



AP-286

**APPLICATION
NOTE**

80186/188 Interface to Intel Microcontrollers

PARVIZ KHODADADI
APPLICATIONS ENGINEER

October 1986



Order Number: 231784-001

Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products.

Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.

*Other brands and names are the property of their respective owners.

†Since publication of documents referenced in this document, registration of the Pentium, OverDrive and iCOMP trademarks has been issued to Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641
or call 1-800-879-4683

**80186/188
INTERFACE TO INTEL
MICROCONTROLLERS**

CONTENTS	PAGE
1.0 INTRODUCTION	1
1.1 System Overview	2
1.2 Application Examples	2
2.0 OVERVIEW OF THE 80186, 80C51, 8052, AND 8044	2
2.1 The 80186 Internal Architecture	2
2.2 The MCS-51 Internal Architecture	3
2.3 The 8044 Internal Architecture	3
3.0 80186/MICROCONTROLLER INTERACTION	4
4.0 SYSTEM INTERFACE	5
4.1 Command/Status Transfers	5
4.2 Data/Parameter Transfer	5
4.3 Interrupts	6
5.0 COMMAND AND STATUS	7
5.1 Commands	7
5.1.1 Acknowledging Interrupt	7
5.1.2 Operations	7
5.1.3 Illegal Commands	9
5.2 Status	9
5.2.1 Interrupt	9
5.2.2 DMA Operation	9
5.2.3 Error	9
5.2.4 Request to Send	9
5.2.5 Clear to Send	9
5.2.6 Event	9
6.0 HARDWARE DESCRIPTION	10
6.1 Reset	10
6.2 Sending Commands	10
6.3 DMA Transfers	11
6.4 Reading Status	11



CONTENTS	PAGE
7.0 80186/8044 INTERFACE	12
7.1 Configuring the 8044	12
7.2 Transferring a Message with the 8044	13
7.3 Receiving a Message with the 8044	13
7.4 Dumping the 8044 Registers	14
7.5 Aborting an Operation	14
7.6 Disabling Transmission or Reception	14
7.7 Handling Interrupts	15

CONTENTS	PAGE
8.0 8044 IN EXPANDED OPERATION	15
8.1 Transmitting a Message in Expanded Operation	15
8.2 Receiving a Message in Expanded Operation	15
9.0 CONCLUSION	15
APPENDIX A: SOFTWARE	A-1



1.0 INTRODUCTION

Systems which require I/O processing and serial data transmission are very software intensive. The communication task and I/O operations consume a lot of the system's intelligence and software. In many conventional systems the central processing unit carries the burden of all the communication and I/O operations in addition to its main routines, resulting in a slow and inefficient system.

In an ideal system, tasks are divided among processors to increase performance and achieve flexibility. One attractive solution is the combination of the Intel highly integrated 80186 microprocessor and the Intel 8-bit microcontrollers such as the 80C51, 8052, or 8044. In such a system, the 80186 provides the processing power and the 1 Mbyte memory addressability, while the controller provides the intelligence for the I/O operations and data communication tasks. The 80186 runs application programs, performs the high level communication tasks, and provides the human interface. The microcontroller performs 8-bit math and single bit boolean operations, the low level communication tasks, and I/O processing.

This application note describes an efficient method of interfacing the 16-bit 80186 high integration microprocessor to the 80C51, 8052, or the microcontroller-based 8044 serial communication controller. The interface hardware shown in Figure 1.1, is very simple and may be implemented with a programmable logic device or a gate-array. The 80186 and the microcontroller may run asynchronously and at different speeds. With this technique data transfers up to 200 Kbytes per second can be achieved between a 12 MHz microcontroller and an 8 MHz 80186.

The 8-bit 80188 high integration microprocessor can also be used with the same interface technique. The performance of the interface is the same since an 8-bit bus is used.

Interface to the 8044, 80C51, and the 8052 is identical because they have identical pinouts (some pins have alternate functions). As an example, the software procedures for the 8044/80186 interface, which is the building block for the application driver, is supplied in this Application Note.

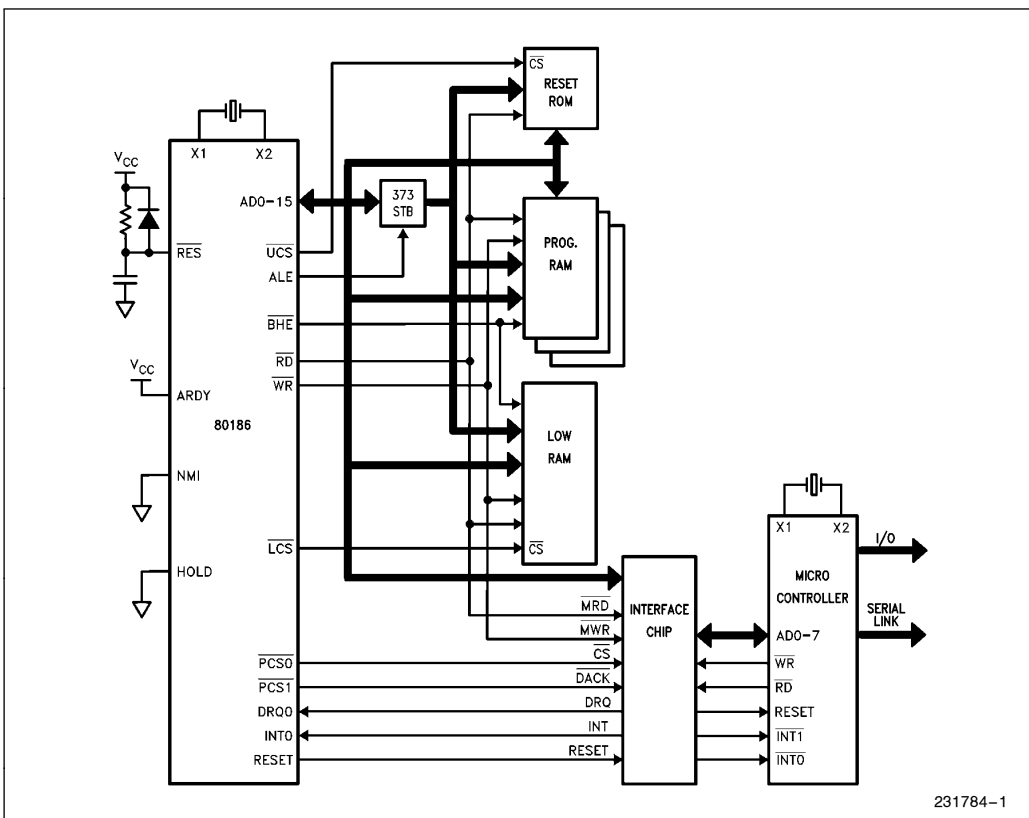


Figure 1.1. 80186/Microcontroller Based System

1.1 System Overview

The 80186 and the microcontrollers are processors. They each access memory and have address/data, read, and write signals. There are three common ways to interface multiple processors together:

- 1) First In First Out (FIFO)
- 2) Dual Port RAM (DPRAM)
- 3) Slave Port

The FIFO interface, compared to DPRAM, requires less TTL and is easier to interface; however, FIFOs are expensive. The DPRAM interface is also expensive and even more complex. When DPRAM is used, the address/data lines of each processor must be buffered, and hardware logic is needed to arbitrate access to DPRAM. The slave port interface given here is cheaper and easier than both FIFO and DPRAM alternatives.

The 80186 processor, when interfaced to this circuit, views the microcontroller as a peripheral chip with 8-bit data bus and no address lines (see Figure 1.1). It can read status and send commands to the microcontroller at any time. The microcontroller becomes a slave co-processor while keeping its processing power and serial communication capabilities.

The microcontrollers, with the interface hardware, have a high level command interface like many other data communication peripherals. For example, the 80186 can send the microcontroller commands such as Transmit or Configure. This means the designer does not have to write low level software to perform these tasks, and it offloads the 80186 to serve other functions in the application.

1.2 Application Examples:

The combination of the 80186 and a microcontroller basically provides all the functions that are needed in a system: a 16-bit CPU, 8-bit CPU, DMA controller, I/O ports, and a serial port. The 80C51 and the 8052 have an on-chip asynchronous channel, while the 8044 has an intelligent SDLC serial channel. In addition, many other functions such as timers, counters, and interrupt controllers are integrated in both the 80186 and the microcontrollers.

Applications of the system described above are in the area of robotics, data communication networks, or serial communication backplanes. A typical example is copiers. Different segments of the copy machine like the motor, paper feed, diagnostics, and error/warning displays are all controlled by microcontrollers. Each segment receives orders from and replies to the central processor which consists of the 80186 interfaced with a microcontroller.

Another common application is in the area of process controllers. An example is a central control unit for a multiple story building which controls the heating, cooling, and lighting of each room in each floor. In each room a microcontroller performs the above functions based on the orders received from the central processor. Depending on the throughput and type of the serial communication required, the 8044 or the 80C51 (8052) may be selected for the application.

2.0 OVERVIEW OF THE 80186, 80C51, 8052, AND 8044

This section briefly discusses the features of the microcontrollers and the 80186. For more information about these products please refer to the Intel Microcontroller and Microsystem components hand-books. Readers familiar with the above products may skip this section.

2.1 The 80186 Internal Architecture

The 80186 contains an enhanced version of Intel's popular 8086 CPU integrated with many other features common to most systems (Figure 2.1). The 16-bit CPU can access up to 1 Mbyte of memory and execute instructions faster than the 8086. With speed selection of 8, 10, and 12.5 MHz, this highly integrated product is the most popular 16-bit microprocessor for embedded control applications.

The on-chip DMA controller has two channels which can each be shared by multiple devices. Each channel is capable of transferring data up to 3.12 Mbytes per second (12.5 MHz speed). It offers the choice of byte or word transfer. It can be programmed to perform a burst transfer of a block of data, transfer data per specified time interval, or transfer data per external request.

The on-chip interrupt controller responds to both external interrupts and interrupts requested by the on-chip peripherals such as the timers and the DMA channels. It can be configured to generate interrupt vector addresses internally like the microcontrollers or externally like the popular 8259 interrupt controller. It can be configured to be a slave controller to an external interrupt controller (iRMX 86 mode) or be master for one or two 8259s which in turn may be masters for up to 8 more 8259s. When configured in master mode, each channel can support up to 64 external interrupts (128 total).

Three 16-bit timers are also integrated on the chip. Timer 0 and timer 1 can be configured to be 16-bit counters and count external events. If configured as timers, they can be started by software or by an external event. Timer 0 and 1 each contain a timer output pin. Transitions on these pins occur when the timers reach one of the two possible maximum counts. Timer

2 can be used as a prescaler for timer 0 and 1 or can be used to generate DMA requests to the on-chip DMA channel.

Finally, the integrated clock generator, the wait state generator, and the chip select logic reduce the external logic necessary to build a processing system.

2.2 The MCS-51 Internal Architecture

The 80C51BH, as shown in Figure 2.2, consists of an 8-bit CPU which can access up to 64 Kbytes of data memory (RAM) and 64 Kbytes of program memory (ROM). In addition, 4 Kbytes of ROM and 128 bytes of RAM are built onto the chip.

The on-chip interrupt controller supports five interrupts with two priority levels. There are two timers integrated in the 80C51. Timer 0 and 1 can be configured as 8-bit or 16-bit timers or event counters.

Finally the integrated full duplex asynchronous serial channel provides the human interface or communica-

tion capability with other microcontrollers. The UART supports data rates up to 500 kHz (with 15 MHz crystal) and can distinguish between address bytes and data bytes.

The 8052 has the same features as the 80C51 except it has 8 Kbytes of on-chip ROM and 256 bytes of on-chip RAM. In addition the 8052 has another timer which may be configured as the baud rate generator for the serial port.

