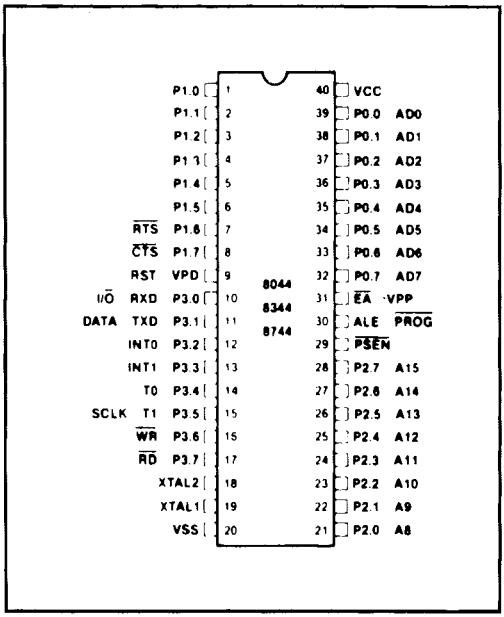


Figure 3. Pin Configuration

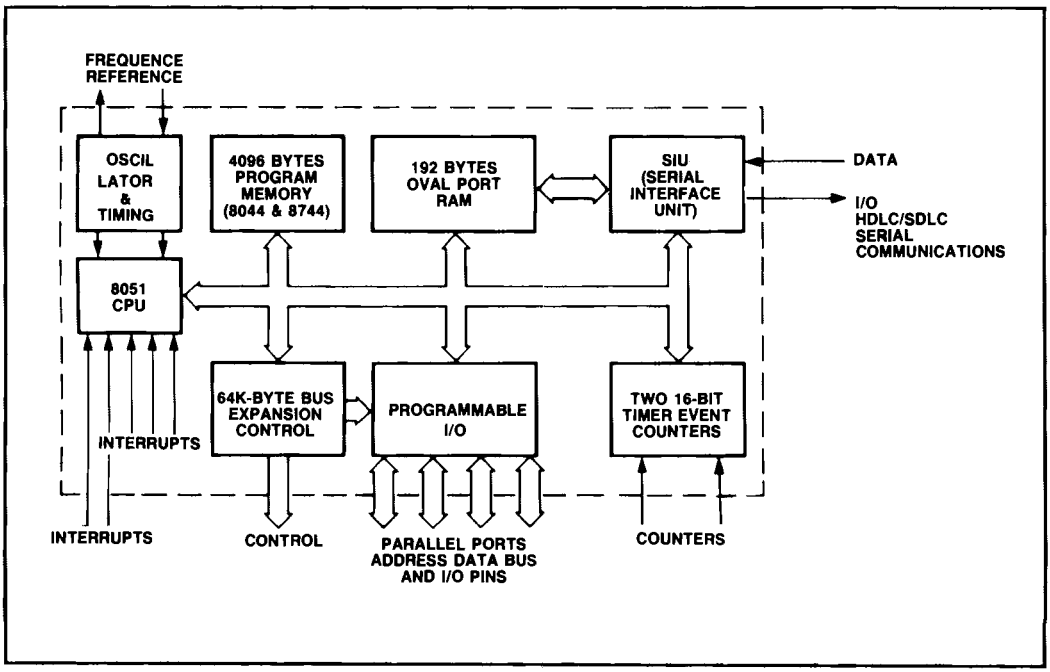


Figure 4. Block Diagram

Functional Description

General

The 8044 integrates the powerful 8051 microcontroller with an intelligent Serial Communication Controller to provide a single-chip solution which will efficiently implement a distributed processing or distributed control system. The microcontroller is a self-sufficient unit containing ROM, RAM, ALU, and its own peripherals. The 8044's architecture and instruction set are identical to the 8051's. The 8044 replaces the 8051's serial interface with an intelligent SDLC/HDLC Serial Interface Unit (SIU). 64 more bytes of RAM have been added to the 8051 RAM array. The SIU can communicate at bit rates up to 2.4 M bps. The SIU works concurrently with the Microcontroller so that there is no throughput loss in either unit. Since the SIU possesses its own intelligence, the CPU is off-loaded from many of the communications tasks, thus dedicating more of its computing power to controlling local peripherals or some external process.

The Microcontroller

The microcontroller is a stand-alone high-performance single-chip computer intended for use in sophisticated real-time application such as instrumentation, industrial control, and intelligent computer peripherals.

The major features of the microcontroller are:

- 8-bit CPU
- on-chip oscillator
- 4K bytes of ROM
- 192 bytes of RAM
- 32 I/O lines
- 64K address space for external Data Memory
- 64K address space for external Program Memory
- two fully programmable 16-bit timer/counters
- a five-source interrupt structure with two priority levels
- bit addressability for Boolean processing
- 1 μsec instruction cycle time for 60% of the instructions 2 μsec instruction cycle time for 40% of the instructions
- 4 μsec cycle time for 8 by 8 bit unsigned Multiply/Divide

Internal Data Memory

Functionally the Internal Data Memory is the most flexible of the address spaces. The Internal Data Memory space is subdivided into a 256-byte Internal Data RAM address space and a 128-byte Special Function Register address space as shown in Figure 5.

The Internal Data RAM address space is 0 to 255. Four 8-Register Banks occupy locations 0 through 31. The stack can be located anywhere in the Internal Data RAM address space. In addition, 128 bit locations of the on-chip RAM are accessible through Direct Addressing. These bits reside in Internal Data RAM at byte locations 32 through 47. Currently locations 0 through 191 of the Internal Data RAM address space are filled with on-chip RAM.

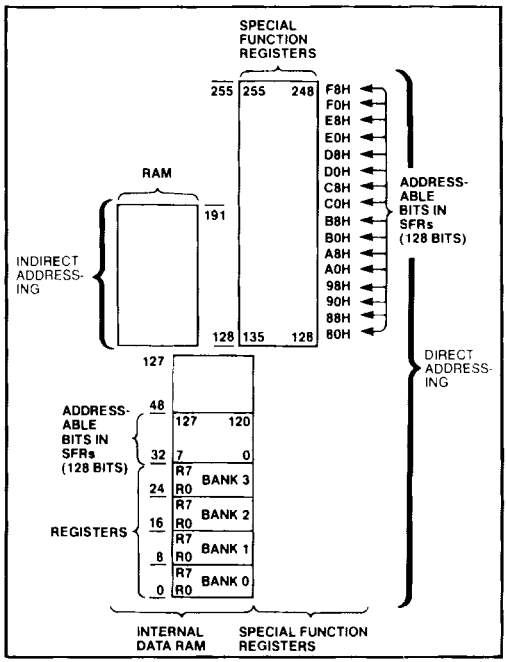


Figure 5. Internal Data Memory Address Space

Parallel I/O

The 8044 has 32 general-purpose I/O lines which are arranged into four groups of eight lines. Each group is called a port. Hence there are four ports; Port 0, Port 1, Port 2, and Port 3. Up to five lines from 1 and Port 2 are dedicated to supporting the serial channel when the SIU is invoked. Due to the nature of the serial port, two of Port 3's I/O lines (P3.0 and P3.1) do not have latched outputs. This is true whether or not the serial channel is used.

Table 1. MCS®-51 Instruction Set Description

ARITHMETIC OPERATIONS

Mnemonic		Description	Byte	Cyc
ADD	A,Rn	Add register to Accumulator	1	1
ADD	A,direct	Add direct byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#data	Add immediate data to Accumulator	2	1
ADDC	A,Rn	Add register to Accumulator with Carry	1	1
ADDC	A,direct	Add direct byte to A with Carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with Carry flag	1	1
ADDC	A,#data	Add immediate data to A with Carry flag	2	1
SUBB	A,Rn	Subtract register from A with Borrow	1	1
SUBB	A,direct	Subtract direct byte from A with Borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A with Borrow	1	1
SUBB	A,#data	Subtract immed data from A with Borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
INC	DPTR	Increment Data Pointer	1	2
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
MUL	AB	Multiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1

LOGICAL OPERATIONS

Mnemonic		Destination	Byte	Cyc
ANL	A,Rn	AND register to Accumulator	1	1
ANL	A,direct	AND direct byte to Accumulator	2	1
ANL	A,@Ri	AND indirect RAM to Accumulator	1	1
ANL	A,#data	AND immediate data to Accumulator	2	1
ANL	direct,A	AND Accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to Accumulator	1	1
ORL	A,direct	OR direct byte to Accumulator	2	1

LOGICAL OPERATIONS (CONTINUED)

Mnemonic		Destination	Byte	Cyc
ORL	A,@Ri	OR indirect RAM to Accumulator	1	1
ORL	A,#data	OR immediate data to Accumulator	2	1
ORL	direct,A	OR Accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive-OR register to Accumulator	1	1
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	1
XRL	A,@Ri	Exclusive-OR indirect RAM to A	1	1
XRL	A,#data	Exclusive-OR immediate data to A	2	1
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate A Left through the Carry flag	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate A Right through Carry flag	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1

DATA TRANSFER

Mnemonic		Description	Byte	Cyc
MOV	A,Rn	Move register to Accumulator	1	1
MOV	A,direct	Move direct byte to Accumulator	2	1
MOV	A,@Ri	Move indirect RAM to Accumulator	1	1
MOV	A,#data	Mov immediate data to Accumulator	2	1
MOV	Rn,A	Move Accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move Accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2

Table 1. (Cont.)

DATA TRANSFER (CONTINUED)

Mnemonic		Description	Byte	Cyc
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move Accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	2
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A	1	2
MOVC	A,@A+PC	Move Code byte relative to PC to A	1	2
MOVX	A,@Ri	Move External RAM (8-bit addr) to A	1	2
MOVX	A,@DPTR	Move External RAM (16-bit addr) to A	1	2
MOVX	@Ri,A	Move A to External RAM (8-bit addr)	1	2
MOVX	@DPTR,A	Move A to External RAM (16-bit addr)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with Accumulator	1	1
XCH	A,direct	Exchange direct byte with Accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with A	1	1
XCHD	A,@Ri	Exchange low-order Digit ind RAM w A	1	1

BOOLEAN VARIABLE MANIPULATION

Mnemonic		Description	Byte	Cyc
CLR	C	Clear Carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set Carry flag	1	1
SETB	bit	Set direct Bit	2	1
CPL	C	Complement Carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to Carry flag	2	2
ANL	C,/bit	AND complement of direct bit to Carry	2	2
ORL	C/bit	OR direct bit to Carry flag	2	2
ORL	C,/bit	OR complement of direct bit to Carry	2	2
MOV	C,/bit	Move direct bit to Carry flag	2	1
MOV	bit,C	Move Carry flag to direct bit	2	2

PROGRAM AND MACHINE CONTROL

Mnemonic		Description	Byte	Cyc
ACALL	addr11	Absolute Subroutine Call	2	2
LCALL	addr16	Long Subroutine Call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute Jump	2	2
LJMP	addr16	Long Jump	3	2
SJMP	rel	Short Jump (relative addr)	2	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if Accumulator is Zero	2	2
JNZ	rel	Jump if Accumulator is Not Zero	2	2
JC	rel	Jump if Carry flag is set	2	2
JNC	rel	Jump if No Carry flag	2	2
JB	bit,rel	Jump if direct Bit set	3	2
JNB	bit,rel	Jump if direct Bit Not set	3	2
JBC	bit,rel	Jump if direct Bit is set & Clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & Jump if Not Equal	3	2
CJNE	A,#data,rel	Comp, immed, to A & Jump if Not Equal	3	2
CJNE	Rn,#data,rel	Comp, immed, to reg & Jump if Not Equal	3	2
CJNE	@Ri,#data,rel	Comp, immed, to ind, & Jump if Not Equal	3	2
DJNZ	Rn,rel	Decrement register & Jump if Not Zero	2	2
DJNZ	direct,rel	Decrement direct & Jump if Not Zero	3	2
NOP		No operation	1	1

Notes on data addressing modes:

Rn —Working register R0-R7
direct —128 internal RAM locations, any I/O port, control or status register
@Ri —Indirect internal RAM location addressed by register R0 or R1
#data —8-bit constant included in instruction
#data16 —16-bit constant included as bytes 2 & 3 of instruction
bit —128 software flags, any I/O pin, control or status bit

Notes on program addressing modes:

addr16 —Destination address for LCALL & LJMP may be anywhere within the 64-K program memory address space
Addr11 —Destination address for ACALL & AJMP will be within the same 2-K page of program memory as the first byte of the following instruction
rel —SJMP and all conditional jumps include an 8-bit offset byte, Range is +127-128 bytes relative to first byte of the following instruction

Port 0 and Port 2 also have an alternate dedicated function. When placed in the external access mode, Port 0 and Port 2 become the means by which the 8044 communicates with external program memory. Port 0 and Port 2 are also the means by which the 8044 communicates with external data memory. Peripherals can be memory mapped into the address space and controlled by the 8044.

Timer/Counters

The 8044 contains two 16-bit counters which can be used for measuring time intervals, measuring pulse widths, counting events, generating precise periodic interrupt requests, and clocking the serial communications. Internally the Timers are clocked at 1/12 of the crystal frequency, which is the instruction cycle time. Externally the counters can run up to 500 KHz.

Interrupt System

External events and the real-time driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a sophisticated multiple-source, two priority level, nested interrupt system is provided. Interrupt response latency ranges from 3 μsec to 7 μsec when using a 12 MHz clock.

All five interrupt sources can be mapped into one of the two priority levels. Each interrupt source can be enabled or disabled individually or the entire interrupt system can be enabled or disabled. The five interrupt sources are: Serial Interface Unit, Timer 1, Timer 2, and two external interrupts. The external interrupts can be either level or edge triggered.

Serial Interface Unit (SIU)

The Serial Interface Unit is used for HDLC/SDLC communications. It handles Zero Bit Insertion/Deletion, Flags, automatic address recognition, and a 16-bit cyclic redundancy check. In addition it implements in hardware a subset of the SDLC protocol such that it responds to many SDLC frames without CPU intervention. In certain applications it is advantageous to have the CPU control the reception or transmission of every single frame. For this reason the SIU has two modes of operation: "AUTO" and "FLEXIBLE" (or "NON-AUTO"). It is in the AUTO mode that the SIU responds to SDLC frames without CPU intervention; whereas, in the FLEXIBLE mode the reception or transmission of every single frame will be under CPU control.

There are three control registers and eight parameter registers that are used to operate the serial interface. These registers are shown in Figure 5 and Figure 6. The control registers set the modes of operation and provide status information. The eight parameter registers buffer the station address, receive and transmit control bytes, and point to the on-chip transmit and receive buffers.

Data to be received or transmitted by the SIU must be buffered anywhere within the 192 bytes of on-chip RAM. Transmit and receive buffers are not allowed to "wrap around" in RAM; a "buffer end" is generated after address 191 is reached.

AUTO Mode

In the AUTO mode the SIU implements in hardware a subset of the SDLC protocol such that it responds to many SDLC frames without CPU intervention. All AUTO mode responses to the primary station will conform to IBM's SDLC definition. The advantages of the AUTO mode are that less software is required to implement a secondary station, and the hardware generated response to polls is much faster than doing it in software. However, the Auto mode can not be used at a primary station.

To transmit in the AUTO mode the CPU must load the Transmit Information Buffer, Transmit Buffer Start register, Transmit Buffer Length register, and set the Transmit Buffer Full bit. The SIU automatically responds to a poll by transmitting an information frame with the P/F bit in the control field set. When the SIU receives a positive acknowledgement from the primary station, it automatically increments the Ns field in the NSNR register and interrupts the CPU. A negative acknowledgement would cause the SIU to retransmit the frame.

To receive in the AUTO mode, the CPU loads the Receive Buffer Start register, the Receive Buffer Length register, clears the Receive Buffer Protect bit, and sets the Receive Buffer Empty bit. If the SIU is polled in this state, and the TBF bit indicates that the Transmit Buffer is empty, an automatic RR response will be generated. When a valid information frame is received the SIU will automatically increment Nr in the NSNR register and interrupt the CPU.

While in the AUTO mode the SIU can recognize and respond to the following commands without CPU intervention: I (Information), RR (Receive Ready), RNR (Receive Not Ready), REJ (Reject), and UP (Unnumbered Poll). The SIU can generate

REGISTER NAMES	SYMBOLIC ADDRESS	BIT ADDRESS	BYTE ADDRESS	
B REGISTER	B	247 through 240	240	(F0H)
ACCUMULATOR	ACC	231 through 224	224	(E0H)
*THREE BYTE FIFO	FIFO		223	(DFH)
	FIFO		222	(DEH)
	FIFO		221	(DDH)
TRANSMIT BUFFER START	TBS		220	(DCH)
TRANSMIT BUFFER LENGTH	TBL		219	(DBH)
TRANSMIT CONTROL BYTE	TCB		218	(DAH)
* SIU STATE COUNTER	SIUST		217	(D9H)
SEND COUNT RECEIVE COUNT	NSNR	223 through 216	216	(D8H)
PROGRAM STATUS WORD	PSW	215 through 208	208	(D0H)
*DMA COUNT	DMA CNT		207	(CFH)
STATION ADDRESS	STAD		206	(CEH)
RECEIVE FIELD LENGTH	RFL		205	(CDH)
RECEIVE BUFFER START	RBS		204	(CCH)
RECEIVE BUFFER LENGTH	RBL		203	(CBH)
RECEIVE CONTROL BYTE	RCB		202	(CAH)
SERIAL MODE	SMD		201	(C9H)
STATUS REGISTER	STS	207 through 200	200	(C8H)
INTERRUPT PRIORITY CONTROL	IP	191 through 184	184	(B8H)
PORT 3	P3	183 through 176	176	(B0H)
INTERRUPT ENABLE CONTROL	IE	175 through 168	168	(A8H)
PORT 2	P2	167 through 160	160	(A0H)
PORT 1	P1	151 through 144	144	(90H)
TIMER HIGH 1	TH1		141	(8DH)
TIMER HIGH 0	TH0		140	(8CH)
TIMER LOW 1	TL1		139	(8BH)
TIMER LOW 0	TL0		138	(8AH)
TIMER MODE	TMOD		137	(89H)
TIMER CONTROL	TCON	143 through 136	136	(88H)
DATA POINTER HIGH	DPH		131	(83H)
DATA POINTER LOW	DPL		130	(82H)
STACK POINTER	SP		129	(81H)
PORT 0	P0	135 through 128	128	(80H)

SFR's CONTAINING DIRECT ADDRESSABLE BITS

*ICE Support Hardware registers. Under normal operating conditions there is no need for the CPU to access these registers.

Figure 5. Mapping of Special Function Registers

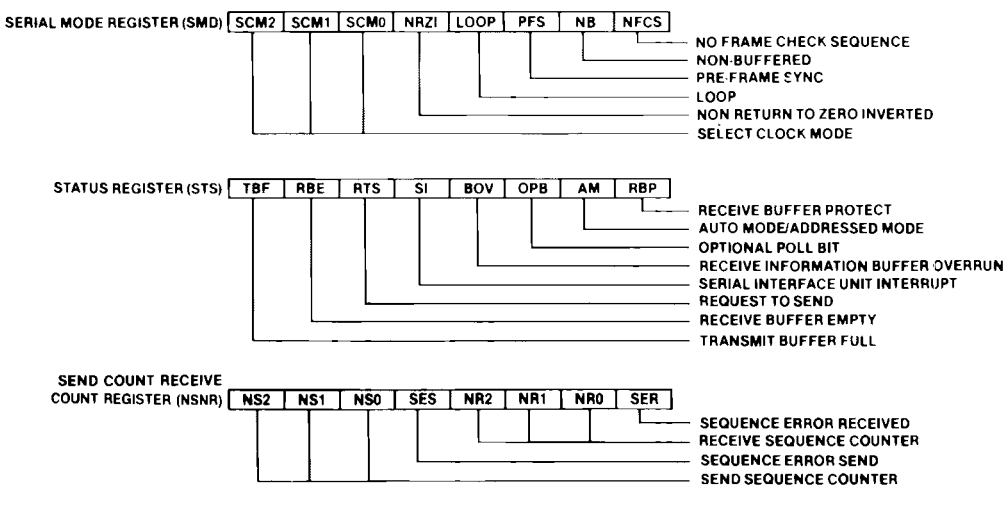


Figure 6. Serial Interface Unit Control Registers

the following responses without CPU intervention: I (Information), RR (Receive Ready), and RNR (Receive Not Ready).

When the Receive Buffer Empty bit (RBE) indicates that the Receive Buffer is empty, the receiver is enabled, and when the RBE bit indicates that the Receive Buffer is full, the receiver is disabled. Assuming that the Receive Buffer is empty, the SIU will respond to a poll with an I frame if the Transmit Buffer is full. If the Transmit Buffer is empty, the SIU will respond to a poll with a RR command if the Receive Buffer Protect bit (RBP) is cleared, or an RNR command if RBP is set.

FLEXIBLE (or NON-AUTO) Mode

In the FLEXIBLE mode all communications are under control of the CPU. It is the CPU's task to encode and decode control fields, manage acknowledgements, and adhere to the requirements of the HDLC/SDLC protocols. The 8044 can be used as a primary or a secondary station in this mode.

To receive a frame in the FLEXIBLE mode, the CPU must load the Receive Buffer Start register, the Receive Buffer Length register, clear the Receive Buffer Protect bit, and set the Receive Buffer Empty bit. If a valid opening flag is received and the address field matches the byte in the Station Address register or the address field contains a broadcast address, the 8044 loads the control field in the receive control byte register, and loads the I field in the receive buffer. If there is no CRC error, the SIU interrupts the CPU, indicating a frame has just been received. If there is a CRC error, no interrupt occurs. The Receive Field Length register provides the number of bytes that were received in the information field.

To transmit a frame, the CPU must load the transmit information buffer, the Transmit Buffer Start register, the Transmit Buffer Length register, the Transmit Control Byte, and set the TBF and the RFS bit. The SIU, unsolicited by an HDLC/SDLC frame, will transmit the entire information frame, and interrupt the CPU, indicating the completion of transmission. For supervisory frames or unnumbered frames, the transmit buffer length would be 0.

CRC

The FCS register is initially set to all 1's prior to calculating the FCS field. The SIU will not interrupt the CPU if a CRC error occurs (in both AUTO and FLEXIBLE modes). The CRC error is cleared upon receiving of an opening flag.

Frame Format Options

In addition to the standard SDLC frame format, the 8044 will support the frames displayed in Figure 7. The standard SDLC frame is shown at the top of this figure. For the remaining frames the information field will incorporate the control or address bytes and the frame check sequences; therefore these fields will be stored in the Transmit and Receive buffers. For example, in the non-buffered mode the third byte is treated as the beginning of the information field. In the non-addressed mode, the information field begins after the opening flag. The mode bits to set the frame format options are found in the Serial Mode register and the Status register.

EXTENDED ADDRESSING

To realize an extended control field or an extended address field using the HDLC protocol, the FLEXIBLE mode must be used. For an extended control field, the SIU is programmed to be in the non-buffered mode. The extended control field will be the first and second bytes in the Receive and Transmit Buffers. For extended addressing the SIU is placed in the non-addressed mode. In this mode the CPU must implement the address recognition for received frames. The addressing field will be the initial bytes in the Transmit and Receive buffers followed by the control field.

The SIU can transmit and receive only frames which are multiples of 8 bits. For frames received with other than 8-bit multiples, a CRC error will cause the SIU to reject the frame.

SDLC Loop Networks

The SIU can be used in a ADLC loop as a secondary or primary station. When the SIU is placed in the Loop mode it receives the data on pin 10 and transmits the data one bit time delayed on pin 11. It can also recognize the Go ahead signal and change it into a flag when it is ready to transmit. As a secondary station the SIU can be used in the AUTO or FLEXIBLE modes. As a primary station the FLEXIBLE mode is used; however, additional hardware is required for generating the Go Ahead bit pattern. In the Loop mode the maximum data rate is 1 Mbps clocked or 375 Kbps self-clocked.

SDLC Multidrop Networks

The SIU can be used in a SDLC non-loop configuration as a secondary or primary station. When the SIU is placed in the non-loop mode, data is received and transmitted on pin 11, and pin 10 drives a tri-state buffer. In non-loop mode, modem interface pins, RTS and CTS, become available.

