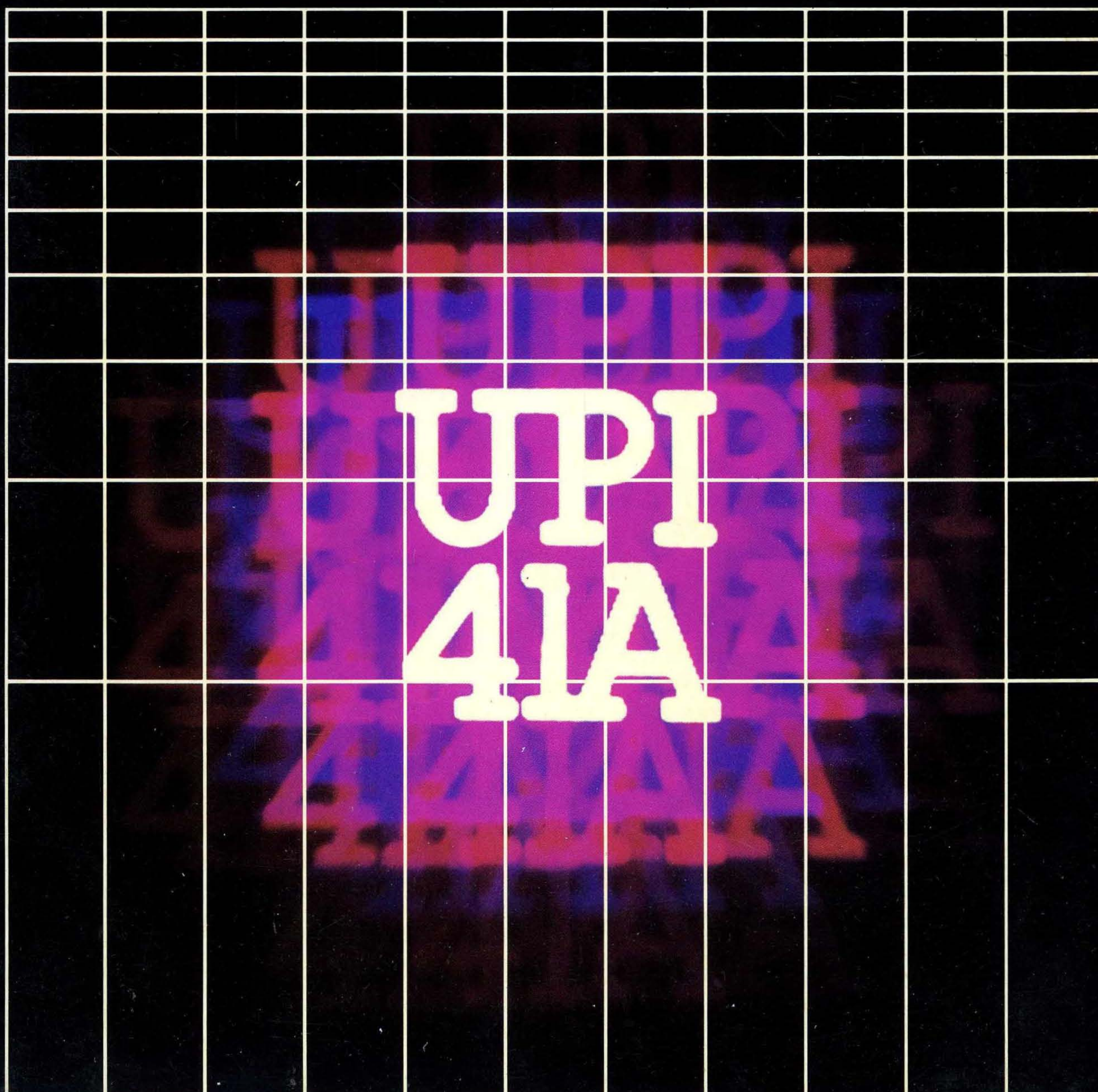
intel®

UPI-41A User's Manual



9800504-02 Rev.B

intel®

UPI 41A™ USER'S MANUAL

APRIL 1980

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

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

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

The following are trademarks of Intel Corporation and may only be used to describe Intel products:

i	Intellec	Multimodule
ICE	iSBC	PROMPT
ICS	Library Manager	Promware
im	MCS	RMX
Insite	Megachassis	UPI
Intel	Micromap	μScope
Intelevision		

and the combinations of ICE, iCS, MCS or RMX and a numerical suffix.

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

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

Table of Contents

CHAPTER 7

CHAPTER 8

CHAPTER 1
INTRODUCTION

1

CHAPTER 1
INTRODUCTION

Accompanying the introduction of microprocessors such as the 8080, 8085, and 8086 there has been a rapid proliferation of intelligent peripheral devices. These special purpose peripherals extend CPU performance and flexibility in a number of important ways.

Table 1-1. Intelligent Peripheral Devices

8255 (GPIO)	Programmable Peripheral Interface
8251 (USART)	Programmable Communication Interface
8253 (TIMER)	Programmable Interval Timer
8257 (DMA)	Programmable DMA Controller
8259	Programmable Interrupt Controller
8272 (DDFDC)	Programmable Floppy Disk Controller
8273 (SDLC)	Programmable Synchronous Data Link Controller
8275 (CRT)	Programmable CRT Controller
8279 (PKD)	Programmable Keyboard/Display Controller
8291, 8292, 8293	Programmable GPIB System Talker, Listener, Controller

Intelligent devices like the 8272 floppy disk controller and 8273 synchronous data link controller (see Table 1-1) can preprocess serial data and perform control tasks which off-load the main system processor. Higher overall system throughout is achieved and software complexity is greatly reduced. The intelligent peripheral chips simplify master processor control tasks by performing many functions externally in peripheral hardware rather than internally in main processor software.

Intelligent peripherals also provide system flexibility. They contain on-chip mode registers which are programmed by the master processor during system initialization. These control registers allow the peripheral to be configured into many different operation modes. The user-defined program for the peripheral is stored in main system memory and is transferred to the peripheral's registers whenever a mode change is required. Of course, this type of flexibility requires software overhead in the master system which tends to limit the benefit derived form the peripheral chip.

In the past, intelligent peripherals were designed to handle very specialized tasks. Separate chips were designed for communication disciplines, parallel I/O, keyboard encoding, interval timing, CRT control, etc. Yet, in spite of the large number of devices available and the increased flexibility built into these chips, there is still a large number of microcomputer peripheral control tasks which are not satisfied.

With the introduction of the Universal Peripheral Interface (UPI) microcomputer, Intel has taken the intelligent peripheral concept a step further by providing an intelligent controller that is fully user programmable. It is a complete single-chip microcomputer which can connect directly to a master processor data bus. It has the same advantages of intelligence and flexibility which previous peripheral chips offered. In addition, the UPI is user-programmable: it has 1K bytes of ROM or EPROM memory for program storage plus 64 bytes of RAM memory for data storage or initialization from the master processor. The UPI device allows a designer to fully specify his control algorithm in the peripheral chip without relying on the master processor. Devices like printer controllers and keyboard scanners can be completely self-contained, relying on the master processor only for data transfer.

The UPI family consists of three components:

- 8741A microcomputer with EPROM memory
- 8041A microcomputer with ROM memory
- 8243 I/O expander device

The 8741A and 8041A single chip microcomputers are functionally equivalent except for the type of program memory available with each. These devices have the following main features:

- 8-bit CPU
- 8-bit data bus interface registers
- 1K by 8 bit ROM or EPROM memory
- 64 by 8 bit RAM memory
- Interval timer/event counter
- Two 8-bit TTL compatible I/O ports
- Resident clock oscillator

The 8243 device is an I/O multiplexer which allows expansion of I/O to over 100 lines (if seven devices are used). All three parts are fabricated with N-channel MOS technology and require a single, 5V supply for operation.

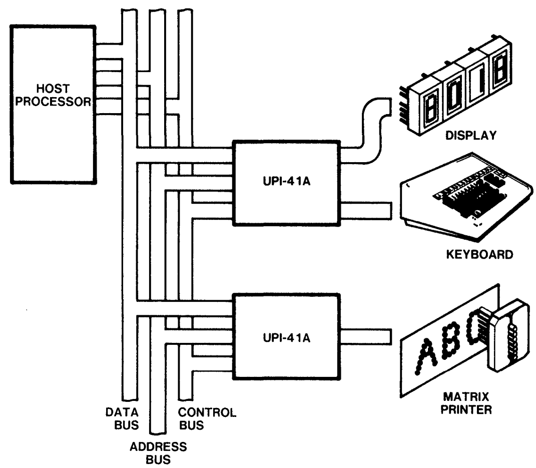


Figure 1-1. Interfacing Peripherals To Microcomputer Systems

INTERFACE REGISTERS FOR MULTI-PROCESSOR CONFIGURATIONS

In the normal configuration, the 8041A/8741A interfaces to the system bus, just like any intelligent peripheral device (see Figure 1-1). The host processor and the 8041A/8741A form a loosely coupled multiprocessor system, that is, communications between the two processors are direct. Common resources are three addressable registers located physically on the 8041A/8741A. These registers are the Data Bus Buffer Input (DBBIN), Data Bus Buffer Output (DBBOUT), and Status (STATUS) registers. The host processor may read data from DBBOUT or write commands and data into DBBIN. The status of DBBOUT and DBBIN plus user-defined status is supplied in STATUS. The host may read STATUS at any time. An interrupt to the UPI processor is automatically generated (if enabled) when DBBIN is loaded.

Because the UPI contains a complete microcomputer with program memory, data memory, and CPU it can function as a "Universal" controller. A designer can program the UPI to control printers, tape transports, or multiple serial communication channels. The UPI can also handle off-line arithmetic processing, or any number of other low speed control tasks.

POWERFUL 8-BIT PROCESSOR

The UPI contains a powerful, 8-bit CPU with 2.5 μsec cycle time and two single-level interrupts. Its instruction set includes over 90 instructions for easy software development. Most instructions are single byte and single cycle and none are more than two bytes long. The instruction set is optimized for bit manipulation and I/O operations. Special instructions are included to allow binary or BCD arithmetic operations, table lookup routines, loop counters, and N-way branch routines.

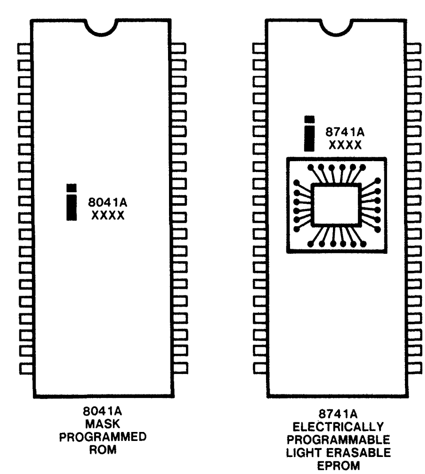


Figure 1-2. Pin Compatible ROM/EPROM Versions

SPECIAL INSTRUCTION SET FEATURES

- For Loop Counters:
  Decrement Register and Jump if not zero.
- For Bit Manipulation:
  AND to A (immediate data or Register)
  OR to A (immediate data or Register)
  XOR to A (immediate data or Register)
  AND to Output Ports (Accumulator)
  OR to Output Ports (Accumulator)
  Jump Conditionally on any bit in A
- For BDC Arithmetic:
  Decimal Adjust A
  Swap 4-bit Nibbles of A
  Exchange lower nibbles of A and Register
  Rotate A left or right with or without Carry
- For Lookup Tables:
  Load A from Page of ROM (Address in A)
  Load A from Current Page of ROM (Address in A)

Features for Peripheral Control

The UPI 8-bit interval timer/event counter can be used to generate complex timing sequences for control applications or it can count external events such as switch closures and position encoder pulses. Software timing loops can be simplified or eliminated by the interval timer. If enabled, an interrupt to the CPU will occur when the timer overflows.

The UPI I/O complement contains two TTL-compatible 8-bit bidirectional I/O ports and two general-purpose test inputs. Each of the 16 port lines can individually function as either input or output under software control. Four of the port lines can also function as an interface for the 8243 I/O expander which provides four additional 4-bit ports that are directly addressable by UPI software. The 8243 expander allows low cost I/O expansion for large control applications while maintaining easy and efficient software port addressing.

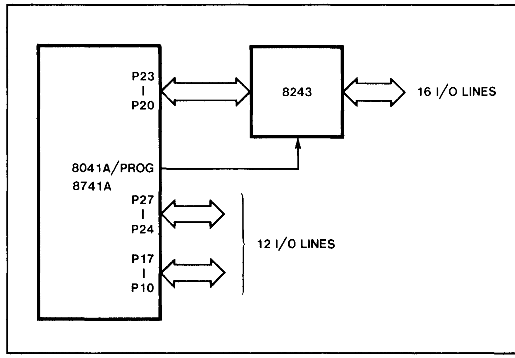


Figure 1-4. 8243 I/O Expander Interface

On-Chip Memory

The UPI's 64 bytes of data memory include dual working register banks and an 8-level program counter stack. Switching between the register banks allows fast response to interrupts. The stack is used to store return addresses and processor status upon entering a subroutine.

The UPI program memory is available in two types to allow flexibility in moving from design to prototype to production with the same PC layout. The 8741A device with EPROM memory is very economical for initial system design and development.

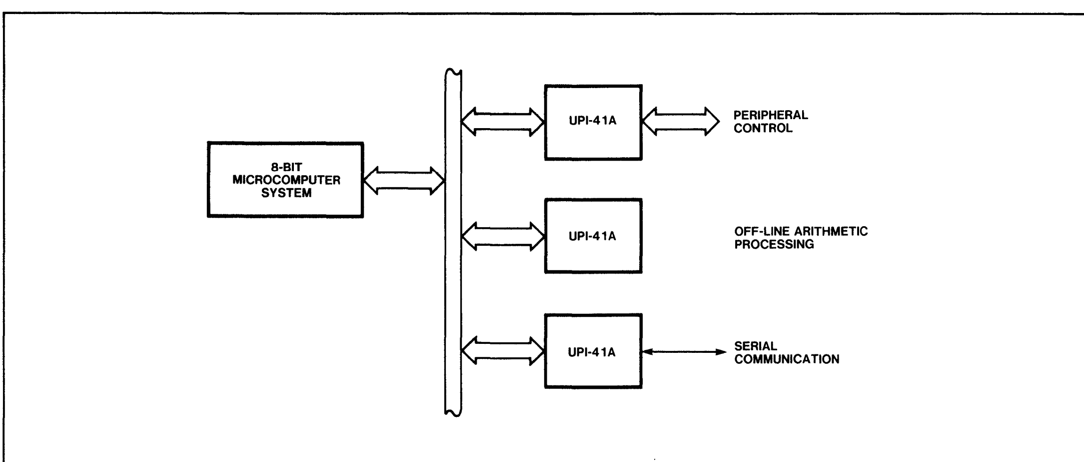


Figure 1-3. Interfaces And Protocols For Multiprocessor Systems

Its program memory can be electrically programmed using the Intel Universal PROM Programmer. When changes are needed, the entire program can be erased using UV lamp and reprogrammed in about 20 minutes. This means the 8741A can be used as a single chip "breadboard" for very complex interface and control problems. After the 8741A is programmed it can be tested in the actual production level PC board and the actual functional environment. Changes required during system debugging can be made in the 8741A program much more easily than they could be made in a random logic design. The system configuration and PC layout can remain fixed during the development process and the turn around time between changes can be reduced to a minimum.

At any point during the development cycle, the 8741A EPROM part can be replaced with the low cost 8041A part with factory mask programmed memory. The transition from system development to mass production is made smoothly because the 8741A and 8041A parts are completely pin compatible. This feature allows extensive testing with the EPROM part, even into initial shipments to customers. Yet, the transition to low-cost ROM is simplified to the point of being merely a package substitution.

PREPROGRAMMED UPI's

The 8292, 8294, and 8295 are 8041A's that are programmed by Intel and sold as standard peripherals. The 8292 is a GPIB controller, part of a three chip GPIB system. The 8294 is a Data Encryption Unit that implements the National Bureau of Standards data encryption algorithm. The 8295 is a dot matrix printer controller designed especially for the LRC 7040 series dot matrix impact printers. These parts illustrate the great flexibility offered by the UPI family.

DEVELOPMENT SUPPORT

The UPI microcomputer is fully supported by Intel with development tools like the UPP PROM programmer already mentioned. An ICE-41A in-circuit emulator is also available to allow UPI software and hardware to be developed easily and quickly. The combination of device features and Intel development support make the UPI an ideal component for low-speed peripheral control applications.

UPI DEVELOPMENT SUPPORT

- 8048/8041A Assembler
- Universal PROM Programmer UPP Series
- ICE-41A Module
- MULTI-ICE
- Insite User's Library
- Application Engineers
- Training Courses

CHAPTER 2
FUNCTIONAL DESCRIPTION

CHAPTER 2
FUNCTIONAL DESCRIPTION

The UPI-41A microcomputer is an intelligent peripheral controller designed to operate in MCS-86, MCS-85, MCS-80, and MCS-48 systems. The UPI'S architecture, illustrated in Figure 2-1, is based on a low cost, single-chip microcomputer with program memory, data memory, CPU, I/O, event timer and clock oscillator in a single 40-pin package. Special interface registers are included which enable the UPI to function as a peripheral to an 8-bit master processor.

This chapter provides a basic description of the UPI microcomputer and its system interface registers. Unless otherwise noted the descriptions in this section apply to both the 8741A (with UV erasable program memory) and the 8041A (with factory mask programmed memory). These two devices are so similar that they can be considered identical under most circumstances. All functions described in this chapter apply to both the 8041A and 8741A.

PIN DESCRIPTION

The 8741A and 8041A are packaged in 40-pin Dual In-Line (DIP) packages. The pin configuration for both devices is shown in Figure 2-2. Figure 2-3 illustrates the UPI Logic Symbol.

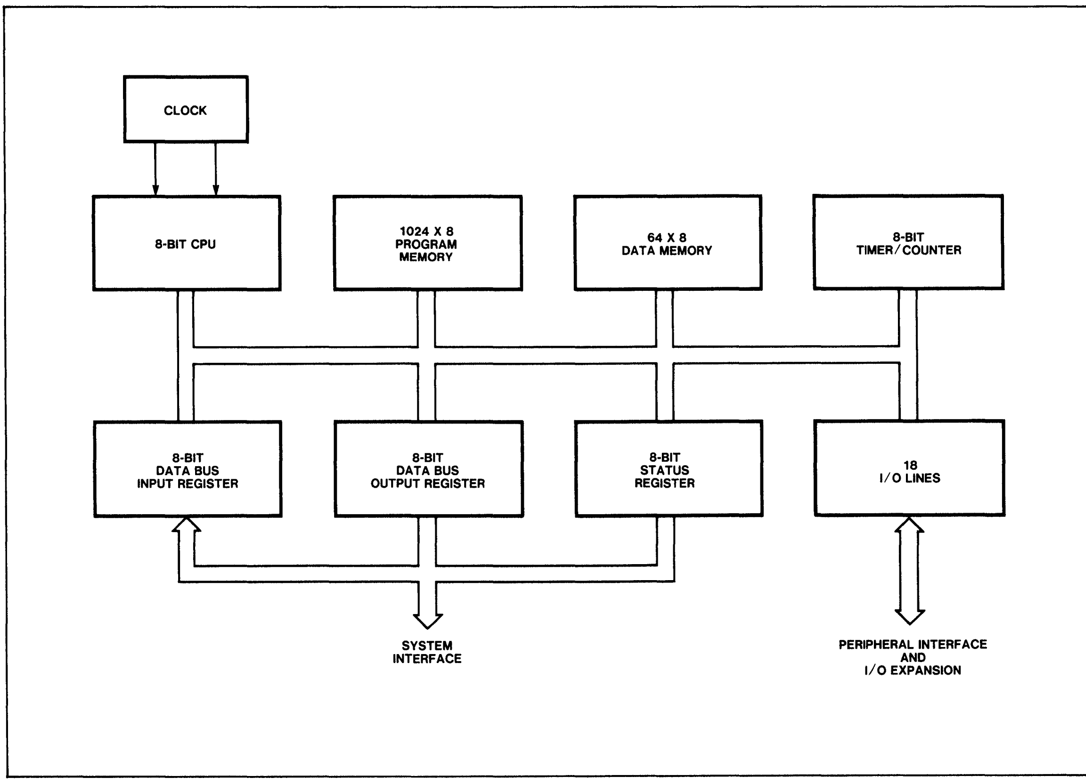


Figure 2-1. UPI-41A Single Chip Microcomputer

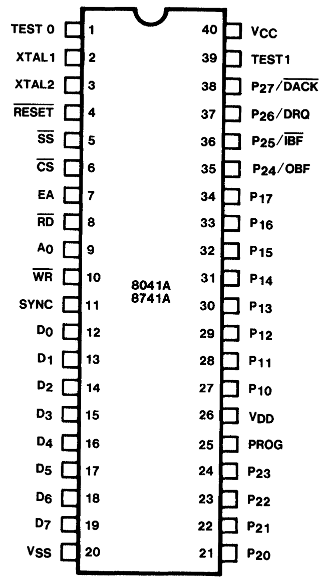


Figure 2-2. Pin Configuration

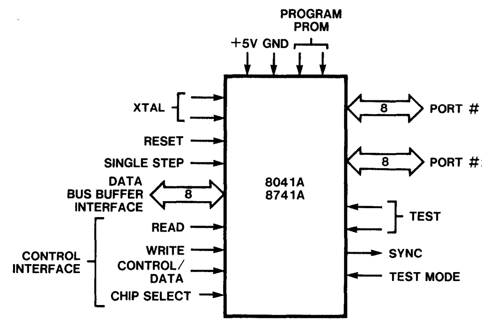


Figure 2-3. Logic Symbol

The following section summarizes the functions of each UPI-41A pin. NOTE that several pins have two or more functions which are described in separate paragraphs.

Table 2-1. Pin Description

Symbol	Pin No.	Type	Name and Function
D_0-D_7 (BUS)	12-19	I/O	**Data Bus:** Three-state, bidirectional DATA BUS BUFFER lines used to interface the UPI-41A microcomputer to an 8-bit master system data bus.
P_{10}-P_{17}	27-34	I/O	**Port 1:** 8-bit, PORT 1 quasi-bidirectional I/O lines.
P_{20}-P_{27}	21-24 35-38	I/O	**Port 2:** 8-bit, PORT 2 quasi-bidirectional I/O lines. The lower 4 bits (P_{20}-P_{23}) interface directly to the 8243 I/O expander device and contain address and data information during PORT 4-7 access. The upper 4 bits (P_{24}-P_{27}) can be programmed to provide interrupt Request and DMA Handshake capability. Software control can configure P_{24} as Output Buffer Full (OBF) interrupt, P_{25} as Input Buffer Full ($\overline{\text{IBF}}$) interrupt, P_{26} as DMA Request (DRQ), and P_{27} as DMA ACKnowledge ($\overline{\text{DACK}}$).
$\overline{\text{WR}}$	10	I	**Write:** I/O write input which enables the master CPU to write data and command words to the UPI-41A INPUT DATA BUS BUFFER.
$\overline{\text{RD}}$	8	I	**Read:** I/O read input which enables the master CPU to read data and status words from the OUTPUT DATA BUS BUFFER or status register.
$\overline{\text{CS}}$	6	I	**Chip Select:** Chip select input used to select one UPI-41A microcomputer out of several connected to a common data bus.
A_0	9	I	**Command/Data Select:** Address input used by the master processor to indicate whether byte transfer is data (A_0=0) or command (A_0=1).
TEST 0, TEST 1	1 39	I	**Test Inputs:** Input pins which can be directly tested using conditional branch instructions. **Frequency Reference:** TEST 1 (T_1) also functions as the event timer input (under software control). TEST 0 (T_0) is used during PROM programming and verification in the 8741A.

Table 2-1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
XTAL 1, XTAL 2	2 3	I	**Inputs:** Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
SYNC	11	O	**Output Clock:** Output signal which occurs once per UPI-41A instruction cycle. SYNC can be used as a strobe for external circuitry; it is also used to synchronize single step operation.
EA	7	I	**External Access:** External access input which allows emulation, testing and PROM/ROM verification.
PROG	25	I/O	**Program:** Multifunction pin used as the program pulse input during PROM programming. During I/O expander access the PROG pin acts as an address/data strobe to the 8243.
$\overline{\text{RESET}}$	4	I	**Reset:** Input used to reset status flip-flops and to set the program counter to zero. $\overline{\text{RESET}}$ is also used during PROM programming and verification.
$\overline{\text{SS}}$	5	I	**Single Step:** Single step input used in the 8741A in conjunction with the SYNC output to step the program through each instruction.
V_{CC}	40		**Power:** +5V main power supply pin.
V_{DD}	26		**Power:** +5V during normal operation. +25V during programming operation. Low power standby pin in ROM version.
V_{SS}	20		**Ground:** Circuit ground potential.

The following sections provide a detailed functional description of the UPI microcomputer. Figure 2-4 illustrates the functional blocks within the UPI device.

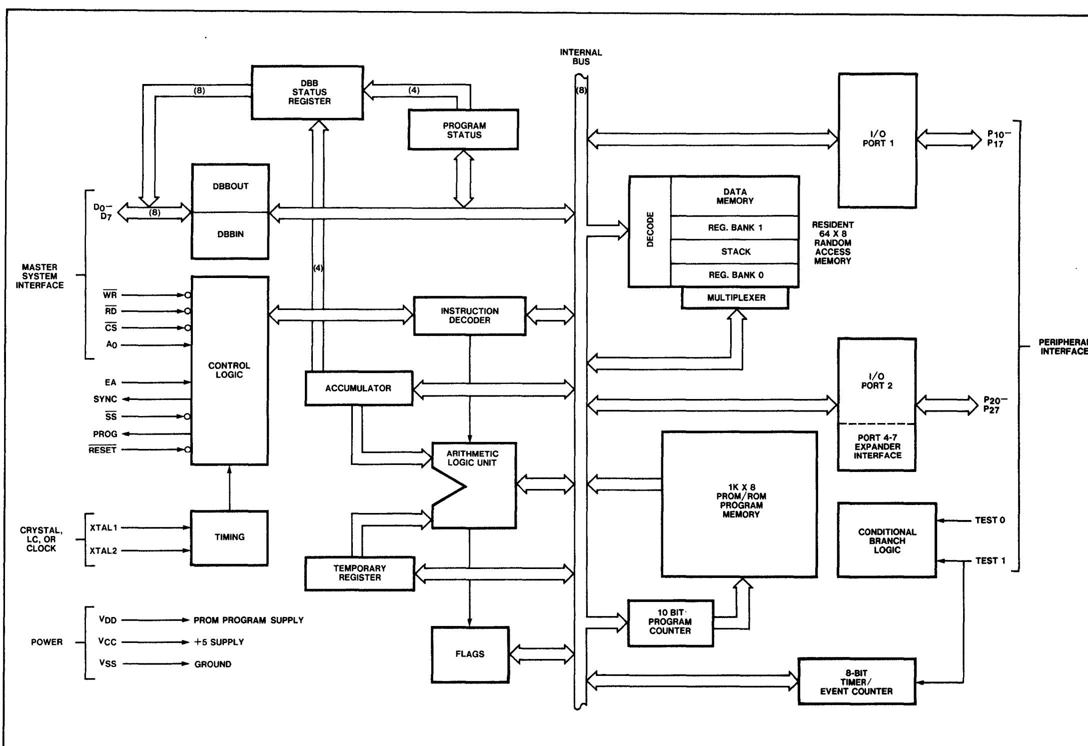


Figure 2-4. UPI-41A™ Block Diagram

CPU SECTION

The CPU section of the UPI-41A microcomputer performs basic data manipulations and controls data flow throughout the single chip computer via the internal 8-bit data bus. The CPU section includes the following functional blocks shown in Figure 2-4:

- Arithmetic Logic Unit (ALU)
- Instruction Decoder
- Accumulator
- Flags

Arithmetic Logic Unit (ALU)

The ALU is capable of performing the following operations:

- ADD with or without carry
- AND, OR, and EXCLUSIVE OR
- Increment, Decrement
- Bit complement
- Rotate left or right
- Swap
- BCD decimal adjust

In a typical operation data from the accumulator is combined in the ALU with data from some other source on the UPI-41A internal bus (such as a register or an I/O port). The result of an ALU operation can be transferred to the internal bus or back to the accumulator.

If an operation such as an ADD or ROTATE requires more than 8 bits, the CARRY flag is used as an indicator. Likewise, during decimal adjust and other BCD operations the AUXILIARY CARRY flag can be set and acted upon. These flags are part of the Program Status Word (PSW).

Instruction Decoder

During an instruction fetch, the operation code (opcode) portion of each program instruction is stored and decoded by the instruction decoder. The decoder generates outputs used along with various timing signals to control the functions performed in the ALU. Also, the instruction decoder controls the source and destination of ALU data.

Accumulator

The accumulator is the single most important register in the processor. It is the primary source of data to the ALU and is often the destination for results as well. Data to and from the I/O ports and memory normally passes through the accumulator.

PROGRAM MEMORY

The UPI-41A microcomputer has 1024 8-bit words of resident, read-only memory for program storage. Each of these memory locations is directly addressable by a 10-bit program counter. Depending on the type of application and the number of program changes anticipated, two types of program memory are available:

- 8041A with mask programmed ROM Memory
- 8741A with electrically programmable EPROM Memory

The 8041A and 8741A are functionally identical parts and are completely pin compatible. The 8041A has ROM memory which is mask programmed to user specification during fabrication. The 8741A is electrically programmed by the user using the Universal PROM Programmer (UPP series) with a UPP-848 Personality Card. It can be erased using ultraviolt light and reprogrammed at any time.

A program memory map is illustrated in Figure 2-5. Memory is divided into 256 location 'pages' and three locations are reserved for special use:

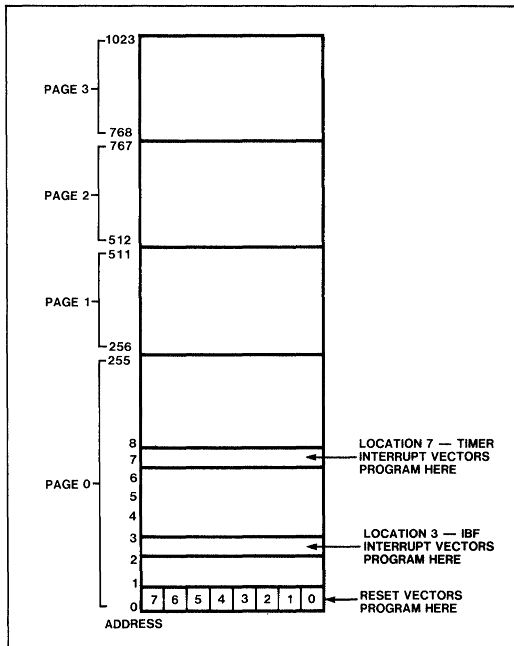


Figure 2-5. Program Memory Map

INTERRUPT VECTORS

1) **Location 0**
   Following a $\overline{\text{RESET}}$ input to the processor, the next instruction is automatically fetched from location 0.

2) **Location 3**
   An interrupt generated by an Input Buffer Full (IBF) condition (when the IBF interrupt is enabled) causes the next instruction to be fetched from location 3.

3) **Location 7**
A timer overflow interrupt (when enabled) will cause the next instruction to be fetched from location 7.

Following a system $\overline{\text{RESET}}$, program execution begins at location 0. Instructions in program memory are normally executed sequentially. Program control can be transferred out of the main line of code by an input buffer full (IBF) interrupt or a timer interrupt, or when a jump or call instruction is encountered. An IBF interrupt (if enabled) will automatically transfer control to location 3 while a timer interrupt will transfer control to location 7.

All conditional JUMP instructions and the indirect JUMP instruction are limited in range to the current 256-location page (that is, they alter PC bits 0–7 only). If a conditional JUMP or indirect JUMP begins in location 255 of a page, it must reference a destination on the following page.

Program memory can be used to store constants as well as program instructions. the UPI-41A instruction set contains an instruction (MOVP3) designed specifically for efficient transfer of look-up table information from page 3 of memory.

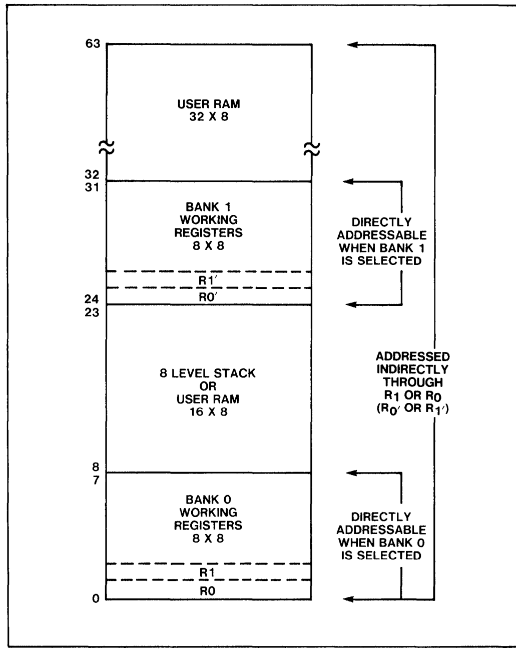


Figure 2-6. Data Memory Map

DATA MEMORY

The UPI-41A universal peripheral interface has 64 8-bit words of random access data memory. This memory contains two working register banks, an 8-level program counter stack and a scratch pad memory, as shown in Figure 2-6. The amount of scratch pad memory available is variable depending on the number of addresses nested in the stack and the number of working registers being used.

Addressing Data Memory

The first eight locations in RAM are designated as working registers R_0–R_7. These locations (or registers) can be addressed directly by specifying a register number in the instruction. Since these locations are easily addressed, they are generally used to store frequently accessed intermediate results. Other locations in data memory are addressed indirectly by using R_0 or R_1 to specify the desired address. Since all RAM locations (including the eight working registers) can be addressed by 6 bits, the two most significant bits (6 and 7) of the addressing registers are ignored.

Working Registers

Dual banks of eight working registers are included in the UPI-41A data memory. Locations 0–7 make up register bank 0 and locations 24–31 form register bank 1. A $\overline{\text{RESET}}$ signal automatically selects register bank 0. When bank 0 is selected, references to R_0–R_7 in UPI-41A instructions operate on locations 0–7 in data memory. A "select register bank" instruction is used to select between the banks during program execution. If the instruction SEL RB1 (Select Register Bank 1) is executed, then program references to R_0–R_7 will operate on locations 24–31. As stated previously, registers 0 and 1 in the active register bank are used as indirect address registers for all locations in data memory.

Register bank 1 is normally reserved for handling interrupt service routines, thereby preserving the contents of the main program registers. The SEL RB1 instruction can be issued at the beginning of an interrupt service routine. Then, upon return to the main program, an RETR (return & restore status) instruction will automatically restore the previously selected bank. During interrupt processing, registers in bank 0 can be accessed indirectly using R_0' and R_1'.

If register bank 1 is not used, registers 24–31 can still serve as additional scratch pad memory.

Program Counter Stack

RAM locations 8–23 are used as an 8-level program counter stack. When program control is temporarily passed from the main program to a subroutine or interrupt service routine, the 10-bit program counter and bits 4–7 of the program status word (PSW) are stored in two stack locations. When control is returned to the main program via an RETR instruction, the program counter and PSW bits 4–7 are restored. Returning via an RET instruction does not restore the PSW bits, however. The program counter stack is addressed by three stack pointer bits in the PSW (bits 0–2). Operation of the program counter stack and the program status word is explained in detail in the following sections.

The stack allows up to eight levels of subroutine 'nesting'; that is, a subroutine may call a second subroutine, which may call a third, etc., up to eight levels. Unused stack locations can be used as scratch pad memory. Each unused level of subroutine nesting provides two additional RAM locations for general use.

The following sections provide a detailed description of the Program Counter Stack and the Program Status Word.

PROGRAM COUNTER

The UPI-41A microcomputer has a 10-bit program counter (PC) which can directly address any of the 1024 locations in program memory. The program counter always contains the address of the next instruction to be executed and is normally incremented sequentially for each instruction to be executed when each instruction fetches occurs.

When control is temporarily passed from the main program to a subroutine or an interrupt routine, however, the PC contents must be altered to point to the address of the desired routine. The stack is used to save the current PC contents so that, at the end of the routine, main program execution can continue. The program counter is initialized to zero by a $\overline{\text{RESET}}$ signal.

PROGRAM COUNTER STACK

The Program Counter Stack is composed of 16 locations in Data Memory as illustrated in Figure 2-7. These RAM locations (8 through 23) are used to store the 10-bit program counter and 4 bits of the program status word.

An interrupt or CALL to a subroutine causes the contents of the program counter to be stored in one of the 8 register pairs of the program counter stack.

A 3-bit Stack Pointer which is part of the Program Status Word (PSW) determines the stack pair to be used at a given time. The stack pointer is initialized by a $\overline{\text{RESET}}$ signal to 00H which corresponds to RAM locations 8 and 9.

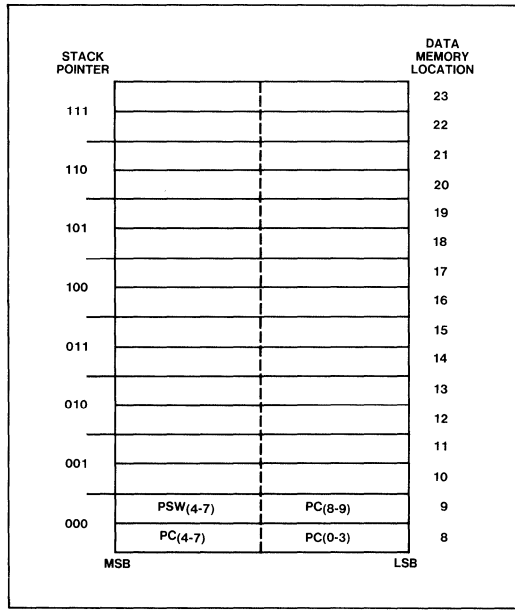


Figure 2-7. Program Counter Stack

The first call or interrupt results in the program counter and PSW contents being transferred to RAM locations 8 and 9 in the format shown in Figure 2-7. The stack pointer is automatically incremented by 1 to point to locations 10 and 11 in anticipation of another CALL.

Nesting of subroutines within subroutines can continue up to 8 levels without overflowing the stack. If overflow does occur the deepest address stored (locations 8 and 9) will be overwritten and lost since the stack pointer overflows from 07H to 00H. Likewise, the stack pointer will underflow from 00H to 07H.

The end of a subroutine is signaled by a return instruction, either RET or RETR. Each instruction will automatically decrement the Stack Pointer and transfer the contents of the proper RAM register pair to the Program Counter.

PROGRAM STATUS WORD

The 8-bit program status word illustrated in Figure 2-8 is used to store general information about program execution. In addition to the 3-bit Stack Pointer discussed previously, the PSW includes the following flags:

- CY — Carry
- AC — Auxiliary Carry
- F_0 — Flag 0
- BS — Register Bank Select

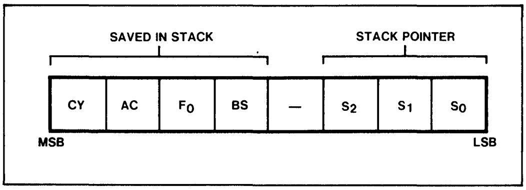


Figure 2-8. Program Status Word

The Program Status Word (PSW) is actually a collection of flip-flops located throughout the machine which are read or written as a whole. The PSW can be loaded to or from the accumulator by the MOV A, PSW or MOV PSW,A instructions. The ability to write directly to the PSW allows easy restoration of machine status after a power-down sequence.

The upper 4 bits of the PSW (bits 4, 5, 6, and 7) are stored in the PC Stack with every subroutine CALL or interrupt vector. Restoring the bits on a return is optional. The bits are restored if an RETR instruction is executed, but not if an RET is executed.

PSW bit definitions are as follows:

- Bits 0–2 Stack Pointer Bits S_0, S_1, S_2
- Bit 3 Not Used
- Bit 4 Working Register Bank
  0 = Bank 0
  1 = Bank 1
- Bit 5 Flag 0 bit (F_0)
  This is a general purpose flag which can be cleared or complemented and tested with conditional jump instructions. It may be used during data transfer to an external processor.
- Bit 6 Auxiliary Carry (AC)
  The flag status is determined by an ADD instruction and is used by the Decimal Adjustment instruction DAA.
- Bit 7 Carry (CY)
  The flag indicates that a previous operation resulted in overflow of the accumulator.

CONDITIONAL BRANCH LOGIC

Conditional Branch Logic in the UPI-41A allows the status of various processor flags, inputs, and other hardware functions to directly affect program execution. The status is sampled in state 3 of the first cycle.

Table 2-2 lists the internal conditions which are testable and indicates the condition which will cause a jump. In all cases, the destination address must be within the page of program memory (256 locations) in which the jump instruction occurs.

OSCILLATOR AND TIMING CIRCUITS

The 8041A's internal timing generation is controlled by a self-contained oscillator and timing circuit. A choice of crystal, L-C or external clock can be used to derive the basic oscillator frequency.

The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 2-9. Figure 2-10 shows instruction cycle timing.

Oscillator

The on-board oscillator is a series resonant circuit with a frequency range of 1 to 6 MHz. Pins XTAL 1 and XTAL 2 are input and output (respectively) of a high gain amplifier stage. A crystal or inductor and capacitor connected between XTAL 1 and XTAL 2 provide the feedback and proper phase shift for os-

Table 2-2. Conditional Branch Instructions

Device	Instruction Mnemonic		Jump Condition Jump if:
Accumulator	JZ	addr	All bits zero
	JNZ	addr	Any bit not zero
Accumulator bit	JBb	addr	Bit "b" = 1
Carry flag	JC	addr	Carry flag = 1
	JNC	addr	Carry flag = 0
User flag	JFO	addr	F_0 flag = 1
	JF1	addr	F_1 flag = 1
Timer flag	JTF	addr	Timer flag = 1
Test Input 0	JT0	addr	$T_0 = 1$
	JNT0	addr	$T_0 = 0$
Test Input 1	JT1	addr	$T_1 = 1$
	JNT1	addr	$T_1 = 0$
Input Buffer flag	JNIBF	addr	IBF flag = 0
Output Buffer flag	JOBF	addr	OBF flag = 1

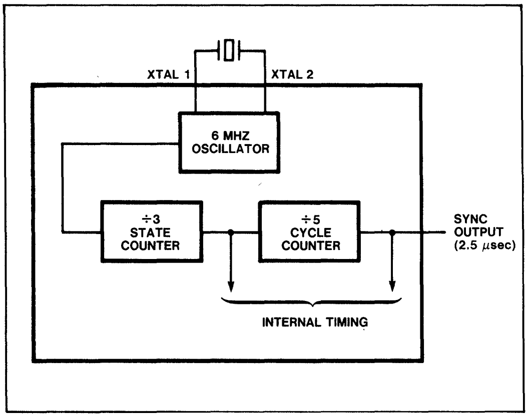


Figure 2-9. Oscillator Configuration

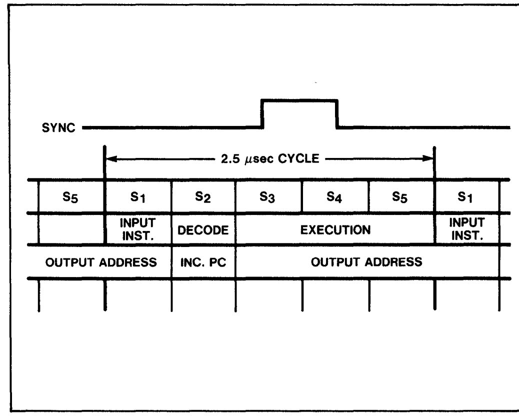


Figure 2-10. Instruction Cycle Timing

cillation. Recommended connections for crystal or L-C are shown in Figure 2-11.

State Counter

The output of the oscillator is divided by 3 in the state counter to generate a signal which defines the state times of the machine.

Each instruction cycle consists of five states as illustrated in Figure 2-10 and Table 2-3. The overlap of address and execution operations illustrated in Figure 2-10 allows fast instruction execution.

Cycle Counter

The output of the state counter is divided by 5 in the cycle counter to generate a signal which defines a machine cycle. This signal is call SYNC and is available continously on the SYNC output pin. It can be used to synchronize external circuitry or as a general purpose clock output. It is also used for synchronizing single-step in the 8741A.

Table 2-3. Instruction Timing Diagram

INSTRUCTION	CYCLE 1					CYCLE 2				
	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
IN A,Pp	Fetch Instruction	Increment Program Counter	—	Increment Timer	—	—	Read Port	—	—	—
OUTL Pp,A	Fetch Instruction	Increment Program Counter	—	Increment Timer	Output To Port	—	—	—	—	—
ANL Pp, DATA	Fetch Instruction	Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	Increment Program Counter	Output To Port	—
ORL Pp, DATA	Fetch Instruction	Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	Increment Program Counter	Output To Port	—
MOVD A,Pp	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	—	—	Read P2 Lower	—	—	—
MOVD Pp,A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data To P2 Lower	—	—	—	—	—
ANLD Pp,A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data	—	—	—	—	—
ORLD Pp,A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data	—	—	—	—	—
J (Conditional)	Fetch Instruction	Increment Program Counter	Sample Condition	Increment Timer	—	Fetch Immediate Data	—	Update Program Counter	—	—
IN A,DBB	Fetch Instruction	Increment Program Counter	—	Increment Timer	—					
OUT DBB,A	Fetch Instruction	Increment Program Counter	—	Increment Timer	Output To Port					
STRT T STRT CNT	Fetch Instruction	Increment Program Counter	—	—	Start Counter					
STOP TCNT	Fetch Instruction	Increment Program Counter	—	—	Stop Counter					
EN I	Fetch Instruction	Increment Program Counter	—	Enable Interrupt	—					
DIS I	Fetch Instruction	Increment Program Counter	—	Disable Interrupt	—					
EN DMA	Fetch Instruction	Increment Program Counter	—	DMA Enabled DRQ Cleared	—					
EN FLAGS	Fetch Instruction	Increment Program Counter	—	OBF, $\overline{IBF}$ Output Enabled	—					

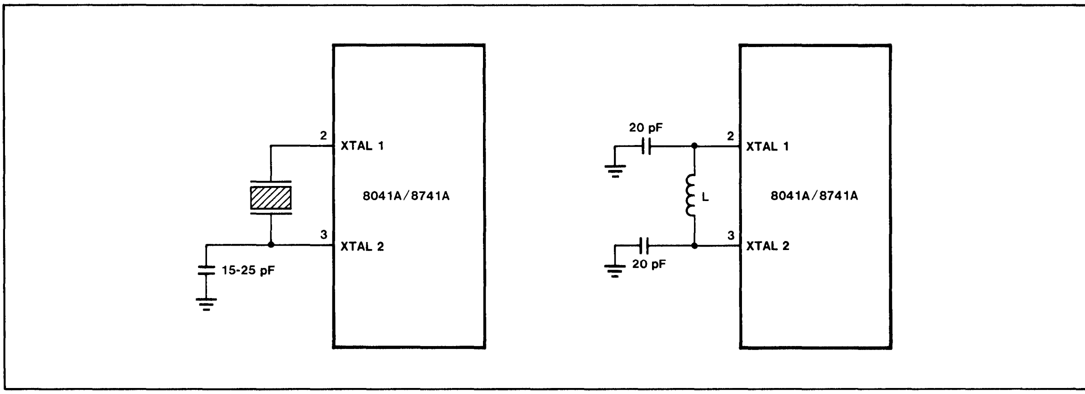


Figure 2-11. Recommended Crystal and L-C Connections

Frequency Reference

The external crystal provides high speed and accurate timing generation. A crystal frequency of 5.9904 MHz is useful for generation of standard communication frequencies by the 8741A and 8041A. However, if an accurate frequency reference and maximum processor speed are not required, an inductor and capacitor may be used in place of the crystal as shown in Figure 2-11.

A recommended range of inductance and capacitance combinations is given below:

- L = 130 μH corresponds to 3 MHz
- L = 45 μH corresponds to 5 MHz

An external clock signal can also be used as a frequency reference to the 8741A or 8041A; however, the levels are *not* TTL compatible. The signal must be in the 1–6 MHz frequency range and must be connected to pins XTAL 1 and XTAL 2 by buffers with a suitable pull-up resistor to guarantee that a logic "1" is above 3.8 volts. The recommended connection is shown in Figure 2-12.

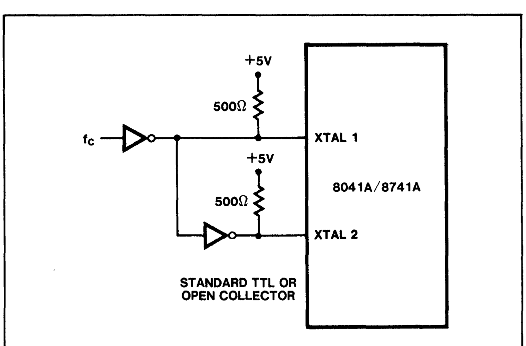


Figure 2-12. Recommended Connection For External Clock Signal

INTERVAL TIMER/EVENT COUNTER

The 8041A has a resident 8-bit timer/counter which has several software selectable modes of operation. As an interval timer, it can generate accurate delays from 80 microseconds to 20.48 milliseconds without placing undue burden on the processor. In the counter mode, external events such as switch closures or tachometer pulses can be counted and used to direct program flow.

Timer Configuration

Figure 2-13 illustrates the basic timer/counter configuration. An 8-bit register is used to count pulses from either the internal clock and prescaler or from an external source. The counter is presettable and readable with two MOV instructions which transfer the contents of the accumulator to the counter and vice-versa. The counter is initialized solely by the MOV T,A instruction; it is not cleared by a $\overline{\text{RESET}}$ signal. The counter is stopped by a $\overline{\text{RESET}}$ or STOP TCNT instruction and remains stopped until restarted either as a timer (START T instruction) or as a counter (START CNT instruction). Once started, the counter will increment to its maximum count (FFH) and overflow to zero continuing its count until stopped by a STOP TCNT instruction or $\overline{\text{RESET}}$.

The increment from maximum count to zero (overflow) results in setting the Timer Flag (TF) and generating an interrupt request. The state of the overflow flag is testable with the conditional jump instruction, JTF. The flag is reset by executing a JTF or by a $\overline{\text{RESET}}$ signal.

The timer interrupt request is stored in a latch and ORed with the input buffer full interrupt request. The timer interrupt can be enabled or disabled independent of the IBF interrupt by the EN TCNTI and

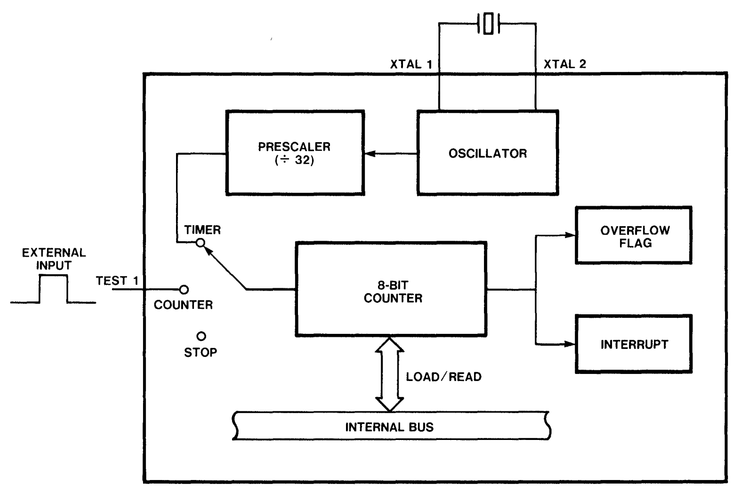


Figure 2-13. Timer Counter

DIS TCTNI instructions. If enabled, the counter overflow will cause a subroutine call to location 7 where the timer service routine is stored. If the timer and Input Buffer Full interrupts occur simultaneously, the IBF source will be recognized and the call will be to location 3. Since the timer interrupt is latched, it will remain pending until the DBBIN register has been serviced and will immediately be recognized upon return from the service routine. A pending timer interrupt is reset by the initiation of a timer interrupt service routine.

Event Counter Mode

The START CNT instruction connects the TEST 1 input pin to the counter input and enables the counter. Note that this instruction does not clear the counter. The counter is incremented on high to low transition of TEST 1. The maximum count rate is one count per three instruction cycles (every 7.5 microseconds when using a 6 MHz crystal). There is no minimum frequency limit. The TEST 1 input must remain high for a minimum of 500 ns (at 6 MHz) during a count cycle.

Timer Mode

The START T instruction connects an internal clock to the counter input and enables the counter. The input frequency is derived from a divide by 32 prescaler connected to the 400 kHz machine cycle clock. The configuration is illustrated in Figure 2-13. The resulting 12.5 kHz clock provides a counter increment every 80 μsec. Various delays and timing sequences between 80 μsec and 20.48 msec can easily be generated with a miniumum of software timing loops. Times longer than 20 msec can be accurately measured by accumulating multiple overflows in a register under software control. For time resolution less than 80 μsec an external clock can be applied to the TEST 1 input and the counter can be operated in the event counter mode. The 2.5 μsec SYNC output divided by 3 or more can serve as the external clock. Software loops can also be used to "fine tune" long delays generated by the timer.

TEST 1 Event Counter Input

The TEST 1 pin is multifunctional. It is automatically initialized as a test input by a $\overline{\text{RESET}}$ signal and can be tested using UPI-41A conditional branch instructions.

In the second mode of operation, illustrated in Figure 2-13, the TEST 1 pin is used as an input to the internal 8-bit event counter. The Start Counter (STRT CNT) instruction controls an internal switch which connects TEST 1 through an edge detector to the 8-bit internal counter. Note that this instruction does not inhibit the testing of TEST 1 via conditional Jump instructions.

In the counter mode the TEST 1 input is sampled once per instruction cycle. After a high level is detected, the next occurence of a low level at TEST 1 will cause the counter to increment by one.

The event counter functions can be stopped by the Stop Timer/Counter (STOP TCNT) instruction. When this instruction is executed the TEST 1 pin becomes a test input and functions as previously described.

TEST INPUTS

There are two multifunction pins designated as Test Inputs, TEST 0 and TEST 1. In the normal mode of operation, status of each of these lines can be directly tested using the following conditional Jump instructions:

- JT0 Jump if TEST 0 = 1
- JNT0 Jump if TEST 0 = 0
- JT1 Jump if TEST 1 = 1
- JNT1 Jump if TEST 1 = 0

The test inputs are TTL compatible. An external logic signal connnected to one of the test inputs will be sampled at the time the appropriate conditional jump instruction is executed. The path of program execution will be altered depending on the state of the external signal when sampled.

INTERRUPTS

The 8041A/8741A has the following internal interrupts:

- Input Buffer Full (IBF) interrupt
- Timer Overflow interrupt

The IBF interrupt forces a CALL to location 3 in program memory; a timer-overflow interrupt forces a CALL to location 7. The IBF interrupt is enabled by the EN I instruction and disabled by the DIS I instruction. The timer-overflow interrupt is enabled and disabled by the EN TNCTI and DIS TCNTI instructions, respectively.

Figure 2-14 illustrates the internal interrupt logic. An IBF interrupt request is generated whenever $\overline{WR}$ and $\overline{CS}$ are both low, regardless of whether interrupts are enabled. The interrupt request is cleared upon entering the IBF service routine only. That is, the DIS I instruction does not clear a pending IBF interrupt.

Interrupt Timing Latency

When the IBF interrupt is enabled and an IBF interrupt request occurs, an interrupt sequence is initiated as soon as the currently executing instruction is completed. The following sequence occurs:

- A CALL to location 3 is forced.
- The program counter and bits 4–7 of the Program Status Word are stored in the stack.
- The stack pointer is incremented.

Location 3 in program memory should contain an

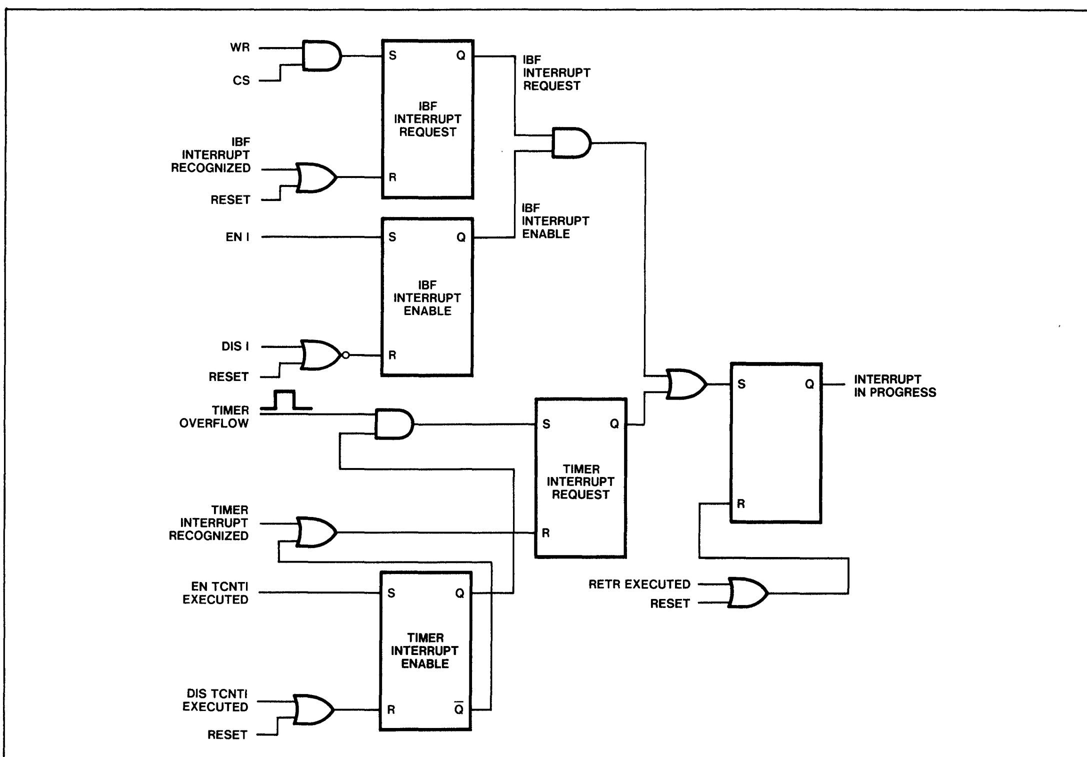


Figure 2-14. Interrupt Logic

unconditional jump to the beginning of the IBF interrupt service routine elsewhere in program memory. At the end of the service routine, an RETR (Return and Restore Status) instruction is used to return control to the main program. This instruction will restore the program counter and PSW bits 4–7, providing automatic restoration of the previously active register bank as well. RETR also re-enables interrupts.

A timer-overflow interrupt is enabled by the EN TCNTI instruction and disabled by the DIS TCNTI instruction. If enabled, this interrupt occurs when the timer/counter register overflows. A CALL to location 7 is forced and the interrupt routine proceeds as described above.

The interrupt service latency is the sum of current instruction time, interrupt recognition time, and the internal call to the interrupt vector address. The worst case latency time for servicing an interrupt is 7 clock cycles. Best case latency is 4 clock cycles.

Interrupt Timing

Interrupt inputs may be enabled or disabled under program control using EN I, DIS I, EN TCNTI and DIS TCNTI instructions. Also, a $\overline{\text{RESET}}$ input will disable interrupts. An interrupt request must be removed before the RETR instruction is executed to return from the service routine, otherwise the processor will re-enter the service routine immediately. Thus, the $\overline{\text{WR}}$ and $\overline{\text{CS}}$ inputs should not be held low longer than the duration of the interrupt service routine.

The interrupt system is single level. Once an interrupt is detected, all further interrupt requests are latched but are not acted upon until execution of an RETR instruction re-enables the interrupt input logic. This occurs at the beginning of the second cycle of the RETR instruction. If an IBF interrupt and a timer-overflow interrupt occur simultaneously, the IBF interrupt will be recognized first and the timer-overflow interrupt will remain pending until the end of the interrupt service routine.

External Interrupts

An external interrupt can be created using the UPI-41A timer/counter in the event counter mode. The counter is first preset to FFH and the EN TCNTI instruction is executed. A timer-overflow interrupt is generated by the first high to low transition of the TEST 1 input pin. Also, if an IBF interrupt occurs during servicing of the timer/counter interrupt, it will remain pending until the end of the service routine.

Host Interrupts And DMA

If needed, two external interrupts to the host system can be created using the EN FLAGS instruction. This instruction allocates two I/O lines on PORT 2 (P_{24} and P_{25}). P_{24} is the Output Buffer Full interrupt request line to the host system; P_{25} is the Input Buffer empty interrupt request line. These interrupt outputs reflect the internal status of the OBF flag and the IBF inverted flag. Note, these outputs may be inhibited by writing a "0" to these pins. Reenabling interrupts is done by writing a "1" to these port pins. Interrupts are typically enabled after power on since the I/O ports are set in a "1" condition. The EN FLAG's effect is only cancelled by a device RESET.

DMA handshaking controls are available from two pins on PORT 2 of the UPI-41A microcomputer. These lines (P_{26} and P_{27}) are enabled by the EN DMA instruction. P_{26} becomes DMA request (DRQ) and P_{27} becomes DMA acknowledge ($\overline{\text{DACK}}$). The UPI program initiates a DMA request by writing a "1" to P_{26}. The DMA controller transfers the data into the DBBIN data register using DACK which acts as a chip select. The EN DMA instruction can only be cancelled by a chip RESET.

RESET

The $\overline{\text{RESET}}$ input on the 8041A/8741A provides a means for internal initialization of the processor. An automatic initialization pulse can be generated at power turn-on simply by connecting a 1 μfd capacitor between the $\overline{\text{RESET}}$ input and ground as shown in Figure 2-15. It has an internal pull-up resistor to charge the capacitor and a Schmitt-trigger circuit to generate a clean transition.

If an external $\overline{\text{RESET}}$ pulse is used it must hold the $\overline{\text{RESET}}$ input low for at least 10 milliseconds after the power supply is within tolerance. Figure 2-15 illustrates a configuration using an external TTL gate to generate the $\overline{\text{RESET}}$ input. This configuration can be used to derive the $\overline{\text{RESET}}$ signal from the 8224 clock generator in an 8080 system.

The $\overline{\text{RESET}}$ input performs the following functions:

- Sets Program Counter to zero.
- Sets the Stack Pointer to zero
- Selects Register Bank 0
- Sets PORTS 1 and 2 to the Input Mode
- Disables interrupts.
- Stops the timer.
- Clears the timer flag.
- Clears F_0 and F_1 flip-flops.

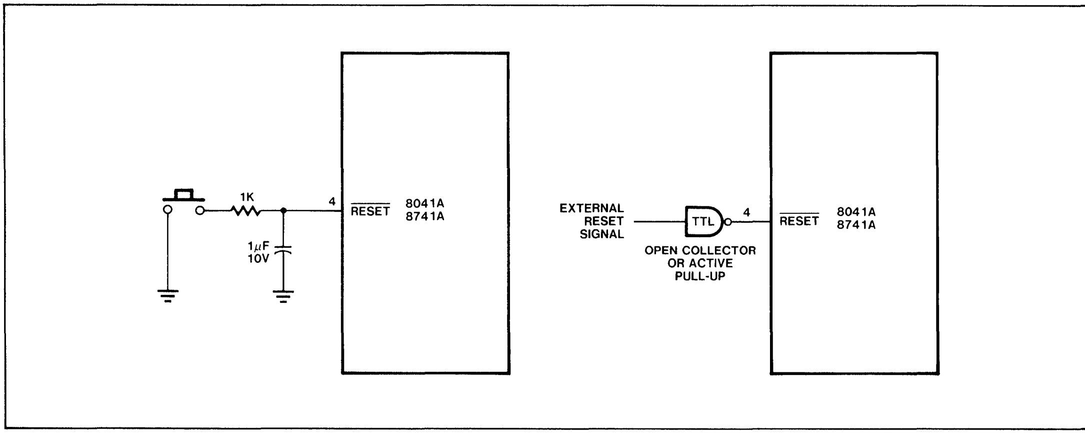


Figure 2-15. External Reset Configuration

DATA BUS BUFFER

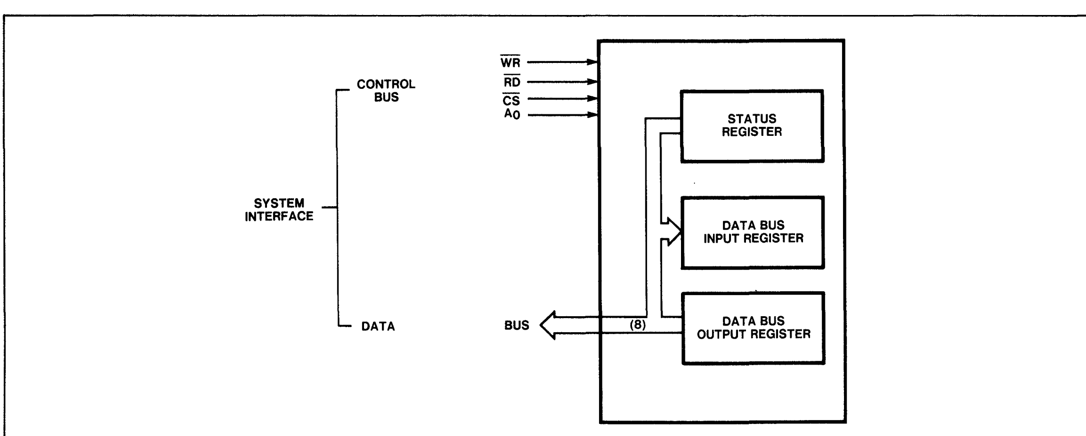
Two 8-bit data bus buffer registers, DBBIN and DBBOUT, serve as temporary buffers for commands and data flowing between it and the master processor. Externally, data is transmitted or received by the DBB registers upon execution of an INput or OUTput instruction by the master processor. Four control signals are used:

- A_0 Address input signifying control or data
- $\overline{CS}$ Chip Select
- $\overline{RD}$ Read strobe
- $\overline{WR}$ Write strobe

Transfer can be implemented with or without UPI program interference by enabling or disabling an internal UPI interrupt. Internally, data transfer between the DBB and the UPI accumulator is under software control and is completely asynchronous to the external processor timing. This allows the UPI software to handle peripheral control tasks independent of the main processor while still maintaining a data interface with the master system.

Configuration

Figure 2-16 illustrates the internal configuration of the DBB registers. Data is stored in two 8-bit buffer registers, DBBIN and DBBOUT. DBBIN and DBBOUT may be accessed by the external processor using the $\overline{WR}$ line and the $\overline{RD}$ line, respectively. The data bus is a bidirectional, three-state bus which can be connected directly to an 8-bit microprocessor system. Four control lines ($\overline{WR}$, $\overline{RD}$, $\overline{CS}$, A_0) are used by the external processor to transfer data to and from the DBBIN and DBBOUT registers.



2-16. Data Bus Buffer Configuration

An 8-bit register containing status flags is used to indicate the status of the DBB registers. The eight status flags are defined as follows:

- **OBF Output Buffer Full** This flag is automatically set when the 8041A loads the DBBOUT register and is cleared when the master processor reads the data register.
- **IBF Input Buffer Full** This flag is set when the master processor writes a character to the DBBIN register and is cleared when the 8041A INputs the data register contents to its accumulator.
- **F_0** This is a general purpose flag which can be cleared or toggled under 8041A software control. The flag is used to transfer 8041A status information to the master processor.
- **F_1 Command/Data** This flag is set to the condition of the A_0 input line when the master processor writes a character to the data register. The F_1 flag can also be cleared or toggled under 8041A program control.
- **ST4 Through ST7** These bits are user defined status bits. They are defined by the MOV STS,A instruction.

All flags in the status register are automatically cleared by a $\overline{\text{RESET}}$ input.

SYSTEM INTERFACE

Figure 2-17 illustrates how an 8041A can be connected to a standard 8080-type bus system. Data lines D_0–D_7 form a three-state, bidirectional port which can be connected directly to the system data bus. The UPI bus interface has sufficient drive capability (400 μA) for small systems, however, a larger system may require buffers.

Four control signals are required to handle the data and status information transfer:

- $\overline{\text{WR}}$ I/O WRITE signal used to transfer data from the system bus to the UPI DBBIN register and set the F_1 flag in the status register.
- $\overline{\text{RD}}$ I/O READ signal used to transfer data from the DBBOUT register or status register to the system data bus.
- $\overline{\text{CS}}$ CHIP SELECT signal used to enable one 8041A out of several connected to a common bus.
- A_0 Address input used to select either the 8-bit status register or DBBOUT register during an I/O READ. Also, the signal is used to set the F_1 flag in the status register during an I/O WRITE.

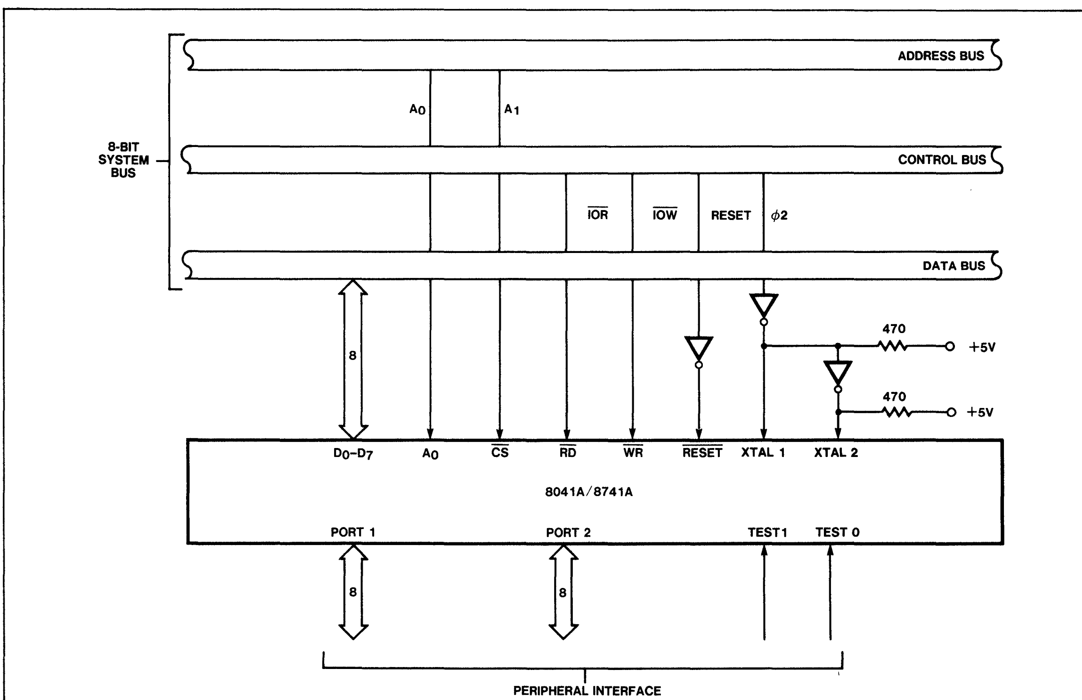


Figure 2-17. Interface to 8080 System Bus

The $\overline{WR}$ and $\overline{RD}$ signals are active low and are standard MCS-80 peripheral control signals used to synchronize data transfer between the system bus and peripheral devices.

The $\overline{CS}$ and A_0 signals are decoded from the address bus of the master system. In a system with few I/O devices a linear addressing configuration can be used where A_0 and A_1 lines are connected directly to A_0 and $\overline{CS}$ inputs (see Figure 2-17).

Data Read

Table 2-4 illustrates the relative timing of a DBBOUT Read. When $\overline{CS}$, A_0, and $\overline{RD}$ are low, the contents of the DBBOUT register is placed on the three-state Data lines D_0–D_7 and the OBF flag is cleared.

The master processor uses $\overline{CS}$, A_0, $\overline{WR}$, and $\overline{RD}$ to control data transfer between the DBBOUT register and the master system. The following operations are under master processor control:

Table 2-4. Data Transfer Controls

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_0	
0	0	1	0	Read DBBOUT register
0	0	1	1	Read STATUS register
0	1	0	0	Write DBBIN data register
0	1	0	1	Write DBBIN command register
1	x	x	x	Disable DBB

Status Read

Table 2-4 shows the logic sequence required for a STATUS register read. When $\overline{CS}$ and $\overline{RD}$ are low with A_0 high, the contents of the 8-bit status register appears on Data lines D_0–D_7.

Data Write

Table 2-4 shows the sequence for writing information to the DBBIN register. When $\overline{CS}$ and $\overline{WR}$ are low, the contents of the system data bus is latched into DBBIN. Also, the IBF flag is set and an interrupt is generated, if enabled.

Command Write

During any write (Table 2-4), the state of the A_0 input is latched into the status register in the F_1 (command/data) flag location. This additional bit is used to signal whether DBBIN contents are command ($A_0 = 1$) or data ($A_0 = 0$) information.

INPUT/OUTPUT INTERFACE

The UPI-41A has 16 lines for input and output functions. These I/O lines are grouped as two 8-bit TTL compatible ports: PORTS 1 and 2. The port lines can individually function as either inputs or outputs under software control. In addition, the lower 4 lines of PORT 2 can be used to interface to an 8243 I/O expander device to increase I/O capacity to 28 or more lines. The additional lines are grouped as 4-bit ports: PORTS 4, 5, 6, and 7.

PORTS 1 and 2

PORTS 1 and 2 are each 8 bits wide and have the same I/O characteristics. Data written to these ports by an OUTL Pp,A instruction is latched and remains unchanged until it is rewritten. Input data is sampled at the time the IN, A,Pp instruction is executed. Therefore, input data must be present at the PORT until read by an INput instruction. PORT 1 and 2 inputs are fully TTL compatible and outputs will drive one standard TTL load.

Circuit Configuration

The PORT 1 and 2 lines have a special output structure (shown in Figure 2-18) that allows each line to serve as an input, an output, or both, even though outputs are statically latched.

Each line has a permanent high impedance pull-up (50KΩ) which is sufficient to provide source current for a TTL high level, yet can be pulled low by a standard TTL gate drive. Whenever a "1" is written to a line, a low impedance pull-up (5K) is switched in momentarily (500 ns) to provide a fast transition from 0 to 1. When a "0" is written to the line, a low impedance pull-down (300Ω) is active to provide TTL current sinking capability.

To use a particular PORT pin as an input, a logic "1" must first be written to that pin.

NOTE: A $\overline{RESET}$ intializes all PORT pins to the high impedance logic "1" state.

An external TTL device connected to the pin has sufficient current sinking capability to pull-down the pin to the low state. An IN A,Pp instruction will sample the status of PORT pin and will input the proper logic level. With no external input connected, the IN A,Pp instruction inputs the previous output status.

This structure allows input and output information on the same pin and also allows any mix of input and output lines on the same port. However, when inputs and outputs are mixed on one PORT, a PORT write will cause the strong internal pull-ups to turn on at all inputs. If a switch or other low impedance device is connected to an input, a PORT write ("1" to an input) could cause current limits on internal lines to

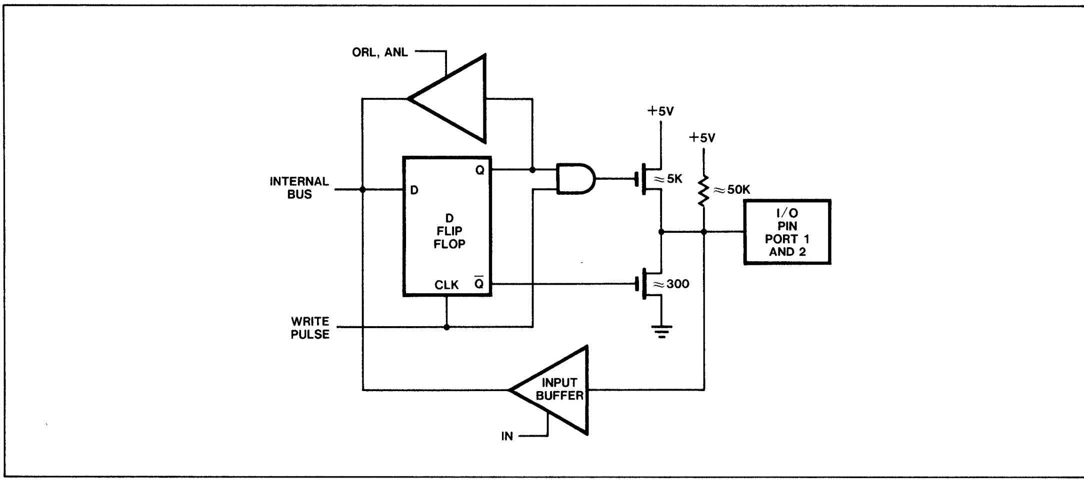


Figure 2-18. Quasi-Bidirectional Port Structure

be exceeded. Figure 2-19 illustrates the recommended connection when inputs and outputs are mixed on one PORT.

The bidirectional port structure in combination with the UPI-41A logical AND and OR instructions provides an efficient means for handling single line inputs and outputs within an 8-bit processor.

PORTS 4, 5, 6, and 7

By using an 8243 I/O expander, 16 additional I/O lines can be connected to the UPI-41A and directly addressed as 4-bit I/O ports using UPI-41A instructions. This feature saves program space and design time, and improves the bit handling capability of the UPI-41A.

The lower half of PORT 2 provides an interface to the 8243 as illustrated in Figure 2-20. The PROG pin is used as a strobe to clock address and data information via the PORT 2 interface. The extra 16 I/O lines are referred to in UPI software as PORTS 4, 5, 6, and 7. Each PORT can be directly addressed and can be ANDed and ORed with an immediate data mask. Data can be moved directly to the accumulator from the expander PORTS (or vice-versa).

The 8243 I/O ports, PORTS 4, 5, 6, and 7, provide more drive capability than the UPI-41A bidirectional ports. The 8243 output is capable of driving about 5 standard TTL loads.

