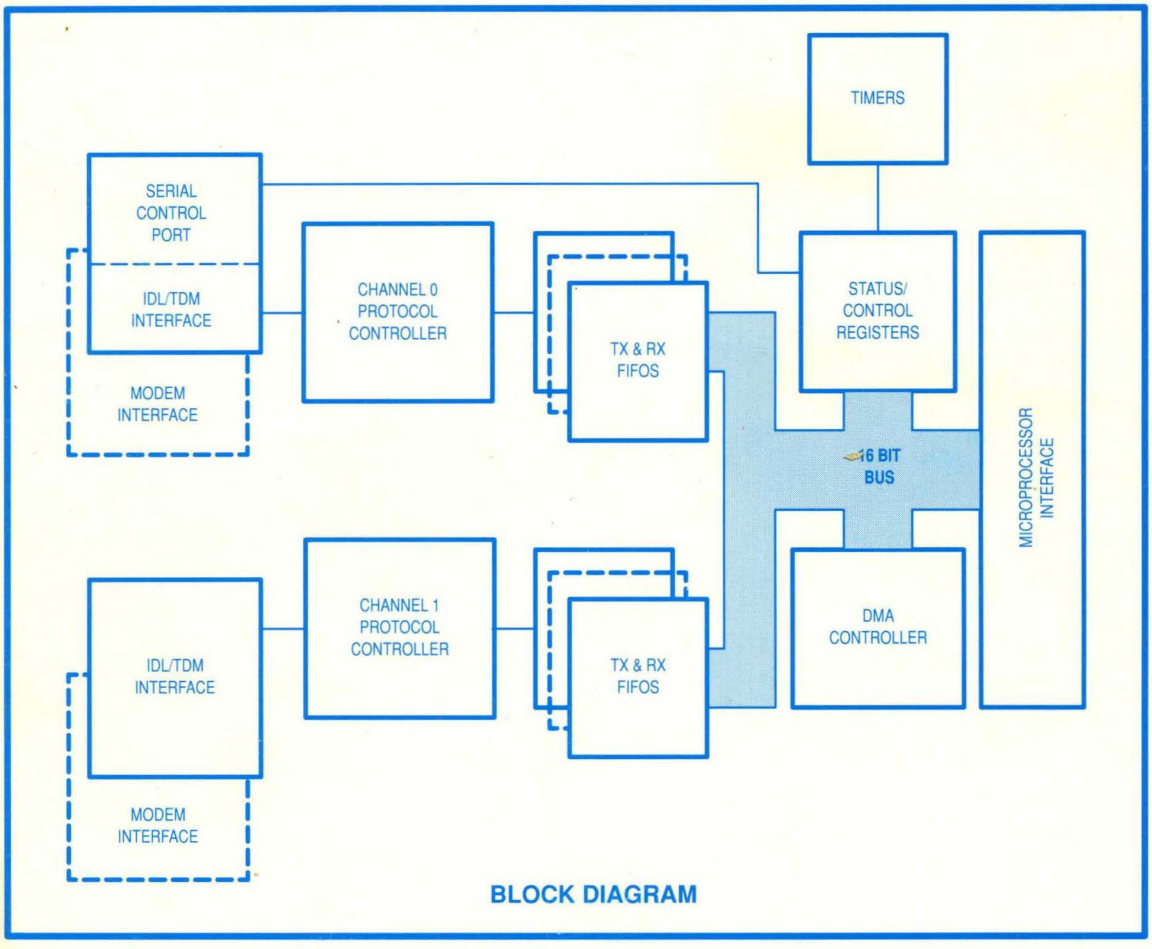
MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Order this document
by MC145488/D

MC145488

Dual Data Link Controller (DDLC)



BLOCK DIAGRAM

MOTOROLA

Replaces ADI1524R1

MC145488
Dual Data Link Controller (DDLC)
MOTOROLA

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Concluded)

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Concluded)

SECTION 1
OVERVIEW

1.1 INTRODUCTION

The MC145488 is a two-channel ISDN LAPD controller with an on-chip direct memory access (DMA) controller. It is intended for ISDN terminal and switch applications where one or two channels of data will use HDLC-type protocols. The DDLC is ideally suited for use with the MC145474 S/T transceiver. The interchip digital link (IDL) easily connects the chips together, providing a powerful layer one/layer two ISDN solution. A serial control port is provided to efficiently control the MC145474 or other ISDN family devices. The DDLC is compatible with 68000 and 80186 bus structures.

Places where significant changes or clarifications have been made between this data sheet and the Advance Information data sheet are indicated by change bars in the left margin.

1.2 FEATURES

- Two Independent Full-Duplex Bit-Oriented Protocol Controllers Supports: HDLC, SDLC, CCITT X.25, CCITT Q.921 (LAPD), and V.120 at Basic and Primary Rates
- Four Channel On-Chip DMA Controller
  - 64 Kbyte Address Range with Expansion Control
  - Internal Programmable Wait-State Generator
  - Two Buffer Descriptors for Each Receiver Channel
- Compatible with 68000 and 80186 Bus Structures
  - Nonmultiplexed 16- or 8-Bit Data Bus
- Bit-Level HDLC Processing Including:
  - Flag Generation/Detection
  - Abort Generation/Detection
  - Zero Insertion/Deletion
  - CRC-CCITT Generation/Checking
  - Residue Bit Handler
- TEI/SAPI Address Comparison
  - Three Address Comparison
  - Wildcard Bits for Block Comparisons
- Transparent Mode for Codec Compatibility
- Programmable Interrupt Vector Generation
- Two Independent Timers Configurable as a Watchdog Timer
- Flexible Serial Interface with:
  - IDL Interface for Connection to Other ISDN Family Devices
  - Timeslot Interface for Connection to PBX-Type Backplanes
  - Modem Interface for Other Applications
- Supports CCITT Specification 1.460
- Supports DMI Specification 3.1 Modes 0, 1, 2, and 3
- Serial Control Port for ISDN Family Device Control
- Low-Power CMOS with Automatic Power-Down

1.3 DDLC OVERVIEW

The MC145488 Dual Data Link Controller (DDLC) is a high-performance two-channel protocol controller with an on-chip direct memory access controller (DMAC). Each channel has a full-duplex transceiver with independent protocol controllers to handle the bit-level tasks of HDLC-type bit-oriented protocols including LAPB and LAPD. Each channel also has dedicated DMA controllers for transmit and receive. A transparent mode is provided which bypasses the protocol circuitry so that serial data may be directly DMAed between the host processor's memory and the serial interface. The DDLC's microprocessor interface is configurable to 68000 or 80186 systems and may be used in eight-bit or sixteen-bit modes.

Each channel has a serial data interface which operates up to primary rate speeds in three modes: IDL, Timeslot, and Modem. In the IDL (interchip digital link) mode for ISDN applications, the IDL bus is supported. While in the IDL D mode, access control lines to the D-channel, through the MC145474 S/T transceiver, are included. The timeslot mode is used to connect the DDLC to PBX-type PCM highway backplanes. Both long-frame and short-frame timing are supported as well as synchronous transmit and receive. In the modem mode, each channel has its own separate transmit and receive clock inputs along with modem control lines (RTS, CTS, and CD). The two channels are independent and may be in different interface modes.

A serial control port (SCP) is provided to pass control information to other devices in a system. The SCP is compatible with Motorola's Serial Peripheral Interface (SPI) and National Semiconductor's Microwire Plus*. Two internal timers may be used for general purpose, low resolution timing of HDLC-type protocols. One of the timers may be configured as a watchdog timer to reset the entire system in the event of a hardware or software failure.

Power consumption is an important aspect of ISDN terminal designs, and the DDLC was designed to use the minimum power possible while maintaining maximum functionality. The DDLC keeps power consumption to a minimum with an automatic power-down feature that turns off sections of circuitry that are not being used. Only those circuits that are actually used, such as when the CS* pin is activated for a register read/write or when the DMA controller performs a bus transaction, does the chip enter the normal power state for the duration of the access and for any time required for internal processing.

Two internal loopback functions and special chip and system test modes are available. The loopbacks are controlled by the host for on-line maintenance. The test modes are activated by bits in the master control register and provide access to the internal state machines.

1.4 HDLC PROTOCOL OVERVIEW

HDLC (High-Level Data Link Control) and its descendants, LAPB (Link Access Protocol-Balanced) and LAPD (Link Access Protocol for the D-channel), are bit-oriented synchronous protocols which are finding increased usage in data communications systems. LAPB and LAPD share the basic format of HDLC but differ in certain aspects of the software implementations. See Figure 1-1.

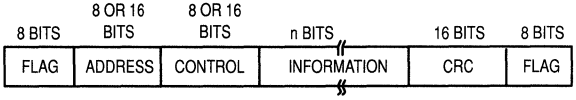


Figure 1-1. HDLC Frame Format

*Microwire is a trademark of National Semiconductor Corp.

The DDLC in the packet mode transmits and receives data in a format called a frame or packet. All frames start with an opening flag and end with a closing flag. Between the flags, a frame contains an address field, control field, information field, and a cyclic redundancy check field (CRC).

Flag — The flag is the unique binary pattern (01111110). It provides the frame boundary and a reference for the position of each field of the frame. The DDLC transmitter generates a flag pattern internally and the opening flag and closing flags are appended to a frame automatically. Two successive frames can share one flag as a closing flag of the first frame and the opening flag of the succeeding frame. The receiver searches for a flag on a bit-by-bit basis and recognizes a flag at any time. The receiver establishes frame synchronization with every flag. The flags mark boundaries and references for each data field but they are not transferred to the receive FIFOs or memory.

Address Field — The 8- or 16-bits following the opening flag comprise the address field. The address field is used to distinguish between the various devices in a network. Devices receiving frame not addressed to them may discard and ignore the frames. The DDLC has address recognition circuitry included which relieves the host from this task.

Control Field — The 8- or 16-bits following the address field are the control field. Commands and responses between the devices in a network are exchanged in this field.

Information Field — This field follows the control field and precedes the CRC field. The information field contains the "data" to be transferred and may be a null field. Additionally, the information field is not necessarily an integral number of octets in length. The DDLC receives non-octet frames as octets and LSB-justifies the extra "orphan" bits at the end of the field. The number of orphan bits is indicated in a residue register.

Cycle Redundancy Check Field — The 16-bits preceding the closing flag are the CRC field. This field detects bit errors in the address, control, and information fields. Checking is with the standard CCITT polynominal $\times 16 \times 12 \times 5 + 1$ for both the transmitter and receiver. Both the transmitter and receiver polynomial register are initialized to all '1s' when a flag character is detected, then begins calculating the CRC. The transmitter calculates the CRC on all bits of the frame except for the flags and transmits the complement of the resulting remainder as the CRC field. The receiver performs the similar computation on all bits except for the flags and compares the result to F0B8 (hex). When the result matches F0B8, the frame is accepted and the buffer is closed. CRC generation and checking are automatically performed by the DDLC. The CRC field is transferred to the buffer in memory. If desired, frames with CRC errors can be discarded automatically. All CRC errors are logged in a counter. For network-level error checking, the transmitter can generate faulty CRCs to simulate transmission errors.

Zero Insertion and Deletion — Zero insertion and deletion, which allows the content of the frame to be transparent, is automatically performed by the DDLC. A binary '0' is inserted by the transmitter after any succession of five '1s' within a frame (between flags). This eliminates the possibility of data imitating a flag character. The receiver deletes all '0s' that were inserted by the transmitter to regenerate the original data.

Abort — The function of prematurely terminating a data frame is called an abort. The transmitter aborts a frame by sending between seven and fourteen consecutive '1s'. When the receiver detects an abort character, it responds by clearing the FIFO and clearing the buffer in memory. It then begins searching for a new frame.

Idle and Interframe Time Fill — For LAPB and other applications, there are three states that the data link may be in: in-frame, inter-frame time fill, and idle. In-frame is the period from e beginning of an opening flag and the end of a closing flag. Inter-frame time fill is the period between frames

when continuous flags are transmitted. Idle is an out-of-frame period when continuous '1s' are on the link. In LAPD, on the D-channel, there are only two states: in-frame and idle. Continuous flags are not transmitted between frames.

SECTION 2
BLOCK DIAGRAM DESCRIPTION

This section describes the internal blocks of the DDLC. The blocks include two protocol controllers which handle the bit-level aspects of HDLC-like packet protocols and four FIFOs which buffer the data, a four-channel DMA controller and a microprocessor interface block which connects the DDLC to the host system. The serial interface block is described in detail in Section 6. Figure 2-1 is a simplified block diagram of the DDLC. While the DDLC has two data transceivers, only one is shown for simplicity.

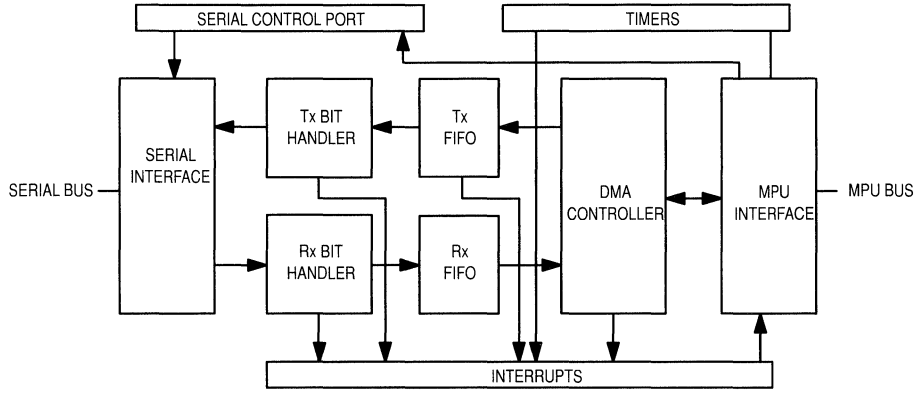


Figure 2-1. DDLC Block Diagram (One Transceiver Shown)

2.1 TRANSMIT BIT HANDLER

Two identical bit-level protocol transmitters are provided which perform HDLC-type framing. This section describes the operation of one transmitter, but it applies to both.

2.1.1 Packet Operation

The transmitter is designed to operate with as little intervention from the host processor as possible. To transmit a frame of data, the host merely informs the DDLC of the starting address of the data frame in memory and the length of the frame in bytes. The DDLC then transmits an opening flag and the data (LSB first) from memory. When the transmitter detects that the end of the data buffer has been reached, a CRC field and a closing flag are appended. Zeros are automatically inserted after ever string of five ones in the data to prevent imitation of a flag or abort character. The transmitter generates an abort character if the FIFO underruns. During inter-frame periods, the DDLC has the capability of transmitting either continuous flags (7E hex) or continuous marks (FF hex) as selected by the Inter-Frame Time Fill bit. For network-level fault isolation when the Force CRC Error bit is set, faulty CRCs are generated simulating a transmission error. The transmitter

also continuously monitors the $\overline{CTS}$ pin and indicates the current state of the pin with the $\overline{CTS}$ Status bit in the Transmit Status register.

2.1.1.1 STATE DIAGRAM. Figure 2-2 is the state diagram for the transmitter in packet operation. IDLE is entered any time the Transmit Enable bit is low or when Mark Idle is selected and no data is ready to transmit. When data is ready or Flag Idle is selected, the SEND FLAG 1 state is entered. At least one flag character will be sent before the SEND DATA state is entered. The state machine remains in the SEND DATA state until an End-of-Frame (EOF) Tag is received through the FIFO from the DMA controller. SEND CRC then SEND FLAG 2 states are entered. After sending the closing flag, if data from another frame is ready, SEND DATA is entered allowing the closing flag to be shared with the following frame's opening flag. If no data is ready, either IDLE or SEND FLAG 1 is entered depending on the setting of the Inter-Frame Time Fill bit.

A FIFO underrun causes the state machine to enter the ABORT state where an abort character is sent to clear the link. When the modem control signals are used, if the $\overline{CTS}$ pin goes inactive for more than one bit-time, an abort condition is generated and the current state is exited for the ABORT state. See Section 2.1.1.3 for a further discussion of flow control.

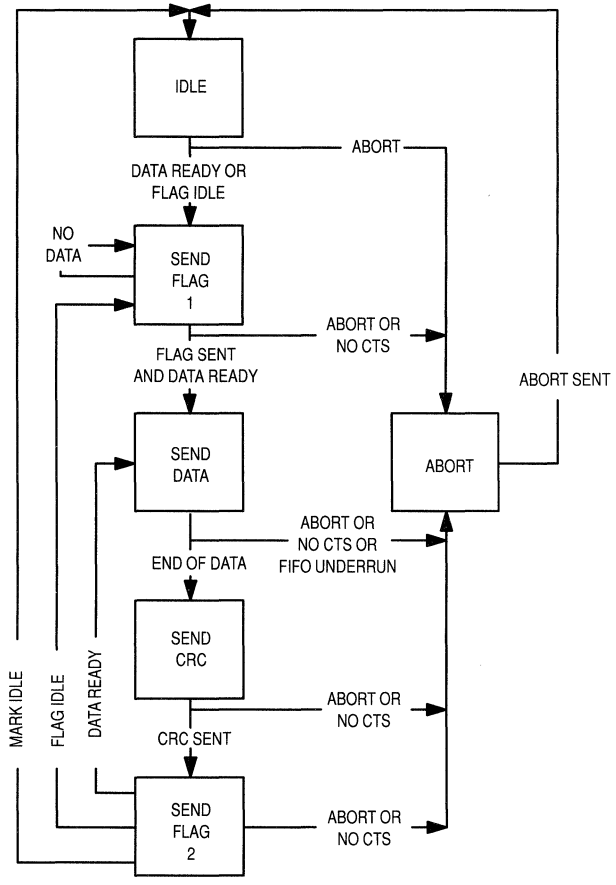


Figure 2-2. Transmitter State Diagram HDLC Operation

2.1.1.2 ABORT CHARACTER. The DDLC transmits an abort character which is compatible with DMI 3.1 modes 2 or 3 for restricted B-channel applications. In restricted access, no more than seven '0s' in a row may be transmitted. Because the HDLC protocol format may have any number of '0s' contiguously in the data field of a frame, DMI modes 2 and 3 invert all data before transmission. The DDLC inverts all transmitted data bits when the Data Invert bit of the Transmit Control register is set to '1'. Normally, with the data inverted, the maximum number of '0s' transmitted is six, the number of '1s' (inverted to '0s') of a flag (01111110) character. However, an HDLC abort character is seven to fourteen contiguous 's', which when inverted may be too many '0s', so the DDLC transmits an abort character of 01111111. This ensures that the maximum number of '0s' (inverted from '1s') is limited to seven even if the preceding data was five '1s'. In LAPB operation, the character following the abort character will always be a flag with the first bit of a '0'. When the DDLC is configured for ISDN D-channel use, an abort character of 11111111 is followed by idle (continuous '1s').

In certain applications a software-generated abort may be desired. The following is the recommended procedure to generate an abort: 1) clear the Buffer Ready bit and set the Force Abort bit, 2) reload the DMA pointers to the next frame's buffer, and 3) set the Buffer Ready bit. The DMA pointers must be updated before the Buffer Ready bit is set. This procedure disables any pending DMA requests then causes the transmitter to begin an abort sequence. At the same time the state machine clears the FIFO. The user then prepares the next frame for transmission. Simply negating the Transmit Enable bit will cause an abort sequence, but it is uncontrolled. The host processor does not know when the idle bits are actually transmitted and if the transmitter is re-enabled before at least seven '1s' were transmitted, a valid abort may not be transmitted.

When the Transmit DMA Complete Interrupt is enabled, if an abort sequence is generated due to negation of the $\overline{\text{CTS}}$ pin or a FIFO underrun, the Buffer Ready bit in the Transmit Control register is reset after the abort has been completely transmitted. The host is then free to prepare a buffer descriptor for the next frame to be transmitted, either a new frame or the aborted one. Control of the Buffer Ready bit is different if the Transmit Frame Complete Interrupt is enabled. This is discussed in Section 2.1.1.3.2.

2.1.1.3 FLOW CONTROL MECHANISMS. The DDLC provides two flow control mechanisms: one for basic rate ISDN applications and the other for standard modem applications. The following paragraphs describe the operation of the two schemes.

2.1.1.3.1 ISDN D-Channel Contention. When the DDLC is operating on the D-channel with the companion MC145474 S/T transceiver, the DREQ and DGNT lines must be used to comply with the basic rate D-channel contention algorithm. When the DDLC has a data frame to transmit, it asserts DREQ (high). The MC145474 transceiver monitors the S/T interface for activity on the D-channel and indicates the the channel is free by asserting DGNT (high). While DREQ is high, the DDLC samples DGNT on the falling edge of IDL SYNC. Transmission from the DDLC begins in the IDL D-bit time slots when DREQ and DGNT are both active. If a D-channel collision is detected on the S/T interface by the MC145474 transceiver, DGNT is negated (low). ISDN D-channel contention rules state that when a collision is detected, the colliding device must immediately stop transmitting and retransmit the frame at a later time when the D-channel is again available. When a collision is indicated, the DDLC automatically aborts the frame in progress and prepares to retransmit the entire frame when the D-channel again becomes available. This is done without interrupting the host. The Transmit Frame Complete Interrupt must be enabled to allow this operation. When the packet has been successfully transmitted a Transmit Frame Complete Interrupt is generated when the last '0' of the closing flag is transmitted.

If a software generated abort is desired while in D-channel operation, Transmit DMA Complete should be enabled and Transmit Frame Complete should be disabled before the Force Abort bit is set. This ensures that the aborted frame will not be retransmitted after the abort character is sent. After the Force Abort bit is cleared by the transmitter, the host is free to prepare a buffer descriptor for the next frame to be transmitted.

2.1.1.3.2 Modem Flow Control. The transmitter indicates to a modem that it has data ready to transmit with signals similar to D-channel operation. In this mode, Request-To-Send ($\overline{RTS}$) is directly controlled by the Transmit Enable (TE) bit. When TE is high, the $\overline{RTS}$ pin is asserted (low). During inter-frame periods, either flags or marks, as selected, are transmitted but the $\overline{RTS}$ pin remains asserted until the user negates the TE bit. Transmission of a frame, if one is ready, actually begins when the modem asserts Clear-to-Send ($\overline{CTS}$ low). If $\overline{CTS}$ is negated for more than one Tx CLK period while a frame is in transmission, the frame is aborted and the DMA pointers are reset so that the frame can be retransmitted without interrupting the host.

Section 2.1.1.2 indicates that while the Transmit DMA Complete Interrupt is enabled, the Buffer Ready bit is reset after the abort character is transmitted. This operation is modified while the Transmit Frame Complete Interrupt is enabled, in which case the Buffer Ready bit remains set after an abort character is sent. When a collision is detected, an abort character is sent immediately followed by automatic retransmission of the colliding frame.

2.1.1.4 INTERRUPTS. There is one interrupt generated by the transmitter state machine. Transmit Frame Complete indicates that an entire frame and its closing flag have been successfully transmitted. It is issued when the state machine exits the SEND FLAG 2 state. Ordinarily this interrupt is used for basic rate ISDN D-channel operation. Two other interrupts associated with the transmitter are generated by the DMA controller and are discussed in Section 2.3.2.

2.1.2 Transmit FIFO

The transmitter has a FIFO which buffers it from the DMA controller. It is four characters deep and nine bits wide. The ninth bit is a Tag bit which is attached to the last byte of a frame by the DMA controller. When the host has a message to send, the DMA controller fills the transmit FIFO and attempts to keep if full. A comparator in the controller keeps track of the FIFO occupancy and requests that the DMA controller load a word of data (16-bits) when the FIFO has two empty bytes. In 8-bit operation the DMA controller fetches one byte when there are one or more bytes available in the FIFO. At 64 kbps, DMA requests from a transmit FIFO are generated at approximately 250 ms intervals. If the DMA controller does not service a request and the FIFO underruns, an interrupt is queued and the current frame in transmission is aborted.

2.1.3 Transparent Operation

The transmitter has the capability of operating with unframed data such as PCM-encoded voice or proprietary protocols. Raw data may be transmitted from memory with byte alignment maintained through the FIFO and transmitter. Byte alignment signals must be provided. In he modem mode, the alignment signal is externally generated. In IDL and timeslot operation, it is internally generated but user programmable. See Section 6.1.3 for further discussion. The DDLC transmits data continuously as long as there is data to transmit. To keep a continuous stream of data flowing, the Transmit DMA Complete Interrupt should be enabled. This interrupt occurs at least 16 bit-times before the FIFO becomes empty and the state machine enters the IDLE state. At 654 kbps, 250 ms is available to prepare the next buffer for transmission before the FIFO empties

and idle characters (either marks or flags) are transmitted. In 16 bit mode the DDLC transmits an even number of bytes. If an odd number is programmed it will transmit the least significant byte of the last 16 bit word read from memory by the DMA controller.

2.1.4 Interframe Time Fill

The bit sequence that is transmitted between frames is determined by the value of the ITF bit in the Transmit Control register. The interframe time fill can be either the X.25 flag character (7E hex) or the LAPD marks ('1') idle. See Section 3.8 for further details on the ITF control bit.

2.2 RECEIVE BIT HANDLER

The receiver provides the complementary functions to the transmitter. This section describes the operation of one receiver, but it applies to both receivers.

2.2.1 Packet Operation

The receiver is reset and idle until the Receive Enable bit is set, at which time it begins searching for a flag character. When a flag is found, the selected address field of the frame, if desired, is checked and if a match is found, the DDLC passes the frame of data to the allocated buffer in memory. If no address match is found the DDLC clears the FIFO, resets the DMA pointers and searches for a new frame. Zeros inserted by the transmitter are removed from the data before placing the bytes in memory. When the closing flag is detected, the CRC field is checked and if found to be correct, the DDLC queues an interrupt indicating that a good data frame has been received and is in memory. If the CRC is found to be in error, the DDLC resets the buffer pointers to the start of the buffer and searches for a new frame. This in essence clears the buffer and discards the frame in error. In certain instances, frames with CRC errors will be of interest. A control bit is provided so that frames with CRC errors can be passed to memory. The Receive on CRC Error bit enables this feature and the CRC Error bit indicates that the frame has an error. All CRC errors are logged in the CRC Error register. The DDLC rejects undersize frames. If a frame of less than four octets between flags is found, it usually means that a bit error occurred, so OUT OF FRAME is entered.

If an abort character is found, the buffer pointers in the DMA controller are reset and the aborted frame is ignored. The FIFO is cleared and the receiver begins searching for a flag. No interrupt is generated.

2.2.1.1 STATE DIAGRAM. Figure 2-3 shows the state diagram of the receiver. When the Receiver Enable bit is initially set, the state machine is in the OUT OF FRAME state. The pattern detector continuously searches for flags and when it finds one, the FLAG state is entered. If the next character is not a flag, the machine enters the IN FRAME state. Address recognition is sampled on the first or second byte after entering this state. If the address does not match, the OUT OF FRAME state is entered and the receiver searches for a new frame. When a flag is found in the IN FRAME state, an end of frame is assumed and the CRC checker is sampled and an End-of-Frame Tag is generated and passed to the FIFO. Then the FLAG state is entered. The receiver detects and reacts to aborted frames. If an abort is detected, the state machine enters OUT OF FRAME. Section 2.3.4 has more discussion of the different End-of-Frame Tag signals.

2.2.1.2 NON-OCTET ALIGNED PACKETS. The receiver has the capability of operating in non-octet aligned packet systems. The residue bit count indicating the number of orphan data bits at the end of the information field is placed in the RC bits of the Receive Status register. These bits are

valid until overwritten by another frame. In non-octet aligned systems the software should check the residue count soon after receiving a Receive Buffer Complete Interrupt to ensure that the residue count is not overwritten by the next frame. Orphan bits are LSB justified in memory. Note: The DDLC cannot transmit non-octet aligned frames.

2.2.1.3 ADDRESS RECOGNITION (FILTERING). The receiver can filter received frames by comparing their address fields to user programmable addresses. Two addresses may be programmed with another (broadcast, FF hex) hardwired into the receiver. Address filtering may be performed on either the first OR second octet following the opening flag of a frame. The ABS bit in the Receive Control register when high enables filtering on the first octet and when low filters on the second octet. Typically, in ISDN terminal applications, the TEI address (second) field will be of interest. In network applications, the SAPI (first) field will be checked. A separate Wildcard register allows selected bits of Compare Address 0 to be ignored during the comparison procedure. This provides a simple way to widen the "bandwidth" of the address filter allowing blocks of addresses to be passed to memory. If received frames are rejected by address recognition, the receiver is reset and searches for a new frame. Address recognition may be disabled by clearing the Address Compare Enable bit to '0'.

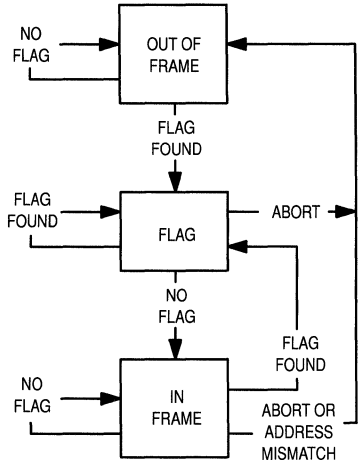


Figure 2-3. Receiver State Diagram
HDLC Operation

2.2.2 Receive FIFO

The receiver has a FIFO which is similar to the transmitter's. It is four characters deep and ten bits wide (eight hits for data and two bits for the Tag). Serial bytes are produced by the receiver and converted to parallel. As each byte is formed, it is pushed into the FIFO. A comparator in the controller keeps track of the occupancy of the FIFO and requests that the DMA controller place a word of data (16-bits) in memory when there are two or more bytes in the FIFO. In 8-bit operation the FIFO requests service when one or more bytes of data read ready to be placed in memory. If the FIFO overruns because the DMA controller did not service a request, an interrupt is queued. When a receiver is operating at 65 kbps in the 16-bit mode, DMA requests from that FIFO occur at approximately 250 ms intervals.

2.2.3 Transparent Operation

The DDLC provides transparent operation for passing raw octet-aligned serial data to memory via DMA. This feature is useful for storing PCM voice or proprietary protocols in memory. When using the DDLC to pass PCM voice to memory, maximum buffer size of 4096 bytes should be used. Ordinarily, this will cause an interrupt every 500 ms. When the Buffer Overrun Interrupt is generated, the DMA controller automatically switches to the alternate buffer (if it is available) and begins placing data in it. No data will be lost in the change over from one buffer to the other. Note that the Receive DMA Complete Interrupt is not generated in transparent operation. Because the transparent mode requires that data to be in eight-bit quantities, synchronization procedures for defining octet boundaries are required. In the modem mode, the sync signal is externally generated and input on the CD* pin. In IDL the timeslot operation, the sync signal is internally generated but user programmable. Once byte alignment is obtained in the receiver, it is maintained through the DMA controller into memory. Section 6.1.3 contains more details regarding transparent operation.

2.2.4 Interrupts

There are two interrupts generated by the receiver. Receive idle indicates that 15 or more consecutive '1s' were received. This interrupt is considered normal operation. The current status of receive idle and carrier detect is available in the Receive Status register, but it must be kept in mind that they can change immediately after being read. The carrier detect pin also generates an interrupt when it changes state. Section 5.3 details all interrupts.

2.3 DMA CONTROLLER

In order to relieve the host's software from critically timed data transfers to or from the protocol controllers, the DDLC provides four DMA channels, one for each transmitter and receiver.

2.3.1 DMA Operation

When the DMA controller detects a service request from one of the FIFOs, it prepares the address and data from the transfer then requests ownership of the system bus from the host. When ownership is granted, the DMA controller assumes control of the bus and transfers data either to or from memory. Transfers are 16 bits or 8 bits depending on the selected bus width. In a 16-bit system, all transfers are word-wide. When the number of bytes in a received frame is odd, the last byte is placed in the most significant byte of the last word. The least significant byte will contain unknown data. The receive byte count will contain the correct number of bytes received (including the CRC). When odd length frames are transmitted, the last word read from memory will have the last byte transmitted in the most significant byte and the least significant byte of that word will be discarded. For operation in 80186 systems, see Section 5.3.

The DMA controller uses a round robin strategy to service internal DMA requests. A channel that was just serviced will not be polled again until all other channels have ben polled and serviced, if needed. The DDLC services one DMA request per bus arbitration cycle. For example, if two channels have request pending, when the DDLC assumes ownership of the bus, one channel is serviced then the bus is relinquished. The DDLC re-requests the bus no sooner than three MCLK cycles after the bus was relinquished allowing another bus master to begin a cycle. This type of operation improves system performance and guarantees that the DDLC will be well behaved. The DDLC prepares the data to be transferred (on receive DMA requests) and calculates the address before requesting the bus. This requires minimum overhead (dead) cycles while the host is transferring ownership of the bus. The next bus owner may assume ownership immediately after the DDLC has relinquished the bus.

It is impossible to precisely predict what the DDLC bus occupancy will be, but worst case with both channels operating full-duplex at 64 kbps (aggregate rate of 256 kbps) in a 16-bit 68000 system with a 12 MHz MCLK, approximately 0.66% of the host bus bandwidth will be consumed by the DDLC. This figure assumes 1-1/2 front-end overhead cycles and no wait-states. While the DDLC was designed to operate with both channels at 64 kbps, it is possible to operate at much higher data rates. Bus occupancy increases linearly with data rate. At very high data rates latency from the bus request to the bus grant and interrupt service latency become the limiting factors. It must also be kept in mind that the DDLC can generate interrupts quickly, especially with a large number of small data packets.

2.3.2 Buffer Descriptors

As previously stated, the DDLC has four DMA channels. Pointer registers and counters are required so that the DMA controller knows where to place or fetch data in memory.

2.3.2.1 TRANSMIT BUFFER DESCRIPTORS. When the host has a frame of data to transmit, it informs the DMA controller where the data resides in memory. A 16-bit register, the Transmit Base Address register points to the first word of the transmitted frame and provides 64 kbyte address range. The host programs the address of the first word to be transmitted into this register. The length of the data frame must also be given to the DMA controller so a 12-bit Transmit Frame Length register is used to indicate the length of the frame in bytes. Frames of up to 4096 bytes in length may be transmitted.

Back to back frames can be transmitted by updating the transmit buffers when the Transmit DMA Complete interrupt is generated. The transmit framer works as follows. The last byte read into the transmit FIFO due to a DMA termination is tagged. The transmit framer detects this tagged byte in the FIFO and automatically appends a closing CRC and flag after the tagged byte is sent. If the transmit buffers have been updated and new untagged bytes are in the transmit FIFO a new transmit frame is started. The DDLC guarantees that at least one flag will occur between transmitted frames.

Note: Once a transmit buffer descriptor has been prepared, it must not be disturbed until the Transmit DMA Complete or Transmit Frame Complete interrupts are generated. Modifying the contents of an active buffer descriptor may result in data being transmitted from unintended areas of memory.

2.3.2.2 RECEIVE BUFFER DESCRIPTORS. The receive buffer descriptors have a 16-Bit Receive Buffer Base Address register, a 12-Bit Buffer Length register, and a 12-Bit Frame Length register. The 16-Bit Base Address register provides 64 kbyte address range and contains the address of the first word of the data buffer which will accept a data frame. The 12-Bit Frame Length register indicates the length of the memory buffer in bytes. Buffers of up to 4096 bytes may be built. The DMA controller will never place data outside of the boundaries set-up by these two registers. The Frame Length register indicates the number of bytes including the CRC) received.

Each channel has a pair of buffer descriptors. These may be used alternately so that while one buffer is filling, another buffer is ready in-waiting. If back-to-back data frames are received, after the first buffer has been closed, the second is immediately ready for the next frame. There must be at least one buffer ready to accept data when the Rx Enable bit is set. Figure 2-4 describes the activity of the receiver with four buffers in memory. Initially, Buffer A Start Address is set to AB00 hex and Buffer B is set to AD00. The Length register is set to 200 hex. After the first frame is received, Buffer A is set to B300 hex to prepare for the third frame. After the second frame is received, the fourth

buffer is prepared with an address of B500. This process continues for the entire session. In a typical ISDN B-channel system at 64 kbps with frames of 256 bytes in length, interrupts requesting new buffer allocations will occur at approximately 32 ms intervals.

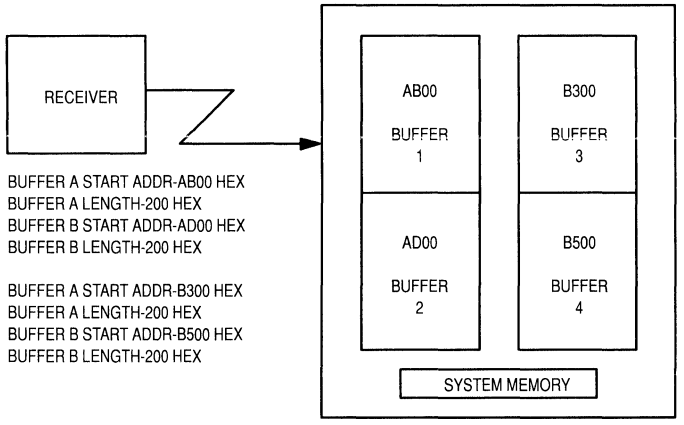


Figure 2-4. Alternate Receive Buffer Operation

If a package is coming in and no buffers are ready, the receive FIFO will overrun, the Receiver Enable bit is reset, and an interrupt is queued indicating the overrun. If both descriptors are ready, then Buffer A will be filled. if a received frame is larger than a buffer, the Buffer (A or B) Overrun Interrupt will be queued, but the receiver will continue to receive and the DMA controller will place the data in the alternate buffer (if it is available). If an alternate buffer is not ready, the Rx FIFO Overrun Interrupt will be generated and the receiver is reset.

Two buffer ready bits, Buffer A Rdy and Buffer B Rdy of the Receive Control register, are provided which allow the DMA controller to operate on that buffer. DMA activity on a receive channel begins when there is data in the FIFO and a buffer ready bit is set. As the buffer fills, the byte count increments. When the last byte has been placed in memory and the CRC has been checked OK, the selected buffer ready bit, Buffer A Rdy or Buffer B Rdy is cleared halting future DMA activity on that buffer until it is again prepared by the host.

Once a data frame has been completely received, the number of bytes received is indicated in the Frame Length register. The number in this register is valid only when the Receive DMA Complete bit (Buffer A or Buffer B) in the Receive Status register is set to '1'. If non-octet aligned frames are received, the number of residue bits (valid bits) in the last byte is indicated in the RC bits of the Receive Status register. The residue bits in memory will be least significant bit justified. For octet-aligned systems, the residue count should always be '0'.

Note: As with the transmitter, once a receive buffer descriptor has been prepared, it must not be disturbed until the closing flag has been found and DMA activity on the buffer has stopped. This is indicated by the Receive DMA Complete (Buffer A or B) interrupt. Modifying the contents of an active receive buffer descriptor may result in data being placed intl unintended areas of memory.

2.3.2.3 ADDRESS EXPANSION. The DDLC provides signals for expansion of the 64 kbyte address space. The $\overline{\text{OWN}}$ pins are activated with timing identical to the address pins to enable external address circuitry onto the address bus. Using the $\overline{\text{OWN}}$ pins with the R/$\overline{\text{W}}$ pin, the transmit and receive buffers may all be on separate 64 kbyte pages in memory. Section 7 describes address expansion techniques in more detail.

2.3.3 Transmit Channel Operation

Figure 2-5 is a simplified state diagram of the DMA controller's operation when a transmit channel requests service. The DMA controller continuously polls the four channels for DMA requests. When a transmit request is found an address pointer into the buffer is prepared and compared to the ending address of the buffer. If the pointer is pointing to the last byte of the buffer, an EOF Tag is prepared to inform the transmitter and the DMA Complete interrupt is generated. The bus is then requested. When it is granted a read cycle is performed. Data read from memory is placed into the FIFO in ascending address order. If during the bus cycle a bus error or address error is detected, the Buffer Ready bit is cleared halting further DMA requests from that channel. A System Error interrupt is then queued.

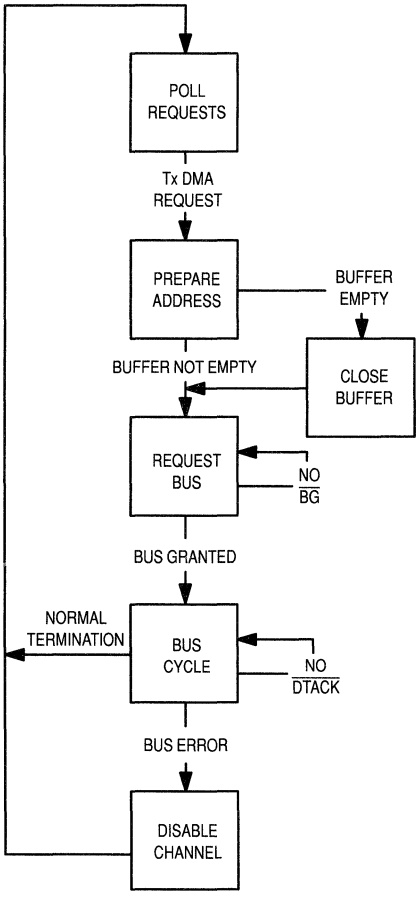


Figure 2-5. Transmit DMA State Diagram

An internal counter keeps track of the number of bytes transferred to the FIFO. When its count equals the number in the Frame Length register, an End-of-Frame Tag is generated and loaded into the FIFO along with the last byte. The Buffer Ready bit is reset, a Transmit DMA Complete interrupt is queued, and further DMA activity on that channel is suspended. The host prepares the next frame by setting up another buffer descriptor and setting the Buffer Ready bit high.

Four interrupts are produced by the transmitter DMA channel Transmit DMA Complete indicates that the last byte of data has been transferred from the buffer into the transmit FIFO. FIFO underrun indicates the DMA requests were not serviced and the FIFO underran. Bus error is generated when the $\overline{BERR}$ pin is activated during a DMA cycle, and address error is generated when either $\overline{IACK}$ or $\overline{CS}$ are activated during a DMA cycle.

Communication from the DMA controller to the transmitter is through the Tag bit as there is an unknown time delay from the DMA controller's activities to the time when the transmitter actually operates on the data. The EOF Tag informs the transmitter to close the frame with a CRC field and flag. Note that if the software reacts to the DMA Complete interrupt and prepares the next frame's buffer, the end of the previous frame may still be in the FIFO while the first two bytes of the next frame are loaded. When the transmitter finds this situation, after closing the first frame it shares the flag between frames and begins the next frame.

This section has only described the internal operation of the DMA controller. For details about DMA from the MPU's point of view, see Section 5.1.2.

2.3.4 Receive Channel Operation

Figure 2-6 is a simplified state diagram for operation of the DMA controller when a receive channel requests service. The DMA controller first gets data from the FIFO and arranges it into ascending order for 68000 or 80186 operation. While it is preparing the data, it examines the Tag bits. If the Tag indicates that an error was received, the byte counter is reset enabling the buffer to be reused. If a normal EOF is found, the Buffer Ready (A or B) bit is cleared and Receive DMA Complete interrupt is queued. If no Tag is found, a pointer to the next buffer location is prepared. If the pointer indicates that the buffer is full, a Buffer Overrun interrupt is queued and the alternate buffer is activated (if it is available). Only after the data and address are ready is Bus Request asserted. A write cycle is then executed when ownership is granted. if a bus error or address error is detected during a bus cycle, a System Error interrupt is queued and the RxEn bit is cleared resetting the receiver.

As with the transmitter, all communication between the receiver and DMA blocks is through the Tag bits in the FIFO. Because three end-of-frame conditions are reported, two bits are used. When a CRC Error Tag is detected, the Receive CRC Error bit in the Receive Status register is set (if RCE in the Control register is set) and the CRC Error Log counter is incremented. If RCE is not set, the Receive Byte Counter is reset enabling the buffer to be reused. When an Abort Tag is detected, the Byte Counter is reset effectively clearing the buffer for reuse. When a normal EOF is found, the Buffer Ready (A or B) is cleared, and a Receive DMA Complete interrupt is queued. When the next frame arrives, the alternate buffer will be used.