FRAME OPTION	NFCS	NB	AM'	FRAME FORMAT
Standard SDLC NON-AUTO Mode	0	0	0	F A C I FCS F
Standard SDLC AUTO Mode	0	0	1	F A C I FCS F
Non-Buffered Mode NON-AUTO Mode	0	1	1	F A I FCS F
Non-Addressed Mode NON-AUTO Mode	0	1	0	F I FCS F
No FCS Field NON-AUTO Mode	1	0	0	F A C I F
No FCS Field AUTO Mode	1	0	1	F A C I F
No FCS Field Non-Buffered Mode NON-AUTO Mode	1	1	1	F A I F
No FCS Field Non-Addressed Mode NON-AUTO Mode	1	1	0	F I F

Mode Bits:
- AM — "AUTO" Mode/Addressed Mode
- NB — Non-Buffered Mode
- NFCS — No FCS Field Mode

Key to Abbreviations:
- F = Flag (01111110)
- A = Address Field
- C = Control Field
- I = Information Field
- FCS = Frame Check Sequence

Note 1: The AM bit function is controlled by the NB bit. When NB = 0, AM becomes AUTO mode select, when NB = 1, AM becomes Address mode select.

Figure 7. Frame Format Options

Data Clocking Options

The 8044's serial port can operate in an externally clocked or self clocked system. A clocked system provides to the 8044 a clock synchronization to the data. A self-clocked system uses the 8044's on-chip Digital Phase Locked Loop (DPLL) to recover the clock from the data, and clock this data into the Serial Receive Shift Register.

In this mode, a clock synchronized with the data is externally fed into the 8044. This clock may be generated from an External Phase Locked Loop, or possibly supplied along with the data. The 8044 can transmit and receive data in this mode at rates up to 2.4 Mbps.

This self clocked mode allows data transfer without a common system data clock. An on-chip Digital Phase Locked Loop is employed to recover the data clock which is encoded in the data stream. The DPLL will converge to the nominal bit center within eight bit transitions, worst case. The DPLL requires a reference clock of either 16 times (16x) or 32 times (32x) the data rate. This reference clock may be externally applied or internally generated. When internally generated either the 8044's internal logic clock (crystal frequency divided by two) or the timer 1 overflow is used as the reference clock. Using the internal timer 1 clock the data reates can vary from 244 to 62.5 Kbps. Using the internal logic clock at a 16x sampling rate, receive data can either be 187.5 Kbps, or 375 Kbps. When the reference clock for the DPLL is externally applied the data rates can vary from 0 to 375 Kbps at a 16x sampling rate.

To aid in a Phase Locked Loop capture, the SIU has a NRZI (Non Return to Zero Inverted) data encoding and decoding option. Additionally the SIU has a preframe sync option that transmits two bytes of alternating 1's and 0's to ensure that the receive station DPLL will be synchronized with the data by the time it receives the opening flag.

Control and Status Registers

There are three SIU Control and Status Registers:

Serial Mode Register (SMD)

Status/Command Register (STS)

Send/Receive Count Register (NSNR)

The SMD, STS, and NSNR registers are all cleared by system reset. This assures that the SIU will power up in an idle state (neither receiving nor transmitting).

These registers and their bit assignments are described below.

SMD: Serial Mode Register (byte-addressable)

Bit: 7	6	5	4	3	2	1	0
SCM2	SCM1	SCM0	NRZI	LOOP	PFS	NB	NFCS

The Serial Mode Register (Address C9H) selects the operational modes of the SIU. The 8044 CPU can both read and write SMD. The SIU can read SMD but cannot write to it. To prevent conflict between CPU and SIU access to SMD, the CPU should write SMD only when the Request To Send (RTS) and Receive Buffer Empty (RBE) bits (in the STS register) are both false (0). Normally, SMD is accessed only during initialization.

The individual bits of the Serial Mode Register are as follows:

Bit #	Name	Description
SMD.0	NFCS	No FCS field in the SDLC frame.
SMD.1	NB	Non-Buffered mode. No control field in the SDLC frame.
SMD.2	PFS	Pre-Frame Sync mode. In this mode, the 8044 transmits two bytes before the first flag of a frame, for DPLL synchronization. If NRZI is enabled, 00H is sent; otherwise, 55H is sent. In either case, 16 pre-frame transitions are guaranteed.
SMD.3	LOOP	Loop configuration.
SMD.4	NRZI	NRZI coding option.
SMD.5	SCM0	Select Clock Mode — Bit 0
SMD.6	SCM1	Select Clock Mode — Bit 1
SMD.7	SCM2	Select Clock Mode — Bit 2

The SCM bit decode as follows:

SCM 2	1	0	Clock Mode	Data Rate (Bits/sec)*
0	0	0	Externally clocked	0-2.4M**
0	0	1	Reserved	
0	1	0	Self clocked, timer overflow	244-62.5K
0	1	1	Reserved	
1	0	0	Self clocked, external 16x	0-375K
1	0	1	Self clocked, external 32x	0-187.5K
1	1	0	Self clocked, internal fixed	375K
1	1	1	Self clocked, internal fixed	187.5K

*Based on a 12 Mhz crystal frequency
**0-1M bps in loop configuration

STS: Status/Command Register (bit-addressable)

Bit: 7	6	5	4	3	2	1	0
TBE	RBE	RTS	SI	BOV	OPB	AM	RBP

The Status/Command Register (Address C8H) provides operational control of the SIU by the 8044 CPU, and enables the SIU to post status information for the CPU's access. The SIU can read STS, and can alter certain bits, as indicated below. The CPU can both read and write STS asynchronously. However, 2-cycle instructions that access STS during both cycles ('JBC/B, REL' and 'MOV /B,C.') should not be used, since the SIU may write to STS between the two CPU accesses.

The individual bits of the Status/Command Register are as follows:

Bit #	Name	Description
STS.0	RBP	Receive Buffer Protect. Inhibits writing of data into the receive buffer. In AUTO mode, RBP forces an RNR response instead of an RR.
STS.1	AM	AUTO Mode/Addressed Mode. Selelcts AUTO mode where AUTO mode is allowed. If NB is true, (=1), the AM bit selects the addressed mode. AM may be cleared by the SIU.
STS.2	OPB	Optional Poll Bit. Determines whether the SIU will generate an AUTO response to an optional poll (UP with P=0). OPB may be set or cleared by the SIU.
STS.3	BOV	Receive Buffer Overrun. BOV may be set or cleared by the SIU.
STS.4	SI	SIU Interrupt. This is one of the five interrupt sources to the CPU. The vector location = 23H. SI may be set by the SIU. It should be cleared by the CPU before returning from an interrupt routine

STS.5 RTS Request To Send. Indicates that the 8044 is ready to transmit or is transmitting. RTS may be read or written by the CPU. RTS may be read by the SIU, and in AUTO mode may be written by the SIU.

STS.6 RBE Receive Buffer Empty. RBE can be thought of as Receive Enable. RBE is set to one by the CPU when it is ready to receive a frame, or has just read the buffer, and to zero by the SIU when a frame has been received.

STS.7 TBF Transmit Buffer Full. Written by the CPU to indicate that it has filled the transmit buffer. TBF may be cleared by the SIU.

NSNR: Send/Receive Count Register (Bit-addressable)

Bit: 7 6 5 4 3 2 1 0

NS2	NS1	NS0	SES	NR2	NR1	NR0	SER

The Send/Receive Count Register (Address D8H) contains the transmit and receive sequence numbers, plus tally error indications. The SIU can both read and write NSNR. The 8044 CPU can both read and write NSNR asynchronously. However, 2-cycle instructions that access NSNR during both cycles ('JBC /B, REL', and 'MOV /B,C') should not be used, since the SIU may write to NSNR between the two 8044 CPU accesses.

The individual bits of the Send/Receive Count Register are as follows:

Bit #	Name	Description
NSNR.0	SER	Receive Sequence Error: NS (P) ≠ NR (S)
NSNR.1	NR0	Receive Sequence Counter — Bit 0
NSNR.2	NR1	Receive Sequence Counter — Bit 1
NSNR.3	NR2	Receive Sequence Counter — Bit 2
NSNR.4	SES	Send Sequence Error: NP (P) ≠ NS (S) and NP (P) ≠ NS (S) + 1
NSNR.5	NS0	Send Sequence Counter — Bit 0
NSNR.6	NS1	Send Sequence Counter — Bit 1
NSNR.7	NS2	Send Sequence Counter — Bit 2

Parameter Registers

There are eight parameter registers that are used in connections with SIU operation. All eight registers may be read or written by the 8044 CPU. RFL and RCB are normally loaded by the SIU.

The eight parameter registers are as follows:

STAD: Station Address Register (byte-addressable)

The station Address register (Address CEH) contains the station address. To prevent access conflict, the CPU should access STAD only when the SIU is idle (RTS = 0 and RBE = 0). Normally, STAD is accessed only during initialization.

TBS: Transmit Buffer Start Address Register (byte-addressable)

The Transmit Buffer Start address register (Address DCH) points to the location in on-chip RAM for the beginning of the I-field of the frame to be transmitted. The CPU should access TBS only when the SIU is not transmitting a frame (when TBF = 0).

TBL: Transmit Buffer Length Register (byte-addressable)

The Transmit Buffer Length register (Address DBH) contains the length (in bytes) of the I-field to be transmitted. A blank I-field (TBL = 0) is valid. The CPU should access TBL only when the SIU is not transmitting a frame (when TBF = 0)

NOTE: The transmit and receive buffers are not allowed to "wrap around" in the on-chip RAM. A "buffer end" is automatically generated if address 191 (BFH) is reached.

TCB: Transmit Control Byte Register (byte-addressable)

The Transmit Control Byte register (Address DAH) contains the byte which is to be placed in the control field of the transmitted frame, during NON-AUTO mode transmission. The CPU should access TCB only when the SIU is not transmitting a frame (when TBF = 0). The N_S and N_R counters are not used in the NON-AUTO mode.

RBS: Receive Buffer Start Address Register (byte-addressable)

The Receive Buffer Start address register (Address CCH) points to the location in on-chip RAM where the beginning of the I-field of the frame being received is to be stored. The CPU should write RBS only when the SIU is not receiving a frame (when RBE = 0).

RBL: Receive Buffer Length Register (byte-addressable)

The Receive Buffer Length register (Address CBH contains the length (in bytes) of the area in on-chip RAM allocated for the received I-field. RBL = 0 is valid. The CPU should write RBL only when RBE = 0.

RFL: Receive Field Length Register (byte-addressable)

The Received Field Length register (Address CDH) contains the length (in bytes) of the received I-field that has just been loaded into on-chip RAM. RFL is loaded by the SIU. RFL = 0 is valid. RFL should be accessed by the CPU only when RBE = 0.

RCB: Receive Control Byte Register (byte-addressable)

The Received Control Byte register (Address CAH) contains the control field of the frame that has just been received. RCB is loaded by the SIU. The CPU can only read RCB, and should only access RCB when RBE = 0.

ICE Support Registers.

The 8044 In-Circuit Emulator (ICE-44) allows the user to exercise the 8044 application system and monitor the execution of instructions in real time.

The emulator operates with Intel's Intellec® development system. The development system interfaces with the user's 8044 system through an in-cable buffer box. The cable terminates in a 8044 pin-compatible plug, which fits into the 8044 socket in the user's system.

With the emulator plug in place, the user can exercise his system in real time while collecting up to 255 instruction cycles of real-time data. In addition, he can single-step the program.

Static RAM is available (in the in-cable buffer box) to emulate the 8044 internal and external program memory and external data memory. The designer can display and alter the contents of the replacement memory in the buffer box, the internal data memory, and the internal 8044 registers, including the SFRs.

SIUST: SIU State Counter (byte-addressable)

The SIU State Counter (Address D9H) reflects the state of the internal logic which is under SIU control. Therefore, care must be taken not to write into this register. This register provides a useful means for debugging 8044 receiver problems.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . 0 to 70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin With
Respect to Ground (Vss) . . . . . −0.5V to +7V
Power Dissipation . . . . . 2 Watts

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5V ± 10%, V_{SS} = 0V)

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
VIL	Input Low Voltage	−0.5		0.8	V	
VIH	Input High Voltage (Except RST/VPD and XTAL2)	2.0		VCC + 0.5	V	
VIH1	Input High Voltage To RST/VPD For Reset, XTAL2	2.5			V	XTAL1 to VSS
VPD	Power Down Voltage To RST/VPD	4.5		5.5	V	VCC = 0V
VOL	Output Low Voltage Ports 1, 2, 3 (Note 1)			0.45	V	IOL = 1.6mA
VOL1	Output Low Voltage Port 0 ALE, \PSEN (Note 1)			0.45	V	IOL = 3.2mA
VOH	Outpt High Voltage Ports 1, 2, 3	2.4			V	IOH = −80μA
VOH1	Output High Voltage Port 0, ALE, \PSEN	2.4			V	IOH = −400μA
IIL	Logical 0 Input Current Ports 1, 2, 3			−800	μA	XTAL1 at VSS VIL = 0.45V
IIH1	Input High Current,T0 RST/VPD For Reset			500	μA	Vin = VCC − 1.5V
ILI	Input Leakage Current To Port 0, \EA			10	μA	0.45V < Vin < VCC
ICC	Power Supply Current		125	200	mA	TA = 25°C
IPD	Power Down Current		15	30	mA	
CIO	Capacitance of I/O Buffer			10	pF	fc = 1MHz
IIL2	Logical 0 Input Current XTAL 2			−2.5	mA	XTAL1 = VSS VIL = 0.45V

Note 1: VOL is degraded when the RUPI-44 rapidly discharges external capacitance. This A.C. noise is most pronounced during emission of address data. When using external memory, locate the latch or buffer as close to the RUPI-44 as possible.

Datum	Emitting Ports	Degraded I/O Lines	VOL (peak) (max)
Address	P2, P0	P1, P3	.8V
Write Data	P0	P1, P3, ALE	.8V

A.C. CHARACTERISTICS (T_A 0°C to 70°C, V_{CC} = 5V ± 10% V_{SS} = 0V, C_L for Port 0, ALE and $\overline{PSEN}$ Outputs = 100pF; C_L for All Other Outputs = 80 pF)

Program Memory

Symbol	Parameter	12 MHz Clock			Variable Clock 1/TCLCL = 1.2 MHz to 12 MHz		
		Min	Max	Units	Min	Max	Units
TLHLL	ALE Pulse Width	127		ns	2TCLCL-40		ns
TAVLL	Address Setup to ALE	43		ns	TCLCL-30		ns
TLLAX[1]	Address Hold After ALE	48		ns	TCLCL-35		ns
TLLIV	ALE To Valid Instr In		233	ns		4TCLCL-100	ns
TLLPL	ALE To $\overline{PSEN}$	58		ns	TCLCL-25		ns
TPLPH	$\overline{PSEN}$ Pulse Width	215		ns	3TCLCL-35		ns
TPLIV	$\overline{PSEN}$ To Valid Instr In		125	ns		3TCLCL-125	ns
TPXIX	Input Instr Hold After $\overline{PSEN}$	0		ns	0		ns
TPXIZ[2]	Input Instr Float After $\overline{PSEN}$		63	ns		TCLCL-20	ns
TPXAV[2]	Address Valid After $\overline{PSEN}$	75		ns	TCLCL-8		ns
TAVIV	Address To Valid Instr In		302	ns		5TCLCL-115	ns
TAZPL	Address Float To $\overline{PSEN}$	0		ns	0		ns

Notes:
1. TLLAX for access to program memory is different from TLLAX for data memory.
2. Interfacing RUPI-44 devices with float times up to 75ns is permissible. This limited bus contention will not cause any damage to Port 0 drivers.

External Data Memory

Symbol	Parameter	12 MHz Clock			Variable Clock 1/TCLCL = 1.2 MHz to 12 MHz		
		Min	Max	Units	Min	Max	Units
TRLRH	$\overline{RD}$ Pulse Width	400		ns	6TCLCL-100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		ns	6TCLCL-100		ns
TLLAX[1]	Address Hold After ALE	48		ns	TCLCL-35		
TRLDV	$\overline{RD}$ To Valid Data In		250	ns		5TCLCL-165	ns
TRHDX	Data Hold After $\overline{RD}$	0		ns	0		ns
TRHDZ	Data Float After $\overline{RD}$		97	ns		2TCLCL-70	ns
TLLDV	ALE To Valid Data In		517	ns		8TCLCL-150	ns
TAVDV	Address To Valid Data In		585	ns		9TCLCL-165	ns
TLLWL	ALE To $\overline{WR}$ or $\overline{RD}$	200	300	ns	3TCLCL-50	3TCLCL + 50	ns
TAVWL	Address To $\overline{WR}$ or $\overline{RD}$	203		ns	4TCLCL-130		ns
TWHLH	$\overline{WR}$ or $\overline{RD}$ High To ALE High	43	123	ns	TCLCL-40	TCLCL + 40	ns
TDVWX	Data Valid To $\overline{WR}$ Transition	33		ns	TCLCL-50		ns
TQVWH	Data Setup Before $\overline{WR}$	433		ns	7TCLCL-150		ns
TWHQX	Data Hold After $\overline{WR}$	33		ns	TCLCL-50		ns
TRLAZ	Address Float After $\overline{RD}$		0	ns		0	ns

Note 1. TLLAX for access to program memory is different from TLLAX for access data memory.

Serial Interface

Symbol	Parameter	Min	Max	Units	
TDCY	Data Clock	420		ns	
TDCL	Data Clock Low	180		ns	
TDCH	Data Clock High	120		ns	

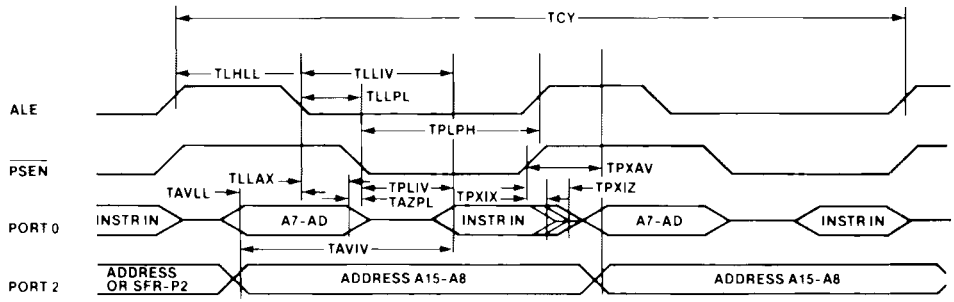
Serial Interface (Continued)

tTD	Transmit Data Delay		80	ns	
tDSS	Data Setup Time	60		ns	
tDHS	Data Hold Time	40		ns	

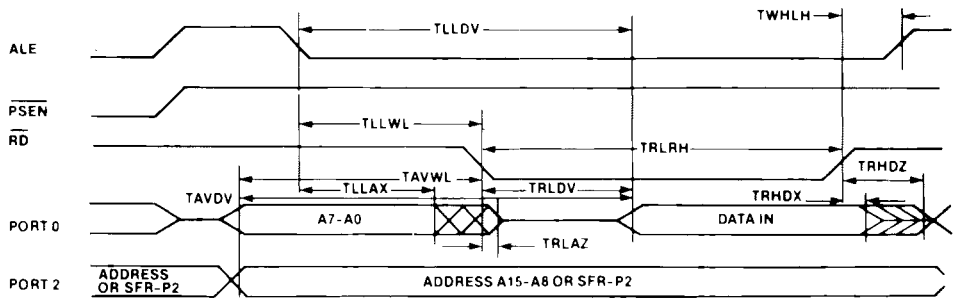
WAVEFORMS

Memory Access

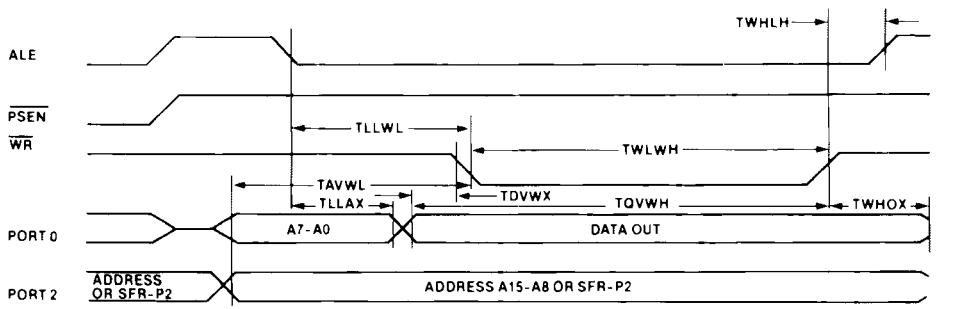
Program Memory Read Cycle



Data Memory Read Cycle

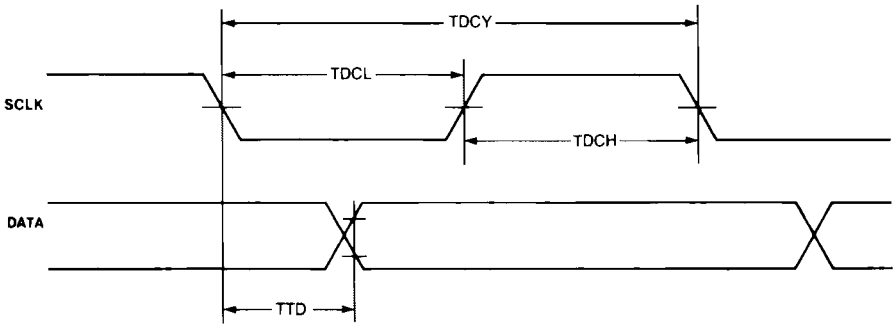


Data Memory Write Cycle

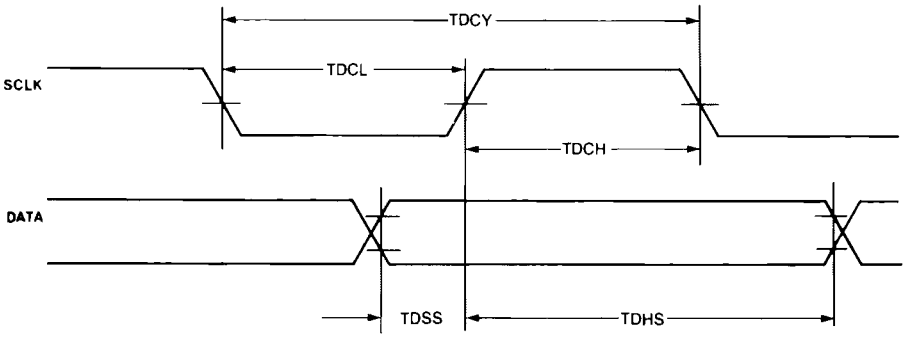


SERIAL I/O WAVEFORMS

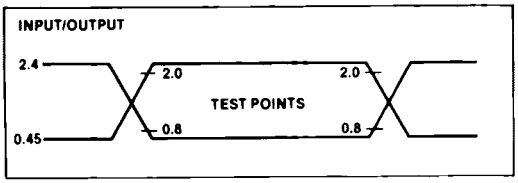
Synchronous Data Transmission

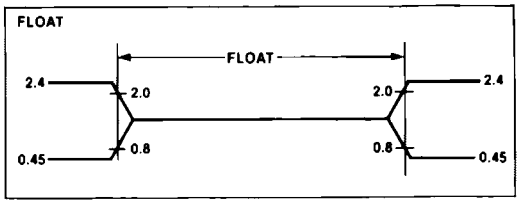


Synchronous Data Reception



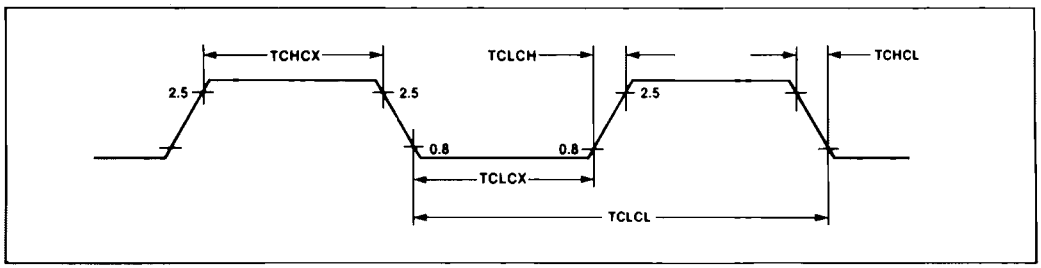
AC TESTING INPUT, OUTPUT, FLOAT WAVEFORMS





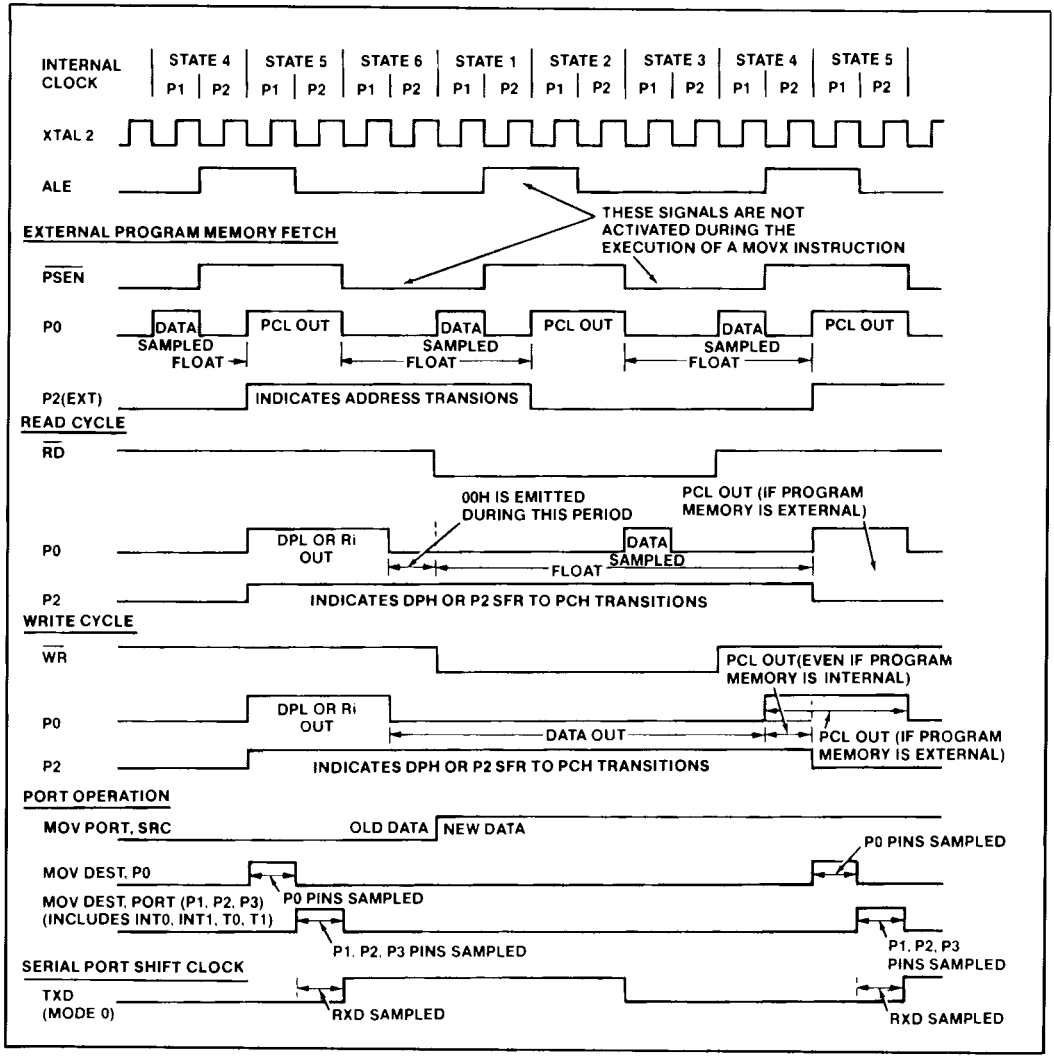
AC testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0."
Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0."

