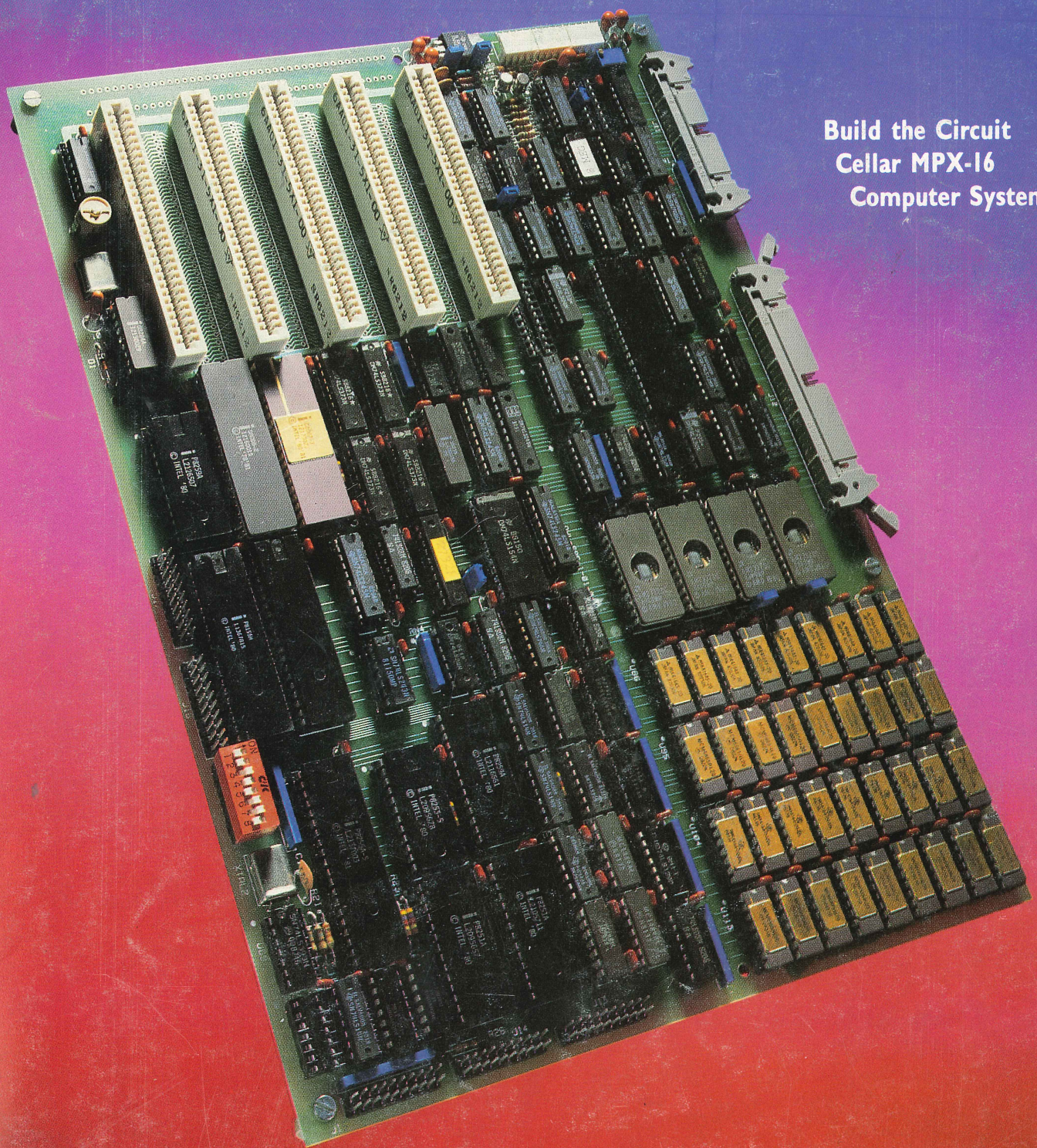
BYTE®

the small systems journal

NOVEMBER 1982 Vol. 7, No. 11
$2.95 in USA
$3.50 in Canada/£1.85 in U.K.
A McGraw-Hill Publication



Build the Circuit Cellar MPX-16 Computer System

GRAPHICS

Build the Circuit Cellar MPX-16 Computer System

Part 1

Any peripheral device designed to be installed in the IBM Personal Computer can be plugged into this 8088-based system.