2.3 The 8044 Internal Architecture

The 8044 has all the features of the 80C51. In addition the on-chip RAM size is increased to 192 bytes and an intelligent HDLC/SDLC serial channel (SIU) replaces the 80C51 serial port (see Figure 2.3). It supports data rates up to 2.4 Mbps when an external clock is used and 375 Kbps when the clock is extracted from the data line. The serial port can be used in half duplex point to point, multipoint, or one-way loop configurations.

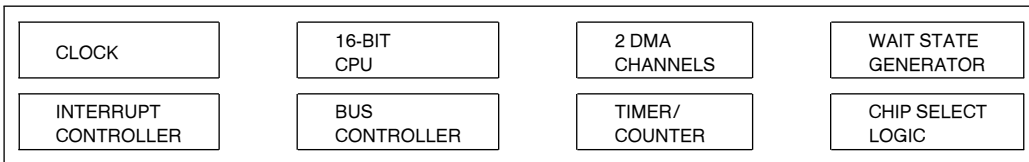


Figure 2.1. 80186 Block Diagram

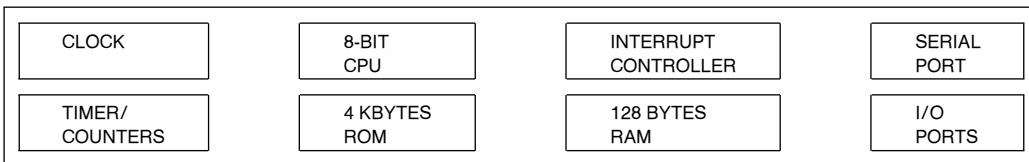


Figure 2.2. 80C51 Block Diagram

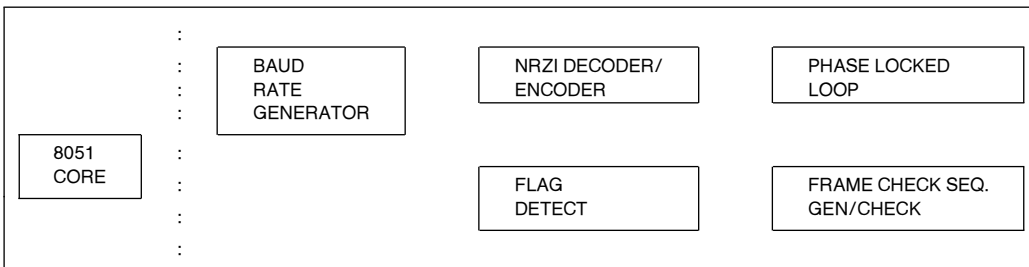


Figure 2.3. 8044 Block Diagram



Figure 2.4. 8044 Automatic Response to SDLC Commands

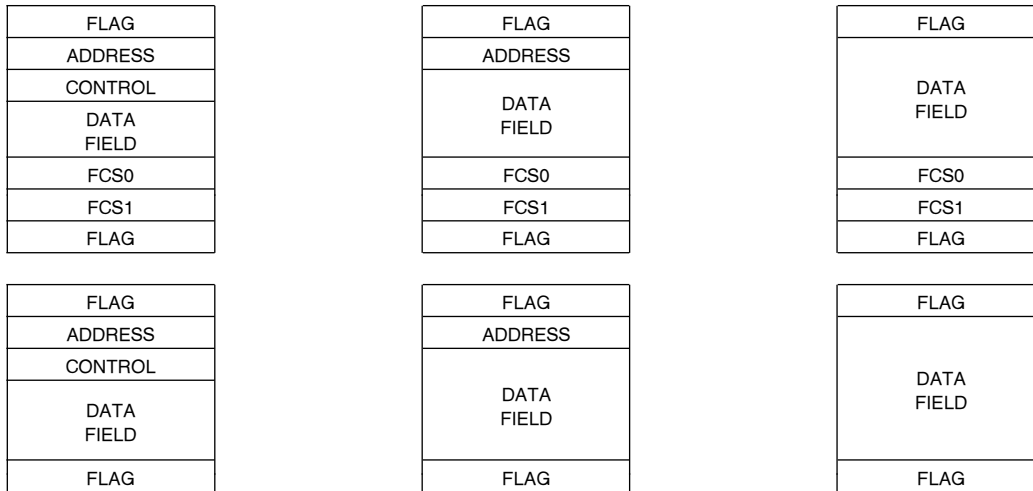


Figure 2.5. The 8044 Frame Formats

The SIU is called an intelligent channel because it responds to some SDLC commands automatically without the CPU intervention when it is set in auto mode. These automatic responses substantially reduce the communication software. Figure 2.4 gives the commands and the automatic responses.

The 8044 supports many types of frames including the standard SDLC format. Figure 2.5 shows the types of frames the 8044 can transmit and receive. If a format with an address byte is chosen, the 8044 performs address filtering during reception and transmits the contents of the station address register during transmission automatically. If a format with FCS bytes is chosen, the 8044 performs Cyclic Redundancy Check (CRC) during reception and calculates the FCS bytes during transmission of a frame in hardware. Two preamble bytes (PFS) may optionally be added to the frames. Formats that include the station address and the control byte are supported both in the auto and flexible modes.

3.0 80186/MICROCONTROLLER INTERACTION

The 80186 communicates with the microcontroller (8044, 80C51 or 8052) through the system's memory and the Command/Data and Status registers. The CPU creates a data structure in the memory, programs the DMA controller with the start address and byte count of the block, and issues a command to the microcontroller. A hypothetical block diagram of a microcontroller when used with the interface hardware is given in Figure 3.1.

Chip select and interrupt lines are used to communicate between the microcontroller and the host. The inter-

rupt is used by the microcontroller to draw the 80186's attention. The Chip Select is used by the 80186 to draw the microcontroller's attention to a new command.

There are two kinds of transfers over the bus: Command/Status and data transfers. Command/Status transfers are always performed by the CPU. Data transfers are requested by the microcontroller and are typically performed by the DMA controller.

The CPU writes commands using CS and WR signals and interrupts the microcontroller. The microcontroller reads the command, decodes it and performs the necessary actions. The CPU reads the status register using CS and RD signals (see Figure 4.1).

To initiate a command like TRANSMIT or CONFIGURE, a write operation to the microcontroller is issued by the CPU. A read operation from the CPU gives the status of the microcontroller. Section 5 discusses details on these commands and the status.

Any parameters or data associated with the command are transferred between the system memory and the microcontroller using DMA. The 80186 prepares a data block in memory. Its first byte specifies the length of the rest of the block. The rest of the block is the information field. The CPU programs the DMA controller with the start address of the block, length of the block and other control information and then issues the command to the microcontroller.

When the microcontroller requires access to the memory for parameter or data transfer, it activates the 80186 DMA request line and uses the DMA controller to achieve the data transfer. Upon completion of an operation, the microcontroller interrupts the 80186. The CPU then reads results of the operation and status of the microcontroller.



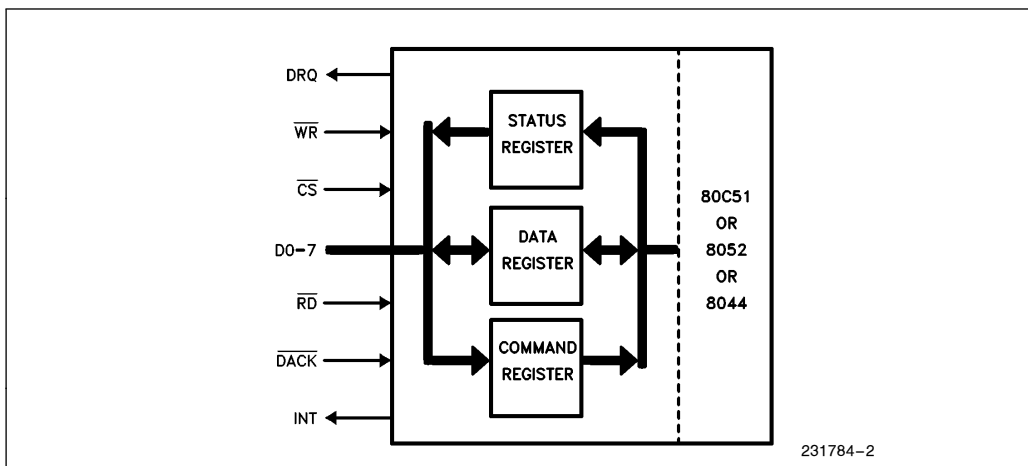


Figure 3.1. Microcontroller Plus the Interface Hardware Block Diagram

4.0 SYSTEM INTERFACE

There are two kinds of transfers over the bus: command/status and data transfers. The command/status transfers are always initiated and performed by the 80186. The data transfers are requested by the microcontroller using the DMA request (DRQ) line. In relatively slow systems the 80186 might also perform the data transfers. In that case, the request from the microcontroller will serve as an interrupt to the CPU. This mode of operation depends on the serial data rate.

The system interface performs command/status transfers, data/parameter transfers, and interrupts. This section describes the interface between the 80186 and a microcontroller shown in Figure 1.1. Section 6 describes the interface hardware.

4.1 Command/Status Transfers

The 80186 controls the microcontroller by writing into the command/data register and reading from the status register. The CPU writes a command by activating the chip select (PCS0), putting the command onto the data bus, and activating the WR signal. The command byte is latched into the command/data register, and the microcontroller is interrupted. In the interrupt service routine, the microcontroller reads the command byte from the command/data register, decodes the command byte, and activates the DRQ for data or parameter transfer if the decoded command requires such transfer.

At the end of parameter transfer the microcontroller updates the status register and interrupts the 80186.

4.2 Data/Parameter Transfer

Data/parameter transfers are controlled by a pair of REQUEST/ACKNOWLEDGE lines: DMA Request line (DRQ) and DMA Acknowledge line (DACK). Data and parameters are transferred via the Command/Data register to or from memory.

In order to request a transfer from memory, the microcontroller activates the DRQ pin. The DRQ signal goes active after a read operation by the microcontroller. In response, the 80186 DMA controller performs a byte transfer from the memory to the Command/Data register. Data is transferred on the bus and written into the Command/Data register on the rising edge of the 80186 WR signal (\overline{MWR}), which is activated by the DMA controller. Figure 4.2 shows the write timing.

In order to request a transfer to memory, the microcontroller activates the DRQ signal and outputs the data into the Command/Data latch. When the microcontroller WR signal goes active, DRQ is set. In response, the DMA performs the data transfer and resets the DRQ signal. Figure 4.3 shows the read timing.



4.3 Interrupt

The microcontroller reports on completion of an event by updating the status register and raising the interrupt signal assuming this signal is initially low. The interrupt is cleared by the command from the CPU where

the INTERRUPT ACKNOWLEDGE bit is set (MD7). The INTA bit is the most significant bit of the command byte. Figure 4.4 and 4.5 show the interrupt timing. Note that it is the responsibility of the CPU to clear the interrupt in order to prevent a deadlock.

80186 Pin Name			Function
\overline{CS}	\overline{RD}	\overline{WR}	
1	X	X	No Transfer to/from Command/Status
0	1	1	
0	0	0	Illegal
0	0	1	Read from Status Register
0	1	0	Write to Command/Data Register
\overline{DACK}	\overline{RD}	\overline{WR}	
1	X	X	No Transfer
0	1	1	
0	0	0	Illegal
0	0	1	Data Read from DMA Channel
0	1	0	Data Write to DMA Channel

NOTE:
Only one of CS, DACK may be active at any time.

