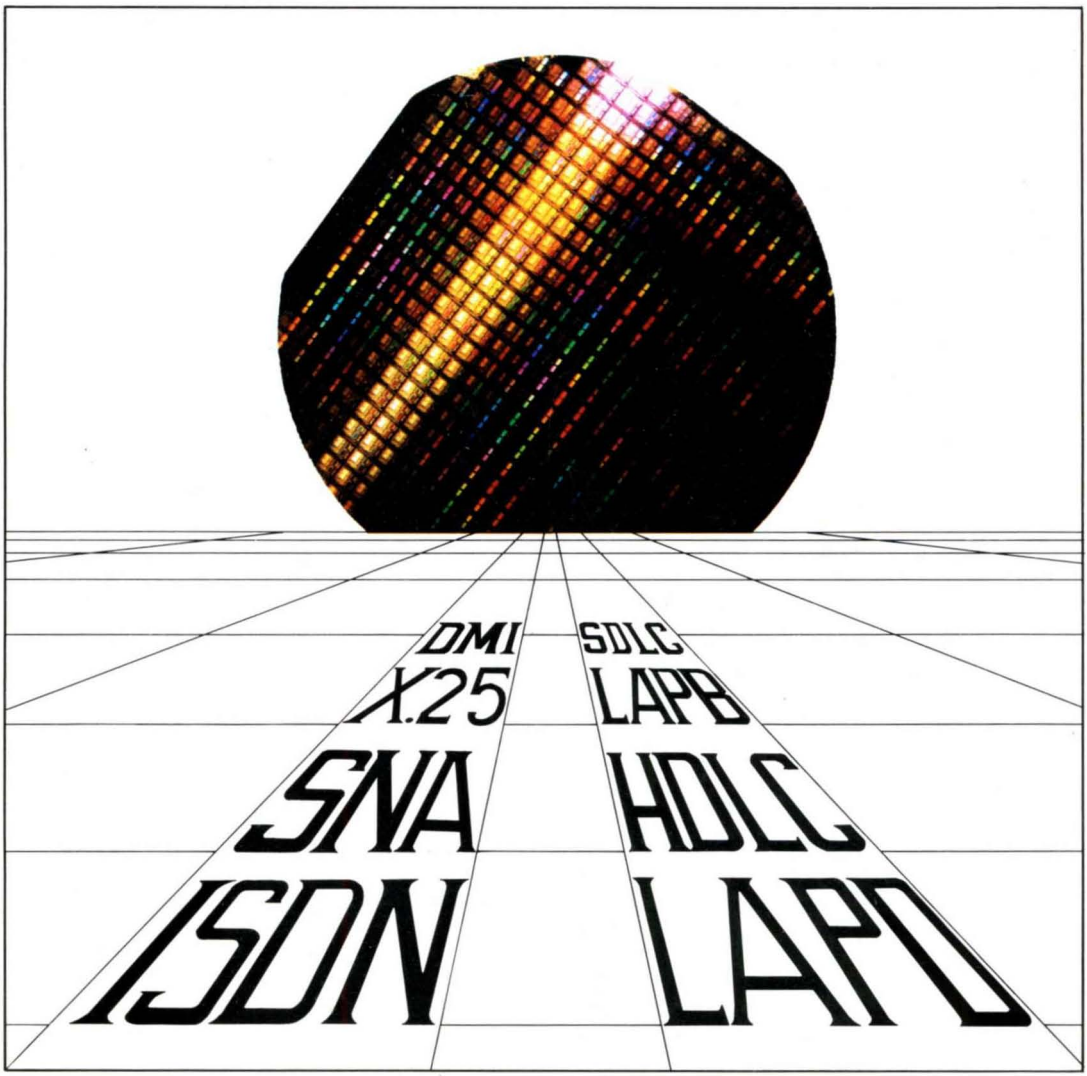
Am79C401
Integrated Data Protocol Controller

Technical Manual

Advanced
Micro
Devices



Am79C401
Integrated Data Protocol Controller

Technical Manual

901 Thompson Place, P. O. Box 3453, Sunnyvale, California 94088-3000
(408) 732-2400 TWX: 910-339-9280 TELEX: 34-6306

TABLE OF CONTENTS

CHAPTER 3. APPLICATIONS

CHAPTER 4. PROGRAMMING THE IDPC

Chapter 1

INTRODUCTION

The Am79C401 Technical Manual provides information to the user concerning the operation and programming of the major functional modules contained in the Am79C401 Integrated Data Protocol Controller (IDPC). This manual is divided into five chapters plus an appendix.

Introduction—This chapter provides an overview of the IDPC, concentrating on how it fits into various system architectures.

Hardware—The Hardware chapter covers the specifics of each major functional block, emphasizing how each block is controlled, and the operation and generation of external interface signals.

Applications—This chapter provides detailed design examples, concentrating on the IDPC's external interfaces.

Programming the IDPC—The Programming chapter contains three sections. The first section introduces the programmable features of the IDPC. The second section provides a series of tutorials on system-level operation. The final section provides detailed programming examples for the major functional blocks.

Low-Level Device Driver—This chapter contains the Reference Guide for the Am79LLD401 Low-Level Device Driver. This software interfaces the IDPC to the AmLink™ LAPD/LAPB software package. The source code, with an unlimited binary distribution license for both software packages, may be purchased from AMD for a nominal one-time fee.

Appendix—The Appendix contains pin definitions and user accessible register descriptions for the Am79C401 IDPC.

INTRODUCTION TO THE Am79C401 IDPC

When designing equipment for packet data networks, designers are concerned with performance, flexibility and cost. The Am79C401 Integrated Data Protocol Controller (IDPC) from Advanced Micro Devices addresses these three concerns by integrating three key building blocks into a single integrated circuit. As shown in Figure 1-1, the Am79C401 IDPC consists of three major blocks: the Data Link Controller (DLC), the Universal Synchronous/Asynchronous Receiver/Transmitter (USART), and the Dual-Port Memory Controller (DPMC). The DLC is the heart of the Am79C401 IDPC, responsible for processing bit

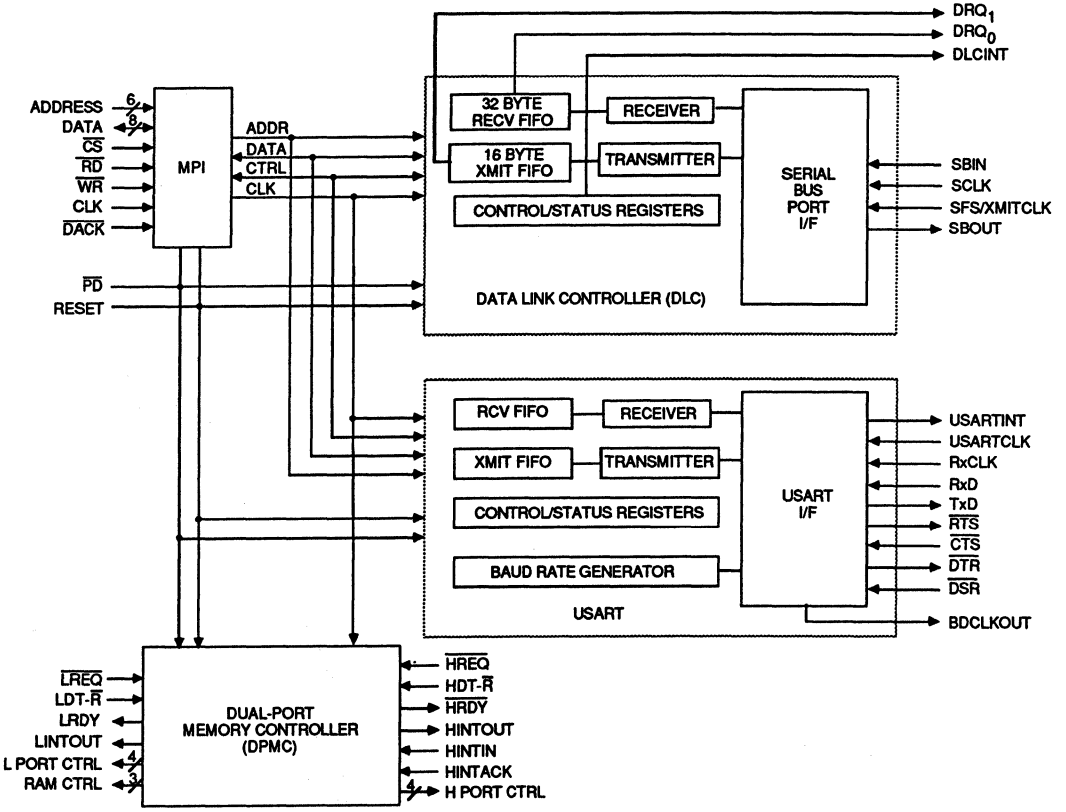


Figure 1-1. IDPC Block Diagram

AmLink is a trade mark of Advanced Micro Devices, Inc.

oriented protocols such as the HDLC and its derivatives SDLC, LAPB, and LAPD. The USART block is a super-set of the industry standard 8250 UART. The USART is useful in building terminal adaptor devices that connect existing terminals to Bit-Oriented Protocol based networks such as X.25, ISDN, and SNA. The Dual-Port Memory Controller (DPMC) provides the circuitry required to convert inexpensive static RAM into dual-port memory. Dual-port memory is required whenever the communications processor shares the system bus with a host processor. The dual-port memory allows messages and data to be passed between the two processors. An example of this type of application is an X.25 network interface, installed in a personal computer. By integrating these basic building blocks into a single integrated circuit, the cost of building data communications products is significantly reduced.

Data Link Controller

Designers of data communication equipment supporting bit-oriented protocols such as HDLC, SDLC, LAPB, and LAPD have had to choose between performance and flexibility. The protocol controller integrated circuits currently on the market offer either performance or flexibility, but not both. The reason for this is that in order for one protocol controller IC to have the flexibility to handle all of the various protocols, the higher layers of the protocols must be handled by software running on an associated microprocessor. Handling part of the protocol processing in software can substantially degrade performance, as measured by over-all throughput. In an attempt to increase performance, some IC manufacturers have designed IC's that are dedicated to one specific protocol, performing most, if not all, of the protocol processing in hardware. While this improves throughput, the flexibility to handle multiple protocols, or even slight variations in the target protocol, is sacrificed. In the development of the Am79C401 IDPC, Advanced Micro Devices has taken a different approach to the problem. Three key elements of this approach are the careful partitioning of the tasks to be performed in hardware versus software, separating the movement of data from the processing of packets, and the optimization of the hardware/software interface.

The Hardware/Software Boundary—Careful consideration is required in determining which of the bit-oriented protocol functions to perform in hardware versus software. Many of the functions, such as flag and abort detection, zero bit insertion, and CRC generation and checking, are best performed in hardware. Other functions are best left to the software, including: sequence number checking, transmission of acknowledgment packets, and re-transmission of non-acknowledged packets. There are two reasons for handling these functions in software: 1) each of the various protocols handles these functions in a slightly different manner, and 2) the amount of hardware required increases prohibitively as the window size increases. (Window size refers to the number of packets one transmitter can send out before an acknowledgment is received regarding the first packet sent. For example, if the window size is four, the transmitter can send four packets, then it must stop transmitting until an acknowledgment is received for the first packet. This requires the transmitter to provide hardware to store a history of all outstanding packets—an expensive proposition considering window sizes of eight or more are not uncommon.) Most protocol controllers that are designed to process multiple protocols divide the above mentioned tasks between hardware and software in a similar fashion. The DLC in the Am79C401 IDPC provides hardware support for several additional functions that are often delegated to software. Two examples of these are: minimum packet size checking and maximum packet size checking. By handling these tasks in hardware, the software does not need to perform a bounds check every time a new byte of data is received. Aside from the obvious advantage of reduced software overhead, this also allows the software to be partitioned into two separate functions: data movement, and packet processing. As you will see later, this is the key to providing high performance, while retaining the flexibility to handle multiple protocols.

Separating Data Movement From Packet Processing—The key factor in determining overall throughput is the rate at which packets can be processed. For reasons of flexibility, this processing takes place in software (dedicated hardware is faster, but expensive and inflexible). In the International Standards Organization's Open Systems Interconnection (ISO-OSI) seven layer model, the bit-oriented protocol resides at layer 2. Layer 2 is given unpacketed data from layer 3, which it packets and transmits, via layer 1. On the receive side, packets are received, verified, then returned to a non-packet format, and passed up to layer 3. While this is an over simplified view of a fairly complex process, it does point out a key fact: the layer 2 software deals with packets, not the movement of individual bytes of data. If the software can be partitioned such that it deals only with completely received packets of data, not bits and pieces of packets, the time spent processing the packet can be substantially reduced.

The DLC in the IDPC has been designed to completely separate the software involved in data movement from the software responsible for packet processing. One piece of software is responsible for the movement of data, often via DMA, while a separate software module processes the status information regarding complete packets. In the IDPC, a special status reporting mechanism has been designed that stores the status information concerning a packet, and reports it to the packet processing software after the entire packet has been received, moved through the 32 byte receive FIFO, and stored in an off-chip buffer. Up until the time that an entire packet has been received, moved from the receive FIFO, and placed in buffer memory, the packet processing software is uninvolved. In fact, the DLC does not notify the software about the packet until the last byte of the received packet has been placed in off-chip memory. At this time, the DLC notifies the software that a packet has been received, and provides the appropriate status information concerning the packet. In this manner, the packet processing software is presented with a complete packet of data and status information pertaining to that packet at the same time. If the status information indicates that the packet has been received without errors, it is acknowledged, and the data are passed to the user. If the packet contained errors, or was aborted, all that is required is to re-assign the buffer location in memory. The DLC can store status information for up to four previously received packets before the microprocessor has to read the status from the first packet. This greatly increases the maximum allowed interrupt latency.

Optimization Of The Hardware/Software Interface—The percentage of the processor's time that must be spent interfacing to the DLC is critical to performance. Three

major factors affect this overhead: time spent identifying which register contains the pertinent information, time spent accessing that register, and time spent locating the desired information within the register. The time required to identify the register containing the condition that caused the interrupt is based on the efficiency of the interrupt reporting structure. In the DLC, the source of an interrupt is reported via the Interrupt Source Register. This register contains bits that directly point to status registers that can generate interrupts. Additionally, the actual status information for the two most common interrupt generating events is reported directly in the interrupt source register. These two conditions, which comprise 95% of all interrupts, are the valid packet received and valid packet transmitted indicators. Once the source of an interrupt has been identified, the appropriate status register must be read. The time required to read the register can be cut in half if the register is directly mapped into the processor's address space, as opposed to indirectly accessed via a pointer register. In the DLC, all registers are directly memory mapped. The third factor contributing to the efficiency of the software interface is the time required to find information once a register has been read. The key to reducing this time is to organize the individual registers such that the most often required information is in either the least significant or most significant bit locations. Once a status register is read, the software typically performs a test, shift, test routine until it finds a bit that is set. The time spent finding the set bit depends on the number of shift/tests required. If the most frequent conditions are indicated by bits closest to one end of the register, the performance can be more than doubled.

Both flexibility and performance are attained by optimizing the partition between software and hardware, separating the movement of data from the processing of packet status, and optimizing the interface between software and hardware resident status registers.

USART

The function served by the USART depends on the application of the IDPC. If the IDPC is embedded in a terminal or host computer (such as a PC), the USART provides a second serial channel, separate from the DLC. In terminal adaptor applications, existing terminals that are not "network-ready" are interfaced to networks such as SNA, X.25, or ISDN. In this case, the USART provides the connection to the terminal, while the DLC provides the network interface.

The USART is a super-set of the industry standard 8250 UART. The 8250 UART provides basic asynchronous RS-232 serial data communication service including baud rate generation, and is the standard UART used in the IBM PC and its compatibles. The IDPC USART starts with this base, and adds three features:

- **Four-Byte Transmit and Receive FIFO Buffers**—The FIFOs increase software performance by reducing the number of interrupts that must be serviced, and increasing the time allowed to respond to an interrupt.

- **Special Character Recognition**—It is normal practice to embed control characters into the serial data stream between a computer and a terminal or printer. This requires the software to inspect each received character to determine if it is a control character, resulting in substantial overhead. The special character recognition hardware in the USART performs this function automatically, eliminating the software overhead. The user can designate up to 128 separate characters as being "special." Whenever a designated character is received, a maskable interrupt is generated to notify the user.

- **Synchronous/Transparent Mode**—In terminal adaptor applications, it is often desirable to place data onto the network exactly as they are received from the terminal, with all framing bits included. This is referred to as a transparent channel. The USART synchronous/transparent mode provides this transparent channel by blindly receiving data in eight bit portions. On every cycle of the receive clock, a data bit is received into the USART, including framing bits and idle bits. When eight bits have been received, they are loaded into the FIFO. After several of these eight bit portions of data have been received, they can be combined into a packet and transmitted over a network via the DLC. On the other side of the network, the receiving DLC processes the packet and places the field containing the data into memory. The USART on the receiving end can then retransmit the eight bit blocks of data, without the addition of framing bits. The result is a data stream that is identical to that received by the original USART, allowing any protocol to be transmitted over the network.

Dual-Port Memory Controller

When packet network hardware is built into a computer, such as a card installed in a PC, it is desirable to use a dedicated microprocessor to perform the communication tasks. This reduces the overhead placed on the host system's microprocessor by the communication functions. Normally, layers 1, 2, and 3 of the ISO-OSI model will run on the communication processor, while layers 4 and above will be handled by the host processor.

With the software functions divided between two processors, a communication mechanism is required that allows commands and data to be passed back and forth. The most straightforward vehicle is a shared memory interface, with a means for each processor to alert the other. To implement the shared memory, a section of each processor's memory must be common, and thus accessible by both processors, for example a dual-port RAM. A means is required to allow one processor to indicate to the other that a message (command or data) is available. A system of interprocessor interrupts provides this function. The shared memory is divided into buffer spaces and a set of mailboxes. The buffers are used to pass data to be transmitted and data that have been received back and forth between the host and the communications processor. The mailboxes are used for passing commands and status. When one processor has either a command or some status information for the other processor, it is placed in the appropriate mailbox. The sending processor then generates an interrupt to the other processor. The receiving processor responds by reading the mailbox and clearing the interrupt.

The IDPC's Dual-Port Memory Controller (DPMC) provides the support hardware to build a low-cost shared memory interface. The DPMC's bus arbitration unit allows low-cost static RAM to be used as dual-port memory.

Hardware is provided for implementation of the interprocessor interrupt system.

The DPMC performs the memory bus access arbitration between the communications and the host processors. Each processor accesses the RAM as if it were the RAM's sole owner transparent to software. The DPMC generates the RAM cycle timing, and outputs the appropriate chip select, output enable and write enable signals. In the event of conflicting access requests, the DPMC holds off one of the processors for one memory cycle time, by deactivating that processor's Ready signal. The interconnection between the RAM, the host's system bus, and the communication processor's address/data bus, is made via the bus interface blocks, (see Figure 1-2). These blocks consist of buffers and latches that control the flow of addresses and data between the two processors' address and data busses and the RAM. The DPMC generates the control signals for the bus interface blocks.

The DPMC also provides hardware support for the interprocessor interrupt structure. The communications processor can generate an interrupt to the host processor by setting a bit in a register located in the IDPC. The setting of this bit drives an IDPC pin (HINTOUT), which is connected to an interrupt request line to the host processor. The host processor can clear the interrupt request by pulsing a pin on the IDPC. The host can generate an interrupt request to the communications processor by pulsing another pin on the IDPC. The communications processor clears this interrupt request by writing to a register in the IDPC.

Applications

The IDPC provides the building blocks necessary to build terminal adaptors and embedded communications processors. Additionally, the IDPC can be used in applications requiring separate synchronous and asynchronous communication channels. Whether the network is SNA, X.25, ISDN, or any other bit-oriented protocol based network, two types of devices are needed: terminal adaptors that allow non-network compatible equipment to be interfaced to the network, and second, embedded communications processors which integrate the network interface directly into the computer or terminal.

Terminal Adaptor

The terminal adaptor is a self-contained device that allows non-network equipped terminals, or computers, to be connected to a network. Figure 1-3 shows the block diagram of a terminal adaptor, including: a transceiver, providing the physical layer 1 connection to the network; an HDLC protocol controller; a USART, providing the terminal interface; and a microprocessor, with RAM and ROM, to process both user data, and call control. The HDLC protocol controller and the USART are provided by the Am79C401 IDPC. In this example, an 80188 microprocessor provides the processing power.

Embedded Communication Processor

When the network interface is built into the computer or terminal, the communication processor is connected directly to the host's system bus. Figure1-4 shows the block diagram of an embedded communication processor. The DLC in the Am79C401 IDPC provides HDLC packet protocol processing for the network. The Dual-Port Memory Controller supports a shared memory interface to the host processor. In this example, the network software runs on the 80188 microprocessor.

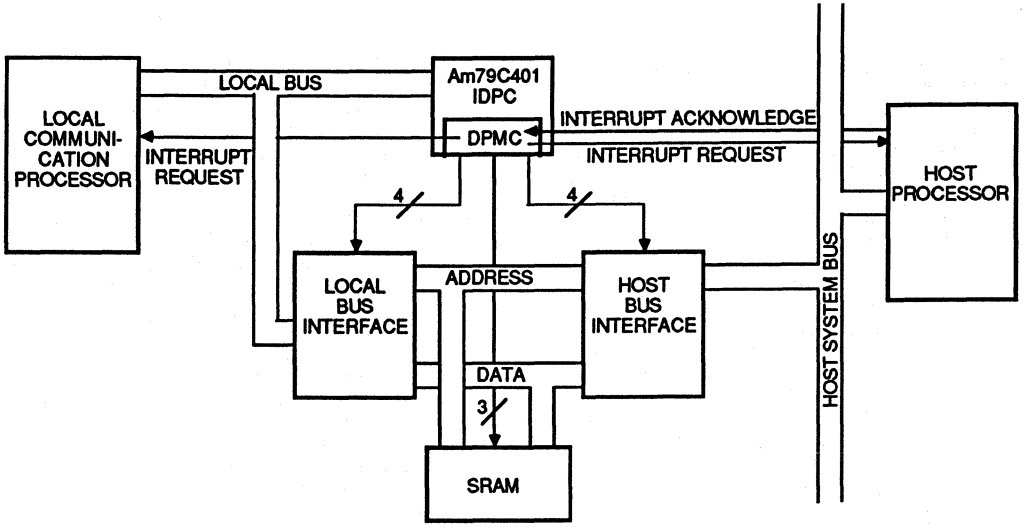


Figure 1-2. DPMC Memory System Interconnection

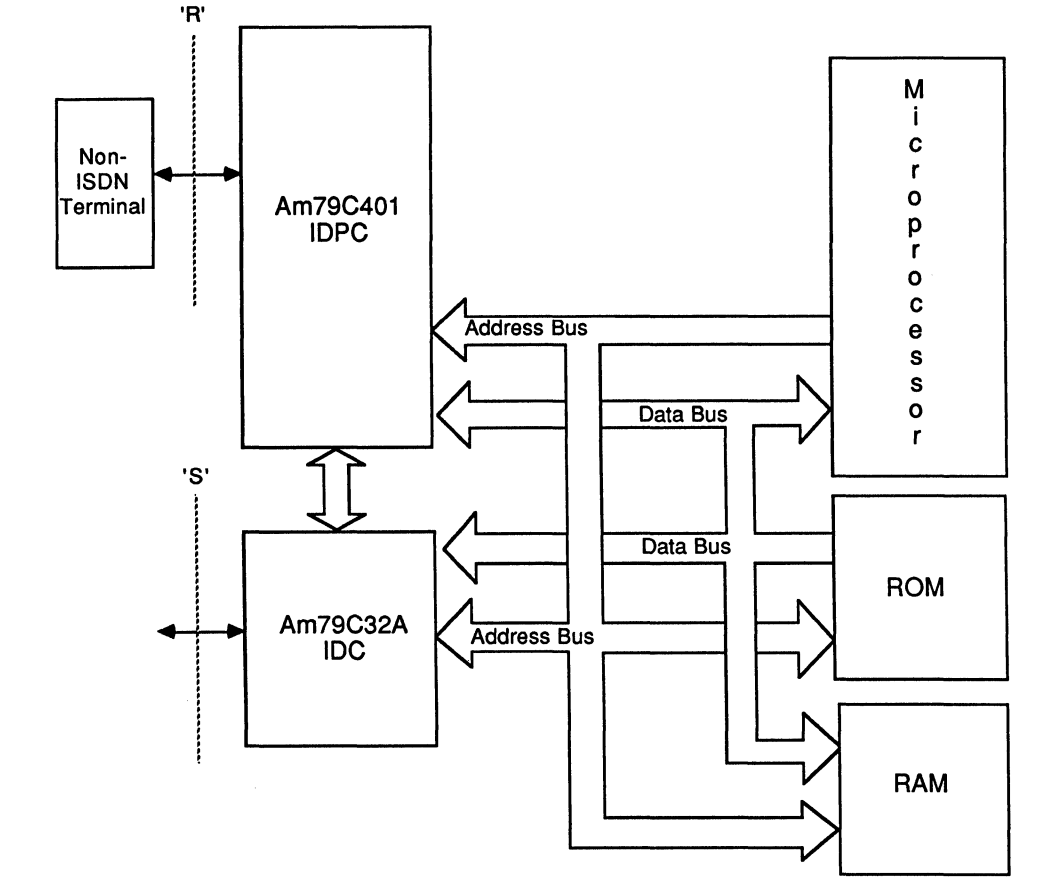


Figure 1-3. ISDN Term Adaptor Application

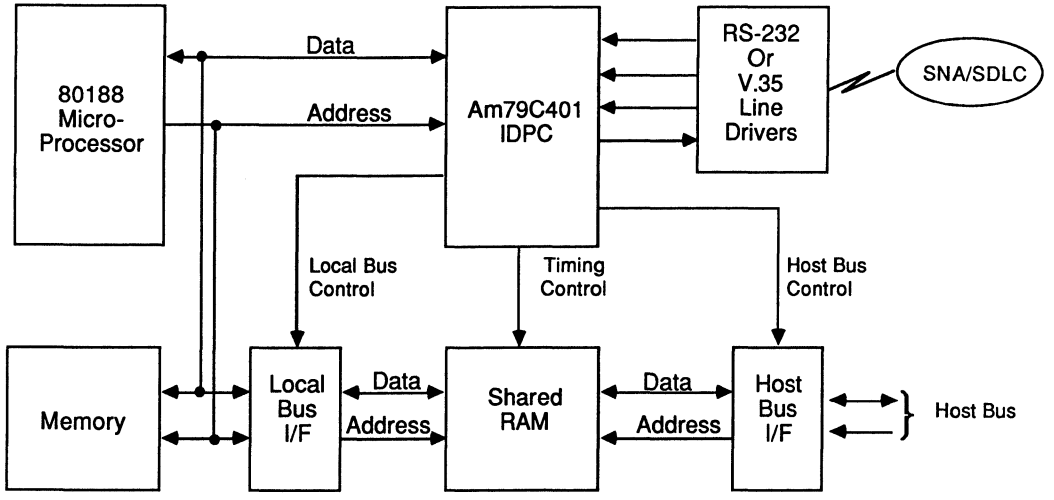


Figure 1-4. Typical SNA Application

Chapter 2

HARDWARE

DATA LINK CONTROLLER

OVERVIEW

The DLC portion of the IDPC has the task of providing a full-duplex interface, simultaneous transmit and receive, between the Serial Bus Port (SBP) and the internal parallel bus of the IDPC. Through the use of a 32-byte receive FIFO, a 16-byte transmit FIFO and optionally, two external DMA channels, the DLC provides transparent movement of data to and from external memory and the SBP. The DLC performs low-level (ISO Layer 2-) bit oriented protocol processing on this data. The major protocols supported are SDLC, HDLC, LAPB (X.25), and LAPD. Figure 2-1 shows the major functional blocks of the DLC in relation to the rest of the IDPC. The DLC and FIFOs sit on the Main Internal Bus. All programmable registers and the FIFO data registers can be accessed via the bus. These registers are mapped directly into the CPU's memory space.

The next section provides an overview of bit-oriented protocols. It is recommended that the reader at least scan over this section to insure a common vocabulary. Following this overview, the DLC Transmitter and Receiver will be discussed in detail. The FIFOs and Serial Bus Port will also be discussed in the transmitter and receiver sections since they share a common control structure.

Overview Of Bit-Oriented Protocol Processing

This discussion deals with the general characteristics of Bit-Oriented Protocols (BOP) as well as identifying the specific variations between the major BOPs that the DLC is designed to handle. The major BOPs are SDLC, HDLC, LAPB (X.25), and LAPD.

Bit-oriented protocols provide a set of rules and techniques that facilitate the transfer of data over a communications network. For the purposes of this discussion we will not be concerned with the workings of the upper level of the protocols—sequence numbers, acknowledges, and the like—since these are the responsibility of the software that runs on the local CPU. We will concentrate on the aspects of the protocols that affect the hardware of the DLC.

The BOPs transmit data in chunks called packets. These packets are delimited by unique flag characters and contain an address, some control information, the data itself, and an error detection code. The address identifies the sender or the receiver of the data. The control information is used by higher levels of the protocol to manage the flow of data. The data, which are contained in the information field, are user information. Packets that are used for protocol control often omit the information field; this is the only optional field. The error detection code is a Cyclical Redundancy Check (CRC). (The DLC uses the CCITT-CRC code.) In addition to addresses, control, data, and error checking, the BOPs employ such mechanisms as flags, bit stuffing, and abort characters. The following section is a glossary of BOP terms and functions. These items will be used throughout the description of the DLC.

General Terminology

Frame—In the bit-oriented protocol environment, data are transmitted in frames. Protocols such as SDLC, HDLC, LAPB (X.25), and LAPD share the same basic frame format, shown below.

OPENING FLAG 01111110	ADDRESS (1 or 2 bytes)	CONTROL (1 or 2 bytes)	INFO (OPTIONAL)	FRAME CHECK SEQUENCE (16 bits)	CLOSING FLAG 01111110

Flag (General)—The eight-bit flag character is identical for all of the above mentioned protocols. It is exactly

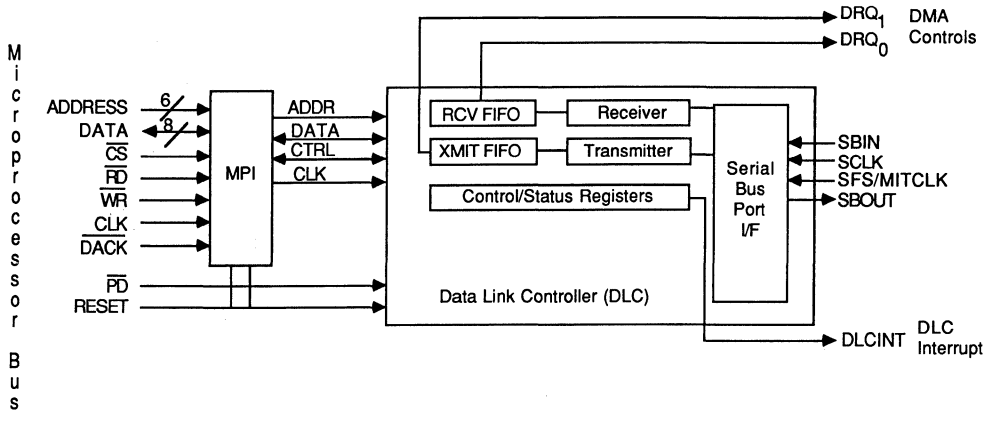


Figure 2-1. DLC Block Diagram

01111110. Its bit pattern is unique within a packet because the zero-bit insertion technique used, described later, does not allow six contiguous ONEs to be present in the packet portion of a frame. The flag character can perform three functions: opening flag, closing flag, and inter-packet fill character.

Opening Flag—The opening flag is defined as the last, perhaps only, flag prior to a non-flag, non-abort character. (The abort character is defined below.) All valid packets must begin with a flag. The opening flag indicates the beginning of a packet. When flags are being used as inter-frame fill characters, a non-flag, non-abort character must be received before the preceding flag can be identified as an opening flag.

Address—The principal difference between the lower levels of the various BOPs is the address field. All addresses are of an integer number of bytes in length. In general, an address can be one, two, or N bytes long.

7	6	5	4	3	2	1 C/R* 0/1	0 EA* 0	BYTE 1
7	6	5	4	3	2	1	0 EA* 0	BYTE 2
7	6	5	4	3	2	1	0 EA* 1	BYTE N

* C/R, EA bits, described below, are not used in all BOPs (SDLC for example). These bit positions are treated as normal address bits for these protocols.

The length of an N byte long address is determined by the value of the least significant bit in each byte of the address. This bit, called the Extended Address (EA) bit, identifies the last byte of the address. All of the bytes of an N byte long address will have the EA bit cleared to a ZERO except the last byte of the address. The presence of an EA bit set to a ONE indicates that that byte is the last byte of the address. The length of the Address field affects the detection of a short frame (refer to short frame definition).

In some protocols the second bit (bit 1) of the first byte of the address is used to indicate whether the frame is a command or a response. This bit, called the Command/Response bit (C/R), can be either a ONE or a ZERO without invalidating the address.

Control Field—The control field immediately follows the address field. The DLC treats the control field as packet data. That is, the DLC does not take any action in response to the contents of the control field. The control field can be either one or two bytes long. The length of the control field has an impact on the detection of a short frame.

Information Field—When present, the Information field, follows the control field and precedes the frame check sequence. The information field contains the data that are being transmitted between users. The information field can be up to 64K bytes long (minus address and control field lengths) in the IDPC.

Frame Check Sequence—The Frame Check Sequence (FCS) is a 16-bit word that is produced by the CRC generator and checked by the CRC checker. Mathematically, it is the ONEs complement of the sum [modulo 2] of the following:

The remainder of

$$X^K [X^{15} + X^{14} + X^{13} + \ldots + X^2 + X + 1]$$

divided [modulo 2] by the generator polynomial

$$X^{16} + X^{12} + X^5 + 1,$$

where K is the number of bits in the frame existing between, but not including, the final bit of the opening flag and the first bit of the FCS, excluding bits inserted for transparency.

-AND-

The remainder after multiplication by X^{16} and then division [modulo 2] by the generator polynomial

$$X^{16} + X^{12} + X^5 + 1,$$

of the content of the frame, between but not including the last bit of the opening flag and the first bit of the FCS, excluding bits inserted for transparency.

Refer to CCITT Recommendation X.25, paragraph 2.2.7.

Closing Flag—The closing flag is the last field in the frame. It indicates the end of the frame and signals that the FCS should be checked.

Packet—A packet is a frame minus the opening and closing flags.

Mark Idle—When frames are not being transmitted over the link, the link is said to be Idle. When the link is idle, the transmitter can be programmed to send an all ONEs pattern (refer to the "DLC Programming" section in Chapter 4). This is referred to as a Mark Idle (MI) condition. Specifically, an MI is defined as being at least 15 contiguous ONEs.

Flag Idle—Prior to and between frames, back to back flags can be transmitted over the link. This is referred to as a Flag Idle (FI) condition and is selected by program control (refer to the "DLC Programming" section in Chapter 4).

In-Frame—The DLC receiver is said to be in-frame when the first non-flag, non-abort character is received after the receipt of at least one flag. In-frame is valid until the closing flag is detected, an abort character is received or an error is detected. The DLC transmitter is said to be in-frame from the time that it starts to send an opening flag until the last bit of the closing flag has been transmitted, assuming that the transmitter is not commanded to send an abort sequence.

Out-Of-Frame—The DLC receiver or transmitter is said to be out-of-frame any time it is not in-frame.

Abort Character—Any pattern of at least seven contiguous ONE bits is said to be an abort character. An abort

character is a physical entity, not to be confused with the abort condition, which is an action. The abort condition, simply called an abort, is described below. It is important to note that there is a subtle difference between an abort character and a mark idle condition. Back to back, abort characters do not necessarily constitute a mark idle condition. A repeating pattern of seven ONEs followed by a ZERO (01111111011111101111111 . . .) is a series of abort characters, but not a mark idle. The DLC sends at least one "01111111" when commanded to send an abort.

Abort—The abort condition is an action that takes place in response to the detection of an abort character while the DLC receiver is in-frame. An abort causes the termination and discarding of the packet being received. Aborts are asynchronous events in that they can be detected on bit boundaries as well as byte boundaries.

Transparency (Zero-Bit Insertion/Deletion)—Zero-bit insertion/deletion, often referred to as bit stuffing, is a technique used to provide data transparency. By this we mean a method by which packet data patterns are prevented from appearing as flags, aborts, or mark idles when they appear in the received data stream. Flags, aborts, and the mark idle condition all consist of six or more contiguous ONE bits. The bit stuffing technique examines the contents of a packet to be transmitted, on a bit by bit basis, from the first bit after the opening flag to the last bit of the FCS, and inserts a ZERO in the bit stream after any pattern of five contiguous ONEs, thus insuring that six or more ONEs do not appear in the data stream. The receiver, in turn, examines the data stream and removes the inserted ZEROs that follow five contiguous ONE bits. The implication of this is that flag, abort, and mark idle generation and detection must take place on the network side of the zero insertion and deletion units.

Short Frame—The BOPs specify minimum lengths for valid packets. This is usually four, five, or six bytes. Any frame that is received with fewer than this legal minimum number of bytes in its packet is called a short frame and is considered an error which should be discarded.

Long Frame—On a theoretical basis, a frame can be any length greater than the specified minimum. As a practical matter, however, a maximum packet length must be set to prevent buffer overrun. This length is dynamic and can vary on a data call-by-data-call basis. Any received frame whose packet exceeds this maximum length is referred to as a long frame, and is considered an error. Note that the detection of a long frame error takes place as soon as the maximum legal number of bytes has been exceeded, not when the entire frame has been received.

Non-Integer Number of Bytes Received—If a closing flag is detected and a Non-Integer Number of Bytes has been received, that is to say that the character preceding the flag contained fewer than eight bits, a non-integer number of bytes condition exists. Some protocols allow this condition as a normal mode of operation—it is referred to as bit residue. Other protocols consider this condition an error.

Order of Bit Transmission—The bytes are transmitted in ascending numerical order; inside a byte, the least significant bit (bit 0) is transmitted first. (NOTE: The FCS is numbered and transmitted in reverse to this convention.)

Transmitter

The transmitter portion of the DLC resides between the off-chip memory and the data communications network (the transmitter includes the FIFO and Serial Bus Port). The CPU, under software control, builds a data block in memory that contains the address, control, and information fields of a packet. This block of data is moved, byte at a time, into the Transmit FIFO via DMA or programmed I/O. The transmitter sends the opening flag, transmits the block of data, generates and sends the FCS (if selected) and transmits the closing flag. If desired, the polarity of the data stream can be inverted as it is being transmitted. Between packets, the transmitter can be programmed to output an all ONEs pattern, mark idle, or back to back flags (flag Idle). The transmission of a packet can be terminated by sending an abort sequence in response to the send abort bit being set in the Command/Control Register (bit 0). Transmission of packets containing a non-integer number of bytes is supported by programming the Residual Bit Control/Status Register.

The remainder of this section is divided into two parts. The first part is a description of the transmitter's operation, including exception conditions—Figure 2-2 shows a state diagram of this operation. This is followed by a detailed description of the hardware blocks. See Figure 2-3.

Transmitter Operation

We will start the discussion of the transmitter operation by covering the normal flow of events. This will be followed by a section covering exceptions and error conditions.

NORMAL OPERATION—When the IDPC comes out of hardware reset, or is reset by the CPU (bit 6 of the DLC Command/Control Register), the transmitter is disabled, and is in state 0a—sending mark idle.

NOTE: When the transmitter is disabled the only action taken is to remove all drive from the SBOUT pin (open-drain). All other operations proceed in a normal fashion.

Initialization—The CPU initializes the transmitter by selecting data inversion or non-inversion (bit 0 of the Serial Bus Port (SBP) Control Register), selecting the SBP channel configuration (bits 5-1 in the SBP Control Register), selecting whether CRC generation is to be used, and selecting either flag or mark idle (bit 3 of the DLC Command/Control Register; the default is mark idle). The initialization sequence is detailed in the "DLC operational sequences" section of Chapter 4. The Transmit Byte Count Register is used to specify the length of the packet to be transmitted, excluding FCS bytes, and is programmed only when the packet length to be transmitted is different from the previous packet transmitted, or following an abort. Bytes are counted by a counter in the FIFO as they are placed into the FIFO's buffer (transmit byte counter). When the count equals the value programmed into the Transmit Byte Count Register, that byte is tagged as the last non-FCS byte in the packet. A more detailed description of this operation is presented in "DLC Programmable Operations" section of Chapter 4. Data inversion/non-inversion and SBP channel configuration do not affect the operational sequence of the transmitter. The SBP is described later in this chapter. The flag Idle/mark

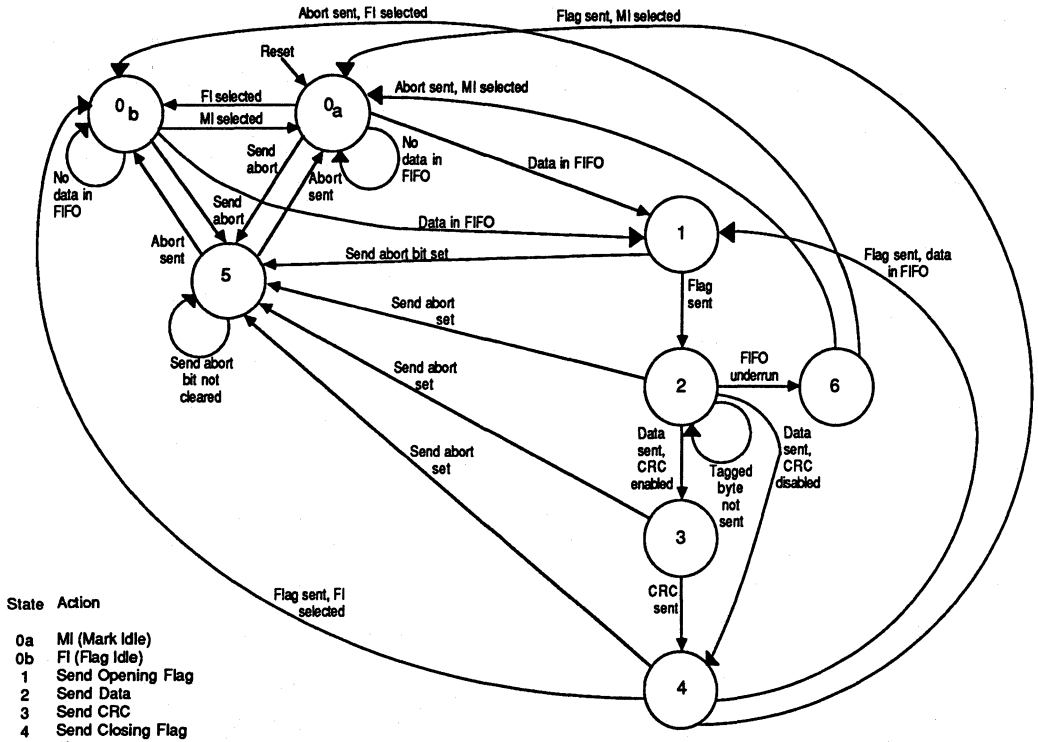


Figure 2-2. DLC Transmitter State Diagram

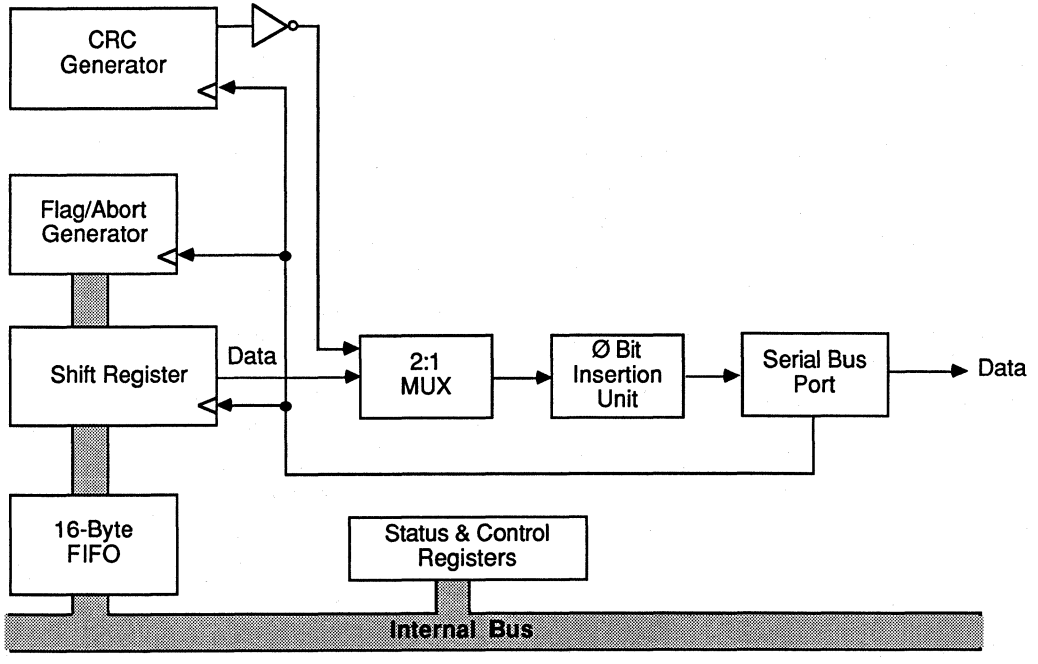


Figure 2-3. DLC Transmitter

idle selection does affect the operational sequence and is described below.