EXTERNAL CLOCK DRIVE XTAL2



Symbol	Parameter	Variable Clock Freq = 1.2 MHz to 12 MHz		Unit
		Min	Max	
TCLCL	Oscillator Period	83.3	833.3	ns
TCHCX	High Time	20	TCLCL-TCLCX	ns
TCLCX	Low Time	20	TCLCL-TCHCX	ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

CLOCK WAVEFORMS



This diagram indicates when signals are clocked internally. The time it takes the signals to propagate to the pins, however, ranges from 25 to 125 ns. This propagation delay is dependent on variables such as temperature and pin loading. Propagation also varies from output to output and component to component. Typically though, (T_A = 25°C, fully loaded) RD and WR propagation delays are approximately 50 ns. The other signals are typically 85 ns. Propagation delays are incorporated in the AC specifications.

THE BITBUS™ INTERCONNECT SERIAL CONTROL BUS SPECIFICATION

REV.	REVISION HISTORY	PRINT DATE
A	Original Issue.	1/84

1.0 GENERAL

1.1 SCOPE

Connection of dedicated controllers and process I/O points to a control computer has been a long time problem. Traditional methods have used a standard parallel bus to interconnect I/O expansion boards, which provide the required signal conditioning. In many cases, these buses are satisfactory; however, in many other cases the characteristics of these parallel buses create significant limitations. Electrically, parallel buses limit the number of boards that can be transparently (i.e. in the same backplane) added to the system. Mechanical packaging for parallel buses often makes I/O cabling cumbersome, due to the close proximity of boards within a backplane. Most importantly, the fixed form factor and bus interface logic requirements of a parallel bus put a lower bound on future system cost reduction, regardless of the advancing capabilities of future VLSI. The BITBUS™ interconnect is a serial control bus specifically designed to address these problems.

The BITBUS interconnect allows up to 250 nodes to be easily interconnected over a physically distributed domain. Distribution capability ranges from 30 meters for the synchronous mode of operation to thousands of meters for the self clocked mode of operation. The different modes of operation are optimized for a wide range of applications covering the spectrum from high speed servo motor control in robotics, to long distance environmental control. In all cases the BITBUS interconnect is optimized for the high speed transfer of short control messages in a hierarchical system.

The level of specification for the BITBUS interconnect is somewhat different than for traditional parallel bus structures. In the past, serial connections have been slow, software intensive and hard to use. A goal of the BITBUS interconnect is to provide an easy to use, high performance serial interconnect that is transparent to the application programmer. For this purpose, this specification not only defines electrical and data link protocol aspects of the bus, as is typical for parallel buses, but also specifies a message structure and protocol for a multitasking environment, and a set of high level commands for remote I/O access and application task control. This allows standard high level software interfaces to be written, off-loading the application programmer of this complication. More importantly, this high level of specification allows the standard interface to be driven into silicon, further reducing the interface overhead, cost and complexity.

This specification has been prepared for those users intending to design or evaluate products that will be compatible with the BITBUS interconnect. The intent of this specification is to specify the interface requirements only. The implementation of interface logic and software is left to the user.

1.2 DEFINITIONS

System — A set of interconnected elements which achieve a given objective through performing a specific function. In this specification a BITBUS system consists of one master node and from one to 250 slave nodes attached to the same BITBUS interconnect.

Message-Passing — The transfer and control of structured data between two tasks.

Task — An entity which competes for system resources in order to execute a program.

Protocol — The rules by which information is exchanged across an interface.

Operation — The process whereby information is transferred between two elements across an interface.

Interface — A shared boundary between two elements within a system.

Bit-Cell-Time — The time interval required to transfer one bit of data on a serial line.

2.0 FUNCTIONAL OVERVIEW

The BITBUS interconnect concept allows highly reliable, low cost distributed expansion of process I/O points and intelligent I/O controllers. This section provides an overall understanding of the concept by describing the hierarchical model, bus elements, modes of operation and levels of specification. Detailed specifications are provided in subsequent sections.

2.1 HIERARCHICAL MODEL

The BITBUS interconnect is defined to provide a high speed serial control bus for hierarchical systems. This model, shown in figure 1, implies that a single master must communicate with multiple slaves. These slaves may consist of simple I/O points or they may be highly intelligent controllers. By adopting this architectural model, the BITBUS interconnect is able to provide a highly capable, low cost solution, especially for those applications that match the model. This is achieved through a significant reduction in protocol complexity and overhead that would otherwise be necessary to implement a more complex multiple access type protocol. Evidence of this simplicity can be seen by comparing the level of protocol support provided in hardware for the BITBUS interconnect as compared to other more complex protocols.

The hierarchical model for the BITBUS interconnect may consist of one or multiple levels as shown in figures 1a and 1b respectively. The multiple level hierarchy is extremely useful in many applications, especially where BITBUS interconnects must run at different speeds. This specification provides the necessary hooks to implement such a structure with minimal overhead. In this specification a BITBUS system refers to only the single level hierarchy. The multiple level hierarchy consists of multiple BITBUS systems.

The goal of the BITBUS interconnect is to provide a message passing interface between tasks at the master node and tasks at multiple slave nodes within the hierarchical model. This is accomplished through an order/reply message protocol. Tasks on the master node issue orders to tasks on the slave nodes and tasks on the slave nodes respond with replies. This level of support effectively hides the serial nature of the BITBUS interconnect from the application programmer, resulting in a relatively simple interface.

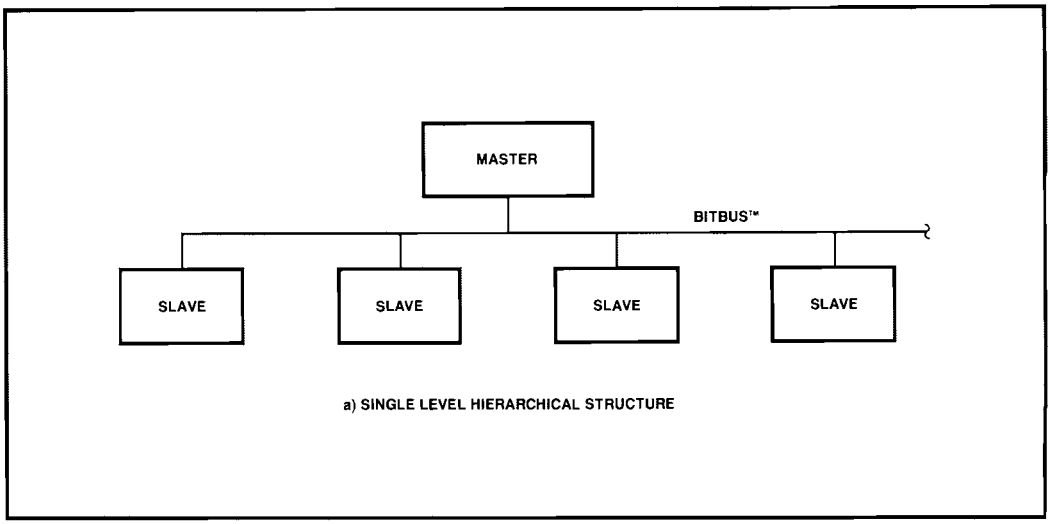


Figure 1a. BITBUS™ Interconnect Hierarchy

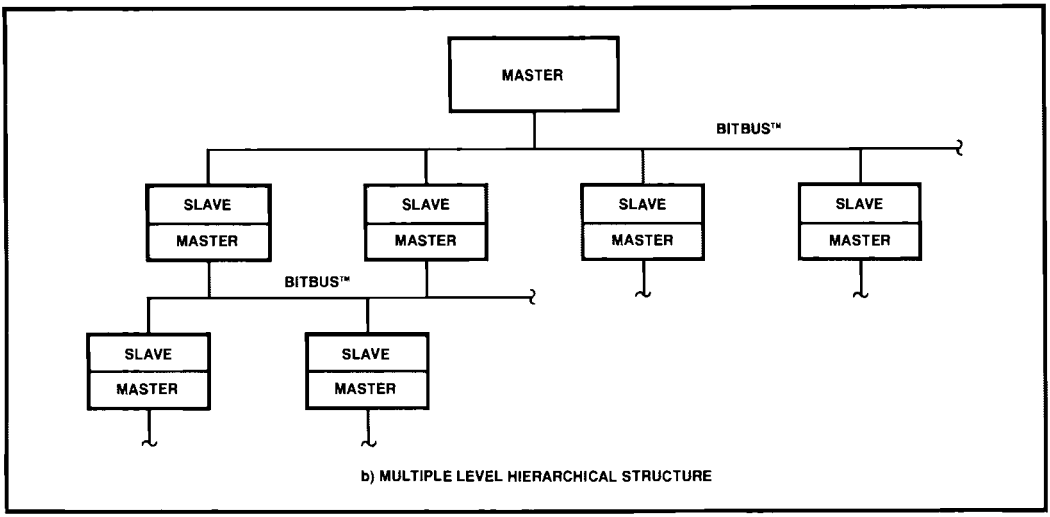


Figure 1b. BITBUS™ Interconnect Hierarchy

2.2 BUS ELEMENTS

The basic element in a BITBUS system is a node. A node may consist of either a device, a device and an extension or a repeater. There are two types of devices: the master device and a slave device. This section explains the unique elements of the BITBUS system: the master device, the slave device, the extension and the repeater. Figure 2 illustrates these elements.

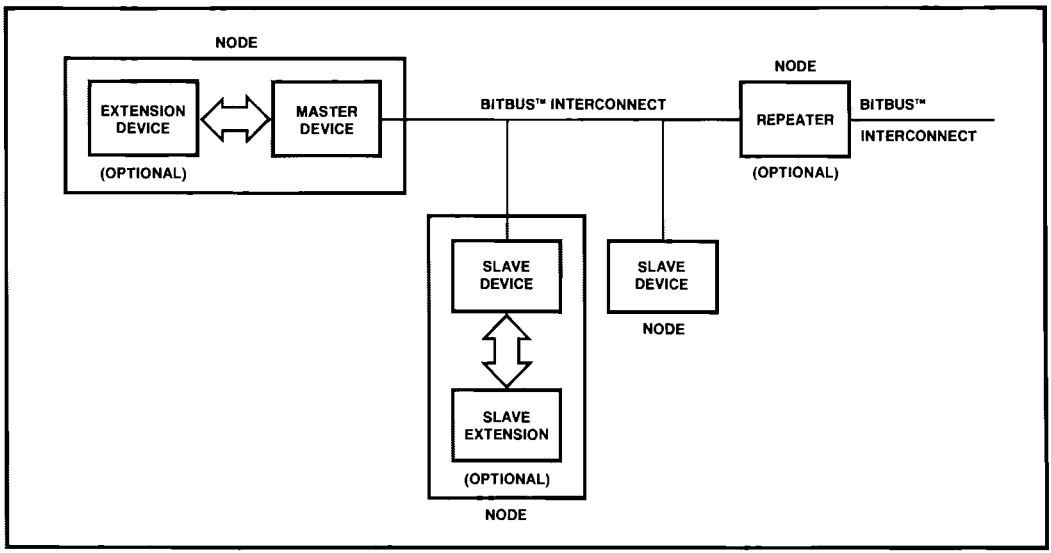


Figure 2. BITBUS™ Interconnect Elements

2.2.1 MASTER DEVICE

The master device controls all BITBUS interconnect operations. There is only one master device per BITBUS system. The master device initiates a transaction by sending an order message to a slave device. It then continues to poll the slave device until a reply message is returned.

During normal operation, each transfer from the master device is acknowledged by the addressed slave device. If a reply is not immediately available, a simple data link acknowledgement is returned. This frees the master device to perform other operations, such as sending another message or polling another slave device. In general, this scheme maximizes the available bus bandwidth within the BITBUS system.

The master device may be stationary in one node or this function may be passed between nodes. Mastership passing in a BITBUS system is performed via a scheme similar to token passing; however, it reduces system reliability (e.g. possibility of lost "tokens") and is therefore in general discouraged. On the other hand, despite the lower reliability, there may be specific cases where passing of mastership is desirable. One example is the implementation of a redundant master. In this case, it may be possible to increase to overall system mean time between failure MTBF, even with the reduced reliability of passing mastership. This is especially true if all potential masters can be isolated to a small section of cable isolated from the rest of the system by a repeater. Note that mastership may not be passed through repeaters.

2.2.2 SLAVE DEVICE

A slave device is a replier in a BITBUS system. There may be up to 250 slaves per system (addresses 1-250). As a replier, the slave may not spontaneously initiate a transfer. Rather, it may only respond to orders from the master device with replies. This approach significantly simplifies the slave device protocol requirements allowing powerful, low cost solutions to be implemented using highly integrated microcontrollers.

2.2.3 EXTENSION

An extension is a secondary processor within a node. The interface between a master or slave device and its extension is not part of this specification. The reason for including the extension in this description is to identify the need for a control field within the standard BITBUS message format to efficiently route messages to a master or slave device or their respective extensions. This allows low cost nodes to be constructed with a single processor and high capability nodes to be constructed with an application processor (extension) and a BITBUS interface processor. By differentiating tasks that run on these two processors, the BITBUS interface processor may contain tasks to off-load the extension processor while the extension processor maintains the capability to send messages directly through to the BITBUS interconnect. This capability may also be used to implement gateways between multiple BITBUS systems creating the multiple level hierarchy shown in Figure 1b.

2.2.4 REPEATERS

A repeater is a node used to regenerate (not reclock) the BITBUS interface signals. Repeaters are used to extend the distance or node count within a BITBUS system. Repeaters are referred to as nodes since they load a segment just as a master or slave node does. Repeaters are only allowed in the self clocked mode of operation. Section 2.3.2 provides more details on repeater operation.

2.3 MODES OF OPERATION

The BITBUS interconnect may operate in one of two modes: synchronous mode and self clocked mode. The purpose for multiple modes is to provide a wide range of performance/distance options for a variety of applications. In this section each mode is presented with its corresponding features and signal line definitions. Note that in both modes of operation all BITBUS interconnect signals are differential. The terminology used to specify the signal pairs is NAME and NAME*. State definition for the signal pairs is provided in section 3.1.1 of this specification.

2.3.1 SYNCHRONOUS MODE

The synchronous mode of operation is optimized for the highest performance over a relatively short distance. This mode can interconnect up to 28 nodes over 30 meters at transmission speeds between 500 kbits/sec and 2.4 mbits/sec. A typical system is shown in figure 3.

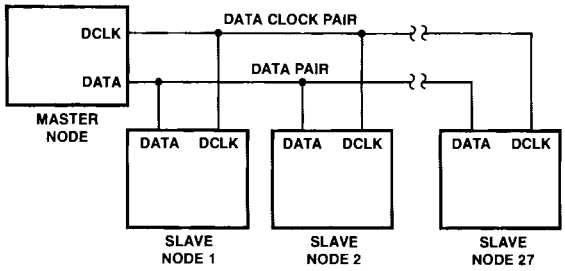


Figure 3. Synchronous Mode Connection

The synchronous mode uses two differential signal pairs: one for data (DATA, DATA*) and one for the data clock (DCLK, DCLK*). The data signal pair carries data which is referenced to the data clock signal pair. Data changes on the "falling edge" of the data clock pair and is sampled on the "rising edge". A sample interface circuit is shown in figure 4. Note that the data clock source is always at the transmitting node.

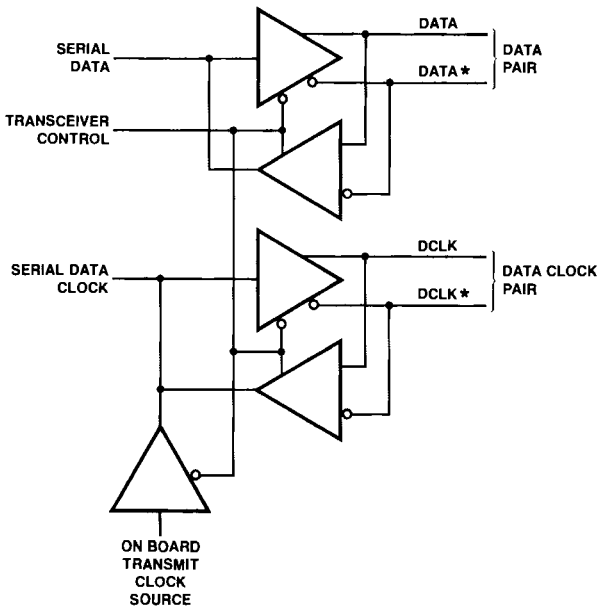


Figure 4. Typical Synchronous Mode Interface

2.3.2 SELF CLOCKED MODE

The self clocked mode of operation is provided to allow longer distance operation. There are two standard bit rates for this mode of operation: 375 kbits/sec and 62.5 kbits/sec. The BITBUS interconnect supports segments up to 300 meters in length at 375 kbits/sec and 1200 meters in length at 62.5 kbits/sec. Each segment supports up to 28 nodes. By connecting segments via repeaters, this mode supports up to 250 nodes over several thousand meters. A typical system is shown in figure 5.

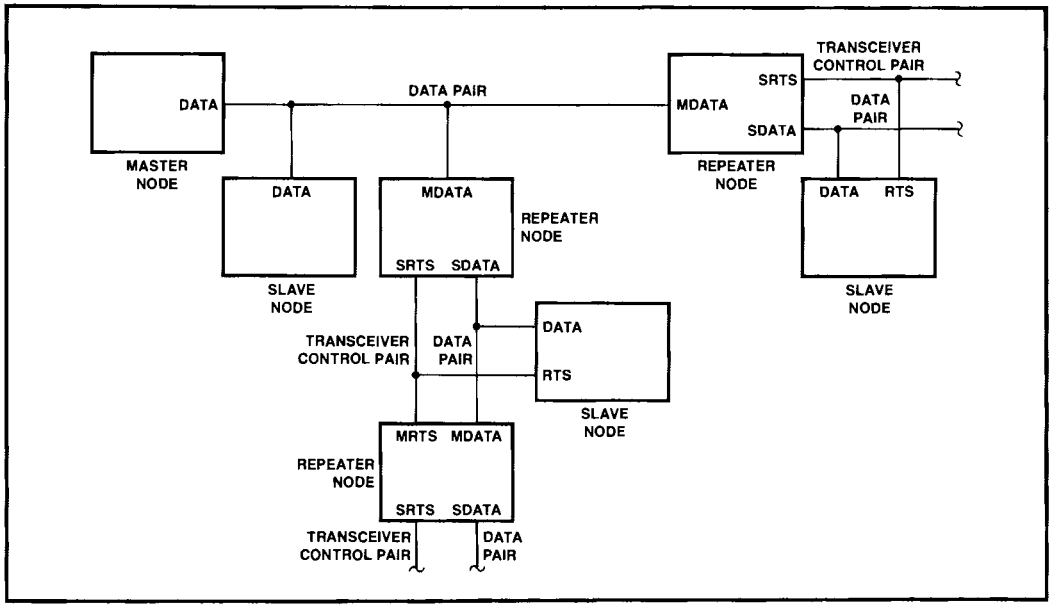


Figure 5. Self Clocked Mode Connection

The self clocked mode uses two differential signal pairs: one for data (DATA, DATA*) and one for transceiver control (RTS, RTS*). The data signal lines carry Non-Return to Zero Inverted (NRZI) encoded data. This encoding method combines clock and data onto the same signal pair. The transceiver control signal pair is used to control transceivers within the repeaters. When repeaters are not used, the tranceiver control pair may be omitted. A sample interface circuit is shown in figure 6.

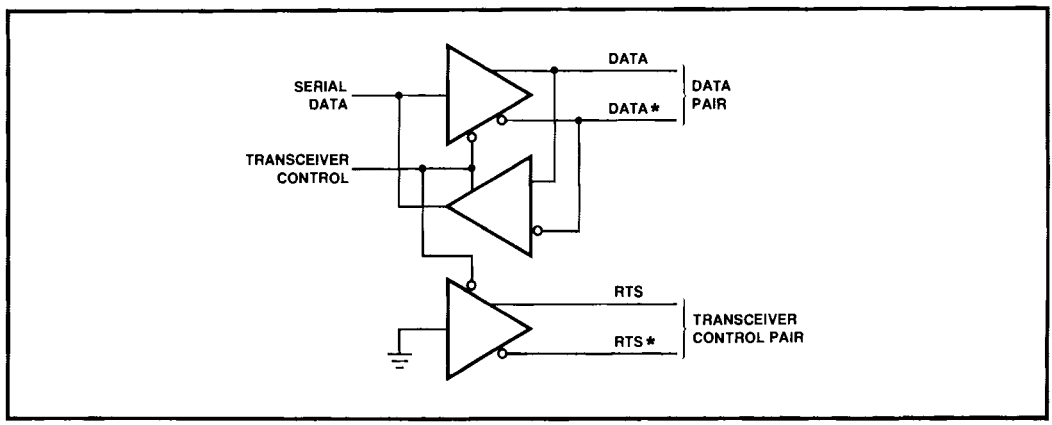


Figure 6. Typical Self Clocked Mode Interface

A BITBUS repeater may or may not provide electrical isolation, depending on the application requirements. In either case, when a slave device is not transmitting, biasing resistors enable the repeater tranceivers away from the master device, allowing the master device to transmit to all slaves. When a slave device responds, it reverses the polarity of the transceiver control pair, which reverses the direction of all repeaters between it and the master device. Figure 7 shows a sample repeater circuit.