Figure 4.1. Data Bus Control Signals and Their Functions

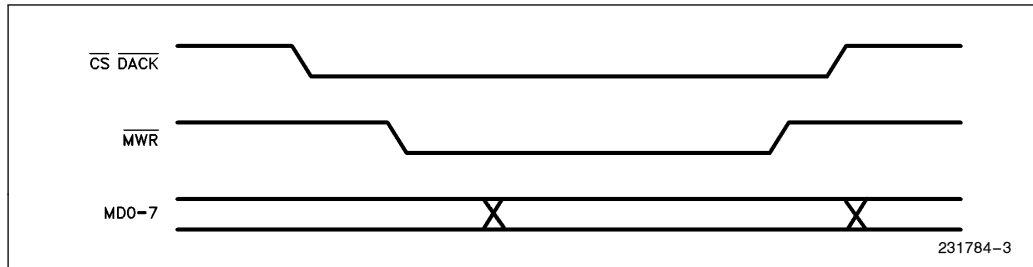


Figure 4.2. Write Timing

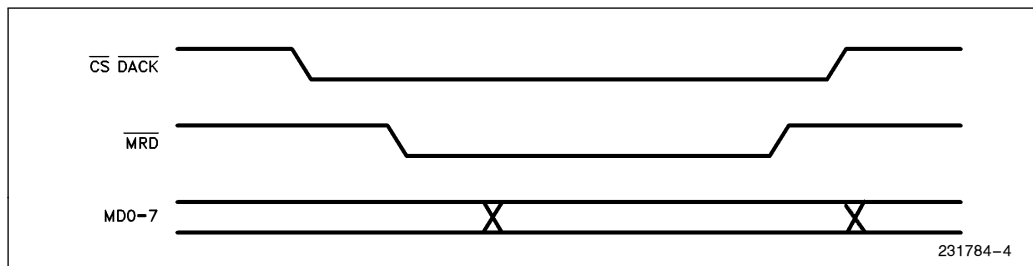


Figure 4.3. Read Timing



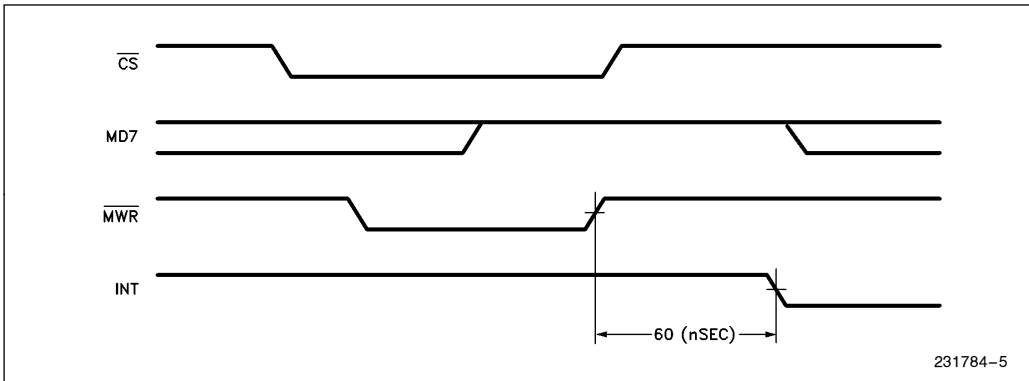


Figure 4.4. Interrupt Timing (Going Inactive)

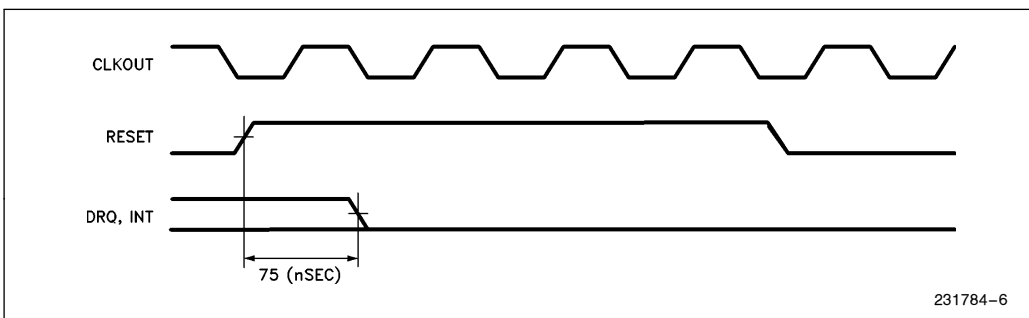


Figure 4.5. Reset Timing

5.0 COMMANDS AND STATUS

This section specifies the format of the commands and status. The commands and status given here are similar to most common coprocessors and data communication peripherals (e.g., the 82588 and 82586). The user may add more commands or redefine the formats for his/her own specific application.

5.1 Commands

A command is given to the microcontroller by writing it into the Command/Data register and interrupting the microcontroller. The command can be issued at any time; but in case it is not accepted, the operation is treated like a NOP and will be ignored (although the INT will be updated).

Format:

7	6	5	4	3	2	1	0
INTA	X	X	X	OPERATION			

5.1.1 ACKNOWLEDGING INTERRUPT (BIT 7)

The INTA bit, if set, causes the interrupt hardware signal and the interrupt bit to be cleared. This is the

only way to clear the interrupt bit and reset the 80186 interrupt signal other than by a hardware reset.

5.1.2 OPERATIONS (BITS 0-3)

The OPERATION field initiates a specific operation. The microcontroller executes the following commands in software:

- NOP
 - ABORT
 - TRANSMIT*
 - CONFIGURE*
 - DUMP*
 - RECEIVE*
 - TRA-DISABLE
 - REC-DISABLE
- *Requires DMA operation.

The above operations except ABORT are executed only when the microcontroller is not executing any other operation. Abort is accepted only when the CPU is performing a DMA operation.



Operations that require parameter transfer (e.g., CONFIGURE and DUMP) or data transfer (e.g., TRANSMIT and RECEIVE) are called parametric operations. The remaining are called non-parametric operations.

An operation is initiated by writing into the command register. This causes the microcontroller to execute the command decode instructions. Some of the operations cause the microcontroller to read parameters from memory. The parameters are organized in a block that starts with an 8-bit byte count. The byte count specifies the length of the rest of the block. Before beginning the operation, the DMA pointer of the DMA channel must point to the byte count. There is no restriction on the memory structure of the parameter block as long as the microcontroller receives the next byte of the block for every DMA request it generates. Transferring the bytes is the job of the 80186 DMA controller.

The microcontroller requests the byte-count and determines the length of the parameter block. It then requests the parameters.

Upon completion of the operation, (when interrupt is low) the microcontroller updates the status, raises the interrupt signal, and goes idle.

NOP

This operation does not affect the microcontroller. It has no parameters and no results.

ABORT

This operation attempts to abort the completion of an operation under execution. It is valid for CONFIGURE, TRANSMIT, DUMP, and RECEIVE. It is ignored for any of the above if transfer of parameters has already been accomplished. The microcontroller, upon reception of the ABORT command, stops the DMA operation and issues an Execution-Aborted interrupt.

TRANSMIT

This operation transmits one message. A message may be transmitted as an SDLC frame by the 8044, or in ASYNC protocol by the 80C51 or the 8052 serial port.

Figure 5.1 shows the format of the Transmit block. A typical transmit operation parameter block includes the destination address and the control byte in the information field. As an example, see the 8044 transmit block in Figure 7.2.

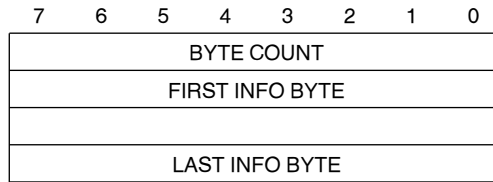


Figure 5.1. Format of Transmit Block

The transmit operation will either complete the execution or be aborted by a specific ABORT operation. A Transmit-Done or Execution-Aborted interrupt is issued upon completion of this operation.

CONFIGURE

This operation configures the microcontroller's internal registers. The length and the part of the configuration block that is modified are determined by the first two bytes of the command parameter (see Figure 5.2). The FIRST BYTE specifies the first register in the configure block that will be configured, and the BYTE COUNT specifies the number of registers that will be configured starting with the FIRST BYTE. For example, if the FIRST BYTE is 1 and the BYTE COUNT is the length of the configure block, then all of the registers are updated. If FIRST BYTE is 4 and BYTE COUNT is 2, then only the fourth register in the configure block is updated. Minimum byte count is 2.

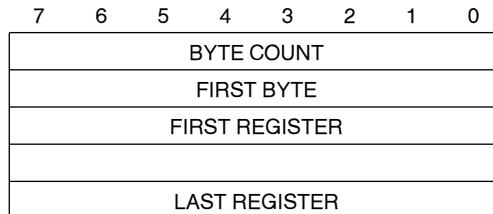


Figure 5.2. Format of Configure Block

A Configure-Done interrupt is issued when the operation is done unless ABORT was issued during the DMA operation.

DUMP

This operation causes dumping of a set of microcontroller internal registers to system memory. Figure 7.4 shows the format of the 8044 DUMP block.

The DUMP operation will either complete the execution or be aborted by a specific ABORT operation. A Dump-Done or Execution-Aborted interrupt is issued upon completion of this operation.



RECEIVE

This operation enables the reception of frames. It is ignored if the microcontroller's serial channel is already in reception mode.

The serial port receives only frames that pass the address filtering. The microcontroller transfers the received information and the byte count to the system memory using DMA. The completion of frame reception causes a Receive-Done event.

REC-DISABLE

This operation causes reception to be disabled. If transfer of data to the 80186 memory has already begun, then it is treated like the ABORT command. This operation has no parameters. REC-DISABLE is accepted only when the microcontroller's serial port is in receive mode.

TRA-DISABLE

This operation causes the transmission process to be aborted. If the microcontroller is fetching data from 80186 memory, then it is treated like the ABORT command. This operation has no parameters. It is accepted only when the serial port is in transmit mode.

5.1.3 ILLEGAL COMMANDS

Parametric and non-parametric commands except ABORT will be rejected (interrupt will not be set) if the microcontroller is already executing a command.

ABORT is rejected if issued when the microcontroller is not requesting DMA operation, or a non-Parametric execution is performed, or transfer of parameters/data has already been accomplished.

DMA operations shall not be aborted by any non-parametric or parametric command except by the ABORT command.

REC-DISABLE and TRA-DISABLE will not be accepted if the serial channel is idle.

5.2 Status

The microcontroller provides the information about the last operation that was executed, via the status register.

The microcontroller reports on these events by updating a status register and raising the INTERRUPT signal. Information from the status register is valid provided the interrupt signal is high or bit 0 of the status being read is set.

Format:

7	6	5	4	3	2	1	0
CTS*	RTS*	E	EVENT	DMA	INT		

*8044 only

5.2.1 INTERRUPT (BIT 0)

The interrupt bit is set together with the hardware interrupt signal. Setting the INT bit indicates the occurrence of an event. This bit is cleared by any command whose INTA bit is set. Status is valid only when this bit is set.

5.2.2 DMA OPERATION (BIT 1)

The DMA bit, when set, indicates that a DMA operation is in progress. This bit is set if the command received by the microcontroller requires data or parameter transfer. If this bit is clear, DRQ will be inactive. The DMA bit, when cleared, indicates the completion of a DMA operation.

5.2.3 ERROR (BIT 5)

The E bit, if set, indicates that the event generated for the operation that was completed contains a warning, or the operation was not accepted.

5.2.4 REQUEST TO SEND (BIT 6)

The RTS bit, if clear, indicates that the serial channel is requesting a transmission.

5.2.5 CLEAR TO SEND (BIT 7)

The CTS bit indicates that, if the RTS bit is clear, the serial port is active and transmitting a frame.

5.2.6 EVENT (BITS 2-4)

The event field specifies why the microcontroller needs the attention of the 80186.

The following events may occur:

- CONFIGURE-DONE
- TRANSMIT-DONE
- DUMP-DONE
- RECEIVE-DONE
- RECEPTION-DISABLED
- TRANSMISSION-DISABLED
- EXECUTION-ABORTED



CONFIGURE-DONE

This event indicates the completion of a CONFIGURE operation.

TRANSMIT-DONE

This event indicates the completion of the TRANSMIT operation.