Operational Sequence—After the transmitter is reset (bit 6 of the DLC Command/Control Register, or hardware reset), the transmitter goes to state 0a. The transmitter will remain in state 0 until data have been placed in the FIFO. At that time the transmitter will go to state 1.

With the transition to state 1, the transmitter is said to be in-frame. In state 1 the transmitter sends the opening flag. When this flag has been sent, state 2 is entered.

While in state 2, data are unloaded from the FIFO into an eight-bit parallel-to-serial shift register. Serial data are clocked out of the shift register, through a 2-to-1 multiplexor, and into the zero-bit insertion unit. The zero-bit insertion unit inserts a ZERO bit into the data stream after any pattern of five contiguous ONE bits. The data are then fed into the Serial Bus Port (SBP) where they are optionally inverted and transmitted to the data communications network. The SBP can be programmed to transmit data on one of 31 multiplexed time slots, or to transmit data non-multiplexed. The transmitter leaves state 2 when the last byte of the packet up to the first FCS byte has been shifted out of the parallel-to-serial shift register.

If CRC generation has been selected (bit 5 of the DLC Command/Control Register) the transmitter will enter state 3. If CRC generation is disabled, state 4 is entered directly from state 2. In state 3, the contents of the CRC generator is fed to the zero-bit insertion unit following the original packet, now completed, data stream. After the 16 bits of the FCS have been transmitted, the valid packet sent bit is set (bit 4 in the Interrupt Source Register) and state 4 is entered. The valid packet sent indication can generate a maskable interrupt.

While in state 4, one flag character, the closing flag, is transmitted. The transmitter will transition to either state 0a or 0b when the transmission of the closing flag completes. If data are present in the FIFO, a new packet, state 1 is entered. If no data are present in the FIFO, state 0 is entered. The selection of the flag idle or mark idle inter-frame fill (bit 3 of the DLC Command/Control Register) selects between states 0a and 0b.

EXCEPTION TO NORMAL OPERATION—There are six exceptions to the normal flow of events described above: abort, local loop back, remote loop back, transmitter disabled while in-frame, FIFO Underrun, and Residual bit operation. Of these, only FIFO Underrun is an error condition.

1) Abort—The user can terminate the transmission of a packet by requesting that an abort be sent (bit 0 of the DLC Command/Control Register). When a send abort request is received the transmitter enters state 5 where the transmitter will begin transmitting abort characters (01111111 with 1 being the first bit sent). This action takes place on the next bit boundary after the Send abort bit is set by software; the Transmit FIFO, transmit byte counter, and Transmit Byte Count Register will be cleared. Abort characters will continue to be sent until this bit is cleared. The transmitter will go out of frame when transmission of the abort begins. When the Send abort bit is cleared the transmitter will enter state 0b if flag Idle is selected or data are present in the FIFO (a new packet); state 0a is entered otherwise. In all cases at least one abort character will be transmitted, even if the Send abort bit is set and cleared by consecutive CPU instructions. (The abort is used to tell the receiver on the other end of the link that the packet currently being received is to be terminated and discarded.) While sending an abort has no meaning when the transmitter is out of frame (not sending a packet), the request will be honored. It will have no meaning at the receive end if the receiver is out of frame.

2) Local Loop Back—For test purposes the DLC can be placed in a local loop back mode of operation (bit 6 of the SBP Control Register). In this mode the output of the transmitter is routed directly to the receiver. The receiver is disconnected from the SBIN pin to prevent incoming data from interfering with the Loop Back; the receiver enable bit must still be set. The transmit clock is used as the timing reference for both the transmitter and the receiver. Packets can then be transmitted normally. The receiver receives the packet just as if it were originating from outside the IDPC.

3) Remote Loop Back—Remote loop back, selected by setting bit 7 of the SBP Control Register, causes any activity on the SBIN input to the receiver to be echoed on the SBOUT output pin. The DLC transmitter is disconnected from the SBOUT pin. When the SBP is operating in multiplexed channel mode, each received bit, conditioned by SFS/XMITCLK, is transmitted on the next falling edge of the receive clock, i.e., data received at the SBIN pin on the rising edge of SCLK is clocked out of the SBOUT pin by the subsequent falling edge of SCLK. When the SBP is operating in the non-multiplexed mode, data bits received via SBIN (clocked in by the positive going edge of the receiver clock, SCLK), are clocked out on a bit-by-bit basis using the negative edge of the same clock (SCLK). The receiver can still receive data while in this state.

If an attempt is made to use the transmitter while in remote loop back mode, the transmitter will function normally, but no data will leave the IDPC.

4) Transmitter Disabled While In-Frame—*This is a legal operation*—The transmitter will continue to process the frame normally and will disable the SBOUT pin as soon as the closing flag has been sent. Once the closing flag is transmitted, the transmitter returns to state 0 and disconnects the SBOUT pin (places it in an open-drain condition with no ability to be driven Low).

5) FIFO Underrun—*This is an error condition*—A FIFO Underrun occurs when the transmitter attempts to unload a byte of data from an empty FIFO while in frame. This condition is reported via bit 4 of the FIFO Status Register and a maskable interrupt is generated. This causes the FIFO Status Register bit to be set in the Interrupt Source Register, if the underrun interrupt has been enabled in the FIFO Status Interrupt Enable Register. When the FIFO underrun is detected the transmitter enters state 6 where one abort character (01111111) is transmitted and the transmitter reenters state 0. The transmit byte counter and Transmit Byte Count Register are also cleared.

6) Transmission of Residual Bits—Some protocols require the transmission of packets containing a non-integer number of bytes (the number of bits in the Information field of the packet is not evenly divisible by eight). The DLC supports Bit Residue operation by allowing the user

to specify the number of valid bits in the last byte of the packet (prior to the FCS field). When the DLC transmits the last byte of the packet, only the specified number of bits is sent. The number of residual bits is specified in the Residual Bit Control/Status Register.

Transmitter Block Description

The hardware blocks of the transmitter will be discussed in the order that data flow through the unit, from FIFO to Serial Bus Port (refer to Figure 2-3). The transmitter supports data rates from DC to CLK divided by 5 (this is the theoretical maximum data rate; data rates in excess of those specified in the Am79C401 Data Sheet are not guaranteed).

FIFO

The Transmit FIFO consists of the FIFO buffer, the Transmit Byte Count Register, the transmit byte counter, and the DMA data request generation logic.

Buffer—The buffer is 16 bytes deep and nine bits wide (eight data bits plus one tag bit, the tag indicating the last byte of a transmit packet). Data are loaded into the buffer, FIFO Data Register, by the local processor, via PIO (Programmed I/O), or DMA. Data are unloaded from the buffer into the parallel-to-serial shift register. The buffer is cleared on reset, when an abort is transmitted, or when a Transmit FIFO underrun occurs. The tag is set by hardware and is used for internal housekeeping.

Threshold—Associated with the buffer is a Threshold reached signal. This signal is active whenever the number of bytes in the buffer is at or below the threshold level (programmed into the FIFO Threshold Register. The Threshold reached signal is used by the data request generation logic (DMA control), described below, as an indication that the buffer should be reloaded. The Threshold reached signal is reported in the FIFO Status Register, bit 2. A maskable interrupt is generated when the level in the FIFO falls to the threshold level. This is useful for program controlled transfers.

Data Register—The user addressable location of the FIFO is termed the Data Register. The buffer generates a status signal that reflects whether or not the Data Register is empty or available. This signal, buffer available, is reported in bit 3 of the FIFO Status Register. The bit is set anytime the Data Register is empty and the last byte of a packet is not in the FIFO. BA is cleared when the FIFO is full.

Underrun—If the parallel-to-serial shift register attempts to unload a byte from an empty buffer, an underrun condition exists. This causes an error to be reported via bit 4 of the FIFO Status Register. A maskable interrupt is generated by the setting of this bit. In response to the underrun, an abort is generated. This causes the Transmit Byte Count Register and the transmit byte counter to be reset to ZERO, as well as the FIFO to be cleared.

Transmit Byte Count Register—The Transmit Byte Count Register (TBCR) holds the length of the packet to be transmitted (exclusive of the opening flag, FCS, and closing flag). This value is loaded into the TBCR by software. The TBCR is cleared when the DLC is reset, an abort is transmitted, or a transmit FIFO underrun occurs. When the transmitter is out-of-frame, the content of the TBCR is loaded into the transmit byte counter at the same time it is written into the TBCR. The contents of the TBCR are automatically loaded into the transmit byte counter when the last byte of a packet (tagged as such) is removed from the FIFO buffer. (This also insures that the correct value is loaded into the TBC if the TBCR is updated while the transmitter is in-frame). This load is delayed if the TBCR is being written at this time as an internal reload.

Transmit Byte Counter—The Transmit Byte Counter (TBC) is used to count the number of bytes loaded into the buffer for a given packet. The TBC is loaded from the Transmit Byte Count Register, and decremented once for each byte loaded into the buffer. When the TBC reaches ZERO, the byte that caused the TBC to reach ZERO is tagged by hardware as the last byte of the packet. This tag is created by setting the ninth FIFO bit position of that byte to a ONE. The ninth bit position holds this tag, which travels with the last data byte through the buffer. The tag is used to load the TBC from the TBCR and indicate the end of a packet to the DLC.

DMA Request—The data request generation logic is used to generate the Data Request (DRQ_1) signal. When active, DRQ_1 indicates to the DMA that the buffer is available for the loading of data. The DRQ_1 signal is activated when the TBC is not ZERO -AND- the FIFO does not contain a tagged byte -AND- the level in the buffer is at or below the programmed threshold (bits 3-0 of the FIFO Threshold Register). DRQ_1 remains active until the TBC = 0 -OR- the buffer becomes full. When the level in the buffer falls to the threshold and there is more data in the packet to be loaded into the buffer, DRQ_1 will go active. DRQ1 will remain active until the buffer is completely full or the last byte of the packet is loaded into the buffer. This insures that there can never be data from more than one packet in the buffer at any one time since even if the TBCR is written before the last byte of the packet has been transmitted, DRQ_1 will remain inactive until the tagged byte is removed from the buffer. DRQ_1 is indirectly made inactive by reset, abort, and transmit FIFO underrun, since the TBC is cleared to ZERO by these conditions. DRQ_1 will become active as soon as the TBCR is written (becomes non-ZERO).

NOTE: Care must be taken to insure that DRQ_1, the transmitter DMA request line, is deactivated early enough to prevent the transfer of one to many bytes of data. This can occur because the DMA controller does not write the last byte of data into the transmit FIFO until the second half of the DMA cycle. Data are read from RAM during the first half cycle, and deposited into the Transmit FIFO during the second half cycle, leaving little time for the DLC to deactivate DRQ_1. This problem can be prevented in two ways: 1) use of the DMA acknowledge output from the DMA controller—connected to the DACK/ pin on the IDPC, the DMA acknowledge signal is activated at the beginning of the DMA cycle, allowing time for the DLC to deactivate DRQ_1. 2) adding a wait-state to the DMA cycle. If the DMA controller does not provide an acknowledge output, or one cannot be generated, a wait-state can be inserted to provide more time prior to the DMA controller sampling DRQ_1.

The DLC will deactivate DRQ_1 during the last cycle when either the DACK/ pin is activated, or when the WR/ and CS/ pins become active.

The receiver does not have this problem since data are read from the receive FIFO during the first half of the DMA cycle. In this case, the Receive DMA request line, DRQ_0, is deactivated during the last cycle when the RD/ and CS/ become active.

8-Bit Parallel-to-Serial Shift Register

Data to be transmitted are moved out of the FIFO and loaded into an 8-bit shift register, one byte at a time. This byte is shifted out of the shift register serially. The shift register output is fed to the CRC generator and to a 2-to-1 multiplexor.

If the FIFO buffer becomes empty before the last byte of the packet has been loaded into the shift register, an underrun error is reported, (see FIFO description for error handling details). An underrun causes an abort to be transmitted, the FIFO to be flushed, and the TBCR and TBC to be set to zero.

CRC Generator

The CRC generator produces a 16-bit word referred to as the Frame Check Sequence (FCS). The mathematical equation describing this operation is provided in the review of BOPs at the beginning of this chapter.

CRC generation can be disabled by clearing bit 5 of the DLC Command/Control Register. The FCS field is transmitted after the Information (I) field and just prior to the closing flag.

2-to-1 Multiplexor

The outputs of the parallel-to-serial shift register and the CRC generator are fed into the zero insertion unit via a 2-to-1 multiplexor. During the data portion of a packet, (we will refer to the address, control, and information fields as the "data"), the multiplexor is passing data from the shift register. After the last bit of the data portion of the packet has been shifted out of the shift register, the FCS is passed out of the CRC generator, if the CRC generator is enabled.

Zero-Bit Insertion Unit

To maintain data transparency, the transmitter examines the frame content between the opening and closing flag, including the address, control, information, and FCS fields, and inserts a 0 bit after all sequences of five contiguous ONEs. This prevents the data stream from simulating flags and aborts.

Serial Bus Port

The Serial Bus Port (SBP) sits at the output of the transmitter. The SBP performs several functions related to time slot assignment, clock selection, data inversion, enabling the transmitter, and loop back testing.

Time Slot Multiplexor (TSM)—The output of the zero-bit insertion logic is routed through the TSM where it is assigned one of 31 time slots, or transmitted as is, referred to as the non-multiplexed mode. The SBP is designed to connect directly to the SBP of the Am79C30A DSC. Up to 31 time slots combine to form a frame, where data are transmitted during one of the 8-bit time long windows (see Figure 2-4). The Serial Frame Sync (SFS) input provides a reference indicating the location of the first eight bits of the frame. (The SFS/XMITCLK pin serves as either the SFS input in multiplexed mode, or the transmit clock input in non-multiplexed mode.) The transmitter can be programmed to place data on any one of up to 31 time slots via bits 0-4 of the SBP Control Register. The TSM is re-synchronized by each frame syncronization pulse, allowing frames of from 1 to 31 channels to be used. In the multiplexed mode, the SCLK pin provides the transmit clock source. This clock source is gated with the selected time slot to provide the transmit clock. If time slot 0 is selected, data are transmitted for as long as the SFS signal is active, instead of for eight bits at a time. If the SFS input is held active for 16-bit times instead of 8 each frame, the transmitter will send out 16 bits per frame, as opposed

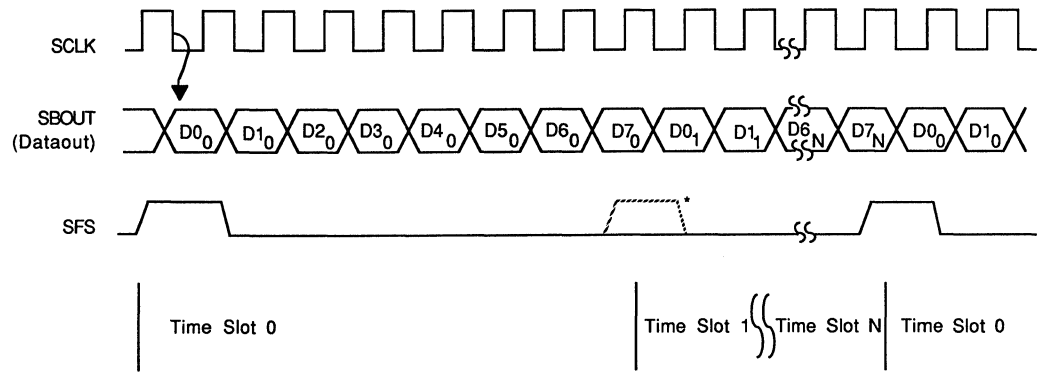


Figure 2-4. Transmit SBP Timeslot Channel Multiplexing

to 8. By doing this the DSC can place the data on both of the two B channels, on an every-other-byte basis, effectively doubling the data rate. In the non-multiplexed mode, bits 1-5 all set to ONEs in the SBP Control Register, data are transmitted continuously. In this mode, the transmit clock is input on the SFS/XMITCLK pin. When the DLC is used in the DSC or DEC, the TSM must be hard-wired into the non-multiplexed mode.

Data are always transmitted on the falling edge of the transmit clock.

Inversion—before data have passed through the TSM they are fed through a programmable inverter. If bit 5 of the SBP Control Register is set to 1, the data will be inverted.

Mark Idle Insertion—Whenever the transmitter is enabled (bit 1 of the DLC Command/Control Register) and is out of frame (and the closing flag or abort has been sent) with mark idle selected (bit 3 of the DLC Command/Control Register), the transmitter's output will be forced High. This takes place before the programmable data inverter.

Transmitter Enable—The transmitter is enabled and disabled via bit 1 in the DLC Command/Control Register. Whenever the transmitter is disabled, the SBOUT pin, which is an open-drain pin, is forced High.

Local Loop Back—The DLC can be placed in a local loop back configuration for test purposes. This is done by setting bit 6 to a 1 in the SBP Control Register. Local loop back disconnects the SBIN and SBOUT pins, (SBOUT is driven High - SBOUT is an open drain pin), and connects the transmitter output and receiver input together. The selected transmitter clock (see above) is used as the receive clock.

Remote Loop Back—The DLC can be placed in a remote loop back configuration for test purposes. This is done by setting bit 7 to a 1 of the SBP Control Register. Remote loop back disables the transmitter and echoes whatever is received at the SBIN pin out the SBOUT pin.

NOTE: The receiver can be either enabled or disabled without affecting operation. Data are transmitted on the falling edge of SCLK. SCLK performs both transmit and receive functions.

Receiver

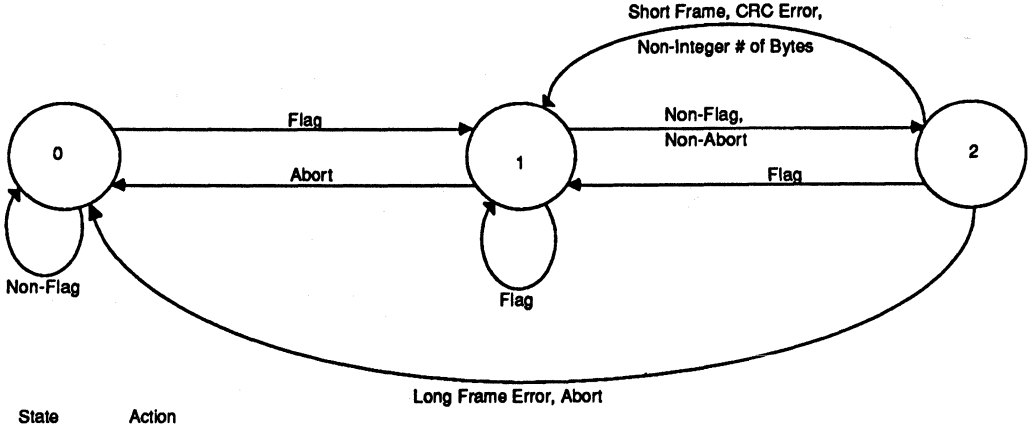
The receive portion of the DLC takes serial data from the Serial Bus Port (SBP), processes them, and allows them to be moved to off-chip memory (the receiver includes the SBP and the FIFO). Dedicated hardware modules are used to perform the bit-level operations on each frame of data as it is received (mark idle detection, data inversion, flag/abort recognition, zero-bit deletion, CRC checking, and address recognition). A 32-byte deep FIFO is used as a buffer between this bit rate dependent processing and the packet at a time processing performed by the CPU. Data can be moved from the FIFO to memory either by DMA, or programmed I/O.

This portion of the document is divided into two parts: a description of receiver operation—including normal operation as well as exceptions (see Figure 2-5) and a detailed discussion of the hardware functional blocks (see Figure 2-6). Throughout the discussions of the receiver, reference will be made to bits in control, status, and command registers; these registers are described in the data sheet. If you have not already reviewed the section on bit-oriented protocols, it is recommended that you do this before proceeding.

Receiver Operation

This section will begin with a discussion of the normal flow of events, from the time that the receiver is reset through the receipt of a frame of data. This will be followed by a discussion of the exceptions to this operational flow.

NORMAL RECEIVER OPERATION—When the receiver comes out of hardware reset, or is reset by software (bit 6



State	Action
0	Hunt for Flag (No Flag Sync)
1	Hunt for Non-Flag, Non-Abort
2	In-frame (Look for Flag)

Figure 2-5. DLC Receiver State Diagram

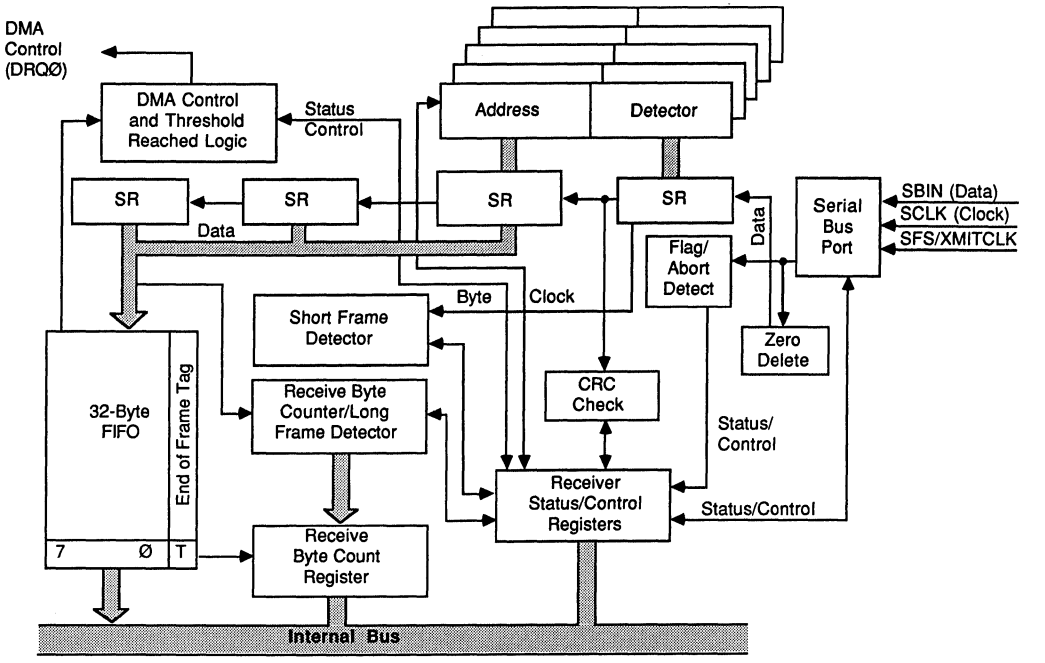


Figure 2-6. DLC Receiver Block Diagram

of the DLC Command/Control Register), the receiver is disabled and is in state 0.

NOTE: When the receiver is disabled (by clearing bit 2 of the DLC Command/Control Register), the connection between the SBIN pin and the receiver is severed. This is the only effect that disabling the receiver has on the remainder of the DLC logic. All other receiver functions work in the same manner as they do when the receiver is on.

Initialization—The user, via software running on the external CPU, initializes the receiver prior to operation by selecting data inversion/non-inversion (bit 0 of the SBP Control Register), specifying SBP channel configuration (bits 1-5, of the SBP Control Register), enabling CRC check if desired (bit 4 in the DLC Command/Control Register), selecting the desired address mode (Address Control Register), loading the address(es) to be recognized (Address Register(s)), specifying the minimum packet size (Minimum Packet Size Register), specifying the maximum packet size (Maximum Packet Size Register), and finally enabling the receiver (bit 2 in the DLC Command/Control Register). The initialization sequence is described in the "DLC Operational Sequences" section of Chapter 4.

Operational Sequence—The receiver starts operation in state 0. In state 0 the receiver examines the incoming data stream, clocked in from the SBIN pin on the rising edge of SCLK (SCLK pin), on a bit-by-bit basis for the presence of a flag character. No data are passed beyond the flag/abort detection unit in state 0. The detection of a flag causes a transition to state 1.

In state 1 the data stream is inspected on a character-by-character basis for the presence of a non-flag, non-abort character, (character boundaries are established by the receipt of a flag). If the character following the flag is another flag, the receiver remains in state 1. If the character is an abort, the receiver re-enters state 0. If the character is not a flag or an abort, the receiver is said to be in-frame, and state 2 is entered.

In state 2, data are passed beyond the flag/abort detector to the zero-bit deletion unit. Here, the next bit following any five contiguous ONEs is deleted. This bit should always be a ZERO and was inserted by the transmitter to prevent data patterns from being detected as flag or abort characters, which have six and seven contiguous ONE bits respectively. The first one or two characters following the opening flag of the packet are normally the address field. While the address field can be more than two bytes long, the receiver can examine only the first two bytes of any address; any remaining bytes are treated as data. If address recognition is enabled (bits 0-4 of the Address Control Register), these characters are tested by the address recognition unit for a match with one of the five enabled preprogrammed addresses, four programmable addresses and the broadcast address. If there is not a match, the receiver returns to state 0, looking for flags. The packet currently being transmitted is ignored and no status is reported on it. However, if there was an address match, or address detection was disabled, in which case all frames are accepted, the frame is received and is placed into the FIFO, one byte at a time, including the address, control, information, and FCS fields). Each received character is loaded into the FIFO when it reaches the last eight bits of the serial-to-parallel shift register, with the exception of the last character, discussed below.

State 2 is exited normally whenever the flag/abort detector receives a flag character. If a flag is detected the receiver enters state 1. Back-to-back packets can share opening and closing flags. At the time the flag is detected, the three previous characters still in the shift register are immediately loaded into the FIFO, assuming FCS pass through has been selected, (bit 7 of the DLC Command/Control Register). If FCS pass-through has not been selected, only the first of these three bytes is moved into the FIFO. In either case, the last byte placed in the FIFO is tagged as the end of the packet. The tag takes the form of a ninth bit appended to each word in the FIFO. If the character preceding the two byte FCS field contained less than eight bits, the actual number of bits received is reported in bit positions 0-2 of the Residual Bit Status/Control Register. If CRC checking has been enabled, the output of the CRC comparator is valid at this time, and its status (error or not) is recorded. These last two characters loaded into the FIFO are the Frame Check Sequence (FCS), if CRC check is enabled.

When a packet has been received with either a closing flag, an abort, or a long frame error, its length and status are latched. This information is presented to the user when the last byte of the packet, tagged as such, is read from the FIFO by DMA or programmed I/O. A maskable interrupt indicating the receipt of a packet, and its status, is generated at this time. The delay in status reporting is required since the user's software operates at a packet level and has not received the complete packet until the last byte has been moved from the FIFO to memory. In normal operation, the FIFO is automatically unloaded by the DMA and the user is not interested in the status of a packet until it has been completely transferred to memory. The discussion of the FIFO/DMA interaction is presented in the "DMA Operation" part of the "DLC Programmable Operations" section of Chapter 4.

EXCEPTIONS TO NORMAL OPERATION—During the course of normal operation, six error or exception conditions can occur. These are: the receipt of an abort character while in-frame, a CRC error, a short frame error, a long frame error, a non-integer number of bytes error (if residual bit operation is not allowed by the protocol in-use), and a FIFO overrun error. In addition to these six cases, the receiver can be placed in two test modes: local loop back, and remote loop back.

It should be noted that any packet received with 24 or fewer bits (whether an error exists or not) is discarded from the serial-to-parallel shift register without the reporting of status.

Abort—When an abort is received while the receiver is in-frame (state 2), the packet is terminated. The abort takes precedence over all receive errors. As a result of this termination several actions are taken:

1. The contents of the shift register are moved to the FIFO. The last byte is tagged as such as it is placed into the FIFO.

2. The receiver returns to state 0.

3. The status, including the abort received bit in the Receive Link Status Register, and byte count are latched (see the "DLC Programmable Operations" section of Chapter 4 for a discussion of the three stage delayed reporting mechanism).

4. When the last byte of the aborted packet is read from the FIFO, a maskable interrupt is generated.

CRC Error—When the closing flag of a packet is detected, the CRC checker has finished its work. If CRC checking is enabled (bit 4 in the DLC Command/Control Register), the output of the CRC checker is tested at this time. If an error has occurred, this error condition is latched for delayed reporting.

Short Frame Error—When a packet is terminated with a flag, and the packet has fewer characters (exclusive of flags) than is programmed into the Minimum Receive Packet Size Register, and more than 24 bits, a short frame error is reported. If the packet had 24 or fewer bits it is discarded without notification to the user. This is possible since no data have been placed into the FIFO at this time. If the short frame contained more than 24 bits, it is terminated the same way that a normal packet is, with the exception that the short frame error is latched for delayed reporting. The receiver returns to state 1.

Long Frame Error—The receiver contains a Maximum Receive Packet Size Register which is programmed to specify the maximum acceptable packet length. If the number of bytes received equals this count and a flag or an abort is not detected at this time, a long frame error exists and the packet is terminated. This termination is the same as for a normal frame with the exception that the long frame error status condition is latched for delayed reporting.

Non-Integer Number of Bytes Error—If the protocol being used allows for bit residue, this is not an error. Otherwise, if a flag is detected on a non-byte boundary (when from 1 to 7 bits of a character have been received), a non-integer number of bytes error exists. The packet is terminated as normal with the exception that the short character is loaded into the FIFO as is, it is tagged as the last byte, and the non-integer number of bytes error status is latched for delayed reporting. Note that if this error occurs in a short frame, the rules governing the handling of short frames (above) take precedence and only the short frame error is reported.

FIFO Overrun—When a byte has been shifted into the last 8 bit positions of the shift register it is moved into the FIFO buffer. If only the last location in the FIFO buffer is available when this load is attempted, a FIFO overrun condition exists. When this happens the packet is terminated, the byte in the shift register is placed into the FIFO and tagged as the last byte in the packet, and status is latched, including the overrun condition indicator, for delayed reporting. The receiver then returns to state 0; if a flag is detected at the same time as the overrun, then state 1 is entered.

Local Loop Back—For test purposes the output of the DLC transmitter can be looped back to the receiver. This mode is selected by setting bit 6 in the SBP Control Register. When in the local loop back mode, the receiver is isolated from its input (SBIN pin). Refer to "Serial Bus Port" part of the "Transmitter Block Description" section for details concerning data clocks.

NOTE: The receiver must be enabled for local loop back to work.

Remote Loop Back—For test purposes, the input to the receiver can be fed directly to the output pin of the transmitter (SBOUT). This mode is entered when bit 7 of the SBP Control Register is set. The operation of the receiver is unaffected by this action. Refer to the "Serial Bus Port" part of the "Transmitter Block Description" section for details concerning data clocks.

Receiver Block Description

The hardware blocks of the receiver will be discussed in the order that data flow through the unit, from the Serial Bus Port to the FIFO (refer to Figure 2-6).

Serial Bus Port

The Serial Bus Port (SBP) receives serial data from the SBIN pin, processes them and sends them to the flag/abort detection Unit and the zero-bit deletion unit. The SBP performs three operations on the data: mark idle detection, programmable data inversion, and time slot demultiplexing. Data are clocked into the SBP by the rising edge of SCLK (SBIN pin).

Mark Idle Detection—The mark idle detector examines the incoming data stream for the presence of 15 or more contiguous ONE bits, whenever the receiver is out of frame. The detection of a mark idle condition sets bit 0 in the Receive Link Status Register. If enabled, a maskable interrupt is generated in response to a negative to positive transition of this bit.

Data Inversion—The Programmable data inverter simply inverts the received data on a bit by bit basis. Setting bit 5 in the SBP Control Register causes this inversion.

Time Slot Demultiplexor—The Time Slot Demultiplexor (TSD) operates in one of two modes: multiplexed or non-multiplexed. When in the multiplexed mode (selected by bits 0-4 of the SBP Control Register), the incoming data are valid during one of up to 31 eight bit long time slots (see Figure 2-7). The Serial Frame Sync/Transmit Clock (SFS/XMITCLK) pin provides a frame sync pulse which is active at the beginning of the frame. This defines the frame boundaries. The active time slot is selected by bits 0-4 of the SBP Control Register. Time slot 0 is treated as a special case in which data can be received more than eight bits at a time. When time slot 0 is selected, SFS is sampled at the beginning of the ninth bit time. If SFS is sampled active, another eight bits of data will be received. This allows 16 bits of data to be received each frame. In an ISDN application, this allows the concatenation of both 64 kbps B-channels, thus doubling the data rate.

In the non-multiplexed mode, data are received as a continuous stream, clocked by SCLK. Non-multiplexed operation is selected by setting bits 0-4 of the SBP Control Register. In this mode, the SFS/XMITCLK input is not used by the receiver. It is used as the transmit clock input by the transmitter, thus giving separate receive and transmit clocks.

Flag/Abort Detection Unit

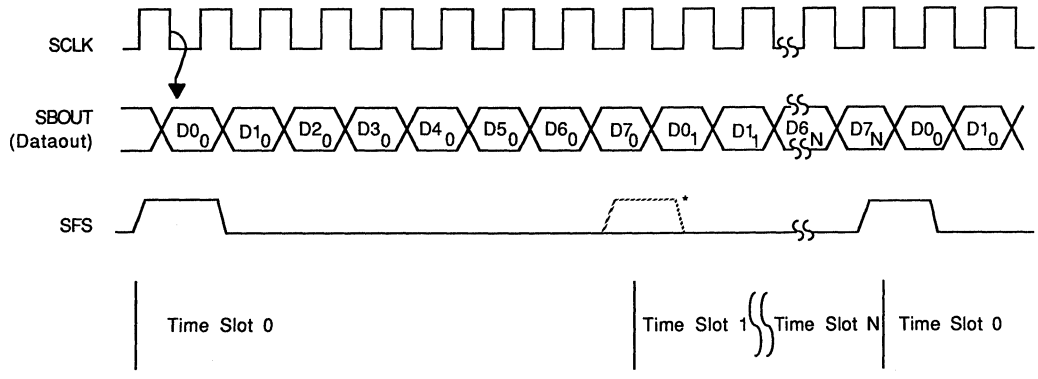
Receive data are fed from the SBP to the flag/abort detection unit and to the zero-bit deletion unit. The flag/abort detection unit is built around an eight-bit shift register into which the serial receive data are shifted on the rising edge of SCLK. The contents of the shift register are tested for the presence of either a flag or an abort character. This test takes place every time a bit is shifted into the shift register. In the case of abort detection, only the first seven bits are tested.

The detection of a flag is not directly reported in a status register. It serves several functions:

1. When the receiver is in state 0 (looking for a flag), the receipt of a flag causes a transition to state 1 (looking for a non-flag, non-abort character).

2. When the receiver is in-frame (state 2), the receipt of a flag causes the frame to end and the receiver to go to state 1. The packet is checked for short frame errors and, if enabled, CRC errors.

The receipt of an abort character causes the receiver to enter state 0 (looking for a flag). If the abort character is



* 16 Bit Channel Concatenation

Figure 2-7. Receive SBP Timeslot Channel Multiplexing

received while the receiver is in state 2 (in-frame), an abort condition exists. This causes the frame to be terminated. Assuming at least 24 bits preceded the abort, bit 0 in the Receive Frame Status Register is set. (Abort is a delayed status condition and is not reported until the last byte of the aborted packet is read from the FIFO.)

Zero-Bit Deletion Unit

In order to prevent valid data patterns from being detected as either flags or aborts, a technique called bit stuffing is used. The transmitter examines the data stream between the opening and closing flags, exclusively. If five consecutive ONE bits are detected, a ZERO is inserted after the fifth ONE. The zero-bit deletion unit in the receiver removes this added ZERO.

The received data are fed from the SBP to the zero-bit deletion unit. A three bit counter monitors the data stream for the presence of five consecutive ONEs. If this event occurs, the next bit is deleted from the data stream, normally a ZERO.

Short Frame Byte Counter

The Short Frame Byte Counter (SFBC) is a four-bit counter that counts the number of characters received after the opening flag. If a frame ends in a flag, -AND- the number of bytes received is less than the value programmed in the Minimum Packet Size Register, -AND- data have been placed in the FIFO (receive byte counter > 0), a short frame error is reported. Bit 3 of the Receive Frame Status Register is set when the last byte of the packet is read from the FIFO. If for some reason the software reloads the Minimum Packet Size Register while the receiver is in-frame, the previously loaded value will still be used for the packet that is being received. The new value will be used for the next packet. This is a delayed-stacked status condition and is not reported to the user until the last byte of the packet has been removed from the receive FIFO.

CRC Checker

The CRC checker is virtually identical to the CRC Generator in the DLC transmitter. As data are received they are fed through the zero-bit deletion logic, and into the checker. When the closing flag of the frame is detected the 16-bit Frame Check Sequence (FCS) has just been shifted into the checker. At this time the content of the CRC checker is tested. If an error exists, bit 2 of the Receive Frame Status Register is set. CRC checking is enabled by setting bit 4 in the DLC Command/Control Register. This is a delayed-stacked status condition and is not reported to the user until the last byte of the packet has been removed from the Receive FIFO.

Serial-to-Parallel Shift Register

The output of the zero-bit deletion unit is fed into a 32-bit shift register which converts the serial bit stream into bytes. The first 16 bits of the shift register are presented in parallel to the address detection unit for comparison. For one byte addresses, either the first or the second eight bits of the shift register are compared. The parallel output of the shift register is fed to the Receive FIFO a byte at a time.

Any packet that terminates, with a flag or an abort, before any data have been loaded into the FIFO buffer (receive byte counter is ZERO), is handled specially. In this case, no data are allowed to be placed into the FIFO, and no status is reported.

Address Detection Unit

The address detection unit is used to identify packets that are addressed to the receiver. Depending on programming, the first one or two bytes of each received packet is compared against up to five address registers: four user programmable and one broadcast. If the incoming packet's address field matches one of the address registers (if enabled - see below) the packet is received. If no match occurs the packet is discarded and the receiver re-enters state 0, looking for a flag.

Each of the five comparison units consists of a two byte comparator and an address register. The broadcast detector is hard-wired to look for a 111111X0, 11111111 address in two byte mode, a 111111X1 address in one byte/first byte mode, and a 11111111 address in one byte/second byte mode. (X = 1 if the address detection unit is programmed to include the C/R bit in the comparison, otherwise it is a don't care; the right-most bit is the least significant.) Associated with each comparison unit is an enable bit that turns that particular recognition unit on or off. These bits reside in the Address Control Register. If all five enable bits are cleared (disabled) the receiver will accept all packets. Bit 5 of the Address Control Register selects whether the address is one or two bytes long. If one byte addressing is selected, the comparison is made on either the first or second eights, as indicated by Bit 7 of the Address Control Register. Also, bit 6 of the Address Control Register causes the second bit (Bit 1) of the first byte of all addresses to be ignored. This is required since some BOPs use this bit position to indicate whether the packet is a Command or a Response (C/R). When this ignore C/R bit control bit is set, bit 1 of the first byte of all addresses is considered a don't care.

Address comparison takes place when the serial-to-parallel shift register has received 16 bits following the opening flag. The identity of the particular comparator that makes the match with the incoming address is reported in bits 0-2 of the Interrupt Source Register. This is a delayed-stacked status condition and is not reported to the user until the last byte of the packet has been removed from the Receive FIFO.

Receive FIFO

The Receive FIFO sits between the serial-to-parallel shift register and the microprocessor interface and consists of the FIFO buffer, the receive byte counter, and the data request control logic.

FIFO Buffer—The FIFO buffer is 32 bytes deep and is loaded by the serial-to-parallel shift register and unloaded at the receive FIFO Data Register by the CPU or the DMA.

Data Available—The presence of data in the Data Register is indicated by the setting of the data available bit (bit 1) in the FIFO Status Register. This bit is cleared by two conditions: 1) The FIFO buffer becoming empty, and 2) the last byte of a packet being read from the FIFO. In the latter, data available remains cleared until the user reads the least significant byte of the Receive Byte Count Register. This provides an indication to the user, operating in Programmend I/O (PIO) mode (instead of using DMA), that the last byte of a packet has been read, and it is time

to process that packet's status information. Data available generate a maskable interrupt.

End of Packet (EOP) Tag—When the receiver terminates the receipt of a packet, normally or abnormally, and data from that packet have been placed in the FIFO, the last byte of the packet is tagged when it is placed into the buffer. Each buffer location contains a ninth bit to accommodate this tag. The presence of a tagged bit in the buffer forces data request (described below) active.

Threshold Reached—Associated with the FIFO buffer is a threshold reached signal. This signal is active whenever the number of bytes of data in the buffer is equal to or greater-than the threshold level programmed in the FIFO Threshold Register. When threshold reached is active, bit 0 in the Receive FIFO Status Register is set to 1. A maskable interrupt is generated when the threshold reached bit transitions from ZERO to ONE. The threshold reached signal is also used in the generation of Data Request to the DMA.

Overrun—An overrun condition occurs if the shift register to FIFO buffer transfer does not take place before the first bit of the next character is shifted into the shift register. Under no circumstances is data in the FIFO buffer over written, i.e., no data are lost.

End Of Packet In FIFO Indication—In non-DMA operation, it is important to tell the user when the end of a packet has been detected, or declared in the case of an error. This indication is provided by the EOP bit in the FIFO Status Register, bit 5. The EOP bit is set when the last byte of a packet is loaded into the FIFO from the serial-to-parallel shift register (tagged as such). It is cleared when the tagged byte is read from the FIFO and there are no other tagged bytes present. The EOP bit causes a maskable interrupt to be generated.

DMA Request Control—The FIFO is responsible for the generation of a DMA Request signal (DRQ_0) that controls the operation of the DMA (when used). DMA request active informs the DMA that it should unload the buffer. DMA request goes active when the threshold reached signal becomes active, -OR-, a byte tagged as the end of a packet is present in the buffer. DMA Request remains active until the buffer becomes empty, -OR-, when the tagged byte has been removed. In the case where the DMA request signal becomes inactive because the last byte of a packet has been read from the FIFO, it will remain inactive until the user reads the status information for that packet. Specifically, the DMA Request signal is prevented from going active until the least significant byte of the Receive Byte Count Register is read-independent of the presence of data from subsequent packets being in the FIFO. This mechanism insures synchronization between packet data and status.

Receive Byte Counter—A 16-bit counter is provided in the FIFO to maintain a count of the number of bytes that have been placed in the buffer from the packet that is currently being received. When the last byte of the packet (tagged as such) is removed from the FIFO buffer, the content of the counter is loaded into the Receive Byte Count Register (see below). This is a delayed-stacked status condition and is not reported to the user until the last byte of the packet has been removed from the Receive FIFO.

The Receive Byte Count is used in the identification of long frames and frames that have been terminated prior to any data being placed in the buffer, and for software to determine the length of a received frame.

Long Frame Error—A long frame is defined as a frame that has not been terminated when the number of received bytes equals the value programmed into the Maximum Packet Size Register. (Note that the numeric value programmed into the RBCR is three smaller than the maximum packet size; i.e., if the RBCR is loaded with the number 17, the actual maximum packet size will be 20.) The user will never be able to receive more bytes in a packet than the number specified in the Maximum Packet Size Register. The byte that caused the long frame error is the last byte in the packet. The long frame error is reported by setting bit 4 in the Receive Frame Status Register. This is a delayed-stacked status condition and is not reported to the user until the last byte of the packet has been removed from the Receive FIFO.

Packets Shorter Than 24 Bits—There is no reporting of error and status conditions for any packet that is terminated before data have been placed in the FIFO buffer. When a packet is terminated the receive byte counter is inspected. If it is ZERO, indicating that no data have been placed in the FIFO, status is not reported for that packet.

Receive Byte Count Register—The Receive Byte Count Register reports the length of the received packet to software. This is a delayed-stacked status condition and is not reported to the user until the last byte of the packet has been removed from the receive FIFO.

USART

This section provides an overview of the USART features. Subsequent sections describe the hardware modules and operating modes.

Overview

The USART is similar to an 8250 with several added features. The additions include a Synchronous/transparent mode, a special character recognition unit, and transmit and receive FIFOs.

The 8250 features that are not supported in the IDPC are the ring indicate and receive line signal detect inputs, as well as the general purpose Output 1 and 2 lines.

Features

The USART has the following features:

- 5-, 6-, 7-, or 8-bit characters
- Even, odd, or no parity
- 1, 1½, or 2 stop bits
- RTS, CTS, DSR, and DTR handshake lines
- Synchronous/transparent operation
- Special character recognition
- Four-byte transmit and receive FIFOs
- Software reset
- Break generation

Interrupts

An interrupt is generated in response to the following conditions:

- Change in CTS
- Change in DSR
- Parity error
- Receive FIFO threshold reached
- Receive FIFO time-out
- Transmit FIFO threshold reached
- Transmit shift register empty
- Break detection
- Special character detected
- Framing error
- Buffer overrun

FIFOs

The USART receiver and transmitter each have a 4-byte deep FIFO. Each FIFO has a programmable threshold level, which can generate a maskable interrupt. The receive FIFO has a time-out which generates an interrupt if the level in the FIFO remains above zero, and below the programmed threshold, for more than a specified time.

Special Character Recognition

A unique feature of the USART is its ability to detect special characters. Up to 128 user defined characters can be detected on the fly. As characters are received, they are tested against a table of special characters. If the character received has been flagged as special, bit 5 of the USART Status Register is set, and a maskable interrupt is generated.

Operational Modes

The USART has two primary modes of operation: asynchronous and synchronous/transparent.

Asynchronous Operation

In the asynchronous mode the receive and transmit shift registers are clocked at a rate that is 16 times the baud rate. Asynchronous operation is selected by clearing bit 2 of the USART Control Register to ZERO. The source of the clock can be either the internal baud rate generator or an external input (receive clock input, RXCLK). The receive clock select is bit 0 of the USART Control Register, the transmit clock select is bit 1 of the USART Control Register.