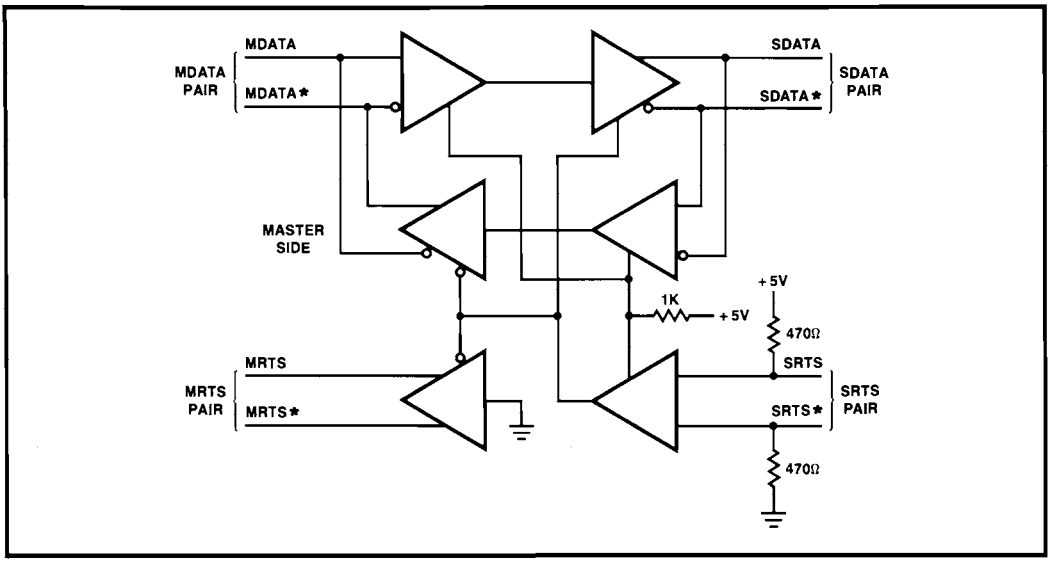


Figure 7. Typical Repeater

2.4 LEVELS OF SPECIFICATION

The BITBUS interconnect is designed to provide a highly capable, easy to use interface to the application programmer. In order to be easy to use, this interface must hide the complexity of controlling the serial link from the user through a standard hardware/software interface. In order to be highly capable, the BITBUS interconnect must be defined at a high enough level to allow this hardware and software interface to be efficient, and to a large degree be driven into hardware. These requirements result in the need for the following levels of specification:

Electrical Interface
Data Link Protocol
Message Protocol
Remote Access and Control
Mechanical

2.4.1 ELECTRICAL INTERFACE

The BITBUS electrical interface specification defines the mechanisms by which bits are transferred. This section of the specification is simplified by the fact that the BITBUS interconnect consists of only two signal pairs and the electrical interface is based to a large degree on existing standards. This level of the specification is optimized for low cost implementations (i.e. low cost cable and low cost transceivers).

2.4.2 DATA LINK PROTOCOL

The BITBUS data link protocol specification defines the mechanisms by which packets are reliably transferred across the electrical interface. Reliability aspects include error detection, and in many

cases error recovery. This is important to the BITBUS interconnect definition since it makes the application less sensitive to the serial nature of the interconnect. This section of the specification is also based to a large degree on existing standards.

2.4.3 MESSAGE PROTOCOL

The BITBUS message protocol specification defines the mechanism by which tasks pass messages over the data link. The protocol is optimized for communication between tasks on the master node and tasks on slave nodes. In actual implementations this same protocol can be used to communicate between tasks on the same node as well, allowing the serial bus to appear transparent. The purpose for providing this level in the bus specification is to allow a standard high level user interface to be efficiently implemented, and eventually driven into silicon. This section of the specification is based on a simple high performance protocol, using a well established order/reply mechanism.

2.4.4 REMOTE ACCESS AND CONTROL

The remote access and control (RAC) specification defines a set of high level commands to perform a set of standard operations over the BITBUS interconnect. Remote access commands allow standard I/O operations (e.g. read, write, etc.) to be performed on up to 256 I/O ports at each slave node without application software. For higher capability systems remote control commands allow slave node tasks to be downloaded, uploaded and controlled from the master device.

2.4.5 MECHANICAL

The BITBUS mechanical specifications are minimal. The purpose for these specifications is to define standard connectors for the BITBUS interconnect and an optional I/O board form factor for higher levels of compatibility.

3.0 ELECTRICAL SPECIFICATION

The BITBUS interconnect electrical specification requires definition of data encoding techniques and DC/AC parameters for the interface logic and interconnect cable. This section provides the required definition and specifically points out where existing standards are used as a basis for the specification.

3.1 DATA ENCODING TECHNIQUES

The BITBUS interconnect uses different data encoding techniques for the two modes of operation. Each is discussed separately below, along with a general definition of signal pair state.

3.1.1 GENERAL SIGNAL PAIR STATE DEFINITION

All signals on the BITBUS interconnect are differential pairs. The naming convention for these pairs identifies the two signal lines as NAME and NAME*. An active signal (logical 1) is present when the NAME signal line is at a higher electrical potential than the NAME* signal line. An inactive signal (logical 0) is present when the NAME signal line is at a lower electrical potential than the NAME* signal line.

3.1.2 SYNCHRONOUS MODE

Data encoding is relatively simple in the synchronous mode. The logical value of each bit is simply indicated by the level of the data signal pair at the bit cell center. The bit cell center is defined as the rising edge of the data clock pair. A logical zero is represented by the DATA signal being at a lower electrical potential than the DATA* signal at the bit cell center. A logical one is represented by the DATA signal being at a higher electrical potential than the DATA* signal at the bit cell center. These relationships are shown in figure 8. Data encoding for the synchronous mode of operation

must also provide zero bit insertion/deletion. This is required to support the frame delimiting flags defined in section 4 of this specification. The flags are defined as the bit pattern 01111110. In order to guarantee uniqueness of this bit pattern, there may be no more than five consecutive ones in the bit stream (other than flags). This is accomplished using zero bit insertion (by the transmitter) and deletion (by the receiver). Specifically, the transmitter inserts a zero into the bit stream anytime it detects five consecutive ones (except for flags) regardless of the next bit value. Receivers then remove any zero from the bit stream that is preceded by five ones. The receiver must also detect the 01111110 bit pattern as a frame delimiting flag.

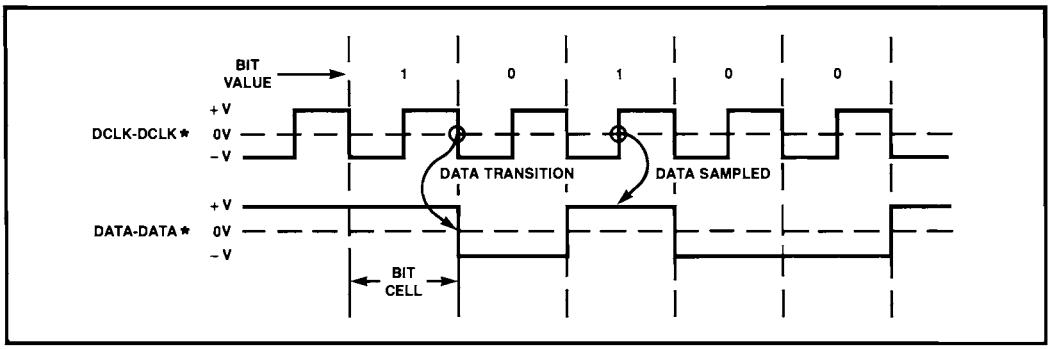


Figure 8. Synchronous Mode Data Encoding

3.1.3 SELF CLOCKED MODE

The self clocked mode of operation uses standard NRZI encoding with zero bit insertion/deletion. This encoding method combines serial data and serial clock information onto a single signal pair (DATA, DATA*). A logical zero is represented by a change in the polarity of the data signal pair from the proceeding bit cell (i.e. a transition occurs on the bit cell boundary). A logical one is represented by no change in the polarity of the data signal pair from the preceding bit cell. These relationships are shown in figure 9. As in the synchronous mode, zero bit insertion/deletion is provided to guarantee uniqueness of the frame delimiting flag pattern.

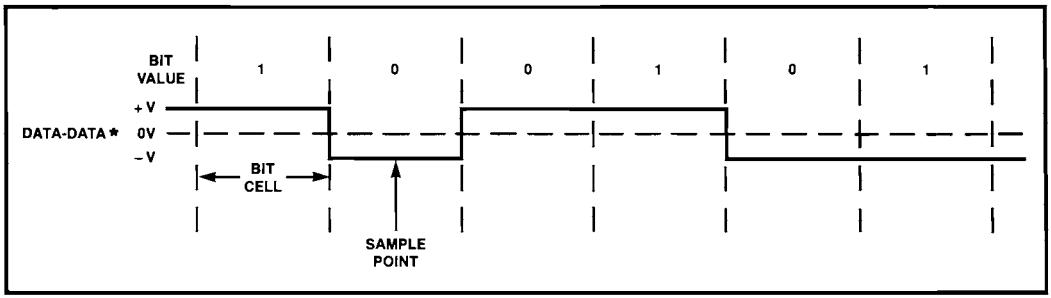


Figure 9. Self Clocked Mode Data Encoding

The self clocked mode of operation requires that receivers recover the serial clock from the data bit stream. The BITBUS interconnect is defined with the assumption that a digital phase locked loop (DPLL) with a 16× reference clock will be used for this purpose. For proper operation, the DPLL must be synchronized (i.e. be in phase with the transmitter clock) before, and remain synchronized during the transmission of a frame. Initial synchronization of the receivers DPLL is guaranteed by the transmission of a preframe sync (PFS) prior to the frame. The PFS consists of a minimum of eight zeros (i.e. transitions) that allow the DPLL to adjust (one reference clock at a time) until synchronized.

Once synchronized, the DPLL uses the transitions (i.e. zeros) within the bit stream to make fine adjustments of ±1 reference clock cycle. Note that zero bit insertion/deletion guarantees a transition at least every seven bit cells, allowing a reasonable tolerance on the reference clock.

3.2 DC SPECIFICATIONS

The DC specifications for the BITBUS interconnect are based on the RS485 electrical standard. This section defines the characteristics of a standard load, a transmitter, a receiver, the interconnect cable and the terminating and biasing resistors.

3.2.1 STANDARD LOAD SPECIFICATION

The standard unit load specification is used as a basis to define the load presented by a node to the BITBUS interconnect. Figure 10 shows the I/V (current verses voltage) characteristics of the standard unit load. Note that this is 1.125 RS485 unit loads and it is specified over the input voltage range of +12 volts to −7 volts.

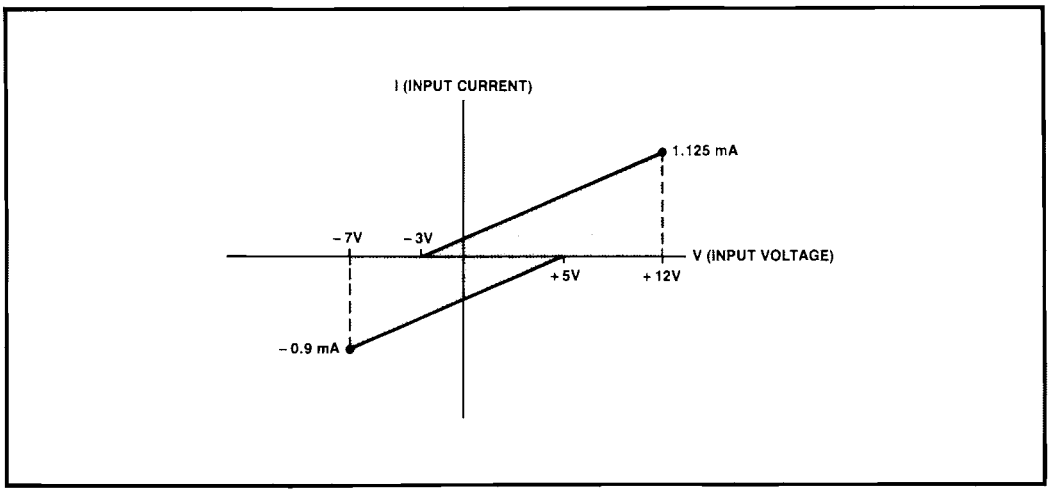


Figure 10. Standard Unit Data

The shaded region in figure 10 indicates the acceptable region for the I/V characteristic representing one standard unit load. Actual implementations may present a load which is a multiple of or fraction of this standard. The actual load specification of a connection is determined by plotting its I/V characteristics, and then scaling the I axis (i.e. multiply I axis by scaling factor) until the I/V characteristic is contained within the shaded region and touching the boundary. The factor used to scale the I axis is the number of standard unit loads.

The standard unit load is used to define the BITBUS interconnect load. The following sections specify how the standard unit load is applied to the various signal pairs, and how interface loads are computed for the synchronous and self clocked modes of operation.

3.2.1.1 Data And Data Clock Pair Specification

The load for the data pair (DATA, DATA*) and data clock pair (DCLK, DCLK*) consist of a disabled transmitter and a receiver as shown in figures 4, 6 and 7. The DC characteristics of these loads are specified as a multiple of the standard unit load shown in figure 10. The standard unit load specification has been defined such that most implementations with standard off the shelf components will be one standard unit load.

3.2.1.2 Transceiver Control Pair Specification

The transceiver control pair (RTS, RTS*) is not symmetrical (i.e. there is not a transmitter and receiver at each node) like the data and data clock pairs. Rather, this pair has one "receiver only" node (in the repeater as shown in figure 7) and multiple "transmitter only" nodes (as shown in figures 6 and 7). The DC characteristics of the transmitter only loads are specified as a multiple of 1/9 the standard unit load shown in figure 10. For example, if a node presents 2/9 standard unit loads on this pair it is counted as 2 BITBUS interconnect loads. The DC characteristics of the receiver only node shall not exceed 25 standard unit loads, including the biasing resistors specified in section 3.2.5. With this definition, 27 receiver only nodes and one transmitter only node are equivalent to 28 standard unit loads ($27 \times 1/9 + 25 = 28$).

3.2.1.3 Synchronous Mode Load

The load presented to the BITBUS interconnect in synchronous mode is defined by determining the number of standard unit loads for each signal pair. The largest number is the specification. For example if the data signal pair presents a load equal to the standard unit load, and the data clock signal pair presents a load equal to 2 standard unit loads, the interface would be specified as 2 loads.

3.2.1.4 Self Clocked Mode Load

The load presented to the BITBUS interconnect in the self clocked mode is also defined by determining the number of standard unit loads for each signal pair. For example if the data signal pair presents a load equal to the standard unit load, and the transceiver control pair presents a load equal to 2/9 standard unit loads, the interface would be specified as 2 loads. If the tranceiver control pair were not used (i.e. no repeater connections) the interface would be specified as 1 load.

3.2.2 TRANSMITTER SPECIFICATION

The standard BITBUS transmitter shall meet the requirements of an RS485 generator. It shall be capable of driving a 60 ohm termination (120 ohms at each end of the cable) and up to 32 RS485 unit loads (28 standard unit loads in Figure 10). For reference, the DC requirements are repeated in this specification. For the figures in this section all output voltages apply to both logic states and the symbol Δ in front of a voltage refers to the difference between that voltage in the two logic states.

3.2.2.1 Open Circuit Specification

The transmitter open circuit test configuration is shown in figure 11. All relevant test parameters are shown in the figure. This test is used to verify the output voltage characteristics when no load is applied.

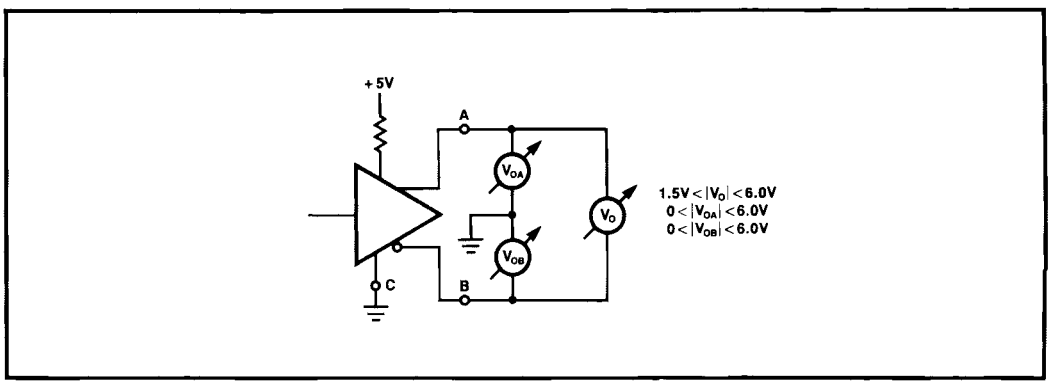


Figure 11. Transmitter Open Circuit Test Configuration

3.2.2.2 Test Termination Specification

The transmitter shall be able to drive the two test configurations shown in figures 12 and 13. All relevant test parameters are shown in the figures.

In figure 12 the transmitter drives a 54 ohm test load consisting of two 27 ohm resistors in series. This test termination is used to verify the output drive (V_O) capability and the signal pair offset voltage (V_{OS}) characteristics.

In figure 13 the transmitter drives a 60 ohm termination plus a load that simulates 32 RS485 unit loads through the RS485 common mode voltage range of +12 to −7 volts. This test termination is used to verify output drive capability under worst case command mode conditions.

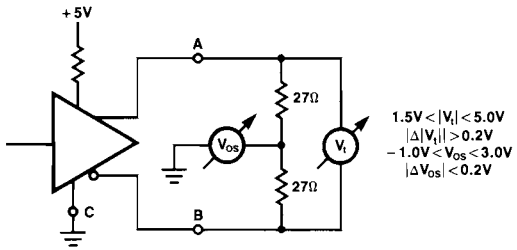


Figure 12. Transmitter Test Termination Configuration 1

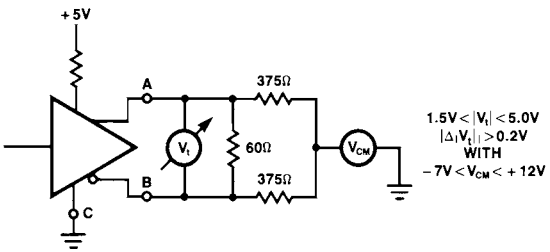


Figure 13. Transmitter Test Termination Configuration 2

3.2.2.3 Fault Conditions Specification

The transmitter shall be able to withstand, without permanent damage, the following two conditions:

a) Outputs shorted together.

b) Output(s) directly connected to a current limited voltage source within the +12 to −7 volt range.

Figure 14 shows the test configurations.

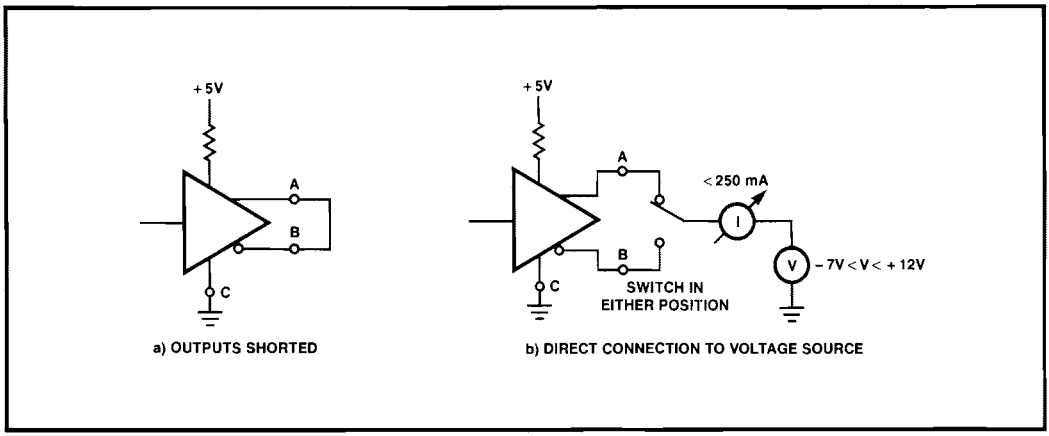


Figure 14. Transmitter Fault Conditions

3.2.3 RECEIVER SPECIFICATION

The standard BITBUS receiver shall meet the requirements of an RS485 receiver. Receiver load considerations are included in section 3.2.1. For reference, the input sensitivity and balance requirements are repeated in this section.

3.2.3.1 Sensitivity Specification

The receiver sensitivity specification is shown in figure 15. The allowable range of input voltages appearing at the receiver inputs, referenced to the receiver ground, shall be between + 12 and − 7 volts. Within this range a differential voltage of 200 millivolts or more shall be detectable as a valid logic state. The states are indicated in figure 15.

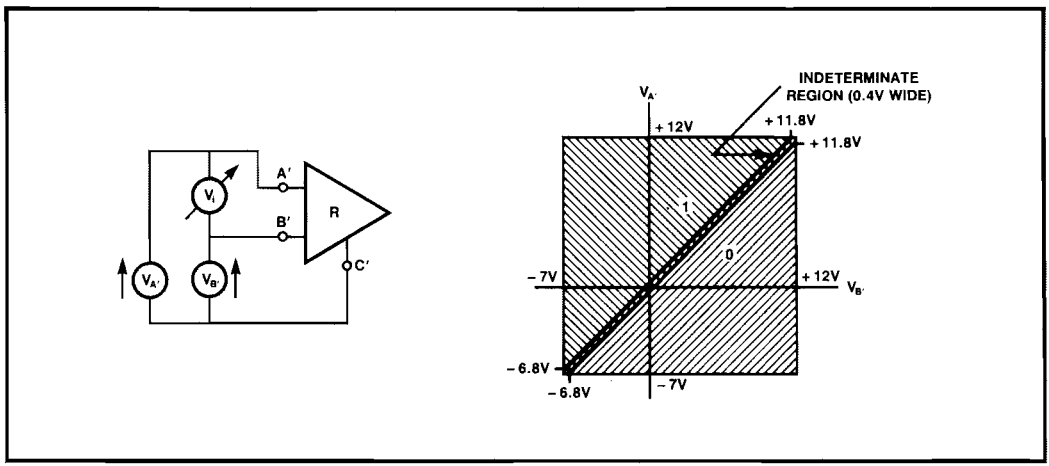


Figure 15. Receiver Sensitivity Specification

3.2.3.2 Balance Specification

The receiver balance specification is shown in figure 16. For this test the input resistor shown is 1500 ohms divided by the number of RS485 unit loads of the receiver. Within the common mode input range of + 12 to − 7 volts, a differential input of 400 millivolts or more shall be detectable as a valid logic state. The states are indicated in figure 16.

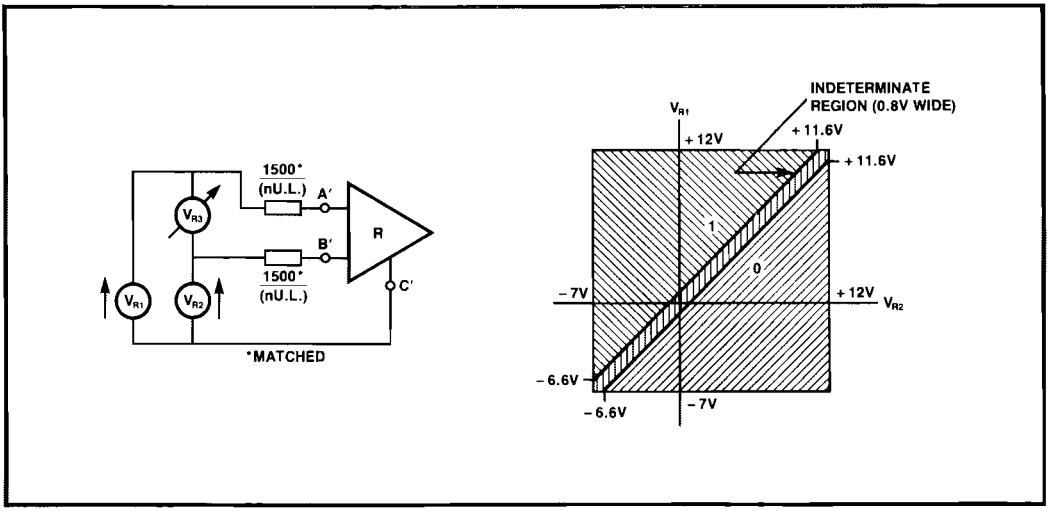


Figure 16. Receiver Balance Specification

3.2.4 CABLE CHARACTERISTICS

The BITBUS is defined to operate on a variety of cables including a low cost twisted pair (shielded or unshielded depending on application requirements) or backplane traces. Specific cable characteristics are not rigidly specified in order to allow flexibility for various implementations (i.e. only the interfaces are rigidly specified). When selecting a cable, the following aspects must be considered. Generally these parameters can be found in the cable data sheet.

- Characteristic Impedance
- Attenuation
- Resistance

3.2.4.1 Characteristic Impedance

Cables should be chosen with a characteristic impedance of 120 ohms or more to allow for properly matched terminations. Cables with lower characteristic impedance may be used in systems that guarantee low enough attenuation to prevent the reflections caused by the mismatch from switching the logic state. In severe environments, such as backplanes, greater mismatch can be tolerated as long as the maximum round trip propagation delay is less than one-half the signal rise time specified in section 3.3.1.1.

3.2.4.2 Attenuation

Cable must be chosen to support the desired bit rates over the desired distance. The specified BITBUS interconnect distances are based on the characteristics of readily available low cost twisted pair cables.

3.2.4.3 Resistance

Cable must be chosen with a low enough DC resistance to guarantee sufficient voltage across the termination to be detected by receivers. This parameter, the attenuation and the magnitude of reflections need to be considered together to determine the worst case noise margin at the receivers.