If the E bit is set, it indicates that the transmit buffer was already full.

DUMP-DONE

This event indicates that the DUMP operation is completed.

RECEIVE-DONE

This event indicates that a frame has been received and stored in memory.

The format of the received message is indicated in Figure 5.3.

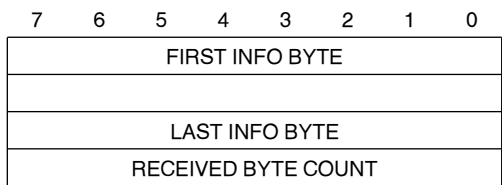


Figure 5.3. Format of Receive Block

Following the byte count, a few more bytes relating to the received frame such as the source address and the control byte may be transferred to the system memory using DMA. As an example, see the 8044 receive block in Figure 7.3.

Note that the format of a frame received by the microcontroller serial channel is configured by the CONFIGURE command.

If the E bit is set, buffer overrun has occurred.

RECEPTION-DISABLED

This event is issued as a result of a RCV-DISABLE operation that causes part of a frame to be disabled.

If the E bit is set, the serial port was already disabled, and the RCV-DISABLE is not accepted.

TRANSMISSION-DISABLED

This event is issued as a result of a TRA-DISABLE operation that causes transmission of a frame to be disabled.

The E bit, if set, indicates that the TRA-DISABLE operation was not accepted since the serial port was already idle, or transmission of a frame has already been accomplished.

EXECUTION-ABORTED

This event indicates that the execution of the last operation was aborted by the ABORT command.

If the E bit is set, ABORT was issued when the microcontroller was not executing any commands.

6.0 HARDWARE DESCRIPTION

The interface hardware shown in Figures 6.1 and 6.2 are identical. The difference is the status register. In Figure 6.2, an external latch is used to latch the status byte. This hardware is recommended if an extra I/O port on the microcontroller is required for some other applications, or external program and data memory is required for the microcontroller. The hardware shown in Figure 6.1 makes use of one of the microcontroller's I/O ports (Port 1) to latch the status to minimize hardware. The discussion of Sections 1 through 5 apply to both schematics.

6.1 Reset

After an 80186 hardware reset, the microcontroller is also reset. The on-chip registers are initialized as explained in the Intel Microcontroller Handbook. The reset signal also clears the 80186 interrupt and the microcontroller interrupt signals by resetting FF3 (Flip-Flop 3) and FF2 (Flip-Flop 2). Figure 4.5 shows the RESET timing.

6.2 Sending Commands

A bidirectional latched transceiver (74ALS646) is used for the Command/Data register. When the 80186 writes a command to the Command/Data register, it interrupts the microcontroller. The interrupt is generated only when bit 7 (INTA) of the command byte is set. When the 80186 PCS0 and WR signals go active to write the command, FF2 will be set and FF3 will be cleared. The output of FF3 is the interrupt to the 80186 and the INT status bit. The INT bit is cleared immediately to indicate that the status is no longer valid. The output of FF2 is the interrupt to the microcontroller. A high to low transition on this line will interrupt the microcontroller. The interrupt signal will be cleared as soon as the microcontroller reads the command from the Command/Data register.



6.3 DMA Transfers

In the interrupt service routine the command is decoded. If it requires a DMA transfer, the microcontroller sets the DMA bit of the status register which activates the DMA request signal. DRQ active causes the 80186 on-chip DMA to perform a fetch and a deposit bus cycle. The first DMA cycle clears the DRQ signal (FF1 is cleared). When the microcontroller performs a read or write operation, the output of the FF1 will be set, and DRQ goes active again.

The DMA controller transfers a byte from system memory to the Command/Data register. Data is latched when the 80186 PCS1 and WR signals go active. PCS1 and WR active also clear FF1. The microcontroller monitors the output of FF1 by polling the P3.3 pin. When FF1 is cleared the microcontroller reads the byte from the Command/Data register. The P3.3 pin is also the interrupt pin. If a slow rate of transfer is acceptable, every DMA transfer can be interrupt driven to allow the microcontroller to perform other tasks.

The DMA controller transfers a byte from the Command/Data register to system memory by activating

the 80186 PCS1 and RD signals. PCS1 and RD active also clear FF1. When FF1 is cleared the microcontroller writes the next byte to the Command/Data register.

When all the data is transferred, the microcontroller clears the DMA status bit to disable DRQ. It then updates the status, sets the INT bit, and interrupts the 80186.

If the interface hardware in Figure 6.1 is used P1.1 is the DMA status bit and P1.0 is the INT bit. The microcontroller enables or disables them by writing to port 1. In Figure 6.2, DRQ or INT is disabled or enabled by writing to the 74LS374 status register. Note that the INT status bit is cleared by the hardware when the 80186 writes a command.

6.4 Reading Status

The command is written and the status is read with the same chip select (PCS0), although the status is read through the 74LS245 transceiver and the command is written to the Command/Data register.

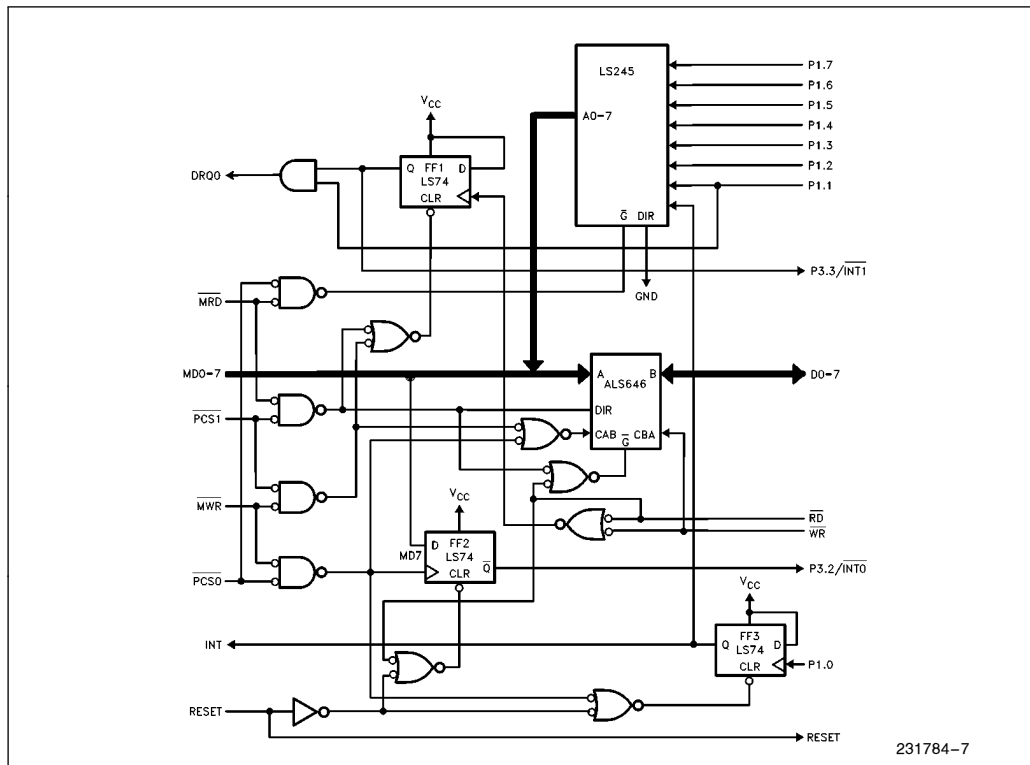


Figure 6.1. Hardware Interface (Port 1 is the Status Register)

For other microcontrollers the format of the configure block should be modified accordingly. For example, the 80C51 serial port registers (e.g., T2CON, SCON) replace the 8044 SIU registers in the configure block.

7	6	5	4	3	2	1	0
BYTE COUNT							
FIRST BYTE							
STS							
SMD							
STATION ADDRESS							
TRANSMIT BUFFER START							
TRANSMIT BUFFER LENGTH							
RECEIVE BUFFER START							
RECEIVE BUFFER LENGTH							
INTERRUPT PRIORITY							
INTERRUPT ENABLE							
TIMER/COUNTER MODE							
TIMER/COUNTER MODE							
PROCESSOR STATUS WORD							

Figure 7.1. Format of the 8044 Configure Block

7.2 Transmitting a Message with the 8044

A message is a block of data which represents a text file or a set of instructions for a remote node or an application program which resides on the 8044 program memory. A message can be a frame (packet) by itself or can be comprised of multiple frames. An SDLC frame is the smallest block of data that the 8044 transmits. The 8044 can receive commands from the 80186 to transmit and receive messages. The 8044 on-chip CPU can be programmed to divide messages into frames if necessary. Maximum frame size is limited by the transmit or receive buffer.

To transmit a message, the 80186 prepares a transmit data block in memory as shown in Figure 7.2. Its first byte specifies the length of the rest of the block. The next two bytes specify the destination address of the node the message is being sent to and the control byte of the message. The 80186 programs the DMA controller with the start address of the block, length of the block and other control information and then issues the Transmit command to the 8044.

Upon receiving the command, the 8044 fetches the first byte of the block using DMA to determine the length of the rest of the block. It then fetches the destination address and the control byte using DMA.

The 8044 fetches the rest of the message into the on-chip transmit buffer. The size and location of the transmit buffer in the on-chip RAM is configured with the Configure command. The 8044 CPU then enables the Serial Interface Unit (SIU) to transmit the data as an SDLC frame. The SIU sends out the opening flag, the station address, the SDLC control byte, and the contents of transmit buffer. It then transmits the calculated CRC bytes and the closing flag. The 8044 CPU and the SIU operate concurrently. The CPU can fetch bytes from system memory or execute a command such as TRANSMIT-DISABLE while the SIU is active.

Upon completion of transmission, the SIU updates the internal registers and interrupts the 8044 CPU. The 8044 then updates the status and interrupts the 80186. Note that baud rate generation, zero bit insertion, NRZI encoding, and CRC calculation are automatically done by the SIU.

7.3 Receiving a Message with the 8044

To receive a message, the 80186 allocates a block of memory to store the message. It sets the DMA channel and sends the Receive command to the 8044.

Upon reception of the command, the 8044 enables its serial channel. The 8044 receives and passes to memory all frames whose address matches the individual or broadcast address and passes the CRC test.

The SIU performs NRZI decoding and zero bit deletion, then stores the information field of the received frame in the on-chip receive buffer. At the end of reception, the CPU requests the transfer of data bytes to 80186 memory using DMA. After transferring all the bytes, the 8044 transfers the data length, source address, and control byte of the received frame to the memory (see Figure 7.3). Upon completion of the transfers, the 8044 updates the status register and raises the interrupt signal to inform the 80186.

If the SIU is not ready when the first byte of the frame arrives, then the whole frame is ignored. Disabling reception after the first byte was passed to memory causes the rest of the frame to be ignored and an interrupt with Receive-Aborted event to be issued.



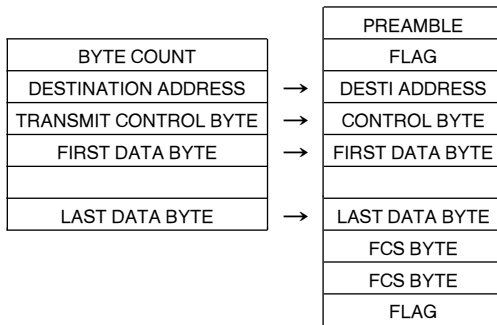


Figure 7.2. The 8044 Transmit Frame Structure and Location of Data Element in System Memory

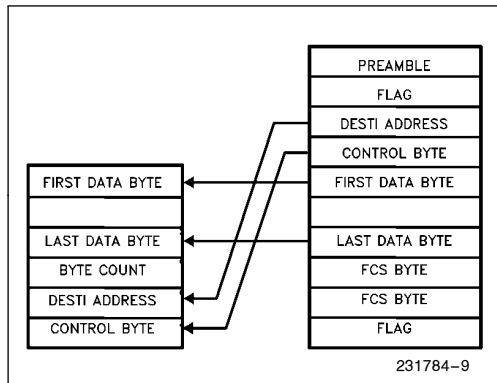


Figure 7.3. The 8044 Receive Frame Structure and Location of Received Data Element in System Memory

7.4 Dumping the 8044 Registers

Upon reception of the Dump command, the 8044 transfers the contents of its internal registers to the system memory (See Figure 7.4).