Steve Ciarcia
POB 582
Glastonbury, CT 06033

Let's see. What's next? A computer-controlled bird bath? An early-warning radar transponder? How about a satellite-tracking system? Something simple.

After this series of articles is over, I am going to write about uncomplicated construction projects for a while. You'd think that after doing 50 or so projects over the past four years I'd have learned to recognize when uncontrolled invention was getting the upper hand, as it did in this month's project.

I was caught up in the fervor that resulted from the introduction of the IBM Personal Computer. As I had already written two articles on the Intel 8088 microprocessor used in the IBM machine, I quickly decided to jump on the bandwagon and purchase the first IBM PC (as it's called by its owners) that I could get my hands on. I've found myself in agreement with the prevailing opinion that the IBM PC is a solid design and well supported, but it's relatively expensive to upgrade.

The design of the MPX-16 had to be a team effort.

Somewhere along the way I had the absurdly ambitious idea of presenting a Circuit Cellar construction project on building a full computer system based, like the IBM PC, on the Intel 8088 microprocessor. (After all, I've done many microprocessor projects before.) And somewhere further along the way I decided to do it.

Design Concepts

Certain questions had to be addressed, of course. Should I try for a 10-chip design or splurge and make it 20 chips? What kind of expansion-bus scheme should the system use? What about supporting software? Could I design a small 8088-based computer and call it a development system?

The initial stages of design moved very quickly, and in a few weeks I had put together a prototype of a 64K-byte 8088-based trainer or development system. It was a compact design with limited input/output (I/O) capability but with relatively little expansion potential, lacking an expansion bus. It could have served well as a Circuit Cellar project. However, owning a so-called development system has come to mean that you are on your own: you won't get much support for either software or hardware. If the project was to have any real significance, support had to be available, and the burden of providing support would have been mine.

IBM and IBM Personal Computer are trademarks of International Business Machines Corporation.