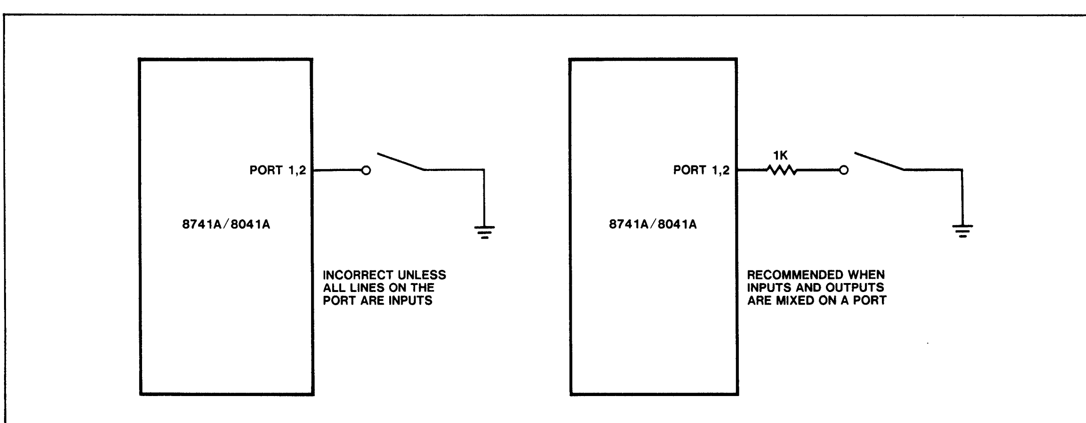


Figure 2-19. Recommended PORT Input Connections

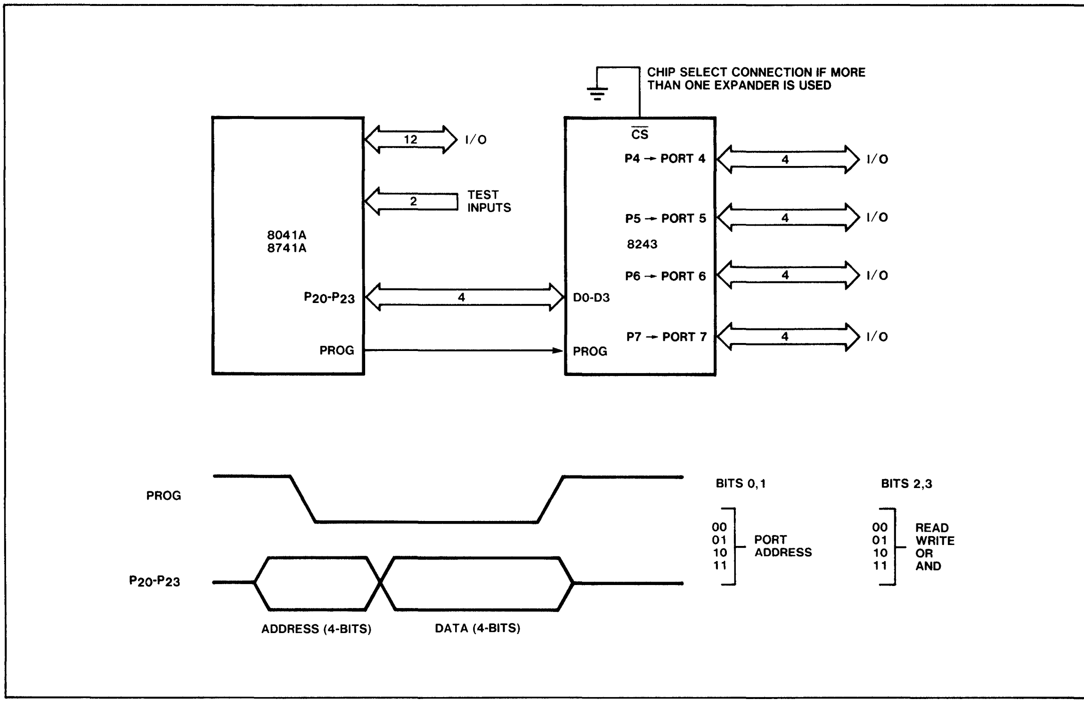


Figure 2-20. 8243 Expander Interface

Multiple 8243's can be connected to the PORT 2 interface. In normal operation, only one of the 8243's would be active at the time an Input or Output command is executed. The upper half of PORT 2 is used to provide chip select signals to the 8243's. Figure 2-21 shows how four 8243's could be connected. Software is needed to select and set the proper PORT 2 pin before an INPUT or OUTPUT command to PORTS 4–7 is executed. In general, the software overhead required is very minor compared to the added flexibility of having a large number of I/O pins available.

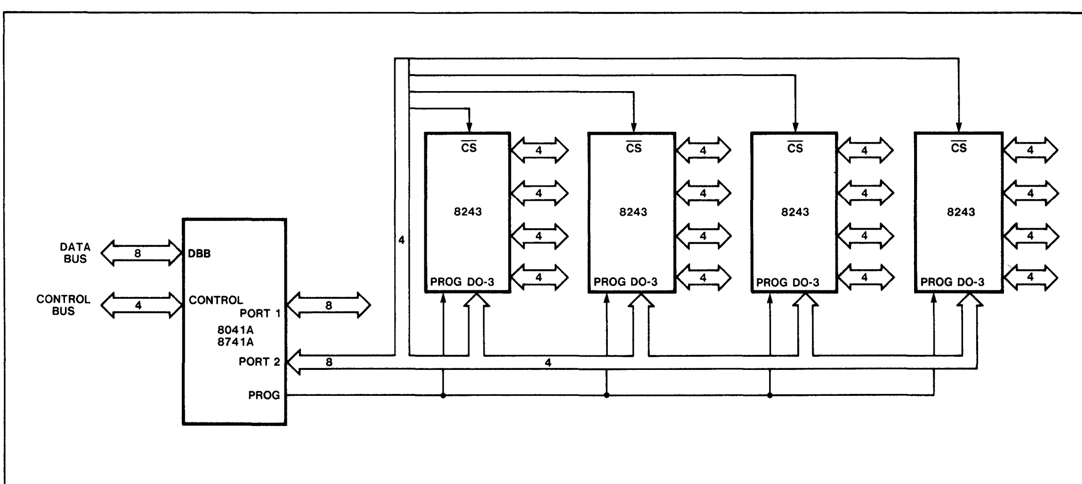


Figure 2-21. Multiple 8243 Expansion

CHAPTER 3
INSTRUCTION SET

CHAPTER 3
INSTRUCTION SET

The UPI-41A Instruction Set is opcode-compatible with the MCS-48 set except for the elimination of external program and data memory instructions and the addition of the data bus buffer instructions. It is very straightforward and efficient in its use of program memory. All instructions are either 1 or 2 bytes in length (over 70% are only 1 byte long) and over half of the instructions execute in one machine cycle. The remainder require only two cycles and include Branch, Immediate, and I/O operations.

The UPI-41A Instruction Set efficiently handles the single-bit operations required in control applications. Special instructions allow port bits to be set or cleared individually. Also, any accumulator bit can be directly tested via conditional branch instructions. Additional instructions are included to simplify loop counters, table look-up routines and N-way branch routines.

The UPI-41A Microcomputer handles arithmetic operations in both binary and BCD for efficient interface to peripherals such as keyboards and displays.

The instruction set can be divided into the following groups:

- Data Moves
- Accumulator Operations
- Flags
- Register Operations
- Branch Instructions
- Control
- Timer Operations
- Subroutines
- Input/Output Instructions

Data Moves (See Instruction Summary)

The 8-bit accumulator is the control point for all data transfers within the UPI-41A. Data can be transferred between the 8 registers of each working register bank and the accumulator directly (i.e., with a source or destination register specified by 3 bits in the instruction). The remaining locations in the RAM array are addressed either by R_0 or R_1 of the active register bank. Transfers to and from RAM require one cycle.

Constants stored in Program Memory can be loaded directly into the accumulator or the eight working registers. Data can also be transferred directly between the accumulator and the on-board timer/counter, the Status Register (STS), or the Program Status Word (PSW). Transfers to the STS register alter bits 4-7 only. Transfers to the PSW alter machine status accordingly and provide a means of restoring status after an interrupt or of altering the stack pointer if necessary.

Accumulator Operations

Immediate data, data memory, or the working registers can be added (with or without carry) to the accumulator. These sources can also be ANDed, ORed, or exclusive ORed to the accumulator. Data may be moved to or from the accumulator and working registers or data memory. The two values can also be exchanged in a single operation.

The lower 4 bits of the accumulator can be exchanged with the lower 4 bits of any of the internal RAM locations. This operation, along with an instruction which swaps the upper and lower 4-bit halves of the accumulator, provides easy handling of BCD numbers and other 4-bit quantities. To facilitate BCD arithmetic a Decimal Adjust instruction is also included. This instruction is used to correct the result of the binary addition of two 2-digit BCD numbers. Performing a decimal adjust on the result in the accumulator produces the desired BCD result.

The accumulator can be incremented, decremented, cleared, or complemented and can be rotated left or right 1 bit at a time with or without carry.

A subtract operation can be easily implemented in UPI-41A software using three single-byte, single-cycle instructions. A value can be subtracted from the accumulator by using the following instructions:

- Complement the accumulator
- Add the value to the accumulator
- Complement the accumulator

Flags

There are four user accessible flags:

- Carry
- Auxiliary Carry
- F_0
- F_1

The Carry flag indicates overflow of the accumulator, while the Auxiliary Carry flag indicates overflow between BCD digits and is used during decimal adjust operations. Both Carry and Auxiliary Carry are part of the Program Status Word (PSW) and are stored in the stack during subroutine calls. The F_0 and F_1 flags are general-purpose flags which can be cleared or complemented by UPI instructions. F_0 is accessible via the Program Status Word and is stored in the stack with the Carry flags. F_1 reflects the condition of the A_0 line, and caution must be used when setting or clearing it.

Register Operations

The working registers can be accessed via the accumulator as explained above, or they can be loaded with immediate data constants from program memory. In addition, they can be incremented or decremented directly, or they can be used as loop counters as explained in the section on branch instructions.

Additional Data Memory locations can be accessed with indirect instructions via R_0 and R_1.

Branch Instructions

The UPI-41A Instruction Set includes 17 jump instructions. The unconditional jump instruction allows jumps anywhere in the 1K words of program memory. All other jump instructions are limited to the current page (256 words) of program memory.

Conditional jump instructions can test the following inputs and machine flags:

- TEST 0 input pin
- TEST 1 input pin
- Input Buffer Full flag
- Output Buffer Full flag
- Timer flag
- Accumulator zero
- Accumulator bit
- Carry flag
- F_0 flag
- F_1 flag

The conditions tested by these instructions are the instantaneous values at the time the conditional jump instruction is executed. For instance, the jump on accumulator zero instruction tests the accumulator itself, not an intermediate flag.

The decrement register and jump if not zero (DJNZ) instruction combines decrement and branch operations in a single instruction which is useful in implementing a loop counter. This instruction can designate any of the 8 working registers as a counter and can effect a branch to any address within the current page of execution.

A special indirect jump instruction (JMPP @A) allows the program to be vectored to any one of several different locations based on the contents of the accumulator. The contents of the accumulator point to a location in program memory which contains the jump address. As an example, this instruction could be used to vector to any one of several routines based on an ASCII character which has been loaded into the accumulator. In this way, ASCII inputs can be used to initiate various routines.

Control

The UPI-41A Instruction Set has six instructions for control of the DMA, interrupts, and selection of working register banks.

The UPI-41A provides two instructions for control of the external microcomputer system. IBF and OBF flags can be routed to PORT 2 allowing interrupts of the external processor. DMA handshaking signals can also be enabled using lines from PORT 2.

The IBF interrupt can be enabled and disabled using two instructions. Also, the interrupt is automatically disabled following a RESET input or during an interrupt service routine.

The working register bank switch instructions allow the programmer to immediately substitute a second 8 register bank for the one in use. This effectively provides either 16 working registers or the means for quickly saving the contents of the first 8 registers in response to an interrupt. The user has the option of switching register banks when an interrupt occurs. However, if the banks are switched, the original bank will automatically be restored upon execution of a return and restore status (RETR) instruction at the end of the interrupt service routine.

Timer

The 8-bit on-board timer/counter can be loaded or read via the accumulator while the counter is stopped or while counting.

The counter can be started as a timer with an internal clock source or as an event counter or timer with an external clock applied to the TEST 1 pin. The instruction executed determines which clock source is used. A single instruction stops the counter whether it is operating with an internal or an external clock source. In addition, two instructions allow the timer interrupt to be enabled or disabled.

Subroutines

Subroutines are entered by executing a call instruction. Calls can be made to any address in the 1K word program memory. Two separate return instructions determine whether or not status (i.e., the upper 4 bits of the PSW) is restored upon return from a subroutine.

Input/Output Instructions

Two 8-bit data bus buffer registers (DBBIN and DBBOUT) and an 8-bit status register (STS) enable the UPI-41A universal peripheral interface to communicate with the external microcomputer system. Data can be INputted from the DBBIN register to the accumulator. Data can be OUTputted from the accumulator to the DBBOUT register.

The STS register contains four user-definable bits (ST_4–ST_7) plus four reserved status bits (IBF, OBF, F_0, and F_1). The user-definable bits are set from the accumulator.

The UPI-41A peripheral interface has two 8-bit static I/O ports which can be loaded to and from the accumulator. Outputs are statically latched but inputs to the ports are sampled at the time an IN instruction is executed. In addition, immediate data from program memory can be ANDed and ORed directly to PORTS 1 and 2 with the result remaining on the port. This allows "masks" stored in program memory to be used to set or reset individual bits on the I/O ports. PORTS 1 and 2 are configured to allow input on a given pin by first writing a "1" to the pin.

Four additional 4-bit ports are available through the 8243 I/O expander device. The 8243 interfaces to the UPI-41A peripheral interface via four PORT 2 lines which form an expander bus. The 8243 ports have their own AND and OR instructions like the on-board ports, as well as move instructions to transfer data in or out. The expander AND or OR instructions, however, combine the contents of the accumulator with the selected port rather than with immediate data as is done with the on-board ports.

Instruction Set Description

The following section provides a detailed description of each UPI instruction and illustrates how the instructions are used.

For further information about programming the UPI, consult the *8048/8041A Assembly Language Manual.*

Table 3-1. Symbols and Abbreviations Used

Symbol	Definition
A	Accumulator
C	Carry
DBBIN	Data Bus Buffer Input
DBBOUT	Data Bus Buffer Output
F_0, F_1	FLAG 0, FLAG 1 (C/D flag)
I	Interrupt
P	Mnemonic for "in-page" operation
PC	Program Counter
Pp	Port designator (p = 1, 2, or 4–7)
PSW	Program Status Word
Rr	Register designator (r = 0–7)
SP	Stack Pointer
STS	Status register
T	Timer
TF	Timer Flag
T_0, T_1	TEST 0, TEST 1
#	Immediate data prefix
@	Indirect address prefix
(( ))	Double parentheses show the effect of @, that is, @RO is shown as ((RO)).
( )	Contents of

Table 3-2. Instruction Set Summary

Mnemonic		Operation Description	Bytes	Cycles
Accumulator				
ADD	A,Rr	Add register to A	1	1
ADD	A,@Rr	Add data memory to A	1	1
ADD	A,#data	Add immediate to A	2	2
ADDC	A,Rr	Add register to A with carry	1	1
ADDC	A,@Rr	Add data memory to A with carry	1	1
ADDC	A,#data	Add immediate to A with carry	2	2
ANL	A,Rr	And register to A	1	1
ANL	A,@Rr	And data memory to A	1	1
ANL	A,#data	And immediate to A	2	2
ORL	A,Rr	Or register to A	1	1
ORL	A,@Rr	Or data memory to A	1	1
ORL	A,#data	Or immediate to A	2	2
XRL	A,Rr	Exclusive Or register to A	1	1
XRL	A,@Rr	Exclusive Or data memory to A	1	1
XRL	A,#data	Exclusive Or immediate to A	2	2
INC	A	Increment A	1	1
DEC	A	Decrement A	1	1
CLR	A	Clear A	1	1
CPL	A	Complement A	1	1
DA	A	Decimal Adjust A	1	1
SWAP	A	Swap nibbles of A	1	1
RL	A	Rotate A left	1	1
RLC	A	Rotate A left through carry	1	1
RR	A	Rotate A right	1	1
RRC	A	Rotate A right through carry	1	1

Table 3-2. Instruction Set Summary (Con't.)

Mnemonic		Operation Description	Bytes	Cycles
INPUT/OUTPUT				
IN	A,Pp	Input port to A	1	2
OUTL	Pp,A	Output A to port	1	2
ANL	Pp,#data	And immediate to port	2	2
ORL	Pp,#data	Or immediate to port	2	2
IN	A,DBB	Input DBB to A, clear IBF	1	1
OUT	DBB,A	Output A to DBB, Set OBF	1	1
MOV	STS,A	A_4–A_7 to bits 4–7 of status	1	1
MOVD	A,Pp	Input Expander port to A	1	2
MOVD	Pp,A	Output A to Expander port	1	2
ANLD	Pp,A	And A to Expander port	1	2
ORLD	Pp,A	Or A to Expander port	1	2
DATA MOVES				
MOV	A,Rr	Move register to A	1	1
MOV	A,@Rr	Move data memory to A	1	1
MOV	A,#data	Move immediate to A	2	2
MOV	Rr,A	Move A to register	1	1
MOV	@Rr,A	Move A to data memory	1	1
MOV	Rr,#data	Move immediate to register	2	2
MOV	@Rr,#data	Move immediate to data memory	2	2
MOV	A,PSW	Move PSW to A	1	1
MOV	PSW,A	Move A to PSW	1	1
XCH	A,Rr	Exchange A and registers	1	1
XCH	A,@Rr	Exchange A and data memory	1	1
XCHD	A,@Rr	Exchange digit of A and register	1	1
MOVP	A,@A	Move to A from current page	1	2
MOVP3	A,@A	Move to A from Page 3	1	2
TIMER/COUNTER				
MOV	A,T	Read Timer/Counter	1	1
MOV	T,A	Load Timer/Counter	1	1
STRT	T	Start Timer	1	1
STRT	CNT	Start Counter	1	1
STOP	TCNT	Stop Timer/Counter	1	1
EN	TCNTI	Enable Timer/Counter Interrupt	1	1
DIS	TCNTI	Disable Timer/Counter Interrupt	1	1
CONTROL				
EN	DMA	Enable DMA Handshake Lines	1	1
EN	I	Enable IBF interrupt	1	1
DIS	I	Disable IBF interrupt	1	1
EN	FLAGS	Enable Master Interrupts	1	1
SEL	RB0	Select register bank 0	1	1
SEL	RB1	Select register bank 1	1	1
NOP		No Operation	1	1
REGISTERS				
INC	Rr	Increment register	1	1
INC	@Rr	Increment data memory	1	1
DEC	Rr	Decrement register	1	1
SUBROUTINE				
CALL	addr	Jump to subroutine	2	2
RET		Return	1	2
RETR		Return and restore status	1	2
FLAGS				
CLR C		Clear Carry	1	1
CPL C		Complement Carry	1	1
CLR F0		Clear Flag 0	1	1
CPL F0		Complement Flag 0	1	1
CLR F1		Clear F_1 Flag	1	1
CPL F1		Complement F_1 Flag	1	1

Table 3-2. Instruction Set Summary (Con't.)

Mnemonic		Operation Description	Bytes	Cycles
BRANCH				
JMP	addr	Jump unconditional	2	2
JMPP	@A	Jump indirect	1	2
DJNZ	Rr,addr	Decrement register and jump on non-zero	2	2
JC	addr	Jump on Carry=1	2	2
JNC	addr	Jump on Carry=0	2	2
JZ	addr	Jump on A Zero	2	2
JNZ	addr	Jump on A not Zero	2	2
JT0	addr	Jump on T_0=1	2	2
JNT0	addr	Jump on T_0=0	2	2
JT1	addr	Jump on T_1=1	2	2
JNT1	addr	Jump on T_1=0	2	2
JF0	addr	Jump on F_0 Flag=1	2	2
JF1	addr	Jump on F_1 Flag=1	2	2
JTF	addr	Jump on Timer Flag=1	2	2
JNIBF	addr	Jump on IBF Flag=0	2	2
JOBF	addr	Jump on OBF Flag=1	2	2
JBb	addr	Jump on Accumulator Bit	2	2

ALPHABETIC LISTING

ADD A,Rr Add Register Contents to Accumulator

Opcode:

0	1	1	0	1	r_2	r_1	r_0

The contents of register 'r' are added to the accumulator. Carry is affected.

(A) ← (A) + (Rr) r=0–7

Example:

```
ADDREG: ADD A,R6        ;ADD REG 6 CONTENTS
                        ;TO ACC
```

ADD A,@Rr Add Data Memory Contents to Accumulator

Opcode:

0	1	1	0	0	0	0	r

The contents of the standard data memory location addressed by register 'r' bits 0–5 are added to the accumulator. Carry is affected.

(A) ← (A) + ((Rr)) r=0–1

Example:

```
ADDM: MOV RO,#47        ;MOVE 47 DECIMAL TO REG 0
      ADD A,@RO         ;ADD VALUE OF LOCATION
                        ;47 TO ACC
```

ADD A,#data Add Immediate Data to Accumulator

Opcode:

0	0	0	0	0	0	1	1	•	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

This is a 2-cycle instruction. The specified data is added to the accumulator. Carry is affected.

(A) ← (A) + data

Example:

```
ADDID: ADD A,#ADDER     ;ADD VALUE OF SYMBOL
                        ;'ADDER' TO ACC
```

ADDC A,Rr Add Carry and Register Contents to Accumulator

Opcode:

0	1	1	1	1	r_2	r_1	r_0

The content of the carry bit is added to accumulator location 0. The contents of register 'r' are then added to the accumulator. Carry is affected.

(A) ← (A) + (Rr) + (C)   r=0–7

Example:

```
ADDRGC: ADDC A,R4        ;ADD CARRY AND REG 4
                         ;CONTENTS TO ACC
```

ADDC A,@Rr Add Carry and Data Memory Contents to Accumulator

Opcode:

0	1	1	1	0	0	0	r

The content of the carry bit is added to accumulator location 0. Then the contents of the standard data memory location addressed by register 'r' bits 0–5 are added to the accumulator. Carry is affected.

(A) ← (A) + ((Rr)) + (C)   r=0–1

Example:

```
ADDMC: MOV R1,#40        ;MOV '40' DEC TO REG 1
       ADDC A,@R1        ;ADD CARRY AND LOCATION 40
                         ;CONTENTS TO ACC
```

ADDC A,#data Add Carry and Immediate Data to Accumulator

Opcode:

0	0	0	1	0	0	1	1	•	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

This is a 2-cycle instruction. The content of the carry bit is added to accumulator location 0. Then the specified data is added to the accumulator. Carry is affected.

(A) ← (A) + data + (C)

Example:

```
ADDC A,#255              ;ADD CARRY AND '225' DEC
                         ;TO ACC
```

ANL A,Rr Logical AND Accumulator With Register Mask

Opcode:

0	1	0	1	1	r_2	r_1	r_0

Data in the accumulator is logically ANDed with the mask contained in working register 'r'.

(A) ← (A) AND (Rr)   r=0–7

Example:

```
ANDREG: ANL A,R3         ;'AND' ACC CONTENTS WITH MASK
                         ;MASK IN REG 3
```

ANL A,@Rr Logical AND Accumulator With Memory Mask

Opcode:

0	1	0	1	0	0	0	r

Data in the accumulator is logically ANDed with the mask contained in the data memory location referenced by register 'r', bits 0–5.

(A) ← (A) AND ((Rr))   r=0–1

Example:

```
ANDDM: MOV R0,#0FFH      ;MOVE 'FF' HEX TO REG 0
       ANL A,#0AFH       ;'AND' ACC CONTENTS WITH
                         ;MASK IN LOCATION 63
```

ANL A,#data Logical AND Accumulator With Immediate Mask

Opcode:

0	1	0	1	0	0	1	1	•	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

This is a 2-cycle instruction. Data in the accumulator is logically ANDed with an immediately-specified mask.
(A) ← (A) AND data

Example:

```
ANDID: ANL A,#0AFH       ;'AND' ACC CONTENTS
                         :WITH MASK 10101111
       ANL A,#3+X/Y      ;'AND' ACC CONTENTS
                         ;WITH VALUE OF EXP
                         ;'3+X/Y'
```

ANL Pp,#data Logical AND Port 1–2 With Immediate Mask

Opcode:

1	0	0	1	1	0	p_1	p_0	•	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

This is a 2-cycle instruction. Data on port 'p' is logically ANDed with an immediately-specified mask.
(Pp) ← (Pp) AND data p=1−2

Note: Bits 0-1 of the opcode are used to represent PORT 1 and PORT 2. If you are coding in binary rather than assembly language, the mapping is as follows:

Bits	p1	p0	Port
	0	0	X
	0	1	1
	1	0	2
	1	1	X

Example:

```
ANDP2: ANL P2,#0F0H      ;'AND' PORT 2 CONTENTS
                         ;WITH MASK 'F0' HEX
                         ;(CLEAR P20-23)
```

ANLD Pp,A Logical AND Port 4–7 With Accumulator Mask

Opcode:

1	0	0	1	1	1	p_1	p_0

This is a 2-cycle instruction. Data on port 'p' on the 8243 expander is logically ANDed with the digit mask contained in accumulator bits 0–3.
(Pp) ← (Pp) AND (A0−3) p=4−7

Note: The mapping of Port 'p' to opcode bits p_1,p_0 is as follows:

p1	p0	Port
0	0	4
0	1	5
1	0	6
1	1	7

Example:

```
ANDP4: ANLD P4,A         ;'AND' PORT 4 CONTENTS
                         ;WITH ACC BITS 0-3
```

CALL address Subroutine Call

Opcode:

0	a_9	a_8	1	0	1	0	0	•	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

This is a 2-cycle instruction. The program counter and PSW bits 4–7 are saved in the stack. The stack pointer (PSW bits 0–2) is updated. Program control is then passed to the location specified by 'address'.

Execution continues at the instruction following the CALL upon return from the subroutine.
$((SP)) \leftarrow (PC), (PSW_{4-7})$
$(SP) \leftarrow (SP) + 1$
$(PC_{8-9}) \leftarrow (addr_{8-9})$
$(PC_{0-7}) \leftarrow (addr_{0-7})$

Example: Add three groups of two numbers. Put subtotals in locations 50, 51 and total in location 52.

```
         MOV R0,#50       ;MOVE '50' DEC TO ADDRESS
                          ;REG 0
BEGADD:  MOV A,R1         ;MOVE CONTENTS OF REG 1
                          ;TO ACC
         ADD A,R2         ;ADD REG 2 TO ACC
         CALL SUBTOT      ;CALL SUBROUTINE 'SUBTOT'
         ADD A,R3         ;ADD REG 3 TO ACC
         ADD A,R4         ;ADD REG 4 TO ACC
         CALL SUBTOT      ;CALL SUBROUTINE 'SUBTOT'
         ADD A,R5         ;ADD REG 5 TO ACC
         ADD A,R6         ;ADD REG 6 TO ACC
         CALL SUBTOT      ;CALL SUBROUTINE 'SUBTOT'
              .
              .
              .
SUBTOT:  MOV @R0,A        ;MOVE CONTENTS OF ACC TO
                          ;LOCATION ADDRESSED BY
                          ;REG 0
         INC R0           ;INCREMENT REG 0
         RET              ;RETURN TO MAIN PROGRAM
```

CLR A Clear Accumulator

Opcode:

0	0	1	0	0	1	1	1

The contents of the accumulator are cleared to zero.
$(A) \leftarrow 00H$

CLR C Clear Carry Bit

Opcode:

1	0	0	1	0	1	1	1

During normal program execution, the carry bit can be set to one by the ADD, ADDC, RLC, CPLC, RRC, and DAA instructions. This instruction resets the carry bit to zero.
$(C) \leftarrow 0$

CLR F1 Clear Flag 1

Opcode:

1	0	1	0	0	1	0	1

The F_1 flag is cleared to zero.
$(F_1) \leftarrow 0$

CLR F0 Clear Flag 0

Opcode:

1 0 0 0	0 1 0 1

Flag 0 is cleared to zero.
$(F_0) \leftarrow 0$

CPL A Complement Accumulator

Opcode:

0 0 1 1	0 1 1 1

The contents of the accumulator are complemented. This is strictly a one's complement. Each one is changed to zero and vice-versa.
(A) ← NOT (A)

Example: Assume accumulator contains 01101010.

```
CPLA: CPL A                ;ACC CONTENTS ARE COMPLE-
                           ;MENTED TO 10010101
```

CPL C Complement Carry Bit

Opcode:

1 0 1 0	0 1 1 1

The setting of the carry bit is complemented; one is changed to zero, and zero is changed to one.
(C) ← NOT (C)

Example: Set C to one; current setting is unknown.

```
CT01: CLR C                ;C IS CLEARED TO ZERO
      CPL C                ;C IS SET TO ONE
```

CPL F0 Complement Flag 0

Opcode:

1 0 0 1	0 1 0 1

The setting of Flag 0 is complemented; one is changed to zero, and zero is changed to one.
$F_0 \leftarrow$ NOT (F0)

CPL F1 Complement Flag 1

Opcode:

1 0 1 1	0 1 0 1

The setting of the F_1 Flag is complemented; one is changed to zero, and zero is changed to one.
$(F_1) \leftarrow$ NOT (F_1)

DA A Decimal Adjust Accumulator

Opcode:

0	1	0	1	0	1	1	1

The 8-bit accumulator value is adjusted to form two 4-bit Binary Coded Decimal (BCD) digits following the binary addition of BCD numbers. The carry bit C is affected. If the contents of bits 0–3 are greater than nine, or if AC is one, the accumulator is incremented by six.

The four high-order bits are then checked. If bits 4–7 exceed nine, or if C is one, these bits are increased by six. If an overflow occurs, C is set to one; otherwise, it is cleared to zero.

Example: Assume accumulator contains 9AH.

```
DA  A   ;ACC ADJUSTED TO 01H with C set

C   AC  ACC
0   0   9AH   INITIAL CONTENTS
        06H   ADD SIX TO LOW DIGIT
0   0   A1H
        60H   ADD SIX TO HIGH DIGIT
1   0   01H   RESULT
```

DEC A Decrement Accumulator

Opcode:

0	0	0	0	0	1	1	1

The contents of the accumulator are decremented by one.

(A) ← (A) − 1

Example: Decrement contents of data memory location 63.

```
MOV R0,#3FH      ;MOVE '3F' HEX TO REG 0
MOV A,@R0        ;MOVE CONTENTS OF LOCATION 63
                 ;TO ACC
DEC A            ;DECREMENT ACC
MOV @R0,A        ;MOVE CONTENTS OF ACC TO
                 ;LOCATION 63
```

DEC Rr Decrement Register

Opcode:

1	1	0	0	1	r_2	r_1	r_0

The contents of working register 'r' are decremented by one.

(Rr) ← (Rr) − 1   r=0−7

Example:

```
DECR1: DEC R1    ;DECREMENT ADDRESS REG 1
```

DIS I Disable IBF Interrupt

Opcode:

0	0	0	1	0	1	0	1

The input Buffer Full interrupt is disabled. The interrupt sequence is not initiated by $\overline{WR}$ and $\overline{CS}$, however, an IBF interrupt request is latched and remains pending until an EN I (enable IBF interrupt) instruction is executed.

Note: The IBF flag is set and cleared independent of the IBF interrupt request so that handshaking protocol can continue normally.

DIS TCNTI Disable Timer/Counter Interrupt

Opcode:

0	0	1	1	0	1	0	1

The timer/counter interrupt is disabled. Any pending timer interrupt request is cleared. The interrupt sequence is not initiated by an overflow, but the timer flag is set and time accumulation continues.

DJNZ Rr, address Decrement Register and Test

Opcode:

1	1	1	0	1	r_2	r_1	r_0	•	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

This is a 2-cycle instruction. Register 'r' is decremented and tested for zero. If the register contains all zeros, program control falls through to the next instruction. If the register contents are not zero, control jumps to the specified address within the current page.

(Rr) ← (Rr) − 1

If R ≠ 0, then;

(PC_{0-7}) ← addr

Note: A 10-bit address specification does not cause an error if the DJNZ instruction and the jump target are on the same page. If the DJNZ instruction begins in location 255 of a page, it will jump to a target address on the following page. Otherwise, it is limited to a jump within the current page.

Example: Increment values in data memory locations 50–54.

```
        MOV R0,#50          ;MOVE '50' DEC TO ADDRESS
                            ;REG 0
        MOV R3,#05          ;MOVE '5' DEC TO COUNTER
                            ;REG 3
INCRT:  INC @R0             ;INCREMENT CONTENTS OF
                            ;LOCATION ADDRESSED BY
                            ;REG 0
        INC R0              ;INCREMENT ADDRESS IN REG 0
        DJNZ R3,INCRT       ;DECREMENT REG 3——JUMP TO
                            ;'INCRT' IF REG 3 NONZERO
        NEXT——              ;'NEXT' ROUTINE EXECUTED
                            ;IF R3 IS ZERO
```

EN DMA Enable DMA Handshake Lines

Opcode:

1	1	1	0	0	1	0	1

DMA handshaking is enabled using P_{26} as DMA request (DRQ) and P_{27} as DMA acknowledge ($\overline{DACK}$). The $\overline{DACK}$ line forces $\overline{CS}$ and A_0 low internally and clears DRQ.

EN FLAGS Enable Master Interrupts

Opcode:

1	1	1	1	0	1	0	1

The Output Buffer Full (OBF) and the Input Buffer Full (IBF) flags (IBF is inverted) are routed to P_{24} and P_{25}. For proper operation, a "1" should be written to P_{25} and P_{24} before the EN FLAGS instruction. A "0" written to P_{24} or P_{25} disables the pin.

EN I Enable IBF Interrupt

Opcode:

0	0	0	0	0	1	0	1

The Input Buffer Full interrupt is enabled. A low signal on $\overline{WR}$ and $\overline{CS}$ initiates the interrupt sequence.

EN TCNTI Enable Timer/Counter Interrupt

Opcode:

0	0	1	0	0	1	0	1

The timer/counter interrupt is enabled. An overflow of this register initiates the interrupt sequence.

IN A,DBB Input Data Bus Buffer Contents to Accumulator

Opcode:

0	0	1	0	0	0	1	0

Data in the DBBIN register is transferred to the accumulator and the Input Buffer Full (IBF) flag is set to zero.
(A) ← (DBB)
(IBF) ← 0

Example:

```
INDBB: IN A,DBB          ;INPUT DBBIN CONTENTS TO
                         ;ACCUMULATOR
```

IN A,Pp Input Port 1–2 Data to Accumulator

Opcode:

0	0	0	0	1	0	p_1	p_0

This is a 2-cycle instruction. Data present on port 'p' is transferred (read) to the accumulator.
(A) ← (Pp) p=1–2 (see ANL instruction)

Example:

```
INP12: IN A,P1           ;INPUT PORT 1 CONTENTS
                         ;TO ACC
       MOV R6,A          ;MOVE ACC CONTENTS TO
                         ;REG 6
       IN A,P2           ;INPUT PORT 2 CONTENTS
                         ;TO ACC
       MOV R7,A          ;MOVE ACC CONTENTS TO REG 7
```

INC A Increment Accumulator

Opcode:

0	0	0	1	0	1	1	1

The contents of the accumulator are incremented by one.
(A) ← (A) + 1

Example: Increment contents of location 10 in data memory.

```
INCA: MOV R0,#10         ;MOV '10' DEC TO ADDRESS
                         ;REG 0
      MOV A,@R0          ;MOVE CONTENTS OF LOCATION
                         ;10 TO ACC
      INC A              ;INCREMENT ACC
      MOV @R0,A          ;MOVE ACC CONTENTS TO
                         ;LOCATION 10
```

INC Rr Increment Register

Opcode:

0	0	0	1	1	r_2	r_1	r_0

The contents of working register 'r' are incremented by one.

(Rr) ← (Rr) + 1 r=0–7

Example: INCR0: INC R0 ;INCREMENT ADDRESS REG 0

INC @Rr Increment Data Memory Location

Opcode:

0	0	0	1	0	0	0	r

The contents of the resident data memory location addressed by register 'r' bits 0–5 are incremented by one.

((Rr)) ← ((Rr)) + 1 r=0–1

Example:
```
INCDM: MOV R1,#OFFH     ;MOVE ONES TO REG 1
       INC @R1          ;INCREMENT LOCATION 63
```

JBb address Jump If Accumulator Bit is Set

Opcode:

b_2	b_1	b_0	1	0	0	1	0	•	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

This is a 2-cycle instruction. Control passes to the specified address if accumulator bit 'b' is set to one.

(PC_{0-7}) ← addr if b=1

(PC) ← (PC) + 2 if b=0

Example:
```
JB4IS1: JB4 NEXT        ;JUMP TO 'NEXT' ROUTINE
                        ;IF ACC BIT 4=1
```

JC address Jump If Carry Is Set

Opcode:

1	1	1	1	0	1	1	0	•	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

This is a 2-cycle instruction. Control passes to the specified address if the carry bit is set to one.

(PC_{0-7}) ← addr if C=1

(PC) ← (PC) + 2 if C=0

Example:
```
JC1: JC OVERFLOW        ;JUMP TO 'OVFLOW' ROUTINE
                        ;IF C=1
```

JF0 address Jump If Flag 0 Is Set

Opcode:

1	0	1	1	0	1	1	0	•	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

This is a 2-cycle instruction. Control passes to the specified address if flag 0 is set to one.

(PC_{0-7}) ← addr if F_0=1

Example:
```
JF0IS1: JF0 TOTAL       ;JUMP TO 'TOTAL' ROUTINE
                        ;IF F0=1
```

JF1 address Jump If C/D Flag (F1) Is Set

Opcode:

0	1	1	1	0	1	1	0	•	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

This is a 2-cycle instruction. Control passes to the specified address if the C/D flag (F_1) is set to one.

$(PC_{0-7}) \leftarrow$ addr if $F_1=1$

Example: JF 1IS1: JF1 FILBUF ;JUMP TO 'FILBUF'
;ROUTINE IF $F_1=1$

JMP address Direct Jump Within 1K Block

Opcode:

a_{10}	a_9	a_8	0	0	1	0	0	•	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

This is a 2-cycle instruction. Bits 0–9 of the program counter are replaced with the directly-specified address.

$(PC_{8-9}) \leftarrow$ addr 8–9

$(PC_{0-7}) \leftarrow$ addr 0–7

Example:

```
JMP SUBTOT        ;JUMP TO SUBROUTINE 'SUBTOT'
JMP $-6           ;JUMP TO INSTRUCTION SIX LOCATIONS
                  ;BEFORE CURRENT LOCATION
JMP 2FH           ;JUMP TO ADDRESS '2F' HEX
```

JMPP @A Indirect Jump Within Page

Opcode:

1	0	1	1	0	0	1	1

This is a 2-cycle instruction. The contents of the program memory location pointed to by the accumulator are substituted for the 'page' portion of the program counter (PC 0–7).

$(PC_{0-7}) \leftarrow ((A))$

Example: Assume accumulator contains OFH

```
JMPPAG: JMPP @A   ;JMP TO ADDRESS STORED IN
                  ;LOCATION 15 IN CURRENT PAGE
```

JNC address Jump If Carry Is Not Set

Opcode:

1	1	1	0	0	1	1	0	•	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

This is a 2-cycle instruction. Control passes to the specified address if the carry bit is not set, that is, equals zero.

$(PC_{0-7}) \leftarrow$ addr if C=0

Example:

```
JCO: JNC NOVFLO   ;JUMP TO 'NOVFLO' ROUTINE
                  ;IF C=0
```

JNIBF address Jump If Input Buffer Full Flag Is Low

Opcode:

1	1	0	1	0	1	1	0	•	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

This is a 2-cycle instruction. Control passes to the specified address if the Input Buffer Full flag is low (IBF=0).

$(PC_{0-7}) \leftarrow$ addr if IBF=0

Example:

```
LOC 3:JNIBF LOC 3  ;JUMP TO SELF IF IBF=0
                   ;OTHERWISE CONTINUE
```

JNT0 address Jump If TEST 0 Is Low

Opcode:

0 0 1 0	0 1 1 0	•	a_7 a_6 a_5 a_4	a_3 a_2 a_1 a_0

This is a 2-cycle instruction. Control passes to the specified address, if the TEST 0 signal is low. Pin is sampled during SYNC.

(PC_{0-7}) ← addr   if T_0=0

Example: JT0LOW: JNT0 60   ;JUMP TO LOCATION 60 DEC
;IF T_0=0

JNT1 address Jump If TEST 1 Is Low

Opcode:

0 1 0 0	0 1 1 0	•	a_7 a_6 a_5 a_4	a_3 a_2 a_1 a_0

This is a 2-cycle instruction. Control passes to the specified address if the TEST 1 signal is low. Pin is sampled during SYNC.

(PC_{0-7}) ← addr   if T_1=0

Example: JT1LOW: JNT1 0BBH   ;JUMP TO LOCATION 'BB' HEX
;IF T_1=0

JNZ address Jump If Accumulator Is Not Zero

Opcode:

1 0 0 1	0 1 1 0	•	a_7 a_6 a_5 a_4	a_3 a_2 a_1 a_0

This is a 2-cycle instruction. Control passes to the specified address if the accumulator contents are nonzero at the time this instruction is executed.

(PC_{0-7}) ← addr   if A≠0

Example: JACCNO: JNZ 0ABH   ;JUMP TO LOCATION 'AB' HEX
;IF ACC VALUE IS NONZERO

JOBF Address Jump If Output Buffer Full Flag Is Set

Opcode:

1 0 0 0	0 1 1 0	•	a_7 a_6 a_5 a_4	a_3 a_2 a_1 a_0

This is a 2-cycle instruction. Control passes to the specified address if the Output Buffer Full (OBF) flag is set (=1) at the time this instruction is executed.

(PC_{0-7}) ← addr   if OBF=1

Example: JOBFHI: JOBF 0AAH   ;JUMP TO LOCATION 'AA' HEX
;IF OBF=1

JTF address Jump If Timer Flag Is Set

Opcode:

0 0 0 1	0 1 1 0	•	a_7 a_6 a_5 a_4	a_3 a_2 a_1 a_0

This is a 2-cycle instruction. Control passes to the specified address if the timer flag is set to one, that is, the timer/counter register overflows to zero. The timer flag is cleared upon execution of this instruction. (This overflow initiates an interrupt service sequence if the timer-overflow interrupt is enabled.)

(PC_{0-7}) ← addr   if TF=1

Example: JTF1: JTF TIMER   ;JUMP TO 'TIMER' ROUTINE
;IF TF=1

JTO address   Jump If TEST 0 Is High

Opcode:	0 0 1 1	0 1 1 0	•	a_7 a_6 a_5 a_4	a_3 a_2 a_1 a_0

This is a 2-cycle instruction. Control passes to the specified address if the TEST 0 signal is high (=1). Pin is sampled during SYNC.

$(PC_{0-7}) \leftarrow$ addr   if $T_0=1$

Example: JTOHI: JTO 53   ;JUMP TO LOCATION 53 DEC
;IF $T_0=1$

JT1 address   Jump If TEST 1 Is High

Opcode:	0 1 0 1	0 1 1 0	•	a_7 a_6 a_5 a_4	a_3 a_2 a_1 a_0

This is a 2-cycle instruction. Control passes to the specified address if the TEST 1 signal is high (=1). Pin is sampled during SYNC.

$(PC_{0-7}) \leftarrow$ addr   if $T_1=1$

Example: JT1HI: JT1 COUNT   ;JUMP TO 'COUNT' ROUTINE
;IF $T_1=1$

JZ address   Jump If Accumulator Is Zero

Opcode:	1 1 0 0	0 1 1 0	•	a_7 a_6 a_5 a_4	a_3 a_2 a_1 a_0

This is a 2-cycle instruction. Control passes to the specified address if the accumulator contains all zeros at the time this instruction is executed.

$(PC_{0-7}) \leftarrow$ addr   if A=0

Example: JACCO: JZ OA3H   ;JUMP TO LOCATION 'A3' HEX
;IF ACC VALUE IS ZERO

MOV A,#data   Move Immediate Data to Accumulator

Opcode:	0 0 1 0	0 0 1 1	•	d_7 d_6 d_5 d_4	d_3 d_2 d_1 d_0

This is a 2-cycle instruction. The 8-bit value specified by 'data' is loaded in the accumulator.

(A) $\leftarrow$ data

Example: MOV A,#OA3H   ;MOV 'A3' HEX TO ACC

MOV A,PSW   Move PSW Contents to Accumulator

Opcode:	1 1 0 0	0 1 1 1

The contents of the program status word are moved to the accumulator.

(A) $\leftarrow$ (PSW)

Example: Jump to 'RB1SET' routine if bank switch, PSW bit 4, is set.

```
BSCHK: MOV A,PSW      ;MOV PSW CONTENTS TO ACC
       JB4 RB1 SET    ;JUMP TO 'RB1SET' IF ACC
                      ;BIT 4=1
```

MOV A, Rr Move Register Contents to Accumulator

Opcode:

1	1	1	1	1	r_2	r_1	r_0

Eight bits of data are moved from working register 'r' into the accumulator.

(A) ← (Rr) r=0–7

Example:
```
MAR: MOV A,R3              ;MOVE CONTENTS OF REG 3
                           ;TO ACC
```

MOV A,@Rr Move Data Memory Contents to Accumulator

Opcode:

1	1	1	1	0	0	0	r

The contents of the data memory location addressed by bits 0–5 of register 'r' are moved to the accumulator. Register 'r' contents are unaffected.

(A) ← ((Rr)) r=0–1

Example: Assume R1 contains 00110110.
```
MADM: MOV A,@R1            ;MOVE CONTENTS OF DATA MEM
                           ;LOCATION 54 TO ACC
```

MOV A,T Move Timer/Counter Contents to Accumulator

Opcode:

0	1	0	0	0	0	1	0

The contents of the timer/event-counter register are moved to the accumulator. The timer/event-counter is not stopped.

(A) ← (T)

Example: Jump to "EXIT" routine when timer reaches '64', that is, when bit 6 is set—assuming initialization to zero.
```
TIMCHK: MOV A,T            ;MOVE TIMER CONTENTS TO
                           ;ACC
        JB6 EXIT           ;JUMP TO 'EXIT' IF ACC BIT
                           ;6=1
```

MOV PSW,A Move Accumulator Contents to PSW

Opcode:

1	1	0	1	0	1	1	1

The contents of the accumulator are moved into the program status word. All condition bits and the stack pointer are affected by this move.

(PSW) ← (A)

Example: Move up stack pointer by two memory locations, that is, increment the pointer by one.
```
INCPTR: MOV A,PSW          ;MOVE PSW CONTENTS TO ACC
        INC A              ;INCREMENT ACC BY ONE
        MOV PSW,A          ;MOVE ACC CONTENTS TO PSW
```

MOV Rr,A Move Accumulator Contents to Register

Opcode:

1	0	1	0	1	r_2	r_1	r_0

The contents of the accumulator are moved to register 'r'.

```
(Rr) ← (A)                     r=0–7
```

Example:

```
MRA MOV R0,A                   ;MOVE CONTENTS OF ACC TO
                               ;REG 0
```

MOV Rr,#data Move Immediate Data to Register

Opcode:

1	0	1	1	1	r_2	r_1	r_0	•	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

This a 2-cycle instruction. The 8-bit value specified by 'data' is moved to register 'r'.

```
(Rr) ← data                    r=0–7
```

Example:

```
MIR4: MOV R4,#HEXTEN           ;THE VALUE OF THE SYMBOL
                               ;'HEXTEN' IS MOVED INTO
                               ;REG 4
MIR5: MOV R5;#PI*(R*R)         ;THE VALUE OF THE
                               ;EXPRESSION 'PI*(R*R)'
                               ;IS MOVED INTO REG 5
MIR6: MOV R6,#OADH             ;'AD' HEX IS MOVED INTO
                               ;REG 6
```

MOV @Rr,A Move Accumulator Contents to Data Memory

Opcode:

1	0	1	0	0	0	0	r

The contents of the accumulator are moved to the data memory location whose address is specified by bits 0–5 of register 'r'. Register 'r' contents are unaffected.

```
((Rr)) ← (A)                   r=0–1
```

Example: Assume R0 contains 11000111.

```
MDMA: MOV @R,A                 ;MOVE CONTENTS OF ACC TO
                               ;LOCATION 7 (REG)
```

MOV @Rr,#data Move Immediate Data to Data Memory

Opcode:

1	0	1	1	0	0	0	r	•	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

This is a 2-cycle instruction. The 8-bit value specified by 'data' is moved to the standard data memory location addressed by register 'r', bit 0–5.

```
((Rr)) ← data                  r=0–1
```

Example: Move the hexadecimal value AC3F to locations 62–63.

```
MIDM: MOV R0,#62               ;MOVE '62' DEC TO ADDR REGO
      MOV @RO,#OACH            ;MOVE 'AC' HEX TO LOCATION 62
      INC RO                   ;INCREMENT REG 0 TO '63'
      MOV @RO,#3FH             ;MOVE '3F' HEX TO LOCATION 63
```

MOV STS,A Move Accumulator Contents to STS Register

Opcode:

1	0	0	1	0	0	0	0

The contents of the accumulator are moved into the status register. Only bits 4-7 are affected.
$(STS_{4-7}) \leftarrow (A_{4-7})$

Example: Set ST_4–ST_7 to "1".

```
MSTS: MOV A,#0F0H          ;SET ACC
      MOV STS,A            ;MOVE TO STS
```

MOV T,A Move Accumulator Contents to Timer/Counter

Opcode:

0	1	1	0	0	0	1	0

The contents of the accumulator are moved to the timer/event-counter register.
$(T) \leftarrow (A)$

Example: Initialize and start event counter.

```
INITEC: CLR A              ;CLEAR ACC TO ZEROS
        MOV T,A            ;MOVE ZEROS TO EVENT COUNTER
        STRT CNT           ;START COUNTER
```

MOVD A,Pp Move Port 4–7 Data to Accumulator

Opcode:

0	0	0	0	1	1	p_1	p_0

This is a 2-cycle instruction. Data on 8243 port 'p' is moved (read) to accumulator bits 0–3. Accumulator bits 4–7 are zeroed.
$(A_{0-3}) \leftarrow Pp$ p=4–7
$(A_{4-7}) \leftarrow 0$

Note: Bits 0–1 of the opcode are used to represent PORTS 4–7. If you are coding in binary rather than assembly language, the mapping is as follows:

Bits p_1	p_0	Port
0	0	4
0	1	5
1	0	6
1	1	7

Example:

```
INPPT5: MOVD A,P5          ;MOVE PORT 5 DATA TO ACC
                           ;BITS 0-3, ZERO ACC BITS 4-7
```

MOVD Pp,A Move Accumulator Data to Port 4, 5, 6 and 7

Opcode:

0	0	1	1	1	1	p_1	p_0

This is a 2-cycle instruction. Data in accumulator bits 0–3 is moved (written) to 8243 port 'p'. Accumulator bits 4–7 are unaffected. (See NOTE above regarding port mapping.)
$(Pp) \leftarrow (A_{0-3})$ p=4–7

Example: Move data in accumulator to ports 4 and 5.

```
OUTP45: MOVD P4,A          ;MOVE ACC BITS 0-3 TO PORT 4
        SWAP A             ;EXCHANGE ACC BITS 0-3 AND 4-7
        MOVD P5,A          ;MOVE ACC BITS 0-3 TO PORT 5
```

MOVP A,@A Move Current Page Data to Accumulator

Opcode:

1	0	1	0	0	0	1	1

This is a 2-cycle instruction. The contents of the program memory location addressed by the accumulator are moved to the accumulator. Only bits 0–7 of the program counter are affected, limiting the program memory reference to the current page. The program counter is restored following this operation.

(A) ← ((A))

Note: This is a 1-byte, 2-cycle instruction. If it appears in location 255 of a program memory page, @A addresses a location in the following page.

Example:

```
MOV128: MOV  A,#128      ;MOVE '128' DEC TO ACC
        MOVP A,@A        ;CONTENTS OF 129TH LOCATION
                         ;IN CURRENT PAGE ARE MOVED TO
                         ;ACC
```

MOVP3 A,@A Move Page 3 Data to Accumulator

Opcode:

1	1	1	0	0	0	1	1

This is a 2-cycle instruction. The contents of the program memory location within page 3, addressed by the accumulator, are moved to the accumulator. The program counter is restored following this operation.

(A) ← ((A)) within page 3

Example: Look up ASCII equivalent of hexadecimal code in table contained at the beginning of page 3. Note that ASCII characters are designated by a 7-bit code; the eighth bit is always reset.

```
TABSCH: MOV A,#OB8H      ;MOVE 'B8' HEX TO ACC (10111000)
        ANL A,#7FH       ;LOGICAL AND ACC TO MASK BIT
                         ;7 (00111000)
        MOVP3, A,@A      ;MOVE CONTENTS OF LOCATION
                         ;'38' HEX IN PAGE 3 TO ACC
                         ;(ASCII '8')
```

Access contents of location in page 3 labelled TAB1. Assume current program location is not in page 3.

```
TABSCH: MOV A,#TAB1      ;ISOLATE BITS 0-7
                         ;OF LABEL
                         ;ADDRESS VALUE
        MOVP3 A,@A       ;MOVE CONTENT OF PAGE 3
                         ;LOCATION LABELED 'TAB1'
                         ;TO ACC
```

NOP The NOP Instruction

Opcode:

0	0	0	0	0	0	0	0

No operation is performed. Execution continues with the following instruction.

ORL A,Rr Logical OR Accumulator With Register Mask

Opcode:

0	1	0	0	1	r_2	r_1	r_0

Data in the accumulator is logically ORed with the mask contained in working register 'r'.

(A) ← (A) OR (Rr)   r=0–7

Example:

```
ORREG: ORL A,R4          ;'OR' ACC CONTENTS WITH
                         ;MASK IN REG 4
```

ORL A,@Rr Logical OR Accumulator With Memory Mask

Opcode:

0	1	0	0	0	0	0	r

Data in the accumulator is logically ORed with the mask contained in the data memory location referenced by register 'r', bits 0–5.

(A) ← (A) OR ((Rr)) r=0–1

Example:
```
ORDM: MOVE R0,#3FH       ;MOVE '3F' HEX TO REG 0
      ORL A,@R0          ;'OR' ACC CONTENTS WITH MASK
                         ;IN LOCATION 63
```

ORL A,#data Logical OR Accumulator With Immediate Mask

Opcode:

0	1	0	0	0	0	1	1	•	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

This is a 2-cycle instruction. Data in the accumulator is logically ORed with an immediately-specified mask.

(A) ← (A) OR data

Example:
```
ORID: ORL A,#'X'         ;'OR' ACC CONTENTS WITH MASK
                         ;01011000 (ASCII VALUE OF 'X')
```

ORL Pp,#data Logical OR Port 1–2 With Immediate Mask

Opcode:

1	0	0	0	1	0	p_1	p_0	•	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

This is a 2-cycle instruction. Data on port 'p' is logically ORed with an immediately-specified mask.

(Pp) ← (Pp) OR data p=1–2 (see OUTL instruction)

Example:
```
ORP1: ORL P1,#0FFH       ;'OR' PORT 1 CONTENTS WITH
                         ;MASK 'FF' HEX (SET PORT 1
                         'TO ALL ONES)
```

ORLD Pp,A Logical OR Port 4–7 With Accumulator Mask

Opcode:

1	0	0	0	1	1	p_1	p_0

This is a 2-cycle instruction. Data on 8243 port 'p' is logically ORed with the digit mask contained in accumulator bits 0–3,

(Pp) (Pp) OR (A_{0-3}) p=4–7 (See MOVD instruction)

Example:
```
ORP7: ORLD P7,A          ;'OR' PORT 7 CONTENTS
                         ;WITH ACC BITS 0-3
```

OUT DBB,A Output Accumulator Contents to Data Bus Buffer

Opcode:

0	0	0	0	0	0	1	0

Contents of the accumulator are transferred to the Data Bus Buffer Output register and the Output Buffer Full (OBF) flag is set to one.

(DBB) ← (A)

OBF ← 1

Example:
```
OUTDBB: OUT DBB,A        ;OUTPUT THE CONTENTS OF
                         ;THE ACC TO DBBOUT
```

OUTL Pp,A Output Accumulator Data to Port 1 and 2

Opcode:

0	0	1	1	1	0	p_1	p_0

This is a 2-cycle instruction. Data residing in the accumulator is transferred (written) to port 'p' and latched.
(Pp) ← (A)   P=1–2

Note: Bits 0–1 of the opcode are used to represent PORT 1 and PORT 2. If you are coding in binary rather than assembly language, the mapping is as follows:

Bits	p1	p0	Port
	0	0	X
	0	1	1
	1	0	2
	1	1	X

Example:

```
OUTLP: MOV A,R7       ;MOVE REG 7 CONTENTS TO ACC
       OUTL P2,A      ;OUTPUT ACC CONTENTS TO PORT2
       MOV A,R6       ;MOVE REG 6 CONTENTS TO ACC
       OUTL P1,A      ;OUTPUT ACC CONTENTS TO PORT 1
```

RET Return Without PSW Restore

Opcode:

1	0	0	0	0	0	1	1

This is a 2-cycle instruction. The stack pointer (PSW bits 0–2) is decremented. The program counter is then restored from the stack. PSW bits 4–7 are not restored.
(SP) ← (SP) – 1
(PC) ← ((SP))

RETR Return With PSW Restore

Opcode:

1	0	0	1	0	0	1	1

This is a 2-cycle instruction. The stack pointer is decremented. The program counter and bits 4–7 of the PSW are then restored from the stack. Note that RETR should be used to return from an interrupt, but should not be used within the interrupt service routine as it signals the end of an interrupt routine.
(SP) ← (SP) – 1
(PC) ← ((SP))
(PSW_{4-7}) ← ((SP))

RL A Rotate Left Without Carry

Opcode:

1	1	1	0	0	1	1	1

The contents of the accumulator are rotated left one bit. Bit 7 is rotated into the bit 0 position.
(A_{n+1}) ← (A_n)   n=0–6
(A_0) ← (A_7)

Example: Assume accumulator contains 10110001.

```
RLNC: RL A            ;NEW ACC CONTENTS ARE 01100011
```

RLC A Rotate Left Through Carry

Opcode:

1	1	1	1	0	1	1	1

The contents of the accumulator are rotated left one bit. Bit 7 replaces the carry bit; the carry bit is rotated into the bit 0 position.

$(A_{n+1}) \leftarrow (A_n)$ n=0–6

$(A_0) \leftarrow (C)$

$(C) \leftarrow (A7)$

Example: Assume accumulator contains a 'signed' number; isolate sign without changing value.

```
RLTC: CLR C          ;CLEAR CARRY TO ZERO
      RLC A          ;ROTATE ACC LEFT, SIGN
                     ;BIT (7) IS PLACED IN CARRY
      RR A           ;ROTATE ACC RIGHT — VALUE
                     ;(BITS 0–6) IS RESTORED,
                     ;CARRY UNCHANGED, BIT 7
                     ;IS ZERO
```

RR A Rotate Right Without Carry

Opcode:

0	1	1	1	0	1	1	1

The contents of the accumulator are rotated right one bit. Bit 0 is rotated into the bit 7 position.

$(A_n) \leftarrow (A_{n+1})$ n=0–6

$(A_7) \leftarrow (A_0)$

Example: Assume accumulator contains 10110001.

```
RRNC: RRA            ;NEW ACC CONTENTS ARE 11011000
```

RRC A Rotate Right Through Carry

Opcode:

0	1	1	0	0	1	1	1

The contents of the accumulator are rotated right one bit. Bit 0 replaces the carry bit; the carry bit is rotated into the bit 7 position.

$(A_n) \leftarrow (A_{n+1})$ n=0–6

$(A_7) \leftarrow (C)$

$(C) \leftarrow (A_0)$

Example: Assume carry is not set and accumulator contains 10110001.

```
RRTC: RRCA           ;CARRY IS SET AND ACC
                     ;CONTAINS 01011000
```

SEL RB0 Select Register Bank 0

Opcode:

1	1	0	0	0	1	0	1

PSW BIT 4 is set to zero. References to working registers 0–7 address data memory locations 0–7. This is the recommended setting for normal program execution.

$(BS) \leftarrow 0$

SEL RB1 Select Register Bank 1

Opcode:

1	1	0	1	0	1	0	1

PSW bit 4 is set to one. References to working registers 0–7 address data memory locations 24–31. This is the recommended setting for interrupt service routines, since locations 0–7 are left intact. The setting of PSW bit 4 in effect at the time of an interrupt is restored by the RETR instruction when the interrupt service routine is completed.

(BS) ← 1

Example: Assume an IBF interrupt has occurred, control has passed to program memory location 3, and PSW bit 4 was zero before the interrupt.

```
LOC3: JMP INIT              ;JUMP TO ROUTINE 'INIT'
          .
          .
          .
INIT: MOV R7,A              ;MOV ACC CONTENTS TO
                            ;LOCATION 7
      SEL RB1               ;SELECT REG BANK 1
      MOV R7,#OFAH          ;MOVE 'FA' HEX TO LOCATION 31
          .
          .
          .
      SEL RBO               ;SELECT REG BANK 0
      MOV A,R7              ;RESTORE ACC FROM LOCATION 7
      RETR                  ;RETURN--RESTORE PC AND PSW
```

STOP TCNT Stop Timer/Event Counter

Opcode:

0	1	1	0	0	1	0	1

This instruction is used to stop both time accumulation and event counting.

Example: Disable interrupt, but jump to interrupt routine after eight overflows and stop timer. Count overflows in register 7.

```
START: DIS TCNTI            ;DISABLE TIMER INTERRUPT
       CLR A                ;CLEAR ACC TO ZERO
       MOV T,A              :MOV ZERO TO TIMER
       MOV R7,A             :MOVE ZERO TO REG 7
       STRT T               ;START TIMER
MAIN: JTF COUNT             ;JUMP TO ROUTINE 'COUNT'
                            ;IF TF=1 AND CLEAR TIMER FLAG
      JMP MAIN              ;CLOSE LOOP
COUNT: INC R7               ;INCREMENT REG 7
       MOV A,R7             ;MOVE REG 7 CONTENTS TO ACC
       JB3 INT              ;JUMP TO ROUTINE 'INT' IF ACC
                            ;BIT 3 IS SET (REG 7=8)
       JMP MAIN             ;OTHERWISE RETURN TO ROUTINE
                            ;MAIN
          .
          .
          .
INT: STOP TCNT              ;STOP TIMER
     JMP 7H                 ;JUMP TO LOCATION 7 (TIMER
                            ;INTERRUPT ROUTINE)
```

STRT CNT Start Event Counter

Opcode:

0	1	0	0	0	1	0	1

The TEST 1 (T_1) pin is enabled as the event-counter input and the counter is started. The event-counter register is incremented with each high to low transition on the T_1 pin.

Example: Initialize and start event counter. Assume overflow is desired with first T_1 input.

```
STARTC: EN TCNTI          ;ENABLE COUNTER INTERRUPT
        MOV A,#OFFH       ;MOVE 'FF' HEX (ONES) TO
                          ;ACC
        MOV T,A           ;MOVE ONES TO COUNTER
        STRT CNT          ;INPUT AND START
```

STRT T Start Timer

Opcode:

0	1	0	1	0	1	0	1

Timer accumulation is initiated in the timer register. The register is incremented every 32 instruction cycles. The prescaler which counts the 32 cycles is cleared but the timer register is not.

Example: Initialize and start timer.

```
STARTT: EN TCNTI          ;ENABLE TIMER INTERRUPT
        CLR A             :CLEAR ACC TO ZEROS
        MOV T,A           ;MOVE ZEROS TO TIMER
        STRT T            ;START TIMER
```

SWAP A Swap Nibbles Within Accumulator

Opcode:

0	1	0	0	0	1	1	1

Bits 0–3 of the accumulator are swapped with bits 4–7 of the accumulator.

$(A_{4-7}) \longleftrightarrow (A_{0-3})$

Example: Pack bits 0–3 of locations 50–51 into location 50.

```
PCKDIG: MOV R0,#50        ;MOVE '50' DEC TO REG 0
        MOV R1,#51        ;MOVE '51' DEC TO REG 1
        XCHD A,@R0        ;EXCHANGE BIT 0-3 OF ACC
                          ;AND LOCATION 50
        SWAP A            ;SWAP BITS 0-3 AND 4-7 OF ACC
        XCHD A,@R1        ;EXCHANGE BITS 0-3 OF ACC AND
                          ;LOCATION 51
        MOV @R0,A         ;MOVE CONTENTS OF ACC TO
                          ;LOCATION 51
```

XCH A,Rr Exchange Accumulator-Register Contents

Opcode:

0	0	1	0	1	r_2	r_1	r_0

The contents of the accumulator and the contents of working register 'r' are exchanged.

$(A) \longleftrightarrow (Rr)$ $r=0-7$

Example: Move PSW contents to Reg 7 without losing accumulator contents.

```
XCHAR7: XCH A,R7          ;EXCHANGE CONTENTS OF REG 7
                          ;AND ACC
        MOV A,PSW         ;MOVE PSW CONTENTS TO ACC
        XCH A,R7          ;EXCHANGE CONTENTS OF REG 7
                          ;AND ACC AGAIN
```

XCH A,@Rr Exchange Accumulator and Data Memory Contents

Opcode:

0	0	1	0	0	0	0	r

The contents of the accumulator and the contents of the data memory location addressed by bits 0–5 of register 'r' are exchanged. Register 'r' contents are unaffected.

(A) ⟷ ((Rr))   r=0–1

Example: Decrement contents of location 52.

```
DEC52: MOV R0,#52         ;MOVE '52' DEC TO ADDRESS
                          ;REG 0
       XCH A,@R0          ;EXCHANGE CONTENTS OF ACC
                          ;AND LOCATION 52
       DEC A              ;DECREMENT ACC CONTENTS
       XCH A,@R0          ;EXCHANGE CONTENTS OF ACC
                          ;AND LOCATION 52 AGAIN
```

XCHD A,@Rr Exchange Accumulator and Data Memory 4-bit Data

Opcode:

0	0	1	1	0	0	0	r

This instruction exchanges bits 0–3 of the accumulator with bits 0–3 of the data memory location addressed by bits 0–5 of register 'r'. Bits 4–7 of the accumulator, bits 4–7 of the data memory location, and the contents of register 'r' are unaffected.

$(A_{0-3}) \longleftrightarrow ((Rr_{0-3}))$ r=0–1

Example: Assume program counter contents have been stacked in locations 22–23.

```
XCHNIB: MOV R0,#23        ;MOVE '23' DEC TO REG 0
        CLR A             ;CLEAR ACC TO ZEROS
        XCHD A,@R0        ;EXCHANGE BITS 0-3 OF ACC
                          ;AND LOCATION 23 (BITS 8-11
                          ;OF PC ARE ZEROED, ADDRESS
                          ;REFERS TO PAGE 0)
```

XRL A,Rr Logical XOR Accumulator With Register Mask

Opcode:

1	1	0	1	1	r_2	r_1	r_0

Data in the accumulator is EXCLUSIVE ORed with the mask contained in working register 'r'.

(A) ← (A) XOR (Rr)   r=0–7

Example:

```
XORREG: XRL A,R5          ;'XOR' ACC CONTENTS WITH
                          ;MASK IN REG 5
```

XRL A,@Rr Logical XOR Accumulator With Memory Mask

Opcode:

1	1	0	1	0	0	0	r

Data in the accumulator is EXCLUSIVE ORed with the mask contained in the data memory location addressed by register 'r', bits 0–5.

(A) ← (A) XOR ((Rr))   r=0–1

Example:

```
XORDM: MOV R1,#20H        ;MOVE '20' HEX TO REG 1
       XRL A,@R1          ;'XOR' ACC CONTENTS WITH MASK
                          ;IN LOCATION 32
```

XRL A,#data Logical XOR Accumulator With Immediate Mask

Opcode:

1	1	0	1	0	0	1	1	•	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	

This is a 2-cycle instruction. Data in the accumulator is EXCLUSIVE ORed with an immediately-specified mask.

(A) ← (A) XOR data

Example:

```
XORID: XOR A,#HEXTEN        ;XOR CONTENTS OF ACC WITH
                            ;MASK EQUAL VALUE OF SYMBOL
                            ;'HEXTEN'
```

CHAPTER 4
SINGLE STEP, PROGRAMMING AND POWER-DOWN MODES

4

CHAPTER 4
SINGLE-STEP, PROGRAMMING, AND POWER-DOWN MODES

SINGLE-STEP (8741A EPROM Only)

The 8741A has a single-step mode which allows the user to manually step through his program one instruction at a time. While stopped, the address of the next instruction to be fetched is available on PORT 1 and the lower 2 bits of PORT 2. The single-step feature simplifies program debugging by allowing the user to easily follow program execution.

Figure 4-1 illustrates a recommended circuit for single-step operation, while Figure 4-2 shows the timing relationship between the SYNC output and the $\overline{SS}$ input. During single-step operation, PORT 1 and part of PORT 2 are used to output address information. In order to retain the normal I/O functions of PORTS 1 and 2, a separate latch can be used as shown in Figure 4-3.

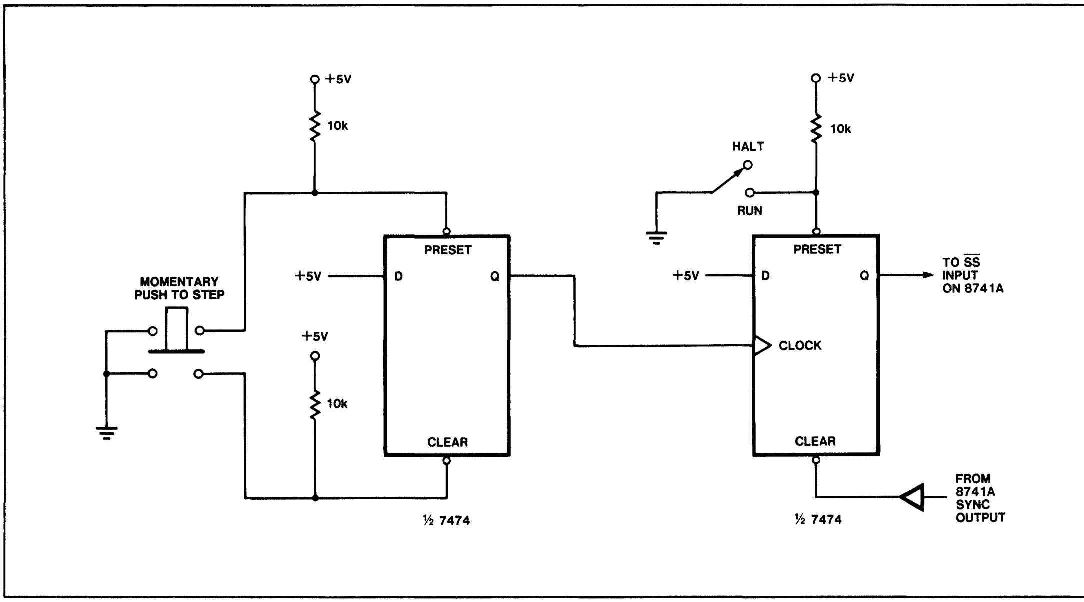


Figure 4-1. Single-Step Circuit

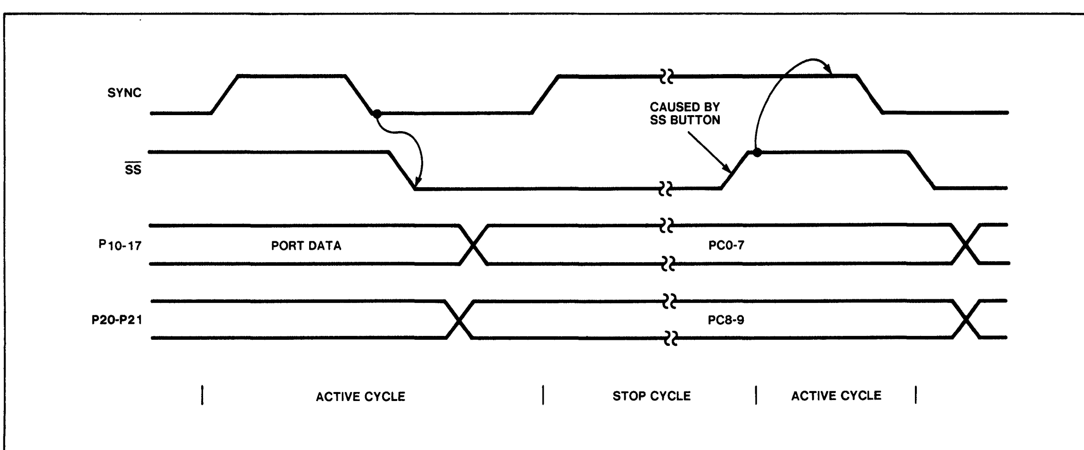


Figure 4-2. Single-Step Timing

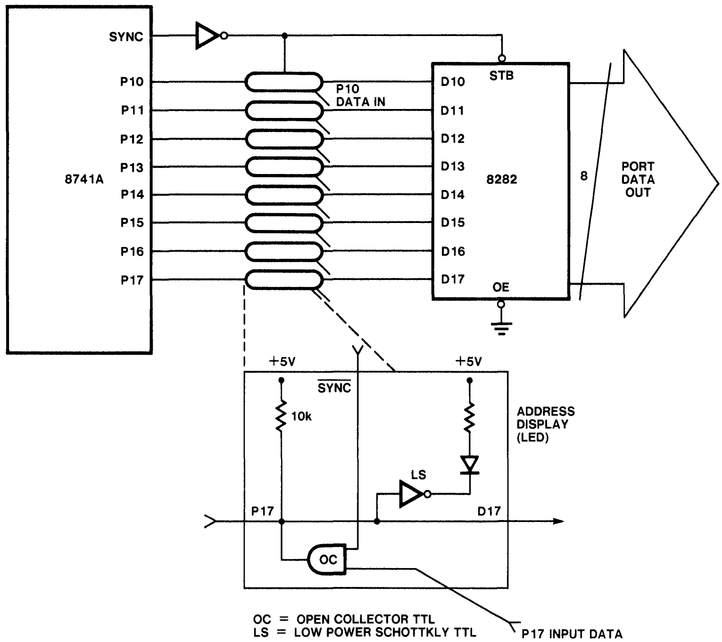


Figure 4-3. Latching Port Data

Timing

The sequence of single-step operation is as follows:

1) The processor is requested to stop by applying a low level on $\overline{SS}$. The $\overline{SS}$ input should not be brought low while SYNC is high. (The 8741A samples the $\overline{SS}$ pin in the middle of the SYNC pulse).

2) The processor responds to the request by stopping during the instruction fetch portion of the next instruction. If a double cycle instruction is in progress when the single-step command is received, both cycles will be completed before stopping.

3) The processor acknowledges it has entered the stopped state by raising SYNC high. In this state, which can be maintained indefinitely, the 10-bit address of the next instruction to be fetched is present on PORT 1 and the lower 2 bits of PORT 2.

4) $\overline{SS}$ is then raised high to bring the processor out of the stopped mode allowing it to fetch the next instruction. The exit from stop is indicated by the processor bringing SYNC low.

5) To stop the processor at the next instruction $\overline{SS}$ must be brought low again before the next SYNC pulse—the circuit in Figure 4-1 uses the trailing edge of the previous pulse. If $\overline{SS}$ is left high, the processor remains in the "RUN" mode.

Figure 4-1 shows a schematic for implementing single-step. A single D-type flip-flop with preset and clear is used to generate $\overline{SS}$. In the RUN mode $\overline{SS}$ is held high by keeping the flip-flop preset (preset has precedence over the clear input). To enter single-step, preset is removed allowing SYNC to bring $\overline{SS}$ low via the clear input. Note that SYNC must be buffered since the SN7474 is equivalent to 3 TTL loads.

The processor is now in the stopped state. The next instruction is initiated by clocking "1" into the flip-flop. This "1" will not appear on $\overline{SS}$ unless SYNC is high (i.e., clear must be removed from the flip-flop). In response to $\overline{SS}$ going high, the processor begins an instruction fetch which brings SYNC low. $\overline{SS}$ is then reset through the clear input and the processor again enters the stopped state.

PROGRAMMING, VERIFYING AND ERASING EPROM (8741A EPROM ONLY)

The internal Program Memory of the 8741A may be erased and reprogrammed by the user as explained in the following sections. See the data sheet for more detail.

Programming

The programming procedure consists of the following: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. Figure 4-4 illustrates the programming and verifying sequence. The following is a list of the pins used for programming and a description of their functions:

- XTAL 1, XTAL 2 — Clock Input (1 to 6 MHz)
- $\overline{\text{RESET}}$ — Initialization and Address Latching
- TEST 0 — Selection of Program or Verify Mode
- EA — Activation of Program/Verify Modes
- D_0–D_7 — Address and Data Input; Data Output During Verify
- P_{20}, P_{21} — Address Input
- V_{DD} — Programming Power Supply
- PROG — Program Pulse Input

NOTE: All set-up and hold times are 4 cycles.

The detailed Program/Verify sequence is as follows:

1) V_{DD} = 5V; Clock Running 4MHz. Crystal or External Clock; $\overline{\text{RESET}}$ = 0V; V_{DD} = 5V; A_0 = 0V; CS = 5V
TEST 0 = 5V; EA = 5V, D_0–D_7 and PROG floating.

2) Insert 8741A in programming socket.

3) TEST 0 = 0V (Select Program Mode)

4) EA = 23V (Activate Program Mode), PROG will float.

5) Address applied to D_0–D_7, P_{20}, P_{21}.

6) $\overline{\text{RESET}}$ = 5V (Latch Address).

7) Data applied to D_0–D_7.

8) V_{DD} = 25V (Programming Power).

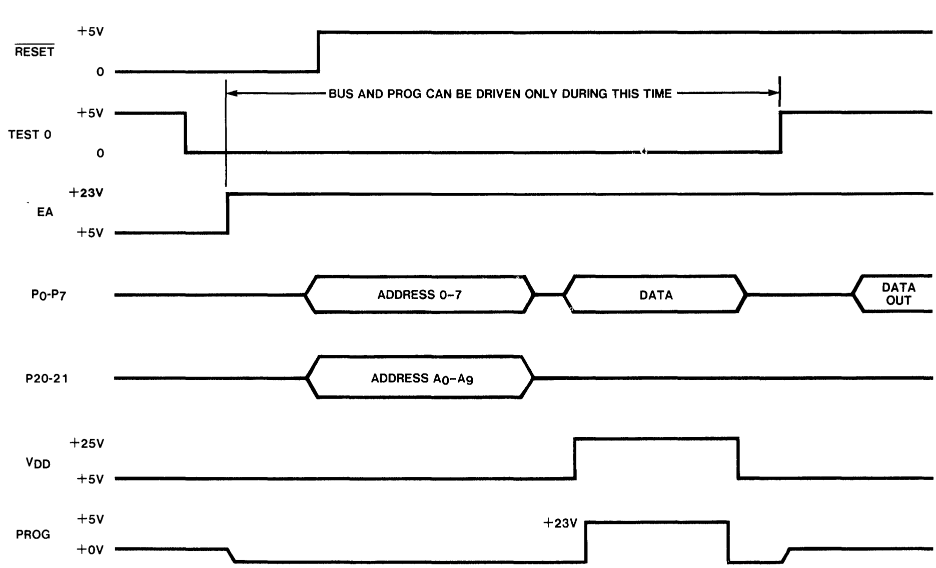


Figure 4-4. Programming Sequence

9) PROG = 0V followed by 50ms pulse to 23V.

10) V_{DD} = 5V.

11) TEST 0 = 5V (Verify Mode).

12) Read and Verify Data on D_0–D_7.

13) TEST 0 = 0V.

14) $\overline{\text{RESET}}$ = 0V and repeat from step 5.

15) Programmer should be at conditions of step 1 when 8741A is removed from socket.

WARNING

An attempt to program a mis-socketed 8741A will result in severe damage to the part. An indication of a properly socketed part is the appearance of the SYNC clock output. The lack of this clock may be used to disable the programmer.