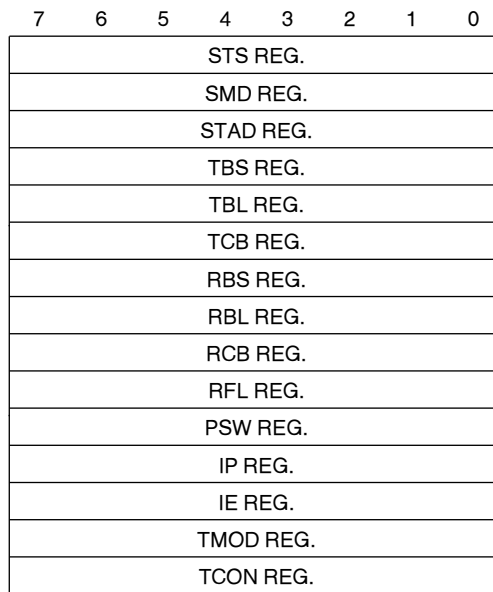


Figure 7.4. Format of the 8044 Dumped Registers

7.5 Aborting an Operation

To abort a DMA operation, the 80186 sends an Abort command to the Command/Data latch and interrupts the 8044. During a DMA operation, the 8044 puts the external interrupt to high priority; therefore, the Abort interrupt will suspend the execution of the operation in progress and update the status register with the Execution-Aborted event. It then returns the 8044 program counter to a location before the aborted operation started. The Abort software procedure given in Appendix A gives the details of the execution of the ABORT command.

7.6 Disabling the Transmission or Reception

Transmission of a frame is aborted if the 80186 sends a TRANSMIT-DISABLE command to the 8044. The command causes the 8044 to clear the Transmit Buffer



Full (TBF) bit. During transmission, if the TBF bit is cleared, the SIU will discontinue the transmission and interrupt the 8044 CPU.

The RECEIVE-DISABLE command causes the 8044 to clear the Receive Buffer Empty (RBE) bit. The SIU aborts the reception, if the RBE bit is cleared by the CPU.

When transmission or reception of a frame is discontinued, the SIU interrupts the 8044 CPU. The CPU then updates the status and interrupts the 80186.

7.7 Handling Interrupts

When the 80186 sends a command, it sets the 8044 external interrupt flag. The 8044 services the interrupt at its own convenience. In the interrupt service routine the 8044 executes the appropriate instructions for a given command. During execution of a command the 8044 ignores any command, except ABORT, sent by the 80186 (see section 5.1.2). This is accomplished by clearing the interrupt flag before the 8044 returns from the interrupt service routine. During DMA operations the 8044 sets the external interrupt to high priority. An interrupt with high priority can suspend execution of an interrupt service routine with low priority. The ABORT command given by the 80186 will interrupt the execution of the DMA transfer in progress. Upon completion of ABORT, execution of the last operation will not be resumed (see Appendix A). Note that any other command given during the DMA operation will also abort the operation in progress and should be avoided.

8.0 8044 IN EXPANDED OPERATION

To increase the number of information bytes in a frame, the 8044 can be operated in Expanded mode. In Expanded operation the system memory can be used as the transmit and receive buffer instead of the 8044 internal RAM. AP-283, "Flexibility in Frame Size Operation with the 8044", describes Expanded operation in detail.

8.1 Transmitting a Message in Expanded Operation

In Expanded operation the 8044 transmits the frame while it is fetching the data from the system memory using DMA. An internal transmit buffer is not necessary. The system memory can be used as the transmit buffer by the 8044.

Upon receiving the Transmit command, the 8044 enables the SIU and fetches the first data byte from the Command/Data register. The SIU transmits the opening flag, station address, and the control byte if the frame format includes these fields. It then transmits the

fetches the data. The 8044 CPU fetches the next byte while the previously fetched byte is being transmitted by the SIU. The CPU fetches the remaining bytes using DMA, then the SIU transmits them simultaneously until the end of message is reached. The SIU then transmits the FCS bytes, the closing flag and interrupts the 8044 CPU. The 8044 updates the status with the Transmit-Done event and interrupts the 80186. If the DMA does not keep up with transmission, the transmission is an underrun.

8.2 Receiving a Message in Expanded Operation

In Expanded operation the DMA controller transfers data to the system memory while the 8044 SIU is receiving them.

To receive a message, the 80186 allocates a block of memory for storing the message. It sets the DMA channel and sends the Receive command to the 8044.

Upon reception of the command, the 8044 enables its serial channel and waits for a frame. The SIU performs flag detection, address filtering, zero bit deletion, NRZI decoding, and CRC checking as it does in Normal operation.

After the SIU receives the first byte of the frame, the 8044 CPU requests the transfer of the byte to memory using DMA. The 80186 DMA moves the information byte into the system memory while the SIU is receiving the next byte. The next byte is transferred to the memory after the SIU receives it. When the entire frame is received, the SIU checks the received Frame Check Sequence bytes. If there is no CRC error, the SIU updates the 8044 registers and interrupts the 8044 CPU. The CPU updates the status and interrupts the 80186.

9.0 CONCLUSION

This application note describes an efficient way to interface the 80186 and the 80188 microprocessors to the Intel 8-bit microcontrollers like the 80C51, 8052, and 8044. To illustrate this point the 80186 microprocessor interface to the 8044 microcontroller based serial communication chip was described. The hardware interface given here is very general and can interface the 8-bit microcontrollers to a variety of Intel microprocessors and DMA controllers. The microcontrollers with this interface hardware have the same benefits as both the Intel UPI-41/42 family and data communication peripheral chips such as the 82588 and the 82568 LAN controllers. Like the Intel UPI chips, they can be easily interfaced to microprocessors, and like the data communication peripherals, they execute high level commands. A similar approach can be used to interface Intel microprocessors to the 16-bit 8096 microcontroller.

APPENDIX A SOFTWARE

The software modules shown here implement the execution of commands and status explained in sections 5 and 7. The 80186 software provides procedures to send commands and read status. The 8044 software decodes and executes the commands, updates the status, and interrupts the 80186. The procedures given here are called by higher level software drivers. For example, an 80186 application program may use the Transmit command to send a block of data to an application program that resides in the 8044 ROM or in another remote node. The application programs and the drivers that perform the communication tasks run asynchronously since all communication tasks are interrupt-driven.

Figure A-1 shows how to assign the ports and control registers for an 80186-based system. The software is written for an Intel iSBC 186/51 computer board. The 8044 hardware is connected to the computer board iSBX connector.

Figure A-2 shows the 80186 command procedures. These procedures are used by the data link driver.

Figure A-3 shows how the DMA controller is loaded and initialized for data and parameter transfer from the 80186 memory to the 8044. This procedure is used by the TRANSMIT and CONFIGURE commands.

Figure A-4 shows how the DMA controller is loaded and initialized for data and parameter transfer from the 8044 to the 80186 memory. This procedure is used by the RECEIVE and DUMP commands.

Figure A-5 shows an interrupt service routine which handles interrupts resulting from various events. Note that this routine is not complete. The user should write the software to respond to events.

Figure A-6 shows an example of the 80186 software. It shows how to start various operations. This is not a data link driver, but it gives the procedures needed to write a complete driver.

Figure A-7 shows how to initialize the 8044. The user application program should be inserted here.

Figures A-8 through A-13 show the 8044 external interrupt service routine. In this routine a command received from the 80186 is decoded, and one of the command procedures shown in Figures A-9 through A-13 is executed.

Figure A-14 shows the serial channel (SIU) interrupt service routine. Note that execution of TRANSMIT, RECEIVE, and TRANSMIT-DISABLE commands are completed in this routine.

```

NAME COM_DRIVER

; ** 80186 SOFTWARE FOR THE 80186/MICROCONTROLLER INTERFACE

; * 8044 BOARD CONNECTED TO THE SBX1 OF THE SBC 186/51 BOARD.
; * SBX1 INTO TIED TO 80130 IR[0-7]. CONNECT JUMPER 30 TO 46.
; * 80186 DMA CHANNEL 1 USED. CONNECT JUMPER 202 TO 203.

TRUE      EQU  OFFFFH
FALSE     EQU  0H

; 8044 REGISTERS

CMD_44    EQU  080H      ; ADDRESS OF THE COMMAND REGISTER
ST_44     EQU  080H      ; ADDRESS OF THE STATUS REGISTER
DATA_44   EQU  0D4H      ; ADDRESS OF THE DATA REGISTER

; EVENTS

CON_DONE  EQU  01H      ; CONFIGURE_DONE
TRA_DONE  EQU  02H      ; TRANSMIT_DONE
DUM_DONE  EQU  03H      ; DUMP_DONE
REC_DONE  EQU  04H      ; RECEIVE_DONE
REC_DISA  EQU  05H      ; RECEPTION_DISABLE
TRA_DISA  EQU  06H      ; TRANSMISSION_DISABLE
ABO_DONE  EQU  07H      ; EXECUTION_ABORTED
; COMMANDS (INTA=1)

ABO_CMD   EQU  080H      ; ABORT
REC_DIS_CMD EQU  081H      ; RECEIVE_DISABLE
XMIT_DIS_CMD EQU  082H      ; TRANSMIT_DISABLE
REC_CMD   EQU  083H      ; RECEIVE
TRA_CMD   EQU  084H      ; TRANSMIT
DUM_CMD   EQU  085H      ; DUMP
CON_CMD   EQU  086H      ; CONFIGURE
NOP_CMD   EQU  087H      ; NOP

; 80186 DMA CHANNEL 1 REGISTERS

SL_DMA1   EQU  OFFD0H      ; SOURCE ADDRESS (LO WORD)
SH_DMA1   EQU  OFFD2H      ; SOURCE ADDRESS (HI WORD)
DL_DMA1   EQU  OFFD4H      ; DESTINATION ADDRESS (LO WORD)
DH_DMA1   EQU  OFFD6H      ; DESTINATION ADDRESS (HI WORD)
CNT_DMA1  EQU  OFFD8H      ; TRANSFER COUNT ADDRESS
CTL_DMA1  EQU  OFFDAH      ; CONTROL ADDRESS

; 80186 INTERRUPT CONTROLLER REGISTERS

CTL0_INTR EQU  OFF38H      ; INT 0 CONTROL ADDRESS
CTL1_INTR EQU  OFF3AH      ; INT 1 CONTROL REGISTER
MASK_INTR EQU  OFF28H      ; INT MASK REGISTER
EOI_INTR  EQU  OFF22H      ; INT EOI REGISTER
NSPEC_BIT EQU  08000H      ; NON-SPECIFIC EOI

; 80130 INTERRUPT CONTROLLER REGISTERS

EOI_SINTR EQU  0E0H      ; INT EOI REGISTER
MASK_SINTR EQU  0E2H      ; MASK REGISTER

RD_IRR    EQU  010H      ; COMMAND TO 80130 TO READ IRR REG
RD_ISR    EQU  011H      ; COMMAND TO 80130 TO READ ISR REG

IV_BASE   EQU  20H      ; BASE OF 80130 INT CONTROLLER VECTOR
    
```