When transparent mode is used no tags are generated so the Rx DMA Complete interrupt is never generated. The DMA controller generates a Buffer Overrun interrupt when the buffer actually fills. At 654 kbps the host has approximately 375 ms before the FIFO overruns or 250 ms before the alternate buffer is activated.

2.4 MICROPROCESSOR INTERFACE

The microprocessor block interfaces the internal 16-bit bus to the host's 8- or 16-bit bus. All timing conversion and buffering is also performed. This block has three modes of operation: system slave, system master, and interrupt generator. This section will describe the operation of these three modes. Operation of the DDLC in an 80186 system is described in Section 5.2.

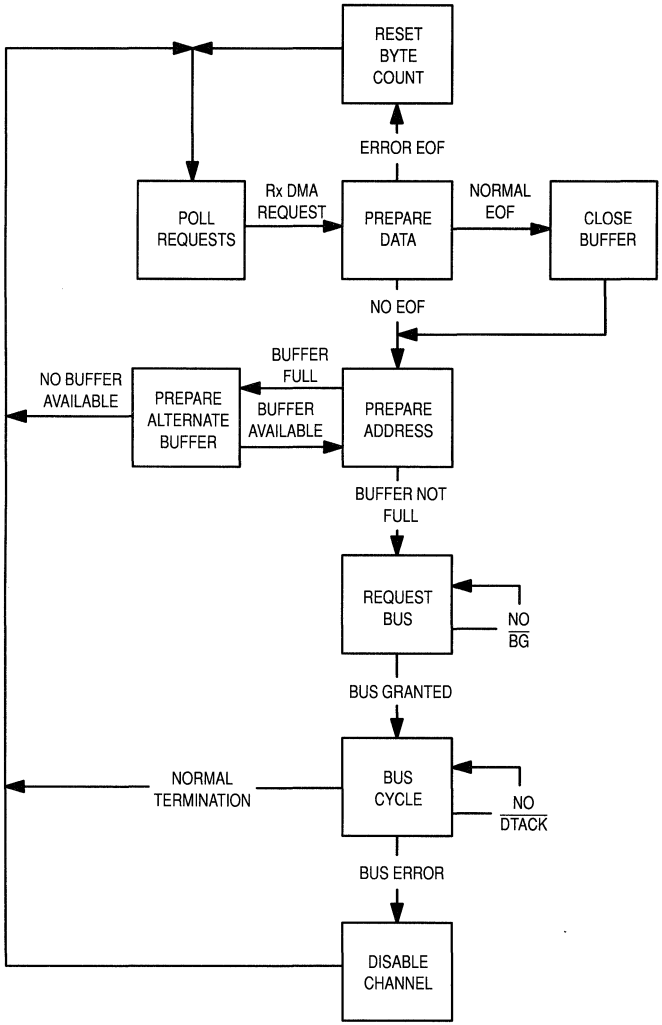


Figure 2-6. DMA Receive State Diagram

2.4.1 System Slave Mode

When the DDLC is in this mode, it appears as fast memory to the host processor. The host can read or write from or to the registers in the DDLC. This mode is entered when the $\overline{CS}$ pin is activated. Internal address decoding circuitry is selected and the desired register is connected to the internal bus for access by the host. Section 5.1.1 describes 68000 operation and Section 5.2.1 describes 80186 operation.

2.4.2 System Master (DMA) Mode

During DMA operation the DDLC becomes a system master and controls the system bus. When one of the internal FIFOs requests a DMA transfer, the DDLC negotiates with the system host for

ownership of the bus. After successful negotiation, one DMA request is serviced then the bus is relinquished. The DDLC has the capability of reading or writing data from or to memory. If the memory system is slow, the DDLC inserts wait states (user selectable) until the memory is ready to complete the access. The DDLC has the capability of recovering from system faults such as address or bus errors. If one of these faults are detected, the DDLC disables the particular DMA channel which encountered the fault and queues an interrupt indicating the fault. The points in the DMA register which had the fault remain where they were when the fault was detected, so the host may investigate the problem. A system fault on any channel will not affect the operation of any other channel. Sections 5.1.2 and 5.2.3 describe the operation of the DDLC as a bus master and the bus negotiation procedure.

2.4.3 Interrupt Operation

The DDLC has 27 vectored interrupt sources to inform the host of its status. One group of interrupts is normal operation interrupts. These inform the host that particular task was completed and that new tasks are desired. Another group is bit handler faults which inform the host that a DDLC channel detected a fault from which it cannot recover without assistance from the host. A third group is the timer and SCP interrupts. The last group of interrupts is the system faults. These include bus errors and address errors. When a DMA channel encountered one of these errors, it stops operation on the affected channel and informs the host that the channel has been disabled. All interrupts are maskable. In the case of system faults, all channels' interrupts are enabled or masked as a group.

The interrupts are presented to the host as a vector number in an interrupt acknowledge cycle. The interrupts are encoded into four bits so the DDLC vector space consumes 16 out of 256 locations. For applications not using vectored interrupts, the equivalent vector number is accessible in the Master Status register.

Interrupts are queued internally and a priority mechanism is used to ensure that the most important interrupt is serviced first. Interrupt operation of the DDLC is described more fully in Section 5.3.

All interrupt status bits have a feature where they may be set by the user to generate an immediate interrupt. This is useful for software debugging.

2.5 SERIAL INTERFACE

The serial interface block has a variety of configurations which make it compatible with most common interfaces. Each channel's serial interface is independent so two different configurations may be active simultaneously. The serial interface has an IDL mode, a timeslot mode, and a general purpose modem mode. The operation of these modes is fully described in Section 6.

2.6 SERIAL CONTROL PORT

A Serial Control Port, similar to the Serial Peripheral Interface (SPI) on Motorola single-chip microprocessors, is provided to communicate with external devices via a serial ink. The SCP functions are multiplexed onto other serial pins so when the SCP is enabled, certain modem control features will be lost. Section 6.4 describes in detail the operation of the SCP.

2.7 TIMERS

Two timers are provided for general purpose low-resolution protocol uses. The clock to the timer is derived from the Master MPU Clock (MCLK). The baud rate generator in the SCP block is used

to drive the timer divide chain. This clock is then divided by 1024 and applied to an eight-bit down-counter. The counter is readable and writable by the host and may be set to any value. The counter counts down toward zero from the current value. A non-maskable interrupt is generated when the counter underflows from FF to FE. The timers continue counting down after reaching FE. The status bit from the previous interrupt must be cleared before a new interrupt is generated. The timer function and interrupt are enabled by setting the Timer Enable bit in the Timer register to one. The timer interrupt status bits must be read while set before they may be cleared. See Section 5.3.3.

The timers are intended for low accuracy uses such as protocol timers. If the host is accessing the timer for a read or write and a count-down pulse arrives, the pulse is ignored. For its intended application, this will not cause any difficulty. Figure 2-7 describes the clock selection choices for the timers. A new count value can be written to the Timer register at any time while it is active. The timer will continue counting down from the new value.

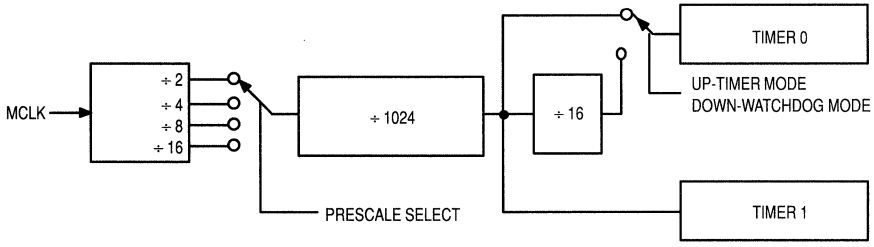


Figure 2-7. Timer Clock Selection

2.7.1 Watchdog Timer

Timer 0 may be configured as a watchdog timer for the entire host system. When the Watchdog Enable bit is set, an extra divide by 16 is added to the clock input of the counter. When the counter underflows from FF to FE, the Reset pin becomes an output for 16 MCLK cycles and a logic low is output. This provides a system reset to the host. The host can write any value (except FE hex) to the Timer register to setup any timeout. Timeouts of up to 5.6 seconds are available with a 12 MHz MCLK. See Section 5.4 for timing information on watchdog generated resets.

2.8 POWER CONSUMPTION

The DDLC is designed utilizing high-performance CMOS technology. As as result, average power consumption is very low. However, because there are wide address and data buses, peak currents may exceed 150 mA for short periods of time (less than 20 ns) while the drivers are charging or discharging the buses.

The bit handler sections of the chip are only clocked during active bit-times. In the IDL or timeslot modes, even though there is a continuous clock running, internal circuitry is active only while the timeslots are enabled. The average power consumption is thus proportional to the bit rate; i.e., for an eight-bit time slot in an IDL system, the bit rate is 64 kbps. The peak bit rate may be 2.56 MHz and during an active timeslot the power consumption will be proportional to the peak bit rate. However, averaged over a 125 ms frame, power consumption will drop to a level proportional to the average bit rate.

To further reduce power consumption, sections of the circuitry are not clocked rather than just held in reset during states when they are not needed. For example, while a receiver is searching for a frame, the CRC checker or that receiver is disabled and not clocked. The internal 16-bit bus remains idle until the host requests a read or write cycle or until a DMA cycle is requested.

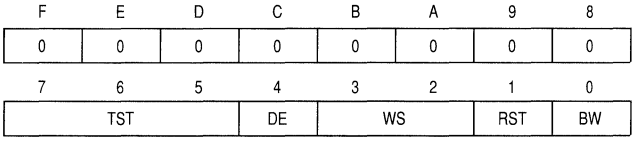
Three separate V_{DD}/V_{SS} pairs are provided. Each powers a separate section of the chip. Section 4.3 details which pins power which sections. It is expected that the user will utilize good grounding and power decoupling techniques when designing the DDLC into a system.

SECTION 3
REGISTER DESCRIPTIONS

The DDLC has several user accessible registers. These registers control the blocks or indicate status. Other registers, used by the DMA section, are used as buffer descriptors and counters. This section provides a description of the registers and the bits in those registers. The address for each register is the hexadecimal offset from the base address of the chip select. The registers may be accessed as 8-bit registers. All status bits, except where noted, may be written with '1' to generate an interrupt. This is useful for system debugging.

3.1 SYSTEM CONTROL REGISTER (Address 00 hex)

This register holds the various bits which control the MPU interface and test functions. When read as a 16-bit register, the most significant byte is 00. This register is cleared to 0000 hex after a reset.



BW **Bus Width** — This read-only bit is a status bit which indicates the data bus width. When this bit is '0', the data bus is 8-bits wide and the upper data byte pins are disabled. When this bit is '1', the data bus is 16-bits wide. Eight-bit operation is selected by connecting the D8 pin to V_{SS}. 16-bit operation is selected by writing FFFF to address 6. The mode is then permanently latched. Only hardware reset can clear 16-bit operation.

RST **Master Reset** — This bit resets the DDLC when set to '1'. A software reset sequence begins when the write cycle is completed. This bit is reset to '0' after the sequence is completed. Section 5.4 describes the sequence.

WS **Wait Select** — These bits choose the number of wait states the DMA controller inserts into direct memory access cycles. The bits are encoded as:

b3	b2	Wait States
0	0	0
0	1	1
1	0	2
1	1	4

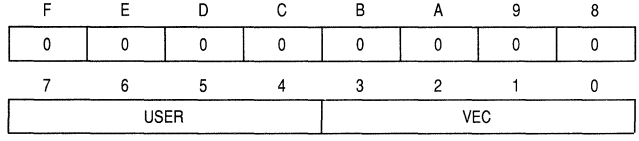
Section 5.1.3.4 describes the operation of the wait state generator.

DE **DTACK Enable** — When this bit is '0', the $\overline{\text{DTACK}}$ signal is ignored so DMA accesses are completed after the programmed wait states have terminated. When this bit is '1', DMA accesses are completed after the programmed wait states have terminated AND the $\overline{\text{DTACK}}$ pin is asserted. The $\overline{\text{DTACK}}$ pin operates normally in the 68000 host processor mode as an output regardless of the setting of this bit. Section 5.1.3.4 discusses $\overline{\text{DTACK}}$ functionality.

TST **Test0-Test2** — These bits control factory test modes are not user accessible.

3.2 MASTER STATUS REGISTER (INTERRUPT VECTOR) (Address 02 hex)

This register contains a code indicating the highest priority interrupt source at the time that this register is read. When an interrupt acknowledge cycle is executed, the contents of the least significant byte are placed on the data bus. When read as a 16-bit register, the upper byte reads 00. After a reset, this register is cleared to 0000 hex.



VEC **Vector** — These bits contain the interrupt source coded as follows:

b3	b2	b1	b1	
0	0	0	0	No Interrupt
0	0	1	0	SCP Complete
0	0	x	1	Timer Interrupt
0	1	x	x	Bit Handler Normal Operation
1	0	x	x	Bit Handler Fault
1	1	x	x	System Fault

x indicates that the respective bit may be 0 or 1 as described below.

b0 — This bit is set to '0' when the interrupting source is a receiver and a '1' when the source is a transmitter.

b1 — This bit indicates the channel which caused the interrupt. An interrupt from channel 0 causes the bit to be set to '0' and channel 1 causes the bit to be set to '1'.

USER **User Programmable** — These bits are user programmable so the programmer may place the DDLC's vectors anywhere in the vector space.

Note: After reset this register reads 00 hex. As such, the DDLC **does not** generate an uninitialized interrupt (0F hex) as some 68000 family peripherals do.

3.3 INTERRUPT ENABLE REGISTER (Address 04 hex)

This register controls the generation of interrupts to the host. If a bit is set, the respective interrupt is enabled. In some cases, a bit controls the interrupts for both channels. In every case, even if an interrupt is disabled, the status bit associated with the disabled interrupt will reflect the current state of the DDLC. If an interrupt is disabled, it will not be encoded into the vector.

F	E	D	C	B	A	9	8
SCPC	CD1	CD0	RI1	RI0	BOV	FUN	FOV
7	6	5	4	3	2	1	0
TFC1	TFC0	TDC1	TDC0	RDC1	RDC0	AE	BE

BE — **Bus Error** — When this bit is set to '1', bus errors from all four channels can generate interrupts. When this bit is cleared to '0', all bus error interrupts are disabled. Each DMA channel can generate a separate bus error interrupt which is encoded in the vector number.

AE — **Address Error** — When this bit is set to '1', address errors from all four channels can generate interrupts. When this bit is cleared to '0', all address error interrupts are disabled. Each DMA channel can generate a separate address error interrupt which is encoded in the vector number.

RDC0 — **Rx Ch 0 DMA Complete** — When this bit is set to '1', an interrupt is queued when the DMA controller has transferred a complete data frame to memory from Rx channel 0. When this bit is cleared to '0', the interrupt is withheld. This bit should be cleared when transparent operation is used. Use the Buffer Overrun interrupt to indicate that a buffer has filled. In transparent operation this interrupt is not generated.

RDC1 — **Rx Ch 1 DMA Complete** — When this bit is set to '1', an interrupt is queued when the DMA controller has transferred a complete data frame to memory from Rx channel 1. When this bit is cleared to '0', the interrupt is withheld. This bit should be cleared when transparent operation is used. Use the Buffer Overrun interrupt to indicate that a buffer has filled. In transparent operation this interrupt is not generated.

TDC0 — **Tx Ch 0 DMA Complete** — When this bit is set to '1', an interrupt is queued when the DMA controller has transferred the last byte of a buffer to the channel 0 Tx FIFO. When this bit is cleared to '0', the interrupt is withheld.

TDC1 — **Tx Ch 1 DMA Complete** — When this bit is set to '1', an interrupt is queued when the DMA controller has transferred the last byte of a buffer to the channel 1 Tx FIFO. When this bit is cleared to '0', the interrupt is withheld.

TFC0 — **Tx Ch 0 Frame Complete** — When this bit is set to '1', an interrupt is queued when the channel 0 bit handler has transmitted the last zero of the closing flag of a complete data frame. When this bit is cleared to '0', the interrupt is withheld. This bit when set modifies the operation of the transmitter state machine during abort or collision events. See Sections 2.1.1.2 and 2.1.1.3.1 for a further discussion.

TFC1 **Tx Ch 1 Frame Complete** — When this bit is set to '1', an interrupt is queued when the channel 1 bit handler has transmitted the last zero of the closing flag of a complete data frame. When this bit is cleared to '0', the interrupt is withheld. This bit when set modifies the operation of the transmitter state machine during abort or collision events. See Sections 2.1.1.2 and 2.1.1.3.1 for further discussion.

FOV **Rx FIFO Overrun** — When this bit is set to '1', an interrupt is queued if either channel 0 Rx FIFO or channel 1 Rx FIFO overruns. The channel which generated the interrupt is encoded in the vector number. When this bit is cleared to '0', an interrupt from either receive channel's FIFO is withheld.

FUN **Tx FIFO Underrun** — When this bit is set to '1', an interrupt is queued if the channel 0 Tx FIFO or channel 1 Tx FIFO underruns. The channel which generated the interrupt is encoded in the vector number. When this bit is cleared to '0', an interrupt from either transmit channel's FIFO is withheld.

BOV **Rx Buffer Overrun** — When this bit is set to '1', an interrupt is queued if a received data frame is too large for the active data buffer. The channel which generated the interrupt is encoded in the vector number. When this bit is cleared to '0', a buffer overrun interrupt from either receive channel is withheld. This bit normally is set when transparent operation is selected.

RI0 **Rx Ch 0 Idle** — When this bit is set to '1', an interrupt is enabled when the channel 0 receiver detects 15 or more contiguous '1s' indicating that the receive channel is (or was) idle. When this bit is cleared to '0', the interrupt is withheld.

RI1 **Rx Ch 1 Idle** — When this bit is set to '1', an interrupt is enabled when the channel 1 receiver detects 15 or more contiguous '1s' indicating that the receive channel is (or was) idle. When this bit is cleared to '0', the interrupt is withheld.

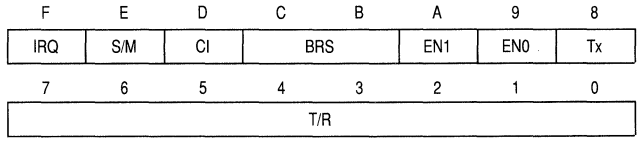
CD0 **Carrier Detect 0** — When this bit is set to '1', an interrupt is enabled when the channel 0 carrier detect pin changes state. When this bit is cleared to '0' the interrupt is withheld.

CD1 **Carrier Detect 1** — When this bit is set to '1', an interrupt is enabled when the channel 1 carrier detect pin changes state. When this bit is cleared to '0' the interrupt is withheld.

SCPC **SCP Complete** — When this bit is set to '1', an interrupt is queued when the SCP module has completed an exchange with an external device. Interrupts may be generated in master or slave mode.

3.4 SCP STATUS/CONTROL REGISTER & Tx/Rx REGISTER (Address 10 hex)

This register contains the SCP Tx/Rx holding register and eight status/control bits. After a reset, the register is cleared to 0000 hex. The Serial Control Port may be used only when the SCP Enable bit in Serial Interface Control Register 0 is set to '1'.



T/R **Tx/Rx Register** — The host writes this 8-bit register with the data to be transferred over the SCP link to an external device. Data is exchanged with the slave device MSB first. Upon completion of the transfer this register contains the data received from the addressed device. If the host reads this register while a transfer is in progress, 00 hex is presented to the host.

Tx **Transmit** — This bit when set to '1' causes the SCP controller to issue eight clock pulses and exchange the data in the Tx/Rx register with the selected slave device's data. This bit is cleared to '0' when the exchange is complete. Note: The host should wait 2 SCP clock periods (as selected by the BRS bits) after this bit is automatically cleared before re-enabling this bit after an SCP transfer. This bit has no function in the slave mode.

EN0 **Enable 0** — This bit directly controls the state of the $\overline{\text{SCP EN}}$ pin. When this bit is set to '1', $\overline{\text{SCP EN}}$ is asserted (low). When this bit is cleared to '0', $\overline{\text{SCP EN}}$ is negated (high). $\overline{\text{SCP EN}}$ may be used as a general purpose output controlled by this bit when the SCP function is not enabled. This bit has no function in the slave mode.

EN1 **Enable 1** — This bit directly controls the state of the $\overline{\text{SCP EN}}$ (Ch 1 $\overline{\text{CD}}$) pin while the SCP Enable bit in Serial Interface Control Register 1 (40 hex) is set. While this bit is set to '1', $\overline{\text{SCP EN}}$ is asserted (low). While this bit is cleared to '0', $\overline{\text{SCP EN}}$ is negated (high). When the SCP Enable bit in Serial Interface Control Register 1 is negated, this bit has no function. This bit operates in master and slave mode.

BRS **Baud Rate Select** — These bits select the SCP clock frequency by choosing a tap from the master clock input (MCLK) divide chain. The output of this clock selector is also used for the two timers. See Section 2.7. The divide ratios are encoded as follows:

b11	b10	MCLK Divider
0	0	÷2
0	1	÷4
1	0	÷8
1	1	÷16

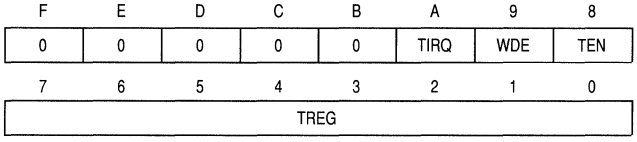
CI **Clock Invert** — This bit inverts the SCP clock polarity. While this bit is low, transmitted data bits shift on rising clock edges and received bits are sampled on falling edges. When the SCP is idle, the clock is low. While this bit is high, transmitted data bits are shifted on falling edges and received bits are sampled on rising edges. When the SCP is idle, the clock is high.

S/M **Slave/Master Control** — This bit selects master or slave mode. When this bit is '0' the master mode is selected. When '1', the slave mode is selected. See Section 6.4 for a further discussion of SCP modes.

IRQ **SCP Interrupt Status** — This bit indicates the current status of the SCP interrupt. When a transfer is complete this bit is set. This bit must be reset to '0' by the host only after it was read with the value '1'. It must be reset to '0' by the host to clear this interrupt.

3.5 TIMER REGISTER
(Address 12 — Channel 0)
(Address 14 — Channel 1)

These registers contain a timer for each channel. The timer is a simple count-down timer which generates an interrupt when the count reaches FE hex. The clock source for the timers is the SCP clock divided by 1024. This generates maximum interrupt intervals of up to 500 ms with an 8 MHz MCLK. The resolution of the counter varies from 250 µs to 2 ms depending on the prescaler tap. The timers can be read or written at any time. Note: If a count-down pulse occurs while a timer register is being accessed (read or write), the pulse is ignored.



TREG **Timer Register** — These bits are the eight-bit counter register. They may be read or written at any time. The contents of this register are unaffected by reset and contain unknown data at power-up.

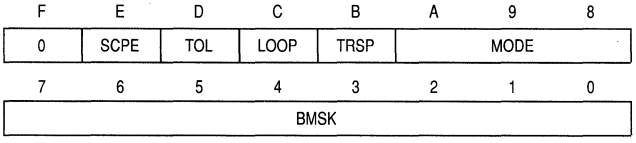
TEN **Timer Enable** — When this bit is asserted to '1', the timer is enabled and will generate an interrupt when the counter reaches FE hex. When this bit is cleared to '0', the timer is disabled and cannot generate interrupts. This bit is cleared by reset.

WDE **Watchdog Enable (Timer 0 Only)** — This bit enables the watchdog function on timer 0. When the watchdog is enabled, an extra ÷16 is inserted into the prescaler chain. Timeouts up to 8 seconds may be generated with the watchdog. This bit is cleared by reset.

TIRQ **Timer Interrupt Status** — This bit indicates that the timer generated an interrupt. This bit may be reset to '0' by the host only after it was read with the value '1'. It must be reset to '0' by the host to clear this interrupt.

3.6 SERIAL INTERFACE CONTROL REGISTER
(Address 20 hex — Channel 0)
(Address 40 hex — Channel 1)

This register contains the control bits for the serial interface blocks for each transceiver channel and is cleared to 0000 hex after a reset.



BMSK **Bit Mask** — These bits select the individual bit positions in an 8-bit IDL B-channel or timeslot that are active. Each bit in this field corresponds directly to a bit in the timeslot. Each bit which is asserted to '1' is disabled and each bit cleared to '0' is enabled. **Note:** Serial transmission is LSB first so the LSB of this field corresponds to the first bit of the B-channel timeslot. This register is also used in the timeslot or IDL transparent mode. See Section 6.1.3 for further discussion.

MODE **Serial Mode Select** — These bits select the mode of operation for the serial interface. For a detailed description of the operation of these modes, refer to Section 6. **Note:** The transparent mode should be initiated and terminated only while the transmitter and receiver enable bits are negated. See Sections 2.1.3, 2.2.3, and 6.1.3 for further discussion.

The mode select bits are encoded as follows:

b10	b9	b8	Mode
0	0	0	Timeslot
0	0	1	IDL B1-Channel
0	1	0	IDL B2-Channel
0	1	1	IDL D-Channel
1	0	0	Modem
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

TRSP **Transparent Mode** — This bit when asserted to '1' enables the transparent mode for both the transmitter and receiver. All HDLC framing and deframing circuitry is bypassed in the transparent mode. When this bit is cleared to '0', normal operation resumes.

LOOP **Loopback** — While this bit is asserted to '1', a loopback from the transmitter to receiver is enabled. While in loopback, the receiver input pin is disabled and the output of the transmitter is connected to the receiver. The transmit pin of the affected channel is controlled by the Transmit on Loop bit (b13). When this bit is cleared to '0', loopback is disabled and normal operation resumes. **Note:** Loopback should be initiated and terminated only while the transmitter and receiver enable bits are negated. Timing of the transmit and receive channels must be identical for correct loopback operation. Provision must be made to assert appropriate handshake inputs for loopbacks to fuction.

TOL **Transmit on Loop** — While this bit is asserted to '1', the transmitter pin transmits data normally while in loopback. When this bit is cleared to '0', the transmitter pin is forced to '1' during loopback.

SCPE **SCP Enable (Channel 0 Only)** — This bit, while asserted to '1' enables the Serial Control Port. The channel 0 $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ pin assume the SCP functions.

SCP Enable (Channel 1 Only) — This bit, while asserted to '1' changes channel 1 $\overline{\text{CD}}$ into a general purpose output controlled by the EN1 bit in the SCP Control register. This may be independent of SCP operation.

3.7 TRANSMIT CONTROL REGISTER
(Address 22 hex — Channel 0)
(Address 42 hex — Channel 1)

This section describes two registers, one for each channel. Each bit performs an identical function for both channels. After reset, this register is cleared to 0000 hex.

F	E	D	C	B	A	9	8
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
0	DI	FA	MD	FCE	ITF	BR	TE

TE **Transmit Enable** — This bit when asserted to '1' activates the transmitter. If TE is negated to '0' while a frame is in transmission, the transmitter immediately flushes the FIFO and enters the idle state. The DMA pointers are reset to their initial values and the DMA controller is disabled for that channel. This is equivalent to resetting the transmitter. During the time when TE is '0' the TxD pin is high impedance. This bit directly controls the $\overline{\text{RTS}}$ pin. When TE is high $\overline{\text{RTS}}$ is low and vice versa. When ISDN D-channel operation is selected the DREQ pin is asserted and negated on a frame-by-frame basis as frames are ready for transmission.

BR **Buffer Ready** — This bit when asserted to '1' by the host indicates that a buffer is ready for transmission and allows the DMA controller to service this channel. The DMA controller fills the transmit FIFO and the transmitter begins outputting data when DGNT ($\overline{\text{CTS}}$) is asserted. When data in the buffer has been completely transferred to the FIFO, this bit is cleared by the DMA controller and an interrupt is queued. If the host clears this bit while a frame is in transmission, DMA activity on the channel stops. Sometime alter, the FIFO will underrun and the frame is aborted. If a bus error or address error is detected during a DMA cycle involving this transmit channel, this bit is cleared by the DMA controller disabling further DMA activity on this channel. The buffer pointers will remain where they were at the time of the system error.

ITF **Interframe Time Fill** — This bit controls the inter-frame time fill character to be transmitted. While this bit is set to '1', marks ('1s') are transmitted during inter-frame periods. Marks idle is used for LAPD operation. While this bit is '0', the transmitter fills inter-frame periods with flag characters (7E hex). Flags idle is used in X.25 operation.

FCE **Force CRC Error** — While this bit is asserted to a '1', transmitted frames contain forced CRC errors. This feature is useful for system debugging. Normally, this bit is cleared so normal CRCs are transmitted.

MD **Mask Drive** — This bit controls the drive characteristics of masked-off B-channel or timeslot bits. While this bit is set to '1', masked bits are driven to a logic 1. While cleared to a '0', masked bits are high-impedance. This feature allows other devices to share a timeslot on an IDL or PCM highway bus.

FA **Force Abort** — When this bit is asserted to a '1', an abort character is transmitted and the transmitter returns to the idle state after the abort was transmitted. This bit is reset by the transmitter after the abort character is sent. See Section 2.1.1.2 for a description of the preferred abort sequence.

DI **Data Invert** — When this bit is asserted to a '1', all transmitted data is inverted. Zeros become ones and ones become zeros. This feature is useful in DMI mode 2 and 3 operation. When this bit is cleared to a '0', data is transmitted normally.

3.8 RECEIVE CONTROL REGISTER (Address 24 — Channel 0) (Address 44 — Channel 1)

This section describes two registers, one for each channel. Each bit performs an identical function for both channels. These registers are cleared to 0000 hex after a reset.

F	E	D	C	B	A	9	8
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
0	DI	ABS	ACE	RCE	BBR	BAR	RE

RE **Receive Enable** — This bit when asserted to '1', enables the receiver. When the HDLC mode is operating, the flag search state is entered. In transparent mode the receiver immediately begins accepting data and places it in the FIFO. IF a bus error or address error is detected by the DMA controller this bit is cleared. This is equivalent to a receiver reset.

BAR **Buffer A Ready** — This set-only bit controls DMA activity on Buffer A's descriptor. When asserted to '1' by the host, it indicates that Buffer A is ready to accept data frames and that DMA activity on this buffer may begin. After a complete frame has been received and placed in memory, this bit is negated to '0' by the DDLC's DMA controller and an interrupt is queued. When the receiver is disabled, this bit is cleared by the receive state machine.

Note: The A and B buffers may be used in a ping-pong fashion so that while a frame is being received, an alternate buffer is prepared for the next frame. Continuous data in the transparent mode or back-to-back frames in the HDLC mode are easily accommodated with this technique. Once the programmer has set either of these bits, the buffer descriptors associated with these bits should not be disturbed until the Rx DMA Complete interrupt is generated and all DMA activity on that buffer has stopped. The receiver will place data in whichever buffer (A or B) is ready, and if both are ready, Buffer A is filled. When re-enabling one of these bits after preparing a new buffer descriptor, the alternate bit should be cleared to '0'. Because these bits are set-only, writing a '0' ensures that the alternate buffer is not accidentally enabled, and if it is already enabled, there will be no change.

RCE **Receive Enable on CRC Error** — Normally, if a CRC error is detected in a frame, the DDLC discards the frame in error. The receive buffer is cleared and reused for the next data frame. When debugging a system, however, it is useful to examine frames with CRC errors to help locate the source of the errors. This bit when asserted to '1' enables data frames to be placed in memory in the presence of CRC errors. The CRC Error bit in the Receive Status register indicates that the received frame had a CRC error. When this bit is negated to '0', all received frames with CRC errors are discarded and the buffer descriptor is reused on the next received frame. In all cases, independent of the setting of this bit, when a frame is received with a CRC error, the error is logged in the CRC Error Log register.

ACE **Address Compare Enable** — In the HDLC mode, when this bit is asserted to '1', the address field of a frame is compared against the contents of the Address Compare registers. The Address Byte Select bit selects which address field is tested. A data frame with an address that matches one of these addresses is passed to memory normally. If the address does not match, the FIFO is cleared, the DMA pointers are reset to their initial values and the receiver enters the flag search state to look for a new frame. Note that the LAPD TEI Broadcast Address (FF hex) is hardwired into the address compare circuitry so three addresses can be tested. If comparison of the address field is not desired, negating this bit to '0' disables address filtering. This bit has no function in the transparent mode.

ABS **Address Byte Select** — This bit selects the octet in the received data frame which is tested by the address compare circuitry. While this bit is '1', the first octet following the opening flag is tested. While this bit is '0', the second octet following the opening flag is tested. While the Address Compare Enable bit is '0' this bit has no significance.

DI **Data Invert** — When this bit is asserted to '1', all received data is inverted. Zeros become ones and ones become zeros. When this bit is cleared to '0', data is received normally.

3.9 TRANSMIT STATUS REGISTER (Address 26 hex — Channel 0) (Address 46 hex — Channel 1)

This section describes two registers, one for each channel. The two channels operate identically.

F	E	D	C	B	A	9	8
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
0	0	FU	TFC	CTS	BE	AE	TDC

TDC **Transmit DMA Complete** — When the DMA controller has transferred the last byte of a buffer into the FIFO, it asserts this bit to '1' and an interrupt is queued. If the host wishes to send back-to-back frames, another buffer can be prepared and activated before the FIFO goes empty and the transmitter begins transmitting inter-frame time fill characters. This bit must be reset to '0' by the host to clear the interrupt from this channel.

AE — **Address Error** — This bit when asserted to '1' by the DMA controller indicates that an address error occurred when the DMA controller had ownership of the host's bus. An address error occurs when the $\overline{CS}$ pin or $\overline{IACK}$ pin is asserted while a DMA cycle is in progress. If an address error occurs, DMA activity on this channel is suspended and an interrupt is queued. This bit must be reset to '0' by the host to clear the interrupt from this channel.

BE — **Bus Error** — This bit when asserted to '1' by the DMA controller indicates that a bus error occurred when a DMA transaction was active on the respective channel. A bus error occurs when the $\overline{BERR}$ pin is activated while the DMA controller has ownership of the host's bus. When a bus error is detected, the active BR bit is reset and DMA activity on that channel is suspended. This bit must be reset to '0' by the host after it is read to clear the bus error interrupt from this channel. **Note:** This function is not available in 80186 mode operation.

CTS — **$\overline{CTS}$ Status** — This bit indicates the current status of the $\overline{CTS}$ (DGNT) pin associated with each channel. The $\overline{CTS}$ (DGNT) pin is sampled on every falling edge of the MCLK. This pin when set to a '1' indicates that the pin ($\overline{CTS}$ and DGNT), depending on the mode is asserted. $\overline{CTS}$ is an active low signal. DGNT is an active high signal. **Note:** The state of this bit can change immediately after it is read.

TFC — **Transmit Frame Complete** — Only when a data frame has been successfully transmitted by the HDLC transmitter and the inter-frame fill state has been entered is this bit asserted to a '1' by the transmitter and an interrupt is queued. If a data frame is aborted by the Layer One device, as in an ISDN S/T Bus D-channel collision, the DDLC will retransmit the aborted frame without the host processor's intervention. This bit must be reset to '0' by the host to clear the interrupt from this channel. See Section 2.1.1.3.1 for a further discussion.

FU — **FIFO Underrun** — This bit when asserted to '1' by the transmitter indicates that the Tx FIFO had an underrun. An interrupt is queued when this bit is set. If an underrun occurs, the frame in transmission is aborted (see Section 2.1.1.2). The BR bit is negated suspending DMA activity on that channel. This bit must be reset to '0' by the host to clear the interrupt from this channel.

3.10 RECEIVE STATUS REGISTER
(Address 28 hex — Channel 0)
(Address 48 hex — Channel 1)

This section describes two registers, one for each channel. Each bit performs an identical function for each channel.

F	E	D	C	B	A	9	8
CD	RI	CE	CDIRQ	FO	RC		

7	6	5	4	3	2	1	0
0	0	BE	AE	BBO	BAO	RBC	RAC

RAC **Rx Buffer A Complete** — This bit when asserted to '1' by the DMA controller indicates that the receiver has successfully received and placed an entire data frame in memory. An interrupt is queued when this bit is set. This bit must be reset to '0' by the host to clear the interrupt from this channel. This function is not used in transparent operation.

RBC **Rx Buffer B Complete** — This bit when asserted to '1' by the DMA controller indicates that the receiver has successfully received and placed an entire data frame in memory. An interrupt is queued when this bit is set. This bit must be reset to '0' by the host to clear the interrupt from this channel. This function is not used in transparent operation.

BAO **Buffer A Overrun** — If the receive data frame exceeds the length of the Receive Buffer A, the DMA controller sets this bit to a '1' indicating that the received frame was too large for the buffer. If Buffer B is enabled, it is activated and the overflow data is placed in that buffer. The BAR bit is cleared by the DMA controller and further DMA activity on this buffer is suspended after the overrun is detected. This bit must be reset to '0' by the host to clear the interrupt from this channel. In transparent operation this bit indicates that the buffer was completely filled. If Buffer B is available and data continues, it is activated and further data is placed there.

BBO **Buffer B Overrun** — If the receive data frame exceeds the length of the Receive Buffer B, the DMA controller sets this bit to a '1' indicating that the received frame was too large for the buffer. If Buffer A is enabled, it is activated and the overflow data is placed in that buffer. The BBR bit is cleared by the DMA controller and further DMA activity on this buffer is suspended after the overrun is detected. This bit must be reset to '0' by the host to clear the interrupt from this channel. In transparent operation that bit indicates that the buffer was complete filled. If Buffer A is available and data continues, it is activated and further data is placed there.

AE **Address Error** — This it when asserted to '1' by the DMA controller indicates that an address error occurred while the DMA controller had ownership of the host's bus. An address error occurs when the $\overline{CS}$ pin or $\overline{IACK}$ pin is activated while a DMA cycle is in progress. If an address error occurs, the DMA controller clears the active Buffer Ready (BAR or BBR) bit, DMA activity on that buffer is suspended and an interrupt is queued. This bit must be reset to '0' by the host to clear the interrupt from this channel.

BE **Bus Error** — This bit when asserted to '1' by the DMA controller indicates that a bus error occurred while a DMA transaction was active on the respective channel. A bus error occurs when the $\overline{BERR}$ pin is activated while the DMA controller has ownership of the host's bus. When a bus error is detected, the DMA controller clears the active Buffer Ready (BAR or BBR) bit and DMA activity on that channel is suspended. This bit must be reset to '0' by the host to clear the interrupt from this channel. Note: This function is not available in 80186 mode operation.

RC **Residue Count** — These three bits indicate the number of residue bits received. These bits have significance only when the RAC or RBC bits are set by the receiver. If both RAC and RBC are at a '1' level at the same time, the frame which had a non-zero residue count cannot be determined. The user must read the residue count soon after the end-of-frame interrupt as receiving another short frame may cause RC to be overwritten.

FO **FIFO Overrun** — This bit, when asserted to a '1', indicates a receiver FIFO overrun occurred and data was lost. An interrupt is generated when an overrun occurs. The active Buffer Ready (BAR or BBR) bit is cleared and further DMA activity on this channel is suspended. A FIFO overrun can occur only if no buffer is prepared for the data in the host's memory or if the DMA controller cannot gain access to the bus. This bit must be reset to '0' by the host to clear the interrupt from this channel.

CDIRQ **CD IRQ Status** — This bit, when high, indicates that the $\overline{CD}$ pin changed state and that an interrupt was queued. The current status of the $\overline{CD}$ pin is available on bit 15. This interrupt is generated every time the $\overline{CD}$ pin changes state. The $\overline{CD}$ pin is not available on either channel 0 or channel 1 when the SCP function is enabled or when the timeslot mode is selected.

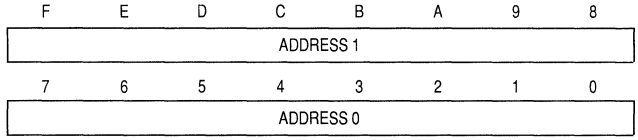
CE **CRC Error** — This bit when asserted to '1' by the receiver indicates that the received frame had a CRC error. This bit only operates when the RCE bit is set to a '1' by the host and has significance only when RAC or RBC are set. If both RAC and RBC are the same logic level at the same time, the frame which had a CRC error cannot be determined.

RI **Receive Idle** — This bit is asserted to '1' by the receiver when 15 or more consecutive '1s' are detected indicating that the channel is (or was) idle. An interrupt is queued when this bit is set. This bit remains set until it is cleared to '0' by the host. If the idle condition remains when this bit is cleared by the host, another interrupt will be generated. The communication channel may change state at any time and may become active immediately after reading this bit.

CD **CD Status** — This bit indicates the current status of the $\overline{CD}$ pin. When the $\overline{CD}$ pin is low this bit is high and vice versa. The $\overline{CS}$ pin is sampled when this register is read. A maskable interrupt is generated every time the $\overline{CD}$ pin changes state. The $\overline{CD}$ pin is not available on either channel 0 or channel 1 when the SCP function is enabled or when the timeslot mode is selected.

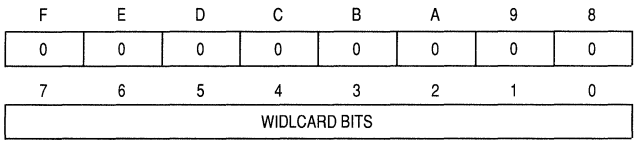
3.11 ADDRESS COMPARE REGISTER (Address 2A hex — Channel 0) (Address 4A hex — Channel 1)

This section refers to two 16-bit registers, one for each channel.



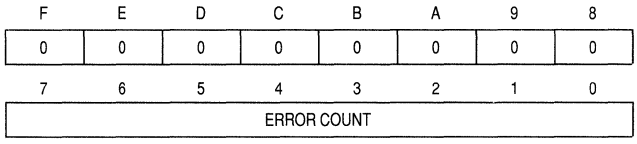
This register contains the two addresses which are compared against incoming data frame address fields. If a true comparison is made, the data frame is passed to the FIFO. Compare Address 1 resides in bit positions b8-b15 and Compare Address 0 resides in bit positions b0-b7 of this register. Note that address FF hex is hardwired into the address compare circuitry effectively providing a third address. See the Address Byte Select bit description in Sections 3.8 and 2.2.1.3 for a further discussion of the address compare feature.

3.12 ADDRESS WILDCARD BIT REGISTER
(Address 2C hex — Channel 0)
(Address 4C hex — Channel 1)



This eight bit register contains the wildcard (don't care) bits for Address Compare 0. Any bit that is set to '1' generates a true result after the comparison. With this feature, blocks of addresses can be compared with only one address specified and certain of the bits ignored. There is a one-for-one correspondence between the bits in the address comparison register and the wildcard register. Recall the data bits are received LSB first.

3.13 CRC ERROR COUNT REGISTER
(Address 2E hex — Channel 0)
(Address 4E hex — Channel 1)



This register contains the current number of receive CRC errors detected since it was reset. Each counter counts up to 255 and stops if more errors are detected. This register is reset to 0000 hex on reset (hardware or software) or any time it is written by the host. The upper byte always reads 00 hex.

3.14 DMA BASE ADDRESS REGISTERS
(Address 32 hex — Channel 0 Transmit)
(Address 38 hex — Channel 0 Receive Buffer A)
(Address 3C hex — Channel 0 Receive Buffer B)
(Address 52 hex — Channel 1 Transmit)
(Address 58 hex — Channel 1 Receive Buffer A)
(Address 5C hex — Channel 1 Receive Buffer B)

These 16-bit registers contain the addresses of the start of DMA buffers. For transmitters, the first two bytes of a frame to be transmitted reside at this location. For receivers, the first two bytes received will be placed in this location. For 16-bit systems, the base address must be **even** addresses (A0 = 0). Care must be taken by the programmer so that these registers are not disturbed while the buffer ready bit associated with that register is set to '1'. If these registers are disturbed while the DMA controller is active, data may be written to unintended areas of memory (on receive channels) or read from unintended areas (transmitters). These registers are not disturbed by a reset. The programmer should place valid addresses in these registers before asserting the associated Buffer Ready bit to prevent DMA activity into unintended areas of memory.