Verification

Verification is accomplished by latching in an address as in the Programming Mode and then applying "1" to the TEST 0 input. The word stored at the selected address then appears on the D_0–D_7 lines. Note that verification can be applied to both ROM's and EPROM's independently of the programming procedure. See the data sheet.

Erasing

The program memory of the 8741A may be erased to zeros by exposing its translucent lid to shortwave ultraviolet light.

EPROM Light Sensitivity

The erasure characteristics of the 8741A EPROM are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000 Angstrom range. Data shows that constant exposure to room level fluorescent lighting could erase the typical 8741A in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 8741A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels (available from Intel) should be placed over the 8741A window to prevent unintentional erasure.

The recommended erasure procedure for the 8741A is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms. The integrated dose (i.e., UV intensity × exposure time) for erasure should be a minimum of 15W-sec/cm^2 power rating. The erasure time with this dosage is approximately 15 minutes using an ultraviolet lamp with a 12,000 μW/cm^2 power rating. The 8741A should be placed within 1 inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

External Access

The 8041A/8741A has an External Access (EA) pin which will put the processor into a test mode when a high level is applied. This allows the user to effectively disable the internal program memory.

The External Access mode is useful in testing because it allows the user to disable the internal application program and test processor functions directly. In addition, program memory can be read externally, independent of the processor.

This mode is invoked by connecting the EA pin to 5V. The current program counter contents then come out on PORTS 10-17 and PORTS 20-21 (PORT 10 is the least significant and PORT 21 the most significant bits). The desired instruction opcode is placed on D_0–D_7. This instruction is executed in place of the internal program memory contents. The I/O port data and program address are multiplexed on the 8741A but not on the 8041A.

Upon reset with EA = 5V, the 8041A sends out program counter contents 0FFH as the first address rather than 000H. The second address is 001H. Therefore, the first and second instructions should be located at 0FFH and 001H respectively. The 8741A outputs 000H as the first address after reset.

Reading and/or writing the Data Bus Buffer registers is still allowed although only when D_0–D_7 are not being sampled for opcode data. In practice, since this sampling time is not known externally, reads or writes on the system bus are done during SYNC high time. Approximately 600ns are available for each read or write cycle.

POWER DOWN MODE (8041A ROM ONLY)

Extra circuitry is included in the 8041A ROM version to allow low power, standby operation. Power is removed from all system elements except the 64-byte data RAM in the low power mode. Thus, the contents of RAM can be maintained while typically drawing only 10 to 15% of normal power.

The V_{CC} pin serves as the 5V supply pin for most of the 8041A circuitry while the V_{DD} pin supplies only the RAM array. In normal operation, both pins are at 5 volts. To enter the Power-Down mode, the V_{CC} pin is grounded while only V_{DD} is maintained at 5 volts. Applying a $\overline{RESET}$ signal to the processor inhibits access to RAM and thereby guarantees that the memory is not inadvertently altered during the transition when power is removed from V_{CC}. Figure 4-5 illustrates a recommended Power-Down sequence. The sequence typically occurs as follows:

1) Imminent power supply failure is detected by user defined circuitry. The signal must occur early enough to guarantee the 8041A can save all necessary data before V_{CC} falls outside normal operating tolerance.

2) A "Power Failure" signal is used to interrupt the processor (via a timer overflow interrupt, for instance) and call a Power Failure service routine.

3) The Power Failure routine saves all important data and machine status in the RAM array. The routine may also initiate transfer of a backup supply to the V_{DD} pin and indicate to external circuitry that the Power Failure routine is complete.

4) A $\overline{RESET}$ signal is applied by external hardware to guarantee data will not be altered as the power supply falls out of limits. $\overline{RESET}$ must be low until V_{CC} reaches ground potential.

Recovery from the Power-Down mode can occur as any other power-on sequence. An external 1 μfd capacitor on the $\overline{RESET}$ input will provide the necessary initialization pulse.

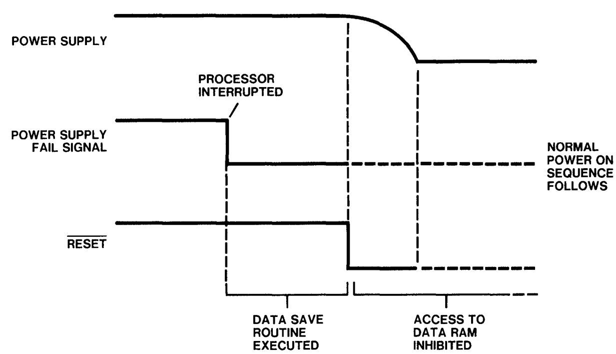


Figure 4-5. Power-Down Sequence

CHAPTER 5
SYSTEM OPERATION

5

CHAPTER 5
SYSTEM OPERATION

BUS INTERFACE

The UPI-41A Microcomputer functions as a peripheral to a master processor by using the data bus buffer registers to handle data transfers. The DBB configuration is illustrated in Figure 5-1. The UPI-41A Microcomputer's 8 three-state data lines (D_7–D_0) connect directly to the master processor's data bus. Data transfer to the master is controlled by 4 external inputs to the UPI:

- A_0 Address Input signifying command or data
- $\overline{CS}$ Chip Select
- $\overline{RD}$ Read strobe
- $\overline{WR}$ Write strobe

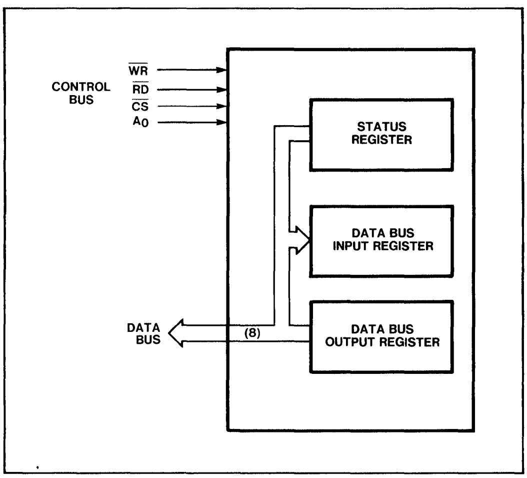


Figure 5-1. Data Bus Register Configuration

The master processor addresses the UPI-41A Microcomputer as a standard peripheral device. Table 5-1 shows the conditions for data transfer:

Table 5-1. Data Transfer Controls

$\overline{CS}$	A_0	$\overline{RD}$	$\overline{WR}$	Condition
0	0	0	1	Read DBBOUT
0	1	0	1	Read STATUS
0	0	1	0	Write DBBIN data, set $F_1 = 0$
0	1	1	0	Write DBBIN command set $F_1 = 1$
1	x	x	x	Disable DBB

Reading the DBBOUT Register

The sequence for reading the DBBOUT register is shown in Figure 5-2. This operation causes the 8-bit contents of the DBBOUT register to be placed on the system Data Bus. The OBF flag is cleared automatically.

Reading STATUS

The sequence for reading the UPI-41A Microcomputer's 8 STATUS bits is shown in Figure 5-3. This operation causes the 8-bit STATUS register contents to be placed on the system Data Bus as shown.

Write Data to DBBIN

The sequence for writing data to the DBBIN register is shown in Figure 5-4. This operation causes the system Data Bus contents to be transferred to the DBBIN register and the IBF flag is set. Also, the F_1 flag is cleared ($F_1 = 0$) and an interrupt request is generated. When the IBF interrupt is enabled, a jump to location 3 will occur. The interrupt request is cleared upon entering the IBF service routine or by a system RESET input.

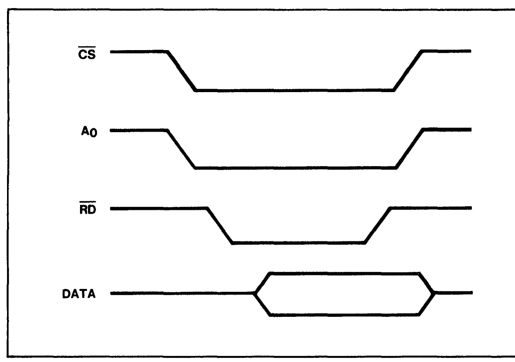


Figure 5-2. DBBOUT Read

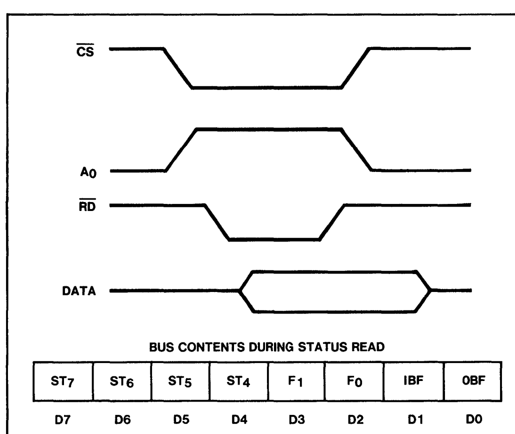


Figure 5-3. Status Read

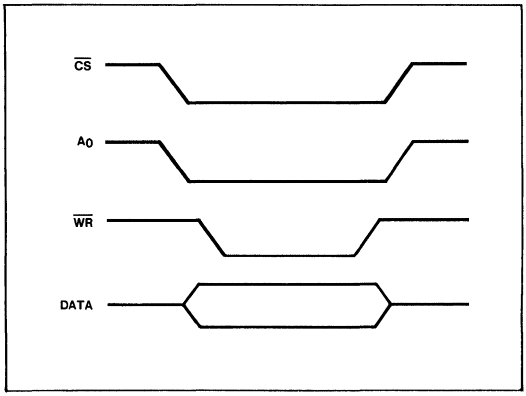


Figure 5-4. Writing Data to DBBIN

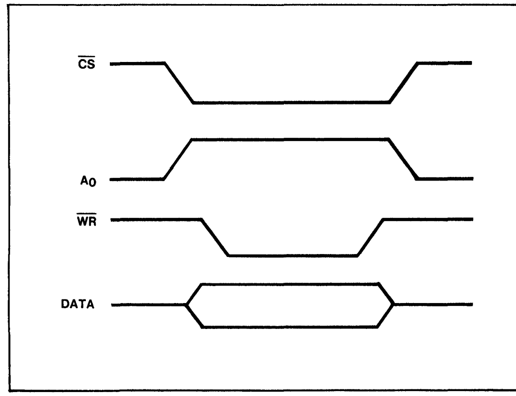


Figure 5-5. Writing Commands to DBBIN

Writing Commands to DBBIN

The sequence for writing commands to the DBBIN register is shown in Figure 5-5. This sequence is identical to a data write except that the A_0 input is latched in the F_1 flag (F_1 = 1). The IBF flag is set and an interrupt request is generated when the master writes a command to DBB.

OPERATIONS OF DATA BUS REGISTERS

The UPI-41A Microcomputer controls the transfer of DBB data to its accumulator by executing INput and OUTput instructions. An IN A,DBB instruction causes the contents to be transferred to the UPI accumulator and the IBF flag is cleared.

The OUT DBB,A instruction causes the contents of the accumulator to be transferred to the DBBOUT register. The OBF flag is set.

The UPI's data bus buffer interface is applicable to a variety of microprocessors including the 8086, 8088, 8085, 8080, and 8048.

A description of the interface to each of these processors follows.

DESIGN EXAMPLES

8085A Interface

Figure 5-6 illustrates an 8085A system using a UPI-41A. The 8085A system uses a multiplexed address and data bus. During I/O the 8 upper address lines (A_8–A_{15}) contain the same I/O address as the lower 8 address/data lines (A_0–A_7); therefore I/O address decoding is done using only the upper 8 lines to eliminate latching of the address. An 8205 decoder provides address decoding for both the UPI-41A and the 8237. Data is transferred using the two DMA handshaking lines of PORT 2. The 8237 performs the actual bus transfer operation. Using the UPI-41A's OBF master interrupt, the UPI-41A notifies the 8085A upon transfer completion using the RST 5.5 interrupt input. The $\overline{\text{IBF}}$ master interrupt is not used in this example.

8088 Interface

Figure 5-7 illustrates a UPI-41A interface to an 8088 minimum mode system. Two 8-bit latches are used to demultiplex the address and data bus. The address bus is 20-lines wide. For I/O only, the lower 16 address lines are used, providing an addressing range of 64K. UPI address selection is accomplished using an 8205 decoder. The A_0 address line of the bus is connected to the corresponding UPI input for register selection. Since the UPI-41A is polled by the 8088, neither DMA nor master interrupt capabilities of the UPI-41A are used in the figure.

8086 Interface

The UPI-41A can be used on an 8086 maximum mode system as shown in figure 5-8. The address and data bus is demultiplexed using three 8282 latches providing separate address and data buses. The address bus is 20-lines wide and the data bus is 16-lines wide. Multiplexed control lines are decoded by the 8288. The UPI's $\overline{\text{CS}}$ input is provided by linear selection. Note that the UPI-41A is both I/O mapped and memory mapped as a result of the linear addressing technique. An address decoder may be used to limit the UPI-41A to a specific I/O mapped address. Address line A_1 is connected to the UPI's A_0 input. This insures that the registers of the UPI will have even I/O addresses. Data will be transferred on D_0–D_7 lines only. This allows the I/O registers to be accessed using byte manipulation instructions.

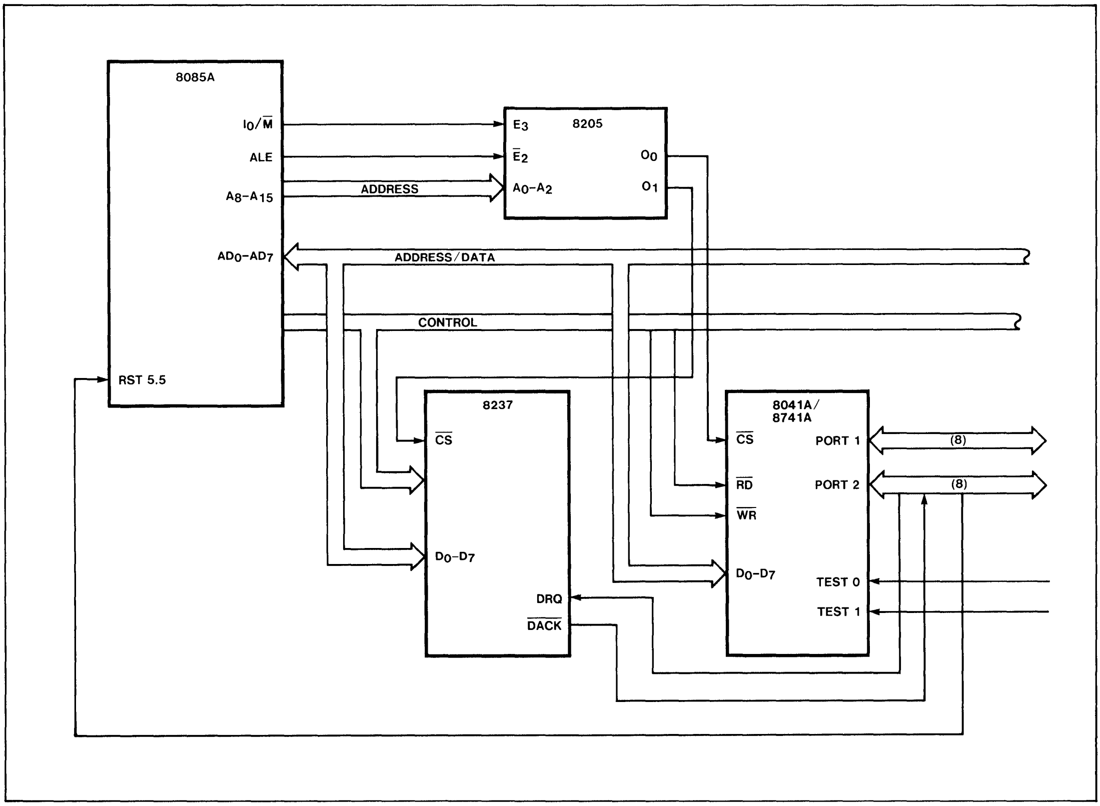


Figure 5-6. 8041A To 8085A System

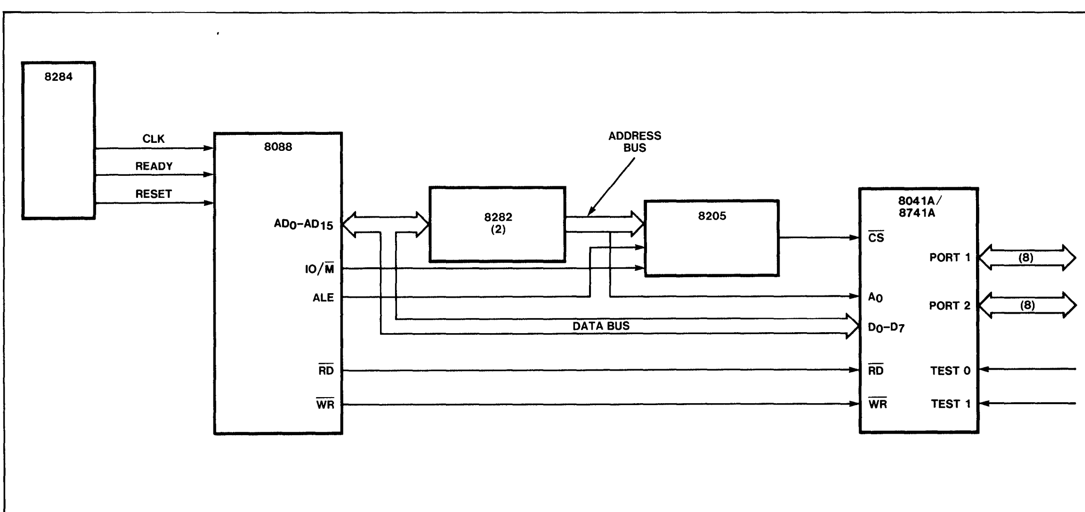


Figure 5-7. 8041A To 8088 Minimum Mode System

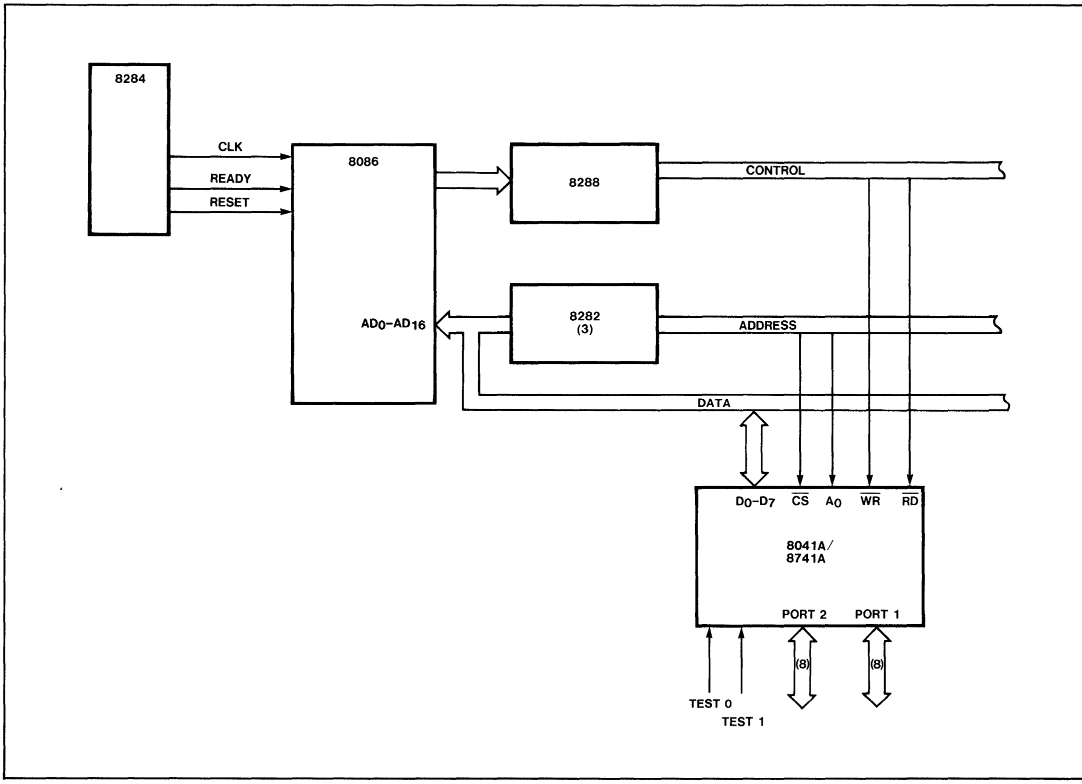


Figure 5-8. 8041A To 8086 Maximum Mode Systems

8080 Interface

Figure 5-9 illustrates the interface to an 8080A system. In this example, a crystal and capacitor are used for UPI-41A timing reference and power-on RESET. If the 2-MHz 8080A 2-phase clock were used instead of the crystal, the UPI-41A would run at only 30% full speed.

The A_0 and $\overline{CS}$ inputs are direct connections to the 8080 address bus. In larger systems, however, either of these inputs may be decoded from the 16 address lines.

The $\overline{RD}$ and $\overline{WR}$ inputs to the UPI can be either the $\overline{IOR}$ and $\overline{IOW}$ or the $\overline{MEMR}$ and $\overline{MEMR}$ signals depending on the I/O mapping technique to be used.

The UPI can be addressed as an I/O device using INput and OUTput instructions in 8080 software.

8048 Interface

Figure 5-10 shows the UPI interface to an 8048 master processor.

The 8048 $\overline{RD}$ and $\overline{WR}$ outputs are directly compatible with the UPI. Figure 5-11 shows a distributed processing system with up to seven 8041A's connected to a single 8048 master processor.

In this configuration the 8048 uses PORT 0 as a data bus. I/O PORT 2 is used to select one of the seven 8041A's when data transfer occurs. The 8041A's are programmed to handle isolated tasks and, since they operate in parallel, system throughput is increased.

GENERAL HANDSHAKING PROTOCOL

1) Master reads STATUS register ($\overline{RD}$, $\overline{CS}$, A_0 = (0, 0, 1)) in polling or in response to either an $\overline{IBF}$ or an OBF interrupt.

2) If the UPI-41A DBBIN register is empty (IBF flag = 0), Master writes a word to the DBBIN register ($\overline{WR}$, $\overline{CS}$, A_0 = (0, 0, 1) or (0, 0, 0)). If A_0 = 1, write command word, set F_1. If A_0 = 0, write data word, F_1 = 0.

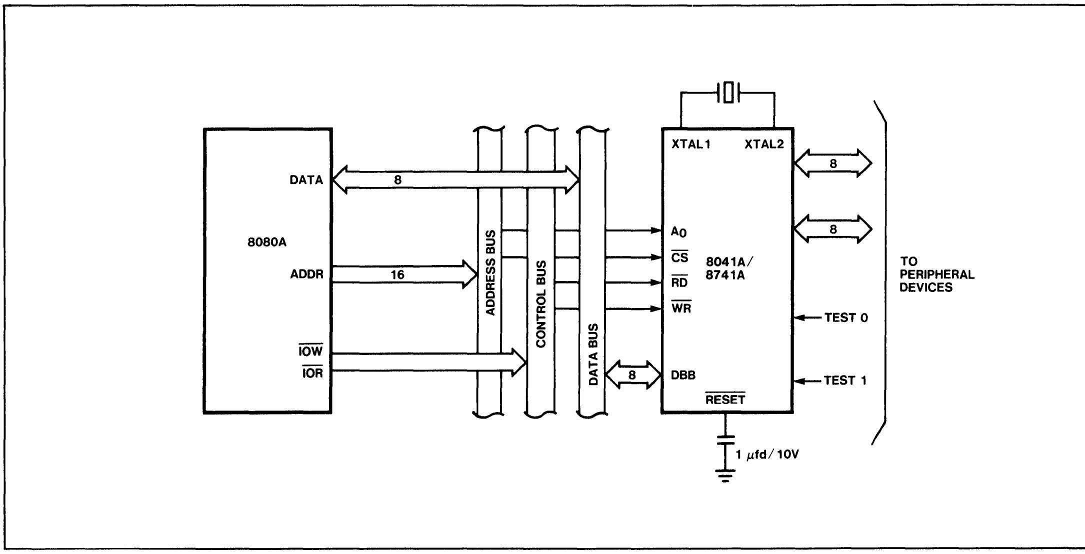


Figure 5-9. 8080A-8041A Interface

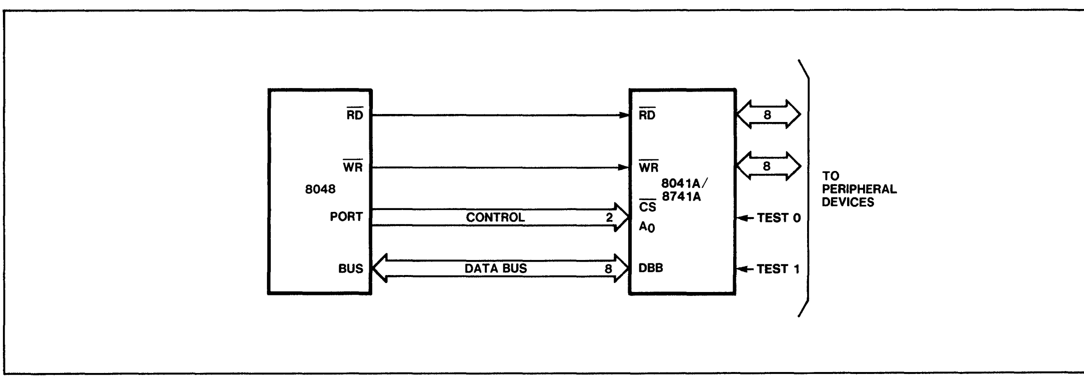


Figure 5-10. 8048-8041A Interface

3) If the UPI-41A DBBOUT register is full (OBF flag = 1), Master reads a word from the DBBOUT register ($\overline{RD}$, $\overline{CS}$, A_0 = (0, 0, 0)).

4) UPI-41A recognizes IBF (via IBF interrupt or JNIBF). Input data or command word is processed, depending on F_1; IBF is reset. Repeat step 1 above.

5) UPI-41A recognizes OBF flag = 0 (via JOBF). Next word is output to DBBOUT register, OBF is set. Repeat step 1 above.

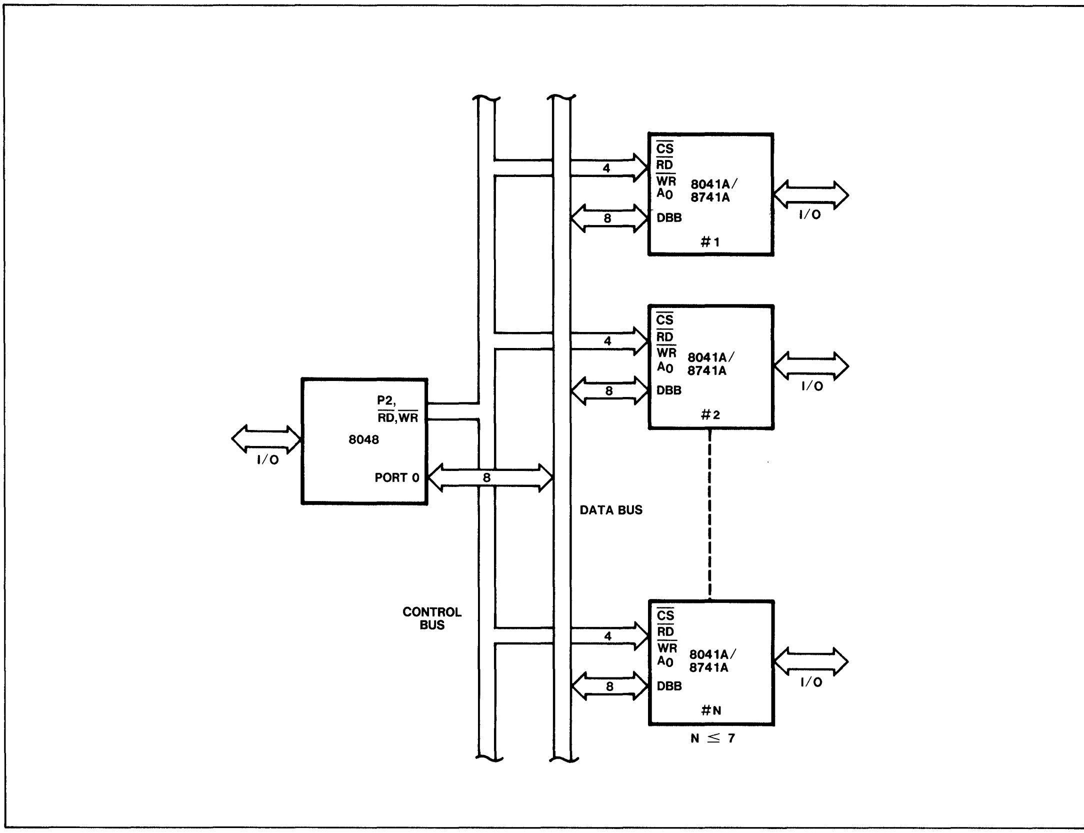


Figure 5-11. Distributed Processor System

CHAPTER 6
APPLICATION NOTES

6

Chapter 6
APPLICATIONS

ABSTRACTS

The UPI-41A is designed to fill a wide variety of low to medium speed peripheral interface applications where flexibility and easy implementation are important considerations. The following examples illustrate some typical applications.

Keyboard Encoder

Figure 6-1 illustrates a keyboard encoder configuration using the UPI and the 8243 I/O expander to scan a 128-key matrix. The encoder has switch matrix scanning logic, N-key rollover logic, ROM look-up table, FIFO character buffer, and additional outputs for display functions, control keys or other special functions.

PORT 1 and PORTs 4–7 provide the interface to the keyboard. PORT 1 lines are set one at a time to select the various key matrix rows.

When a row is energized, all 16 columns (i.e., PORTs 4–7 inputs) are sampled to determine if any switch in the row is closed. The scanning software is code efficient because the UPI instruction set includes individual bit set/clear operations and expander PORTs 4–7 can be directly addressed with single, 2-byte instructions. Also, accumulator bits can be tested in a single operation. Scan time for 128 keys is about 10 ms. Each matrix point has a unique binary code which is used to address ROM when a key closure is detected. Page 3 of ROM contains a look-up table with useable codes (i.e., ASCII, EBCDIC, etc.) which correspond to each key. When a valid key closure is detected the ROM code corresponding to that key is stored in a FIFO buffer in data memory for transfer to the master processor. To avoid stray noise and switch bounce, a key closure must be detected on two consecutive scans before it is considered valid and loaded into the FIFO buffer. The FIFO buffer allows multiple keys to be processed as they are depressed without regard to when they are released, a condition known as N-key rollover.

The basic features of this encoder are fairly standard and require only about 500 bytes of memory. Since the UPI is programmable and has additional memory capacity it can handle a number of other functions. For example, special keys can be programmed to give an entry on closing as well as opening. Also, I/O lines are available to control a 16-digit, 7-segment display. The UPI can also be programmed to recognize special combinations of characters such as commands, then transfer only the decoded information to the master processor.

A complete keyboard application has been developed for the UPI-41A. A description is included in this section. The code for the application is available in the Intel Insite Library (program AB 147).

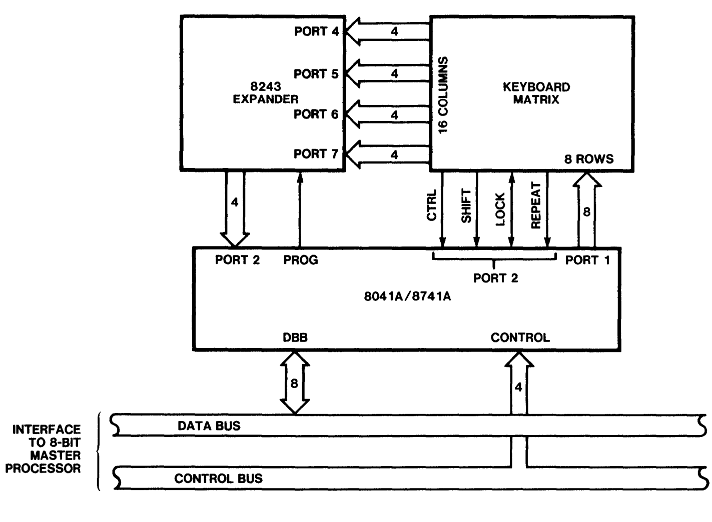


Figure 6-1. Keyboard Encoder Configuration

Matrix Printer Interface

The matrix printer interface illustrated in Figure 6-2 is a typical application for the UPI-41A. The actual printer mechanism could be any of the numerous dot-matrix types and similar configurations can be shown for drum, spherical head, daisy wheel or chain type printers.

The bus structure shown represents a generalized, 8-bit system bus configuration. The UPI's three-state interface port and asynchronous data buffer registers allow it to connect directly to this type of system for efficient, two-way data transfer.

The UPI's two on-board I/O ports provide up to 16 input and output signals to control the printer mechanism. The timer/event counter is used for generating a timing sequence to control print head position, line feed, carriage return, and other sequences. The on-board program memory provides character generation for 5 × 7, 7 × 9, or other dot matrix formats. As an added feature a portion of the 64 × 8-bit data memory can be used as a FIFO buffer so that the master processor can send a block of data at a high rate. The UPI can then output characters from the buffer at a rate the printer can accept while the master processor returns to other tasks.

The 8295 Printer Controller is an example of an 8041A preprogrammed as a dot matrix printer interface.

Tape Cassette Controller

Figure 6-3 illustrates a digital cassette interface which can be implemented with the UPI-41A. Two sections of the tape transport are controlled by the UPI: digital data/command logic, and motor servo control.

The motor servo requires a speed reference in the form of a monostable pulse whose width is proportional to the desired speed. The UPI monitors a prerecorded clock from the tape and uses its on-board interval timer to generate the required speed reference pulses at each clock transition.

Recorded data from the tape is supplied serially by the data/command logic and is converted to 8-bit words by the UPI, then transferred to the master processor. At 10 ips tape speed the UPI can easily handle the 8000 bps data rate. To record data, the UPI uses the two input lines to the data/command logic which control the flux direction in the recording head. The UPI also monitors 4 status lines from the tape transport including: end of tape, cassette

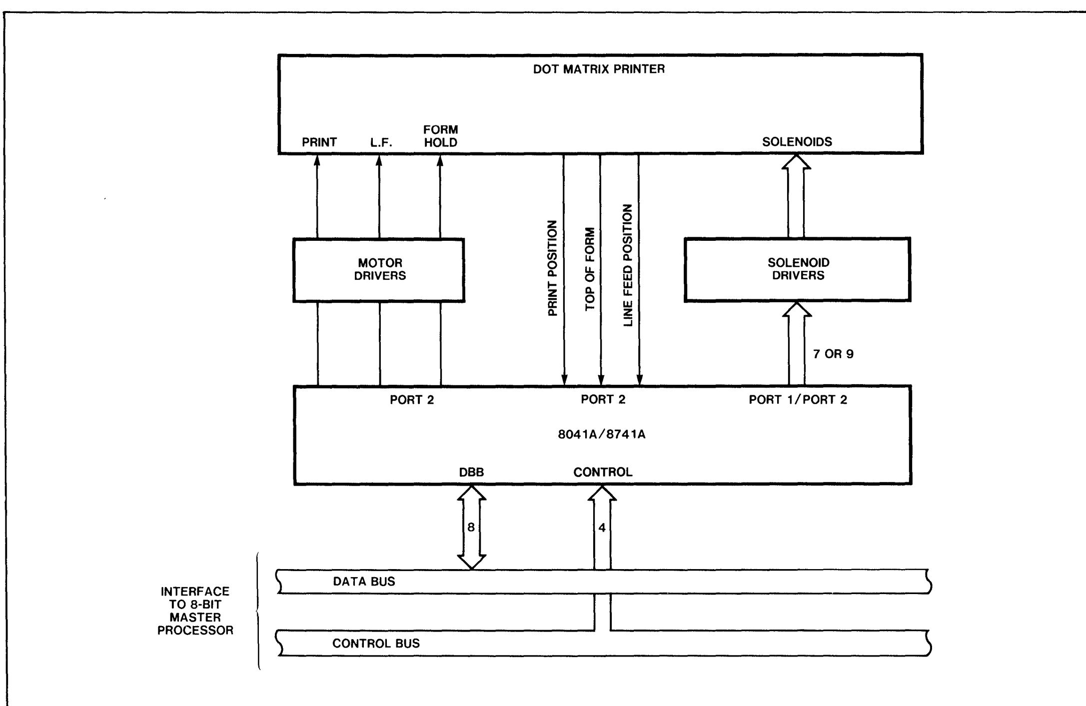


Figure 6-2. Matrix Printer Controller

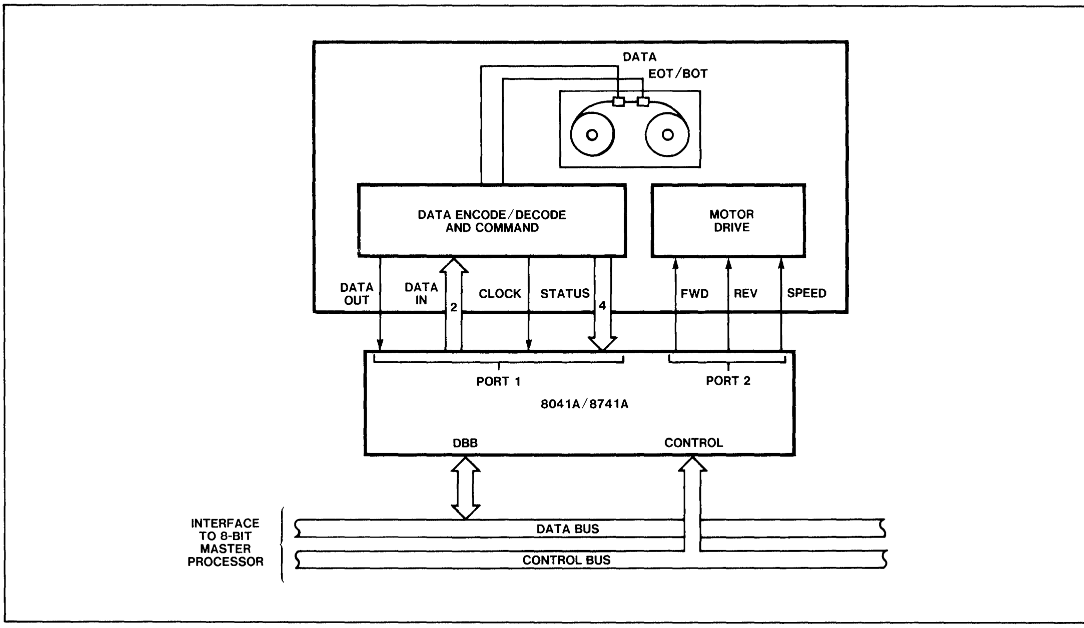


Figure 6-3. Tape Transport Controller

inserted, busy, and write permit. All control signals can be handled by the UPI's two I/O ports.

Universal I/O Interface

Figure 6-4 shows an I/O interface design based on the UPI. This configuration includes 12 parallel I/O lines and a serial (RS232C) interface for full duplex data transfer up to 1200 baud. This type of design can be used to interface a master processor to a broad spectrum of peripheral devices as well as to a serial communication channel.

PORT 1 is used strictly for I/O in this example while PORT 2 lines provide five functions:

- P_{23}-P_{20} I/O lines (bidirectional)
- P_{24} Request to send (RTS)
- P_{25} Clear to Send (CTS)
- P_{26} Interrupt to master
- P_{27} Serial data out

The parallel I/O lines make use of the bidirectional port structure of the UPI. Any line can function as an input or output. All port lines are automatically initialized to 1 by a system $\overline{\text{RESET}}$ pulse and remain

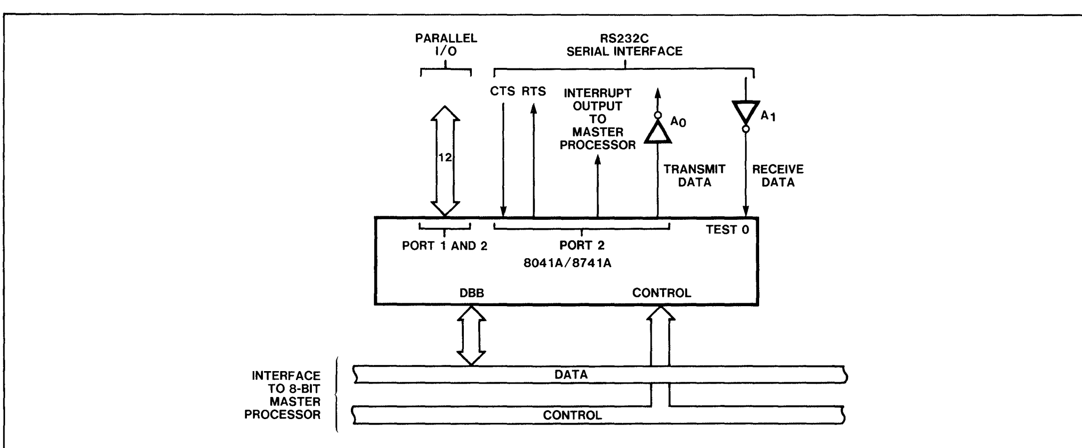


Figure 6-4. Universal I/O Interface

latched. An external TTL signal connected to a port line will override the UPI's 50K-ohm internal pull-up so that an INPUT instruction will correctly sample the TTL signal.

Four PORT 2 lines function as general I/O similar to PORT 1. Also, the RTS signal is generated on PORT 2 under software control when the UPI has serial data to send. The CTS signal is monitored via PORT 2 as an enable to the UPI to send serial data. A PORT 2 line is also used as a software generated interrupt to the master processor. The interrupt functions as a service request when the UPI has a byte of data to transfer or when it is ready to receive. Alternatively, the EN FLAGS instruction could be used to create the OBF and $\overline{IBF}$ interrupts on P_{24} and P_{25}.

The RS232C interface is implemented using the TEST 0 pin as a receive input and a PORT 2 pin as a transmit output. External packages (A_0, A_1) are used to provide RS232C drive requirements. The serial receive software is interrupt driven and uses the on-chip timer to perform time critical serial control. After a start bit is detected the interval timer can be preset to generate an interrupt at the proper time for sampling the serial bit stream. This eliminates the need for software timing loops and allows the processor to proceed to other tasks (i.e., parallel I/O operations) between serial bit samples. Software flags are used so the main program can determine when the interrupt driven receive program has a character assembled for it.

This type of configuration allows system designers flexibility in designing custom I/O interfaces for specific serial and parallel I/O applications. For instance, a second or third serial channel could be substituted in place of the parallel I/O if required. The UPI's data memory can buffer data and commands for up to 4 low-speed channels (110 baud teletypewriter, etc.)

Application Notes

The following application notes illustrate the various applications of the UPI family. Other related publications including the *8048 Family Application Handbook* are available through the Intel Literature Department.

INTRODUCTION TO THE UPI-41A™

Introduction

Since the introduction in 1974 of the second generation of microprocessors, such as the 8080, a wide range of peripheral interface devices have appeared. At first, these devices solved application problems of a general nature; i.e., parallel interface (8255), serial interface (8251), timing (8253), interrupt control (8259). However, as the speed and density of LSI technology increased, more and more intelligence was incorporated into the peripheral devices. This allowed more specific application problems to be solved, such as floppy disk control (8271), CRT control (8275), and data link control (8273). The advantage to the system designer of this increased peripheral device intelligence is that many of the peripheral control tasks are now handled externally to the main processor in the peripheral hardware rather than internally in the main processor software. This reduced main processor overhead results in increased system throughput and reduced software complexity.

In spite of the number of peripheral devices available, the pervasiveness of the microprocessor has been such that there is still a large number of peripheral control applications not yet satisfied by dedicated LSI. Complicating this problem is the fact that new applications are emerging faster than the manufacturers can react in developing new, dedicated peripheral controllers. To address this problem, a new microcomputer-based Universal Peripheral Interface (UPI-41A) device was developed.

In essence, the UPI-41A acts as a slave processor to the main system CPU. The UPI contains its own processor, memory, and I/O, and is completely user programmable; that is, the entire peripheral control algorithm can be programmed locally in the UPI, instead of taxing the master processor's main memory. This distributed processing concept allows the UPI to handle the real-time tasks such as encoding keyboards, controlling printers, or multiplexing displays, while the main processor is handling non-real-time dependent tasks such as buffer management or arithmetic. The UPI relies on the master only for initialization, elementary commands, and data transfers. This technique results in an overall increase in system efficiency since both processors—the master CPU and the slave UPI—are working in parallel.

This application note presents three UPI-41A applications which are roughly divided into two groups: applications whose complexity and UPI code space requirements allow them to either stand alone or be incorporated as just one task in a "multi-tasking" UPI, and applications which are complete UPI applications in themselves. Applications in the first group are a simple LED display and sensor matrix controllers. A combination serial/parallel/ I/O device is an application in the second group. Each application illustrates different UPI configurations and features. However, before the application details are presented, a section on the UPI/master protocol requirements is included. These protocol requirements are key to UPI software development. For convenience, the UPI block diagram is reproduced in Figure 1 and the instruction set summary in Table 1.

UPI-41 vs. UPI-41A

The UPI-41A is an enhanced version of the UPI-41. It incorporates several architectural features not found on the "non-A" device:

- Separate Data In and Data Out data bus buffer registers
- User-definable STATUS register bits
- Programmable master interrupts for the OBF and $\overline{\text{IBF}}$ flags
- Programmable DMA interface to external DMA controller.

The separate Data In (DBBIN) and Data Out (DBBOUT) registers greatly simplify the master/ UPI protocol compated to the UPI-41. The master need only check IBF before writing to DBBIN and OBF before reading DBBOUT. No data bus buffer lock-out is required.

The most significant nibble of the STATUS register, undefined in the UPI-41, is user-definable in UPI-41A. It may be loaded directly from the most significant nibble of the Accumulator (MOV STS,A). These extra four STATUS bits are useful for transferring additional status information to the master. This application note uses this feature extensively.

A new instruction, EN FLAGS, allows OBF and $\overline{\text{IBF}}$ to be reflected on PORT 2 BIT 4 and PORT 2 BIT 5 respectively. This feature enables interrupt-driven data transfers when these pins are interrupt sources to the master.

By executing an EN DMA instruction PORT 2 BIT 6 becomes a DRQ (DMA Request) output and PORT 2 BIT 7 becomes $\overline{\text{DACK}}$ (DMA Acknowledge). Setting DRQ requests a DMA cycle to an external DMA controller. When the cycle is granted, the DMA controller returns DACK plus either $\overline{\text{RD}}$ (Read) or $\overline{\text{WR}}$ (Write). DACK automatically forces

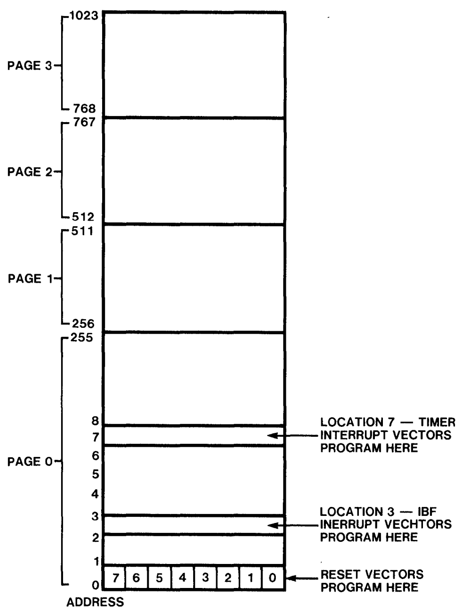


Figure 1A. Program Memory Map

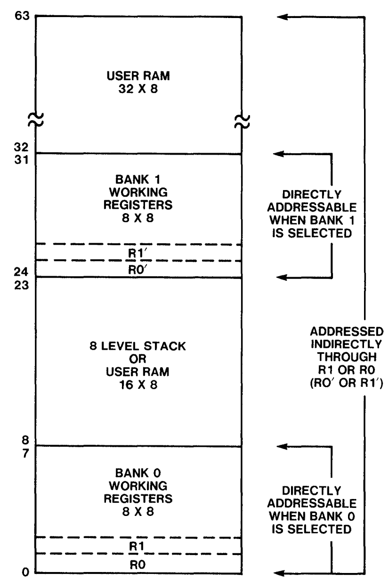


Figure 1B. Data Memory Map

$\overline{CS}$ and A_0 low internally and clears DRQ. This selects the appropriate data buffer register (DBBOUT for DACK and $\overline{RD}$, DBBIN for DACK and $\overline{WR}$) for the DMA transfer.

Like the "non-A", the UPI-41A is available in both ROM (8041A) and EPROM (8741A) Program Memory versions. This application note deals exclusively with the UPI-41A since the applications use the "A"s enhanced features.

UPI/MASTER PROTOCOL

As in most closely coupled multiprocessor systems, the various processors communicate via a shared resource. This shared resource is typically specific locations in RAM or in registers through which status and data are passed. In the case of a master processor and a UPI-41A, the shared resource is 3 separate, master-addressable, registers internal to the UPI. These registers are the status register (STATUS), the Data Bus Buffer Input register (DBBIN), and the Data Bus Output register (DBBOUT). [Data Bus Buffer direction is relative to the UPI]. To illustrate this register interface, consider the 8085A/UPI system in Figure 2.

Looking into the UPI from the 8085A, the 8085A sees only the three registers mentioned above. If the 8085A wishes to issue a command to the UPI, it does so by writing the command to the DBBIN register according to the decoding of Table 2. Data for the UPI is also passed via the DBBIN register. (The UPI differentiates commands and data by examining the A_0 pin. Just how this is done is covered shortly.) Data from the UPI for the 8085A is passed in the DBBOUT register. The 8085A may interrogate the UPI's status by reading the UPI's STATUS register. Four bits of the STATUS register act as flags and are used to handshake data and commands into and out of the UPI. The STATUS register format is shown in Figure 3.

BIT 0 is OBF (Output Buffer Full). This flag indicates to the master when the UPI has placed data in the DBBOUT register. OBF is set when the UPI writes to DBBOUT and is reset when the master reads DBBOUT. The master finds meaningful data in the DBBOUT register only when OBF is set.

The Input Buffer Full (IBF) flag is BIT 1. The UPI uses this flag as an indicator that the master has written to the DBBIN register. The master uses IBF

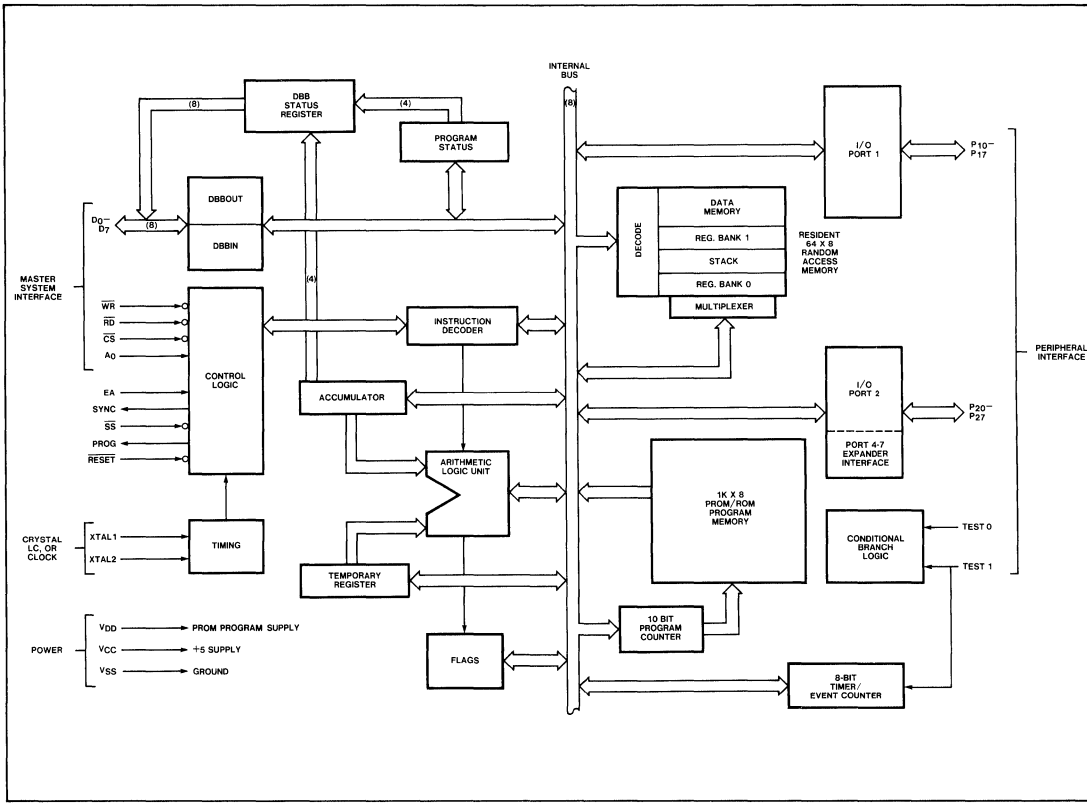


Figure 1C. UPI-41A Block Diagram

to indicate when the UPI has accepted a particular command or data byte. The master should examine IBF before outputting anything to the UPI. IBF is set when the master writes to DBBIN and is reset when the UPI reads DBBIN. The master must wait until IBF=0 before writing new data or commands to DBBIN. Conversely, the UPI must ensure IBF=1 before reading DBBIN.

The third STATUS register bit is F_0 (FLAG 0). This is a general purpose flag that the UPI can set, reset, and test. It is typically used to indicate a UPI error or busy condition to the master.

FLAG 1 (F_1) is the final dedicated STATUS bit. Like F_0 the UPI can set, reset, and test this flag. However, in addition, F_1 reflects the state of the A_0 pin whenever the master writes to the DBBIN register. The UPI uses this flag to delineate between master command and data writes to DBBIN.

The remaining four STATUS register bits are user definable. Typical uses of these bits are as status in-

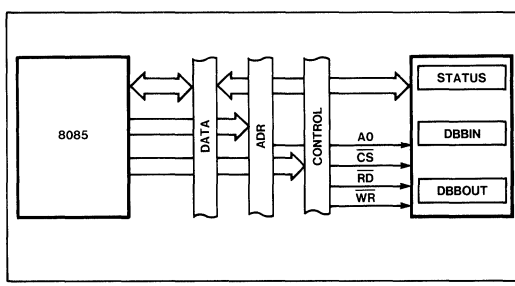


Figure 2. Register Interface

dicators for individual tasks in a multitasking UPI or as UPI generated interrupt status. These bits find a wide variety of uses in the upcoming applications.

Looking into the 8085A from the UPI, the UPI sees the two DBB registers plus the IBF, OBF, and F_1 flags. The UPI can write from its accumulator to DBBOUT or read DBBIN into the accumulator. The UPI cannot read OBF, IBF, or F_1 directly, but these flags may be tested using conditional jump

Table 1. Instruction Set Summary

Mnemonic	Description	Bytes	Cycles
Accumulator			
ADD A,R_r	Add register to A	1	1
ADD A,@R_r	Add data memory to A	1	1
ADD A,#data	Add immediate to A	2	2
ADDC A,R_r	Add register to A with carry	1	1
ADDC A @R_r	Add data memory to A with carry	1	1
ADDC A,#data	Add immed. to A with carry	2	2
ANL a,R_r	AND register to A	1	1
ANL A,@R_r	AND data memory to A	1	1
ANL A,#data	AND immediate to A	2	2
ORL A,R_r	OR register to A	1	1
ORL A@R_r	OR data memory to A	1	1
ORL A,#data	OR immediate to A	2	2
XRL A,R_r	Exclusive OR register to A	1	1
XRL A,@R_r	Exclusive OR data memory to A	1	1
XRL A,#data	Exclusive OR immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap digits of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
Input/Output			
IN A,P_p	Input port to A	1	2
OUTL P_p,A	Output A to port	1	2
ANL P_p,#data	AND immediate to port	2	2
ORL P_p,#data	OR immediate to port	2	2
IN A,DBB	Input DBB to A, clear IBF	1	1
OUT DBB,A	Output A to DBB, set OBF	1	1
MOV STS,A	A_4-A_7 to Bits 4-7 of Status	1	1
MOVD A,P_p	Input Expander port to A	1	2
MOVD P_p,A	Output A to Expander port	1	2
ANLD P_p,A	AND A to Expander port	1	2
ORLD P_p,A	OR A to Expander port	1	2
Data Moves			
MOV A,R_r	Move register to A	1	1
MOV A,@R_r	Move data memory to A	1	1
MOV A,#data	Move immediate to A	2	2
MOV R_r,A	Move A to register	1	1
MOV @R_r,A	Move A to data memory	1	1
MOV R_r,#data	Move immediate to register	2	2
MOV @R_r,#data	Move immediate to data memory	2	2
MOV A,PSW	Move PSW to A	1	1
MOV PSW,A	Move A to PSW	1	1
XCH A,R_r	Exchange A and register	1	1
XCH A,@R_r	Exchange A and data memory	1	1
XCHD A@R_r	Exchange digit of A and register	1	1
MOVP A,@A	Move to A from current page	1	2
MOVP3, A,@A	Move to A from page 3	1	2

Mnemonic	Description	Bytes	Cycles
Timer/Counter			
MOV A,T	Read Timer/Counter	1	1
MOV T,A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1
Control			
EN DMA	Enable DMA Handshake Lines	1	1
EN I	Enable IBF Interrupt	1	1
DIS I	Disable IBF Interrupt	1	1
EN FLAGS	Enable Master Interrupts	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1
Registers			
INC R_r	Increment register	1	1
INC @R_r	Increment data memory	1	1
DEC R_r	Decrement register	1	1
Subroutine			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
Flags			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear F1 Flag	1	1
CPL F1	Complement F1 Flag	1	1
Branch			
JMP ADDR	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R,addr	Decrement register and skip	2	2
JC addr	Jump on Carry=1	2	2
JNC addr	Jump on Carry=0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0=1	2	2
JNT0 addr	Jump on T0=0	2	2
JT1 addr	Jump on T1=1	2	2
JNT1 addr	Jump on T1=0	2	2
JF0 addr	Jump on F0 Flag=1	2	2
JF1 addr	Jump on F1 Flag=1	2	2
JTF addr	Jump on Timer Flag=1,Clear Flag	2	2
JNIBF addr	Jump on IBF Flag=0	2	2
JOBF addr	Jump on OBF Flag=1	2	2
JBb addr	Jump on Accumulator Bit	2	2

Table 2. Register Decoding

$\overline{CS}$	A0	$\overline{RD}$	$\overline{WR}$	REGISTER
0	0	0	1	READ DBBOUT
0	1	0	1	READ STATUS
0	0	1	0	WRITE DBBIN (DATA)
0	1	1	0	WRITE DBBIN (COMMAND)
1	X	X	X	NO ACTION

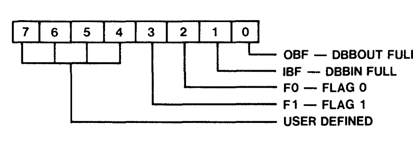


Figure 3. Status Register Format

instructions. The UPI should make sure that OBF is reset before writing new data into DBBOUT to ensure that the master has read previous DBBOUT data. IBF should also be tested before reading DBBIN since DBBIN data is valid only when IBF is set. As was mentioned earlier, the UPI uses F_1 to differentiate between command and data contents in DBBIN when IBF is set. The UPI may also write the upper 4-bits of its accumulator to the upper 4-bits of the STATUS register. These bits are thus user definable.

The UPI can test the flags at any time during its internal program execution. It essentially "polls" the STATUS register for changes. If faster response is needed to master commands and data, the UPI's internal interrupt structure can be used. If IBF interrupts are enabled, a master write to DBBIN (either command or data) sets IBF which generates an internal CALL to location 03H in program memory. At this point, working register contents can be saved using bank switching, the accumulator saved in a spare working register, and the DBBIN register read and serviced. The interrupt logic for the IBF interrupt is shown in Figure 4. A few observations concerning this logic are appropriate. Note that if the master writes to DBBIN while the UPI is still servicing the last IBF interrupt (a RETR instruction has not been executed), the IBF Interrupt Pending line is made high which causes a new CALL to 03H as soon as the first RETR is executed. No EN I (Enable Interrupt) instruction is needed to rearm the interrupt logic as is needed in an 8080 or 8085A system; the RETR performs this function. Also note that executing a DIS I to disable further IBF interrupts does not clear a pending interrupt. Only a CALL to location 03H or RESET clears a pending IBF interrupt.

Keeping in mind that the actual master/UPI protocol is dependent on the application, probably the best way to illustrate correct protocol is by example. Let's consider using the UPI as a simple parallel I/O device. (This is a trivial application but it embodies all of the important protocol considerations.) Since the UPI may be either interrupt or non-interrupt driven internally, both cases are considered.

Let's take the easiest configuration first; using the UPI PORT 1 as an 8-bit output port. From the UPI's point-of-view, this is an input-only application since all that is required is that the UPI input data from the master. Once the master writes data to the UPI, the UPI reads the DBBIN register and transfers the data to PORT 1. No testing for commands versus data is needed since the UPI "knows" it only performs one task—no commands are needed.

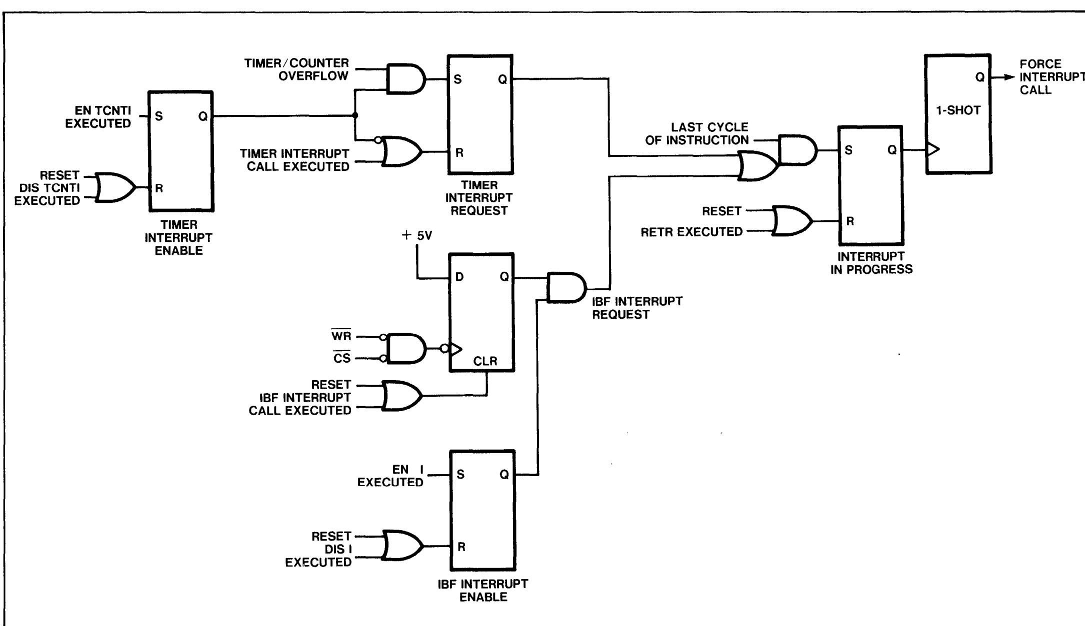


Figure 4. UPI-41A Interrupt Structure

Non-interrupt driven UPI software is shown in Figure 5A while Figure 5B shows interrupt based software. For Figure 5A, the UPI simply waits until it sees IBF go high indicating the master has written a data byte to DBBIN. The UPI then reads DBBIN, transfers it to PORT 1, and returns to waiting for the next data. For the interrupt-driven UPI, Figure 5B, once the EN I instruction is executed, the UPI simply waits for the IBF interrupt before handling the data. The UPI could handle other tasks during this waiting time. When the master writes the data to DBBIN, an IBF interrupt is generated which performs a CALL to location 03H. At this point the UPI reads DBBIN (no testing of IBF is needed since an IBF interrupt implies that IBF is set), transfers the data to PORT 1, and executes an RETR which returns program flow to the main program.

Software for the master 8085A is included in Figure 5C. The only requirement for the master to output data to the UPI is that it check the UPI to be sure the previous data had been taken before writing new data. To accomplish this the master simply reads the STATUS register looking for IBF=0 before writing the next data.

```
;
; UPI INPUT ONLY EXAMPLE—PORT 1 USED AS OUTPUT PORT
;       UPI POLLS IBF FOR DATA
;
RESET:  JNIBF  RESET      ; WAIT ON IBF FOR INPUT
        IN     A,DBB      ; INPUT THERE, SO READ IT
        OUTL   P1,A       ; TRANSFER DATA TO PORT 1
        JMP    RESET      ; GO WAIT FOR NEXT DATA
```

Figure 5A. Single Output Port Example—Polling

```
;
; UPI INPUT ONLY EXAMPLE—PORT 1 USED AS OUTPUT PORT
;       DATA INPUT IS INTERRUPT-DRIVEN ON IBF
;
RESET:  EN     I          ; ENABLE IBF INTERRUPTS
        JMP    RESET+1    ; LOOP WAITING FOR INPUT
IBFINT: IN     A,DBB      ; READ DATA FROM DBBIN
        OUTL   P1,A       ; TRANSFER DATA TO PORT 1
        RETR              ; RETURN WITH RESTORE
```

Figure 5B. Single Output Port Example—Interrupt

```
;
; 8085 SOFTWARE FOR UPI INPUT-ONLY EXAMPLE
;       DATA FOR OUTPUT IS PASSED IN REG. C
;
UPIOUT: IN     STATUS     ; READ UPI STATUS
        ANI    IBF        ; LOOK AT IBF
        JNZ    UPIOUT     ; WAIT FOR IBF=0
        MOV    A,C        ; GET DATA FROM C
        OUT    DBBIN      ; OUTPUT DATA TO DBBIN
        RET               ; DONE, RETURN
```

Figure 5C. 8085A Code for Single Output Port Example

Figure 6A illustrates the case where UPI PORT 2 is used as an 8-bit input port. This configuration is termed UPI output-only as the master does not write (input) to the UPI but simply reads either the STATUS or the DBBOUT registers. In this example only the OBF flag is used. OBF signals the master that the UPI has placed new port data in DBBOUT. The UPI loops testing OBF. When OBF is clear, the master has read the previous data and UPI then reads its input port (PORT 2) and places this data in DBBOUT. It then waits on OBF until the master reads DBBOUT before reading the input port again. When the master wishes to read the input port data, Figure 6B, it simply checks for OBF being set in the STATUS register before reading DBBOUT. While this technique illustrates proper protocol, it should be noted that it is not meant to be a good method of using the UPI as an input port since the master would never get the newest status of the port.

```
;
; UPI OUTPUT ONLY EXAMPLE—PORT 2 USED AS INPUT PORT
;       PORT DATA IS AVAILABLE IN DBBOUT
;
RESET:  JOBF   RESET      ; LOOP IF OBF=1 (DATA NOT READ)
        IN     A,P2       ; DBBOUT CLEAR, READ PORT
        OUT    DBB,A      ; TRANSFER PORT DATA TO DBBOUT
        JMP    RESET      ; WAIT FOR MASTER TO READ DATA
```

Figure 6A. Single Input Port Example

```
;
; 8085 SOFTWARE FOR UPI OUTPUT—ONLY EXAMPLE
;       INPUT DATA RETURNED IN REG. A
;
UPIIN:  IN     STATUS     ; READ UPI STATUS
        ANI    OBF        ; LOOK AT OBF
        JZ     UPIIN      ; WAIT UNTIL OBF=1
        IN     DBBOUT     ; READ DBBOUT
        RET               ; RETURN WITH DATA IN A
```

Figure 6B. 8085A Single Input Port Code

The above examples can easily be combined. Figure 7 shows UPI software to use PORT 1 as an output port simultaneously with PORT 2 as an input port. The program starts with the UPI checking IBF to see if the master has written data destined for the output port into DBBIN. If IBF is set, the UPI reads DBBIN and transfers the data to the output port (PORT 1). If IBF is not set or once the data is transferred to the output port if it was, OBF is tested. If OBF is reset (indicating the master has read DBBOUT), the input port (PORT 2) is read and transferred to DBBOUT. If OBF is set, the master has yet to read DBBOUT so the program just loops back to test IBF.

The master software is identical to the separate input/output examples; the master must test IBF

```
;
; UPI INPUT/OUTPUT EXAMPLE—PORT 1OUTPUT, PORT 2 INPUT
;
RESET:  JNIBF  OUT1     ; IF IBF=0, DO OUTPUT
        IN     A, DBB   ; IF IBF=1, READ DBBIN
        OUTL   P1, A    ; TRANSFER DATA TO PORT 1
OUT1:   JOBF   RESET    ; IF OBF=1, GO TEST IBF
        IN     A, P2    ; IF OBF=0, READ PORT 2
        OUT    DBB, A   ; TRANSFER PORT DATA TO DBBOUT
        JMP    RESET    ; GO CHECK FOR INPUT
```

Figure 7. Combination Output/Input Port Example

and OBF before writing output port data into DBBIN or before reading input port from DBBOUT respectively.

In all of the three examples above, the UPI treats information from the master solely as data. There has been no need to check if DBBIN information is a command rather than data since the applications do not require commands. But what if both PORTs 1 and 2 were used as output ports? The UPI needs to know into which port to put the data. Let's use a command to select which port.

Recall that both commands and data pass through DBBIN. The state of the A_0 pin at the time of the write to DBBIN is used to distinguish commands from data. By convention, DBBIN writes with $A_0=0$ are for data, and those with $A_0=1$ are commands. When DBBIN is written into, F_1 (FLAG 1) is set to the state of A_0. The UPI tests F_1 to determine if the information in the DBBIN register is data or command.

For the case of two output ports, let's assume that the master selects the desired port with a command prior to writing the data. (We could just use F_1 as a port select but that would not illustrate the subtle differences between commands and data). Let's define the port select commands such that BIT 1=1 if the next data is for PORT 1 (Write PORT 1=0000 0010) and BIT 2=1 if the next data is for PORT 2 (Write PORT 2=0000 0100). (The number of the set bit selects the port.) Any other bits are ignored. This assignment is completely arbitrary; we could use any command structure, but this one has the advantage of being simple.

Note that the UPI must "remember" from DBBIN write to write which port has been selected. Let's use F_0 (FLAG 0) for this purpose. If a Write PORT 1 command is received, F_0 is reset. If the command is Write PORT 2, F_0 is set. When the UPI finds data in DBBIN, F_0 is interrogated and the data is loaded into the previously selected port. The UPI software is shown in Figure 8A.

```
;
; UPI DUAL OUTPUT PORT EXAMPLE—BOTH PORT 1 AND 2 OUTPUTS
;        COMMAND SELECTS DESIRED PORT
;        WRITE PORT 1—0000 0010 (02H)
;        WRITE PORT 2—0000 0100 (04H)
;
;        FLAG 0 USED TO REMEMBER WHICH PORT WAS SELECTED
;        BY LAST COMMAND.
;
RESET:  JNIBF  RESET    ; WAIT FOR MASTER INPUT
        IN     A, DBB   ; READ INPUT
        JF1    CMD      ; IF F1=1, COMMAND INPUT
        JFO    PORT2    ; INPUT IS DATA, TEST FO
        OUTL   P1,A     ; FO=0, SO OUTPUT TO PORT 1
        JMP    RESET    ; WAIT FOR NEXT INPUT
PORT2:  OUTL   P2,A     ; FO=1, SO OUTPUT TO PORT 2
        JMP    RESET    ; WAIT FOR NEXT INPUT
CMD:    JB1    PT1      ; TEST COMMAND BITS (BIT 1)
        JB2    PT2      ; TEST BIT 2
        JMP    RESET    ; NEITHER BIT SET, WAIT FOR INPUT
PT1:    CLR    FO       ; PORT 1 SELECTED, CLEAR FO
        JMP    RESET    ; WAIT FOR INPUT
PT2:    CLR    FO       ; PORT 2 SELECTED, SET FO
        CPL    FO
        JMP    RESET    ; WAIT FOR INPUT
```

Figure 8A. Dual Output Port Example

Initially, the UPI simply waits until IBF is set indicating the master has written into DBBIN. Once IBF is set, DBBIN is read and F_1 is tested for a command. If $F_1=1$, the DBBIN byte is a command. Assuming a command, BIT 1 is tested to see if the command selected PORT 1. If so, F_0 is cleared and the program returns to wait for the data. If BIT 1=0, BIT 2 is tested. If BIT 2 is set, PORT 2 is selected so F_0 is set. The program then loops back waiting for the next master input. This input is the desired port data. If BIT 2 was not set, F_0 is not changed and no action is taken.

When IBF=1 is again detected, the input is again tested for command or data. Since it is necessarily data, DBBIN is read and F_0 is tested to determine which port was previously selected. The data is then output to that port, following which the program waits for the next input. Note that since F_0 still selects the previous port, the next input could be more data for that port. The port selection command could be thought of as a port select flip-flop control; once a selection is made, data may be repeatedly written to that port until the other port is selected. Master software, Figure 8B, simply must check IBF before writing either a command or data to DBBIN. Otherwise, the master software is straightforward.

For the sake of completeness, UPI software for implementing two input ports is given in Figure 9. This case is simpler than the dual output case since the UPI can assume that all writes to DBBIN are port selection commands so no command/data testing is required. Once the Port Read command is input, the selected port is read and the port data is placed in DBBOUT. Note that in this case F_0 is used as a UPI error indicator. If the master happened to issue an invalid command (a command without either BIT 1 or 2 set), F_0 is set to notify the master that the UPI did not know how to interpret the command. F_0 is also set if the master commanded a port read before it had read DBBOUT from the previous command. The UPI simply tests OBF just prior to loading DBBOUT and if OBF=1, F_0 is set to indicate the error.

All of the above examples are, in themselves, rather trivial applications of the UPI although they could easily be incorporated as one of several tasks in a UPI handling multiple small tasks. We have covered them primarily to introduce the UPI concept and to illustrate some master/UPI protocol. Before moving on to more realistic UPI applications, let's discuss two UPI features that do not directly relate to the master/UPI protocol but greatly enhance the UPI's transfer capability.

In addition to the OBF and IBF bits in the STATUS register, these flags can also be made available directly on two port pins. These port pins can then be used as interrupt sources to the master. By executing an EN FLAGS instruction, PORT 2 pin 4 reflects the condition of OBF and PORT 2 pin 5 reflects the inverted condition of IBF ($\overline{IBF}$). These dedicated outputs can then be enabled or disabled via their respective port bit values; i.e., P_{24} reflects OBF as long as an instruction is executed which sets P_{24} (i.e. ORL P2,#10H). The same action applies to the $\overline{IBF}$ output except P_{25} is used. Thus P_{24} may serve as a DATA AVAILABLE interrupt output. Likewise for P_{25} as a READY-TO-ACCEPT-DATA interrupt. This greatly simplifies interrupt-driven master-slave data transfers.

```
;
; 8085 SOFTWARE FOR DUAL OUTPUT PORT EXAMPLE
;       THIS ROUTINE WRITES DATA IN REG. C TO PORT 1
;       (SAME ROUTINE FOR PORT 2—JUST CHANGE COMMAND)
;
PORT1:  IN    STATUS       ; READ UPI STATUS
        ANI   IBF          ; LOOK AT IBF
        JNZ   PORT1        ; WAIT UNTIL IBF=0
        MVI   A, 00000010B ; LOAD WRITE PORT1 CMD
        OUT   UPICMD       ; OUTPUT TO UPI COMMAND PORT
P1:     IN    STATUS       ; READ UPI STATUS AGAIN
        ANI   IBF          ; LOOK AT IBF
        JNZ   P1           ; WAIT UNTIL COMMAND ACCEPTED
        MOV   A, C         ; GET DATA FROM C
        OUT   DBBIN        ; OUTPUT TO DBBIN
        RET                ; DONE, RETURN
```

Figure 8B. 8085A Dual Output Port Example Code

The UPI also supports a DMA transfer interface. If an EN DMA instruction is executed, PORT 2 pin 6 becomes a DMA Request (DRQ) output and P_{27} becomes a high impedance DMA Acknowledge ($\overline{DACK}$) input. Any instruction which would normally set P_{26} now sets DRQ. DRQ is cleared when $\overline{DACK}$ is low and either $\overline{RD}$ or $\overline{WR}$ is low. When $\overline{DACK}$ is low, $\overline{CS}$ and A0 are forced low internally which allows data bus transfers between DBBOUT or DBBIN to occur, depending upon whether $\overline{WR}$ or $\overline{RD}$ is true. Of course, the function requires the use of an external DMA controller.

```
;
; UPI DUAL INPUT PORT EXAMPLE—BOTH PORT 1 AND 2 INPUTS
;       COMMAND SELECTS WHICH PORT IS TO BE READ
;       FLAG 0 USED AS ERROR FLAG
;
RESET:  JNIBF  RESET    ; WAIT FOR INPUT
        CLR    F0       ; CLEAR ERROR FLAG
        IN     A, DBB   ; READ INPUT (COMMAND)
        JB1    PT1      ; TEST BIT 1 (PORT 1)
        JB2    PT2      ; TEST BIT 2 (PORT 2)
ERROR:  CPL    F0       ; ERROR—COMPLEMENT F0
        JMP    RESET    ; WAIT FOR INPUT
PT1:    IN     A, P1    ; READ PORT 1
        JOBF   ERROR    ; TEST OBF BEFORE LOADING DBBOUT
        OUT    DBB, A   ; LOAD PORT 1 DATA INTO DBBOUT
        JMP    RESET    ; WAIT FOR INPUT
PT2:    IN     A, P2    ; READ PORT 2
        JOBF   ERROR    ; TEST OBF BEFORE LOADING DBBOUT
        OUT    DBB, A   ; LOAD PORT 2 DATA INTO DBBOUT
        JMP    RESET    ; WAIT FOR INPUT
```

Figure 9. Dual Input Port Example

Now that we have discussed the aspects of the UPI protocol and data transfer interfaces, let's move on to the actual applications.

EXAMPLE APPLICATIONS

Each of the following three sections presents the hardware and software details of a UPI application. Each application utilizes one of the protocols mentioned in the last section. The first example is a simple 8-digit LED display controller. This application requires only that the UPI perform input operations from the DBBIN; DBBOUT is not used. The reverse is true for the second application: a sensor matrix controller. The final application involves both DBBOUT and DBBIN operations: a combination serial/parallel I/O device.