3.2.5 TERMINATION AND BIASING

The BITBUS interconnect requires the cable to be terminated for proper operation. There are two cases to be considered: the signal pair termination and the transceiver control pair biasing.

3.2.5.1 Termination

All BITBUS interconnect cables must be terminated at both ends for proper operation. The terminations shall be located at the extreme ends of the cable. The value of each termination shall be 120 ohms or greater and should be chosen to match the characteristic impedance of the cable as closely as possible.

3.2.5.2 Transceiver Control Biasing

In addition to terminations, the transceiver control pair shall have biasing resistors at its receiver only node. These resistors shall be 470 ohms ±5%, one being connected to +5 volts ±5% and the other connected to ground as shown in figure 7. The receiver only node may be located anywhere on the cable segment.

3.3 AC SPECIFICATIONS

The AC specifications for the BITBUS interconnect requires definition of the signal line characteristics, transmitter enable timing, self clocked mode timing, and repeater timing.

3.3.1 SIGNAL LINE CHARACTERISTICS

The BITBUS signal lines must maintain a reasonable level of signal integrity to guarantee proper operation. Specifically, there needs to be bounded rise and fall times and reflection guidelines.

3.3.1.1 Rise And Fall Time Specification

The BITBUS signal lines shall have rise and fall times between 25 and 100ns as shown in figure 17. This measurement shall be made in the test configuration shown in figure 18.

In actual systems, the rise and fall times may be slower than the above specification as long as the following two conditions are met.

a) The rise and fall time of any node shall meet the above specification (Figure 17) into the test load (Figure 18).
b) The rise and fall times in the actual system shall not be less than 0.3 times the bit cell width, measured anywhere in the system.

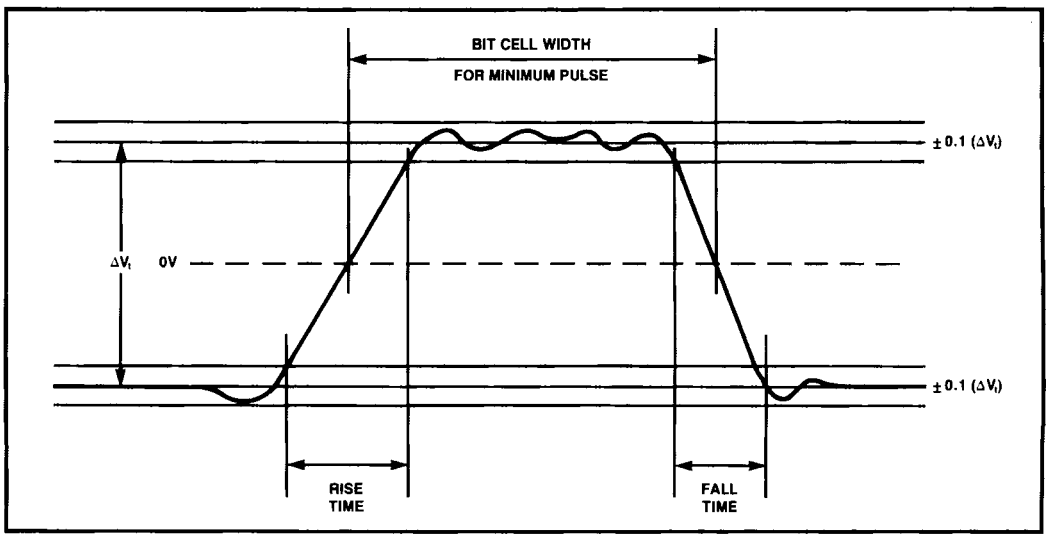


Figure 17. Rise and Fall Specification

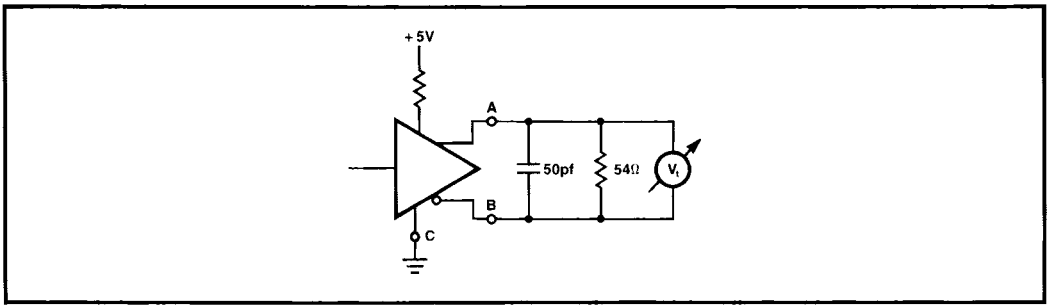


Figure 18. Test Configuration for Rise and Fall Time Measurement

3.3.1.2 Reflection Guidelines

Transitions on the BITBUS signal lines shall meet the requirements shown in figure 17 when driving the test load in figure 18. The transition from 10% to 90% of the final steady state value shall be monotonic. Ringing shall be limited to ±10% around the final steady state value.

RS485 is a robust standard, providing a significant DC noise margin. In practical systems where reflections may be unavoidable these noise margins shall be used to guarantee proper operation. Potential reflections (due to lumped capacitive loads, mismatched terminations, or excessively long stubs) should be considered together with the DC losses specified in section 3.2.4 when choosing cable. In all cases, the voltage at any node shall remain in the +12 volt to −7 volt common mode range except for pulses of less than 15 microseconds, at less than 1% duty cycle, which shall be bounded to ±25 volts.

3.3.2 TRANSMITTER ENABLE TIMING

The BITBUS interconnect is designed to operate without transceiver control signals (repeaters accepted). In order to guarantee proper operation, this requires specifications for transmitter turn on and turn off in order to prevent a "dribbling" (enabled after end of frame) transmitter from corrupting the data of another transmitter. Figure 19 summarizes these parameters.

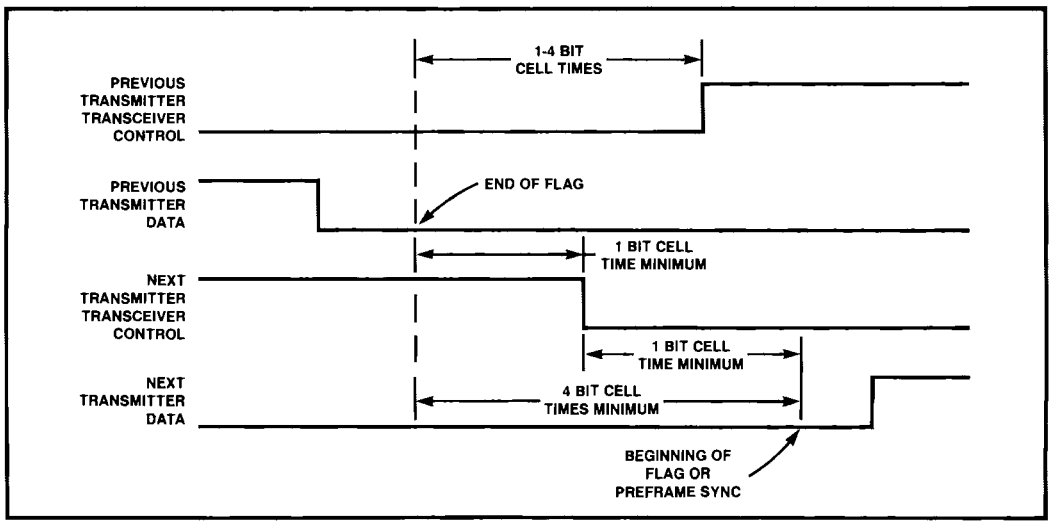


Figure 19. Transmitter Enable Timing

Transmitters shall be enabled a least one bit cell time prior to valid data transmission (opening flag in synchronous mode, preframe synch in self clocked mode). A transmitter shall not be enabled until at least one bit cell time after detecting the closing flag of a previous frame. Finally a transmitter shall guarantee that the first valid bit of data is not transmitted until any previous transmitter has been disabled.

Transmitters shall be disabled between one and four bit cell times after the end of the closing flag. Note that this specification leads to a potential three bit cell time contention period during turn around. This is acceptable based on the fault condition specification of section 3.2.2.3.

3.3.3 SYNCHRONOUS MODE TIMING

The synchronous mode of operation requires specification of the data clock signal pair timing and the data pair signal timing.

The data clock signal pair timing is shown in figure 20. All receivers shall be able to receive at speeds up to 2.4 Mbits/sec. Transmitters may transmit between 500 kbits/sec and 2.4 Mbits/sec.

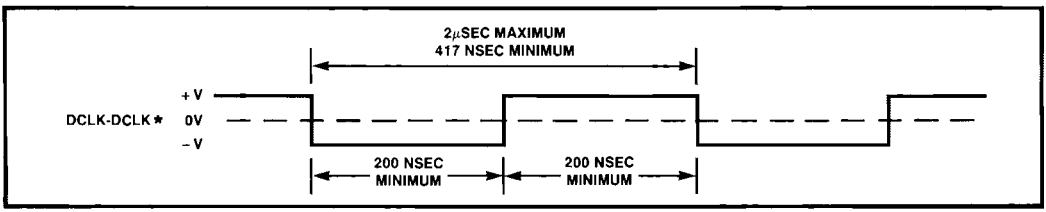


Figure 20. Data Clock Signal Pair Specification for Synchronous Mode

The data signal pair timing is shown in figure 21. The data signal pair is specified with respect to clock edges on the data clock signal pair. Data is changed on the "falling edge" and sampled on the "rising edge" of the data clock pair. These specifications assume balanced delays throughout the system. Specifically, the transmitters for the two signal pairs shall be in the same piece of silicon, the receivers for the two signal pairs shall be in the same piece of silicon and the conductors for the two signal pairs shall be of the same type, the same length and equally loaded.

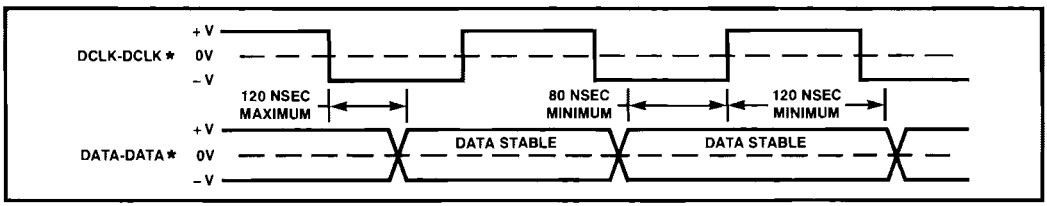


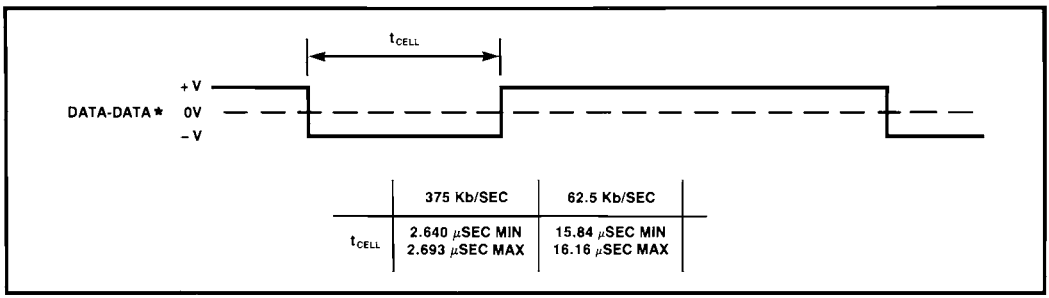
Figure 21. Data Signal Pair Specification for Synchronous Mode

3.3.4 SELF CLOCKED MODE TIMING

The self clocked mode of operation requires specification of the data signal pair and the transceiver control signal pair.

The data signal pair combines both data and clock information in the self clocked mode of operation. The clock information on the signal pair is the transmitter's clock. The receiver uses a separate reference clock to recover the data. The tolerance of each of these clocks shall be ± 1% for both standard bit rates (375 kbit/sec and 62.5 kbits/sec). The transmitter timing is shown in figure 22.

The transceiver control signal pair shall be driven whenever the transmitter is driven. These specifications are provided in section 3.3.2



	375 Kb/SEC	62.5 Kb/SEC
t_{CELL}	2.640 μSEC MIN 2.693 μSEC MAX	15.84 μSEC MIN 16.16 μSEC MAX

Figure 22. Data Signal Pair Specification for Self Clocked Mode

3.3.5 REPEATER TIMING

The self clocked mode of operation requires timing specifications for repeaters. Specifically, the skew between the two logic transitions and the skew between the data signal pair and the transceiver control pair need to be specified.

The worst case skew permitted through a repeater between the high to low transition time and the low to high transition time shall be less than ±50 nsec as shown in figure 23. The worst case skew between the data signal pair and the transceiver control signal pair shall be less than ±250 nsec as shown in figure 24.

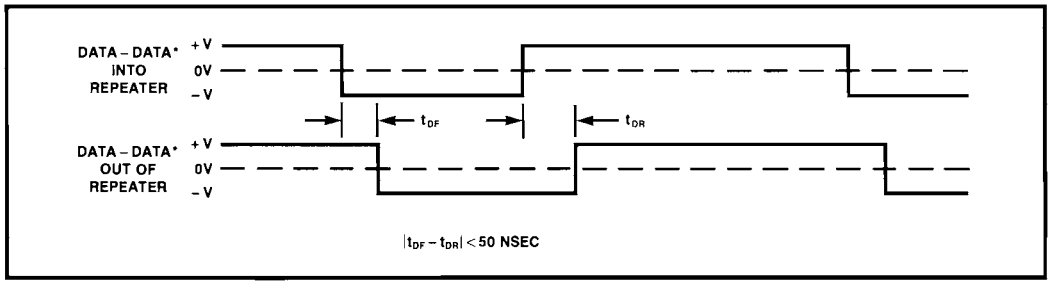


Figure 23. Repeater Skew on Transitions

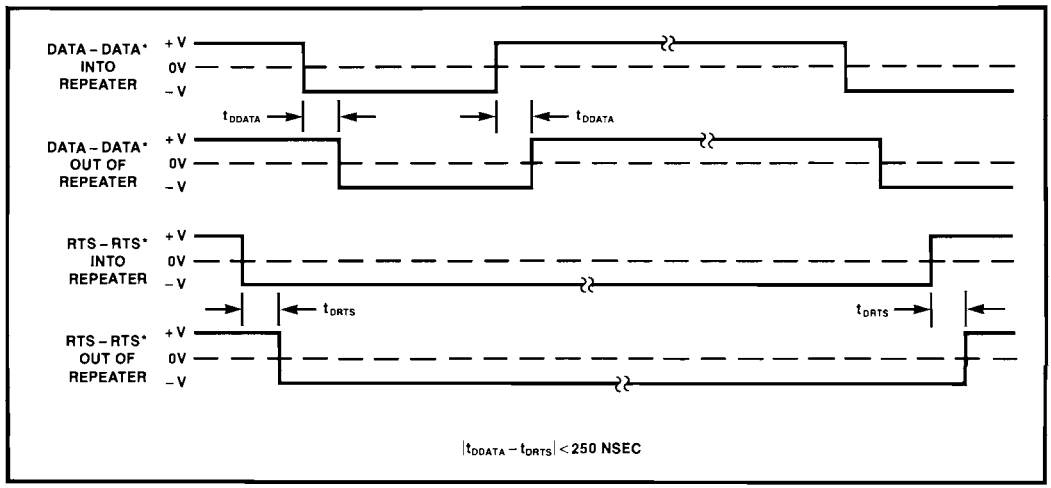


Figure 24. Repeater Skew Between Signal Pairs

The above specifications are for a standard repeater skew. Actual implementations of repeaters may be multiples of or a fraction of this standard. In all cases the specifications shown in table 1 shall apply for the maximum number of standard repeater skews between the master and any slave.

Table 1. Number of Repeaters

SPEED (kbit/sec)	MAX # OF STANDARD REPEATERS
375	2
62.5	10

4.0 DATA LINK PROTOCOL SPECIFICATION

The BITBUS data link protocol is a subset of the IBM Synchronous Data Link Control (SDLC) standard. SDLC is used as a basis for the BITBUS data link protocol because it is a highly reliable and proven protocol. The BITBUS interconnect does not (and is not intended to) maintain strict SDLC compatibility.

Based on SDLC, the BITBUS data link protocol connects a single master device to multiple slave devices in a multidrop (i.e. bus) topology. The data link protocol is specifically responsible for encapsulation of messages (messages are defined in section 5 of this specification) into frames and the control of frame transfer over the data link. To accomplish this, the specification defines the concept of system state, a general frame format, specific control fields and basic bus operations.

4.1 SYSTEM STATE

The ability of the BITBUS interconnect to connect nodes over a physically distributed domain requires some aspects of the system state to be formally defined. Being a hierarchical system, this state is defined from the point of view of the master device. This section provides an overview of the concept of system state. A detailed definition is provided in section 4.4.1 of this specification.

4.1.1 MASTER DEVICE STATE

The master device has full knowledge and control of its own state. Operation of a slave device does not require any information about the master device state. This implies that the master device state information is not transferred on the BITBUS interconnect and, therefore, is not a part of this specification.

4.1.2 SLAVE DEVICE STATE

The state of a slave device on the BITBUS interconnect needs to be known by the master device for proper operation. However, being physically separated, it is impossible for the slave device state to be precisely known at all times by the master device (e.g. in the event of a local reset, power down, etc.). In order to update the master device on a slave device state, it is necessary to define a state transfer mechanism over the BITBUS interconnect. The master device uses this mechanism to keep a record of the last known state for each slave device. This state is assumed for the next transfer and checked. If incorrect, the appropriate recovery action is taken. Slave device state consists of two parts: slave device mode and sequence count.

4.1.2.1 Slave Device Mode

A slave device is always in one of two modes: normal disconnect mode (NDM) or normal response mode (NRM). Below is a brief description of each mode. A formal state diagram is provided in section 4.4.1 of this specification.

A slave device enters NDM after a local reset or when it detects an irrecoverable protocol error. In this mode, a slave is awaiting a specific command from the master device to enter NRM. A slave device may not exchange messages with the master device in this mode.

A slave device enters NRM only after receiving a specific command from the master device. Upon entering NRM, a slave device is "synchronized" with the master device, meaning that all sequence counts match (they are all initialized to 0). In this mode, a slave device may exchange messages with the master device as long as "synchronization" is maintained (i.e. no sequence count errors).

4.1.2.2 Sequence Counts

In NRM, sequence counts are used by the master device and each slave device to guarantee that frames are not lost or duplicated. Below is a brief description of how sequencing works. Further details are provided in section 4.3 of this specification.

Sequencing is performed by the master device and each slave device via two pairs of 3 bit sequence counts. Each slave device keeps an N_r (number received) sequence count and an N_s (number sent) sequence count. The master device keeps a corresponding pair of counts for each slave it communicates with. A slave device is "synchronized" with the master device when the sequence counts are correct. The sequence counts at a slave device are considered part of the slave device state. The sequence counts at the master device are master device's best knowledge of slave device state.

The N_r sequence count indicates the sequence count of the next expected incoming message. The N_s sequence count indicates the sequence count of the next message awaiting acknowledgement (may or may not be outstanding). Each time a transfer occurs in NRM these numbers are included and verified resulting in three possible outcomes: correct sequence count, recoverable sequence error or irrecoverable sequence error. The irrecoverable sequence count error case requires the master device to resynchronize with the slave device by causing it to return to NDM then re-enter NRM. The other cases allow the slave device to remain in NRM. Further details are provided later in this specification.

4.2 FRAME FORMAT

The data link protocol is responsible for creating frames to be transferred across the BITBUS interconnect. All frames use the frame format shown in figure 25. All fields are composed of one or more bytes (plus possible zero bit insertion), with the least significant bit (LSB) of each byte transmitted first.

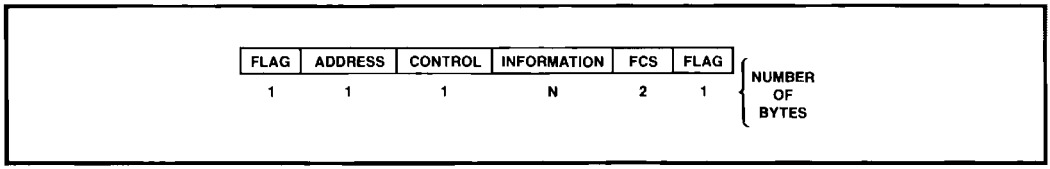


Figure 25. Standard Frame Format

4.2.1 FLAG FIELDS

The flag fields are used to delimit the frame. These fields (one at the beginning of the frame and one at the end of the frame) contain the unique bit pattern 01111110. The uniqueness of this bit pattern is guaranteed by using zero bit insertion/deletion in all other fields as described in section 3 of this specification. These fields are required in all frames.

4.2.2 ADDRESS FIELD

The address field contains the address of the slave device involved in the transfer. For a transmitting master device, this field identifies the destination slave device. For a transmitting slave device, this

field identifies the source slave device to the master device. If this field does not match the slave device address the frame is ignored. The address field is eight bits long and may contain values from 0 to 255. Values 0 and 251-255 are reserved by Intel. All others may be used without restriction. This field is required in all frames.

4.2.3 CONTROL FIELD

The control field is used for command and status exchange between the master device and slave devices. This field is eight bits long and is used for three classes of operations: synchronization, supervision and message transfer. Below is an overview of these operations. Details are provided in section 4.3 of this specification. This field is required in all frames.

4.2.3.1 Synchronization

The transfer of sequenced messages between the master device and a slave device requires that the slave device be properly synchronized to the master device. This synchronization process is performed using unnumbered frames (i.e. frames with unnumbered control fields). As the name implies, unnumbered frames do not use the sequencing feature.

4.2.3.2 Supervision

After the master device is synchronized with a slave device, it is often necessary to exchange status information in the absence of messages. This is done with supervisory frames (i.e. frames with supervisory control fields). These frames are used by the master device to poll a slave device and by a slave device to acknowledge receipt of a valid frame (i.e. address match and no CRC error) from the master device.

4.2.3.3 Information

Information frames (i.e. with information control fields) are used by the master device or a slave device to transfer messages (messages are defined in section 5 of this specification). These frames are only used after synchronization as are supervisory frames. In addition to a message and its sequence count, information frames carry the same status information as supervisory frames. If fact, information frames may be considered as a superset of supervisory frames.

4.2.4 INFORMATION FIELD

The information field is used to carry the BITBUS message. Details of this field are provided in section 5 of this specification. This field is required for information frames and is not used for supervisory or unnumbered frames.

4.2.5 FRAME CHECK SEQUENCE FIELD

The frame check sequence (FCS) field provides the lowest level of error detection on the BITBUS interconnect. This field contains a 16 bit cyclic redundancy check (CRC). The transmitting node generates and sends this field, while the receiving node checks it for correctness. A receiving node ignores an incoming frame if the CRC is incorrect. The CRC is generated by the standard CRC-CCITT polynomial $X^{16} + X^{12} + X^5 + 1$. This field is required in all frames.

4.3 CONTROL FIELD DEFINITION

Operations on the BITBUS interconnect are performed using three types of control fields: unnumbered, supervisory and information. This section specifies these fields in detail and explains their usage.

For reference to SDLC, this specification uses a subset of the defined control fields with the poll/final bit always set. This implies that frames sent by the master device always expect a response from

the addressed slave device (poll bit set) and that frames sent by slave devices always return control of the link (final bit set) to the master device.

4.3.1 UNNUMBERED FRAMES

Unnumbered frames are used on the BITBUS interconnect for synchronizing slave devices with the master device. This section specifies the control field format for these frames and defines the unnumbered operations (commands and responses) supported on the BITBUS interconnect.

4.3.1.1 Unnumbered Control Field Format

The control field format for unnumbered frames is shown in figure 26. This field may specify one of several unnumbered operations.

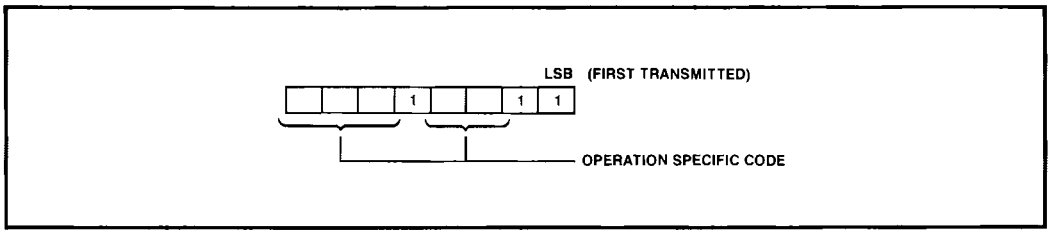


Figure 26. Unnumbered Control Field Format

4.2.1.2 Unnumbered Operations

The BITBUS interconnect supports two unnumbered commands (SNRM and DISC) and two unnumbered responses (UA and FRMR). The control fields for these operations are summarized in table 2.