3.15 TRANSMIT FRAME LENGTH REGISTER
(Address 30 hex — Channel 0)
(Address 50 hex — Channel 1)

These 12-bit registers hold the length of the frames to be transmitted in bytes. The host writes this data into the register during the transmit buffer descriptor setup routine. After the length has been written, it should not be disturbed until the transmit complete interrupt is generated. The upper four bits always read 0 hex. These registers are not disturbed by a reset. The programmer must place valid data in this registers before asserting the associated. Buffer Ready bit to present DMA activity into unintended areas of memory.

3.16 RECEIVE BUFFER LENGTH REGISTER
(Address 36 hex — Channel 0)
(Address 56 hex — Channel 1)

These 12-bit registers contain the maximum length of the receive buffers for each channel. The DMA controller compares the current byte count to the value in this register before a DMA request is made and if the byte count is equal to the buffer length than a buffer overrun is declared and an interrupt is generated. Care must be taken by the programmer so that this register is not disturbed while the RE bit of the associated channel is set to '1'. These registers are not disturbed by a reset. The programmer must place valid data in this registers before asserting the associated Buffer Ready bit to prevent DMA activity into unintended areas of memory.

3.17 RECEIVE FRAME LENGTH REGISTERS (READ ONLY)
(Address 3A hex — Channel 0 Buffer A)
(Address 3E hex — Channel 0 Buffer B)
(Address 5A hex — Channel 1 Buffer A)
(Address 5E hex — Channel 1 Buffer B)

These 12-bit read-only registers contain the length in bytes of the received data frames in their associated buffers. These registers are updated on every new byte placed in memory by the DMA controller. When the associated Receive Buffer (BAR or BBR) bit is set and the Receive DMA Complete interrupt is set, this register contains the final length of the frame including the two CRC bytes. If the received frame is not an integral number of octets long, the number of residue bits is found in the Rx Status register. These registers may be read at any time, but they contain meaningful information only after the Rx DMA Complete interrupt has been generated. The upper four bits always read 0 hex. These registers are cleared to 0000 after a reset operation.

3.18 TRANSMIT BYTE COUNT (READ ONLY)
(Address 34 hex — Channel 0)
(Address 54 hex — Channel 1)

These 12-bit read-only registers contain the count of bytes transmitted in the current frame. The count is cleared to '0' at the beginning of a frame. These registers are provided for diagnostic purposes. The upper four bits always read 0 hex. These registers are cleared to 0000 after a reset operation.

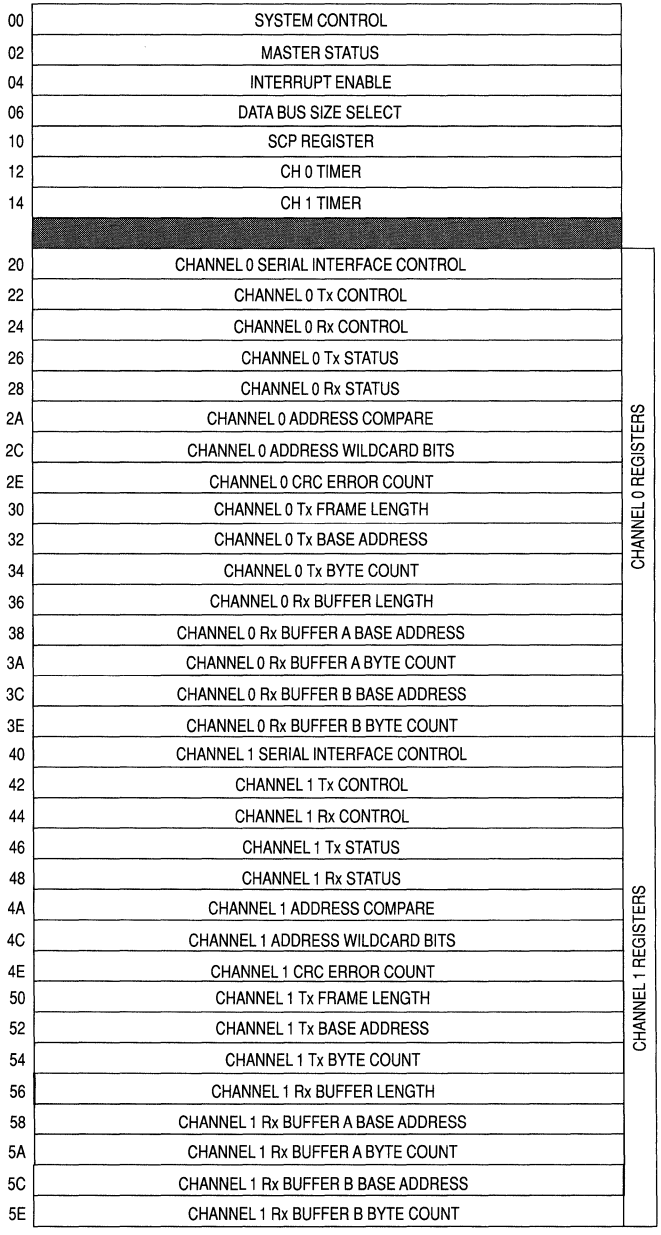


Figure 3-1. Register Memory Map

Table 3-1. Reset Operation Results

Addr (hex)	Register Name	Reset Results				Comments
00	System Control	0000	0000	0000	0000	
02	Master Status	0000	0000	0000	0000	
04	Interrupt Enable	0000	0000	0000	0000	
06	Data Bus Size Select	XXXX	XXXX	XXXX	XXXX	Used to confirm 16 bit bus
10	SCP Register	0000	0000	0000	0000	
12	Channel 0 Timer	0000	0000	XXXX	XXXX	Counter contents unaffected
14	Channel 1 Timer	0000	0000	XXXX	XXXX	Counter contents unaffected
20	Ch 0 Serial Interface Control	0000	0000	0000	0000	
22	Ch 0 Tx Control	0000	0000	0000	0000	
24	Ch 0 Rx Control	0000	0000	0000	0000	
26	Ch 0 Tx Status	0000	0000	0000	0000	
28	Ch 0 Rx Status	0000	0000	0000	0000	
2A	Ch 0 Address Compare	0000	0000	0000	0000	
2C	Ch 0 Address Wildcard Bits	0000	0000	0000	0000	
2E	Ch 0 CRC Error Count	0000	0000	0000	0000	
30	Ch 0 Tx Frame Length	0000	XXXX	XXXX	XXXX	Frame length unaffected
32	Ch 0 Tx Base Address	XXXX	XXXX	XXXX	XXXX	Address unaffected
34	Ch 0 Tx Byte Count	0000	0000	0000	0000	
36	Ch 0 Rx Buffer Length	0000	XXXX	XXXX	XXXX	Buffer length unaffected
38	Ch 0 Rx Buffer A Base Address	XXXX	XXXX	XXXX	XXXX	Address unaffected
3A	Ch 0 Rx Buffer A Byte Count	0000	0000	0000	0000	
3C	Ch 0 Rx Buffer B Base Address	XXXX	XXXX	XXXX	XXXX	Address unaffected
3E	Ch 0 Rx Buffer B Byte Count	0000	0000	0000	0000	
40	Ch 1 Serial Interface Control	0000	0000	0000	0000	
42	Ch 1 Tx Control	0000	0000	0000	0000	
44	Ch 1 Rx Control	0000	0000	0000	0000	
46	Ch 1 Tx Status	0000	0000	0000	0000	
48	Ch 1 Rx Status	0000	0000	0000	0000	
4A	Ch 1 Address Compare	0000	0000	0000	0000	
4C	Ch 1 Address Wildcard Bits	0000	0000	0000	0000	
4E	Ch 1 CRC Error Count	0000	0000	0000	0000	
50	Ch 1 Tx Frame Length	0000	XXXX	XXXX	XXXX	Frame length unaffected
52	Ch 1 Tx Base Address	XXXX	XXXX	XXXX	XXXX	Address unaffected
54	Ch 1 Tx Byte Count	0000	0000	0000	0000	
56	Ch 1 Rx Buffer Length	0000	XXXX	XXXX	XXXX	Buffer length unaffected
58	Ch 1 Rx Buffer A Base Address	XXXX	XXXX	XXXX	XXXX	Address unaffected
5A	Ch 1 Rx Buffer A Byte Count	0000	0000	0000	0000	
5C	Ch 1 Rx Buffer B Base Address	XXXX	XXXX	XXXX	XXXX	Address unaffected
5E	Ch 1 Rx Buffer B Byte Count	0000	0000	0000	0000	

NOTES

1. X indicates that the bit is unaffected by Reset. After power-up, the value is unknown.
2. This table applies to a hardward reset by the $\overline{RST}$ pin or a software reset by the MRST bit in the Master Status/Control Register

SECTION 4
SIGNAL DESCRIPTIONS

This section describes the input and output signals of the DDLC. Active low signals are indicated by an overbar (¯), e.g., $\overline{RST}$. Many of the signal pins are multiplexed with several functions available depending on the operating mode. Each multiplexed pin is described in its various mode settings separately. Pin numbers referenced in the following discussion refer to PLCC package pins.

4.1 MPU INTERFACE PINS

This section describes the operation of the MPU interface pins. Since the DDLC has two modes of operation, Motorola 68000 and Intel 80186, both descriptions will be provided. In most cases, the pin names vary depending on the mode. Table 4-4 at the end of this section is a complete cross-reference of pin names and aliases for the various operating modes. For a complete discussion of both 68000 and 80186 functional timing see Section 5. Parametric specifications are presented in Section 8.

4.1.1 68000 Operation

This mode is selected by connecting a pull-up resistor to $\overline{BGACK}$. A logic high value is internally latched when the DDLC exits the reset state. Any devices connected to $\overline{BGACK}$ should not be actively pulling down during reset. Table 4-1 describes 68000 mode pin functions.

Table 4-1. 68000 Mode Pin Functions

Signal Name	Mnemonic	Input/Output	Active Sate	Driver Type
Address Bus	A1-A15	Input/Output		Three-State
Data Bus	D0-D15	Input/Output		Three-State
Master Clock	MCLK	Input		
Chip Select	$\overline{CS}$	Input	Low	
Address Strobe	$\overline{AS}$	Input/Output	Low	Three-State*
Upper Data Strobe	$\overline{UDS}$	Input/Output	Low	Three-State*
Lower Data Strobe	$\overline{LDS}$	Input/Output	Low	Three-State*
Read/Write	R/$\overline{W}$	Input/Output	High/Low	Three-State*
Data Transfer Acknowledge	$\overline{DTACK}$	Input/Output	Low	Three-State*
Bus Request	$\overline{BR}$	Output	Low	Open-Drain
Bus Grant	$\overline{BG}$	Input	Low	
Bus Grant Acknowledge	$\overline{BGACK}$	Input/Output	Low	Three-State*
Bus Owner 0	$\overline{OWN\ 0}$	Output	Low	Three-State*
Bus Owner 1	$\overline{OWN\ 0}$	Output	Low	Three-State*
Interrupt Request	$\overline{IRQ}$	Output	Low	Open-Drain
Interrupt Acknowledge	$\overline{IACK}$	Input	Low	
Reset	$\overline{RST}$	Input/Output	Low	Open-Drain
Bus Error	$\overline{BERR}$	Input	Low	

*These pins have an internal pull-up device momentarily activated immediately before going high-impedance for improved risetime.

4.1.1.1 ADDRESS AND DATA BUS

A1-A15 — Address Bus

These bidirectional pins are the address lines. When the DDLC is a bus master these pins are outputs. They present the address of the memory location to be read or written. When the DDLC is a bus slave, these pins are inputs. Internal registers are accessed by placing the address of the selected register on the bus and then reading or writing the register's contents. When these pins are inputs, only the lower six bits are decoded and the upper nine bits are thus "don't care".

D0-D15 — Data Bus

These bidirectional pins are data lines. They are inputs when the host owns the bus and writes to the DDLC or when the DDLC owns the bus and reads a memory location. They are outputs when the host owns the bus and reads the DDLC or when the DDLC owns the bus and writes to memory. During an IACK cycle, pins D0-D7 present the interrupt vector number to the host with timing identical to an odd address read. For operation with the 8-bit 68008 bus, see $\overline{\text{UDS}}$ and $\overline{\text{LDS}}$ pin descriptions below.

4.1.1.2 CONTROL BUS

R/$\overline{\text{W}}$ (Read/Write)

This bidirectional pin controls the direction of data flow on the data pins. High indicates that the bus cycle is a read cycle and low indicates a write cycle. While the host owns the bus this pin is an input. While the DDLC owns the bus, this pin is an output. At the end of a DMA cycle, this pin is pulled high for 1/2 MCLK cycle before going high-impedance.

$\overline{\text{AS}}$ (Address Strobe)

This bidirectional pin is an output while the DDLC owns the bus. It is the address strobe which indicates while active, that there is a valid address on the bus. During a bus arbitration cycle after $\overline{\text{BG}}$ has been asserted, $\overline{\text{AS}}$ is an input which is used along with $\overline{\text{BGACK}}$ to determine if the DDLC has been given ownership of the bus. At the end of a DMA cycle, this pin is pulled high for 1/2 MCLK cycle before going high-impedance.

$\overline{\text{UDS}}$ and $\overline{\text{LDS}}$ (Upper and Lower Data Strobes)

These bidirectional pins are outputs when the DDLC owns the bus and inputs while the host owns the bus. These pins are data strobes which indicate that data on the bus is valid. When $\overline{\text{UDS}}$ is low it indicates that data on the bus in the upper byte (D7-D15) is valid. When $\overline{\text{LDS}}$ is low it indicates that data on the bus in the lower byte (D0-D7) is valid. When both $\overline{\text{UDS}}$ and $\overline{\text{LDS}}$ are low, all 16-bits on the data bus are valid and a word transaction is performed. Data is latched into the DDLC on the rising edge of these signals. In the 8-bit bus mode, $\overline{\text{UDS}}$ becomes the A0 (Address 0) pin and $\overline{\text{LDS}}$ becomes the $\overline{\text{DS}}$ (Data Strobe) pin. At the end of a DMA cycle, these pins are pulled high for 1/2 MCLK cycle before going high-impedance.

$\overline{\text{DTACK}}$ (Data Transfer Acknowledge)

This bidirectional pin is an output when the host owns the bus. It indicates that the DDLC has completed the data transfer, either read or write, and the host processor may terminate the access cycle. While the DDLC owns the bus this pin is an input and is sampled on the rising edge of State 4. If $\overline{\text{DTACK}}$ is detected high (inactive), wait states are inserted. The pin is sampled on every rising edge of MCLK while wait states are inserted until a logic low is found, at which time the DDLC

terminates its access cycle normally. At the end of a DMA cycle, an active pull-up device is momentarily enabled before going high-impedance.

4.1.1.3 BUS ARBITRATION

$\overline{BR}$ (Bus Request)

This active-low open-drain pin is an output indicating that the DDLC is requesting the bus from the present bus owner. It is negated after the DDLC has asserted $\overline{BGACK}$ so the next bus master may begin negotiation for ownership of the bus. In eight-bit 68008 systems, $\overline{BGACK}$ and $\overline{BR}$ should be connected together producing the required $\overline{BR}$ function.

$\overline{BG}$ (Bus Grant)

This input is sampled on MCLK falling edges by the DDLC after $\overline{BR}$ has been asserted. While $\overline{BG}$ is active, it indicates that the bus ownership will be transferred to the DDLC.

$\overline{BGACK}$ (Bus Grant Acknowledge)

This bidirectional pin is an active-low open-drain output while the DDLC is indicating that it has taken ownership of the system bus. Before the DDLC can assume bus ownership, it monitors $\overline{BGACK}$ and $\overline{AS}$ after $\overline{BG}$ has been asserted to a logic low and waits for $\overline{BGACK}$ and $\overline{AS}$ to be negated indicating that the present owner has relinquished control of the bus. An internal pull-up device is momentarily activated when this pin is negated to improve risetime.

This pin is also used to select 68000 or 80186 modes of operation. To select the 68000 mode, a pull-up resistor is connected from this pin to VDD. The level of this pin is internally latched upon exiting the reset state and remains until the DDLC is again reset (hardware or software).

$\overline{OWN}$ 0 (Bus Owner Channel 0)

This active-low output pin indicates that channel 0 of the DDLC has ownership of the bus. It is similar to the $\overline{BGACK}$ pin, but is intended for use by an external function code generator or address expansion circuit. At the end of a DMA cycle, this pin is pulled high for 1/2 MCLK cycle before going high-impedance. This pin is driven high during DMA cycles associated with channel 1.

$\overline{OWN}$ 1 (Bus Owner Channel 1)

This active-low output pin indicates that channel 1 of the DDLC has ownership of the bus. It is similar to the $\overline{BGACK}$ pin, but is intended for use by an external function code generator or address expansion circuit. At the end of a DMA cycle, this pin is pulled high for 1/2 MCLK cycle before going high-impedance. This pin is driven high during DMA cycles associated with channel 0.

4.1.1.4 INTERRUPT SIGNALS

$\overline{IRQ}$ (Interrupt Request)

This active-low open-drain output indicates that the DDLC has a pending interrupt to the host processor. It remains low as long as there is at least one pending interrupt.

$\overline{IACK}$ (Interrupt Acknowledge)

This input is similar to the $\overline{CS}$ pin, except that while it is activated, the interrupt vector number is placed on the data bus in the lower byte (D0-D7). During the IACK cycle, the $\overline{CS}$ or address pins

are not used, but the timing of the vector fetch is the same as a normal odd address read. If $\overline{IRQ}$ is high (no interrupt is pending) while $\overline{IACK}$ is asserted, or if $\overline{IACK}$ and $\overline{CS}$ are active at the same time, $\overline{IACK}$ is ignored.

4.1.1.5 MISCELLANEOUS SIGNALS

MCLK (MPU Clock)

This input is the master clock for the DDLC. All timing states are derived from this clock. Normally, the source of this signal is the microprocessor's master system clock. The DDLC expects a clock signal of nominal 50 percent duty cycle. All other clock inputs to the DDLC may be asynchronous to this clock.

$\overline{BERR}$ (Bus Error)

This input is used to pass the status of the MPU bus to the DDLC while the DDLC owns the bus. It is sampled on every falling edge of MCLK after bus state 2 until $\overline{DTACK}$ is active. If a bus error is detected while the DDLC owns the bus, the DMA channel that was active when the error was found is disabled and an interrupt is queued.

$\overline{RST}$ (Reset)

This bidirectional active-low pin is used to reset all internal state machines and selected registers to know states and values. All bus interface pins go high impedance while the reset line is active. While the watchdog timer is enabled, if the timer underflows to FF hex, this pin becomes an open-drain output and goes low for 16 MCLK cycles. This is intended to reset the entire host system.

$\overline{CS}$ (Chip Select)

This input, while at a logic low, indicates that the base address of the DDLC has been selected and the internal address decode logic selects a register for a read or write access cycle. If this pin is detected active while the DDLC owns the bus, an address error is assumed.

4.1.2 80186 Operation

This mode is selected by connecting $\overline{BGACK}$ (pin 41) to VSS. The logic level of this pin is internally latched when the DDLC exits the reset state. Table 4-2 describes the pin functions in the 80186 mode.

4.1.2.1 ADDRESS AND DATA BUS

A0-A15 — Address Bus

These bidirectional pins are the address lines. While the DDLC is a bus master these pins are outputs. They present the address of the memory location to be read or written. While the DDLC is a bus slave, these pins are inputs. The internal registers are accessed by placing the address of the selected register on the bus and then reading or writing the register's contents. While these pins are inputs, only the lower seven bits are decoded and the upper nine bits are thus "don't care".

D0-D15 — Data Bus

These bidirectional pins are the data lines. They are inputs while the host owns the bus and writes to the DDLC or while the DDLC owns the bus and reads a memory location. They are outputs while the host owns the bus and reads the DDLC or while the DDLC owns the bus and writes to memory. During an IACK cycle, pins D0-D7 present the interrupt vector number to the host.

Table 4-2. 80186 Mode Pin Functions

Signal Name	Mnemonic	Input/Output	Active Sate	Driver Type
Address Bus	A1-A15	Input/Output		Three-State
Data Bus	D0-D15	Input/Output		Three-State
Master Clock	MCLK	Input		
Chip Select	$\overline{CS}$	Input	Low	
Write Strobe	$\overline{WR}$	Input/Output	Low	Three-State*
Address 0	A0	Input/Output		Three-State*
High Byte Enable	$\overline{BHE}$	Input/Output	Low	Three-State*
Read/Write	R/$\overline{W}$	Input/Output	High/Low	Three-State*
Ready	$\overline{RDY}$	Input	Low	
Hold Request	$\overline{HREQ}$	Output	Low	Open-Drain
Hold Acknowledge	$\overline{HACK}$	Input	Low	
80186 Select	$\overline{186}$	Input	Low	
Bus Owner 0	$\overline{OWN\ 0}$	Output	Low	Three-State*
Bus Owner 1	$\overline{OWN\ 1}$	Output	Low	Three-State*
Interrupt Request	$\overline{IRQ}$	Output	Low	Open-Drain
Interrupt Acknowledge	$\overline{IACK}$	Input	Low	
Reset	$\overline{RST}$	Input/Output	Low	Open-Drain
Read Strobe	$\overline{RD}$	Input/Output	Low	Three-State*

*These pins have an internal pull-up device momentarily activated immediately before going high-impedance for improved risetime.

4.1.2.2 CONTROL BUS

$\overline{RD}$ (Read Strobe)

This bidirectional active-low pin indicates that the address on the bus is valid and the bus owner will input data on the rising edge of this signal. This pin is an input while the DDLC is in the host processor mode and an output while in the DMA mode. During DMA operation an internal pull-up device is momentarily activated when this pin is negated to improve risetime.

$\overline{WR}$ (Write Strobe)

This bidirectional active-low pin indicates that the address and data on the bus is valid. The addressed device should latch the data on the rising edge of this signal. While the DDLC is in the host processor mode, this pin is an input. While in the DMA mode, this pin is an output. During DMA operation an internal pull-up device is momentarily activated when this pin is negated to improve risetime.

R/$\overline{W}$ (Read/Write Control)

This three-state output pin is intended to control the direction of external bus transceivers. The DDLC is performing a memory read while this pin is high and a memory write while this pin is low.

This pin has an internal pull-up device activated for 1/2 MCLK cycle before going high-impedance at the end of a DMA access. While the DDLC is not the bus owner this pin is high-impedance.

$\overline{\text{BHE}}$ (High-Byte Enable)

This bidirectional active-low pin indicates that the data on D8-D15 pins is valid. This pin is an input while the DDLC is in the host processor mode and an output in the DMA mode and is used only for 16-bit operation. During DMA operation an internal pull-up device is momentarily activated while this pin is negated to improve risetime.

$\overline{\text{RDY}}$ (Ready)

This active-low pin is an input in the DMA mode. Memory signals that it is ready to complete a DMA cycle by asserting this pin. This pin is not used in the host processor mode.

4.1.2.3 BUS ARBITRATION

$\overline{\text{HREQ}}$ (Hold Request)

This active-low output pin indicates that the DDLC is requesting ownership of the bus. It remains asserted for as long as the DDLC owns the bus, then it is negated.

$\overline{\text{HACK}}$ (Hold Acknowledge)

This active-low input is used to determine when the DDLC has been granted bus ownership. The DMA controller samples this pin while HREQ is asserted, and when found asserted for two sample periods, the DMA cycle begins.

$\overline{\text{OWN}}$ $\overline{0}$ (Bus Owner Channel 0)

This active-low output pin indicates that channel 0 of the DDLC has ownership of the bus. It is intended for use by an external function code generator and address expansion circuitry and remains asserted as long as the DDLC owns the bus. An internal pull-up device is momentarily activated when this pin is negated to improve risetime. This pin is driven high when $\overline{\text{OWN}}$ $\overline{1}$ is active.

$\overline{\text{OWN}}$ $\overline{1}$ (Bus Owner Channel 1)

This active-low output pin indicates that channel 1 of the DDLC has ownership of the bus. It is intended for use by an external function code generator and address expansion circuitry and remains asserted as long as the DDLC owns the bus. An internal pull-up device is momentarily activated when this pin is negated to improve risetime. This pin is driven high when $\overline{\text{OWN}}$ $\overline{0}$ is active.

4.1.2.4 INTERRUPT SIGNALS

$\overline{\text{IRQ}}$ (Interrupt Request)

This active-low open-drain output indicates that the DDLC has a pending interrupt to the host processor. It remains low as long as there is at least one pending interrupt.

$\overline{\text{IACK}}$ (Interrupt Acknowledge)

This input is similar to the $\overline{\text{CS}}$ pin, except that when it is activated, the interrupt vector number is placed on the data bus in the lower byte (D0-D7). During the IACK cycle, the $\overline{\text{CS}}$ or address pins are not used, but the timing of the vector fetch is the same as a normal odd address read. If $\overline{\text{IRQ}}$ is high (no interrupt is pending) when $\overline{\text{IACK}}$ is asserted, or if $\overline{\text{IACK}}$ and $\overline{\text{CS}}$ are active at the same time, $\overline{\text{IACK}}$ is ignored.

4.1.2.5 MISCELLANEOUS SIGNALS

MCLK (Master Clock)

This input is the master clock for the DDLC. All timing states are derived from this clock. Normally, the source of this signal is the microprocessor's master system clock. The DDLC expects a clock signal of nominal 50 percent duty cycle. All other clock inputs to the DDLC may be asynchronous to this clock.

$\overline{\text{RST}}$ (Reset)

This bidirectional active-low pin is used to reset all internal state machines and selected registers to known states and values. All bus interface pins go high impedance while the reset line is active. While the watchdog timer is enabled, this pin becomes an open-drain output if the timer underflows and goes low for 16 MCLK cycles. This is intended to reset the entire host system.

$\overline{\text{CS}}$ (Chip Select)

This input, while at a logic low, indicates that the base address of the DDLC has been selected and the internal address decode logic selects a register for a read or write access cycle. If this pin is detected active while the DDLC owns the bus, an address error is assumed.

4.2 SERIAL COMMUNICATION PINS

Table 4-3 describes the various names of the serial communication pins. The pin number is in the left column and the several operational modes are in the top row. Some pins, primarily the modem control pins change function as the operating modes are changed. While the SCP is enabled, modem control pins on channel 0 assume SCP functions. The SCP can be selected independently of any other mode, except for the transparent modem mode where the byte sync pins become SCP functions. When the SCP is not activated, $\overline{\text{SCP EN}}$ 0 may be used as a general purpose output (GPO).

When the SCP is enabled, it overrides the functions on pins: 50, 53, 54, 57, and 58 as defined by any one of the selected serial interface modes. By enabling the SCP, pin 54 becomes SCP TxD regardless of which serial interface mode is selected. The same occurs with pins 57, 58, and pin 50 for Channel 1. On channel 0 this means that Timeslot mode (because RX and TX enables), IDL D mode (because of DREQ and DGNT) and Transparent Modem mode (because of the BSYNC signals) cannot be used while the SCP is enabled. IDL B mode and Packet Modem modes can still be used.

4.2.1 IDL Bus Mode

In the IDL mode, each channel has six pins allocated to serial communication with an external data transceiver, typically the MC145474 S/T transceiver. Both channel's interfaces operate identically and are described together. Operation in the packet and transparent modes is identical from an interface point of view.

Table 4-3. Serial Interface Pins

PLCC Pin #	IDL B Mode	IDL D Mode	Timeslot Mode	Packet Modem	Transparent Modem	SCP Enabled
Channel 1						
44	IDL SYNC 1 (I)	IDL SYNC (I)	Rx CLK 1 (I)	Rx CLK 1 (I)	Rx CLK 1 (I)	Note (I)
45	IDL CLK 1 (I)	IDL CLK 1 (I)	Tx CLK 1 (I)	Tx CLK 1 (I)	Tx CLK 1 (I)	Note (I)
46	$\overline{\text{CTS}}$ 1 (I)	DGNT 1 (I)	Tx EN 1 (I)	$\overline{\text{CTS}}$ 1 (I)	Tx BSYNC 1 (I)	Note (I)
47	$\overline{\text{RTS}}$ 1 (O)	DREQ 1 (O)	$\overline{\text{RTS}}$ 1 (O)	$\overline{\text{RTS}}$ 1 (O)	$\overline{\text{RTS}}$ 1 (O)	Note (O)
48	IDL Rx 1 (I)	IDL Rx 1 (I)	RxD 1 (I)	RxD 1 (I)	RxD 1 (I)	Note (I)
49	IDL Tx 1 (O)	IDL Tx 1 (O)	TxD 1 (O)	TxD 1 (O)	TxD 1 (O)	Note (O)
50	$\overline{\text{CD}}$ 1 (I)	$\overline{\text{CD}}$ 1 (I)	Rx EN 1 (I)	$\overline{\text{CD}}$ 1 (I)	Rx BSYNC 1 (I)	$\overline{\text{SCP EN}}$ 1 (O)
53	GPO (O)	GPO (O)	GPO (O)	GPO (O)	GPO (O)	$\overline{\text{SCP EN}}$ 0 (I/O)
Channel 0						
54	$\overline{\text{CD}}$ 0 (I)	$\overline{\text{CD}}$ 0 (I)	Rx EN 0 (I)	$\overline{\text{CD}}$ 0 (I)	Rx BSYNC 0 (I)	SCP TxD (O)
55	IDL Tx 0 (O)	IDL Tx 0 (O)	TxD 0 (O)	TxD 0 (O)	TxD 0 (O)	Note (O)
56	IDL Rx 0 (I)	IDL Rx 0 (I)	RxD 0 (I)	RxD 0 (I)	RxD 0 (I)	Note (I)
57	$\overline{\text{RTS}}$ 0 (O)	DREQ 0 (O)	$\overline{\text{RTS}}$ 0 (O)	$\overline{\text{RTS}}$ 0 (O)	$\overline{\text{RTS}}$ 0 (O)	SCP CLK (I/O)
58	$\overline{\text{CTS}}$ 0 (I)	DGNT 0 (I)	Tx EN 0 (I)	$\overline{\text{CTS}}$ 0 (I)	Tx BSYNC 0 (I)	SCP RxD (I)
59	IDL CLK 0 (I)	IDL CLK 0 (I)	Tx CLK 0 (I)	Tx CLK 0 (I)	Tx CLK 0 (I)	Note (I)
60	IDL SYNC 0 (I)	IDL SYNC 0 (I)	Rx CLK 0 (I)	Rx CLK 0 (I)	Rx CLK 0 (I)	Note (I)

NOTE: Functionality of this pine is not affected by SCP enable. (I) indicates input, (O) indicates output.

IDL SYNC 0 (IDL Frame Synchronization Channel 0)
IDL SYNC 1 (IDL Frame Synchronization Channel 1)

This input synchronizes the operation of the serial interface circuitry to the IDL bus. A pulse one IDL clock period wide is presented to this pin every 125 µs. The synchronization pulse is rising edge aligned with the IDL CLK signal.

IDL CLK 0 (IDL Clock Channel 0)
IDL CLK 1 (IDL Clock Channel 1)

This input is a clock signal typically at 1.544, 2.048, or 2.56 MHz although any frequency between 1.5 and 4.5 MHz will operate properly. This signal is expected to be nominally 50% duty cycle. It normally is asynchronous to the MCLK signal.

IDL Rx 0 (IDL Receive Data Channel 0)
IDL Rx 1 (IDL Receive Data Channel 1)

Data from the IDL bus is input on this pin. Data bits are presented to this pin on rising edges of the IDL CLK and latched into the DDLC on falling edges. This pin ignores all bit times other than the ones it is programmed to receive.

IDL Tx 0 (IDL Transmit Data Channel 0)
IDL Tx 1 (IDL Transmit Data Channel 1)

Data is output onto the IDL bus from this pin. Data bits are presented on the rising edges of the IDL CLK. Data bits are presented only in the selected time-slots in the IDL frame. At all other times this pin is high-impedance. When the TE bit is '0' this pin is always high impedance.

DREQ 0 (D-Channel Request Channel 0)
DREQ 1 (D-Channel Request Channel 1)

In the IDL D-channel mode this active-high output indicates to the Layer One transceiver that the DDLC has a frame of data to transmit on the D-channel. When the frame has been transmitted the DDLC negates DREQ to a logic low. If a frame in transmission is aborted by the Layer One device, the DDLC negates DREQ to a logic low and recycles the frame for retransmission. When the frame is again ready for transmission, DREQ is set to a logic one. In the IDL B-channel mode this pin assumes the functionality of RTS. See Section 4.2.3.

DGNT 0 (D-Channel Grant Channel 0)
DGNT 1 (D-Channel Grant Channel 1)

In the IDL D-channel mode this output indicates to the DDLC that the Layer One transceiver is granting permission for the DDLC to transmit on the D-channel. The first bit of the data frame is transmitted during the IDL frame immediately following the assertion of DGNT. If DGNT is found negated while DREQ is asserted, the frame in transmission is aborted and the DMA pointers are reset to prepare for retransmission. In the IDL B-channel mode this pin assumes the functionality of CTS. See Section 4.2.3.

4.2.2 Timeslot Mode

The timeslot mode is compatible with most PBX-type PCM highways. The two channels are independent. Additionally, the transmitter and receiver of each channel may be clocked from a different source. There is no frequency or phase relationship required between transmit and receive clocks. The two channels are identical in operation, so pins of like function are described together. Operation in the packet and transparent modes is identical from an interface point of view.

Tx CLK 0 (Transmit Clock Channel 0)
Tx CLK 1 (Transmit Clock Channel 1)

This signal is the clock for the transmitted data. Data bits are shifted on rising clock edges. This signal must be a continuous clock with a nominal 50 percent duty cycle.

Tx EN 0 (Transmit Enable Channel 0)
Tx EN 1 (Transmit Enable Channel 1)

This signal is an active high enable which synchronizes the transmitter to the eight bit-times of a timeslot. The enable may be of two different types: short-frame and long-frame. In short-frame operation the enable is a positive pulse, one Tx CLK period wide which immediately precedes the eight-bit timeslot. The edges of the pulse are aligned with a rising clock edge. In long-frame operation, the enable is a positive pulse which is eight bit-times wide. The eight data bits of a timeslot are transmitted while the enable is high. Rising and falling edges of the enable are aligned to rising edges of the clock. The DDLC automatically senses either short- or long-frame operation and adjusts its timing for the next eight-bit timeslot. See Section 6.2 for a further discussion.

TxD 0 (Transmit Data Channel 0)
TxD 1 (Transmit Data Channel 1)

This signal is the transmitted data. Data is presented on the eight bits of a timeslot on rising clock edges. During periods outside of the timeslot or when the TE bit is '0' this signal is high-impedance.

Rx CLK 0 (Receive Clock Channel 0)
Rx CLK 1 (Receive Clock Channel 1)

This signal is the clock for the received data. Data bits are sampled on falling clock edges. This signal must be a continuous clock with a nominal 50 percent duty cycle.

Rx EN 0 (Receive Enable Channel 0)
Rx EN 1 (Receive Enable Channel 1)

This signal is an active high enable which synchronizes the receiver to the eight bit-times of a timeslot. Like Tx Enable this signal may be of two different types: short-frame and long-frame. Operation is essentially similar to Tx Enable.

RxD 0 (Receive Data Channel 0)
RxD 1 (Receive Data Channel 1)

Received data is applied to this pin. This pin is sampled on falling edges of the Rx CLK during the timeslot. During all other periods this pin is ignored.

4.2.3 Modem Mode (Packet Operation)

The modem mode is a general purpose interface which is compatible with most modems. Logic levels are inverted from the IDL and timeslot modes because inverting transceivers such as EIA-232 devices are normally used. Operation in the transparent mode is different from packet mode and is described in the next section. As with the timeslot mode, transmit and receive are independent and have no mutual clocking requirements.

Tx CLK 0 (Transmit Clock Channel 0)
Tx CLK 1 (Transmit Clock Channel 1)

This signal is the clock for the transmitter. The transmitter emits a new data bit on every falling clock edge while $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ are both at a logic low. This signal may be any frequency from 0 Hz to MCLK/2.

TxD 0 (Transmit Data Channel 0)
TxD 1 (Transmit Data Channel 1)

This signal is the transmitted data output. Data is output on every falling clock edge while both $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ are both at a logic low. While either $\overline{\text{RTS}}$ or $\overline{\text{CTS}}$ are at a logic high, this pin outputs a logic high.

Rx CLK 0 (Receive Clock Channel 0)
Rx CLK 1 (Receive Clock Channel 1)

This signal is the clock for the receiver and may be any frequency from 0 Hz to MCLK/2.

RxD 0 (Receive Data Channel 0)
RxD 1 (Receive Data Channel 1)

This is the data input for the receiver. It samples this pin on every rising clock edge while the receiver is enabled.

$\overline{\text{KTS}}$ 0 (Request to Send Channel 0)
$\overline{\text{RTS}}$ 1 (Request to Send Channel 1)

This output, while at a logic low, indicates that the transmitter has been enabled by setting the TE bit. Once the transmitter is enabled, this signal remains low until the transmitter is disabled by the TE bit. Transmission begins when both $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ are low and when data is ready.

$\overline{\text{CTS}}$ 0 (Clear to Send Channel 0)
$\overline{\text{CTS}}$ 1 (Clear to Send Channel 1)

The transmitter transmits packet data while this signal is low. If this signal goes high for more than one bit-time, the packet in transmission is aborted and retransmitted in its entirety when $\overline{\text{CTS}}$ again goes low.

$\overline{\text{CD}}$ 0 (Carrier Detect Channel 0)
$\overline{\text{CD}}$ 1 (Carrier Detect Channel 1)

This input is a general purpose input. For modem applications this input is used to indicate when the carrier is active. This pin changes function while the SCP is enabled or while transparent operation is selected. See Sections 4.2.4 and 4.2.5 for a further description.

4.2.4 Modem Mode (Transparent Operation)

When transparent operation is selected five of the seven serial interface pins retain their packet operation functionality. $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ change operation to enable a byte synchronization signal to the transmitter and receiver respectively.

Tx $\overline{\text{BSYNC}}$ 0 (Transmit Byte Sync Channel 0)
Tx $\overline{\text{BSYNC}}$ 1 (Transmit Byte Sync Channel 1)

This input is used to pass byte synchronization information to the transmitter. This pin is sampled on rising Tx CLK edges and when found low for one sample, 1-1/2 Tx CLK edges (on a falling edge) later the first bit of a byte is transmitted. Synchronization is maintained in the absence of subsequent sync signals. If another sync signal is detected, the transmitter resynchronizes to that signal regardless of its alignment to previous sync signals. See Section 6.3 for further details.

Rx $\overline{\text{BSYNC}}$ 0 (Receive Byte Sync Channel 0)
Rx $\overline{\text{BSYNC}}$ 1 (Receive Byte Sync Channel 1)

This input is used to pass byte synchronization information to the receiver. This pin is sampled on rising Rx CLK edges and when found low for one sample, 1-1/2 Rx CLK edges later (on a rising edge), a byte boundary is defined. Rx BSYNC actually resets a modulo-8 counter in the DDLC so synchronization is maintained in the absence of subsequent sync signals. If sync pulses occur more often than every eight Rx CLKs, the internal counter will never generate a load signal to the FIFO and data will never be placed in memory. See Section 6.3 for further details.

4.2.5 Serial Control Port

When the Serial Control Port (SCP) is enabled (SCPE high in the Channel 0 Serial Interface Control register) some serial interface pins change functionality to the SCP mode. When the SCP is enabled, SCP functions override any other selected function. See Section 6.4 for a further discussion.

$\overline{\text{SCP}}$ $\overline{\text{EN}}$ 0 (SCP Enable 0) (Master Mode)
$\overline{\text{Slave Select}}$ (Slave Mode)

In the SCP master mode this pin is an open-drain output. It is normally high except when an SCP transmission is in progress when this pin is programmed to a logic low for the duration of the 8-bit SCP exchange. This pin is controlled by the EN0 bit in the SCP Status/Control register. When EN0 is high, $\overline{\text{SCP}}$ $\overline{\text{EN}}$ 0 is low. When EN0 is low, $\overline{\text{SCP}}$ $\overline{\text{EN}}$ 0 is high. This pin may be used as a general purpose output as it is not multiplexed with any other function and always available.

In the SCP slave mode this pin is an input which indicates that the DDLC has been selected for a transfer. The external SCP master must not enable this pin while the SCP CLK is in the active state (1 while CI is 0 and 0 while CI is 1). An interrupt is generated on the rising edge of this signal.

$\overline{\text{SCP}}$ $\overline{\text{EN}}$ 1 (SCP Enable 1)

This active low output pin is an enable signal for the SCP. This pin is normally high except when an SCP transmission is in progress when this pin goes to a logic low for the duration of the 8-bit SCP exchange. This pin is controlled by the EN1 bit in the SCP Status/Control register. When EN1 is high, $\overline{\text{SCP}}$ $\overline{\text{EN}}$ 1 is low. When EN1 is low, $\overline{\text{SCP}}$ $\overline{\text{EN}}$ 1 is high. This pin is not affected by SCP master or slave operation.

SCP CLK (SCP Clock)

In the master mode this pin is an output which is the clock for the SCP. It is derived from e MCLK pin and the baud rate is selected by the SCP Baud Rate Select bits in the SCP Status/Control register. The polarity of this signal is selectable by the CI bit so data bits may be input or output on either but opposite edges. When the Tx bit is set, a burst of eight clock pulses is generated.

In the slave mode this pin is the clock input to the SCP. Any frequency between 0 Hz and MCLK is acceptable for proper operation. This clock may be continuous or gated.

SCP RxD (SCP Receive Data)

This input latches new SCP data on edges of the SCP CLK as selected by the CI bit in the SCP Status/Control register.

SCP TxD (SCP Transmit Data)

This output presents new data bits on SCP CLK edges as selected by the CI bit of the SCP Status/Control register.

4.3 POWER SUPPLY

V_{DD} — The most positive power supply pin, normally +5 V with respect to V_{DD}. There are three V_{DD} pins on the DDLC.

V_{SS} — The most negative power supply pin, normally 0 V. There are three V_{SS} pins on the DDLC.

Pins 2 and 35 are V_{DD} and V_{SS} respectively for the DMA and MPU sections of the chip. Pins 9 and 21 are V_{DD} and V_{SS} respectively for the pin drivers which can sink and source high peak currents. Pin 52 is V_{DD} and pin 51 is V_{SS} for the serial interface and bit handler sections. Powering the separate sections of the chip in this manner helps minimize noise problems arising from asynchronous clocking schemes. Recall that normally the serial sections are asynchronous from the MPU/DMA sections.

Section 9 shows the pin assignments for the MC145488.

Table 4-4. MPU Interface Pin Names

PLCC Pin #	68000 Mode	80186 Mode
61	Address 15	Address 15
62	Address 14	Address 14
63	Address 13	Address 13
64	Address 12	Address 12
65	Address 11	Address 11
66	Address 10	Address 10
67	Address 9	Address 9
68	Address 8	Address 8
1	Address 7	Address 7
3	Address 6	Address 6
4	Address 5	Address 5
5	Address 4	Address 4
6	Address 3	Address 3
7	Address 2	Address 2
8	Address 1	Address 1
10	Data 0	Data 0
11	Data 1	Data 1
12	Data 2	Data 2
13	Data 3	Data 3
14	Data 4	Data 4
15	Data 5	Data 5
16	Data 6	Datat 6
17	Data 7	Data 7
18	Data 8	Data 8

PLCC Pin #	68000 Mode	80186 Mode
19	Data 9	Data 9
20	Data 10	Data 10
22	Data 11	Data 11
23	Data 12	Data 12
24	Data 13	Data 13
25	Data 14	Data 14
26	Data 15	Data 15
27	Master Clock	Master Clock
28	Chip Select	Chip Select
29	Read/Write	Read/Write
30	Address Strobe	Write Strobe
31	Lower Data Strobe	High Byte Enable
32	Upper Data Strobe	Address 0
33	Reset	Reset
34	Interrupt Acknowledge	Interrupt Acknowledge
36	Interrupt Request	Interrupt Request
37	Data Transfer Acknowledge	Ready
38	Bus Error	Read Strobe
39	Bus Request	Hold Request
40	Bus Grant	Hold Acknowledge
41	Bus Grant Acknowledge	80186 Select
42	Owner 0	Owner 0
43	Owner 1	Owner 1

SECTION 5
BUS OPERATION

This section describes the operation of the DDLC microprocessor interface in the Motorola 68000 environment and the Intel 80186 environment during host processor operation, DMA operation, bus arbitration, and interrupt operation. Functional timing diagrams are included to clarify the timing of the various operations, but are not intended to be parametric definitions. For detailed timing parameters, see Section 8.