Synchronous/Transparent Operation

In synchronous/transparent operation the receive shift register is clocked at the same rate as the data. This means that the data and clock must be in sync with each other. Data are latched into the receive shift register on the rising edge of the clock. The source of the receive clock can be programmed to be either the internal baud rate generator or the external receive clock input (RXCLK). Normally, for synchronous/transparent operation the receive clock source will be the external RxCLK because the data must be synchonous to the clock. Synchronous mode is selected by setting bit 2 of the USART Control Register.

Data Clocking—The clock used by the transmit shift register is also 1X the data rate. Data is shifted out of the shift register on the falling edge of the clock. The transmit clock can be provided by either the baud rate generator or the external receive clock input (RXCLK).

Data Transmission—Data are transmitted as a steady stream of bits with no framing start and stop bits involved. When the transmit shift register is loaded, its contents are transmitted directly. The next data byte is concatenated onto the previous one. When the shift register and FIFO are empty the line is placed in a marking (ONEs) condition.

Data Reception—Data are received as a steady stream of bits with no framing involved and therefore no character boundaries. As eight bits are received into the shift register, they are loaded into the FIFO. When the line is idle, marking, the receiver is receiving and moving to the FIFO bytes containing all ONEs. This mode is useful in low speed synchronous applications since the end to end link—IDPC USART, to packet network, to IDPC USART—appears as a piece of wire to the two end users. Data are sampled and transferred as long as receive clock pulses are received and the receiver is enabled.

USART Functional Description

Figure 2-8 shows the functional block diagram of the USART. The major blocks are the receiver (with special character recognizer), the transmitter, the modem control, the interrupt controller, and the baud rate generator.

Receiver

The receiver performs a serial to parallel conversion on the incoming data, verifies framing, buffers the data in a FIFO, and detects break conditions and special characters.

Receiver Enable

The receiver can be enabled and disabled via bit 7 in the USART Control Register.

Shift Register

The shift register does a serial data to parallel data conversion. The serial data are clocked into the shift register by either the data sample strobe in asynchronous mode or the rising edge of the receive clock in synchronous/transparent mode.

Framing Error—Asynchronous mode only. If RXD is sampled Low on the next bit time after the last bit of a character is received, a framing error exists and is reported via bit 3 of the Line Status Register. The character with the framing error is not loaded into the FIFO.

Fill Bits—Asynchronous mode only. When the USART receives characters containing less than eight data bits, the additional high order bits in the 8-bit byte that is to be loaded into the receive FIFO are set to ZERO.

Synchronous/Transparent Operation—In the synchronous/transparent mode, the RXD input is sampled on every rising edge of the 1X receive clock. Data are shifted into the shift register on every clock cycle. In this mode, there are no start or stop bits. One byte of data is received and loaded into the FIFO every eight clock times.

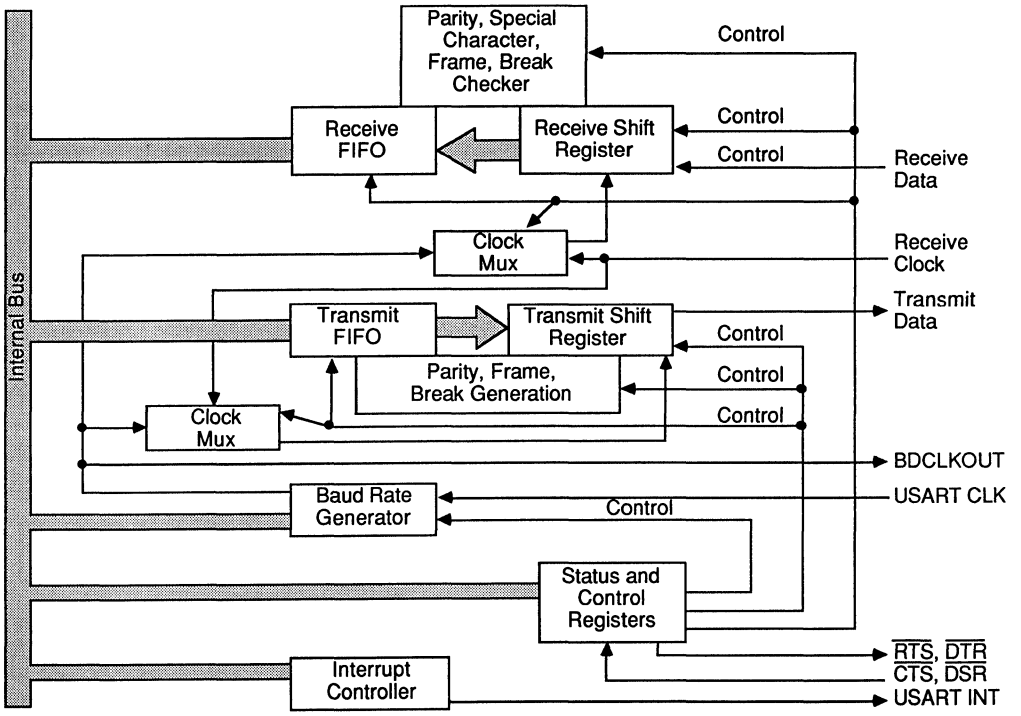


Figure 2-8. USART Block Diagram

Receive FIFO

Received data are loaded into a four byte FIFO.

Threshold Interrupt—An interrupt condition flag is set in the Interrupt Identification Register (bits 1-3) when the number of characters in the FIFO has reached the level designated in the receive FIFO threshold field of the USART Control Register (bits 3 and 4). This maskable interrupt takes the place of the receiver data available interrupt in the 8250. Bit 3 in the USART Status Register is set when the FIFO threshold is reached, and cleared when the FIFO level falls back below the threshold.

Time-out—A time-out is generated internally if the level in the receive FIFO is non-zero and less than the programmed threshold, and no characters have been received for 2048 receive clock cycles in the asynchronous mode. There is no time-out in the synchronous/transparent mode. The time-out sets bit 0 in the USART Status Register and generates a maskable interrupt.

Data Register—Data are read out of the FIFO, from the Receive FIFO Data Register, by the external CPU. The Receive FIFO Data Register is the equivalent of the Receiver Buffer Register in a conventional 8250. The presence of valid data in the Receive FIFO Data Register is indicated by (receive data available bit 0) in the Line Status Register.

Overrun—If the FIFO is full when a newly received character is to be loaded into the FIFO, an overrun error is reported via bit 1 in the Line Status Register.

Special Character Received and Parity Error Flags—The FIFO is ten bits wide: eight data bits, one special character flag, and one parity error flag. Parity, framing, and special character conditions are checked when the data are loaded into the FIFO. The presence of a character that has a parity error or is a special character is reported in the Line Status Register. If enabled, interrupts are generated when the condition is detected. Only the eight data bits can be read by the user. While special character and parity error interrupts are generated when the character is loaded into the FIFO, the parity error present and special character available status bits in the USART Status Register are not set until the character is at the FIFO output (see Figure 2-9). This allows the user to identify which character caused the interrupt.

Special Character Recognition

When a valid character has been received, the lower seven bits of its bit pattern are used as a pointer into a 128-bit RAM. If the RAM bit addressed by the data is set, the character is flagged as "special" by setting bit 7 (special character in FIFO) in the Line Status Register when the character is loaded into the FIFO. An interrupt is generated only if the special character enable bit is set

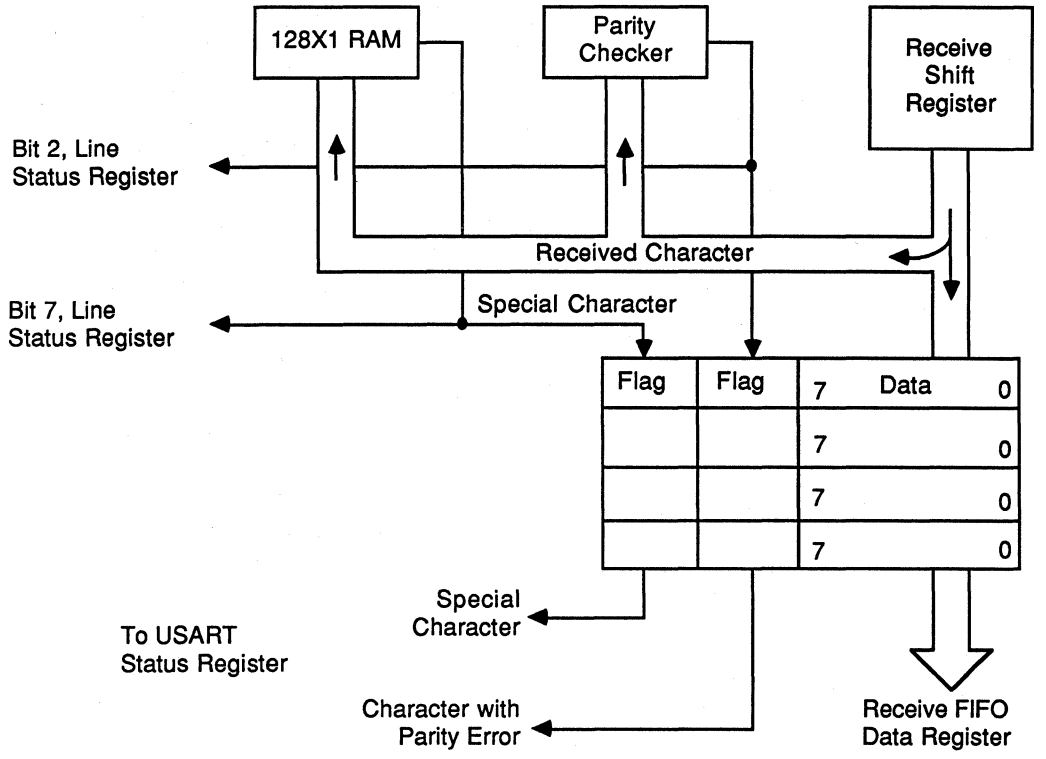


Figure 2-9. Parity Error and Special Character Reporting

(bit 5 in the Interrupt Enable Register). A second bit, bit 2 in the USART Status Register, is used to identify which character in the FIFO is special. This bit is not set until the special character is at the output of the FIFO. Bits in the 128-bit RAM are set and cleared by the external CPU.

Parity

Parity is checked on all received characters as they are loaded into the receive FIFO in asynchronous mode only. If a violation has occurred, parity is enabled (bit 3 of the Line Control Register), and the parity error bit is set (bit 2 of the Line Status Register). If the receiver line status interrupt is enabled (bit 2 of the Interrupt Enable Register), an interrupt will be generated. A second status bit, bit 1 in the USART Status Register, is set when the character containing the parity error reaches the output of the FIFO. This allows the user to identify which character in the FIFO contains the parity error. The selection of even or odd parity is made via bit 4 of the Line Control Register.

Stick Parity—The USART can be placed in a test mode that forces the parity bit to be generated as follows when parity is enabled:

If even parity is selected, the parity bit is always transmitted as a 0. The receiver expects the parity bit received to be a 0 (i.e., if a 1 is received, an error is generated). For odd parity, the reverse conditions apply.

Frame Errors

Framing is valid only in the asynchronous mode of operation. Framing is not checked in the synchronous/transparent mode.

Bit 3 of the Line Status Register is set if the received character does not have a valid stop bit, and it is not a break condition. A character with a frame error is not loaded into the receive FIFO. An interrupt is generated if the line status interrupt enable bit is set (bit 2 of the Interrupt Enable Register).

Break Detection

Break detection is valid only in asynchronous mode. Break detection does not take place in synchronous/transparent mode.

Bit 4 in the Line Status Register is set if the receive data input is held spacing (0) for more than a full character time (start bit + data bits + parity bit + stop bits). The receive line status interrupt enable bit (bit 2 of the Interrupt Enable Register), must be set to generate an interrupt.

Transmitter

Data that have been placed into the transmit FIFO are loaded into a parallel-to-serial shift register and shifted out by the programmed transmit clock. Parity can be generated and appended to the data. The character length and number of stop bits are programmable. Break indications can be generated by the transmitter.

Shift Register

The shift register clock can come from either the baud rate generator or from the transmit clock input pin. The input source for the shift register clock is 16 times the data rate in asynchronous mode, and 1 times the data rate in synchronous/transparent mode. Synchronous/transparent operation is selected via bit 2 of the USART Control Register. In asynchronous mode, the transmit logic automatically divides the clock by 16. Data are shifted out, LSB first, on the falling edge of the clock. Clock source selection is made via the transmit clock selection (bit 1) in the USART Control Register.

Bit 6 of the Line Status Register is set when the FIFO is empty and the last bit has been shifted out of the shift register. An interrupt is generated by this condition if bit 6 (transmit shift register empty) of the Interrupt Enable Register is set.

Transmit FIFO

Data Movement—Data to be transmitted are loaded into the transmit FIFO by the CPU. As the shift register becomes empty it is reloaded from the FIFO.

Threshold—When the number of bytes in the FIFO is equal to or less than a programmable threshold, the transmit FIFO threshold reached bit is set in the Line Status Register. An interrupt is generated if bit 1, transmit FIFO threshold reached, of the Interrupt Enable Register is set when the FIFO level falls to the programmed threshold level. The transition causes the interrupt, not the level in the FIFO being at or below the threshold. The threshold is programmed via bits 5 and 6 of the USART Control Register. Line Status Register bit 5 is the equivalent of the transmitter holding register empty bit in the 8250.

Parity—If selected, parity is generated as the data are moved from the FIFO to the shift register.

Frame Generation

Frame generation takes place only in the asynchronous mode of operation. The number of stop bits and character length is programmed into the transmitter. These parameters also hold for the receiver. The number of stop bits is programmed in the Line Control Register bit 2. The character length is programmed by bits 0 and 1 of the Line Control Register.

Break Generation

When the send break bit (bit 6 of the Line Control Register) is set by the CPU, the USART will transmit an all ZEROs pattern until the send break bit is reset by the CPU. The USART will finish transmitting the current character when the send break request is received. (A minimum of ten contiguous ZERO bits will always be sent when a break is requested.) The transmitter will return High for at least one bit time following the transmission of a break before a new character will be sent. This allows the start bit of the new character to be detected. Break generation causes the transmit FIFO to be cleared.

Modem Control And Status Registers

The Modem Control Register provides handshake signals for use in controlling communications between the IDPC and the terminal. These signals are: RTS/, CTS/, DSR/, and DTR/. RTS/ and DTR/ are outputs. They are controlled by the CPU via bits 1and 0 in the Modem Control Register, respectively. CTS/ and DSR/ are inputs. Their status can be read at Modem Status Register bits 4 and 5, respectively. The CTS/ and DSR/ inputs can generate a modem status interrupt if they have changed since the Modem Status Register was last read. This interrupt is enabled via Interrupt Enable Register bit 3. The change in CTS and change in DSR bits in the Modem Status Register (0,1) reflect the fact that the status of CTS/ or DSR/ has changed since the Modem Status Register was last read. Reading the Modem Status Register clears these bits.

Interrupt Controller

The USART generates one interrupt request to the CPU. The Interrupt Enable Register is used to mask individual interrupt sources. The interrupt request will be active until the source of the interrupt is cleared. Bits 1, 2, and 3 of the Interrupt Identification Register define the source of the interrupt. When cleared, bit 0 indicates that an interrupt is pending.

BIT 3 2 1	SOURCE	PRIORITY
000	MODEM STATUS	FOURTH
001	XMIT FIFO THRESH. RCHD.	THIRD
010	RECEIVE FIFO THRESH. RCHD.	SECOND
011	RECEIVE LINE STATUS	FIRST
100	RECEIVE FIFO TIME-OUT	FIFTH
101	LAST CHARACTER SENT	SIXTH

Data Clocks

The clock(s) used to transmit and receive data can come from one of two sources, either the receive clock input (RXCLK pin), or the baud rate generator. Clock selection is made via bits 0 and 1 in the USART Control Register.

Baud Rate Generator

The baud rate generator is a programmable divider that receives its input from the UASRTCLK pin and provides the clock to the USART receiver and transmitter. The input clock is divided by a programmable 16-bit (1-65535) divider. The programmable divider is configured by loading the Divisor Latch LSB and Divisor Latch MSB registers. These registers are accessed by setting the Divisor Latch Access Bit (DLAB) (bit 7 in the Line Control Register), and then writing to USART addresses 0 and 1. These are the Data Registers and Interrupt Enable Register addresses when the DLAB bit is cleared.

There is no enable/disable control for the baud rate generator.

In the asynchronous mode, the baud rate generator must be programmed to a value 16 times the data rate.

The output of the baud rate generator is fed to the receiver, the transmitter, and the BDCLKOUT pin.

If the baud rate generator is programmed to divide by one, the input signal (USARTCLK) is passed straight through unaltered to the receiver and transmitter via the clock selection multiplexors.

Clock Selection

The sources of the transmitter and receiver clocks are independently selectable. For example, when bit 0 is set in the USART Control Register, the receiver uses the output of the baud rate generator for its clock. When bit 0 is cleared, the RXCLK input is used. The same options apply for the transmitter, except that in this case bit 1 in the USART Control Register specifies the clock source.

DUAL-PORT MEMORY CONTROLLER

When the IDPC is used in host-based systems, the local CPU and any external host communicate with one another via shared memory (dual-port RAM). This memory is an external SRAM that can be accessed by either the local processor or the host CPU. The Dual-Port Memory Controller (DPMC) provides the control functions necessary to allow an ordinary SRAM to function as a dual-port device. These functions include the memory cycle timing generation, control of the buffers and latches required to isolate the host's system bus from the local processor's bus, and generation of the ready control signals back to the host and the local processor (see Figure 2-10).

In addition to arbitrating accesses to the shared RAM, the DPMC provides a semaphore mechanism (bidirectional interprocessor interrupts) that is used to coordinate the passing of high-level messages to and from the local processor and the host.

Memory Cycle Arbitration and Control

The discussion of the RAM cycle arbitration and control is divided into three parts: operational sequences, cycle timing and generation, and resolution of conflicting requests.

Operational Sequences

The DPMC generates the cycle timing for all accesses to the shared RAM. The length of each cycle is fixed and is independent of the cycle times of either the Local (L) processor or the Host (H). Memory cycles are generated in response to a request from either the local processor or the host. In the case of conflicting requests, the DPMC arbitrates the conflict, granting the first cycle to one requester while holding off the other via the appropriate ready line. The DPMC will arbitrate in favor of the local processor, referred to as the L-port, if the memory was idle in the prior cycle. This means that if the L-port has a request pending (via the LREQ/ input) at the time when the arbitration mechanism is ready to start the next memory cycle following a period of inactivity, the L-port will be granted the cycle regardless of a request from the host (H-port). If

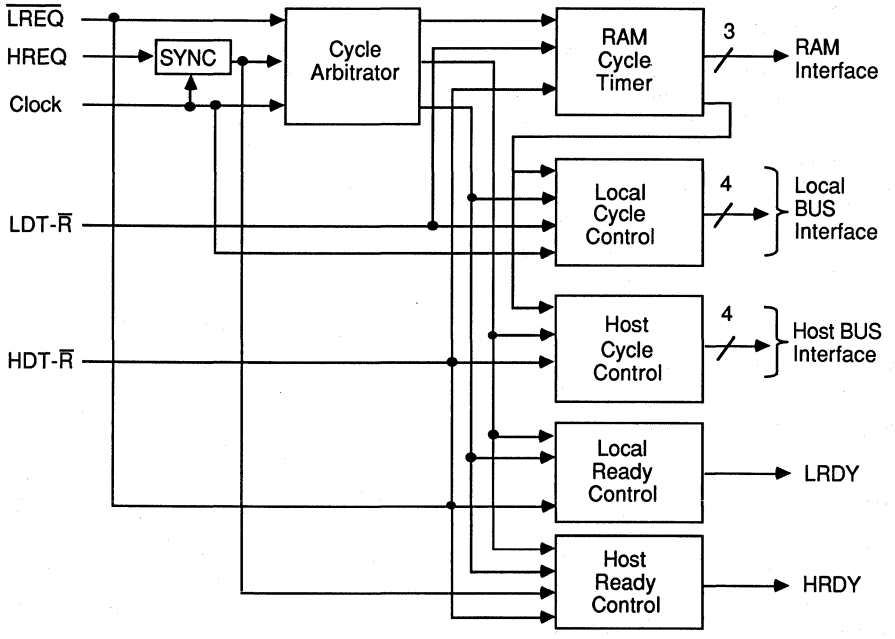


Figure 2-10. Dual-Port Memory Controller Block Diagram

a request from the host (HREQ input pin) is present, or becomes present during the existing cycle (L-cycle), the next cycle will be granted to the host (H-cycle). If an L-cycle request is received in the middle of an H-cycle the local processor is held off, via its ready line, until the H-cycle has completed.

L-cycle requests must be synchronous to the IDPC clock. This is not a problem since the IDPC clock is the same as the local processor clock and the memory cycle timing is generated from the IDPC clock. H-cycle requests are assumed to be asynchronous to the IDPC clock and are synchronized internally.

Memory Cycle Timing

The memory cycle is two IDPC clock times in length, with at least one clock time dead space inbetween any two cycles. The DPMC is designed to operate within the timing constraints of a 100 nanosecond SRAM, having the following timing specifications in nanoseconds.

Read cycle time	100 Min
Address access time	100 Max
Chip select to output	100 Max
Output enable to output valid	50 Max
Write cycle time	100 Min
Chip select to end of write	80 Min
Address valid to end of write	80 Min
Write pulse width	60 Min

Figure 2-11 shows the basic cycle timing.

Starting a Cycle—While the memory is idle the DPMC samples the LREQ/ and HREQ inputs on the falling edge of every IDPC clock cycle. If a request is present a cycle is started. The starting of a cycle causes the following actions to take place:

1. RAMCS/ is driven active (Low).

2. Either LABE/ or HABE/ is driven active (Low).

RAMCS/ provides the chip select control output to the RAM. The chip select is provided to take advantage of the power down mode available in most SRAMs. LABE/ and HABE/ are the address buffer enable controls that place the appropriate address on the memory bus. RAMCS/ and the address buffer enable signals (LABE/ or HABE/) remain active until the end of the memory cycle.

Determining Direction—On the next falling edge of the IDPC clock the active port's direction control input line (LDT-R/ or HDT-R/) is sampled. This determines whether the cycle is a read or write cycle.

Write Cycle—If the direction control is sampled High (write) the following actions are taken:

1. RAMWE/ is driven active (Low).

2. LDBE/ or HDBE/ is driven active (Low).

RAMWE/ is the RAM write strobe. It returns to its inactive (High) state at the end of the cycle. LDBE/ and HDBE/ are the data buffer enable controls that place the data to be written into the RAM on the memory bus. They also return to their inactive (High) state at the end of the cycle.

Read Cycle—If the direction control line is sampled Low (read), the following happens:

1. RAMOE/ is driven active (Low).

2. LDLE or HDLE is driven active (High).

3. LDLOE/ or HDLOE/ is driven active (Low).

RAMOE/ enables the RAM output drivers. LDLE and HDLE place the appropriate data bus latch in its transparent state. LDLOE/ and HDLOE/ enable the data bus latch outputs back to the local or host system bus. RAMOE/, LDLE, and HDLE are cleared at the end of the cycle. LDLOE/ and HDLOE/ are cleared when the cycle request (LREQ/ or HREQ) is removed.

Ending the Cycle—The memory cycle ends on the next falling edge of the IDPC clock. Note that the end of the cycle is independent of the state of the LREQ/ and HREQ inputs. These inputs will remain active until the end of the local or host system bus cycle. The fact that they remain active will not cause a new cycle to be started.

The LREQ/ and HREQ inputs are sampled on each successive falling edge of the IDPC clock to determine if a new cycle is to be started.

Conflicting Request Resolution

A conflict will occur in the event that the L-port requests a cycle while an H-cycle is in progress, or the H-port requests a cycle while either an L-cycle is in progress or an L-port request is present.

If LREQ/ becomes active while an H-cycle is in progress LRDY is immediately driven inactive (Low). LRDY is

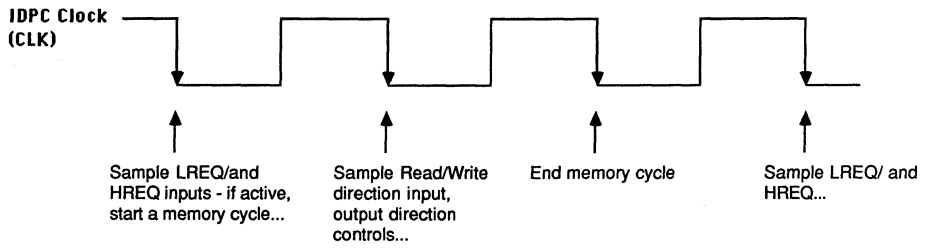


Figure 2-11. DPMC Cycle Timing

returned active at the start of the next memory cycle (which will be an L-cycle).

NOTE: For an 80188, LRDY is connected to the 80188's SRDY input (synchronous ready). SRDY is sampled on the falling edge of the 80188's clock. Therefore, LRDY does not return active until after the falling edge of the IDPC clock that starts the memory cycle. LRDY meets the setup time to the next falling edge of the 80188's clock in order to be sampled active and end the 80188's cycle.

The case in which HREQ becomes active while an L-cycle is in progress is handled exactly the same as above, except that HRDY is used as the control signal instead of LRDY and is held inactive until the end of the H-cycle.

The case where HREQ is active prior to the start of a cycle and LREQ/ also becomes active, causes HRDY to be driven inactive (Low) as soon as LREQ/ becomes active, assuming that the immediately previous cycle was an idle cycle. (If LREQ/ is already active—before the L-cycle starts—HRDY is driven inactive as soon as HREQ becomes active.) HRDY is returned active at the end of the H-cycle.

Interprocessor Interrupts

All communication between the local processor and the host takes place through mailboxes located in shared RAM. A mechanism is required to inform the recipient that there is a message in his mailbox. Interrupts are used for this task.

Operational Sequences

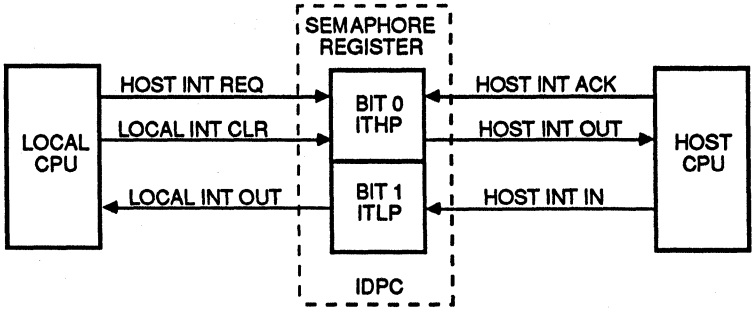
Message passing takes on two forms: local processor sending to the host, and host sending to the local processor. When the local processor wishes to send a message to the host, it first places the message in the host's mailbox and then generates an interrupt request to the host. The mailbox is located in the shared RAM—the message can either be placed directly in the predefined, by software, mailbox location, or a pointer to the message can be placed in the mailbox. The host reads the message and clears the interrupt request. Conversely, when the host wishes to send a message to the local processor, it places the message in the local processor's mailbox and generates an interrupt request to the local processor. The local processor reads the message and clears the interrupt request. The DPMC provides the hardware to facilitate the generation and clearing of these interrupt requests.

Interrupt Generation

Figure 2-12 shows the interconnection of the interprocessor interrupt mechanism to the host and local processors.

Local Processor to Host Interrupt—The local processor generates an interrupt to the host by writing a ONE to bit 0 in the Semaphore Register. The setting of this bit activates the host interrupt output (HINTOUT pin). The host clears the bit, and therefore the HINTOUT pin, by pulsing the host interrupt acknowledge input (HINTACK pin). The Semaphore Register can be read and written by the local processor, but not by the host.

Host to Local Processor Interrupt—The host generates an interrupt to the local processor by pulsing the host interrupt input (HINTIN pin). This sets bit 1 in the Semaphore Register and activates the local interrupt output (LINTOUT pin). The local processor clears the interrupt request (generated by the LINTOUT line) by clearing bit 1 in the Semaphore Register.



NOTES: Local Interrupt Clear and Host Processor Request are writes to the Semaphore Register by the local processor.

ITLP = Interrupt to Local Processor

ITHP = Interrupt to Host Processor

Figure 2-12. DPMC Inter-Processor Interrupt Structure

Chapter 3

APPLICATIONS

The applications chapter is divided into three sections: hardware interfacing, system applications, and AMD-provided software. The hardware interfacing section provides details of interfacing with IDPC's Microprocessor Interface (MPI), Serial Bus Port (SBP), and Dual-Port Memory Controller (DPMC). The system architecture section presents system examples of IDPC based embedded communication processors and terminal adaptors, as well as an introduction to ISDN software. The AMD provided software section is a brief overview of the software packages available from AMD to support the IDPC.

HARDWARE INTERFACING

This section provides examples of how to interface to the MPI including DMA, the SBP, and the DPMC. For reference, the IDPC pin descriptions are provided in the appendix.

IDPC/80188 Interface

The IDPC has been designed to interface cleanly with the 80188/801886 processor. Figure 3-1 shows the interconnection of the IDPC, 80188, and Am79C30A DSC.

Microprocessor Bus—The 80188 has a multiplexed address/data bus, which must be de-multiplexed in order to connect to the non-multiplexed bus of the IDPC, memory, etc. This is accomplished with an octal transparent latch (74LS373); the 80188 ALE signal provides the latch control. The 80188 RD/ and WR/ signals are directly connected to the IDPC signals of the same name. The IDPC CS/ input is generated by the 80188 internal programmable chip select generation logic; the IDPC is mapped directly into either the 80188's I/O or memory address space.

Clock—The 80188 has an on-chip oscillator that uses a 2X crystal to generate a master clock. This master clock is

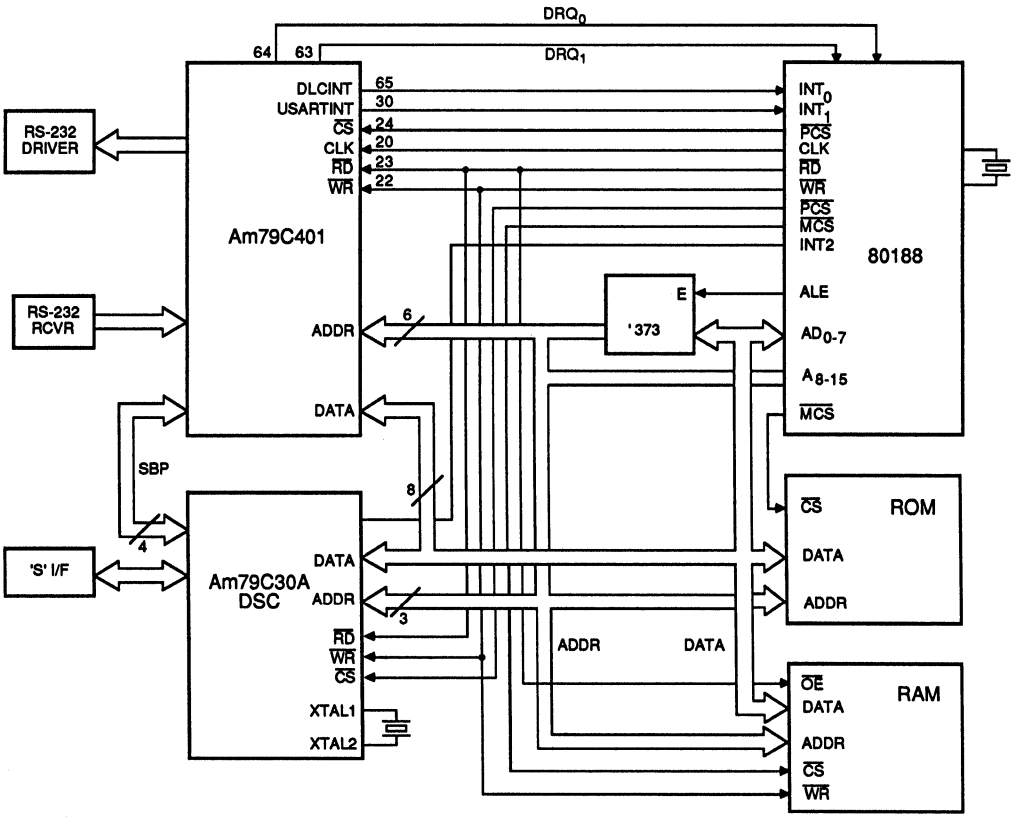


Figure 3-1. Am79C401, 80188, Am79C30A Interconnection Diagram

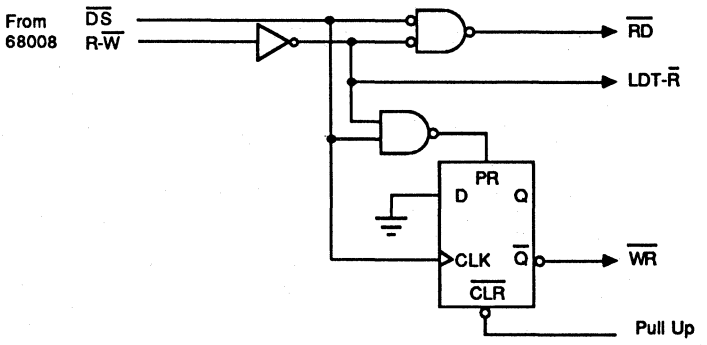


Figure 3-2. 68008, Am79C401 Read, Write, DPMC-Direction Control Signal Generation

provided as an output (CLKOUT), with all 80188 bus timing being related to it. CLKOUT is used as the IDPC CLK input. It is important to use this clock as the IDPC's master clock if the DPMC is used, because the DPMC assumes that requests to shared memory from the 80188 (LREQ/) are synchronous to the CLK input.

DPMC Signals—Three DPMC signals connect to the 80188: LREQ/, LDT-R/, and LRDY. LREQ/ is the local request input and is generated by the 80188 whenever the 80188 is accessing shared memory. One of the 80188's programmable address decode signals can be used for this purpose. LDT-R/ is the local bus direction input (Write = T, Read = R). The 80188 generates a bus status signal (S1/) which indicates early in the bus cycle whether a read or write is to be performed; S1/ is connected directly to LDT-R/. The last DPMC signal to be connected to the 80188 is the LRDY output. LRDY indicates to the 80188 that its request for a shared memory cycle cannot be serviced immediately, and that wait-states should be generated. LRDY is connected to the 80188 SRDY input, SRDY is the synchronous ready input. LRDY meets the setup and hold time requirements of the SRDY input since it is generated synchronously to CLK, which is connected to the 80188 CLKOUT output. Note: LRDY is an open-drain output and must be pulled-up to the +5V supply.

DMA Control Signals—The IDPC's DMA request outputs (DRQ0 = DLC receiver, DRQ1 = DLC transmitter) connect directly to the 80188 DMA controller's DMA request inputs (DRQ0, DRQ1). The DACK/ input is more of a problem, since the 80188 DMA controller does not generate an acknowledge signal. In some cases, the 80188 clock is operated at a slow enough speed that the DACK/ signal is not required. If this is not the case, a wait-state can be added to the DMA cycle, or a DACK/ signal can be constructed from other 80188 signals. Details of the DACK/ signals operation and timing requirements are presented in the DMA interface section, later in this chapter.

An interrupt acknowledge signal can be generated for the 80188 by building a signal that is active only when the 80188's DMA controller is reading RAM memory space (assuming that the only time the DMA controller reads the RAM is when it is loading the DLC Transmit FIFO). Such a signal can be built by ANDing the 80188's RD/, DEN/, and S6 with one of the 80188's chip select outputs. RD/ indicates a read cycle, S6 indicates a DMA cycle, and the 80188 chip select output is programmed to indicate an access to RAM. The DEN/ signal is active only while the RD/, S6, and chip select signals are stable, providing a clean DACK/ strobe.

IDPC/68000 Interface

Microprocessor Bus—The 68000 microprocessor bus is significantly different from the 80188 bus. Aside from the timing differences, the interface signals are different. Specifically, the 80188 has separate read and write strobes which provide direction and timing, while the 68000 has a single combined read/write signal indicating direction, plus a data strobe providing cycle timing. In order to interface the IDPC to the 68000, read and write strobes must be built. The example above shows an interface for a 10 MHZ 68008.

Figure 3-2 shows a simple circuit for generating the read (RD/) and write (WR/) strobes. The RD/ strobe is built by ANDing the data strobe (DS/) with the direction signal (R-W/). Figure 3-3 shows the timing diagram. The RD/ strobe is brought LOW (active) by the falling edge of DS/ qualified by R-W/ being HIGH, and returned HIGH by the rising edge of DS/. The critical parameter is the deactivation of RD/. DS/ is guaranteed to return inactive (HIGH) 20 ns (MIN) prior to the address bus becoming invalid. The IDPC requires an address hold time of 15 ns (MAX). This means that the maximum allowed propagation delay of the AND gate (AND of LOWs) is 5 ns plus the minimum propagation delay of the address bus buffer (if a buffer is present).

The WR/ strobe is slightly more complicated since the 68008 outputs the DS/ signal late in the write cycle, and the IDPC initiates its write cycle from the leading edge (falling) of WR/. The solution is to use the R-W/ signal which is generated sufficiently early to create the leading edge of WR/. The Q/ output (WR/) of a D-flip-flop is pre-set by the falling edge or R-W/, which is inverted and NANDed with DS/. The combination of R-W/ with the absence of DS/ is required to prevent the Flip-Flop from being held in pre-set since R-W/ is active longer than DS/. WR/ is returned inactive by the rising edge of DS/. The deactivation of WR/ has the same timing constraint that the deactivation of RD/

has (see above). WR/ must be deactivated quickly when DS/ returns HIGH in order to meet the IDPC's 15 ns address hold time.

DPMC Interface—The DPMC/68008 interface requires the generation of two signals: LREQ/, and LDT-R/. The local access request (LREQ/) is built by decoding the address space of the shared memory, gated with the 68008's address strobe (AS/). The local direction control LDT-R/ is simply the inverted form of the R-W/ signal (the inverted R-W/ signal is available as a by-product of the generation of RD/ and WR/ - see Figure 3-2). The only complication is in guaranteeing that LREQ/ will meet the the set-up time to the falling edge of CLK, start of S3, assuming that the IDPC clock and the 68008 clock are the same. The AS/ strobe, which is the gating factor in the generation of LREQ/, is generated too late to insure proper operation. The solution is to latch the address decode-AS/ combination with the rising edge of CLK, start of S4, insuring that LREQ/ will be stable prior to the subsequent falling edge of CLK - see Figure 3-4. As a consequence of this, one full clock cycle wait-state must be added to the 68008 bus cycle. The wait-state is required since the DPMC generates a 2-clock memory cycle, which must end at or before the falling edge of S7.

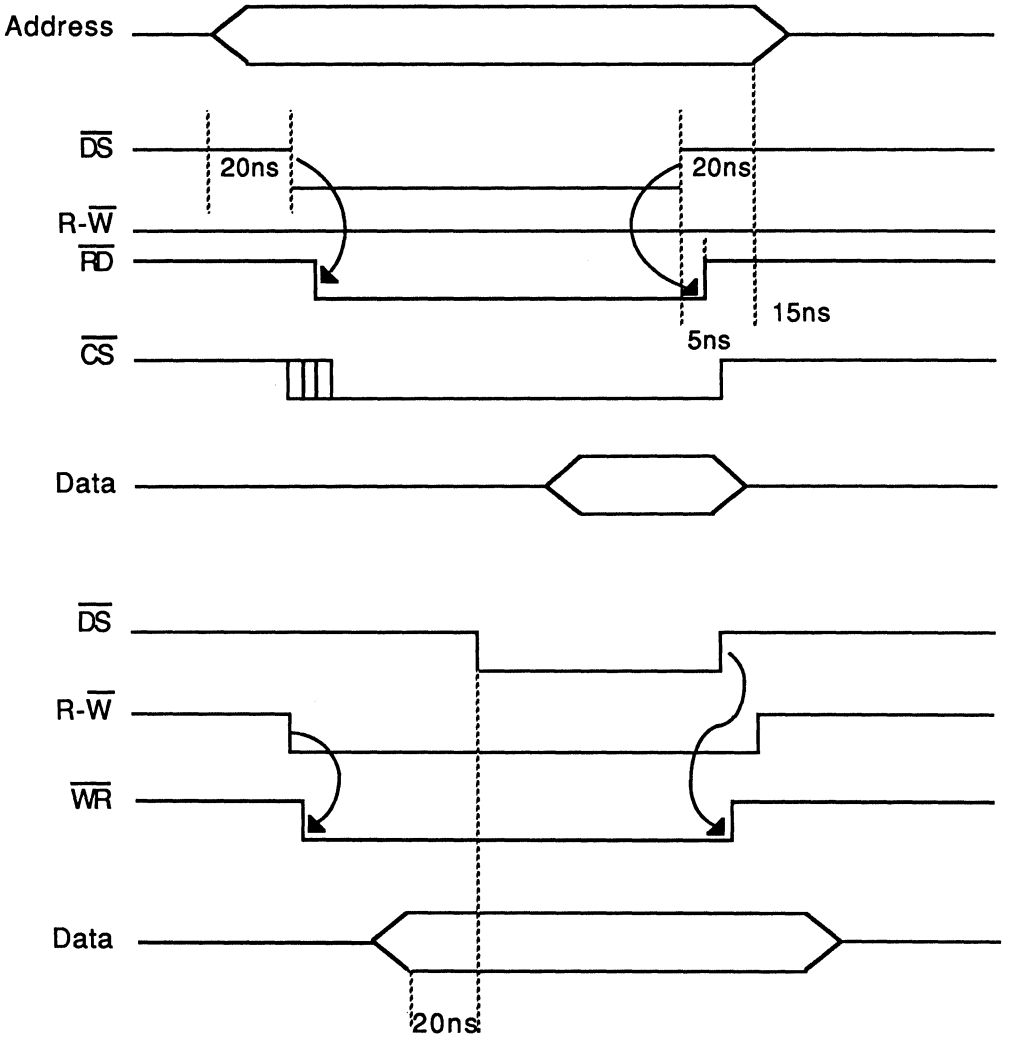


Figure 3-3. Read, Write timing

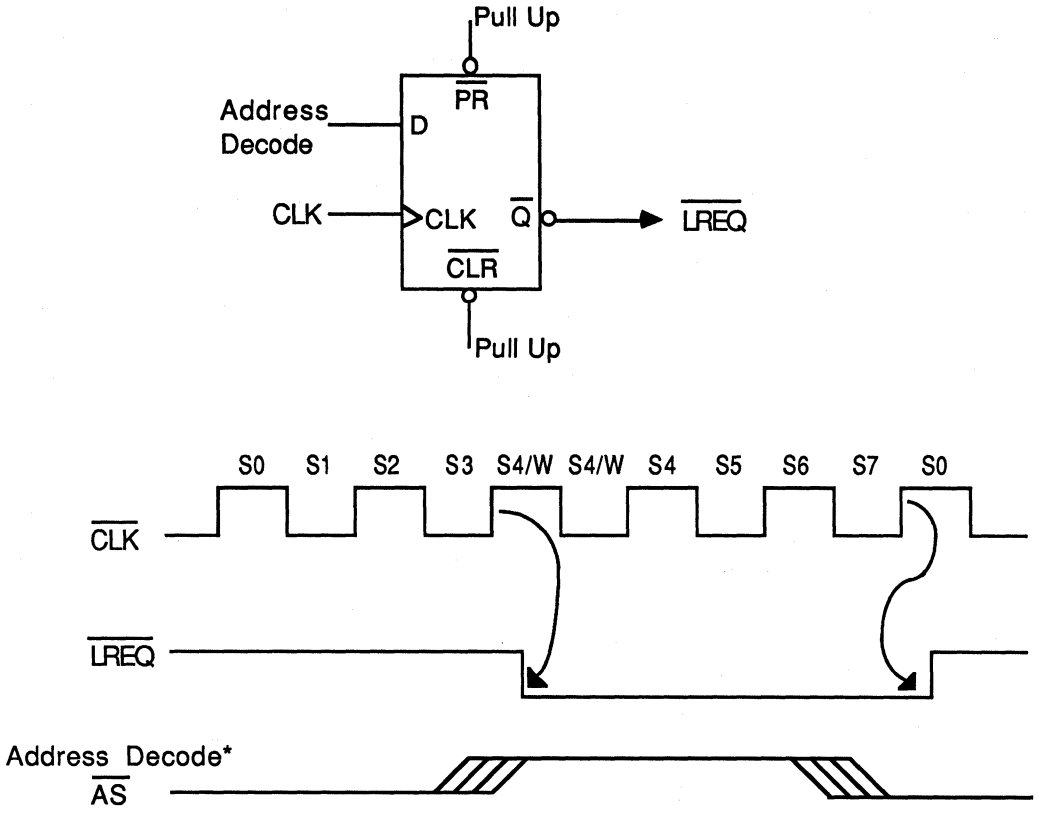


Figure 3-4. 68008, DPMC Local Access Request Generation

IDPC/DMA Interface

The IDPC can be connected directly to most DMA controllers that support source and destination synchronized transfers. Care must be taken to insure that the deactivation of the DRQ1 (transmitter DMA request line) occurs early enough to stop the DMA controller in time to prevent the transfer of one too many bytes of data. This can occur because the DMA controller does not write the last byte of data into the transmit FIFO until the second half of the DMA cycle. Data are read from RAM during the first half cycle and deposited into the transmit FIFO during the second half cycle, leaving little time for the DLC to deactivate DRQ1 prior to the DMA controller sampling the request input. This problem can be prevented in two ways:

1) Use of the DMA acknowledge output from the DMA controller - connected to the DACK/ pin on the IDPC. The DMA acknowledge signal is activated at the beginning of the DMA cycle, allowing time for the DLC to deactivate DRQ1.