Table 2. Unnumbered Control Fields

OPERATION	COMMAND	RESPONSE	CONTROL VALUE FIELD
SNRM	✓		93H
DISC	✓		53H
UA		✓	73H
FRMR		✓	97H

The frame reject (FRMR) command is sent to the master device by a slave device that detects an invalid control field in an otherwise valid frame. The response is used to respond to any unnumbered frame while in NRM, any supervisory or information frame while in NDM, any unsupported control field, or an irrecoverable sequence count error. Upon receiving this command, the master device shall initiate resynchronization with the slave device.

The unnumbered acknowledge (UA) response is used by a slave device to acknowledge receipt of a valid unnumbered command while in NDM.

The disconnect (DISC) command is sent by the master device to a slave device to initiate resynchronization. This command causes the slave device to go to, or stay in, NDM. It is used by the master device when it detects the need to resynchronize (e.g. after reset, irrecoverable sequence error, etc.) or in response to an FRMR from a slave device. When the master device receives an UA in response to a DISC it knows that the addressed slave device is in NDM.

The set normal response mode (SNRM) command is sent by the master device to synchronize a slave device. If a slave device is in NDM, this command causes it to enter NRM, allowing it to exchange messages with the master device. If a slave device is already in NRM, this command is invalid.

4.3.2 SUPERVISORY FRAMES

Supervisory frames are used on the BITBUS interconnect for passing status between the master device and a slave device while in NRM. These frames are used by a master device to poll slave devices and by slave devices to acknowledge receipt of a valid frame from the master device. This section specifies the control field format for these frames and defines the specific control fields that are supported.

4.3.2.1 Supervisory Control Field Format

The control field format for supervisory frames is shown in Figure 27. This field is used to specify the supervisory operation (RR or RNR) as well as to acknowledge receipt of a previous message via the N_r sequence count.

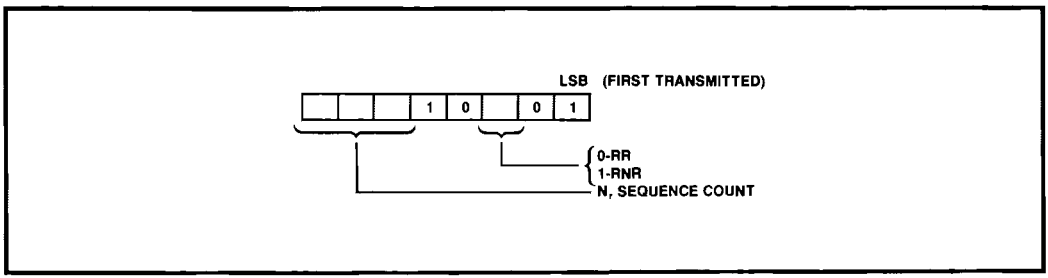


Figure 27. Supervisory Control Field Format

4.3.2.2 Supervisory Operation

The BITBUS interconnect supports two supervisory operations: receiver ready (RR) and receiver not ready (RNR).

The master device uses the RR supervisory frame to poll a slave device for an information frame. Slave devices use the RR supervisory frame to acknowledge valid reception (i.e. address match, valid CRC and available buffer space) of a previous supervisory or information frame when a message is not available (If a message is available an information frame is sent unless the incoming frame was a RNR supervisory frame).

The RNR supervisory frame is used to indicate that a buffer is presently unavailable to receive an otherwise valid frame. The master device uses the RNR supervisory frame to poll a slave device for data link status only. A slave device shall not respond to a RNR supervisory frame with an information frame. A slave device uses the RNR supervisory frame to indicate that the last frame was recognized, however there was no buffer available to store it. In this case the master device shall retransmit the frame.

Both RR and RNR frames contain an N_r sequence count. This sequence count acknowledges receipt of N_r-1 frames (i.e. N_r equals the next expected incoming frame). The receiver of a supervisory frame shall always compare this value to its N_s sequence count for correctness.

In the case that all messages have been acknowledged, the incoming N_r sequence count should equal the N_s sequence count. If the previous frame sent by the receiving node was an information frame, then the incoming N_r sequence count should equal the N_s sequence count plus 1. If this case is true, the previous frame is acknowledged meaning the N_s sequence can be incremented and the message buffer released. If after an information frame the incoming N_r sequence count is equal to the N_s sequence count, the information frame was not received and shall be retransmitted. Any other N_r sequence count value is detected as an irrecoverable sequence error requiring the master device to resynchronize with the slave device.

4.3.3 INFORMATION FRAMES

Information frames are used on the BITBUS interconnect for message transfer. These are the only frames that contain an information (I) field. The content of the I field is the BITBUS message as defined in section 5 of this specification. This section specifies the control field format for these frames and defines the operation of the sequence counts.

4.3.3.1 Information Control Field Format

The control field format for information frames is shown in Figure 28. The control field contains an N_r sequence count to acknowledge valid reception of $N_r - 1$ frames and an N_s sequence count corresponding to the attached information field (i.e. message).

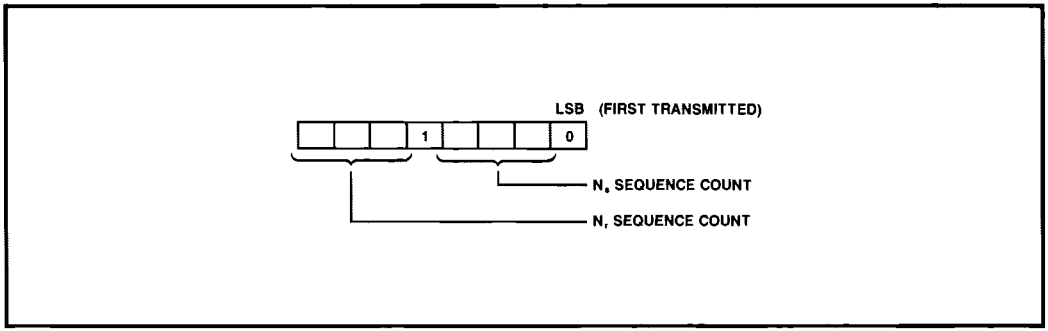


Figure 28. Information Control Field Format

4.3.3.2 Information Operation

Information frames are a superset of supervisory frames. They carry the same acknowledgement status as supervisory frames via the N_r sequence count. This count shall always be compared to the receiver's N_s sequence count as it is for supervisory frames.

In addition to the N_r sequence count, information frames carry a message in the information field and a corresponding N_s sequence count. Upon receiving an information frame, a receiver shall compare the incoming N_s count with its N_r count. If the counts match, the message shall be passed on. If the counts do not match, error recovery action shall proceed as follows. If the incoming sequence count N_s equals the receiver's sequence count $N_r - 1$, the receiving node may acknowledge receipt of the frame, then discards it (this case occurs when an information frame is received and its acknowledge is corrupted, making the second information frame a duplicate). If the incoming sequence count N_s does not equal the receiver's sequence count N_r or $N_r - 1$, then an irrecoverable sequence error is detected. This case requires the master device to resynchronize with the slave device before further message transfers can occur.

4.4 DATA LINK OPERATION

This section summarizes the data link operation on the BITBUS interconnect. This summary includes a precise definition of slave state, slave state transition, and sequenced operations in NRM. In addition sample synchronization sequences, sample transfer sequences and a summary of error conditions and the corresponding recovery actions are presented.

4.4.1 SLAVE STATE DESCRIPTION

The state transition diagram for a slave device is shown in Figure 29. As previously discussed, the state diagram has only two basic states: normal response mode (NRM) and normal disconnect mode (NDM). Table 3 lists the responses to all possible incomming frames in each of the states. Note that

this table assumes that the slave device has recognized a valid incoming frame. That is, the address field matched the node address and the CRC field was correct.

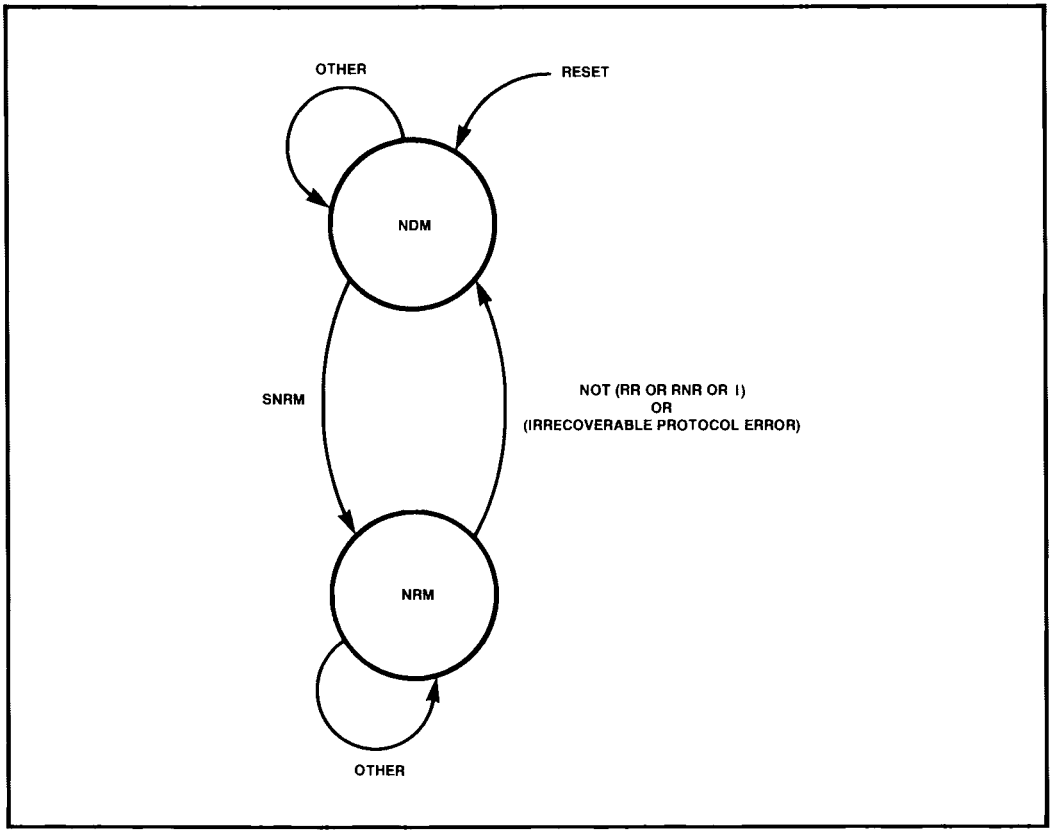


Figure 29. Slave Device State Diagram

Table 3. Slave Device State Transitions and Responses

SLAVE STATE	INCOMING FRAME	NEXT STATE	RESPONSE
NDM	DISC	NDM	UA
NDM	SNRM	NRM	UA
NDM	Other	NDM	FRMR
NRM	INFORMATION	NRM	RR,RNR,I
NRM	RR	NRM	RR,RNR,I
NRM	RNR	NRM	RR,RNR
NRM	INFORMATION,RR,RNR with irrocoverable sequence error	NDM	FRMR
NRM	Other	NDM	FRMR

When in NRM, the slave device state also includes its N_r and N_s sequence counts. For simplicity, these states are not shown but instead, a flow chart is provided in figure 30. This flow chart, along with table 3, completely specifies the slave device interface.

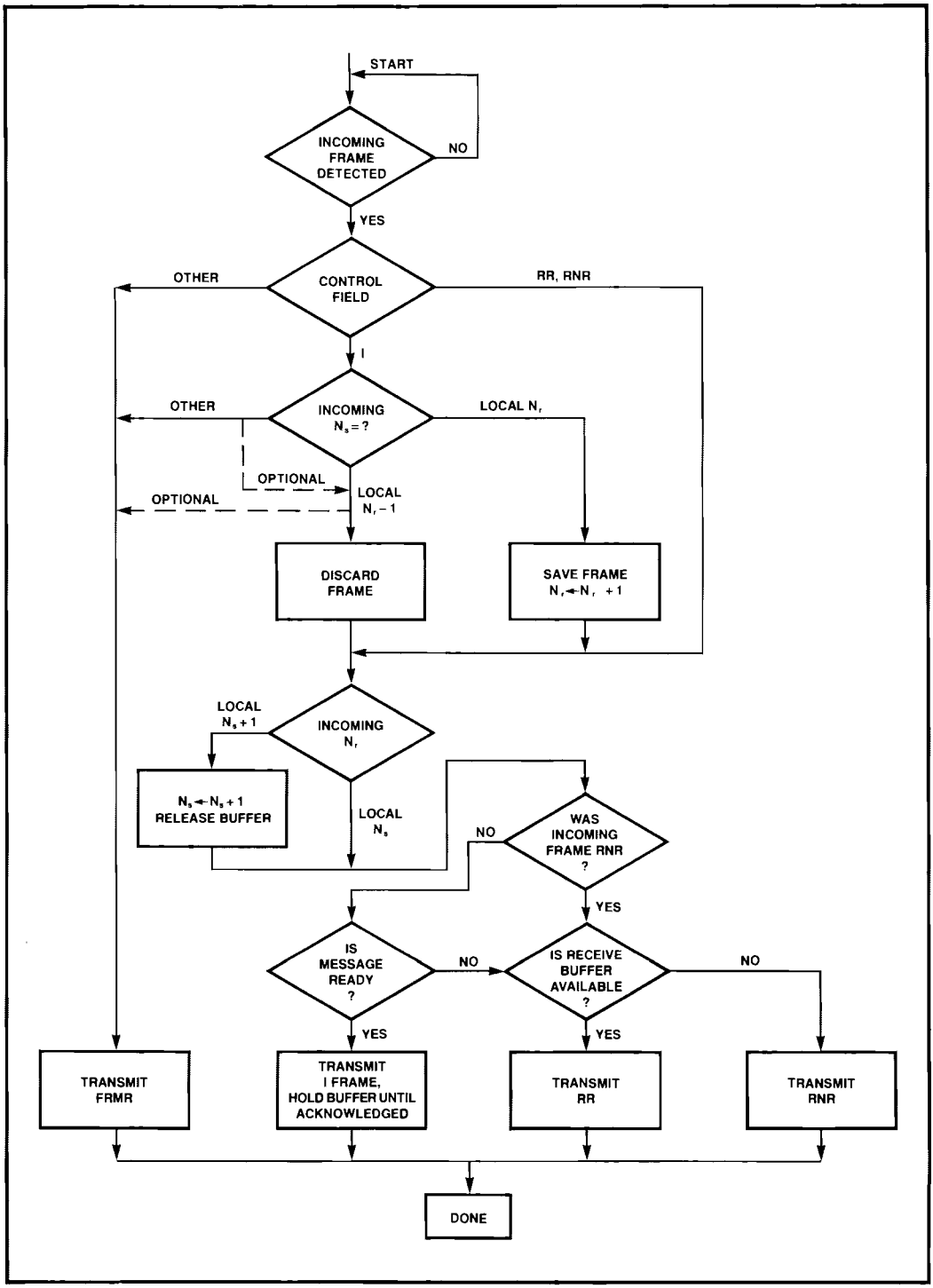


Figure 30. Slave Response to Incoming Frame in NRM

4.4.2 SYNCHRONIZATION SEQUENCE

The synchronization sequence on the BITBUS interconnect is used to establish a known state at the slave device (NRM with sequence counts equal to zero), in order to allow sequenced message transfer. Synchronization is required anytime the master device is reset or anytime the master device or slave device detect an irrecoverable protocol error. Synchronization is strictly a data link protocol feature and, therefore, does not appear at any higher levels of this specification. Figure 31 shows a typical synchronization sequence.

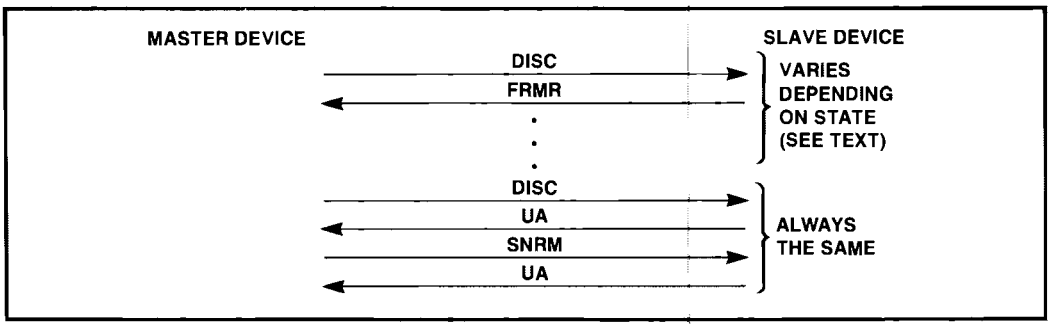


Figure 31. Typical Initialization Sequence

The master device may initiate the synchronization sequence by sending a DISC. The response from the slave device to a disconnect is a FRMR if it is in NRM or an UA if it is in NDM. The master device shall continue sending DISC frames until a UA response is returned verifying that the slave is in NDM. The master device then completes the sequence by sending an SNRM. Upon receiving the SNRM while in NDM, a slave device enters NRM and responds with a UA. Once the master device receives a UA to an SNRM, it knows that the slave device is synchronized.

The slave device may initiate the synchronization sequence by responding to an incoming frame with a FRMR frame. The master device responds to a FRMR with a DISC and proceeds through the sequence described above.

4.4.3 TRANSFER SEQUENCES

This section provides several transfer sequence examples. These examples serve as a basis for the message protocol defined in section 5 of this specification. Three examples are provided in figure 32: message transfer from a master device to a slave device, message transfer from a slave device to a master device and piggybacking of messages on acknowledges or polls. For these examples, it is assumed that the slave device is synchronized and that incoming frames are valid. Errors are discussed further in section 4.4.4 of this specification.

In the first example, the master device sends a message to a slave device. The information frame that carries the message also carries the sequence counts $N_s = 0$ and $N_r = 0$. The slave device verifies that the incoming N_r equals its N_s and that the incoming N_s equals its N_r. The slave then increments N_r to indicate it received the information frame and returns an RR frame with sequence count $N_r = 1$ to acknowledge the frame. Upon receiving the N_r sequence count, the master device sees that its information frame has been received, increments its N_s sequence count for the next message and releases the transmit buffer.

In the second example, the master device is polling a slave device with an RR frame for a message. Upon receiving the RR frame, the slave device verifies the incoming N_r sequence count and responds with an information frame containing the message and sequence counts $N_s = 0$ and $N_r = 1$.

Upon receiving the information frame, the master device verifies the incoming N_r and N_s sequence counts. It then increments its N_r to indicate that it has received the information frame. On the next transmission to the slave device, the master device sends its sequence count $N_r = 1$ to acknowledge receipt of the information frame. Upon detecting this, the slave device increments its N_s sequence count and releases the transmit buffer.

In the third example, the link efficiency is increased by piggybacking acknowledges and polls onto information frames. At both master device and slave device, an incoming frame carries an N_r sequence count which causes the N_s sequence count to be incremented and the last transmit buffer released. In addition, the N_s sequence count causes the N_r sequence count to be incremented. These results are then returned on the next information frame and the process is repeated.

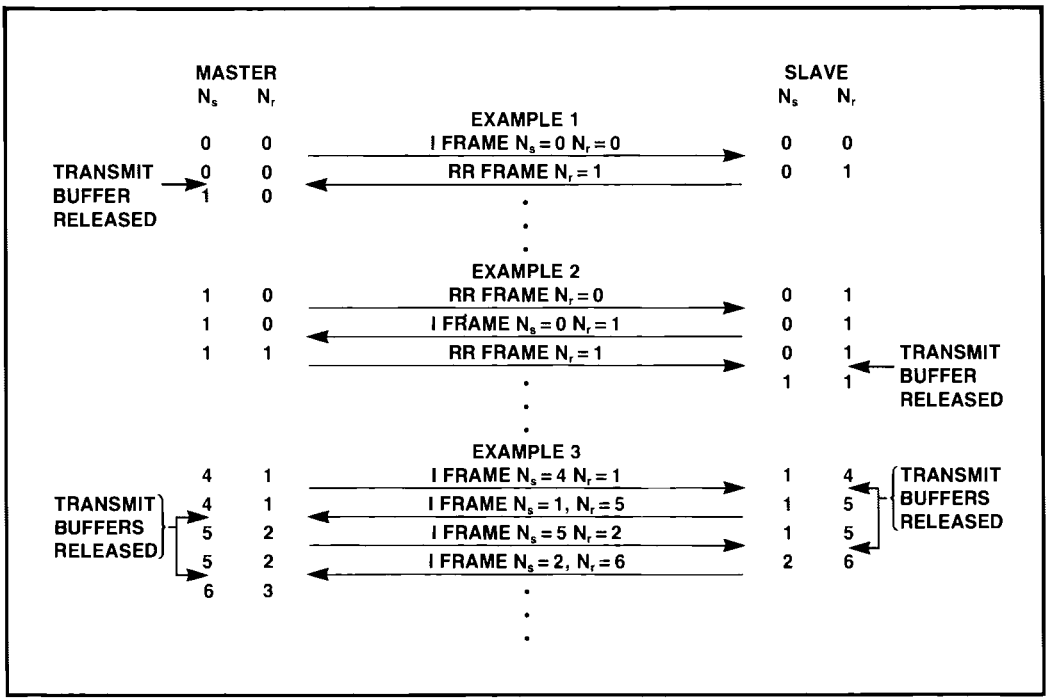


Figure 32. Transfer Sequence Examples

4.4.4 DATA LINK ERROR SUMMARY

The following section summarizes the possible data link error conditions and specifies the corresponding action. Specific error conditions are time out, invalid control field and sequence count.

4.4.4.1 Time Out

A time out error occurs when the master device does not receive a response to a transmitted frame within 10 milliseconds. This time is measured from the time transmission is completed until the response is received and the CRC verified.

Time out errors can occur under the following conditions: the addressed slave does not exist, the address slave discards the frame due to a CRC error, or a CRC error is detected by the master device in a response. In all cases the recovery action taken by the master device is to retransmit the frame a second time. If two consecutive errors are detected an irrecoverable protocol error is returned to the user.

4.4.4.2 Invalid Control Field

An invalid control field detected by either the master device or a slave device is considered an irrecoverable protocol error. Detection of this error condition at a slave device causes the slave device to enter NDM and respond with a FRMR. Detection at the master device causes it to assume that the slave is in NDM, requiring resynchronization.

4.4.4.3 Sequence Count

Sequence count errors are recoverable in some cases as indicated in figure 30. Cases that are not recoverable are considered irrecoverable protocol errors. Detection of an irrecoverable condition at a slave device causes it to enter NDM and respond with a FRMR. Detection at the master device causes it to assume that the slave is in NDM, requiring resynchronization.

5.0 MESSAGE PROTOCOL

The BITBUS message protocol is designed to provide a task to task message interface between a master node and multiple slave nodes using an order/reply structure. That is, the master node issues orders to slave nodes which respond with replies. This structure is built on top of the data link protocol using information frames for the message transfer. This section specifies the order/reply structure (including error handling) and the standard message format.

5.1 ORDER/REPLY STRUCTURE

The BITBUS message protocol is based on an order/reply structure in a multitasking environment. An order is defined as a message that is sent by a task on the master node, over the BITBUS interconnect, to a task on a slave node. A reply is defined as a message that is sent by as task on a slave node, over the BITBUS interconnect, to a task on the master node, in response to an order from that master node task. Master node tasks may exist on either the master device or master device extension (collectively referred to as the master node). Likewise, slave node tasks may exist on either the slave device or slave device extension (collectively referred to as the slave node). Note that orders and replies may have a more general definition when used between tasks on a given node. This special case is not part of this specification.

Every order on the BITBUS interconnect requires a corresponding reply; however, replies need not be returned in the same sequence that orders are issued. For example, if the master node issues orders A,B,C to a slave node, the responses may be returned as A,B,C or B,C,A or C,A,B or etc. Based on this requirement, a master device need only poll a slave node when one or more orders are outstanding. This condition exists when the sequence counts N_r and N_s for a given slave node are not equal. When the sequence counts N_r and N_s are equal, the slave node has no more replies and need not be polled. This algorithm limits unnecessary polling and increases system performance.

The data link protocol provides sequence counts that are 3 bits long. This limits the number of outstanding messages to any given slave node to seven. The slave node is responsible for enforcing this limit by not providing a receive buffer for the eighth message.

The BITBUS message protocol provides sufficient control to route orders from the master node to a slave device or slave device extension and replies from a slave node to a master device or master device extension. This capability allows local messages within a node to co-exist in a common gateway implementation. Furthermore, multiple level hierarchies (as shown in figure 1) are easily supported using the same master device software. The details of these implementations are not part of this specification.