The core master processor system with which these applications were developed is the iSBC 80/30 single board computer. This board provides an especially convenient UPI environment since it contains a dedicated socket specifically interfaced for the UPI-41A. The 80/30 uses the 8085A as the master processor. The I/O and peripheral complement on the 80/30 include 12 vectored priority interrupts (8 on an 8259 Programmable Interrupt Controller and 4 on the 8085A itself), an 8253 Programmable Interval Timer supplying three 16-bit programmable timers (one is dedicated as a programmable baud rate generator), a high speed serial channel provided by a 8251 Programmable USART, and 24 parallel I/O lines implemented with an 8255A Programmable Parallel Interface. The memory complement contains 16K bytes of RAM using 2117 16K bit Dynamic RAMs and the 8202 Dynamic RAM Controller, and up to 8K bytes of ROM/EPROM with sockets compatible with 2716, 2758, or 2332 devices. The 80/30's RAM uses a dual port architecture. That is, the memory can be considered a global system resource, accessible from the on-board 8085A as well as from remote CPUs and other devices via the MULTIBUS. The 80/30 contains MULTIBUS control logic which allows up to 16 80/30s or other bus masters to share the same system bus. (More detailed information on the iSBC 80/30 and other iSBC products may be found in the latest Intel *Systems Data Catalog.*)

A block diagram of the iSBC 80/30 is shown in Figure 10. Details of the UPI interface are shown in Figure 11. This interface decodes the UPI registers in the following format:

Register	Operations
Read STATUS	IN E5H
Write DBBIN (command)	OUT E5H
Read DBBOUT (data)	IN E4H
Write DBBIN (data)	OUT E4H

8-Digit Multiplexed LED Display

The traditional method of interfacing an LED display with a microprocessor is to use a data latch along with a BDC-to-7-segment decoder for each digit of the display. Thus two ICs, seven current limiting resistors, and about 45 connections are required for each digit. These requirements are, of course, multiplied by the total number of digits desired. The obvious disadvantages of this method are high parts count and high power dissipation since each digit is "ON" continuously. Instead, a scheme of time multiplexing the display can be used to decrease both parts count and power dissipation.

Display multiplexing basically involves connecting the same segment (a, b, c, d, e, f, or g) of each digit in parallel and driving the common digit element (anode or cathode) of each digit separately. This is shown schematically in Figure 12. The various digits of the display are not all on at once; rather, only one digit at a time is energized. As each digit is energized, the appropriate segments for that digit are turned on. Each digit is enabled in this way, in sequence, at a rate fast enough to ensure that each digit appears to be "ON" continuously. This implies that the display must be "refreshed" at periodic intervals to keep the digits flicker-free. If the CPU had to handle this task, it would have to suspend normal processing, go update the display, and then return to its normal flow. This extra burden is ideally handled by a UPI. The master CPU could simply give characters to the UPI and let the UPI do the actual segment decoding, display multiplexing, and refreshing.

As an example of this technique, Figure 13 shows the UPI controlling an 8-digit LED display. All digit segments are connected in parallel and are driven through segment drivers by the UPI PORT 1. The lower 3 bits of PORT 2 are inputs to a 3-to-8 decoder which selects an individual digit through a digit driver. A fourth PORT 2 line is used as a decoder enable input. The remaining PORT 2 lines plus the TEST 0 and TEST 1 inputs are available for other tasks.

Internally, the UPI uses the counter/timer in the interval timer mode to define the interval between display refreshes. Once the timer is loaded with the desired interval and started, the UPI is free to handle other tasks. It is only when a timer overflow interrupt occurs that the UPI handles the short display multiplexing routine. The display multiplexing can be considered a background task which is entirely interrupt-driven. The amount of time spent multiplexing is such that there is ample time to handle a non-timer task in the UPI foreground. (We'll discuss this timing shortly.)

When a timer interrupt occurs, the UPI turns off all digits via the decoder enable. The next digit's segment contents are retrieved from the internal data memory and output via PORT 1 to the segment drivers. Finally, the next digit's location is placed on PORT 2 (P_{20}–P_{22}) and the decoder enabled. This displays the digit's segment information until the next interrupt. The timer is then restarted for the next interval. This process continues repeatedly for each digit in sequence.

As a prelude to discussing the UPI software, let's examine the internal data memory structure used in this application, Figure 14. This application requires only 14 of the 64 total data memory locations. The top eight locations are dedicated to the Display Map; one location for each digit. These locations contain the segment and decimal point information for each character. Just how characters are loaded into this section of memory is covered shortly. Register R_7 of Register Bank 1 is used as the temporary Accumulator store during the interrupt service routines. Register R_3 stores the digit number of the next digit to be displayed. R_2 is a temporary storage register for characters during input routine. R_0 is

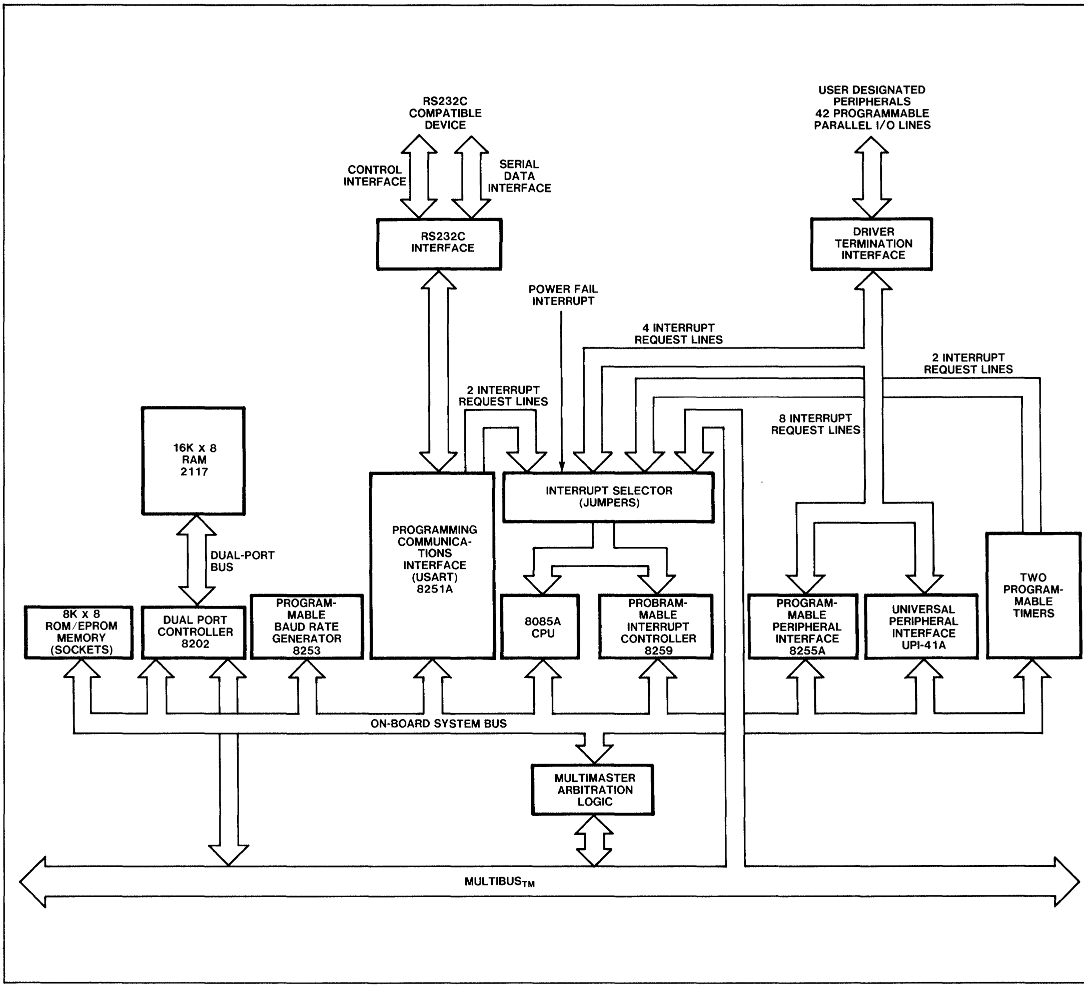


Figure 10. iSBC 80/30 Block Diagram

the offset pointer pointing to the Display Map location of the next digit. That makes 12 locations so far. The remaining two locations are the two stack locations required to store the return address plus status during the timer and input interrupt service routines. The remaining unused locations, all of Register Bank 0, 14 bytes of stack, 4 in Register Bank 1, and 24 general purpose RAM locations, are all available for use by any foreground task.

The UPI software consists of only three short routines. One, INIT, is used strictly during initialization. DISPLA is the multiplexing routine called at a timer interrupt. INPUT is the character input handler called at an IBF interrupt. The flow charts for these routines are shown in Figures 14A through 14C.

INIT initializes the UPI by simply turning off all segment and digit drivers, filling the Display Map with blank characters, loading and starting the timer, and enabling both timer and IBF interrupts. Although the flow chart shows the program looping at this point, it is here that the code for any foreground task is inserted. The only restrictions on this foreground task are that it not use I/O lines dedicated to the display and that it not require dedicated use of the timer. It could share the timer if precautions are taken to ensure that the display will still be refreshed at the required interval.

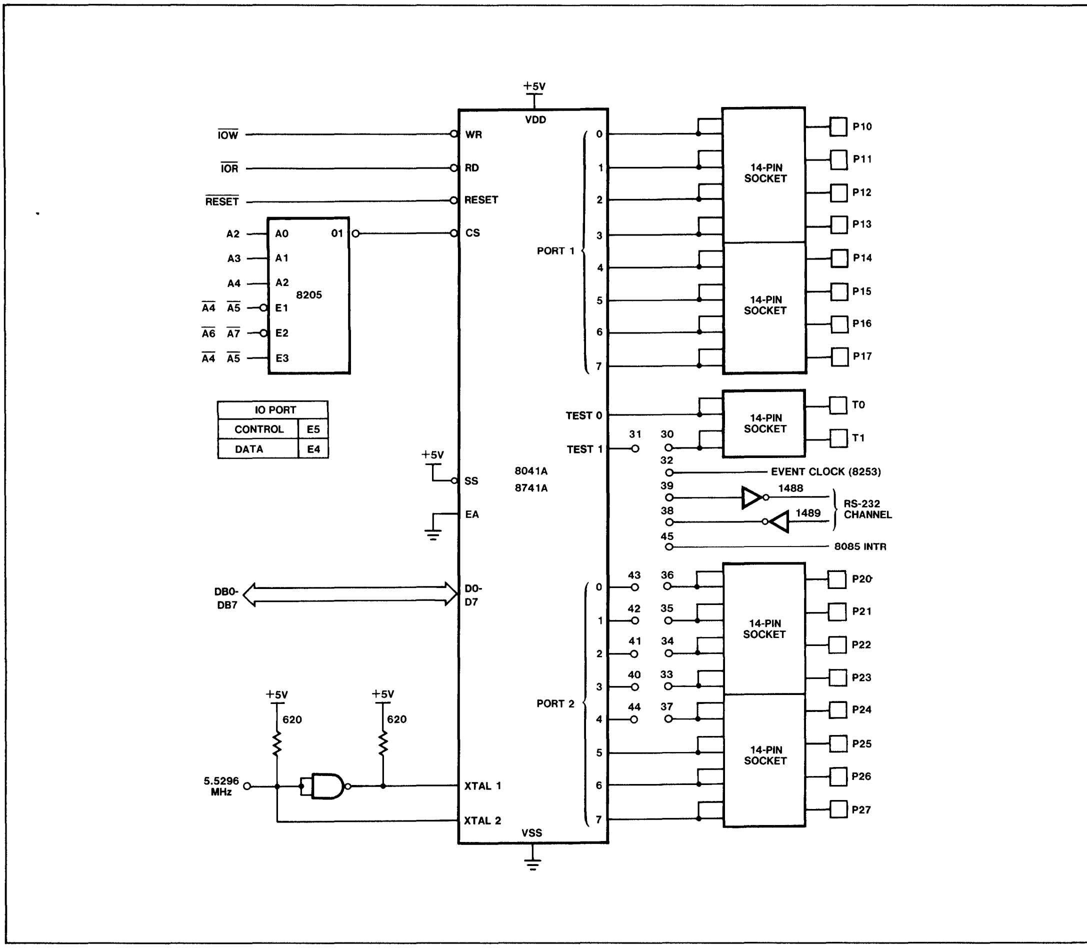


Figure 11. UPI Interface on iSBC 80/30

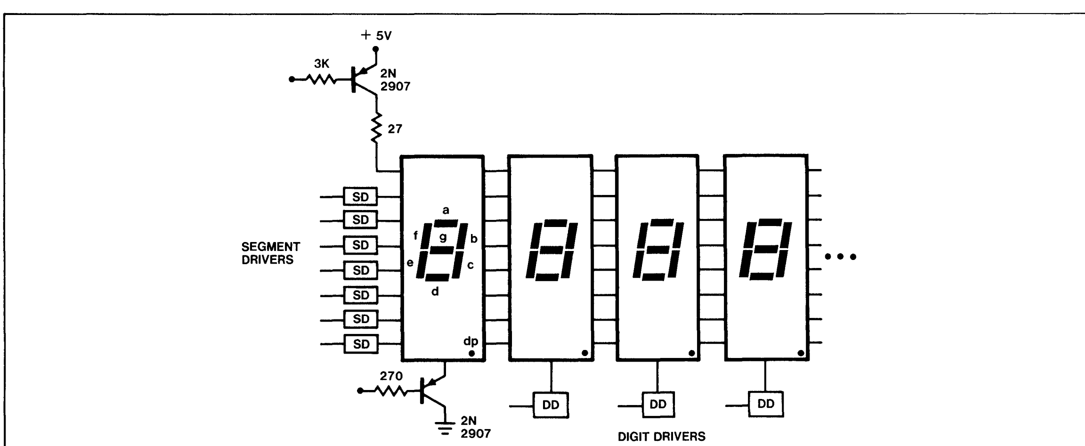


Figure 12. LED Multiplexing

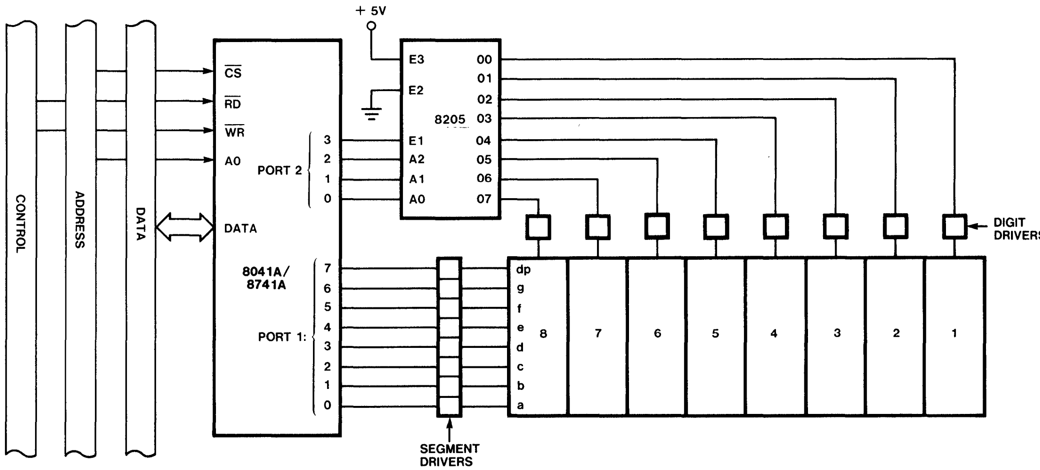
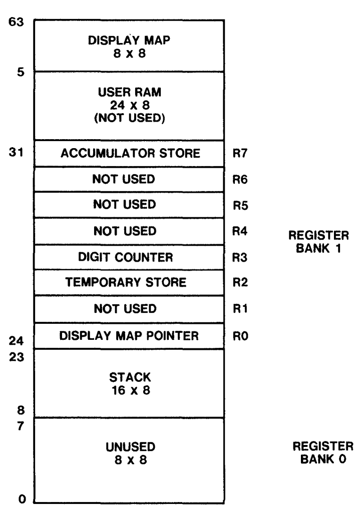


Figure 13. UPI Controlled 8-Digit LED Display



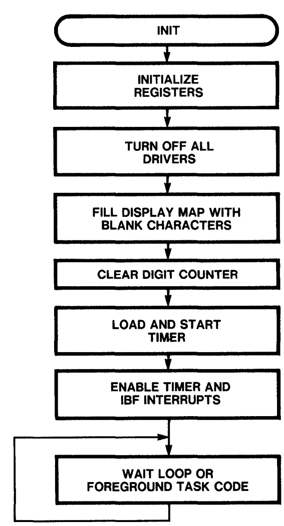


Figure 14. LED Display Controller Data Memory Allocation

Figure 14A. INIT Routine Flow

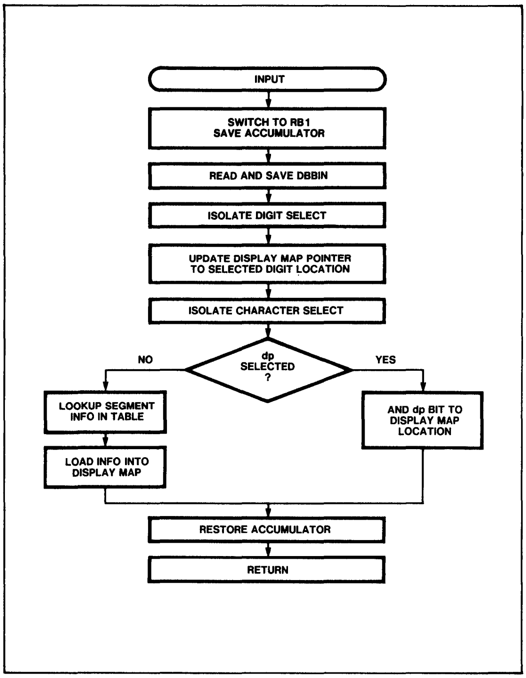


Figure 14B. INPUT Routine Flow

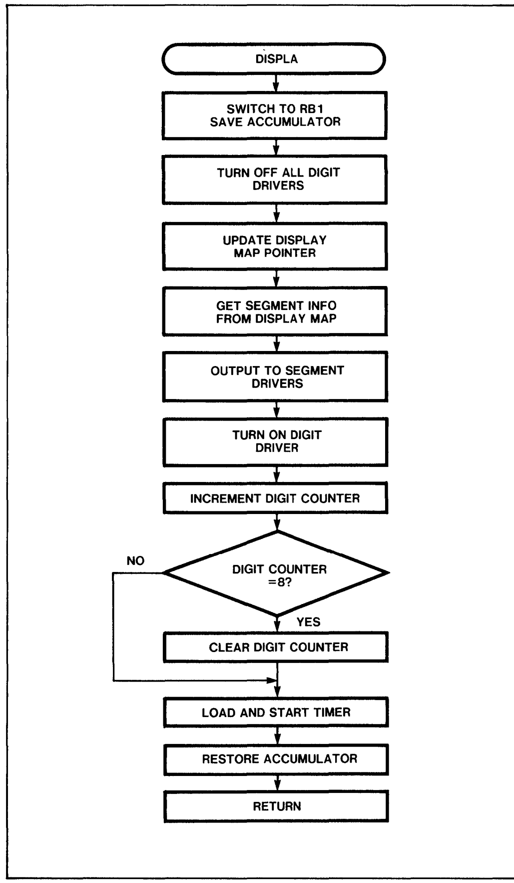
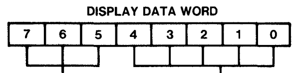


Figure 14C. DISPLA Routine Flow

The INPUT routine handles the character input. It is called when an IBF interrupt occurs. After the usual swapping of register banks and saving of the accumulator, DBBIN is read and stored in register R_2. DBBIN contains the Display Data Word. The format for this word, Figure 15, has two fields: Digit Select and Character Select. The Digit Select field selects the digit number into which the character from the Character Select field is placed. Notice that the character set is not limited strictly to numerics, some alphanumeric capability is provided. Once DBBIN is read, the offset for the selected digit is computed and placed in the Display Map Pointer R_0. Next the segment information for the selected character is found through a look-up table starting in page 3 of the program memory. This segment information is then stored at the location pointed at by the Display Map Pointer. If the Character Select field specified a decimal point, the segment corresponding to the decimal point is ANDed into the present segment information for that digit. After the accumulator is restored, execution is returned to the main program.

The DISPLA routine simply implements the multiplexing actions described earlier. It is called whenever a timer interrupt occurs. After saving pre-interrupt status by switching register banks and storing the Accumulator, all digit drivers are turned off. The Display Map Pointer is then updated using the Current Digit Register to point at that digit's segment information in the Display Map. This information is output to PORT 1; the segment drivers. The number of the current digit, R_3, is then sent to the digit select decoder and the decoder is enabled. This turns on the current digit. The digit counter is incremented and tested to see if all eight digits have been refreshed. If so, the digit counter is reset to zero. If not, nothing is done. Finally, the timer is loaded and restarted, the Accumulator is restored, and the routine returns execution to the main program. Thus DISPLA refreshes one digit each time it is CALLed by the timer interrupt. The digit remains on until the next time DISPLA is executed.

The UPI software listing is included as Appendix A1. Appendix A2 shows the 8085A test routine used



DIGIT SELECT

7	5	6	DIGIT
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

CHARACTER SELECT

4	3	2	1	0	CHAR
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	2
0	0	0	1	1	3
0	0	1	0	0	4
0	0	1	0	1	5
0	0	1	1	0	6
0	0	1	1	1	7
0	1	0	0	0	8
0	1	0	0	1	9
0	1	0	1	0	A
0	1	0	1	1	b
0	1	1	0	0	C
0	1	1	0	1	d
0	1	1	1	0	E
0	1	1	1	1	F
1	0	0	0	0	•
1	0	0	0	1	G
1	0	0	1	0	H
1	0	0	1	1	I
1	0	1	0	0	J
1	0	1	0	1	L
1	0	1	1	0	n
1	0	1	1	1	o
1	1	0	0	0	P
1	1	0	0	1	r
1	1	0	1	0	t
1	1	0	1	1	U
1	1	1	0	0	y
1	1	1	0	1	-
1	1	1	1	0	
1	1	1	1	1	blank

Figure 15. LED Display Controller Display Data Word Format

to display the contents of a display buffer on the display. The 8085A software takes care of the display digit numbering. Since the application is input-only for the UPI, the only protocol required is that the master must test IBF before writing a Display Data Word into DBBIN.

On the iSBC 80/30, the UPI frequency is at 5.5296 MHz. To obtain a flicker-free display, the whole display must be refreshed at a rate of 50 Hz or greater.

If we assume a 50 Hz refresh rate and an 8-digit display, this means the DISPLA routine must be CALLed 50×8 or 400 times/sec. This transfers, using the timer interval of 87 μs at 5.5296 MHz, to a timer count of 227. (Recall from the UPI-41A *User's Manual* that the timer is an "8-bit up-counter".) Hence the TIME equate of 227D in the UPI listing. Obviously, different frequency sources or display lengths would require that this equate be modified.

With the UPI running at 5.5296 MHz, the instruction cycle time is 2.713 μs. The DISPLA routine requires 28 instruction cycles, therefore, the routine executes in 76 μs. Since DISPLA is CALLed 400 times/sec, the total time spent refreshing the display during one second is then 30 ms or 3% of the total UPI time. This leaves 97.0% for any foreground tasks that could be added.

While the basic UPI software is useful just as it stands, there are several enhancements that could be incorporated depending on the application. Auto-incrementing of the digit location could be added to the input routine to alleviate the need for the master to keep track of digit numbers. This could be (optionally) either right-handed or left-handed entry a la TI or HP calculators. The character set could be easily modified by simply changing the lookup table. The display could be expanded to 16 digits at the expense of one additional PORT 2 digit select line, the replacement of the 3-to-8 decoder with a 4-to-16 decoder, and 8 more Display Map locations.

Now let's move on to a slightly more complex application that is UPI output-only—a sensor matrix controller.

Sensor Matrix Controller

Quite often a microprocessor system is called upon to read the status of a large number of simple SPST switches or sensors. This is especially true in a process or industrial control environment. Alarm systems are also good examples of systems with a large sensor population. If the number of sensors is small, it might be reasonable to dedicate a single input port pin for each sensor. However, as the number of sensors increase, this technique becomes very wasteful. A better arrangement is to configure the sensors in a matrix organization like that shown in Figure 16. This arrangement of 16 sensors requires only 4 input and 4 output lines; half the number needed if dedicated inputs were used. The line saving becomes even more substantial as the number of sensors increases.

In Figure 16, the basic operation of the matrix involves scanning individual row select lines in sequence while reading the column return lines. The state of any particular sensor can then be determined by decoding the row and column information. The typical configuration pulls up the column return lines and the selected row is held low. Deselected rows are held high. Thus a return line remains high for an open sensor on the selected row and is pulled low for a closed sensor. Diode isolation is used to prevent a phantom closure which would occur when a sensor is closed on a selected row and there are two or more closures on a deselected row. Germanium diodes are used to provide greater noise margin at the return line input.

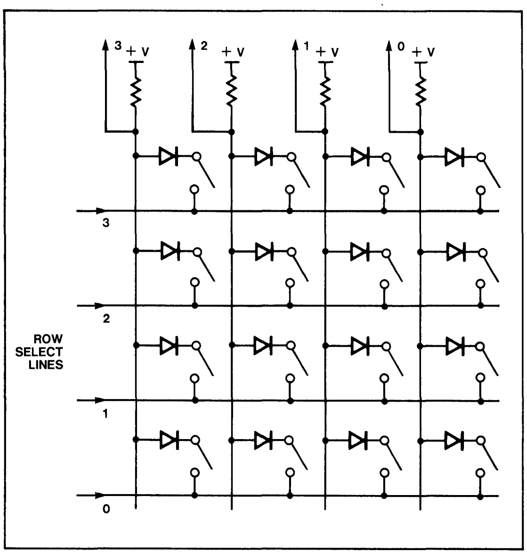


Figure 16. 4X4 Sensor Matrix

If the main processor was required to control such a matrix it would periodically have to output at the row port and then read the column return port. The processor would need to maintain in memory a map of the previous state of the matrix. A comparison of the new return information to the old information would then be made to determine whether a sensor change had occurred. Any changes would be processed as needed. A row counter and matrix map pointer also require maintenance each scan. Since in most applications sensors change very slowly compared to most processing actions, the processor probably would scan the rows only periodically with other tasks being processed between scans.

Rather than require the processor to handle the rather mundane tasks of scanning, comparing, and decoding the matrix, why not use a dedicated processor? The UPI is perfect.

Figure 17 shows a UPI configuration for controlling up to 128 sensors arranged in a 16X8 matrix. The 4-to-16 line decoder is used as the row selector to save port pins and provides the expansion to 128 sensors over the maximum of 64 sensors if the port had been used directly. It also helps increase the port drive capability. The column return lines go directly into PORT 1. Features of this design include complete matrix management. As the UPI scans the matrix it compares its present status to the previous scan. If any change is detected, the location of the change is decoded and loaded, along with the sensor's present state, into DBBOUT. This byte is called a Change Word. The Master processor has only to read one byte to determine the status and coordinate of a changed sensor. If the master had not read a previous Change Word in DBBOUT (OBF=1) before a new sensor change is detected, the new Change

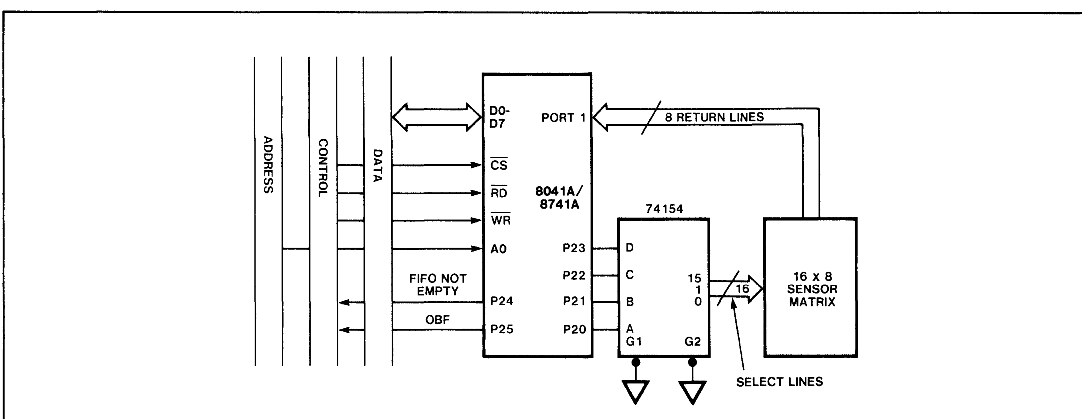


Figure 17. 128 Sensor Matrix Controller

Word is loaded into an internal FIFO. This FIFO buffers up to 40 changes before it fills. The status of the FIFO and OBF is made available to the master either by polling the UPI STATUS register, Figure 18A, or as interrupt sources on port pins P_{24} and P_{25} respectively, Figure 17. The FIFO NOT EMPTY pin and bit are true as long as there are changes not yet read in the FIFO. As long as the FIFO is not empty, the UPI monitors OBF and loads new Change Words from the FIFO into DBBOUT. Thus, the UPI provides complete FIFO management.

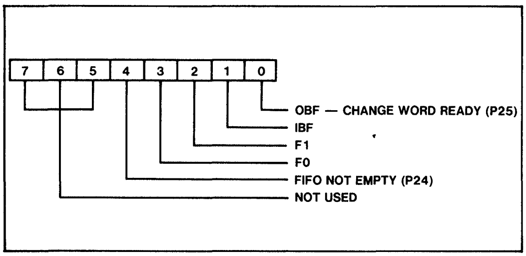


Figure 18A. Sensor Matrix Status Register Format

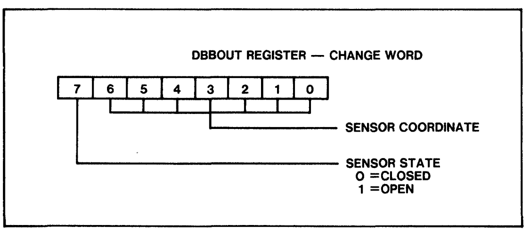


Figure 18B. Sensor Matrix Change Word Format

Internally, the matrix scanning software is programmed to run as a foreground task. This allows the timer/counter to be used by any background task although the hardware configuration leaves only 2 inputs (TEST 0 and TEST 1) plus 2 I/O port pins available. Also, to add a background task, the FIFO would have to be made smaller to accommodate the needed register and data memory space. (It would be possible however to turn the table here and make the scanning software timer/counter interrupt-driven where the timer times the scan interval.)

The data memory organization for this application is shown in Figure 19. The upper 16 bytes form the Matrix Map and store the sensor states from the previous scan; one bit for each sensor. The Change Word FIFO occupies the next 40 locations. (The top and bottom addresses of this FIFO are treated as equate variables in the program so that the FIFO size may easily be changed to accommodate the register needs of other tasks.) Register R_0 serves as a pointer into the matrix map area for comparisons and updates of the sensor status. R_1 is a general FIFO pointer. The FIFO is implemented as a circular buffer with In and Out pointer registers which are stored in R_4 and R_5 respectively. These registers are moved into FIFO pointer R_1 for actual transfers into or out of the FIFO. R_2 is the Row Select Counter. It stores the number of the row being scanned.

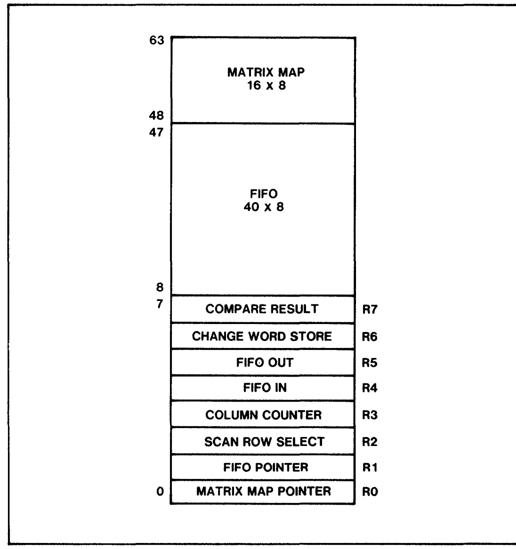


Figure 19. Sensor Matrix Data Memory Map

Register R_3 is the Column Counter. This counter is normally set to 00H; however, when a change is detected somewhere in a particular row, it is used to inspect each sensor status bit individually for a change. When a changed counter sensor bit is found, the Row Select Counter and Column Counter are combined to give the sensor's matrix coordinate. This coordinate is temporarily stored in the Change Word Store, register R_6. Register R_7 is the Compare Result. As each row is scanned, the return information is Exclusive-OR'd with the return information from the previous scan of that row. The result of this operation is stored in R_7. If R_7 is zero, there have been no changes on that row. A non-zero result indicates at least one changed sensor.

The basic program operation is shown in the flow chart of Figure 20. At RESET, the software initializes the working registers, the ports, and clears the STATUS register. To get a starting point from which to perform the sensor comparisons, the current status of the matrix is read and stored in the Matrix Map. At this point, the UPI begins looking for changed sensors starting with the first row.

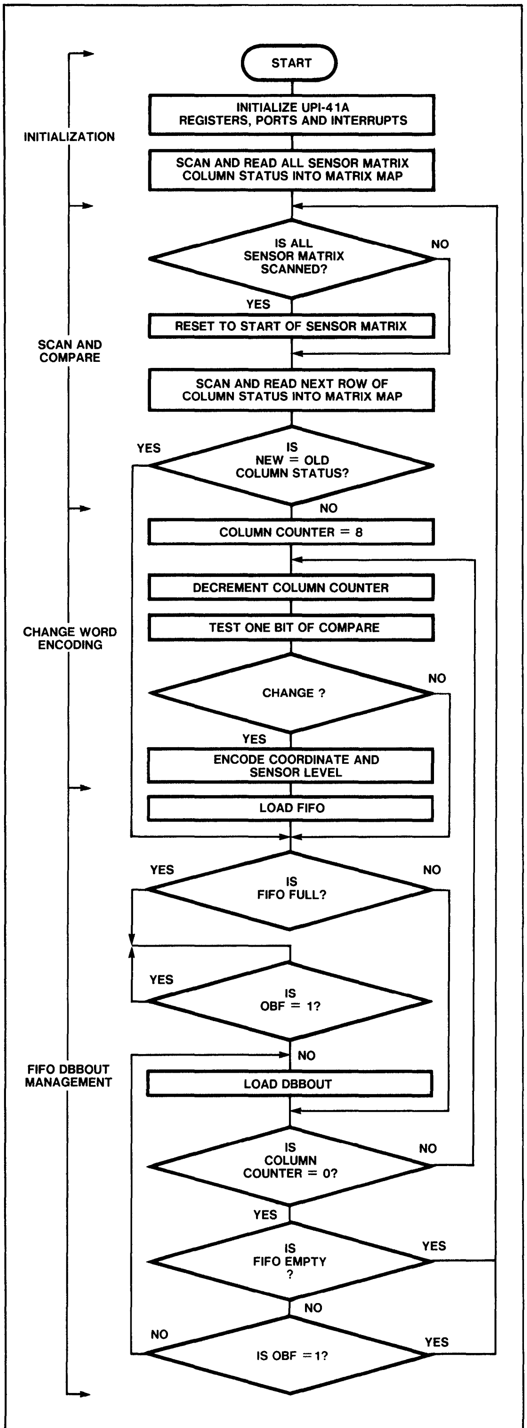


Figure 20. Sensor Matrix Controller Flow Chart

Before delving further into the flow, let's pause to describe the general format of the operation. The UPI scans the matrix one row at a time. If no changes are detected on a particular row, the UPI simply moves to the next row after checking the status of DBBOUT and the FIFO. If a change is detected, the UPI must check each bit (sensor) within the row to determine the actual sensor location. (More than one sensor on the scanned row could have changed.) Rather than test all 8 bits of the row before checking the DBBOUT and FIFO status again, the UPI performs the status check in between each of the bit tests. This ensures the fastest response to the master reading previous Change Words from DBBOUT and the FIFO.

With this general overview in mind, let's go first thru the flow chart assuming we are scanning a row where no changes have occurred. Starting at the Scan-and-Compare section, the UPI first checks if the entire matrix has been scanned. If it has, the various pointers are reset. If not, the address of the next row is placed on PORTs 20 thru 23. This selects the desired row. The state of the row is then read on PORT 1; the column return lines. This present state is compared to the previous state by retrieving the previous state from the matrix map and performing an Exclusive-OR with the present state. Since we are assuming that no change has occurred, the result is zero. No coordinate decoding is needed and the flow branches to the FIFO-DBBOUT Management section.

The FIFO-DBBOUT Management section simply maintains the FIFO and loads DBBOUT whenever Change Words are present in the FIFO and DBBOUT is clear (OBF=0). The section first tests if the FIFO is full. (If we assume our "no-change" row is the first row scanned, the FIFO obviously would not be full.) If it is, the UPI waits until OBF=0, at which point the next Change Word is retrieved from the FIFO and placed in DBBOUT. This "unfills" the FIFO making room for more Change Words. At this point, the Column Counter, R_3, is checked. For rows with no changes, the Column Counter is always zero so the test simply falls through. (We cover the case for changes shortly.) Now the FIFO is tested for being empty. If it is, there is no sense in any further tests so the flow simply goes back up to scan the next row. If the FIFO is not empty, DBBOUT is tested again through OBF. If a Change Word is in DBBOUT waiting for the master to read it, nothing can be done and the flow likewise branches up for the next row. However, if the DBBOUT is free and remembering that the previous test showed that the FIFO was not empty, DBBOUT is loaded with the next Change Word and the last two conditional tests repeat.

Now let's assume the next row contains several changed sensors. Like before, the row is selected, the return lines read, and the sensor status compared to the previous scan. Since changes have occurred, the Exclusive-OR result is now non-zero. Any 1's in the result reflect the positions of the changed sensors. This non-zero result is stored in the Compare Result register, R_7. At this point, the Column Counter is preset to 8. To determine the changed sensors' locations, the Compare Result register is shifted bit-by-bit to the left while decrementing the Column Counter. After each shift, BIT 7 of the result is tested. If it is a one, a changed sensor has been found. The Column Counter then reflected the sensor's matrix column position while the Scan Row Select register holds it row position. These registers are then combined in R_6, the Change Word Store, to form the sensor's matrix coordinate section of the Change Word. The 8th bit of the Change Word Store is coded with the sensor's present state (Figure 18). This byte forms the complete Change Word. It is loaded into the next available FIFO position. If BIT 7 of the Compare Result had been zero, that particular sensor had not changed and the coordinate decoding is not performed.

In between each shift, test, and coordinate encode (if necessary), the FIFO-DBBOUT Management is performed. It is the Column Counter test within this section that routes the flow back up to the Change Word Encoding section if the entire Compare Result (row) has not been shifted and tested.

The FIFO is implemented as a circular buffer with IN and OUT pointers (R_4 and R_5 respectively). The operations of the FIFO is best understood using an example, Figure 21. This series of figures show how the FIFO, DBBOUT, and OBF interact as changes are detected and Change Words are read by the master. The letters correspond to sequential Change Words being loaded into the FIFO. Note that the figures show only a 4×8 FIFO however, the principles are the same in the 40×8 FIFO.

Figure 21A shows the condition where no Change Words have been loaded into the FIFO or DBBOUT. In Figure 21B a change, "A", has been detected, decoded, and loaded into the FIFO at the location equal to the value of the FIFO-IN pointer. The FIFO-OUT pointer is reset to the bottom of the FIFO since it had reached the FIFO top. Now that a Change Word is in the FIFO, OBF is checked to see if DBBOUT is empty. Because OBF=0, DBBOUT is empty and the Change Word is loaded from the FIFO location pointed at by the FIFO-OUT pointer. This is shown in Figure 21C. Loading DBBOUT automatically sets OBF. OBF remains set until the master reads DBBOUT. Figures 21D and 21E show two more Change Words loaded into the FIFO. In Figure 21F the first Change Word is finally read by the master resetting OBF. This allows the next Change Word to be loaded into DBBOUT. Note that each time the FIFO is loaded, the FIFO-IN pointer increments. Each time DBBOUT is read the FIFO-OUT pointer increments unless there are no more Change Words in the FIFO. Both pointers wrap-around to the bottom once they reach the FIFO top. The remaining figures show more Change Words being loaded into the FIFO. When the entire FIFO fills and DBBOUT can not be loaded (OBF=1), scanning stops until the master reads DBBOUT making room for more Change Words.

As was mentioned earlier, two interrupt outputs to the master are available: Change Word Ready (P_{25}, OBF) and FIFO NOT EMPTY (P_{24}). The Change Word Ready interrupt simply reflects OBF and is handled automatically by the UPI since an EN FLAGS instruction is executed during initialization. The FIFO NOT EMPTY interrupt is generated and cleared as appropriate, each pass through the FIFO management code.

No debouncing is provided although it could be added. Rather, the scan time is left as an equate variable so that it could be varied to account for both debounce time and expected sensor change rates. The minimum scan time for this application is 2msec when using a 6MHz clock. Since the matrix controller is coded as a foreground task, scan time simply uses a software delay loop.

The UPI software is included as Appendix B1. Appendix B2 is 8085A test software which builds a Change Word buffer starting at BUFSRT. This software simply polls the STATUS register looking for Change Word Ready to go true. DBBOUT is then read and loaded into the buffer. Now let's move on to an application which combines both the foreground and background concepts.

Combination I/O Device

The final UPI application was designed especially to add additional serial and parallel I/O ports to the iSBC 80/30. This UPI simulates a full-duplex UART (Universal Asynchronous Receiver/Transmitter) combined with an 8-bit parallel I/O port. Features of the UART include: software selectable baud rates (110, 300, 600, or 1200 baud), double buffering for both the transmitter and receiver, and receiver testing for false start bit, framing, and overrun errors. For parallel I/O, one 8-bit port is programmable for either input or output. The output port is statically latched and the input port is sampled.

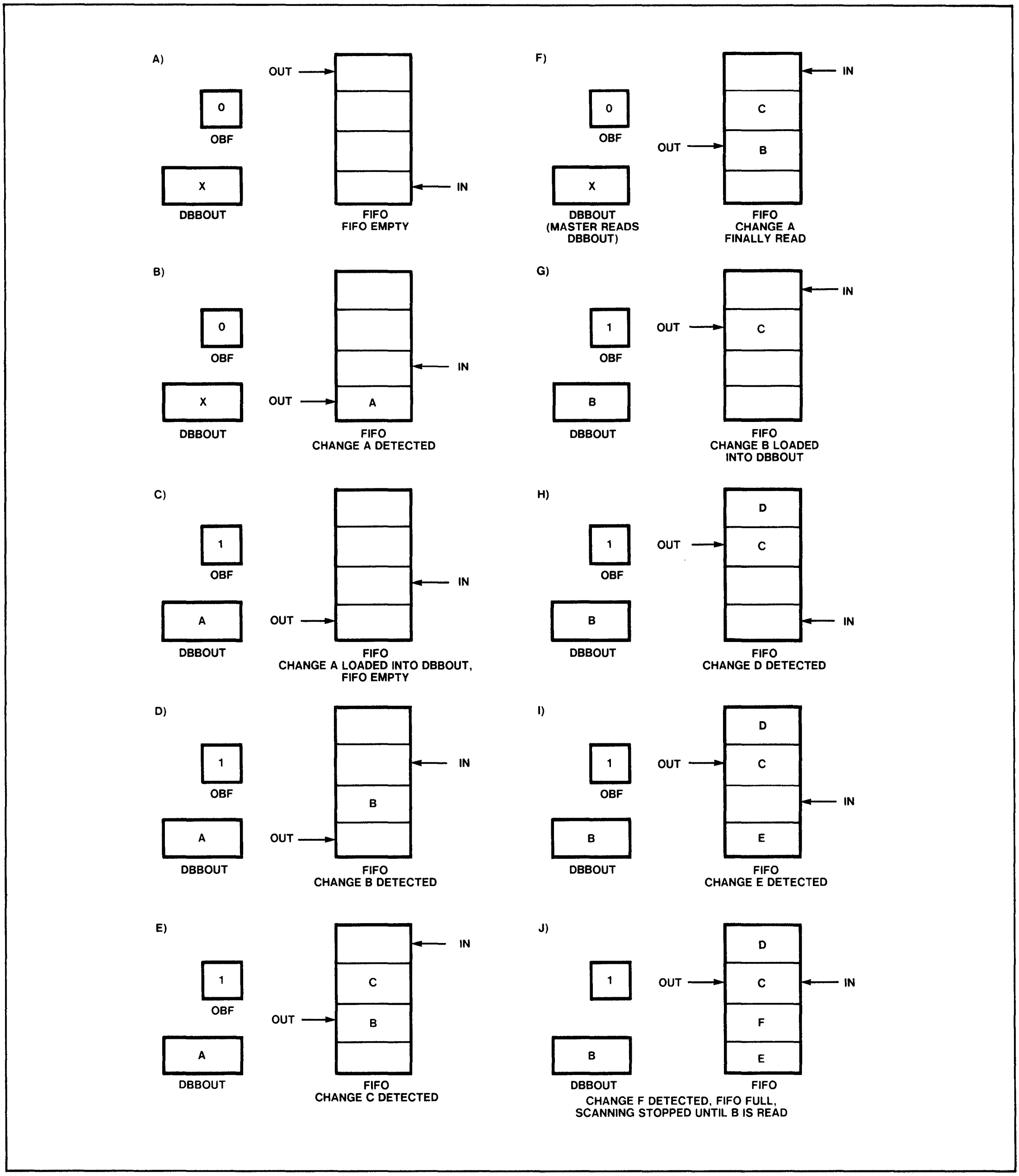


Figure 21A-J. FIFO Operation Example

Figure 22 shows the interface of this combination I/O device to the dedicated UPI socket on the iSBC 80/30. The only external requirement is a 76.8 kHz source which serves as the baud rate standard. The internal baud rates are generated as multiples of this external clock. This clock is obtained from one of the 8253 counters. Otherwise, an RS-232 driver and receiver already available for UPI use in serial I/O applications. Sockets are also provided for termination of the parallel port.

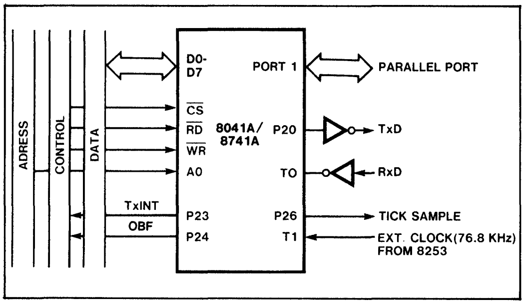


Figure 22. Combination I/O Device

There are three commands for this application. Their format is shown in Figure 23. The CONFIGURE command specifies the serial baud rate and the parallel I/O direction. Normally this command is issued once during system initialization. The I/O command causes a parallel I/O operation to be performed. If the parallel port direction is out, the UPI expects the data byte immediately following an I/O command to be data for the output port. If the port is in the input direction, an I/O command causes the port to be read and the data placed in DBBOUT. The RESET ERROR command resets the serial receiver error bits in the STATUS register.

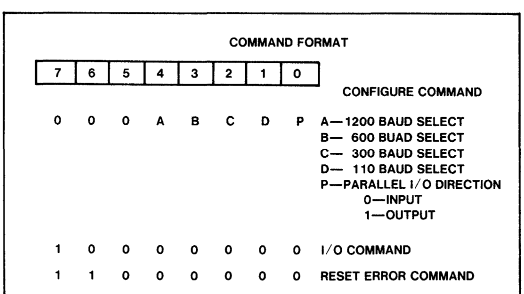
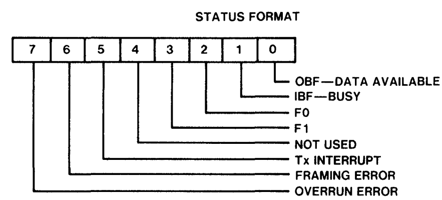


Figure 23. Combination I/O Command Format

The STATUS register format is shown in Figure 24. Looking at each bit, BIT 0 (OBF) is the DATA AVAILABLE flag. It is set whenever the UPI places data into DBBOUT. Since the data may come from either the receiver or the parallel input port, the F_0 and F_1 flags (BITs 2 and 3) code the source. Thus, when the master finds OBF set, it must decode F_0 and F_1 to determine the source.



F0	F1	OPERATION (BF = 1)
0	0	NO OPERATION
0	1	PARALLEL I/O DATA
1	0	SERIAL I/O DATA
1	1	COMMAND ERROR

Figure 24. STATUS Register Format

BIT 1 (IBF) functions as a busy bit. When IBF is set, no writes to DBBIN are allowed. BIT 5 is the TxINT (Transmitter Interrupt) bit. It is asserted whenever the transmitter buffer register is empty. The master uses this bit to determine when the transmitter is ready to accept a data character.

BITS 6 and 7 are receiver error flags. The framing error flag, BIT 6, is set whenever a character is received with an invalid stop bit. BIT 7, overrun error, is set if a character is received before the master has read a previous character. If an overrun occurs, the previous character is overwritten and lost. Once an error occurs, the error flag remains set until reset by a RESET ERROR command. A set error flag does not inhibit receiver operation however.

Figure 25 shows the port pin definition for this application. PORT 1 is the parallel I/O port. The UART uses PORT 2 and the Test inputs. P_{20} is the transmitter data out pin. It is set for a mark and reset for a space. P_{23} is a transmitter interrupt output. This pin has the same timing as the TxINT bit in the STATUS register. It is normally used in interrupt-driven systems to interrupt the master processor when the transmitter is ready to accept a new data character.

The OBF flag is brought out on P_{24} as a master interrupt when data is available in DBBOUT. P_{26} is a diagnostic pin which pulses at four times the selected baud rate. (More about this pin later.) The receiver data input uses the TEST 0 input. One of the PORT 2 pins could have been used, however, the

PORT PIN DEFINITION

PORT	BIT	FUNCTION
1	0-7	PARALLEL I/O
2	0	Tx Data
	1	NOT USED
	2	NOT USED
	3	Tx INTERRUPT
	4	OBF INTERRUPT
	5	NOT USED
	6	NOT USED (TICK SAMPLE)
	7	NOT USED
T0		Rx DATA
T1		EXTERNAL CLOCK (76.8 kHz)

Figure 25. Combination I/O Port Definition

software can test the TEST 0 in one instruction without first reading a port.

The TEST 1 input is the baud rate external source. The UART divides this input to determine the timing needed for the selected baud rate. The input is a non-synchronous 76.8 kHz source.

Internally, when the CONFIGURE command is received and the selected baud rate is determined, the internal timer/counter is loaded with a baud rate constant and started in the event counter mode. Timer/counter interrupts are then enabled. The baud rate constant is selected to provide a counter interrupt at four times the desired baud rate. At each interrupt, both the transmitter and receiver are handled. Between interrupts, any new commands and data are recognized and executed.

As a prelude to discussing the flow charts, Figure 26 shows the register definition. Register Bank 0 serves the UART receiver and parallel I/O while Register Bank 1 handles the UART transmitter and commands. Looking at RB0 first, R_3 is the receiver status register, RxSTS. Reflected in the bits of this register is the current receiver status in sequential order. Figure 27 shows this bit definition. BIT 0 is the Rx flag. It is set whenever a possible start bit is received. BIT 1 signifies that the start bit is good and character construction should begin with the next received bit. BIT 1 is the Good Start flag. BIT 2 is the Byte Finished flag. When all data bits of a character are received, this flag is set. When all the bits, data and stop bits are received, the assembled character is loaded into the holding register (R_4 in Figure 27) BIT 3, the Data Ready flag, is set. The foreground routine which looks for commands and data continuously, looks at this bit to determine when the receiver has received a character. BITS 4 and 5 signify any error conditions for a particular character.

Address	Contents	Register	Bank
63–32	USER RAM (NOT USED)		
31	AC TEMP. STORE	R7	REGISTER BANK 1
30	COMMAND STORE	R6	
29	Tx STATUS — TxSTS	R5	
28	Tx BUFFER	R4	
27	Tx SERIALIZER	R3	
26	Tx TICK COUNTER	R2	
25	BAUD RATE CONSTANT	R1	
24	NOT USED	R0	
23–8	STACK (ONE LEVEL USED)		
7	STATUS STORE	R7	REGISTER BANK 0
6	Rx DESERIALIZER	R6	
5	Rx TICK COUNTER	R5	
4	Rx HOLDING	R4	
3	Rx STATUS—RxSTS	R3	
2	NOT USED	R2	
1	NOT USED	R1	
0	NOT USED	R0	

Figure 26. Combination I/O Register Map

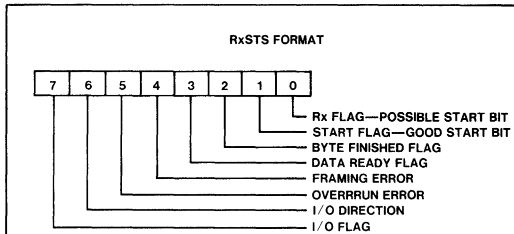


Figure 27. RxSTS Register

The parallel I/O port software uses BITS 6 and 7. BIT 6 codes the I/O direction specified by the last CONFIGURE command. BIT 7 is set whenever an I/O command is received. The foreground routine tests this bit to determine when an I/O operation has been requested by the master.

As was mentioned, R_4 is the receiver holding register. Assembled characters are held in this register until the foreground routine finds DBBOUT free, at which time the data is transferred from R_4 to DBBOUT. R_5 is the receiver tick counter. Recall that counter interrupts occur at four times the baud rate. Therefore, once a start bit is found, the receiver only needs to look at the data every four interrupts or tick counts. R_5 holds the current tick count.

R_6 is the receiver de-serializing register. Data characters are assembled in this register. R_6 is preset to 80H when a good start bit is received. As each bit is sampled every four timer ticks, they are rotated into the leftmost bit of R_6. The software knows the character assembly is complete when the original preset bit rotates into the carry.

An image of the upper 4 bits of the STATUS register is stored in R_7. These bits are the TxINT, Framing and Overrun bits. This image is needed since the UPI may load the upper 4 STATUS register bits from its accumulator; however, it cannot read STATUS directly.

In Register Bank 1 (Figure 26), R1 holds the baud rate constant which is found from decoding the baud rate select bits of the CONFIGURE command. The counter is reloaded with this constant every timer tick. Like the receiver, the transmitter only needs to update the transmitter output every four ticks. R_2 holds the transmitter tick count. The value of R_2 determines which portion of the data is being transmitted; start bit, data bits, or stop bit. The transmit serializer is R_3. R_3 holds the data character as each character bit is transmitted.

R_4 is the transmitter holding register. It provides the double buffering for the transmitter. While transmitting one character, it is possible to load the next character into R_4 via DBBIN. The TxINT bit in STATUS and pin on PORT 2 reflect the "fullness" of R_4. If the holding register is empty, the interrupt bit and pin are set. They are reset when the master writes a new data byte for the transmitter into DBBIN. The transmitter status register (TxSTS) is R_5. Like RxSTS,TxSTS contains flag bits which indicate the current state of the transmitter. This flag bit format is shown in Figure 28.

TxSTS BIT 0 is the Tx flag. It is set whenever the transmitter is transmitting a character. It is set from the beginning of the start bit until the end of the stop bit. BIT 1 is the Tx request flag. This bit is set by the foreground routine when it transfers a new character from DBBIN to the Tx holding register, R_4. The transmitter software uses this flag to tell if new data is available. It is reset when the transmitter transfers the character from the holding register to the serializer.

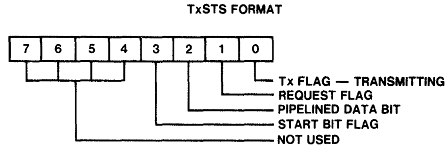


Figure 28. TxSTS Register

BIT 2 is the pipelined Tx data bit. The transmitter uses a pipelining technique which sets up the next output level in BIT 2 after processing the current timer tick. The output level is always changed at the same point after a timer tick interrupt. This technique ensures that no bit timing distortion results from different length processing paths through the receiver and transmitter routines.

BIT 3 of TxSTS is the Start Bit flag. It is set by the transmitter when the start bit space is set up in the pipelined data bit. This allows the transmitter to differentiate between the start bit and the data bits on following timer ticks.

The flow charts for this application are shown in Figures 29A–F. At reset, the INIT routine is executed which initializes the registers and port pins. After initialization, IBF and OBF are tested in MNLOOP. These flags are tested continually in this loop. If IBF is set, F_1 is tested for command or data and execution is transferred to the appropriate routine (CMD or DATA). If IBF=0, OBF is checked. If OBF=0 (DBBOUT is free), the Rx data ready and I/O flags in RxSTS are tested. If Rx data ready is set, the received data is retrieved from the Rx holding register and transferred to DBBOUT. Any error flags associated with that data are also transferred to STATUS. If the I/O flag is set and the I/O direction is input, PORT 1 is read and the data transferred to DBBOUT. In either case, F_0 and F_1 are set to indicate the data source.

If IBF is set by a command write to DBBIN, CMD reads the command and decodes the desired operation. If an I/O operation is specified, the I/O flag is set to indicate to the MNLOOP and DATA routines that an I/O operation is to be performed. If the command is a CONFIGURE command, the constant for the selected baud rate is loaded into both Baud Rate Constant register and the timer/counter. The timer/counter is started in the event counter mode and timer/counter interrupts are enabled. In addition, the I/O port is initialized to all 1's if the I/O direction bit specifies an input port. If the command is a RESET ERROR command, the two error flags in STATUS are cleared.

If the IBF flag is set by a data write, the DATA routine reads DBBIN and places the data in the appropriate place. If the I/O flag is set, the data is for the output port so the port is loaded. If the I/O flag is reset, the data is for the UART transmitter. Data for the transmitter resets the TxINT bit and pin plus sets the Tx request flag in TxSTS. The data is transferred to the Tx holding register, R_4.

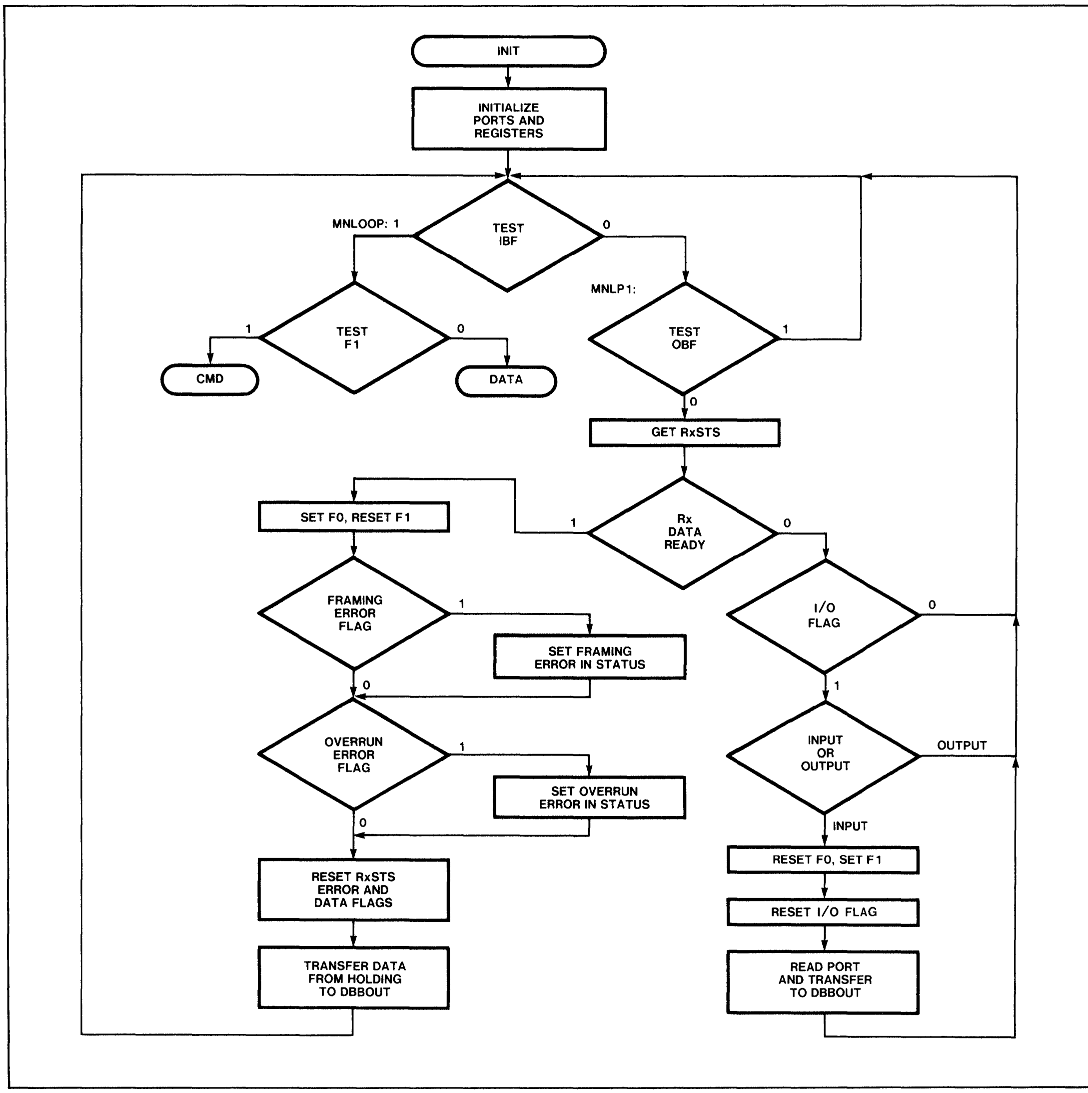


Figure 29A. INIT Flow Chart

Once a CONFIGURE command is received and the counter started, timer/counter interrupts start occurring at four times the selected baud rate. These interrupts cause a vector to the TIMINT routine, Figure 29D. A 76.8 kHz counter input provides a 13.02 μs counter resolution. Since it requires several UPI instruction cycles to reload the counter, the counter is set to two counts less than the desired baud rate and the counter is reloaded in TIMINT synchronous with the second low-going transition after the interrupt. Once the counter is reloaded, an output port (P_{26}) is toggled to give an external indication of internal counter interval. This is a helpful diagnostic feature. After the tick sample output, the pipelined transmitter data in TxSTS is output to the TxD pin. Although this occurs every timer tick, the pipelined data is changed only every fourth tick.

The receiver is now handled, Figure 29E. The Rx flag in RxSTS is examined to see if the receiver is currently in the process of receiving a character. If it is not, the RxD input is tested for a space condition which might indicate a possible start bit. If the input is a mark, no start bit is possible and execution

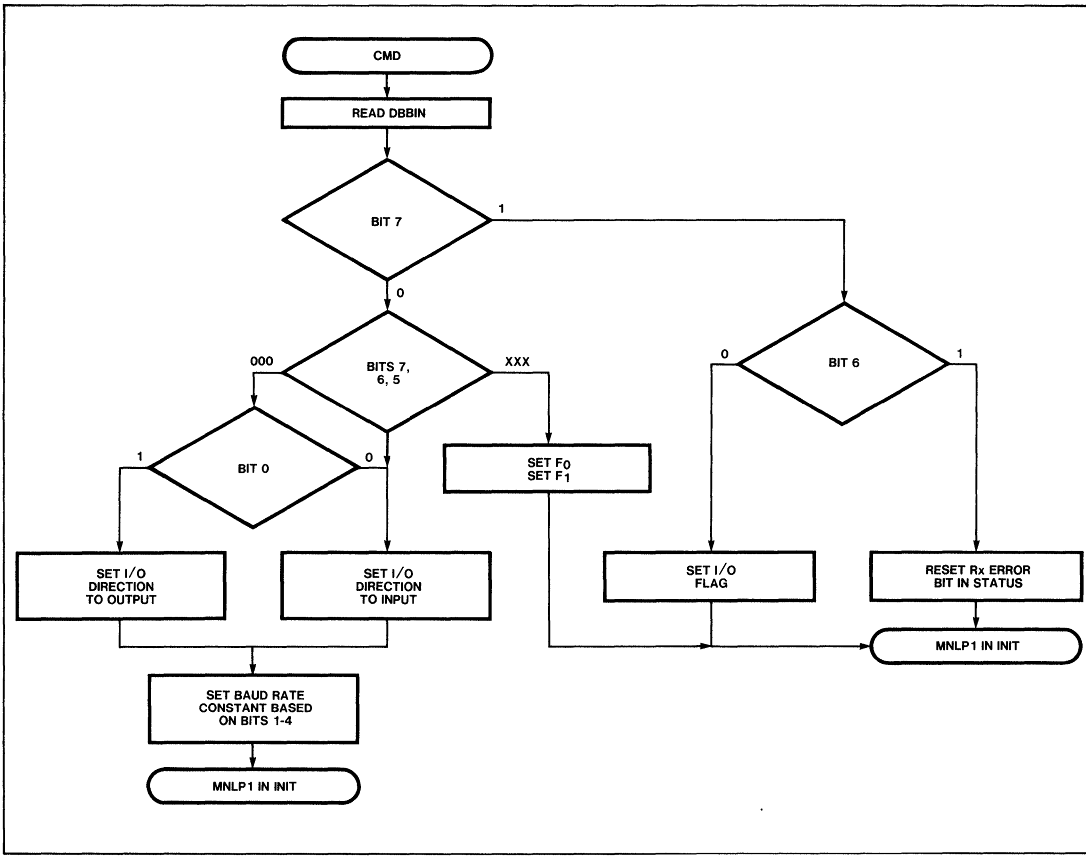


Figure 29B. CMD Flow Chart

branches to the transmitter flow, XMIT. If the input is a space, the Rx flag is set before proceeding with XMIT.

If the Rx flag is found set when entering RCV, the receiver is in the process of receiving a character. If so, the start bit flag is then tested to determine if a good start bit was received. The Rx tick counter is initialized to 4 and the Rx deserializer is set to 80H. A mark indicates a bad start bit; the Rx flag is reset to abort the reception.

If the start bit flag is set, the program is somewhere in the middle of the received character. Since the data should be sampled every fourth timer tick, the tick counter is decremented and tested for zero. If non-zero no sample is needed and execution continues with XMIT. If zero, the tick counter is reset to four. Now the byte finished flag is tested to determine if the data sample is a data or stop bit. If reset, the sample is a data bit. The sample is done and the new bit rotated into the Rx deserializer. If this rotate

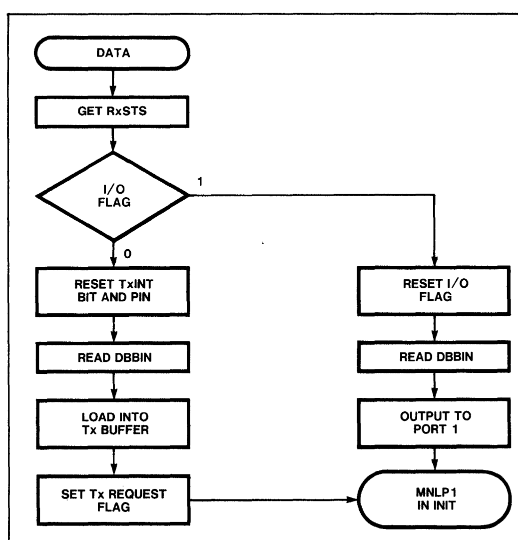


Figure 29C. Data Flow Chart

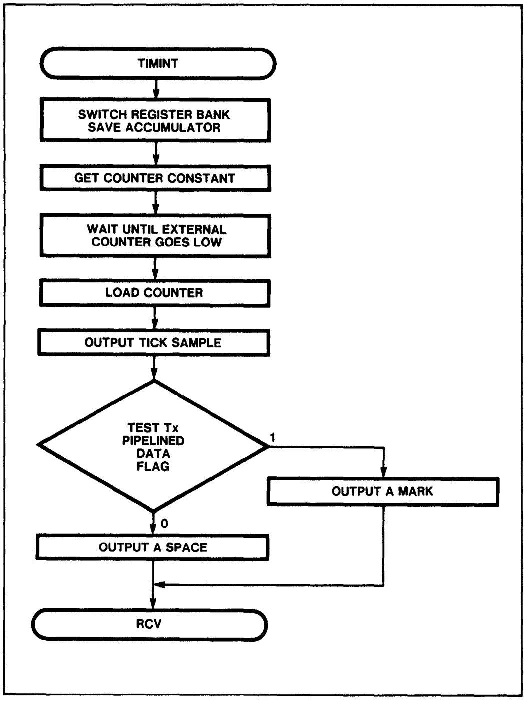


Figure 29D. TIMINT Flow Chart

sets the carry, that data bit was the last so the byte finished flag is set. If the carry is reset, the data bit is not the last so execution simply continues with XMIT.

Had the byte finished flag been set, this sample is for the stop bit. The RxD input is tested and if a space, the framing error flag is set. Otherwise, it is reset. Next, the Rx data ready flag is tested. If it is set, the master has not read the previous character so the overrun error flag is set. Then the Rx data ready flag is set and the received data character is transferred into the Rx holding register. The Rx, start bit, and byte finished flags are reset to get ready for the next character.

Execution of the transmitter routine, XMIT, follows the receiver, Figure 29F. The transmitter starts by checking the start bit flag in TxSTS. Recall that the actual transmit data is output at the beginning of the timer routine. The start bit flag indicates whether the current timer tick interrupt started the start bit. If it is set, the pipelined data output earlier in the routine was the start of the start bit so the flag is reset and the Tx tick counter is initialized. Nothing else is done this timer tick so the routine returns to the foreground.

If the start bit flag is reset, the Tx tick counter is incremented and tested. The test is performed modulo 4. If the counter mod 4 is not zero, it has not been four ticks since the transmitter was handled last so the routine simply returns. If the counter mod 4 is zero, it is time to handle the transmitter and the Tx flag is tested.

The Tx flag indicates whether the transmitter is active. If the transmitter is inactive, no character is currently being transmitted so the Tx request flag is tested to see if a new character is waiting in the Tx buffer. If no character is waiting (Tx request flag=0), the Tx interrupt pin and bit are set before returning to the foreground. If there is a character waiting, it is retrieved from the buffer and placed in the Tx serializer. The Tx request flag is reset while the Tx and start bit flags are set. A space is placed in the Tx pipelined data bit so a start bit will be output on the next tick. Since the Tx buffer is now empty, the Tx interrupt bit and pin are set to indicate the availability of the buffer to the master. The routine then returns to the foreground.

If the tick counter mod 4 is zero and the Tx flag indicates the transmitter is in the middle of a character, the tick counter is checked to see what transmitter operation is needed. If the counter is 28H (40D), all data bits plus the stop bits are complete. The character is therefore done and the Tx flag is reset. If the counter is 24H (36D), the data bits are complete and the next output should be a mark for the stop bit so a mark is loaded into the Tx pipelined data bit.

If neither of the above conditions are met for the counter, the transmitter is some place in the data field, so the next data bit is rotated out of the Tx serializer into the pipelined data bit. The next tick outputs this bit.

At this point the program execution is returned to the foreground.

That completes the discussion of the combination I/O device flow charts. The UPI software listing is shown in Appendix C1. Appendix C2 is example 8085A driver software.

Several observations concerning the drivers are appropriate. Notice that since the receiver and input port of the UPI use the OBF flag and interrupt output, the interrupt and flag are cleared when the master reads DBBOUT. This is not true for the transmitter. There is always some time after a master write of new transmitter data before the transmitter bit and pin are cleared. Thus in an interrupt-driven system, edge-sensitive interrupts should be

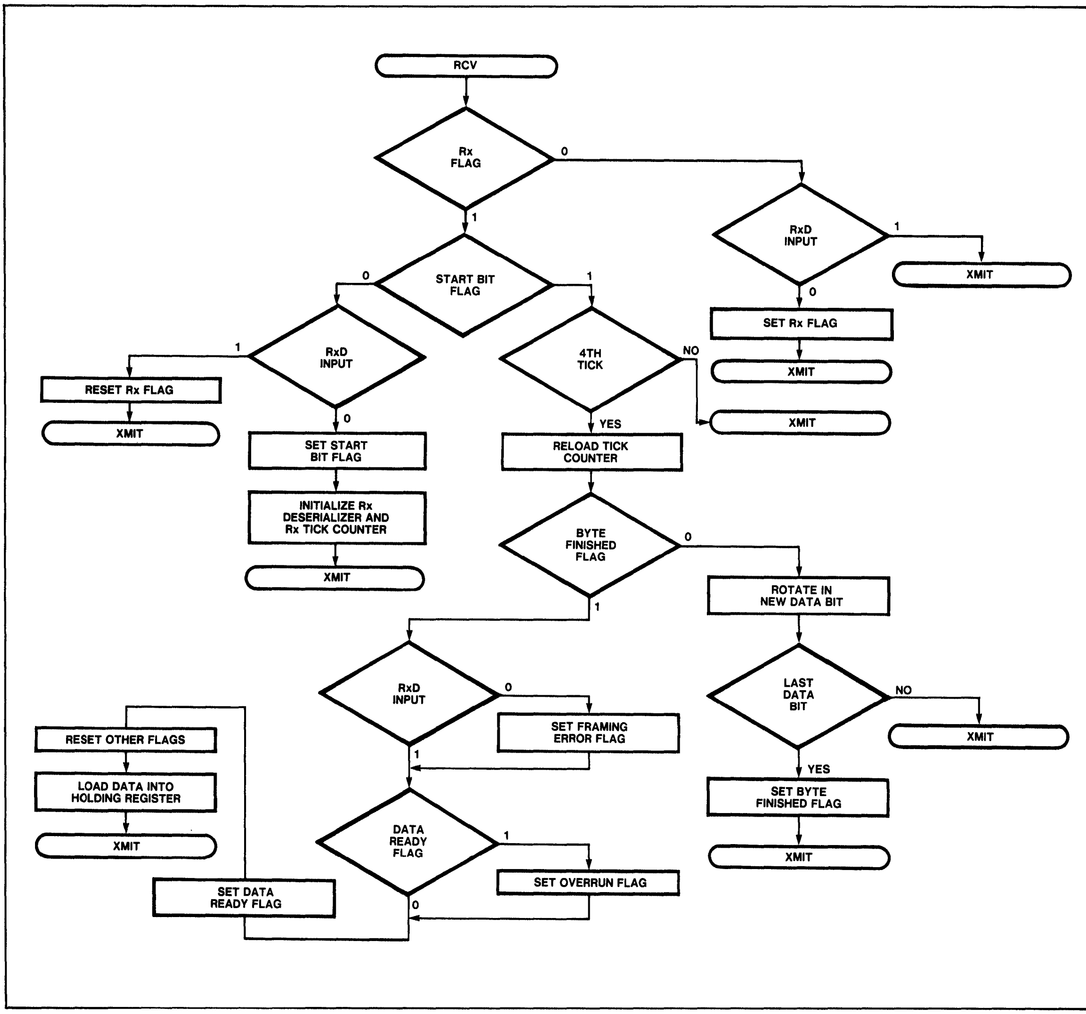


Figure 29E. RCV Flow Chart

used. For polled-systems, the software must wait after writing new data for IBF=0 before re-examining the Tx interrupt flag in STATUS.

Notice that this application uses none of the user data memory above Register Bank 1 and only 361 bytes of program memory. This leaves the door open for many improvements. Improvements that come to mind are increased buffering of the transmit or received data, modem control pins, and parallel port handshaking inputs.

This completes our discussion of specific UPI applications. Before concluding, let's look briefly at two debug techniques used during the development of these applications that you might find useful in your own designs.

DEBUG TECHNIQUES

Since the UPI is essentially a single-chip microcomputer, the classical data, address, and control buses are not available to the outside world during normal operation. This fact normally makes debugging a UPI design difficult; however, certain "tricks" can be included in the UPI software to ease this task.

If a UPI is handling multiple tasks, it is usually easier to code and debug each task individually. This is fairly standard procedure. Since each task usually utilizes only a subset of the total number of I/O pins,

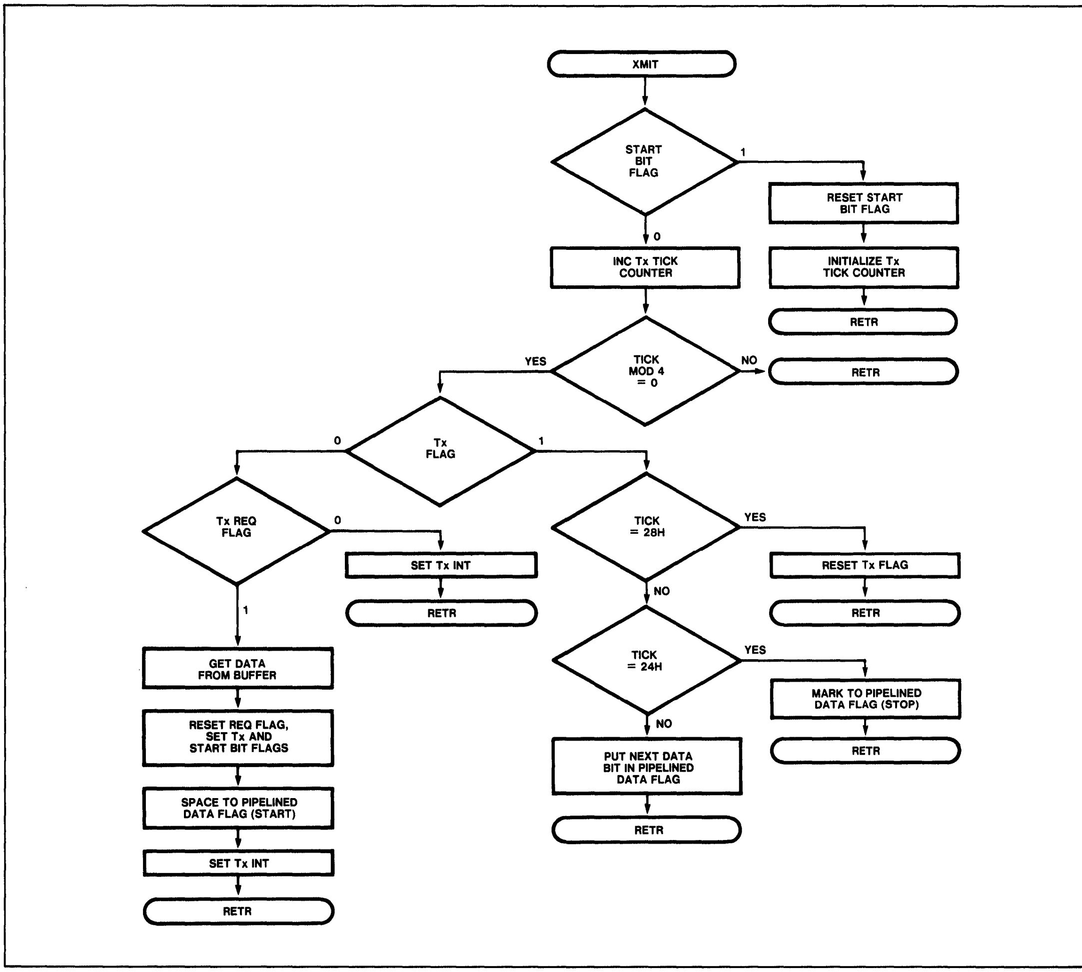


Figure 29F. XMIT Flow Chart

coding only one task leaves some I/O pins free. Port output instructions can then be added in the task code being debugged which toggle these unused pins to determine which section of task code is being executed at any particular time. The task can also be made to "wait" at various points by using an extra pin as an input and adding code to loop until a particular input condition is met.