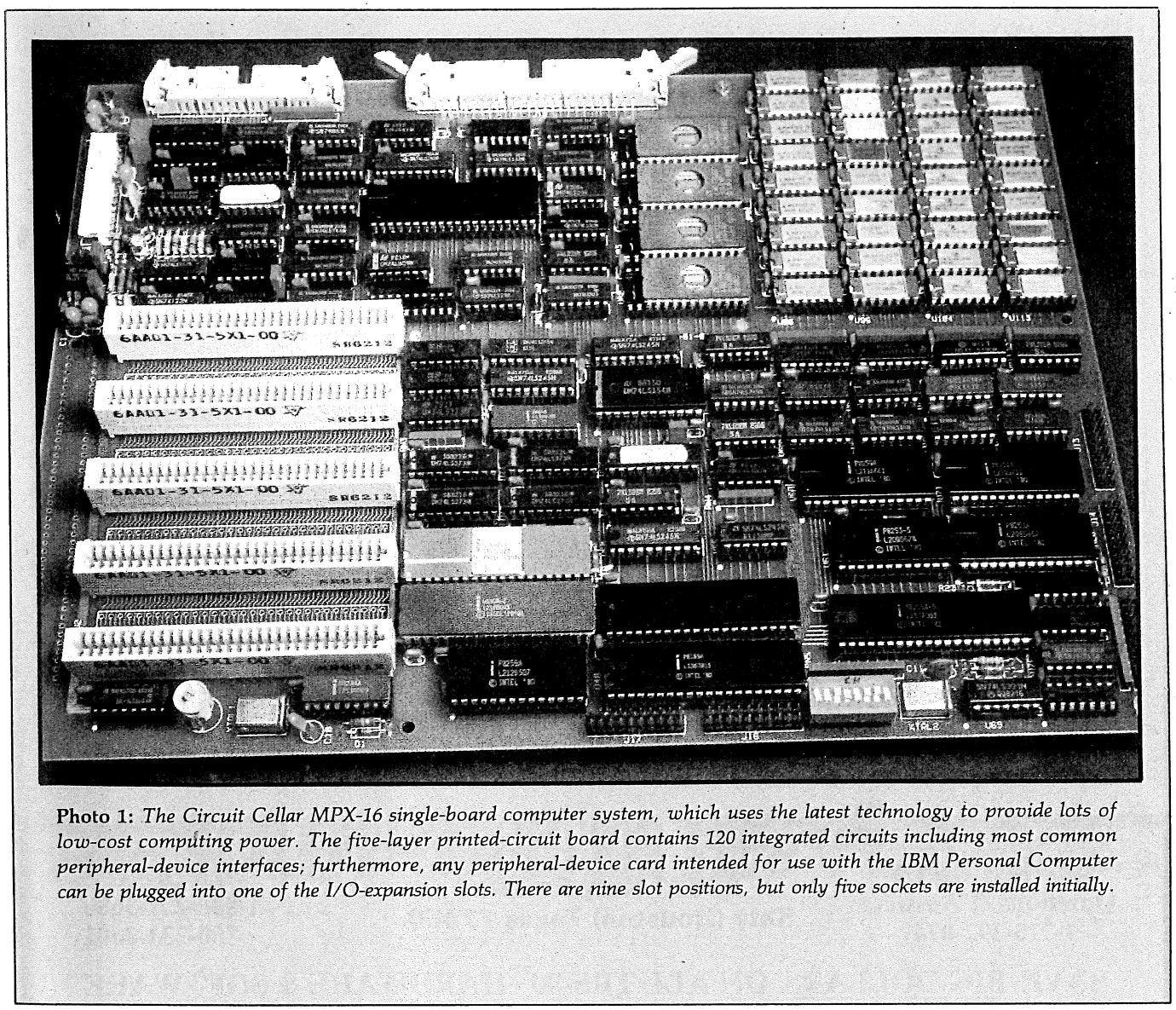


Photo 1: *The Circuit Cellar MPX-16 single-board computer system, which uses the latest technology to provide lots of low-cost computing power. The five-layer printed-circuit board contains 120 integrated circuits including most common peripheral-device interfaces; furthermore, any peripheral-device card intended for use with the IBM Personal Computer can be plugged into one of the I/O-expansion slots. There are nine slot positions, but only five sockets are installed initially.*

There was only one answer: compatibility. The project would have to be compatible with peripheral-device expansion boards and software designed for some popular computer system. The logical choice, given that I intended to use an 8088, was to make my project compatible with the I/O-expansion bus of the IBM Personal Computer. (The only real alternative was the S-100 bus, but somehow in my fervor the IBM PC route seemed more natural. S-100 fans should look up reference 1.) Consequently, IBM PC memory and I/O-expansion boards, available from numerous sources, could be used to expand this new computer.

But in making this choice, I opened Pandora's box. I was already committed to producing the article, but making my little prototype bus-compatible with the IBM PC was like fitting the *Queen Mary* into a bathtub. Scratch one prototype; start thinking about the "system board."

Ten minutes later, I realized that this would have to be a team effort. I would need assistance in developing the design, the documentation, and the software, so I enlisted help from a few friends and other engineers to form the design team.

At that point, team (or rather, committee) dynamics came into play. If you give a committee 3 square inches of empty space on the printed-circuit board, they'll want to increase performance by packing 10 more integrated circuits into it. Essentially that's what happened to my little trainer board. Not only would the resulting system be bus-compatible with the Personal Computer, but it would overcome some of the expansion weaknesses of the IBM machine by incorporating many peripheral devices as part of the basic design. Instead of a board that could be expanded into a system, this new computer would be a complete system that had been shrunk to fit on a single board.