Several sources of error exist in the message protocol. Specific examples are irrecoverable protocol errors or missing tasks. In all cases, the BITBUS message protocol detects the errors and handles

them in the same way. If an error occurs while delivering an order, the complete order is immediately converted to a reply and returned to the originating task. The reply is identical to the order except for the error code in the response field. If an error is detected on a reply, it is simply discarded (note that this is a very rare case, except for catastrophic error, since the reply retraces the path of the order). In either case, there is a small possibility that a reply is not returned. The task originating the order must account for this by setting a time out period for recovery. Note that this time out is not related to the data link time out specified in section 4 of this specification. Its value is determined by the application.

5.2 MESSAGE FORMAT

The standard message format on the BITBUS interconnect is shown in figure 33. This message structure is sent across the BITBUS interconnect in the information field of information frames. All fields shown, except the data field, are required in all messages.

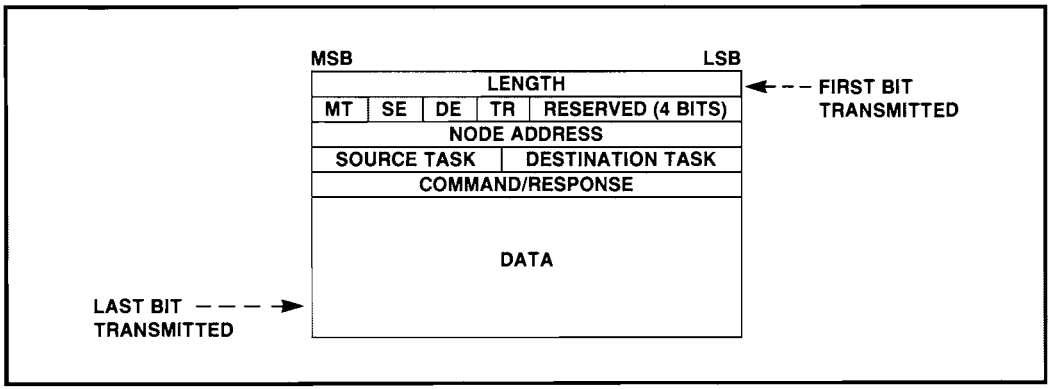


Figure 33. Message Format

5.2.1 LENGTH

The length field specifies the total message length. This eight bit field may contain values between 7 and 255. The value in this field equals the number of bytes in the data field plus 7. This value allows implementation to easily add two bytes for local message manipulation such as buffer control or queuing. All implementations shall support a data field of up to 13 bytes, corresponding to a length field equal to 20.

5.2.2 MESSAGE TYPE (MT)

The message type field is used to specify whether the message is an order or a reply. The master node always sends orders and, therefore, shall set this bit to 0. A slave node always sends replies and, therefore, shall clear this bit to 1.

5.2.3 SOURCE EXTENSION (SE)

The source extension field indicates whether the source of an order, or the destination of a reply, is the master device or the master device extension. This bit is set to 1 to indicate the master device extension and cleared to 0 to indicate the master device. This bit is unchanged between an order and its corresponding reply.

5.2.4 DESTINATION EXTENSION (DE)

The destination extension field indicates whether the destination of an order, or source of a reply, is a slave device or a slave device extension. This bit is set to 1 to indicate a slave device extension

and cleared to 0 to indicate a slave device. This bit is unchanged between an order and its corresponding reply.

5.2.5 TRACK (TR)

The track field is used to provide message control at a master or slave device which may be required by some implementations. This bit shall be cleared to 0 when sending a message. It shall be set to one upon receiving a message from the BITBUS interconnect.

5.2.6 RESERVED

These four bits are reserved for possible future enhancements. They shall be cleared to zero when sending a message and their value is not guaranteed upon receiving a message.

5.2.7 NODE ADDRESS

The node address field specifies the destination node for orders and the source node for replies (i.e. it is unchanged between order and reply). This eight bit field contains the same value as the address field in the data link frame. Valid entries are 1 through 250. The values 0 and 251-255 are reserved by Intel.

5.2.8 SOURCE TASK

The source task field identifies the task that has generated an order or is to receive a reply. This four bit field allows up to sixteen tasks to generate orders from the master device and the master device extension. Specific implementations may support as few as one or as many as sixteen tasks.

5.2.9 DESTINATION TASK

The destination task field identifies the task that is to receive an order or has generated a reply. This four bit field allows up to sixteen tasks to receive orders at each slave device and slave device extension. Specific implementations may support as few as one or as many as sixteen tasks.

5.2.10 COMMAND/RESPONSE

The command/response field is used by both user tasks and the message protocol. Under normal conditions, this field is used only by the user tasks. The message protocol only uses this field for reporting errors. Table 4 indicates the usage of this field. Note that this table only applies to responses. Commands are always defined by the user task.

Table 4. Message Protocol Responses

CONDITION	ERROR
No error (set by user)	00H
User defined	01H-7FH
No destination task	80H
Protocol Error	91H
No destination device	93H
Reserved by Intel	81H-90H,92H, 94H-0FFH

5.2.11 DATA

The data field is defined by the contents of the command field. Minimum support requires this field to be capable of handling 13 bytes. Implementation may extend it to as much as 248 bytes provided that the longer messages are not sent to nodes that cannot support them. This field is the only optional field in the message.

6.0 REMOTE ACCESS AND CONTROL

The remote access and control (RAC) interface for the BITBUS interconnect defines a set of high level commands and responses to perform general purpose operations at a slave node. The remote access interface allows I/O read and write capabilities as well as memory download and upload. The remote control interface allows control and monitoring of tasks at a slave node. The RAC interface is built on top of the message protocol defined in section 5 of this specification. In fact, RAC is simply a special task defined at slave nodes. This section specifies the commands, responses and the general RAC message format.

6.1 MESSAGE FORMAT FOR RAC

The RAC interface is built on top of the standard message protocol. The message format follows the standard format in figure 33 with the following fields further defined: destination task, command/ response, and data.

6.1.1 DESTINATION TASK

The BITBUS interconnect, by convention, defines task 0 as the RAC task. This makes its presence easily detectable and allows it to be addressed in the same way for all implementations. If the RAC task is not supported at a slave node, task 0 may be used for another function.

6.1.2 COMMAND/RESPONSE

The command/response field is used to identify the RAC command in an order message and the RAC response in reply messages. The various commands and responses supported by the RAC interface are specified in sections 6.2 and 6.3, respectively.

6.1.3 DATA

The contents of the data field are determined by the RAC command. The length field is used to identify the exact number of data bytes. The minimum BITBUS interconnect support provides up to 13 bytes in this field. Details of this field are provided in sections 6.2 and 6.3 of this specification.

6.2 RAC COMMANDS

Table 5 lists the RAC commands supported on the BITBUS interconnect. Commands 00H through 0BFH are general purpose commands controlled by Intel. At this time, 15 are specifically defined, and the rest are reserved. Commands 0C0H through 0FFH are available to the user for custom RAC commands. These values will not be standardized. Specific implementations of RAC may support all or any subset of the defined commands. The only specific requirements are that the implementation return an error response for unsupported commands and that all supported general purpose commands conform to this specification.

6.2.1 CREATE TASK

The create task command is used to initialize and begin execution of a task at a slave node. This command assumes that the task to be created is already in the slave node memory. The command is simply the mechanism used to inform the operating system at the slave node that a given task in memory is to become active.

The data field in a create task order contains an address pointer (length field equal to 7 plus pointer length) to the task descriptor in the slave node's memory. The length of the pointer is determined by the implementation requirements at the slave node. This pointer is sent most significant byte first. The actual task descriptor may vary between implementations and, therefore, is not part of this specification.

Table 5. RAC Commands

COMMAND	ACCESS	CONTROL	VALUE
Reset Slave		✓	00H
Create Task		✓	01H
Delete Task		✓	02H
Get Function ID		✓	03H
RAC Protect		✓	04H
Read I/O	✓		05H
Write I/O	✓		06H
Update I/O	✓		07H
Upload Memory	✓		08H
Download Memory	✓		09H
OR I/O	✓		0AH
AND I/O	✓		0BH
XOR I/O	✓		0CH
Status Read	✓		0DH
Status Write	✓		0EH
Reserved			0FH-0BFH
User Defined			0C0H-0FFH

After successful completion of a create task command, a reply message is returned to the originating task on the master node with 00H in the response field and the data field unchanged (i.e. address pointer is returned).

6.2.2 DELETE TASK

The delete task command performs the inverse function to the create task command. It is used to indefinitely terminate execution of a task at a slave node. This allows the task number associated with the deleted task to be reused (i.e. with a create task command).

The data field in a delete task order contains a single byte (length field equal to 8) identifying the task number to be deleted. This field may contain values between 0 and 15; however, beware of deleting task 0 as it is the RAC task.

After successful completion of a delete task command, a reply message is returned to the originating task on the master node with 00H in the response field and the data field unchanged (i.e. eight bit task number is returned).

6.2.3 GET FUNCTION IDs

Function IDs are logical IDs used to identify a given task by its function rather than its physical address. The concept of function ID allows functions to maintain a constant identifier, even if they are moved to a new physical address (node address and/or task number). The get function IDs command causes the slave device, or slave device extension, to respond with a list of function ID codes for the tasks currently in existence.

The get function IDs order message carries a dummy data field equal in length to the number of function IDs to be returned. The length field is used to identify the number of bytes (value in length field equals number of bytes plus 7). Carrying the dummy data field allows the slave node to allocate a single buffer that is sufficiently large to receive the order message, as well as return the reply message.

After execution of the get function IDs command, a reply message is returned to the originating task on the master node with 00H in the response field and the function ID codes in the data field. This first data byte corresponds to task 0, the second data byte to task 1, etc. The number of data bytes returned is always equal to the number sent in the order message.

The assignment of function IDs is summarized in table 6. As an example, consider a slave device that supports up to 8 tasks. Task 0 is RAC, task 1 is a user task with function ID 81H, tasks 2 and 3 are present but have no function ID and tasks 4-7 are not used. For this case, the length field in the order message and reply message is 15. The reply message would have a data field as shown in figure 34.

Table 6. Function ID Code Assignments

FUNCTION	VALUE
No Task	00H
RAC Task	01H
Reserved by Intel	02H-7FH
User Assigned	80H-0FEH
Task with no Function ID	0FFH

MSB LSB

LENGTH	
MT, SE, DE, TR, RESERVED (4 BITS)	
NODE ADDRESS	
SOURCE TASK, DESTINATION TASK	
TASK NUMBER	COMMAND/RESPONSE
0	01H
1	81H
2	0FFH
3	0FFH
4	00H
5	00H
6	00H
7	00H

TASK FUNCTION
RAC
User Function 81H
Task without function ID
Task without function ID
No Task
No Task
No Task
No Task

Figure 34. GET Function ID Reply Example

6.2.4 RAC PROTECT

The RAC protect command allows the remote access functions at a slave node to be enabled or disabled by a task on the master node. When disabled, the RAC function at a slave node only recognizes the remote control commands. On reset, the RAC function shall be enabled.

The RAC protect order contains a single byte data field and a length field of 8. If the value of this data field is 00H, the RAC function is enabled. If the value in this data field is 01H, the RAC function is disabled. Any other value may produce indeterminate results and should not be used.

After successful completion of an RAC protect command, a reply message is returned to the originating task on the master node with 00H in the response field and the data field unchanged.

6.2.5 RESET SLAVE

The reset slave command initiates a reset at the addressed slave device or slave device extension. The extent of the reset (e.g. software, slave device, slave device extension, entire slave node, etc.) is determined by the implementation. The data field for this command is null and the length field is always 7.

The reset slave command is the only command for which a reply message is not returned upon successful completion. A reply message is not returned in this case since its delivery cannot be predicted while the slave node is in the process of resetting. The only case when a reply message is attempted is in the case of error. This case is discussed in section 6.3 of this specification.

6.2.6 MEMORY COMMANDS

The memory commands allow blocks of data to be moved between the master node and a slave node. Operations include memory upload and memory download. Since these commands share a common format in the message data field, they are presented together.

The general data field format for memory commands is shown in Figure 35. This format includes a 16 bit address pointer and 1 to n bytes of data. The length field in the header is used to specify the actual number of data bytes (e.g. length equal to 9 plus n for n data bytes). The address pointer is used to identify the base address for the data bytes. That is, the pointer is the address associated with data byte 1 and the address of subsequent data bytes is obtained by simply incrementing the pointer.

The address pointer is limited to a 64K address range as this is believed to be sufficient for the vast majority of BITBUS interconnect applications. In the few cases where a larger address range is needed, it can be created by assigning a location in the I/O or status space as an offset register, thus allowing the memory commands to operate in 64K byte segments or pages.

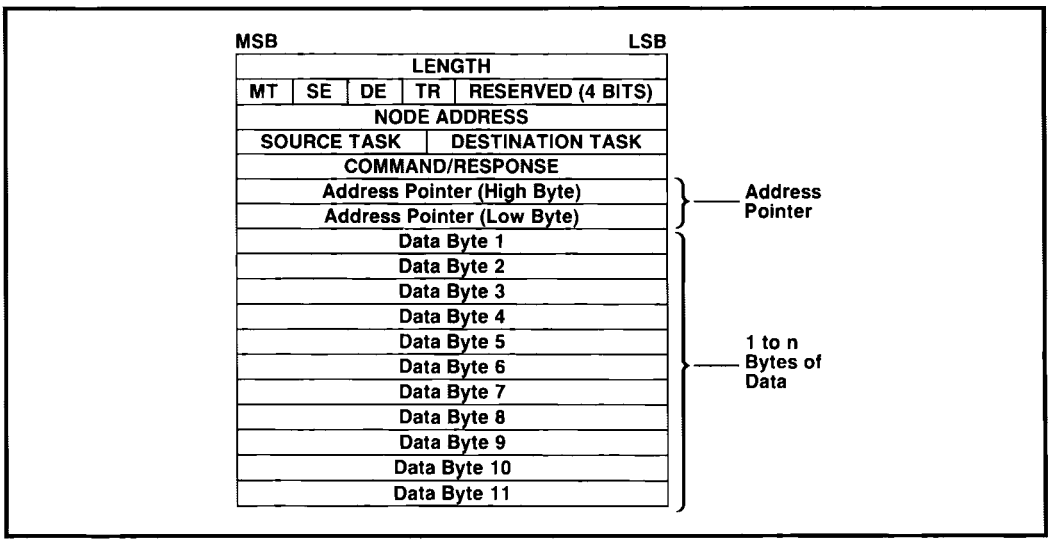


Figure 35. Data Field Format for Memory Commands

6.2.6.1 Memory Upload

The memory upload command causes a slave device, or slave device extension, to read a specified number of data bytes from its local memory and return them in a reply message. The specific number of bytes to be read is identified in the length field of the order message (i.e. length field equals number of bytes to be read plus 9).

The data field in a memory upload order message contains a 16 bit address pointer and n invalid data bytes, where n is the number of bytes requested. The invalid data bytes are carried to simplify buffer allocation at the slave node by allowing a single buffer to be allocated that is sufficient for delivering the order and returning the reply.

At the slave node, the RAC task reads sequentially the specified number of data bytes. The first byte is read from the location specified by the address pointer and subsequent bytes are located by incrementing the address pointer. Upon successful completion, a reply message is returned with an 00H response field and a data field with the original address pointer and the requested data bytes.

6.2.6.2 Memory Download

The memory download command causes a slave device, or slave device extension, to write n bytes of data into its local memory. The specific number of bytes to be written is identified in the length field of the order message (i.e. length field equals number of bytes to be written plus 9). In addition to the data bytes, the data field contains a 16 bit address pointer.

At the slave node, the RAC task writes the data bytes sequentially starting at the location specified by the address pointer. Subsequent addresses are located by incrementing the address pointer for each byte. Upon successful completion, a reply message is returned with an 00H response field and a data field containing the contents of the order message.

6.2.7 I/O COMMANDS

The I/O commands allow the master node to access up to 256 I/O ports on each slave device and each slave device extension. Operations include read I/O, write I/O, update I/O, OR I/O, AND I/O, and XOR I/O. Since these commands all share a common format in the message data field, they are presented together.

The general data field format for I/O commands is shown in figure 36. This format allows a single command to be executed on one or more ports in a single message. The data field is comprised of one or more pairs of bytes. The port address fields identify the I/O ports on which the operation is to be performed. The data byte fields contain the data for the operation or act as place holders for the data that results from an operation.

Upon successful completion of an I/O command, a reply message is returned with an 00H response field and a data field with the same port address fields as the order message and data byte fields as specified by the command. In all cases, the reply message length is equal to that of the order message.

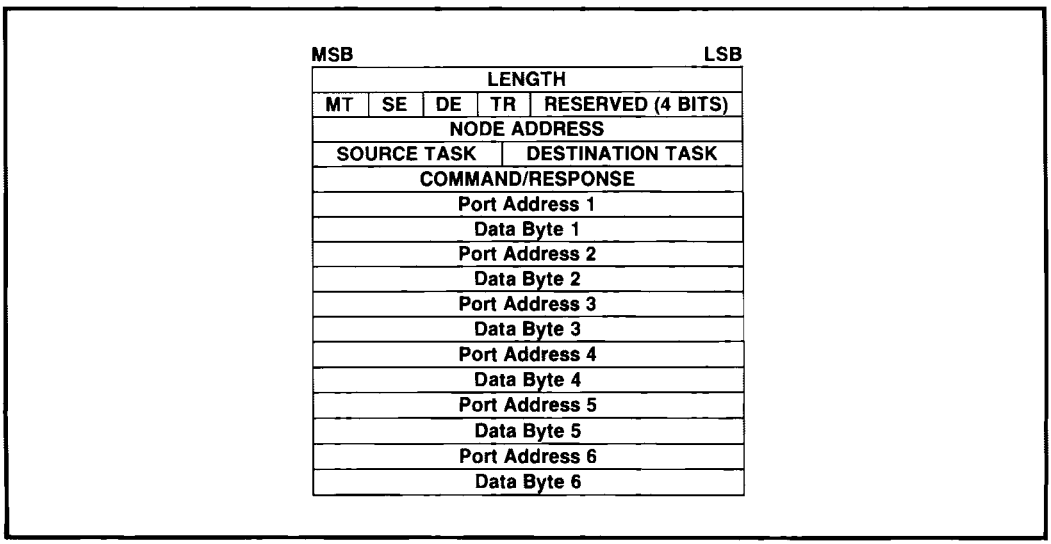


Figure 36. Data Field Format for I/O Commands

6.2.7.1 Read I/O

The read I/O command causes a slave node to read the specified I/O ports. The data byte fields in the order message are undefined. They are carried only to simplify buffer allocation at the slave node. The data byte fields in the reply message contain the data read from the specified ports.

6.2.7.2 Write I/O

The write I/O command causes a slave node to write the data bytes fields to the specified I/O ports. The data byte fields in the order message contain the data to be written. The data byte fields in the reply message are unchanged from the order message.

6.2.7.3 Update I/O

The update I/O command causes a slave node to write the data byte fields to the specified I/O ports then reread the ports. The data byte fields in the order message contain the data to be written. The data byte fields in the reply message contain the data reread from the I/O port after the write operation.

6.2.7.4 Logical Operations

The logical operation commands provide bit set, bit reset and bit toggle operations. The data byte fields for these operations contain a mask. The command causes the slave node to read the specified I/O port, perform the specified logical operation with the mask in the data byte field, write the result back to the I/O port, then finally reread the port. This final value is entered in the data byte field for the reply message.

Bit operations are defined to logically set, clear and toggle I/O bits. The electrical levels of these operations depends on the implementation. When no signal inversion exists between the internal CPU state and the I/O pin, the operations of set and clear correspond to the electrical high and low states respectively. These implementations are referred to as "active high." If a signal inversion occurs between the internal CPU state and the I/O pin, the electrical definitions of set and clear are reversed. These implementations are referred to as "active low."

From the logical point of view, bit operations are defined as follows. A bit is set by using an OR I/O command with a one in the bit position to be set. A bit is cleared by using an AND I/O command with a zero in the bit position to be cleared. A bit is toggled by using an XOR I/O command with a one in the bit position to be toggled. Note that it is possible to operate on multiple bits with the same byte in a single operation.

6.2.8 STATUS COMMANDS

The status commands allow the master node to access up to 256 bytes of memory on each slave device and slave device extension. These operations can be used to create a shared data structure between the master and slave nodes. Specific operation include read status and write status. Since these commands share a common format in the message data field, they are presented together.

The general data field format for status commands is shown in figure 37. This format allows a single command to operate on one or more locations with a single message. The data field is comprised of one or more pairs of bytes. The pairs include an address field and a data byte field. The address fields identify the locations at which the operation specified by the command is performed. The data byte field contains the data for the operation or acts as a place holder for the data that results from the operation.

Upon successful completion of a status command, a reply message is returned with an 00H response field and a data field with the same address fields as the order message and data byte fields as specified by the command. In all cases, the reply message length is equal to that of the order message.

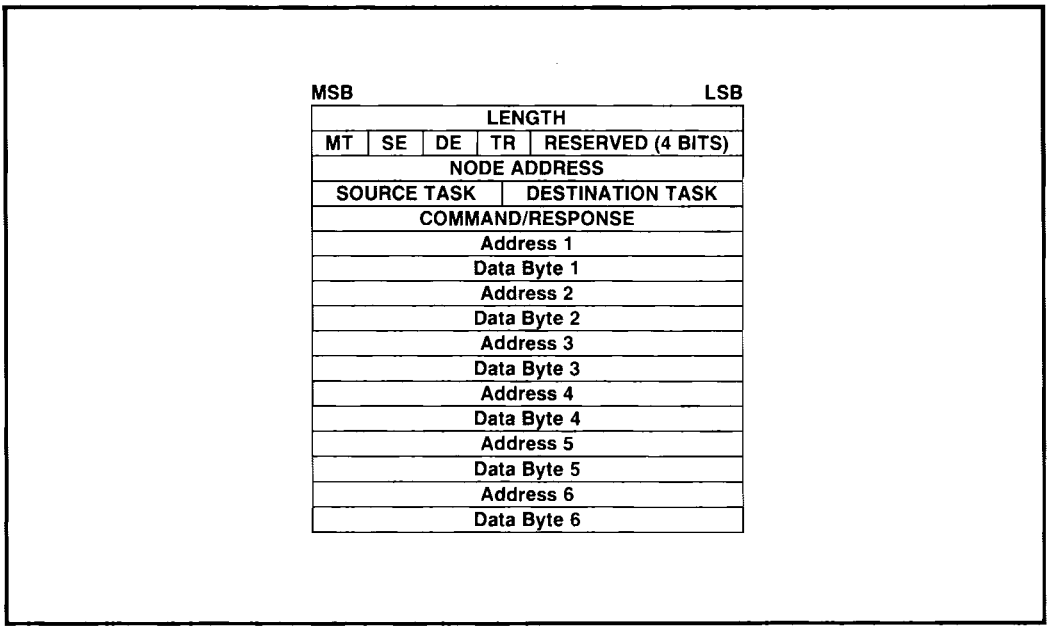


Figure 37. Data Field Format for Status Commands

6.2.8.1 Read Status

The read status command causes a slave node to read the specified memory locations. The data byte fields in the order message are undefined. They are carried only to simplify buffer allocation at the slave node. The data byte fields in the reply message contain the data read from the specified addresses.

6.2.8.2 Write Status

The write status command causes a slave node to write the data byte fields to the specified addresses. The data byte fields in the order message contain the data to be written. The data byte fields in the reply message are unchanged from the order message.

6.3 RAC RESPONSES

In general, RAC order messages expect RAC reply messages. The only cases where reply messages are not returned are for a successfully delivered reset slave order message or under certain error conditions as specified in section 5 (section 5 also specifies how to handle this case). When a reply message is returned, it indicates one of three cases: no error, message protocol error, or RAC error. Reply messages for the no error case are specified for each command in section 6.2. Reply messages for message protocol errors are specified in section 5. This section specifies the reply messages for the various RAC error conditions.

In general, RAC errors generate a reply message similar to message protocol errors. The reply message for RAC errors is simply the order message with the error response inserted in place of the command. the data field is returned intact. Table 7 summarizes the RAC error responses.

6.3.1 NO TASK

No task is generated in response to a delete task command in the case where the task to be deleted does not exist.

Table 7. RAC Error Responses

RESPONSE	VALUE
No Task	80H
Task Overflow	81H
Register Bank Overflow	82H
Duplicate Function ID	83H
No Buffers	84H
RAC Protected	95H
Unknown RAC Command	96H

6.3.2 TASK OVERFLOW

Task overflow is generated in response to a create task command in the case where the slave device or slave device extension cannot support another task. This response indicates that it is necessary to delete a task before another can be created.