In the discussions that follow, the words **assert** and **negate** are used extensively. They indicate the logical value of the signal regardless of whether the voltage level on the pin is high or low.

5.1 68000 SYSTEM OPERATION

Operation in the Motorola 68000 environment is chosen by providing a pull-up resistor from the $\overline{BGACK}$ pin to V_{DD}. When the DDLC exits the reset state (either hardware or software), the level on the $\overline{BGACK}$ pin is internally latched. This requires that any other $\overline{BGACK}$ pins in the system be high-impedance or at a logic high during reset. The following sections describe the operation of the DDLC in the 68000 mode. The DDLC can operate in either a 16-bit or an 8-bit system.

All DDLC MPU interface circuitry timing is derived from the MCLK pin. Normally this pin will be connected to the host's master clock, but it may operate asynchronously to the host if all bus timing requirements are satisfied. In the timing diagrams that follow, MCLK is assumed to be identical to the host's clock.

A few words are needed to describe the 68000 memory architecture. All memory may be thought of as 16-bit-wide word locations which reside only on even addresses. However, the 68000 does have the capability of accessing bytes. Most significant bytes are located on even (byte) addresses and least significant bytes are located on odd (byte) addresses. For example, if the base address of the DDLC is placed at A100 (hex), the System Control register will be located at address A100, the Master Status register will be located at A102, and so on. The interrupt number (bits b0-b7) of the Master Status register is actually located at address A103. This address is not directly accessible by the 68000 since there is no A0 address lines to indicate even or odd addresses. Two data strobe pins are provided to solve this dilemma. Upper Data Strobe ($\overline{UDS}$) indicates when asserted, that the most significant byte of the 16-bit data bus is active and Lower Data Strobe ($\overline{LDS}$) indicates when asserted, that the least significant byte of the bus is active. If the interrupt vector is read in a byte-wide operation, only the $\overline{LDS}$ pin would be asserted indicating that the least significant (odd) byte is valid. This is effectively address A103. For word-wide (16-bits) operations, both $\overline{UDS}$ and $\overline{LDS}$ are asserted indicating that the entire 16-bit word is active.

For 8-bit 68008 systems, the memory architecture is composed of 8-bit locations. Even and odd addresses are used so an A0 pin is provided to access every byte. When the DDLC is used in the 68008 mode, the $\overline{UDS}$ pin becomes the least significant address pin (A0) and $\overline{LDS}$ becomes the Data Strobe ($\overline{DS}$). The DDLC's internal 16-bit b us is converted to an external 8-bit bus which is active on pins D0-D7. Pins D8-D15 become deactivated.

5.1.1 Bus Width Selection

The DDLC is in the 8-bit mode after reset. To remain in 8-bit mode, it is recommended that D8 of the data bus be connected to VSS. 16-bit operation is selected by writing FFFF to address 6. Address 6 is a dummy register which is used to qualify 16-bit operation. Once the 16-bit mode is selected, it is internally latched until the DDLC is reset by pulling the $\overline{RST}$ pin low. A software reset does not clear 16-bit operation.

5.1.2 Host Operation

The DDLC is the host processor mode when it is a memory mapped slave peripheral to the bus master. Any time the DDLC is not a bus master (actually performing a DMA access), the $\overline{CS}$ and $\overline{IACK}$ pins are continually sampled. A read or write cycle begins immediately when either of these pins are detected in the asserted state. The DDLC either accepts or presents data from or to the bus, depending on the level of the R/$\overline{W}$ pin. All internal registers are accessible to the host as decoded by address pins A1-A6. Operation of the $\overline{IACK}$ pin is similar to a read cycle and is described in Section 5.3.4.

5.1.2.1 WRITE CYCLE. During a host processor write cycle, the DDLC accepts data from e host and places it into the selected register. All internal registers are writable with the exception of the frame length registers which are read only. Figure 5-1 illustrates the timing relationships between the various bus interface pins in a typical 68000 write cycle. Timing is identical for even byte, odd byte, or word operations so the $\overline{UDS}$ and $\overline{LDS}$ pins are shown together. If an unoccupied internal address is selected, the DDLC will not indicate an error condition and the written data will be lost.

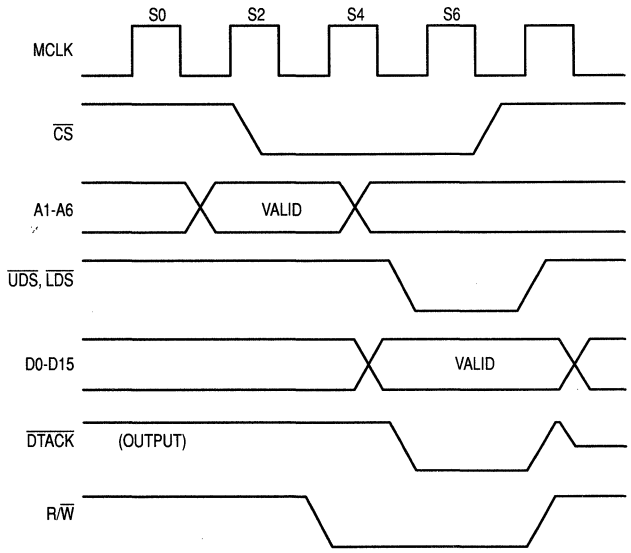


Figure 5-1. 68000 Host Write Cycle

While in the MPU mode of operation, the DDLC samples $\overline{CS}$ on falling edges of the MCLK. A write cycle begins on the MCLK edge on which $\overline{CS}$ is first detected low with R/$\overline{W}$ low. On the next rising MCLK edge, the address is latched and $\overline{DTACK}$ is asserted. Then data is allowed to enter the internal bus. When the data strobes ($\overline{UDS}$ and/or $\overline{LDS}$) are asserted, the latch of the selected register is opened and the data enters the register. The register's latch is closed-off and the data is retained when the data strobes are negated. When $\overline{CS}$ is negated, $\overline{DTACK}$ is negated and the cycle is complete. $\overline{DTACK}$ is an open-drain output in the MPU mode. When it is negated, a pull-up device is momentarily connected to provide a fast risetime. $\overline{CS}$ must return to a negated state for at least one MCLK cycle before another DDLC access cycle may begin.

5.1.2.2 READ CYCLE. During a host processor read cycle, the DDLC places data from the selected internal register onto the data bus. All internal registers of the DDLC are readable. Figure 5-2 describes the timing of a typical 68000 read cycle. Timing is identical for even byte, odd byte, or word operations. If the host selects an unoccupied internal register, the DDLC will not indicate an error and will place unknown data on the bus. In the 16-bit mode, the DDLC always presents 16-bits to the bus regardless of which data strobe pin ($\overline{UDS}$ or $\overline{LDS}$) was asserted during the cycle.

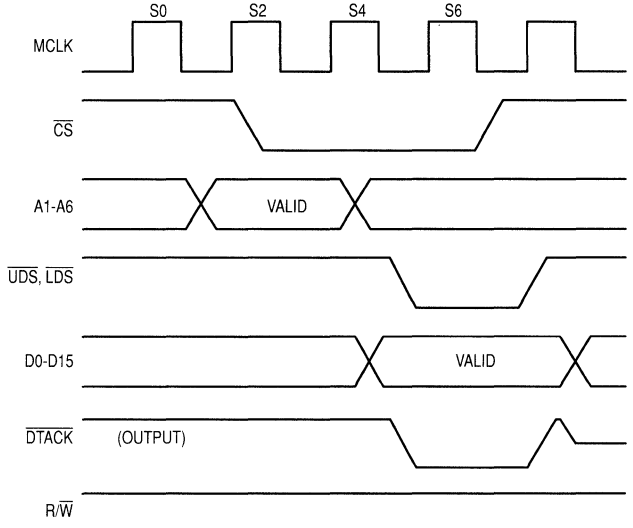


Figure 5-2. 68000 Host Read Cycle

A read cycle begins when $\overline{CS}$ and the data strobes are asserted and R/$\overline{W}$ is high. The DDLC decodes A1-A6 and selects a register and outputs the data. On the rising MCLK edge after $\overline{CS}$ was detected asserted, $\overline{DTACK}$ is asserted. $\overline{DTACK}$ is negated after the rising edge of $\overline{CS}$ and the cycle is complete. In some systems, the data strobes will be delayed from $\overline{CS}$. When this occurs, the data will be placed on the bus when the data strobes are asserted.

5.1.3 DMA Operation

In the DMA mode, the DDLC operates as a system bus master. When the internal FIFOs need data or have data to place in memory, the DDLC's DMA controller requests ownership of the system bus. When ownership is granted, the DDLC takes control of the bus and executes a memory cycle. The DDLC services one DMA request then relinquishes the bus. If another DMA request is pending

internally, the DDLC will not re-request the bus for at least three MCLK cycles after the bus has been relinquished from the previous access.

The internal address generators can operate on address ranges of up to 64 kbytes. Two pins, $\overline{OWN}$ 0 and $\overline{OWN}$ 1 are provided to indicate which communication channel is the bus owner. These signals may be used to control an external device, such as an address page register onto the address bus so each channel may operate on a different memory page. While a DMA cycle is in progress, only one $\overline{OWN}$ pin will be asserted. The other will be driven to a logic high for the duration of the DMA cycle. For systems that need them, function codes may also be enabled with the OWN pins. Section 7 describes applications for these pins.

Note: In 16-bit systems, the DDLC does not check for odd word boundaries or generate odd word addresses. The system designer must ensure that the DDLC is not supplied with odd pointers in 16-bit systems. For 8-bit systems, odd byte addresses are proper and the A0 address pin selects even or odd bytes. For both 16-bit and 8-bit systems, it is suggested that the programmer use only even addresses for transmit and receive buffers and even lengths for receive buffers. This will maintain software compatibility between the two systems.

In the following descriptions, MCLK cycles are used as a reference. All internal timing is generated from the MCLK. The MCLK used by the DDLC is not necessarily synchronous to the host's clock. Correct operation of the DDLC in a system is maintained if the proper synchronization times are met for input pins.

5.1.3.1 DMA WRITE CYCLE. A DMA write cycle is performed when a receiver FIFO has data to be placed in a buffer in memory. Figure 5-3 describes the timing of a DDLC write cycle. Bus arbitration is assumed to have been completed before the cycle begins.

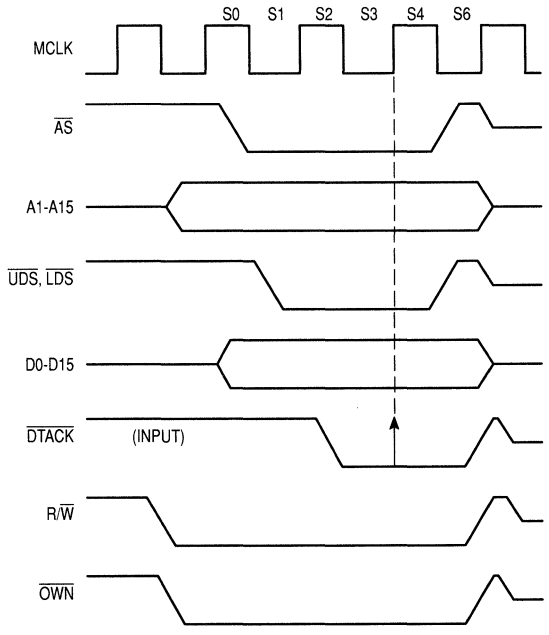


Figure 5-3. 68000 DMA Write Cycle

A DMA write cycle is three MCLK periods long with the R/$\overline{W}$ level and address presented to the bus one-half clock early. At the beginning of S0, data is presented to the bus and address strobe is asserted. On the next MCLK edge, the data strobes are asserted. In the 16-bit mode, the DDLC always makes 16-bit data transactions, so both strobes are asserted at the same time. $\overline{DTACK}$ is sampled on the rising edge of S4, and if found asserted, the cycle is completed with $\overline{AS}$ and the data strobes negated on the falling edge of S4. The Address and R/$\overline{W}$ forced to high-impedance on the rising edge of S5. All control bus pins have an internal active pull-up to provide a fast risetime before going high-impedance. If $\overline{DTACK}$ is inactive on the rising edge of S4, a wait-state is inserted and $\overline{DTACK}$ is resampled at the end of the wait-state. This process continues until $\overline{DTACK}$ is found active or until a bus error is found. Operation with the internal wait-state generator is described in Section 5.1.3.4.

In 16-bit operation, all write transactions are 16-bits wide. For this reason, memory buffers should be an even number of bytes long. If a received frame is an odd number of bytes long, the last received byte will be written to memory in a 16-bit transaction. The data in the lower byte (higher byte address) in this case, will be unknown data. The received byte count will, however, indicate the correct number of bytes received in the frame.

5.1.3.2 DMA READ CYCLE. The DDLC performs a DMA read cycle when a transmit FIFO needs data to transmit. Figure 5-4 describes the timing of a DDLC read cycle. Bus arbitration is assumed to be completed before the cycle begins.

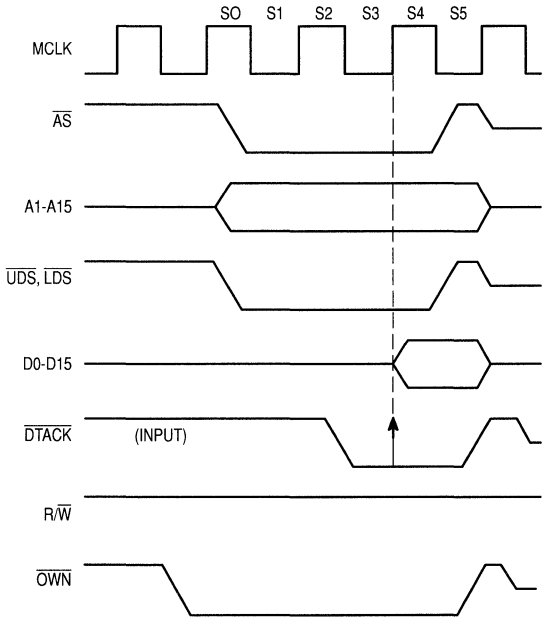


Figure 5-4. 68000 DMA Read Cycle

A DMA read cycle begins when the address is driven on the bus and the R/$\overline{W}$ pin is driven high. One-half MCLK later, at the beginning of S0, the address and data strobes are driven low. $\overline{DTACK}$ is sampled on the rising edge of S4, and if found asserted the address and data strobes are negated

on the falling edge of S4 and the data is latched into the FIFO. One-half MCLK later, the address pins and R/$\overline{W}$ pin go high-impedance. If $\overline{DTACK}$ is inactive on the rising edge of S4, a wait-state is inserted and $\overline{DTACK}$ is resampled at the end of the wait-state. This process continues until $\overline{DTACK}$ is found active or a bus error is found. Operation with the internal wait-state generator is described in Section 5.1.3.4. As with DMA write cycles, all DMA read transactions in the 16-bit mode are 16-bits wide. If only 8-bits of data are needed, for an odd byte length frame, the extra byte that was read is discarded.

5.1.3.3 BUS ERROR RECOVERY. The DDLC provides a bus error pin for systems that detect bus errors. The $\overline{BERR}$ pin is sampled at the same time as $\overline{DTACK}$, and if $\overline{BERR}$ is found to be asserted, the DDLC resamples $\overline{BERR}$ one-half clock period later to qualify it. If $\overline{BERR}$ is found to still be asserted, the DDLC terminates the DMA cycle and queues an interrupt. For DMA write cycles (associated with a receiver) the receiver's RxEn bit is cleared to reset the receiver. For DMA read cycles (associated with a transmitter) the transmitter's BR bit is cleared to disable further DMA requests from at channel. The transmitter is still enabled and will enter the idle state after transmitting an abort character.

The DDLC also responds to address errors which are defined as the assertion of $\overline{CS}$ or $\overline{IACK}$ while the DDLC is in a DMA cycle. The $\overline{CS}$ and $\overline{IACK}$ pins are sampled at the same time as $\overline{BERR}$ and if found asserted, an address error is declared and an interrupt is queued.

5.1.3.4 WAIT-STATE GENERATOR AND $\overline{DTACK}$ CONTROL. The DDLC provides a programmable internal wait-state generator for systems that have slow memory. The wait-state generator inserts wait-states before S4 which are essentially equivalent to repeated S2-S3 states. Four wait-state selections are available including 0, 1, 2, or 4 wait-states. Additionally, $\overline{DTACK}$ may be enabled to operate with the wait-state generator, so $\overline{DTACK}$ is sampled after the wait-states have been executed. The cycle then terminates only after $\overline{DTACK}$ is asserted. the $\overline{DTACK}$ pin may be internally disabled so the DMA cycle will terminate after the wait-state have executed regardless of the state of $\overline{DTACK}$. Figure 5-5 describes the operation of the wait-state generator with one programmed wait-state and a delayed $\overline{DTACK}$ signal. **Sp** indicates programmed wait-states and **Sw** indicates wait-state generated because of the delayed $\overline{DTACK}$ signal. The downward pointing arrows indicate the sampling points for $\overline{DTACK}$.

5.1.4 Bus Arbitration

An internal DMA scheduler continuously polls the four data channels for service requests. When a request is found, the scheduler prepares the data and address for the DMA access and requests the system bus from the current bus owner. Once the DDLC has ownership of the bus, it services one internal request then relinquishes ownership of the bus.

Figure 5-6 describes the operation of the 68000 arbitration mechanism. 68008 bus arbitration is slightly different in that only two pins are used for bus arbitration; $\overline{BR}$ and $\overline{BT}$. When using the DDLC in a 68008 system, the $\overline{BGACK}$ and $\overline{BR}$ pins should be connected together to provide the proper $\overline{BR}$ signal to the processor. The signal edges overlap so the combined signal will be glitch-free.

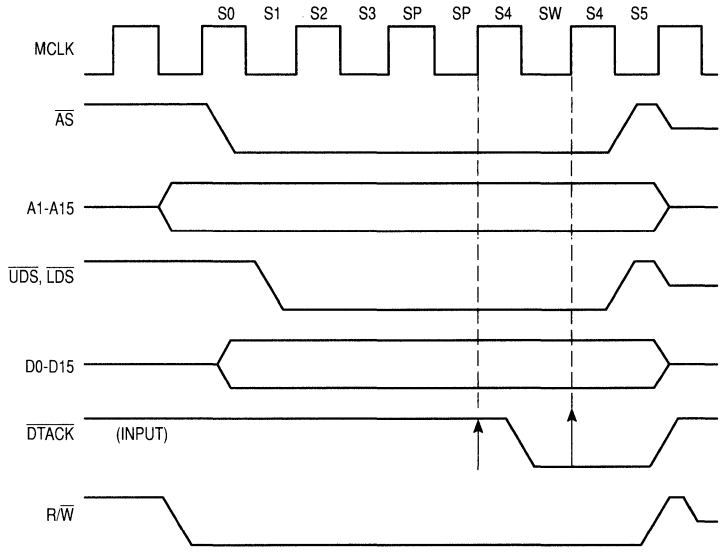


Figure 5-5. 68000 DMA Write Cycle with One Programmed Wait-State and a Delayed DTACK

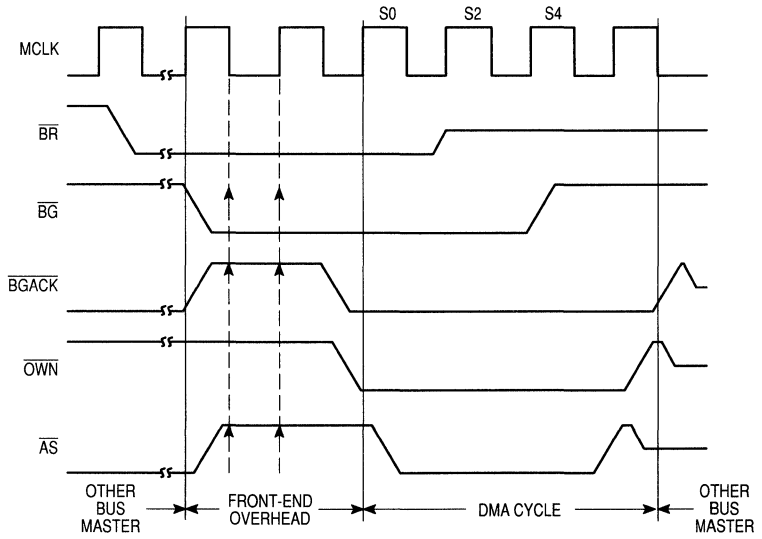


Figure 5-6. 68000 Bus Arbitration

Bus arbitration begins when the $\overline{BR}$ pin is asserted indicating that the DDLC is requesting the bus. The DDLC then samples for $\overline{BG}$ to be asserted, $\overline{BGACK}$ to be negated, and $\overline{AS}$ to be negated on successive MCLK edges. The upward pointing arrows indicate the sampling points of the three signals. When the signals are at the proper levels for two successive samples, the DDLC immediately assumes ownership of the bus and asserts $\overline{BGACK}$, $\overline{OWN}$ 0, or $\overline{OWN}$ 1, sets the level of R/$\overline{W}$ and places the address on the bus. On the next rising edge, the cycle begins with the assertion of address strobe. At the end of the DMA cycle, the DDLC negates $\overline{BGACK}$ and the $\overline{OWN}$ pins.

The DDLC verifies $\overline{BG}$ (or $\overline{HACK}$) by sampling $\overline{BG}$ (or $\overline{HACK}$) on a falling edge of MCLK followed by sampling on the next rising edge of MCLK. If $\overline{BG}$ (or $\overline{HACK}$) are not detected on a falling edge of MCLK the DDLC waits until the next falling edge of MCLK to restart verification. $\overline{BG}$ ($\overline{HACK}$) must be asserted during both successive clock edges in order for the DDLC to initiate a DMA cycle.

The DDLC will not re-request the bus for at least three MCLK cycles after relinquishing the bus even if another internal DMA request is pending. This ensures that another DMA requesting device has access to the bus and that the DDLC will not be a bus "hog".

5.2 80186 SYSTEM OPERATION

The DDLC also can operate as an 80186/88 peripheral. This mode is entered by connecting the $\overline{BGACK}$ pin to V_{SS}. The level of this pin is latched when the DDLC exits the reset state. The DDLC fully supports the 80186/88 bus structure except that the address and data buses are not multiplexed. The DDLC is designed so that it will reside on the demultiplexed global bus of an Intel system.

5.2.1 Host Processor Operation

In this mode, the DDLC is a peripheral device to the 80186 host processor. Internal registers may reside in either the I/O space or the memory space depending on how the $\overline{CS}$ pin is decoded by external circuitry. The DDLC is in this mode of operation when either the $\overline{CS}$ or $\overline{IACK}$ pins are activated. Operation of the $\overline{IACK}$ pin is described in Section 5.3.5.

The $\overline{RDY}$ pin is an input and is thus not used in the host processor mode. The DDLC, however, appears to be fast memory and does not require wait states to be inserted by the host.

The DDLC operates in an 8-bit or 16-bit mode, the $\overline{BHE}$ pin is unused, and the A0 pin directs the data to the even or odd bytes of the internal registers. In the 16-bit mode, the registers are accessed on even addresses. Both AO and $\overline{BHE}$ are active low during a 16-bit access. Selection of 8-bit operation is described in Section 5.1.

5.2.1.1 WRITE CYCLE. During a write cycle, the DDLC accepts data from the data bus and places it into the selected register. Figure 5-7 describes the timing relationships of the various bus interface pins. As with the 68000 mode, data written to an unoccupied internal address is lost without indication.

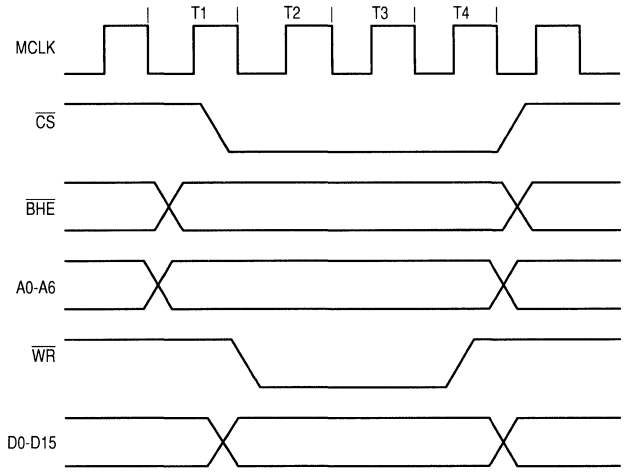


Figure 5-7. 80186 Host Write Cycle

A write cycle begins when the $\overline{CS}$ pin is activated and the address is on the bus. The $\overline{WR}$ pin is activated by the host and the internal circuitry selects the proper register for a write access. The data and address must be stable before $\overline{WR}$ is asserted as this pin directly controls the latch function of the internal registers. Data is latched into the DDLC on the rising edge of $\overline{WR}$. For byte accesses, the A0 and $\overline{BHE}$ pins direct the DDLC's address decoder to the proper byte of the selected register.

5.2.1.2 READ CYCLE. During a host read cycle, the DDLC takes data from an addressed internal register and places it on the bus. Figure 5-8 describes the timing of a read cycle. If an unoccupied internal address is read, the DDLC will placed unknown data on the bus.

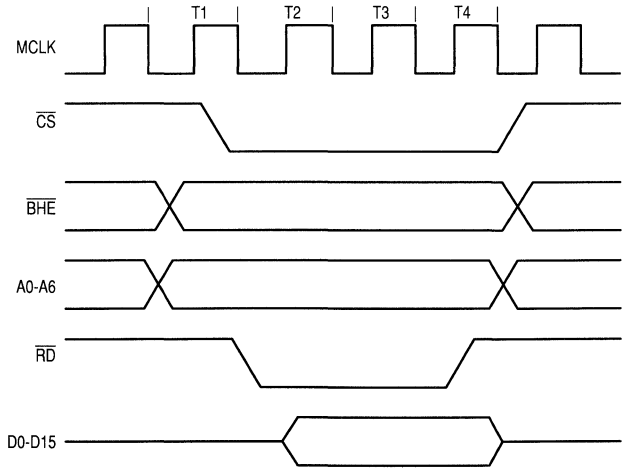


Figure 5-8. 80186 Host Read Cycle

A read cycle begins when the $\overline{CS}$ pin is asserted. Data is available after $\overline{CS}$ is detected asserted. The $\overline{Rd}$ pin directly controls the output drivers of the data pins and when it is asserted, the data is placed on the bus. For byte accesses in a 16-bit system, the AO and $\overline{BHE}$ pins direct the DDLC's address decoder to the proper byte of the selected register.

5.2.2 DMA Operation

In the DMA mode of operation, the DDLC is the system bus master. When an internal transmitter or receiver needs service, it generates an internal DMA request. The DMA controller calculates the address to be placed on the address bus and prepares the data to be placed on the data bus (if the request was from a receiver). Then the DDLC requests the system bus by asserting $\overline{HREQ}$. When the system bus has been given to the DDLC ($\overline{HACK}$ asserted), the internal DMA request is serviced with a single bus cycle, then the bus is relinquished. There is one MCLK cycle between the time $\overline{HACK}$ is asserted and the time the bus cycle begins. When the bus is relinquished, the next bus master should wait at least one-half MCLK cycle for the bus to settle before beginning its cycle.

Note: In 16-bit systems, the DDLC does not have the capability of operating with odd-address word boundaries. All word (16-bit) accesses must be on even addresses. For 8-bit systems, either even or odd addresses are proper, but the programmer should start all buffers on even addresses to maintain compatibility with 16-bit systems. In 16-bit systems, the DDLC will always read or write 16-bit words from or to memory. In the case of a transmitted frame with an odd number of bytes, only the least significant (valid) byte of the last word read from the memory buffer will be used and the most significant (invalid) byte will be discarded. For received odd byte-length frames, the last received byte will be transferred to memory with a 16-bit write and the least significant byte will contain the last received byte. The most significant byte will be unknown data. The receive byte count register always contains the correct number of bytes received.

The DDLC automatically aligns the data (depending on whether the 68000 or 80186 mode is selected) so that succeeding received data bytes are placed in ascending order in memory. Transmitted bytes are retrieved from memory in ascending order and are internally rearranged so that they are transmitted in the correct order.

5.2.2.1 DMA WRITE CYCLE. When the DDLC is the bus owner, a DMA write cycle may be performed to transfer data received from the serial bit-handler to a memory buffer. Figure 5-9 describes the timing relationships of the various signals during a DMA write cycle in the 80186 mode. The timing diagram assumes that bus arbitration was completed before the cycle started.

A write cycle begins with the DDLC placing the address on the bus and asserting the $\overline{OWN}$ pin at the beginning of T1. On the rising edge during T1, the data is placed on the bus. At the beginning of T2, $\overline{WR}$ is asserted. The $\overline{RDY}$ pin is sampled on the rising edge of MCLK during T3 and if found asserted, the cycle is terminated with $\overline{WR}$ negated at the beginning of T4. On the next rising MCLK edge, the address and data are removed. Since the DDLC performs one transaction per bus request, all address, control, and data lines go high-impedance at the end of the cycle. The DDLC has separate pins for address and data, so no demultiplexing control signals are required. R/$\overline{W}$ is provided to control the direction of a system bus transceiver. Note that one of the two $\overline{OWN}$ pins is asserted during every DMA cycle, so the $\overline{OWN}$ pins may be used to enable or disable other devices on the bus. The inactive $\overline{OWN}$ pin is driven to a logic high state while a DMA cycle is active.

5.2.2.2 DMA READ CYCLE. A DMA read cycle is performed when a transmitter needs data. Timing is similar to a write cycle and is described in Figure 5-10.

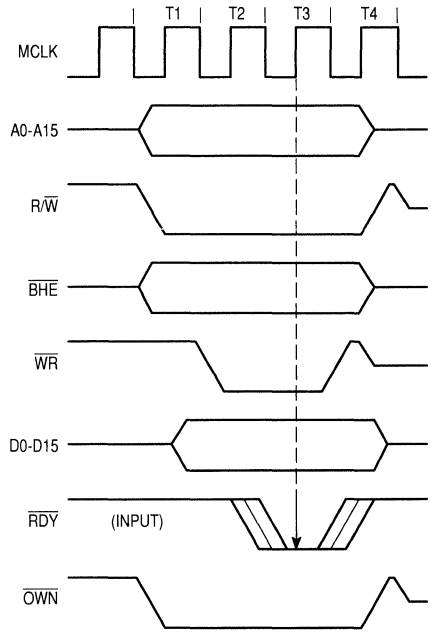


Figure 5-9. 80186 DMA Write Cycle

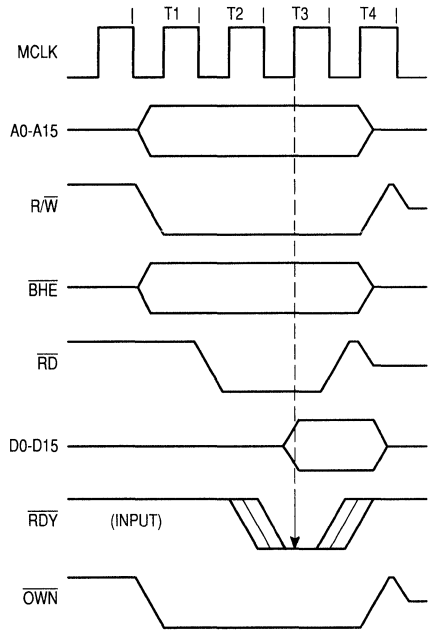


Figure 5-10. 80186 DMA Read Cycle

A read cycle begins with the DDLC placing the address on the bus at the beginning of T1. At the beginning of T2, $\overline{RD}$ is asserted. On the rising MCLK edge during T3, $\overline{RDY}$ is sampled and if found asserted, the cycle completes with $\overline{RD}$ being negated at the beginning of T4. Data is latched into the DDLC on the rising edge of $\overline{RD}$. On the rising MCLK edge during T4, the address is removed from the bus and all address, control, and data lines go to the high-impedance state. The operation of the $\overline{RDY}$ pin is described in Section 5.2.2.3.

5.2.2.3 WAIT-STATE GENERATOR AND READ CONTROL. The DDLC provides a programmable internal wait-state generator for systems that have slow memory. The wait-state generator inserts wait-states before the rising MCLK edge during T3 which are essentially equivalent to repeating the second half of T2 and the first half of T3. Four wait-state selections are available including 0, 1, 2, or 4 wait-states. Additionally, the $\overline{RDY}$ pin may be enabled with the wait-state generator so $\overline{RDY}$ will be sampled after the wait-states have been executed and the cycle will terminate when $\overline{RDY}$ is found to be asserted. $\overline{RDY}$ can be internally enabled in which case the DMA cycle will terminate after the wait-states have been run regardless of the state of $\overline{RDY}$. Figure 5-112 describes the operation of the wait-state generator with one programmed wait-state and a delayed $\overline{RDY}$ signal in a ready cycle. **Tp** indicates the programmed wait-state. T3 is repeated because when $\overline{RDY}$ was sampled the first time, it was not asserted. The downward pointing arrows indicate the sampling points for $\overline{RDY}$. Operation in a write cycle is similar.

5.2.3 Bus Arbitration

In the 80186 mode of operation, the DDLC uses a hold request/hold acknowledge bus arbitration protocol. When the DDLC has an internal DMA request, $\overline{HREQ}$ is activated. $\overline{HACK}$ is activated by

the bus arbiter when the DDLC has been granted ownership of the bus. The DDLC assumes ownership when $\overline{HACK}$ is active and holds $\overline{HREQ}$ active until the DMA transaction has been completed. Figure 5-12 describes the peration of the bus arbitration pins on the DDLC. The $\overline{OWN}$ pins indicate that the DDLC is the bus owner and is activated after $\overline{HACK}$ has been found asserted for two sample periods. The downward pointing arrows indicate the same points for $\overline{HACK}$.

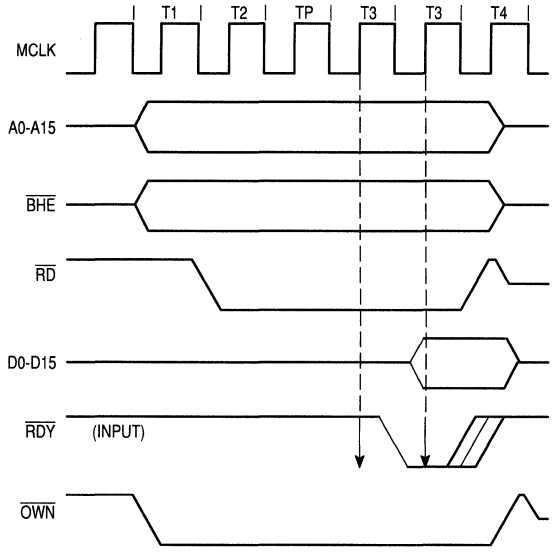


Figure 5-11. 80186 DMA Cycle with One Programmed Wait-State and Delayed RDY

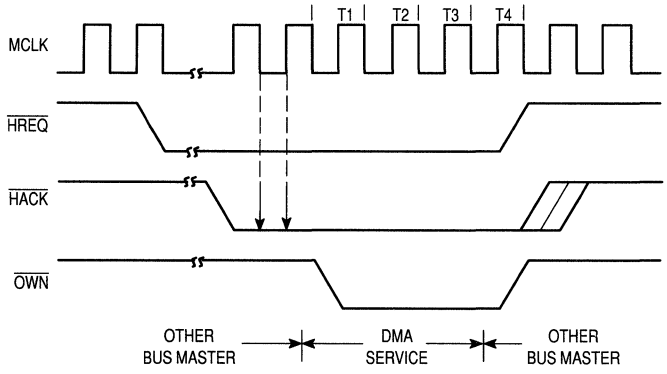


Figure 5-12. 80186 Mode Bus Arbitration

A bus arbitration cycle begins when an internal DMA request is generated by a transmitter or receiver. $\overline{HREQ}$ is asserted low and the DDLC samples $\overline{HACK}$ on falling MCLK edges. The DDLC assumes ownership of the bus after $\overline{HACK}$ has been found asserted for two sample periods. The

$\overline{\text{OWN}}$ pin is asserted low at the beginning of T1. After the DMA activity has been completed $\overline{\text{HREQ}}$ and $\overline{\text{OWN}}$ are negated on the rising MCLK edge during T4. Some time later, $\overline{\text{HACK}}$ is negated by the bus arbiter and the next bus master may negotiate for the bus.

5.3 INTERRUPT OPERATION

The DDLC communicates with the host through status and control bits. When a particular status bit changes state, sometimes the host needs to service the status change quickly so interrupts are typically used. The DDLC has a variety of interrupts which may be used by the host to get fast service on certain status changes. All interrupt-generating status bits may be set by the user. This generates an immediate interrupt if the associated interrupt enable bit is set. This feature is useful for software debugging.

5.3.1 Interrupt Sources

There are 27 interrupt sources in the DDLC. As discussed in Section 2.4.3, the interrupts are grouped into several categories. Normal operation interrupts are generated when the DDLC has completed a task and is informing the host of the completion or when a change has been detected in the state of the physical link. An example of this category is Rx DMA complete. Bit-handler interrupts are fault conditions in the reception or transmission of frames from which the DDLC cannot recover on its own. An example is a transmit FIFO underrun. System interrupts are detected faults of the microprocessor bus, an example of which is bus error. Timer interrupts are generated when either timer counts down from 00 to FF. A complete list of interrupt sources is described below in Table 5-1.

Table 5-1. Interrupt Priority and Vector Values

Priority	Interrupt Name	Vector
1	Rx Ch 0 Bus Error	XC
2	Rx Ch 1 Bus Error	XE
3	Tx Ch 0 Bus Error	XD
4	Tx Ch 1 Bus Error	XF
5	Rx Ch 0 Address Error	XC
6	Rx Ch 1 Address Error	XE
7	Tx Ch 0 Address Error	XD
8	Tx Ch 1 Address Error	XF
9	Rx Ch 0 DMA Complete	X4
10	Rx Ch 1 DMA Complete	X6
11	Tx Ch 0 DMA Complete	X5
12	Tx Ch 1 DMA Complete	X7
13	Tx Ch 0 Frame Complete	X5
14	Tx Ch 1 Frame Complete	X7
15	Rx Ch 0 FIFO Overrun	X8
16	Rx Ch 1 FIFO Overrun	XA
17	Tx Ch 0 FIFO Underrun	X9
18	Tx Ch 1 FIFO Underrun	XB
19	Rx Ch 0 Buffer Overrun	X8
20	Rx Ch 1 Buffer Overrun	XA
21	Rx Ch 0 Idle	X4
22	Rx Ch 1 Idle	X6
23	Ch 0 CD Changed State	X4
24	Ch 1 CD Changed State	X6
25	SCP Complete	X2
26	Timer 1	X3
27	Timer 0	X1

NOTE: X indicates user programmable bits.

5.3.1.1 NORMAL INTERRUPTS. These interrupts are generated in response to changes in the link state. A discussion of each follows.

Receive DMA Complete — Each channel has a Receive DMA Complete interrupt. This is generated when the receiver has detected a frame's closing flag and the DMA controller has finished placing the last data byte in memory. The DDLC is informing the host that a complete frame was received and is in memory at the locations described in the buffer descriptor. This interrupt is not generated in the transparent mode as flag detection is not used.

Transmit DMA Complete — This is an early interrupt which indicates that the DMA controller has transferred the last byte of data from memory to the FIFO. This interrupt is used when the host wishes to send back-to-back frames. At 64 kbps, the host has several hundred microseconds from the time this interrupt is generated until the FIFO empties the transmitter closes the frame with a CRC field and flag to prepare the next frame's buffer descriptor.

Transmit Frame Complete — For ISDN S/T interface D-channel applications, where back-to-back frames are illegal, Transmit Frame Complete is used. This interrupt indicates that the closing flag of a frame was successfully transmitted. On the D-channel, where collisions may be found, the host needs to know when a frame was actually transmitted. If the DMA Complete interrupt was enabled and a collision occurred while the last byte of data was being transmitted, the interrupt would have occurred before the collision and the host may have already prepared the next frame for transmission. When the DDLC prepared to retransmit the frame that collided, the wrong frame would be sent. This interrupt is not generated in the transparent mode.

Receive Idle — Receive Idle indicates that the generating channel's receiver detected at least 15 contiguous '1s' and that the link has gone into the idle state. This interrupt is used only in LAPB where inter-frame periods are filled with flags. In this case idle is distinct from inter-frame periods. In LAPD, where inter-frame periods are filled with '1s', there is no distinction between inter-frame and idle periods. In LAPD applications, this interrupt should be masked.

Carrier Detect Changed State — This interrupt is generated when the carrier detect pin changes state. The interrupt status bit is bit 12 of the Receive Status register. The current state of the $\overline{CD}$ pin is available on bit 15 of the same register. When the SCP is disabled and the serial interface is not in the timeslot mode, the $\overline{CD}$ pin is available for general purpose input.

5.3.1.2 TIMER INTERRUPTS. There are two timers in the DDLC. The timers are low resolution devices intended for long-term timeouts for HDLC-type protocols. The source of the clock to the timers is the SCP prescaler which is then further divided by 1024. The host can preset the value in the counter at any time and the counter will begin counting down toward zero from that value. When a timer register counts down past zero, an interrupt is queued. The timer interrupt status bits operate identically to the other interrupt-generating bits. The status bit must be read while it is set before it may be cleared. **Note:** The timer will ignore a count pulse if the pulse occurs while the register is being accessed by the host for a read or write operation.

5.3.1.3 BIT-HANDLER FAULT INTERRUPTS. These interrupts are generated when the transmit or receive state machines detect errors that cannot be serviced without host intervention.

Receive FIFO Overrun — This indicates that the DMA controller did not service the FIFO's request for data and data was lost. When this interrupt is generated, future DMA activity on the interrupting channel is suspended until the host re-enables the suspended channel.

Transmit FIFO Overrun — This indicates that the DMA controller did not service the FIFO's request for service. The frame in transmission is aborted and the transmitter enters the idle state. FIFO interrupts indicate that the system is having hardware or software malfunctions or that the system is approaching its throughput limit.

Receive Buffer Overrun — The Receive Buffer Overrun interrupt indicates that the received frame was too large for the allocated buffer in memory and usually indicates that a closing flag was corrupted in transmission. When this interrupt is generated, DMA activity on that buffer is suspended and if the alternate buffer is available, data will continue to be received and will be placed in the alternate buffer. This interrupt is used in the transparent mode to indicate that a block of data was received.

5.3.1.4 SYSTEM FAULT INTERRUPTS. The interrupts for all four channels are enabled or masked together as either bus or address errors. A bus error is defined as the activation of the $\overline{BERR}$ pin while a DMA cycle is in progress. The timing of sampling of the $\overline{BERR}$ pin is the same as for the $\overline{DTACK}$ pin. External circuitry determines which system faults allow the $\overline{BERR}$ pin to be asserted. If $\overline{BERR}$ is found to be asserted in a DMA cycle, the DMA channel that was active when the fault occurred is disabled. Address errors are defined to be activation of the $\overline{IACK}$ or $\overline{CS}$ pins while a DMA cycle is in progress. This implies that the DMA controller is pointing to a register inside the DDLC. When an address error is detected, the DMA channel that was active when the fault occurred is disabled. For transmitters this means that DMA activity stops, the transmitter generates an abort character then enters the idle state. Receivers are reset by clearing the RxEn bit.

5.3.1.5 SCP COMPLETE INTERRUPT. This interrupt is generated when an SCP transfer is complete. It operates similarly in master and slave modes. In master mode, the interrupt is generated after the eighth bit has been transmitted. In slave mode, the rising edge of slave select generates the interrupt.