Design Characteristics

The result of our effort is called the Circuit Cellar MPX-16 Computer

1. 5-MHz Intel 8088 main processor
2. optional Intel 8087 numeric coprocessor
3. 256K-byte on-board-user-memory capacity, with parity
4. two RS-232C serial input/output ports
5. three parallel input/output ports
6. on-board controller for either 5¼-inch or 8-inch single- or double-density floppy-disk drives (up to four)
7. supports the CP/M-86 operating system directly, with BIOS in EPROM
8. nine expansion slots (five connectors provided), bus-compatible with IBM Personal Computer
9. sockets for 64K bytes of 24- or 28-pin EPROM
10. four independent DMA channels
11. sixteen levels of vectored interrupts

Table 1: *Features of the MPX-16 computer system.*

System, shown in photo 1. Consisting of a single 9- by 12-inch five-layer printed-circuit board containing 120 integrated circuits (ICs), the MPX-16 is completely compatible with the expansion bus of the IBM Personal Computer and contains the following features: provision for an optional Intel 8087 math coprocessor, 256K bytes of RAM (random-access read/write memory), serial and parallel I/O ports, floppy-disk controller, expansion slots, and support for Digital Research's CP/M-86 operating system. (A more detailed list of features appears in table 1.)

The MPX-16 constitutes a complete, single-board computer system, using the latest technology to provide lots of low-cost computing power. It is designed to utilize all the expansion peripherals that are available for the IBM machine, and because it has so many capabilities built in, you don't have to use up expansion slots for simple jobs like interfacing a printer. Programmers, however, will undoubtedly want more memory. To meet this demand, additional memory boards can be plugged in to provide the system with one full megabyte of user memory. A hard-disk drive can be added easily, and an 8087 mathematics coprocessor can be inserted to multiply the system's raw computing power by a factor of 10 to 100.

The MPX-16 is designed initially to use CP/M-86, but it will ultimately accommodate Microsoft's MS-DOS and any other software that does not use unique features of the IBM Personal Computer. The greatest difference is this: as a stand-alone system, the MPX-16 communicates with the user through a serially interfaced display terminal instead of through a memory-mapped video display and separate keyboard. The BIOS (basic input/output system) module of CP/M-86 is contained in a set of EPROMs (erasable programmable read-only memories) on the board.

The MPX-16 is almost complete on a single board. In addition, you need merely a power supply, a serial terminal, and one floppy-disk drive. To start operation, you just turn on the power, insert a CP/M-86 disk, and start the bootstrap operation. For the sake of appearance, though, you may

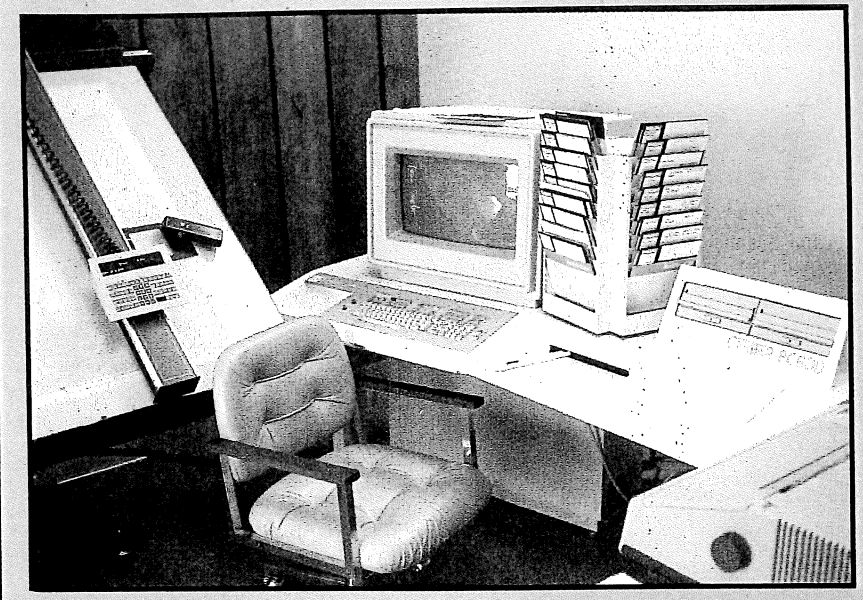


Photo 2: *The Gerber Scientific Instrument Company PC-800 CAD (computer-aided design) machine used to lay out the MPX-16's five-layer printed-circuit board.*