6.3.3 REGISTER BANK OVERFLOW

Register Bank Overflow is generated in response to a create task command that cannot be executed due to a lack of register bank resources.

6.3.4 DUPLICATE FUNCTION ID

Duplicate Function ID is generated in response to a create task command in the case where the task to be created has the same function ID (0FFH excepted) as a task currently recognized by the operating system.

6.3.5 NO BUFFERS

No buffers is generated in response to a create task command that cannot be executed due to a lack of memory resources.

6.3.6 RAC PROTECTED

RAC protected is generated in response to any remote access command when the RAC function is disabled. Before the remote access command can be recognized, an RAC protect command must be sent to enable the RAC function.

6.3.7 UNKNOWN RAC COMMAND

Unknown RAC command is generated in response to any RAC command that is not recognized. This response is used for both undefined and unimplemented commands.

7.0 MECHANICAL SPECIFICATION

The BITBUS interconnect provides mechanical specifications for cable connectors and a standard printed board. The cable connectors are used to connect cables to printed boards or cables to other cables. The standard printed board is specified to allow multiple manufacturers to build complementary products that fit into a consistent packaging scheme.

7.1 CONNECTOR SPECIFICATIONS

The BITBUS interconnect requires specification of two standard connectors: one for cable to printed board connection and one for cable to cable connections. Both connectors support the same signals. Four pins are used for the two differential signal pairs specified in section 2 of this specification for

the BITBUS interconnect. Four additional pins are provided for power distribution to low power nodes. Finally a high impedance ground (100 ohms to ground) is provided as specified in the RS485 specification (not required in all implementations). Figure 38 and 39 show the two connectors and table 8 lists the pin assignments.

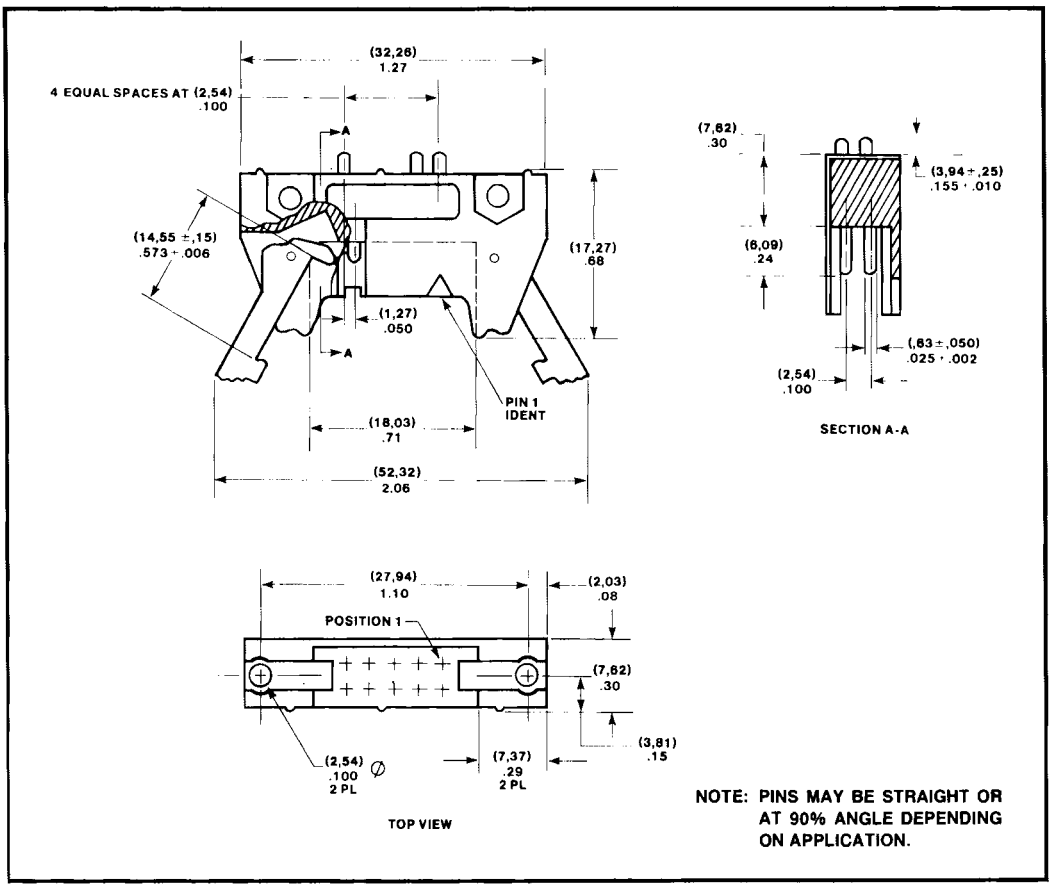


Figure 38. Printed Board Connector

Table 8. Connector Pin Assignments

SIGNAL	PRINTED BOARD CONNECTOR PIN #	CABLE CONNECTOR PIN #
+12V ± 5%	1	1
+12V ± 5%	2	6
GND	3	2
GND	4	7
DATA*	5	3
DATA	6	8
DCLK*/RTS*	7	4
DCLK/RTS	8	9
RGND (100 ohms to GND)	9,10	5

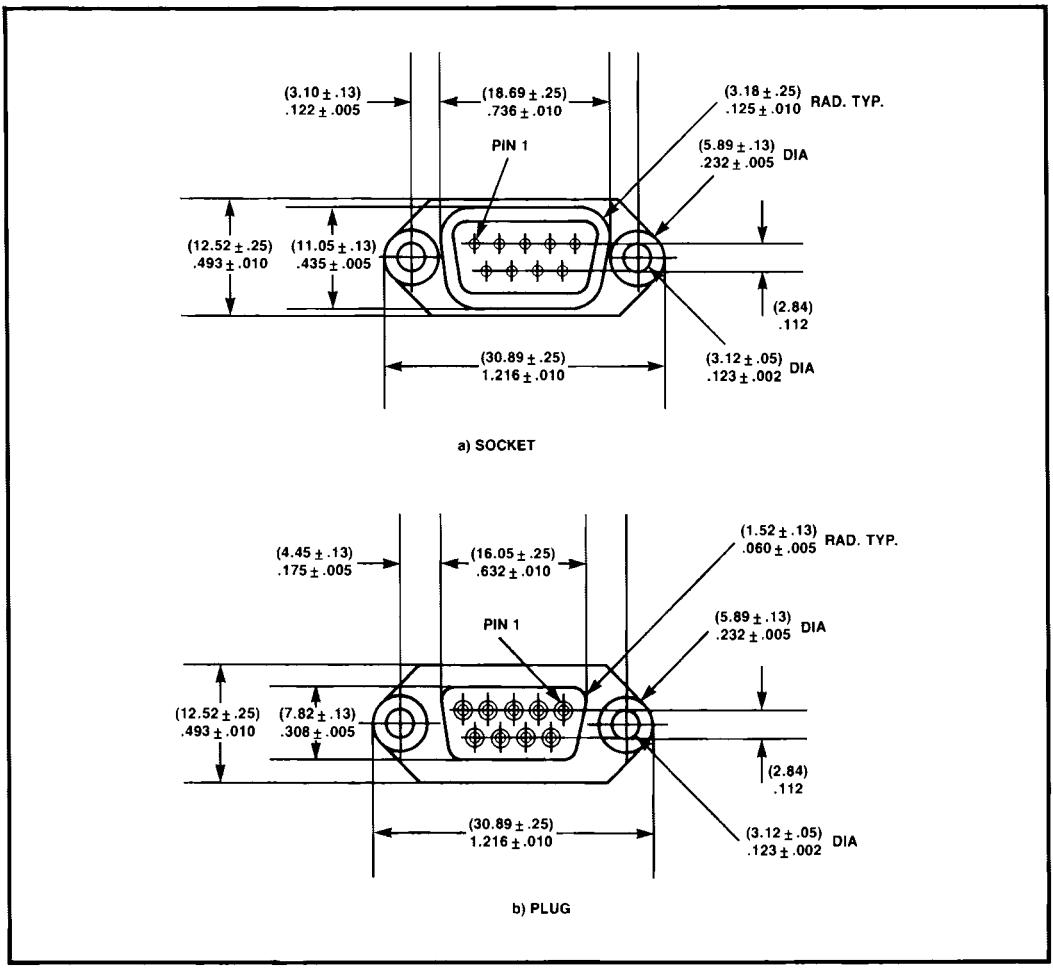


Figure 39. Cable to Cable Connector

The standard cable to printed board connector is a 10 pin latching pin and socket type with strain reliefs. This connector can support either flat cable or discrete wire connection.

The standard cable to cable connector is a 9 pin D-subminiature type. Versions of this connector are also available for flat cable or discrete wire connection.

7.2 I/O BOARD SPECIFICATION

The BITBUS I/O board is designed to be compatible with the single-high Multibus II board form factor. This definition is based on the International Electrotechnical Commission (IEC) standards. This section provides an overview of the mechanical dimensions of this board and presents standard pin assignments. For further details (e.g. board spacing, component height, ejector tabs, front panels, etc.) refer to the Mechanical Specification in the Multibus II Bus Architecture Specification.

7.2.1 BOARD DIMENSIONS

Figure 40 shows the board dimensions for the standard BITBUS I/O board. This board is a standard single-high, 220mm deep form factor.

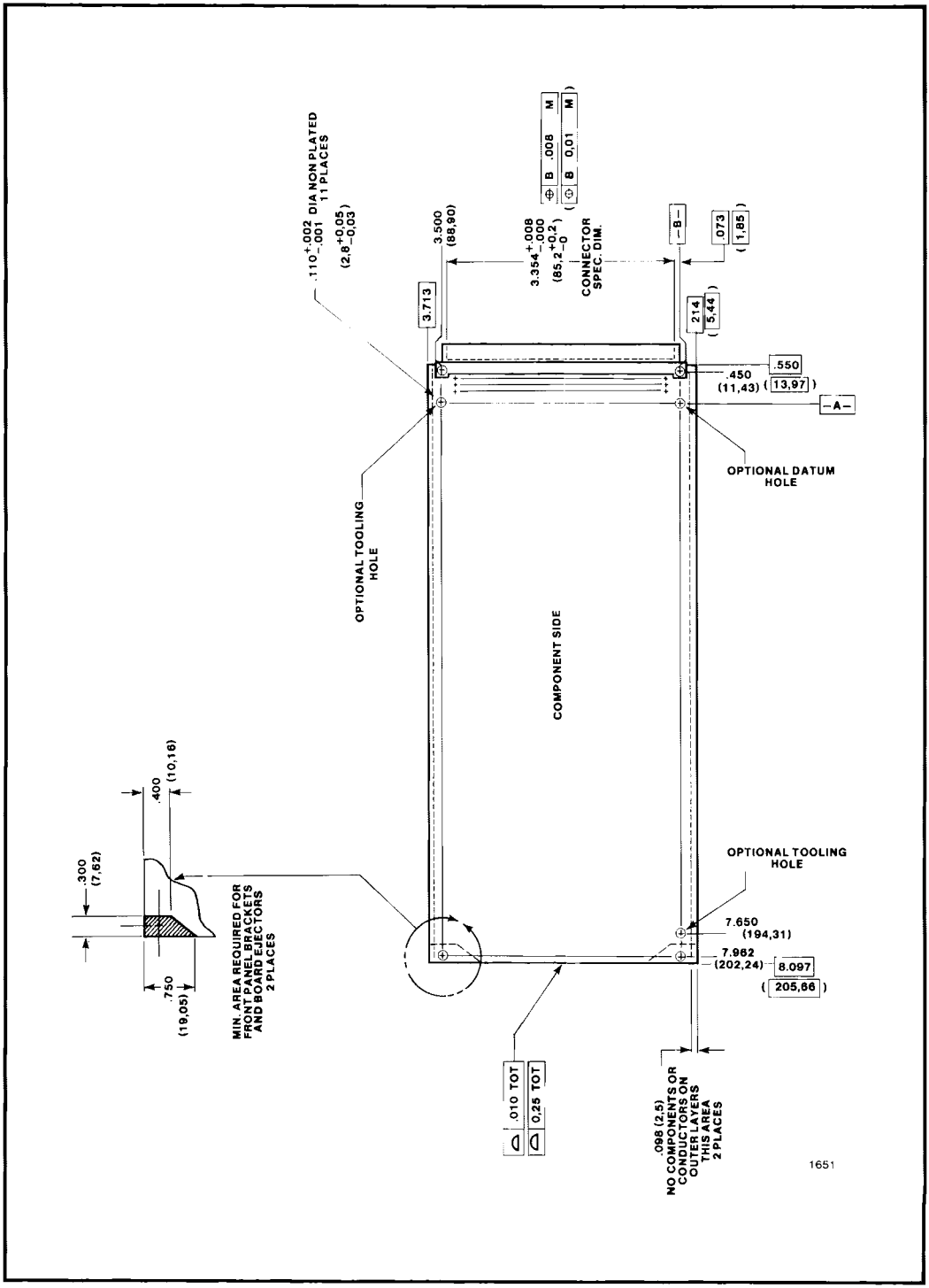


Figure 40. Standard Board Dimensions

7.2.2 CONNECTORS

The BITBUS I/O boards use two piece, 64-pin connectors. Figure 41 and 42 show the dimensional specification for the connectors. The right angle connectors on the printed board are IEC standard 603-2-IEC-C064-M; the receptacle connectors are IEC standard 603-2-IEC-C064-F. Figure 43 shows the relationship between connectors and boards in a subrack. Note that compatible recepticle connectors are available for flat cable, discrete wire or wire wrap connections in addition to the backplane version shown.

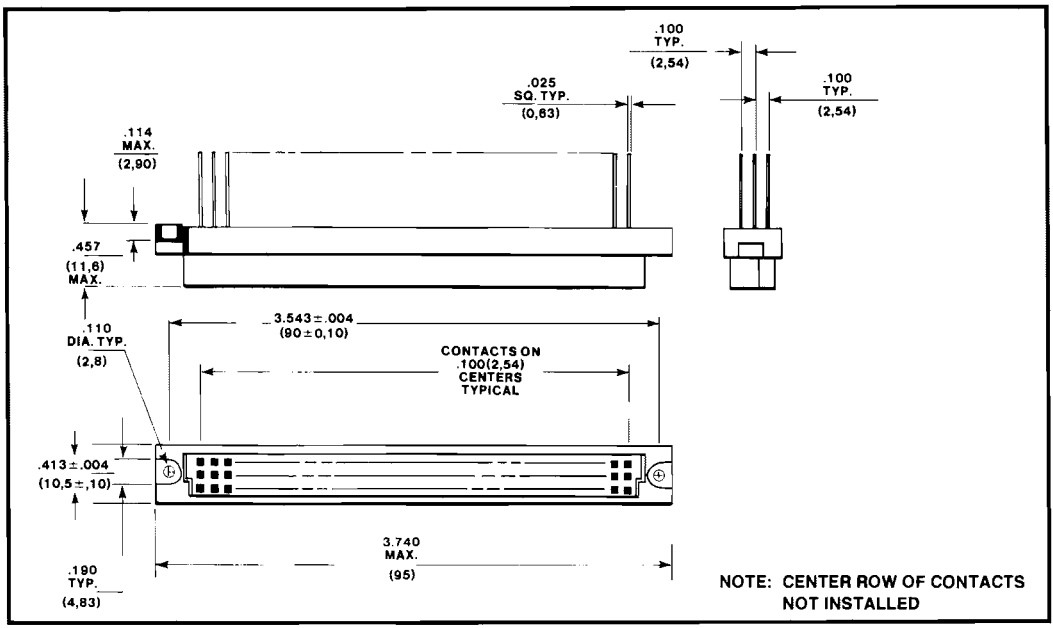


Figure 41. Board Connector

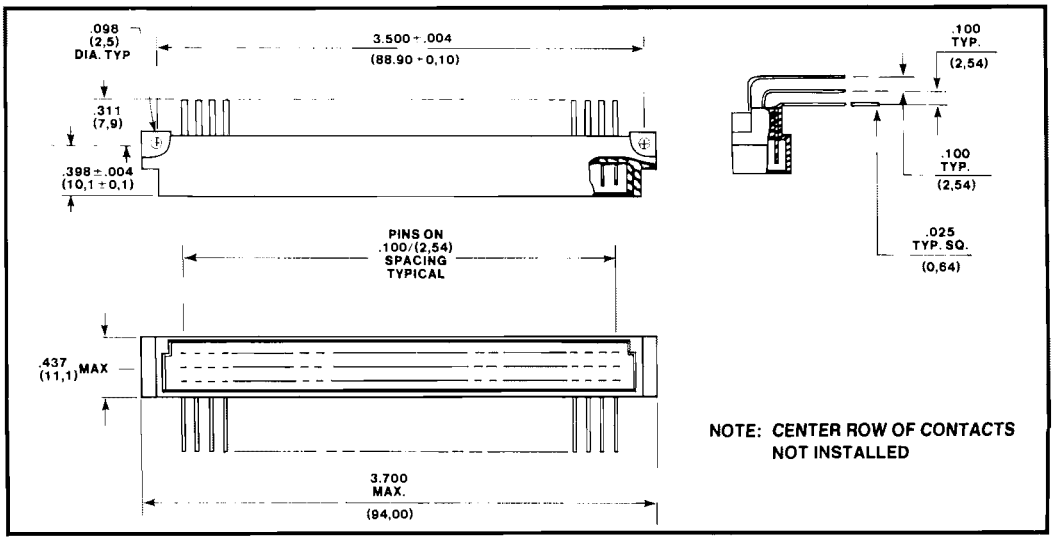


Figure 42. Backplane Connector

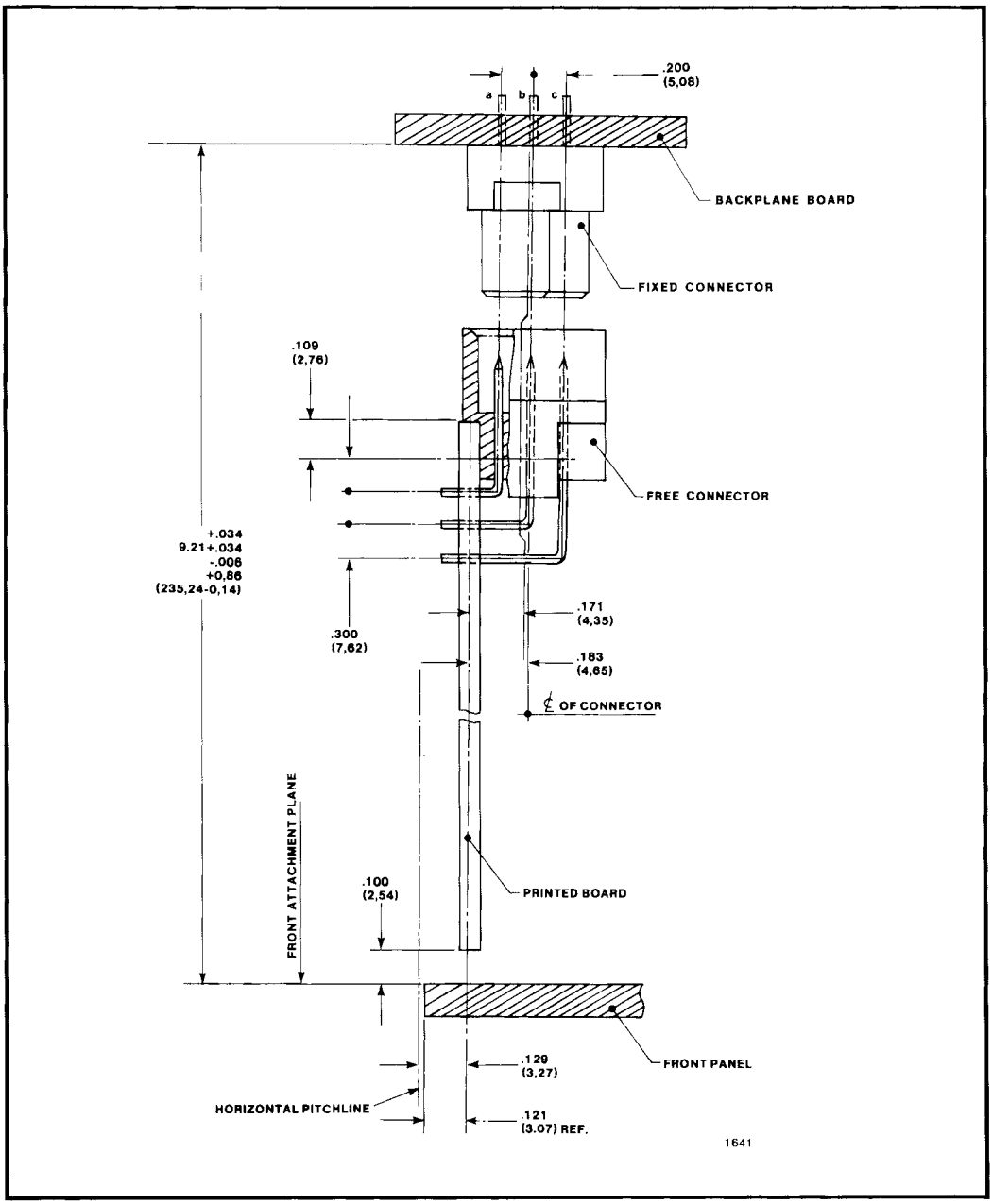


Figure 43. Relationship of Mechanical Components

7.2.3 PIN ASSIGNMENTS

The standard BITBUS I/O board defines standard pin assignment for the 64 pin connector as shown in table 9. This pin assignment defines pins for the BITBUS interconnect as defined in section 2 of this specification, power and ground (1 amp per pin), I/O signals and signal ground returns. The I/O signal lines are defined as required by each implementation.

Table 9. I/O Board Pin Assignment

PIN #	SIGNAL	PIN #	SIGNAL
1a	GND	1c	GND
2a	+5V	2c	+5V
3a	DATA	3c	DATA*
4a	DLCK/RTS	4c	DLCK*/RTS*
5a	I/O	5c	RGND
6a	I/O	6c	I/O
7a	I/O	7c	GND
8a	I/O	8c	I/O
9a	I/O	9c	GND
10a	I/O	10c	I/O
11a	I/O	11c	GND
12a	I/O	12c	I/O
13a	I/O	13c	GND
14a	I/O	14c	I/O
15a	I/O	15c	GND
16a	I/O	16c	I/O
17a	I/O	17c	GND
18a	I/O	18c	I/O
19a	I/O	19c	GND
20a	I/O	20c	I/O
21a	I/O	21c	GND
22a	I/O	22c	I/O
23a	I/O	23c	GND
24a	I/O	24c	I/O
25a	I/O	25c	GND
26a	I/O	26c	I/O
27a	I/O	27c	GND
28a	I/O	28c	I/O
29a	I/O	29c	GND
30a	+12V	30c	−12V
31a	+5V	31c	+5V
32a	GND	32c	GND

8.0 LEVELS OF COMPLIANCE

The BITBUS interconnect specification contains many features that are required for compatibility and other features that are only required for additional capabilities. This section clarifies the various levels of compliance that may be supported by various implementations of the BITBUS interconnect. This information is included to allow use of the BITBUS interconnect products of varying capabilities manufactured by diverse vendors.

8.1 CONCEPT OF LEVEL OF COMPLIANCE

The concept of level of compliance is provided to guarantee compatibility of BITBUS interface products that support the specification at different levels. The basic rule is that all products are required

to support a minimum level of the specification. This results in a minimal level at which all BITBUS interconnect products can interact. Furthermore, enhancements beyond this minimal level are controlled. This results in a guaranteed compatibility between a group of products that all support the same enhanced capabilities.

8.2 COMPLIANCE LEVELS

This section presents the various levels of compliance for each level of the specification. Each capability supported beyond the minimum shall be documented in the product data sheet.

8.2.1 ELECTRICAL INTERFACE

The electrical interface supports variability in bit rates, load characteristics, and repeater characteristics. The bit rates specified in section 2 include 2.4 mbits/sec synchronous and 375 kbits/sec and 62.5 kbits/sec self clocked. Of these only 375 kbits/sec is required. The others may be optionally supported. The load for a BITBUS node shall be specified in terms of the standard load defined in section 3.2.1. Repeaters are specified in terms of the standard repeater delays defined in section 3.3.5.

8.2.2 DATA LINK PROTOCOL

The data link protocol has no optional features. The full protocol shall be supported for compatibility.

8.2.3 MESSAGE PROTOCOL

The message protocol supports variability in message length. All implementation shall support the standard 5 byte header and 13 byte data field. Implementation may optionally support larger data fields.

8.2.4 REMOTE ACCESS AND CONTROL

Support of the remote access and control function is completely optional. The only requirement is that supported commands shall meet the specification.

8.2.5 MECHANICAL

The mechanical specification contains the option of supporting the standard I/O board specification. As with RAC, the only requirement is that if supported, it shall meet the specification.