One example of using an extra pin as an output is included in the combination serial/parallel device code. During initial development the receiver was not receiving characters correctly. Since this could be caused by incorrect sampling, three lines of code were added to toggle BIT 6 of PORT 2 at each tick of the sample clock. This code is at lines 184 and 185 of the listing. Thus by looking at the location of the tick sample pulse with respect to the received bit, the UPI sampling interval can be observed. The tick sample time was incorrect and the code was modified accordingly. Similar techniques could be applied at other locations in the program.

The EPROM version of the UPI (8741A) also contains another feature to aid in debug: the capability to single step thru a program. The user may step thru the program instruction-by-instruction. The address of the next instruction to be fetched is available on PORT 1 and the lower 2 bits of PORT 2. Figure 30 shows the timing used in the discussion below. When the single step input, $\overline{SS}$, is brought low, the internal processor responds by stopping during the fetch portion of the next instruction. This action is acknowledged by the processor raising the SYNC

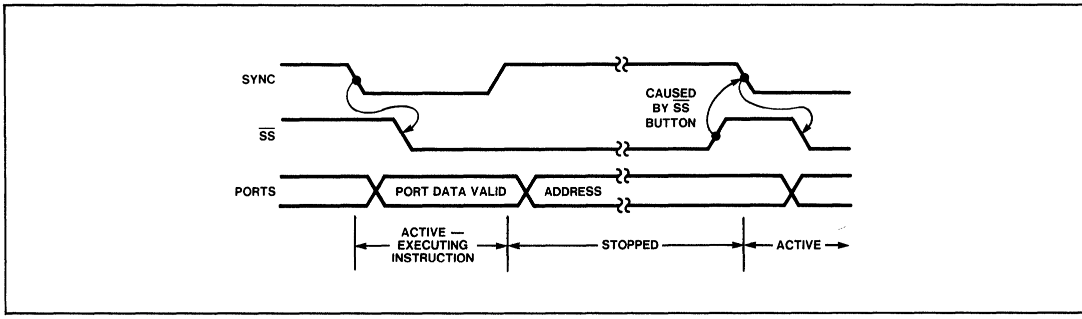


Figure 30. Single Step Timing

output. The address of the instruction to be fetched is then placed on the port pins. This state may be held indefinitely. To step to the next instruction, $\overline{SS}$ is raised high, which causes SYNC to go low, which is then used to return $\overline{SS}$ low. This allows the processor to advance to the next instruction. If $\overline{SS}$ is left high, the processor continues to execute at normal speed until $\overline{SS}$ goes low.

To preserve port functionality, port data is valid while SYNC is low. Figure 31 shows the external circuitry required to implement single step while preserving port functionality. S_1 is the RUN/STOP switch. When in the RUN position, the 7474 is held preset so $\overline{SS}$ is high and the UPI executes normally. When switched to STOP, the preset is removed and the next low-going transition of SYNC causes the 7474 to clear, lowering $\overline{SS}$. While sync is low, the port data is valid and the current instruction is executing. Low SYNC is also used to enable the tri-state buffers when the ports are used as inputs. When execution is complete, SYNC goes high. This transition latches the valid port data in the 74LS374s. SYNC going high also signifies that the address of the next instruction will appear on the port pins. This state can be held indefinitely with the address data displayed on the LEDs.

When the S_2 is depressed, the 7474 is set which causes $\overline{SS}$ to go high. This allows the processor to fetch and execute the instruction whose address was displayed. SYNC going low during execution, clears

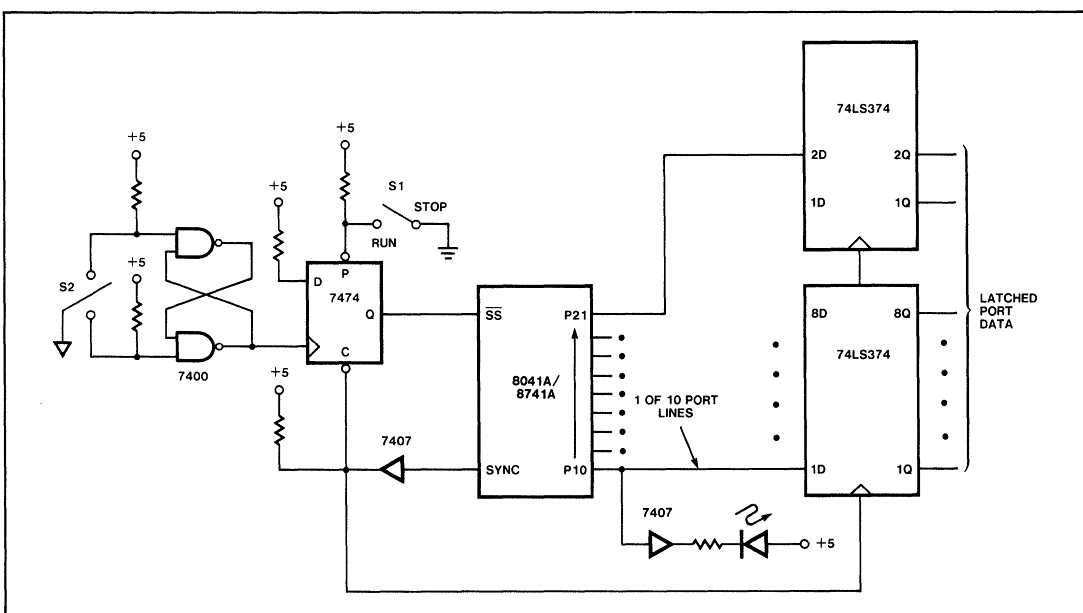


Figure 31. Single Step External Circuitry

the 7474 lowering $\overline{SS}$. Thus the processor again stops when execution is complete and the next fetch is started.

All UPI functions continue to operate while single stepping (the processor is actually executing NOPs internally while stopped). Both IBF and timer/counter interrupts can be serviced. The only change is that the interval timer is prescaled on single stepped instructions and, of course, will not indicate the correct intervals in real time. The total number of instruction which would have been executed during a given interval is the same however.

The single step circuitry can be used to step through a complete program; however, this might be a time-consuming job if the program is long or if only a portion is to be examined. The circuitry could easily be modified to incorporate the output toggling technique to determine when to run and stop. If you would like to step thru a particular section of code, an extra port pin could replace switch S_1. Extra instructions would then be added to lower the port when entering the code section and raise the port when exiting the section. The program would then stop when that section of code is reached allowing it to be stepped through. At the end of the section, the program would execute at normal speed.

CONCLUSION

Well, that's it. Machine readable (floppy disk or paper tape) source listings of UPI software for these applications are available in Insite, the Intel library of user-donated programs. Also available in Insite are the source listings for some of Intel's pre-programmed UPI products.

For information about Insite, write to:
Insite
Intel Corp.
3065 Bowers Ave.
Santa Clara, Ca 95051

```
:F1:ASM48 :F3:LED PRINT(:LP:) NOOBJECT


ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0              PAGE    1

  LOC  OBJ         LINE        SOURCE STATEMENT

$MOD41A
;       ***********************************************
;       *    UPI-41A 8-DIGIT LED DISPLAY CONTROLLER    *
;       ***********************************************
;
;
;
;THIS PROGRAM USES THE UPI-41A AS A LED DISPLAY CONTROLLER
;WHICH SCANS AND REFRESHES EIGHT SEVEN-SEGMENT LED DISPLAYS.
;THE CHARACTERS ARE DEFINED BY INPUT FROM A MASTER CPU IN THE
;FORM OF ONE EIGHT BIT WORD PER DIGIT-CHARACTER SELECTION.
;
;
;
;*******************************************************************
;
;REGISTER DEFINITIONS:
;       REGISTER               RB1                        RBO
;       --------               ---                        ---
;       RO              DISPLAY MAP POINTER               NOT USED
;       R1              NOT USED                          NOT USED
;       R2              DATA WORD AND CHARACTER STORAGE   NOT USED
;       R3              DIGIT COUNTER                     NOT USED
;       R4              NOT USED                          NOT USED
;       R5              NOT USED                          NOT USED
;       R6              NOT USED                          NOT USED
;       R7              ACCUMULATOR STORAGE               NOT USED
;********************************************************************
;
;PORT PIN DEFINITIONS:
;       PIN             PORT 1 FUNCTION         PORT 2 FUNCTION
;       ---             ---------------         ---------------
;       PO-7            SEGMENT DRIVER CONTROL  DIGIT DRIVER CONTROL
;
$EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0              PAGE     2

  LOC  OBJ          LINE          SOURCE STATEMENT

                      ;************************************************************************
                      ;DISPLAY DATA WORD BIT DEFINITION:
                      ;       BIT               FUNCTION
                      ;       ---               --------
                      ;       0-4               CHARACTER SELECT
                      ;       5-7               DIGIT SELECT
                      ;
                      ;CHARACTER SELECT:
                      ;                 D4  D3  D2  D1  DO  CHARACTER
                      ;                 0   0   0   0   0       0
                      ;                 0   0   0   0   1       1
                      ;                 0   0   0   1   0       2
                      ;                 0   0   0   1   1       3
                      ;                 0   0   1   0   0       4
                      ;                 0   0   1   0   1       5
                      ;                 0   0   1   1   0       6
                      ;                 0   0   1   1   1       7
                      ;                 0   1   0   0   0       8
                      ;                 0   1   0   0   1       9
                      ;                 0   1   0   1   0       A
                      ;                 0   1   0   1   1       B
                      ;                 0   1   1   0   0       C
                      ;                 0   1   1   0   1       D
                      ;                 0   1   1   1   0       E
                      ;                 0   1   1   1   1       F
                      ;                 1   0   0   0   0       .
                      ;                 1   0   0   0   1       G
                      ;                 1   0   0   1   0       H
                      ;                 1   0   0   1   1       I
                      ;                 1   0   1   0   0       J
                      ;                 1   0   1   0   1       L
                      ;                 1   0   1   1   0       N
                      ;                 1   0   1   1   1       O
                      ;                 1   1   0   0   0       P
                      ;                 1   1   0   0   1       R
                      ;                 1   1   0   1   0       T
                      ;                 1   1   0   1   1       U
                      ;                 1   1   1   0   0       Y
                      ;                 1   1   1   0   1       -
                      ;                 1   1   1   1   0       /
                      ;                 1   1   1   1   1       "BLANK"
                      ;
                      ;DIGIT SELECT:
                      ;                 D7  D6  D5   DIGIT NUMBER
                      ;                 0   0   0         1
                      ;                 0   0   1         2
                      ;                 0   1   0         3
                      ;                 0   1   1         4
                      ;                 1   0   0         5
                      ;                 1   0   1         6
                      ;                 1   1   0         7
                      ;                 1   1   1         8
                      ;************************************************************************
                      $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0              PAGE    3

 LOC  OBJ          LINE         SOURCE STATEMENT

                        ;***********************************************************************
                        ;                       EQUATES
                        ;THE FOLLOWING CODE DESIGNATES "TIME" AS A VARIABLE. THIS
                        ;ADJUSTS THE AMOUNT OF CYCLES THE TIMER COUNTS BEFORE
                        ;A TIMER INTERRUPT OCCURS AND REFRESHES THE DISPLAY. APPROXIMATELY
                        ;50 TIMES PER SECOND.
 FFF1                   TIME    EQU     -0FH    ;TIMER VALUE 2.5MSEC
                        ;***********************************************************************
                        ;                       INTERRUPT BRANCHING
                        ;THIS PORTION OF MEMORY IS DEDICATED FOR USE OF RESET AND
                        ;INTERRUPT BRANCHING. WHEN THE INTERRUPTS ARE ENABLED THE
                        ;CODE AT THE FOLLOWING DESIGNATED SPOTS ARE EXECUTED WHEN A
                        ;RESET OR A INTERRUPT OCCURS.
 0000                           ORG     0               ;
 0000 0409                      JMP     START           ;RESET
 0002 00                        NOP                     ;
 0003 0436                      JMP     INPUT           ;IBF INTERRUPT
 0005 00                        NOP                     ;
 0006 00                        NOP                     ;
 0007 041D                      JMP     DISPLA          ;TIMER INTERRUPT
                        ;***********************************************************************
                        ;                       INITIALIZATION
                        ;THE FOLLOWING CODE SETS UP THE UPI-41 AND DISPLAY HARDWARE
                        ;INTO OPERATIONAL FORMAT. THE DISPLAY IS TURNED OFF, THE DISPLAY
                        ;MAP IS FILLED WITH "BLANK" CHARACTERS, THE TIMER SET AND THE
                        ;INTERRUPTS ARE ENABLED.
                        ;
 0009 D5                START:  SEL     RB1             ;
 000A 8A08                      ORL     P2,#08H         ;TURN DIGIT DRIVERS OFF
 000C B838                      MOV     R0,#38H         ;DISPLAY MAP POINTER,BOTTOM OF DISPLAY MAP
 000E 23FF              BLKMAP: MOV     A,#0FFH         ;FF="BLANK"
 0010 A0                        MOV     @R0,A           ;BLANK TO DISPLAY MAP
 0011 18                        INC     R0              ;INCREMENT DISPLAY MAP POINTER
 0012 F8                        MOV     A,R0            ;DISPLAY MAP POINTER TO ACCUMULATOR
 0013 B20E                      JB5     BLKMAP          ;BLANK DISPLAY MAP TILL FILLED
 0015 BB00                      MOV     R3,#00H         ;SET DIGIT COUNTER TO 0
 0017 23F1                      MOV     A,#TIME         ;TIMER VALUE
 0019 62                        MOV     T,A             ;LOAD TIMER
 001A 55                        STRT    T               ;START TIMER
 001B 25                        EN      TCNTI           ;ENABLE TIMER INTERRUPT
 001C 05                        EN      I               ;ENABLE IBF INTERRUPT
                        ;***********************************************************************
                        ;                       USER PROGRAM
                        ;A USERS PROGRAM WOULD INITIALIZE AT THIS POINT.  THE FOLLOWING
                        ;CODE IS UND CONCLUDED WITH
                        ;SYNC CHARACTERS (0AAH).  A CHECKSUM BYTE IMMEDIATELY PRECEEDS THE
                        ;FINAL SYNC.  WHEN READING, THE CONTROLLE****************************************
                        $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0          PAGE    4

LOC  OBJ        LINE        SOURCE STATEMENT

                    ;**************************************************************************
                    ;                           DISPLAY ROUTINE
                    ; THIS PORTION OF THIS PROGRAM IS AN INTERRUPT ROUTINE WHICH IS
                    ;ACTED UPON WHEN THE TIMER COUNT IS COMPLETED. THE ROUTINE UPDATES
                    ;ONE DISPLAY DIGIT FROM THE DISPLAY MAP PER INTERRUPT SEQUENTIALLY,
                    ;THUS EIGHT TIMER INTERRUPTS WILL HAVE REFRESHED THE ENTIRE DISPLAY.
                    ;REGISTER BANK 1 IS SELECTED AND THE ACCUMULATOR IS SAVED UPON
                    ;ENTERING THE ROUTINE. ONCE THE DISPLAY HAS BEEN REFRESHED THE TIMER
                    ;IS RESET AND THE ACCUMULATOR AND PRE-INTERRUPT REGISTER BANK IS RESTORED.
                    ;
001D D5             DISPLA: SEL     RB1             ;REGISTER BANK 1
001E AF                     MOV     R7,A            ;SAVE ACCUMULATOR
001F 8A08                   ORL     P2,#08H         ;TURN DIGIT DRIVERS OFF
0021 FB                     MOV     A,R3            ;DIGIT COUNTER TO ACCUMULATOR
0022 4338                   ORL     A,#38H          ;"OR" TO GET DISPLAY MAP ADDRESS
0024 A8                     MOV     R0,A            ;DISPLAY MAP POINTER
0025 F0                     MOV     A,@R0           ;GET CHARACTER FROM DISPLAY MAP
0026 39                     OUTL    P1,A            ;OUTPUT CHARACTER TO SEGMENT DRIVERS
0027 FB                     MOV     A,R3            ;DIGIT COUNTER VALUE TO ACCUMULATOR
0028 3A                     OUTL    P2,A            ;OUTPUT TO DIGIT DRIVERS
0029 1B                     INC     R3              ;INCREMENT DIGIT COUNTER
002A D307                   XRL     A,#07H          ;CHECK IF AT LAST DIGIT
002C 9630                   JNZ     SETIME          ;RESET TIMER IN NOT LAST DIGIT
002E BB00                   MOV     R3,#00H         ;RESET DIGIT COUNTER
0030 23F1           SETIME: MOV     A,#TIME         ;TIMER VALUE
0032 62                     MOV     T,A             ;LOAD TIMER
0033 55                     STRT    T               ;START TIMER
0034 FF                     MOV     A,R7            ;RESTORE ACCUMULATOR
0035 93                     RETR                    ;RETURN
                    ;**************************************************************************
                    $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0              PAGE    5

 LOC  OBJ         LINE        SOURCE STATEMENT

                       ;
                       ;*************************************************************************
                       ;                 INPUT CHARACTER AND DIGIT ROUTINE
                       ; THIS PORTION OF THE PROGRAM IS AN INTERRUPT ROUTINE WHICH
                       ;IS ACTED UPON WHEN THE IBF BIT IS SET.  THE ROUTINE GETS THE
                       ;DISPLAY DATA WORD FROM THE DBB AND DEFINES BOTH THE DIGIT AND
                       ;THE CHARACTER TO BE DISPLAYED.  THIS IS DONE BY MEANS OF A
                       ;CHARACTER LOOP-UP TABLE AND A DISPLAY MAP FOR DIGIT AND CHARACTER
                       ;LOCATION.  SPECIAL CONSIDERATION IS TAKEN FOR A DECIMAL POINT WHICH IS
                       ;SIMPLY ADDED TO THE EXISTING CHARACTER IN THE DISPLAY MAP.  REGISTER
                       ;BANK 1 IS SELECTED AND THE ACCUMULATOR IS SAVED UPON ENTERING
                       ;THE ROUTINE.  ONCE THE DATA WORD HAS BEEN FULLY DEFINED THE ACCUMULATOR
                       ;AND THE PRE-INTERRUPT REGISTER BANK IS RESTORED.
                       ;
0036 D5                INPUT:  SEL    RB1            ;REGISTER BANK 1
0037 AF                        MOV    R7,A           ;SAVE ACCUMULATOR
0038 22                        IN     A,DBB          ;GET DATA
0039 AA                        MOV    R2,A           ;SAVE DATA WORD
003A 47                        SWAP   A              ;DEFINE DIGIT LOCATION
003B 77                        RR     A              ;
003C 5307                      ANL    A,#07H         ;
003E 4338                      ORL    A,#38H         ;
0040 A8                        MOV    R0,A           ;DIGIT LOCATION IN DIGIT POINTER
0041 FA                        MOV    A,R2           ;SAVED DATA WORD TO ACCUMULATOR
0042 531F                      ANL    A,#1FH         ;DEFINE CHARACTER LOOK-UP-TABLE LOC.
0044 E3                        MOVP3  A,@A           ;GET CHARACTER
0045 AA                        MOV    R2,A           ;SAVE CHARACTER
0046 D37F                      XRL    A,#7FH         ;IS CHARACTER DECIMAL POINT
0048 C64E                      JZ     DPOINT         ;
004A FA                        MOV    A,R2           ;SAVED CHARACTER TO ACCUMULATOR
004B A0                        MOV    @R0,A          ;CHARACTER TO DISPLAY MAP
004C 0451                      JMP    RETURN         ;
004E FA                DPOINT: MOV    A,R2           ;SAVED CHARACTER TO ACCUMULATOR
004F 50                        ANL    A,@R0          ;"AND" WITH OLD CHARACTER
0050 A0                        MOV    @R0,A          ;BACK TO DISPLAY MAP
0051 FF                RETURN: MOV    A,R7           ;RESTORE ACCUMULATOR
0052 93                        RETR
                       ;*************************************************************************
                       $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0                 PAGE    6

  LOC  OBJ                SOURCE STATEMENT

                  ;**********************************************************************
                  ;                              LOOK-UP TABLE
                  ; THIS LOOK-UP TABLE ORIGINATES IN PAGE 3 OF THE UPI-41 PROGRAM
                  ;MEMORY.  IT IS USED TO DEFINE THE CORRECT LEVEL OF EACH SEGMENT
                  ;AND DECIMAL POINT FOR A SELECTED CHARACTER FROM THE INPUT ROUTINE.
                  ;INVERSE LOGIC IS USED BECAUSE OF THE SPECIFIC DRIVER CIRCUITRY, THUS
                  ;A 1 ON A GIVEN SEGMENT MEANS IT IS OFF AND A 0 MEANS IT IS ON.
                  ;
                                                    ;*******SEGMENTS********
  0300                        ORG     300H          ;DP  G  F  E  D  C  B  A
  0300 C0         CH0:        DB      0C0H          ;1   1  0  0  0  0  0  0
  0301 F9         CH1:        DB      0F9H          ;1   1  1  1  1  0  0  1
  0302 A4         CH2:        DB      0A4H          ;1   0  1  0  0  1  0  0
  0303 B0         CH3:        DB      0B0H          ;1   0  1  1  0  0  0  0
  0304 99         CH4:        DB      99H           ;1   0  0  1  1  0  0  1
  0305 92         CH5:        DB      92H           ;1   0  0  1  0  0  1  0
  0306 82         CH6:        DB      82H           ;1   0  0  0  0  0  1  0
  0307 F8         CH7:        DB      0F8H          ;1   1  1  1  1  0  0  0
  0308 80         CH8:        DB      80H           ;1   0  0  0  0  0  0  0
  0309 98         CH9:        DB      98H           ;1   0  0  1  1  0  0  0
  030A 88         CHA:        DB      88H           ;1   0  0  0  1  0  0  0
  030B 83         CHB:        DB      83H           ;1   0  0  0  0  0  1  1
  030C C6         CHC:        DB      0C6H          ;1   1  0  0  0  1  1  0
  030D A1         CHD:        DB      0A1H          ;1   0  1  0  0  0  0  1
  030E 86         CHE:        DB      86H           ;1   0  0  0  0  1  1  0
  030F 8E         CHF:        DB      8EH           ;1   0  0  0  1  1  1  0
  0310 7F         CHDP:       DB      7FH           ;0   1  1  1  1  1  1  1
  0311 C2         CHG:        DB      0C2H          ;1   1  0  0  0  0  1  0
  0312 89         CHH:        DB      89H           ;1   0  0  0  1  0  0  1
  0313 FB         CHI:        DB      0FBH          ;1   1  1  1  1  0  1  1
  0314 E1         CHJ:        DB      0E1H          ;1   1  1  0  0  0  0  1
  0315 C7         CHL:        DB      0C7H          ;1   1  0  0  0  1  1  1
  0316 AB         CHN:        DB      0ABH          ;1   0  1  0  1  0  1  1
  0317 A3         CHO:        DB      0A3H          ;1   0  1  0  0  0  1  1
  0318 8C         CHP:        DB      8CH           ;1   0  0  0  1  1  0  0
  0319 AF         CHR:        DB      0AFH          ;1   0  1  0  1  1  1  1
  031A 87         CHT:        DB      87H           ;1   0  0  0  0  1  1  1
  031B C1         CHU:        DB      0C1H          ;1   1  0  0  0  0  0  1
  031C 91         CHY:        DB      91H           ;1   0  0  1  0  0  0  1
  031D BF         CHDASH:     DB      0BFH          ;1   0  1  1  1  1  1  1
  031E FD         CHAPOS:     DB      0FDH          ;1   1  1  1  1  1  0  1
  031F FF         BLANK:      DB      0FFH          ;1   1  1  1  1  1  1  1
                  ;*************************************************************************
                              END

USER SYMBOLS
BLANK  031F   BLKMAP 000E   CH0    0300   CH1    0301   CH2    0302   CH3    0303   CH4    0304   CH5    0305
CH6    0306   CH7    0307   CH8    0308   CH9    0309   CHA    030A   CHAPOS 031E   CHB    030B   CHC    030C
CHD    030D   CHDASH 031D   CHDP   0310   CHE    030E   CHF    030F   CHG    0311   CHH    0312   CHI    0313
CHJ    0314   CHL    0315   CHN    0316   CHO    0317   CHP    0318   CHR    0319   CHT    031A   CHU    031B
CHY    031C   DISPLA 001D   DPOINT 004E   INPUT  0036   RETURN 0051   SETIME 0030   START  0009   TIME   FFF1

ASSEMBLY COMPLETE,   NO ERRORS
```

```
:F1:ASM48 :F3:SENSOR NOOBJECT PRINT(:LP:)


ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0                PAGE    1

 LOC  OBJ          LINE        SOURCE STATEMENT

                        $MOD41A
                        ;         **********************************************
                        ;         *       UPI-41A SENSOR MATRIX CONTROLLER     *
                        ;         **********************************************
                        ;
                        ;        THIS PROGRAM USES THE UPI-41A AS A SENSOR MATRIX CONTROLLER.
                        ;IT HAS MONITORING CAPABILITIES OF UP TO 128 SENSORS.  THE COORDINATE
                        ;AND SENSOR STATUS OF EACH DETECTED CHANGE IS AVAILABLE TO THE MASTER
                        ;MICROPROCESSOR IN A SINGLE BYTE.  A 40X8 FIFO QUEUE IS PROVIDED FOR
                        ;DATA BUFFERING.  BOTH HARDWARE OR POLLED INTERRUPT METHODS CAN BE USED
                        ;TO NOTIFY THE MASTER OF A DETECTED SENSOR CHANGE.
                        ;
                        ;*********************************************************************
                        ;
                        ;REGISTER DEFINITIONS:
                        ;        REGISTER               RBO                   RBI
                        ;        --------               ---                   ---
                        ;           R0          MATRIX MAP POINTER         NOT USED
                        ;           R1          FIFO POINTER               NOT USED
                        ;           R2          SCAN ROW SELECT            NOT USED
                        ;           R3          COLUMN COUNTER             NOT USED
                        ;           R4          FIFO-IN                    NOT USED
                        ;           R5          FIFO-OUT                   NOT USED
                        ;           R6          CHANGE WORD                NOT USED
                        ;           R7          COMPARE                    NOT USED
                        ;
                        ;*********************************************************************
                        ;
                        ;PORT PIN DEFINITIONS:
                        ;
                        ;PIN         PORT 1 FUNCTION     PIN        PORT 2 FUNCTION
                        ;---         ---------------     ---        ---------------
                        ;P0-7        COLUMN LINE INPUTS  P0-3       ROW SELECT OUTPUTS
                        ;                                P4         FIFO NOT EMPTY INTERRUPT
                        ;                                P5         OBF INTERRUPT
                        ;                                P6-7       NOT USED
                        ;
                        ;*********************************************************************

                        $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0                PAGE    2

 LOC  OBJ          LINE        SOURCE STATEMENT

                        ;*********************************************************************
                        ;
                        ;CHANGE WORD BIT DEFINITION:
                        ;
                        ;                  BIT                   FUNCTION
                        ;                  ---                   --------
                        ;                  D0-6                  SENSOR COORDINATE
                        ;                  D7                    SENSOR STATUS
                        ;
                        ;***********************************************************************
                        ;
                        ;STATUS REGISTER BIT DEFINITION:
                        ;
                        ;                  BIT                   FUNCTION
                        ;                  ---                   --------
                        ;                  D0                    OBF
                        ;                  D1-3                  IBF, F0, F1 (NOT USED)
                        ;                  D4                    FIFO NOT EMPTY
                        ;                  D5-7                  USED DEFINED (NOT USED)
                        ;
                        ;***********************************************************************
                        ;
                        ;                            EQUATES
                        ;
                        ;THE FOLLOWING CODE DESIGNATES THREE VARIABLES; SCANTM,FIFOBA
                        ;AND FIFOTA.  SCANTM ADJUSTS THE LENGTH OF A DELAY BETWEEN
                        ;SCANNING SWITCH.  THIS SIMULATES DEBOUNCE FUNCTIONS.  FIFOBA
                        ;IS THE BOTTOM ADDRESS OF THE FIFO.  FIFOTA IS THE TOP ADDRESS
                        ;OF THE FIFO. THIS MAKES IT POSSIBLE TO HAVE A FIFO 3 TO 40
                        ;BYTES IN LENGTH.
                        ;
                        ;*********************************************************************
                        ;
 000F                   SCANTM  EQU     0FH             ;SCAN TIME ADJUST
 0008                   FIFOBA  EQU     08H             ;FIFO BOTTOM ADDRESS
 002F                   FIFOTA  EQU     2FH             ;FIFO TOP ADDRESS

                        $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0              PAGE     3

 LOC  OBJ         LINE        SOURCE STATEMENT

                       ;**********************************************************************
                       ;
                       ;                               INITIALIZATION
                       ;
                       ;THE PROGRAM STARTS AT THE FOLLOWING CODE UPON RESET.  WITHIN
                       ;THIS INITIALIZATION SECTION THE REGISTERS THAT MAINTAIN THE MATRIX
                       ;MAP,FIFO AND ROW SCANNING ARE SET UP.  PORT 1 IS SET HIGH FOR USE
                       ;AS AN INPUT PORT FOR THE COLUMN STATUS.  BIT 4 OF STATUS REGISTER IS
                       ;WRITTEN TO CONVEY A FIFO EMPTY CONDITION.  THE INITIAL COLUMN STATUS
                       ;OF ALL THE ROWS IN THE SENSOR MATRIX IS THEN READ INTO THE MATRIX
                       ;MAP.  ONCE THE MATRIX MAP IS FILLED THE OBF INTERRUPT (PORT 2-4) IS
                       ;ENABLED.
                       ;
                       ;***********************************************************************
                       ;
 0000                          ORG    0
 0000 B83F             INITMX: MOV    R0,#3FH         ;MATRIX MAP POINTER REGISTER,TOP ADDRESS
 0002 BA0F                     MOV    R2,#0FH         ;SCAN ROW SELECT REGISTER,TOP ROW
 0004 BC08                     MOV    R4,#FIFOBA      ;FIFO INPUT ADDRESS REGISTER,BOTTOM OF FIFO
 0006 BD2F                     MOV    R5,#FIFOTA      ;FIFO OUTPUT ADDRESS REGISTER,TOP OF FIFO
 0008 89FF                     ORL    P1,#0FFH        ;INITIALIZE PORT 1 HIGH FOR INPUTS
 000A 2300                     MOV    A,#00H          ;INITIALIZE STATUS REGISTER, FIFO EMPTY
 000C 90                       MOV    STS,A           ;WRITE TO STATUS REGISTER, BITS 4-7
 000D FA               FILLMX: MOV    A,R2            ;SCAN ROW SELECT TO ACCUMULATOR
 000E 3A                       OUTL   P2,A            ;OUTPUT SCAN ROW SELECT TO PORT 2
 000F 09                       IN     A,P1            ;INPUT COLUMN STATUS PORT 1
 0010 A0                       MOV    @R0,A           ;LOAD MATRIX MAP WITH COLUMN STATUS
 0011 FA                       MOV    A,R2            ;CHECK SCAN ROW SELECT REGISTER VALUE FOR 0
 0012 C618                     JZ     OBFINT          ;IF 0 ENABLE OBF INTERRUPT
 0014 C8                       DEC    R0              ;DECREMENT TO NEXT MATRIX MAP ADDRESS
 0015 CA                       DEC    R2              ;DECREMENT TO SCAN NEXT ROW
 0016 040D                     JMP    FILLMX          ;FILL NEXT MATRIX MAP ADDRESS
 0018 BA10             OBFINT: MOV    R2,#10H         ;BIT 4 HIGH IN ROW SCAN SELECT REGISTER
 001A FA                       MOV    A,R2            ;ROW SCAN SELECT VALUE TO ACCUMULATOR
 001B 3A                       OUTL   P2,A            ;INITIALIZE PORT 2, BIT 4 FOR "EN FLAGS"
 001C F5                       EN     FLAGS           ;ENABLE OBF INTERRUPT PORT 2, BIT 4

                       $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0                PAGE    4

 LOC  OBJ         LINE        SOURCE STATEMENT

                      ;*************************************************************************
                      ;
                      ;                             SCAN AND COMPARE
                      ;
                      ;THE FOLLOWING CODE IS THE SCAN AND COMPARE SECTION OF THE PROGRAM.
                      ;UPON ENTERING THIS SECTION A CHECK IS MADE TO SEE IF THE ENTIRE MATRIX
                      ;HAS BEEN SCANNED.  IF SO THE REGISTERS THAT MAINTAIN THE MATRIX MAP AND ROW
                      ;SCANNING ARE RESET TO THE BEGINNING OF THE SENSOR MATRIX.  IF THE ENTIRE
                      ;MATRIX HASNT BEEN SCANNED THE REGISTERS INCREMENT TO SCAN THE NEXT ROW.
                      ;FROM THIS POINT ON THE ROW SCAN SELECT REGISTER IS USED FOR TWO FUNCTIONS.
                      ;BITS 0-3 FOR SCANNING AND BITS 4 AND 5 FOR THE EXTERNAL INTERRUPTS.  THUSLY
                      ;ALL USAGE OF THE REGISTERS IS DONE BY LOGICALLY MASKING IT SO AS TO ONLY
                      ;AFFECT THE FUNCTION DESIRED.  ONCE THE REGISTERS ARE RESET, ONE ROW OF THE
                      ;SENSOR MATRIX IS SCANNED.  A DELAY IS EXECUTED TO ADJUST FOR SCAN TIME
                      ;(DEBOUNCE).  A BYTE OF COLUMN STATUS IS THEN READ INTO THE MATRIX MAP.
                      ;AT THE TIME THE NEW COLUMN STATUS IS COMPARED TO THE OLD.  THE RESULT IS
                      ;STORED IN THE COMPARE REGISTER.  THE PROGRAM IS THEN ROUTED ACCORDING TO
                      ;WHETHER OR NOT A CHANGE WAS DETECTED.
                      ;
                      ;************************************************************************

 001D FA              ADJREG: MOV     A,R2            ;SCAN ROW SELECT TO ACCUMULATOR
 001E 530F                    ANL     A,#0FH          ;CHECK FOR 0 SCAN VALUE ONLY,NOT INTERRUPT
 0020 C626                    JZ      RSETRG          ;IF 0 RESET REGISTERS
 0022 C8                      DEC     R0              ;DECREMENT MATRIX MAP POINTER
 0023 CA                      DEC     R2              ;DECREMENT SCAN ROW SELECT
 0024 042C                    JMP     SCANMX          ;SCAN MATRIX
 0026 B83F            RSETRG: MOV     R0,#3FH         ;RESET MATRIX MAP POINTER REGISTER,TOP ADDRESS
 0028 FA                      MOV     A,R2            ;SCAN ROW SELECT TO ACCUMULATOR
 0029 430F                    ORL     A,#0FH          ;RESET SCAN ROW SELECT,NO INTERRUPT CHANGE
 002B AA                      MOV     R2,A            ;SCAN ROW SELECT REGISTER
 002C FA              SCANMX: MOV     A,R2            ;SCAN ROW SELECT TO ACCUMULATOR
 002D 3A                      OUTL    P2,A            ;OUTPUT SCAN ROW SELECT TO PORT 2
 002E BB0F                    MOV     R3,#SCANTM      ;SET DELAY FOR OUTPUT SCAN TIME
 0030 EB30            DELAY2: DJNZ    R3,DELAY2       ;DELAY
 0032 09                      IN      A,P1            ;INPUT COLUMN STATUS FROM PORT 1 TO ACCUMULATOR
 0033 20                      XCH     A,@R0           ;STORE NEW COLUMN STATUS SAVE OLD IN ACCUMULATOR
 0034 D0                      XRL     A,@R0           ;COMPARE OLD WITH NEW COLUMN STATUS
 0035 AF                      MOV     R7,A            ;SAVE COMPARE RESULT IN COMPARE REGISTER
 0036 C669                    JZ      CHFFUL          ;IF THE SAME, CHECK IF FIFO IS FULL

                      $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0              PAGE    5

 LOC  OBJ         LINE        SOURCE STATEMENT

                       ;*************************************************************************
                       ;
                       ;                          CHANGE WORD ENCODING
                       ;
                       ;THE FOLLOWING CODE IS THE CHANGE WORD ENCODING SECTION.  THIS
                       ;SECTION IS ONLY EXECUTED IF A CHANGE WAS DETECTED.  THE COLUMN COUNTER
                       ;IS SET AND DECREMENTED TO DESIGNATE EACH OF THE 8 COLUMNS.  THE COMPARE
                       ;REGISTER IS LOOKED AT ONE BIT AT A TIME TO FIND THE EXACT LOCATION OF
                       ;THE CHANGE(S).  WHEN A CHANGE IS FOUND IT IS ENCODED BY GIVING IT A
                       ;COORDINATE FOR ITS LOCATION.  THIS IS DONE BY COMBINING THE PRESENT VALUE
                       ;IN THE ROW SCAN SELECT REGISTER AND THE COLUMN COUNTER.  THE ACTUAL STATUS
                       ;OF THAT SENSOR IS ESTABLISHED BY LOOKING AT THE CORRESPONDING BYTE IN
                       ;THE MATRIX MAP.  THIS STATUS IS COMBINED WITH THE COORDINATE TO ESTABLISH
                       ;THE CHANGE WORD.  THE CHANGE WORD IS THEN STORED IN THE CHANGE WORD REGISTER.
                       ;
                       ;**********************************************************************
                       ;
0038 BB08                      MOV    R3,#08H        ;SET COLUMN COUNTER REGISTER TO 8
003A CB                RRLOOK: DEC    R3             ;DECREMENT COLUMN COUNTER
003B F0                        MOV    A,@R0          ;COLUMN STATUS TO ACCUMULATOR
003C 77                        RR     A              ;ROTATE COLUMN STATUS RIGHT
003D A0                        MOV    @R0,A          ;ROTATED COLUMN STATUS BACK TO MATRIX MAP
003E FF                        MOV    A,R7           ;COMPARE REGISTER VALUE TO ACCUMULATOR
003F 77                        RR     A              ;ROTATE COMPARE VALUE RIGHT
0040 AF                        MOV    R7,A           ;ROTATED COMPARE VALUE TO COMPARE REGISTER
0041 F245                      JB7    ENCODE         ;TEST BIT 7 IF CHANGE DETECTED ENCODE CHANGE WORD
0043 0469                      JMP    CHFFUL         ;IF NO CHANGE IS DETECTED CHECK FOR FIFO FULL
0045 FA                ENCODE: MOV    A,R2           ;SCAN ROW SELECT TO ACCUMULATOR  0000XXXX
0046 530F                      ANL    A,#0FH         ;ROTATE ONLY SCAN VALUE
0048 E7                        RL     A              ;ROTATE LEFT                     000XXXX0
0049 E7                        RL     A              ;ROTATE LEFT                     00XXXX00
004A E7                        RL     A              ;ROTATE LEFT                     0XXXX000
004B 4B                        ORL    A,R3           ;ESTABLISH MATRIX COORDINANT     0XXXXXXX
                                                     ;(OR) COLUMN COUNTER VALUE WITH ACCUMULATOR
004C AE                        MOV    R6,A           ;SAVE COORDINANT IN CHANGE WORD REGISTER
004D F0                        MOV    A,@R0          ;COLUMN STATUS FROM MATRIX MAP TO ACCUMULATOR
004E 5380                      ANL    A,#80H         ;O ALL BITS BUT BIT 7
0050 4E                        ORL    A,R6           ;(OR) SENSOR STATUS WITH COORDINATE FOR COMPLETED CHANGE WORD
0051 AE                        MOV    R6,A           ;SAVE CHANGE WORD                 XXXXXXXX

                       $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0              PAGE    6

 LOC  OBJ         LINE        SOURCE STATEMENT

                      ;*********************************************************************
                      ;
                      ;                    FIFO-DBBOUT MANAGEMENT
                      ;
                      ;THE FOLLOWING CODE IS THE FIFO-DBBOUT MANAGEMENT SECTION OF THE
                      ;PROGRAM.  THIS SECTION TAKES AN ENCODED CHANGE WORD AND LOADS IT INTO
                      ;THE FIFO.  THE FIFO NOT EMPTY INTERRUPT IS THEN SET AND THE FIFO-IN
                      ;POINTER GETS UPDATED. A FIFO FULL CONDITION IS THEN CHECKED FOR AND
                      ;ROUTED ACCORDINGLY.  IF BOTH THE FIFO AND OBF HAVE CHANGE WORDS THE
                      ;PROGRAM LOCKS UP UNTIL THIS HAS CHANGED.  IF THE FIFO ISNT FULL COLUMN
                      ;COUNTER= 0, FIFO EMPTY AND OBF CONDITIONS ARE CHECKED.  THE FIFO-OUT
                      ;POINTER IS SET AND DBBOUT IS LOADED IF THE FIFO ISNT EMPTY AND OBF ISNT
                      ;SET.  IF THIS ISNT THE SITUATION, PROGRAM FLOW IS ROUTED BACK TO THE
                      ;THE SCAN AND COMPARE SECTION TO SCAN THE NEXT ROW.
                      ;
                      ;*********************************************************************
                      ;
0052 FC               LOADFF: MOV    A,R4           ;FIFO INPUT ADDRESS TO ACCUMULATOR
0053 A9                       MOV    R1,A           ;FIFO POINTER USED FOR INPUT
0054 FE                       MOV    A,R6           ;CHANGE WORD TO ACCUMULATOR
0055 A1                       MOV    @R1,A          ;LOAD FIFO AT FIFO INPUT ADDRESS
0056 2310             STATNE: MOV    A,#10H         ;BIT 4 FOR FIFO NOT EMPTY
0058 90                       MOV    STS,A          ;WRITE TO STATUS REGISTER, FIFO NOT EMPTY
0059 8A20             INTRH1: ORL    P2,#20H        ;FIFO NOT EMPTY INTERRUPT PORT 2-5 HIGH
005B FA                       MOV    A,R2           ;ROW SCAN SELECT TO ACCUMULATOR
005C 4320                     ORL    A,#20H         ;SAVE INTERRUPT, NO CHANGE TO SCAN VALUE
005E AA                       MOV    R2,A           ;ROW SCAN SELECT REGISTER
005F 232F             ADJFIN: MOV    A,#FIFOTA      ;FIFO TOP ADDRESS TO ACCUMULATOR
0061 DC                       XRL    A,R4           ;COMPARE WITH CURRENT FIFO INPUT ADDRESS
0062 C667                     JZ     RSFFIN         ;IF THE SAME RESET FIFO INPUT REGISTER
0064 1C                       INC    R4             ;NEXT FIFO INPUT ADDRESS
0065 0469                     JMP    CHFFUL         ;CHECK FIFO FULL
0067 BC08             RSFFIN: MOV    R4,#FIFOBA     ;RESET FIFO INPUT REGISTER,BOTTOM OF FIFO
0069 FC               CHFFUL: MOV    A,R4           ;FIFO INPUT ADDRESS TO ACCUMULATOR
006A DD                       XRL    A,R5           ;COMPARE INPUT WITH OUTPUT FIFO ADDRESS
006B 967D                     JNZ    CHCNTR         ;IF NOT SAME CHECK COLUMN COUNTER VALUE
006D 866D             CHOBF1: JOBF   CHOBF1         ;IF OBF IS 1 THEN CHECK OBF
006F 232F             ADJFOT: MOV    A,#FIFOTA      ;FIFO TOP ADDRESS TO ACCUMULATOR
0071 DD                       XRL    A,R5           ;COMPARE TOP TO OUTPUT FIFO ADDRESS
0072 C677                     JZ     RSFFOT         ;IF THE SAME RESET FIFO OUTPUT REGISTER
0074 1D                       INC    R5             ;NEXT FIFO OUTPUT ADDRESS
0075 0479                     JMP    LOADDB         ;LOAD DBBOUT
0077 BD08             RSFFOT: MOV    R5,#FIFOBA     ;RESET FIFO OUTPUT ADDRESS TO BOTTOM OF FIFO
0079 FD               LOADDB: MOV    A,R5           ;OUTPUT FIFO ADDRESS TO ACCUMULATOR
007A A9                       MOV    R1,A           ;FIFO POINTER USED FOR OUTPUT
007B F1                       MOV    A,@R1          ;CHANGE WORD TO ACCUMULATOR
007C 02                       OUT    DBB,A          ;CHANGE WORD TO DBBOUT
007D FB               CHCNTR: MOV    A, R3          ;COLUMN COUNTER TO ACCUMULATOR
007E 963A                     JNZ    RRLOOK         ;IF NOT 0 FINISH CHANGE WORD ENCODING
0080 2308             CHFFEM: MOV    A,#FIFOBA      ;FIFO BOTTOM ADDRESS TO ACCUMULATOR

                      $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3 0                    PAGE    7

 LOC  OBJ          LINE        SOURCE STATEMENT

 0082 DC            252          XRL      A,R4              ;COMPARE FIFO INPUT ADDRESS WITH FIFO BOTTOM ADD
 0083 C68C          253          JZ       ADJFEM            ;IF THE SAME, ADJUST TO CHECK FOR FIFO EMPTY
 0085 FC            254          MOV      A,R4              ;FIFO INPUT ADDRESS TO ACCUMULATOR
 0086 07            255          DEC      A                 ;DECREMENT FIFO INPUT ADDRESS IN ACCUMULATOR
 0087 DD            256          XRL      A,R5              ;COMPARE INPUT TO OUTPUT FIFO ADDRESSES
 0088 C691          257          JZ       STATMT            ;IF SAME,WRITE STATUS REGISTER FOR FIFO EMPTY
 008A 049C          258          JMP      CHOBF2            ;CHECK OBF
 008C 232F          259 ADJFEM:  MOV      A,#FIFOTA         ;FIFO TOP ADDRESS TO ACCUMULATOR
 008E DD            260          XRL      A,R5              ;COMPARE TOP TO OUTPUT FIFO ADDRESS
 008F 969C          261          JNZ      CHOBF2            ;IF NOT SAME THEN FIFO IS NOT EMPTY, CHECK OBF
 0091 2300          262 STATMT:  MOV      A,#00H            ;CLEAR BIT 0 FOR FIFO EMPTY
 0093 90            263          MOV      STS,A             ;WRITE TO STATUS REGISTER
 0094 9ADF          264 INTRLO:  ANL      P2,#0DFH          ;FIFO EMPTY, INTERRUPT PORT 2-5 LOW
 0096 FA            265          MOV      A,R2              ;SCAN ROW SELECT TO ACCUMULATOR
 0097 53DF          266          ANL      A,#0DFH           ;SAVE INTERRUPT, NO CHANGE TO SCAN VALUE
 0099 AA            267          MOV      R2,A              ;SCAN ROW SELECT REGISTER
 009A 041D          268          JMP      ADJREG            ;ADJUST REGISTERS
 009C 861D          269 CHOBF2:  JOBF     ADJREG            ;IF OBF=1 THEN ADJUST REGISTERS
 009E 046F          270          JMP      ADJFOT            ;ADJUST FIFO OUT ADDRESS TO LOAD DBBOUT
                    271
                    272          END

USER SYMBOLS
ADJFEM 008C    ADJFIN 005F    ADJFOT 006F    ADJREG 001D    CHCNTR 007D    CHFFEM 0080    CHFFUL 0069    CHOBF1 006D
CHOBF2 009C    DELAY2 0030    ENCODE 0045    FIFOBA 0008    FIFOTA 002F    FILLMX 000D    INITMX 0000    INTRH1 0059
INTRLO 0094    LOADDB 0079    LOADFF 0052    OBFINT 0018    RRLOOK 003A    RSETRG 0026    RSFFIN 0067    RSFFOT 0077
SCANMX 002C    SCANTM 000F    STATMT 0091    STATNE 0056

ASSEMBLY COMPLETE,    NO ERRORS
```

PROGRAMMABLE KEYBOARD INTERFACE

- **Simultaneous Keyboard and Display Operations**
- **Interface Signals for Contact and Capacitive Coupled Keyboards**
- **128-Key Scanning Logic**
- **10.7msec Matrix Scan Time for 128 Keys and 6MHz Clock**
- **Eight Character Keyboard FIFO**
- **N-Key Rollover with Programmable Error Mode on Multiple New Closures**
- **Sixteen or Eight Character Seven-Segment Display Interface**
- **Right or Left Entry Display RAM**
- **Depress/Release Mode Programmable**
- **Interrupt Output on Key Entry**

This application is a general purpose programmable keyboard and display interface device designed for use with 8-bit microprocessors like the MCS-80 and MCS-85. The keyboard portion can provide a scanned interface to 128-key contact or capacitive-coupled keyboards. The keys are fully debounced with N-key rollover and programmable error generation on multiple new key closures. Keyboard entries are stored in an 8-character FIFO with overrun status indication when more than 8 characters are entered. Key entries set an interrupt request output to the master CPU.

The display portion of the UPI-41A provides a scanned display interface for LED, incandescent and other popular display technologies. Both numeric displays and simple indicators may be used. The UPI-41A has a 16×4 display RAM which can be

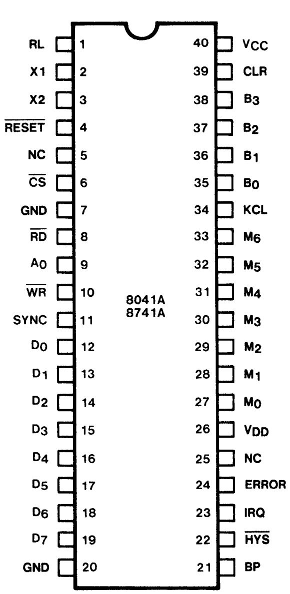


Figure 1. Pin Configuration

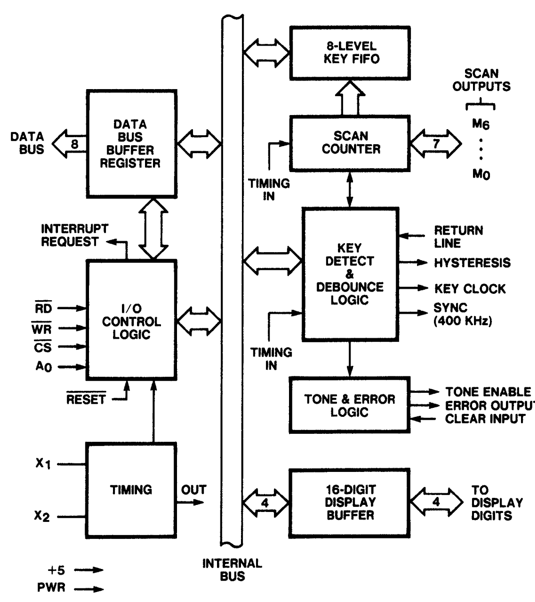


Figure 2. Block Diagram

loaded or interrogated by the CPU. Both right entry calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto increment of the display RAM address.

ORDERING INFORMATION:

This part may be ordered as an 8041A with ROM code number 8278. The source code is available through Insite.

Throughout this application of the UPI-41A, it will be referred to by its ROM code number, 8278. The 8278 is packaged in a 40-pin DIP. The following is a brief functional description of each pin.

PRINCIPLES OF OPERATION

The following is a description of the major elements of the Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 1.

I/O Control and Data Buffers

The I/O control section uses the $\overline{CS}$, A_0, $\overline{RD}$, and $\overline{WR}$ lines to control data flow to and from the various internal registers and buffers (see Table 2). All data flow to and from the 8278 is enabled by $\overline{CS}$. The 8-bits of information being transferred by the CPU is identified by A_0. A logic one means information is command or status. A logic zero means the information is data. $\overline{RD}$ and $\overline{WR}$ determine the direction of data flow through the Data Bus Buffer (DBB). The

Table 1. Pin Description

Signal	Pin. No.	Type	Name and Function
D_0-D_7	12-19	I/O	**Data Bus:** Three-state, bi-directional data bus lines used to transfer data and commands between the CPU and the 8278.
$\overline{WR}$	10	I	**Write:** Write strobe which enables the master CPU to write data and commands between the CPU and the 8278.
$\overline{RD}$	8	I	**Read:** Read strobe which enables the master CPU to read data and status from the 8278 internal registers.
$\overline{CS}$	6	I	**Chip Select:** Chip select input used to enable reading and writing to the 8278.
A_0	9	I	**Control/Data:** Address input used by the CPU to indicate control or data.
$\overline{RESET}$	4	I	**Reset:** A low signal on this pin resets the 8278.
X_1, X_2	2,3	I	**Freq. Reference Inputs:** Inputs for crystal, L-C or external timing signal to determine internal oscillator frequency.
IRQ	23	O	**Interrupt Request:** Interrupt Request Output to the master CPU. In the keyboard mode the IRQ line goes low with each FIFO read and returns high if there is still information in the FIFO or an ERROR has occurred.
M_0–M_6	27-33	O	**Matrix Scan Lines:** Matrix scan outputs. These outputs control a decoder which scans the key matrix columns and the 16 display digits. Also, the Matrix scan outputs are used to multiplex the return lines from the key matrix.
RL	1	I	**Keyboard Return Line:** Input from the multiplexer which indicates whether the key currently being scanned is closed.
$\overline{HYS}$	22	O	**Hysteresis:** Hysteresis output to the analog detector. (Capacitive keyboard configuration). A "0" means the key currently being scanned has already been recorded.
KCL	34	O	**Key Clock:** Key Clock output to the analog detector (capacitive keyboard configuration) used to reset the detector before scanning a key.
SYNC	11	O	**Output Clock:** High frequency (400 kHz) output signal used in the key scan to detect a closed key (capacitive keyboard configuration).
B_0-B_3	35-38	O	**Display Outputs:** These four lines contain binary coded decimal display information synchronized to the keyboard column scan. The outputs are for multiplexed digital displays.
ERROR	24	O	**Error Signal:** This line is high whenever two new key closures are detected during a single scan or when too many characters are entered into the keyboard FIFO. It is reset by a system RESET pulse or by a "1" input on the CLR pin or by the CLEAR ERROR command.
CLR	39	I	**Clear Error:** Input used to clear an ERROR condition in the 8278.
BP	21	O	**Tone Enable:** Tone enable output. This line is high for 10ms following a valid key closure; it is set high and remains high during an ERROR condition.
V_{CC}, V_{DD}	40,26	I	**Power:** +5 volt power input: +5V ± 10%.
GND	20,7	I	**Ground:** Signal ground.

DBB register is a bi-directional 8-bit buffer register which connects the internal 8278 bus buffer register to the external bus. When the chip is not selected ($\overline{CS}$ = 1) the DBB is in the high impedance state. The DBB acts as an input when ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$) = (1, 0, 0) and an output when ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$) = (0, 1, 0).

Table 2. I/O Control and Data Buffers

$\overline{CS}$	A_0	$\overline{WR}$	$\overline{RD}$	Condition
0	0	1	0	Read DBB Data
0	1	1	0	Read STATUS
0	0	0	1	Write Data to DBB
0	1	0	1	Write Command to DBB
1	X	X	X	Disable 8278 Bus, High Impedance

Scan Counter

The scan counter provides the timing to scan the keyboard and display. The four MSB's (M_3-M_6) scan the display digits and provide column scan to the keyboard via a 4 to 16 decoder. The three LSB's (M_0-M_2) are used to multiplex the row return lines into the 8278.

Keyboard Debounce and Control

The 8278 system configuration is shown in Figure 3. The rows of the matrix are scanned and the outputs are multiplexed by the 8278. When a key closure is detected, the debounce logic waits about 12 msec to check if the key remains closed. If it does, the address of the key in the matrix is transferred into a FIFO buffer.

FIFO and FIFO Status

The 8278 contains an 8X8 FIFO character buffer. Each new entry is written into a successive FIFO location and each is then read out in the order of entry. A FIFO status register keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or key entries will be recognized as an error. The status can be read by a $\overline{RD}$ with $\overline{CS}$ low and A_0 high. The status logic also provides a IRQ signal to the master processor whenever the FIFO is not empty.

Display Address Registers and Display RAM

The Display Address registers hold the address of the word currently being written or read by the CPU and the two 4-bit nibbles being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The display RAM can be directly read by the CPU after the correct mode and address is set. Data entry to the display can be set to either left or right entry.

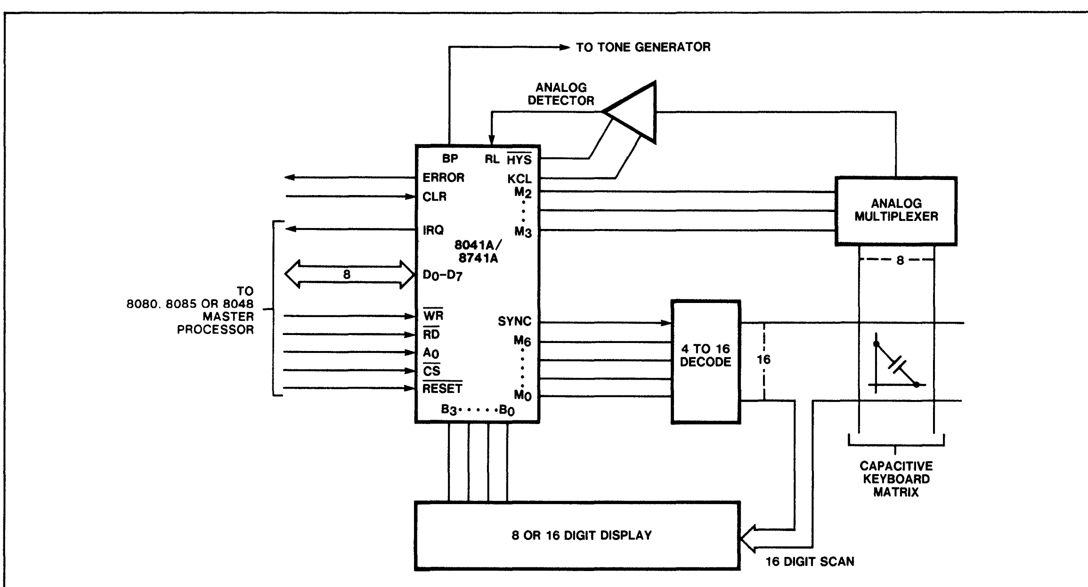


Figure 3. System Configuration for Capacitive-Coupled Keyboard

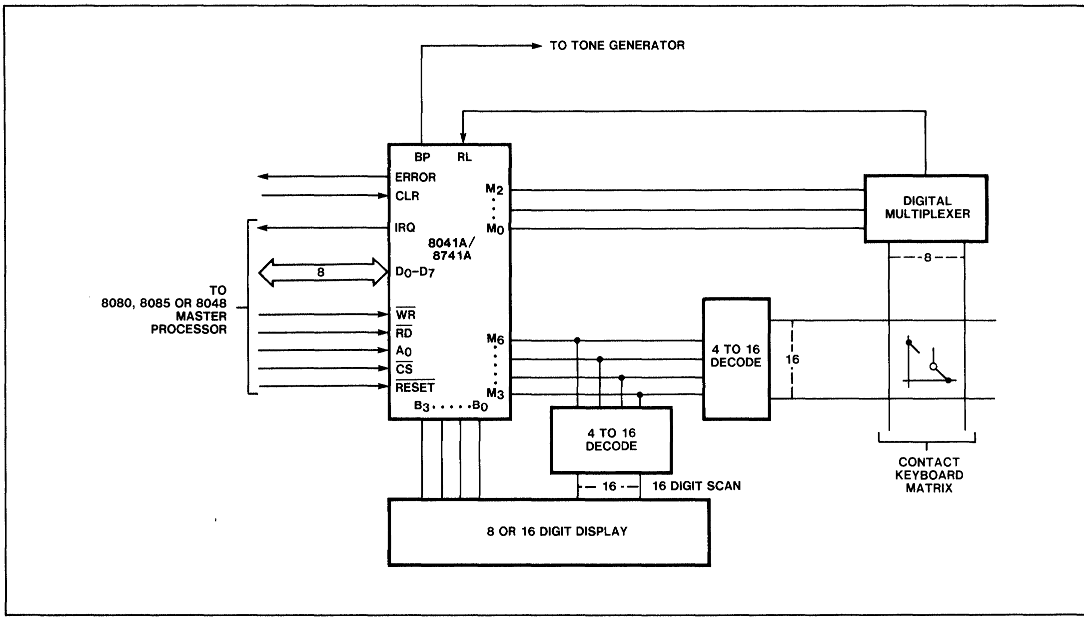
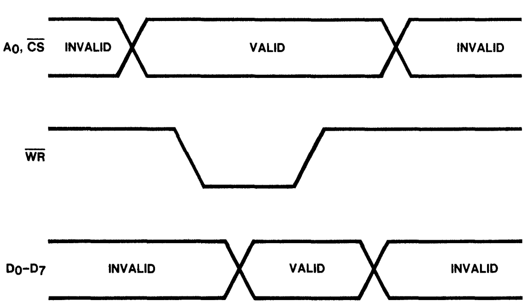


Figure 4. System Configuration for Contact Keyboard

COMMANDS

The 8278 operating mode is programmed by the master CPU using the A_0, WR and D_0-D_7 inputs as shown below:



The master CPU presents the proper command on the D_0-D_7 data lines with A_0 =1 and then sends a $\overline{WR}$ pulse. The command is latched by the 8278 on the rising edge of the $\overline{WR}$ and is decoded internally to set the proper operating mode. See the 8041A/8741A data sheet for timing details.

Command Summary

KEYBOARD/DISPLAY MODE SET

CODE	0	0	0	N	E	I	D	K

Where the mode set bits are defined as follows:

K—the keyboard mode select bit
- 0—normal key entry mode
- 1—special function mode: Entry on key closure and on key release

D—the display entry mode select bit
- 0—left display entry
- 1—right display entry

I—the interrupt request (IRQ) output enable bit.
- 0—enable IRQ output
- 1—disable IRQ output

E—the error mode select bit
- 0—error on multiple key depression
- 1—no error on multiple key depression

N—the number of display digits select
- 0—16 display digits
- 1—8 display digits

NOTE:
The default mode following a RESET input is all bits zero:

0	0	0	0	0	0	0	0

READ FIFO COMMAND

CODE	0	1	0	0	0	0	0	0

READ DISPLAY COMMAND

CODE	0	1	1	AI	A_3	A_2	A_1	A_0

Where AI indicates Auto Increment and A_3-A_0 is the address of the next display character to be read out.

AI = 1 AUTO increment
AI = 0 no AUTO increment

WRITE DISPLAY COMMAND

CODE								
	1	0	0	AI	A_3	A_2	A_1	A_0

Where AI indicates Auto Increment and A_3-A_0 is the address of the next display character to be written.

CLEAR/BLANK COMMAND

CODE								
	1	0	1	UD	BD	CD	CF	CE

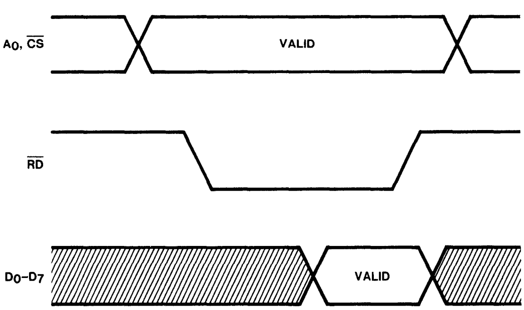
Where the command bits are defined as follows:

CE = Clear ERROR
CF = Clear FIFO
CD = Clear Display to all High
BD = Blank Display to all High
UD = Unblank Display

The display is cleared and blanked following a Reset.

Status Read

The status register in the 8278 can be read by the master CPU using the A_0, $\overline{RD}$, and D_0-D_7 inputs as shown below:



The 8278 places 8-bits of status information on the D_0-D_7 lines following (A_0, $\overline{CS}$, $\overline{RD}$) = 1, 0, 0 inputs from the master.

Status Format

S_3	S_2	S_1	S_0	B	KE	IBF	OBF
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Where the status bits are defined as follows:

IBF = Input Buffer Full Flag
OBF = Output Buffer Full Flag
KE = Keyboard Error Flag (multiple depression)
B = BUSY Flag
S_3-S_0 = FIFO Status

STATUS DESCRIPTION

The S_3-S_0 status bits indicate the number of entries (0 to 8) in the 8-level FIFO. A FIFO overrun will lock status at 1111. The overrun condition will prevent further key entries until cleared.

A multiple key closure error will set the KE flag and prevent further key entries until cleared.

The IBF and OBF flags signify the status of the 8278 data buffer registers used to transfer information (data, status or commands) to and from the master CPU.

The IBF flag is set when the master CPU writes Data or Commands to the 8278. The IBF flag is cleared by the 8278 during its response to the Data or Command.

The OBF flag is set when the 8278 has output data ready for the master CPU. This flag is cleared by a master CPU Data READ.

The Busy flag in the status register is used as a LOCKOUT signal to the master processor during response to any command or data write from the master.

The master must test the Busy flag before each read (during a sequence) to be sure that the 8278 is ready with valid DATA.

The ERROR and TONE outputs from the 8278 are set high for either type of error. Both types of error are cleared by the CLR input, by the CLEAR ERROR command, or by a reset. The FIFO and Display buffers are cleared independently of the Errors.

FIFO status is used to indicate the number of characters in the FIFO and to indiate whether an error has occurred. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO. The character read will be the last one entered. FIFO status will remain at 0000 and the error condition will not be set.

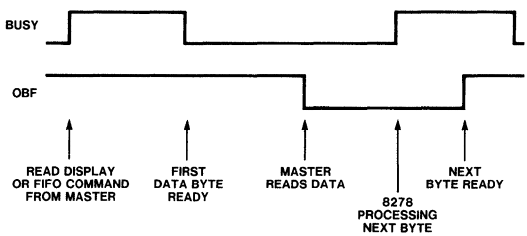
Data Read

The master CPU can read DATA from the 8278 FIFO or Display buffers by using the A_0, $\overline{RD}$, and D_0-D_7 inputs.

The master sends a $\overline{RD}$ pulse with A_0 = 0 and CS = 0 and the 8278 responds by outputting data on lines D_0-D_7. The data is strobed by the trailing edge of $\overline{RD}$.

DATA READ SEQUENCE

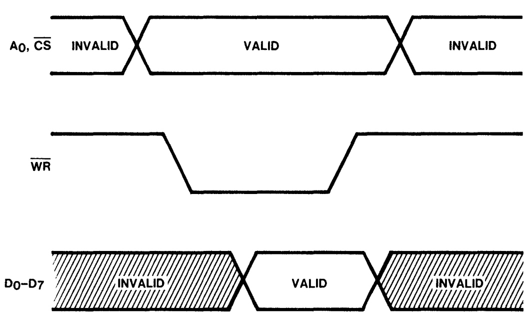
Before reading data, the master CPU must send a command to select FIFO or Display data. Following the command, the master must read STATUS and test the BUSY flag and the OBF flag to verify that the 8278 has responded to the previous command. A typical DATA READ sequence is as follows:



After the first read following a Read Display or Read FIFO command, successive reads may occur as soon as OBF rises.

Data Write

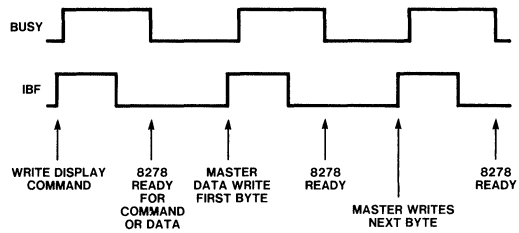
The master CPU can write DATA to the 8278 Display buffers by using the A_0, $\overline{WR}$ and D_0-D_7 inputs as follows:



The master CPU presents the Data on the D_0-D_7 lines with A_0=0 and then sends a $\overline{WR}$ pulse. The data is latched by the 8278 on the rising edge of $\overline{WR}$.

DATA WRITE SEQUENCE

Before writing data to the 8278, the master CPU must first send a command to select the desired display entry mode and to specify the address of the next data byte. Following the commands, the master must read STATUS and test the BUSY flag (B) and IBF flag to verify that the 8278 has responded. A typical sequence is shown below.



INTERFACE CONSIDERATIONS

Scanned Keyboard Mode

With N-key rollover each key depression is treated independently from all others. When a key is depressed the debounce logic waits for a full scan of 128 keys and then checks to see if the key is still down. If it is, the key is entered into the FIFO.

If two key closures occur during the same scan the ERROR output is set, the KE flag is set in the Status word, the TONE output is activated and IRQ is set, and no further inputs are accepted. This condition is cleared by a high signal on the CLEAR input or by a system $\overline{RESET}$ input or by the CLEAR ERROR command.

In the special function mode both the key closure and the key release cause an entry to the FIFO. The release is entered with the MSB=1.

Any key entry triggers the TONE output for 10ms.

The $\overline{HYS}$ and KCL outputs enable the analog multiplexer and detector to be synchronized for interface to capacitive coupled keyboards.

Data Format

In the scanned keyboard mode, the code entered into the FIFO corresponds to the position or address of the switch in the keyboard. The MSB is relevant only for special function keys in which code "0" signifies closure and "1" signifies release. The next four bits are the column count which indicates which column the key was found in. The last three bits are from the row counter.

BIT	7	6	5	4	3	2	1	0
	SF	M_6	M_5	M_4	M_3	M_2	M_1	M_0

SF: 1 FOR SPECIAL FUNCTION MODE AND KEY RELEASED
0 FOR KEY DEPRESSED

Display

Display data is entered into a 16×4 display register and may be entered from the left, from the right or

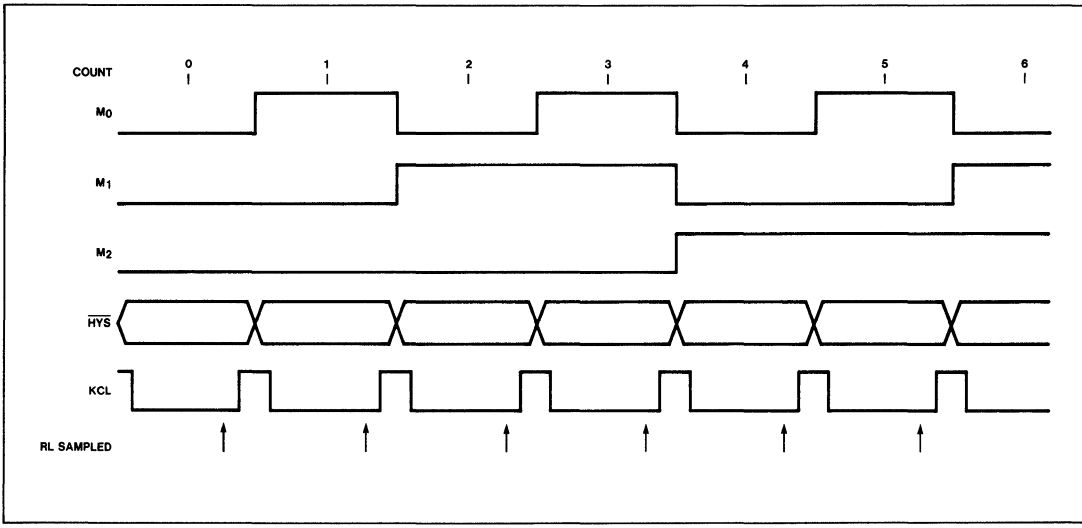


Figure 5. Keyboard Timing

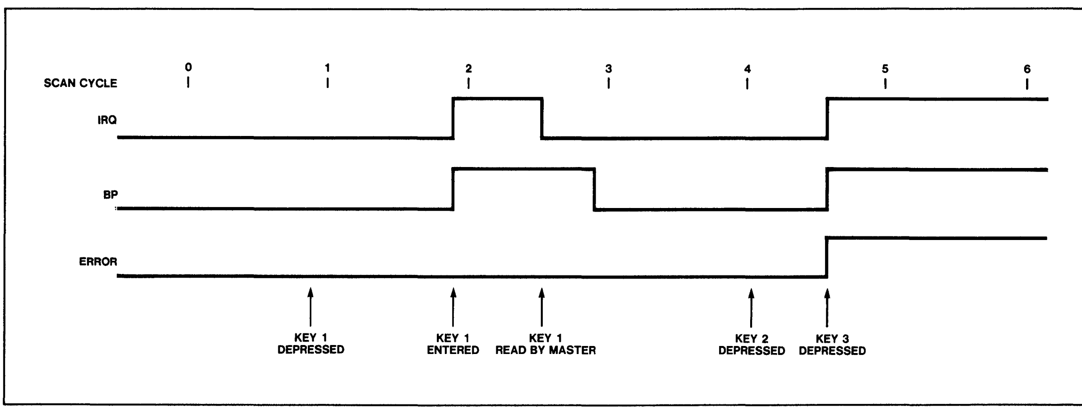


Figure 6. Key Entry and Error Timing

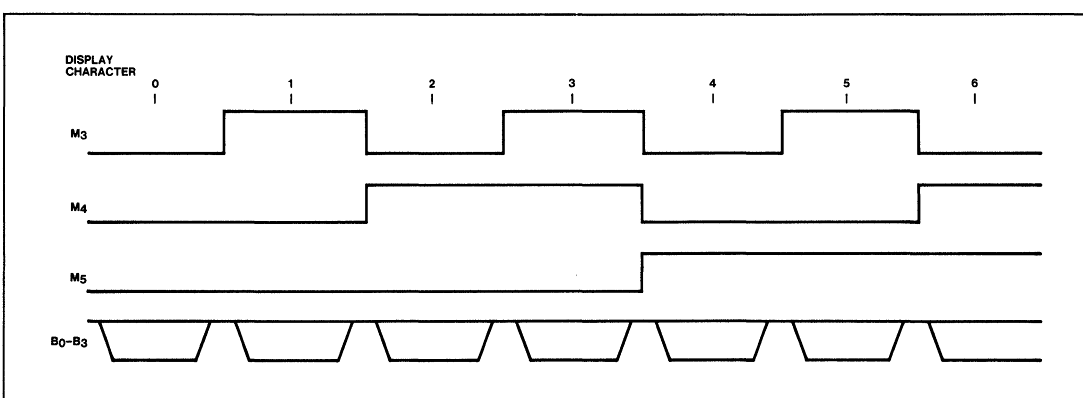


Figure 7. Display Timing

into specific locations in the display register. A new data character is put out on B_0-B_3 each time the M_6-M_3 lines change (i.e., once every 0.75ms with a 6 MHz crystal). Data is blanked during the time the column select lines change by raising the display outputs. Output data is positive true.

LEFT ENTRY

The left entry mode is the simplest display format in that each display position in the display corresponds to a byte (or nibble) in the Display RAM. ADDRESS 0 in the RAM is the left-most display character and ADDRESS 15 is the right-most display character. Entering characters from position zero causes the display to fill from the left. The 17th character is entered back in the left-most position and filling again proceeds from there.

RIGHT ENTRY

Right entry is the method used by most electronic calculators. The first entry is placed in the right-most display character. The next entry is also placed in the right-most character after the display is shifted left one character. The left-most character is shifted off the end and is lost.

DISPLAY RAM ADDRESS	1	2		14	15	0
1ST ENTRY						1
DISPLAY RAM ADDRESS	2	3		15	0	1
2ND ENTRY					1	2
DISPLAY RAM ADDRESS	3	4		0	1	2
3RD ENTRY				1	2	3
DISPLAY RAM ADDRESS	0	1		13	14	15
16TH ENTRY	1	2		14	15	16
DISPLAY RAM ADDRESS	1	2		14	15	0
17TH ENTRY	2	3		15	16	17
DISPLAY RAM ADDRESS	2	3		15	0	1
18TH ENTRY	3	4		16	17	18

Note that now the display position and register address do not correspond. Consequently, entering a character to an arbitrary position in the Auto Increment mode may have unexpected results. Entry starting at Display RAM ADDRESS 0 with sequential entry is recommended. A Clear Display command should be given before display data is entered if the number of data characters is not equal to 16 (or 8) in this mode.

AUTO INCREMENT

In the Left Entry mode, Auto Incrementing causes the address where the CPU will next write to be incremented by one and the character appears in the next location. With non-Auto Incrementing the entry is both to the same RAM address and display position. Entry to an arbitrary address in the Left Entry—Auto Increment mode has no undesirable side effects and the result is predictable:

DISPLAY RAM ADDRESS	0	1	2	3	4	5	6	7
1ST ENTRY	1							
DISPLAY RAM ADDRESS	0	1	2	3	4	5	6	7
2ND ENTRY	1	2						
DISPLAY RAM ADDRESS	0	1	2	3	4	5	6	7
COMMAND 10010101	1	2						

ENTER NEXT AT LOCATION 5 AUTO INCREMENT

DISPLAY RAM ADDRESS	0	1	2	3	4	5	6	7
3RD ENTRY	1	2				3		
DISPLAY RAM ADDRESS	0	1	2	3	4	5	6	7
4TH ENTRY	1	2				3	4	

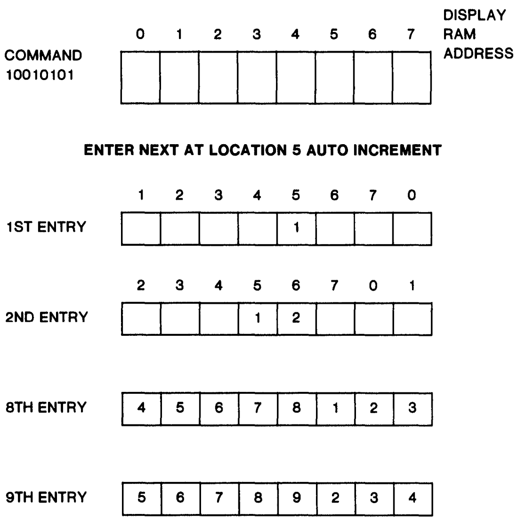
In the Right Entry mode, Auto Incrementing and non-Incrementing have the same effect as in the Left Entry except that the address sequence is interrupted.

DISPLAY RAM ADDRESS	1	2	3	4	5	6	7	0
1ST ENTRY								1
DISPLAY RAM ADDRESS	2	3	4	5	6	7	0	1
2ND ENTRY							1	2
DISPLAY RAM ADDRESS	2	3	4	5	6	7	0	1
COMMAND 10010101							1	2

ENTER NEXT AT LOCATION 5 AUTO INCREMENT

DISPLAY RAM ADDRESS	3	4	5	6	7	0	1	2
3RD ENTRY			3			1	2	
DISPLAY RAM ADDRESS	4	5	6	7	0	1	2	3
4TH ENTRY		3	4		1	2		

Starting at an arbitrary location operates as shown below.



Entry appears to be from the initial entry point.

CHAPTER 7
DATA SHEETS

8041A/8641A/8741A UNIVERSAL PERIPHERAL INTERFACE 8-BIT MICROCOMPUTER

- **8-Bit CPU plus ROM, RAM, I/O, Timer and Clock in a Single Package**
- **One 8-Bit Status and Two Data Registers for Asynchronous Slave-to-Master Interface**
- **DMA, Interrupt, or Polled Operation Supported**
- **1024 × 8 ROM/EPROM, 64 × 8 RAM, 8-Bit Timer/Counter, 18 Programmable I/O Pins**
- **Fully Compatible with MCS-48™, MCS-80™, MCS-85™, and MCS-86™ Microprocessor Families**
- **Interchangeable ROM and EPROM Versions**
- **3.6 MHz 8741A-8 Available**
- **Expandable I/O**
- **RAM Power-Down Capability**
- **Over 90 Instructions: 70% Single Byte**
- **Single 5V Supply**

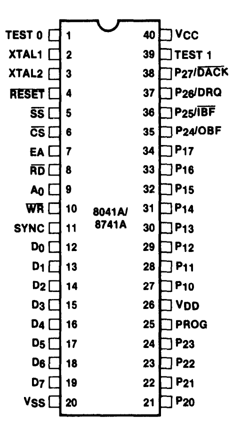
The Intel® 8041A/8741A is a general purpose, programmable interface device designed for use with a variety of 8-bit microprocessor systems. It contains a low cost microcomputer with program memory, data memory, 8-bit CPU, I/O ports, timer/counter, and clock in a single 40-pin package. Interface registers are included to enable the UPI device to function as a peripheral controller in MCS-48™, MCS-80™, MCS-85™, MCS-86™, and other 8-bit systems.