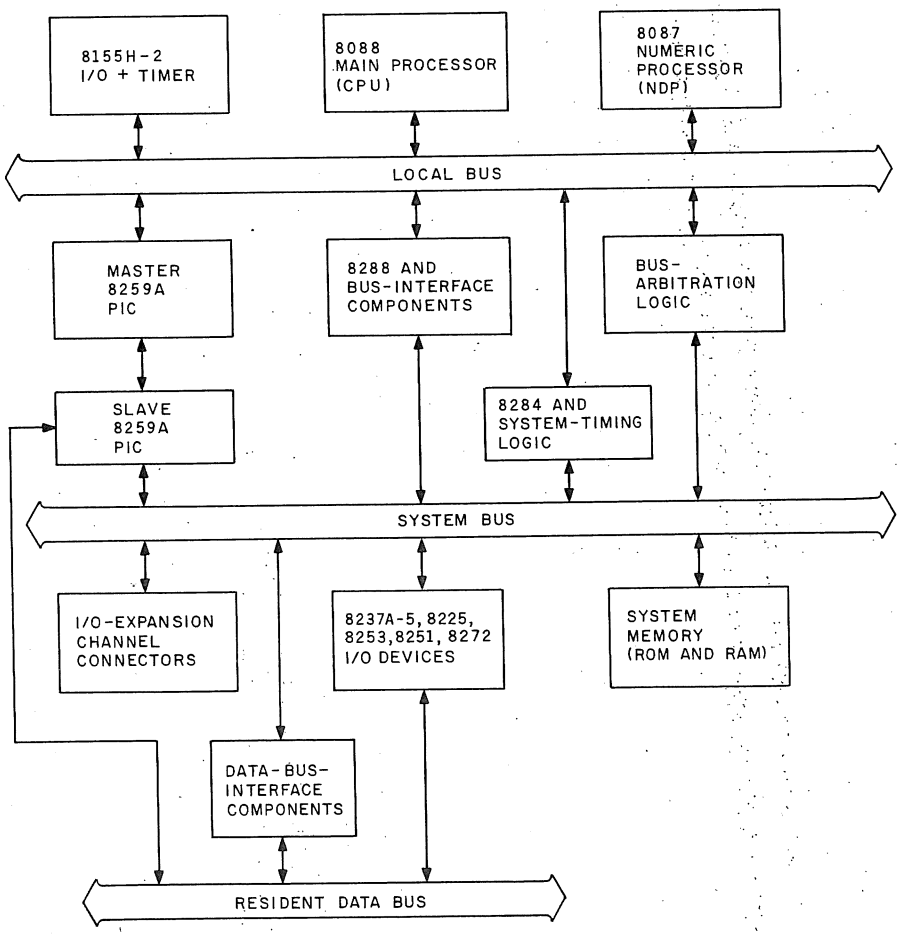


Figure 1: *Simplified, high-level block diagram of the Circuit Cellar MPX-16 computer system. The abbreviation "PIC" stands for "programmable interrupt controller."*

want to wrap up the whole thing in a suitable enclosure; one should be available by the time you read this.

Pragmatic Considerations

Obviously, it is impossible to describe the construction of such a powerful computer in detail in a single article. Even dividing it into three parts, as I plan, will be a difficult task; it will take us three months to print schematic diagrams of the entire computer in the magazine. I'll try to be as explicit as I can concerning how the circuitry works, but you must understand from the outset that this is no beginner's project.

The condensation of information here is counterbalanced by the support available from The Micromint, where you can get assembled and tested systems, blank printed-circuit boards, and complete documentation containing all the circuit diagrams plus much more detail than can be included in these brief articles.

Finally, before I start the details, I'd like to say something about the MPX-16's circuit board. Printed-circuit boards are available for building most recent Circuit Cellar projects, and this project is no exception. The only departure from the norm this time is in the complexity of the board.

The MPX-16 contains 120 IC packages. To keep its size manageable, we had to use a multilayer printed-circuit board instead of the relatively simple double-sided boards used in smaller-scale projects. With the aid of a Gerber Scientific Instrument Company PC-800 CAD (computer-aided design) machine, shown in photo 2, we eventually arrived at a 9- by 12-inch board with five layers of connecting traces. This is significant because multilayer boards cost about 10 times as much as standard double-sided boards. But even with an expensive circuit board, I believe that the MPX-16 has unbeatable performance for its cost.