5.3.2 Interrupt Request ($\overline{IRQ}$) Pin

The $\overline{IRQ}$ pin is an active-low, open-drain pin which indicates that there is a pending interrupt from one of the DDLC's sources. $\overline{IRQ}$ is the logical NOR of all enabled interrupt sources. As long as there

is at least one pending interrupt, $\overline{IRQ}$ remains low. When the host processor requests the interrupt vector with either an $\overline{IACK}$ cycle or by reading the Master Status register, the highest priority interrupt pending at that instant $\overline{CS}$ or $\overline{IACK}$ are asserted will be encoded into bits b0-b3. The $\overline{IRQ}$ pin will remain asserted until all interrupting sources have been cleared. It is the host's responsibility, in the interrupt service routine, to clear the status bit which caused the interrupt or the same interrupt will immediately cause another interrupt.

5.3.3 Interrupt Vector

The 27 interrupts are encoded into four bits of an eight bit vector. The most significant bits of the vector are user programmable so the block of vectors may be placed anywhere in the vector map. When the vector is read either with an IACK or a read cycle, the highest pending interrupt will be encoded and latched at the instant that the internal read enable signal is decoded. Every time the vector is read, the highest pending interrupt is encoded.

After reset, the interrupt vector register reads 00 hex. This is different from other 68000 peripherals which generate vector number 0F hex (uninitialized interrupt). The programmer is expected to initialize the interrupt vector map before enabling any of the DDLC interrupts.

When an interrupt generating status bit is set, the bit must be cleared before that particular interrupt is cleared. **A status bit may only be cleared if it was set when it was read.** Stated another way, a status bit must be read in the set state before it may be cleared by writing a '0' into that bit position of the status register. This rule eliminates the possibility that an interrupt generating status bit can be accidentally cleared and thus lost. If a status bit is written with a '0' before it is read while it is a '1', the write operation for that bit will be ignored.

5.3.4 68000 Interrupt Acknowledge Cycle

An interrupt acknowledge cycle has timing identical to a read cycle. However, the vector number is placed only on the lower byte of the data bus (pins D0-D7). The $\overline{LDS}$ pin indicates that data is valid on the lower byte. The $\overline{IACK}$ pin and $\overline{CS}$ are mutually exclusive and should not be asserted at the same time. If they are asserted simultaneously, the $\overline{CS}$ pin has precedence. Figure 5-13 describes an IACK cycle in a 68000 system.

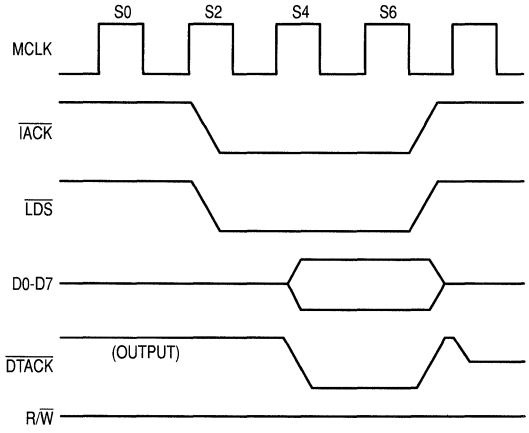


Figure 5-13. 68000 IACK Cycle

An interrupt acknowledge cycle is virtually identical to a read cycle except that it begins when $\overline{IACK}$ is asserted. The DDLC presents the interrupt vector number to the data bus on pins D0-D7 and asserts $\overline{DTACK}$ on the next rising MCLK edge after the $\overline{IACK}$ pin was detected low. The cycle is completed when $\overline{LDS}$ and $\overline{IACK}$ are negated.

5.3.5 80186 Interrupt Acknowledge Cycle

The 80186 interrupt acknowledge cycle is similar to read cycle except that the $\overline{INTA}$ signal is issued one clock earlier than the $\overline{RD}$ strobe. Because the DDLC responds to the $\overline{INTA}$ pin identically to the $\overline{RD}$ pin, the data will be placed on the bus one MCLK edge after the $\overline{INTA}$ pin was detected asserted. Figure 5-14 describes one INTA cycle. The 80186 actually performs two cycles separated by two wait states. The DDLC will respond to each cycle identically.

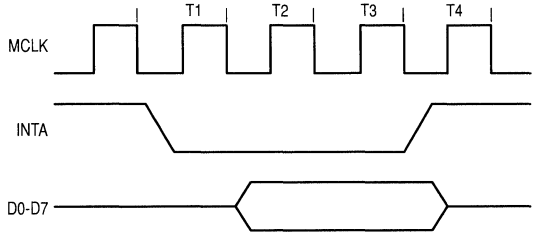


Figure 5-14. 80186 INTA Cycle

The DDLC monitors the $\overline{INTA}$ pin continuously and when it is found asserted (with $\overline{IRQ}$ asserted) presents the interrupt vector on the next MCLK edge.

5.4 RESET OPERATION

The DDLC may be reset by a hardware signal, or by software. Additionally, with the watchdog timer activated, the DDLC may reset the entire host system if the timer underflows. Table 3-1 describes the state of all register bits after a reset operation (either hardware or software). A software reset takes two MCLK cycles to complete. Figure 5-15 describes the timing of a watchdog timer generated reset. $\overline{RESET}$ becomes an output and is held low for 16 MCLK cycles then it is released.

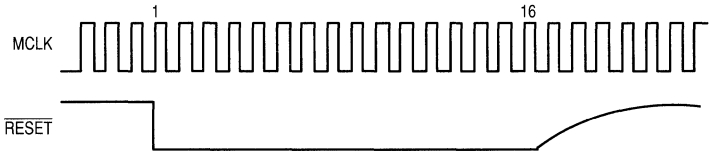


Figure 5-15. Watchdog-Generated Reset

SECTION 6
SERIAL INTERFACE OPERATION

The serial interface has several modes of operation. The following sections describe the operation of each mode. Functional timing diagrams are included to clarify the operation. For detailed parametric specifications see Section 8.

The DDLC's two channels each have three modes of operation for their serial interfaces: IDL, timeslot, and modem. In the IDL mode the DDLC is compatible with Motorola's ISDN family. When IDL D-channel operation is selected, D-channel access control pins (DREQ and DGNT) are also provided. The timeslot mode provides compatibility with typical PBX backplane timings. Both long-frame and short-frame synchronization is supported. In the modem modem, each channel supports a standard seven-pin synchronization modem interface including three modem control signals. See Table 4-4 for a description of the serial interface in functions. The two DDLC channels are identical but independent. The following discussions describe one channel, but they apply to both.

6.1 INTERCHIP DIGITAL LINK (IDL) INTERFACE FOR ISDN APPLICATIONS

The IDL interface is used to interconnect various devices on ISDN board-level systems. (See Figure 6-1.) The IDL distributes several channels of information from the ISDN basic access interface (S/T bus) that have been processed by the MC145474 S/T transceiver to the various devices on the bus using a timeslot technique. Full-duplex data is transferred in 20-bit frames every 125 µs providing 160 kbps full-duplex bandwidth. Figure 6-2 describes the operation of the IDL frames. A continuous clock at 2.56 MHz (or other standard frequency) shifts the data for both transmit and receive.

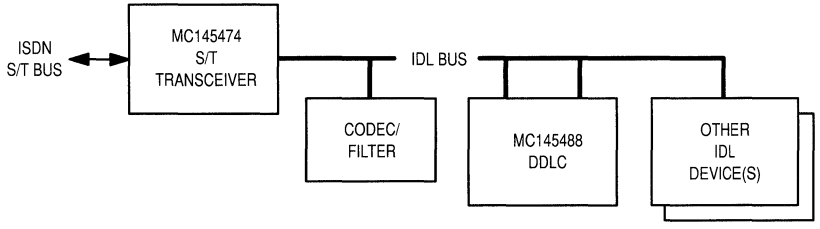


Figure 6-1. Typical IDL Bus Application

6.1.1 Functional Description

Two 64 kbps B-channels and one 16 kbps D-channel along with two 8 kbps auxiliary channels are passed over the IDL interface. The DDLC is an IDL slave which is clocked by the bus master (typically the MC145474 S/T transceiver). Figure 6-3 indicates the relationship of the four IDL signals.

IDL SYNC is an input which accepts a positive one clock period wide pulse. The 20 clock periods following the pulse contain the IDL frame. Sync occurs every 125 µs so that 8000 frames per

second are exchanged between the bus master and the slaves. IDL CLK is an input which receives a square wave at 2.56 MHz from the IDL bus master. For ISDN terminal applications, the MC145474 S/T transceiver is the IDL bus master which provides this clock. In other applications, frequencies such as 1.536 or 2.048 MHz may be used. IDL Tx is an output which is active only on assigned bits in the frames. For all other bit times, it is high-impedance. IDL Rx is the data input which samples bits only on assigned timeslot bit-times.

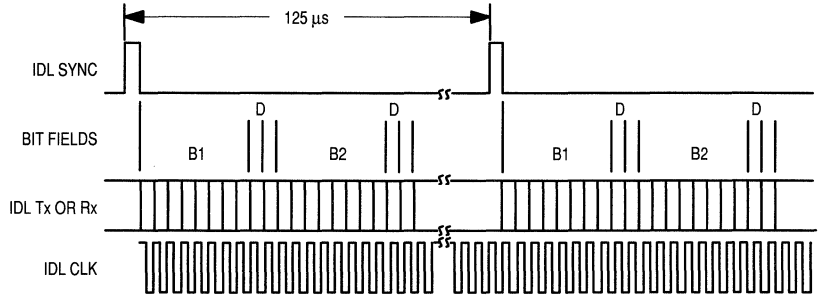


Figure 6-2. IDL Frame Timing

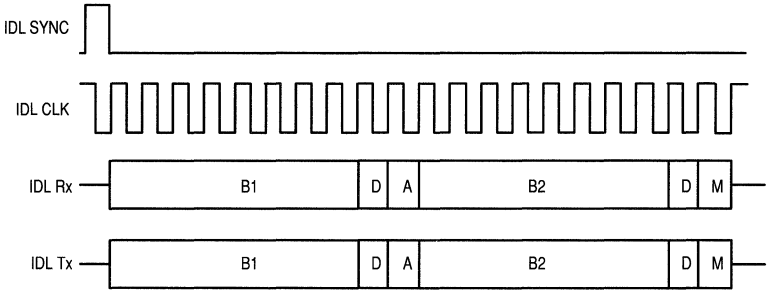


Figure 6-3. IDL Bus Signals

The 20-bit frames are composed of five bit-fields. The first eight bits contain data on the 64 kbps B1-channel. The ninth bit contains one bit of the D-channel. The tenth bit is an auxiliary bit which is not used by the DDLC. Bits 11-18 contain the 65 kbps 32-channel. The 19th bit is the second bit of the D-channel and the 20th bit is another auxiliary bit which is not used by the DDLC. When the DDLC operates on the D-channel, it uses bit 9 and bit 19 of the IDL frame. The two bits cannot be separated. The D-channel is a 16 kbps channel which is used for communication between an ISDN terminal and the network. When the DDLC is configured for use with the IDL D-channel, it directly supports the D-channel contention algorithm of the MC145474 S/T transceiver. Each channel may operate on either of the B-channels or the D-channel. With both channels configured in the IDL mode, the DDLC has the capability of handling data on either B-channel and the D-channel or on both B-channels simultaneously. If the two channels operate on the same IDL frame, care must be taken so that the transmitters are not programmed to output data in the same bit-times.

6.1.2 Subrate Multiplexing (CCITT I.460)

Normally, all eight bits of an ISDN B-channel are grouped into one 65 kbps bearer channel. However, a B-channel may be thought of as composed of eight 8 kbps channels. CCITT

Specification I.460 outlines a technique where the individual bits in a B-channel are accessed and treated as individual channels. The DDLC meets the requirements of I.460 as it can access individual bits in the B-channels for multiplexer or other non-ISDN applications. A Bit Mask register is provided for each channel which selects the individual bits in the 8-bit field which are active. The B-channel bit mask registers are programmed with '0s' in the positions where the bits are active and '1s' where the bits are ignored. Note that the least significant bit is transmitted first so the LSB of the Mask register corresponds to the first bit transmitted in an IDL frame. For example, if bits 2 and 3 are to be active in a particular application, the coding in the Bit Mask register would be 1111 0011 (F3 hex). The receiver will ignore all bits in bit positions other than b2 and b3 and the transmitter will transmit only on these two bit positions. On all other positions, it will be high-impedance or '1s' depending on the setting of the Mask Drive bit in the Transmit Control register. When high-impedance is selected, other devices on the IDL bus may transmit during the inactive bit-times. For applications without other devices sharing a B-channel, '1s' may be transmitted eliminating the need for a pull-up resistor on the bus. The bit mask feature is usable on IDL B-channels. In D-channel operation, the bit mask is disabled.

Figure 6-4 depicts subrate multiplexing where b2 and b3 of the B1-channel are active. Other devices on the IDL bus may transmit on the first two and last four bits of the B1-channel or the B2 or D-channels. Subrate multiplexing is not supported when operated in transparent mode.

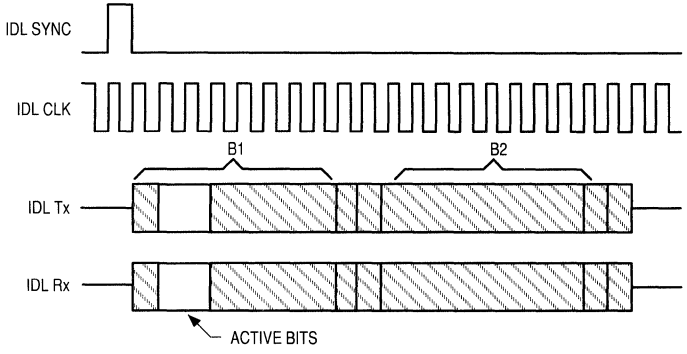


Figure 6-4. IDL Subrate Multiplexing

6.1.3 Transparent Operation

The DDLC provides a method to pass unformatted (unframed) data between the serial interface and memory. Internal synchronization signals are generated which delineate byte boundaries. Received data is organized into bytes and passed to memory through the DMA controller. Bytes from memory are transmitted upon command by the internal sync signal. The user has control of the byte orientation within an 8-bit timeslot (transparent operation is not supported with D-channel operation.) A bit pattern is programmed into the Bit Mask register which corresponds to the byte alignment desired. Other alignments are possible by moving the position of the zero in the Bit Mask register. Table 6-1 describes the possible bit patterns and their consequent alignments. Transmit and receive use the same Mask register so their byte alignments are the same. The timeslot byte alignment needs to be programmed during DDLC initialization.

Table 6-1. Timeslot Byte Alignment

msb	lsb		
0111	1111	7F hex	Aligned to Timeslot Boundary (b0)
1011	1111	BF hex	Aligned to b1
1101	1111	DF hex	Aligned to b2
1110	1111	EF hex	Aligned to b3
1111	0111	F7 hex	Aligned to b4
1111	1011	FB hex	Aligned to b5
1111	1101	FD hex	Aligned to b6
1111	1110	FE hex	Aligned to b7

Figure 6-5 depicts the case where 7F (hex) is loaded into the Bit Mask register. Data bytes are aligned such that the seventh bit in a timeslot is shifted into the b0 position of the data bytes stored in memory. All other received bits follow sequentially. As Figure 6-5 shows, the internal sync signal is actually offset by 1/2 bit time to eliminate internal edge skew. The user does not see this offset in the IDL or timeslot modes, however, in the modem mode this is important. Section 6.3.1 describes this more fully.

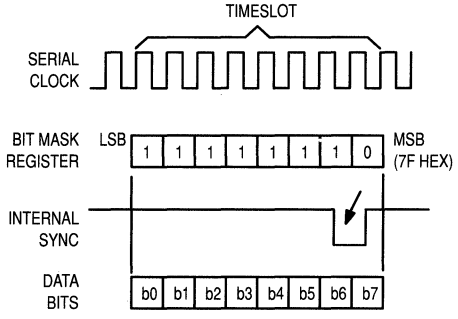


Figure 6-5. Transparent Byte Alignment

6.2 TIMESLOT INTERFACE

A PBX PCM highway timeslot interface may be enabled for network applications. Timeslots of eight bits each are enabled so the DDLC shares a bus with other devices. The transmitter transmits up to eight bits in a timeslot, then goes high-impedance. The receiver samples bits only in its assigned timeslot and ignores all other bits. Subrate multiplexing may be used in timeslot operation in a similar manner as the IDL mode. Figure 6-6 describes a typical PCM bus application.

Two separate continuous clocks (not shown) at 1.536 MHz, clock the transmitters and receivers on the bus. Individual devices are activated for their assigned eight-bit timeslot. Figure 6-7 shows a device active on timeslot 2. The other 23 devices are active on the PCM highway in their assigned timeslots.

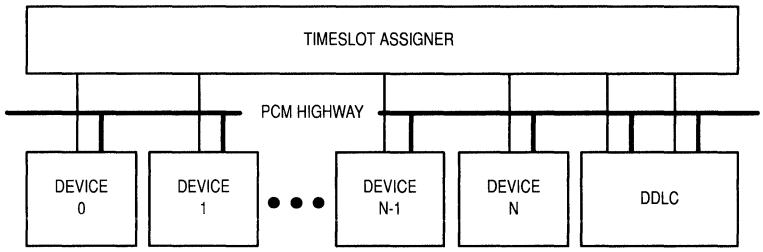


Figure 6-6. PCM Highway Application

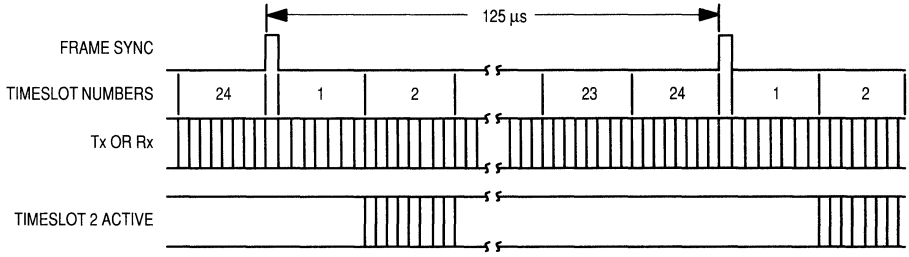


Figure 6-7. PCM Bus

External circuits called Timeslot Assigners (TSACs) count timeslots using the frame sync to keep track of frame boundaries. They output enable signals to the devices on the bus so that each device is active for only its assigned slot. Two modes of operation are common in industry, long-frame and short-frame. TSACs are available which provide long- or short-frame synchronization signals. The DDLC supports both operating modes and automatically switches between them on a frame-by-frame basis. Note: When switching between modes, one frame time is needed to synchronize operation.

6.2.1 Mode Selection

When the timeslot mode is first elected, short-frame operation is assumed. An eight-bit transfer is made when the enable signal goes high. During the third bit-time, the enable signal is sampled and if it is low, short-frame operation is selected for the next frame. If the enable signal is high, long-frame operation is selected. The serial interface samples the enable signal on every eight-bit transfer and adjusts its timing for the next transfer. For best operation, the operating mode should be fixed to either long- or short-frame operation. The transmitter and receiver are independent of each other so one may be in long-frame and the other may be in short-frame operation.

When the timeslot is first enabled, the Transmit Enable bit in the Transmit D Control register should be '0'. This forces the TxD pin to high impedance. After a 127 µs frame time, the serial interface block of the DDLC is synchronized to the timeslot. The transmitter then can be enabled and will transmit on its proper timeslot.

6.2.2 Long-Frame Operation

In long-frame operation, an enable signal is presented to the DDLC which encloses the eight bits of a timeslot. The first transmitted data bit in a timeslot is enabled on the rising edge of the Tx Enable or first Tx Clock rising edge, whichever is later. The next seven bits are output on succeeding rising clock edges. The Tx Enable signal normally returns low between the seventh and eighth falling clock edges, but it may return low as early as the fifth rising clock edge. If it remains high after the eighth clock, the transmitter continues transmitting bits until the enable goes low. The receiver is similar except that it samples the Rx Data pi on falling clock edges while Rx Enable is high. The long-frame mode expects a continuous clock for proper operation. For nonstandard applications active timeslots may be separated by as few as four clock pulses. Figure 6-8 describes typical long-frame operation. Separate clocks may be provided to the transmitter and receiver for asynchronous applications. If a single transmit/receive clock is used, the Tx and Rx clock pins may simply be connected together. Similarly, if transmit and receive are on the same timeslot, Tx Enable and Rx Enable are connected together.

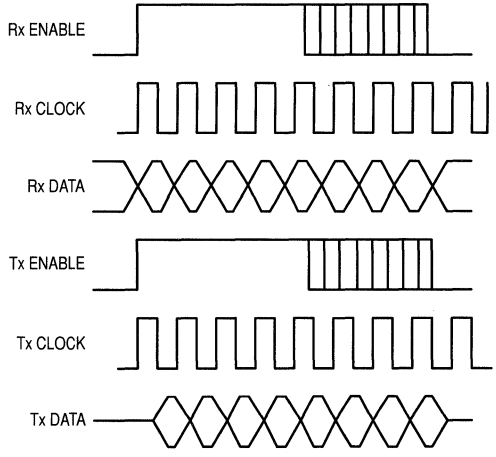


Figure 6-8. Timeslot Operation Long-Frame

6.2.3 Short-Frame Operation

Short-frame operation is similar to IDL B1 operation. A one-clock wide sync signal is presented to the Tx Enable pin during the clock period immediately preceding the eight bits of a timeslot. The transmitter outputs its eight bits on the eight rising clock edges of the clock following the sync. The receiver samples the Rx Data pin on the eight falling clock edges following the sync. Figure 6-9 describes the operation of short-frame operation. As with long-frame active timeslots may be separated by as few as four clock pulses.

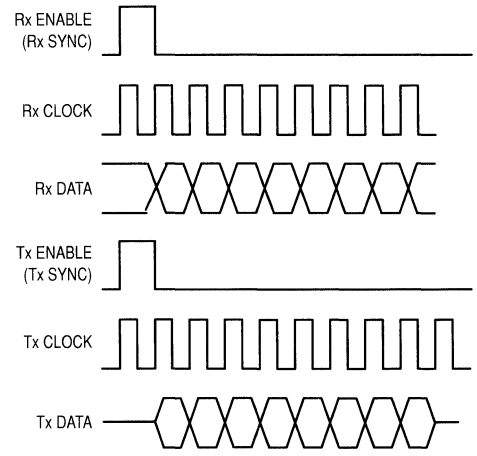


Figure 6-9. Timeslot Operation Short-Frame

6.2.4 Subrate Multiplexing

The subrate multiplexing bit mask function of the IDL interface operates identically in the timeslot mode. See Section 6.1.2 for a complete discussion. Placing a '1' in a particular bit in the Mask register causes the receiver to ignore that bit and the transmitter transmits either a high impedance or '1' for the bit-time depending on the setting of the Mask Drive bit in the Transmit Control register. Subrate multiplexing is not supported when a transmitter/receiver is operated in transparent mode.

6.2.5 Transparent Operation

Transparent operation in the timeslot mode is similar to operation in the IDL mode. A synchronization signal is programmed into the Bit mask register which defines byte alignment through the receiver and transmitter. Because the transmitter and receiver share the Bit Mask register, byte alignment within transmit and receive timeslots must be the same even though the timeslots are not related to each other. See Section 6.1.3.

6.3 MODEM INTERFACE

For those applications that do not use a timeslot-type interface, the DDLC provides a modem-style interface. In the modem mode seven pins transfer data and control information. There are transmit and receive clock inputs (Tx CLK and Rx CLK), transmit and receive data pins (TxD and RxD), and modem control lines ($\overline{RTS}$, $\overline{CTS}$, and $\overline{CD}$). The transmit and receive clocks are independent on each other and may be asynchronous to the microprocessor system clock (MCLK). These clocks may have a frequency of up to MCLK/2. It must be noted, however, that the data throughput of the DDLC is limited by the microprocessor bus bandwidth available to the DDLC and the rate at which the MPU can service interrupts. See Figure 6-10.

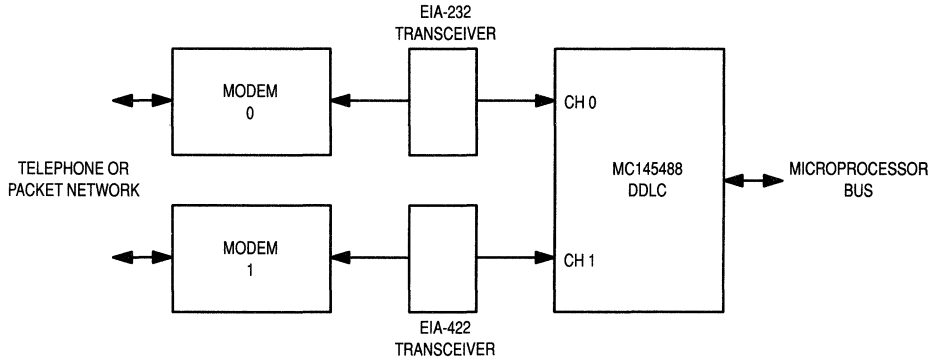


Figure 6-10. Modem Application

Data bits from the TxD pins are presented on falling edges of the Tx CLK (this is inverted from the timeslot and IDL modes). Data bits on the RxD pins are sampled on the rising edges of the Rx CLK (again, this is inverted). The modem control lines are used to control the flow of data. Request-to-Send ($\overline{RTS}$) is an active low output which indicates that the DDLC has data to transmit. Clear-to-Send ($\overline{CTS}$) is an input which indicates that the DDLC has permission to transmit. The CTS bit in the Transmit Status register indicates the current state of this pin. If $\overline{CTS}$ goes inactive (high) for more than one Tx CLK bit time while a frame is being transmitted, the DDLC immediately aborts the frame and prepares to retransmit the frame when $\overline{CTS}$ again goes active. Carrier Detect ($\overline{CD}$) is an input which indicates that the data on the RxD pin is valid. The CD bit of the Receive Status register indicates the current state of this pin.

The bit mask feature is not available in the modem mode. Data is transmitted continuously, not in eight bit clusters, so masking bits is not appropriate.

6.3.1 Transparent Modem Operation

The modem mode can operate with transparent data in a similar manner to the IDI and timeslot modes. Unlike the other two modes, however, the user must provide a synchronization signal to indicate byte alignment of the serial data. The transmitter and receiver maintain the given alignment through the bit handlers and FIFOs to memory. The synchronization signal is a one-clock wide active-low pulse. A byte boundary occurs 1 1/2 clocks following the falling edge of the signal. Figure 6-11 describes the operation of the receive byte sync. Transmitter operation is similar. Notice that the sync signal edges occur on rising clock edges (middle of a bit time) so that proper set-up and hold times will be met when the signal is sampled on a falling clock edge.

Synchronization signals on transmit should occur on eight-bit boundaries but is optional after the first pulse as the transmitter maintains byte sync in the absence of sync signals. Resynchronization occurs on every sync pulse, so if sync pulses occur every eight bits, synchronization is maintained with the original alignment. For receive, sync pulses should also occur on eight-bit boundaries and are optional after the first pulse. The synchronization signal to the receiver resets an internal modulo-eight counter which generates a load signal to the FIFO. If synchronization pulses occur more frequently than every eight bit-times the counter will never generate a load signal and data will not be transferred to the FIFO.

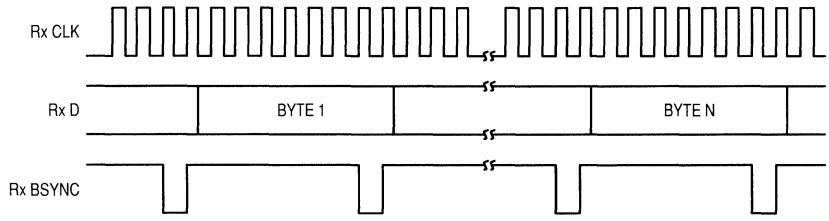


Figure 6-11. Transparent Modem Operation

6.4 SERIAL CONTROL PORT (SCP)

The DDLC has a serial control port which may be used to communicate with external devices through a serial mechanism. The SCP is a four-wire full-duplex communication system which exchanges 8-bit bursts of data with SCP devices. The DDLC may be programmed to be an SCP master which generates the enable and clock signals or an SCP slave which accepts clock pulses from an external master. The SCP is compatible with Motorola's serial peripheral interface, available on most single chip MCUs, and National Semiconductor's Microwire Plus.

Figure 6-12 describes a typical application of the IDL and SCP buses in an ISD terminal. In this example, the SCP bus passes control information to the internal registers of the S/T transceiver and codec/filter to set operating modes and read status information and the DDLC is an SCP master. In other applications the DDLC may be an SCP slave.

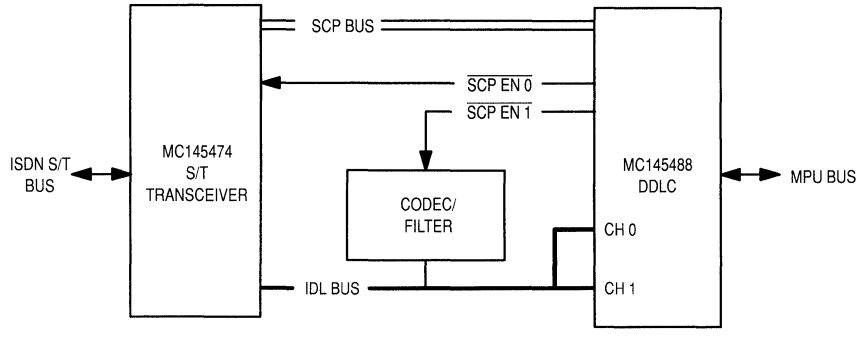


Figure 6-12. ISDN Terminal Application

6.4.1 Serial Control Port Description

The SCP module in the DDLC consists of a baud rate generator, a shift register, a control clock, and several control bits. The shift register is parallel read/write and connects directly to the data bus when addressed. Data may be written or read at any time except when a transfer is in progress. If the register is written at this time, the write is ignored and if the register is read, data reads 00 hex. Control and status bits may be read or written at any time. The control block generates an interrupt on the rising edge of slave select in the slave mode or generates eight clock pulses in the master mode. The baud rate generator selects one of four clock rates derived from MCLK. The serial interface block multiplexes the SCP functions onto the modem control pins when the SCP is enabled. See Figure 6-13.

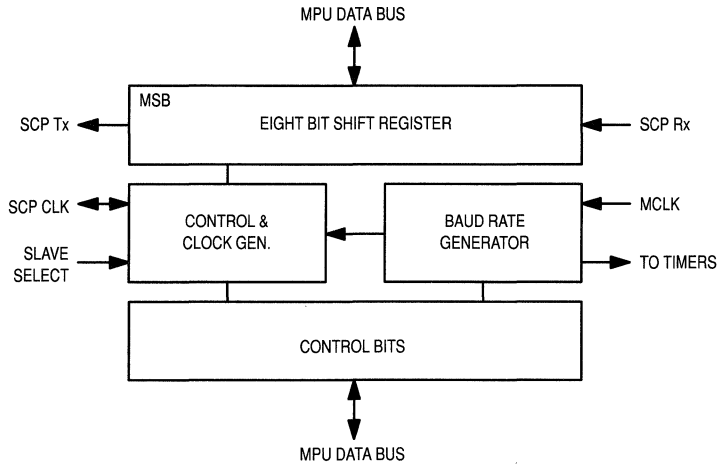


Figure 6-13. SCP Module Block Diagram

The SCP module is enabled by setting the SCPE bit of the Channel 0 Serial Interface Control register to '1'. The modem control pins of channel 0 are forced into the SCP function and override any other functions that may have been selected. The channel 1 $\overline{CD}$ pin is controlled by the SCPE bit of Channel 1 Serial Interface Control register and becomes a second SCP Enable ($\overline{SCP\ EN}$ 1) pin when this bit is set to '1'. $\overline{SCP\ EN}$ 0 or $\overline{SCP\ EN}$ 1 must go low when the SCP clock is in its inactive state. For example, when the SCP clock is used in the normal mode SCP enable must go low when the SCP clock input is low. When the SCP clock is used in the inverted mode SCP enable must go low when the SCP clock input is high.

6.4.2 SCP Master Mode

$\overline{SCP\ EN}$ 0 and $\overline{SCP\ EN}$ 1 are actually general purpose outputs which may be used to enable SCP slave devices. When low, the enable signals indicate that the salve has been selected and should expect a data burst. SCP CLK is an output which provides the shift clock for the data. The clock is normally low until a burst of data is transmitted. Then the clock is active for eight pulses. SCP Tx is an output which transmits data MSB first to the slave device. New data bits are presented on the rising edges of SCP CLK. SCP Rx is the input which accepts data from the slave devices. This pin is sampled on the falling edges of SCP CLK. **Note:** The SCP CLK is selectable as to its polarity. Normally, the transmitter outputs bits on rising edges and the receiver samples bits on falling edges. When the clock polarity is inverted, the transmitter outputs on falling edges and the receiver inputs on rising edges. Figure 6-14 shows the normal polarity case.

When a message is to be sent to a slave device, the information is placed in the SCP Tx/Rx register. Then Enable 0 or Enable 1 in the SCP Status/Control register are set to '1' depending on the application. This activates the selected enable pin to a logic low. The host then activates the Transmit bit and the SCP controller generates eight clock pulses on the SCP CLK pin and shifts the data on the eight rising edges. Data is shifted most significant bit first. The slave transmitter, at the same time, shifts its data (if any is ready) to the DDLC SCP Rx pin on the rising SCP CLK edges. The DDLC samples the SCP Rx pin on the falling SCP CLK edges and replaces the transmitted data with the received data in the Tx/Rx register on a bit by bit basis. After the eight shifts have occurred,

the Transmit bit is cleared indicating that the exchange was completed and the SCP Complete interrupt is generated. The host then resets the Enable 0 or Enable 1 bit which causes the enable pin to return to a logic high. Typically the user will poll the SCP IRQ bit to see when the 8-bit exchange is complete. The IRQ bit is a status bit which generates an interrupt if the SCP Complete interrupt is enabled.

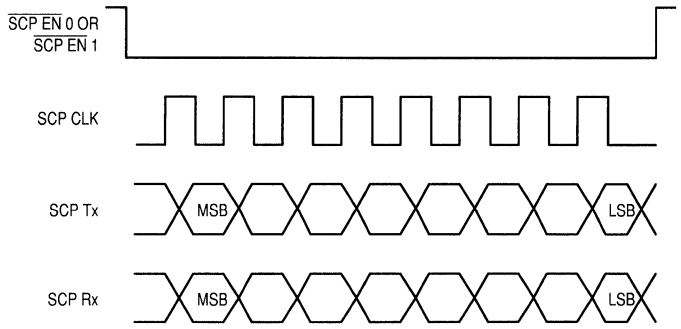


Figure 6-14. SCP Pin Functions — Master Mode

The SCP EN pins are directly controlled by the bits in the SCP Control register. The pins change state within one MCLK after the end of the MPU bus cycle which changed the bits. The SCP Tx bit causes the SCP state machine to start transmitting within one and one-half SCP clocks after the bus cycle which changed the bit. Recall that this can be 24 MCLK periods if the ÷ 16 prescaler tap is selected. **Note:** The SCP register should not be written within one and one-half SCP clocks after the SCP Complete interrupt is generated (this could be up to 24 MCLKs with the Ö 16 prescaler selected).

The user can select an appropriate baud rate for the application. The Baud Rate bits in the SCP Status/Control register select a tap on a divider chain which generates the SCP CLK signal. See Section 3.4 for a listing of the divide ratios available.

6.4.3 SCP Slave Mode

When the SCP is in the slave mode, SCP EN 0 becomes an input, Slave Select. SCP EN 1 remains a general purpose output if it is enabled by setting the SCPE bit in the Serial Interface Control Register 1 (Address 40 hex). SCP CLK also becomes an input. The SCP module now waits for the Slave Select pin to go low and accepts data until the rising edge of slave select. Data bits are input on the SCP Rx pin. Any data in the Tx/Rx register is shifted out, MSB first on the SCP Tx pin, on positive SCP CLK edges (negative edges if the Clock Invert bit is set). When the rising edge of slave select is detected, the SCP Complete interrupt is generated.

In the slave mode any clock frequency from dc to MCLK/2 is acceptable. The clock may be continuous but is utilized internally only when the slave select is low. Data is shifted into and out of a shirt register simultaneously. If slave select is low for fewer than eight clock pulses, data that was in the shift register before slave select went low will remain in the register and will be shifted into the low order bit positions. If slave select remains low for more than eight clock pulses, received data will be shifted out starting on the ninth clock pulse and will continue to shift until slave select goes high.

Figure 6-15 describes a typical SCP slave exchange. In this case the Clock Invert bit is '0'. The SCP Tx pin outputs the first data bit on the first rising clock edge following the falling edge of slave select. The SCP Complete interrupt is generated on the rising edge of slave select. The slave select pin must transition to low while the SCP CLK is inactive (low while CI is 0 and high while CI is high).

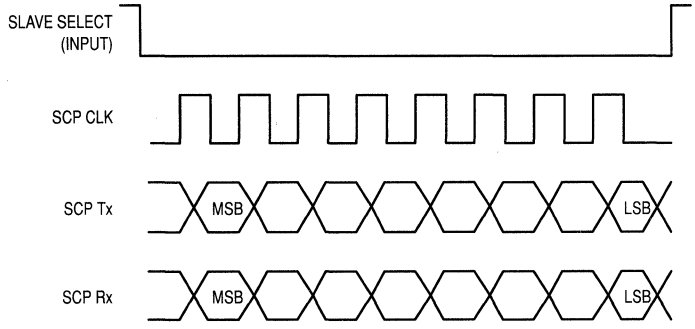


Figure 6-15. SCP Pin Functions — Slave Mode

SECTION 7
APPLICATIONS

The DDLC was designed for ISDN systems, but it finds many non-ISDN applications. This section will describe two ISDN applications. A non-ISDN application such as X.25 PAD is similar in concept to the ISDN devices but the physical interface is different.

7.1 ISDN VOICE/DATA TERMINAL ADAPTER

The DDLC and its companions, the MC145474 S/T transceiver and the MC145554 codec/filter easily connect to build a powerful IDSN terminal with voice and data capability. This section will describe the design of a stand-alone 2B + D terminal adapter which provides an asynchronous EIA-232 interface for an external dumb terminal. The terminal adaptor (TA) performs packet assembly/disassembly (PAD) functions for the terminal and transports data packets on either B-channel or the D-channel. PCM voice is transported on the remaining B-channel. This example assumes that 16K RAM and 64K ROM is sufficient to support firmware and buffering needs. Figure 7-1 is a block diagram of the TA.

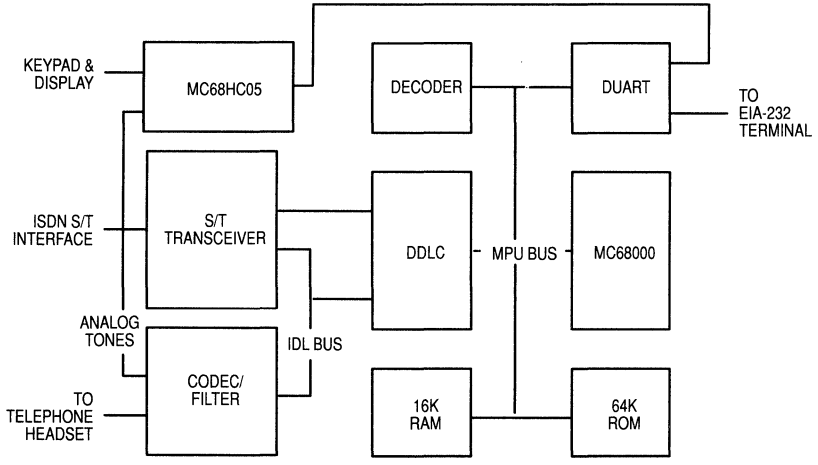


Figure 7-1. ISDN Terminal Adapter and Voice and Data Block Diagram

Terminal adapters may have many configurations. This example converts asynchronous EIA-232 data into V.120 format for transport on either one B-channel at 64 kbps or the D-channel at 16 kbps. Firmware in the TA provides packet assembly/disassembly functions for the asynchronous data. Voice calls may be placed on the second B-channel. Again, firmware handles call setup and teardown signaling with the network. A microcomputer is provided to control a keypad and display. These are used to place voice and data calls and to display local and network status messages. The microcomputer communicates with the MC68000 host through one of the asynchronous ports

on the MC68681 DUART. Call progress signals such as dial tone and Touch Tone® beeps are generated in the microcomputer and passed to the telephone handset through the codec/filter.

This basic design can be expanded to support more asynchronous channels by expanding RAM and ROM (if necessary) and by adding extra DUARTs. V.120 software supports statistical multiplexing which interleaves packets from multiple sources onto the ISDN B- or D-channels.

7.1.1 IDL and SCP Interfaces

The data backbone of the system is the IDL bus. Figure 7-2 describes the interconnection between the S/T transceiver, the DDLC, and the codec/filter. The IDL bus passes actual data which is transferred on the ISDN link to the network. The SCP bus passes status and control information between the MC145474 S/T transceiver and the host processor using the SCP module in the DDLC.

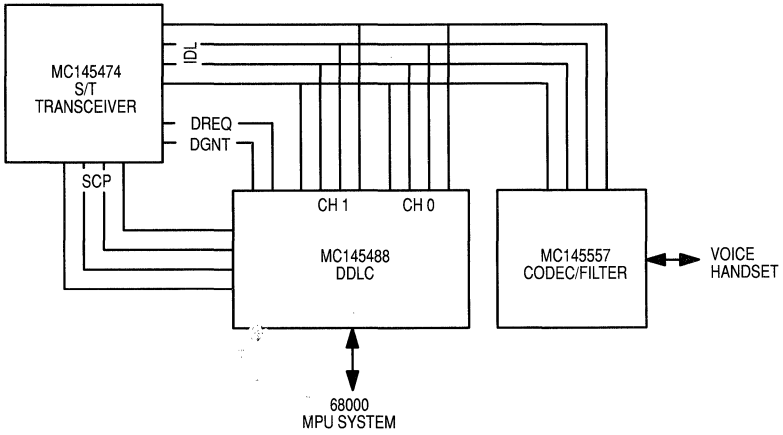


Figure 7-2. IDL, SCP Bus Connections

On both IDL and SCP buses, master transmitters are connected to slave receivers and slave transmitters are connected to master receivers. The MC145474 S/T transceiver is the IDL master and the DDLC is the SCP master. Refer to Table 4-4 which defines pin names and data directions and cross references them to the actual pin numbers on the DDLC. Refer to the MC145474 S/T transceiver and MC145554 codec/filter data sheets for correct pin connections.

All three chips simply connect together on the IDL bus and the DDLC and S/T transceiver connect together on the SCP bus. The codec/filter operates on IDL B1-channel and the DDLC will be programmed with software to communicate on IDL B2 and IDL D-channels.

7.1.2 MPU/DMA Interface

The DDLC was designed to fit into a 68000 host system with minimal glue logic. For the high-performance terminal adapter in this example, virtually all glue logic is contained in two PALs. Because the DDLC is the only bus master other than the 68000, bus arbitration is handled directly by the 68000. Interrupts from the DUART, DDLC, and S/T transceiver are directly connected into the three IPL inputs of the 68000. The Interrupt Acknowledge (IACK) signal to the DDLC is also generated in one of the PALs.

The RAM size is small (16K bytes) so static RAM was chosen for simplicity. Similarly, the ROM is small so two 32K x 8 EPROMs are used. Memory is arranged in even and odd byte address banks to be compatible with the DDLC and 68000 memory structure.

Figure 7-3 describes the memory map used in this example. Eight 16K byte pages are used. ROM occupies the lowest four pages. RAM is actually one 16K byte page, but since full address decoding is not used, an alias page, beginning at 1400, appears. The DDLC and DUART are not fully decoded so they appear on many alias addresses within their respective 16K pages. As the DUART is an 8-bit peripheral, it occupies only the odd bank of its 16K byte page, all accesses to the DUART are on odd addresses.