The UPI-41A™ has 1K words of program memory and 64 words of data memory on-chip. To allow full user flexibility the program memory is available as ROM in the 8041A version or as UV-erasable EPROM in the 8741A version. The 8741A and the 8041A are fully pin compatible for easy transition from prototype to production level designs. The 8641A is a one-time programmable (at the factory) 8741A which can be ordered as the first 25 pieces of a new 8041A order. The substitution of 8641A's for 8041A's allows for very fast turnaround for initial code verification and evaluation results.

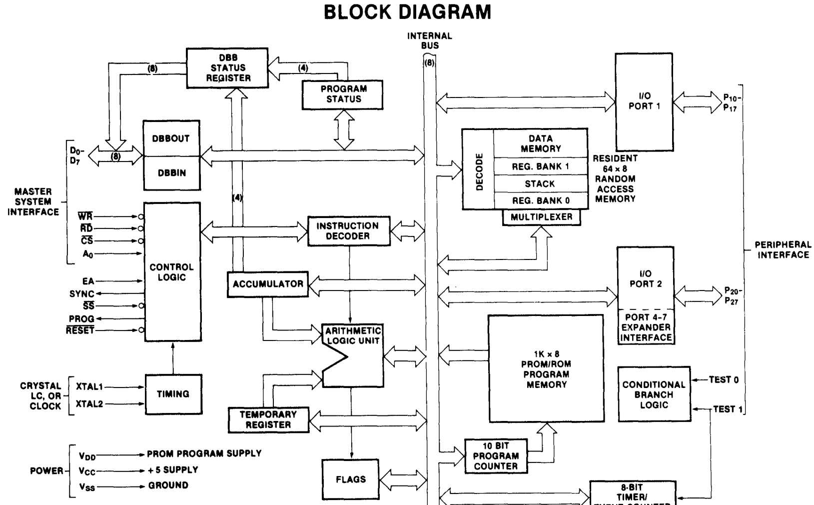
The device has two 8-bit, TTL compatible I/O ports and two test inputs. Individual port lines can function as either inputs or outputs under software control. I/O can be expanded with the 8243 device which is directly compatible and has 16 I/O lines. An 8-bit programmable timer/counter is included in the UPI device for generating timing sequences or counting external inputs. Additional UPI features include: single 5V supply, low power standby mode (in the 8041A), single-step mode for debug (in the 8741A), and dual working register banks.

Because it's a complete microcomputer, the UPI provides more flexibility for the designer than conventional LSI interface devices. It is designed to be an efficient controller as well as an arithmetic processor. Applications include keyboard scanning, printer control, display multiplexing and similar functions which involve interfacing peripheral devices to microprocessor systems.

PIN CONFIGURATION



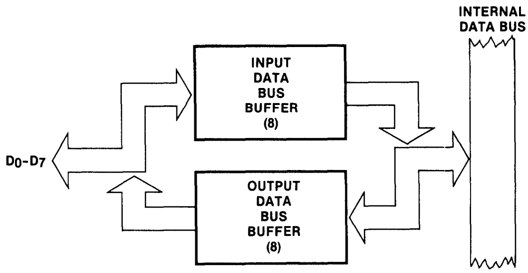
BLOCK DIAGRAM



00188A

UPI-41A™ FEATURES AND ENHANCEMENTS

1. Two Data Bus Buffers, one for input and one for output. This allows a much cleaner Master/Slave protocol.



2. 8 Bits of Status

ST_7	ST_6	ST_5	ST_4	F_1	F_0	IBF	OBF
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

ST_4-ST_7 are user definable status bits. These bits are defined by the "MOV STS, A" single byte, single cycle instruction. Bits 4-7 of the accumulator are moved to bits 4-7 of the status register. Bits 0-3 of the status register are not affected.

MOV STS, A Op Code: 90H

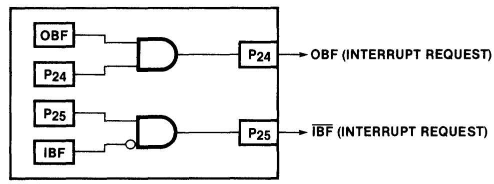
1	0	0	1	0	0	0	0
D_7							D_0

3. $\overline{RD}$ and $\overline{WR}$ are edge triggered. IBF, OBF, F_1 and INT change internally after the trailing edge of $\overline{RD}$ or $\overline{WR}$.

4. P_{24} and P_{25} are port pins or Buffer Flag pins which can be used to interrupt a master processor. These pins default to port pins on Reset.

If the "EN FLAGS" instruction has been executed, P_{24} becomes the OBF (Output Buffer Full) pin. A "1" written to P_{24} enables the OBF pin (the pin outputs the OBF Status Bit). A "0" written to P_{24} disables the OBF pin (the pin remains low). This pin can be used to indicate that valid data is available from the UPI-41A (in Output Data Bus Buffer).

If "EN FLAGS" has been executed, P_{25} becomes the $\overline{IBF}$ (Input Buffer Full) pin. A "1" written to P_{25} enables the $\overline{IBF}$ pin (the pin outputs the inverse of the IBF Status Bit). A "0" written to P_{25} disables the $\overline{IBF}$ pin (the pin remains low). This pin can be used to indicate that the UPI-41A is ready for data.



DATA BUS BUFFER INTERRUPT CAPABILITY

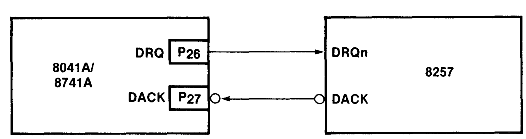
EN FLAGS Op Code: 0F5H

1	1	1	1	0	1	0	1
D_7							D_0

5. P_{26} and P_{27} are port pins or DMA handshake pins for use with a DMA controller. These pins default to port pins on Reset.

If the "EN DMA" instruction has been executed, P_{26} becomes the DRQ (DMA ReQuest) pin. A "1" written to P_{26} causes a DMA request (DRQ is activated). DRQ is deactivated by DACK·RD, DACK·WR, or execution of the "EN DMA" instruction.

If "EN DMA" has been executed, P_{27} becomes the $\overline{DACK}$ (DMA ACKnowledge) pin. This pin acts as a chip select input for the Data Bus Buffer registers during DMA transfers.



DMA HANDSHAKE CAPABILITY

EN DMA Op Code: 0E5H

1	1	1	0	0	1	0	1
D_1							D_0

PIN DESCRIPTION

Signal	Description
$D_0 - D_7$ (BUS)	Three-state, bidirectional DATA BUS BUFFER lines used to interface the UPI-41A to an 8-bit master system data bus.
P_{10}-P_{17}	8-bit, PORT 1 quasi-bidirectional I/O lines.
P_{20}-P_{27}	8-bit, PORT 2 quasi-bidirectional I/O lines. The lower 4 bits (P_{20}-P_{23}) interface directly to the 8243 I/O expander device and contain address and data information during PORT 4-7 access. The upper 4 bits (P_{24}-P_{27}) can be programmed to provide Interrupt Request and DMA Handshake capability. Software control can configure P_{24} as OBF (Output Buffer Full), P_{25} as $\overline{IBF}$ (Input Buffer Full), P_{26} as DRQ (DMA Request), and P_{27} as $\overline{DACK}$ (DMA ACKnowledge).
$\overline{WR}$	I/O write input which enables the master CPU to write data and command words to the UPI-41A INPUT DATA BUS BUFFER.
$\overline{RD}$	I/O read input which enables the master CPU to read data and status words from the OUTPUT DATA BUS BUFFER or status register.
$\overline{CS}$	Chip select input used to select one UPI-41A out of several connected to a common data bus.
A_0	Address input used by the master processor to indicate whether byte transfer is data or command.
TEST 0, TEST 1	Input pins which can be directly tested using conditional branch instructions. T_1 also functions as the event timer input (under software control). T_0 is used during PROM programming and verification in the 8741A.
XTAL1, XTAL2	Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
SYNC	Output signal which occurs once per UPI-41A instruction cycle. SYNC can be used as a strobe for external circuitry; it is also used to synchronize single step operation.
EA	External access input which allows emulation, testing and PROM/ROM verification.
PROG	Multifunction pin used as the program pulse input during PROM programming. During I/O expander access the PROG pin acts as an address/data strobe to the 8243.
$\overline{RESET}$	Input used to reset status flip-flops and to set the program counter to zero. $\overline{RESET}$ is also used during PROM programming and verification.
$\overline{SS}$	Single step input used in the 8741A in conjunction with the SYNC output to step the program through each instruction.
V_{CC}	+5V main power supply pin.
V_{DD}	+5V during normal operation. +25V during programming operation. Low power standby pin in ROM version.
V_{SS}	Circuit ground potential.

UPI™ INSTRUCTION SET

Mnemonic	Description	Bytes	Cycles
ACCUMULATOR			
ADD A,Rr	Add register to A	1	1
ADD A,@Rr	Add data memory to A	1	1
ADD A,#data	Add immediate to A	2	2
ADDC A,Rr	Add register to A with carry	1	1
ADDC A,@Rr	Add data memory to A with carry	1	1
ADDC A,#data	Add immed. to A with carry	2	2
ANL A,Rr	AND register to A	1	1
ANL A,@Rr	AND data memory to A	1	1
ANL A,#data	AND immediate to A	2	2
ORL A,Rr	OR register to A	1	1
ORL A,@Rr	OR data memory to A	1	1
ORL A,#data	OR immediate to A	2	2
XRL A,Rr	Exclusive OR register to A	1	1
XRL A,@Rr	Exclusive OR data memory to A	1	1
XRL A,#data	Exclusive OR immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
INPUT/OUTPUT			
IN A,Pp	Input port to A	1	2
OUTL Pp,A	Output A to port	1	2
ANL Pp,#data	AND immediate to port	2	2
ORL Pp,#data	OR immediate to port	2	2
IN A,DBB	Input DBB to A, clear IBF	1	1
OUT DBB,A	Output A to DBB, set OBF	1	1
MOV STS,A	A_4–A_7 to Bits 4–7 of Status	1	1
MOVD A,Pp	Input Expander port to A	1	2
MOVD Pp,A	Output A to Expander port	1	2
ANLD Pp,A	AND A to Expander port	1	2
ORLD Pp,A	OR A to Expander port	1	2
DATA MOVES			
MOV A,Rr	Move register to A	1	1
MOV A,@Rr	Move data memory to A	1	1
MOV A,#data	Move immediate to A	2	2
MOV Rr,A	Move A to register	1	1
MOV @Rr,A	Move A to data memory	1	1
MOV Rr,#data	Move immediate to register	2	2
MOV @Rr,#data	Move immediate to data memory	2	2
MOV A,PSW	Move PSW to A	1	1
MOV PSW,A	Move A to PSW	1	1
XCH A,Rr	Exchange A and register	1	1
XCH A,@Rr	Exchange A and data memory	1	1
XCHD A,@Rr	Exchange digit of A and register	1	1
MOVP A,@A	Move to A from current page	1	2
MOVP3, A,@A	Move to A from page 3	1	2
TIMER/COUNTER			
MOV A,T	Read Timer/Counter	1	1
MOV T,A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1

00188A

Mnemonic	Description	Bytes	Cycles
CONTROL			
EN DMA	Enable DMA Handshake Lines	1	1
EN I	Enable IBF Interrupt	1	1
DIS I	Disable IBF Interrupt	1	1
EN FLAGS	Enable Master Interrupts	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1
REGISTERS			
INC Rr	Increment register	1	1
INC @Rr	Increment data memory	1	1
DEC Rr	Decrement register	1	1
SUBROUTINE			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
FLAGS			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear F1 Flag	1	1
CPL F1	Complement F1 Flag	1	1
BRANCH			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ Rr, addr	Decrement register and jump	2	2
JC addr	Jump on Carry = 1	2	2
JNC addr	Jump on Carry = 0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0 = 1	2	2
JNT0 addr	Jump on T0 = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 Flag = 1	2	2
JF1 addr	Jump on F1 Flag = 1	2	2
JTF addr	Jump on Timer Flag = 1, Clear Flag	2	2
JNIBF addr	Jump on IBF Flag = 0	2	2
JOBF addr	Jump on OBF Flag = 1	2	2
JBb addr	Jump on Accumulator Bit	2	2

APPLICATIONS

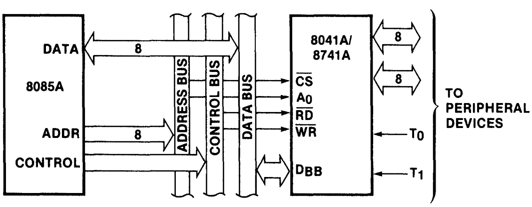


Figure 1. 8085A-8041A Interface

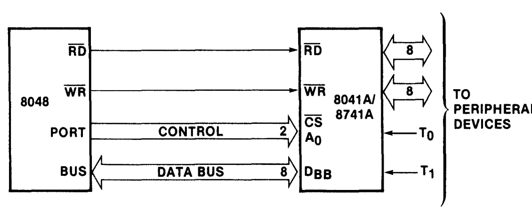


Figure 2. 8048-8041A Interface

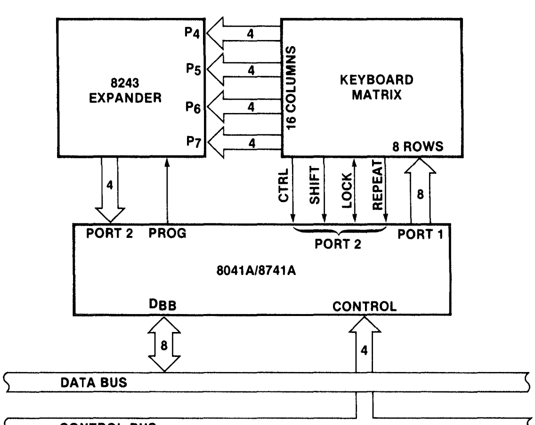


Figure 3. 8041A-8243 Keyboad Scanner

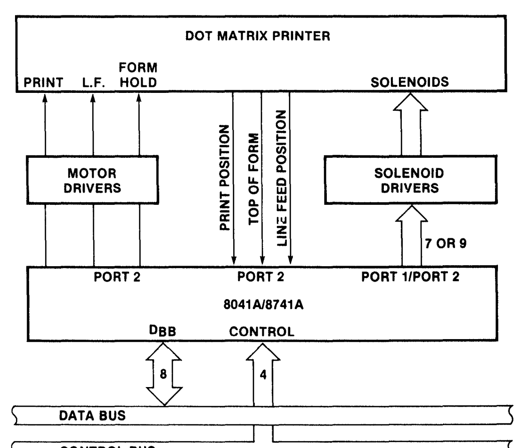


Figure 4. 8041A Matrix Printer Interface

00188A

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ....0°C to 70°C
Storage Temperature .... −65°C to +150°C
Voltage on Any Pin With Respect
to Ground ....0.5V to +7V
Power Dissipation ....1.5 Watt

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

D.C. AND OPERATING CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{SS} = 0V$, 8041A: $V_{CC} = V_{DD} = +5V \pm 10\%$, 8741A: $V_{CC} = V_{DD} = +5V \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except XTAL1, XTAL2, $\overline{RESET}$)	−0.5	0.8	V	
V_{IL1}	Input Low Voltage (XTAL1, XTAL2, $\overline{RESET}$)	−0.5	0.6	V	
V_{IH}	Input High Voltage (Except XTAL1, XTAL2, $\overline{RESET}$)	2.2	V_{CC}		
V_{IH1}	Input High Voltage (XTAL1, XTAL2, $\overline{RESET}$)	3.8	V_{CC}	V	
V_{OL}	Output Low Voltage (D_0-D_7)		0.45	V	$I_{OL} = 2.0$ mA
V_{OL1}	Output Low Voltage ($P_{10}P_{17}$, $P_{20}P_{27}$, Sync)		0.45	V	$I_{OL} = 1.6$ mA
V_{OL2}	Output Low Voltage (Prog)		0.45	V	$I_{OL} = 1.0$ mA
V_{OH}	Output High Voltage (D_0-D_7)	2.4		V	$I_{OH} = -400\ \mu A$
V_{OH1}	Output High Voltage (All Other Outputs)	2.4		V	$I_{OH} = -50\ \mu A$
I_{IL}	Input Leakage Current (T_0, T_1, $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_0, EA)		±10	μA	$V_{SS} \le V_{IN} \le V_{CC}$
I_{OZ}	Output Leakage Current (D_0-D_7, High Z State)		±10	μA	$V_{SS} + 0.45 \le V_{IN} \le V_{CC}$
I_{LI}	Low Input Load Current ($P_{10}P_{17}$, $P_{20}P_{27}$)		0.5	mA	$V_{IL} = 0.8V$
I_{LI1}	Low Input Load Current ($\overline{RESET}$, $\overline{SS}$)		0.2	mA	$V_{IL} = 0.8V$
I_{DD}	V_{DD} Supply Current		15	mA	Typical = 5 mA
$I_{CC} + I_{DD}$	Total Supply Current		125	mA	Typical = 60 mA

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{SS} = 0V$, 8041A: $V_{CC} = V_{DD} = +5V \pm 10\%$, 8741A: $V_{CC} = V_{DD} = +5V \pm 5\%$

DBB READ

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}$↓	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}$↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	$C_L = 150$ pF
t_{RD}	$\overline{RD}$↓ to Data Out Delay		225	ns	$C_L = 150$ pF
t_{DF}	$\overline{RD}$↑ to Data Float Delay		100	ns	
t_{CY}	Cycle Time (Except 8741A-8)	2.5	15	μs	6.0 MHz XTAL
t_{CY}	Cycle Time (8741A-8)	4.17	15	μs	3.6 MHz XTAL

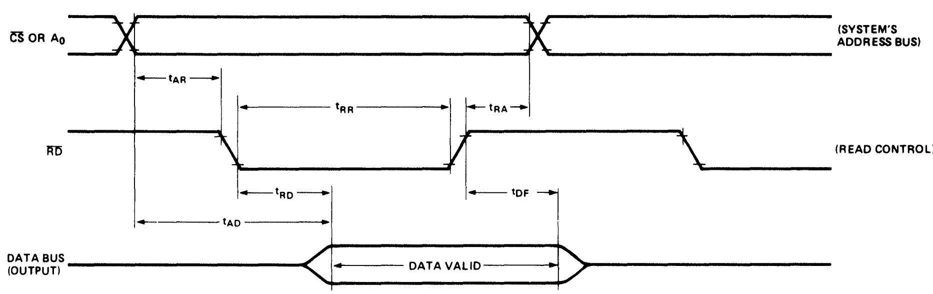
DBB WRITE

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	$\overline{CS}$, A_0 Setup to $\overline{WR}$↓	0		ns	
t_{WA}	$\overline{CS}$, A_0 Hold After $\overline{WR}$↑	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$↑	150		ns	
t_{WD}	Data Hold After $\overline{WR}$↑	0		ns	

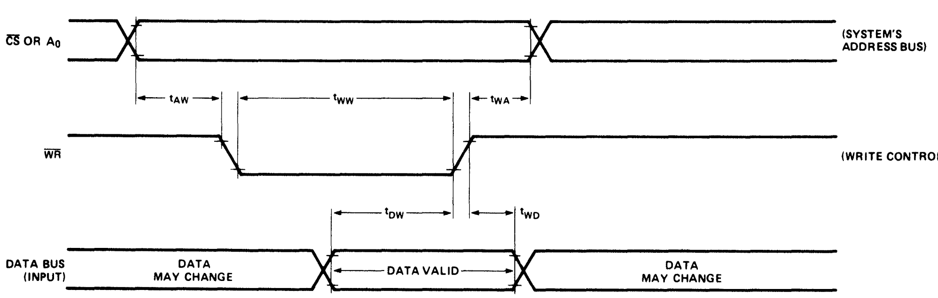
INPUT AND OUTPUT WAVEFORMS FOR A.C. TESTS

WAVEFORMS

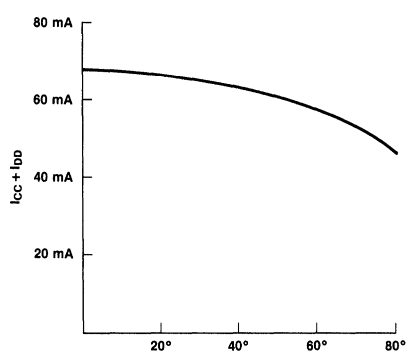
1. READ OPERATION—DATA BUS BUFFER REGISTER.



2. WRITE OPERATION—DATA BUS BUFFER REGISTER.



TYPICAL 8041/8741A CURRENT



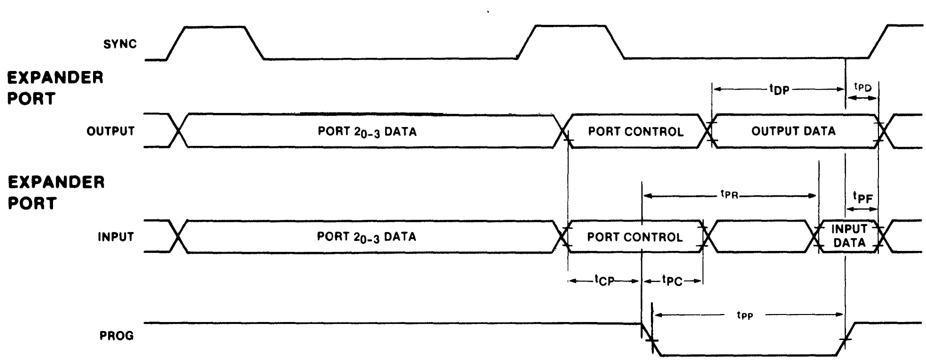
00188A

A.C. CHARACTERISTICS—PORT 2

$T_A = 0°C$ to $70°C$, 8041A: $V_{CC} = +5V \pm 10\%$, 8741A: $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CP}	Port Control Setup Before Falling Edge of PROG	110		ns	
t_{PC}	Port Control Hold After Falling Edge of PROG	100		ns	
t_{PR}	PROG to Time P2 Input Must Be Valid		810	ns	
t_{PF}	Input Data Hold Time	0	150	ns	
t_{DP}	Output Data Setup Time	250		ns	
t_{PD}	Output Data Hold Time	65		ns	
t_{PP}	PROG Pulse Width	1200		ns	

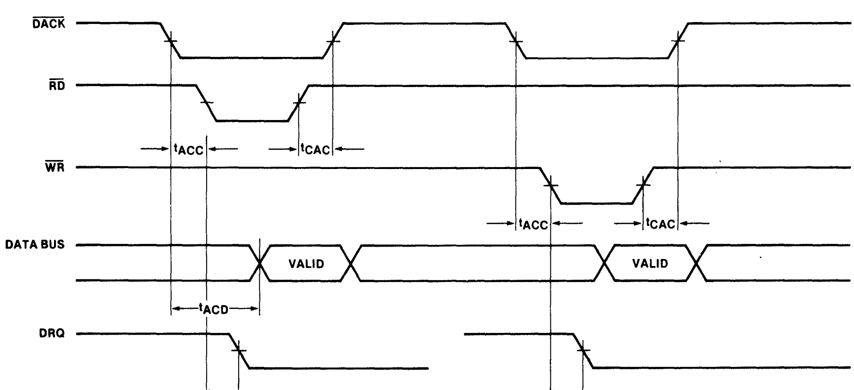
PORT 2 TIMING



A.C. CHARACTERISTICS—DMA

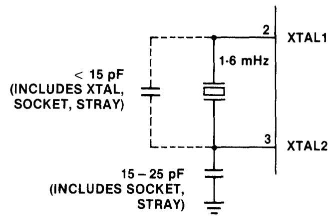
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{ACC}	$\overline{DACK}$ to $\overline{WR}$ or $\overline{RD}$	0		ns	
t_{CAC}	$\overline{RD}$ or $\overline{WR}$ to $\overline{DACK}$	0		ns	
t_{ACD}	$\overline{DACK}$ to Data Valid		225	ns	$C_L = 150$ pF
t_{CRQ}	$\overline{RD}$ or $\overline{WR}$ to DRQ Cleared		200	ns	

WAVEFORMS—DMA



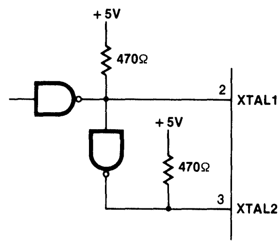
00188A

CRYSTAL OSCILLATOR MODE



CRYSTAL SERIES RESISTANCE SHOULD BE <75Ω AT 6 MHz; <180Ω AT 3.6 MHz.

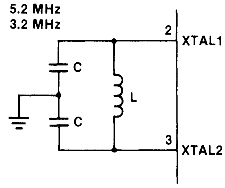
DRIVING FROM EXTERNAL SOURCE



BOTH XTAL1 AND XTAL2 SHOULD BE DRIVEN.
RESISTORS TO V_{CC} ARE NEEDED TO ENSURE $V_{IH} = 3.8V$
IF TTL CIRCUITRY IS USED.

LC OSCILLATOR MODE

L	C	NOMINAL f
45 μH	20 pF	5.2 MHz
120 μH	20 pF	3.2 MHz



$$f \approx \frac{1}{2\pi\sqrt{LC'}}$$

$$C' = \frac{C + 3C_{PP}}{2}$$

$C_{PP} \approx 5 - 10$ pF PIN-TO-PIN CAPACITANCE

EACH C SHOULD BE APPROXIMATELY 20 pF, INCLUDING STRAY CAPACITANCE.

PROGRAMMING, VERIFYING, AND ERASING THE 8741A EPROM

Programming Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL 1	Clock Input (1 to 6MHz)
$\overline{\text{Reset}}$	Initialization and Address Latching
Test 0	Selection of Program or Verify Mode
EA	Activation of Program/Verify Modes
BUS	Address and Data Input Data Output During Verify
P20-1	Address Input
V_{DD}	Programming Power Supply
PROG	Program Pulse Input

WARNING:

An attempt to program a missocketed 8741A will result in severe damage to the part. An indication of a properly socketed part is the appearance of the SYNC clock output. The lack of this clock may be used to disable the programmer.

The Program/Verify sequence is:

1. A_0 = 0V, $\overline{CS}$ = 5V, EA = 5V, $\overline{\text{RESET}}$ = 0V, TEST0 = 5V, V_{DD} = 5V, clock applied or internal oscillator operating, BUS and PROG floating.
2. Insert 8741A in programming socket
3. TEST 0 = 0v (select program mode)
4. EA = 23V (activate program mode)
5. Address applied to BUS and P20-1
6. $\overline{\text{RESET}}$ = 5v (latch address)
7. Data applied to BUS
8. V_{DD} = 25v (programming power)
9. PROG = 0v followed by one 50ms pulse to 23V
10. V_{DD} = 5v
11. TEST 0 = 5v (verify mode)
12. Read and verify data on BUS
13. TEST 0 = 0v
14. $\overline{\text{RESET}}$ = 0v and repeat from step 5
15. Programmer should be at conditions of step 1 when 8741A is removed from socket.

8741A Erasure Characteristics

The erasure characteristics of the 8741A are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8741A in approximately 3 years while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 8741A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8741A window to prevent unintentional erasure.

The recommended erasure procedure for the 8741A is exposure to shortwave ultraviolet light which has a wavelength of 2537Å. The integrated dose (i.e., UV intensity x exposure time) for erasure should be a minimum of 15 w-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 μW/cm^2 power rating. The 8741A should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

A.C. TIMING SPECIFICATION FOR PROGRAMMING

$T_A = 25°C \pm 5°C$, $V_{CC} = 5V \pm 5\%$, $V_{DD} = 25V \pm 1V$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	Address Setup Time to $\overline{RESET}$ ↑	4tcy			
t_{WA}	Address Hold Time After $\overline{RESET}$ ↑	4tcy			
t_{DW}	Data in Setup Time to PROG ↑	4tcy			
t_{WD}	Data in Hold Time After PROG ↓	4tcy			
t_{PH}	$\overline{RESET}$ Hold Time to Verify	4tcy			
t_{VDDW}	V_{DD} Setup Time to PROG ↑	4tcy			
t_{VDDH}	V_{DD} Hold Time After PROG ↓	0			
t_{PW}	Program Pulse Width	50	60	mS	
t_{TW}	Test 0 Setup Time for Program Mode	4tcy			
t_{WT}	Test 0 Hold Time After Program Mode	4tcy			
t_{DO}	Test 0 to Data Out Delay		4tcy		
t_{WW}	$\overline{RESET}$ Pulse Width to Latch Address	4tcy			
t_r, t_f	V_{DD} and PROG Rise and Fall Times	0.5	2.0	μs	
t_{CY}	CPU Operation Cycle Time	5.0		μs	
t_{RE}	$\overline{RESET}$ Setup Time Before EA ↑.	4tcy			

Note: If TEST 0 is high, t_{DO} can be triggered by $\overline{RESET}$ ↑.

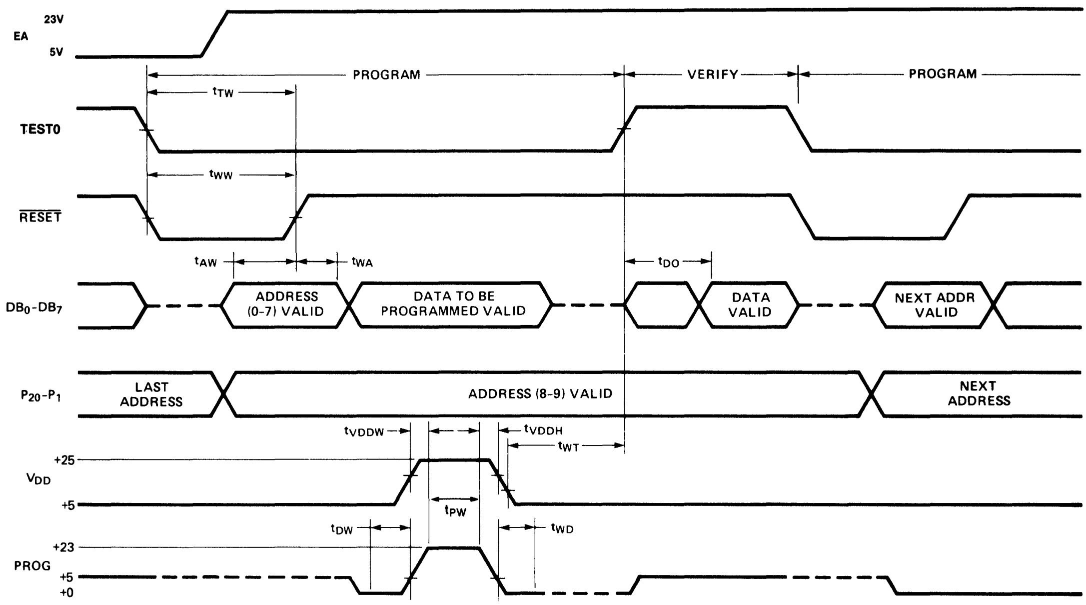
D.C. SPECIFICATION FOR PROGRAMMING

$T_A = 25°C \pm 5°C$, $V_{CC} = 5V \pm 5\%$, $V_{DD} = 25V \pm 1V$

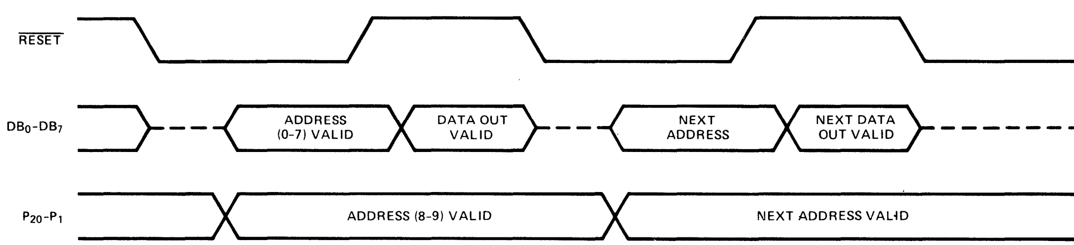
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{DOH}	V_{DD} Program Voltage High Level	24.0	26.0	V	
V_{DDL}	V_{DD} Voltage Low Level	4.75	5.25	V	
V_{PH}	PROG Program Voltage High Level	21.5	24.5	V	
V_{PL}	PROG Voltage Low Level		0.2	V	
V_{EAH}	EA Program or Verify Voltage High Level	21.5	24.5	V	
V_{EAL}	EA Voltage Low Level		5.25	V	
I_{DD}	V_{DD} High Voltage Supply Current		30.0	mA	
I_{PROG}	PROG High Voltage Supply Current		16.0	mA	
I_{EA}	EA High Voltage Supply Current		1.0	mA	

WAVEFORMS FOR PROGRAMMING

COMBINATION PROGRAM/VERIFY MODE (EPROM'S ONLY)



VERIFY MODE (ROM/EPROM)



NOTES:
1. **PROG MUST FLOAT IF EA IS LOW (i.e., ≠23V), OR IF T0 = 5V FOR THE 8741A. FOR THE 8041A PROG MUST ALWAYS FLOAT.**
2. **XTAL1 AND XTAL 2 DRIVEN BY 3.6 MHz CLOCK WILL GIVE 4.17 μsec t_{CY}. THIS IS ACCEPTABLE FOR 8741A-8 PARTS AS WELL AS STANDARD PARTS.**
3. **AO MUST BE HELD LOW (i.e., = 0V) DURING PROGRAM/VERIFY MODES.**

The 8741A EPROM can be programmed by either of two Intel products:

1. PROMPT-48 Microcomputer Design Aid, or
2. Universal PROM Programmer (UPP series) peripheral of the Intellec® Development System with a UPP-848 Personality Card.

00188A

8243
MCS-48™ INPUT/OUTPUT EXPANDER

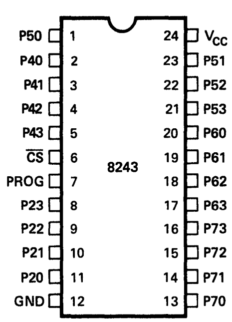
- **Low Cost**
- **Simple Interface to MCS-48™ Microcomputers**
- **Four 4-Bit I/O Ports**
- **AND and OR Directly to Ports**
- **24-Pin DIP**
- **Single 5V Supply**
- **High Output Drive**
- **Direct Extension of Resident 8048 I/O Ports**

The Intel® 8243 is an input/output expander designed specifically to provide a low cost means of I/O expansion for the MCS-48™ family of single chip microcomputers. Fabricated in 5 volts NMOS, the 8243 combines low cost, single supply voltage and high drive current capability.

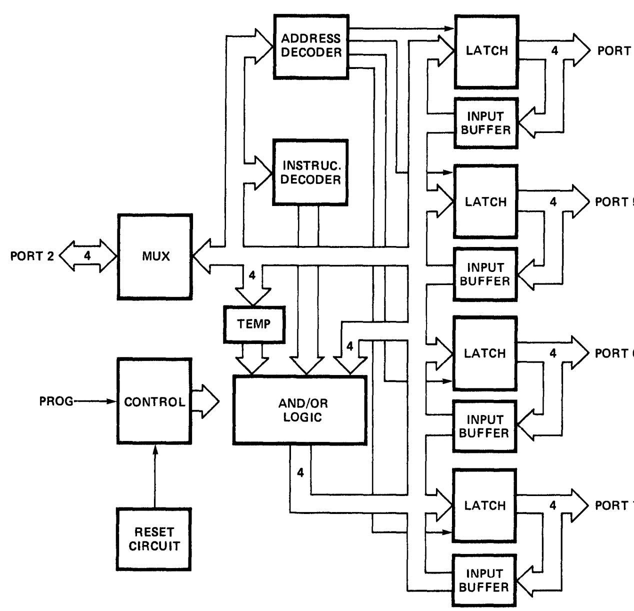
The 8243 consists of four 4-bit bidirectional static I/O ports and one 4-bit port which serves as an interface to the MCS-48 microcomputers. The 4-bit interface requires that only 4 I/O lines of the 8048 be used for I/O expansion, and also allows multiple 8243's to be added to the same bus.

The I/O ports of the 8243 serve as a direct extension of the resident I/O facilities of the MCS-48 microcomputers and are accessed by their own MOV, ANL, and ORL instructions.

PIN CONFIGURATION



BLOCK DIAGRAM



00214A

PIN DESCRIPTION

Symbol	Pin No.	Function
PROG	7	Clock Input. A high to low transistion on PROG signifies that address and control are available on P20-P23, and a low to high transition signifies that data is available on P20-23.
$\overline{CS}$	6	Chip Select Input. A high on CS inhibits any change of output or internal status.
P20-P23	11-8	Four (4) bit bi-directional port contains the address and control bits on a high to low transition of PROG. During a low to high transition contains the data for a selected output port if a write operation, or the data from a selected port before the low to high transition if a read operation.
GND	12	0 volt supply.
P40-P43 P50-P53 P60-P63 P70-P73	2-5 1,23-21 20-17 13-16	Four (4) bit bi-directional I/O ports. May be programmed to be input (during read), low impedance latched output (after write) or a tri-state (after read). Data on pins P20-23 may be directly written, ANDed or ORed with previous data.
V_{CC}	24	+5 volt supply.

FUNCTIONAL DESCRIPTION

General Operation

The 8243 contains four 4-bit I/O ports which serve as an extension of the on-chip I/O and are addressed as ports 4-7. The following operations may be performed on these ports:

- Transfer Accumulator to Port.
- Transfer Port to Accumulator.
- AND Accumulator to Port.
- OR Accumulator to Port.

All communication between the 8048 and the 8243 occurs over Port 2 (P20-P23) with timing provided by an output pulse on the PROG pin of the processor. Each transfer consists of two 4-bit nibbles:

The first containing the "op code" and port address and the second containing the actual 4-bits of data.

A high to low transition of the PROG line indicates that address is present while a low to high transition indicates the presence of data. Additional 8243's may be added to the 4-bit bus and chip selected using additional output lines from the 8048/8748/8035.

Power On Initialization

Initial application of power to the device forces input/output ports 4, 5, 6, and 7 to the tri-state and port 2 to the input mode. The PROG pin may be either high or low when power is applied. The first high to low transition of PROG causes device to exit power on mode. The power on sequence is initiated if V_{CC} drops below 1V.

P21	P20	Address Code	P23	P22	Instruction Code
0	0	Port 4	0	0	Read
0	1	Port 5	0	1	Write
1	0	Port 6	1	0	ORLD
1	1	Port 7	1	1	ANLD

Write Modes

The device has three write modes. MOVD Pi, A directly writes new data into the selected port and old data is lost. ORLD Pi,A takes new data, OR's it with the old data and then writes it to the port. ANLD Pi,A takes new data AND's it with the old data and then writes it to the port. Operation code and port address are latched from the input port 2 on the high to low transition of the PROG pin. On the low to high transition of PROG data on port 2 is transferred to the logic block of the specified output port.

After the logic manipulation is performed, the data is latched and outputed. The old data remains latched until new valid outputs are entered.

Read Mode

The device has one read mode. The operation code and port address are latched from the input port 2 on the high to low transition of the PROG pin. As soon as the read operation and port address are decoded, the appropriate outputs are tri-stated, and the input buffers switched on. The read operation is terminated by a low to high transition of the PROG pin. The port (4, 5, 6 or 7) that was selected is switched to the tri-stated mode while port 2 is returned to the input mode.

Normally, a port will be in an output (write mode) or input (read mode). If modes are changed during operation, the first read following a write should be ignored; all following reads are valid. This is to allow the external driver on the port to settle after the first read instruction removes the low impedance drive from the 8243 output. A read of any port will leave that port in a high impedance state.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias. . . . . 0°C to 70°C
Storage Temperature . . . . −65°C to +150°C
Voltage on Any Pin
With Respect to Ground . . . . −0.5V to +7V
Power Dissipation . . . . 1 Watt

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

D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ±10%

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}+0.5	V	
V_{OL1}	Output Low Voltage Ports 4-7			0.45	V	I_{OL} = 5 mA*
V_{OL2}	Output Low Voltage Port 7			1	V	I_{OL} = 20 mA
V_{OH1}	Output High Voltage Ports 4-7	2.4			V	I_{OH}= 240μA
I_{IL1}	Input Leakage Ports 4-7	-10		20	μA	V_{in} = V_{CC} to 0V
I_{IL2}	Input Leakage Port 2, CS, PROG	-10		10	μA	V_{in} = V_{CC} to 0V
V_{OL3}	Output Low Voltage Port 2			.45	V	I_{OL}= 0.6 mA
I_{CC}	V_{CC} Supply Current		10	20	mA	
V_{OH2}	Output Voltage Port 2	2.4				I_{OH} = 100μA
I_{OL}	Sum of all IO_L from 16 Outputs			80	mA	5 mA Each Pin

*See following graph for additional sink current capability

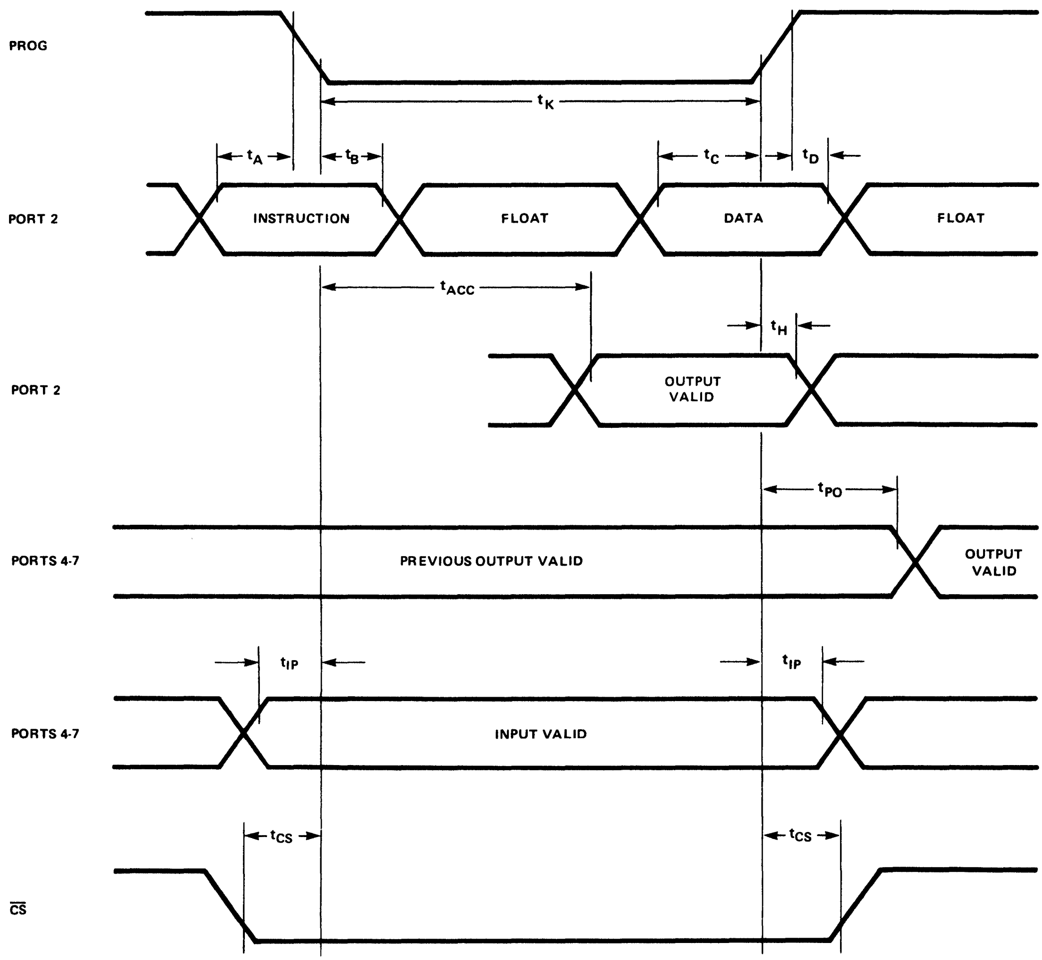
A.C. CHARACTERISTICS

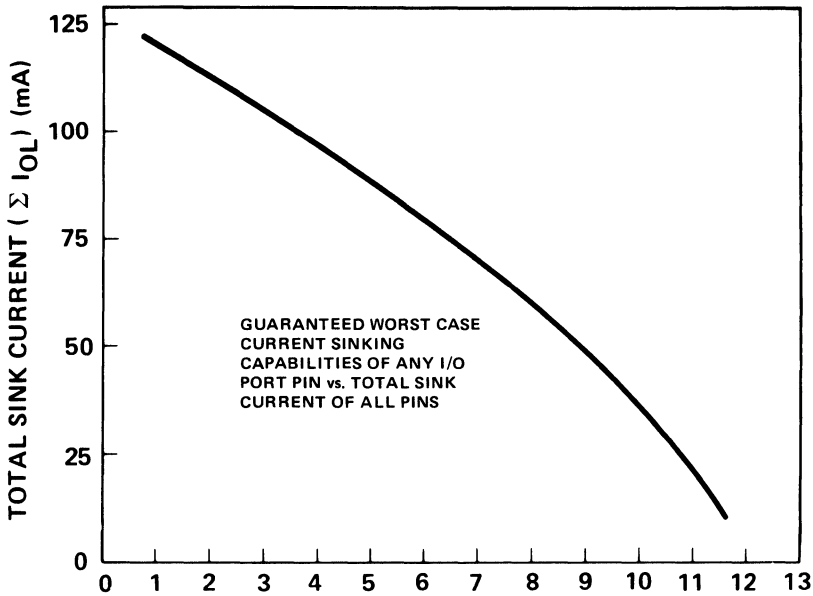
T_A = 0°C to 70°C, V_{CC} = 5V ±10%

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_A	Code Valid Before PROG	100		ns	80 pF Load
t_B	Code Valid After PROG	60		ns	20 pF Load
t_C	Data Valid Before PROG	200		ns	80 pF Load
t_D	Data Valid After PROG	20		ns	20 pF Load
t_H	Floating After PROG	0	150	ns	20 pF Load
t_K	PROG Negative Pulse Width	700		ns	
t_{CS}	CS Valid Before/After PROG	50		ns	
t_{PO}	Ports 4-7 Valid After PROG		700	ns	100 pF Load
t_{LP1}	Ports 4-7 Valid Before/After PROG	100		ns	
t_{ACC}	Port 2 Valid After PROG		650	ns	80 pF Load

00214A

WAVEFORMS


PROG
t_K
t_A
t_B
t_C
t_D
PORT 2
INSTRUCTION
FLOAT
DATA
FLOAT
t_{ACC}
t_H
PORT 2
OUTPUT VALID
t_{PO}
PORTS 4-7
PREVIOUS OUTPUT VALID
OUTPUT VALID
t_{IP}
t_{IP}
PORTS 4-7
INPUT VALID
t_{CS}
t_{CS}
$\overline{CS}$
00214A



MAXIMUM SINK CURRENT ON ANY PIN@.45V
MAXIMUM I_{OL} WORST CASE PIN (mA)

Sink Capability

The 8243 can sink 5 mA@.45V on each of its 16 I/O lines simultaneously. If, however, all lines are not sinking simultaneously or all lines are not fully loaded, the drive capability of any individual line increases as is shown by the accompanying curve.

For example, if only 5 of the 16 lines are to sink current at one time, the curve shows that each of those 5 lines is capable of sinking 9 mA@.45V (if any lines are to sink 9 mA the total I_{OL} must not exceed 45 mA or five 9 mA loads).

Example: How many pins can drive 5 TTL loads (1.6 mA) assuming remaining pins are unloaded?

$I_{OL} = 5 \times 1.6$ mA $= 8$ mA
$\varepsilon I_{OL} = 60$ mA from curve
pins = 60 mA ÷ 8 mA/pin = 7.5 = 7

In this case, 7 lines can sink 8 mA for a total of 56 mA. This leaves 4 mA sink current capability which can be divided in any way among the remaining 8 I/O lines of the 8243.

Example: This example shows how the use of the 20 mA sink capability of Port 7 affects the sinking capability of the other I/O lines.

An 8243 will drive the following loads simultaneously.

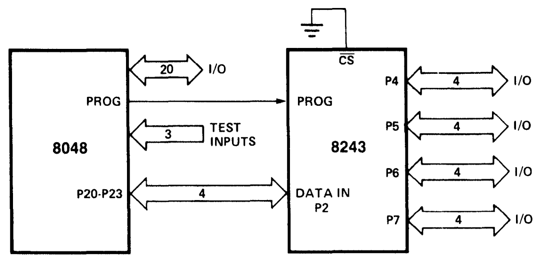
2 loads — 20 mA@1V (port 7 only)
8 loads — 4 mA@.45V
6 loads — 3.2 mA@.45V
Is this within the specified limits?

$\varepsilon I_{OL} = (2 \times 20) + (8 \times 4) + (6 \times 3.2) = 91.2$ mA.
From the curve: for $I_{OL} = 4$ mA, $\varepsilon I_{OL} \approx 93$ mA since 91.2 mA < 93 mA the loads are within specified limits.

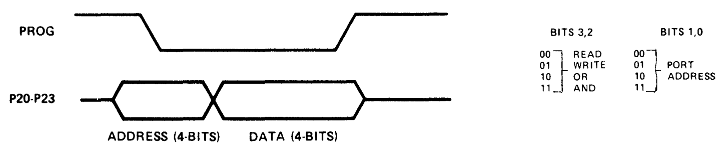
Although the 20 mA@1V loads are used in calculating εI_{OL}, it is the largest current required@.45V which determines the maximum allowable εI_{OL}.

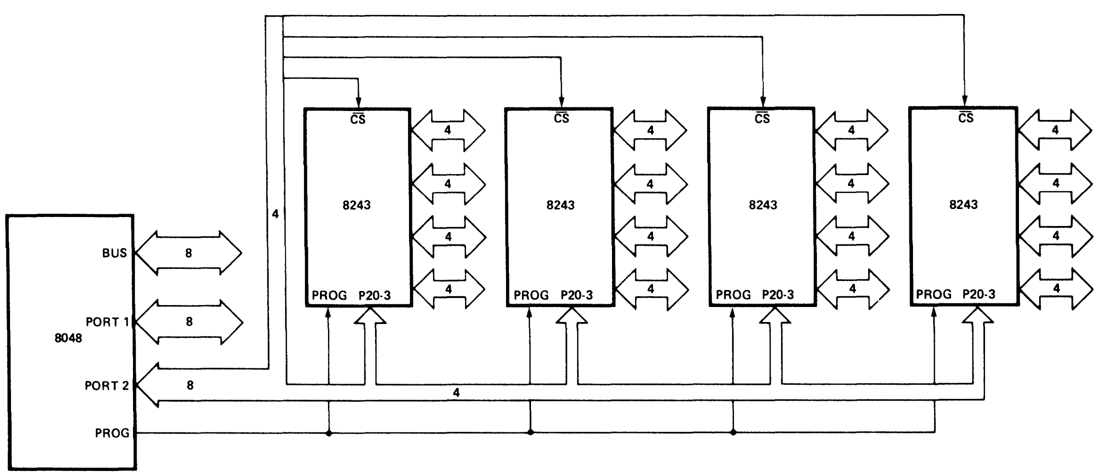
Note: A 10 to 50KΩ pullup resistor to +5V should be added to 8243 outputs when driving to 5V CMOS directly.

EXPANDER INTERFACE



OUTPUT EXPANDER TIMING





USING MULTIPLE 8243's

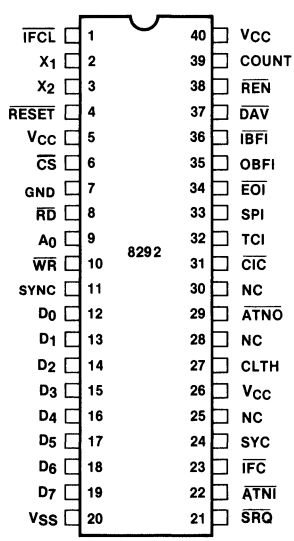
00214A

8292
GPIB CONTROLLER

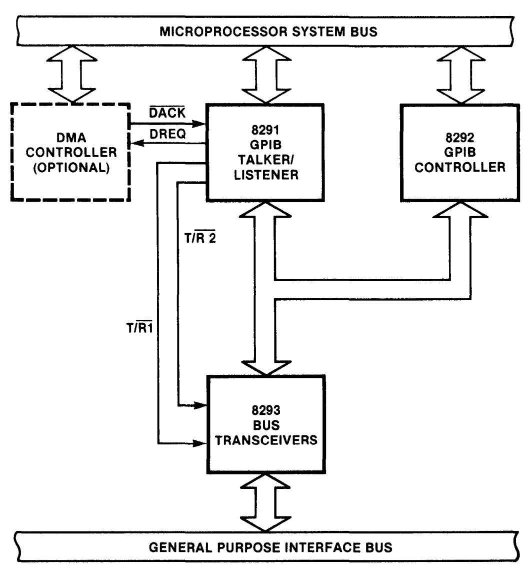
- **Complete IEEE Standard 488 Controller Function**
- **Interface Clear (IFC) Sending Capability Allows Seizure of Bus Control and/or Initialization of the Bus**
- **Responds to Service Requests (SRQ)**
- **Sends Remote Enable (REN), Allowing Instruments to Switch to Remote Control**
- **Complete Implementation of Transfer Control Protocol**
- **Synchronous Control Seizure Prevents the Destruction of Any Data Transmission in Progress**
- **Connects with the 8291 to Form a Complete IEEE Standard 488 Interface Talker/Listener/Controller**

The 8292 GPIB Controller is a microprocessor-controlled chip designed to function with the 8291 GPIB Talker/Listener to implement the full IEEE Standard 488 controller function, including transfer control protocol. The 8292 is a pre-programmed Intel® 8041A.

PIN CONFIGURATION



8291, 8292 SYSTEM DIAGRAM



00741C

PIN DESCRIPTION

Symbol	I/O	Pin No.	Function
$\overline{IFCL}$	I	1	IFC Received (latched) — The 8292 monitors the IFC Line (when not system controller) through this pin.
X_1, X_2	I	2, 3	Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
$\overline{RESET}$	I	4	Used to initialize the chip to a known state during power on.
$\overline{CS}$	I	6	Chip Select Input — Used to select the 8292 from other devices on the common data bus.
$\overline{RD}$	I	8	I/O write input which allows the master CPU to read from the 8292.
A_0	I	9	Address Line — Used to select between the data bus and the status register during read operations and to distinguish between data and commands written into the 8292 during write operations.
$\overline{WR}$	I	10	I/O read input which allows the master CPU to write to the 8292.
SYNC	O	11	8041A instruction cycle synchronization signal; it is an output clock with a frequency of XTAL ÷ 15.
D_0–D_7	I/O	12–19	8 bidirectional lines used for communication between the central processor and the 8292's data bus buffers and status register.
V_{SS}	P.S.	7, 20	Circuit ground potential.
$\overline{SRQ}$	I	21	Service Request — One of the IEEE control lines. Sampled by the 8292 when it is controller in charge. If true, SPI interrupt to the master will be generated.
$\overline{ATNI}$	I	22	Attention In — Used by the 8292 to monitor the GPIB ATN control line. It is used during the transfer control procedure.
$\overline{IFC}$	I/O	23	Interface Clear — One of the GPIB management lines, as defined by IEEE Std. 488-1978, places all devices in a known quiescent state.
SYC	I	24	System Controller — Monitors the system controller switch.
CLTH	O	27	CLEAR LATCH Output — Used to clear the $\overline{IFCR}$ latch after being recognized by the 8292. Usually low (except after hardware Reset), it will be pulsed high when $\overline{IFCR}$ is recognized by the 8292.
$\overline{ATNO}$	O	29	Attention Out — Controls the ATN control line of the bus through external logic for tcs and tca procedures. (ATN is a GPIB control line, as defined by IEEE Std. 488-1978.)

Symbol	I/O	Pin No.	Function
V_{CC}	P.S.	5, 26, 40	+5V supply input. ±10%.
COUNT	I	39	Count Input — When enabled by the proper command the internal counter will count external events through this pin. High to low transition will increment the internal counter by one. The pin is sampled once per three internal instruction cycles (7.5μsec sample period when using 6 MHz XTAL). It can be used for byte counting when connected to NDAC, or for block counting when connected to the EOI.
$\overline{REN}$	O	38	The Remote Enable bus signal selects remote or local control of the device on the bus. A GPIB bus management line, as defined by IEEE Std. 488-1978.
$\overline{DAV}$	I/O	37	DAV Handshake Line — Used during parallel poll to force the 8291 to accept the parallel poll status bits. It is also used during the tcs procedure.
$\overline{IBFI}$	O	36	Input Buffer Not Full — Used to interrupt the central processor while the input buffer of the 8292 is empty. This feature is enabled and disabled by the interrupt mask register.
OBFI	O	35	Output Buffer Full — Used as an interrupt to the central processor while the output buffer of the 8292 is full. The feature can be enabled and disabled by the interrupt mask register.
$\overline{EOI2}$	I/O	34	End Or Identify — One of the GPIB management lines, as defined by IEEE Std. 488-1978. Used with ATN as Identify Message during parallel poll.
SPI	O	33	Special Interrupt — Used as an interrupt on events not initiated by the central processor.
TCI	O	32	Task Complete Interrupt — Interrupt to the control processor used to indicate that the task requested was completed by the 8292 and the information requested is ready in the data bus buffer.
CIC	O	31	Controller In Charge — Controls the S/R input of the SRQ bus transceiver. It can also be used to indicate that the 8292 is in charge of the GPIB bus.

GENERAL DESCRIPTION

The 8292 is an Intel 8041A which has been programmed as a GPIB Controller interface element. It is used with the 8291 GPIB Talker/Listener and two 8293 GPIB Transceivers to form a complete IEEE-488 Bus Interface for a microprocessor. The electrical interface is performed by the transceivers, data transfer is done by the talker/listener, and control of the bus is done by the 8292. Figure 1 is a typical controller interface using Intel's GPIB peripherals.

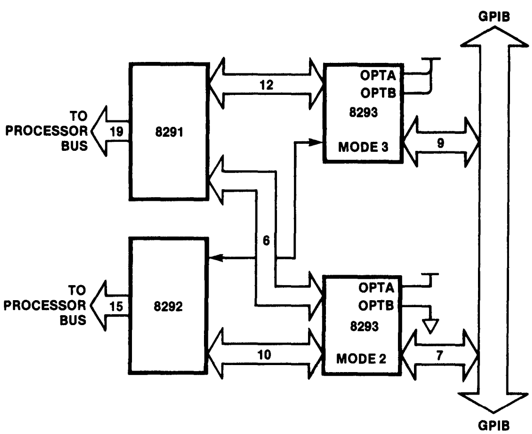


Figure 1. Talker/Listener/Controller Configuration

The internal RAM in the 8041A is used as a special purpose register bank for the 8292. Most of these registers (except for the interrupt flag) can be accessed through commands to the 8292. Table 1 identifies the registers used by the 8292 and how they are accessed.

Interrupt Status Register

D_7							D_0
SYC	ERR	SRQ	EV	X	IFCR	IBF	OBF

The 8292 can be configured to interrupt the microprocessor on one of several conditions. Upon receipt of the interrupt the microprocessor must read the 8292 interrupt status register to determine which event caused the interrupt, and then the appropriate subroutine can be performed. The interrupt status register is read with A_0 high. With the exception of OBF and IBF, these interrupts are enabled or disabled by the SPI interrupt mask. OBF and IBF have their own bits in the interrupt mask (OBFI and $\overline{\text{IBFI}}$).

OBF Output Buffer Full. A byte is waiting to be read by the microprocessor. This flag is cleared when the output data bus buffer is read.

IBF Input Buffer Full. The byte previously written by the microprocessor has not been read yet by the 8292. If another byte is written to the 8292 before this flag clears, data will be lost. IBF is cleared when the 8292 reads the data byte.

IFCR Interface Clear Received. The GPIB system controller has set IFC. The 8292 has become idle and is no longer in charge of the bus. The flag is cleared when the IACK command is issued.

EV Event Counter Interrupt. The requested number of blocks or data bytes has been transferred. The EV interrupt flag is cleared by the IACK command.

SRQ Service Request. Notifies the 8292 that a service request (SRQ) message has been received. It is cleared by the IACK command.

ERR Error occurred. The type of error can be determined by reading the error status register. This interrupt flag is cleared by the IACK command.

SYC System Controller Switch Change. Notifies the processor that the state of the system controller switch has changed. The actual state is contained in the GPIB Status Register. This flag is cleared by the IACK command.

TABLE 1. 8292 REGISTERS.

READ FROM 8292

Register	D_7							D_0	A_0
INTERRUPT STATUS	SYC	ERR	SRQ	EV	X	IFCR	IBF	OBF	1
ERROR FLAG	X	X	USER	X	X	$TOUT_3$	$TOUT_2$	$TOUT_1$	0*
CONTROLLER STATUS	CSBS	CA	X	X	SYCS	IFC	REN	SRQ	0*
GPIB (BUS) STATUS	REN	DAV	EOI	X	SYC	IFC	ANTI	SRQ	0*
EVENT COUNTER STATUS	D	D	D	D	D	D	D	D	0*
TIME OUT STATUS	D	D	D	D	D	D	D	D	0*

WRITE TO 8292

Register	D_7							D_0	A_0
INTERRUPT MASK	1	SPI	TCI	SYC	OBFI	IBFI	0	SRQ	0
ERROR MASK	0	0	USER	0	0	$TOUT_3$	$TOUT_2$	$TOUT_1$	0
COMMAND FIELD	1	1	1	OP	C	C	C	C	1
EVENT COUNTER	D	D	D	D	D	D	D	D	0*
TIME OUT	D	D	D	D	D	D	D	D	0*

Note: These registers are accessed by a special utility command, see page 6.

Interrupt Mask Register

1	SPI	TCI	SYC	OBFI	$\overline{\text{IBFI}}$	0	SRQ

D_7 D_0

The Interrupt Mask Register is used to enable features and to mask the SPI and TCI interrupts. The flags in the Interrupt Status Register will be active even when masked out. The Interrupt Mask Register is written when A_0 is low and reset by the RINM command. When the register is read, D_1 and D_7 are undefined. An intertupt is enabled by setting the corresponding register bit.

SRQ Enable interrupts on SRQ received.

$\overline{\text{IBFI}}$ Enable interrupts on input buffer empty.

OBFI Enable interrupts on output buffer full.

SYC Enable interrupts on a change in the system controller switch.

TCI Enable interrupts on the task completed.

SPI Enable interrupts on special events.

NOTE: The event counter is enabled by the GSEC command, the error interrupt is enabled by the error mask register, and IFC cannot be masked (it will always cause an interrupt).

Controller Status Register

CSBS	CA	X	X	SYCS	IFC	REN	SRQ

D_7 D_0

The Controller Status **Register** is used to determine the status of the controller function. This register is accessed by the RCST command.

SRQ Service Request line active (CSRS).

REN Sending Remote Enable.

IFC Sending or receiving interface clear.

SYCS System Controller Switch Status (SACS).

CA Controller Active (CACS + CAWS + CSWS).

CSBS Controller Stand-by State (CSBS, CA) = (0,0) — Controller Idle

GPIB Bus Status Register

REN	DAV	EOI	X	SYC	IFC	ATNI	SRQ

D_7 D_0

This register contains GPIB bus status information. It can be used by the microprocessor to monitor and manage the bus. The GPIB Bus Register can be read using the RBST command.

Each of these status bits reflect the current status of the corresponding pin on the 8292.

SRQ Service Request

ATNI Attention In

IFC Interface Clear

SYC System Controller Switch

EOI End or Identify

DAV Data Valid

REN Remote Enable

Event Counter Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

The Event Counter Register contains the initial value for the event counter. The counter can count pulses on pin 39 of the 8292 (COUNT). It can be connected to EOI or NDAC to count blocks or bytes respectively during standby state. A count of zero equals 256. This register cannot be read, and is written using the WEVC command.

Event Counter Status Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

This register contains the current value in the event counter. The event counter counts back from the initial value stored in the Event Counter Register to zero and then generates an Event Counter Interrupt. This register cannot be written and can be read using a REVC command.

Time Out Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

The Time Out Register is used to store the time used for the time out error function. See the individual timeouts (TOUT1, 2, 3) to determine the units of this counter. This Time Out Register cannot be read, and it is written with the WTOUT command.

Time Out Status Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

This register contains the current value in the time out counter. The time out counter decrements from the original value stored in the Time Out Register. When zero is reached, the appropriate error interrupt is generated. If the register is read while none of the time out functions are active, the register will contain the last value reached the last time a function was active. The Time Out Status Register cannot be written, and it is read with the RTOUT command.

Error Flag Register

X	X	USER	X	X	$TOUT_3$	$TOUT_2$	$TOUT_1$

D_7 D_0

Four errors are flagged by the 8292 with a bit in the Error Flag Register. Each of these errors can be masked by the Error Mask Register. The Error Flag Register cannot be written, and it is read by the IACK command when the error flag in the Interrupt Status Register is set.

TOUT1 Time Out Error 1 occurs when the current controller has not stopped sending ATN after receiving the TCT message for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 1800 t_{CY}. After flagging the error, the 8292 will remain in a loop trying to take control until the current controller stops sending ATN or a new command is written by the microprocessor. If a new command is written, the 8292 will return to the loop after executing it.

TOUT2 Time Out Error 2 occurs when the transmission between the addressed talker and listener has not started for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 45 t_{CY}. This feature is only enabled when the controller is in the CSBS state.

TOUT3 Time Out Error 3 occurs when the handshake signals are stuck and the 8292 is not succeeding in taking control synchronously for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 1800 t_{CY}. The 8292 will continue checking $\overline{ATNI}$ until it becomes true or a new command is received. After performing the new command, the 8292 will return to the $\overline{ATNI}$ checking loop.

USER User error occurs when request to assert IFC or REN was received and the 8292 was not the system controller.

Error Mask Register

0	0	USER	0	0	$TOUT_3$	$TOUT_2$	$TOUT_1$
D_7							D_0

The Error Mask Register is used to mask the interrupt from a particular type of error. Each type of error interrupt is enabled by setting the corresponding bit in the Error Mask Register. This register can be read with the RERM command and written with A_0 low.

Command Register

1	1	1	OP	C	C	C	C
D_7							D_0

Commands are performed by the 8292 whenever a byte is written with A_0 high. There are two categories of commands distinguished by the OP bit (bit 4). The first category is the operation command (OP = 1). These commands initiate some action on the interface bus. The second category is the utility commands (OP = 0). These commands are used to aid the communication between the processor and the 8292.

OPERATION COMMANDS

Operation commands initiate some action on the GPIB interface bus. It is using these commands that the control functions such as polling, taking and passing control, and system controller functions are performed. A TCI interrupt is generated upon successful completion of each of these functions.

F0 — SPCNI — Stop Counter Interrupts

This command disables the internal counter interrupt so that the 8292 will stop interrupting the master on event counter underflows. However, the counter will continue counting and its contents can still be used.

F1 — GIDL — Go To Idle

This command is used during the transfer of control procedure while transferring control to another controller. The 8292 will respond to this command only if it is in the active state. $\overline{ATNO}$ will go high, and $\overline{CIC}$ will be high so that this 8292 will no longer be driving the ATN line on the GPIB interface bus.

F2 — RST — Reset

This command has the same effect as asserting the external reset on the 8292. For details, refer to the reset procedure described later.

F3 — RSTI — Reset Interrupts

This command resets any pending interrupts and clears the error flags. The 8292 will not return to any loop it was in (such as from the time out interrupts).

F4 — GSEC — Go To Standby, Enable Counting

The function causes $\overline{ATNO}$ to go high and the counter will be enabled. If the 8292 was not the active controller, this command will exit immediately. If the 8292 is the active controller, the counter will be loaded with the value stored in the Event Counter Register, and the internal interrupt will be enabled so that when the counter reaches zero, the SPI interrupt will be generated. SPI will be generated every 256 counts thereafter until the controller exits the standby state or the SPCNI command is written. An initial count of 256 (zero in the Event Counter Register) will be used if the WEVC command is not executed. If the data transmission does not start, a TOUT2 error will be generated.

F5 — EXPP — Execute Parallel Poll

This command initiates a parallel poll by asserting ATN and EOI (IDY message) true. The 8291 should be previously configured as a listener. Upon detection of DAV true, the 8291 enters ACDS and latches the parallel poll response (PPR) byte into its data in register. The master will be interrupted by the 8291 BI interrupt when the PPR byte is available. No interrupts except the $\overline{IBFI}$ will be generated by the 8292. The 8292 will respond to this command only when it is the active controller.

F6 — GTSB — Go To Standby

If the 8292 is the active controller, $\overline{ATNO}$ will go high then TCI will be generated. If the data transmission does not start, a TOUT2 error will be generated.

F7 — SLOC — Set Local Mode

If the 8292 is the system controller, then REN will be asserted false for at least 100 μsec. If it is not the system controller, the User Error bit will be set in the Error Flag Register.

F8 — SREM — Set Interface To Remote Control

This command will set REN true if this 8292 is the system controller. If not, the User Error bit will be set in the Error Flag Register.

F9 — ABORT — Abort All Operation, Clear Interface

This command will cause IFC to be asserted true for at least 100 μsec if this 8292 is the system controller. If it is in CIDS, it will take control over the bus (see the TCNTR command).

FA — TCNTR — Take Control

The transfer of control procedure is coordinated by the master with the 8291 and 8292. When the master receives a TCT message from the 8291, it should issue the TCNTR command to the 8292. The following events occur to take control:

1. The 8292 checks to see if it is in CIDS, and if not, it exits.
2. Then $\overline{\text{ATNI}}$ is checked until it becomes high. If the current controller does not release ATN for the time specified by the Time Out Register, then a TOUT1 error is generated. The 8292 will return to this loop after an error or any command except the RST and RSTI commands.
3. After the current controller releases ATN, the 8292 will assert $\overline{\text{ATNO}}$ and $\overline{\text{CIC}}$ low.
4. Finally, the TCI interrupt is generated to inform the master that it is in control of the bus.

FC — TCASY — Take Control Asynchronously

TCAS transfers the 8292 from CSBS to CACS independent of the handshake lines. If a bus hangup is detected (by an error flag), this command will force the 8292 to take control (asserting ATN) even if the AH function is not in ANRS (Acceptor Not Ready State). This command should be used very carefully since it may cause the loss of a data byte. Normally, control should be taken synchronously. After checking the controller function for being in the CSBS (else it will exit immediately), $\overline{\text{ATNO}}$ will go low, and a TCI interrupt will be generated.

FD — TCSY — Take Control Synchronously

There are two different procedures used to transfer the 8292 from CSBS to CACS depending on the state of the 8291 in the system. If the 8291 is in "continuous AH cycling" mode (Aux. Reg. A0 = A1 = 1), then the following procedure should be followed:

1. The master microprocessor stops the continuous AH cycling mode in the 8291;
2. The master reads the 8291 Interrupt Status 1 Register;
3. If the END bit is set, the master sends the TCSY command to the 8292;
4. If the END bit was not set, the master reads the 8291 Data In Register and then waits for another BI interrupt from the 8291. When it occurs, the master sends the 8292 the TCSY command.

If the 8291 is not in AH cycling mode, then the master just waits for a BI interrupt and then sends the TCSY command. After the TCSY command has been issued, the 8292 checks for $\overline{\text{CSBS}}$. If $\overline{\text{CSBS}}$, then it exits the routine. Otherwise, it then checks the DAV bit in the GPIB status. When DAV becomes false, the 8292 will wait for at least 1.5 μsec. (T10) and then $\overline{\text{ATNO}}$ will go low. If DAV does not go low, a TOUT3 error will be generated.

FE — STCNI — Start Counter Interrupts

This command enables the internal counter interrupt. The counter is enabled by the GSEC command.

UTILITY COMMANDS

All these commands are either Read or Write to registers in the 8292. Upon completion of Read commands, the TCI (Task Completed Interrupt) will be generated. Note that writing to the Error Mask Register and the Interrupt Mask Register are done directly.

E1 — WTOUT — Write To Time Out Register

The byte written to the data bus buffer (with $A_0 = 0$) following this command will determine the time used for the time out function. Since this function is implemented in software, this will not be an accurate time measurement. This feature is enable or disable by the Error Mask Register. No interrupts except for the $\overline{\text{IBFI}}$ will be generated upon completion.

E2 — WEVC — Write To Event Counter

The byte written to the data bus buffer (with $A_0 = 0$) following this command will be loaded into the Event Counter Register and the Event Counter Status for byte counting or EOI counting. Only $\overline{\text{IBFI}}$ will indicate completion of this command.

E3 — REVC — Read Event Counter Status

This command transfers the contents of the Event Counter into the data bus buffer. A TCI is generated when the data is available in the data bus buffer.

E4 — RERF — Read Error Flag Register

This command transfers the contents of the Error Flag Register into the data bus buffer. A TCI is generated when the data is available.

E5 — RINM — Read Interrupt Mask Register

This command transfers the contents of the Interrupt Mask Register into the data bus buffer. This register is available to the processor so that it does not need to store this information elsewhere. A TCI is generated when the data is available in the data bus buffer.

E6 — RCST — Read Controller Status Register

This command transfers the contents of the Controller Status Register into the data bus buffer and a TCI interrupt is generated.

E7 — RBST — Read GPIB Bus Status Register

This command transfers the contents of the GPIB Bus Status Register into the data bus buffer, and a TCI interrupt is generated when the data is available.

E9 — RTOUT — Read Time Out Status Register

This command transfers the contents of the Time Out Status Register into the data bus buffer, and a TCI interrupt is generated when the data is available.

EA — RERM — Read Error Mask Register

This command transfers the contents of the Error Mask Register to the data bus buffer so that the processor does not need to store this information elsewhere. A TCI interrupt is generated when the data is available.

Interrupt Acknowledge

SYC	ERR	SRQ	EV	1	IFCR	1	1

D_7 D_0

Each named bit in an Interrupt Acknowledge (IACK) corresponds to a flag in the Interrupt Status Register. When the 8292 receives this command, it will clear the SPI and the corresponding bits in the Interrupt Status Register. If not all the bits were cleared, then the SPI will be set true again. If the error flag is not acknowledged by the IACK command, then the Error Flag Register will be transferred to the data bus buffer, and a TCI will be generated.

NOTE: XXXX1X11 is an undefined operation or utility command, so no conflict exists between the IACK operation and utility commands.

SYSTEM OPERATION

8292 To Master Processor Interface

Communication between the 8292 and the Master Processor can be either interrupt based communication or based upon polling the interrupt status register in predetermined intervals.

Interrupt Based Communication

Four different interrupts are available from the 8292:

OBFI Output Buffer Full Interrupt
$\overline{IBFI}$ Input Buffer Not Full Interrupt
TCI Task Completed Interrupt
SPI Special Interrupt

Each of the interrupts is enabled or disabled by a bit in the interrupt mask register. Since OBFI and $\overline{IBFI}$ are directly connected to the OBF and IBF flags, the master can write a new command to the input data bus buffer as soon as the previous command has been read.

The TCI interrupt is useful when the master is sending commands to the 8292. The pending TCI will be cleared with each new command written to the 8292. Commands sent to the 8292 can be divided into two major groups:

1. Commands that require response back from the 8292 to the master, e.g., reading register.
2. Commands that initiate some action or enable features but do not require response back from the 8292, e.g., enable data bus buffer interrupts.

With the first group, the TCI interrupt will be used to indicate that the required response is ready in the data bus buffer and the master may continue and read it. With the second group, the interrupt will be used to indicate completion of the required task, so that the master may send new commands.

The SPI should be used when immediate information or special events is required (see the Interrupt Status Register).

"Polling Status" Based Communication

When interrupt based communication is not desired, all interrupts can be masked by the interrupt mask register. The communication with the 8292 is based upon sequential poll of the interrupt status register. By testing the OBF and IBF flags, the data bus buffer status is determined while special events are determined by testing the other bits.

Receiving IFC

The IFC pulse defined by the IEEE-488 standard is at least 100 μsec. In this time, all operation on the bus should be aborted. Most important, the current controller (the one that is in charge at that time) should stop sending ATN or EOI. Thus, IFC must externally gate $\overline{CIC}$ (controller in charge) and $\overline{ATNO}$ to ensure that this occurs.

Reset and Power Up Procedure

After the 8292 has been reset either by the external reset pin, the device being powered on, or a RST command, the following sequential events will take place:

1. All outputs to the GPIB interface will go high ($\overline{SRQ}$, $\overline{ATNI}$, $\overline{IFC}$, SYC, CLTH, $\overline{ATNO}$, $\overline{CIC}$, TCI, SPI, $\overline{EOI}$, OBFI, $\overline{IBFI}$, $\overline{DAV}$, $\overline{REV}$).
2. The four interrupt outputs (TCI, SPI, OBFI, $\overline{IBFI}$) and CLTH output will go low.
3. The following registers will be cleared:
   Interrupt Status
   Interrupt Mask
   Error Flag
   Error Mask
   Time Out
   Event Counter (= 256), Counter is disabled.
4. If the 8292 is the system controller, an ABORT command will be executed, the 8292 will become the controller in charge, and it will enter the CACS state.

   If it is not the system controller, it will remain in CIDS.

System Configuration

The 8291 and 8292 must be interfaced to an IEEE-488 bus meeting a variety of specifications including drive capability and loading characteristics. To interface the 8291 and the 8292 without the 8293's, several external gates are required, using a configuration similar to that used in Figure 3.