2) Adding a wait-state to the DMA cycle. If the DMA controller does not provide an acknowledge output, or one cannot be generated, a wait-state can be inserted to provide more time prior to the DMA controller sampling DRQ1.

The DLC will deactivate DRQ1 during the last cycle when either the DACK/ pin is activated, or when the WR/ and CS/ pins become active, whichever occurs first. (Refer to the DRQ timing specifications in the IDPC Data Sheet for the DRQ inactive delay time. Refer to the Data Sheet of the specific DMA controller used in your design to determine how much time is available prior to the DMA controller falsely sampling DRQ1 and starting an unwanted DMA cycle.)

The DMA channel that loads the DLC receive FIFO does not have this problem since data are read from the receive FIFO during the first half of the DMA cycle. In this case the receive DMA request line (DRQ0) is deactivated during the last read cycle at the time that RD/ and CS/ become active.

IDPC/Am79C30A Interface

In ISDN applications, the IDPC's DLC is connected to the Serial Bus Port (SBP) on the Am79C30A DSC, or Am79C32A IDC. This provides the connection between the DLC and the 'S' Interface transceiver on the DSC. The Serial Bus Ports on the IDPC's DLC and the DSC are time slot multiplexed busses, with data input, data output, clock, and frame synchronization signals (the IDPC's SBP is operated in multiplexed mode). The IDPC's SBP is a slave to the DSC's SBP in that the DSC provides the clock and frame synchronization signals. The connection between the IDPC and DSC is shown in Figure 3-5. (Note: The IDPC's SBOUT pin, data output, is open-drain, and must be pulled-up to the +5V supply.)

DPMC/SRAM Interface

The DPMC has complete control of the timing of the shared memory bus cycles. The DPMC generates signals that control the RAM, the host bus interface logic, and the local bus interface logic. Figure 3-6 shows the details of the RAM/local bus/host bus interface.

RAM Control Signals—The DPMC generates three signals that control the RAM chip select - RAMCS/, write enable - RAMWE/, and output enable - RAMOE/. RAMCS/ is generated at the start of a memory access. RAMWE/ and RAMOE/ are generated mid-cycle depending on whether a write or read cycle is in progress. All three signals are deactivated at the end of the cycle.

Local and Host Bus Interface Control Signals—The local and host bus interfaces are identical and consist of three-state buffers that connect the host (or local) address bus to the RAM, a three-state buffer that connects the host (or local) data bus to the RAM (write cycle), and a three-state latch that connects the RAM to the host (or local) data bus (read cycle). Refer to the DPMC section in Chapter 2 for a description of the generation and timing of these signals.

SYSTEM APPLICATIONS

This section is divided into two parts, the first provides an introduction to the hardware architectures of terminal adaptors and embedded communication controllers, and the second covers ISDN system applications.

Hardware System Architecture

The principle application for the IDPC is the connection of equipment to a packet network. In this application, the network interface can be either integrated into a host system, or built as a stand-alone package, which is used to connect non-network ready equipment to the network. The two basic differences between these applications are: 1) the integrated, or embedded, application has a system processor (host) and a communication processor, while the stand-alone device has only a single processor; 2) in the integrated application, the communication between the host system and the communication system takes place over the parallel system bus, while in the stand-

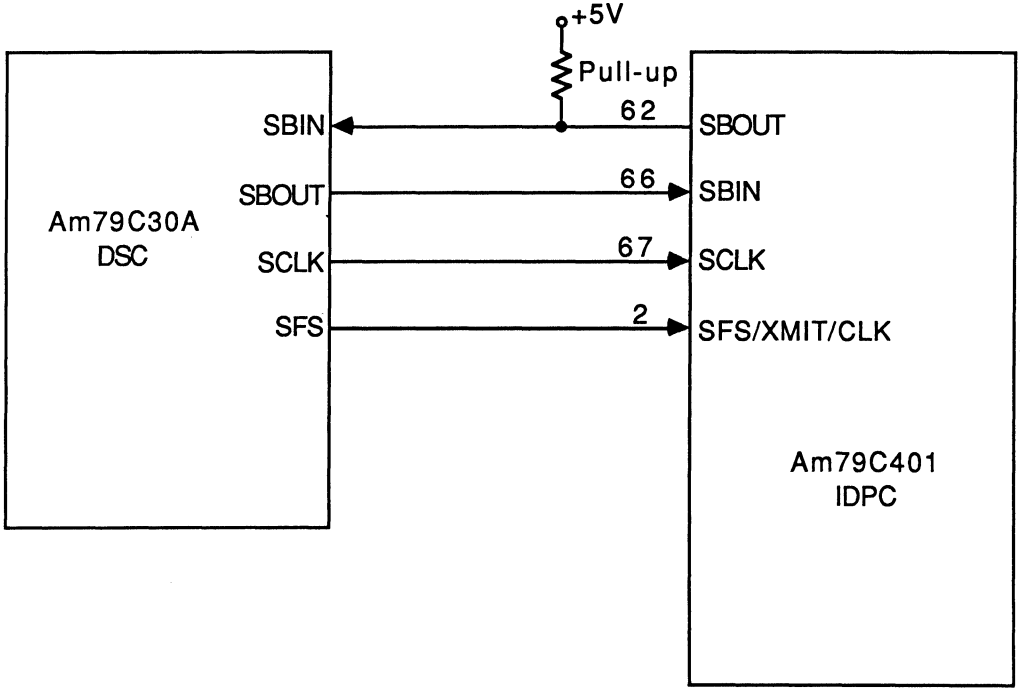


Figure 3-5. Am79C401, Am79C30A/32A Serial Bus Port Connection

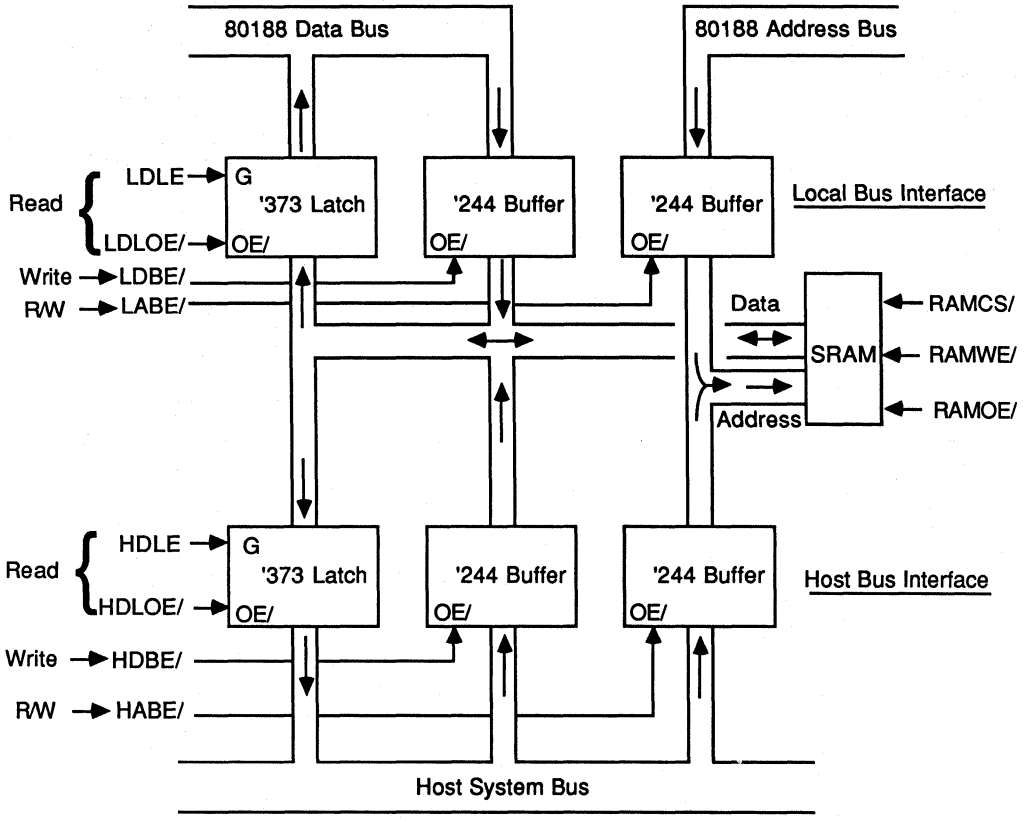


Figure 3-6. DPMC RAM Interface

alone case, a serial communications channel is used. In integrated applications, the Dual-Port Memory Controller (DPMC) provides support for building a shared memory interface between the host system and the communications system. In stand-alone applications, the USART supports the serial channel between the non-network ready device and the communications controller.

Embedded Communication Controllers

When the communication controller is built into the computer or terminal, two interfaces are required: a network interface for connecting to the packet network, and a shared memory interface between the host system's processor and the communications controller's local processor. The network interface is provided by the DLC in the IDPC, and a physical interface transceiver. The IDPC's Dual-Port Memory Controller (DPMC) supports the shared memory interface, allowing messages and data to be passed between the processors.

SNA Example—Figure 3-7 shows the block diagram of an embedded communication processor for an SDLC based SNA network.

Network Interface—The DLC in the IDPC provides SDLC protocol support for data rates up to 2.048 Mbps. A physical layer transceiver is used to connect the DLC's Serial Bus Port to the network wiring. The SNA network software runs on an 80188 microprocessor and associated RAM and ROM.

Shared Memory Interface—Shared memory interfaces consist of two parts, a block of shared memory, and a mechanism for generating and acknowledging interrupts between the processors.

The most common means of sharing a block of memory is to use dual-port memory. If the size of the shared memory block is small (approximately 2K bytes), a dual-port RAM device can be used. If the required block of shared memory is large (8K bytes or more), as is typically the case in communications systems, the use of true dual-port RAM becomes prohibitive because of cost and required board space. The alternative is to use a standard single-port RAM and arbitrate accesses between the local and host processors. The IDPC's DPMC performs this task by performing the arbitration function and generating memory cycles to the RAM. The interface between the host system bus and the local processor's bus is provided by buffers and latches. (Refer to the preceding DPMC interface section for details of the bus interfaces.)

Typically, the shared memory space will be partitioned into buffers and mailboxes. The buffers are used to transfer

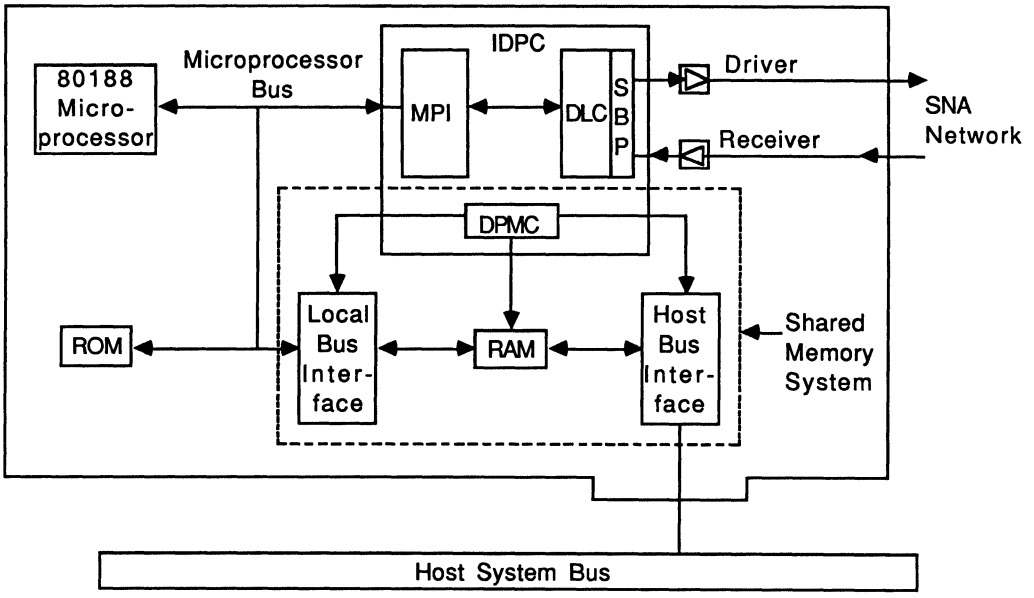


Figure 3-7. SNA Embedded Communication Controller

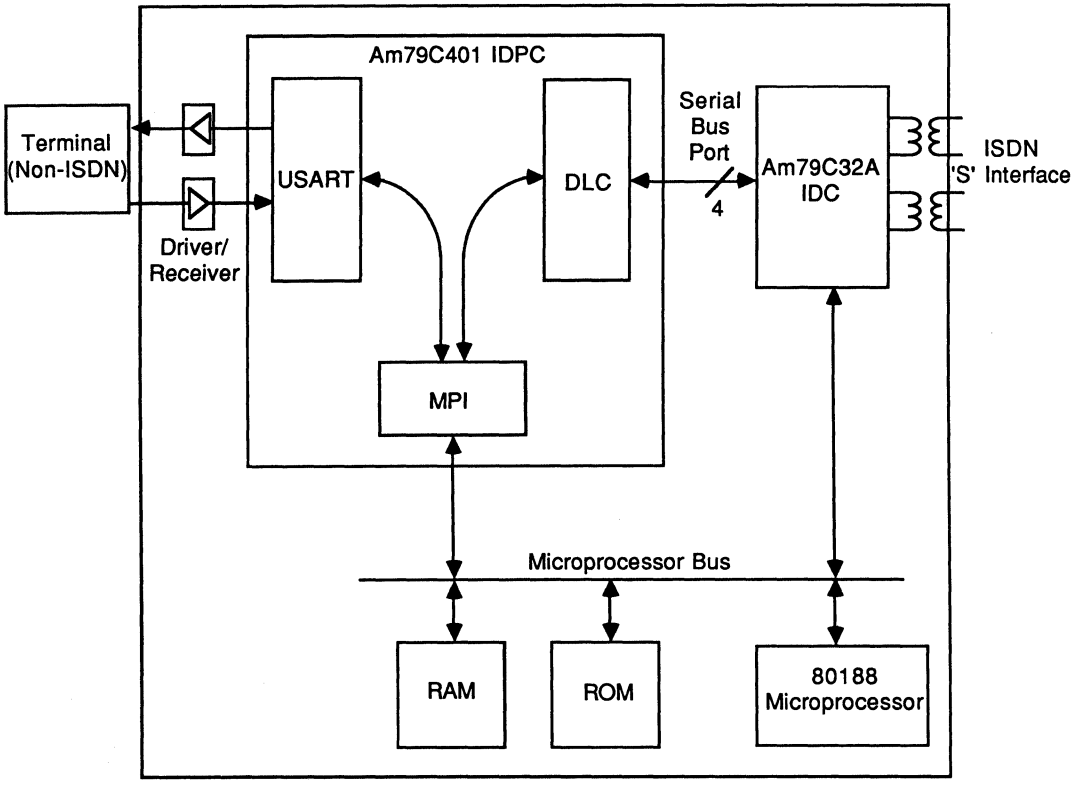


Figure 3-8. Terminal Adaptor

data, while the mailboxes are used to exchange commands. When one processor places a message into the other's mailbox, an interrupt needs to be generated alerting the recipient of its presence. The DPMC contains an interprocessor interrupt mechanism that allows both processors to generate and acknowledge interrupt requests.

Terminal Adaptors

The terminal adaptor is a self-contained device that allows non-network equipped terminals, or computers, to be connected to a network. Figure 3-8 shows the block diagram of a terminal adaptor for the ISDN. (A glossary of ISDN terminology is provided at the end of the ISDN System Architecture section.) The basic building blocks include: An ISDN 'S' Interface transceiver, providing the physical layer 1 connection to the ISDN; a protocol controller for processing the ISDN D-channel (the D-channel is used to for network call control); a B-channel protocol controller (the B-channel carries user data over the ISDN); a USART, providing the terminal interface; and a microprocessor (with RAM and ROM) to process both user data, and call control.

'S' Interface Transceiver and D-Channel Controller—The 'S' Interface transceiver and D-channel protocol controller are provided by the Am79C32A ISDN Data Controller (IDC). The 'S' Interface transceiver provides the physical layer connection to the ISDN network. The D-channel protocol controller provides support for the LAPD packet protocol used over the D-channel. The ISDN D-channel is used for call control functions such as setting up and tearing down the connection. (Refer to the Software Application section for a discussion of the ISDN software structure and D-channel functions.)

If voice facilities are desired in the terminal adaptor, the Am79C30A Digital Subscriber Controller (DSC) would be used in place of the Am79C32A IDC—This will be common, since the ISDN basic rate interface provides two separate 64 kbps channels (in this case one would be used for voice, the other for data). The Am79C30A DSC and Am79C32A are software and pin compatible.

B-Channel Controller and Terminal Interface—The B-channel protocol controller and terminal interface are provided by the IDPC. The IDPC's DLC serves as the B-channel controller. The DLC supports the three major packet protocols commonly used over the ISDN B-channel; these are X.25 (LAPB), V.120 (LAPD), and DMI (a slight variant of LAPD). The DLC connects directly to the serial port on the Am79C32A IDC (or Am79C30A DSC) via the IDPC and IDC/DSC Serial Bus Ports.

The IDPC's USART provides the terminal interface. Asynchronous terminals use the basic 8250 UART functions of the USART block, while synchronous terminals use the USART's synchronous/transparent mode.

Processor and Software—An 80188 microprocessor, with associated RAM and ROM, provides the processing power necessary to process the three separate data streams (terminal to USART, ISDN B-channel, and ISDN D-channel). The 80188 also provides a dual-channel DMA controller, three programmable timers, an interrupt controller, and chip select generation logic. The DMA controller is used to support B-channel data movement between the IDPC's DLC and memory. The programmable timers provide time bases required by the ISDN B- and D-channel software. The interrupt controller and chip select generator reduce the glue logic required to tie the system together.

AMD provides various software packages that support terminal adaptor applications, including: low-level drivers for the IDPC and DSC (or IDC), AmLink LAPD/LAPB layer 2 software, and AmLink3 layer 3 code.

ISDN System Architecture

The ISDN provides a framework for voice and data communication on a global scale. One application of the IDPC in the ISDN is the support of the transmission of user data over the ISDN B-channels. In this application, the IDPC performs protocol processing for the transmission of packetized data. (A glossary of ISDN terminology is provided at the end of this section.)

B-Channel Protocols

Data on the B-channel can take any form, so long as the data rate is 64 kbps, but most applications will use existing protocols to take advantage of available software.

Layer 2—The major layer 2 packet protocols include: SDLC, LAPB, and LAPD. SDLC is the protocol used over IBM's System Network Architecture (SNA). LAPB is used for X.25 networks. LAPD is used for the ISDN D-channel, AT&T's Digital Multiplexed Interface protocol (DMI), and the V.120 protocol. V.120 is significant for two reasons: first, using LAPD for both D-channel and B-channel, only one layer 2 protocol is required, second, V.120 allows the statistical multiplexing of multiple logical channels over a single physical channel. While statistical multiplexing is not new, V.120 represents the first international statistical multiplexing standard.

Layer 3—Each layer 2 protocol has an associated layer 3 protocol. X.25 (LAPB) and DMI (LAPD) use the X.25 Packet Layer Protocol (PLP), while SNA (SDLC) and V.120 have their own specific layer 3 protocols.

Software Requirements For a Voice/Data PC Plug-In Board

The following sections describe the layered structure of communications software, using a PC-based ISDN voice/data application as an example. An IBM-PC running MS-DOS is used as the host environment for this example; however, the basic software structure is applicable to any embedded processor environment.

Figure 3-9 shows the layering (partitioning) of ISDN software in a personal computer (PC) running MS-DOS.

The following basic mechanisms characterize each of the interfaces between software layers:

1. SET OF WELL-DEFINED PRIMITIVES (commands) and associated parameters that each layer uses to communicate with an adjacent layer.

2. MAILBOX in RAM where a layer writes a primitive command code (and associated parameters) to be read by an adjacent layer and where the adjacent layer writes any responses.

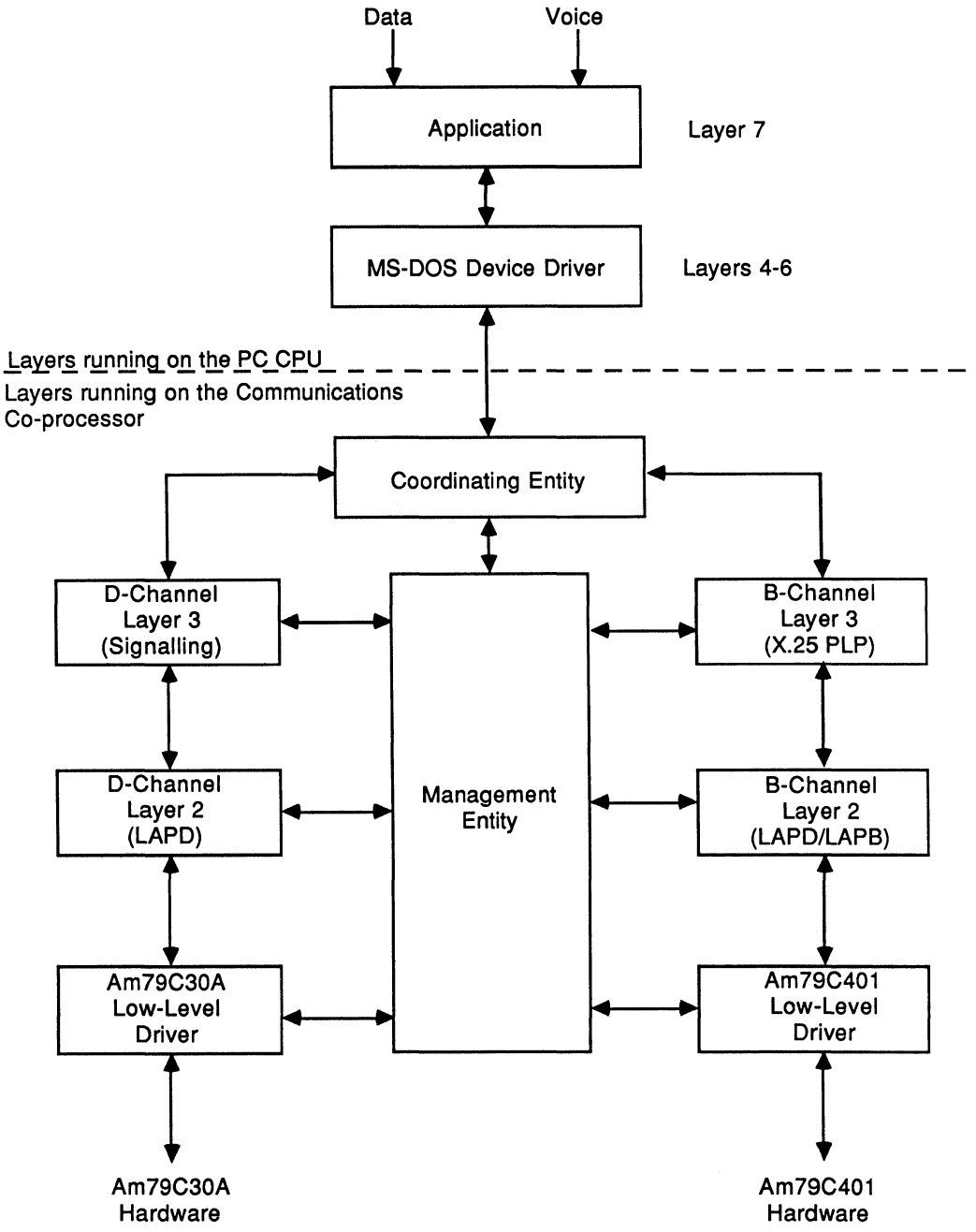


Figure 3-9. Software Layering for ISDN PC Application

3. INTERLAYER NOTIFICATION MECHANISM invoked by a requesting layer to cause an adjacent layer to read a mailbox and execute the primitive previously written there by the requesting layer. This notification mechanism can be a subroutine call, a software interrupt, or a hardware interrupt, depending on the individual interface.

Software Layers

The following is a general description of each of the layers depicted in Figure 3-9, from the top down.

Software running on the PC CPU:

The Application Layer (Layer 7)—interacts with the PC user to get the telephone number to dial and the data to be transferred across the ISDN network. These data may take the form of a disk file or data typed interactively on the PC keyboard.

The MS-DOS Device Driver (Layers 4-6)—converts system call requests from the application layer (e.g., OPEN, WRITE) into primitives for the coordinating entity to execute. Similarly, the MS-DOS device driver receives primitives from the coordinating entity (e.g., containing user data from the far end of the telephone connection) for transfer to the application via a READ system call.

Software running on the Communications Co-Processor:

The Coordinating Entity (CE)—coordinates the activities of the D-channel and the B-channel. For example, the CE ensures that an ISDN call has been set up (other end has answered the phone) before allowing any user data to be transferred on the B-channel. When stimulated by the MS-DOS device driver to initiate a data call, the coordinating entity exchanges primitives with the D-channel layer 3, the management entity, and the B-channel layer 3 to accomplish call setup. Once a call has been set up, the CE transfers user data between the MS-DOS device driver and the B-channel layer 3.

D-Channel Layer 3 (Signalling)—exchanges messages with the ISDN network to set up and tear down voice and data calls. These messages contain information such as the number to dial and information about call progress such as dial tone, ringing or busy, and answered. The CCITT Q.931 specification describes the signalling protocol.

D-Channel Layer 2—provides a reliable, error-controlled data transport service for carrying layer 3 messages to and from the ISDN network. The CCITT Q.921 (LAPD) specification describes the protocol used at layer 2 of the D-channel.

Am79C30A DSC Low-Level Driver (Am79LLD30A)—provides hardware independence to layer 2 by handling all details of programming the Am79C30A DSC registers.

B-Channel Layer 3—provides reliable, error-controlled transfer of user data, independent of the layer 2 protocol in use. An example of the B-channel layer 3 protocol is the X.25 Packet Layer Protocol (X.25 PLP), which is used for both X.25 and the Digital Multiplexed Interface (DMI).

B-Channel Layer 2—provides a reliable, error-controlled data transport service for carrying layer 3 messages to and from the next link. Examples of layer 2 protocols that may run on the B-channel include LAPB, LAPD, or SDLC.

Am79C401 IDPC Low-Level Driver (Am79LLD401)—provides hardware independence to layer 2 by handling all details of programming the Am79C401 IDPC registers.

The Management Entity (ME)—provides the glue necessary to make all the other layers running on the communications co-processor work together smoothly. Among the functions performed by the ME are:

- Provide real-time executive services such as task scheduling, timer services, buffer allocation and inter-layer primitive queuing.
- Collect error statistics such as CRC errors per second and notify other layers if errors exceed pre-selected thresholds.
- Control LLD functions which are not handled by layer 2 such as Am79C30A DSC tone generation to the voice handset (e.g., dial tone).

Software Considerations

In general, each layer "hides" complexity from the adjacent higher layer. In other words, a relatively simple primitive command transmitted from a higher layer may result in a series of complex actions by the lower layer. For example, when a layer 3 entity sends the "transmit user data" primitive to layer 2, the layer 2 entity performs several actions such as transmitting a frame, receiving an acknowledgment frame, and possibly retransmitting the frame if it was not received successfully. The layer 3 entity is never aware of whether a retransmission is required or not; layer 3 assumes that layer 2 takes care of these details.

An advantage of proper software layering is that it allows several different protocols in one layer to share the facilities provided by a single protocol in an adjacent layer. For example, the CCITT layer 2 Q.921 protocol can carry (multiplex) the messages of both of the following layer 3 protocols:

- The D-channel layer 3 signalling protocol (CCITT Q.931) for call setup.
- X.25 layer 3 for transferring user data (e.g., contents of a user disk file) on the D-channel.

Similarly, the Am79C30A DSC and Am79C401 IDPC low-level drivers present a common, hardware-independent interface to layer 2 such that the same layer 2 code may be shared by both D-channel and B-channel.

One of the significant features of ISDN is that different B-channel protocols may be used on different calls made from the same terminal. In addition, there is another set of layered protocols running on the D-channel at the same time that a given set of layers is running on the B-channel.

This multiplicity of protocols running on one communications interface is quite different from the traditional inter-

face running a single integrated protocol (e.g., running X.25 or SNA/SDLC, but not both). At present, some computers have more than one communications protocol available, but each protocol has its own dedicated interface software and hardware.

This potential for different protocols running on the same hardware on a phone call by call basis has implications such as the need for disciplined layering and sufficient processing horsepower and memory in ISDN terminals.

ISDN Software Glossary

B-CHANNEL—ISDN "Bearer" channel on which digitized voice or user data is transported. The B-channel data rate is 64 kbps.

BRI—Basic Rate Interface. ISDN terminal interface consisting of two B-channels and one D-channel (2B + D). Either voice or data may be transported on either B-channel.

CCITT—International Telegraph and Telephone Consultative Committee developing ISDN protocol standards.

COORDINATING ENTITY—ISDN software layer which coordinates the activities of the D-channel and B-channel(s).

D-CHANNEL—ISDN channel on which messages are exchanged between terminal and network to establish voice and data calls on the B-channel(s). Optionally, the D-channel may be used to transport user data. The D-channel data rate is 64 kbps.

DMI—Digital Multiplexed Interface. A set of D-channel and B-channel protocols for the PBX-to-ISDN primary rate interface. Of interest to the BRI terminal world are the DMI B-channel protocols referred to as "Mode 2" and "Mode 3" which BRI terminals may use to communicate with BRI host computers via ISDN and/or PBXs.

HDLC—High-Level Data Link Control. Prototype bit-oriented layer 2 protocol.

ISO—International Standards Organization.

LAPB—Link Access Protocol Balanced. Layer 2 of X.25. Derived from HDLC.

LAPD—Link Access Protocol on the D-Channel. LAPD is also used as the V.120 layer 2 protocol, and is defined by CCITT Q.921 specification. Derived from HDLC.

MANAGEMENT ENTITY—ISDN software entity that provides operating system services and overall coordination to all software layers.

OSI MODEL—Seven-layered Open Systems Interconnection model developed by the ISO describing hierarchy for organizing communications software.

Q.921—CCITT protocol standard describing LAPD. Primary application is as D-channel layer 2 but may also be used on the B-channel. Also referred to as CCITT Recommendation I.441.

Q.931—CCITT protocol standard describing D-channel layer 3 signalling. Also referred to as CCITT Recommendation I.451.

Signalling—D-Channel layer 3 message exchange between terminal and network to set up and tear down voice and data calls. Signalling messages convey such information as RINGING or BUSY and ANSWERED. See Q.931.

V.120—A statistical multiplexing protocol based on LAPD, for terminal adaptation. V.120 also specifies the terminal interface and the layer 3 and 4 protocols.

X.25 PLP—X.25 Packet Layer Protocol. Layer 3 of X.25 (also used as layer 3 of the DMI protocol).

SOFTWARE AVAILABLE FROM AMD

In order to reduce the development cost and time to market of products using the IDPC, Advanced Micro Devices has designed a series of software packages. These packages are available from AMD for a one time license fee, and include source code, documentation, and unlimited binary distribution rights. The packages are:

Am79LLD401 Low Level Device Driver—The Low-Level Driver provides initialization and full control of the IDPC DLC, creating a clean hardware independent interface to the higher layer software.

AmLink LAPD/LAPB—The AmLink LAPD/LAPB software package works with the IDPC and the Am79LLD401 Low-Level Driver to provide a complete LAPD and LAPB solution. AmLink supports the concurrent operation of multiple channels, using multiple data link controllers.

AmLink3™ Layer 3—The AmLink3 package network layer support for both the ISDN D-channel and the X.25 Packet Layer Protocol (PLP).

Am79LLD401 Low-Level Device Driver

The Low-Level Driver isolates higher layer software from the hardware details of the IDPC. The code is written primarily in 'C' (Microsoft 'C' Compiler version 4.0 or higher), with approximately 5% written in 8088 assembly language (Microsoft Macro Assembler version 5.0 or higher). The LLD is primitive driven, interfacing to the layer 2 software (L2) and the operating system (Management Entity - ME) via mailbox structures. These primitives and mailboxes are described in detail in Chapter 5.

The LLD provides Command primitives that:

- Transmit a Buffer
- Initialize the DLC
- DLC Control
- Update Address Recognition
- Abort the Current Transmit
- Load a New Event Enables
- Begin Remote Loopback
- End Remote Loopback
- Begin Local Loopback
- End Local Loopback

In response to hardware conditions, the LLD generates the following Event primitives:

- Transmission Complete
- Packet Received
- Error Status
- Buffer Allocation Request

AmLink LAPD/LAPB

The AmLink LAPD/LAPB software package combines with the IDPC and the Am79LLD401 Low-Level Driver (LLD) to provide a complete solution for layer 2 of the ISO-OSI seven layer communications model (data link layer). The software is written in 'C' (Microsoft 'C' Compiler version 4.0 or higher) and provides complete operating system independence.

Interfaces—As can be seen in Figure 3-10, the AmLink LAPD/LAPB software interfaces to the LLD, the Management Entity (ME), and the layer 3 software via a set of mailboxes. Command and event primitives are passed between software entities via these mailboxes. The mailbox structure and primitives are described in detail in the AmLink Reference Guide (PID #09529).

Flexibility—One of the key features of the AmLink LAPD/LAPB package is the flexibility to handle multiple logical connections over multiple physical channels, with unlimited window sizes. The re-entrant nature of the software and the configurable nature of the link parameters allows a single body of code to simultaneously support multiple physical devices, as well as multiple logical channels over a single physical channel. This allows the statistical multiplexing of multiple separate conversations over a single physical channel.

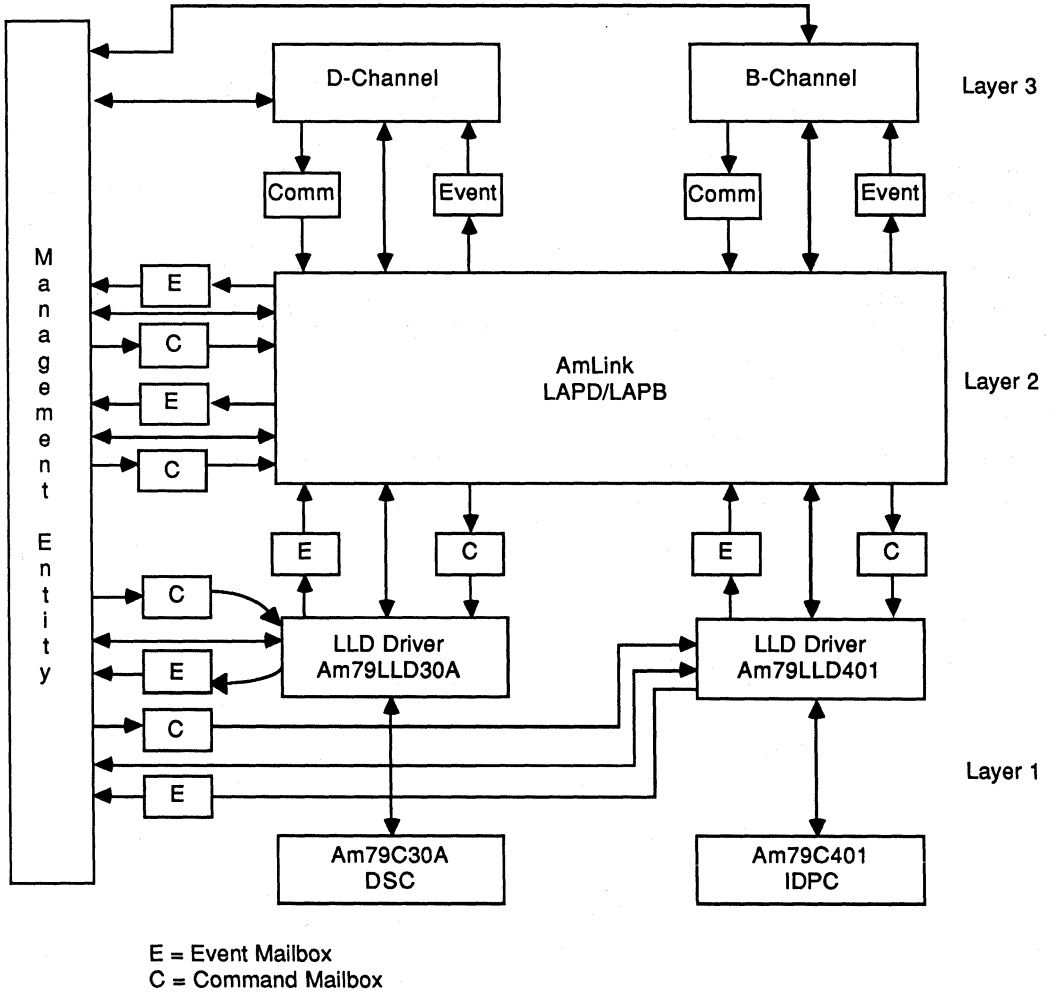


Figure 3-10. AmLink Software Layer Diagram

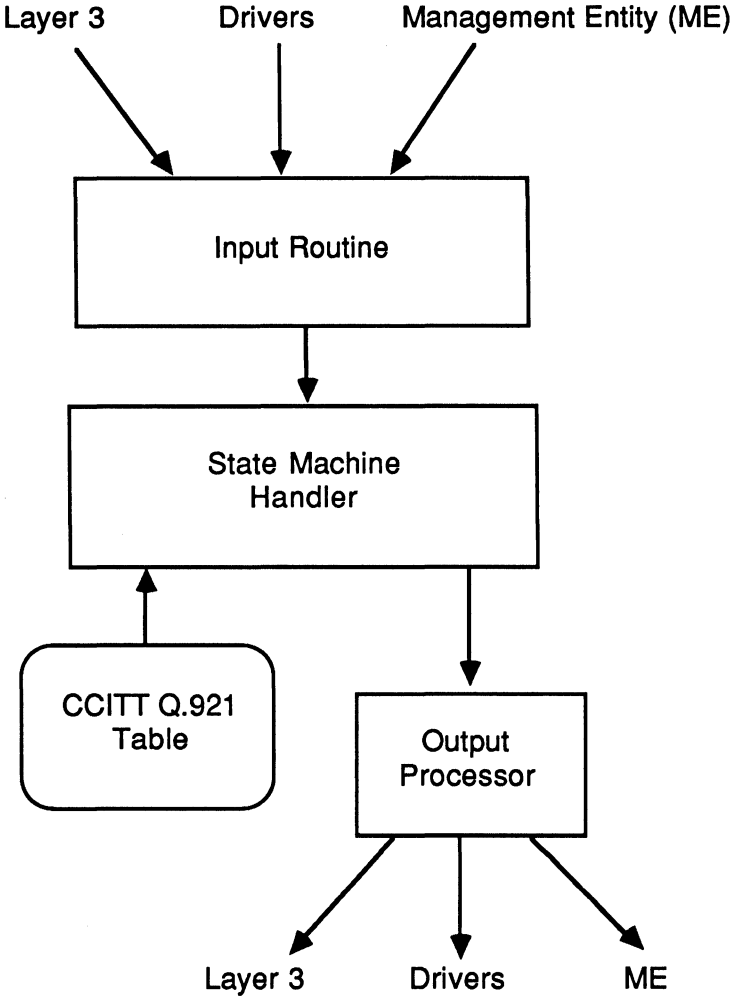


Figure 3-11. AmLink Code Structure

Code Structure—AmLink LAPD/LAPB software uses a state table structure (see Figure 3-11), providing flexibility and maintainability. Inputs can be received from the LLD, ME, or layer 3 entities (via mailboxes). These are processed and fed to the state machine handler. The state machine handler uses these inputs along with the LAPD/LAPB state tables to generate outputs to the output processor. The output processor connects to the LLD, ME, and layer 3 entities via mailboxes.

AmLink3™ Layer 3

The AmLink3 package supports two layer 3 standards, X.25 and Q.931. X.25 is used in both X.25 networks and ISDN (both B- and D-channels). Q.931 is the ISDN call control standard. While the X.25 standard is fairly stable, Q.931 is not. There are significant variations depending on the ISDN switch (PABX or Central Office) in use. For this reason, several versions of the AmLink3 package are available for specific switches.

The software is designed to interface directly to the AmLink LAPD/LAPB primitives using the mailbox structure mentioned above. A similar set of mailboxes and primitives is provided for interfacing to layer 4 software. The mailbox structure and primitives are described in detail in the AmLink3 Reference Guide (PID #10812).

Chapter 4

PROGRAMMING THE IDPC

The IDPC is comprised of three basic modules: the DLC, USART, and DPMC. Each module operates independently of the other modules, and is programmed independently. The following sections cover each module in turn. Each section contains three parts: a discussion of the module's programmable features, a programmable options section discussing their use, and finally, a set of operational sequences providing programming details, including initialization, normal operation, and exception handling.

The IDPC is controlled via internal registers that are written and read by software running on the external "local" processor connected to the IDPC external bus. These internal registers may be mapped into either memory or I/O space, but typically are memory mapped.

The internal registers occupy a 64-byte block located in the local processor's memory address space. The starting address of the memory block is determined by address decode logic (external to the IDPC) that is used to generate the IDPC Chip Select signal (CS/). The registers and their respective memory offset values are listed in Tables 4-1, 4-2, 4-4, and 4-5.

In systems containing more than one microprocessor (e.g., a workstation application with host processor and local processor), normally only the local processor can access the IDPC registers. The host processor, however, can control IDPC operations indirectly by issuing requests to the local processor via shared memory supported by the Dual-Port Memory Controller.

The programmable registers are used for establishing modes of operation, configuring the IDPC, and monitoring/reporting status.

Table 4-1. IDPC Address Map

Offset (Hex)	Block
00-1F	DLC
20-3E	USART
3F	DPMC

DATA LINK CONTROLLER PROGRAMMING

DLC PROGRAMMABLE FEATURES

The DLC is comprised of two sub-blocks: the transmitter and the receiver.

Transmitter Programmable Features

The DLC programmable features include:

- **Transmit Enable**—the transmitter may be disconnected from the output pin, leaving other transmit functions intact (DLC Command/Control Register).
- **Abort**—interrupts a frame by sending at least one Abort character and places the transmitter in the abort condition (DLC Command/Control Register).
- **Flag Idle/Mark Idle**—may be selected as an idle condition between frames (DLC Command/Control Register).
- **CRC Generation**—may be enabled or disabled independent of CRC checking being enabled (DLC Command/Control Register).
- **FIFO Threshold**—user can select threshold of 0 to 15 bytes. When the level of the transmit FIFO falls to this level or below, status is set and a DMA request is generated, unless the last byte of the packet is still in the FIFO (FIFO Threshold Register).
- **Interrupts**—the following transmitter-related interrupts can be selectively enabled and disabled:

- Valid packet sent
- FIFO buffer available
- Transmit threshold reached
- Transmit underrun

Receiver Programmable Features

The DLC receiver programmable features include:

- **Receiver Enable**—when disabled, the receiver is disconnected from the receive data input pin, leaving other receiver functions intact (DLC Command/Control Register).
- **CRC Check**—selectively enables or disables the internal CRC compare operation, independent of CRC generation being enabled (DLC Command/Control Register).
- **CRC Pass Through**—the FCS Field can optionally be placed into the FIFO with the data (DLC Command/Control Register).
- **Address Recognition**—program any combination of four programmable one- or two-byte addresses plus the broadcast address. In the 1-byte mode, either the first or second byte can be selected. The command/response bit (bit 1 of the first byte) can be ignored (optional) (Address Control Register).
- **Minimum Packet Size**—defines the minimum packet size in use. A short frame error is indicated if a packet is received containing fewer than the programmed number of bytes (1-15) (Minimum Receive Packet Size Register).
- **Maximum Packet Size**—defines the maximum packet size in use. This prevents buffer overruns in the event of lost flags or protocol violations (3-65,538) (Maximum Receive Packet Size Register).
- **FIFO Threshold**—select threshold of 2 to 32 bytes. When the level of the Receive FIFO reaches this level or above, status is set and a DMA request is generated (unless the last byte of a packet has already been read from

the FIFO and status for that packet has not yet been read by the user — this forms an interlock that maintains synchronization between packet status and data) (FIFO Threshold Register).

- **Interrupts**—the following DLC receiver interrupts may be selectively enabled or disabled:
- Valid packet received
- Abort received
- Non-integer number of bytes received
- CRC error
- Short frame error
- Long Frame Error
- FIFO buffer overrun error
- Receive threshold reached
- Receive data available
- Change in mark idle
- Change in flag idle
- Change in in-frame
- End-of-packet in receive FIFO

Transmit/Receive Programmable Features

The following programmable features affect both the DLC transmitter and receiver:

- **Inversion**—the output of the transmitter and the input of the receiver will be inverted if this option is selected (Serial Bus Port Control Register).

- **Channel Selection**—up to thirty-one 8-bit time slots for multiplexing transmitted serial data and demultiplexing received serial data may be chosen. In the non-multiplexed mode, received serial data are continuous, not gated, and the SFS/XMITCLK pin is used as a transmit clock input separate from the receive clock input (Serial Bus Port Control Register).

- **Local Loopback**—the DLC can be programmed to route transmitted data to the receiver for diagnostic purposes (Serial Bus Port Control Register).

- **Remote Loopback**—The DLC can be programmed to route received data to the transmit data output for remote testing capabilities (Serial Bus Port Control Register).

- **Reset**—a software reset can be generated to stop all functions, clear the FIFOs, and set all registers to their default values (Command/Control Register).

DLC Register Map

The DLC contains 23 registers, as shown in Table 4-2.