231784-10

231784-11

Figure A-1. Port and Register Definitions for 80186 System

```

;*****
; INTERRUPT TABLE
INTERRUPTS      SEGMENT AT 0
                ORG    (IV_BASE+1)*4H
IV_INTRO       LABEL  DWORD          ; IRI VECTOR
INTERRUPTS      ENDS
;*****

STACK          SEGMENT  STACK  'STACK'
THE_STACK     DW      200H  DUP(?)
TOS           LABEL  WORD
STACK         ENDS
;*****

DATA          SEGMENT  PUBLIC  'DATA'
REC_BUFFER    DB      1024  DUP(?)
CON_BUFFER    DB      08H,01H,00H,0D0H,55H,20H,05H,30H,05H
DUM_BUFFER    DB      0FH  DUP(?)
TRA_BUFFER    DB      07H,55H,11H,01H,02H,03H,04H,05H
CMND_FLAG     DW      FALSE
DATA          ENDS
    
```

231784-12

Figure A-1. Port and Register Definitions for 80186 System (Continued)

```

;*****
CODE          SEGMENT  PUBLIC  'CODE'
ASSUME       CS:CODE,
            & DS:DATA,
            & ES:NOTHING,
            & SS:STACK
;*****

RCV_COMMAND  PROC  FAR
    PUSH  BP
    MOV   BP,SP
    LES  SI,DWORD PTR [BP+6]      ; LOAD BUFFER POINTER
    MOV  AX,WORD PTR[BP+10]      ; LOAD BUFFER SIZE
    MOV  AH,0H
    CALL REC_DMA                 ; CALL REC-DMA
    MOV  AL,REC_CMD              ; LOAD RECEIVE COMMAND
    OUT  CMD_44,AL               ; SEND TO COMMAND/DATA REG
    POP  BP
    RET
RCV_COMMAND  ENDP
;*****

XMIT_COMMAND PROC  FAR
    PUSH  BP
    MOV   BP,SP
    LES  SI,DWORD PTR [BP+6]      ; LOAD BUFFER POINTER
    MOV  AX,WORD PTR[BP+10]      ; LOAD BUFFER SIZE
    MOV  AH,0H
    CALL TRA_DMA                 ; CALL TRA-DMA
    MOV  AL,TRA_CMD              ; LOAD TRANSMIT COMMAND
    OUT  CMD_44,AL               ; SEND TO COMMAND/DATA REG
    POP  BP
    RET
XMIT_COMMAND ENDP
    
```

231784-13

Figure A-2. Setup and Execution of Commands

```

;*****
CONF_COMMAND  PROC  FAR

    PUSH  BP
    MOV   BP,SP
    LES  SI,DWORD PTR[BP+6]      ; LOAD BUFFER POINTER
    MOV  AX,WORD PTR[BP+10]     ; LOAD BUFFER SIZE
    MOV  AH,0H
    CALL TRA_DMA                 ; CALL TRA-DMA
    MOV  AL,CON_CMD             ; LOAD CONFIGURE COMMAND
    OUT  CMD_44,AL              ; SEND TO COMMAND/DATA REG
    POP  BP
    RET

CONF_COMMAND  ENDP

;*****

DUMP_COMMAND  PROC  FAR

    PUSH  BP
    MOV   BP,SP
    LES  SI,DWORD PTR[BP+6]     ; LOAD BUFFER POINTER
    MOV  AX,WORD PTR[BP+10]     ; LOAD BUFFER SIZE
    MOV  AH,0H
    CALL REC_DMA                 ; CALL REC-DMA
    MOV  AL,DUM_CMD             ; LOAD DUMP COMMAND
    OUT  CMD_44,AL              ; SEND TO COMMAND/DATA REG
    POP  BP
    RET

DUMP_COMMAND  ENDP
;***** 231784-14

XMIT_DIS_COMMAND  PROC  FAR

    MOV  AL,XMIT_DIS_CMD        ; LOAD XMIT-DIS COMMAND
    OUT  CMD_44,AL              ; SEND TO COMMAND/DATA REG
    RET

XMIT_DIS_COMMAND  ENDP

;*****

REC_DIS_COMMAND  PROC  FAR

    MOV  AL,REC_DIS_CMD        ; LOAD REC-DIS COMMAND
    OUT  CMD_44,AL              ; SEND TO COMMAND/DATA REG
    RET

REC_DIS_COMMAND  ENDP

;*****

ABOR_COMMAND  PROC  FAR

    MOV  AL,ABO_CMD             ; LOAD ABORT COMMAND
    OUT  CMD_44,AL              ; SEND TO COMMAND/DATA REG
    RET

ABOR_COMMAND  ENDP

;*****

NOP_COMMAND  PROC  FAR

    MOV  AL,NOP_CMD             ; LOAD NOP COMMAND
    OUT  CMD_44,AL              ; SEND TO COMMAND/DATA REG
    RET

NOP_COMMAND  ENDP
;***** 231784-15

```

Figure A-2. Setup and Execution of Commands (Continued)


```

;*****
; ** RECEIVE DMA
; ARGS AX BUFFER SIZE
; ES:SI BUFFER POINTER
;
REC_DMA PROC NEAR
    MOV DX,CNT_DMA1 ; LOAD ADD OF TRANSFER COUNT REG
    OUT DX,AX ; PROGRAM TRANSFER COUNT REGISTER

    XOR BX,BX ; CLEAR BX
    MOV AX,ES ; LOAD SEG ADDRESS OF BUFFER
    SHL AX,1 ; CALCULATE LINEAR ADDRESS OF THE BUFFER
    RCL BX,1
    SHL AX,1
    RCL BX,1
    SHL AX,1
    RCL BX,1
    SHL AX,1
    RCL BX,1
    ADD AX,SI ; ADD THE OFFSET TO BASE
    ADC BX,0
    MOV DX,DL_DMA1 ; LOAD ADDRESS OF DEST POINTER (LO WORD)
    OUT DX,AX ; PROGRAM DEST POINTER REGISTER (LO WORD)
    MOV AX,BX
    MOV DX,DH_DMA1 ; LOAD ADDRESS OF DEST POINTER (HI WORD)
    OUT DX,AX ; PROGRAM DEST POINTER REGISTER (HI WORD)

    MOV AX,DATA_44 ; LOAD ADDRESS OF DATA REGISTER
    MOV DX,SL_DMA1 ; LOAD ADDRESS OF SOURCE POINTER
    OUT DX,AX ; PROGRAM SOURCE POINTER REGISTER (LO WORD)

    XOR AX,AX ; CLEAR AX
    MOV DX,SH_DMA1 ; LOAD ADDRESS OF SOURCE POINTER (HI WORD)
    OUT DX,AX ; PROGRAM SOURCE POINTER REGISTER (HI WORD)

    MOV DX,CTL_DMA1 ; LOAD ADDRESS OF CONTROL REGISTER
    MOV AX,1010001010100110B ; LOAD THE CONTROL WORD
    OUT DX,AX ; PROGRAM THE CONTRL REGISTER
    RET
REC_DMA ENDP
    
```

231784-16

Figure A-3. Loading and Starting the 80186 DMA Controller

```

;*****
; ** TRANSMIT DMA
; ARGS AX BUFFER SIZE
; ES:SI BUFFER POINTER
;
TRA_DMA PROC NEAR
    INC AX
    MOV DX,CNT_DMA1 ; LOAD ADD OF TRANSFER COUNT REG
    OUT DX,AX ; PROGRAM TRANSFER COUNT REGISTER

    XOR BX,BX ; CLEAR BX
    MOV AX,ES ; LOAD SEG ADDRESS OF BUFFER
    SHL AX,1 ; CALCULATE LINEAR ADDRESS OF THE BUFFER
    RCL BX,1
    SHL AX,1
    RCL BX,1
    SHL AX,1
    RCL BX,1
    SHL AX,1
    RCL BX,1
    ADD AX,SI ; ADD THE OFFSET TO BASE
    ADC BX,0
    MOV DX,SL_DMA1 ; LOAD ADDRESS OF SOURCE POINTER (LO WORD)
    OUT DX,AX ; PROGRAM SOURCE POINTER REGISTER (LO WORD)
    MOV AX,BX
    MOV DX,SH_DMA1 ; LOAD ADDRESS OF SOURCE POINTER (HI WORD)
    OUT DX,AX ; PROGRAM SOURCE POINTER REGISTER (HI WORD)

    MOV AX,DATA_44 ; LOAD ADDRESS OF DATA REGISTER
    MOV DX,DL_DMA1 ; LOAD ADDRESS OF DEST POINTER
    OUT DX,AX ; PROGRAM DEST POINTER REGISTER (LO WORD)

    XOR AX,AX ; CLEAR AX
    MOV DX,DH_DMA1 ; LOAD ADDRESS OF DEST POINTER (HI WORD)
    OUT DX,AX ; PROGRAM DEST POINTER REGISTER (HI WORD)

    MOV DX,CTL_DMA1 ; LOAD ADDRESS OF CONTROL REGISTER
    MOV AX,0001011010100110B ; LOAD THE CONTROL WORD
    OUT DX,AX ; PROGRAM THE CONTRL REGISTER
    RET
TRA_DMA ENDP
    
```

231784-17

Figure A-4. Loading and Starting the 80186 DMA Controller

```

;*****
; 80186 INTERRUPT ROUTINE
INT_186:
    PUSH AX
    PUSH DX
    MOV AX,NSPEC_BIT           ; SEND NSPEC END OF INT
    MOV DX,EI_INTR
    OUT DX,AX

    MOV AL,0110001B
    OUT EI_SINTR,AL

    IN AL,ST_44               ; READ THE STATUS
    AND AX,0FFH

; DECODE STATUS AND TAKE APPROPRIATE ACTION

    MOV DX,CTL_DMA1          ; DISABLE DMA
    IN AX,DX
    OR AX,0100B
    AND AX,NOT 010B
    OUT DX,AX

    MOV CMND_FLAG,TRUE

    POP DX
    POP AX
    IRET

```

231784-18

Figure A-5. Interrupt Service Routine

```

;*****
BEGIN:
    CLI
    CLD

; SET ALL REGISTERS SMALL MODEL

    MOV SP,DATA
    MOV DS,SP
    MOV ES,SP
    MOV SP,STACK
    MOV SS,SP
    MOV SP,OFFSET TOS

; SETUP INTERRUPT VECTORS

    PUSH ES
    XOR AX,AX
    MOV ES,AX
    MOV WORD PTR ES:IV_INTR0 +0, OFFSET INT_186
    MOV WORD PTR ES:IV_INTR0 +2, CS
    POP ES

SETUP 80130 INTERRUPT CONTROLLER

    MOV AL,00010011B          ; ICW1
    OUT EI_SINTR,AL
    MUL AL

    MOV AL,IV_BASE           ; ICW2
    OUT MASK_SINTR,AL
    MUL AL

    MOV AL,00000000B         ; ICW4
    OUT MASK_SINTR,AL
    MUL AL

    MOV AL,0FCH              ;MASK
    OUT MASK_SINTR,AL