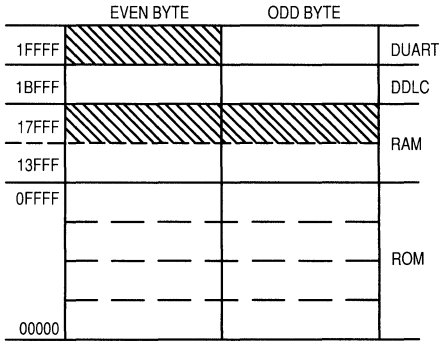


Figure 7-3. 128K Byte Memory Map

The DDLC $\overline{\text{OWN}}$ pins are provided so that the page on which the buffer RAM resides can be accessed. In this example, the OWN pins are used to generate a chip select for the RAM which resides on the fifth page in the system. Expanding the ROM or RAM beyond this example is equate easy. Simply going to 32K byte pages doubles all available blocks without changing the address decoder block. The address pins are just connected to the next most significant address pins on the system bus. Other applications use the $\overline{\text{OWN}}$ pins to generate function codes and further expand the memory. Section 7.2 describe such a system.

7.1.2.1 INTERRUPT VECTORS. This design example has three interrupt generating devices, the MC145474 S/T transceiver, the DUART , and the DDLC. The 68000 has three interrupt input pins which can be encoded to produce up to seven interrupt priority levels. Level 7 is the highest priority and level 1 is the lowest. The DDLC $\overline{\text{IRQ}}$ pin is connected to IPL1 generating a level 2 interrupt and the S/T transceiver $\overline{\text{IRQ}}$ pin is connected to IPL0 generating a level 1 interrupt. Because the DUART will be working at relatively high speeds, up to 19.2 kbps, its interrupt priority is set to level 4 by connecting its $\overline{\text{IRQ}}$ pin to IPL2. After the 68000 recognizes that an interrupt has been generated, it stacks selected internal registers then fetches a vector number from the interrupting device. The vector fetch cycle, known as an interrupt acknowledge cycle, is similar to a read cycle except that the $\overline{\text{IACK}}$ pin on the peripheral device is activated rather than the $\overline{\text{CS}}$ pin.

The DDLC generates up to 2 interrupts and encodes them into 15 unique vectors. Several interrupt handlers may be built in software, each servicing specific interrupts. The DUART also generates its own vectors in a similar manner to the DDLC, but this design uses the autovectoring technique of the 68000 to service the DUART and the S/T transceiver interrupts. Autovectoring operates as follows. When the 68000 fetches a vector from the interrupting device, if it detects $\overline{\text{VPA}}$ asserted

in place of $\overline{\text{DTACK}}$, it terminates the cycle and vectors to the function addressed by the autovector area of the vector map for the current interrupt level. Seven autovectors are provided, one for each priority level. One of the PALs is programmed to generate $\overline{\text{VPA}}$ when a level 1 or level 4 IACK cycle is performed causing an autovector.

The starting addresses of the interrupt handlers must be placed in their proper locations in the interrupt vector map during the TA's restart sequence. The programmer should not enable any interrupts in the system until all vectors have been placed in the vector map.

7.1.2.2 DMA BUS ARBITRATION. The DDLC has a built-in DMA controller which connects to the 68000 quite simply in this TA example. The three DMA arbitration pins, $\overline{\text{BR}}$, $\overline{\text{BG}}$, and $\overline{\text{BGACK}}$ directly connect to their counterparts on the 68000. All three signals need pull-up resistors of about 50 kΩ. A more complex design with multiple MPU bus masters normally uses a bus arbitration module.

7.1.2.3 DECODER PALs. Chip count of this terminal adapter is minimized by using two PALs (programmable array logic). This section describes the operations performed by the PALs and their design equations. Chip selects and control signals for RAM and ROM are generated in the address decoder PAL. $\overline{\text{DTACK}}$ for RAM and ROM is generated in a second PAL. Recall that the DDLC and DUART generate their own $\overline{\text{DTACK}}$ signals. The address decoder PAL is a 20L8 device and the DTACK generator is a 16R4 device. Table 7-1 describes the input and output signals of the PALs, while Table 7-2 defines the design equations.

Table 7-1. System Decoder PALs

Address Decoder (10L8)		DTACK Generator (16R4)	
Inputs	**Outputs**	**Inputs**	**Outputs**
AS	/ROM_CS	ROM_CS	/DTACK0
A2	/RAM_CS	RAM_CS	/DTACK1
A14	/DDLC_CS	MCLK	/RAM_DTACK
A15	/DDLC_IACK	Q1	Q0 (Internal Terms)
A16	/DUART_CS	Q2	Q1 (Internal Terms)
A17	/WE	ENABLE	
OWN0	/OE		
OWN1	/VPA		
RW			
DS			

Table 7-2. PAL Design Equations

Address Decoder (20L8)	
/DUART_CS =	/AS•A16•A15•A14
/DDLC_CS =	/AS•A16•A15•/A14
/DDLC_IACK =	/AS•A17•A2
/ROM_CS =	/AS•/A16
/RAM_CS =	/AS•A16•/A15•OWN 0•OWN 1 + /AS•/A15•/OWN 0•OWN 1 + /AS•/A15•OWN 0•/OWN 1
/VPA =	/AS•A17•/A2
/WE =	/DS•/RW•/AS•A16•/A15•OWN 0•OWN 1 + /DS•/RW•/AS•/A15•/OWN 0•OWN 1 + /DS•/RW•/AS•/A15•OWN 0•/OWN 1
/OE =	/DS•RW•/AS•A16•/A15•OWN 0•OWN 1 + /DS•RW•/AS•/A15•/OWN 0•OWN 1 + /DS•RW•/AS•/A15•OWN 0•/OWN 1
DTACK Generator (16R4)	
Q0 =	/ROM_CS (Clocked by MCLK)
Q1 =	/ROM_CS•Q0 (Clocked by MCLK)
Q2 =	/ROM_CS•Q0•Q1 (Clocked by MCLK)
/DTACK 0 =	Q0•Q1•/ROM_CS (Normal DTACK)
/DTACK 1 =	Q1•Q2•/ROM_CS (Delayed DTACK)
/RAM_DTACK =	/RAM_CS
/OE (Input) =	/ROM_CS
CLK (Input) =	MCLK

NOTE: The three DTACK signals are three-state signals which are high impedance while their respective chip selects are inactive.

The address decoder operates on 16K byte pages allocating four pages to ROM and one page each to the DDLC, the DUART, and RAM. As full address decoding is not used, RAM appears at an alias page beginning at address 14000 and the DUART and DDLC appear at many alias addresses in their respective pages. The IACK cycle is detected, not by using the function codes, but by detecting the assertion of A17. When the 68000 does an interrupt vector fetch, it puts addresses between FFFFF0 and FFFFFF on the bus. This is the only time in this design that the 68000 accesses an address where A17 is asserted, so A17 makes a convenient interrupt fetch detector. When A2 is asserted during an interrupt fetch cycle, indicating a level 2 interrupt vector fetch, $\overline{IACK}$ to the DDLC is asserted. When A2 is negated, $\overline{VPA}$ is asserted low causing the 68000 to vector to the level 1 or level 4 autovector.

DTACK signals are generated from ROM and RAM. Normal and delayed DTACKs are provided for ROM enabling the use of slow EPROMs if desired. Fast static RAM is assumed, so RAM_DTACK is simply RAM_CS converted into an open-drain signal.

7.1.3 User Interface (Microcomputer Keyboard/Display Controller)

A microcomputer, configured as a smart peripheral, is a simple but powerful way to provide a user interface into the TA. Typically a keypad and LCD display provide input means and feedback to the

user. Ordinary telephones provide audio feedback (signalling tones, etc.) to the user, so the microcomputer in this example also generates these tones. While the microcomputer is controlling the keyboard and display, it communicates with the 68000 host to pass keypad inputs to the TA and status signals to the display. A simple protocol using the asynchronous port of the MC68HC05 and one of the two DUART channels is an inexpensive and reliable way to provide two-way communication between the processors.

To be user friendly, an ISDN terminal adapter must be similar in operation to an ordinary telephone. Because it has several features not on ordinary telephones, it will have special function keys and a display of some sort. When the user wishes to place a call, the user places the TA in the off-hook state and dials the number — just like an ordinary telephone. The MCU reports the hookswitch status to the host to prepare the TA for an outgoing call. When the host acknowledges the status, the MCU reports keypad inputs to the host so call-setup messages can be prepared for transmission to the network. Terminating a call is also similar to that of any ordinary phone. Voice and data calls are handled similarly.

A simple message protocol should be implemented between the host and MCU, consisting of a status byte followed by data bytes. Optional error checking can be provided, but probably is not necessary. A typical speed for the data link would be 9600 baud with operation in full duplex me. Almost any message format may be used, but it must be simple enough so that it does not burden either the MCU or the host. The MCU should be the link master, so that if the 68000 needs service, or has a message, it should request permission to transmit before continuing.

A basic set of messages from the MCU to the host would consist of status information such as "Key x Pushed", "Go Ahead", and "Message ACKnowledged". Others may be implemented as needed. Messages from the host include "Display Message xxx", "Enable Tone yy", "Message ACKnowledged", and "Set Test Mode", among others. Details of how the message protocol is implemented is beyond the scope of this data sheet. Typically, the host processor in a system does not handle low level tasks such as keyboard scanning and display driving. Using a microcomputer as a smart peripheral makes more sense from both the hardware and software aspects. The MCU has quite a bit of power which may be exploited by handling certain aspects of the TA state control in its firmware. Specific partitioning is up to the system designer.

Using an asynchronous link between the host and MCU makes good use of available hardware. The DUART has a second channel which would not normally be used and the MCU has an asynchronous communication module on-chip. Separation of functions between the host, which is controlling the PAD functions, and the MCU which is controlling the keyboard and display is clean. Only when a key is pressed, or when a new message is to be displayed is there any communication between the 68000 and MCU. This leaves the host free to handle the PAD functions both for signaling and data transmission with interruptions only during state changes requested by the user or by the network.

7.1.4 Real-Time Operating System

A small real-time operating system coordinates the PAD functions with real-time inputs from the microcomputer. The operating system is capable of controlling several instances of layer 2 (LAPD) and layer 3, each logically independent from one another (even though they may use the same physical ROM code). Each process has its own environment which distinguishes it from other similar processes. The low-level device drivers operate in an interrupt environment and communicate with their host processes through mailbox structures. Requests from the outside world (user or network) are handled by the operating system. Appropriate processes are spawned or terminated as

required. Figure 7-4 describes the interaction of the operating system with the processes, their environments, and the device drivers.

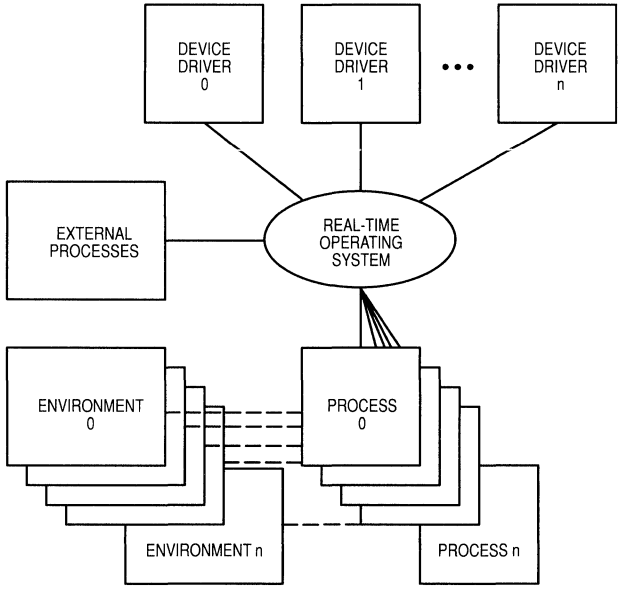


Figure 7-4. Real-Time Multitasking Operating System

The operating system keeps track of all processes. For example, if a voice call is in progress and a data call is requested, the operating system will spawn new logical instances of layer 1, 2, and 3 processes to control the data call on the B-channel. When the data call is terminated, the instances that were spawned are terminated. The voice call is independent and ignorant of the data call. The operating system keeps track of and controls the priority of each process and adjusts it as needed. Recall that network signaling messages have higher priority than packet messages.

Figure 7-5 is a high-level view of the software/hardware interface. The operating system is not shown, as the individual processes and device drivers are not aware of its presence. From the processors' point of view it serves as a post office which passes messages between layers and from layers to device drivers. Requests from the outside are processed through the management entity.

7.1.5 Summary

This section has described a conceptual design for a stand-alone ISDN terminal adapter. It is intended only to provide suggestions as to how a design might be done. The PAL equations are examples which are based on proven 68000 designs. The software which resides in ROM is simple to describe but complex to write. No attempt has been made to describe the complexities of LAPD or layer 3 software. The next section will go into some detail regarding layer 2 software along with the low-level device drivers.

Notice that the layer 1 device driver logically connects to the S/T transceiver even though it physically communicates via the DDLC's SCP block.

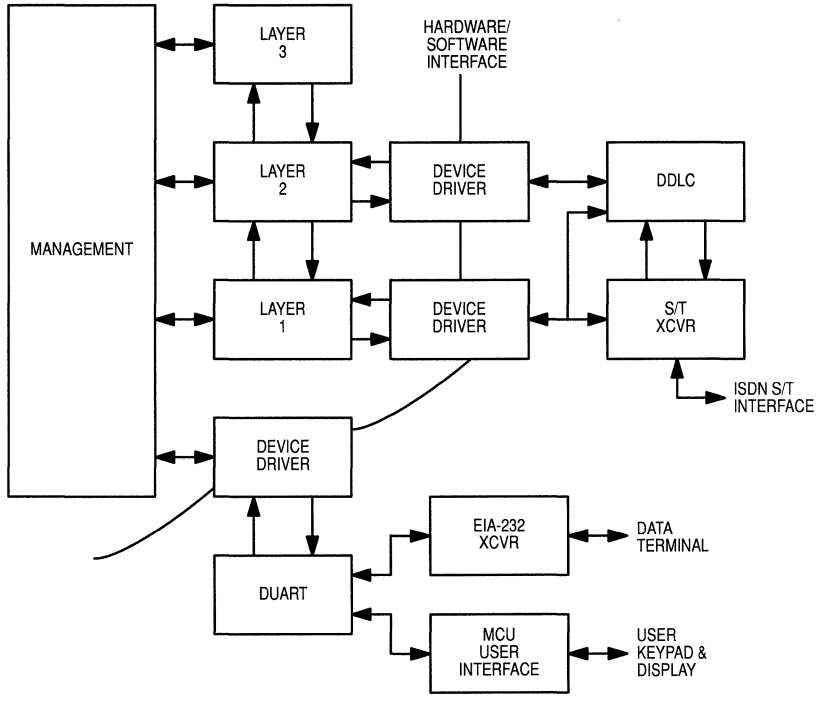


Figure 7-5. Terminal Adapter Software/Hardware Model

7.2 ISDN TERMINAL CARD FOR THE MACINTOSH SE

The Macintosh SE combined with an ISDN terminal card provides a powerful desktop communication tool. This section describes the design of an ISDN terminal which has the capability of transferring data at 65 kpbs over an ISDN B-channel while a voice call may be made on the second B-channel. Call set-up and tear-down is handled on the D-channel. The card uses the Macintosh's 68000 CPU and system memory. Application software resides on disk and is loaded into memory when the application is started.

The following discussion assumes that the reader is familiar with the Macintosh environment. The Macintosh is a powerful, elegant computer with a complex software package. Once the operating system and ROM Toolbox are understood the Macintosh is a programmer's dream. It is easy to build powerful, robust applications which have the "Mac-Look".

In many ways the stand-alone terminal adapter described in the previous section is similar to the Macintosh SE TA. This section will provide detailed descriptions of the hardware and software. Figure 7-6 is a block diagram of the terminal card hardware.

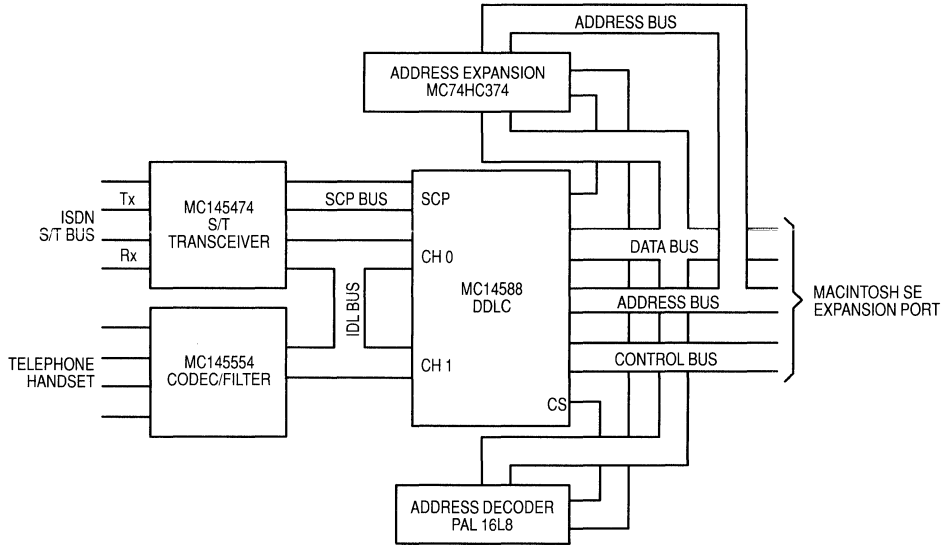


Figure 7-6. Macintosh SE ISDN Terminal Card Hardware Block Diagram

7.2.1 Terminal Card Hardware Description

This section describes the hardware of the terminal card. Brief digressions into three low-level software routines are included to clarify the hardware/software relationship.

7.2.1.1 THE MACINTOSH SE EXPANSION BUS. The Macintosh SE is the first Macintosh computer which provides an expansion slot so peripheral devices may be added to the system. Connection to the mother board is via a single 96-pin VME-type connector. Pin functions are unique to the Macintosh SE and are in no way compatible with the VME bus. For a detailed description of the expansion slot consult *Designing Cards and Drivers for the Macintosh II and Macintosh SE*, Addison-Wesley Publishing Company, 1987. This book is referred to as DC&D throughout the discussion.

7.2.1.2 DATA, ADDRESS, AND CONTROL BUSES. Basically, the expansion slot provides all MC68000 bus signals plus a few extras which are used by the custom circuitry in the Macintosh. For the most part, the DDLC is directly compatible with the signals provided. The data bus of the DDLC is directly connected to the data bus of the expansion port. No data transceivers are necessary.

The address bus in the Macintosh is 23 bits wide. Address decoders inside the Macintosh utilize all 23 lines to select specific devices or RAM areas. Since the DDLC provides 16 address lines, address expansion is required. A 64K byte block of RAM is used by the DDLC's DMA controller to hold data that is ready to transmit or that has just been received. An LS374 Octal Latch is used as an address expansion register which is enabled onto the address bus during DMA cycles when the DDLC owns the bus. The *DDLCbufferAlloc()* routine allocates buffer space from the operating

system and loads the upper byte of the buffer's base address into the '374 (see Figures 7-7 and 7-8). When a DMA cycle is executed the outputs of the '374 are enabled by the OWN pins onto the upper byte of the address bus providing effectively 23 address lines.

```
/ ********************************************************************************
  *                typedef of the complete DDLC register structure                *
  ********************************************************************************/

typedef short int INTEGER;
typedef unsigned BIT_FIELD;

/ *
  *      Individual register unions
  * /
    typedef union  {                              /* system control register */
      INTEGER SysCntr;
      struct    {
        BIT_FIELD                          : 8; /* 8-f: Dummy Bits */
        BIT_FIELD TEST                     : 3; /* 5-7: Factory Test Modes */
        BIT_FIELD DTACKEn                  : 1; /* 4: DTACK Enable */
        BIT_FIELD WAIT                     : 2; /* 2-3: Wait State Select */
        BIT_FIELD RESET                    : 1; /* 1: Software Reset */
        BIT_FIELD BW                       : 1; /* 0: Bus Width (Status) */
      } sys_control;
    } SYSCONTROL;

    Typedef union  {                              /* master status register (vector) */
      INTEGER Vector;
      struct    {
        BIT_FIELD                          : 8; /* 8-f: Dummy Bits */
        BIT_FIELD VecUser                  : 4; /* 4-7: User Programmable Bits */
        BIT_FIELD VecNum                   : 4; /* 0-3: Vector Number */
      } mstr_stat;
    } MASTERSTATUS;

    typedef union  {                              /* interrupt enable register */
      INTEGER IntEn;
      struct    {
        BIT_FIELD SCPC                     : 1; /* f: SCP Interrupt Enable */
        BIT_FIELD CD1                      : 1; /* e: CD1 Interrupt Enable */
        BIT_FIELD CD0                      : 1; /* d: CD0 Interrupt Enable */
        BIT_FIELD Rx1Idl                   : 1; /* c: Ch 1 Idle Interrupt Enable */
        BIT_FIELD Rx0Idl                   : 1; /* b: Ch 0 Idle Interrupt Enable */
        BIT_FIELD RxBOvr                   : 1; /* a: Receive Buffer Overrun Int Enable */
        BIT_FIELD TxFUnr                   : 1; /* 9: Transmit FIFO Underrun Int Enable */
        BIT_FIELD RxFOvr                   : 1; /* 8: Receive FIFO Overrun Int Enable */
        BIT_FIELD Tx1FC                    : 1; /* 7: Ch 1 Tx Frame Complete Int Enable */
        BIT_FIELD Tx0FC                    : 1; /* 6: Ch 0 Tx Frame Complete Int Enable */
        BIT_FIELD Tx1DMAC                  : 1; /* 5: Ch 1 Tx DMA Complete Int Enable */
        BIT_FIELD Tx0DMAC                  : 1; /* 4: Ch 0 Tx DMA Complete Int Enable */
        BIT_FIELD Rx1DMAC                  : 1; /* 3: Ch 1 Rx DMA Complete Int Enable */
        BIT_FIELD Rx0DMAC                  : 1; /* 2: Ch 0 Rx DMA Complete Int Enable */
        BIT_FIELD AddrErr                  : 1; /* 1: Address Error Interrupt Enable */
        BIT_FIELD BusErr                   : 1; /* 0: Bus Error Interrupt Enable */
      } int_enable;
    } IRQE;
```

Figure 7-7. DDLC typedef (C Language)

```
    typedef union  {                          /* scp status/control register */
      INTEGER SCPReg;
      struct   {
        BIT_FIELD                          : 1; /* f: SCP Complete */
        BIT_FIELD                          : 1; /* e: Slave/Master Select */
        BIT_FIELD CI                       : 1; /* d: SCP Clock Invert */
        BIT_FIELD BR                       : 1; /* b-c: SCP Baud Rate Select */
        BIT_FIELD En1                      : 1; /* a: SCP Enable 1 */
        BIT_FIELD En0                      : 1; /* 9: SCP Enable 0 */
        BIT_FIELD TxEn                     : 1; /* 8: SCP Transmit Enable */
        BIT_FIELD TxRx                     : 1; /* 0-7: SCP Tx/Rx Register */
      } scpreg;
    } SCP;

    typedef union  {                          /* timer register */
      INTEGER timer;
      struct   {
        BIT_FIELD                          : 5; /* b-f: Dummy Bits */
        BIT_FIELD TIRQ                     : 1; /* a: Timer Interrupt Status Bit */
        BIT_FIELD WDE                      : 1; /* 9: Watchdog Enable (chan 0 only) */
        BIT_FIELD TEN                      : 1; /* 8: Timer Enable */
        BIT_FIELD count                    : 8; /* 7-0: 8-bit Timer Register */
      } Timer;
    } TIMER;

/ *
  *      Structure of the channel's status/control and buffer descriptor register
  * /
    typedef struct  {                         /* channel status/control registers */
     union  {
      INTEGER SerCont;
      struct   {
        BIT_FIELD                          : 1; /* f: Dummy Bits */
        BIT_FIELD SCPE                     : 1; /* e: SCP Enable */
        BIT_FIELD TOL                      : 1; /* d: Transmit on Loop */
        BIT_FIELD LOOP                     : 1; /* c: Loopback */
        BIT_FIELD TRSP                     : 1; /* b: Transparent Operation */
        BIT_FIELD MODE                     : 1; /* 8-a: Mode Bits */
        BIT_FIELD MASK                     : 1; /* 0-7: Bit Mask Register */
      } serial_control;
    } SerControl;
     union  {                                 /* transmit control register */
      INTEGER TxCntr;
      struct   {
        BIT_FIELD                          : 9; /* 7-f: Dummy Bits */
        BIT_FIELD DI                       : 1; /* 6: Data Invert */
        BIT_FIELD FA                       : 1; /* 5: Force Abort */
        BIT_FIELD MD                       : 1; /* 4: Mask Drive */
        BIT_FIELD FCE                      : 1; /* 3: Force CRC Error */
        BIT_FIELD ITF                      : 1; /* 2: Inter-frame Time Fill */
        BIT_FIELD BR                       : 1; /* 1: Buffer Ready */
        BIT_FIELD TE                       : 1; /* 0: Transmit Enable */
      } tx_control;
    } TxCont;
```

Figure 7-7. DDLC typedef (C Language) (Continued)

```
union  {                                    /* receive control register */
 INTEGER RxCntr;
 struct    {
   BIT_FIELD                          : 9; /* 7-f: Dummy Bits */
   BIT_FIELD DI                       : 1; /* 6: Data Invert */
   BIT_FIELD ABS                      : 1; /* 5: Address Byte Select */
   BIT_FIELD ACE                      : 1; /* 4: Address Compare Enable */
   BIT_FIELD RCE                      : 1; /* 3: Receive on CRC Error */
   BIT_FIELD BBR                      : 1; /* 2: Buffer B Ready */
   BIT_FIELD BAR                      : 1; /* 1: Buffer A Ready */
   BIT_FIELD RE                       : 1; /* 0: Receiver Enable */
 } rx_control;
} RxCont;
union  {                                    /* transmit status register */
 INTEGER TxStatus;
 struct    {
   BIT_FIELD                          : 10; /* 6-f: Dummy Bits */
   BIT_FIELD FUN                      : 1;  /* 5: FIFO Underrun */
   BIT_FIELD TFC                      : 1;  /* 4: Transmit Frame Complete */
   BIT_FIELD CTS                      : 1;  /* 3: CTS Status */
   BIT_FIELD BE                       : 1;  /* 2: Bus Error */
   BIT_FIELD AE                       : 1;  /* 1: Address Error */
   BIT_FIELD TDC                      : 1;  /* 0: Transmit DMA Complete */
 } tx_stat;
} TxStat;
union  {                                    /* receive status register */
 INTEGER RxStatus;
 struct    {
   BIT_FIELD CD                       : 1; /* f: CD pin status */
   BIT_FIELD RI                       : 1; /* e: Receiver Link Idle */
   BIT_FIELD CE                       : 1; /* d: CRC Error */
   BIT_FIELD CDIRQ                    : 1; /* c: CD Interrupt Status */
   BIT_FIELD FO                       : 1; /* b: FIFO Overrun */
   BIT_FIELD RC                       : 3; /* 8-a: Residue Count */
   BIT_FIELD                          : 2; /* 6-7: Dummy Bits */
   BIT_FIELD BE                       : 1; /* 5: Bus Error */
   BIT_FIELD AE                       : 1; /* 4: Address Error */
   BIT_FIELD BBO                      : 1; /* 3: Buffer B OVerrun */
   BIT_FIELD BA0                      : 1; /* 2: Buffer A OVerrun */
   BIT_FIELD RBC                      : 1; /* 1: Buffer B Complete */
   BIT_FIELD RAC                      : 1; /* 0: Buffer A Complete */
 } rx_status;
} RxStat;
 struct    {                                /* address compare registers */
   char AddComp1:                           /* 8-f: Compare Address 1 */
   char AddComp0:                           /* 0-7: Compare Address 0 */
} AddCmp;
 union  {                                   /* wildcard register */
 INTEGER Wild;
 struct    {
   BIT_FIELD                          : 8; /* 8-f: Dummy Bits */
   BIT_FIELD Widle Card          : 8;       /* 0-7: Address Wildcard Bits */
 } widlbits;
} WildBits;
```

Figure 7-7. DDLC typedef (C Language) (Continued)

```
      union  {                                    /* CRC error count */
       INTEGER CRCErr;
       struct    {
         BIT_FIELD                          : 8; /* 8-f: Dummy Bits */
         BIT_FIELD CRCError                 : 8; /* 0-7: CRC Error Count */
       } crcerrors;
      } CRCErrors;
      INTEGER TxFrLen;                           /* Transmit Frame Length */
      INTEGER TxBase;                            /* Transmit Buffer Base Address */
      INTEGER TxByteCnt;                         /* Transmit Byte Count */
      INTEGER RxBufLen;                          /* Receive Buffer Length */
      INTEGER RxABase;                           /* Receive Buffer A Base Address */
      INTEGER RxAByteCnt;                        /* Receive Buffer A Byte Count */
      INTEGER RxBBase;                           /* Receive Buffer B Base Address */
      INTEGER RxBByteCnt;                        /* Receive Buffer B Byte Count */
}     CH_REGS;


/ *
  *       typedef of the complete DDLC register structure
  * /
typedef struct {
  SYSCONTROL System;                             /* system control register */
  MASTERSTATUS MStat                             /* master status register */
  IRQE Interrupt;                                /* interrupt enable register */
  INTEGER Select16;                              /* dummy register for 16-bit bus
  INTEGER unused_space0[4];                     /* unused space */
  SCP scp;                                       /* scp register */
  TIMER timeout [2];                             /* timer registers  */
  INTEGER unused_space1[5];                     /* unused space */
  CH_REGS Ch_Regs[2];                            /* channel status/control registers */
} DDLC

/ *
  *       definition of global DDLC pointer
  * /
DDLC *ddlc_ptr;
```

Figure 7-7. DDLC typedef (C Language) (Continued)

```
#define AddrExp     0x00528000L
#define AddrMask    0x00ff0000L

/ *
   * DDLCbufferAlloc allocates nonrelocatable memory that
   * resides completely on a single 64k byte page.
   * The requested size must be <= 64K bytes.
   *
   * A global constat <(void) *AddrExp> should contain the
   * address of the Address Expansion Register. Typically, this
   * value will be 528000h or 52C000h.
   *
   * Calling Syntax:  void *DDLCbufferAlloc (unsigned long BufferSize);
   * Functions called internally:  mlalloc( ), relalloc( )
   * Return value:  if success:  pointer to allocated buffer
   *                if failure:  NULL
   */
void *DDLCbufferAlloc (BufferSize)
unsigned long BufferSize;
{
   int *AddrExpReg;                    /* pointer to Address Exp Register */
   void *bit_buffer;                   /* pointer to original requested buffer */
   unsigned long int extra_page;       /* temporary data */
   unsigned long int first_page;       /* temporary data */

   AddrExpReg = (volid *) AddrExp;
   if (BufferSize > (long) 0x00010000) return (void *) NULL);

   /* allocate twice as much as requested */
   if((big_buffer = (void *) mlalloc(BufferSize<<1)) == (void *) NULL)
      return(big_buffer);  /* allocation failure return NULL */
   extra_page = ((unsigned long) big_buffer + BufferSize) & AddrMask;
   first_page = (unsigned long) big_buffer & AddrMask;
   if (extra_page > first_page) {
      long int length;

      /* requested buffer overlaps 64k page boundary; use the higher page */
      *AddrExpReg = (int) (extra_page >> 16);  /* load Address Expansion Register */
      length = extra_page - (unsigned long) big_buffer + BufferSize;

      /* deallocate unused space above requested buffer */
      if (relalloc(big_buffer, length) == (void *) NULL)
        return((void *) NULL);  /* reallocation failure; return NULL */
      return((void *) extra_page);
   }
   else {
      /* requested buffer fits on a single 64k page */
      *AddrExpReg = (int) (first_page >> 16);  /* load Address Expansion Register */
      if(relalloc(big_buffer, BufferSize) == (void *) NULL)
        return ((void *) NULL);  /* reallocation failure; return NULL */
      return(big_buffer);
   }
}
```

Figure 7-8. DDLC bufferAlloc( ) Procedure (C Language)

Since the OWN pins are driven high or low during a DMA cycle they cannot be simply connected together to control one '374; diodes are used to "wire-OR" them together. One of the OWN pins is active when the DDLC is the bus master and is executing a DMA cycle. At all other times, the address expansion register is held off of the address bus in a high-impedance state.

Both channels share the same 64K byte buffer area in this design example. If the two channels need separate pages, two address expansion registers can be provided, each controlled by its OWN pin. See Figure 7-9. *DDLCbufferAlloc()* should be appropriately modified so that it can load page addresses into two separate '374 registers. This scheme can be expanded so that each receiver and transmitter has its own buffer page by using the R/$\overline{W}$ pin to further decode the OWN pins.

The Macintosh control bus has a few special signals which are not used directly by the 68000. EXT.DTK disables the Macintosh's internal DTACK generator allowing an external device to generate its own DTACK. The chip select line to the DDLC is connected to EXT.DTK so the DDLC's internal DTACK generator may be used (DC&D 13-10). Timing of the Macintosh bus is slightly different from the 68000. In particular $\overline{AS}$ of an external bus master (the DDLC) must not change around the rising edge of the Macintosh's S3 state (DC&D 13-14). Since the DDLC is not synchronized to the Macintosh bus and its $\overline{AS}$ signal changes on a rising edge, the MCLK to the DDLC is derived from the C8M signal and inverted.

7.2.1.3 INTERRUPTS. The DDLC provides several interrupt vectors to the host processor. The Macintosh cannot however use these vectors directly. Instead, the autovector capability of the 6800 is used (DC&D 13-16). A gate array in the Macintosh generates a VPA signal to the 68000 whenever an interrupt vector fetch is made which initiates autovector operation. The processor fetches an address located in the autovector area and jumps to that location. There are even autovector locations in the system. All 27 interrupts generated by the DDLC are grouped into the level 5 autovector (vector location 0074h). Similarly, interrupts generated by the MC145474 S/T transceiver are assigned to the level 3 auto-vector (vector location 006ch). The Macintosh system uses level 1, level 2, and level 4 interrupts for its own sues. Level 1 the lowest priority.

When a level 5 interrupt is detected, the interrupt handler in the terminal card's software package looks at the Master Status register in the DDLC and jumps to the appropriate routine based on the value in the register. When a level 3 interrupt is generated, a special routine accesses the S/T transceiver through the SCP in the DDLC. Interrupt handlers will be discussed in greater detail in Section 7.2.2.2.5.

7.2.1.4 PAL DEVICE. A 16L8 PAL is used to generate chip select signals to the DDLC and '374 address expansion register. The base address of the DDLC is 520000h and the address of the address expansion register is 528001h. 520000h was chosen as the base address because it is in an area that does not have RAM and is not used for Apple factory testing (DC&D 13-16). The PAL also generates an inverted MCLK for the DDLC. When the card is installed in the SEcond Wave ExpanSE chassis (available from Second Wave, Austin, Texas, (512) 343-9661) the inverted clock is not used and MCLK is connected directly to the C8M signal of the chassis.

The PAL is programmed to provide a chip select for a second address expansion register even though it is not used in this example. The design equations for the PAL are:

/520000H = /A23•A22•/A21•A20•/A19•/A18•A17•/A15•/A14•/AS
/528001h = /A23•A22•/A21•A20•/A19•/A18•A17•/A15•/A14•/AS•/LDS
/52C001h = /A23•A22•/A21•A20•/A19•/A18•A17•/A15•/A14•/AS•/LDS
/MCLK = /C8M

7.2.1.5 SERIAL BUSES. Two serial buses connect the S/T transceiver, DDLC, and codec/filter. The IDL bus transports data while the SCP bus transports control information. The following paragraphs describe the operation of the two buses.

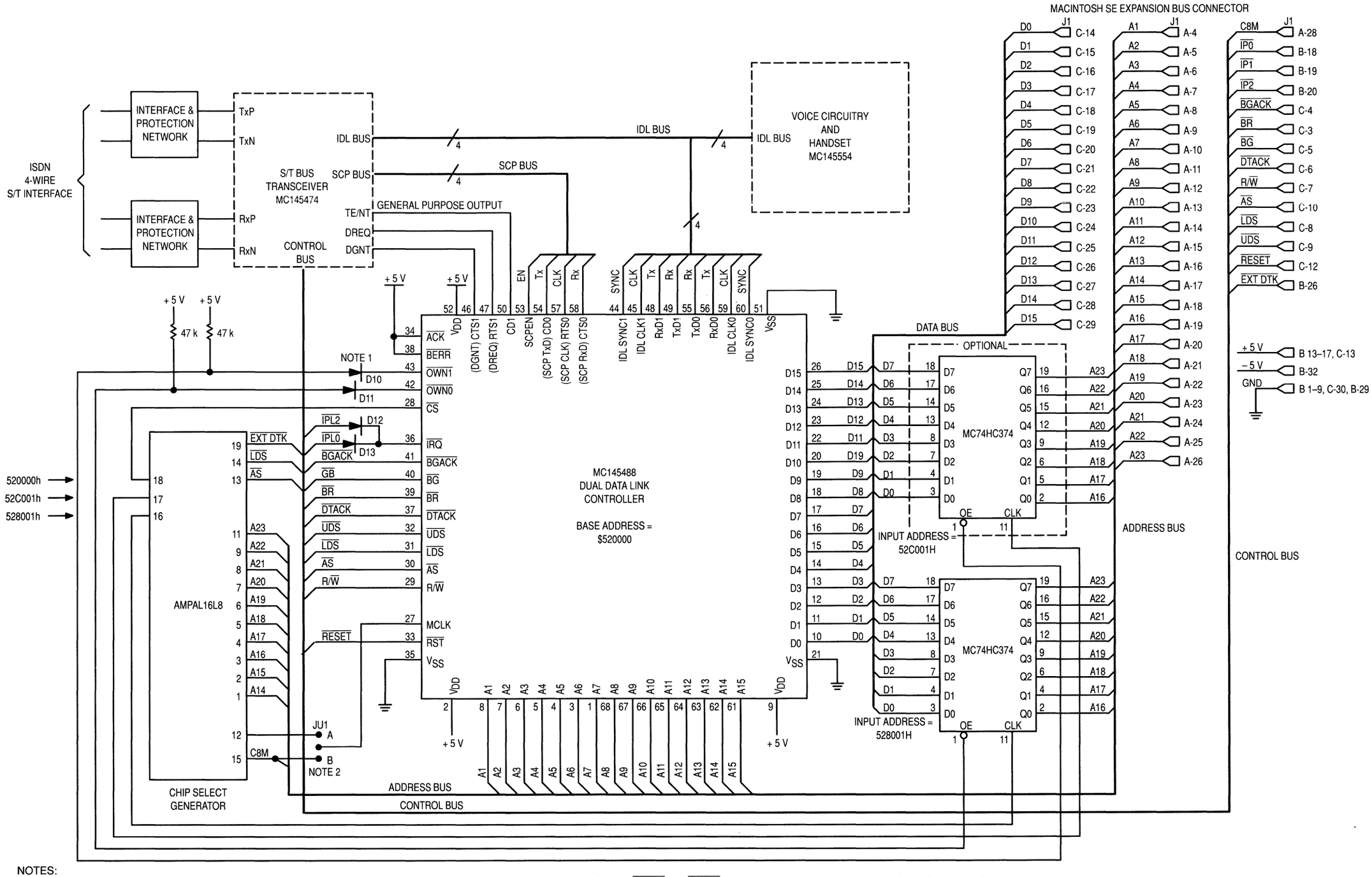
7.2.1.5.1 IDL Bus. The IDL bus is a 4-wire serial bus which transports serial data between the DDLC, S/T transceiver, and codec/filter. This bus is described in detail in Section 6.1. Since the S/T transceiver is the IDL master, its Tx pin is connected to the Rx pins of the DDLC and codec/filter. Likewise, the S/T transceiver's Rx pin is connected to the other devices' Tx pins. The SYNC pins of the devices and CLK pins are connected to their respective mates. Recall that the DDLC has two independent IDL interfaces. Since they are on the same bus, they are simply connected together. Of course, the two channels must not be programmed to operate on the same IDL timeslots at the same time.

The codec/filter operates on the IDL B1-channel, so the DDLC is programmed to use the other B-channel and the D-channel. The S/T transceiver has the capability to switch B-channels internally, so the DDLC and codec/filter can access either B-channel on the ISDN S/T bus.

7.2.1.5.2 SCP Bus. The Macintosh's 68000 communicates with the MC145474 S/T transceiver's internal registers through the DDLC's SCP port. Operation of the SCP from the DDLC's point of view is described in detail in Section 6.4. Figures 7-10 and 7-11 describe two C routines which communicate to the S/T transceiver. STReadWrite() is a routine which accepts the data direction (read or write), the S/T transceiver register address, and the data to be written. *scp_transfer()* actually performs the data exchange with the S/T transceiver. Transceiver nibble registers need a one-byte access; direction, address, and data (on a write) are all in one byte. Byte registers need two accesses where the first eight-bit exchange passes the direction and address to the transceiver and the second exchange passes the data. For more details on the operation of the S/T transceiver consult the MC145474 data sheet.

scp_transfer() does not use the interrupt capability of the SCP module. In most cases when the DDLC is the SCP master, polling the Tx bit may be more efficient. In other applications where the DDLC is an SCP slave, the SCPC interrupt is probably the most efficient way of detecting that an asynchronous SCP exchange occurred.

7.2.1.6 S/T TRANSCEIVER AND LINE INTERFACE BLOCKS. The MC145474 S/T transceiver connects directly into the system through the SCP and IDL buses. Figure 7-12 describes the connections. The $\overline{\text{IRQ}}$ signal is assigned to interrupt level 3 by "wire-ORing" $\overline{\text{IPL0}}$ and $\overline{\text{IPL1}}$ through diodes. DREQ and DGNT are connected to DDLC channel 1 DREQ and DGNT pins (RTS and CTS) and the SCP and IDL buses are connected. The ISET resistor is selected to be 33 k which sets the transmit current to be compatible with 1:1 transformers. A 15.36 MHz crystal is connected to generate the master clock for the S/T interface. This clock generates the timing for the IDL bus and is asynchronous to the Macintosh's master clock. The TE/$\overline{\text{NT}}$ pin is connected to +5 V selecting the TE mode of operation. Figure 7-13 describes the interface and protection blocks which protect the S/T transceiver from high-level transient signals that exist on the twisted pairs of the S/T bus. The terminating resistor on the receive pair may or may not be needed, depending on the installation.



NOTES:
1. If only one address expansion register is used, the anodes of the diodes can be connected together effectively "Wire-ORing" $\overline{OWN0}$ and $\overline{OWN1}$ together onto the output enable pin of the 374. If two 374s are used, the diodes may be removed and the OWN pins connected directly to the 374 Output enable pins. The pull-up pins must still be used as the OWN pins are high-impedance while not in DMA.
2. JU1 is a clock selector for the DDLC. The PAL is programmed to be an inverter from pin 15 to pin 12. When this board is installed directly into a Macintosh SE, the MCLK should be inverted from the C8M signal on the expansion connector. If a Second Wave ExpanSE chassis is used, MCLK should not be inverted.