DLC Programmable Operations

The following section provides an introduction to programming the DLC to perform basic operations, including:

- Address recognition
- DMA operation
- Non-DMA operation
- Receive packet status stacking mechanism
- Receive packet status processing

Address Recognition

The DLC receiver can be programmed to inspect the addresses of incoming packets. If an address match occurs, the DLC will receive the packet, if no match occurs, the packet is ignored. The following programmable options are available:

First Byte Address Detection—The first byte after the opening flag is inspected.

Table 4-2. DLC Registers

Offset (Hex)	Register Name	Size (Bytes)	Type
00	Command/Control Register	1	Read/Write
01	Address Control Register	1	Read/Write
02	Link Address Recognition Register 0	2	Read/Write
04	Link Address Recognition Register 1	2	Read/Write
06	Link Address Recognition Register 2	2	Read/Write
08	Link Address Recognition Register 3	2	Read/Write
0A	Serial Bus Port Control Register	1	Read/Write
0B	Minimum Receive Packet Size Register	1	Read/Write
0C	Maximum Receive Packet Size Register	2	Read/Write
0E	Interrupt Source Interrupt Enable Register	1	Read/Write
0F	Receive Frame Interrupt Enable Register	1	Read/Write
10	Receive Link Interrupt Enable Register	1	Read/Write
11	FIFO Status Interrupt Enable Register	1	Read/Write
12	Transmit Byte Count Register	2	Read/Write
14	FIFO Threshold Register	1	Read/Write
15	Interrupt Source Register	1	Read Only
16	Receive Byte Count Register	2	Read Only
18	Receive Frame Status Register	1	Read Only
19	Receive Link Status Register	1	Read Only
1A	FIFO Status Register	1	Read Only
1B	Receive FIFO Data Register	1	Read Only
1C	Transmit FIFO Data Register	1	Write Only
1D	Residual Bit Control Status Register	1	Read/Write
1E-1F	Reserved	2	—

Second Byte Address Detection—The second byte after the opening flag is inspected.

First Two Byte Address Detection—Both the first and second bytes after the opening flag is inspected.

Ignore Command/Response Bit—In some protocols, the second bit in the first address byte (Bit 1) is used to indicate both whether the packet is a command or a response. The command/response bit is not considered as part of the address field. The DLC can be programmed to optionally ignore the command/response bit when making address comparisons.

The DLC has five address detectors. Four of these are user programmable, the fifth is hard-wired to an all ones value (broadcast). The command/response bit (Bit 1) of the broadcast address detector can optionally be ignored. Additionally, in two byte mode, Bit 0 of the first broadcast address byte (extended address bit) is expected to be zero. Each address detector can be enabled or disabled. Additionally, if address recognition is disabled, all packets are received regardless of their address.

Programming—Table 4-3 lists the various addressing options, and the registers/bits that are used to program them.

Table 4-3. Addressing Options

One/two byte addressing	Bit 5, DLC Address Control Register
First/Second byte (one byte mode)	Bit 7, DLC Address Control Register
Command/Response bit checking	Bit 6, DLC Address Control Register
Address detector #0 enable	Bit 0, DLC Address Control Register
Address detector #1 enable	Bit 1, DLC Address Control Register
Address detector #2 enable	Bit 2, DLC Address Control Register
Address detector #3 enable	Bit 3, DLC Address Control Register
Broadcast address detector enable	Bit 4, DLC Address Control Register

Addresses are programmed into the four address detectors via the four Link Address Registers (two bytes each).

Address Reporting—When a packet is received, the identification of the address detector that matched the packets address is reported via a three-bit field in the Interrupt Source Register.

000 Address Detector 0
001 Address Detector 1
010 Address Detector 2
011 Address Detector 3
100 Broadcast Detector
101 Not Used
110 No Packet Received
111 Packet Received, All Address Detectors Disabled

DMA Operation

DMA can be used to move data in and out of the DLC FIFOs. Each FIFO generates an output signal that indicates to a DMA controller that data need to be moved. The receive FIFO generates DRQ_0, the transmit FIFO generates DRQ_1.

Receiver DMA Operation—The DRQ_0 signal is activated by two conditions, the level in the Receive FIFO rising to the programmed threshold, or the last byte of a packet being placed in the FIFO. In the case where the FIFO threshold caused the activation, DRQ_0 will remain active until the FIFO is empty. In the case of an end of packet (EOP), DRQ_0 will be active only until the last byte of the packet has been read from the FIFO— regardless of the level of data in the FIFO, or of the presence of additional EOP indicators in the FIFO. In the case where DRQ_0 was activated by an EOP condition, it will remain inactive until the least significant byte of the Receive Byte Count Register is read, preventing data from the next packet from being read out of the FIFO until the status of the current packet is read from the stacked status registers (refer to the description of delayed-stacked status reporting). This insures that the status information for a given packet stays in synchronization with that packet's data.

The DLC receiver does not require an acknowledge signal from the DMA. This is because the DMA reads the Receive FIFO at the beginning of the DMA cycle, and then writes the data to memory in the second half of the DMA cycle. This gives the FIFO sufficient time to deactivate DRQ_0 when the last byte of data (or EOP byte) is read from the FIFO, preventing the DMA from attempting a follow-on read of the now empty FIFO.

Transmitter DMA Operation—Unlike the receive FIFO, the transmit FIFO is set up to allow data from only one packet to be in the FIFO at one time. This greatly simplifies operation in the event of an abort or underrun condition. This does not prevent the DMA controller from being set up to transfer several packets at a time. The DRQ_1 signal will automatically control the loading of the FIFO such that data from only one packet at a time will be moved into the FIFO.

The DRQ_1 signal is activated when the transmit byte counter is not zero, and the level in the FIFO is at or below the programmed threshold value. DRQ_1 is de-activated when either the TBC reaches zero (the last byte of the packet is placed into the FIFO), or the FIFO becomes full. If DRQ_1 was de-activated by the loading of the last byte of a packet (TBC = 0), it will be reactivated as soon as this last byte is moved out of the FIFO into the parallel-to-serial shift register. Assuming CRC operation, this gives four character times to load the first byte of a new packet into the FIFO to insure the transmission of back-to-back packets.

Unlike the receive FIFO, an acknowledge signal may be required from the transmit DMA Controller. This is because the transmit FIFO is loaded at the end of a DMA cycle (the receive FIFO is serviced at the start of a DMA cycle). Insufficient time is available, once the last Write operation to the FIFO is started, to deactivate DRQ_1 before an additional (unwanted) DMA cycle is started. If the DMA Controller provides an acknowledge signal, this can be connected to the IDPC's DACK/ pin. (DMA acknowledge signals are generated at the beginning of the DMA cycle, providing time to deactivate DRQ_1.) If an acknowledge signal is not available, or cannot be constructed, a wait-state can

be added to the DMA cycle. This will provide sufficient time for the FIFO to detect that a byte is being written into the FIFO Data Register, and deactivate DRQ_1.

Non-DMA Operation

In systems where DMA is not used to move data in and out of the DLC FIFOs, the microprocessor must transfer the data.

Receive FIFO—Data is moved out of the receive FIFO in response to two interrupts: FIFO threshold reached, and end-of-packet in receive FIFO. Both of these interrupts are reported via the FIFO Status Register. In response to a FIFO threshold reached interrupt, the number of bytes to be read is known, the value programmed into the receive FIFO threshold field in the FIFO Threshold Register. In this case, the microprocessor can execute a string move instruction, moving that number of bytes. In the case of an end-of-packet in receive FIFO interrupt, the number of bytes to be moved is not known (it will always be less than or equal to the threshold value). The procedure for unloading the receive FIFO is as follows: the microprocessor reads the Receive FIFO Data Register and stores the byte in memory. Then the data available bit in the FIFO Status Register is tested. Data is moved from the receive FIFO as long as the data available bit is set. The receive FIFO is designed to de-activate the data available bit when the last byte of the packet is read from the receive FIFO, even if there are additional data in the FIFO (these data would belong to a new packet.) The de-activation of the data available bit identifies the packet boundary, indicating that it is time to service the packet status information. When the packet status is serviced, the receive byte counter is always read last. (Reading the least significant byte of the receive byte counter clears the status registers of the data for the packet). When the least significant byte of the receive byte counter is read, the data available bit will return active if there are data from a new packet in the receive FIFO. This mechanism insures that packet data and status are synchronized. A description of the delayed-stacked status mechanism is provided below.

Transmit FIFO—The transmit FIFO is serviced in response to a transmit FIFO threshold reached interrupt. If the number of bytes yet to be loaded into the transmit FIFO exceeds the programmed threshold level (FIFO Threshold Register), a string move can be used to move the data. For example, if the threshold is programmed at 2, a threshold reached interrupt indicates that 14 bytes can be loaded into the transmit FIFO. If the number of bytes remaining to be loaded is fewer than 16 minus the programmed threshold value, or if the number of bytes to be loaded is unknown, a polled procedure is required. In this case, a byte is loaded into the transmit FIFO and then the buffer available bit in the FIFO Status Register is tested. If it is active, another byte can be loaded into the transmit FIFO. The buffer available bit will remain active as long as the transmit FIFO is not full -AND- the transmit byte counter is not zero. (The TBC counts down to zero when all of the data for a given packet have been loaded.) The buffer available bit becoming inactive indicates that the transmit FIFO is full, or that the last byte of the packet has been loaded into the transmit FIFO. If the buffer available bit became inactive because of the TBC reaching zero, it will remain inactive until the last byte of the packet has moved from the transmit FIFO into the serial-to-parallel shift register. This prevents data from more than one packet at a time from being placed into the transmit FIFO.

Receive Packet Status Stacking Mechanism

The DLC receiver contains a mechanism that allows multiple packets to be received without losing the unprocessed status information for previously received packets. Up to four packets can be received before the status of the first packet must be processed. If the status for the first packet has not been processed by the time the closing flag of the fourth packet is detected, the receiver will be prevented from receiving additional packets. The receiver will be re-enabled when the status for the first packet is processed. This status log is referred to as the status stack. All registers or portions of registers that report status information on received packets are stacked registers; these include: The Receive Byte Count Register, the Receive Frame Status Register, the link address and valid packet received bit fields of the Interrupt Source Register, and the received bit residue count field of the Residual Bit Status Control Register.

These registers and bit fields are cleared when they are read. Additionally, they are cleared when the least significant byte of the Receive Byte Count Register is read. In most cases, the receive packet processing software will need to read only the Interrupt Source Register and the Receive Byte Count Register—the Receive Frame Status Register contains error conditions, and needs only to be read if the receive frame status bit (and, therefore, not the valid packet received bit) is set in the Interrupt Source Register, and the received bit residue count field needs only to be read if the protocol in use allows packets to contain a non-integer number of bytes. By clearing out any unread status for the packet when the least significant byte of the Receive Byte Count Register is read, synchronization is maintained between the various status registers.

Receive Packet Status Processing

The receiver presents the status of a received packet to the microprocessor after the packet has been completely received and all of the packet data have been stored in memory. The movement of the last byte of packet data out of the FIFO is the trigger that allows the packet status to be presented to the microprocessor. Interrupts, if enabled, are generated at this time.

Sequence Of Events—In response to a DLC interrupt, the microprocessor will read the Interrupt Source Register. The ISR contains the following packet status information:

- The identification of the address detector that matched the address of the received packet.
- Two bits indicating whether the packet is valid or not.

If the valid packet received bit is set, all that remains for the user to do is to read the Receive Byte Count Register pair. Even in cases where the size of the received packet is known, the least significant byte of the RBCR must be read—reading the LSB of the RBCR clears any unread receive packet status registers.

If the received packet contains an exception condition (CRC error, short frame error, long frame error, abort, non-interger number of bytes [not always an error] or receive FIFO overrun error [causes the packet to be terminated]), the valid packet received bit will not be set; instead, the

receive frame status bit will beset—The Receive Frame Status Register contains only exception conditions.

If the software, upon reading the Interrupt Source Register, finds the receive frame status bit set, the Receive Frame Status Register should be read to determine the exception condition.

After determining the status of the packet, the Receive Byte Count Register is read to determine the size of the received packet. Even if the size is known, the least significant byte of the Receive Byte Count Register must be read in order to clear-out unread status from any of the registers.

Packet Transmission Sequence

The DLC transmitter is designed to work both with and without DMA support. When DMA is used, there are two possible modes of operation: 1) transmitting one packet at a time, 2) transmitting a queue of packets. From the transmitter's point of view, there is no difference between the two modes, the only difference is in how the DMA controller is programmed.

Transmitter Operation—Independent of how data are loaded into the transmit FIFO (DMA or processor controlled I/O) a series of basic operations takes place within the transmitter. The first step in transmission of a packet is to program the length of the packet into the Transmit Byte Count Register—TBCR (the length includes the address, control, and information fields, but not flags or the FCS field). Writing to the TBCR causes the contents of the TBCR to be loaded into a counter, the Transmit Byte Counter - TBC. As soon as the TBC becomes non-zero, the DMA Request 1 (DRQ_1) pin is activated, indicating that the transmit FIFO is ready to receive data. When the first byte of data (typically the first address byte) is loaded into the transmit FIFO, the transmitter starts sending the opening flag. As soon as the opening flag leaves the parallel-to-serial shift register, the first byte of data loaded into the FIFO is moved into the shift register. The TBC is decremented each time a byte is loaded into the transmit FIFO. The DRQ_1 signal will remain active until the transmit FIFO becomes full, or the TBC counts down to zero. Assuming that the length of the packet exceeds the 16 byte depth of the transmit FIFO, DRQ_1 will be reactivated when the level in the transmit FIFO falls to the programmed threshold level, programmed in the FIFO Threshold Register. Once the last byte of the packet is loaded into the transmit FIFO, causing the TBC to be zero, DRQ_1 is de-activated, preventing the DMA from loading additional data into the FIFO. DRQ_1 will remain inactive until the last byte of the packet is loaded into the serial-to-parallel shift register - this prevents data from more than one packet from being in the transmit FIFO at any one time. When the last byte is moved into the shift register, the TBCR automatically reloads the TBC, allowing DRQ_1 to return active since the TBC will no longer contain a zero value. If the user loads a new value into TBCR while the transmitter is transmitting a packet, the TBCR will hold off loading this value into the TBC until the last byte of the packet leaves the transmit FIFO. If the transmit FIFO is empty, the TBCR will automatically load the TBC any time the TBCR is written to by the user.

Transmitting One Packet At A Time Using DMA—To transmit packets one at a time, the DMA controller is programmed to move only the number of bytes in the packet. The TBCR is programmed with this same value; the transmitter needs to know when to end the packet. If the user desires, the DMA controller can be programmed to generate an interrupt when the last byte of the packet is moved into the transmit FIFO. The DMA controller can then be set up to send a new packet, and the TBCR can be reloaded with the length of the new packet. When the last byte of the packet leaves the transmit FIFO, DRQ_1 will be reactivated, and the DMA controller will start loading the transmit FIFO with the new packet.

Transmitting A Queue Of Packets—This can be done only if all of the individual packets in the queue are the same length. The DMA controller is programmed with the total number of bytes in the queue. The TBCR is then programmed with the length of a packet. The transmit FIFO will control the movement of data via DRQ_1. When the last byte of the last packet is loaded into the transmit FIFO, the DMA controller will have counted to the total number of bytes it had been programmed with, and will stop the movement of data. In this way, a queue of packets can be constructed in memory (complete with address and control fields). These packets can then be transmitted without intervention by the microprocessor.

NOTE: The number of packets in the queue is limited by the window size of the protocol in use. Window size refers to the maximum number of packets that can be transmitted before the first packet is acknowledged.

DLC Operational Sequences

The IDPC operational sequences in the sections that follow provide detailed examples of programming the major functional components of the IDPC.

All of the operational sequences except the host part of the interprocessor interrupts sequences are assumed to be executed on an 80188 processor (local processor). The local processor software consists of a set of Interrupt Service Routines (ISVRs) and main loop code that executes when the ISVRs are not.

The operational sequences described below illustrate in detail the programming of the IDPC Data Link Controller (DLC) hardware by an 80188 local processor in a typical application scenario. This scenario assumes:

A) The DLC is used to perform bit-oriented protocol processing.

B) DMA used for both DLC frame reception and transmission. 80188 DMA Channel 0 is used for DLC reception; 80188 DMA Channel 1 is used for DLC transmission.

C) The IDPC DLCINT output pin is connected to one of the local 80188 INTX (INT0-INT3) maskable interrupt input pins to form the DLC interrupt.

D) The local processor has initialized its interrupt controller hardware and interrupt vectors during reset, enabling the external DLC interrupt and internal 80188 DMA Channel 1 interrupts in the process.

E) 80188 DMA Channel 0 interrupt NOT used to indicate packet received; DLC interrupt (valid or receive frame status exception packet received) used for this purpose. This is because only the DLC interrupt can indi-

cate reception of variable length packets and/or packets received with errors.

F) Either the DLC interrupt (valid packet sent), or 80188 DMA Channel 1 interrupt may be used to notify the processor of successful packet transmission. The DMA interrupt is more general in that it allows more than one packet to be transmitted per interrupt. For this reason, the DMA interrupt is used for DLC transmission in this scenario. (If multiple packets are to be transmitted per interrupt, all packets must be the same length and contiguous in memory.)

NOTE: For scenarios in which the DLC interrupt is used for BOTH packet reception and transmission, the DLC interrupt service routine must check both receive and transmit status in the same read of the DLC Interrupt Source Register since one read of that register clears it.

G) Several interrupts that are useful for non-DMA programmed I/O or diagnostic testing are not enabled for regular operation in this scenario.

Refer to the iAPX 86/88,186/188 User Manual Volume 1: Programmer's Reference for descriptions of 80188 DMA and interrupt controller operation.

The DLC operational sequences for this scenario are:

Operational Sequences

DLC Link Initialization (after call setup)
DLC Transmit Packet(s)
DLC Receive Packet—Normal
DLC Receive Packet—Exception

These operational sequences are interdependent. For example, the DLC link initialization sequence must be executed before the DLC transmit packet(s) or receive packet sequences can be executed.

Protocol processing performed by software (e.g., packet sequence number checking, acknowledge (ack) packet transmission) is not described in the DLC operational sequences. Only hardware level processing is described.

Link Initialization

DLC register bits that are flagged in the steps below with an asterisk (*) are configuration dependent. In this operational sequence, these bits are set to values that are arbitrary for this example. Setting such bits to values other than those indicated does not change the validity of this operational sequence.

1. Write the DLC Command/Control Register with the following contents to reset the DLC:

Bit	Value	Function
0	Don't care	
1	Don't care	
2	Don't care	
3	Don't care	
4	Don't care	
5	Don't care	
6	1	Enable DLC Reset
7	Don't care	

2. Write DLC Link Address Recognition Register 0 with the 16-bit B-Channel Layer 2 link address negotiated during call setup.

3. Write the DLC Address Control Register with the following contents:

Bit	Value	Function
0	1	Enable Logical Link 0 Address Recognition
1	0	Disable Logical Link 1 Address Recognition
2	0	Disable Logical Link 2 Address Recognition
3	0	Disable Logical Link 3 Address Recognition
4	0	Disable Broadcast Address Recognition
5	0*	Enable Two-Byte Address Recognition
6	0*	Ignore Command/Response bit
7	0*	First/second byte selection—ignored for two byte address

4. Write the DLC Serial Bus Port Control Register with the following contents:

Bit	Value	Function
0	0*	
1	1*	Select channel 2
2	0*	
3	0*	
4	0*	
5	1*	Invert data
6	0	Disable Local Loop back
7	0	Disable Remote Loop back

5. Write hex E2 (*) (receive FIFO threshold=28 and transmit FIFO threshold=2) to the DLC FIFO Threshold Register.

6. Write six (*) to the DLC Minimum Receive Packet Size Register. This value was negotiated during call setup or by local administration.

7. Write 135 (*) to the DLC Maximum Receive Packet Size Register. (Four bytes L2 header, four bytes L3 header, 128 bytes L3 I-field,and two bytes CRC, minus 3 bytes [the DLC Maximum Receive Packet Size Register is always programmed with a value that is 3 less than the desired maximum packet size].) This value was negotiated during call setup or by local administration.

8. Write the DLC Interrupt Source Interrupt Enable Register with the following contents:

Bit	Value	Function
0-2	Don't Care	Spare
3	1	Enable interrupt on Valid Packet Received
4	0	Disable interrupt on Valid Packet Send
5	1	Enable interrupt on Receive Frame Status Error
6	1	Enable interrupt on FIFO Status Register bit set
7	0	Disable interrupt on Receive Link Status bit set

9. Write the DLC Receive Frame Interrupt Enable Register with the following contents:

Bit	Value	Function
0	1	Enable interrupt on Abort Received
1	1	Enable interrupt on Non-Integer Number of Bytes Received error
2	1	Enable interrupt on Received CRC error
3	1	Enable interrupt on received byte count less than DLC Minimum Receive Packet Size Register error (Short Frame error)
4	1	Enable interrupt on received byte count greater than DLC Maximum Receive Packet Size Register error (Long Frame error)
5	1	Enable interrupt on receive Overrun error
6-7	Don't Care	Spare

10. Write the DLC Receive Link Status Interrupt Enable Register with the following contents:

Bit	Value	Function
0	0	Disable interrupt on Mark Idle detection
1	0	Disable interrupt on Flag Idle detection
2	0	Disable interrupt on In-frame detection
3-7	Don't care	Spare

11. Write the DLC FIFO Status Interrupt Enable Register with the following contents:

Bit	Value	Function
0	0	Disable Interrupt on Receive FIFO Threshold Reached
1	0	Disable Interrupt on Receive FIFO Data Available
2	0	Disable Interrupt on Transmit FIFO Threshold Reached
3	0	Disable Interrupt on Transmit FIFO Buffer Available
4	1	Enable Interrupt on Transmitter Underrun
5	0	Disable Interrupt on EOP in Receive FIFO
6-7	Don't care	Spare

12. Set up the 80188 DMA channel (channel 0) dedicated to receiving frames by initializing the DMA Channel 0 Control Word with the following:

Bit	Value	Function
0	0	Byte Transfer
1	1	Start DMA
2	1	Change bit
3	1	Don't care
4	0	Disable DMA requests from 80188 timer 2
5	1	Receive DMA has higher priority than transmit DMA
6	1	Source Synchronized
7	0	
8	0	Don't interrupt CPU on Transfer Count termination
9	1	Terminate DMA if Transfer Count reaches zero
10	1	Don't change source pointer after each transfer
11	1	
12	1	Source pointer is in memory space
13	1	Increment destination pointer after each transfer
14	0	Do not decrement destination pointer
15	1	Destination pointer is in memory space

13. Allocate from a queue of empty buffers in RAM (or a stack, etc.) a B-Channel receive buffer big enough to hold at least one maximum length packet for the protocol in use, 138 bytes in this example.

14. Load the 80188 DMA Channel 0 Transfer Count Register with the size of the allocated receive buffer. Although the DMA Channel 0 interrupt is not used, the Channel 0 DMA operation is halted in the exceptional event that the Transfer Count reaches zero. This provides a fail-safe mechanism to prevent received frame bytes from overwriting memory past the allocated receive buffer boundary.

15. Load the 80188 DMA Channel 0 Destination Pointer Register with the starting RAM address of the allocated receive buffer.

16. Write the IDPC DLC Receive FIFO Data Register address to the 80188 DMA Channel 0 Source Pointer Register. This address never changes during IDPC operation. This step is thus an example of a DLC initialization step that can be performed once at 80188 reset instead of during every call as in this scenario.

17. Write the IDPC DLC Transmit FIFO Data Register address to the 80188 DMA Channel 1 Destination Pointer Register.

18. Write the DLC Command/Control Register with the following contents:

Bit	Value	Function
0	0	Do not Send Abort
1	1	Transmitter Enable
2	1	Receiver Enabled
3	1*	Flag Idle
4	1*	Enable CRC Check
5	1*	Enable CRC Generate
6	0	Disable DLC Reset
7	0*	Do not pass FCS through to the Receive FIFO

At this point, packets may be transmitted and received.

19. Continuously poll the DLC Receive Link Status Register detected by the DLC receiver. Do this as a precondition for transmitting since the destination end point probably is not yet ready to receive if it is not yet transmitting the proper Idle pattern. The destination end point may be slightly slower than the originating call end point, or vice versa, in starting up the channel.

Transmit Packet(s)

NOTE: Multiple packets may be transmitted in one execution of the following steps with the restrictions that the packets must be contiguous in memory and each packet must be of identical length. If successive packets are either not contiguous or are of different lengths, then the following processing steps must be repeated for each packet.

1. Format packet(s), including headers, somewhere in the local processor's addressable memory.

2. Write the first packet's starting memory address in the 80188 DMA Channel 1 Source Pointer Register.

3. Write the SUM of the lengths of the packets to be transmitted (not including FCS bytes or flags) into the 80188 DMA Channel 1 Transmit Count Register.

4. If the size of each packet to be transmitted is different from the last packet transmitted, write the packet size (not including FCS bytes or flags) to the DLC Transmit Byte Count Register. Note that this size is not the sum of all packets to be transmitted as in step 3, above, but rather the individual packet size.

5. Start 80188 DMA channel 1 by writing the DMA 1 control word with the following:

Bit	Value	Function
0	0	Byte Transfer
1	1	Start DMA
2	1	Change bit
3	1	Don't care
4	0	Disable DMA requests from 80188 timer 2
5	0	Transmit DMA has lower priority than receive DMA
6	0	Destination Synchronized
7	1	
8	1	Interrupt CPU on Transfer Count termination
9	1	Terminate DMA when Transfer Count reaches zero
10	1	Increment source pointer after each transfer
11	0	
12	1	Source pointer is in memory space
13	1	Don't change destination pointer after each transfer
14	1	
15	1	Destination pointer is in memory space

6. The DLC transmits the packets without further local processor intervention. When all packets have been moved via DMA to the DLC, the 80188 DMA Channel 1 interrupt occurs. The interrupt service routine pointed to by the 80188 DMA Channel 1 interrupt vector is invoked.

7. The 80188 DMA Channel 1 interrupt service routine writes a non-specific end-of-interrupt command (hex 8000) to the 80188 interrupt controller EOI Register.

8. (Optional) If additional data are available for transmission, repeat steps 1-5. These steps may be performed immediately at this point in the 80188 DMA Channel 1 interrupt service routine if the DLC is running at a low data rate (e.g., 64 Kbps), or if relatively little processing is required. For example, only steps 2-5 need to be executed if additional packets have already been formatted (packet headers set up) during main loop execution.

   For an alternative to this, the interrupt service routine may not perform any of steps 1-5 at this point. Rather, the interrupt service routine may simply set a global flag in RAM indicating that the 80188 DMA Channel 1 is idle. The local processor main loop, during a periodic poll of this global flag, detects that the flag is set and initiates the next DLC transmission.

9. The interrupt service routine executes an Interrupt Return (IRET) 80188 instruction to exit.

Receive Packet—Normal

1. When the DLC receive logic detects that a packet has been received (closing flag detected), with no errors, the valid packet received bit is set in the DLC Interrupt Source Register. Since this interrupt was enabled in the DLC Interrupt Source Interrupt Enable Register during DLC initialization, the 80188 is interrupted and vectors to the DLC interrupt service routine (DLC ISVR).

2. The DLC ISVR reads the DLC Interrupt Source Register to determine the specific reason for the interrupt. Since this read clears the receive status in the Interrupt Source Register, the ISVR saves this value temporarily in scratchpad RAM so that the receive link address field in the register can be used during packet header processing later.

3. The DLC ISVR determines that the valid packet received bit is set in the DLC Interrupt Source Register. DLC design insures that if this bit is set, no DLC receive exception status bits are set. Thus, no further DLC receive status checking is required.

4. The DLC ISVR immediately stops 80188 DMA Channel 0 by writing the DMA Channel 0 control word with the following:

Bit	Value	Function
0	Don't care	
1	0	Stop DMA
2	1	Change bit 1
3-15	Don't care	

   This stops DMA Channel 0 from activating its data request signal and thus forces the DLC receive FIFO to buffer the next incoming packet until the DMA channel is reinitialized in steps 7-10 below.

5. The ISVR reads the DLC Receive Byte Count Register and temporarily saves in RAM this count of bytes received in the current packet.

6. At this point, some implementations will process the received packet in its entirety before continuing with step 7. This processing includes making sure frame headers and lengths are valid, formatting and transmitting any ack packet, and moving any user data out of the receive frame buffer.

   Other implementations (e.g., at high DLC bit rates) would not perform Layer 2 and above processing during interrupt service routine execution. Rather, these implementations would place the received packet in a queue of unprocessed packets. This queue would be unloaded by receive packet processing software that is not part of the ISVR. This receive packet processing software would be called periodically from the local processor main software loop.

7. The DLC ISVR allocates from a queue of empty buffers in RAM (or a stack, etc.) a B-Channel receive buffer big enough to hold at least one maximum length packet for the protocol in use.

8. The DLC ISVR loads the 80188 DMA Channel 0 Transfer Count Register with the size of the allocated receive buffer. Although the DMA Channel 0 interrupt is not used, the Channel 0 DMA operation will be halted in the exceptional event that the transfer count reaches zero. This provides a fail-safe mechanism to prevent received frame bytes from overwriting memory past the allocated receive buffer boundary.

9. The DLC ISVR loads the 80188 DMA Channel 0 Destination Pointer Register with the starting RAM address of the allocated receive buffer.

10. The DLC ISVR restarts 80188 DMA Channel 0 by writing the DMA Channel 0 Control Word with the following:

Bit	Value	Function
0	0	Byte Transfer
1	1	Start DMA
2	1	Change bit
3	1	Don't care
4	0	Disable DMA requests from 80188 timer 2
5	1	Receive DMA has higher priority than transmit DMA
6	1	Source Synchronized
7	0	
8	0	Don't interrupt CPU on Transfer Count termination
9	1	Terminate DMA if Transfer Count reaches zero
10	1	Don't change source pointer after each transfer
11	1	
12	1	Source pointer is in memory space
13	1	Increment destination pointer after each transfer
14	0	Do not decrement destination pointer
15	1	Destination pointer is in memory space

11. The DLC ISVR writes a non-specific end-of-interrupt command (hex 8000) to the 80188 interrupt controller EOI Register and executes an 80188 IRET instruction to exit.

Receive Packet—Exception

The operational sequence described in this section may be used to handle each of the following DLC receiver exceptional conditions:

- Abort Received (DLC Receive Frame Status Register bit 0)
- Non-Integer Number of Bytes Received (DLC Receive Frame Status Register bit 1)
- CRC Error (DLC Receive Frame Status Register bit 2)
- Short Frame Error (DLC Receive Frame Status Register bit 3)
- Long Frame Error (DLC Receive Frame Status Register bit 4)
- Overrun Error (DLC Receive Frame Status Register bit 5)

The only difference in how these exceptions are handled is that IDPC software may wish to increment a separate counter in RAM for each exception type for maintenance/ diagnostic purposes (see step 10 below).

1. When the DLC receive logic detects that one of the exception conditions has occurred, the corresponding bit is set in the DLC Receive Frame Status Register and the receive frame status bit is set in the DLC Interrupt Source Register. Since DLC interrupt generation on occurrence of each of these exception conditions was enabled during DLC initialization, the 80188 is interrupted and vectors to the DLC interrupt service routine (DLC ISVR).

2. The DLC ISVR reads the DLC Interrupt Source Register to determine the specific reason for the interrupt.

3. The DLC ISVR determines that the receive frame status bit is set in the DLC Interrupt Source Register. This causes the ISVR to read the Receive Frame Status Register.

4. The DLC ISVR immediately stops 80188 DMA Channel 0 by writing the DMA Channel 0 control word with the following:

Bit	Value	Function
0	Don't care	
1	0	Stop DMA
2	1	Change bit 1
3-15	Don't care	

This stops DMA Channel 0 from activating its data request signal and thus forces the DLC receive FIFO to buffer the next incoming packet until the DMA channel is reinitialized in steps 6-9 below. Some data may already have been moved via DMA from the DLC to memory when the exceptional condition occurs. In this scenario, the DLC ISVR assumes that the current receive buffer pointed to by the 80188 DMA Channel 0 Destination Pointer Register has been corrupted when a receive frame status exception occurs. For this reason, the DLC ISVR reinitializes the 80188 DMA Channel 0.

Since there is nothing meaningful that can be done with the corrupted buffer, the buffer is returned to empty status by the DLC ISVR (i.e., the buffer is placed on a free list of available buffers). This effectively discards the partially received packet it contains.

5. The ISVR reads the DLC Receive Byte Count Register to clear it.

6. The DLC ISVR allocates from a queue of empty buffers in RAM (or a stack, etc.) a receive buffer big enough to hold at least one maximum length packet for the protocol in use.

7. The DLC ISVR loads the 80188 DMA Channel 0 Transfer Count Register with the size of the allocated receive buffer. Although the DMA Channel 0 interrupt is not used, the Channel 0 DMA operation will be halted in the unlikely event that the count decrements to zero to prevent received frame bytes from overwriting memory past the allocated receive buffer boundary.

8. The DLC ISVR loads the 80188 DMA Channel 0 Destination Pointer Register with the starting RAM address of the allocated receive buffer.

9. The DLC ISVR restarts 80188 DMA Channel 0 by writing the DMA Channel 0 control word with the following:

Bit	Value	Function
0	0	Byte Transfer
1	1	Start DMA
2	1	Change bit
3	1	Don't care
4	0	Disable DMA requests from 80188 timer 2
5	1	Receive DMA has higher priority than transmit DMA
6	1	Source Synchronized
7	0	
8	0	Don't interrupt CPU on Transfer Count termination
9	1	Terminate DMA if Transfer Count reaches zero
10	1	Don't change source pointer after each transfer
11	1	
12	1	Source pointer is in memory space
13	1	Increment destination pointer after each transfer
14	0	Do not decrement destination pointer
15	1	Destination pointer is in memory space

10. (Optional) The DLC ISVR increments a counter in RAM for the exception that occurred. If this counter reaches a certain threshold, the IDPC software may notify a higher level of software that a certain exception is occurring too frequently.

11. The DLC ISVR writes a non-specific end-of-interrupt command (hex 8000) to the 80188 interrupt controller EOI Register and executes an 80188 IRET instruction to exit.

USART PROGRAMMING

USART Programmable Features

General USART Features

The programmable features of the USART include:

- **Character Length**—5-, 6-, 7-, or 8-bit character sizes may be selected.

- **Parity**—even, odd, and no parity modes may be selected. Additionally, a "stick" parity test mode is available (Line Control Register).

- **Stop Bits**—1 or 2 stop bits may be selected for 6-, 7-, or 8-bit characters; 1 or 1-1/2 stop bits may be selected for use with 5-bit characters (Line Control Register).

- **Handshake Lines**—the USART provides RTS and DTR assertion through software control and allows for status checking of CTS and DSR. The signal names have been assigned the Data Terminal Equipment (DTE) designations in order to be compatible with the 8250 UART. This is simply a naming convention, and does not

prevent the USART from functioning as Data Communication Equipment (DCE).

- **Operational Modes**—the USART may be programmed for asynchronous or synchronous/transparent operation (USART Control Register). The asynchronous mode is similar to that of any UART. The synchronous/transparent mode allows data to be transmitted and received without respect to framing or protocol. In this mode, the USART appears as a simple shift register that transmits eight-bit characters back-to-back from the transmit FIFO, without start or stop bits. Similarly, the data are received eight bits at a time and placed into the receive FIFO. Any framing bits present in the data stream are treated as data and placed into the receive FIFO.

- **Baud Rate Generator**—a programmable internal baud rate generator is provided. The output of the baud rate generator can be used by either the transmitter or receiver, or both (Divisor Latch Register pair). In asynchronous mode, the baud rate generator is programmed to provide a clock that is 16 times the data rate. The receiver uses this 16X clock directly. The transmitter divides this clock by 16 internally. If the baud rate generator is programmed to divide by one, the input clock (USARTCLK) is passed to the output of the baud rate generator unaffected. This allows USARTCLK to be used as second external clock input.

- **Clock Selection**—both the receiver and the transmitter can be clocked from either the output of the baud rate generator or directly from the RxCLK input (USART Control Register).

- **Break Generation**—a register selection is available to permit interruption of data transmission in the asynchronous mode with the break character (Line Control Register).

- **FIFO Thresholds**—each 4-byte transmit and receive FIFO has a selectable threshold value up to 4 bytes (USART Control Register). Upon reaching the threshold value, the USART may be programmed to interrupt the external processor or signify this by setting a status register bit.

- **Break Generation**—a register selection is available to permit interruption of data transmission in the asynchronous mode with the break character (Line Control Register).

- **FIFO Thesholds**—each 4-byte transmit and receive FIFO has a selectable threshold value up to 4 bytes (USART Control Register). Upon reaching the threshold value, the USART may be programmed to interrupt the external processor or signify this by setting a status register bit.

- **Special Character Recognition**—the user may select a set of one or more characters and define them as special characters. The USART can be programmed to interrupt the local processor when a special charcter is detected or to set a status register bit. Up to 128 characters can be selected as special. If 5-, 6-, or 7-bit character lengths are selected, any combination of characters may be selected as special. If an 8-bit character length is used, characters with bit patterns of 0-127 may be selected as special. Special characters are designated by setting bits in a 128-bit map via the Special Character Bit-Map Address Pointer Register, and the Special Character Bit-Map Command Register.

- **Interrupts**—any of the following interrupts may be selectively enabled or disabled:
  - Change in CTS
  - Change in DSR
  - Parity error
  - Receive FIFO threshold reached
  - Receive FIFO time-out
  - Transmit FIFO threshold reached
  - Transmit shift register empty
  - Break detect
  - Special character detected
  - Framing error
  - Buffer overrun

USART Register Map

Table 4-4. USART register map.

Offset (Hex)	Register Name	Size (Bytes)	Type
20	Receive FIFO Data Register (DLAB = 0)*	1	Read Only
	Transmit FIFO Data Register (DLAB = 0)	1	Write Only
	Baud Rate Divisor LSB Register (DLAB = 1)	1	Read/Write
21	Interrupt Enable Register (DLAB = 0)	1	Read/Write
	Baud Rate Divisor MSB Register (DLAB = 1)	1	Read/Write
22	Interrupt Identification Register	1	Read Only
23	Line Control Register	1	Read/Write
24	Modem Control Register	1	Read/Write
25	Line Status Register	1	Read Only
26	Modem Status Register	1	Read Only
27	Control Register	1	Read/Write
28	Status Register	1	Read/Write
29	Special Character Bit-Map ADDR Pointer Register	1	Read/Write
2A	Special Character Bit Map Command Register	1	Read/Write
2B-3E	Reserved	6	—

*Divisor Latch Access Bit (DLAB) in the Line Control Register.

USART Programmable Operations

The following section provides an introduction to programming the USART basic operations/functions, including:

- Baud rate generation
- Clocking options
- Special character recognition
- Modem handshake signals
- Receive FIFO operation

Baud Rate Generation

The baud rate generator divides the USARTCLK input frequency (user defined) by a programmable value. For asynchronous operation, the result of this division must be 16

times the data rate. For example, if the frequency of the USARTCLK input is 12.288 MHz, and the data rate is 19200 bps, the baud rate generator would be programmed to divide by 40. The output of the baud rate generator would be 307,200 Hz, or 16 times 19200 bps. In synchronous/transparent mode, the output of the baud rate generator is programmed to be equal to the data rate. In the example above, the 12.288 MHz input would need to be divided by 640 to produce the desired 19200 Hz clock.

Programming The Baud Rate Generator—The divisor is programmed into the baud rate generator by loading two eight-bit registers, the Baud Rate Divisor LSB and MSB Registers. These registers can be accessed only by first setting the divisor Latch Access Bit (DLAB) in the Line Control Register (Bit 7).

Divide By One Option—It is sometimes desirable to supply both the transmitter and the receiver clocks from separate external sources. The RxCLK input provides one of these clock inputs. The other is provided by programming the baud rate generator to divide by one. In this case, the input to the baud rate generator (USARTCLK) is fed directly to the output of the generator, providing a second clock input.

Clocking Options

The USART transmitter and receiver can be clocked from the RxCLK pin, the USARTCLK pin divided by the baud rate generator, or the non-divided output of the baud rate generator (USARTCLK pin direct). The selection of the clock source is made independently for the transmitter and the receiver. Two bits in the USART Control Register are used to select between the output of the baud rate generator and the RxCLK pin.

Special Character Recognition

As characters are received they are checked against a user programmed list of up to 128 "Special Characters." If a designated character is received, an interrupt is generated (maskable). Characters are designated as special by first loading the character into the Special Character Bit-Map Command Register (for eight-bit characters, the least significant seven bits are used). Once a character has been designated, it can be returned to non-designated status by loading the character into the Special Character Bit-Map Command Register. When a special character is detected, a flag is set in the FIFO. This flag travels with the character as the character moves through the FIFO and is used to identify the character as it is read from the FIFO. (Refer to the discussion of FIFO programming below.)

Modem Handshake Signals

The USART has four general purpose handshake lines. RTS/ and DTR/ are outputs while CTS/ and DSR/ are inputs. Those familiar with UART conventions will recognize that these are Data Terminal Equipment (DTE) designations as opposed to Data Communication Equipment (DCE) designations. This is because the 8250 UART, of which the IDPC USART is a functional super-set, uses these naming designations. In practice, the USART can be either a DTE or DCE since the handshake lines do not directly control the operation of the USART—the lines are used only as input and ouputs under software control. The RTS/ and DTR/ outputs are controlled by setting and clearing the corresponding bits in the Modem Control register. The CTS/ and DSR/ inputs are monitored via four bits in the Modem Status Register. The change in CTS/ and change in DSR/ bits indicate whether the respective input has changed state, in either direction, since the register was last read. A maskable interrupt is generated when either of the bits is set. The actual state of the CTS/ and DSR/ bits can be read via the CTS/ and DSR/ status bits in the Modem Status Register.

Receive FIFO Operation

Special Character and Parity Error Handling—The receive FIFO is 10 bits wide, with eight bits for the received character, and the remaining two bits containing special character and parity error Flags. Only the eight received data bits are directly accessible to the user. The two flag bits are indirectly accessed via bits in the USART Status Register. When a character with a special character or a parity error is detected, it is placed into the FIFO and one, or both, of the flags is set. At this time an interrupt is generated, if enabled. When the interrupt is detected, the user becomes aware that a parity error, for example, has been detected. What the user does not know is which character in the FIFO has the error. To identify the character, the user polls the character with the parity error available bit in the Modem Status Register, as follows: prior to reading a byte of data from the Receive FIFO Data Register, the character with parity error available Bit is polled. If it is not set, the Receive FIFO Data Register is read to remove an error-free character. Again the character with parity error available bit is polled. This cycle is repeated until the character with parity error available bit is set. The bit being set indicates that the character in the Receive FIFO Data Register contains the parity error. Note that the data available bit (Line Status Register) was not polled in this operation. This is possible since we know that there is still data in the receive FIFO until the character with the parity error is read.

Using The Data Available Bit—In normal operation, the data available bit is used only when the number of characters in the receive FIFO is not known, and a special character or parity error interrupt has not been detected. The number of characters available in the receive FIFO is known if a threshold reached interrupt is received. In cases where the data available bit is required, it is polled prior to each read from the Receive FIFO Data Register. When it is no longer polled active, the receive FIFO is empty.

Receive FIFO Time-out—Since, in asynchronous communication, there is no explicit indication of the end of a transmission, one or more of the last characters in a message can be received into the receive FIFO without the user being aware of their existence. This happens only when the level in the receive FIFO remains below the programmed threshold level when the last character is received. For this reason, a time-out interrupt is provided. This maskable interrupt is generated any time a character remains in the receive FIFO for more than 2048 receive clock cycles. Note that the receive clock is 16 times the data rate.

USART Operational Sequences

The operational sequences described below illustrate in detail the programming of the IDPC Universal Synchron-

ous/Asynchronous Receiver Transmitter (IDPC USART) hardware by an 80188 local processor in a typical application scenario.

An asterisk (*) beside various parameters indicates that these are options, chosen arbitrarily for the sake of example.

This scenario assumes:

A) Asynchronous operation with the following parameters:

   * 9600 baud, 7-bit character, even parity, one stop bit, USARTCLK = 12.288MHz.

B) IDPC USART serial interface is attached to a "dumb" ASCII terminal. Since no modem is involved, IDPC USART modem input signals are ignored. IDPC USART modem output signals DTR and RTS are permanently set active.

C) The IDPC USARTINT output pin is connected to one of the local 80188 INTX (INT0-INT3) maskable interrupt input pins to form the "USART interrupt." Both USART receive and transmit operations will be interrupt driven in this example scenario.

D) The local processor has initialized its 80188 interrupt controller hardware and interrupt vectors during reset, enabling the USART interrupt in the process.

E) Several interrupts that are useful for diagnostic testing are not enabled for regular operation in this scenario.

F) When a USART receiver interrupt occurs (e.g., receive FIFO threshold reached or parity error), the local processor USART interrupt service routine (USART ISVR) will unload the USART receive FIFO before another character can be received, e.g., within 520 microseconds at 19.2 Kbps.

   If this interrupt processing performance can be guaranteed, then USART ISVR processing time is significantly reduced, since the USART ISVR does not have to check the parity error and special character received bits in the UART Status Register as each character is read from the receive FIFO (unless the interrupt cause was parity error or special character received).

   If this interrupt processing performance cannot be guaranteed, then the USART receive character(s) operational sequences in this section could miss a parity error or special character received indication. This would occur if a character with a parity error or a special character is received after a USART receiver interrupt is generated but before the FIFO can be unloaded. These operational sequences would fail to check the parity error or special character status bits for the character received after the interrupt was generated. The solution to this problem would be to check the USART Status Register for parity error and special character status for every character that is ever unloaded from the Receive FIFO Data Register.