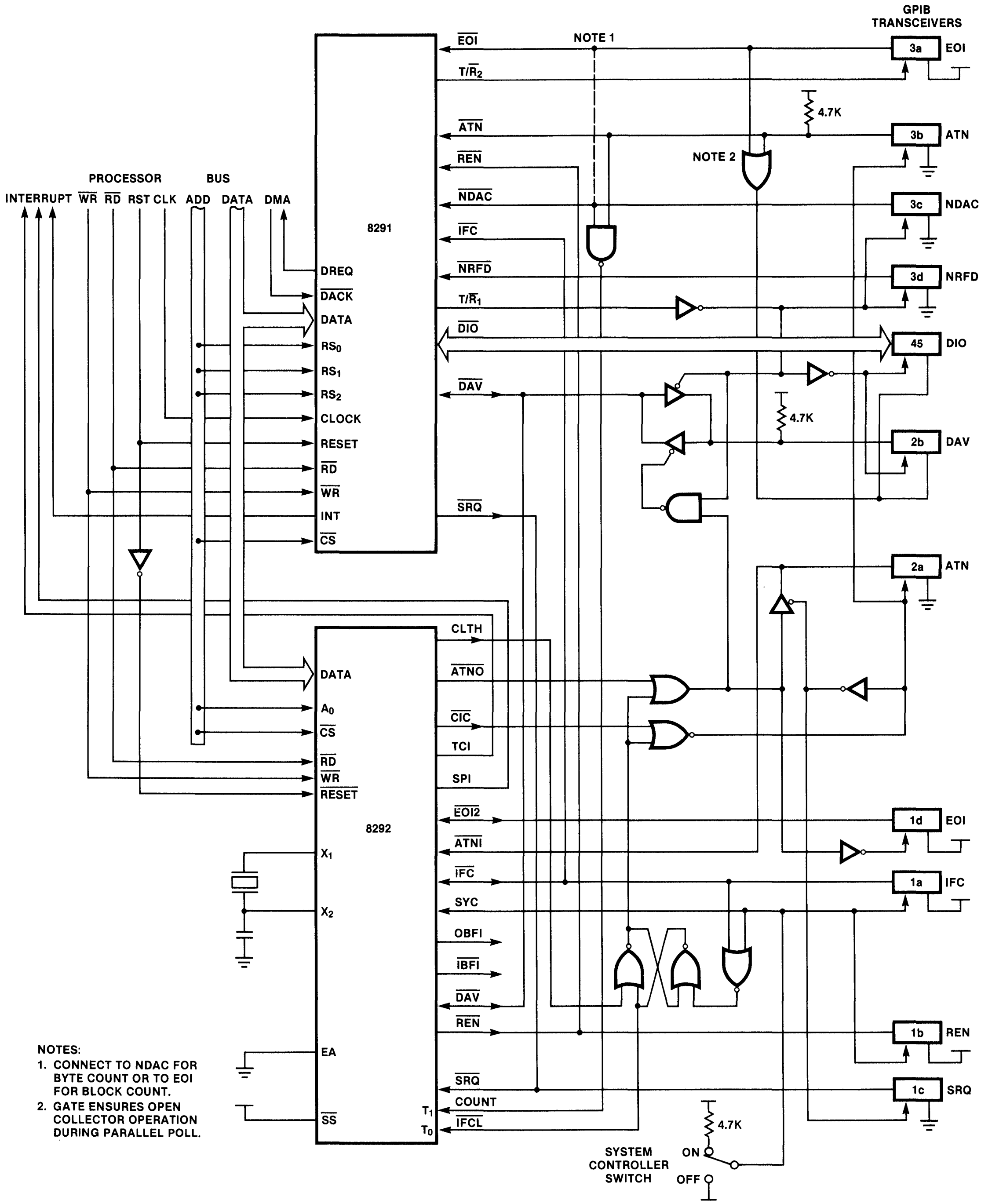
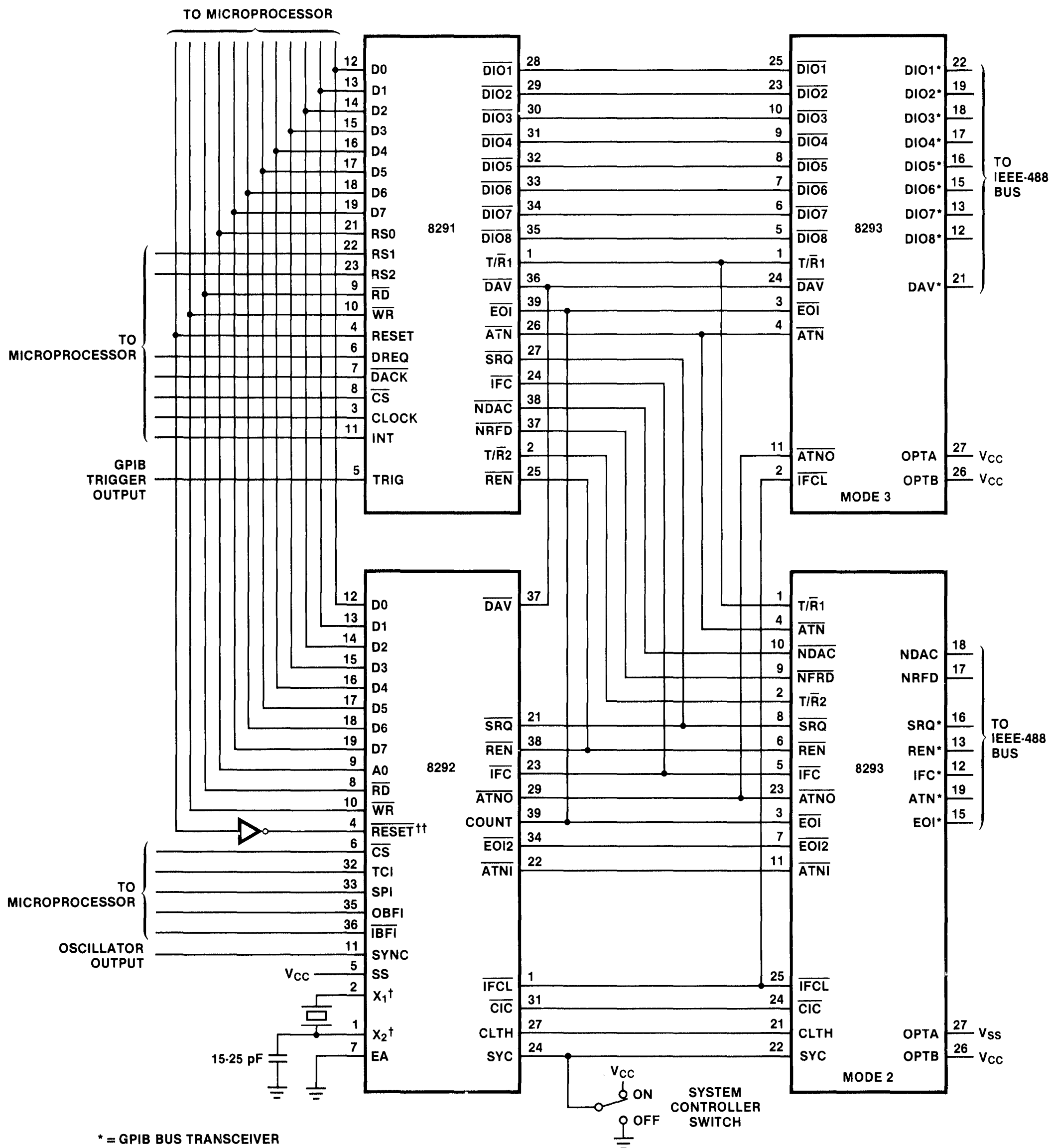


Figure 2. 8291 and 8292 System Configuration

00741C



* = GPIB BUS TRANSCEIVER
† = SEE 8041A DATA SHEET FOR ALTERNATE CRYSTAL CONFIGURATIONS
†† = CAN CONNECT TO SYSTEM RESET SWITCH, SEE 8041A DATA SHEET

Figure 3. 8291, 8292, and 8293 System Configuration

00741C

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias.........0°C to 70°C
Storage Temperature.............−65°C to +150°C
Voltage on Any Pin With Respect
to Ground..........................0.5V to +7V
Power Dissipation..........................1.5 Watt

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

D.C. AND OPERATING CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{SS} = 0V$, 8292: $V_{CC} = \pm 5V \pm 10\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL1}	Input Low Voltage (All Except X_1, X_2, $\overline{RESET}$)	−0.5	0.8	V	
V_{IL2}	Input Low Voltage (X_1, X_2, $\overline{RESET}$)	−0.5	0.6	V	
V_{IH1}	Input High Voltage (All Except X_1, X_2, $\overline{RESET}$)	2.2	V_{CC}	V	
V_{IH2}	Input High Voltage (X_1, X_2, $\overline{RESET}$)	3.8	V_{CC}	V	
V_{OL1}	Output Low Voltage (D_0–D_7)		0.45	V	$I_{OL} = 2.0$ mA
V_{OL2}	Output Low Voltage (All Other Outputs)		0.45	V	$I_{OL} = 1.6$ mA
V_{OH1}	Output High Voltage (D_0–D_7)	2.4		V	$I_{OH} = -400\ \mu A$
V_{OH2}	Output High Voltage (All Other Outputs)	2.4		V	$I_{OH} = -50\ \mu A$
I_{IL}	Input Leakage Current (COUNT, $\overline{IFCL}$, $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_0)		±10	μA	$V_{SS} \leqslant V_{IN} \leqslant V_{CC}$
I_{OZ}	Output Leakage Current (D_0–D_7, High Z State)		±10	μA	$V_{SS} + 0.45 \leqslant V_{IN} \leqslant V_{CC}$
I_{LI1}	Low Input Load Current (Pins 21-24, 27-38)		0.5	mA	$V_{IL} = 0.8V$
I_{LI2}	Low Input Load Current ($\overline{RESET}$)		0.2	mA	$V_{IL} = 0.8V$
I_{CC}	Total Supply Current		125	mA	Typical = 65 mA

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{SS} = 0V$, 8292: $V_{CC} = +5V \pm 10\%$

1. DBB READ

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}$↓	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}$↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	$C_L = 150$ pF
t_{RD}	$\overline{RD}$↓ to Data Out Delay		225	ns	$C_L = 150$ pF
t_{DF}	$\overline{RD}$↑ to Data Float Delay		100	ns	
t_{CY}	Cycle Time	2.5	15	μs	

2. DBB WRITE

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	CS, A_0 Setup to WR↓	0		ns	
t_{WA}	CS, A_0 Hold After WR↑	0		ns	
t_{WW}	WR Pulse Width	250		ns	
t_{DW}	Data Setup to WR↑	150		ns	
t_{WD}	Data Hold After WR↓	0		ns	

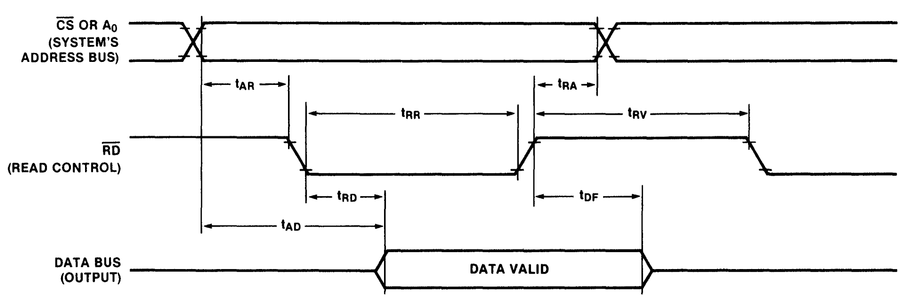
3. COMMAND TIMINGS[1,3]

Code	Name	Execution Time	$\overline{IBF}$I↑	TCI[2]	SPI	$\overline{ATNO}$	$\overline{CIC}$	$\overline{IFC}$	$\overline{REN}$	$\overline{EOI}$	$\overline{DAV}$	Comments
E1	WTOUT	63	24									
E2	WEVC	63	24									
E3	REVC	71	24	51								
E4	RERF	67	24	47								
E5	RINM	69	24	49								
E6	RCST	97	24	77								
E7	RBST	92	24	72								
E8												
E9	RTOUT	69	24	49								
EA	RERM	69	24	49								
F0	SPCNI	53	24									Count Stops After 39
F1	GIOL	88	24	70		↑61	↑61					
F2	RST	94	24		↓52							Not System Controller
F2	RST	214	24	192	↓52	↓179	↓174	↓101				System Controller
F3	RSTI	61	24									
F4	GSEC	125	24	107		↑98						
F5	EXPP	75	24						↓53 ↑59	↓55 ↑57		
F6	GTSB	118	24	100		↑91						
F7	SLOC	73	24	55				↑46				
F8	SREM	91	24	73				↓64				
F9	ABORT	155	24	133		↓120	↓115	↓42				
FA	TCNTR	108	24	86		↓71	↓68					
FC	TCAS	92	24	67		↓55						
FD	TCSY	115	24	91		↓80						
FE	STCNI	59	24									Starts Count After 43
PIN	RESET	29	—	↓7	↓7							Not System Controller
X	IACK	116	—		↓73 ↑98							If Interrupt Pending

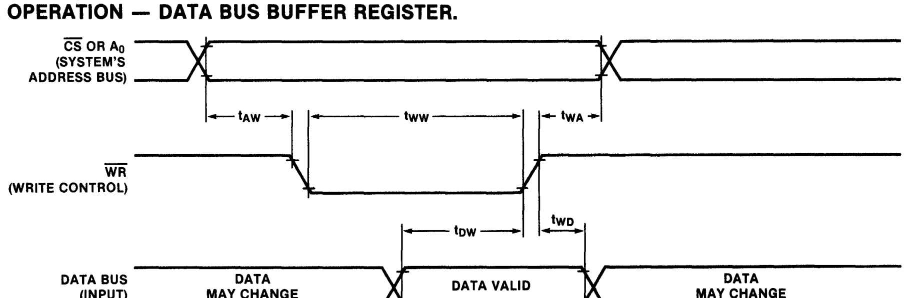
Notes: 1. All times are multiples of t_{CY} from the 8041A command interrupt.
2. TCI clears after 7 t_{CY} on all commands.
3. ↑ indicates a level transition from low to high, ↓ indicates a high to low transition.

WAVEFORMS

1. READ OPERATION — DATA BUS BUFFER REGISTER.



2. WRITE OPERATION — DATA BUS BUFFER REGISTER.



00741C

APPENDIX

The following tables and state diagrams were taken from the IEEE Standard Digital Interface for Programmable Instrumentation, IEEE Std. 488-1978. This document is the official standard for the GPIB bus and can be purchased from IEEE, 345 East 47th St., New York, NY 10017.

C MNEMONICS

Messages	Interface States
pon = power on	CIDS = controller idle state
rsc = request system control	CADS = controller addressed state
rpp = request parallel poll	CTRS = controller transfer state
gts = go to standby	CACS = controller active state
tca = take control asynchronously	CPWS = controller parallel poll wait state
tcs = take control synchronously	CPPS = controller parallel poll state
sic = send interface clear	CSBS = controller standby state
sre = send remote enable	CSHS = controller standby hold state
IFC = interface clear	CAWS = controller active wait state
ATN = attention	CSWS = controller synchronous wait state
TCT = take control	CSRS = controller service requested state
	CSNS = controller service not requested state
	SNAS = system control not active state
	SACS = system control active state
	SRIS = system control remote enable idle state
	SRNS = system control remote enable not active state
	SRAS = system control remote enable active state
	SIIS = system control interface clear idle state
	SINS = system control interface clear not active state
	SIAS = system control interface clear active state
	(ACDS) = accept data state (AH function)
	(ANRS) = acceptor not ready state (AH function)
	(SDYS) = source delay state (SH function)
	(STRS) = source transfer state (SH function)
	(TADS) = talker addressed state (T function)

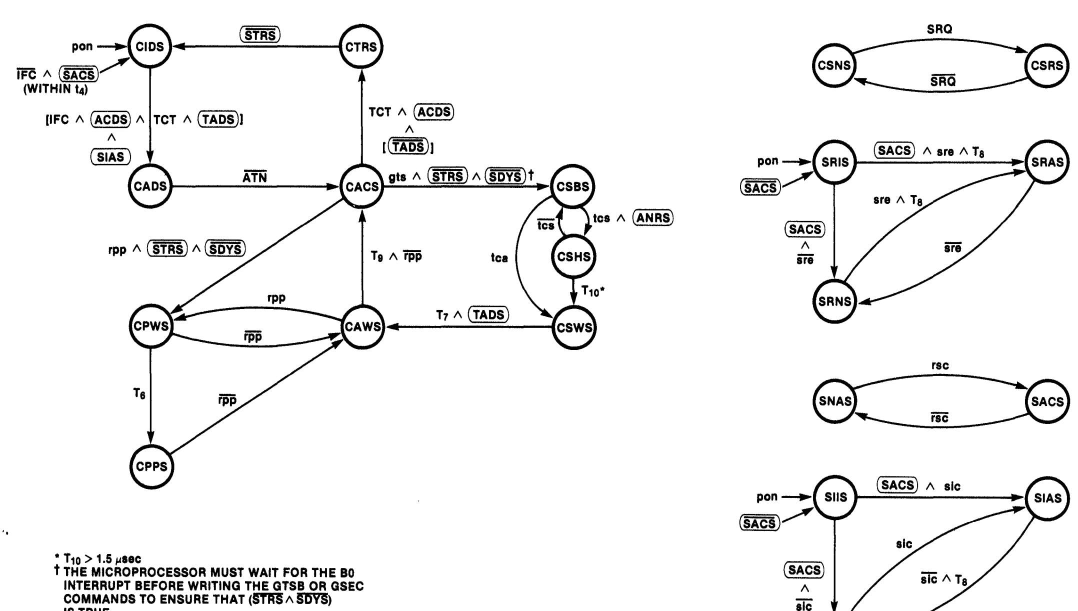


Figure A.1. C State Diagram

00741C

REMOTE MESSAGE CODING

Bus Signal Line(s) and Coding That Asserts the True Value of the Message

Mnemonic	Message Name		TYPE	CLASS	DIO 8	7	6	5	4	3	2	DIO 1	DAV	NRFD	NDAC	ATN	EOI	SRQ	IFC	REN
ACG	Addressed Command Group		M	AC	Y	0	0	0	X	X	X	X	X	X	X	1	X	X	X	X
ATN	Attention		U	UC	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	X
DAB	Data Byte	(Notes 1, 9)	M	DD	D8	D7	D6	D5	D4	D3	D2	D1	X	X	X	0	X	X	X	X
DAC	Data Accepted		U	HS	X	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X
DAV	Data Valid		U	HS	X	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X
DCL	Device Clear		M	UC	Y	0	0	1	0	1	0	0	X	X	X	1	X	X	X	X
END	End		U	ST	X	X	X	X	X	X	X	X	X	X	X	0	1	X	X	X
EOS	End of String	(Notes 2, 9)	M	DD	E8	E7	E6	E5	E4	E3	E2	E1	X	X	X	0	X	X	X	X
GET	Group Execute Trigger		M	AC	Y	0	0	0	1	0	0	0	X	X	X	1	X	X	X	X
GTL	Go to Local		M	AC	Y	0	0	0	0	0	0	1	X	X	X	1	X	X	X	X
IDY	Identify		U	UC	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X
IFC	Interface Clear		U	UC	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	X
LAG	Listen Address Group		M	AD	Y	0	1	X	X	X	X	X	X	X	X	1	X	X	X	X
LLO	Local Lock Out		M	UC	Y	0	0	1	0	0	0	1	X	X	X	1	X	X	X	X
MLA	My Listen Address	(Note 3)	M	AD	Y	0	1	L5	L4	L3	L2	L1	X	X	X	1	X	X	X	X
MTA	My Talk Address	(Note 4)	M	AD	Y	1	0	T5	T4	T3	T2	T1	X	X	X	1	X	X	X	X
MSA	My Secondary Address	(Note 5)	M	SE	Y	1	1	S5	S4	S3	S2	S1	X	X	X	1	X	X	X	X
NUL	Null Byte		M	DD	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X
OSA	Other Secondary Address		M	SE	(OSA = SCG $\wedge$ $\overline{MSA}$)															
OTA	Other Talk Address		M	AD	(OTA = TAG $\wedge$ $\overline{MTA}$)															
PCG	Primary Command Group		M	—	(PCG = ACG $\vee$ UCG $\vee$ LAG $\vee$ TAG)															
PPC	Parallel Poll Configure		M	AC	Y	0	0	0	0	1	0	1	X	X	X	1	X	X	X	X
PPE	Parallel Poll Enable	(Note 6)	M	SE	Y	1	1	0	S	P3	P2	P1	X	X	X	1	X	X	X	X
PPD	Parallel Poll Disable	(Note 7)	M	SE	Y	1	1	1	D4	D3	D2	D1	X	X	X	1	X	X	X	X
PPR1	Parallel Poll Response 1		U	ST	X	X	X	X	X	X	X	1	X	X	X	1	1	X	X	X
PPR2	Parallel Poll Response 2		U	ST	X	X	X	X	X	X	1	X	X	X	X	1	1	X	X	X
PPR3	Parallel Poll Response 3		U	ST	X	X	X	X	X	1	X	X	X	X	X	1	1	X	X	X
PPR4	Parallel Poll Response 4	(Note 10)	U	ST	X	X	X	X	1	X	X	X	X	X	X	1	1	X	X	X
PPR5	Parallel Poll Response 5		U	ST	X	X	X	1	X	X	X	X	X	X	X	1	1	X	X	X
PPR6	Parallel Poll Response 6		U	ST	X	X	1	X	X	X	X	X	X	X	X	1	1	X	X	X
PPR7	Parallel Poll Response 7		U	ST	X	1	X	X	X	X	X	X	X	X	X	1	1	X	X	X
PPR8	Parallel Poll Response 8		U	ST	1	X	X	X	X	X	X	X	X	X	X	1	1	X	X	X
PPU	Parallel Poll Unconfigure		M	UC	Y	0	0	1	0	1	0	1	X	X	X	1	X	X	X	X
REN	Remote Enable		U	UC	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1
RFD	Ready for Data		U	HS	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X	X
RQS	Request Service	(Note 9)	U	ST	X	1	X	X	X	X	X	X	X	X	X	0	X	X	X	X
SCG	Secondary Command Group		M	SE	Y	1	1	X	X	X	X	X	X	X	X	1	X	X	X	X
SDC	Selected Device Clear		M	AC	Y	0	0	0	0	1	0	0	X	X	X	1	X	X	X	X
SPD	Serial Poll Disable		M	UC	Y	0	0	1	1	0	0	1	X	X	X	1	X	X	X	X
SPE	Serial Poll Enable		M	UC	Y	0	0	1	1	0	0	0	X	X	X	1	X	X	X	X
SRQ	Service Request		U	ST	X	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X
STB	Status Byte	(Notes 8, 9)	M	ST	S8	X	S6	S5	S4	S3	S2	S1	X	X	X	0	X	X	X	X
TCT	Take Control		M	AC	Y	0	0	0	1	0	0	1	X	X	X	1	X	X	X	X
TAG	Talk Address Group		M	AD	Y	1	0	X	X	X	X	X	X	X	X	1	X	X	X	X
UCG	Universal Command Group		M	UC	Y	0	0	1	X	X	X	X	X	X	X	1	X	X	X	X
UNL	Unlisten		M	AD	Y	0	1	1	1	1	1	1	X	X	X	1	X	X	X	X
UNT	Untalk	(Note 11)	M	AD	Y	1	0	1	1	1	1	1	X	X	X	1	X	X	X	X

The 1/0 coding on ATN when sent concurrent with multiline messages has been added to this revision for interpretive convenience.

NOTES:

1. D1-D8 specify the device dependent data bits.
2. E1-E8 specify the device dependent code used to indicate the EOS message.
3. L1-L5 specify the device dependent bits of the device's listen address.
4. T1-T5 specify the device dependent bits of the device's talk address.
5. S1-S5 specify the device dependent bits of the device's secondary address.
6. S specifies the sense of the PPR.

   Response = $\overline{S \oplus ist}$

P1-P3 specify the PPR message to be sent when a parallel poll is executed.

P3	P2	P1	PPR Message
0	0	0	PPR1
.	.	.	.
.	.	.	.
.	.	.	.
1	1	1	PPR8

7. D1-D4 specify don't-care bits that shall not be decoded by the receiving device. It is recommended that all zeroes be sent.
8. S1-S6, S8 specify the device dependent status. (DIO7 is used for the RQS message.)
9. The source of the message on the ATN line is always the C function, whereas the messages on the DIO and EOI lines are enabled by the T function.
10. The source of the messages on the ATN and EOI lines is always the C function, whereas the source of the messages on the DIO lines is always the PP function.
11. This code is provided for system use, see 6.3.

intel®

8294
DATA ENCRYPTION UNIT

- **Certified by National Bureau of Standards**
- **80 Byte/Sec Data Conversion Rate**
- **64-Bit Data Encryption Using 56-Bit Key**
- **DMA Interface**
- **3 Interrupt Outputs to Aid in Loading and Unloading Data**
- **7-Bit User Output Port**
- **Single 5V ± 10% Power Supply**
- **Peripheral to MCS-86™, MCS-85™, MCS-80™ and MCS-48™ Processors**
- **Implements Federal Information Processing Data Encryption Standard**
- **Encrypt and Decrypt Modes Available**

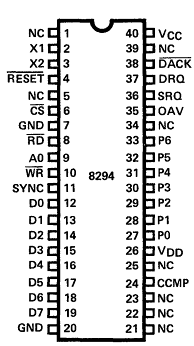
DESCRIPTION

The Intel® 8294 Data Encryption Unit (DEU) is a microprocessor peripheral device designed to encrypt and decrypt 64-bit blocks of data using the algorithm specified in the Federal Information Processing Data Encryption Standard. The DEU operates on 64-bit text words using a 56-bit user-specified key to produce 64-bit cipher words. The operation is reversible: if the cipher word is operated upon, the original text word is produced. The algorithm itself is permanently contained in the 8294; however, the 56-bit key is user-defined and may be changed at any time.

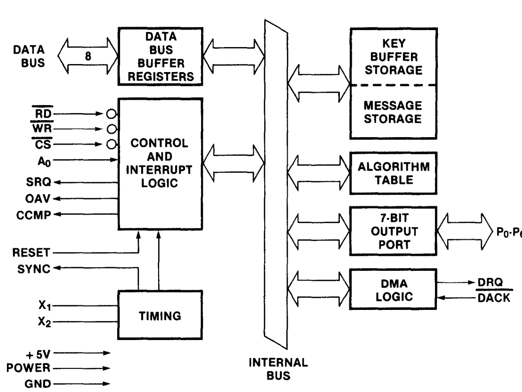
The 56-bit key and 64-bit message data are transferred to and from the 8294 in 8-bit bytes by way of the system data bus. A DMA interface and three interrupt outputs are available to minimize software overhead associated with data transfer. Also, by using the DMA interface two or more DEUs may be operated in parallel to achieve effective system conversion rates which are virtually any multiple of 80 bytes/second. The 8294 also has a 7-bit TTL compatible output port for user-specified functions.

Because the 8294 implements the NBS encryption algorithm it can be used in a variety of Electronic Funds Transfer applications as well as other electronic banking and data handling applications where data must be encrypted.

PIN CONFIGURATION



BLOCK DIAGRAM



00230B

Pin #	Pin Name	I/O	Pin Description
1	NC	—	No connection.
2 3	X1 X2	I	Inputs for crystal, L-C or external timing signal to determine internal oscillator frequency.
4	$\overline{RESET}$	I	A low signal to this pin resets the 8294.
5	NC	—	No connection or tied high.
6	$\overline{CS}$	I	A low signal to this pin enables reading and writing to the 8294.
7	GND	—	This pin must be tied to ground.
8	$\overline{RD}$	I	An active low read strobe at this pin enables the CPU to read data and status from the internal DEU registers.
9	A_0	I	Address input used by the CPU to select DEU registers during read and write operations.
10	$\overline{WR}$	I	An active low write strobe at this pin enables the CPU to send data and commands to the DEU.
11	SYNC	O	High frequency (Clock ÷ 15) output. Can be used as a strobe for external circuitry.
12 13 14 15 16 17 18 19	D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7	I/O	Three-state, bi-directional data bus lines used to transfer data between the CPU and the 8294.
20	GND	—	This pin must be tied to ground.

Pin #	Pin Name	I/O	Pin Description
40	V_{CC}	—	+5 volt power input: +5V ± 10%.
39	NC	—	No connection.
38	$\overline{DACK}$	I	DMA acknowledge. Input signal from the 8257 DMA Controller acknowledging that the requested DMA cycle has been granted.
37	DRQ	O	DMA request. Output signal to the 8257 DMA Controller requesting a DMA cycle.
38	SRQ	O	Service Request. Interrupt to the CPU indicating that the 8294 is awaiting data or commands at the input buffer. SRQ = 1 implies IBF = 0.
35	OAV	O	Output Available. Interrupt to the CPU indicating that the 8294 has data or status available in its output buffer. OAV = 1 implies OBF = 1.
34	NC	—	No connection.
33 32 31 30 29 28 27	P6 P5 P4 P3 P2 P1 P0	O	User output port lines. Output lines available to the user via a CPU command which can assert selected port lines. These lines have nothing to do with the encryption function. At power-on, each line is in a 1 state.
26	V_{DD}	—	+5V power input. (+5V ± 10%) Low power standby pin.
25	NC	—	No connection.
24	CCMP	O	Conversion Complete. Interrupt to the CPU indicating that the encryption/decryption of an 8-byte block is complete.
23	NC	—	No connection.
22	NC	—	No connection.
21	NC	—	No connection.

BASIC FUNCTIONAL DESCRIPTION

OPERATION

The data conversion sequence is as follows:

1. A Set Mode command is given, enabling the desired interrupt outputs.
2. An Enter New Key command is issued, followed by 8 data inputs which are retained by the DEU for encryption/decryption. Each byte must have odd parity.
3. An Encrypt Data or Decrypt Data command sets the DEU in the desired mode.

After this, data conversions are made by writing 8 data bytes and then reading back 8 converted data bytes. Any of the above commands may be issued between data conversions to change the basic operation of the DEU; e.g., a Decrypt Data command could be issued to change the DEU from encrypt mode to decrypt mode without changing either the key or the interrupt outputs enabled.

INTERNAL DEU REGISTERS

Four internal registers are addressable by the master processor: 2 for input, and 2 for output. The following table describes how these registers are accessed.

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	A_0	Register
1	0	0	0	Data input buffer
0	1	0	0	Data output buffer
1	0	0	1	Command input buffer
0	1	0	1	Status output buffer
X	X	1	X	Don't care

The functions of each of these registers are described below.

Data Input Buffer — Data written to this register is interpreted in one of three ways, depending on the preceding command sequence.

1. Part of a key.
2. Data to be encrypted or decrypted.
3. A DMA block count.

Data Output Buffer — Data read from this register is the output of the encryption/decryption operation.

Command Input Buffer — Commands to the DEU are written into this register. (See command summary below.)

Status Output Buffer — DEU status is available in this register at all times. It is used by the processor for poll-driven command and data transfer operations.

STATUS BIT:	7	6	5	4	3	2	1	0
FUNCTION:	X	X	X	KPE	CF	DEC	IBF	OBF

OBF Output Buffer Full; OBF = 1 indicates that output from the encryption/decryption function is available in the Data Output Buffer. It is reset when the data is read.

IBF Input Buffer Full; A write to the Data Input Buffer or to the Command Input Buffer sets IBF = 1. The DEU resets this flag when it has accepted the input byte. Nothing should be written when IBF = 1.

DEC Decrypt; indicates whether the DEU is in an encrypt or a decrypt mode. DEC = 1 implies the decrypt mode. DEC = 0 implies the encrypt mode.

CF Completion Flag; This flag may be used to indicate any or all of three events in the data transfer protocol.

1. It may be used in lieu of a counter in the processor routine to flag the end of an 8-byte transfer.
2. It must be used to indicate the validity of the KPE flag.
3. It may be used in lieu of the CCMP interrupt to indicate the completion of a DMA operation.

KPE Key Parity Error; After a new key has been entered, the DEU uses this flag in conjunction with the CF flag to indicate correct or incorrect parity.

COMMAND SUMMARY

1 — Enter New Key

OP CODE: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | (MSB … LSB)

This command is followed by 8 data byte inputs which are retained in the key buffer (RAM) to be used in encrypting and decrypting data. These data bytes must have odd parity represented by the LSB.

2 — Encrypt Data

OP CODE: | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | (MSB … LSB)

This command puts the 8294 into the encrypt mode.

3 — Decrypt Data

OP CODE: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | (MSB … LSB)

This command puts the 8294 into the decrypt mode.

4 — Set Mode

OP CODE: | 0 | 0 | 0 | 0 | A | B | C | D | (MSB … LSB)

where:

A is the OAV (Output Available) interrupt enable
B is the SRQ (Service Request) interrupt enable
C is the DMA (Direct Memory Access) transfer enable
D is the CCMP (Conversion Complete) interrupt enable

This command determines which interrupt outputs will be enabled. A "1" in bits A, B, or D will enable the OAV, SRQ, or CCMP interrupts respectively. A "1" in bit C will allow DMA transfers. When bit C is set the OAV and SRQ interrupts should also be enabled (bits A,B = 1). Following the command in which bit C, the DMA bit, is set, the 8294 will expect one data byte to specify the number of 8-byte blocks to be converted using DMA.

5 — Write to Output Port

OP CODE:

1	P_6	P_5	P_4	P_3	P_2	P_1	P_0
MSB							LSB

This command causes the 7 least significant bits of the command byte to be latched as output data on the 8294 output port. The initial output data is 1111111. Use of this port is independent of the encryption/decryption function.

After the Enter New Key command is issued, 8 data bytes representing the new key are written to the data input buffer (most significant byte first). After the eighth byte is accepted by the DEU, CF goes true (CF = 1). The CF bit goes false again when KPE is valid. The CPU can then check the KPE flag. If KPE = 1, a parity error has been detected and the DEU has not accepted the key. Each byte is checked for odd parity, where the parity bit is the LSB of each byte.

Since the CF bit is used in this protocol to indicate the validity of the KPE flag, it may not be used to flag the end of the 8 byte key entry. CF = 1 only as long as KPE is invalid. Therefore, the CPU might not detect that CF = 1 and the key entry is complete before KPE becomes valid. Thus, a counter should be used, as in Figure 2, to flag the end of the new key entry. Then, CF is used to indicate a valid KPE flag.

PROCESSOR/DEU INTERFACE PROTOCOL

ENTERING A NEW KEY

The timing sequence for entering a new key is shown in Figure 1. A flowchart showing the CPU software to accommodate this sequence is given in Figure 2.

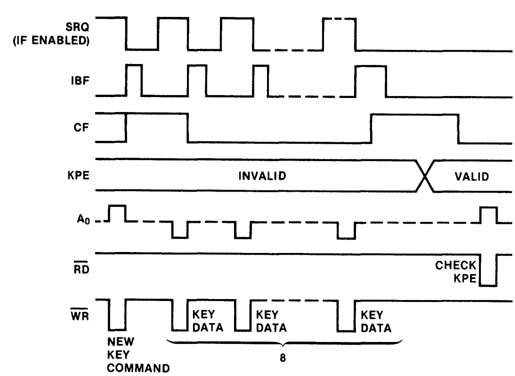


Figure 1. Entering a New Key

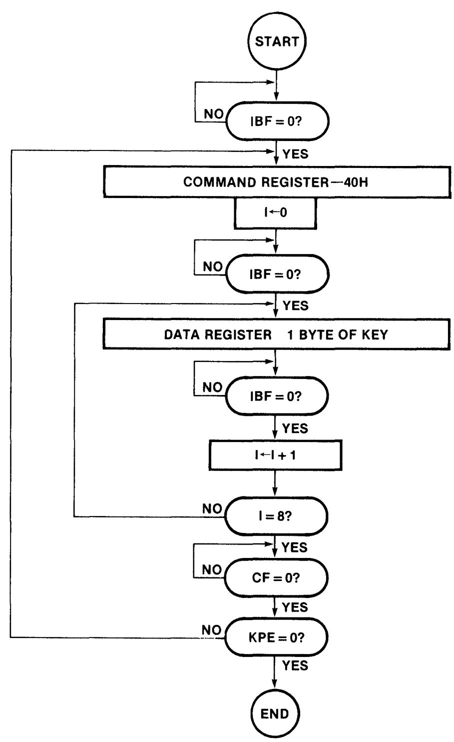


Figure 2. Flowchart for Entering a New Key

ENCRYPTING OR DECRYPTING DATA

Figure 3 shows the timing sequence for encrypting or decrypting data. The CPU writes 8 data bytes to the DEU's data input buffer for encryption/decryption. CF then goes true (CF = 1) to indicate that the DEU has accepted the 8-byte block. Thus, the CPU may test for IBF = 0 and CF = 1 to terminate the input mode, or it may use a software counter. When the encryption/decryption is complete, the CCMP and OAV interrupts are asserted and the OBF flag is set true (OBF = 1). OAV and OBF are set false again after each of the converted data bytes is read back by the CPU. The CCMP interrupt is set false, and remains false, after the first read. After 8 bytes have been read back by the CPU, CF goes false (CF = 0). Thus, the CPU may test for CF = 0 to terminate the read mode. Also, the CCMP interrupt may be used to initiate a service routine which performs the next series of 8 data reads and 8 data writes.

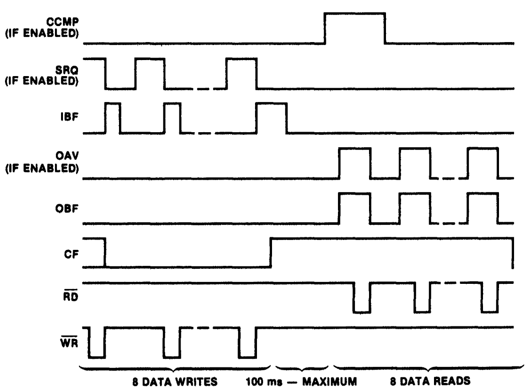
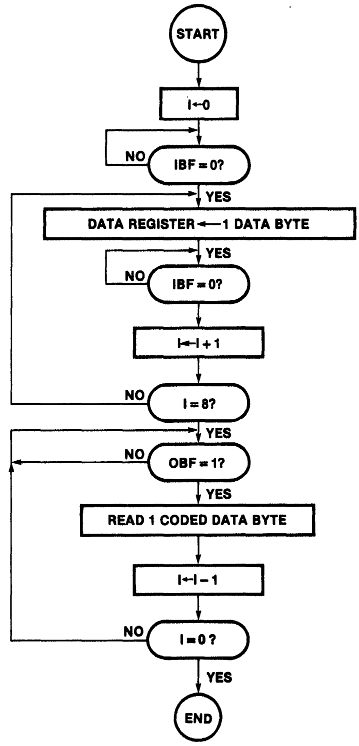


Figure 3. Encrypting/Decrypting Data

Figure 4 offers two flowcharts outlining the alternative means of implementing the data conversion protocol. Either the CF flag or a software counter may be used to end the read and write modes.

SRQ = 1 implies IBF = 0, OAV = 1 implies OBF = 1. This allows interrupt routines to do data transfers without checking status first. However, the OAV service routine must detect and flag the end of a data conversion.



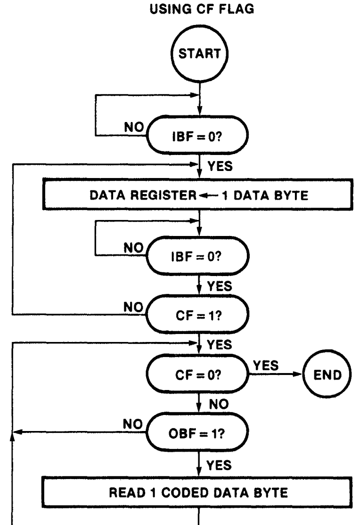


Figure 4. Data Conversion Flowcharts

USING DMA

The timing sequence for data conversions using DMA is shown in Figure 5. This sequence can be better understood when considered in conjunction with the hardware DMA interface in Figure 6. Note that the use of the DMA feature requires 3 external AND gates and 2 DMA channels (one for input, one for output). Since the DEU has only one DMA request pin, the SRQ and OAV outputs are used in conjunction with two of the AND gates to create separate DMA request outputs for the 2 DMA channels. The third AND gate combines the two active-low $\overline{DACK}$ inputs.

To initiate a DMA transfer, the CPU must first initialize the two DMA channels as shown in the flowchart in Figure 7. It must then issue a Set Mode command to the DEU enabling the OAV, SRQ, and DMA outputs. The CCMP interrupt may be enabled or disabled, depending on whether that output is desired. Following the Set Mode command, there must be a data byte giving the number of 8-byte blocks of data (n<256) to be converted. The DEU then generates the required number of DMA requests to the 2 DMA channels with no further CPU intervention. When the requested number of blocks has been converted, the DEU will set CF and assert the CCMP interrupt (if enabled). CCMP then goes false again with the next write to the DEU (command or data). Upon completion of the conversion, the DMA mode is disabled and the DEU returns to the encrypt/decrypt mode. The enabled interrupt outputs, however, will remain enabled until another Set Mode command is issued.

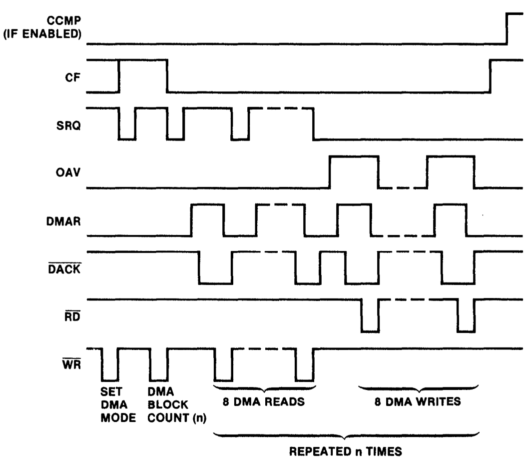


Figure 5. DMA Sequence

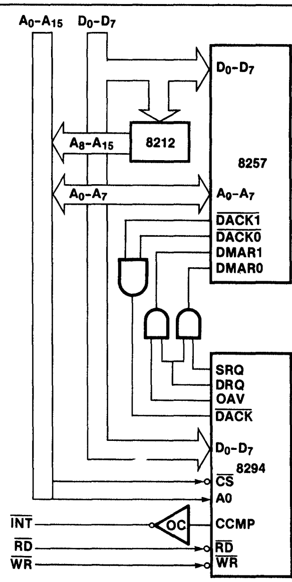


Figure 6. DMA Interface

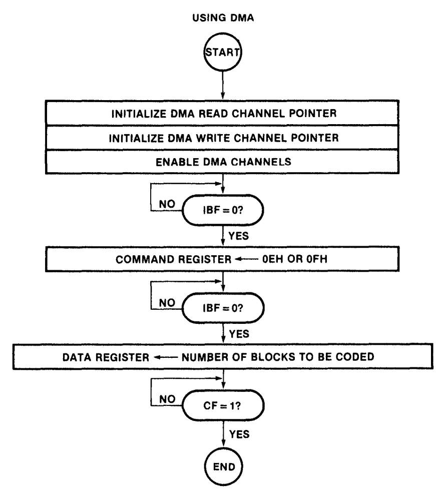


Figure 7. DMA Flowchart

SINGLE BYTE COMMANDS

Figure 8 shows the timing and protocol for single byte commands. Note that any of the commands is effective as a pacify command in that they may be entered at any time, except during a DMA conversion. The DEU is thus set to a known state. However, if a command is issued out of sequence, an additional protocol is required (Figure 9). The CPU must wait until the command is accepted (IBF = 0). A data read must then be issued to clear anything the preceding command sequence may have left in the Data Output Buffer.

CPU/DEU INTERFACES

Figures 10 through 13 illustrate four interface configurations used in the CPU/DEU data transfers. In all cases SRQ will be true (if enabled) and IBF will be false when the DEU is ready to accept data or commands.

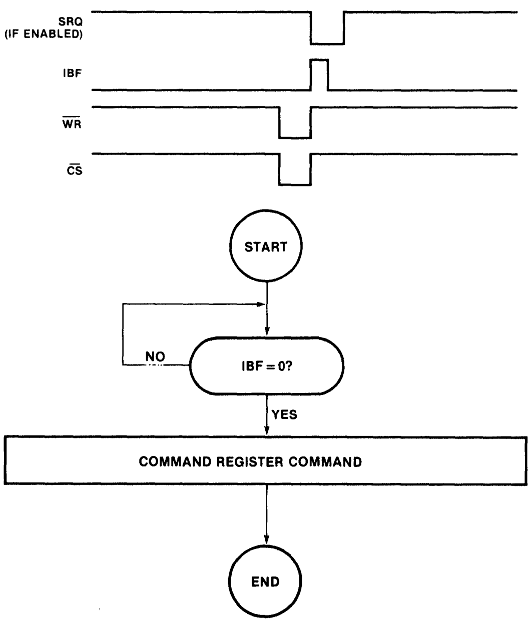


Figure 8. Single Byte Commands

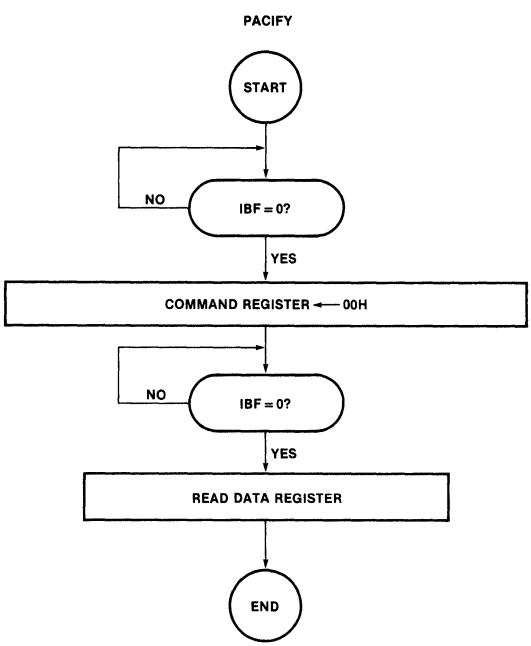


Figure 9. Pacify Protocol

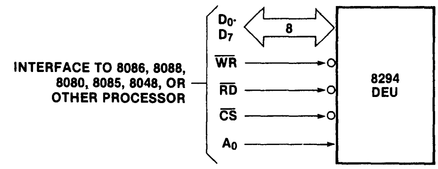


Figure 10. Polling Interface

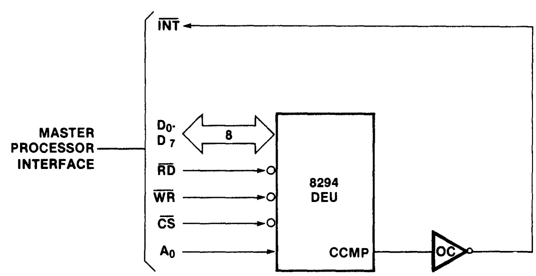


Figure 11. Single Interrupt Interface

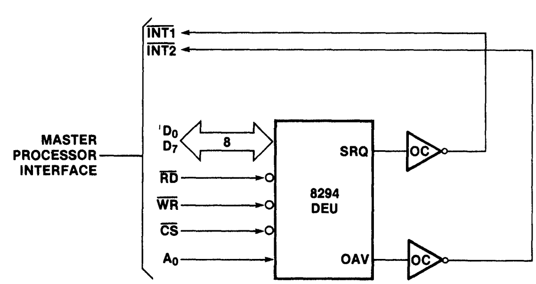
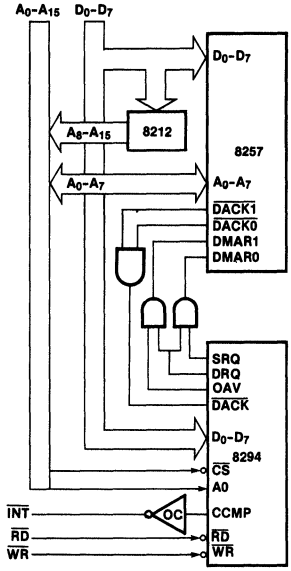


Figure 12. Dual Interrupt Interface

00230B



DMAR0 IS FOR MEMORY TO DEU DATA TRANSFER
DMAR1 IS FOR DEU TO MEMORY DATA TRANSFER
USE OF CCMP IS OPTIONAL

Figure 13. DMA Interface

OSCILLATOR AND TIMING CIRCUITS

The 8294's internal timing generation is controlled by a self-contained oscillator and timing circuit. A choice of crystal, L-C or external clock can be used to derive the basic oscillator frequency.

The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 14.

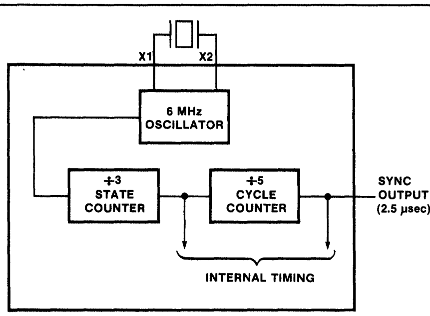
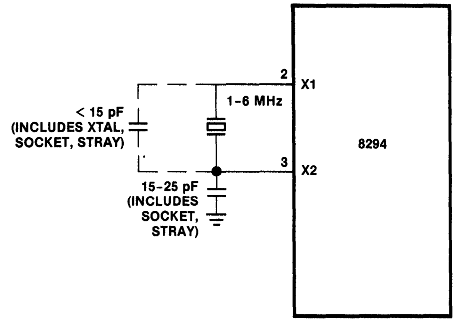


Figure 14. Oscillator Configuration

OSCILLATOR

The on-board oscillator is a series resonant circuit with a frequency range of 1 to 6 MHz. Pins X1 and X2 are input and output (respectively) of a high gain amplifier stage. A crystal or inductor and capacitator connected between X1 and X2 provide the feedback and proper phase shift for oscillation. Recommended connections for crystal or L-C are shown in Figure 15.



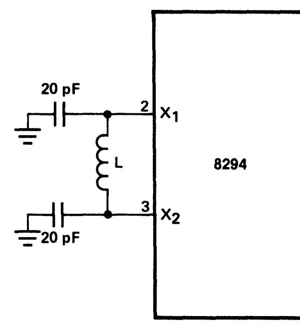


Figure 15. Recommended Crystal and L-C Connections

A recommended range of inductance and capacitance combinations is given below:

L = 120μH corresponds to 3 MHz
L = 45μH corresponds to 5 MHz

An external clock signal can also be used as a frequency reference to the 8294; however, the levels are *not* compatible. The signal must be in the 1 MHz-6 MHz frequency range and must be connected to pins X1 and X2 by buffers with a suitable pull-up resistor to guarantee that a logic "1" is above 3.8 volts. The recommended connection is shown in Figure 16.

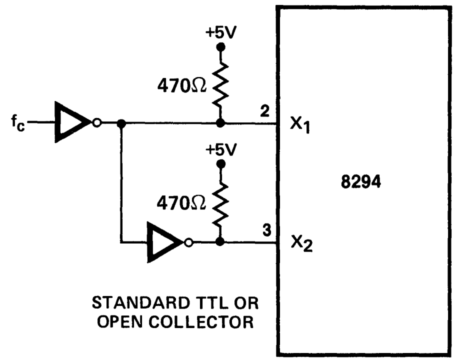


Figure 16. Recommended Connection for External Clock Signal

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .... 0°C to 70°C
Storage Temperature .... −65°C to +150°C
Voltage on Any Pin With
Respect to Ground .... 0.5V to +7V
Power Dissipation .... 1.5 Watt

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

D.C. AND OPERATING CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage (All Except X_1, X_2, $\overline{RESET}$)	−0.5		0.8	V	
V_{IL1}	Input Low Voltage (X_1, X_2, $\overline{RESET}$)	−0.5		0.6	V	
V_{IH}	Input High Voltage (All Except X_1, X_2, $\overline{RESET}$)	2.2		V_{CC}	V	
V_{IH1}	Input High Voltage (X_1, X_2, $\overline{RESET}$)	3.8		V_{CC}	V	
V_{OL}	Output Low Voltage (D_0-D_7)			0.45	V	$I_{OL} = 2.0\,mA$
V_{OL1}	Output Low Voltage (All Other Outputs)			0.45	V	$I_{OL} = 1.6\,mA$
V_{OH}	Output High Voltage (D_0-D_7)	2.4			V	$I_{OH} = -400\,\mu A$
V_{OH1}	Output High Voltage (All Other Outputs)	2.4			V	$I_{OH} = -50\,\mu A$
I_{IL}	Input Leakage Current ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$, $\overline{A_0}$)			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{OZ}	Output Leakage Current (D_0-D_7, High Z State)			±10	μA	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$
I_{DD}	V_{DD} Supply Current		5	15	mA	
$I_{DD} + I_{CC}$	Total Supply Current		60	125	mA	
I_{LI}	Low Input Load Current (Pins 24, 27-38)			0.5	mA	$V_{IL} = 0.8V$
I_{LI1}	Low Input Load Current ($\overline{RESET}$)			0.2	mA	$V_{IL} = 0.8V$

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

DBB READ

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}$ ↓	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}$ ↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	$C_L = 150pF$
t_{RD}	$\overline{RD}$ ↓ to Data Out Delay		225	ns	$C_L = 150pF$
t_{DF}	$\overline{RD}$ ↑ to Data Float Delay		100	ns	
t_{CY}	Cycle Time	2.5	15	μs	6MHz Crystal

DBB WRITE

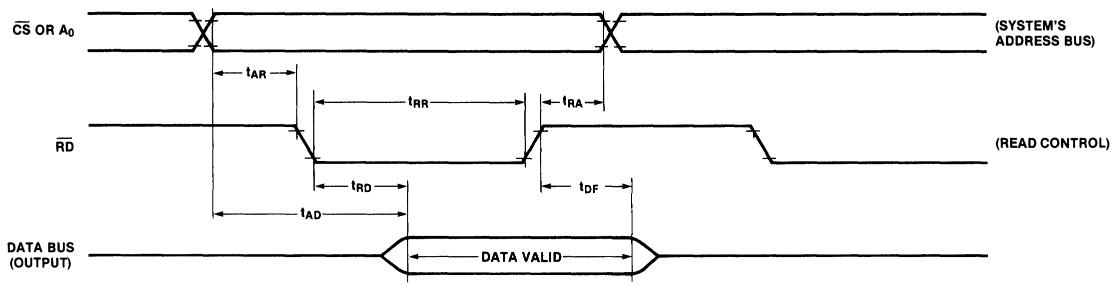
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	$\overline{CS}$, A_0 Setup to $\overline{WR}$ ↓	0		ns	
t_{WA}	$\overline{CS}$, A_0 Hold After $\overline{WR}$ ↑	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$ ↑	150		ns	
t_{WD}	Data Hold to $\overline{WR}$ ↑	0		ns	

DMA AND INTERRUPT TIMING

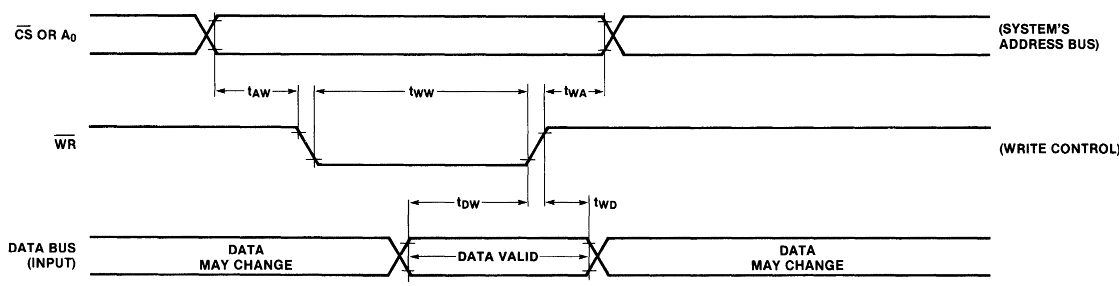
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{ACC}	$\overline{DACK}$ Setup to Control	0		ns	
t_{CAC}	$\overline{DACK}$ Hold After Control	0		ns	
t_{ACD}	$\overline{DACK}$ to Data Valid		225	ns	
t_{CRQ}	Control L.E. to DRQ T.E.		200	ns	
t_{CI}	Control T.E. to Interrupt T.E.		$t_{CY} + 500$	ns	

WAVEFORMS

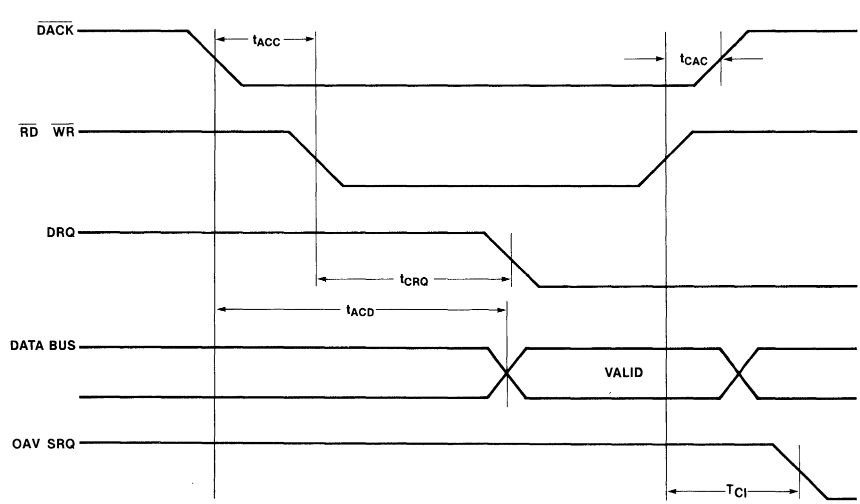
1. READ OPERATION — OUTPUT BUFFER REGISTER.



2. WRITE OPERATION — INPUT BUFFER REGISTER.



DMA AND INTERRUPT TIMING



00230B

intel®

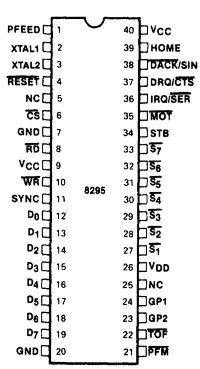
8295
DOT MATRIX PRINTER CONTROLLER

- **Interfaces Dot Matrix Printers to MCS-48™, MCS-80/85™, MCS-86™ Systems**
- **40 Character Buffer On Chip**
- **Serial or Parallel Communication with Host**
- **DMA Transfer Capability**
- **Programmable Character Density (10 or 12 Chararcters/Inch)**
- **Programmable Print Intensity**
- **Single or Double Width Printing**
- **Programmable Multiple Line Feeds**
- **3 Tabulations**
- **2 General Purpose Outputs**

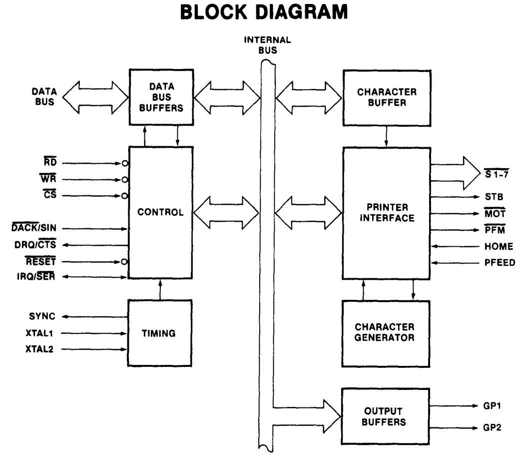
The Intel® 8295 Dot Matrix Printer Controller provides an interface for microprocessors to the LRC 7040 Series dot matrix impact printers. It may also be used as an interface to other similar printers.

The chip may be used in a serial or parallel communication mode with the host processor. In parallel mode, data transfers are based on polling, interrupts, or DMA. Furthermore, it provides internal buffering of up to 40 characters and contains a 7 x 7 matrix character generator accommodating 64 ASCII characters.

PIN CONFIGURATION



BLOCK DIAGRAM



00231B

PIN DESCRIPTION

Name	I/O	Pin #	Description
PFEED	I	1	Paper feed input switch.
XTAL1	I	2	Inputs for a crystal to set internal oscillator frequency. For proper operation use 6 MHz crystal.
XTAL2		3	
$\overline{RESET}$	I	4	Reset input, active low. After reset the 8295 will be set for 12 characters/inch single width printing, solenoid strobe at 320 msec.
NC	—	5	No connection or tied high.
$\overline{CS}$	I	6	Chip select input used to enable the $\overline{RD}$ and $\overline{WR}$ inputs except during DMA.
GND	—	7	This pin must be tied to ground.
$\overline{RD}$	I	8	Read input which enables the master CPU to read data and status. In the serial mode this pin must be tied to V_{CC}.
V_{CC}	—	9	+5 volt power input: +5V ± 10%.
$\overline{WR}$	I	10	Write input which enables the master CPU to write data and commands to the 8295. In the serial mode this pin must be tied to V_{SS}.
SYNC	O	11	2.5 μs clock output. Can be used as a strobe for external circuitry.
D_0	I/O	12	Three-state bidirectional data bus buffer lines used to interface the 8295 to the host processor in the parallel mode. In the serial mode D_0-D_2 sets up the baud rate.
D_1		13	
D_2		14	
D_3		15	
D_4		16	
D_5		17	
D_6		18	
D_7		19	
GND	—	20	This pin must be tied to ground.
V_{CC}	—	40	+5 volt power input: +5V ± 10%.

Name	I/O	Pin #	Description
HOME	I	39	Home input switch, used by the 8295 to detect that the print head is in the home position.
$\overline{DACK}$/SIN	I	38	In the parallel mode used as DMA acknowledgement; in the serial mode, used as input for data.
DRQ/$\overline{CTS}$	O	37	In the parallel mode used as DMA request output pin to indicate to the 8257 that a DMA transfer is requested; in the serial mode used as clear-to-send signal.
IRQ/$\overline{SER}$	O	36	In parallel mode it is an interrupt request input to the master CPU; in serial mode it should be strapped to V_{SS}.
$\overline{MOT}$	O	35	Main motor drive, active low.
STB	O	34	Solenoid strobe output. Used to determine duration of solenoids activation.
$\overline{S_7}$	O	33	Solenoid drive outputs; active low.
$\overline{S_6}$		32	
$\overline{S_5}$		31	
$\overline{S_4}$		30	
$\overline{S_3}$		29	
$\overline{S_2}$		28	
$\overline{S_1}$		27	
V_{DD}	—	26	+5V power input (+5V ± 10%). Low power standby pin.
NC	—	25	No connection.
GP1	O	24	General purpose output pins.
GP2	O	23	
$\overline{TOF}$	I	22	Top of form input, used to sense top of form signal for type T printer.
$\overline{PFM}$	O	21	Paper feed motor drive, active low.

FUNCTIONAL DESCRIPTION

The 8295 interfaces microcomputers to the LRC 7040 Series dot matrix impact printers, and to other similar printers. It provides internal buffering of up to 40 characters. Printing begins automatically when the buffer is full or when a carriage return character is received. It provides a modified 7x7 matrix character generator. The character set includes 64 ASCII characters.

Communication between the 8295 and the host processor can be implemented in either a serial or parallel mode. The parallel mode allows for character transfers into the buffer via DMA cycles. The serial mode features selectable data rates from 110 to 4800 baud.

The 8295 also offers two general purpose output pins which can be set or cleared by the host processor. They can be used with various printers to implement such functions as ribbon color selection, enabling form release solenoid, and reverse document feed.

COMMAND SUMMARY

Hex Code	Description
00	Set GP1. This command brings the GP1 pin to a logic high state. After power on it is automatically set high.
01	Set GP2. Same as the above but for GP2.
02	Clear GP1. Sets GP1 pin to logic low state, inverse of command 00.
03	Clear GP2. Same as above but for GP2. Inverse command 01.
04	Software Reset. This is a pacify command. This command is not effective immediately after commands requiring a parameter, as the Reset command will be interpreted as a parameter.
05	Print 10 characters/in. density.
06	Print 12 characters/in. density.
07	Print double width characters. This command prints characters at twice the normal width, that is, at either 17 or 20 characters per line.
08	Enable DMA mode; must be followed by two bytes specifying the number of data characters to be fetched. Least significant byte accepted first.

Hex Code	Description
09	Tab character.
0A	Line feed.
0B	Multiple Line Feed; must be followed by a byte specifying the number of line feeds.
0C	Top of Form. Enables the line feed output until the Top of Form input is activated.
0D	Carriage Return. Signifies end of a line and enables the printer to start printing.
0E	Set Tab #1, followed by tab position byte.
0F	Set Tab #2, followed by tab position byte. Should be greater than Tab #1.
10	Set Tab #3, followed by tab position byte. Should be greater than Tab #2.
11	Print Head Home on Right. On some printers the print head home position is on the right. This command would enable normal left to right printing with such printers.
12	Set Strobe Width; must be followed by strobe width selection byte. This command adjusts the duration of the strobe activation.

PROGRAMMABLE PRINTING OPTIONS

CHARACTER DENSITY

The character density is programmable at 10 or 12 characters/inch (32 or 40 characters/line). The 8295 is automatically set to 12 characters/inch at power-up. Invoking the Print Double-Width command halves the character density (5 or 6 characters/inch). The 10 char/in or 12 char/in command must be re-issued to cancel the Double-Width mode. Different character density modes may not be mixed within a single line of printing.

PRINT INTENSITY

The intensity of the printed characters is determined by the amount of time during which the solenoid is on. This on-time is programmable via the Set Strobe-Width command. A byte following this command sets the solenoid on-time according to Table 1. Note that only the three least significant bits of this byte are important.

D7—D3	D2	D1	D0	Solenoid On (microsec)
x	0	0	0	200
x	0	0	1	240
x	0	1	0	280
x	0	1	1	320
x	1	0	0	360
x	1	0	1	400
x	1	1	0	440
x	1	1	1	480

Table 1.

TABULATIONS

Up to three tabulation positions may be specified with the 8295. The column position of each tabulation is selected by issuing the Set Tab commands, each followed by a byte specifying the column. The tab positions will then remain valid until new Set Tab commands are issued.

Sending a tab character (09H) will automatically fill the character buffer with blanks up to the next tab position. The character sent immediately after the tab character will thus be stored and printed at that position.

CPU TO 8295 INTERFACE

Communication between the CPU and the 8295 may take place in either a serial or parallel mode. However, the selection of modes is inherent in the system hardware; it is not software programmable. Thus, the two modes cannot be mixed in a single 8295 application.

PARALLEL INTERFACE

Two internal registers on the 8295 are addressable by the CPU: one for input, one for output. The following table describes how these registers are accessed.

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Register
1	0	0	Input Data Register
0	1	0	Output Status Register

Input Data Register—Data written to this register is interpreted in one of two ways, depending on how the data is coded.

1. A command to be executed (0XH or 1XH).
2. A character to be stored in the character buffer for printing (2XH, 3XH, 4XH, or 5XH). See the character set, Table 2.

Output Status Register—8295 status is available in this register at all times.

STATUS BIT:	7	6	5	4	3	2	1	0
FUNCTION:	x	x	PA	DE	x	x	IBF	x

PA—Parameter Required; PA = 1 indicates that a command requiring a parameter has been received. After the necessary parameters have been received by the 8295, the PA flag is cleared.

DE—DMA Enabled; DE = 1 whenever the 8295 is in DMA mode. Upon completion of the required DMA transfers, the DE flag is cleared.

IBF—Input Buffer Full; IBF = 1 whenever data is written to the Input Data Register. No data should be written to the 8295 when IBF = 1.

A flow chart describing communication with the 8295 is shown in Figure 1.

The interrupt request output (IRQ, Pin 36) is available on the 8295 for interrupt driven systems. This output is asserted true whenever the 8295 is ready to receive data.

To improve bus efficiency and CPU overhead, data may be transferred from main memory to the 8295 via DMA cycles. Sending the Enable DMA command (08H) activates the DMA channel of the 8295. This command must be followed by two bytes specifying the length of the data string to be transferred (least significant byte first). The 8295 will then assert the required DMA requests to the 8257 DMA controller without further CPU intervention. Figure 2 shows a block diagram of the 8295 in DMA mode.

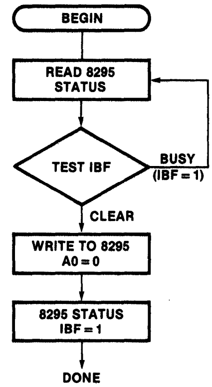


Figure 1. Host to 8295 Protocol Flowchart

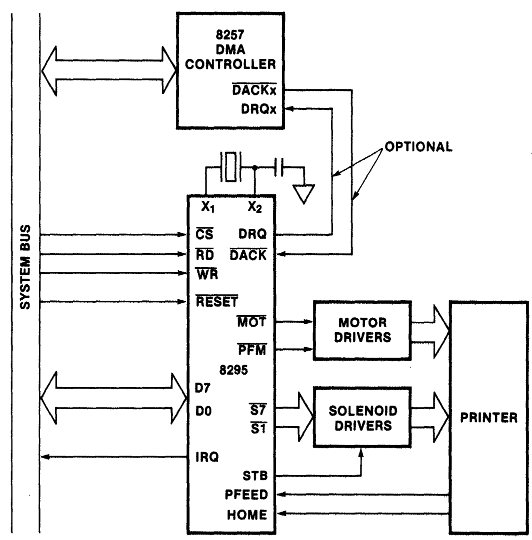


Figure 2. Parallel System Interface

Data transferred in the DMA mode may be either commands or characters or a mixture of both. The procedure is as follows:

1. Set up the 8257 DMA controller channel by sending a starting address and a block length.
2. Set up the 8295 by issuing the "Enable DMA" command (08H) followed by two bytes specifying the block length (least significant byte first).

The DMA enabled flag (DE) will be true until the assigned data transfer is completed. Upon completion of the transfer, the flag is cleared and the interrupt request (IRQ) signal is asserted. The 8295 then returns to the non-DMA mode of operation.

SERIAL INTERFACE

The 8295 may be hardware programmed to operate in a serial mode of communication. By connecting the IRQ/$\overline{SER}$ pin (pin 36) to logic zero, the serial mode is enabled immediately upon power-up. The serial Baud rate is also hardware programmable; by strapping pins 14, 13, and 12 according to Table 2, the rate is selected. $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ must be strapped as shown in Figure 3.

Pin 14	Pin 13	Pin 12	Baud Rate
0	0	0	110
0	0	1	150
0	1	0	300
0	1	1	600
1	0	0	1200
1	0	1	2400
1	1	0	4800
1	1	1	4800

Table 2.

The serial data format is shown in Figure 3. The CPU should wait for a clear to send signal ($\overline{CTS}$) from the 8295 before sending data.

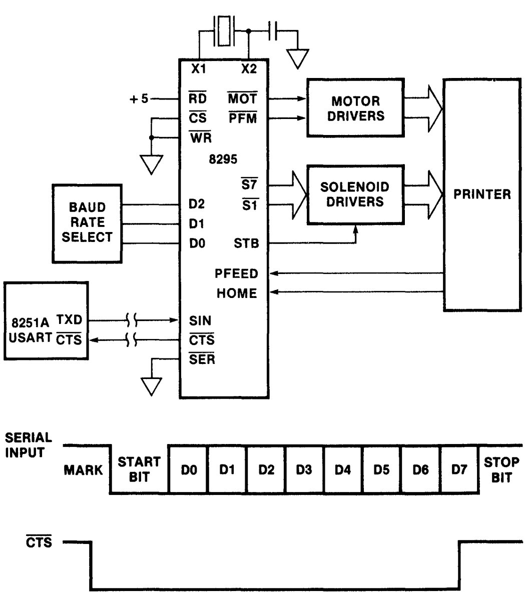


Figure 3. Serial Interface to UART (8251A)

8295 TO PRINTER INTERFACE

The strobe output signal of the 8295 determines the duration of the solenoid outputs, which hold the data to the printer. These solenoid outputs cannot drive the printer solenoids directly. They should be buffered through solenoid drivers as shown in Figure 4. Recommended solenoid and motor driver circuits may be found in the printer manufacturer's interface guide.

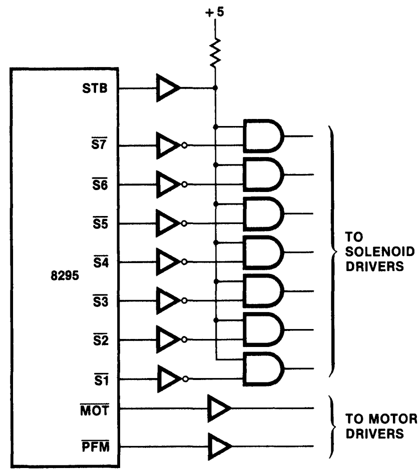


Figure 4. 8295 To Printer Solenoid Interface

OSCILLATOR AND TIMING CIRCUITS

The 8295's internal timing generation is controlled by a self-contained oscillator and timing circuit. A 6 MHz crystal is used to derive the basic oscillator frequency. The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 5. The recommended crystal connection is shown in Figure 6.

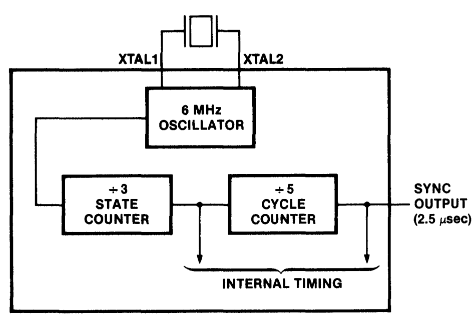


Figure 5. Oscillator Configuration

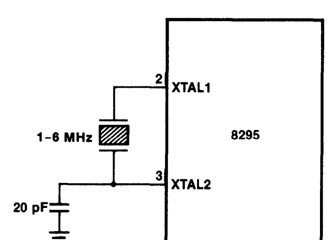


Figure 6. Recommended Crystal Connection

8295 CHARACTER SET

Hex Code	Print Char.	Hex Code	Print Char.	Hex Code	Print Char.	Hex Code	Print Char.
20	space	30	0	40	@	50	P
21	!	31	1	41	A	51	Q
22	"	32	2	42	B	52	R
23	#	33	3	43	C	53	S
24	$	34	4	44	D	54	T
25	%	35	5	45	E	55	U
26	&	36	6	46	F	56	V
27	,	37	7	47	G	57	W
28	(	38	8	48	H	58	X
29	)	39	9	49	I	59	Y
2A	*	3A	:	5A	J	5A	Z
2B	+	3B	;	4B	K	5B	[
2C	'	3C	<	4C	L	5C	\
2D	-	3D	=	4D	M	5D	]
2E	.	3E	>	4E	N	5E	↑
2F	/	3F	?	4F	O	5F	—

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ....0°C to 70°C
Storage Temperature .... −65° to +150°C
Voltage on Any Pin With
Respect to Ground ....0.5V to +7V
Power Dissipation ....1.5 Watt

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

D.C. AND OPERATING CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage (All Except X_1, X_2, $\overline{RESET}$)	−0.5		0.8	V	
V_{IL1}	Input Low Voltage (X_1, X_2, $\overline{RESET}$)	−0.5		0.6	V	
V_{IH}	Input High Voltage (All Except X_1, X_2, $\overline{RESET}$)	2.2		V_{CC}	V	
V_{IH1}	Input High Voltage (X_1, X_2, $\overline{RESET}$)	3.8		V_{CC}	V	
V_{OL}	Output Low Voltage (D_0–D_7)			0.45	V	$I_{OL} = 2.0mA$
V_{OL1}	Output Low Voltage (All Other Outputs)			0.45	V	$I_{OL} = 1.6mA$
V_{OH}	Output High Voltage (D_0–D_7)	2.4			V	$I_{OH} = -400\mu A$
V_{OH1}	Output High Voltage (All Other Outputs)	2.4			V	$I_{OH} = -50\mu A$
I_{IL}	Input Leakage Current ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$, $\overline{A_0}$)			±10	μA	$V_{SS} \leqslant V_{IN} \leqslant V_{CC}$
I_{OZ}	Output Leakage Current (D_0–D_7, High Z State)			±10	μA	$V_{SS} + 0.45 \leqslant V_{IN} \leqslant V_{CC}$
I_{DD}	V_{DD} Supply Current		5	15	mA	
$I_{DD} \div I_{CC}$	Total Supply Current		60	125	mA	
I_{LI}	Low Input Load Current (Pins 24, 27–38)			0.5	mA	$V_{IL} = 0.8V$
I_{LI1}	Low Input Load Current ($\overline{RESET}$)			0.2	mA	$V_{IL} = 0.8V$

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

DBB READ

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}$ ↓	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}$ ↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	$C_L = 150$ pF
t_{RD}	$\overline{RD}$ ↓ to Data Out Delay		225	ns	$C_L = 150$ pF
t_{DF}	$\overline{RD}$ ↑ to Data Float Delay		100	ns	
t_{CY}	Cycle Time	2.5	15	μs	

DBB WRITE

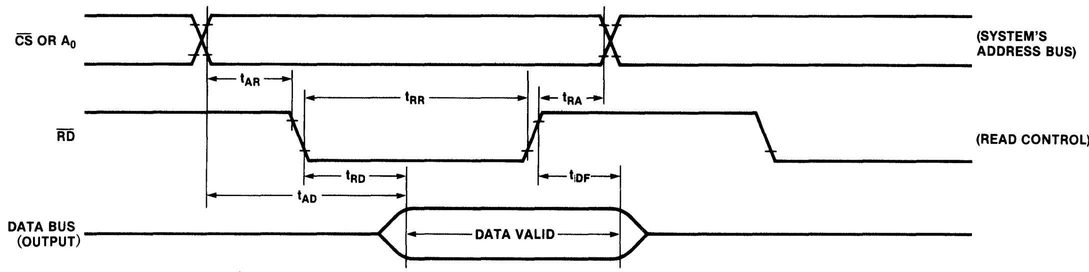
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	$\overline{CS}$, A_0 Setup to $\overline{WR}$ ↓	0		ns	
t_{WA}	$\overline{CS}$, A_0 Hold After $\overline{WR}$ ↑	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$ ↑	150		ns	
t_{WD}	Data Hold to $\overline{WR}$ ↑	0		ns	

DMA AND INTERRUPT TIMING

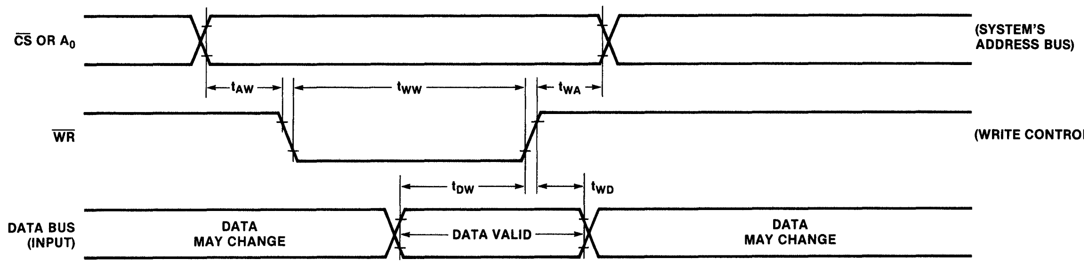
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{ACC}	$\overline{DACK}$ Setup to Control	0		ns	
t_{CAC}	$\overline{DACK}$ Hold After Control	0		ns	
t_{CRQ}	$\overline{WR}$ to DRQ Cleared		200	ns	
t_{ACD}	$\overline{DACK}$ to Data Valid		225	ns	

WAVEFORMS

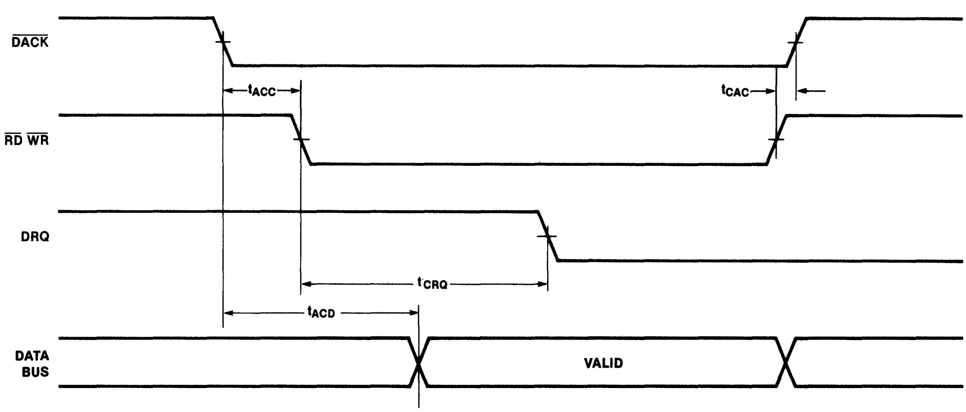
1. READ OPERATION — OUTPUT BUFFER REGISTER.