Figure 7-9. Macintosh SE ISDN Terminal Card (MPU Section)

```
/ *
  *       STReadWrite( )
  *
  *       STReadWrite is a low level device driver for communication
  *       with the MC145474 S/T Transceiver
  *
  *       Calling Syntax:  int STReadWrite (int dir, int addr, int data);
  *           int dir:    data direction 1 = read, 0 = write
  *           int addr:   address of S/T transceiver regsiter to be accessed.
  *           int data:   data to be written. (Don't care on a read)
  *
  *       Functions called internally:  scp_transfer( );
  *       Return value:  data read from S/T Transceiver
  * /

int STReadWrite(int direction, int address, int data)
{
     char data_byte = 0;  /* temporary storage of transmitted data */
     if(direction) data_byte = (char) 0x80;        /* if read, set the direction bit */
     if(address) >= 0x0070) {                      /* byte register access to address */
       data_byte |= (char)(address & 0x007F);      /* 70h to 7Fh (mask unused bits) */
       scp_transfer(data_byte);                    /* first eight-bit exchange. . . */
                                                   /* . . .send the address */
       data_byte = (char) data;                    /* now prepare the data. . . */
     }                                             /* . . .for the second exchange */

     else {                                        /* nibble register access */
       data_byte |= (char) (address << 4);         /* address 0 to 6h */
       data_byte |= (char) (data & 0x0f);          /* set the transmitted data. . . */
     }
     return (scp_transfer(data_byte));             /* begin the eight-bit exchange. . . */
}
```

Figure 7-10. STReadWrite( ) Procedure (C Language)

```
/ *
  *       scp_transfer( )
  *
  *       scp_transfer prepares the DDLC for communication
  *       with the MC145474 S/T Transceiver and performs the transfer.
  *
  *       Calling Syntax:  int scp_transfer(int data);
  *           int data: address or data to be transferred to S/T Transceiver
  *
  *       This function expects a global pointer to the DDLC register structure:
  *           extern DDLC *ddlc_ptr;
  *
  *       Functions called internally:  none
  *       Return value:  data read from S/T Transceiver
  *           for write cycles this is unknown data
  * /


#define ON = 1;
#define OFF = 0;


int scp_transfer(char data)
{
     extern DDLC *ddlc_ptr;                        /* pointer to DDLC register structure */

     ddlc_ptr->scp.scpreg.TxRx = data;             /* load the data into the DDLC */
     ddlc_ptr->scp.SCPReg.En0 = ON;                /* set the En0 bit */
     ddlc_ptr->scp.SCPReg.TxEn = ON;               /* enable the SCP transmitter */
     while ( ddlc_ptr->scp.scpreg.TxEn);           /* wait for the exchange. . . */
                                                   /* . . .to complete */
     ddlc_ptr->scp.scpreg.En0 = OFF                /* turn off En0 bit */

     return (ddlc_ptr->scp.SCPReg);                /* return the data (for write cycles */
                                                   /* this is unknown data) */
```

Figure 7-11. scp transfer( ) Procedure (C Language)

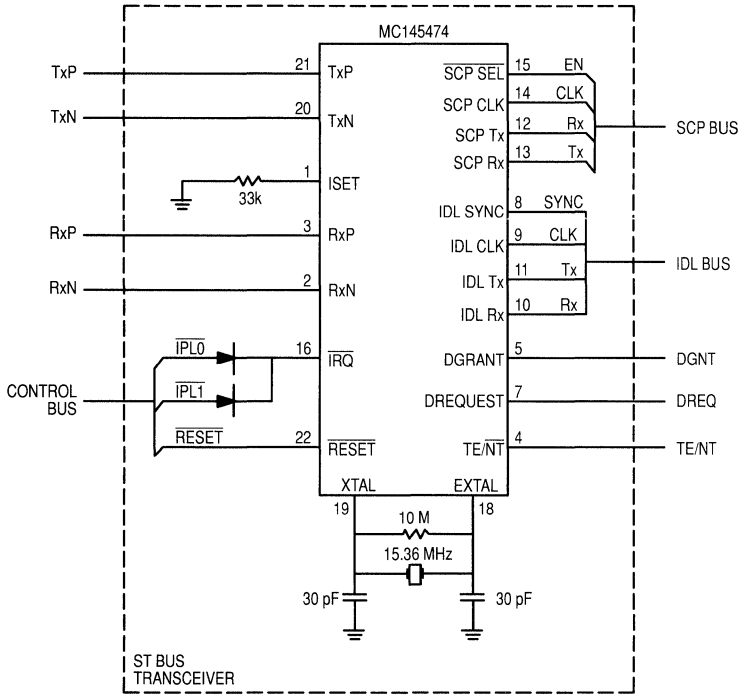


Figure 7-12. Macintosh SE ISDN Terminal Card (S/T Transceiver Section)

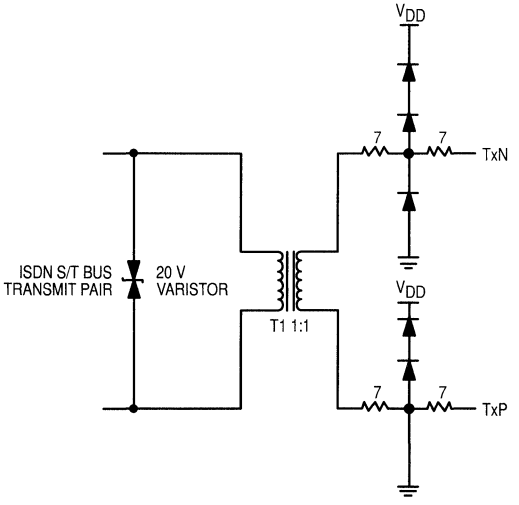


Figure 7-13. ISDN Transmit Pair Interface and Protection

7.2.1.7 CODEC/FILTER BLOCK. The MC145554 is a general-purpose codec/filter which converts analog voice signals into digital data that is transmitted over the S/T bus B-channels. It communicates with the S/T transceiver via the IDL bus on the B1-channel. Clock signals for the codec/filter's A/D and D/A are provided by the IDL CLK. The S/T transceiver is programmed to generate a 2.048 MHz IDL CLK.

Figure 7-14 shows the connection of the codec/filter to the IDL and the analog circuitry for the handset. R1 and R2 set the gain of the signal from the microphone. R5 sets the bias current through the microphone. R4 sets the attenuation for the received signal which is applied to the speaker. R3 sets the sidetone (feedback from mouth to ear). Resistor values shown are selected for a typical telephone handset with a carbon microphone. If an electret microphone is used R1, R2, and R5 will change values to compensate for the different bias and gain characteristics of that microphone.

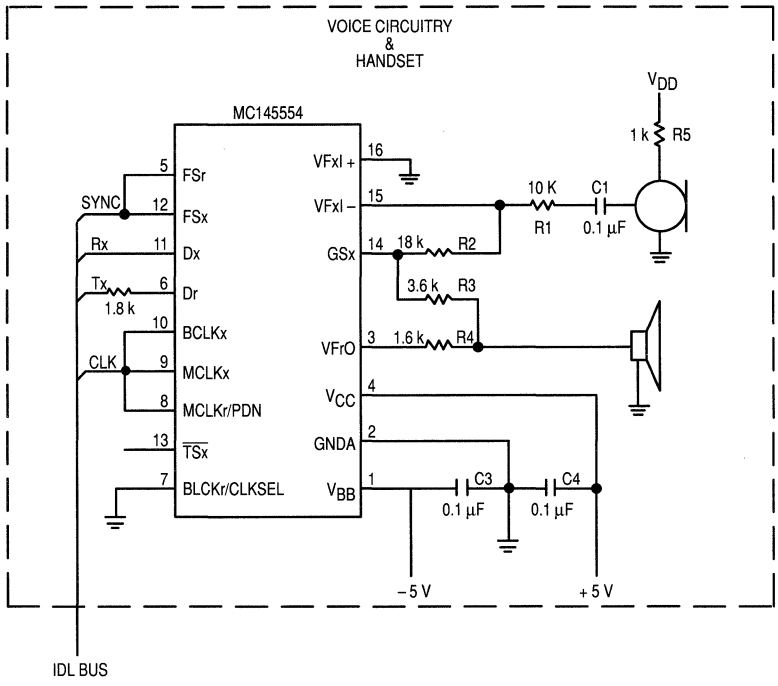


Figure 7-14. Macintosh SE ISDN Terminal Card (Codec/Filter Section)

Bypass capacitors C2 and C3 should be layed out close to the codec/filter chip. Placement of the ground paths around the codec/filter are critical to achieving the best possible analog performance.

7.2.2 Terminal Card Software

The following section describes the software package of the ISDN terminal card. Figure 7-15 is an overview of the software.

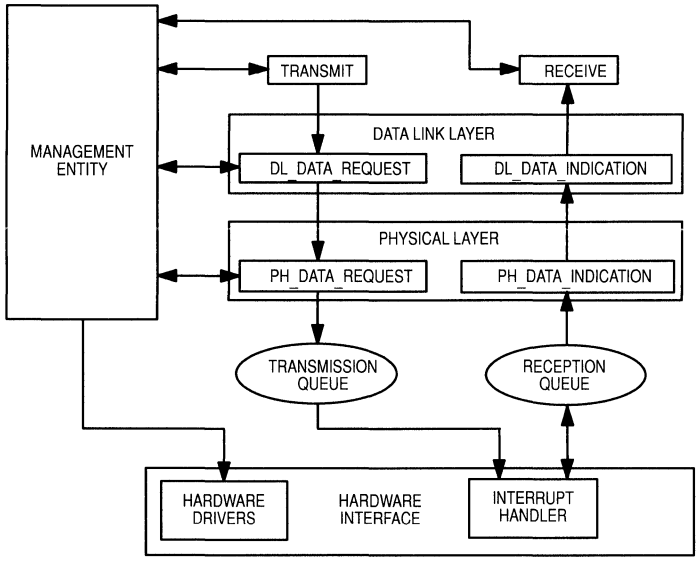


Figure 7-15. Terminal Card Software Block Diagram

- TRANSMIT and RECEIVE are the highest level routines which control the sequence of events required for proper transmission or reception of files.
- Data link layer, that implements a HDLC data link protocol, with a sliding window (numbered) data flow control and go-back-N automatic retransmission error recovery. Two primitives are included in this functional block: DL_DATA_REQUEST and DL_DATA_INDICATION. The first is called by the application whenever a packet containing data from a file needs to be transmitted. The procedure checks the state of the transmission window, and determines whether the packet may be accepted by the data link layer. The second procedure is called by the application to determine if a received data packet is available for processing. If it is the case, and before returning control to the application, the DL_DATA_INDICATION procedure sends an acknowledge message to the peer entity at the other end of the data link with an acknowledge number corresponding to the received frame.
- Physical layer provides a reliable data path between the upper data link layer and the hardware interface. Two primitives are implemented within this block: PH_DATA_REQUEST and PH_DATA_INDICATION. The first is executed whenever the upper data link layer requests that a packet be given to the DDLC for later transmission. The second is called by the same entity to determine whether or not a frame has been received that requires further processing.
- Transmission queue and a **reception queue**, which are used to exchange data between the application and the DDLC. When a packet of data is ready to be given to the DDLC (to be later transmitted as a frame), it is placed in the transmission queue, from where the appropriate handler will access it and give it to the DDLC to be transmitted. When a data frame is received, it is placed in the reception queue by the interrupt handler, so that is may be accessed by the application.

- Hardware interface is responsible for handling the hardware present on the card (mainly the DDLC itself). Within this block, the **interrupt handler** processes all of the interrupts originated from the DDLC as part of its normal operation: transmit frame complete interrupt, frame received interrupt, etc. Additional **hardware drivers** are required to initialize the DDLC, enable/disable its interrupts, and perform other maintenance functions.
- Management entity, that provides for "additional" services requires to link all of the functional blocks of the application.

7.2.2.2 DESCRIPTION OF THE INDIVIDUAL BLOCKS. What follows is a discussion of the issues concerning the implementation of the functional blocks of the software.

7.2.2.2.1 Transmit Procedures. The transmit functional block of the DDLC demo software is actually composed of three main procedures: transmission, transmit, and transmit_packet. The sequence of actions implemented by these procedures is as follows:

Transmission: The transmission procedure is the first one to be executed (right after the **transmit** button is selected from the main window). It calls a procedure from the **management entity** to "install" the terminal card, and after that, displays the Macintosh standard file package so that the user may select which file is to be transmitted. The procedure then opens both data and resource forks of the file, and sends a numbered frame containing both the **file type** and **file creator** of the file to the data link layer. This "finder-related" information is required by the peer station to be able to create a file with the same characteristics as the one whose data is to be received. It must be emphasized that this first message is numbered: indeed, the transmission procedure will provide the data link layer with a data packet that will be processed exactly as if it were a file data packet. This way, the data link layer will retransmit the first "setup" frame until the proper acknowledge is received.

Having done this, the transmission procedure is ready to start sending the file. To do this, it executes successive calls to the **transmit** procedure, requesting transmission of the data fork and resource fork of the file, respectively.

Transmit: This procedure controls the transmission of a file fork (data or resource). It receives two input parameters:

- The DDLC channel through which the data is to be sent.
- The file reference number of the file fork to be transmitted.

The procedure first determines the length of the file. If it is longer than 32K bytes, a 32K byte memory block is requested from the operating system, and the transmission of the file will proceed in blocks of 32K byte each. Otherwise the application requests a memory block long enough to load the file fork into memory. The transmit procedure initializes a status window, which displays the following information: the name of the file being transmitted, whether its data fork or resource fork is being transmitted, and how much (in percentage) of the file has been transmitted. Then it interleaves calls to the **transmit_packet** procedure with calls to a **management entity** procedure to update the information contained in the status window (the user has the option of suspending the transmission of a file at any time).

Transmit_packet: This procedure receives three input parameters:

- The DDLC channel through which the data is to be sent
- A pointer to the buffer containing the data packet to be transmitted.
- The length (in bytes) of the packet to be transmitted.

The procedure transfers the data intended for transmission (which is in the memory buffer the file was read to), along with a packet header, to another memory buffer, and passes it to the data link layer for further processing.

The sequence of actions involved in the transmission of a file is schematically described in Figure 7-16.

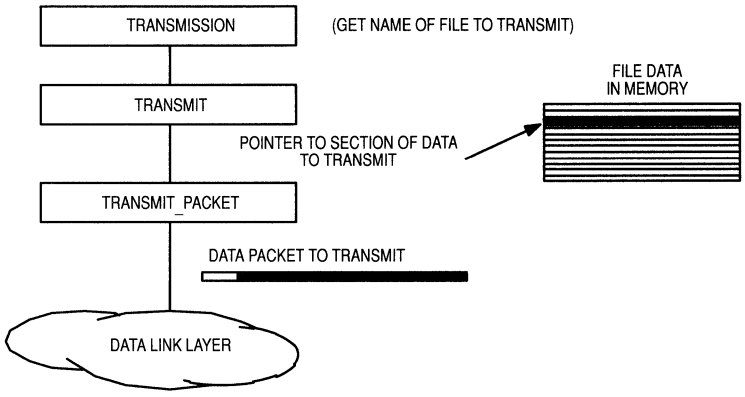


Figure 7-16. Transmit Procedure

7.2.2.2.2 Reception Procedure. The reception functional block of the DDLC demo software is actually composed of two main procedures: reception and receive. The sequence of actions implemented by these procedures is the following.

Reception: This procedure first displays the Macintosh standard file interface to request the user to input a name to be given to the file (whose data is to be received). After this, it waits for a "preliminary" frame to be received. Recall that the program will wait until either the proper frame is received or the user clicks the quit button (the latter will cause the program to return to the main window). This preliminary frame contains the file type and creator of the file that is to store the data to be received.

After the preliminary frame has been received, the reception procedure initializes a status window, similar to the one displayed during the transmission sequence, which displays the following information: the name of the file being received, whether its data fork or resource fork is being received, and host much (in percentage) of the file has been received. Then, it executes consecutive calls to the procedure receive, to process the reception of both the data and the resource forks of a file (it implicitly assumes that the data fork is sent first), and then returns to the main window.

Receive: This procedure is called to control the reception of a file fork (either data or resource). It receives two inputs parameters:

- The file reference number of the file fork to be received.
- The length of the file to be received (in bytes).

The procedure requests a 32K byte buffer of memory to store the incoming file, and successively checks the data link layer (by issuing calls to the procedure DL_DATA_INDICATION) to see if any data packets are available for processing. In doing so, it interleaves calls to a management entity procedure to update the information contained in the status window (the user has the option of suspending the reception of a file at any time) and updates the information contained in the status window.

The sequence of actions involved in the reception of a file is schematically described in Figure 7-17.

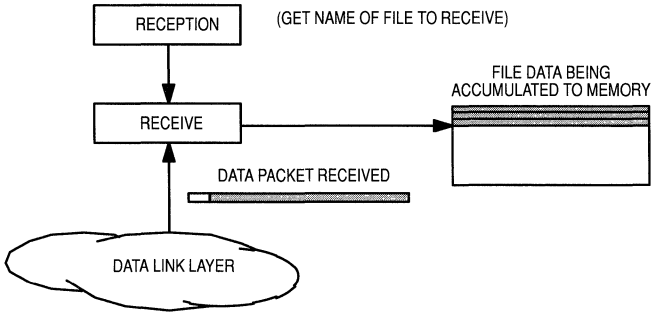


Figure 7-17. Reception Procedure

7.2.2.2.3 Data Link Layer. From a procedural point of view, this is the most important part of software. The data link layer provides a reliable interface between the application's dat transfer needs and the available physical resources. Furthermore, it provides with flow control and error recovery so that a (potentially faulty) physical link becomes an error-free physical link. It is made up of two main procedures.

DL_DATA_REQUEST: This procedure is called by the application whenever a data packet needs to be sent through the link. It receives three input parameters:

- The DDLC channel through which the data is to be sent.
- The pointer to the memory buffer that contains the data packet.
- The length (in bytes) of the data packet to be sent.

Before trying to process a data packet, the data link first checks to see if any pending acknowledges have been received (if so, it updates the transmission window) and the checks the state of the transmission window. If the window is exhausted (i.e., the maximum number of frames is pending to receive acknowledge), it returns control to the application, with a value that indicates that the message couldn't be processed. Otherwise, the data packet is first copied in an internal storage area of the data link, along with a time stamp indicating the moment of acceptance of the packet and then the packet is duplicated to a buffer in the memory area of the DDLC, with a data link header inserted into it. Finally, the data packet is passed to the physical layer, for actual transmission by the DDLC.

DATA_LINK_INDICATION: This procedure is called by the application to check for data frames received by the DDLC, requiring to be processed. It receives two input parameters:

- The DDLC channel whose reception status needs to be checked.
- A pointer to a data structure used to pass information regarding frames received to the application.

It accesses the physical layer to determine if there are any received frames in the reception queue. If so, the physical layer passes the first enqueued frame to the data link layer, which in turn passes it to the application.

As it may be deduced, whether on transmission or reception of a file, the application allocates the data link software a minimum processor time so data packets can be transmitted/received "as soon as possible". In absence of a real-time operating system, this is the best way to allocate a minimum of CPU time to the communication functions of the application.

7.2.2.2.4 Physical Layer. This functional entity is responsible for handling the data structures used as exchange between the application and the DDLC (i.e., the transmission and reception queues). As is the case of the data link layer, it is made up of two main procedures.

PH_DATA_REQUEST: This procedure is called by the data link layer to enqueue a data packet that is to be transmitted by the DDLC. Two input parameters are needed:

- The DDLC channel through which the data is to be sent.
- A pointer to the transmit descriptor of the data packet.

Since this procedure needs to access the transmission queue, it requires that the CPU interrupts be disabled, to provide for mutual exclusion with the interrupt handler. After disabling the interrupts, it checks the queue. If its empty, it means that no packet is in the process of being transmitted. It also means that the interrupt handler will not be able to service this particular packet (this last issue will be understood when the interrupt handler is explained in detail, later on), so its transmission must be triggered with an auxiliary driver. If, on the other hand, the transmission queue has at least one element, it means that a packet is in the process of being transmitted, and a transmit complete interrupt will be signalled later on that will cause the interrupt handler to service other packets in the queue. In this case, the procedure simply enqueues the packet descriptor in the transmission queue, and enables the interrupts backs.

PH_DATA_INDICATION: This procedure is called by the data link to check for data packets being received from the physical link. For the same reason as in the previous procedure, this must be executed with the interrupts disabled (it accesses the reception queue, which is also accessed by the interrupt handler). Two input parameters are passed:

- The DDLC channel that needs to be checked for reception.
- A pointer to a data structure used to pass information regarding frames received to the data link layer.

If the procedure finds that a data packet has been enqueued in the reception queue, it is removed and information about it is copied in a message that is passed to the data link layer: a pointer to the storage area that contains the data, and the length in bytes of the data; this last value excludes two bytes corresponding to the CRC (recall that, in receiving frames, the DDLC does transfer the CRC to memory), but includes the data link header of the frame. Finally, it removes the data packet

descriptor from the queue, enables the interrupts, and returns a value that indicates that a valid data packet is being passed to the data link layer. If no data packet is found in the reception queue, it reenables the interrupts, and notifies it to the data link layer.

7.2.2.2.5 Hardware Interface. As pointed out before, the hardware interface is made up of a set of hardware drivers, and the interrupt handler.

Interrupt Handler: The interrupt handler is the part of the code that processes the interrupts generated by the DDLC as part of its normal operation. Prior to start execution, the interrupt handler saves all the CPU registers in the stack, and then executes a utility call, SetUpA5(), that forces register A5 of the 68000 to be loaded with the application's A5 register. This is relevant because any compiler that generates code to execute in the Macintosh is expected to generate global addresses relative to register A5. Thus, by executing the utility call, it is ensured that the interrupt handler will be able to address the global variables of the application.

Due to the hardware design of the Macintosh, any device interrupting the CPU is forced to do it through a so-called "autovectored" interrupt. Because of this, the only way to differentiate between different interrupting sources within the DDLC is by reading its Master Status register. This is the first thing done by the interrupt handler after saving registers and setting register A5. The actions taken by the interrupt handler after this point depend on the actual interrupt source (they are identical for both DDLC channels).

Bit handler fault interrupts and **system fault** interrupts are dealt with by resetting the corresponding bit and resuming the DDLC operation. In some special debug routines these interrupts are serviced differently.

Time 0 interrupts cause a tick counter to be incremented. This counter is used as timeout and timestamp clock by the data link layer (timer1 is not used in the application).

Bit handler normal operation interrupts are handled according to whether they are transmission or reception interrupts:

- Transmission Interrupts: All related bits in the Transmit Status register are cleared, and the transmission queue is checked to see if any more data packets need to be transmitted. If so, the DMA Transmit Base Address register is loaded with the offset of the memory buffer containing the data packet, and the Buffer Ready bit in the Transmit Control register is set, so that the DDLC will start transmission again.

- Reception Interrupts: In this case, the Receive Status register is read to determine whether buffer A or B is filled with data. Once this is done, it requests the buffer descriptor corresponding to the data buffer in which the data has been stored. Then, the interrupt handler accesses the free (unused) receive buffer of the descriptor, loads it in the corresponding Receive Base Address register, and sets the Buffer A Ready or Buffer B Ready bit in the Receive Control register, so that another received frame may be loaded in that buffer.

7.2.2.2.6 Hardware Drivers. This is a set of routines that provide access to the hardware of the DDLC demo board. These routines:

- Reset the DDLC.

- Enable/disable interrupts from the DDLC.

- Enable the individual interrupt sources of the DDLC.
- Initialize the receive buffer registers of the DDLC.
- Control the S/T transceiver through the serial control port of the DDLC.

7.2.2.3 IMPLEMENTATION ISSUES. Two implementation issues of the DDLC demo software will be described in some detail. There are:

- The memory buffer management.
- The flow control and error recovery functions of the data link layer.

7.2.2.3.1 Memory Buffer Management. One of the most important actions in servicing the DDLC is to supply it with buffers to be used during transmission or reception of frames. The way the DDLC drivers interact with the rest of the application is more or less a classical "message passing" problem: packets that are to be sent must be enqueued in some output data structure so that the DDLC drivers will "see" them, whereas packets that have been received must be enqueued in an input data structure so that the application will "see" them. In this kind of message exchange problem, it is generally agreed that it is better to deal with buffer descriptors rather than the buffers themselves. A buffer descriptor usually contains a minimal of information required to completely specify a buffer in memory. Figure 7-18 depicts the information in a buffer descriptor.

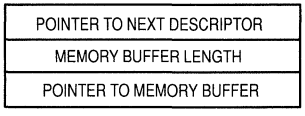


Figure 7-18. Buffer Descriptor

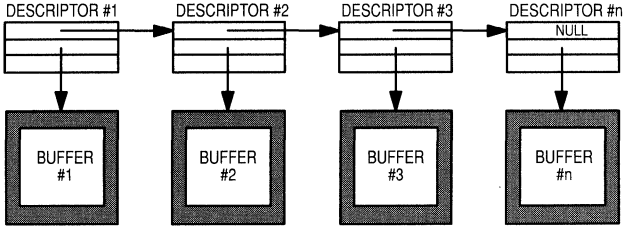


Figure 7-19. Linked List of Descriptors

In particular, descriptors are particularly well suited for queue management (Figure 7-19). Descriptor queues can be either single linked (an element contains only a reference to the next element in the queue) or double linked (an element contains references to both the previous and the next element in the queue). Since operations in the exchange structures of the software occur at either the beginning of the queue (when deleting the descriptor of a received frame) or the end (when appending the descriptor of a frame to be transmitted), it is only necessary to implement a single linked queue structure.

When the application is first called to run (either for transmission or reception of a file), it creates a small pool of descriptors, each one having assigned a memory buffer. Two obvious questions to be asked at this time are: why is it necessary to create pool of descriptors? And why is each descriptor assigned to a memory buffer? We will deal with these questions one at a time.

With the memory allocation facilities of an operating system, it is sometimes possible to avoid such things as pools, since free memory may be allocated and deallocated at run time. However, the Macintosh does not allow calls to the memory manager from an interrupt handler. But from the last section, it is clear that the interrupt handler needs to request a free memory buffer to give it to the

DDLC after a frame is received, otherwise the DDLC runs out of buffers to store incoming frames, and the receiver circuitry is disabled. The solution in this case, is to create a memory management entity that is part of the application itself, and supply it with enough elements to satisfy the worst case requirements of the application. This worst case number is affected by the window size of both the data link and the number of DDLC channels that may be active at any given time. So the application creates two small pools: one for transmit frame descriptors and one for receive frame descriptors. These pools are actually arrays of structures (the structures being the descriptors themselves), addressed in a circular fashion (Figure 7-20). After the last element of the pool is used, the first element of the pool becomes the next-to-be-used descriptor. By making sure that the size of the pool is greater than the maximum number of elements that may be withstanding at any given time, no active descriptor can be mistaken as a free descriptor.

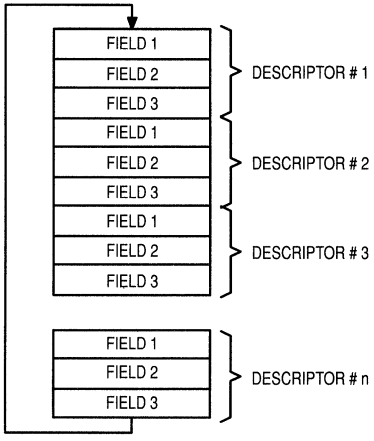


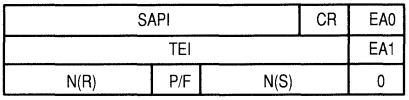
Figure 7-20. Circular Queue of Descriptors

With respect to the linking of a buffer to a descriptor, there is a savings in processing time, since the other solution would be to actually have two pools: one for buffer descriptors and one for the buffers themselves. But that would be worthless, especially when one considers that descriptors are consumed at the same rate as the buffers are. There is, though, an additional complexity introduced here, since when a frame is received, its corresponding descriptor must be accessed. With the descriptor data structure shown in Figure 7-20, the only way to do it is to use the buffer offset as an index to the array that implements the descriptor pool. For small pools, this solution is both simple and fast, but for a larger number of descriptors, additional features must be considered (i.e., a double link between a descriptor and a buffer; another alternative might be a hash table to store the pointers to the descriptors, and use the buffer offset as an input to some hashing function).

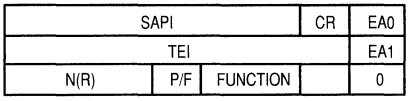
7.2.2.3.2 Flow Control and Error Recovery in the Data Link Layer. As suggested by the layered hierarchy of the ISO model for an open systems interconnection, the data link layer should provide for a reliable logical link between peer processes in the network. Two important issues involved here are flow control and error recovery. The first has to do with how can the data link entity controls the amount of data being sent by a peer process. The second refers to the means by which the same data link entity can react to non-permanent errors present in the link.

A window flow control scheme is used to limit the number of frames that a data link entity may send without waiting for acknowledgement from its peer process. It means that the data link entity may

send a consecutive number of frames equal to the window size (which is usually a multiple of 2, say 2N; in this software, the window size is 8) minus one. When the total number of frames waiting for acknowledge equals the maximum allowed, the window is said to be exhausted, and the sending side of the link is forced to wait for acknowledge frames. Thus, the receiving side controls the flow of frames from the wending side by withholding the acknowledge of frames received. Normally, nothing prohibits the receiving station from acknowledging various frames at a time. For instance, the receiving station could wait for its peer to send as many frames as possible, and then acknowledge them all with a single acknowledge frame. Nonetheless, in this software, frames are acknowledged as soon as they are received, so the sending station is constantly updating its transmission window. The information required to implement the flow control in the link is contained as a header that is appended to the frame prior to its transmission (Figure 7-21). Of all the fields available in the data link header, the only ones used by this software package are the N(S) in the information mode, and the N(R) and the function fields in the supervisory mode (other fields are set, but they are never checked, so they are not functionally active).



DATA LINK HEADER FOR INFORMATION FRAMES



DATA LINK HEADER FOR SUPERVISORY FRAMES

Figure 7-21. Data Link Headers

At any given time, an error may take place in a frame being sent that will cause it to be discarded by the receiving station. Obviously, this frame will not be acknowledged, and after a certain time, the sending station realizes that there was an error related to that frame. An error recovery procedure must be initiated so as to ensure that all frames intended for transmission are received correctly. The way it has been implemented here is through a go-back-N automatic retransmission error control procedure.

Whenever the data link entity receives a packet to be transmitted, it is duplicated in a local storage area, and a time stamp is associated with it. That way, the data link knows how much time all the transmitted frames have been in the data link. If this time ever exceeds a maximum timeout (this will always occur to the "oldest" packet in the dat link), all the packets stored are retransmitted, and their corresponding time stamps are updated.

7.2.2.4 SUMMARY. This section described in some detail the inner workings of the software associated with the ISDN terminal card for the Macintosh SE. It attempted only to provide direction to future ISDN programmers.

The development tool that this data sheets describes is compatible with the ISDN S/T interface from a hardware point of view. The software is in concept similar to what "real" ISDN software would be like. However, multiple logical links were not considered as they add quite a bit more complexity.

SECTION 8
ELECTRICAL SPECIFICATIONS

8.1 MAXIMUM RATINGS

Rating	Symbol	Min	Max	Unit
Supply Voltage	V_{DD}	− 0.3	+ 7.0	V
Input Voltage	V_{in}	− 0.3	+ 7.0	V
Operating Temperature	T_A	− 40	+ 85	°C
Storage Temperature	T_{stg}	− 55	+ 150	°C

8.2 THERMAL CHARACTERISTICS

Rating	Symbol	Value	Symbol	Value	Unit
Thermal Resistance (Still Air)	$R_{\theta JC}$	20*	$R_{\theta JA}$	65*	°C/W

*Estimated.

8.3 DC ELECTRICAL CHARACTERISTICS (T_A = − 40°C to 85°C, V_{DD} = 5.0 V ± 10%)

Rating	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	2.0	V_{DD}	V
Input Low Voltage	V_{IL}	− 0.3	0.8	V
Input Leakage Current @ 5.5 V	I_{in}	—	10	µA
High-Impedance Input Current @ 2.4/.04 V	I_{HZ}	—	10	µA
Output High Voltage (I_{OH} = − 400 µA)	V_{OH}	2.4	—	V
Output Low Votlage (I_{OL} = 5.0 mA)	V_{OL}	—	0.5	V
Power Dissipation	P_D	*	—	mW
Capacitance	C_{in}	—	20	pF

*See Section 8.4.

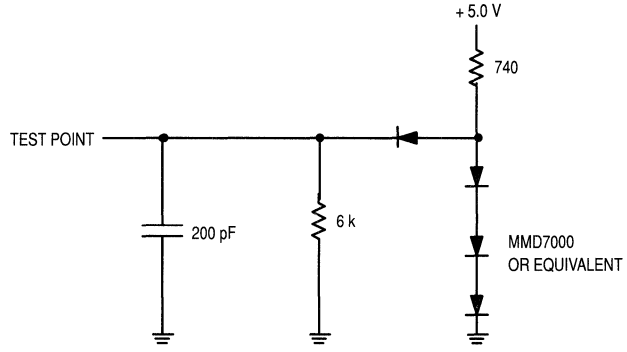


Figure 8-1. Test Load

8.4 POWER CONSIDERATIONS

Maximum Power Dissipation Calculations:

MPU Power Consumption $P_{MPU}(max)$ = 2.5 mW/MHz

Serial Power Consumption $P_{SERIAL}(max)$ = 2.0 mW/100 kbps

Examples:

MCLK = 8 MHz and aggregate serial data flow = 256 kbps

Total Power Dissipation ≤ 25.2 mW

MCLK = 16 MHz and aggregate serial data flow = 4.096 Mbps

Total Power dissipation ≤ 122 mW

8.5 MPU/DMA TIMING CHARACTERISTICS (68000 MODE)

(T_A = 40°C to 85°C, V_{DD} = 5.0 V ± 10%)

Reference Number	Characteristic	Min	Max	Unit	Notes
1	Asynchronous Set-Up Time	20	+	ns	1, 2
2	Address Valid Before $\overline{CS}$ Active	0	—	ns	
3	R/$\overline{W}$ (Write) Before $\overline{UDS}$, $\overline{LDS}$, Active	20	—	ns	
4	R/$\overline{W}$ (Read) Before $\overline{UDS}$, $\overline{LDS}$ Active	20	—	ns	
5	Data Strobe Pulse Width (Write Cycle)	30	—	ns	
6	Data Set-Up Time (Write)	35	—	ns	
7	$\overline{DTACK}$ Active After MCLK High	0	35	ns	3
8	$\overline{DTACK}$ Active After $\overline{CS}$ Active	1/2	1 + 35 ns	MCLK	
9	Data Valid (Read) After Data Strobe Active (Access Time)	—	80	ns	5

Reference Number	Characteristic	Min	Max	Unit	Notes
10	$\overline{DTACK}$ Active Before Data Valid (Read)	—	35	ns	
11	$\overline{CS}$ Active After Data Strobe Inactive (Hold Time)	0	—	ns	7
12	Address Valid After MCLK High (Hold Time)	20	—	ns	4
13	R/$\overline{W}$ Valid After Data Strobe Inactive (Hold Time)	0	—	ns	
14	Data Valid After Data Strobe Inactive (Write Hold Time)	10	—	ns	
15	Data Valid After Data Strobe Inactive (Read Hold Time)	15	—	ns	
15a	Data High-Impedance After Data Strobe Inactive	—	25	ns	
16	Chip Select Inactive to Chip Select Active	1	—	MCLK	
17	$\overline{DTACK}$ Active After Chip Select Inactive (Hold Time)	0	30	ns	6
18	$\overline{DTACK}$ High-Impedance After Chip Select Inactive	—	50	ns	
19	Signal Active Delay	0	45	ns	
20	Signal Inactive Delay	0	55	ns	
21	Signal High-Impedance Delay	0	60	ns	
22	$\overline{BGACK}$ Inactive After MCLK Low	0	30	ns	6
23	$\overline{BGACK}$ High-Impedance Delay	—	50	ns	
24	Address Valid After MCLK Low	0	55	ns	
25	Address Valid Before $\overline{AS}$ Active	0	—	ns	
26	Data Valid After MCLK High	0	55	ns	
27	Data Valid Before $\overline{UDS}$, $\overline{LDS}$ Active	0	—	ns	
28	Address Valid Afer $\overline{AS}$ Inactive (Hold Time)	20	—	ns	
29	Address High-Impedance Delay	0	25	ns	
30	Data (Write) After $\overline{UDS}$, $\overline{LDS}$ Inactive (Hold Time)	20	—	ns	
31	Data (Write) High-Impedance Delay	—	25	ns	
32	Data (Read) Valid Before MCLK Low (Setup Time)	20	—	ns	
33	Data (Read) Valid After $\overline{UDS}$, $\overline{LDS}$ Inactive (Hold Time)	10	—	ns	
34	$\overline{DTACK}$ Valid After MCLK High (Hold Time)	20	—	ns	
35	$\overline{BERR}$ Valid After MCLK Low (Hold Time)	20	—	ns	
37	$\overline{CS}$ and $\overline{IACK}$ Inactive After Address Error Sample Point	20	—	ns	8

NOTES:

1. MPU Mode — this is the time that a signal must be valid before a falling MCLK for the access sequence to begin on this MCLK cycle.
2. DMA Mode — $\overline{BG}$ must be active, $\overline{BGACK}$ and $\overline{AS}$ must be inactive for two successive MCLK edges before a DMA sequence begins on this MCLK cycle.
3. This is for synchronous applications.
4. The address is latched on the rising MCLK edge following the falling edge at which $\overline{CS}$ was detected low.
5. Both $\overline{CS}$ (and $\overline{LDS}$ or $\overline{UDS}$) must be active before data is output.
6. This signal may not reach V_{OH} before going high-impedance.
7. The bus cycle is terminated when $\overline{CS}$ goes high. The rising edge of $\overline{CS}$ may precede the rising edges of $\overline{UDS}$ or $\overline{LDS}$.
8. If either of these signals is active when sampled, an address error interrupt is generated.

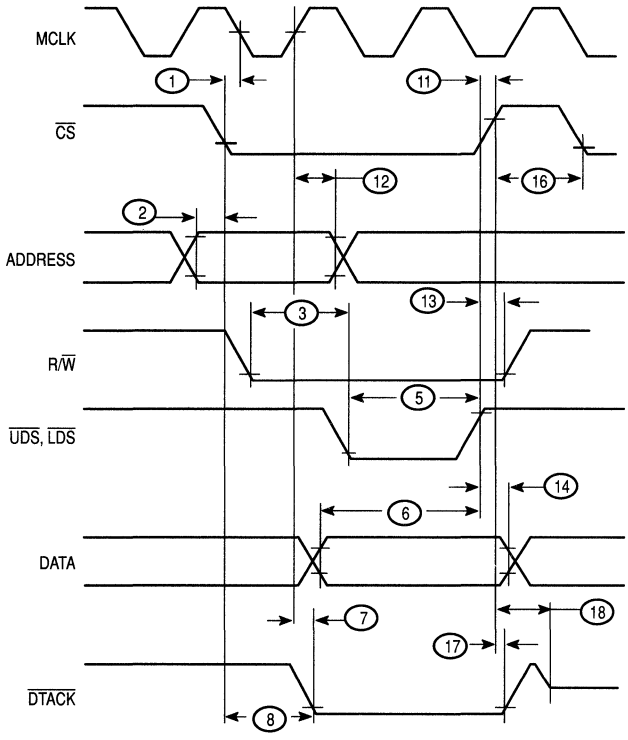


Figure 8-2. 68000 MPU Write Timing

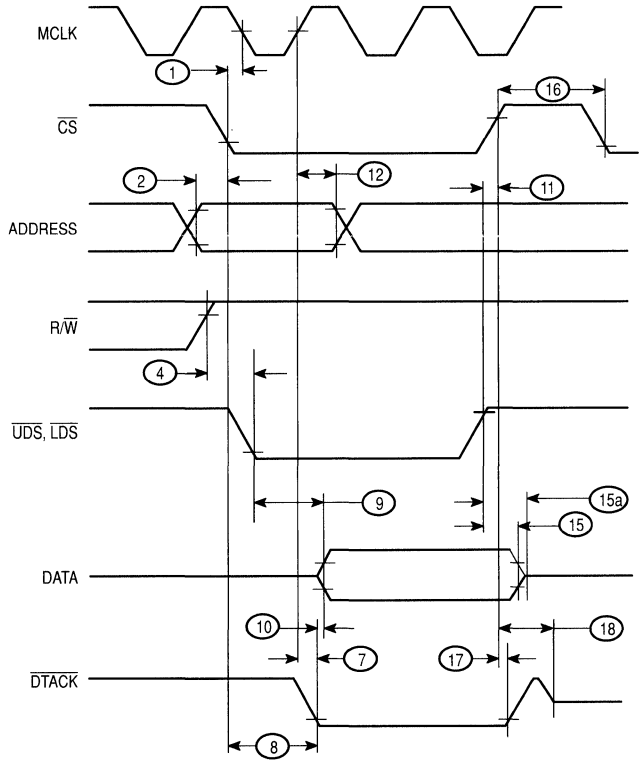


Figure 8-3. 68000 MPU Read Timing

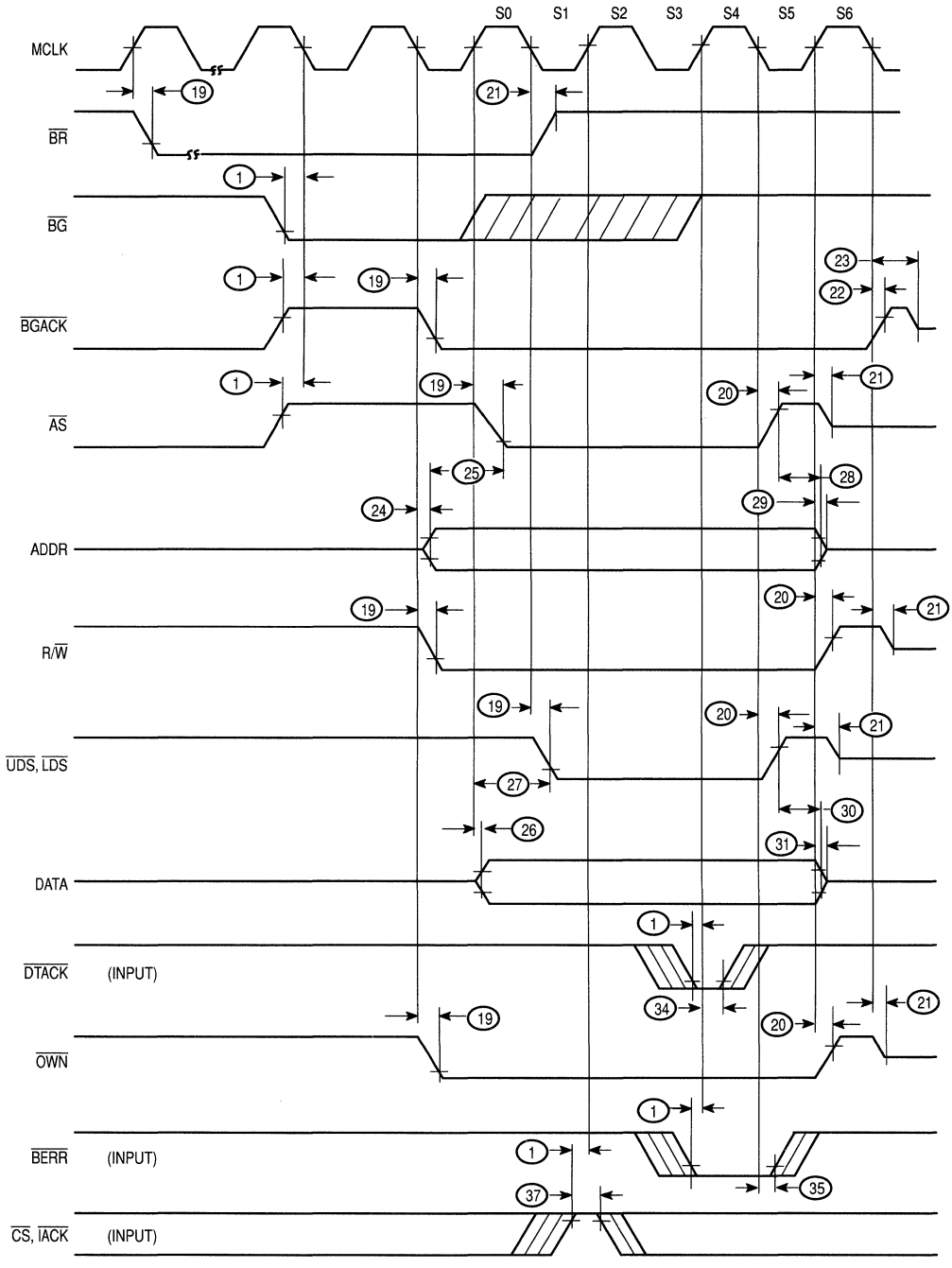


Figure 8-4. 68000 DMA Arbitration and Write Timing

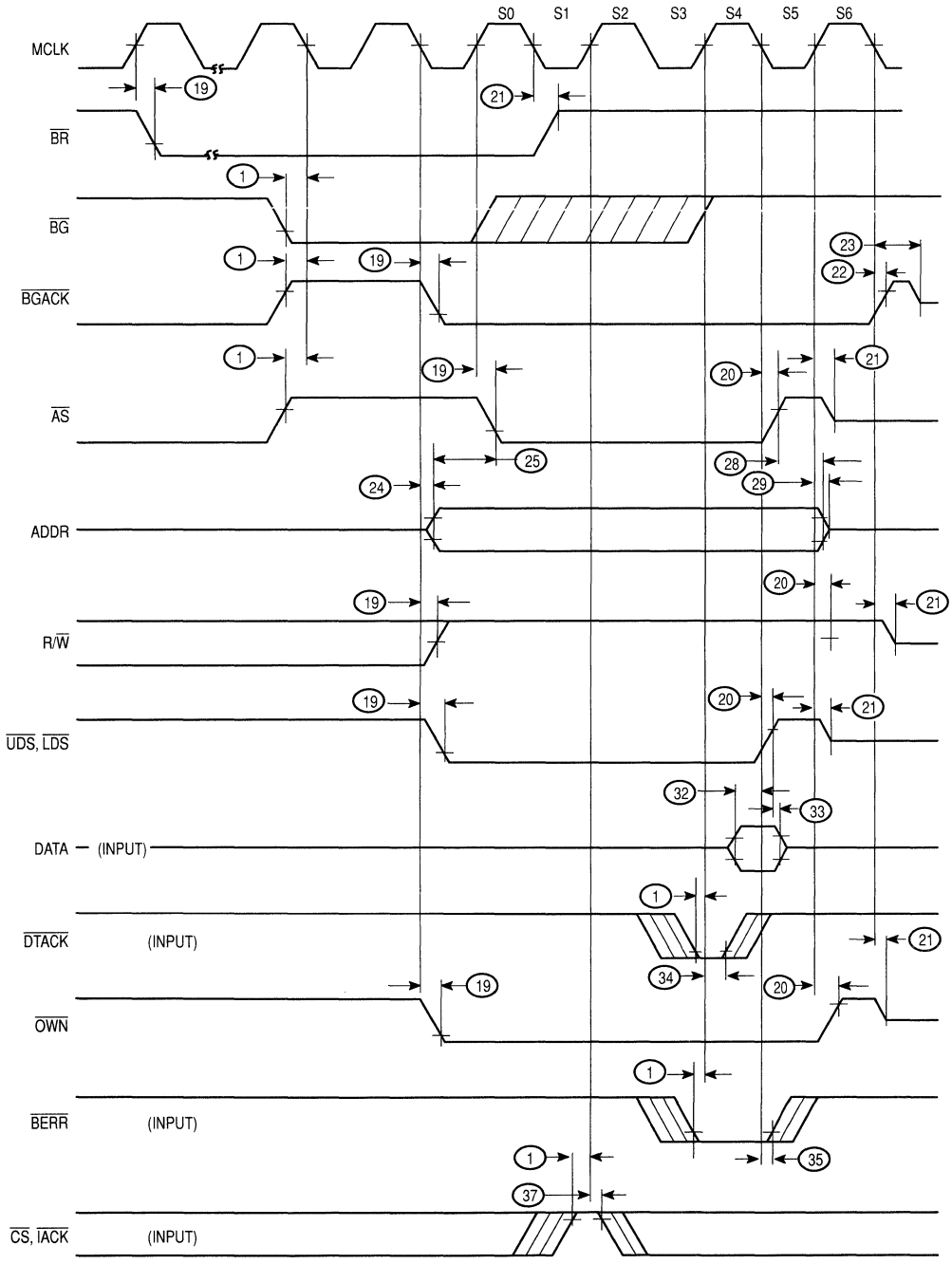


Figure 8-5. 68000 DMA Arbitration and Read Timing

8.6 MCLK TIMING CHARACTERISTICS (T_A = – 40°C to 85°C, V_{DD} = 5.0 V ± 10%)

Refernce Number	Characteristic	Min	Max	Unit	Notes
38	MCLK Rise Time	—	15	ns	
39	MCLK Fall Time	—	15	ns	
40	MCLK Width High	24	—	ns	9
41	MCLK Width Low	24	—	ns	9
42	MCLK Period	48	—	ns	10

NOTES:

9. A 50% duty cycle signal should be provided.
10. The DDLC is a static device. Any clock frequency from dc to 20.5 MHz may be provided.