   Refer to the iAPX 86/88,186/188 User Manual Volume 1: Programmer's Reference for a description of 80188 interrupt controller operation.

The USART operational sequences for this scenario are listed below:

Operational Sequences

Initialization

Transmit Character(s)—Initiate USART Transmission

Transmit Character(s)—Transmit FIFO Threshold Reached Interrupt Service Routine

Receive Character(s)—Receive FIFO Threshold Reached

Receive Character(s)—Receive FIFO Time-out

Receive Character(s)—Special Character Received

Receive Character(s)—Parity Error

Receive Character(s)—Break Received

Receive Character(s)—Framing Error

Receive Character(s)—Overrun Error

These operational sequences are interdependent. For example, the USART initialization sequence must be executed before the USART transmit character(s) or receive character(s) sequences.

The two USART transmit operational sequences—Initiate USART transmission and transmit FIFO threshold reached interrupt service routine (ISVR)—are intended to work together. (The transmit FIFO threshold reached ISVR is a particular execution path through the more general USART ISVR.) The initiate USART transmission sequence would be executed when the USART transmitter is idle to begin transmission (prime the pump). Once transmission starts, the transmit FIFO threshold reached ISVR would continue placing characters in the transmit FIFO as long as there are characters to transmit (keep the pump going). At the point that there are no characters left to transmit, the transmit FIFO threshold reached ISVR would exit without writing any characters to the FIFO (no more to pump). When more characters become available for transmission, the initiate USART transmission sequence would be invoked, starting the USART transmit cycle again.

The seven USART receive operational sequences also work together. Each of the seven sequences would be executed as a different path through the USART interrupt service routine. It is likely that these sequences would share much of the same code, arriving at a common code section by different control paths. The receive FIFO threshold reached and receive FIFO time-out operational sequences would be the most frequently executed paths in most applications.

Initialization

1. Write the USART Interrupt Enable Register with the following contents:

Bit	Value	Function
0	1	Enable interrupt on Receive FIFO Threshold Reached
1	1	Enable interrupt on Transmit FIFO Threshold Reached
2	1	Enable Receive Line Status interrupt
3	0	Disable Modem Status interrupt
4	1	Enable interrupt on Receive FIFO Time-out
5	1	Enable interrupt on Special Character Received
6	0	Disable interrupt on Transmit Shift Register Empty
7	0	Not used

2. Write the USART Line Control Register with the following contents to prepare to load the USART divisor latches:

Bit	Value	Function
0-6	Don't care	N.A.
7	1	Access Divisor Latches

To determine the value X to load into the divisor latches for 9600 baud operation, the following equation is used:

X = 12,288,000 / (9600 x 16) = 80 decimal = 50 hex

12,288,000 is the local processor clock rate per second and 16 is a constant independent of the clock rate.

Write the derived value of 50 hex to the USART Divisor Latch Least Significant Byte Register and 0 hex to the USART Divisor Latch Most Significant Byte Register.

3. Write the USART Line Control Register with the following contents:

Bit	Value	Function
0	0	7-bit character
1	1	
2	0	One Stop Bit
3	1	Enable parity
4	1	Even parity
5	0	No Stick Parity
6	0	Do not send Break (during initialization)
7	0	Do not access Divisor Latches

4. Write the USART Modem Control Register with the following contents:

Bit	Value	Function
0	1	DTR active
1	1	RTS active
2	0	Not used
3	0	Not used
4	0	No loop back
5	0	Not used
6	0	Not used
7	0	Not used

5. Write the USART Control Register with the following contents:

Bit	Value	Function
0	1	Use internal Baud Rate Generator for Receive Clock
1	1	Use internal Baud Rate Generator for Transmit Clock
2	0	Asynchronous operation
3	1	Receive FIFO Threshold = 3
4	1	
5	1	Transmit FIFO Threshold = 1
6	0	
7	0	Do not reset USART

FIFO threshold settings are arbitrary for this scenario.

6. Initialize the 128-bit USART special character bit map.

Write the character into the Special Character Bit-Map Address Pointer Register, using only the seven least significant bits, then write a "1" to bit 0 of the Special Character Bit-Map Command Register. Repeat for all characters to be designated as special.

7. Enable the USART receiver by writing the USART Status Register with the following contents:

Bit	Value	Function
0-6	Don't care	
7	1	Enable Receiver

Transmit Character(s)— Initiate USART Transmission

This operational sequence is executed by the local processor main software loop when a request (transmit request) for USART transmission of an arbitrary number of characters is received from higher level software.

1. When the request to transmit characters is received by the main loop, it checks a temporary temp queue of characters (in RAM) waiting to be transmitted. If there are any characters in this (temp) queue, then the last time the transmit FIFO was filled, there were additional characters available for transmission that could not fit into the full FIFO. This implies that a transmit FIFO threshold reached interrupt is going to occur eventually, at which time characters from the temp queue will be loaded into the FIFO. In this situation (characters already in temp queue when a request for transmission of additional characters is received), the main loop places the additional characters at the end of the temp queue of characters awaiting transmission and exits this operational sequence, moving on to other main loop duties.

   If, on the other hand, there are no characters in the temp queue when the transmit request is received, proceed with step 2.

2. The main loop polls the transmit buffer available bit (bit 4) in the USART Status Register. If this bit is zero, the FIFO is full so the main loop places the characters of this transmit request in the temp queue (of characters awaiting transmission) and exits this operational sequence. Otherwise, continue with step 3.

3. The main loop writes the next character to be transmitted to the USART Transmit FIFO Data Register.

4. If there are additional characters to transmit, go to step 2. Otherwise, exit this operational sequence.

Transmit Character(s)—Transmit Threshold Reached Interrupt Service Routine

This operational sequence is executed by the local processor USART interrupt service routine when a USART transmit threshold reached interrupt occurs.

1. When the number of characters in the USART transmit FIFO decrements to equal the transmit FIFO threshold programmed in the USART Control Register, the transmit threshold reached bit is set in the USART Line Status Register. Since this interrupt was enabled during USART initialization, the 80188 is interrupted and vectors to the USART Interrupt Service Routine (USART ISVR).

2. The USART ISVR reads the USART Interrupt Identification Register to determine the cause of the interrupt. The interrupt IDcode (bits 1-3) indicates that the highest priority USART interrupt pending is transmit FIFO threshold reached.

3. If there are no characters awaiting transmission on the USART (i.e., no characters in the temp queue), the USART ISVR goes to step 7. Otherwise the ISVR continues with step 4.

4. The main loop polls the transmit buffer available bit (bit 4) in the USART Status Register. If this bit is zero, the FIFO is full so the USART ISVR goes to step 7. Otherwise, continue with step 5.

5. The USART ISVR writes the next character from the temp queue of characters awaiting transmission to the USART Transmit FIFO Data Register.

6. If there are additional characters in the temp queue, go to step 4. Otherwise, continue with step 7.

7. The USART ISVR writes a non-specific end-of-interrupt command (hex 8000) to the 80188 interrupt controller EOI Register and executes an 80188 IRET instruction to exit.

Receive Character(s)—Receive FIFO Threshold Reached

1. When the USART receive logic detects that the USART receive FIFO threshold has been reached (i.e., the number of characters in the FIFO equals or exceeds the threshold programmed in the USART Control Register), the receive FIFO threshold reached bit is set in the USART Status Register. Since this interrupt was enabled during USART initialization, the 80188 is interrupted and vectors to the USART interrupt service routine.

2. The USART ISVR reads the USART Interrupt Identification Register to determine the cause of the interrupt. The interrupt IDcode (bits 1-3) indicates that the highest priority USART interrupt pending is receive FIFO threshold reached.

3. The USART ISVR reads the USART Line Status Register. If the receive data available bit (bit 0) is set, continue with step 4. Otherwise go to step 6.

4. The USART ISVR reads a received character from the USART receive FIFO data register and processes the character as appropriate for the application. For a terminal adaptor, this usually means placing the character in a RAM buffer that will eventually be transmitted over the network via the Data Link Controller (DLC). This character does not require special character processing or error processing (e.g., parity error) since these exception conditions generate a different interrupt IDcode from receive FIFO threshold reached.

5. Go to step 3.

6. The USART ISVR writes a non-specific end-of-interrupt command (hex 8000) to the 80188 interrupt controller EOI Register and executes an 80188 IRET instruction to exit.

Receive Character(s)—Receive FIFO Time-out

1. When the USART receive logic detects that a USART receive FIFO time-out has occurred (i.e., about ten character times have elapsed with at least one character in the receive FIFO), the receive FIFO time-out bit is set in the USART Status Register. Since this interrupt was enabled during USART Initialization, the 80188 is interrupted and vectors to the USART interrupt service routine.

2. The USART ISVR reads the USART Interrupt Identification Register to determine the cause of the interrupt. The interrupt IDcode (bits 1-3) indicates that the highest priority USART interrupt pending is receive FIFO time-out.

3. Execute steps 3 through 6 of the USART Receive Character(s) Operational Sequence 1—receive FIFO threshold reached (above) to finish processing this operational sequence.

Receive Character(s)—Special Character Received

1. When the USART receive logic detects that a special character has been received (i.e., character received with corresponding bit set in the USART special character recognition bit map), the special character received bit is set in the USART Line Status Register. Since this interrupt was enabled during USART initialization, the 80188 is interrupted and vectors to the USART interrupt service routine.

2. The USART ISVR reads the USART Interrupt Identification Register to determine the cause of the interrupt. The interrupt IDcode (bits 1-3) indicates that the highest priority USART interrupt pending is receive line status.

3. The USART ISVR reads the USART Line Status Register. The USART ISVR determines that the special character received bit is set in the Line Status Register.

4. The USART ISVR checks the receive data available bit (bit 0) in the USART Line Status Register. If this bit is

set, continue with step 5. Otherwise go to step 9.

5. The USART ISVR reads the USART Status Register. If the special character available bit (bit 2) is set, go to step 8. Otherwise continue with step 6.

6. The USART ISVR reads a received non-special character from the USART Receive FIFO Data Register and processes the character as appropriate for the application. For a terminal adaptor, this usually means placing the character in a RAM buffer that will eventually be transmitted on the B-Channel via the IDPC DLC. This character does not require special character processing since the special character available bit was not set in the USART Status Register.

7. Go to step 4.

8. The USART ISVR reads the received special character from the USART Receive FIFO Data Register. The ISVR performs the application-dependent "special" processing for the character. For example, in a terminal adaptor application, receipt of the special character line feed may cause the terminal adaptor software to packetize a buffer of characters previously received from the USART and transmit the packet(s) using the DLC.

9. The USART ISVR writes a non-specific end-of-interrupt command (hex 8000) to the 80188 interrupt controller EOI Register and executes an 80188 IRET instruction to exit.

Receive Character(s)—Parity Error

1. When the USART receive logic detects that a character has been received with a parity error, the parity error bit is set in the USART Line Status Register. Since this interrupt was enabled during USART initialization, the 80188 is interrupted and vectors to the USART interrupt service routine.

2. The USART ISVR reads the USART Interrupt Identification Register to determine the cause of the interrupt. The interrupt IDcode (bits 1-3) indicates that the highest priority USART interrupt pending is receive line status.

3. The USART ISVR reads the USART Line Status Register. The USART ISVR determines that the parity error bit is set in the Line Status Register.

4. The USART ISVR checks the receive data available bit (bit 0) in the USART Line Status Register. If this bit is set, continue with step 5. Otherwise go to step 10.

5. The USART ISVR reads the USART Status Register. If the character with parity error available bit (bit 1) is set, go to step 8. Otherwise continue with step 6.

6. The USART ISVR reads a received character with no parity error from the USART Receive FIFO Data Register and processes the character as appropriate for the application. For a terminal adaptor, this usually means placing the character in a RAM buffer that will eventually be transmitted via the DLC.

7. Go to step 4.

8. The USART ISVR reads the received character with parity error from the USART Receive FIFO Data Register. The ISVR performs the application-dependent parity error processing for the character. For example, receipt of a character with parity error may result in the character being thrown away.

9. (Optional) The USART ISVR increments a parity error counter in RAM. If this counter reaches a certain threshold, the IDPC software may notify a higher level of software that parity errors are occurring too frequently.

10. The USART ISVR writes a non-specific end-of-interrupt command (hex 8000) to the 80188 interrupt controller EOI Register and executes an 80188 IRET instruction to exit.

Receive Character(s)—Break Received

1. When the USART receive logic detects that a receive break condition has occurred, the Break Interrupt bit is set in the USART Line Status Register. Since this interrupt was enabled during USART Initialization, the 80188 is interrupted and vectors to the USART interrupt service routine.

2. The USART ISVR reads the USART Interrupt Identification Register to determine the cause of the interrupt. The interrupt IDcode (bits 1-3) indicates that the highest priority USART interrupt pending is receive line status.

3. The USART ISVR reads the USART Line Status Register. The USART ISVR determines that the break interrupt bit is set in the Line Status Register.

4. The USART ISVR performs application-dependent break processing. For example, in a terminal adaptor application, receipt of break may cause the terminal adaptor software to transmit a Layer 3 interrupt-type packet over the network via the DLC. Depending on the application, the USART ISVR may also unload the receive FIFO, packetize and transmit all received USART characters over the network before sending the interrupt-type packet.

5. The USART ISVR writes a non-specific end-of-interrupt command (hex 8000) to the 80188 interrupt controller EOI Register and executes an 80188 IRET instruction to exit.

Receive Character(s)—Framing Error

1. When the USART receive logic detects that a framing error (i.e., no stop bit) has occurred, the framing error bit is set in the USART Line Status Register. Since this interrupt was enabled during USART initialization, the 80188 is interrupted and vectors to the USART interrupt service routine.

2. The USART ISVR reads the USART Interrupt Identification Register to determine the cause of the interrupt. The interrupt IDcode (bits 1-3) indicates that the highest priority USART interrupt pending is receive line status.

3. The USART ISVR reads the USART Line Status Register. The USART ISVR determines that the framing error bit is set in the Line Status Register.

4. Since the IDPC USART (unlike the 8250 UART) discards a character with a framing error, no further USART processing is necessary, although some implementations may take this opportunity to unload the receive FIFO.

5. (Optional) The USART ISVR increments a framing error counter in RAM. If this counter reaches a certain threshold, the IDPC software may notify a higher level of software that framing errors are occurring too frequently.

6. The USART ISVR writes a non-specific end-of-interrupt command (hex 8000) to the 80188 interrupt controller EOI Register and executes an 80188 IRET instruction to exit.

Receive Character(s)—Overrun Error

1. When the USART receive logic detects that a character has been received with the receive FIFO full, the overrun error bit is set in the USART Line Status Register. Since this interrupt was enabled during USART initialization, the 80188 is interrupted and vectors to the USART interrupt service routine.

2. The USART ISVR reads the USART Interrupt Identification Register to determine the cause of the interrupt. The interrupt IDcode (bits 1-3) indicates that the highest priority USART interrupt pending is receive line status.

3. The USART ISVR reads the USART Line Status Register. The USART ISVR determines that the overrun error bit is set in the Line Status Register. Since no exception (e.g., special character or parity error) has occurred up until the time of overrun (exception status would still be pending in the Line Status Register, preventing the overrun error bit from being set), all characters in the receive FIFO are error free. Therefore, the receive FIFO is unloaded in steps 4-6 below without checking for exception status.

4. The USART ISVR reads the USART Line Status Register. If the receive Data available bit (bit 0) is set, continue with step 5. Otherwise go to step 7.

5. The USART ISVR reads a received character from the USART Receive FIFO Data Register and processes the character as appropriate for the application. For a terminal adaptor, this usually means placing the character in a RAM buffer that will eventually be transmitted via the IDPC DLC.

6. Go to step 4.

7. (Optional) The USART ISVR increments an overrun error counter in RAM. If this counter reaches a certain threshold, the IDPC software may notify a higher level of software that overrun errors are occurring too frequently.

8. The USART ISVR writes a non-specific end-of-interrupt command (hex 8000) to the 80188 interrupt controller EOI Register and executes an 80188 IRET instruction to exit.

DPMC/INTERPROCESSOR INTERRUPT PROGRAMMING

For a multi-processor application, a common message area in RAM (mailbox) is often used for inter-processor communications. Since this mailbox must be accessed by both processors, dual-port RAM is normally used. The IDPC provides a hardware mechanism, the DPMC, that allows standard single port static RAM to be used as dual-port RAM. When mailbox structures are used, a means is also required for each processor to indicate to its counterpart that there is a message to be read from the mailbox. The DPMC contains hardware for implementing such an interprocessor interrupt structure. The hardware for creating the dual-port RAM function does not have any associated programmable options. The interprocessor interrupt mechanism is programmable. This programming is described below.

DPMC/Interprocessor Interrupt Programmable Features

The DPMC's interprocessor interrupt mechanism contains one programmable register, the Semaphore Register, which is used by the local processor to generate interrupt requests to the host processor, and to clear interrupt requests from the host processor. Additionally, the local processor can poll the Semaphore Register to see whether the host has responded to an interrupt request from the local processor.

DPMC Register Map

Table 4-5 shows the DPMC register map.

Table 4-5
DPMC Register Map

Offset (Hex)	Register Name	Size (Bytes)	Type
3F	Semaphore Register	2	Read/Write

DPMC/Interprocessor Interrupt Programmable Operations

When Bit 0 of the Semaphore Register is set by the local processor, an interrupt request is generated to the host processor (HINTOUT) indicating it has a message in the mailbox area of RAM. Bit 0 of the Semaphore Register is cleared when the host processor acknowledges the interrupt (HINTACK line is pulsed).

When the local processor has a message from the host processor to read, the host processor generates Host-Interrupt-In (HINTIN), which sets bit 1 in the Semaphore Register and activates the LINTOUT pin. LINTOUT is deactivated when the local processor clears bit 1 of the Semaphore Register.

DPMC/Interprocessor Interrupt Programmable Sequences

The following IDPC interprocessor interrupt operational sequences are described below:

Operational Sequences

Host CPU Interrupts Local 80188

Local 80188 Interrupts Host CPU

These operational sequences are interdependent. The following assumptions apply to these sequences:

A) The host CPU can perform a write (memory-mapped or I/O), that is decoded by hardware external to the IDPC such that a strobe is generated on the IDPC HINTIN pin.

B) The IDPC HINTOUT output pin is connected to a host CPU chip interrupt input pin or to a host system interrupt controller.

C) The host CPU can perform a write (memory-mapped or I/O) that is decoded by hardware external to the IDPC such that a strobe is generated on the IDPC HINTACK pin.

D) The IDPC LINTOUT output pin is connected to one of the local 80188 INTX (INT0-INT3) maskable interrupt input pins.

E) Both host CPU and local processor have initialized their respective interrupt controller hardware and interrupt vectors during reset, enabling Interprocessor Interrupts in the process.

Refer to the iAPX 86/88, 186/188 User Manual Volume 1: Programmer's Reference for a description of 80188 interrupt controller operation.

Host CPU Interrupts Local 80188

HOST:

1. The host writes a command and associated parameters to an interprocessor mailbox in RAM on the IDPC external bus. The host also clears an interrupt ack mailbox in RAM on the IDPC external bus that the local processor will write in step 4 below to ackowledge being interrupted.

2. In order to notify the local 80188 processor of the command, the host CPU performs a write that is decoded to strobe the IDPC HINTIN pin. This causes the IDPC to set bit 1 in the IDPC Semaphore Register and activates the IDPC LINTOUT pin.

LOCAL 80188:

3. The 80188 is interrupted since the local 80188 INTX used for incoming interprocessor interrupt was enabled during initialization of the 80188 interrupt controller (see assumptions above). The 80188 vectors to its IDPC Interprocessor interrupt service routine (local interprocessor ISVR).

4. In order to deactivate the IDPC LINTOUT pin, the local interprocessor ISVR clears bit 1 in the IDPC Semaphore Register. The Local Interprocessor ISVR then writes an ack code to the IDPC external bus RAM interrupt ack mailbox. The local 80188 then proceeds to process the command from the host.

HOST:

5. The host CPU polls this ack mailbox until it sees the ack code.

Local 80188 Interrupts Host CPU

LOCAL 80188:

1. The local 80188 writes a command and associated parameters to an interprocessor mailbox in RAM on the IDPC external bus.

2. In order to notify the host CPU of the command, the local 80188 sets bit 0 in the IDPC Semaphore Register. This causes the IDPC to strobe its HINTOUT output pin.

HOST:

3. The host CPU is interrupted since the host interrupt input used for incoming interprocessor interrupt was enabled during initialization of the host interrupt controller (see assumptions above). The host CPU vectors to its IDPC Interprocessor interrupt service routine (host interprocessor ISVR).

4. Through either hardware or software means, the host activates the IDPC HINTACK input pin to acknowledge the interrupt. This causes the IDPC to clear bit 0 in the IDPC Semaphore Register. The host then proceeds to process the command from the local 80188.

LOCAL 80188:

5. To detect that the host CPU has acknowledged the interrupt, the local 80188 polls bit 0 of the IDPC Semaphore Register until it is zero.

Chapter 5

Am79LLD401 LOW-LEVEL DEVICE DRIVER

DISTINCTIVE CHARACTERISTICS

This document describes the user interface to the Low-Level Device Driver (LLD) for the Am79C401 Integrated Data Protocol Controller (IDPC) Data Link Controller (DLC). The IDPC DLC LLD has been implemented using the 'C' programming language to maximize portability and readability with a minimum effect upon performance.

Table 5-1. Summary of IDPC DLC LLD Features

- Written primarily in ANSI 'C'.
- Less than 5% written in Microsoft 8088 Macro Assembler.
- Minimum operating system and processor dependencies.
- Uses 80188/80186 DMA.
- Supports optional logging to a file.
- Interrupt-driven mailbox interface to/from Layer 2 (L2) and Management Entity (ME) routines.
- Compatible with AmLink™ (LAPD/LAPB) Layer 2 software.

GENERAL DESCRIPTION

This document describes the user interface to the Low-Level Device Driver (LLD) for the Am79C401 Integrated Data Protocol Controller (IDPC) Data Link Controller (DLC).

PURPOSE

The IDPC DLC LLD is intended to be used as a general purpose example of IDPC DLC programming. The IDPC DLC LLD source code contains examples illustrating how to use and access the many features of the Am79C401 IDPC DLC hardware.

The IDPC DLC LLD can be used with any Bit-Oriented Protocol (BOP) including AmLink™, LAPD/LAPB implementation from Advanced Micro Devices. In Integrated Services Digital Network (ISDN) applications, the IDPC DLC LLD is used to support the packet protocol for the B-Channel. (The Am79C30A Digital Subscriber Controller (DSC) LLD provides the same services for the D-Channel.) The interfaces provided by the DSC and IDPC DLC LLDs use the same primitives so that both D-Channel and B-Channel can use the same Layer 2 software. The DSC and IDPC DLC LLDs provide a hardware independent interface to upper layer protocols such as LAPD.

SYSTEM REQUIREMENTS

The IDPC DLC LLD places relatively few requirements on the target system. The IDPC DLC LLD requires that the Operating System (OS) provide a method for requesting and returning memory buffers. No other OS services are required. The system requirements are summarized below:

- 4 kBytes of RAM/ROM for object code.
- 64 bytes of shared RAM for the configuration table.
- 256 bytes of shared RAM for mailboxes and initialization parameter blocks.
- One or more shared memory buffers of MAXPACKSZ (Maximum Packet Size) for data transfers.
- 128 bytes of stack RAM.
- Memory allocation service from the OS.
- Event interrupt generation routines. 80188/80186 DMA hardware (both channels).

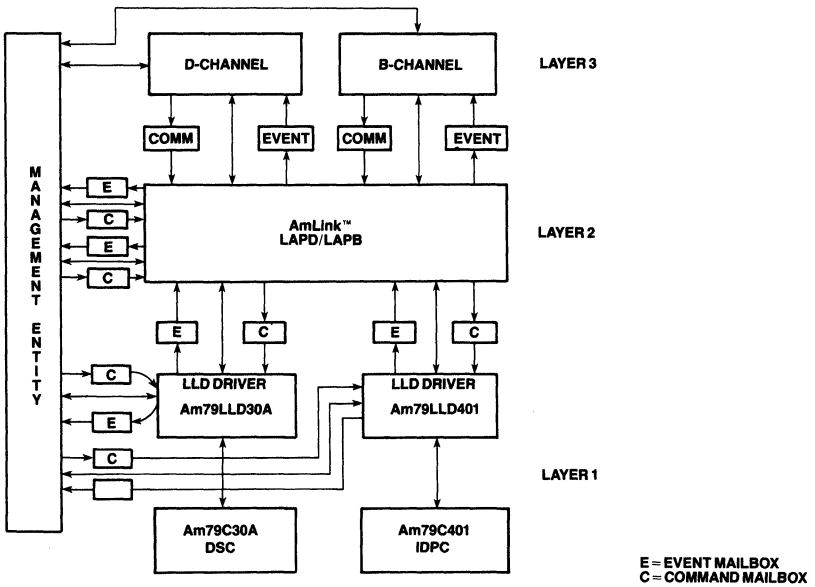


Figure 5-1. Inter-Layer Mailbox Interface

ARCHITECTURE

Communications with the IDPC DLC LLD by Layer 2 (L2) or Management Entity (ME) routines are performed via mailboxes and interrupts. As shown in Figure 5-1, the IDPC DLC LLD includes four mailboxes. One pair of mailboxes is required for the L2 interface and a second pair is required for the ME interface. Each interface pair includes a command and an event mailbox. The mailbox structure is described in a later section.

The ME is a set of routines which are provided by the user to perform connection and layer management functions. These are generally system-dependent functions that are used to tie the various D-Channel and B-Channel protocol layers together in the user's system drivers.

Command mailboxes are used to allow commands to be sent from the L2 or ME routines to the IDPC DLC LLD. Commands are loaded into the mailbox by the calling routine. An interrupt is then generated to inform the IDPC DLC LLD that a command is available for processing. The IDPC DLC LLD acknowledges the receipt of the command through the same mailbox.

Event mailboxes are used to send status information from the IDPC DLC LLD to the L2 or ME routines. The IDPC DLC LLD loads the event information into the proper mailbox then generates an interrupt which alerts the L2 or ME layer that an event has occurred and is available. The receiving routine, L2 or ME, acknowledges the event in the same mailbox.

The L2-DLC LLD interface (mailbox pair) is used primarily for data transfer primitives. The ME-DLC LLD interface is used to pass control and status information.

TARGET ENVIRONMENT

The IDPC DLC LLD can be used in either a single processor environment or a multi-processor system. In the single processor system, inter-layer communications are signaled using software interrupts. In a multi-processing system, hardware interrupts are used. In either case, the operation of the IDPC DLC LLD is the same. The interrupt handler used to process the mailbox message (command or event) is the same for both software- and hardware-based systems.

The IDPC LLD is initially implemented on an 80188/86 processor, using the 'C' programming language to enhance portability. The two primary processor dependencies are:

- Word (16-bits) and long word (32-bits) byte order
- Address segmentation

DEVELOPMENT ENVIRONMENT

The IDPC DLC LLD is implemented using Microsoft 'C' Compiler Version 4.0 and the Microsoft Macro Assembler Version 5.0. Note that the byte order for all addresses specified in this document are in Microsoft 'C' "FAR" format:

Lowest Address Byte:	Low-Order Byte	OFFSET
	High-Order Byte	OFFSET
	Low-Order Byte	SEGMENT
Highest Address Byte:	High-Order Byte	SEGMENT

FUNCTIONAL DESCRIPTION

This section shows the IDPC DLC LLD interfaces and services accessible to the user and describes how to use the IDPC DLC LLD.

POR CONFIGURATION AND INITIALIZATION

Prior to using the IDPC DLC LLD, several initialization tasks must be performed. These are generally executed during the system Power-On Reset (POR) initialization sequence. In order to perform these tasks, the user must know the following information about the IDPC DLC LLD:

- Address of IDPC DLC LLD code.
- Offset of POR initialization routine.
- Offset of IDPC DLC hardware interrupt handler.
- Offset of 80188 DMA hardware interrupt handler.
- Offset of Layer 2 command input handler.
- Offset of Management Entity command input handler.
- Structure of RAM Interface Block (RIB).

The IDPC DLC LLD code base address depends upon the user's system design. It may be located in firmware or in RAM. The IDPC DLC LLD is position independent.

The IDP POR initialization routine, referred to as illdinit() is located at offset TBD from the base address of the IDPC DLC LLD code. This routine must be called to initialize the IDPC DLC LLD.

The IDPC DLC hardware interrupt handler is located at offset TBD from the IDPC DLC LLD code base address. This address must be installed in the processor vector table prior to calling the IDPC POR initialization routine.

The 80188 DMA hardware interrupt handler is located at offset TBD from the IDPC DLC LLD code base address. This address of this routine must be installed in the processor vector table prior to calling the IDPC POR initialization routine.

The Layer 2 command input handler is located at offset TBD from the IDPC DLC LLD code base address. This address of this routine must be installed in the processor vector table prior to calling the IDPC POR initialization routine.

The Management Entity input command handler is located at offset TBD from the IDPC DLC LLD code base address. This address of this routine must be installed in the processor vector table prior to calling the IDPC POR initialization routine.

The RAM Interface Block (RIB) is the structure through which the IDPC DLC LLD is accessed by either Layer 2 or the Management Entity. The IDPC POR initialization routine installs default values into the RIB when it is called.

Using the above information, the user must perform or provide the following functions:

- Install the interrupt vectors for the IDPC DLC LLD.
- Provide routines to generate L2 and ME event interrupts.
- Build or provide the IDPC DLC LLD configuration table.
- Execute the illdinit( ) routine.

The user must install interrupt vectors for the IDPC DLC hardware interrupt handler, the 80188/86 DMA hardware interrupt handler, L2 & ME command input handlers, and L2 and ME event handlers. The L2/ME event handlers reside in the L2 (AmLink or other user-written Layer 2) code and the user-written ME code.

The user is required to provide routines which will be called by the DLC LLD to generate the interrupts (software or hardware) for L2 or ME events. Each routine should be in the form of a subroutine which passes no parameters. This allows the IDPC DLC LLD to be independent of the type of interrupt used to generate the event interrupt.

The user is also required to service requests for memory allocation by the IDPC DLC LLD. The IDPC DLC LLD requests memory of size specified by the Maximum Packet Size field (in the DLC Initialization Parameter Block (IPB)) to support data reception. When a packet is received without an error, the IDPC DLC LLD passes the buffer to Layer 2; if a packet is received with error, the LLD reuses the buffer for the next packet reception.

The IDPC DLC LLD accesses the configuration table (see Table 5-2) via a pointer to the configuration table located at the known address TBD. The configuration table itself can be located in RAM or in firmware.

The 'illdinit()' routine uses information from the 64 byte user-supplied configuration table (see Table 5-2) to initialize the IDPC hardware, install default values into the IDPC DLC Initialization Parameter Block, and initialize the IDPC DLC LLD Private RAM area.

After the 'illdinit()' routine is executed, the IDPC DLC LLD is in an idle state. Before transmitting or receiving any packets, the user must execute the DLC Init and DLC Control ME Command primitives.

Table 5-2. IDPC DLC LLD Configuration Table Format

Offset	Description	Size (bytes)
00	Addr of IDPC DLC LLD Private RAM	4
04	Addr of DLC LLD RAM Interface Block	4
08	Addr of L2 Event Generator	4
0C	Addr of ME Event Generator	4
10	Addr of IDPC DLC Hardware Registers	4
14	Addr of 80188/86 DMA Hardware Registers	4
18	Reserved	40

Address of the IDPC DLC Private Data RAM

This RAM is 64 bytes in length. This area is not required to reside in shared memory.

Address of IDPC DLC LLD RAM Interface Block

This field contains the address of the shared RAM where the mailboxes and DLC Initialization Parameter Block are located. This area must be at least 256 bytes in length. The structure of the RAM Interface Block (RIB) is described in Table 5-3.

Address of the LAYER 2 Interrupt Generator Routine

This field contains a pointer to a user-supplied routine which is called by the IDPC DLC LLD to generate an L2 event interrupt. No parameters are passed; the routine should return via a return from subroutine instruction. It is required that this routine return with all processor registers in the same state as when the routine was called.

Address of the MANAGEMENT ENTITY Interrupt Generator Routine

This field contains a pointer to a user-supplied routine which is called by the IDPC DLC LLD to generate an ME event interrupt. No parameters are passed; the routine should return via a return from subroutine instruction. It is required that this routine return with all processor registers in the same state as when the routine was called.

Address of IDPC DLC Device Hardware

This field contains the base address of the Am79C401 IDPC DLC hardware registers . The IDPC DLC LLD uses this pointer to access the Am79C401 device.

Address of 80188/86 DMA Device Hardware

This field contains the base address of the 80188/86 DMA hardware registers . The IDPC DLC LLD uses this pointer to access the DMA device.

Table 5-3. RAM Interface Block Structure

Offset	Description	Size (Bytes)
00	L2-DLC LLD Command Mailbox	18
12	DLC LLD -L2 Event Mailbox	18
24	ME-DLC LLD Command Mailbox	18
36	DLC LLD -ME Event Mailbox	18
48	Reserved	8
50	DLC Initialization Parameter Block	14
5E	Reserved	162

LAYER 2 to LLD Command Mailbox

This block of RAM is used as the L2-DLC LLD mailbox to pass commands from the Layer 2 protocol to the IDPC DLC LLD. This mailbox is primarily used for B-Channel data transmissions. The IDPC DLC LLD L2 command input handler services the commands passed in this mailbox.

LLD to LAYER 2 Event Mailbox

This block of RAM is used as the DLC LLD L2 event mailbox to pass event information from the IDPC DLC LLD to Layer 2. Mailbox structures are described in Table 5-5. This mailbox is used primarily for receiving B-Channel data.

Management Entity to LLD Command Mailbox

This block of RAM is used as the ME-DLC LLD mailbox to pass commands from the Management Entity to the IDPC DLC LLD. Commands are passed for IDPC DLC LLD setup and initialization. The IDPC DLC LLD ME command input handler services the commands passed in this mailbox.

LLD to Management Entity Event Mailbox

This block of RAM is used as the DLC LLD-ME mailbox which is used to pass event information from the IDPC DLC LLD to the management entity. This mailbox is used primarily to pass IDPC DLC LLD status information back to the management routines.

DLC Initialization Parameter Block

The data in this block (see Table 5-4) provide control information for the IDPC DLC module. Default values are loaded and installed by the 'illdinit()' routine. The user may modify the values in the IPB; however, these are not installed until the user executes the DLC Init Command.

The L2 Address Length, L2 Address Select and C/R Address Bit Ignore Enable fields are also used by the LLD during execution of the Update Address Recognition Command.

Table 5-4. DLC Initialization Parameter Block Structure

Offset	Description	Size (Bytes)
00	Maximum Packet Size	2
02	L2 Address Length	1
03	L2 Address Select	1
04	CRC Check Enable	1
05	CRC Pass-Through Enable	1
06	CRC Generator Enable	1
07	Mark or Flag Idle Select	1
08	C/R Address Bit Ignore Enable	1
09	B-Channel Select	1
0A	Invert Enable	1
0B	Minimum Packet Size	1
0C	Transmit FIFO Threshold	1
0D	Receive FIFO Threshold	1

MAXIMUM PACKET SIZE - Maximum length packet, (including CRC bytes, if any), that is legal for the Layer 2+ protocol in use.

LAYER 2 ADDRESS LENGTH - Number of bytes in the packet address field for the Layer 2+ protocol in use.

SINGLE ADDRESS BYTE SELECT - Selects which packet address byte to compare during address recognition when the Layer 2 Address Length is equal to one. Set this parameter to 1 for first packet address byte select, 2 for second byte select.

CRC CHECK ENABLE - Set to 1 to enable CRC checking during packet reception, 0 for CRC Check Disable.

CRC PASS-THROUGH ENABLE - Set to 1 for pass CRC bytes to Layer 2+ as last two bytes of each received packet. Set to 0 to disable passing any CRC bytes.

CRC GENERATOR ENABLE - Set to 1 to enable CRC Generation during packet transmission, 0 for CRC Generate Disable.

MARK IDLE/FLAG IDLE SELECT - Set to 1 to generate mark idle pattern (all 1 bits) when not transmitting a packet, 0 to generate flag idle pattern.

IGNORE C/R ADDRESS BIT ENABLE - Set to 1 to ignore the C/R bit in the Layer 2 packet address during address recognition. Set to 0 to also compare the C/R bit during address recognition.

B-CHANNEL SELECT - Set to a value from 0 to 30 (decimal) to select multiplexed B-Channel 0 through 30 respectively. Set to 31 (decimal) to select non-multiplexed operation (e.g., SNA).

INVERT ENABLE - Set to 1 to enable inversion of the transmitted and received serial bit streams. Set to 0 to disable inversion.

MINIMUM PACKET SIZE - Minimum length packet (including CRC bytes, if any) that is legal for the Layer 2+ protocol in use.

TRANSMIT FIFO THRESHOLD - Set to a value from 0 to 15 (decimal) to set the fullness threshold (0 to 15 bytes) at which the DLC Transmitter requests service from the LLD software or DMA for additional bytes to be loaded into the transmit FIFO.

RECEIVE FIFO THRESHOLD - Set to a value 0, 1, 2 . . . 15 (decimal) to set the fullness threshold (32, 2, 4 . . . 30 bytes) at which the DLC Receiver requests service from the LLD software or DMA for bytes to be unloaded from the receive FIFO.

MAILBOX INTERFACES

As described earlier, the primary interface to the IDPC DLC LLD services is implemented using mailboxes. This section describes the procedure for using the mailboxes which consist of an 18-byte structure containing the format shown in Table 5-5.

A mailbox is used to transfer commands and event information between the IDPC DLC LLD and either the L2 protocols or the ME routines. These routines may be in separate tasks or processes when using a multi-tasking operating system. This means that the mailboxes must be accessible to both the IDPC DLC LLD and L2/ME. This is generally no problem in a single processor implementation; however, in a multi-processing system or a system with memory-management, the mailboxes must be placed in shared-memory.

Table 5-5. Mailbox Structure

Offset	Component	Size (Bytes)
0	Command/Event Code	1
1	Receipt Code	1
2	Parameters	16

Command/Event Code

The first byte in a mailbox is the Command/Event Code. This byte determines what command is to be performed or what event has occurred. Each IDPC DLC LLD primitive is assigned a unique Command/Event Code. Table 5-7 lists the IDPC DLC LLD command codes and Table 5-8 provides a summary of the IDPC DLC LLD event codes.

Receipt Code

The second byte in a mailbox is the receipt code. This byte indicates to the routine issuing the command that command has been received and validated.

Upon issuing a command/event, the routine places the value 0xFF into the Receipt Code. The issuing routine then monitors the Receipt Code to determine if the command/event has been received. A value of '00' indicates that the command/event has been received or, for some commands, executed. Any other value indicates an error condition.

Table 5-6. Valid Mailbox Receipt Codes

Code	Description
00	Command/event received or complete.
01	Illegal command/event.
02	Illegal parameter(s).
03-FE	Reserved.
FF	Command/event not received or complete.

Parameters

The last 16 bytes of a mailbox are reserved for parameters. The contents of these bytes are dependent upon the actual command or event issued.

COMMAND SEQUENCES

Commands are passed from either the Management Entity (ME) or the Layer 2 (L2) protocol to the IDPC DLC LLD. Figure 5-2 shows a typical command sequence. Note that the Receipt Code returned by the IDPC DLC LLD may indicate that the command has simply been received or that execution is complete. This depends upon the particular command issued.

Table 5-7 lists the valid command codes to which the IDPC DLC LLD will respond. Each command is described in detail in a later section.

Figure 5-2. Typical Command Sequence

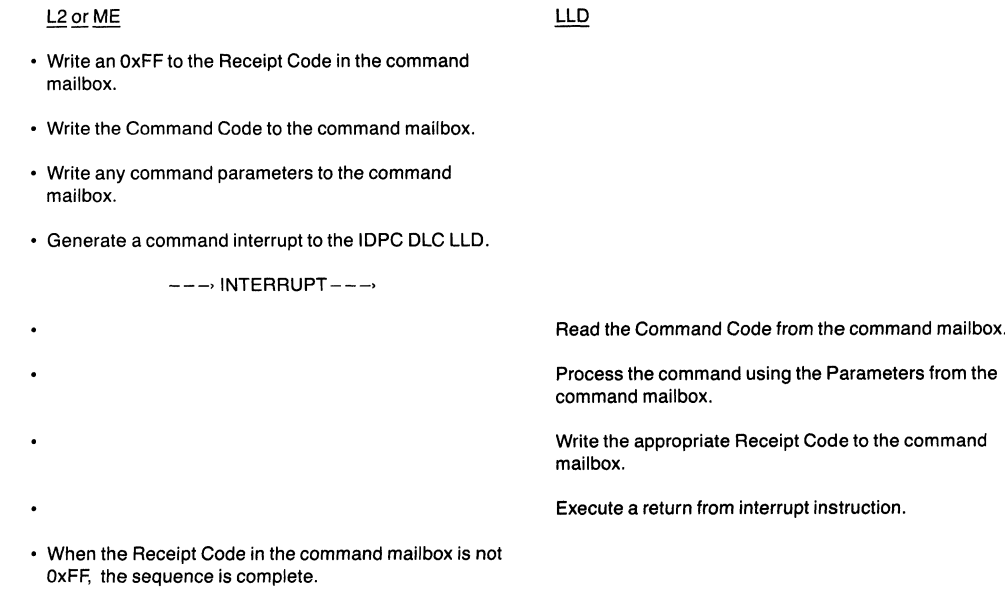


Table 5-7. IDPC DLC LLD Command Codes Summary

Code (Hex)	Description	MB I/F	Module
00	Transmit a Buffer	L2	DLC
01	Initialize the DLC	ME	DLC
02	DLC Control	ME	DLC
03	Update Address Recognition	ME	DLC
04	Abort the Current Transmit	ME	DLC
05	Load a New Event Enables	ME	DLC
06	Begin Remote Loopback	ME	DLC
07	End Remote Loopback	ME	DLC
08	Begin Local Loopback	ME	DLC
09	End Local Loopback	ME	DLC

EVENT SEQUENCES

Events are messages passed from the IDPC DLC LLD back to either the L2 protocol or the ME. Figure 5-3 shows a typical event sequence. Note that the Receipt Code returned by the L2 or ME may indicate that the command has simply been received or that some data are available. This depends upon the particular event issued.

Table 5-8 lists the valid event codes which the IDPC DLC LLD will generate. Each event is described in detail in a later section.

Figure 5-3. Typical Event Sequence

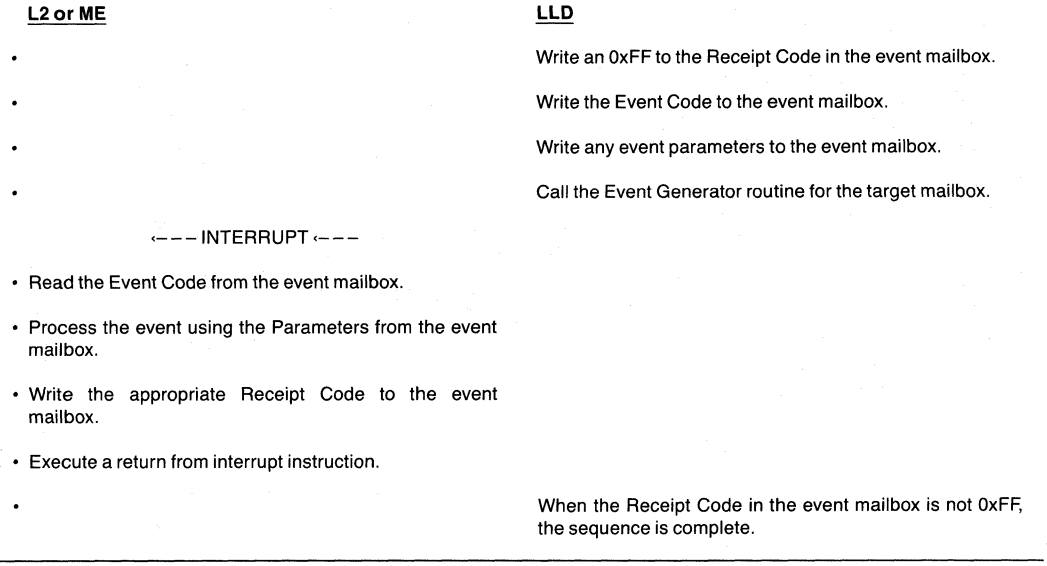


Table 5-8. IDPC DLC LLD Event Codes Summary

Code (Hex)	Description	MB I/F	Module
0	Transmission Complete	L2	DLC
1	Packet Received	L2	DLC
B	Error Status	ME	DLC
C	Buffer Allocation Request	ME	DLC

PROGRAMMING

This section describes the command and event primitives used with the IDPC DLC LLD. These primitives are compatible with those used by AmLink LAPD/LAPB and those provided by the DSC LLD.

Each primitive description contains the following information:

PRIMITIVE CODE:	Command or event code for the primitive.
MAILBOX:	Mailbox to be used with this primitive.
INPUTS:	Input parameters for this primitive. These are identified as mailbox parameter bytes 0 to 15.
OUTPUTS:	Output parameters for this primitive. These are identified as mailbox parameter bytes 0 to 15.
RECEIPT CODES:	Possible Receive Codes for this primitive.
DESCRIPTION:	Describes the functions and services provided by this primitive.
NOTES:	Describes any special considerations related to this primitive.