2. WRITE OPERATION — INPUT BUFFER REGISTER.

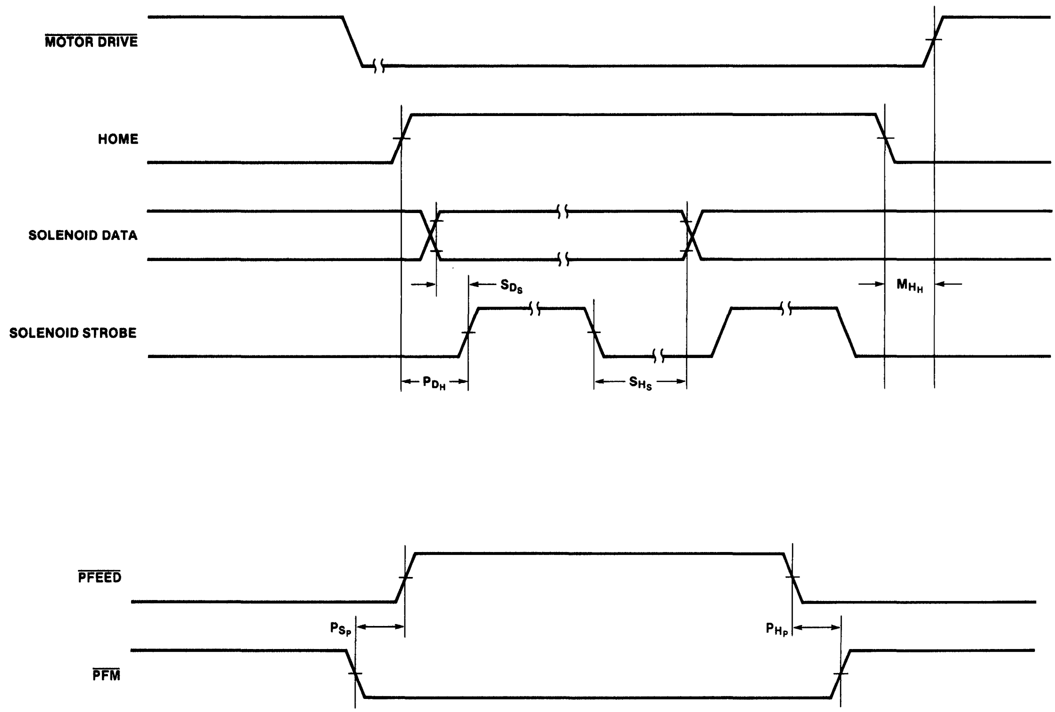


DMA AND INTERRUPT TIMING



00231B

PRINTER INTERFACE TIMING AND WAVEFORMS



Symbol	Parameter	Typical
P_{DH}	Print delay from home inactive	1.8 ms
S_{DS}	Solenoid data setup time before strobe active	25 μs
S_{HS}	Solenoid data hold after strobe inactive	>1 ms
M_{HA}	Motor hold time after home active	3.2 ms
P_{SP}	PFEED setup time after PFM active	58 ms
P_{HP}	PFM hold time after PFEED active	9.75 ms

CHAPTER 8
SYSTEM SUPPORT

8

ICE-41A™
UPI-41A IN-CIRCUIT EMULATOR

Extends Intellec microcomputer development system debug power to user configured system via external cable and 40-pin plug, replacing user UPI-41A™ devices

Emulates user system UPI-41A™ devices in real time

Allows user configured system to use static RAM memory for program debug

Provides hardware comparators for user designated break conditions

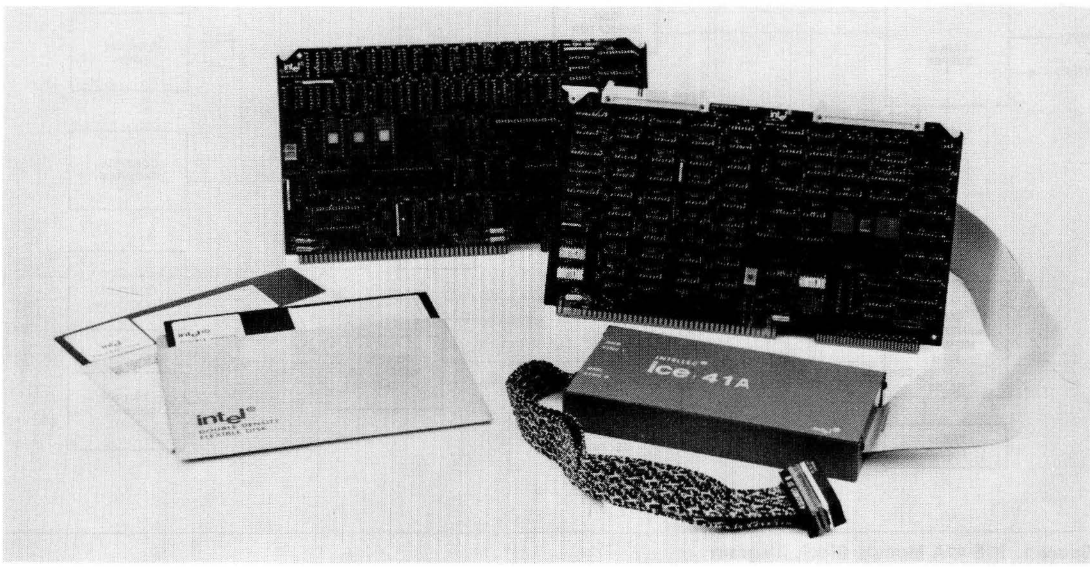
Eliminates need for extraneous debugging tools residing in user system

Collects address, data, and UPI-41A™ status information on machine cycles emulated

Provides capability to examine and alter UPI-41A™ registers, memory, and flag values, and to examine pin and port values

Integrates hardware and software efforts early in engineering cycle to save development time

The ICE-41A UPI-41A In-Circuit Emulator module is an Intellec system resident module that interfaces to any user configured UPI-41A system. The ICE-41A module interfaces with a UPI-41A pin-compatible plug which replaces the UPI-41A device in the system. With the ICE-41A plug in place, the designer has the capability to execute the system in real time while collecting up to 255 instruction cycles of real time trace data. In addition, he can single step the system program during execution. Static RAM memory is available through the ICE-41A module to store UPI-41A programs. The designer may display and alter the contents of program memory, internal UPI-41A registers and flags, and I/O ports. Powerful debug capability is extended into the UPI-41A system while ICE-41A debug hardware and software remain inside the Intellec system. Symbolic reference capability allows the designer to use symbols rather than absolute values when examining and modifying memory, registers, flags, and I/O ports in the system.



FUNCTIONAL DESCRIPTION

Debug Capability Inside User System

Intellec memory is used for the execution of the ICE-41A software. The Intellec CRT console and the file handling capabilities provide the designer with the ability to communicate with the ICE-41A module and display information on the operation of the prototype system. The ICE-41A module block diagram is shown in Figure 1.

Symbolic Debugging

Symbol Table — ICE-41A software allows the user to make symbolic references to I/O ports, memory addresses, and data in his program. The user symbol table which is generated along with the object file during a program assembly can be loaded to Intellec memory for access during emulation. The user may add to this symbol table any additional symbolic values for memory addresses, constants, or variables that he may find useful during system debugging. By referring to symbol memory addresses, the user can examine, change or break at the intended location. In addition, ICE-41A provides symbolic definition of all UPI-41A registers and flags.

Symbolic Reference — Symbolic reference is a great advantage to the system designer. He is no longer burdened with the need to recall or look up addresses of key locations in his program which can change with each assembly. Meaningful symbols from his source program can be used instead. For example, the command:

GO FROM .START TILL CODE. RSLT

begins execution of the program at the address referenced by the label START in the designer's assembly program. A breakpoint is set to occur the first time the microprocessor executes the program memory location referenced by RSLT. The designer does not have to be concerned with the physical locations of START and RSLT. The ICE-41A software driver supplies them automatically from information stored in the symbol table.

Memory Replacement

The 8741/8741A and 8041/8041A contain internal program and data memory. When the UPI-41A microcomputer is replaced by the ICE-41A socket in a system, the ICE-41A module supplies static RAM memory as a replacement for the internal microcomputer memory. The ICE-41A module has enough RAM memory available to emulate up to the total 1K control memory capability of the system.

Real-Time Trace

The ICE-41A module captures trace information while the designer is executing programs in real time. The instructions executed, program counter, port values for port 1 and port 2, and the values of selected UPI-41A status lines are stored for the last 255 instruction cycles executed. When retrieved for display, code is disassembled for user convenience. This provides data for determining how the user system was reacting prior to emulating break.

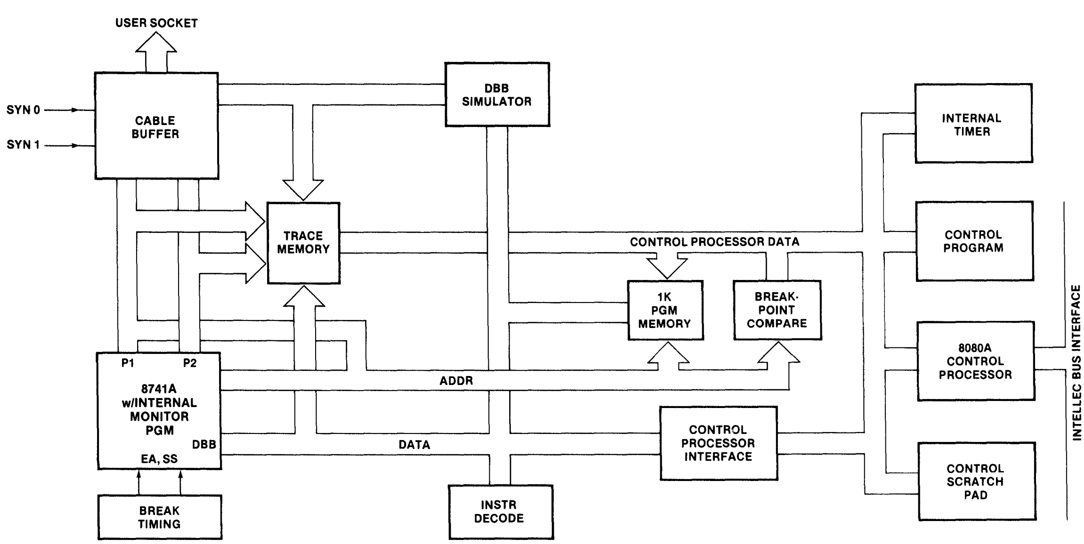


Figure 1. ICE-41A Module Block Diagram

00804A

Integrated Hardware/Software Development

The user prototype systems need no more than a UPI-41A socket and timing logic to begin integration of software and hardware development efforts. Through the ICE-41A module, Intellec system resources can be accessed to replace the prototype system. UPI-41A software development can proceed without the prototype hardware. Hardware designs can be tested using previously tested system software.

Hardware

The ICE-41A module is a microcomputer system utilizing Intel's UPI-41A microprocessor as its nucleus. This system communicates with the Intellec system 8080A processor via direct memory access. Host processor commands and ICE-41A status are interchanged through a DMA channel. ICE-41A hardware consists of two printed circuit boards, the controller board and the emulator board, which reside in the Intellec system chassis. A cable assembly interfaces the ICE-41A module to the user's UPI-41A system. The cable terminates in a UPI-41A pin-compatible plug which replaces any UPI-41A device in the user system.

Controller Board

The ICE-41A module interfaces to the Intellec systems as a peripheral device. The controller board receives commands from the Intellec system and responds through a DMA port. Three 10-bit hardware breakpoint registers are available which can be loaded by the user. While in emulation mode, a hardware comparator is constantly monitoring address lines for a match which will terminate an emulation. The controller board returns real-time trace data, UPI-41A registers, flag and port values, and status information to a control block in the Intellec system when emulation is terminated. This information is available to the user through the ICE-41A interrogation commands. Error conditions, when detected, are automatically displayed on the Intellec system console.

Emulator Board

The emulator board contains the 8741A and peripheral logic required to emulate the UPI-41A device in the user system. A 6 MHz clock drives the emulated UPI-41A device. This clock can be replaced with a user supplied TTL clock in the user system or can be strapped internally for 3 MHz operation.

Cable Card

The cable card is included for cable driving. It transmits address and data bus information to the user system through a 40-pin connector which plugs into the user system in the socket designed for the UPI-41A device.

Software

The ICE-41A software driver is a RAM-based program which provides the user with command language (see Table 1, Table 2, and Table 3) for defining breakpoints, initiating real-time emulation or single step operation, and interrogation and altering user system status recorded during emulation. The ICE-41A command language contains a broad range of modifiers which provide the user with maximum flexibility in defining the operation to be performed. The ICE-41A software driver is available on diskette and operates in 32K of Intellec RAM memory.

Command	Operation
Enable	Activates breakpoint and display registers for use with go and step commands.
Go	Initiates real-time emulation and allows user to specify breakpoints and data retrieval.
Step	Initiates emulation in single instruction increments. Each step is followed by register dump. User may optionally tailor other diagnostic activity to his needs.
Interrupt	Emulates user system interrupt

Table 1. ICE-41A Emulation Commands

Command	Operation
Display	Prints contents of memory, UPI-41A device registers, I/O ports, flags, pins, real-time trace data, symbol table, or other diagnostic data on list device.
Change	Alters contents of memory, register, output port, or flag. Sets or alters breakpoints and display registers.
Base	Establishes mode of display for output data.
Suffix	Establishes mode of display for input data.

Table 2. ICE-41A Interrogation Commands

Command	Operation
Load	Fetches user symbol table and object code from input device.
Save	Sends user symbol table and object code to output device.
Define	Enters symbol name and value to user symbol table.
Move	Moves block of memory data to another area of memory.
Print	Prints user specified portion of trace memory to selected list device.
List	Defines list device.
Exit	Returns program control to ISIS-II.
Evaluate	Converts expression to equivalent values in binary, octal, decimal, and hex.
Remove	Deletes symbols from symbol table.
Reset	Reinitializes ICE-41A hardware.

Table 3. ICE-41A Utility Commands

SPECIFICATIONS

ICE-41A Operating Environment

Required Hardware
Intellec microcomputer development system
System console
Intellec diskette operating system
ICE-41A module

Required Software
System monitor
ISIS-II
ICE-41A diskette-based software

System Clock

Crystal controlled 6.0 MHz or 3.0 MHz internal or user supplied TTL external

Physical Characteristics

Printed Circuit Boards

Width: 12.00 in. (30.48 cm)
Height: 6.75 in. (17.15 cm)
Depth: 0.50 in. (1.27 cm)
Weight: 8.00 lb (3.64 kg)

Cable Buffer Box

Width: 8.00 in. (20.32 cm)
Height: 4.00 in. (10.16 cm)
Depth: 1.25 in. (3.17 cm)
Flat Cable: 4.00 ft (121.92 cm)
User Cable: 15.00 in. (38.10 cm)

Electrical Characteristics

DC Power Requirements

$V_{CC} = +5V, \pm 5\%$
$I_{CC} = 10A$ max; 8A typ
$V_{DD} = +12V, \pm 5\%$
$I_{DD} = 100$ mA max; 60 mA typ
$V_{BB} = -10V$
$I_{BB} = 30$ mA

Input Impedance

@ ICE-41A user socket pins:
$V_{IL} = 0.8V$ max; $I_{IL} = 1.6$ mA
$V_{IH} = 2.0V$ min; $I_{IH} = 40\ \mu A$

@ Bus:
$V_{IL} = 0.8V$ max; $I_{IL} = 250\ \mu A$
$V_{IH} = 2.0V$ min; $V_{IH} = 20\ \mu A$

Output Impedance

@ P1, P2:
$V_{OL} = 0.5V$ max; $I_{OL} = 16$ mA
$V_{OH} = V_{CC}$ (10K pullup)

@ Bus:
$V_{OL} = 0.5V$ max; $I_{OL} = 25$ mA
$V_{OH} = 3.65V$ min; $I_{OH} = 1$ mA

Others

$V_{OL} = 0.5V$ max; $I_{OL} = 16$ mA
$V_{OH} = 2.4V$ max; $I_{OH} = 400\ \mu A$

Equipment Supplied

Controller board
Emulator board
Interface cables and buffer module
Operator's manual
ICE-41A diskette based software

Reference Manuals

9800465 — ICE-41A Operator's Manual (SUPPLIED)

Reference manuals are shipped with each product only if designated SUPPLIED (see above). Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.

ORDERING INFORMATION

Part Number	Description
MDS-41A-ICE	UPI-41A (8741, 8041, 8741A, 8041A) CPU In-circuit emulator, cable assembly and interactive diskette software included

intel®

MULTI-ICE™ SOFTWARE
MULTIPLE-IN-CIRCUIT-EMULATOR

Facilitates software and hardware debugging of multi-processor systems.

Allows two In-Circuit Emulators to operate simultaneously in a single Intellec Microcomputer Development System.

Provides enhanced software features: Symbolic Display of Addresses, Macro Commands, Compound Commands, Software Synchronization of Processes, and INCLUDE File Capability.

Supports In-Circuit Emulator combinations, 85/85 Emulators, 85/49 Emulators (ICE-49™ Emulator supports the design using MCS-48™ chip family), and 85/41A Emulators.

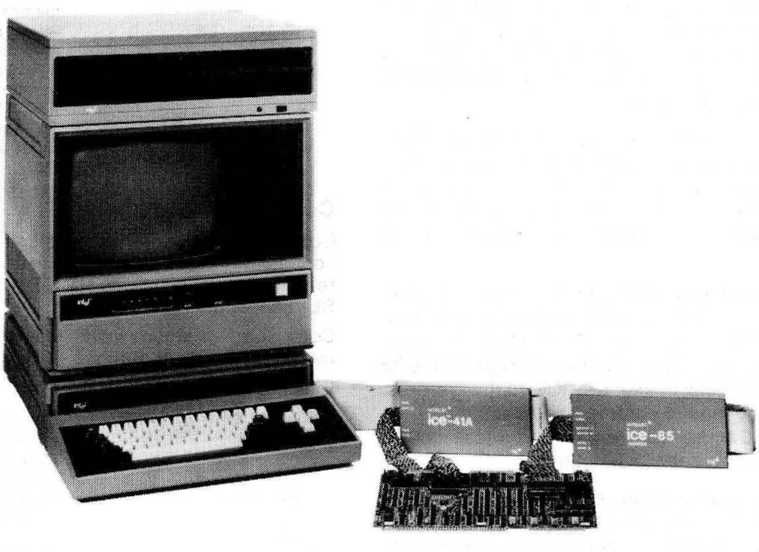
Functions under the supervision of ISIS-II Disk Operating System.

Supports ICE-85™ Emulator Hold Request/Hold Acknowledgement handshake while in both emulation and interrogation modes. (Can be used for Dynamic RAM refresh.)

Multi-ICE In-Circuit Emulator is a software product which allows two Intel In-Circuit Emulators to run simultaneously in a single Intellec Microcomputer Development System. Multi-ICE software used in lieu of the standard ICE software gives users full control of the Intellec Microcomputer Development System, and the two ICE modules for hardware and software debugging of multi-processor systems.

Enhancement features available with Multi-ICE software include a compound command capability which enables the user to "program" a diagnostic or exercise sequence. Also included are repeat and conditional execution of ICE commands, and the ability to invoke the macro commands by name.

A special EPROM set for the ICE-85 Emulator is included. The new firmware will enable the ICE-85 Emulator to support Hold-Request and Hold-Acknowledgement hand-shake protocol both while in emulation and while in interrogation mode. This allows the ICE-85 Emulator to support typical dynamic RAM and DMA applications.

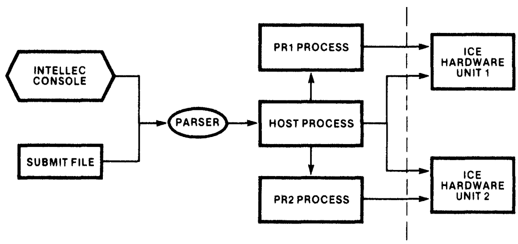


i, ICE, iCS, Insite, Intel, Intellec, iSBC, Library Manager, MCS, Megachassis, Micromap, Multibus, Multimodule, PROMPT, Promware, RMX, UPI, µScope, and the combination of ICE, iCS, iSBC, MCS, or RMX and a numerical suffix are trademarks of Intel Corporation.
© Intel Corporation 1979

MULTI-ICE OPERATION

Multi-ICE software is a debug tool which allows two ICE emulators to begin and stop in sequence. Once started, two ICE emulators emulate simultaneously and independently. Thus, Multi-ICE software permits the debugging of asynchronous or synchronous multiprocessor systems.

A conceptual model for the Multi-ICE software can be illustrated with the following block diagram.



Block Diagram of Multi-ICE™ Operation

There are three processes in the Multi-ICE environment: the Host process and the two ICE processes to control the two ICE hardware modules. The processor for these three processes is the microcomputer in the Intellec Microcomputer Development System. Only the Host process is active when Multi-ICE software is invoked. The Parser interfaces with the console, receives commands from the console or from a file, translates them into intermediate code, and loads the code into the Host command code buffer or ICE command code buffers.

The Host process executes commands from its command code buffer using the execution software and hardware of the Host's current environment, either environment 1 or environment 2 (EN1 or EN2), as required. EN1 and EN2 are the operating environments of the two In-Circuit Emulators.

The user can change the execution environment (from EN1 to EN2 or vice versa) with the SWITCH command. Once the environment is selected, ICE operation is the same as with standard ICE software. In addition, the enhanced software capabilities are available to the user.

The two ICE processes (PR1 and PR2) execute commands from their command code buffers in their own environments (PR1 in EN1 and PR2 in EN2). The main functions of the two ICE execution processes are to control the operations of the two ICE hardware sets. The ACTIVATE command controls the execution of the ICE processes. Commands are passed on to each ICE unit to initiate the desired ICE functions.

The two ICE hardware units accept commands from the Host process or ICE processes. Once emulations start, the two ICE hardware sets will operate until a break condition is met or processing is interrupted by commands from the ICE execution processes.

ENHANCED DIAGNOSTIC SOFTWARE FUNCTIONS

Single ICE™ Module Operation

Multi-ICE software can be used for single ICE operation. The operating procedures will be identical to the Multi-ICE operation. All the enhanced software functions will be available. The performance will be the same as if the standard ICE software is being used.

Symbolic Display of Addresses

The user has the option of displaying a 16-bit address in the form of a symbol name or line number plus a hex number offset.

Macro Command

A macro is a set of commands which is given a name. Thus, a group of commands which is executed frequently may be defined as a macro. Each time the user wants to execute that group of commands, he may just invoke the macro by typing a colon followed by the macro name. Up to ten parameters may be passed to the macro.

Macro commands may be defined at the beginning of a debug session and then can be used throughout the whole session. If the user wants to save the macros for later use, he may use the PUT command to save the macro on diskette, or the user may edit the macro file off-line using the Intellec text editor. Later, the user may use the INCLUDE command to bring in the macro definition file that he created.

Example:

```
*DEFINE MACRO INITMEM      ;This macro clears the
                            memory and then loads the
                            programs.
.*SWITCH = EN1             ;Select environment 1 (ICE
                            Module 1)
.*BYTE 0 TO 100=0          ;Initialize memory to 0.
.*LOAD :F1:DRIVER          ;Load user program into
                            memory for ICE Module 1.
.*SWITCH = EN2             ;Select environment 2 (ICE
                            Module 2)
.*LOAD :F1:DR2             ;Load user program into
                            memory for ICE Module 2.
.*EM                       ;End of Macro
*                          ;To execute this Macro, user
                            types :INITMEM
```

Compound Command

Compound commands provide conditional execution of commands (IF Command) and execution of commands repeatedly until certain conditions are met (COUNT, REPEAT Commands).

Compound commands and Macro commands may be nested any number of times.

Example:

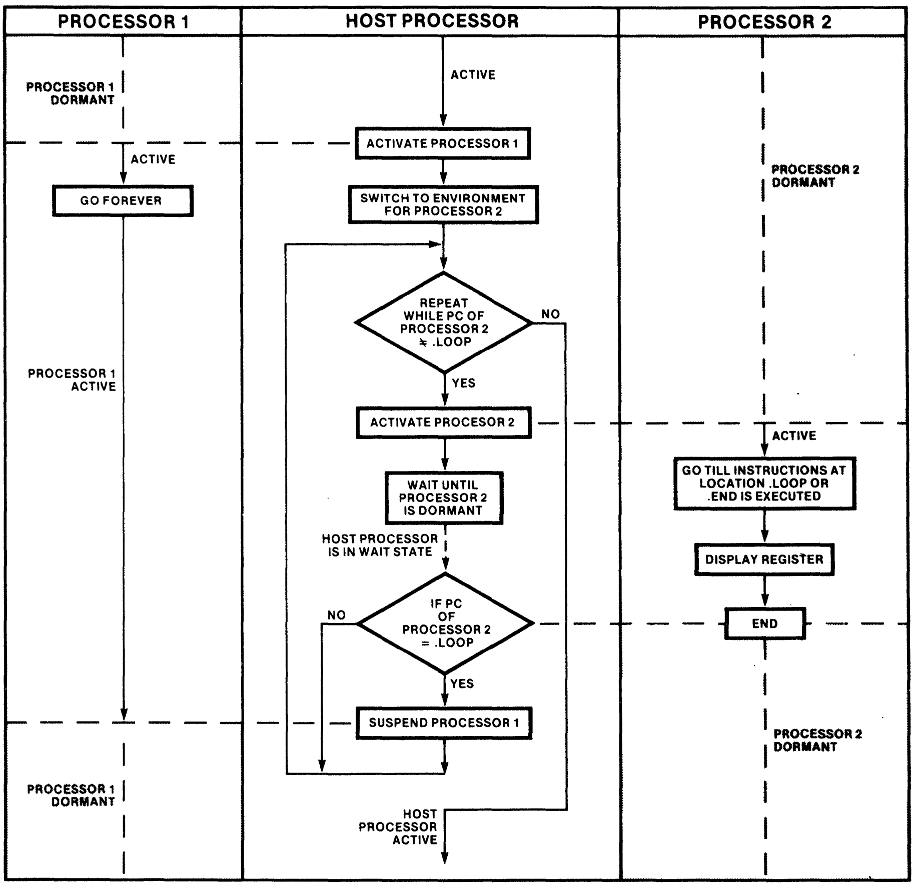
```
*DEFINE .I = 0             ;Define symbol .I to 0
*COUNT 100H                ;Repeat the following
                            commands 100H times.
.*IF .I AND 1 THEN         ;Check if .I is odd
..*BYT .I = .I             ;Fill the memory at location .I
                            to value .I
..*END
.*.I = .I+1                ;Increment .I by 1.
*END                       ;Command executes upon
                            carriage-return after END
```

Software Synchronization of Processes

Up to three processes (Host, PR1 and PR2) can be active simultaneously in the system. An ICE process can be activated (ACTIVATE), suspended (SUSPEND), killed (KILL), or continued (CONTINUE). The Host process can wait for other processes to become dormant before it becomes active again. Through these synchronization commands, the user can create a system test file off-line yet be able to synchronize the three processes when the actual system test is executed.

Example:

The capability of the software synchronization commands is demonstrated by the following example. The flowchart shows the synchronization requirements. The program steps show the actual implementation.



Flowchart of the Example for Demonstrating Multi-ICE™ Synchronization Capability

```
*ACTIVATE PR1                ;Activate PR1
.*GO FROM 800                ;Start ICE Module 1
.*END                        ;End of Activate block
PR1 EMULATION BEGUN
*SWI=EN2                     ;Switch execution Environment to EN2
*REPEAT                      ;Repeat the following block of commands while PC is not equal to .Loop
.*WHILE PC < >.LOOP
.*ACT PR2                    ;Activate PR2
..*GO TILL .LOOP OR .END     ;Go till instruction at location .Loop or at location .END is executed
..*REGISTER                  ;Display the registers
..*END                       ;End of Activate block
.*WAIT PR2                   ;Wait until PR2 is dormant
.*IF PC=.LOOP THEN
..*SUSPEND PR1
..*END                       ;End of IF block
.*END                        ;End of REPEAT block
*
```

INCLUDE File Capability

The INCLUDE command causes input to be taken from the file specified until the end of the file is encountered, at which point, input continues to be taken from the previous source. Nesting of INCLUDES is permitted. Since the command code file can be complex, the ability to edit offline becomes desirable. The INCLUDE command allows the user to pull in command code files and Macro commands created offline which can then be used for the particular debugging session.

Example:

```
*INCLUDE :F1:PROG1          ;Cause input to be taken
                             from file PROG1
*MAP 0 LENGTH 64K=USER      ;Contents of the file PROG1
                             are listed on screen as they
                             are executed.
*MAP IO 0 TO FF = USER
*SWITCH = EN2
*LOAD :F2:LED.HEX
*SWITCH = EN1               ;End of the file PROG1
*                           ;After the end of file is
                             reached, control is returned
                             to console.
```

SPECIFICATIONS

Equipment Supplied:

- Multi-ICE Flexible diskettes
  (one each in single and double density)
  Contains software that supports 85/85 Emulators, 85/49 Emulators, and 85/41A Emulators
- Special EPROM set for one ICE-85 Emulator
- Operator's Manual

MULTI-ICE™ OPERATING ENVIRONMENT

Required Hardware:

Intellec Microcomputer Development System

—Model-800, Model-888

—Series II Model 220, Model 230, and Expansion Chassis

Required Hardware: (Cont'd.)

64K bytes of RAM memory

Flexible disk drive(s)

—Single or double density

System Console

—CRT or hard copy interactive device

ICE-85 Emulator(s), ICE-49 Emulator or ICE-41A Emulator

Optional Hardware:

Printer

Additional flexible disk drives

Required Software:

Intel Systems Implementation Supervisor (ISIS-II)

ORDERING INFORMATION:

Product Code	Description
MDS*-350	Multi-ICE Software

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

intel®

MCS-48™ DISKETTE-BASED SOFTWARE SUPPORT PACKAGE

Extends Intellec® Microcomputer Development System to support MCS-48™ development

Takes advantage of powerful ISIS-II file handling and storage capabilities

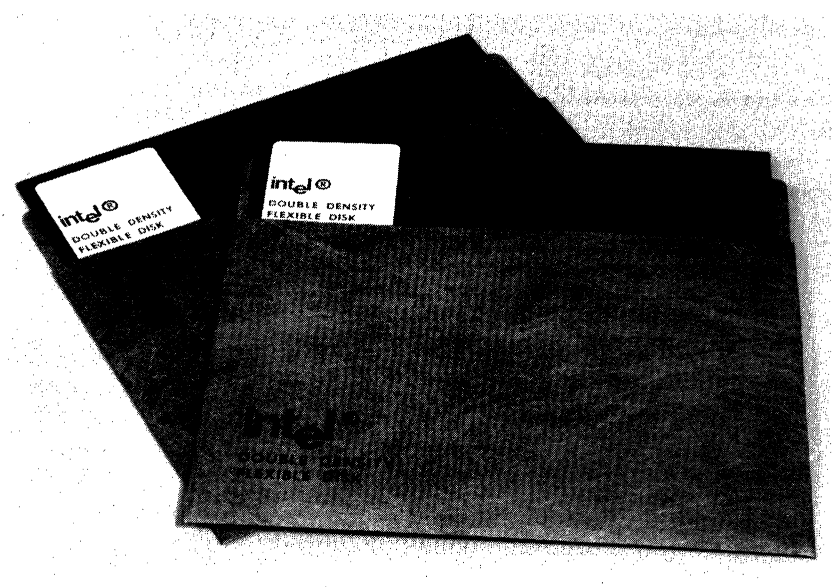
MCS-48 Assembler provides conditional assembly and macro capability

The MCS-48™ Diskette-based Software Support Package (MDS-D48) comes on an Intel® ISIS-II System Diskette and contains the MCS-48 Assembler (ASM48), and the diskette version of the Universal PROM Mapper.

The MCS-48 Assembler (ASM48) translates symbolic 8048 assembly language instructions into the appropriate machine operation codes. In addition to eliminating the errors of hand translation, the ability to refer to program addresses with symbolic names makes it easier to modify programs when adding or deleting instructions. Conditional assembly permits the programmer to specify portions of the master source document which should be included or deleted in variations on a basic system design, such as the code required to handle optional external devices.

Macro capability allows the programmer to define a routine through the use of a single label. ASM48 will assemble the code required by the reserved routine whenever the Macro label is inserted in the text.

Output from the ASM48 is in standard Intel® Hex format. It may be loaded directly to an ICE-48 module for integrated hardware/software debugging. It may also be loaded into the Intellec Development System for 8748 PROM programming using the Universal PROM Programmer.



FUNCTIONAL DESCRIPTION

The MCS-48 assembler translates symbolic 8048 assembly language instructions into the appropriate machine operation codes. The ability to refer to program addresses with symbolic names eliminates the errors of hand translation and makes it easier to modify programs when adding or deleting instructions. Conditional assembly permits the programmer to specify which portions of the master source document should be included or deleted in variations on a basic system design, such as the code required to handle optional external devices. Macro capability allows the programmer use of a single label to define a routine. The MCS-48 assembler will assemble the code required by the reserved routine whenever the macro label is inserted in the text. Output from the assembler is in standard Intel hex format. It may be either loaded directly to an in-circuit emulator (ICE-49) module for integrated hardware/software debugging, or loaded into a Universal PROM Programmer for 8748 PROM programming. A sample assembly listing is shown in Table 1.

```
ISIS-II 8048 MACROASSEMBLER, V1.0                          PAGE 1

LOC   OBJ          SEQ          SOURCE STATEMENT

                    1   ;DECIMAL ADDITION ROUTINE. ADD BCD NUMBER
                    2   ;AT LOCATION 'BETA' TO BCD NUMBER AT 'ALPHA' WITH
                    3   ;RESULT IN 'ALPHA.' LENGTH OF NUMBER IS 'COUNT' DIGIT
                    4   ;PAIRS. (ASSUME BOTH BETA AND ALPHA ARE SAME LENGTH
                    5   ;AND HAVE EVEN NUMBER OF DIGITS OR MSD IS 0 IF
                    6   ;ODD)
                    7   INIT    MACRO   AUGND,ADDND,CNT
                    8           MOV     R0, #AUGND
                    9   L1:     MOV     R1, #ADDND
                   10           MOV     R2, #CNT
                   11           ENDM
                   12   ;
0001E              13   ALPHA   EQU     30
0028               14   BETA    EQU     40
0032               15   COUNT   EQU     5
0100               16           ORG     100H
                   17           INIT    ALPHA, BETA, COUNT
0100  B81E         18+          MOV     R0, #ALPHA
0102  B928         19+ L1:      MOV     R1, #BETA
0104  BA32         20+          MOV     R2, #COUNT
0106  97           21           CLR     C
0107  F0           22   LP:     MOV     A, @R0
0108  71           23           ADDC    A, @R1
0109  57           24           DA      A
010A  A1           25           MOV     @R0, A
010B  18           26           INC     R0
010C  19           27           INC     R1
010D  EA07         28           DJNZ    R2, LP
                                END
USER SYMBOLS
ALPHA  0001E    BETA  0028    COUNT  0005    LP  0107
L1     0102

ASSEMBLY COMPLETE, NO ERRORS


ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE, V1.0             PAGE 1

SYMBOL CROSS REFERENCE

ALPHA  13#   17
BETA   14#   17
COUNT  15#   17
INIT    7#   17
L1     19#
LP     22#   28
```

Table 1. Sample MCS-48 Diskette-Based Assembly Listing

SPECIFICATIONS

Operating Environment

Required Hardware

Intellec Microcomputer Development System
32K RAM (non-macro use)
48K RAM (use of macro facility)
One or two Floppy disk drives
— Single or Double density
System Console
— CRT or interactive hardcopy device

Required Software

ISIS-II Diskette Operating System

Optional Hardware

ICE-49 In-Circuit Emulator
Line Printer
Universal PROM Programmer with 8748 personality card

Shipping Media

Diskette

Reference Manuals

9800255 — MCS-48 and UPI-41 Assembly Language Programming Manual (SUPPLIED)

9800236 — Universal PROM Mapper Operator's Manual

9800306 — ISIS-II User's Guide

Reference manuals are shipped with each product only if designated SUPPLIED (see above). Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.

ORDERING INFORMATION

Product Code	Description
MDS-D48	Diskette-based assembler for MCS-48 family of microprocessors.

MODEL 230
INTELLEC® SERIES II
MICROCOMPUTER DEVELOPMENT SYSTEM

Complete microcomputer development center for Intel 80/85, 8086, and 8048 microprocessor families

LSI electronics board with CPU, RAM, ROM, I/O, and interrupt circuitry

64K bytes RAM memory

Self-test diagnostic capability

Eight-level nested, maskable priority interrupt system

Built-in interfaces for high speed paper tape reader/punch, printer, and universal PROM programmer

Integral CRT with detachable upper/lower case typewriter-style full ASCII keyboard

Powerful ISIS-II Diskette Operating System software with relocating macroassembler, linker, and locater

1 million bytes (expandable to 2.5M bytes) of diskette storage

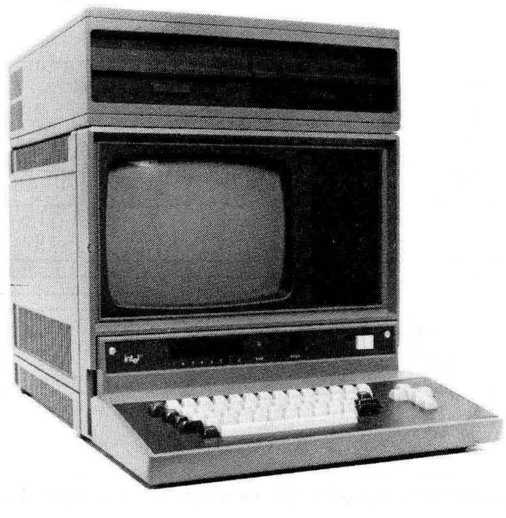
Supports PL/M and FORTRAN high level languages

Standard MULTIBUS™ with multiprocessor and DMA capability

Compatible with standard Intellec/iSBC™ expansion modules

Software compatible with previous Intellec® systems

The Model 230 Intellec Series II Microcomputer Development System is a complete center for the development of microcomputer-based products. It includes a CPU, 64K bytes of RAM, 4K bytes of ROM memory, a 2000-character CRT, a detachable full ASCII keyboard, and dual double density diskette drives providing over 1 million bytes of on-line data storage. Powerful ISIS-II Diskette Operating System software allows the Model 230 to be used quickly and efficiently for assembling and/or compiling and debugging programs for Intel's 80/85, 8086, or 8048 microprocessor families without the need for handling paper tape. ISIS-II performs all file handling operations, leaving the user free to concentrate on the details of his own application. When used in conjunction with an optional in-circuit emulator (ICE) module, the Model 230 provides all the hardware and software development tools necessary for the rapid development of a microcomputer-based product.



00823A

FUNCTIONAL DESCRIPTION

Hardware Components

The Intellec Series II Model 230 is a packaged, highly integrated microcomputer development system consisting of a CRT chassis with a 6-slot cardcage, power supply, fans, cables, and five printed circuit cards. A separate, full ASCII keyboard is connected with a cable. A second chassis contains two floppy disk drives capable of double-density operation along with a separate power supply, fans, and cables for connection to the main chassis. A block diagram of the Model 230 is shown in Figure 1.

CPU Cards — The master CPU card contains its own microprocessor, memory, I/O, interrupt and bus interface circuitry fashioned from Intel's high technology LSI components. Known as the integrated processor board (IPB), it occupies the first slot in the cardcage. A second slave CPU card is responsible for all remaining I/O control including the CRT and keyboard interface. This card, mounted on the rear panel, also contains its own microprocessor, RAM and ROM memory, and I/O interface logic, thus, in effect, creating a dual processor environment. Known as the I/O controller (IOC), the slave CPU card communicates with the IPB over an 8-bit bidirectional data bus.

Memory and Control Cards — In addition, 32K bytes of RAM (bringing the total to 64K bytes) is located on a separate card in the main cardcage. Fabricated from Intel's 16K RAMs, the board also contains all necessary address decoding and refresh logic. Two additional boards in the cardcage are used to control the two double-density floppy disk drives.

Expansion — Two remaining slots in the cardcage are available for system expansion. Additional expansion of 4 slots can be achieved through the addition of an Intellec Series II expansion chassis.

System Components

The heart of the IPB is an Intel NMOS 8-bit microprocessor, the 8080A-2, running at 2.6 MHz. 32K bytes of RAM memory are provided on the board using Intel 16K RAMs. 4K of ROM is provided, preprogrammed with system bootstrap "self-test" diagnostics and the Intellec Series II System Monitor. The eight-level vectored priority interrupt system allows interrupts to be individually masked. Using Intel's versatile 8259 interrupt controller, the interrupt system may be user programmed to respond to individual needs.

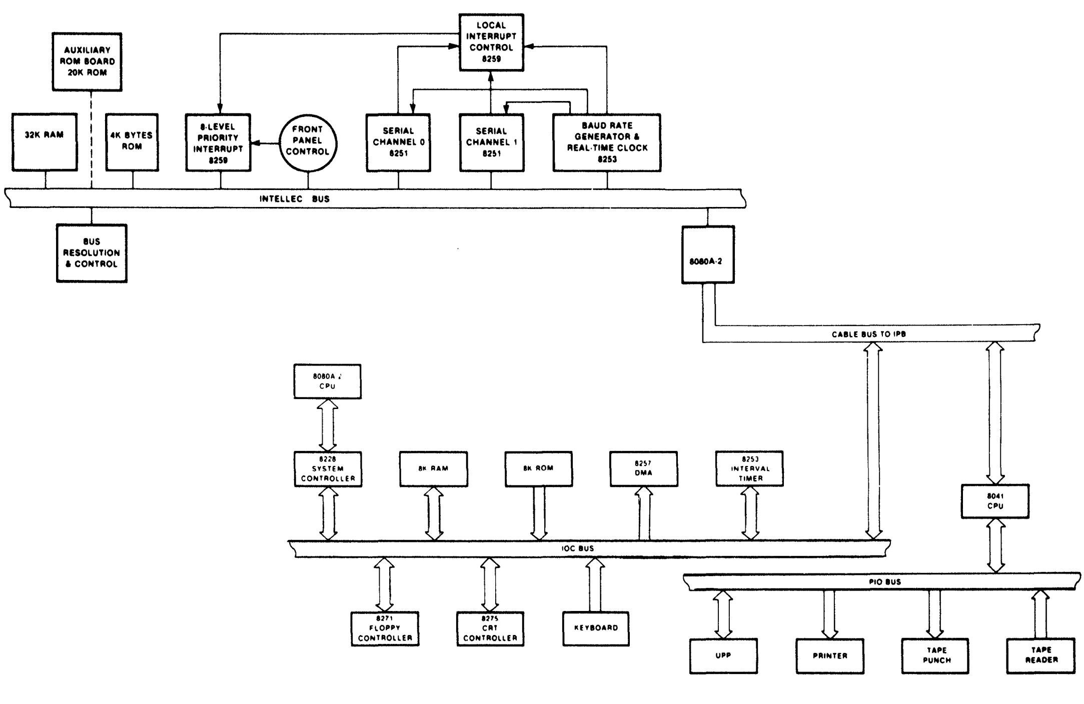


Figure 1. Intellec Series II Model 230 Microcomputer Development System Block Diagram

00823A

Input/Output

IPB Serial Channels — The I/O subsystem in the Model 230 consists of two parts: the IOC card and two serial channels on the IPB itself. Each serial channel is RS232 compatible and is capable of running asynchronously from 110 to 9600 baud or synchronously from 150 to 56K baud. Both may be connected to a user defined data set or terminal. One channel contains current loop adapters. Both channels are implemented using Intel's 8251 USART. They can be programmatically selected to perform a variety of I/O functions. Baud rate selection is accomplished progammatically through an Intel 8253 interval timer. The 8253 also serves as a real-time clock for the entire system. I/O activity through both serial channels is signaled to the system through a second 8259 interrupt controller, operating in a polled mode nested to the primary 8259.

IOC Interface — The remainder of system I/O activity takes place in the IOC. The IOC provides interface for the CRT, keyboard, and standard Intellec peripherals including printer, high speed paper tape reader/punch, and universal PROM programmer. The IOC contains its own independent microprocessor, also an 8080A-2. The CPU controls all I/O operations as well as supervising communications with the IPB. 8K bytes of ROM contain all I/O control firmware. 8K bytes of RAM are used for CRT screen refresh storage. These do not occupy space in Intellec Series II main memory since the IOC is a totally independent microcomputer subsystem.

Integral CRT

Display — The CRT is a 12-inch raster scan type monitor with a 50/60 Hz vertical scan rate and 15.5 kHz horizontal scan rate. Controls are provided for brightness and contrast adjustments. The interface to the CRT is provided through an Intel 8275 single chip programmable CRT controller. The master processor on the IPB transfers a character for display to the IOC, where it is stored in RAM. The CRT controller reads a line at a time into its line buffer through an Intel 8257 DMA controller and then feeds one character at a time to the character generator to produce the video signal. Timing for the CRT control is provided by an Intel 8253 interval timer. The screen display is formatted as 25 rows of 80 characters. The full set of ASCII characters are displayed, including lower case alphas.

Keyboard — The keyboard interfaces directly to the IOC processor via an 8-bit data bus. The keyboard contains an Intel UPI-41 Universal Peripheral Interface, which scans the keyboard, encodes the characters, and buffers the characters to provide N-key rollover. The keyboard itself is a high quality typewriter style keyboard containing the full ASCII character set. An upper/lower case switch allows the system to be used for document preparation. Cursor control keys are also provided.

Peripheral Interface

A UPI-41 Universal Peripheral Interface on the IOC board performs similar functions to the UPI-41 on the PIO board in the Model 210. It provides interface for other standard Intellec peripherals including a printer, high speed paper tape reader, high speed paper tape punch, and universal PROM programmer. Communication between the IPB and IOC is maintained over a separate 8-bit bidirectional data bus. Connectors for the four devices named above, as well as the two serial channels, are mounted directly on the IOC itself.

Control

User control is maintained through a front panel, consisting of a power switch and indicator, reset/boot switch, run/halt light, and eight interrupt switches and indicators. The front panel circuit board is attached directly to the IPB, allowing the eight interrupt switches to connect to the primary 8259, as well as to the Intellec Series II bus.

Diskette System

The Intellec Series II double density diskette system provides direct access bulk storage, intelligent controller, and two diskette drives. Each drive provides ½ million bytes of storage with a data transfer rate of 500,000 bits/second. The controller is implemented with Intel's powerful Series 3000 Bipolar Microcomputer Set. The controller provides an interface to the Intellec Series II system bus, as well as supporting up to four diskette drives. The diskette system records all data in soft sector format. The diskette system is capable of performing seven different operations: recalibrate, seek, format track, write data, write deleted data, read data, and verify CRC.

Diskette Controller Boards — The diskette controller consists of two boards, the channel board and the interface board. These two PC boards reside in the Intellec Series II system chassis and constitute the diskette controller. The channel board receives, decodes and responds to channel commands from the 8080A-2 CPU in the Model 230. The interface board provides the diskette controller with a means of communication with the diskette drives and with the Intellec system bus. The interface board validates data during reads using a cyclic redundancy check (CRC) polynomial and generates CRC data during write operations. When the diskette controller requires access to Intellec system memory, the interface board requests and maintains DMA master control of the system bus, and generates the appropriate memory command. The interface board also acknowledges I/O commands as required by the Intellec bus. In addition to supporting a second set of double density drives, the diskette controller may co-reside with the Intel single density controller to allow up to 2.5 million bytes of on-line storage.

MULTIBUS Capability

All Intellec Series II models implement the industry standard MULTIBUS. MULTIBUS enables several bus masters, such as CPU and DMA devices, to share the bus and memory by operating at different priority levels. Resolution of bus exchanges is synchronized by a bus clock signal derived independently from processor clocks. Read/write transfers may take place at rates up to 5 MHz. The bus structure is suitable for use with any Intel microcomputer family.

SPECIFICATIONS

Host Processor (IPB)

RAM — 64K (system monitor occupies 62K through 64K)
ROM — 4K (2K in monitor, 2K in boot/diagnostic)

Diskette System Capacity (Basic Two Drives)

Unformatted
Per Disk: 6.2 megabits
Per Track: 82.0 kilobits

Formatted
Per Disk: 4.1 megabits
Per Track: 53.2 kilobits

Diskette Performance

Diskette System Transfer Rate — 500 kilobits/sec

Diskette System Access Time
Track-to-Track: 10 ms
Head Settling Time: 10 ms

Average Random Positioning Time — 260 ms

Rotational Speed — 360 rpm
Average Rotational Latency — 83 ms
Recording Mode — M^2FM

Physical Characteristics

Width — 17.37 in. (44.12 cm)
Height — 15.81 in. (40.16 cm)
Depth — 19.13 in. (48.59 cm)
Weight — 73 lb (33 kg)

Keyboard
Width — 17.37 in. (44.12 cm)
Height — 3.0 in. (7.62 cm)
Depth — 9.0 in. (22.86 cm)
Weight — 6 lb (3 kg)

Dual Drive Chassis
Width — 16.88 in. (42.88 cm)
Height — 12.08 in. (30.68 cm)
Depth — 19.0 in. (48.26 cm)
Weight — 64 lb (29 kg)

Electrical Characteristics

DC Power Supply

Volts Supplied	Amps Supplied	Typical System Requirements
+ 5 ± 5%	30	14.25
+ 12 ± 5%	2.5	0.2
− 12 ± 5%	0.3	0.05
− 10 ± 5%	1.5	15
* + 15 ± 5%	1.5	1.3
* + 24 ± 5%	1.7	

*Not available on bus.

AC Requirements — 50/60 Hz, 115/230V AC

Environmental Characteristics

Operating Temperature — 0° to 35°C (95°F)

Equipment Supplied

Model 230 chassis
Integrated processor board (IPB)
I/O controller board (IOC)
32K RAM board
CRT and keyboard
Double density floppy disk controller (2 boards)
Dual drive floppy disk chassis and cables
2 floppy disk drives (512K byte capacity each)
ROM-resident system monitor
ISIS-II system diskette with MCS-80/MCS-85 macroassembler

Reference Manuals

9800558 — A Guide to Microcomputer Development Systems (SUPPLIED)

9800550 — Intellec Series II Installation and Service Guide (SUPPLIED)

9800306 — ISIS-II System User's Guide (SUPPLIED)

9800556 — Intellec Series II Hardware Reference Manual (SUPPLIED)

9800555 — Intellec Series II Hardware Reference Manual (SUPPLIED)

9800301 — 8080/8085 Assembly Language Programming Manual (SUPPLIED)

9800292 — ISIS-II 8080/8085 Assembler Operator's Manual (SUPPLIED)

9800605 — Intellec Series II Systems Monitor Source Listing (SUPPLIED)

9800554 — Intellec Series II Schematic Drawings (SUPPLIED)

Reference manuals are shipped with each product only if designated SUPPLIED (see above). Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.

ORDERING INFORMATION

Part Number	Description
MDS-230	Intellec Series II Model 230 microcomputer development system (110V/60 Hz)
MDS-231	Intellec Series II Model 230 microcomputer development system (220V/50 Hz)

intel®

UPP-103*
UNIVERSAL PROM PROGRAMMER

Replaces UPP-101, UPP-102 Universal PROM Programmers

Intellec® development system peripheral for PROM programming and verification

Provides personality cards for programming all Intel PROM families

Provides zero insertion force sockets for both 16-pin and 24-pin PROMs

Universal PROM mapper software provides powerful data manipulation and programming commands

Provides flexible power source for system logic and programming pulse generation

Holds two personality cards to facilitate programming operations using several PROM types

The UPP-103 Universal PROM Programmer is an Intellec system peripheral capable of programming and verifying all of the Intel programmable ROMs (PROMs). In addition, the UPP-103 programs the PROM memory portions of the 8748 microcomputer, 8741 UPI, the 8755 PROM and I/O chip and the 2920 signal processor. Programming and verification operations are initiated from the Intellec development system console and are controlled by the universal PROM mapper (UPM) program.

FUNCTIONAL DESCRIPTION

Universal PROM Programmer

The basic Universal PROM Programmer (UPP) consists of a controller module, two personality card sockets, a front panel, power supplies, a chassis, and an Intellec development system interconnection cable. An Intel 4040-based intelligent controller monitors the commands from the Intellec System and controls the data transfer interface between the selected PROM personality card and the Intellec memory. A unique personality card contains the appropriate pulse generation functions for each Intel PROM family. Programming and verifying any Intel PROM may be accomplished by selecting and plugging in the appropriate personality card. The front panel contains a power-on switch and indicator, a reset switch, and two zero-force insertion sockets (one 16-pin and one 24-pin or two 24-pin). A central power supply provides power for system logic and for PROM programming pulse generation. The Universal PROM Programmer may be used as a table top unit or mounted in a standard 19-inch RETMA cabinet.

Universal PROM Mapper

The Universal PROM Mapper (UPM) is the software program used to control data transfer between paper tape or diskette files and a PROM plugged into the Universal PROM Programmer. It uses Intellec system memory for intermediate storage. The UPM transfers data in 8-bit HEX, BNPF, or binary object format between paper tape or diskette files and the Intellec system memory. While the data is in Intellec system memory, it can be displayed and changed. In addition, word length, bit position, and data sense can be adjusted as required for the PROM to be programmed. PROMs may also be duplicated or altered by copying the PROM contents into the Intellec system memory. Easy to use program and compare commands give the user complete control over programming and verification operations. The UPM eliminates the need for a variety of personalized PROM programming routines because it contains the programming algorithms for all Intel PROM families. The UPM (diskette based version) is included with the Universal PROM Programmer.

SPECIFICATIONS

Hardware Interface

Data — Two 8-bit unidirectional buses

Commands — 3 write commands, 2 read commands, one initiate command

Physical Characteristics

Width — 6 in. (14.7 cm)

Height — 7 in. (17.2 cm)

Depth — 17 in. (41.7 cm)

Weight — 18 lb (8.2 kg)

Electrical Characteristics

AC Power Requirements — 50–60 Hz; 115/230V AC: 80W

Environmental Characteristics

Operating Temperature — 0°C to 55°C

Optional Equipment

Personality Cards

UPP-816: 2716 personality card

UPP-833: 2732, 2732A personality card

UPP-848: 8748, 8741 personality card with 40-pin adaptor socket

UPP-865: 3602, 3622, 3602A, 3622A, 3621, 3604, 3624, 3604A, 3624A, 3604AL, 3604A-6, 3605, 3605A, 3625, 3625A, 3608, 3628, 3636

UPP-872: 8702A/1702A personality card

UPP-878: 8708/8704/2708/2704 personality card

UPP-955: 8755A personality card with 40-pin adaptor socket

PROM Programming Sockets

UPP-501: 16-pin/24-pin socket pair

UPP-502: 24-pin/24-pin socket pair

UPP-562: Socket adaptor for 3621, 3602, 3622, 3602A, 3622A

UPP-555: Socket adaptor for 3604AL, 3604A-6, 3608, 3628, 3636

UPP-566: Socket adaptor for 3605, 3605A, 3625, 3625A

Equipment Supplied

Cabinet

Power supplies

4040 intelligent controller module

Specified zero insertion force socket pair

Intellec development system interface cable

Universal PROM Mapper program (diskette-based version)

Reference Manuals

9800819 — Universal PROM Programmer User's Manual (SUPPLIED)

ORDERING INFORMATION

Part Number	Description
UPP-103	Universal PROM programmer with 16-pin/24-pin socket pair and 24-pin/24-pin socket pair.

intel

3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080
TWX: 910-338-0026
TELEX: 34-6372

U.S. AND CANADIAN SALES OFFICES

ALABAMA

Intel Corp.
3322 S. Parkway, Ste. 71
Holiday Office Center
Huntsville 35802
Tel: (205) 883-2430

Pen-Tech Associates, Inc.
Holiday Office Center
3322 Memorial Pkwy., S.W.
Huntsville 35801
Tel: (205) 881-9298

ARIZONA

Intel Corp.
10210 N. 25th Avenue, Suite 11
Phoenix 85021
Tel: (602) 997-9695

BFA
4426 North Saddle Bag Trail
Scottsdale 85251
Tel: (602) 994-5400

CALIFORNIA

Intel Corp.
7670 Opportunity Rd.
Suite 135
San Diego 92111
Tel: (714) 268-3563

Intel Corp.*
1651 East 4th Street
Suite 105
Santa Ana 92701
Tel: (714) 835-9642
TWX: 910-595-1114

Intel Corp.*
15335 Morrison
Suite 345
Sherman Oaks 91403
Tel: (213) 986-9510
TWX: 910-495-2045

Intel Corp.*
3375 Scott Blvd.
Santa Clara 95051
Tel: (408) 987-8086
TWX: 910-339-9279
910-338-0255

Earle Associates, Inc.
4617 Ruffner Street
Suite 202
San Diego 92111
Tel: (714) 278-5441

Mac-I
2576 Shattuck Ave.
Suite 4B
Berkeley 94704
Tel: (415) 843-7625

Mac-I
P.O. Box 1420
Cupertino 95014
Tel: (408) 257-9880

Mac-I
11725 Espen Circle
P.O. Box 8763
Fountain Valley 92708
Tel: (714) 839-3341

Mac-I
1321 Centinela Avenue
Suite 1
Santa Monica 90404
Tel: (213) 829-4797

Mac-I
20121 Ventura Blvd., Suite 240E
Woodland Hills 91364
Tel: (213) 347-5900

COLORADO

Intel Corp.*
650 S. Cherry Street
Suite 720
Denver 80222
Tel: (303) 321-8086
TWX: 910-931-2289

Westek Data Products, Inc.
25921 Fern Gulch
P.O. Box 1355
Evergreen 80439
Tel: (303) 674-5255

Westek Data Products, Inc.
1322 Arapahoe
Boulder 80302
Tel: (303) 449-2620

Westek Data Products, Inc.
1228 W. Hinsdale Dr.
Littleton 80120
Tel: (303) 797-0482

CONNECTICUT

Intel Corp.
Peacock Alley
1 Padanaram Road, Suite 146
Danbury 06810
Tel: (203) 792-8366
TWX: 710-456-1199

FLORIDA

Intel Corp.
1001 N.W. 62nd Street, Suite 406
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407

Intel Corp
5151 Adanson Street, Suite 203
Orlando 32804
Tel: (305) 628-2393
TWX: 810-853-9219

FLORIDA (cont.)

Pen-Tech Associates, Inc.
201 S.E. 15th Terrace, Suite K
Deerfield Beach 33441
Tel: (305) 421-4989

Pen-Tech Associates, Inc.
111 So. Maitland Ave., Suite 202
P.O. Box 1475
Maitland 32751
Tel: (305) 645-3444

GEORGIA

Pen Tech Associates, Inc.
Cherokee Center, Suite 21
627 Cherokee Street
Marietta 30060
Tel: (404) 424-1931

ILLINOIS

Intel Corp.*
2250 Golf Road, Suite 815
Rolling Meadows 60008
Tel: (312) 981-7200
TWX: 910-651-5881

Technical Representatives
1502 North Lee Street
Bloomington 61701
Tel: (309) 829-8080

INDIANA

Intel Corp.
2212 Maplecrest Rd.
Ft. Wayne 46815
Tel: (219) 493-2571

Intel Corp.
9101 Wesleyan Road
Suite 204
Indianapolis 46268
Tel: (317) 299-0623

IOWA

Technical Representatives, Inc.
St. Andrews Building
1930 St. Andrews Drive N.E.
Cedar Rapids 52405
Tel: (319) 393-5510

KANSAS

Intel Corp.
9393 W. 110th St., Ste. 265
Overland Park 66210
Tel: (913) 642-8080

Technical Representatives, Inc.
8245 Nieman Road, Suite 100
Lenexa 66214
Tel: (913) 888-0212, 3, & 4
TWX: 910-749-6412

Technical Representatives, Inc.
360 N. Rock Road
Suite 4
Wichita 67206
Tel: (316) 681-0242

KENTUCKY

Lowry & Associates Inc.
Lexington 40593
Tel: (606) 273-3771

MARYLAND

Intel Corp.*
7257 Parkway Drive
Hanover 21076
Tel: (301) 796-7500
TWX: 710-862-1944

Glen White Associates
57 W. Timonium Road, Suite 307
Timonium 21093
Tel: (301) 252-6360

Mesa Inc.
11900 Parklawn Drive
Rockville 20852
Tel: Washington (301) 881-8430
Baltimore (301) 792-0021

MASSACHUSETTS

Intel Corp.*
27 Industrial Ave.
Chelmsford 01824
Tel: (617) 667-8126
TWX: 710-343-6333

EMC Corp.
381 Elliot Street
Newton 02164
Tel: (617) 244-4740
TWX: 922531

MICHIGAN

Intel Corp.*
26500 Northwestern Hwy.
Suite 401
Southfield 48075
Tel: (313) 353-0920
TWX: 910-420-1212
TELEX: 2 31143

Lowry & Associates, Inc.
135 W. North Street
Suite 4
Brighton 48116
Tel: (313) 227-7067

Lowry & Associates, Inc.
3902 Costa NE
Grand Rapids 49505
Tel: (616) 363-9839

MINNESOTA

Intel Corp.
7401 Metro Blvd.
Suite 355
Edina 55435
Tel: (612) 835-6722
TWX: 910-576-2867

Dytec North
1821 University Ave.
Room 163N
St. Paul 55104
Tel: (612) 645-5816

MISSOURI

Intel Corp.
502 Earth City Plaza
Suite 121
Earth City 63045
Tel: (314) 291-1990

Technical Representatives, Inc.
320 Brookes Drive, Suite 104
Hazelwood 63042
Tel: (314) 731-5200
TWX: 910-762-0618

NEW JERSEY

Intel Corp.*
Raritan Plaza
2nd Floor
Raritan Center
Edison 08817
Tel: (201) 225-3000
TWX: 710-480-6238

NEW MEXICO

BFA Corporation
P.O. Box 1237
Las Cruces 88001
Tel: (505) 523-0601
TWX: 910-983-0543

BFA Corporation
3705 Westerfield, N.E.
Albuquerque 87111
Tel: (505) 292-1212
TWX: 910-989-1157

NEW YORK

Intel Corp.*
350 Vanderbilt Motor Pkwy.
Suite 402
Hauppauge 11787
Tel: (516) 231-3300
TWX: 510-227-6236

Intel Corp.
80 Washington St.
Poughkeepsie 12601
Tel: (914) 473-2303
TWX: 510-248-0060

Intel Corp.*
2255 Lyell Avenue
Lower Floor East Suite
Rochester 14606
Tel: (716) 254-6120
TWX: 510-253-7391

Measurement Technology, Inc.
159 Northern Boulevard
Great Neck 11021
Tel: (516) 482-3500

T-Squared
4054 Newcourt Avenue
Syracuse 13206
Tel: (315) 463-8592
TWX: 710-541-0554

T-Squared
2 E. Main
Victor 14564
Tel: (716) 924-9101
TWX: 510-254-8542

NORTH CAROLINA

Intel Corp.
154 Huffman Mill Rd.
Burlington 27215
Tel: (919) 584-3631

Pen-Tech Associates, Inc.
1202 Eastchester Dr.
Highpoint 27260
Tel: (919) 883-9125

Glen White Associates
4009 Barrett Dr.
Raleigh 27609
Tel: (919) 787-7016

OHIO

Intel Corp.*
6500 Poe Avenue
Dayton 45415
Tel: (513) 890-5350
TWX: 810-450-2528

Intel Corp.*
Chagrin-Brainard Bldg., No. 210
28001 Chagrin Blvd.
Cleveland 44122
Tel: (216) 464-2736

Lowry & Associates Inc.
1440 Snow Road
Suite 216
Cleveland 44134
Tel: (216) 398-0506

Lowry & Associates, Inc.
2194 Hewitt Avenue
Dayton 45440
Tel: (513) 429-9040

OHIO (cont.)

Lowry & Associates, Inc.
Columbus 43224
Tel: (614) 436-2051

OREGON

Intel Corp.
10700 S.W. Beaverton
Hillsdale Highway
Suite 324
Beaverton 97005
Tel: (503) 641-8086

PENNSYLVANIA

Intel Corp.*
275 Commerce Dr.
200 Office Center
Suite 300
Fort Washington 19034
Tel: (215) 542-9444
TWX: 510-661-2077

Lowry & Associates, Inc.
Seven Parkway Center
Suite 455
Pittsburgh 15520
Tel: (412) 922-5110

Q.E.D. Electronics
300 N. York Road
Hatboro 19040
Tel: (215) 674-9600

TEXAS

Intel Corp.*
2925 L.B.J. Freeway
Suite 175
Dallas 75234
Tel: (214) 241-9521
TWX: 910-860-5487

Intel Corp.*
6420 Richmond Ave.
Suite 280
Houston 77057
Tel: (713) 784-3400

Industrial Digital Systems Corp.
5925 Sovereign
Suite 101
Houston 77036
Tel: (713) 988-9421

Intel Corp.
313 E. Anderson Lane
Suite 314
Austin 78752
Tel: (512) 454-3628

VIRGINIA

Glen White Associates
Route 2, Box 193
Charlottesville 22901
Tel: (804) 295-7686

Glen White Associates
P.O. Box 10186
Lynchburg 24506
Tel: (804) 384-6920

Glen White Associates
Route 1, Box 322
Colonial Beach 22442
Tel: (804) 224-7764

WASHINGTON

Intel Corp.
Suite 114, Bldg. 3
1603 116th Ave. N.E.
Bellevue 98005
Tel: (206) 453-8086

WISCONSIN

Intel Corp.
150 S. Sunnyslope Rd.
Brookfield 53005
Tel: (414) 784-9060

CANADA

Intel Semiconductor Corp.
Suite 233, Bell Mews
39 Highway 7, Bells Corners
Ottawa, Ontario K2H 8R2
Tel: (613) 829-9714
TELEX: 053-4115

Intel Semiconductor Corp.
50 Galaxy Blvd.
Unit 12
Rexdale, Ontario
M9W 4Y5
Tel: (416) 675-2105
TELEX: 06983574

Multilek, Inc.*
15 Grenfell Crescent
Ottawa, Ontario K2G 0G3
Tel: (613) 226-2365
TELEX: 053-4585

Multilek, Inc.
Toronto
Tel: (416) 245-4622

Multilek, Inc.
Montreal
Tel: (514) 481-1350

*Field Application Location

intel
3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080
TWX: 910-338-0026
TELEX: 34-6372

INTERNATIONAL SALES AND MARKETING OFFICES

INTEL® MARKETING OFFICES

AUSTRALIA
Intel Australia
Suite 2, Level 15, North Point
100 Miller Street
North Sydney, NSW, 2060
Tel: 450-847
TELEX: AA 20097

BELGIUM
Intel Corporation S.A.
Rue du Moulin a Papier 51
Boite 1
B-1160 Brussels
Tel: (02) 660 30 10
TELEX: 24814

DENMARK
Intel Denmark A/S*
Lyngbyvej 32 2nd Floor
DK-2100 Copenhagen East
Tel: (01) 18 20 00
TELEX: 19567

FINLAND
Intel Scandinavia
Sentnerikuja 3
SF - 00400 Helsinki 40
Tel: (0) 558531
TELEX: 123 332

FRANCE
Intel Corporation, S.A.R.L.*
5 Place de la Balance
Silic 223
94528 Rungis Cedex
Tel: (01) 687 22 21
TELEX: 270475

GERMANY
Intel Semiconductor GmbH*
Seidlstrasse 27
8000 Muenchen 2
Tel: (089) 53 891
TELEX: 523 177

Intel Semiconductor GmbH
Mainzer Strasse 75
6200 Wiesbaden 1
Tel: (06121) 700874
TELEX: 04186183

Intel Semiconductor GmbH
Wernerstrasse 67
P.O. Box 1460
7012 Fellbach
Tel: (0711) 580082
TELEX: 7254826

Intel Semiconductor GmbH
Hindenburgstrasse 28/29
3000 Hannover 1
Tel: (0511) 852051
TELEX: 923625

HONG KONG
Intel Trading Corporation
99-105 Des Voeux Rd., Central
18F, Unit B
Hong Kong
Tel: 5-450-847
TELEX: 63869

ISRAEL
Intel Semiconductor Ltd.*
P.O. Box 2404
Haifa
Tel: 972/452 4261
TELEX: 92246511

ITALY
Intel Corporation Italia, S.p.A.
Corso Sempione 39
I-20145 Milano
Tel: 2/34.93287
TELEX: 311271

JAPAN
Intel Japan K.K.*
Flower Hill-Shinmachi East Bldg.
1-23-9, Shinmachi, Setagaya-ku
Tokyo 154
Tel: (03) 426-9261
TELEX: 781-28426

NETHERLANDS
Intel Semiconductor B.V.
Cometongebouw
Westblaak 106
3012 Km Rotterdam
Tel: (10) 149122
TELEX: 22283

NORWAY
Intel Norway A/S
P.O. Box 158
N-2040
Klofta, Norway
Tel: 47 2/981068
TELEX: 18018

SWEDEN
Intel Sweden A.B.*
Box 20092
Enighetsvagen, 5
S-16120 Bromma
Tel: (08) 98 53 90
TELEX: 12261

SWITZERLAND
Intel Semiconductor A.G.
Forchastrasse 95
CH 8032 Zurich
Tel: 1-55 45 02
TELEX: 557 89 ich ch

UNITED KINGDOM
Intel Corporation (U.K.) Ltd.*
Broadfield House
4 Between Towns Road
Cowley, Oxford OX4 3NB
Tel: (0865) 77 14 31
TELEX: 837203

Intel Corporation (U.K.) Ltd.
5 Hospital Street
Nantwich, Cheshire CW5 5RE
Tel: (0270) 62 65 60
TELEX: 36620

INTERNATIONAL DISTRIBUTORS/REPRESENTATIVES

ARGENTINA
Micro Sistemas S.A.
9 De Julio 561
Cordoba
Tel: 54-51-32-880
TELEX: 51837 BICCO

AUSTRALIA
A.J.F. Systems & Components Pty. Ltd.
310 Queen Street
Melbourne
Victoria 3000
Tel:
TELEX:

Warburton Franki
Corporate Headquarters
372 Eastern Valley Way
Chatswood, New South Wales 2067
Tel: 407-3261
TELEX: AA 21299

AUSTRIA
Bacher Elektronische Geraete GmbH
Rotenmulgasse 26
A 1120 Vienna
Tel: (0222) 83 63 96
TELEX: (01) 1532

Rekirsch Elektronik Geraete GmbH
Lichtensteinstrasse 97
A1090 Vienna
Tel: (222) 347646
TELEX: 74759

BELGIUM
Inelco Belgium S.A.
Avenue Val Duchesse, 3
B-1160 Brussels
Tel: (02) 660 00 12
TELEX: 25441

BRAZIL
Icotron S.A.
0511-Av. Mutinga 3650
6 Andar
Pirituba-Sao Paulo
Tel: 261-0211
TELEX: (011) 222 ICO BR

CHILE
DIN
Av. Vic. Mc kenna 204
Casilla 6055
Santiago
Tel: 227 564
TELEX: 3520003

CHINA
C.M. Technologies
525 University Avenue
Suite A-40
Palo Alto, CA 94301

COLOMBIA
International Computer Machines
Adpo. Aereo 19403
Bogota 1
Tel: 232-6635
TELEX: 43439

CYPRUS
Cyprus Eltrom Electronics
P.O. Box 5393
Nicosia
Tel: 21-27982

DENMARK
STL-Lyngso Komponent A/S
Ostmarken 4
DK-2860 Soborg
Tel: (01) 67 00 77
TELEX: 22990

Scandinavian Semiconductor
Supply A/S
Nannasgade 18
DK-2200 Copenhagen
Tel: (01) 83 50 90
TELEX: 19037

FINLAND
Oy Fintronic AB
Melkonkatu 24 A
SF-00210
Helsinki 21
Tel: 0-692 6022
TELEX: 124 224 Ftron SF

FRANCE
Celdis S.A.*
53, Rue Charles Frerot
94250 Gentilly
Tel: (1) 581 00 20
TELEX: 200 485

Feutrier
Rue des Trois Glorieuses
F-42270 St. Priest-en-Jarez
Tel: 167 7746733

Metrologie*
La Tour d'Asnières
4, Avenue Laurent Cely
92606-Asnières
Tel: 791 44 44
TELEX: 611 448

Tekelec Airtronic*
Cite des Bruyeres
Rue Carle Vernet
92310 Sevres
Tel: (1) 534 75 35
TELEX: 204552

GERMANY
Electronic 2000 Vertriebs GmbH
Neumarkter Strasse 75
D-8000 Munich 80
Tel: (089) 434061
TELEX: 522561

Jermyn GmbH
Postfach 1180
D-6277 Camberg
Tel: (06434) 231
TELEX: 484426

Kontron Elektronik GmbH
Breslauerstrasse 2
8057 Eching B
D-8000 Munich
Tel: (89) 319.011
TELEX: 522122

Neye Enatechnik GmbH
Schillerstrasse 14
D-2085 Quickborn-Hamburg
Tel: (04106) 6121
TELEX: 02-13590

GREECE
American Technical Enterprises
P.O. Box 156
Athens
Tel: 30-1-8811271
30-1-8219470

HONG KONG
Schmidt & Co.
28/F Wing on Center
Connaught Road
Hong Kong
Tel: 5-455-644
TELEX: 74766 Schmc Hx

INDIA
Micronic Devices
104/109C, Nirmal Industrial Estate
Sion (E)
Bombay 400022, India
Tel: 486-170
TELEX: 011-5947 MDEV IN

ISRAEL
Eastronics Ltd.*
11 Rozanis Street
P.O. Box 39300
Tel Aviv 61390
Tel: 475151
TELEX: 33638

ITALY
Eledra 3S S.P.A.*
Viale Elvezia, 18
I 20154 Milan
Tel: (02) 34.93.041-31.85.441
TELEX: 332332

JAPAN
Asahi Electronics Co. Ltd.
KMM Bldg. Room 407
2-14-1 Asano, Kokura
Kita-Ku, Kitokyushu City 802
Tel: (093) 511-6471
TELEX: AECKY 7126-16

Hamilton-Avnet Electronics Japan Ltd.
YU and YOU Bldg. 1-4 Horidome-Cho
Nihonbashi
Tel: (03) 662-9911
TELEX: 2523774

Nippon Micro Computer Co. Ltd.
Mutsumi Bldg. 4-5-21 Kojimachi
Chiyoda-ku, Tokyo 102
Tel: (03) 230-0041

Ryoyo Electric Corp.
Konwa Bldg.
1-12-22, Tsukiji, 1-Chome
Chuo-Ku, Tokyo 104
Tel: (03) 543-7711

Tokyo Electron Ltd.
No. 1 Higashikata-Machi
Midori-Ku, Yokohama 226
Tel: (045) 471-8811
TELEX: 781-4473

KOREA
Koram Digital
Room 411 Ahil Bldg.
49-4 2-GA Hoehyun-Dong
Chung-Ku Seoul
Tel: 23-8123
TELEX: K23542 HANSINT

Leewood International, Inc.
C.P.O. Box 4046
112-25, Sokong-Dong
Chung-Ku, Seoul 100
Tel: 28-5927
CABLE: "LEEWOOD" Seoul

NETHERLANDS
Inelco Nether. Comp. Sys. BV
Turfstekerstraat 63
Aalsmeer 1431 D
Tel: (2977) 28855
TELEX: 14693

Koning & Hartman
Koperwerf 30
2544 EN Den Haag
Tel: (70) 210.101
TELEX: 31528

NEW ZEALAND
W. K. McLean Ltd.
P.O. Box 18-065
Glenn Innes, Auckland, 6
Tel: 587-037
TELEX: NZ2763 KOSFY

NORWAY
Nordisk Elektronik (Norge) A/S
Postoffice Box 122
Smedsvingen 4
1364 Hvalstad
Tel: 02 78 62 10
TELEX: 17546

PORTUGAL
Ditram
Componentes E Electronica LDA
Av. Miguel Bombarda, 133
Lisboa 1
Tel: (19) 545313
TELEX: 14347 GESPIC

SINGAPORE
General Engineers Associates
Blk 3, 1003-1008, 10th Floor
P.S.A. Multi-Storey Complex
Telok Blangah/Pasir Panjang
Singapore 5
Tel: 271-3163
TELEX: RS23987 GENERCO

SOUTH AFRICA
Electronic Building Elements
Pine Square
18th Street
Hazelwood, Pretoria 0001
Tel: 789 221
TELEX: 30181SA

SPAIN
Interface
Av. Generalisimo 51 9°
E-Madrid 16
Tel: 456 3151

ITT SESA
Miguel Angel 16
Madrid 10
Tel: (1) 4190957
TELEX: 27707/27461

SWEDEN
AB Gosta Backstrom
Box 12009
10221 Stockholm
Tel: (08) 541 080
TELEX: 10135

Nordisk Electronik AB
Box 27301
S-10254 Stockholm
Tel: (08) 635040
TELEX: 10547

SWITZERLAND
Industrade AG
Gemsenstrasse 2
Postcheck 80 - 21190
CH-8021 Zurich
Tel: (01) 60 22 30
TELEX: 56788

TAIWAN
Taiwan Automation Co.*
3d Floor #75, Section 4
Nanking East Road
Taipei
Tel: 771-0940
TELEX: 11942 TAIAUTO

TURKEY
Turkelek Electronics
Apapurk Boulevard 169
Ankara
Tel: 189483

UNITED KINGDOM
Comway Microsystems Ltd.
Market Street
68-Bracknell, Berkshire
Tel: 44-34451654

G.E.C. Semiconductors Ltd.
East Lane
North Wembley
Middlesex HA9 7PP
Tel: (01) 904-9303/908-4111
TELEX: 28817

Jermyn Industries
Vestry Estate
Sevenoaks, Kent
Tel: (0732) 501.44
TELEX: 95142

Rapid Recall, Ltd.
6 Soho Mills Ind. Park
Woburn Green
Bucks, England
Tel: (6285) 24961
TELEX: 849439

Sintrom Electronics Ltd.*
Arkwright Road 2
Reading, Berkshire RG2 0LS
Tel: (0734) 85464
TELEX: 847395

VENEZUELA
Componentes y Circuitos
Electronicos TTLCA C.A.
Apartado 3223
Caracas 101
Tel: 718-100
TELEX: 21795 TELETIPOS

*Field Application Location

Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051
(408) 987-8080

Intel Corporation
Rue du Moulin à Papier 51, Boite 1,
B-1160 Brussels, Belgium

Intel Corporation
Flower Hill-Shinmachi East Bldg.
1-23-9, Shinmachi, Setagayu-ku
Tokyo 154, Japan

Printed in U.S.A./T-196/480/23K/UP DK