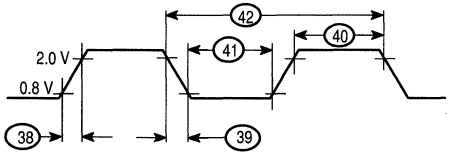


Figure 8-6. MCLK Timing Diagram

8.7 SERIAL INTERFACE (MODEM MODE) TIMING CHARACTERISTICS

Reference Number	Characteristic	Min	Max	Unit	Notes
43	Rx CLK, Tx CLK Rise Time	—	15	ns	
44	Rx CLK, Tx CLK Fall Time	—	15	ns	
45	Rx CLK, Tx CLK Width High	48	—	ns	11
46	Rx CLK, Tx CLK Width Low	48	—	ns	11
47	Rx CLK, Tx CLK Period	96	—	ns	12
48	Rx Data Set-Up Time	20	—	ns	
49	Rx Data Hold Time	20	—	ns	
50	BSYNC Low Set-Up Time	20	—	ns	
51	BSYNC High Set-Up Time	20	—	ns	
52	Tx Data High Delay Time	30	—	ns	
53	Tx Data Low Delay Time	30	—	ns	

NOTES:
11. The Tx CLK and Rx CLK signals should be nominally 50% duty cycle. The width of either phase must be greater than one MCLK period.
12. Tx CLK and Rx CLK may be any frequency up to MCLK/2.

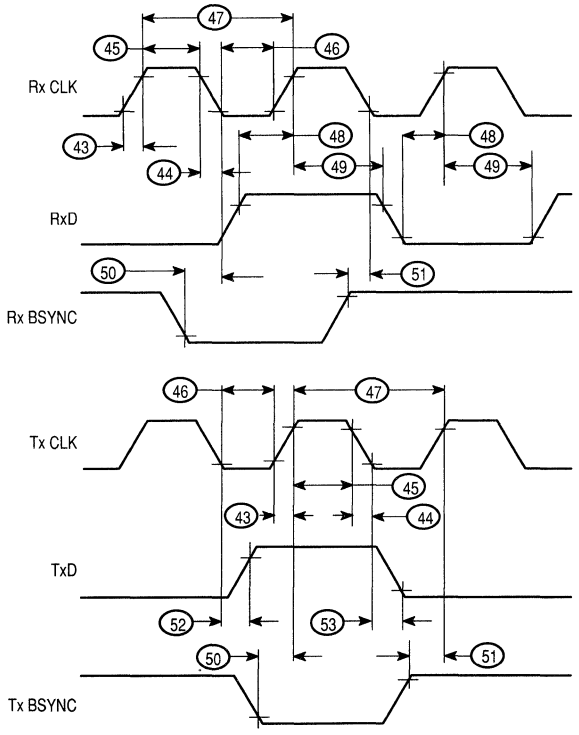


Figure 8-7. Serial Interface (Modem Mode) Timing Diagram

8.8 SERIAL CONTROL PORT TIMING CHARACTERISTICS

($T_A = -40°C$ to $85°C$, $V_{DD} = 5.0\ V \pm 10\%$)

Refernce Number	Characteristic	Min	Max	Unit	Notes
54	Delay from Internal SCP Tx EN to First SCP CLK Edge	1-1/2	2	SCP CLKs	13, 14
55	SCP CLK Period (Master Mode)	MCLK/2	MCLK/16	—	
56	Delay from End of MPU Bus Cycle to $\overline{\text{SCP EN}}$ (1 or 2) Edge	—	75	ns	15
57	Delay from SCP CLK to SCP Tx	—	50	ns	14
58	SCP Rx Set-Up Time	35	—	ns	14
59	SCP Rx Hold Time	20	—	ns	14
60	Time from Last SCP Clock Edge to Next Tx EN	1-1/2	—	SCP CLKs	16
61	Slave Select Active Before Rising SCP CLK Edge	50	—	ns	
62	SCP CLK High Time (Slave Mode)	60	—	ns	17
63	SCP CLK Low Time (Slave Mode)	60	—	ns	17
63a	SCP CLK Period (Slave Mode)	120	—	ns	
63b	Slave Select Inactive Before SCP CLK Rising Edge	50	—	ns	

NOTES:

13. Internal SCP Tx EN is the end of the bus cycle when SCP Tx is set high.
14. This also applies when the SCP CLK is inverted with the CI bit.
15. Delay is measured from the end of the MPU bus cycle when $\overline{\text{SCP EN}}$ 0 or 1 is set or cleared.
16. If the SCP module is re-enabled within this time, erratic behavior will result.
17. The SCP CLK in slave mode should be nominally 50% duty cycle.

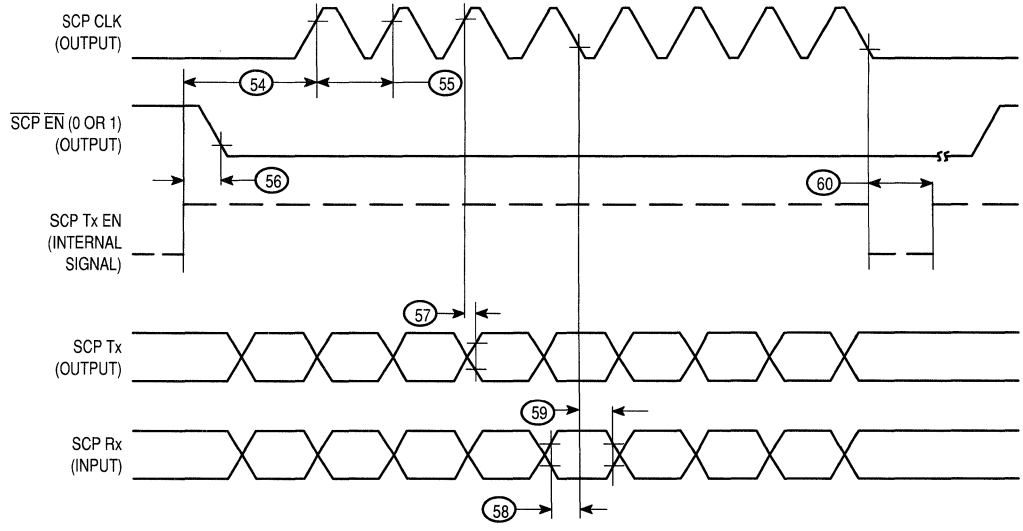


Figure 8-8. SCP Master Mode

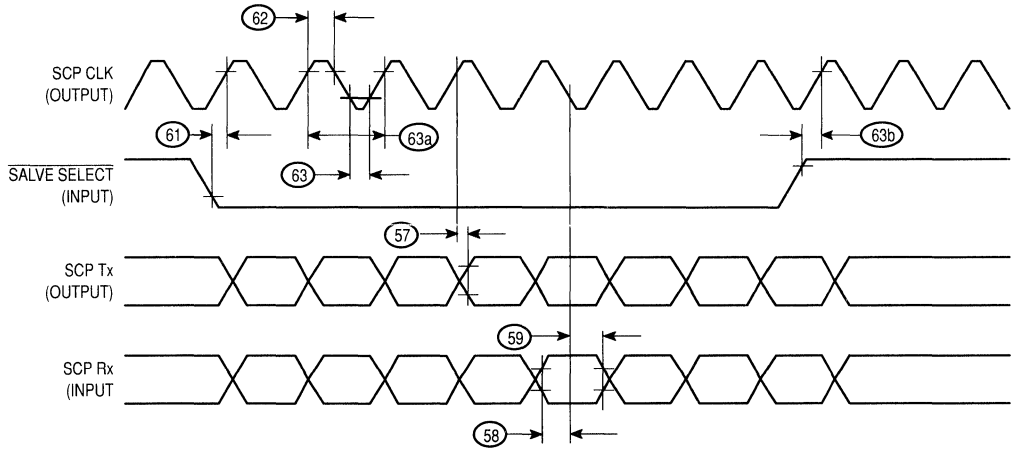


Figure 8-9. SCP Slave Mode

8.9 IDL TIMING CHARACTERISTICS (T_A = – 40°C to 85°C, V_{DD} = 5.0 V ± 10%)

Reference Number	Characteristic	Min	Max	Unit	Notes
64	IDL SYNC Active Before IDL CLK Falling Edge (Set-Up Time)	25	—	ns	
65	IDL SYNC Active After IDL CLK Falling Edge (Hold Time)	25	—	ns	
66	IDL SYNC Inactive Before 4th IDL CLK	25	—	ns	
67	IDL CLK Period	MCLK/2	—	MCLK	
68	IDL CLK Width High	48	—	ns	18
69	IDL CLK Width Low	48	—	ns	18
70	IDL Tx Active Delay	30	—	ns	
71	IDL Tx Time to High Impedance	—	25	ns	
72	IDL Rx Valid Before IDL CLK Falling Edge (Set-Up Time)	25	—	ns	
73	IDL Rx Valid After IDL CLK Falling Edge (Hold Time)	25	—	ns	
74	Time Between Successive IDL SYNCs	24	—	IDL CLK	

NOTES:
18. Each phase must be greater than 1 MCLK period. IDL CLK should be nominally 50% duty cycle.

8.10 SERIAL INTERFACE TIMING CHARACTERISTICS: LONG- AND SHORT-FRAME MODES (T_A = – 40°C to 85°C, V_{DD} = 5.0 V ± 10%)

Reference Number	Characteristic	Min	Max	Unit	Notes
75	Tx or Rx Enable Active After Tx or Rx Clock Falling Edge (Long-Frame Operation)	—	25	ns	
76	Tx or Rx Enable Active Before Tx or Rx Clock Falling Edge (Long-Frame Operation)	25	—	ns	
77	Tx or Rx Enable Active After 4th Tx or Rx Clock Rising Edge (Long-Frame Operation	25	—	ns	
78	Time Between Successive Tx or Rx Enables	12	—	CLK	
79	Tx Data Valid After Tx Enable Active	30	—	ns	19
80	Tx Data Valid After Tx Clock Rising Edge	30	—	ns	
81	Tx Data High Impedance After 9th Tx Clock Rising Edge	—	25	ns	20
82	Tx Data High Impedance After Tx Clock Rising Edge (Subrate Multiplexing Operation)	—	25	ns	
83	Tx Data Valid After Tx Clock Rising Edge (Subrate Multiplexing Operation)	30	—	ns	
84	Tx Data High Impedance After Tx Enable Low	—	25	ns	21
85	Tx or Rx Clock High Time	48	—	ns	22
86	Tx or Rx CLock Low Time	48	—	ns	22
87	Tx or Rx Clock Period	96	—	ns	23
88	Rx Data Valid Before Rx Clock Falling Edge (Set-Up Time)	25	—	ns	
89	Rx Data Valid After Rx Clock Falling Edge (Hold Time)	25	—	ns	
90	Tx or Rx Enable Active Before Tx or Rx Clock Falling Edge (Short-Frame Operation) (Set-Up Time)	25	—	ns	
91	Tx or Rx Enable Active After Tx or Rx Clock Falling Edge (Short-Frame Operation) (Hold Time)	25	—	ns	
92	Tx or Rx Enable Inactive Before 4th Tx or Rx Clock Rising Edge (Short-Frame Operation)	25	—	ns	

NOTES:
19. Tx data becomes valid after Tx clock rising edge or Tx enable high, whichever is later.
20. Data bits continue to be output after the 8th Tx clock as long as Tx enable is high.
21. This applies when Tx enable returns low before the 8th bit time.
22. Each phase must be greater than 1 MCLK period. A 50% duty cycle should be provided.
23. The maximum serial interface clock frequency can be no more than MCLK/2.

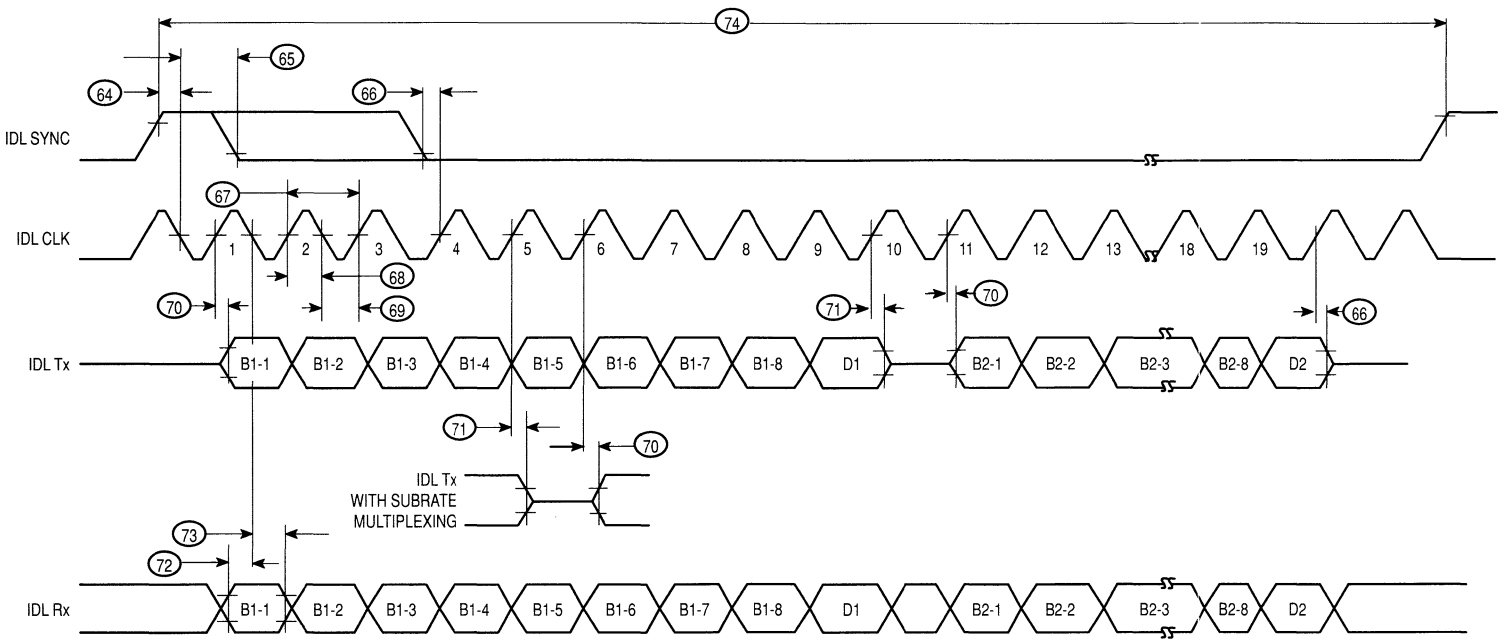


Figure 8-10. IDL Timing

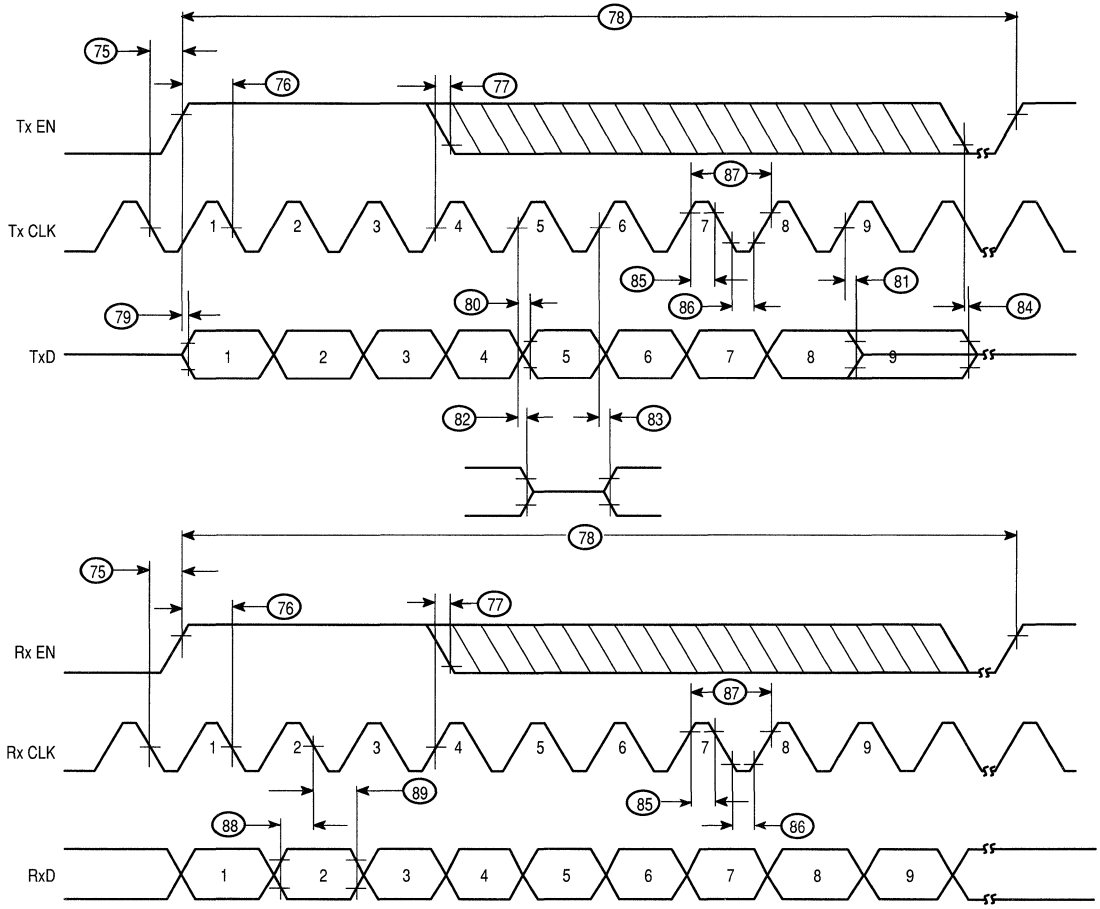


Figure 8-11. Timeslot Mode Timing Long-Frame Operation

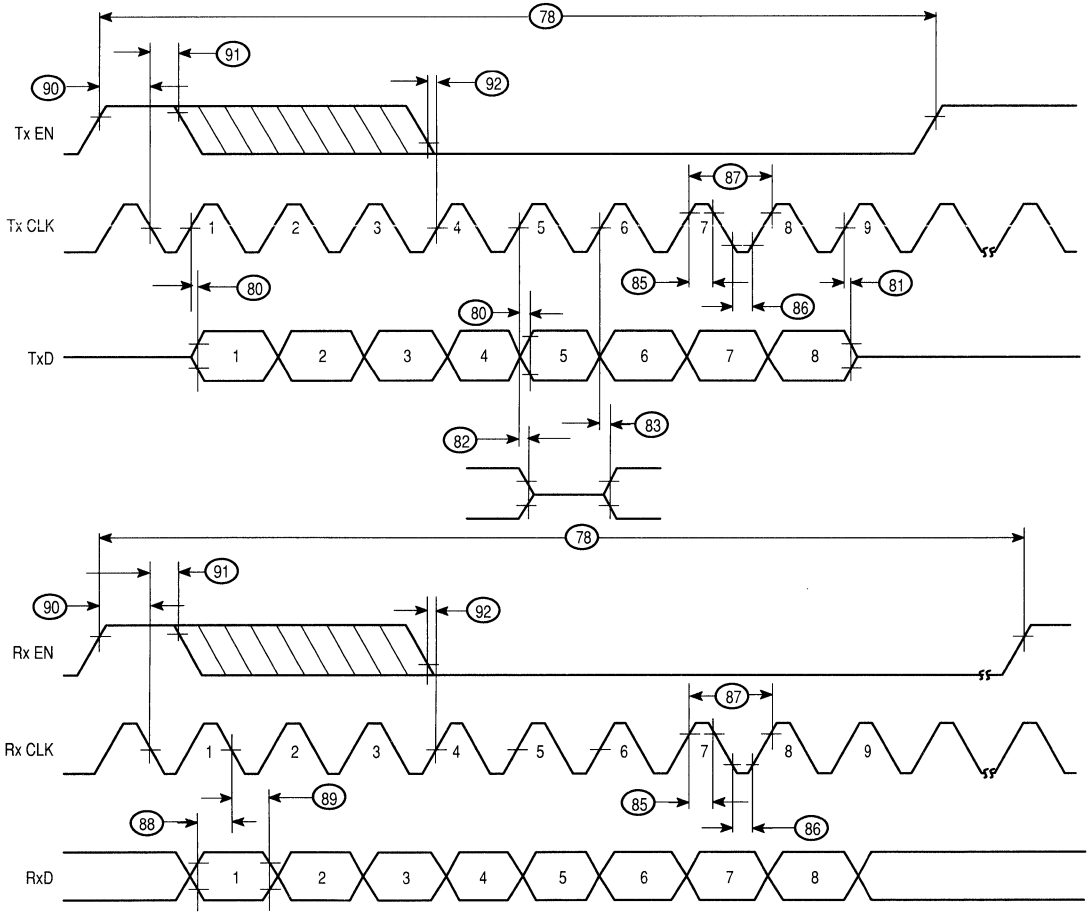


Figure 8-12. Timeslot Mode Timing Short-Frame Operation

8.11 MPU/DMA TIMING CHARACTERISTICS (80186 MODE)

(T_A = – 40°C to 85°C, V_{DD} = 5.0 V ± 10%)

Reference Number	Characteristic	Min	Max	Unit	Notes
93	$\overline{CS}$ Valid Before $\overline{RD}$ or $\overline{WR}$ Active	20	—	ns	
94	Address, $\overline{BHE}$ Valid Before $\overline{RD}$ or $\overline{WR}$ Active	20	—	ns	
95	Data Valid After $\overline{RD}$ Active (Read Access Time)	80	—	ns	
96	$\overline{RD}$ Pulse Width	80	—	ns	
97	$\overline{WR}$ Pulse Width	20	—	ns	
98	$\overline{CS}$ Valid After $\overline{RD}$ or $\overline{WR}$ Active	0	—	ns	
99	Address, $\overline{BHE}$ Valid After $\overline{CS}$ Active	0	—	ns	
100	Data Valid After $\overline{RD}$ Inactive (Read Hold Time)	20	50	ns	
101	Data High Impedance After $\overline{RD}$ Inactive	—	75	ns	
102	Data Valid Before $\overline{WR}$ Inactive (Write Set-Up TIme)	20	—	ns	
103	Data Valid After $\overline{WR}$ Inactive (Write Hold Time)	0	—	ns	
104	$\overline{HREQ}$ Active Delay	—	55	ns	
105	Asynchronous Set-Up Time	20	—	ns	24
106	Address Valid After MCLK Low	0	55	ns	
107	Address Valid Before $\overline{RD}$ or $\overline{WR}$ Active	0	—	ns	
108	Data Valid After MCLK High	0	55	ns	
109	Signal Active Delay	0	55	ns	
110	$\overline{RDY}$ Active Before T3 Rising Edge ($\overline{RDY}$ Set-Up Time)	20	—	ns	
111	$\overline{RDY}$ Active After T3 Rising Edge ($\overline{RDY}$ Hold Time)	20	—	ns	
112	Signal Inactive Delay time	0	55	ns	
113	Signal High-Impedance Delay Time	0	25	ns	
114	Address Valid After $\overline{RD}$ or $\overline{WR}$ Inactive (Address Hold Time)	20	—	ns	
115	Address High-Impedance Delay Time	0	25	ns	
116	Data Valid After $\overline{WR}$ Inactive (Write Data Hold TIme)	20	—	ns	
117	Data High-Impedance Delay Time	—	25	ns	
118	Data Valid Before $\overline{RD}$ Inactive (Read Data Set-Up Time)	20	—	ns	
119	Data Valid After $\overline{RD}$ Inactive (Read Data Hold TIme)	0	—	ns	
120	Chip Select Inactive to Chip Select Active	1	—	MCLK	

NOTES:

24. This is the time required for a signal to be valid before the MCLK edge if it is to be detected on this cycle.

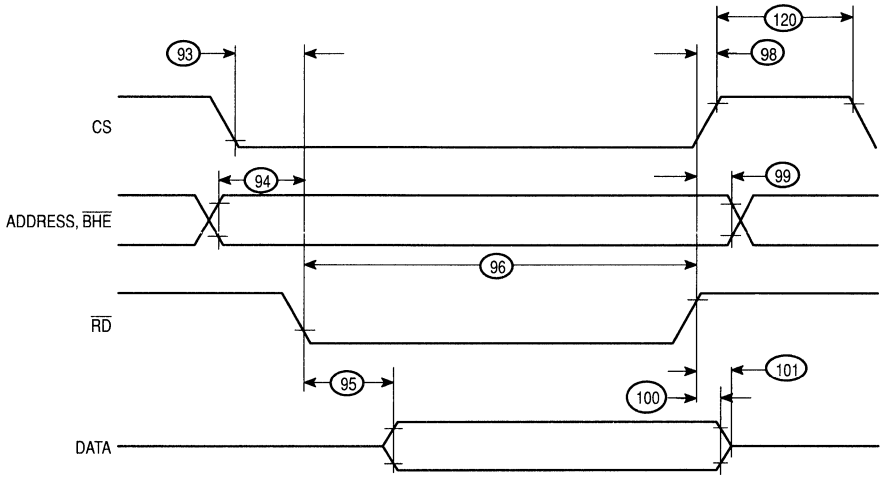


Figure 8-13. MPU Read Timing (80186 Mode)

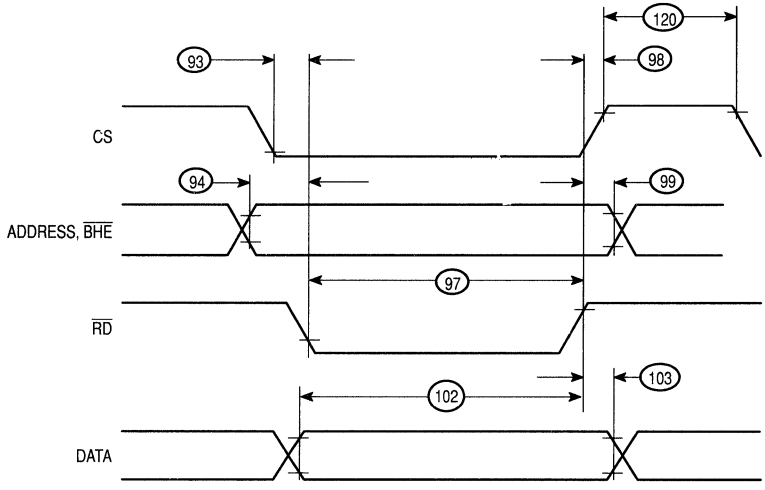


Figure 8-14. MPU Write Timing (80186 Mode)

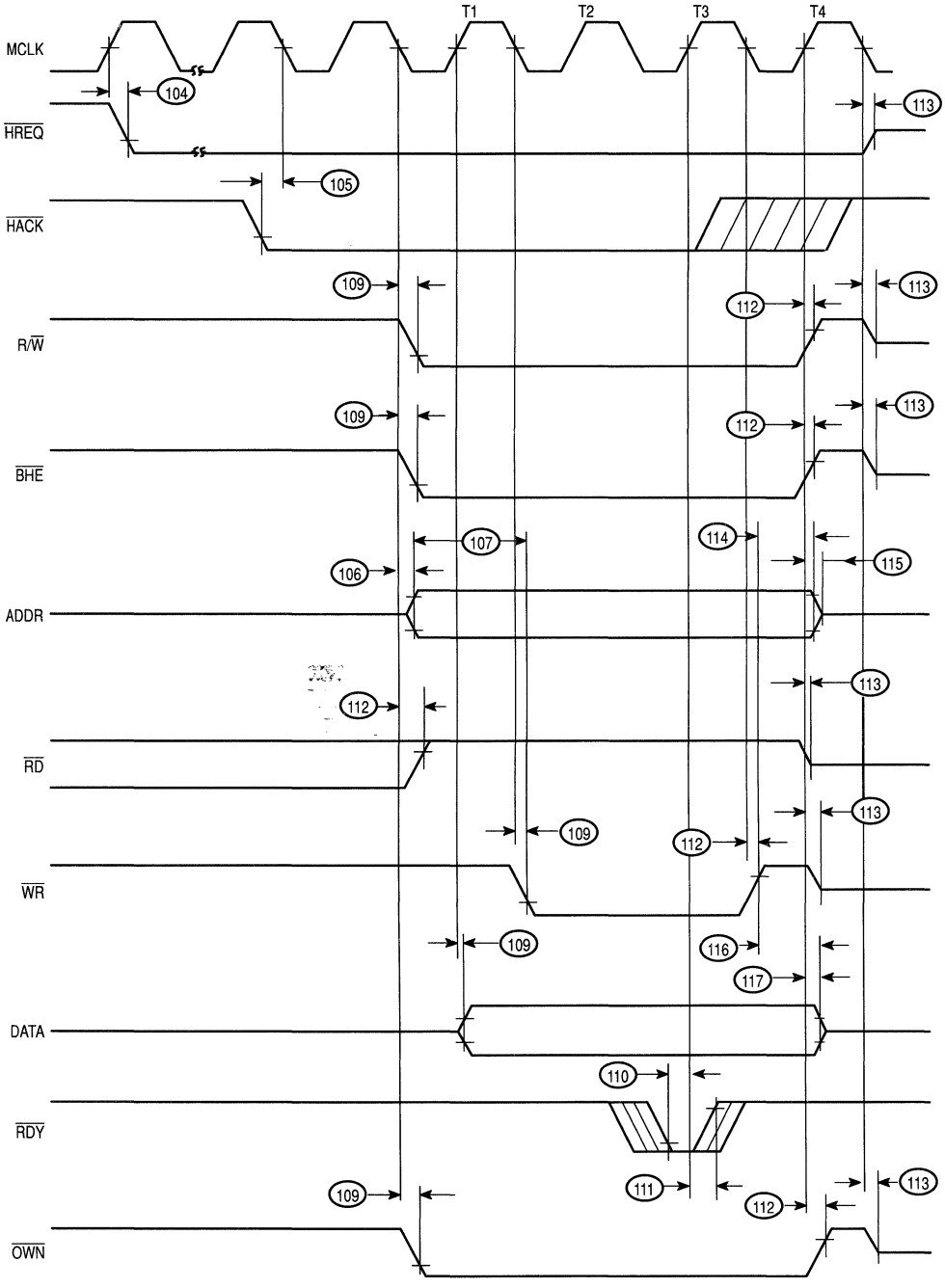


Figure 8-15. DMA Write Timing (80186 Mode)

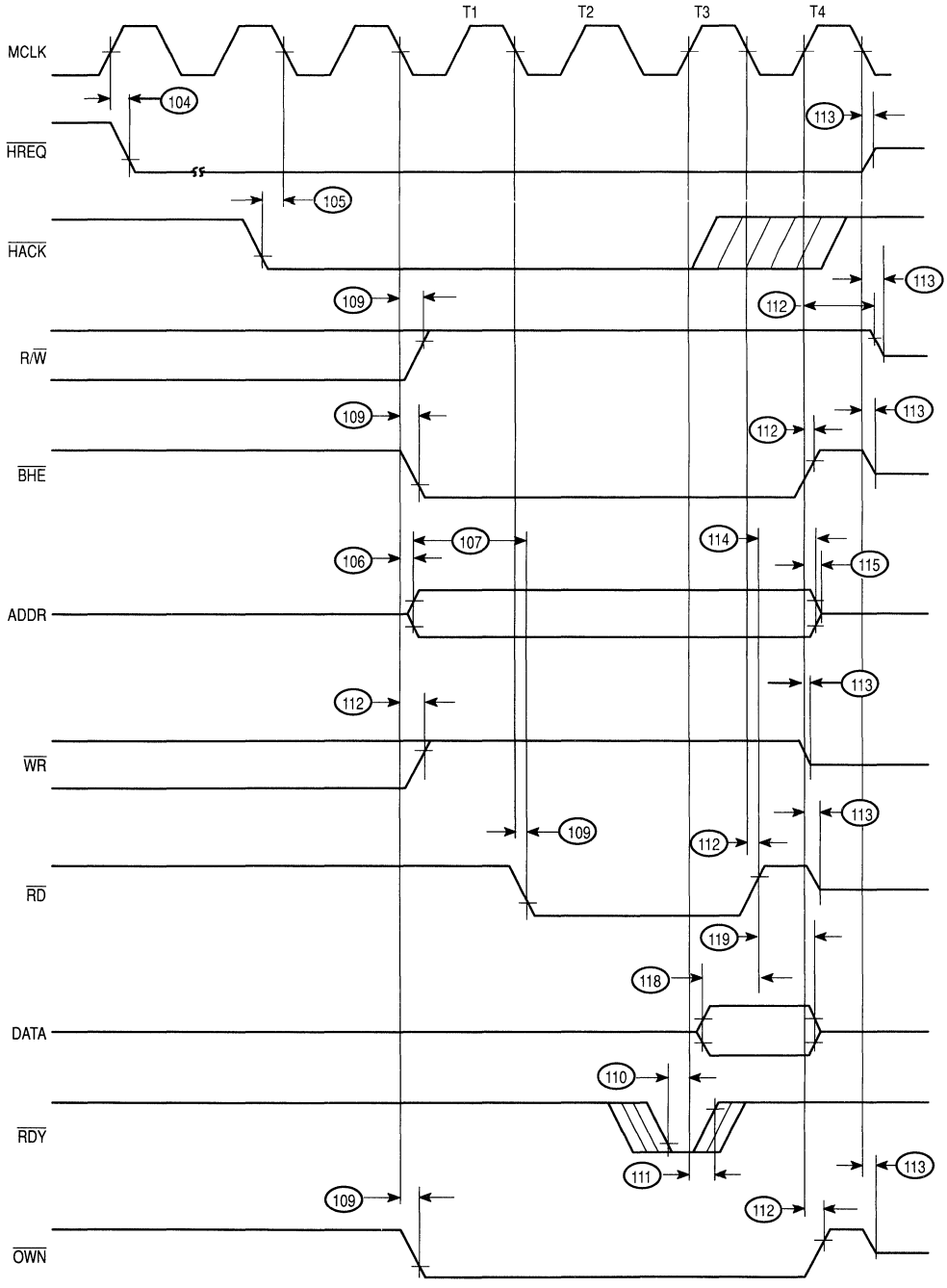
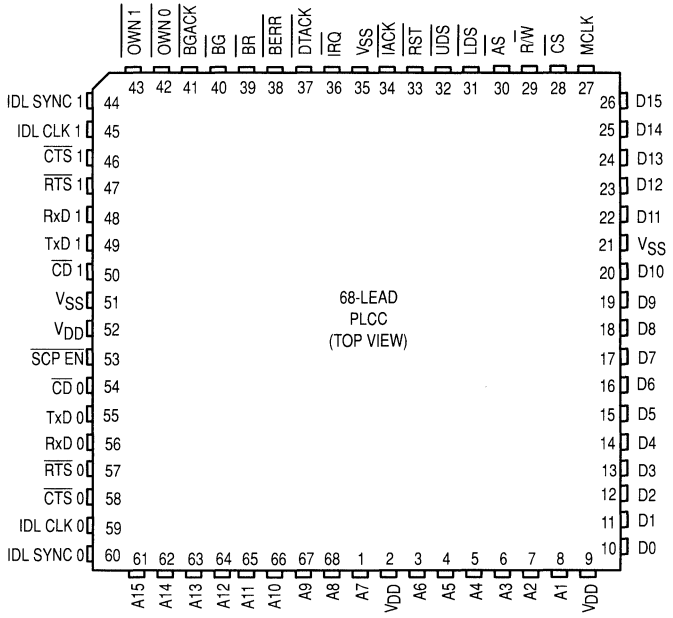


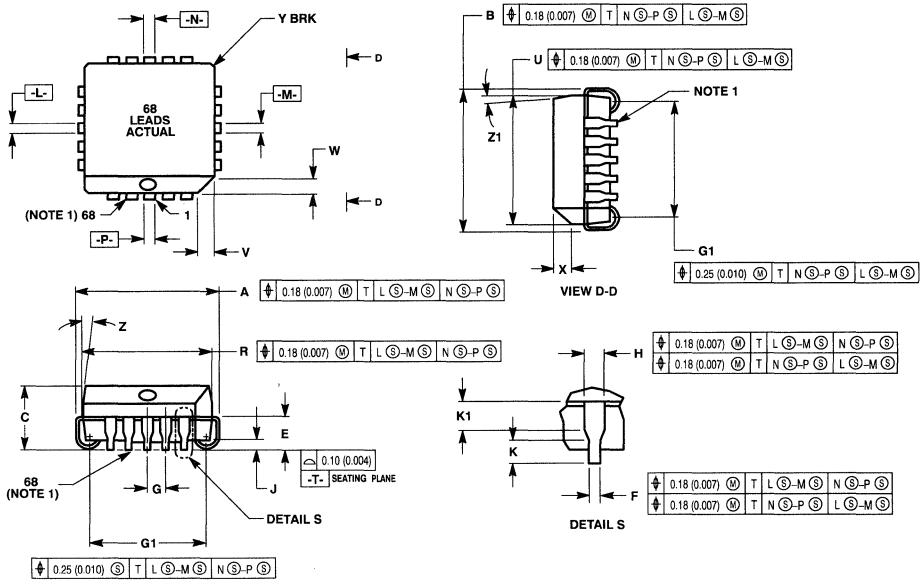
Figure 8-16. DMA Read Timing (80186 Mode)

SECTION 9
MECHANICAL DATA

9.1 PIN ASSIGNMENT



9.2 PACKAGE DIMENSIONS



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	25.02	25.27	0.985	0.995
B	25.02	25.27	0.985	0.995
C	4.20	4.57	0.165	0.180
E	2.29	2.79	0.090	0.110
F	0.33	0.48	0.013	0.019
G	1.27 BSC		0.050 BSC	
H	0.66	0.81	0.026	0.032
J	0.51	—	0.020	—
K	0.64	—	0.025	—
R	24.13	24.28	0.950	0.956
U	24.13	24.28	0.950	0.956
V	1.07	1.21	0.042	0.048
W	1.07	1.21	0.042	0.048
X	1.07	1.42	0.042	0.056
Y	—	0.50	—	0.020
Z	2°	10°	2°	10°
G1	23.12	23.62	0.910	0.930
K1	1.02	—	0.040	—
Z1	2°	10°	2°	10°

NOTES:
1. **DUE TO SPACE LIMITATION, CASE 779-02 SHALL BE REPRESENTED BY A GENERAL (SMALLER) CASE OUTLINE DRAWING RATHER THAN SHOWING ALL 68 LEADS.**
2. DATUMS -L-, -M-, -N-, AND -P- DETERMINED WHERE TOP OF LEAD SHOULDER EXIT PLASTIC BODY AT MOLD PARTING LINE.
3. DIM G1, TRUE POSITION TO BE MEASURED AT DATUM -T-, SEATING PLANE.
4. DIM R AND U DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION IS 0.25 (0.010) PER SIDE.
5. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
6. CONTROLLING DIMENSION: INCH.

Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawaku, Tokyo 141, Japan.
ASIA PACIFIC: Motorola Semiconductors (H.K.) Ltd.; Silicon Harbour Center, No. 2, Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

MOTOROLA

A23449-3 PRINTED IN USA 5/91 IMPERIAL LITHO 79164 20,000 MOS TEL YFABAA

MC145488

◊