The command primitives are:

Commands	Description
XMITBUF	Transmit a buffer.
DLC_INIT	Initialize the DLC.
DLC_CONTROL	Enable/disable DLC transmitter and/or receiver.
UPDATE ADDR_RECOG()	Update address recognition parameters.
XMIT_ABORT	Abort the current buffer transmission.
LOAD EVENT ENABLES	Load event reporting enable/disable bit array.
BEGIN-REMOTE_LOOP	Begin remote Loop back.
END_REMOTE_LOOP	End remote Loop back.
BEGIN_LOCAL_LOOP	Begin local Loop back.
END_LOCAL_LOOP	End local Loop back.

The event primitives are:

Events	Description
PACKET_RCVD	Packet received without error.
XMIT_DONE	Buffer transmitted without error.
ERROR STATUS	A valid address or an end-of address has been received.
BUFFER SERVICE	Buffer allocation/deallocation service.

PRIMITIVE: **XMITBUF**

COMMAND CODE: 0

MAILBOX: L2

INPUTS: Buffer Address (parameter bytes 0-3)

Packet Length (parameter bytes 4-5)

Buffer Length (parameter bytes 6-7)

Buffer Reference Number (parameter bytes 8-9)

OUTPUTS: None

RECEIPT CODES: 00 = Command was received.

01 = Illegal command.

02 = Illegal parameter.

03 = Transmitter busy.

FF = Command not yet received.

DESCRIPTION: This primitive initiates a buffer transmission via the IDPC DLC. If the buffer size is larger than the packet size, the IDPC DLC LLD will automatically send multiple packets until the entire buffer is sent. The buffer size MUST be an integral multiple of the packet size or an illegal parameter receipt code will be returned.

This primitive is equivalent to the PH-DATA request primitive.

If this primitive is issued while the transmitter is busy from a previously issued, but unfinished, XMITBUF, a Transmitter Busy receipt code is returned. The user may re-issue the primitive after the current transmission is complete as indicated by an XMITDONE event.

NOTES:

PRIMITIVE: **DLC_INIT**

COMMAND CODE: 1

MAILBOX: ME

INPUTS: None, uses the DLC Initialization Parameter Block (IPB) in the LLD RAM Interface Block.

OUTPUTS: None.

RECEIPT CODES: 00 = Initialization is complete.

01 = Illegal command.

FF = Command not yet complete.

DESCRIPTION: This primitive places the IDPC DLC into a known state. The primitive installs the DLC IPB parameters, disables address recognition, gets a buffer for the DLC receiver, enables 80188/86 receive DMA Channel 0, and enables all DLC interrupts.

NOTES:

PRIMITIVE: DLC Control

COMMAND CODE: 0x02

MAILBOX: ME

INPUTS: DLC Transmitter Enable/Disable (parameter byte 0) Enable = 1; Disable = 0

DLC Receiver Enable/Disable (parameter byte 1) Enable = 1; Disable = 0

OUTPUTS: None

RECEIPT CODES: 00 = Command is complete.

01 = Illegal command.

02 = Illegal parameter.

FF = Command not yet complete.

DESCRIPTION: This primitive allows the user to enable or disable the DLC transmitter and/or receiver.

NOTES:

PRIMITIVE: UP_ADDR_RECOGNITION

COMMAND CODE: 0x03

MAILBOX: ME

INPUTS: Parameter byte 0 - Enable (=1)/Disable (=0) address recognition for the specified Address Register Number (parameter byte 1).

Parameter byte 1 - Address Register Number. Parameter byte 1 = 0, 1, 2, 3 for Address Recognition Register 0, 1, 2, or 3. Parameter byte 1 = 4 for Broadcast Address Recognition. Parameter bytes 2-3 - Address. Contents to be loaded into the specified address recognition register. If single byte addresses are enabled in the DLC IPB, parameter byte 2 contains that address regardless of which address byte (1st or 2nd) is enabled. Parameter bytes 2-3 are ignored if parameter byte 1 = 4 (Broadcast Address Recognition).

OUTPUTS: None

RECEIPT CODES: 00 = Command is complete.

01 = Illegal command.

02 = Illegal parameter.

FF = Command not yet complete.

DESCRIPTION: This primitive allows the DLC address recognition services to be used. Address recognition can be used as a bandpass filter, allowing only packets with the specified addresses to be received.

The IDPC DLC supports up to four programmable addresses plus a fixed Broadcast Address (all 1's address).

The DLC IPB contains the following fields which condition address recognition:

Address length (single byte or two bytes)

Single byte address select (1st or 2nd)

Ignore C/R address bit

NOTES: See section on DLC IPB for further information.

PRIMITIVE: TRANSMIT ABORT

COMMAND CODE: 4

MAILBOX: ME

INPUTS: None

OUTPUTS: None

RECEIPT CODES: 00 = Command is complete.

01 = Illegal command.

FF = Command not yet complete.

DESCRIPTION: This primitive causes any buffer transmission that was previously issued to be aborted.

Execution of this primitive causes the IDPC DLC LLD to issue an XMITDONE event primitive to the Layer 2.

NOTES:

PRIMITIVE: LOAD_EVENT_ENABLES

COMMAND CODE: 5

MAILBOX: ME

INPUTS: Event Enables. This is an eight byte array. Each bit in the array represents one of 64 events. The bit position for a particular event mask corresponds to the event code for the event. A ONE in a bit position enables the corresponding event to be reported via interrupt to the layer (L2 or ME) appropriate for the particular event. A ZERO in a bit position disables that event from being reported. For instance, the event mask bit for the "Error Status" event (Event Code 0x0B) is byte #1, bit #3.

OUTPUTS: None.

RECEIPT CODES: 00 = Update is complete.

01 = Illegal command.

02 = Illegal parameter.

FF = Command not yet complete.

DESCRIPTION: This primitive causes a new event mask to be used by the IDPC DLC LLD. The event mask allows event reporting to be individually enabled or disabled.

NOTES: Events may still occur even though they are not reported if event reporting is disabled.

PRIMITIVE: **BEGIN_REMOTE_LOOP**

COMMAND CODE: 6

MAILBOX: ME

INPUTS: None

OUTPUTS: None

RECEIPT CODES: 00 = Command is complete.

01 = Illegal command.

FF = Command not yet complete.

DESCRIPTION: This primitive places the IDPC DLC in Remote Loopback. While Remote Loopback is enabled, all received packets will be looped back to the far end transmitter. Received packets will also be received by the local DLC if the DLC Receiver is enabled. Any packets transmitted by the local DLC Transmitter will not be transmitted but will be thrown in the bit bucket.

NOTES:

PRIMITIVE: **END_REMOTE_LOOP**

COMMAND CODE: 7

MAILBOX: ME

INPUTS: None

OUTPUTS: None

RECEIPT CODES: 00 = Command is complete.

01 = Illegal command.

FF = Command not yet complete.

DESCRIPTION: This primitive disables IDPC DLC Remote Loopback. If the Remote Loopback is not enabled, the command does nothing.

NOTES:

PRIMITIVE: **BEGIN_LOCAL_LOOP**

COMMAND CODE: 8

MAILBOX: ME

INPUTS: None

OUTPUTS: None

RECEIPT CODES: 00 = Command is complete.

01 = Illegal command.

FF = Command not yet complete.

DESCRIPTION: This primitive places the IDPC DLC into a Local Loopback mode. In this mode, data transmitted by the IDPC DLC is looped back to the IDPC DLC receiver.

NOTES:

PRIMITIVE: **END_LOCAL_LOOP**

COMMAND CODE: 9

MAILBOX: ME

INPUTS: None

OUTPUTS: None

RECEIPT CODES: 00 = Command is complete.

01 = Illegal command.

FF = Command not yet complete.

DESCRIPTION: This primitive disables IDPC DLC Local Loopback, if enabled.

NOTES:

PRIMITIVE: **XMITDONE**

EVENT CODE: 0

MAILBOX: L2

INPUTS: None

OUTPUTS: None

RECEIPT CODES: 00 = Event has been received.

01 = Illegal event.

FF = Event not yet recognized.

DESCRIPTION: This primitive is used to notify the L2 code that the last Transmit Buffer LLD Command issued has been completed and the IDPC DLC transmitter is ready to transmit another buffer.

This event is issued when normal buffer transmission is finished, when the Transmit Underrun event occurs or when the Transmit Abort command is executed.

NOTES:

PRIMITIVE: **PACKET_RCVD**

EVENT CODE: 1

MAILBOX: L2

INPUTS: Buffer Address (parameter bytes 0-3)

Packet Length (parameter bytes 4-5)

Buffer Reference Number (parameter bytes 6-7)

OUTPUTS: None

RECEIPT CODES: 00 = Event has been received.

01 = Illegal command.

02 = Illegal parameter.

FF = Event not yet received.

DESCRIPTION: This primitive is used to notify the L2 code that a packet has been successfully received.

NOTES: The user should acknowledge this primitive with as little delay as possible. This routine will request a new receiver buffer immediately after the event is acknowledged. If acknowledgment is delayed too long, a receiver overrun error may occur.

PRIMITIVE: ERROR STATUS

EVENT CODE: 0x0B

MAILBOX: ME

INPUTS: None

OUTPUTS: Parameter byte 0 -

Bit 0 - Receiver abort condition.
1 - Receiver non-integer # of bytes.
2 - Reserved.
3 - Receiver CRC error.
4 - Receiver Long Packet error.
5 - Receiver Short Packet error.
6 - Receiver overrun error.
7 - Transmitter Underrun error.

RECEIPT CODES: 00 = Event has been received.

01 = Illegal event.

FF = Event not yet received.

DESCRIPTION: This primitive is used to notify the ME that one or more DLC error or exceptional conditions has occurred.

All of the receiver error/exceptions are mutually exclusive for the IDPC DLC receiver. In other words, only one of bits 0-6 of parameter byte 0 may be set in any single occurrence of this event. However, the Transmitter Underrun error bit may be set simultaneously with one of the receive error/exception bits.

When one of the received error/exception conditions occurs, Layer 2 is not notified. If the Transmitter Underrun condition occurs, a Transmit Done event to Layer 2 is executed in addition to the Error Status event.

NOTES: The Non-Integer Number of Bytes condition indicates that the last byte of a received packet contains less than 8 bits.

The Long Packet error occurs when the number of bytes in a received packet exceeds the Maximum Packet Size field of the DLC IPB.

The Short Packet error occurs when the number of bytes in a received packet is less than the Minimum Packet Size field of the DLC IPB.

PRIMITIVE: IDPC DLC LLD Initialization

COMMAND CODE: Not Applicable

MAILBOX: Not Applicable

INPUTS: None, uses Configuration Table information.

OUTPUTS: None

RECEIPT CODES: 00 = Initialization is complete.

DESCRIPTION: This routine 'illdinit()' is used to initialize the IDPC DLC Initialization Parameter Block (IPB) and IDPC DLC LLD Private RAM area. This routine must be called prior to using any IDPC DLC LLD mailbox services. Normally, the routine should be part of the system POR initialization sequence.

NOTES: The user should issue the DLC INIT command primitive to install the DLC IPB into the IDPC DLC hardware. The user may modify the IPB fields prior to issuing the DLC Init command. Once the DLC Init command has been executed, the DLC CONTROL command must be executed to enable the IDPC transmitter and receiver.

PRIMITIVE: **BUFF_SERVICE**

EVENT CODE: 0x0C

MAILBOX: ME

INPUTS: MODE (parameter byte 0) - A ZERO indicates that an allocation is requested.

SIZE (parameter byte 1 and 2) - The size in bytes of the buffer requested.

OUTPUTS: ADDR (parameter bytes 3-6) - Contains the base address of the buffer requested.

REFNO (parameter byte 7-8) - Contains the reference number of the buffer provided.

RECEIPT CODES: 00 = Initialization is complete.

01 = Illegal command.

02 = Illegal parameter.

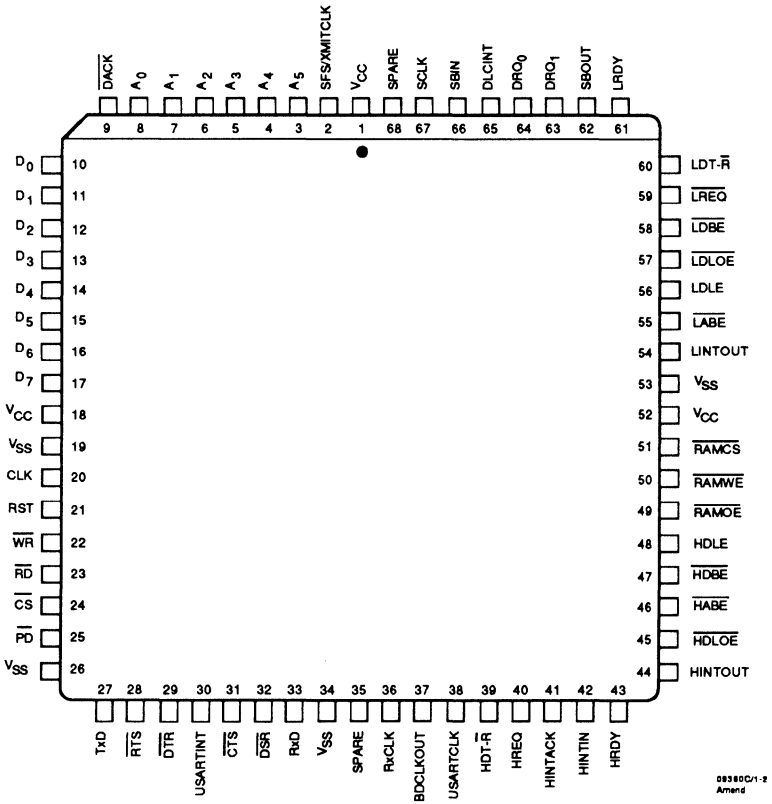
FF = Command not yet complete.

DESCRIPTION: This primitive is used to request that a buffer be allocated to the IDPC DLC LLD. This primitive is used to obtain a buffer for the DLC receiver. The deallocate option is not used by the IDPC DLC LLD. The REFNO is an arbitrary integer associated with the buffer by the ME when the buffer is allocated.

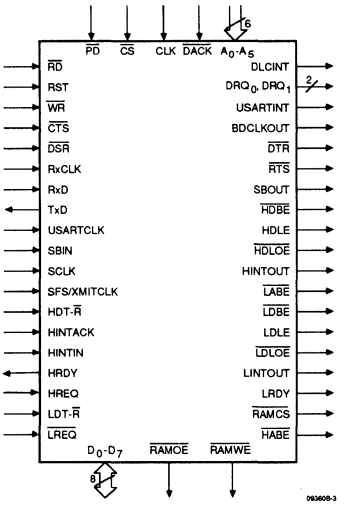
NOTES: The ME must be able to supply a buffer of at least the size programmed in the Maximum Packet Size field in the IDPC DLC Initialization Parameter Block.

APPENDIX

CONNECTION DIAGRAM



LOGIC SYMBOL



PIN DESCRIPTION

The interface pins of the 68-pin IDPC chip can be classified into six major groups which include:

- Processor Bus Interface (25 pins)
- USART Interface (9 pins)
- Serial Bus Port Interface (4 pins)
- Bus Arbitration Control (21 pins)
- Power/Ground (7 pins)

Processor Bus Interface

A_0-A_5 Address Lines (Input)
These six address lines are generated by the external processor to select internal registers of the IDPC. The address lines are valid only when $\overline{CS}$ is active (LOW).

CLK Master Clock (Input)
The Master Clock is an input that provides synchronization and timing for internal IDPC logic functions. CLK is normally the same clock used by the CPU.

$\overline{CS}$ Chip Select (Input; Active LOW)
$\overline{CS}$ is an externally developed signal used to indicate that the IDPC has been selected for a read or write cycle (viewed as memory by the external processor).

D_0-D_7 Data Lines (Input/Output; Three State)
D_0-D_7 are bidirectional data lines used to transfer data between the locally attached processor and the IDPC. The direction of the data transfer is controlled by the read ($\overline{RD}$) and write ($\overline{WR}$) control lines. When $\overline{CS}$ is invalid (HIGH), the data lines remain in a high-impedance state.

$\overline{DACK}$ DMA Acknowledge (Input; Active LOW)
The DACK signal is an indication that the DMA controller is executing a DMA cycle to the DLC Transmit FIFO. This indication occurs early in the DMA cycle, allowing the Transmit FIFO to de-activate the DRQ_1 signal when the last data transfer takes place (before an unwanted DMA cycle is initiated). An equivalent signal is not required for the DLC Receive FIFO operation.

DLCINT DLC Interrupt (Output; Active HIGH)
DLCINT goes active (HIGH) any time the Data Link Controller (DLC) portion of the IDPC sets a status bit and the associated interrupt enable bit is active.

DRQ_0 Receive DMA Request (Output; Active HIGH)
DRQ_0 is an active-HIGH output used by the receive FIFO portion of the DLC to begin a DMA cycle for the receive data. DRQ_0 goes active (HIGH) when the receiver portion of the DLC loads the number of data bytes into the receive FIFO specified by the receive FIFO threshold in the FIFO Threshold Register, or an "end of packet" is loaded into the FIFO.

DRQ_0 is de-activated at reset, when the receive FIFO is emptied, or when the last byte of a packet is transferred from the receive FIFO to external memory.

DRQ_1 Transmit DMA Request (Output; Active HIGH)
DRQ_1 is an active-HIGH signal used by the transmit FIFO of the DLC to request the start of a DMA cycle for the Transmit Data.

DRQ_1 goes HIGH when ALL of the following conditions are met:

1) Transmit byte count is not equal to zero,
2) Last byte of the packet has not been loaded into the FIFO, and
3) The number of bytes in the FIFO is equal to or less than the value programmed into the transmit FIFO threshold.

DRQ_1 is de-activated (LOW) at reset when the FIFO buffer is full, or when the last byte of the packet is loaded into the FIFO.

$\overline{PD}$ Power Down (Input; Active LOW)
When active, this signal disables all internal clocks and places all three-state signals in high-impedance state. HRDY and LRDY are driven active (HIGH) and all interrupt outputs are de-activated. Status and data may be lost but programming is retained.

$\overline{RD}$ Read (Input; Active LOW)
This input is used internally by the IDPC to indicate when read data (output data from the IDPC) is to be latched by the external host (negative to positive transition of $\overline{RD}$). $\overline{RD}$ is qualified internally with an active $\overline{CS}$ input (LOW).

RST Reset (Input; Active HIGH)
When active (HIGH), the reset line forces all functions to terminate and places the IDPC in a default state (described later in this data sheet). HRDY and LRDY are driven active (HIGH) and all three-state outputs are placed in high-impedance state.

USARTINT USART Interrupt (Output; Active HIGH)
USARTINT goes active (HIGH) any time the USART section of the IDPC sets a status bit and the associated interrupt enable bit is active.

$\overline{WR}$ Write (Input; Active LOW)
$\overline{WR}$ is used internally by the IDPC to latch incoming data (D0-D7) during a write cycle. $\overline{WR}$ is qualified internally with an active $\overline{CS}$ input (LOW).

USART Interface

BDCLKOUT Baud Rate Generator Clock Out (Output)
This signal is the output of the final stage of the IDPC's internal baud rate generator. This signal is used as a common clocking source for a modem or other similar application.

$\overline{CTS}$ Clear To Send (Input; Active LOW)
This signal is a TTL-level input to the IDPC. Activity on $\overline{CTS}$ generates a maskable interrupt but does not directly control the USART.

$\overline{DSR}$ Data Set Ready (Input; Active LOW)
$\overline{DSR}$ is a TTL-level input to the IDPC. Activity on $\overline{DSR}$ generates a maskable interrupt but does not directly control the USART.

$\overline{DTR}$ Data Terminal Ready (Output; Active LOW)
$\overline{DTR}$ is a TTL-level output from the the IDPC. This signal is user-controlled and does not directly affect USART operation.

$\overline{RTS}$ Request To Send (Output; Active LOW)
$\overline{RTS}$ is a TTL-level status output from the IDPC. This signal is user-controlled and does not directly affect USART operation.

RxCLK Receive Clock (Input)
RxCLK is an input to the USART portion of the IDPC used in synchronous and asynchronous operation. In asynchronous mode, the RxCLK is 16 times the data rate. In synchronous mode, the RxCLK is synchronized to the incoming data, and the positive-going edge is used to latch the incoming Receive Data (RxD).

RxD Receive Data (Input; Active HIGH)
RxD is the TTL-level serial data input to the IDPC's internal USART. The data are clocked into the IDPC on the positive-going edge of the selected clock source.

TxD Transmit Data (Output; Active HIGH)
TxD is the TTL-level serial data output of the IDPC's inter-

nal USART. The data are clocked out of the IDPC on the negative edge of the selected clock source.

USARTCLK USART Clock (Input)
This pin is the input for the internal baud rate generator. The frequency of this clock source must be integer multiples of the desired baud rate (output of the baud rate generator is the same as the data rate for synchronous operation and 16 times the data rate for asynchronous operation). If the baud rate generator is programmed to divide by one, USARTCLK operates as a direct input to the USART. When the IDPC is used in conjunction with the Am79C30 (DSC), the 12.288 MHz clock output can be used as the USART clock source.

Serial Bus Port Interface

SBIN Serial Data In (Input; Active HIGH)
SBIN is the serial data input to the DLC portion of the IDPC and is clocked into the DLC LSB (bit 0) first on the positive edge of the Serial Clock In (SCLK).

Serial data may be input as free-running data or gated data using the SFS/XMITCLK pin (described in greater detail later in this document). The data will range in speed from 0 to 2.048 Mbps. In applications where an Am79C30A (DSC) is used, SBIN of the IDPC is tied to SBOUT of the DSC.

SBOUT Serial Data Out (Output; Active HIGH, Open Drain)
SBOUT is the serial data output of the DLC portion of the IDPC. The serial data is clocked out, LSB (bit 0) first, on the negative edge of either SCLK or SFS/XMITCLK. SBOUT data will range in speed from 0 to 2.048 Mbps. In applications where an Am79C30A (DSC) is used, SBOUT of the IPDC is tied to SBIN of the DSC.

SCLK Serial Clock In (Input)
SCLK is an input to the IDPC that is used as the clocking source for the DLC.

In one mode of operation, SCLK acts as both the transmit and receive clock synchronized to SFS/XMITCLK. In a second mode of operation, SCLK is used only as the receive clock and is not synchronized to SFS/XMITCLK. The positive edge of SCLK is used to latch receive data on SBIN and the negative edge is used to shift transmit data out on SBOUT.

SFS/XMITCLK Serial Frame Sync/Transmit Clock (Input)
This input clock signal has two different functions depending on the mode of operation selected by bits 0-4 of the SBP Control Register. In the gated mode, this input functions as SFS, the synchronization pulse used to indicate the first of up to 31 independent 8-bit time slots on SBIN and SBOUT.

In the second mode, SFS/XMITCLK is used by the DLC as the input for an independent transmit clock. SFS/XMITCLK is used by the DLC to shift data out on SBOUT (Serial Bus Out), LSB (bit 0) first, on the negative edge. This clock operates from 0 to 2.048 Mbps.

Bus Arbitration Control

$\overline{\text{HDBE}}$ Host Data Bus Enable (Output; Active LOW)
$\overline{\text{HDBE}}$ is an active-LOW output used to enable the data lines from the host processor to the shared RAM data bus. $\overline{\text{HDBE}}$ is driven active as a result of HDT-$\overline{\text{R}}$ being driven HIGH (write cycle). It remains HIGH until the end of the memory cycle.

$\overline{\text{HABE}}$ Host Address Bus Enable (Output; Active LOW)
$\overline{\text{HABE}}$ is driven active LOW by the IDPC as a result of receiving an HREQ from the host processor and is used to enable the address lines from the host processor. $\overline{\text{HABE}}$ remains active until the end of the memory cycle.

HDLE Host Data Latch Enable (Output; Active HIGH)
This active-HIGH output is used to latch data from the RAM to the host processor. HDLE is driven HIGH (the latch is made transparent) as a result of $\overline{\text{HDT-R}}$ going LOW (read cycle). It returns LOW at the end of the memory cycle.

$\overline{\text{HDLOE}}$ Host Data Latch Output Enable (Output; Active LOW)
$\overline{\text{HDLOE}}$ is an active-LOW output from the IDPC used by the host processor to enable the output of the data bus latches back to the host processor. $\overline{\text{HDLOE}}$ is driven active (LOW) when $\overline{\text{HDT-R}}$ is driven LOW (read cycle). It is cleared (HIGH) when HREQ goes inactive (LOW).

HDT-$\overline{\text{R}}$ Host Data Transmit-$\overline{\text{Receive}}$ (Input)
HDT-$\overline{\text{R}}$ indicates the direction of host processor accesses to shared memory. When the signal goes HIGH, it indicates that a RAM write cycle is in progress. As a result, $\overline{\text{RAMWE}}$ and $\overline{\text{HDBE}}$ are driven active (LOW).

When HDT-$\overline{\text{R}}$ goes LOW, it indicates that a RAM read cycle is in progress. At this time $\overline{\text{RAMOE}}$ and $\overline{\text{HDLOE}}$ are driven active (LOW), and HDLE is driven active (HIGH).

HINTACK Host Interrupt Acknowledge (Input; Active HIGH)
HINTACK is generated by the host processor in response to a Host Interrupt Out (HINTOUT) signal from the IDPC. HINTACK is used in the IDPC to clear bit 0 of the Semaphore Register, dropping HINTOUT.

HINTIN Host Interrupt In (Input; Active HIGH)
This signal is used by the host processor to generate an interrupt to the local processor (LINTOUT). When HINTIN is pulsed active (HIGH), it causes bit 1 of the Semaphore Register to be set to a '1' which generates LINTOUT.

HINTOUT Host Interrupt Out (Output; Active HIGH)
When activated, HINTOUT generates an interrupt to the host processor. This signal goes active (HIGH) when the local processor writes a '1' to bit 0 of the Semaphore Register. HINTOUT is de-activated by a pulse on the HINTACK pin. HINTOUT is intended to be connected to an interrupt input on the host processor. HINTOUT is de-activated by reset.

HRDY Host Ready (Output; Active HIGH) Open Drain
HRDY is an active-HIGH output from the IDPC used by the host processor to complete a shared RAM memory cycle. HRDY is normally HIGH. It is driven LOW when a request for the RAM is received from the host processor (HREQ). HRDY is returned HIGH at the end of the memory cycle, or by reset.

HREQ Host Processor Bus Request (Input; Active HIGH)
The HREQ is an active-HIGH input to the IDPC from the host processor requesting access to the shared RAM. HREQ is sampled on the negative edge of every IDPC clock cycle. When sampled active, HREQ drives $\overline{\text{RAMCS}}$ and $\overline{\text{LABE}}$ active (LOW), and HRDY inactive (HIGH). HREQ is an asynchronous input with respect to the master clock and is synchronized internally.

$\overline{\text{LABE}}$ Local Address Bus Enable (Output; Active LOW)
This signal is driven LOW by the IDPC to enable the address lines from the local processor bus onto the memory bus when a Local Processor Bus Request ($\overline{\text{LREQ}}$) is received from the local processor. $\overline{\text{LABE}}$ remains active-LOW until the end of the memory cycle.

$\overline{\text{LDBE}}$ Local Data Bus Enable (Output; Active LOW)
This signal is used to place the data from the local processor onto the shared RAM data bus. $\overline{\text{LDBE}}$ is driven active as a result of LDT-$\overline{\text{R}}$ being driven HIGH (write cycle). The local data bus enable remains HIGH until the end of a memory cycle.

LDLE Local Data Latch Enable (Output; Active HIGH)
This signal goes HIGH to latch data from the RAM onto the local processor data bus. LDLE is driven HIGH (latch made transparent) as a result of LDT-$\overline{\text{R}}$ going LOW (read cycle). LDLE returns LOW at the end of a memory cycle.

$\overline{\text{LDLOE}}$ Local Data Latch Output Enable (Output; Active LOW)
This signal is an active-LOW output from the IDPC that enables the output of the data bus latch onto the local processor. $\overline{\text{LDLOE}}$ is driven active (LOW) when LDT-$\overline{\text{R}}$ is driven LOW (read cycle) and is cleared when $\overline{\text{LREQ}}$ goes inactive (HIGH).

LDT-$\overline{\text{R}}$ Local Data Transmit-$\overline{\text{Receive}}$ (Input)
LDT-$\overline{\text{R}}$ indicates the direction of local processor accesses to shared memory. When the signal goes HIGH it indicates that a RAM write cycle is in progress. As a result, $\overline{\text{RAMWE}}$ and $\overline{\text{LDBE}}$ are driven active (LOW).

When LDT-$\overline{\text{R}}$ goes LOW it indicates that a RAM read cycle is in progress. At this time $\overline{\text{RAMOE}}$ and $\overline{\text{LDLOE}}$ are driven active (LOW), and LDLE is driven active (HIGH).

LINTOUT Local Interrupt Out (Output; Active HIGH)
When activated, LINTOUT generates an interrupt to the local processor. This signal goes active (HIGH) when bit 1 in the Semaphore Register is set to a logic '1' when a pulse from the host is applied to the HINTIN pin. LINTOUT goes LOW when the register bit is cleared to '0' by software (writing a '0' to bit 1 of the Semaphore Register), or after a reset.

LRDY Local Ready (Output; Active HIGH) Open Drain
LRDY is an active-HIGH output from the IDPC used by the local processor to complete a shared RAM memory cycle. LRDY is normally HIGH, and is driven LOW when a request for RAM is received from the local processor ($\overline{\text{LREQ}}$) and the host processor is currently accessing shared RAM.

$\overline{\text{LREQ}}$ Local Processor Bus Request (Input; Active LOW)
This active-LOW signal is generated as an input to the IDPC by the local processor when it requests access to the shared RAM. $\overline{\text{LREQ}}$ is sampled on the negative edge of every IDPC master clock cycle. $\overline{\text{LREQ}}$ must be synchronous to CLK.

$\overline{\text{RAMCS}}$ $\overline{\text{RAM Chip Select}}$ (Output; Active LOW)
This signal is an active-LOW output from the IDPC used by the shared RAM as its chip select. $\overline{\text{RAMCS}}$ is driven LOW when either $\overline{\text{LREQ}}$ or HREQ is sampled active. $\overline{\text{RAMCS}}$ remains active until the end of a memory cycle.

$\overline{\text{RAMOE}}$ $\overline{\text{RAM Output Enable}}$ (Output; Active LOW)
This signal is an active-LOW output signal from the IDPC used by the shared RAM to enable its output drivers. $\overline{\text{RAMOE}}$ is driven active LOW when either LDT-$\overline{\text{R}}$ or HDT-$\overline{\text{R}}$ is driven LOW (read cycle). $\overline{\text{RAMOE}}$ is cleared (HIGH) at the end of the memory cycle.

$\overline{\text{RAMWE}}$ $\overline{\text{RAM Write Enable}}$ (Output; Active LOW)
This signal is an active-LOW output from the IDPC used by the shared RAM as a write strobe. $\overline{\text{RAMWE}}$ is driven LOW when either LDT-$\overline{\text{R}}$ or HDT-$\overline{\text{R}}$ goes HIGH (write cycle). $\overline{\text{RAMWE}}$ remains active until the end of a memory cycle.

Power/Ground

V_{CC} + 5-V Power Supply
V_{SS} Ground

Register Description

The IDPC is controlled via internal registers that are written and read by software running on the external "local" processor connected to the IDPC external bus. These internal registers may be mapped into either memory or I/O space, but typically are memory mapped.

The internal registers occupy a 64-byte block located in the local processor's memory address space. The starting address of the memory block is determined by address decode logic (external to the IDPC) that is used to generate the IDPC Chip Select signal ($\overline{\text{CS}}$). The registers and their respective memory offset values are provided in Tables 1-4.

In systems containing more than one microprocessor (e.g., a workstation application with host processor and local processor), only the local processor can access the IDPC registers. The host processor, however, can control IDPC operations indirectly by issuing requests to the local processor via shared memory supported by the Dual-Port Memory Controller.

The programmable registers are used for establishing modes of operation, configuring the IDPC, and monitoring/reporting status.

Table 1. IDPC Address Map

Offset (Hex)	Block
00-1F	DLC
20-3E	USART
3F	DPMC

Detailed Description of User-Visible DLC Registers

The DLC contains 23 registers, as shown in Table 2.

Table 2. DLC Registers

Offset (Hex)	Register Name	Size (Bytes)	Type
00	Command/Control Register	1	Read/Write
01	Address Control Register	1	Read/Write
02	Link Address Recognition Register 0	2	Read/Write
04	Link Address Recognition Register 1	2	Read/Write
06	Link Address Recognition Register 2	2	Read/Write
08	Link Address Recognition Register 3	2	Read/Write
0A	Serial Bus Port Control Register	1	Read/Write
0B	Minimum Receive Packet Size Register	1	Read/Write
0C	Maximum Receive Packet Size Register	2	Read/Write
0E	Interrupt Source Interrupt Enable Register	1	Read/Write
0F	Receive Frame Interrupt Enable Register	1	Read/Write
10	Receive Link Interrupt Enable Register	1	Read/Write
11	FIFO Status Interrupt Enable Register	1	Read/Write
12	Transmit Byte Count Register	2	Read/Write
14	FIFO Threshold Register	1	Read/Write
15	Interrupt Source Register	1	Read Only
16	Receive Byte Count Register	2	Read Only
18	Receive Frame Status Register	1	Read Only
19	Receive Link Status Register	1	Read Only
1A	FIFO Status Register	1	Read Only
1B	Receive FIFO Data Register	1	Read Only
1C	Transmit FIFO Data Register	1	Write Only
1D	Residual Bit Control Status Register	1	Read/Write
1E-1F	Reserved	2	0

DLC Command/Control Register (00 Hex)

7	6	5	4	3	2	1	0
ENABLE FCS PASS-THRU	DLC RESET	ENABLE CRC GENERATE	ENABLE CRC CHECK	FLAG/ MARK IDLE SELECT	RECVER ENABLE	XMIT ENABLE	SEND ABORT

This register is used to set up and control basic transmitter and receiver functions.

Bit 0: Send Abort (Default = 0)—When set to a '1' the DLC transmitter abort generator transmits abort characters (01111111, LSB on right). If this bit is set and cleared on two successive writes, the DLC will transmit one abort character. The transmitter continues to send aborts as long as this bit is set. Aborts are always sent in whole bytes. Setting and clearing this bit on successive CPU writes will cause one complete abort to be sent.

When this bit is set the DLC transmit FIFO, DLC byte counter, and the DLC Transmit Byte Count Register are cleared.

Bit 1: Transmitter Enable (Default = 0)—When set to a '1', this bit allows data from the DLC to be shifted out to the SBOUT pin under control of SCLK or SFS/XMITCLK.

When this bit is cleared, the SBOUT pin is placed in an open-drain condition. If this bit is cleared and the DLC is transmitting data, the DLC waits until the current frame is complete before disabling the SBOUT pin.

Bit 2: Receiver Enable (Default = 0)—This bit is set to a '1' to allow data from the SBIN pin to be clocked onto the Serial Bus Port.

When cleared to a '0', the DLC receiver is disabled. If the bit is cleared while the DLC receiver is in the middle of a receive frame, the receiver will finish processing the frame before shutting down.

Bit 3: Flag/Mark Idle (Default = 0)—When set to a '1', the DLC transmitter continuously transmits the flag idle pattern when not in frame.

When this bit is cleared to a '0', the DLC transmitter continuously transmits the mark idle pattern when not in frame.

Bit 4: CRC Check Enable (Default = 1)—When this bit is set to a '1', the result of the CRC check is transferred to the CRC error bit (bit 2) in the Receive Frame Status Register.

When this bit is cleared, the CRC error bit in the Receive Frame Status Register is never set.

Bit 5: CRC Generate Enable (Default = 1)—When set, this bit causes the transmit CRC (which is always being calculated) to be transmitted following the byte tagged as EOP in the transmit FIFO.

If this bit is cleared, a closing flag is transmitted immediately following the EOP-tagged byte.

Bit 6: DLC Reset (Default = 0)—When this bit is set to a '1', all DLC FIFOs, latches and status/control bits are forced to their default values. A delay of eleven master clock (CLK) cycles is required before the DLC registers can be accessed.

Bit 7: FCS Pass-Thru Enable (Default = 0)—When set to a '1', this bit allows the frame check sequence (CRC) bytes to be loaded into the FIFO as data (receive side).

When cleared, the frame check sequence is discarded.

DLC Address Control Register (01 Hex)

7	6	5	4	3	2	1	0
FIRST/ SECOND BYTE SELECT	ENABLE C/R BIT CMPARE	1 – 2 BYTE ADDR SELECT	ENABLE BRDCST ADDR DETECT	ENABLE ADDR DETECT 3	ENABLE ADDR DETECT 2	ENABLE ADDR DETECT 1	ENABLE ADDR DETECT 0

All bits in the DLC Address Control Register are set and cleared by software except when initialized to default values as the result of a reset.

The DLC Address Control Register can be written and read by the local processor. When all link address enable bits (bits 0-3) and the broadcast enable bit (bit 4) are cleared to '0', the DLC does not perform address detection, and passes all received frame bytes to the DLC receive FIFO. In this case, bits 5-7 are ignored.

If one or more of the link address enable bits (bits 0-4) are set, then a successful link address compare must occur before any frame bytes can be transferred to the DLC receive FIFO.

Bit 0: Address Register 0 Enable (Default =)—Link address 0 enable.

Bit 1: Address Register 1 Enable (Default =)—Link address 1 enable.

Bit 2: Address Register 2 Enable (Default = 0)—Link address 2 enable.

Bit 3: Address Register 3 Enable (Default = 0)—Link address 3 enable.

NOTE: When set to a '1', bits 0-3 enable comparison of a received frame address with the contents of the DLC Link Address Recognition Registers 0 through 3, respectively.

The comparison of a received frame address with the contents of all enabled Address Recognition Registers is conditioned by bits 5-7 of this register.

Bit 4: Broadcast Address Enable (Default = 1)—When set to a '1', this bit enables comparison of a receive frame address with an all '1's (broadcast address) register. The comparison is conditioned by bits 5-7 of this register. When bits 0-4 are cleared, address detection by the DLC is inhibited. If bit 4 is cleared to a zero and one or more of the enable bits (0-3) is set, then the all '1's pattern is ignored.

Bit 5: Address Size 1-2 (Default = 0)—At least one of the enable bits (0-4) must be set for this bit to have any effect on DLC operation.

If any of the enable bits are set and bit 5 is cleared, then the first two address bytes of each received frame will be compared.

If bit 5 is set to a '1', only one byte is compared (bit 7 specifies whether the first or second byte is compared).

Bit 6: C/R Address Enable (Default = 0)—At least one of the enable bits (0-4) must be set for this bit to have any effect on DLC operation.

If any of the enable bits are set, and the C/R address enable bit is cleared, then bit 1 of the first address byte of each received frame will be ignored.

If this bit is set, then bit 1 of the first received frame address byte must compare successfully along with the other address bits for address recognition to occur.

Bit 7: First/Second Byte Selection (Default = 0)—This bit is effective only when one-byte addressing is selected. When this bit is set, only the second byte is monitored by the address recognizers (first eight bits are don't cares). When this bit is cleared, the first byte only is examined.

DLC LINK Address Recognition Registers

The four registers are each two bytes long with the LSB having the lower address. The LSB of each pair corresponds to the second byte following the flag. The MSB corresponds to the first byte following the flag.

All of the bits in the four Link Address Recognition Registers are set and cleared by software except when initialized to '0's by a DLC reset or IDPC reset.

Each of these four registers has a corresponding enable bit in the DLC Address Control Register (bits 0-3). If the corresponding enable bit is set, then the value in the Link Address Recognition Register is conditioned by bits 5-7 of the DLC Address Control Register. Default = Hex 0000.

DLC Serial Bus Port Control Register (0A Hex)

7	6	5	4	3	2	1	0
ENABLE REMOTE LOOP BACK	ENABLE LOCAL LOOP BACK	INVERT DATA	CHAN SELECT MSB	CHAN SELECT —	CHAN SELECT —	CHAN SELECT —	CHAN SELECT LSB

All bits in the Serial Bus Port Control Register are set and cleared by software, except when initialized to default values by a DLC reset or IDPC reset. This register can be written and read by the local processor.

Bits 0-4: Channel Select—These five bits select Serial Bus Port time slots for multiplexing transmitted serial bit streams/demultiplexing received serial bit streams.

Bits 4	3	2	1	0	Selection
0	0	0	0	0	Channel 0
0	0	0	0	1	Channel 1
0	0	0	1	0	Channel 2
	...	...			
1	1	1	1	0	Channel 30
1	1	1	1	1	Non-Multiplex

In non-multiplexed mode, a single channel is available with the receiver clocked by the SCLK pin and the transmitter clocked by the SFS/XMITCLK pin. For all settings except non-multiplexed, both the transmitter and the receiver are clocked by the SCLK pin.

In multiplexed mode with channel 0 programmed, data from channel 0 and 1 can be concatenated, transmitted, and received 16 bits at a time. This is done automatically by holding SFS/XMIT active during the first bit time of time slot 1.

Bit 5: Data Invert (Default = 0)—When this bit is set to a '1', the transmitted serial bit stream is inverted.

When this bit is set, the receive data stream is inverted. If this bit is cleared to '0', no inversion takes place in either direction.

Bit 6: Local Loop back Enable (Default = 0)—This bit is set to enable loop back for diagnostic purposes. When set, the transmit data path (SBOUT) is connected internally to the receive data path (SBIN is disconnected). The selected transmit clock (either SCLK or SFS/XMITCLK) is used for both the transmit and receive clocks. Clearing this bit restores the system to normal operation.

Bit 7: Remote Loop back Enable (Default = 0)—This bit is set to enable loop back for diagnostic purposes. When set, the SBIN pin is connected directly to SBOUT. In this manner, receive data is presented to SBOUT as transmitted data. In this mode, the appropriate receive clock is SCLK. Receive data may be presented to the DLC receiver depending on the setting of the receive enable bit. However, data from the transmit section is prevented from entering SBOUT in this mode.

DLC Minimum Receive Packet Size Register (0B Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	NOT USED	NOT USED	MIN PKT SIZE MSB	MIN PKT SIZE —	MIN PKT SIZE —	MIN PKT SIZE LSB

This register specifies the Minimum Receive Packet Size Register.

Bits 0-3: Minimum Receive Packet Size (Default = Hex 5)—Bits 0-3 of this register are set and cleared by software except when initialized to a default value by a DLC reset or IDPC hardware reset. This register indicates the minimum packet length (exclusive of opening and closing flags) that can be received without generating a short frame error in the Receive Frame Status Register.

At the time that the short frame interrupt is generated, the Receive Byte Count Register reflects the number of bytes in the short frame.

Bits 3	2	1	0	Selection
0	0	0	1	1 Byte
0	0	1	0	2 Bytes
0	0	1	1	3 Bytes
	...	...		
1	1	1	1	15 Bytes
0	0	0	0	Not Used

NOTE: Although reception of packets containing only 1, 2, or 3 bytes can be programmed, a minimum of 3 bytes must be received before data are moved into the FIFO and the packet is reported.

DLC Maximum Receive Packet Size Register

LSB	7	6	5	4	3	2	1	0
MSB	15	14	13	12	11	10	9	8

This register specifies the Maximum Receive Packet Size Register.

Bits 0-15: Maximum Receive Packet Size Register (Default = Hex 0000)—Bits 0-15 of this register are set and cleared by software except when initialized to a default value by a DLC reset or IDPC hardware reset. This register indicates the maximum packet length (exclusive of opening and closing flags) that can be received without generating a long frame error in the Receive Frame Status Register. The value programmed into the register is equal to the desired packet size minus three.

As each packet byte is received, the contents of the Maximum Receive Packet Size Register are compared with the receive byte counter. A long frame error is generated in the Receive Frame Status Register if the value is exceeded. The received byte that caused the receive byte counter to exceed the maximum length is tagged as the End-of-Packet (EOP) byte, and the DLC receiver looks for the next opening flag. The LSB has the lower order address.

DLC Interrupt Source Interrupt Enable Register (0E Hex)

7	6	5	4	3	2	1	0
ENABLE RECVR LINK STATUS	ENABLE FIFO STATUS	ENABLE RECV FRAME STATUS	ENABLE VALID PACKET SENT	ENABLE VALID PACKET RECVD	NOT USED	NOT USED	NOT USED

Bits 3 and 4 provide single-level interrupt enable/disable control for valid packet received and valid packet sent status conditions. For bits 5-7, the Interrupt Source Interrupt Enable Register contains the first-level enable of a two-level interrupt enable structure. Bits 5-7 enable three corresponding Interrupt Enable Registers:

- Receive Frame Interrupt Enable Register
- Receive Link Interrupt Enable Register
- FIFO Status Interrupt Enable Register

The valid packet received and valid packet sent interrupts have a single-level interrupt enable structure (bits 3 and 4 of the Interrupt Source Interrupt Enable Register).

When an event occurs that causes a bit to be set in one of the three status registers (Receive Frame, Receive Link, and FIFO Status Registers), and both levels of status interrupt enable are set to a '1', the DLC interrupt is generated and the bit corresponding to that register is set in the DLC Interrupt Source Register. Unless both levels of interrupt enable are set, no interrupt is generated.

Bits 0-2—Unassigned

Bit 3: Enable Valid Packet Received Interrupt (Default = 0)—If this bit is set and the valid packet received bit is set in the Interrupt Source Register, a DLC interrupt is generated. If this bit is cleared, setting of the valid packet received bit in the Interrupt Source Register does not result in the generation of an interrupt.

Bit 4: Enable Valid Packet Sent Interrupt (Default = 0)—If this bit is set and the valid packet sent bit is set in the Interrupt Source Register, a DLC interrupt is generated. If this bit is cleared, setting of the valid packet sent bit in the Interrupt Source Register does not result in the generation of an interrupt.