```

231784-19

Figure A-6. Example of Executing Commands

```

; SETUP 80186 INTERRUPT CONTROLLER
      MOV     AX,000000000100000B
      MOV     DX,CTLO_INTR
      OUT     DX,AX

      MOV     DX,CTL1_INTR
      IN      AX,DX
      OR      AX,000000000101000B
      OUT     DX,AX

      MOV     AX,000EDH           ; MASK ALL BUT IO
      MOV     DX,MASK_INTR
      OUT     DX,AX
      STI                    ;ENABLE INTERRUPTS

;*** SEND CONFIGURE COMMAND
      PUSH    WORD PTR CON_BUFFER ; PUSH BUFFER SIZE
      PUSH    DS                ; PUSH BUFFER SEGMENT REGISTER
      PUSH    OFFSET CON_BUFFER ; PUSH OFFSET OF BUFFER
      CALL    CONF_COMMAND      ; CALL CONFIGURE
      ADD     SP,3*2

; WAIT FOR END OF COMMAND
WAIT1:
      CMP     CMND_FLAG,TRUE
      JNE     WAIT1
      MOV     CMND_FLAG,FALSE

;*** SEND DUMP COMMAND
      PUSH    WORD PTR DUM_BUFFER ; PUSH BUFFER SIZE
      PUSH    DS                ; PUSH BUFFER SEGMENT REGISTER
      PUSH    OFFSET DUM_BUFFER ; PUSH OFFSET OF BUFFER
      CALL    DUMP_COMMAND      ; CALL CONFIGURE
      ADD     SP,3*2

WAIT2:
      CMP     CMND_FLAG,TRUE
      JNE     WAIT2
      MOV     CMND_FLAG,FALSE

;*** SEND TRANSMIT COMMAND
      PUSH    WORD PTR TRA_BUFFER ; PUSH BUFFER SIZE
      PUSH    DS                ; PUSH BUFFER SEGMENT REGISTER
      PUSH    OFFSET TRA_BUFFER ; PUSH OFFSET OF BUFFER
      CALL    XMIT_COMMAND      ; CALL COMMAND
      ADD     SP,3*2

WAIT3:
      CMP     CMND_FLAG,TRUE
      JNE     WAIT3
      MOV     CMND_FLAG,FALSE

;*** SEND RECEIVE COMMAND
      PUSH    WORD PTR REC_BUFFER ; PUSH BUFFER SIZE
      PUSH    DS                ; PUSH BUFFER SEGMENT REGISTER
      PUSH    OFFSET REC_BUFFER ; PUSH OFFSET OF BUFFER
      CALL    RECV_COMMAND      ; CALL COMMAND
      ADD     SP,3*2

WAIT4:
      CMP     CMND_FLAG,TRUE
      JNE     WAIT4
      MOV     CMND_FLAG,FALSE

CODE   ENDS
      END     BEGIN

```

231784-21

Figure A-6. Example of Executing Commands (Continued)

```

SDEBUG NOMOD51
$INCLUDE (REG44.PDF)

; THE 8044 SOFTWARE DRIVER FOR THE 80186/8044 INTERFACE.

      ORG 00H           ; LOCATIONS 00 THRU 26H ARE USED
      SJMP INIT        ; BY INTERRUPT SERVICE ROUTINES.
      ORG 03H           ; VECTOR ADDRESS FOR EXT INTO.
      JMP EINT0
      ORG 23H           ; VECTOR ADDRESS FOR SERIAL INT
      JMP SIINT

;***** INITIALIZATION *****

      ORG 26H
INIT:  MOV TCON,#00000001B ; EXT INTO: EDGE TRIGGER
      MOV IE,#00010001B  ; SI=EX0=1
      CLR P1.1           ; CLEAR DRQ STATUS BIT
      SETB EA            ; ENABLE INTERRUPTS
DOT:   SJMP DOT          ; WAIT FOR AN INTERRUPT

```

231784-22

Figure A-7. Initialization Routine

```

;*****EXTERNAL INTERRUPT 0 *****
EINT0: CLR P1.5         ; CLEAR THE E BIT
      MOV DPTR,#100H   ; LOAD DATA POINTER WITH A DUMMY NUMBER
      MOVX A,@DPTR    ; READ THE COMMAND BYTE.
      ANL A,#00001111B ; KEEP THE OPERATION FIELD
      MOV R2,A         ; SAVE COMMAND

; DECODE COMMAND AND JUMP TO THE APPROPRIATE ROUTINE
; COMMAND OPERATION (BITS0-3)
;
;   ABORT          00H
;   REC-DISABLE   01H
;   TRA-DISABLE   02H
;   RECEIVE       03H
;   TRANSMIT      04H
;   DUMP          05H
;   CONFIGURE     06H
;   NOP           07H

      JNB PX0,J1      ; IF INTO IS SET TO PRIORITY 1,
      JMP CABO        ; THEN DMA OPERATION WAS IN PROGRESS.
                      ; EXECUTE ABORT REGARDLESS OF THE
                      ; COMMAND ISSUED.
J1:   CJNE A,#00H,J2  ; EXECUTE ABORT
      JMP CABO        ; THIS LINE WILL BE EXECUTED IF ABORT WAS
                      ; ISSUED WHEN THE 8044 IS NOT EXECUTING
                      ; ANY COMMANDS.
J2:   CJNE A,#01H,J3  ; EXECUTE RECEIVE-DISCONNECT
      JMP CRDIS
J3:   CJNE A,#02H,J4  ; EXECUTE TRANSMIT-DISCONNECT
      JMP CTDIS
J4:   CJNE A,#03H,J5  ; EXECUTE RECEIVE
      JMP CREC
J5:   CJNE A,#04H,J6  ; EXECUTE TRANSMIT
      JMP CTRA
J6:   CJNE A,#05H,J7  ; EXECUTE DUMP
      JMP CDUMP
J7:   CJNE A,#06H,J8  ; EXECUTE CONFIGURE
      JMP CCON
J8:   CJNE A,#07H,J9  ; EXECUTE NOP
      JMP CNOP
J9:   RETI            ; RETURN. OPERATION NOT RECOGNIZED.

```

231784-23

Figure A-8. External Interrupt Service Routine

```

; ** NOP COMMAND
CNOP: CLR IEO ; IGNORE PENDING EXT INTO (IF ANY).
; ANY INTERRUPT (COMMAD) DURING
; EXECUTION OF AN OPERATION IS IGNORED
; RETURN
      RETI

; ** ABORT COMMAND
CABO: JNB PX0,CABOJ1 ; WAS DMA IN PROGRESS?
      CLR PX0 ; YES. EXT INTO: PRIORITY 0
      CLR P1.1 ; CLEAR DMA REQUEST

      SETB P1.2 ; UPDATE STATUS WITH
      SETB P1.3 ; ABORT-DONE EVENT
      SETB P1.4 ; (STATUS=DDH; E=0)

      CLR IEO ; IGNORE PENDING EXT INTO (IF ANY).
      CLR P1.0
      SETB P1.0 ; SET INT BIT AND INTERRUPT 80186
      JB P3.2,$ ; WAIT TILL INTERRUPT IS ACKNOWLEDGED
; EXECUTE THE NEXT "RETI" TWICE
      POP ACC ; POP OUT THE OLD HI BYTE PC
      POP ACC ; POP OUT THE OLD LOW BYTE PC
      MOV B,#HIGH($+10) ; HI BYTE ADDRESS OF CABOJ2
      MOV ACC,#LOW($+7) ; LOW BYTE ADDRESS OF CABOJ2
      PUSH ACC ; PUSH THE ADDRESS OF THE NEXT
      PUSH B ; "RETI" INSTRUCTION INTO STACK
CABOJ2: RETI ; RETURN

CABOJ1: NOP ; DMA WAS NOT IN PROGRESS
      SETB P1.5 ; SET THE E BIT

      SETB P1.2 ; UPDATE STATUS WITH
      SETB P1.3 ; ABORT-DONE EVENT
      SETB P1.4 ; (STATUS=FDH; E=1)

      CLR IEO ; IGNORE PENDING EXT INTO (IF ANY).
      CLR P1.0
      SETB P1.0 ; SET INT BIT AND INTERRUPT 80186
      JB P3.2,$ ; WAIT TILL INTERRUPT IS ACKNOWLEDGED
      RETI
    
```

231784-24

Figure A-9. Execution of NOP and ABORT Commands

```

; ** CONFIGURE COMMAD
CCON: MOV DPTR,#100H ; IGNORE PENDING EXT INTO (IF ANY)
      CLR IEO ; EXT INTO: PRIORITY 1
      SETB PX0 ; PX0 IS SET TO ACCEPT ABORT
; DURING DMA OPERATION.
      SETB P1.1 ; ENABLE DMA REQUEST
      JB P3.3,$ ; WAIT FOR DMA ACK.
      MOVX A,@DPTR ; READ FROM COMMAN/DATA REGISTER
      MOV RO,A ; LOAD BYTE COUNT
      DEC RO ; DECREMENT BYTE COUNT
      JB P3.3,$ ; WAIT FOR DMA ACK.
      MOVX A,@DPTR ; READ FROM COMMAND/DATA REGISTER
      MOV R1,A ; LOAD FIRST-BYTE
      JB P3.3,$ ; WAIT FOR DMA ACK.
      MOVX A,@DPTR ; READ FROM COMMAND/DATA REGISTER
      CJNE R1,#01H,CCONJ1 ; CHECK THE FIRST-BYTE
      MOV STS,A ; UPDATE THE STS REGISTER
      INC R1 ; INC. POINTER TO THE CONF. BLOCK
      DJNZ RO,CCONF4 ; CHECK THE BYTE COUNT
      JMP CCONT1
CCONF4: JB P3.3,CCONF4
      MOVX A,@DPTR
CCONJ1: CJNE R1,#02H,CCONJ2
      MOV SMD,A
      INC R1
      DJNZ RO,CCONF5
      JMP CCONT1
CCONF5: JB P3.3,CCONF5
      MOVX A,@DPTR
CCONJ2: CJNE R1,#03H,CCONJ3
      MOV STAD,A
      INC R1
      DJNZ RO,CCONF6
      JMP CCONT1
CCONF6: JB P3.3,CCONF6
      MOVX A,@DPTR
CCONJ3: CJNE R1,#04H,CCONJ4
    
```

231784-25

Figure A-10. Execution of CONFIGURE Command

```

        MOV     TBS,A
        INC     R1
        DJNZ   R0,CCONF7
        JMP     CCONT1
CCONF7: JB     P3.3,CCONF7
        MOVX   A,@DPTR
CCONJ4: CJNE  R1,#05H,CCONJ5
        MOV     TBL,A
        INC     R1
        DJNZ   R0,CCONF8
        JMP     CCONT1
CCONF8: JB     P3.3,CCONF8
        MOVX   A,@DPTR
CCONJ5: CJNE  R1,#06H,CCONJ6
        MOV     RBS,A
        INC     R1
        DJNZ   R0,CCONF9
        JMP     CCONT1
CCONF9: JB     P3.3,CCONF9
        MOVX   A,@DPTR
CCONJ6: CJNE  R1,#07H,CCONJ7
        MOV     RBL,A
        INC     R1
        DJNZ   R0,CCONFA
        JMP     CCONT1
CCONFA: JB     P3.3,CCONFA
        MOVX   A,@DPTR
CCONJ7: CJNE  R1,#08H,CCONJ8
        MOV     IP,A
        INC     R1
        DJNZ   R0,CCONFB
        JMP     CCONT1
CCONFB: JB     P3.3,CCONFB
        MOVX   A,@DPTR
CCONJ8: CJNE  R1,#09H,CCONJ9
        MOV     IE,A
        INC     R1
        DJNZ   R0,CCONFC
        JMP     CCONT1
CCONFC: JB     P3.3,CCONFC
        MOVX   A,@DPTR
CCONJ9: CJNE  R1,#0AH,CCONJA
        MOV     TMOD,A
        INC     R1
        DJNZ   R0,CCONFD
        JMP     CCONT1
CCONFD: JB     P3.3,CCONFD
        MOVX   A,@DPTR
CCONJA: CJNE  R1,#0BH,CCONJB
        MOV     TCON,A
        INC     R1
        DJNZ   R0,CCONFE
        JMP     CCONT1
CCONFE: JB     P3.3,CCONFE
        MOVX   A,@DPTR
CCONJB: CJNE  R1,#0CH,ERROR1
        MOV     PSW,A
        INC     R1
        DJNZ   R0,ERROR1
        JMP     CCONT1

ERROR1: NOP
        SETB   P1.5          ; ILLEGAL BYTE COUNT
                          ; SET THE E STATUS BIT

CCONT1: NOP
        CLR    P1.1          ; CLEAR DMA REQUEST
        CLR    PX0          ; EXT INTO: PRIORITY 0

        SETB   P1.2          ; UPDATE STATUS WITH
        CLR    P1.3          ; CONFIGURE-DONE EVENT
        CLR    P1.4          ; (STATUS=C5H IF E=0)

        CLR    IE0          ; IGNORE PENDING EXT INTO (IF ANY)
        CLR    P1.0
        SETB   P1.0          ; INTERRUPT THE 80186
        JB     P3.2,$        ; WAIT TILL INTERRUPT IS ACKNOWLEDGED
        RETI                ; RETURN