Bit 5: Enable Receive Frame Status Interrupt (Default = 0)—This bit is set as the first level of enable for the Receive Frame Interrupt Enable Register. If a status bit is set in the Receive Frame Status Register, and the corresponding bit is set in the Receive Frame Interrupt Enable Register, and bit 5 of this register is set, an interrupt is generated and bit 5 of the Interrupt Source Register is set to indicate the interrupt originated in the Receive Frame Status Register.

Bit 6: Enable FIFO Status Interrupt (Default = 0)—This bit is set as the first level of enable for the FIFO Status Interrupt Enable Register. If a status bit is set in the FIFO Status Register, and the corresponding bit is set in the FIFO Status Interrupt Enable Register, and bit 6 of this register is set, an interrupt is generated and bit 6 of the Interrupt Source Register is set to indicate the interrupt originated in the FIFO Status Register.

Bit 7: Enable Receive Link Status (Default = 0)—This bit is set as the first level of enable for the Receive Link Status Enable Register. If a status bit is set in the Receive Link Status Register, and the corresponding bit is set in the Receive Link Status Interrupt Enable Register, and bit 7 of this register is set, an interrupt is generated and bit 7 of the Interrupt Source Register is set to indicate the interrupt originated in the Receive Link Status Register.

DLC Receive Frame Interrupt Enable Register (0F Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	ENABLE OVERRUN ERROR	ENABLE LONG FRAME ERROR	ENABLE SHORT FRAME ERROR	ENABLE CRC ERROR	ENABLE NON-INT # BYTES ERROR	ENABLE ABORT RECVD

The Receive Frame Interrupt Enable Register contains a bit-for-bit image of the Receive Frame Status Register. If a status bit is set in the Receive Frame Status Register corresponding to a set bit in the Receive Frame Interrupt Enable Register, and bit 5 of the first-level enable register (Interrupt Source Interrupt Enable Register) is set, a DLC interrupt is generated and bit 5 of the Interrupt Source Register is set indicating the interrupt originated in the Receive Frame Status Register.

Bit 0: Enable Abort Received Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 5) is set, setting this bit enables a DLC interrupt if the abort received bit (bit 0) is set in the Receive Frame Status Register.

Bit 1: Enable Non-Integer Number Bytes Received Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 5) is set, setting this bit enables a DLC interrupt if the non-integer number bytes received bit (bit 1) is set in the Receive Frame Status Register.

Bit 2: Enable CRC Error Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 5) is set, setting this bit enables a DLC interrupt if the CRC error bit (bit 2) is set in the Receive Frame Status Register.

Bit 3: Enable Short Frame Error Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 5) is set, setting this bit enables a DLC interrupt if the short frame error bit (bit 3) is set in the Receive Frame Status Register.

Bit 4: Enable Long Frame Error Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 5) is set, setting this bit enables a DLC interrupt if the long frame error bit (bit 4) is set in the Receive Frame Status Register.

Bit 5: Enable Overrun Error Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 5) is set, setting this bit enables a DLC interrupt if the overrun error bit (bit 5) is set in the Receive Frame Status Register.

Bits 6-7—Not Used

DLC Receive Link Interrupt Enable Register (10 Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	ENABLE IN-FRAME ERROR	ENABLE FLAG IDLE RECVD	ENABLE MARK IDLE RECVD

This register is used to enable/disable interrupts from the Receive Link Status Register (Default = 0).

Bit 0: Enable Change In Mark Idle Received Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 7) is set, setting this bit enables a DLC interrupt if the change in mark idle received bit (bit 0) is set in the Receive Link Status Register.

Bit 1: Enable Change In Flag Idle Received Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 7) is set, setting this bit enables a DLC interrupt if the change in flag idle received bit (bit 1) is set in the Receive Link Status Register.

Bit 2: Enable Change In In-Frame Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 7) is set, setting this bit enables a DLC interrupt if the change in in-frame bit (bit 2) is set in the Receive Link Status Register.

Bits 3-7—Not Used

DLC FIFO Status Interrupt Enable Register (11 Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	ENABLE EOP RECV FIFO	ENABLE XMIT UNDRUN REACHD	ENABLE XMIT BUFFER AVAIL	ENABLE XMIT TRSHLD REACHD	ENABLE RECV DATA AVAIL	ENABLE RECV TRSHLD REACHD

This register is used to enable/disable interrupts from the FIFO Status Register (Default = 0).

Bit 0: Enable Receive Threshold Reached Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 6) is set, setting this bit enables a DLC interrupt if the receive threshold reached bit (bit 0) is set in the FIFO Status Register.

Bit 1: Enable Receive FIFO Data Available Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 6) is set, setting this bit enables a DLC interrupt if the receive FIFO data available bit (bit 1) is set in the FIFO Status Register.

Bit 2: Enable Transmit Threshold Reached Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 6) is set, setting this bit enables a DLC interrupt if the transmit threshold reached bit (bit 2) is set in the FIFO Status Register.

Bit 3: Enable Transmit Buffer Available Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 6) is set, setting this bit enables a DLC interrupt if the transmit buffer available bit (bit 3) is set in the FIFO Status Register.

Bit 4: Enable Transmit Underrun Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 6) is set, setting this bit enables a DLC interrupt if the transmit underrun bit (bit 4) is set in the FIFO Status Register.

Bit 5: Enable EOP in Receive FIFO Interrupt (Default = 0)—If the first level of interrupt (Interrupt Source Interrupt Enable Register, bit 6) is set, setting this bit enables a DLC interrupt if the EOP in receive FIFO bit (bit 5) is set in the FIFO Status Register.

Bits 6-7—Not Used

DLC Transmit Byte Count Register

LSB	7	6	5	4	3	2	1	0
MSB	15	14	13	12	11	10	9	8

This register is used to specify the length of the packet to be transmitted.

Bits 0-15: Transmit Packet Size (Default = 0)—Bits 0-15 of this register are set and cleared by software except when initialized to a default value by a DLC reset or IDPC hardware reset. This register is written by software when the number of bytes to be transmitted is different from the current value stored in the Transmit Byte Count Register (exclusive of opening and closing flags and FCS bytes).

The contents of this register are written to the transmit byte counter whenever software writes the least significant byte of this register pair (if the transmitter is out of frame), or when an end-of-packet-tagged byte is loaded from the transmit FIFO into the Parallel-to-Serial Shift Register. If software is writing to this register when the EOP-tagged byte is loaded, the transfer to the transmit byte counter is delayed until the software write is complete. The loading of the transmit byte counter takes place when the LSB is written; i.e., write the MSB first. The LSB has the lower address. A transmit FIFO underrun error clears this register.

Transmit Byte Count Decode:

Bits																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Value Selected
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1 Byte
								...								
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	65.535K Bytes
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Not Assigned

DLC FIFO Threshold Register (14 Hex)

7	6	5	4	3	2	1	0
RECV TRSHLD MSB	RECV TRSHLD —	RECV TRSHLD —	RECV TRSHLD LSB	XMIT TRSHLD —	XMIT TRSHLD MSB	XMIT TRSHLD —	XMIT TRSHLD LSB

This register is used to specify the transmit and receive FIFO threshold levels.

Bits 0-3: Transmit Threshold Value (Default = Hex 8)—The contents of this register are set and reset under software control except when initialized by DLC reset or IDPC reset or when an abort is issued.

Bits				
3	2	1	0	Value Selected
0	0	0	1	1 Byte
0	0	1	0	2 Bytes
0	0	1	1	3 Bytes
	...	...		
1	1	1	1	15 Bytes
0	0	0	0	16 Bytes

Bits 4-7: Receive FIFO Threshold (Default = Hex 8)—The receive FIFO threshold counts by two since the receive FIFO buffer is 32 bytes deep.

Bits				
7	6	5	4	Value Selected
0	0	0	1	2 Bytes
0	0	1	0	4 Bytes
0	0	1	1	6 Bytes
	...	...		
1	1	1	1	30 Bytes
0	0	0	0	32 Bytes

DLC Interrupt Source Register (15 Hex)

7	6	5	4	3	2	1	0
RECV LINK STATUS	FIFO STATUS —	RECV FRAME STATUS	VALID PACKET SENT	VALID PACKET RECVD	RECV ADDR MSB	RECV ADDR —	RECV ADDR LSB

This register is used to identify the source of interrupting conditions and to report valid-packet-transmitted, valid-packet-received, and valid-packet address conditions.

Bits 0-2: Receive Link Address Field (Default = 110 (0 = LSB))—The receive link address field is written by hardware whenever a packet is received (with or without errors). This field is a delayed-stacked field.

The link address for up to four received packets can be stored at any given time. The link address field for any packet is not presented to the user until the last byte of that packet is read from the FIFO.

Bits 2	1	0	Definition
0	0	0	Contents of Link Address 0 Recognized
0	0	1	Contents of Link Address 1 Recognized
0	1	0	Contents of Link Address 2 Recognized
0	1	1	Contents of Link Address 3 Recognized
1	0	0	Broadcast Link Address (All '1's) Recognized
1	0	1	Not Used
1	1	0	Default Value—No Packet Received
1	1	1	Packet Received with no Address Recognized enabled (Bits 0-4 of DLC Address Control Register cleared to "0s")

Bit 3: Valid Packet Received (Default = 0)—This bit is reset to its default value when DLC reset is executed or an IDPC reset is received. This bit is set to a '1' when the End-of-Packet-tagged byte is read from the receive FIFO buffer and no receive error has been detected for that packet. This bit is cleared when software reads this register, or a DLC reset or IDPC reset occurs.

Bit 4: Valid Packet Sent (Default = 0)—This bit is set to a '1' when the last bit before the closing flag has been transmitted by the DLC transmitter (transmit byte counter = 0 and no underrun and transmitter out of frame). This bit is cleared when the transmitter goes in-frame, this register is read, a DLC reset is executed, or an IDPC reset occurs.

Bit 5: Receive Frame Status (Default = 0)—This bit is set to a '1' when any bit in the Receive Frame Status Register and both of the corresponding bits in the Receive Frame Interrupt Enable Register and enable receiver frame interrupt bit (bit 5) are set in the Interrupt Source Interrupt Enable Register.

This bit is gated when stage 3 status is actually transferred to stage 4. (See description of delayed status reporting.)

Bit 5 is cleared to 0 when the Receive Frame Status Register is read by software, a DLC reset is executed, or an IDPC reset is received from the processor.

Bit 6: FIFO Status (Default = 0)—This bit is set to a '1' when any bit in the FIFO Status Register is set and both of the corresponding bits in the Receive Frame Interrupt Enable Register and enable FIFO status interrupt (bit 6) are set to a '1' in the Interrupt Source Interrupt Enable Register.

This bit is cleared to '0' when the FIFO Status Register is read by software, a DLC reset is executed, or an IDPC reset is received from the processor.

Bit 7: Receive Link Status (Default = 0)—This bit is set to a '1' when any bit in the Receive Link Status Register is set and both of the corresponding bits in the Receive Frame Interrupt Enable Register and bit 7 (enable received link status interrupt bit) are set in the Interrupt Source Interrupt Enable Register.

This bit is cleared to '0' when the Receive Link Status Register is read by software, a DLC reset is executed, or an IDPC reset is received from the processor.

DLC Receive Byte Count Register

LSB	7	6	5	4	3	2	1	0
MSB	15	14	13	12	11	10	9	8

This register reports the length of the received packet.

Bits 0-15: Receive Byte Count Register (Default = 0)—This 16-bit register indicates the number of bytes received in a packet, not including the opening and closing flags, whether the packet was received in error or not. The actual counter is incremented each time a byte is loaded into the FIFO.

This register is a "read-only" register in respect to the local processor. This register is cleared to '0' when a DLC reset is executed or an IDPC reset is received from the processor.

This register presents information in a delayed fashion. When the last byte of a packet is read from the receive FIFO, the receive byte count is made available to the user. If a new packet is received before the status from the previous packet is read by the user, the status for the new packet is stacked up behind the previous packet. Status for up to four packets can be stacked up at any given time. When the four-deep stack is full, the DLC receiver ignores new packets until the status from at least one packet is read by the user.

There are two mechanisms that ensure synchronization between packet data and status: 1) data from one packet cannot be read from the FIFO until status from the previous packet is read; and 2) when the least-significant byte of the Receive Byte Count Register is read, all of the delayed stacked registers for that packet are cleared (Receive Byte Count Register, Receive Frame Status Register, Residual Bit Register, and the received address field of the Interrupt Source Register). For this reason, the LSB of the Receive Byte Count Register should always be read last. The LSB has the lower address.

Bit Definitions

Bits 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Value Selected
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1 Byte
								. . .								
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	65.535K Bytes
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Not Assigned

DLC Receive Frame Status Register (18 Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	OVERUN ERROR	LONG FRAME ERROR	SHORT FRAME ERROR	CRC ERROR	NON-INT # BYTES RECVD	ABORT RECVD

This is a "read-only" register with the bits being set by hardware. The setting of any bit in this register will result in the setting of bit 5 in the Interrupt Source Register if the corresponding bit is set in the Receive Frame Interrupt Enable Register and the receive frame status bit is set in the Interrupt Source Interrupt Enable Register.

This register is a delayed-stacked register. Status is not reported until the last byte of the packet is read from the FIFO. At that time maskable interrupts are generated. Status for up to four packets can be stacked at any given time.

The bits of this register are cleared to '0' (default setting) when a DLC reset is executed, the IDPC reset pin is activated, or when the register or the LSB of the Receive Byte Count Register is read.

It is possible that more than one receive error may occur simultaneously on the same receive bit. However, only one bit in this register may be set to a '1' at any time. The following table indicates the precedence of the various errors and exception conditions flagged by this register (listed in descending order of precedence):

Bit	Name
0	Abort Received
5	Overrun
3	Short Frame
4	Long Frame
2	CRC Error
1	Non-Integer Number of Bytes

If the Receive Frame Status Register is not read (not normally read for a valid packet) before the LSB of the Receive Byte Count Register, reading the Receive Byte Count Register will clear the Receive Frame Status Register to keep the register in sync (i.e., read Receive Byte Count Register LSB last).

Bit 0: Abort Received (Default = 0)—This bit is set to a '1' as a result of the DLC receiver abort detector detecting an abort character (seven '1's while in-frame) while the DLC receiver is in-frame and at least three bytes have been received.

Bit 1: Non-Integer Number Bytes Received (Default = 0)—This bit is set to a '1' as a result of the DLC receiver flag detector recognizing a closing flag character with at least three bytes received when a non-integer number of bytes has been received in a non-short frame (i.e., at least one but less than eight bits were received after zero bit deletion in the byte immediately preceding the closing flag).

Bit 2: CRC Error (Default = 0)—This bit is set to a '1' as a result of the DLC CRC checker detecting an error when CRC check is enabled in the DLC Command/Control Register.

Bit 3: Short Frame Error (Default = 0)—This bit is set to a '1' as a result of the DLC receiver detecting a short frame error.

Bit 4: Long Frame Error (Default = 0)—This bit is set to a '1' as a result of the DLC receiver detecting a long frame error.

Bit 5: Overrun Error (Default = 0)—This bit is set to a '1' as a result of the DLC receive FIFO detecting an overrun condition (i.e., the receive FIFO contains 32 bytes when receive data needs to be moved into the FIFO from the Parallel-to-Serial Shift Register).

Bits 6-7—Unused

DLC Receive Link Status Register (19 Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	INFRAME RECVD	FLAG IDLE RECVD	MARK IDLE RECVD	CHANGE IN INFRAME	CHANGE IN FLAG IDLE	CHANGE IN MARK IDLE

The Receive Link Status Register reflects the status of the link at the receiver input. Three conditions are monitored: mark idle, flag idle, and in-frame. Bits 5-3 reflect the current status of the link and do not generate interrupts. Bits 2-0 reflect changes in the link since the register was last read; maskable interrupts are associated with these bits. At reset, bits 2-0 are cleared directly and bits 5-3 are cleared defacto by the hardware that sets them.

Bit 0: Change in Mark Idle (Default = 0)—This bit, when set, indicates that the mark idle bit (bit 3) has changed (either set or cleared) since the last time that the register was read. This bit is cleared by reading the register, a DLC reset, or an IDPC reset.

Bit 1: Change In Flag Idle (Default = 0)—This bit, when set, indicates that the flag idle bit (bit 4) has changed (either set or cleared) since the last time that the register was read. This bit is cleared by reading the register, a DLC reset, or an IDPC reset.

Bit 2: Change In In-Frame (Default = 0)—This bit, when set, indicates that the in-frame bit (bit 5) has changed (either set or cleared) since the last time that the register was read. This bit is cleared by reading the register, a DLC reset, or an IDPC reset.

Bit 3: Mark Idle Received (Default = 0)—This bit, when set, indicates that mark idle is currently being received. When cleared, mark idle is not being received.

Bit 4: Flag Idle Received (Default = 0)—This bit, when set, indicates that flag idle is currently being received. When cleared, flag idle is not being received.

Bit 5: In-Frame Received (Default = 0)—This bit, when set, indicates that in-frame is currently being received. When cleared, the receiver is not in-frame.

Bits 6-7—Unused

DLC FIFO Status Register (1A Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	EOP IN RECV FIFO	XMIT UNDRUN	XMIT BUFFER AVAIL	XMIT TRSHLD REACHD	RECV DATA AVAIL	RECV TRSHLD REACHD

Each of the bits of the FIFO Status Register is set and cleared by DLC hardware to indicate the real-time state of the various status conditions that it represents. Bits 6-7 are not assigned.

Upon completion of DLC reset or IDPC reset external input, the bits of this register will be set/cleared to their default values.

There is a FIFO Status Interrupt Enable Register that is a bit-for-bit image of this register. Setting any bit in this register will set bit 6 of the Interrupt Source Register if the corresponding enable bit is set in the FIFO Status Interrupt Enable Register and the enable FIFO status interrupt bit 6 is set in the Interrupt Source Interrupt Enable Register.

Bit 0: Receive Threshold Reached (Default = 0)—This bit is set to a '1' when the number of bytes in the DLC receive FIFO increments to a value equal to or greater than the value in the receive FIFO threshold bit field of the DLC FIFO Threshold Register.

This bit is cleared to '0' when the count of bytes in the receive FIFO byte counter decrements to a value less than the receive threshold value stored in the DLC FIFO Threshold Register.

This status bit is used to condition the DLC receive DMA data request signal.

Bit 1: Receive FIFO Data Available (Default = 1)—This bit is set to a '1' whenever there is a byte available to be read for the DLC Receive FIFO Data Register.

This bit is cleared to a '0' when a byte is read and the receive FIFO is empty. Receive FIFO data available is disabled (cleared) when the last byte of a packet is read from the FIFO. It is not re-enabled until the user reads the LSB of the Receive Byte Count Register. This, in conjunction with the packet received interrupt, provides the non-DMA user with an indication of when the last byte of the packet has been read.

Bit 2: Transmit Threshold Reached (Default = 0)—This bit is set to a '1' when the number of bytes in the DLC transmit FIFO is less than or equal to the count in the transmit FIFO threshold bit field (bits 0-3 of the FIFO Threshold Register).

This bit is cleared to a '0' when the count of bytes in the transmit FIFO increments to a value greater than the transmit FIFO threshold bit field value. This status bit is used to condition the DLC transmit DMA data request signal.

Bit 3: Transmit FIFO Buffer Available (Default = 0)—This bit is set to a '1' whenever the DLC FIFO Data Register is empty, and the transmit byte counter is not equal to zero (i.e., available to be written into). On a write, this bit remains active if the FIFO buffer is not full. This bit is cleared when the last byte of a packet is in the FIFO. This prevents multiple packets from existing in the FIFO at the same time (non-DMA users).

Bit 4: Transmit Underrun (Default = 0)—This bit is set to a '1' if the output location of the transmit FIFO buffer (opposite end of the FIFO from the FIFO Data Register) is empty when a transmitter serial-to-parallel load is attempted. The transmit byte counter is implicitly non-zero for this load to be attempted. This bit is cleared when the FIFO Status Register is read.

An abort is automatically transmitted in response to an underrun.

Bit 5: EOP in Receive FIFO (Default = 0)—This bit, when set to a '1', indicates that the last byte of a packet has been loaded into the receive FIFO. The bit remains set until no EOP tags remain in the FIFO. This is the packet received indication, and is normally used only for non-DMA applications to indicate that the FIFO should be serviced.

Bits 6-7—Not Used

DLC FIFO Data Registers

The Receive FIFO and Transmit FIFO Data Registers are each eight bits in length.

The Receive FIFO Data Register is read by DMA or software to remove one byte at a time from the receive FIFO. If read by software, the user should first poll the receive FIFO data available status bit (bit 1 in the FIFO Status Register), unless data is being read in response to a threshold reached indication in which the number of bytes to be read is known.

The Transmit FIFO Data Register is written by DMA or software to load one byte to the transmit FIFO. If written by software, the user should first poll the transmit FIFO buffer available status bit (bit 3 in the FIFO Status Register) to ensure that a byte slot is available in the FIFO (unless the field is being loaded in response to a threshold reached indication, in which case the number of bytes that can be loaded is known).

DLC Residual Bit Status/Control Register (1D Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	XMIT RESIDUE COUNT MSB	XMIT RESIDUE COUNT —	RECVD RESIDUE COUNT LSB	RECVD RESIDUE COUNT MSB	RECVD RESIDUE COUNT —	RECVD RESIDUE COUNT LSB

Bit residue is the number of bits remaining after the information field is divided into 8-bit bytes. Since microprocessors handle data on 8-bit boundaries, data is moved to and from the IDPC FIFOs and external RAM in 8-bit quantities. Most data communication protocols, however, contain characters 5 to 8 bits in length (extra bit positions contain garbage). In order to use the bandwidth of the data communication network, protocols such as X.25 allow the characters to be stripped of unnecessary bits and concatenated into a "packed" bit stream for transmission.

At the receiving end, the bit stream is unpacked and the characters are once more stored as portions of bytes. The packed information field is no longer aligned on 8-bit boundaries. Since the transmitted and received data are no longer stored in 8-bit chunks, the end of the information field may not end in 8 bits. The leftover bits are referred to as residue bits; however, they represent valid data. The Residual Bit Status/Control Register contains two count fields (receive and transmit) which are used to report the number of residue bits prior to the sending/receiving of the closing flag.

Bits 0-2: Received Bit Residue Count (Default = 000)—These three bits form a "read-only" field displaying the number of residue bits received. This field is cleared to '0's upon reset or by a read of the register or a read of the LSB of the receive byte counter. This field is a delayed-stacked field. Up to four packets may be stacked at any one time.

Bits 2	1	0	Indicated Value
0	0	0	8 Bits
0	0	1	1 Bit
0	1	0	2 Bits
1	1	1	7 Bits

Bits 3-5: Transmitter Bit Residue Count (Default = 000)—These three bits allow the user to specify the number of residue bits to be transmitted in the last byte of the packet (data is loaded into the transmit FIFO in byte quantities). This is a read/write field that is cleared under software control.
Bits 6-7 not used.

Bits 5	4	3	Indicated Value
0	0	0	8 Bits
0	0	1	1 Bit
0	1	0	2 Bits
1	1	1	7 Bits

Detailed Description of User-Visible USART Registers

The USART contains 14 registers, as shown in Table 3.

Table 3. USART Registers

Offset (Hex)	Register Name	Size (Bytes)	Type
20	Receive FIFO Data Register (DLAB = 0)*	1	Read Only
	Transmit FIFO Data Register (DLAB = 0)	1	Write Only
	Baud Rate Divisor LSB Register (DLAB = 1)	1	Read/Write
21	Interrupt Enable Register (DLAB = 0)	1	Read/Write
	Baud Rate Divisor MSB Register (DLAB = 1)	1	Read/Write
22	Interrupt Identification Register	1	Read Only
23	Line Control Register	1	Read/Write
24	Modem Control Register	1	Read/Write
25	Line Status Register	1	Read Only
26	Modem Status Register	1	Read Only
27	Control Register	1	Read/Write
28	Status Register	1	Read/Write
29	Special Character Bit-Map Address Pointer Register	1	Read/Write
2A	Special Character Bit Map Command Register	1	Read/Write
2B-3E	Reserved	6	—

*Divisor Latch Access Bit (DLAB) in the Line Control Register.

USART Receive FIFO Data Register (Default = 0)

MSB	7	6	5	4	3	2	1	0	LSB

The Receive FIFO Data Register is a "read-only" register output side of the receive FIFO. Data received by the USART are read from the FIFO by the CPU at this address.

USART Transmit FIFO Data Register (Default = 0)

MSB	7	6	5	4	3	2	1	0	LSB

The Transmit FIFO Register is a "write-only" input to the transmit FIFO. Data placed in this 8-bit register are transmitted out of the FIFO LSB first.

USART Baud Rate Divisor LSB Register (Default = 0)

MSB	7	6	5	4	3	2	1	0	LSB

The Baud Rate Divisor LSB Register is an 8-bit register used to hold the low-order bits of the number by which the USART clock input (USARTCLK) is to be divided. Bit 0 is the LSB and bit 7 is the MSB.

USART Baud Rate Divisor MSB Register (Default = 0)

MSB	7	6	5	4	3	2	1	0	LSB

The Baud Rate Divisor MSB Register is an 8-bit register used to hold the high-order bits of the number by which the USART clock input (USARTCLK) is to be divided. Bit 0 is the LSB and bit 7 is the MSB.

NOTE: When reset, the register pair is cleared to all zeros, but the baud rate generator will actually divide by 64 until programmed.

Bit Definitions																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Divide By:
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	One
								...								
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	65,535
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reserved

NOTE: Divide-by-one passes through the USARTCLK unaffected. This allows the receiver and transmitter to operate from separate clocks in synchronous mode. A write to either the MSB or LSB divisor causes the baud rate generator to be loaded with a 16-bit value.

USART Interrupt Enable Register (21 Hex, DLAB = 0)

7	6	5	4	3	2	1	0
NOT USED	XMIT LINE STATUS: SHFTREG EMPTY	USART STATUS: SPCL CHAR	USART STATUS: RECV FIFO TIMEOUT	MODEM STATUS: CTS,DSR	RECV LINE STATUS	XMIT FIFO TRSHLD	RECV FIFO TRSHLD

The Interrupt Enable Register is an 8-bit read/write register used to enable specific interrupt sources (Default = 0). Setting a specific bit enables its corresponding interrupt. Clearing a bit disables the interrupt and resets the interrupt pin if the corresponding condition is present.

USART Interrupt Identification Register (22 Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	NOT USED	NOT USED	INTER SOURCE MSB	INTER SOURCE —	INTER SOURCE LSB	INTER PEND

The Interrupt Identification Register is an 8-bit read-only register used to identify which status register contains an interrupt condition. Unused bit positions (bits 4-7) return '0's when this register is read.

Bit 0: Interrupt Pending (Default = 1)—This bit is cleared to a '0' if any interrupt is pending.

Bits 1-3: Interrupt Source (Default = 000)—This 3-bit field identifies the highest priority source of all existing interrupts.

Interrupt Source Decode

Bits 3	2	1	Priority	Source	Reset By*
0	0	0	4th	CTS or DSR	Reading the Modem Status Register
0	0	1	3rd	Transmit FIFO Threshold Reached	Reading this Register and Interrupt Source = 001
0	1	0	2nd	Receive FIFO Threshold Reached	Reading this Register and Interrupt Source = 010
0	1	1	1st**	Overrun, Parity, Special Character Received, Framing, or Break	Reading Line Status Register
1	0	0	5th	Receive FIFO Timeout	Reading USART Status Register
1	0	1	6th	Transmit Shift Register Empty	Reading this Register and Interrupt Source = 101

*All bits are reset by a USART reset or an IDPC reset.

**Simultaneous receipt of a special character or a character with a parity error, and a threshold reached condition, causes the interrupt request to be generated for the special character or parity error prior to the generation of the threshold reached interrupt.

Bits 4-7—Not Used (cleared to '0's)

USART Line Control Register (23 Hex)

7	6	5	4	3	2	1	0
DIV LATCH ACCESS BIT	BREAK	STICK PARITY	EVEN/ ODD PARITY	ENABLE PARITY	NUMBER STOP BITS	CHAR LENGTH MSB	CHAR LENGTH LSB

Bit 0-1: Character Length—Bits 0 and 1 define the character length:

Bits 1	0	Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

Bit 2: Number of Stop Bits—This bit selects the number of stop bits used in serial data transfers:
0 = 1 Stop Bit
1 = 1.5 Stop Bits (5-bit characters) OR 2 Stop Bits (6-, 7-, or 8-bit characters)

Bit 3: Parity Enable—When this bit is set to a '1', parity generation and checking is enabled. When the bit is cleared, parity generation and checking is disabled.

Bit 4: Even Parity Set—This bit is set to select even parity. The bit is cleared to select odd parity.

Bit 5: Stick Parity—If parity is enabled (bit 3 set) and this bit is set, parity is expected to be received opposite to that indicated by bit 4. Parity is transmitted with a value opposite that of bit 4.

Bit 6: Break—This bit is set to request that a break condition be transmitted. The USART will transmit the break pattern immediately after completing any character transmission in progress when this bit is set. The Shift Register and transmit FIFO contents are discarded. The line returns to normal operation when the bit is cleared. Breaks are transmitted only in asynchronous mode.

Bit 7: Divisor Latch Access Bit—This bit is set to access the Baud Rate Divisor Registers and is cleared to access the Receive and Transmit FIFO Data Registers and the Interrupt Enable Register.

USART Modem Control Register (24 Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	NOT USED	LOCAL LOOPBK	RESRVD	RESRVD	$\overline{RTS}$	$\overline{DTR}$

This register specifies modem control parameters (Default = 0).

Bit 0: $\overline{DTR}$—When this bit is set, $\overline{DTR}$ goes active-LOW.

Bit 1: $\overline{RTS}$—When this bit is set to a '1', $\overline{RTS}$ goes active-LOW.

Bits 2-3—Reserved

Bit 4: Local Loop back—Setting this bit to a '1' places the USART in a local loop back condition for diagnostic purposes.

Bits 5-7—Not Used

USART Line Status Register (25 Hex)

7	6	5	4	3	2	1	0
SPCHL CHAR IN FIFO	XMIT SHIFT REG EMPTY	XMIT TRSHLD REACHD	BREAK DETECT	FRAMNG ERROR	PARITY ERROR IN FIFO	RECV BUFFER OVERUN	RECV DATA AVAIL

The Line Status Register contains flag bits that are set to indicate the presence of a condition that can generate an interrupt if the appropriate interrupt enable bits are set in the Interrupt Enable Register. Bits 1 through 4 and 7 are cleared by reading the Line Status Register. Bit 5 is cleared when the condition goes away, but the interrupt is cleared by reading the Interrupt Identification Register (when the Interrupt Identification Register is reporting this interrupt). Bits 0 and 6 are cleared when the associated conditions are no longer present.

Bit 0: Receive Data Available (Default = 0)—This bit is set to a '1' when receive data is available in the Receive FIFO Data Register.

Bit 1: Receive Buffer Overrun Error (Default = 0)—This bit is set to a '1' when an overrun error results in lost receive data.

Bit 2: Character With Parity Error Loaded Into FIFO (Default = 0)—This bit is set when a parity error is detected and the character is loaded into the FIFO.

Bit 3: Framing Error (Default = 0)—This bit is set to a '1' when an invalid stop bit is detected. A character with a framing error is not loaded into the FIFO.

Bit 4: Break Condition Detected (Default = 0)—This bit is set to a '1' when a break condition is detected.

Bit 5: Transmit FIFO Threshold Reached (Default = 1)—This bit is cleared when the number of bytes in the transmit FIFO rises above the programmed threshold. The bit is reset to a '1' when the FIFO level falls to the threshold.

Bit 6: Transmit FIFO Shift Register Empty (Default = 1)—This bit is set to a '1' when the Transmit Shift Register is empty (last character transmitted) and cleared when the Transmit Shift Register and FIFO are no longer empty.

Bit 7: Special Character Loaded Into Receive FIFO (Default = 0)—This bit is set when a special character is loaded into the receive FIFO and cleared when the Line Status Register is read.

USART Modem Status Register (26 Hex)

7	6	5	4	3	2	1	0
RESRVD	RESRVD	$\overline{DSR}$ STATUS	$\overline{CTS}$ STATUS	RESRVD	RESRVD	CHANGE IN $\overline{DSR}$	CHANGE IN $\overline{CTS}$

The 8-bit Modem Status Register is used to indicate the condition of the link handshake input signals and any change in their status. Bits 0 and 1 default to '0' on reset; bits 4 and 5 reflect the input status.

Bit 0: Change in $\overline{CTS}$ (Default = 0)—This bit is set if the $\overline{CTS}$ line has changed since this register was last read.

Bit 1: Change in $\overline{DSR}$ (Default = 0)—This bit is set to a '1' when a change in $\overline{DSR}$ has occurred since this register was last read.

Bits 2-3—Reserved

Bit 4: $\overline{CTS}$ Line Status—This bit is set to a '1' if $\overline{CTS}$ is active-LOW and cleared to a '0' if $\overline{CTS}$ is inactive.

Bit 5: $\overline{DSR}$ Line Status—This bit is set to a '1' if $\overline{DSR}$ is active-LOW and cleared to a '0' if $\overline{DSR}$ is inactive.

Bits 6-7—Reserved

USART Control Register (27 Hex)

7	6	5	4	3	2	1	0
RESET	XMIT FIFO TRSHLD MSB	XMIT FIFO TRSHLD LSB	RECV FIFO TRSHLD MSB	RECV FIFO TRSHLD LSB	SYNC/ ASYNC SELECT	XMIT CLK SOURCE	RECV CLK SOURCE

The 8-bit USART Control Register is used to control all non-8250-UART functions. Additionally, this register contains the USART software reset bit.

Bit 0: Receive Clock Source (Default = 0)—This bit is set to a '1' to select the internal baud rate generator. The bit is cleared to '0' to select the external clock (RxCLK).

Bit 1: Transmit Clock Source (Default = 0)—This bit is set to a '1' to select the internal baud rate generator. The bit is cleared to '0' to select the external clock (RxCLK).

Bit 2: Sync Select (Default = 0)—This bit is set to a '1' to select synchronous mode and cleared to a '0' to select asynchronous mode.

Bits 3-4: Receive FIFO Threshold (Default = 11)—These two bits are used to select the receive FIFO threshold. When the number of bytes in the FIFO is greater than or equal to this value, receive FIFO threshold reached status is generated.

Bit 4	Bit 3	Value
0	1	1 Byte
1	0	2 Bytes
1	1	3 Bytes
0	0	4 Bytes

Bits 5-6: Transmit FIFO Threshold (Default = 00)—This field is used to hold a 2-bit count that reflects the transmit FIFO threshold. When the number of bytes remaining in the transmit FIFO is less than or equal to this level, transmit FIFO threshold reached status is reported.

Bit 6	Bit 5	Value
0	0	0 Bytes
0	1	1 Byte
1	0	2 Bytes
1	1	3 Bytes

Bit 7: Reset (Default = 0)—This bit is set by software to initiate a USART reset operation (identical to a reset initiated by hardware via the $\overline{RST}$ pin, except only the USART is affected).

USART Status Register (28 Hex)

7	6	5	4	3	2	1	0
RECVR ENABLE	NOT USED	NOT USED	XMIT BUFFER AVAIL	RECV FIFO TRSHLD REACHD	SPCHL CHAR AVAIL	CHAR W/ PARITY ERROR AVAIL	RECV FIFO TIME-OUT

The USART Status Register reports status conditions that do not occur in an 8250 UART. This register also contains the "character with parity error available" status bit. The default = 00010000. Bits 1-4 are cleared when the corresponding condition no longer exists.

Bit 0: Receive FIFO Timeout Has Occurred (Default = 0)—This bit is set to a '1' when a receive FIFO timeout has occurred. The bit is cleared when this register is read. The timeout occurs when the level in the receive FIFO is below the threshold and no characters are received in at least 2048 receiver clocks.

Bit 1: Character with Parity Error Available (Default = 0)—This bit is set when a character with a parity error reaches the output of the receive FIFO. This bit is cleared when the character is read from the FIFO.

Bit 2: Special Character Available (Default = 0)—This bit is set when a special character reaches the output of the receive FIFO. This bit is cleared when the character is read from the FIFO.

Bit 3: Receive FIFO Threshold (Default = 0)—This bit is set to a '1' when the level of the receive FIFO reaches the selected receive FIFO threshold. The bit is cleared when the number of bytes in the receive FIFO falls below the threshold value.

Bit 4: Transmit Buffer Available (Default = 1)—This bit is set whenever the FIFO Data Register is empty, and is cleared when the FIFO is full.

Bits 5-6—Not Used

Bit 7: Receiver Enable/Disable (Default = 0)—This bit is set to enable the USART receiver, and is cleared to disable the receiver.

USART Special Character Bit-Map Address Pointer Register (29 Hex)

7	6	5	4	3	2	1	0
NOT USED	SPCHL CHAR BIT 6	SPCHL CHAR BIT 5	SPCHL CHAR BIT 4	SPCHL CHAR BIT 3	SPCHL CHAR BIT 2	SPCHL CHAR BIT 1	SPCHL CHAR BIT 0

This register is used to set a pointer into the 128-bit special character bit map (Default = 0).

The character field is the address pointer into the bit map. A character is designated as a special character by first writing the address (which is the character itself) into bits 0-7 of the Special Character Bit-Map Address Pointer Register, and then by setting bit 0 of the Special Character Bit-Map Command Register. Once designated, a special character can be returned to normal status by clearing bit 0 of the Special Character Bit-Map Command Register (after the pointer is set).

NOTE: When the receiver enable bit is set (bit 7 of USART Status Register), reading the Special Character Command Register returns all '1's regardless of the actual state of the special character addressed. This is done to prevent simultaneous read/writes between the MPI and the internal logic.

A special character can be read or written to *only* when the receiver enable bit (USART Status Register bit 7) is cleared.

USART Special Character Bit-Map Command Register (2A Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	SET/ CLEAR BIT MAP

This register sets and clears the bit pointed to by the Special Character Bit-Map Address Pointer Register (Default = 0).

The register that designates a special character is set using the special characer bit-map pointer. When this register is read by the user, the state of the bit in the bit map (pointed to by the Special Character Bit-Map Pointer Register) is returned in bit location 0.

NOTE: All special characters are cleared on reset.
Detailed Description of User-Visible DPMC Register
The DPMC contains one user-visible register (the Semaphore Register) used to control inter-processor communications.

Table 4. DPMC Registers

Offset (Hex)	Register Name	Size (Bytes)	Type
3F	Semaphore Register	1	Read/Write

DPMC Semaphore Register (3F Hex)

7	6	5	4	3	2	1	0
NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	INTRPT TO LOCAL PROC	INTRPT TO HOST PROC

The Semaphore Register controls interrupt requests between the host processor and the local processor in a multi-processor application. These interrupts coordinate processor-to-processor communication via shared memory. This register is cleared to '0's by a hardware reset.

Bit 0: Interrupt to Host Processor (Default = 0)—This bit is set to a '1' by the local processor to initiate communications with the host processor. When set, the HINTOUT pin goes active-HIGH. The bit is cleared by the HINTACK pin (from the host) going HIGH (pulse). This bit can be read by the local processor.

Bit 1: Interrupt to Local Processor (Default = 0)—This bit is set to a '1' when the HINTIN pin from the host processor goes active (pulse). When this bit is set, the LINTOUT pin goes active-HIGH. This bit is cleared by the local processor by writing a '0' to it. LINTOUT goes inactive when this bit is cleared.

Bits 2-7—Not Used

INDEX

Notes

Sales Offices

North American

ALABAMA	(205)	882-9122
ARIZONA	(602)	242-4400
CALIFORNIA,		
Culver City	(213)	645-1524
Newport Beach	(714)	752-6262
San Diego	(619)	560-7030
San Jose	(408)	452-0500
Woodland Hills	(818)	992-4155
CANADA, Ontario,		
Kanata	(613)	592-0060
Willowdale	(416)	224-5193
COLORADO	(303)	741-2900
CONNECTICUT	(203)	264-7800
FLORIDA,		
Clearwater	(813)	530-9971
Ft Lauderdale	(305)	776-2001
Melbourne	(305)	729-0496
Orlando	(305)	859-0831
GEORGIA	(404)	449-7920
ILLINOIS,		
Chicago	(312)	773-4422
Naperville	(312)	505-9517
INDIANA	(317)	244-7207
KANSAS	(913)	451-3115
MARYLAND	(301)	796-9310
MASSACHUSETTS	(617)	273-3970
MINNESOTA	(612)	938-0001
MISSOURI	(913)	451-3115
NEW JERSEY	(201)	299-0002
NEW YORK,		
Liverpool	(315)	457-5400
Poughkeepsie	(914)	471-8180
Woodbury	(516)	364-8020
NORTH CAROLINA	(919)	878-8111
OHIO,		
Columbus	(614)	891-6455
Dayton	(513)	439-0470
OREGON	(503)	245-0080
PENNSYLVANIA,		
Allentown	(215)	398-8006
Willow Grove	(609)	662-2900
SOUTH CAROLINA	(803)	772-6760
TEXAS,		
Austin	(512)	346-7830
Dallas	(214)	934-9099
Houston	(713)	785-9001
WASHINGTON	(206)	455-3600
WISCONSIN	(414)	792-0590

International

BELGIUM, Bruxelles	TEL	(02) 771-91-42
	FAX	(02) 762-37-12
	TLX	61028
FRANCE, Paris	TEL	(1) 49-75-10-10
	FAX	(1) 49-75-10-13
	TLX	263282
WEST GERMANY,		
Hannover area	TEL	(0511) 736085
	FAX	(0511) 721254
	TLX	922850
München	TEL	(089) 4114170
	FAX	(089) 406490
	TLX	523883
Stuttgart	TEL	(0711) 62 33 77
	FAX	(0711) 625187
	TLX	721882
HONG KONG	TEL	852-5-8654525
	FAX	852-5-8654335
	TLX	67955AMDAPHX
ITALY, Milan	TEL	(02) 3390541
		(02) 3533241
	FAX	(02) 3498000
	TLX	315286
JAPAN,		
Kanagawa	TEL	462-47-2911
	FAX	462-47-1729
Tokyo	TEL	(03) 345-8241
	FAX	(03) 342-5196
	TLX	J24064AMDTKOJ
Osaka	TEL	06-243-3250
	FAX	06-243-3253
KOREA, Seoul	TEL	82-2-784-7598
	FAX	82-2-784-8014
LATIN AMERICA,		
Ft. Lauderdale	TEL	(305) 484-8600
	FAX	(305) 485-9736
	TEL	5109554261 AMDFTL
NORWAY, Hovik	TEL	(02) 537810
	FAX	(02) 591959
	TLX	79079
SINGAPORE	TEL	65-2257544
	FAX	65-2246113
	TLX	RS55650 MMI RS
SWEDEN,		
Stockholm	TEL	(08) 733 03 50
	FAX	(08) 733 22 85
	TLX	11602
TAIWAN	TLX	886-2-7122066
	FAX	886-2-7122017
UNITED KINGDOM,		
Manchester area	TEL	(0925) 828008
	FAX	(0925) 827693
	TLX	628524
London area	TEL	(04862) 22121
	FAX	(0483) 756196
	TLX	859103

North American Representatives

CANADA		
Burnaby, B.C.		
DAVETEK MARKETING	(604)	430-3680
Calgary, Alberta		
VITEL ELECTRONICS	(403)	278-5833
Kanata, Ontario		
VITEL ELECTRONICS	(613)	592-0090
Mississauga, Ontario		
VITEL ELECTRONICS	(416)	676-9720
Quebec		
VITEL ELECTRONICS	(514)	636-5951
IDAHO		
INTERMOUNTAIN TECH MKGT	(208)	888-6071
INDIANA		
ELECTRONIC MARKETING CONSULTANTS, INC	(317)	253-1668
IOWA		
LORENZ SALES	(319)	377-4666
KANSAS		
Merriam		
LORENZ SALES	(913)	384-6556
Wichita		
LORENZ SALES	(316)	721-0500
KENTUCKY		
ELECTRONIC MARKETING CONSULTANTS, INC	(317)	253-1668
MICHIGAN		
MIKE RAICK ASSOCIATES	(313)	644-5040
MISSOURI		
LORENZ SALES	(314)	997-4558
NEBRASKA		
LORENZ SALES	(402)	475-4660
NEW MEXICO		
THORSON DESERT STATES	(505)	293-8555
NEW YORK		
NYCOM, INC	(315)	437-8343
OHIO		
Centerville		
DOLFUSS ROOT & CO	(513)	433-6776
Columbus		
DOLFUSS ROOT & CO	(614)	885-4844
Strongsville		
DOLFUSS ROOT & CO	(216)	238-0300
PENNSYLVANIA		
DOLFUSS ROOT & CO	(412)	221-4420
UTAH		
R^2 MARKETING	(801)	595-0631

Advanced Micro Devices, Inc. 901 Thompson Place, P.O. Box 3453, Sunnyvale, CA 94088, USA
Tel: (408) 732-2400 • TWX: 910-339-9280 • TELEX: 34-6306 • TOLL FREE: (800) 538-8450
APPLICATIONS HOTLINE TOLL FREE: (800) 222-9323 • (408) 749-5703

WCP-7500-8/88-0 Printed in USA

ADVANCED MICRO DEVICES, INC.
901 Thompson Place
P.O. Box 3453
Sunnyvale,
California 94088-3453
(408) 732-2400
TWX: 910-339-9280
TELEX: 34-6306
TOLL-FREE
(800) 538-8450

APPLICATIONS HOTLINE
(800) 222-9323
(408) 749-5703

Printed in USA
09559A