```

231784-26

231784-27

Figure A-10. Execution of CONFIGURE Command (Continued)

```

; ** DUMP COMMAND

CDUMP:  MOV  A,STS           ; LOAD THE FIRST DUMP REG INTO ACC
        MOVX @DPTR,A       ; WRITE TO THE COMMAND/DATA REGISTER
        CLR  IE0           ; IGNORE PENDING EXT INTO (IF ANY)
        SETB PX0          ; INTRERRUPT 0: PRIORITY 1
        SETB P1.1         ; ENABLE DMA REQUEST
        JB   P3.3,$        ; WAIT FOR DMA ACK
        MOV  A,SMD
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,STAD
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,TBS
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,TBL
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,TCB
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,RBS
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,RBL
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,RCE
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,RFL
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,PSW
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,IEP
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,IE
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,TMOD
        MOVX @DPTR,A
        JB   P3.3,$
        MOV  A,TCON
        MOVX @DPTR,A
        JB   P3.3,$
        CLR  P1.1         ; DISABLE DRQ
        CLR  PX0          ; EXTERNAL INTO: PRIORITY 0

        SETB P1.2         ; UPDATE STATUS WITH
        SETB P1.3         ; DUMP-DONE EVENT
        CLR  P1.4         ; (STATUS=CDH)

        CLR  IE0           ; IGNORE PENDING EXT INTO
        CLR  P1.0
        SETB P1.0         ; INTERRUPT THE 80186
        JB   P3.2,$        ; WAIT TILL INTERRUPT IS ACKNOWLEDGED
        RETI              ; RETURN
    
```

231784-28

231784-29

Figure A-11. Execution of DUMP Command

```

; ** RECEIVE COMMAND.
CREC:  JNB  RBE,CRECJ1      ; IS SIU ALREADY IN RECEIVE MODE?
        SETB P1.5          ; YES. SET THE E BIT
CRECJ1: SETB RBE           ; NO. ENABLE RECEPTION
        CLR  RBP           ; CLEAR RECEIVE BUFFER PROTECT BIT
        CLR  IEO           ; IGNORE PENDING EXT INTO (IF ANY)
        RETI              ; RETURN. UPDATE STATUS IN THE
                          ;SIU INTERRUPT ROUTINE.

; ** TRANSMIT COMMAND.
CTRA:  MOV  R1,TBS         ; LOAD TRANSMIT BUFFER START
        CLR  IEO           ; IGNORE PENDING EXT INTO (IF ANY)
        SETB PXO           ; EXT INTO: PRIORITY 1
        SETB P1.1         ; ENABLE DMA REQUEST
        JB   P3.3,$        ; WAIT FOR DMA ACK.
        MOVX A,@DPTR       ; READ FROM COMMAND/DATA REG.
        MOV  RO,A          ; LOAD THE BYTE COUNT
        DEC  A             ; SUBTRACT 2 FROM THE BYTE
        MOV  TBL,A        ; COUNT AND LOAD INTO XMIT
        MOV  TBL,A        ; LOAD BUFFER LENGTH
CTRAJ2: JB   P3.3,CTRAJ2   ; WAIT FOR DMA ACK.
        MOVX A,@DPTR       ; READ FROM COMMAND/DATA REG.
        MOV  STAD,A        ; LOAD DESTINATION ADDRESS
        DEC  RO            ; DECREMENT THE BYTE COUNT
CTRAJ3: JB   P3.3,CTRAJ3   ; WAIT FOR DMA ACK.
        MOVX A,@DPTR       ; READ FROM COMMAND/DATA REG.
        MOV  TCB,A        ; LOAD THE TRANSMIT CONTROL BYTE
        DJNZ RO,CTRAJ4     ; IS THERE ANY INFO. BYTE?
        SJMP CTRAJ5        ; NO.
CTRAJ4: JB   P3.3,CTRAJ4   ; YES. WAIT FOR DMA ACK.
        MOVX A,@DPTR       ; READ FROM COMMAND/DATA REG.
        MOV  @R1,A        ; MOVE DATA TO THE TRANSMIT BUFFER
        INC  R1            ; INC. POINTER TO BUFFER
        DJNZ RO,CTRAJ4     ; LAST BYTE FETCHED INTO THE BUFFER?
        ; NO. FETCH THE NEXT BYTE
CTRAJ5: CLR  P1.1         ; YES. DISABLE DMA REQUEST
        CLR  PXO           ; EXT INTO: PRIORITY 0
        SETB TBF           ; SET TRANSMIT BUFFER FULL
        SETB RTS           ; ENABLE TRANSMISSION
        CLR  IEO           ; IGNORE PENDING EXT INTO (IF ANY)
        RETI              ; RETURN. UPDATE STATUS IN THE
                          ;SIU INTERRUPT ROUTINE

```

231784-30

Figure A-12. Execution of RECEIVE and TRANSMIT Commands

```

; ** TRANSMIT-DISCONNECT COMMAND
CTDIS:  JB   TBF,CTDIJ1    ; IS TRANSMIT BUFFER ALREADY EMPTY?
        SETB P1.5          ; YES, SET THE E BIT
CTDIJ1: CLR  TBF           ; NO. CLEAR TRANSMIT BUFFER
        CLR  IEO           ; IGNORE PENDING EXT INTO (IF ANY)
        RETI              ; RETURN. UPATE STATUS IN THE
                          ;SIU INTERRUPT ROUTINE.

; ** RECEIVE-DISCONNECT COMMAND
CRDIS:  JB   RBE,CRDIJ1    ; IS RECEIVE BUFFER ALREADY EMPTY?
        SETB P1.5          ; YES. SET THE E BIT
CRDIJ1: CLR  RBE           ; NO. CLEAR RECEIVE BUFFER

        SETB P1.2         ; UPDATE STATUS WITH
        CLR  P1.3         ; RECEPTION-DISABLED EVENT
        SETB P1.4         ; (STATUS=D5 IF E=0)

        CLR  IEO           ;
        CLR  P1.0         ;
        SETB P1.0         ; INTERRUPT THE 80186
        JB   P3.2,$        ; WAIT TILL INTERRUPT IS ACKNOWLEDGED
        RETI              ; RETURN

```

231784-31

Figure A-13. Execution of RECEIVE-DISCONNECT and TRANSMIT-DISCONNECT Commands


```

;***** SERIAL CHANNEL (SIU) INTERRUPT *****
SIINT:  CLR  SI                ;
        MOV  A,R2             ; LOAD THE OPERATION FIELD
        CJNE A,#03H,SINTJ1   ; RECEIVE COMMAND PENDING?
        JMP  SIREC           ; YES.
SINTJ1: CJNE A,#02H,SINTJ2   ; TRANSMIT-DISCONNECT PENDING?
        JMP  SITDIS         ; YES.
SINTJ2: JMP  SITRA           ; TRANSMIT COMMAND IS PENDING

; ** TRANSMISSION IS DISABLED
SITDIS: JB  RTS,SINTJ3       ; REQUEST TO SEND ENABLED?
        JNB TBF,SINTJ3       ; YES. TRANSMISSION DISABLED?
        ; YES.
        CLR  P1.2             ; UPDATE STATUS WITH
        SETB P1.3             ; TRANSMISSION-DISABLED EVENT
        SETB P1.4             ; (STATUS=D9H)

        CLR  IE0              ; IGNORE PENDING EXT INTO
        CLR  P1.0             ;
        SETB P1.0             ; INTERRUPT THE 80186
        JB  P3.2,$           ; WAIT TILL INTERRUPT IS ACKNOWLEDGED
        RETI

; ** A FRAME IS TRANSMITTED
SITRA:  JB  RTS,SINTJ3       ; A FRAME TRANSMITTED?
        ; YES.
        CLR  P1.2             ; UPDATE STATUS WITH
        SETB P1.3             ; TRANSMIT-DONE EVENT
        SETB P1.4             ; (STATUS=C9).

        CLR  IE0              ;
        CLR  P1.0             ;
        SETB P1.0             ; INTERRUPT THE 80186
        JB  P3.2,$           ; WAIT TILL INTERRUPT IS ACKNOWLEDGED
        RETI
; ** A FRAME IS RECEIVED
SIREC:  JB  RBE,SINTJ3       ; RECEIVE BUFFER FULL?
        JNB BOV,SINTJ4       ; YES. BUFFER OVERRUN?
        SETB P1.5             ; YES. SET THE E BIT
SINTJ4: MOV  R0,RFL           ; LOAD R0 WITH RECEIVE BYTE COUNT
        MOV  R1,RBS           ; LOAD R1 WITH RECEIVE BUFFER ADDRESS
        CLR  IE0              ; IGNORE PENDING EXT INTO (IF ANY)
        SETB PX0              ; EXT INTO: PRIORITY 1

        MOV  A,@R1            ; MOVE FIRST BYTE INTO ACC.
        MOVX @DPTR,A         ; WRITE TO THE COMMAND/DATA REG
        SETB P1.1             ; ENABLE DMA REQUEST
        INC  R1               ; INC POINTER TO RECEIVE BUFFER
        JB  P3.3,$           ; WAIT FOR DMA ACK.
        DJNZ RO,CINTJ7       ; LAST BYTE MOVED?
        SJMP CINTJ8         ; YES

CINTJ7: MOV  A,@R1            ; LOAD RECEIVED DATA INTO ACC.
        MOVX @DPTR,A         ; WRITE TO THE COMMAND/DATA REG.
        INC  R1               ; INC POINTER TO RECEIVE BUFFER
        JB  P3.3,$           ; WAIT TILL DMA ACK
        DJNZ RO,CINTJ7       ; LAST BYTE MOVED TO COMMAND/DATA REG?
        ; NO. DEPOSIT THE NEXT BYTE

CINTJ8: MOV  A,RFL           ; LOAD BYTE COUNT
        MOVX @DPTR,A         ; WRITE TO THE COMMAND/DATA REG
        JB  P3.3,$           ; WAIT FOR DMA ACK.
        MOV  A,STAD           ; LOAD STATION ADDRESS
        MOVX @DPTR,A         ; WRITE TO THE COMMAND/DATA REG
        JB  P3.3,$           ; WAIT FOR DMA ACK.
        MOV  A,RCB           ; LOAD RECEIVE CONTROL BYTE
        MOVX @DPTR,A         ; WRITE TO THE COMMAND/DATA REG
        JB  P3.3,$           ; WAIT FOR DMA ACK.
        CLR  P1.1             ; CLEAR DMA REQUEST
        CLR  PX0              ; EXTERNAL INTERRUPT: PRIORITY 0
    
```

231784-32

231784-33

Figure A-14. Serial Channel Interrupt Routine

```
        CLR  P1.2          ; UPDATE STATUS WITH
        CLR  P1.3          ; RECEIVE-DONE EVENT
        SETB P1.4          ; (STATUS=D1H IF E=0)
        CLR  IE0           ; IGNORE PENDING EXT INTO
        CLR  P1.0
        SETB P1.0          ; INTERRUPT THE 80186
        JB   P3.2,$        ; WAIT TILL INTERRUPT IS ACKNOWLEDGED
        RETI

SINTJ3: NOP
        RETI
END
```

231784-34

Figure A-14. Serial Channel Interrupt Routine (Continued)



INTEL, SUPPLY FILLER



INTEL CORPORATION, 2200 Mission College Blvd., Santa Clara, CA 95052; Tel. (408) 765-8080

INTEL CORPORATION (U.K.) Ltd., Swindon, United Kingdom; Tel. (0793) 696 000

INTEL JAPAN k.k., Ibaraki-ken; Tel. 029747-8511