MPX-16 Overview

The functional organization of the MPX-16's onboard components is illustrated in two levels of detail. Figure 1 shows a simplified, high-

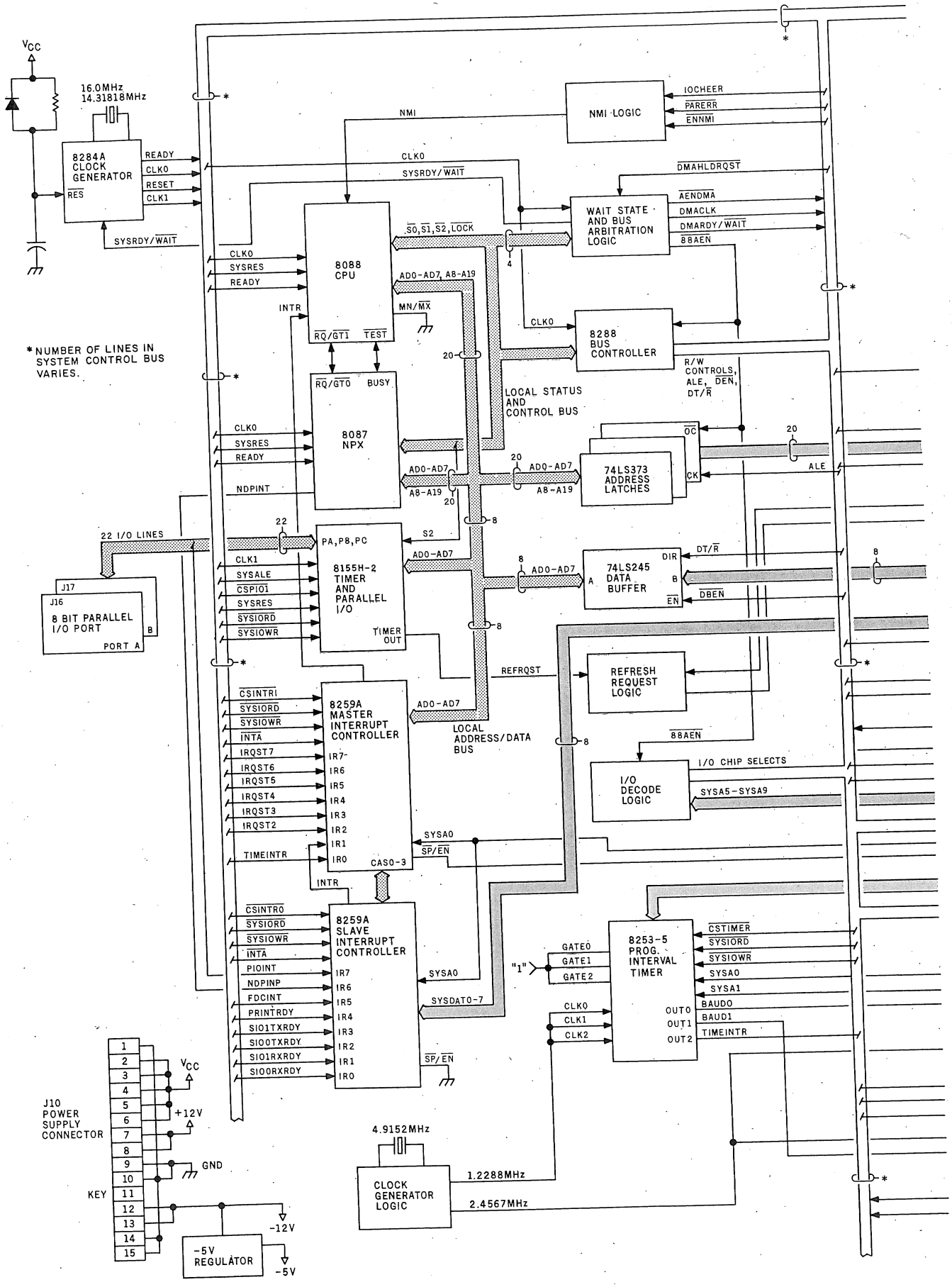
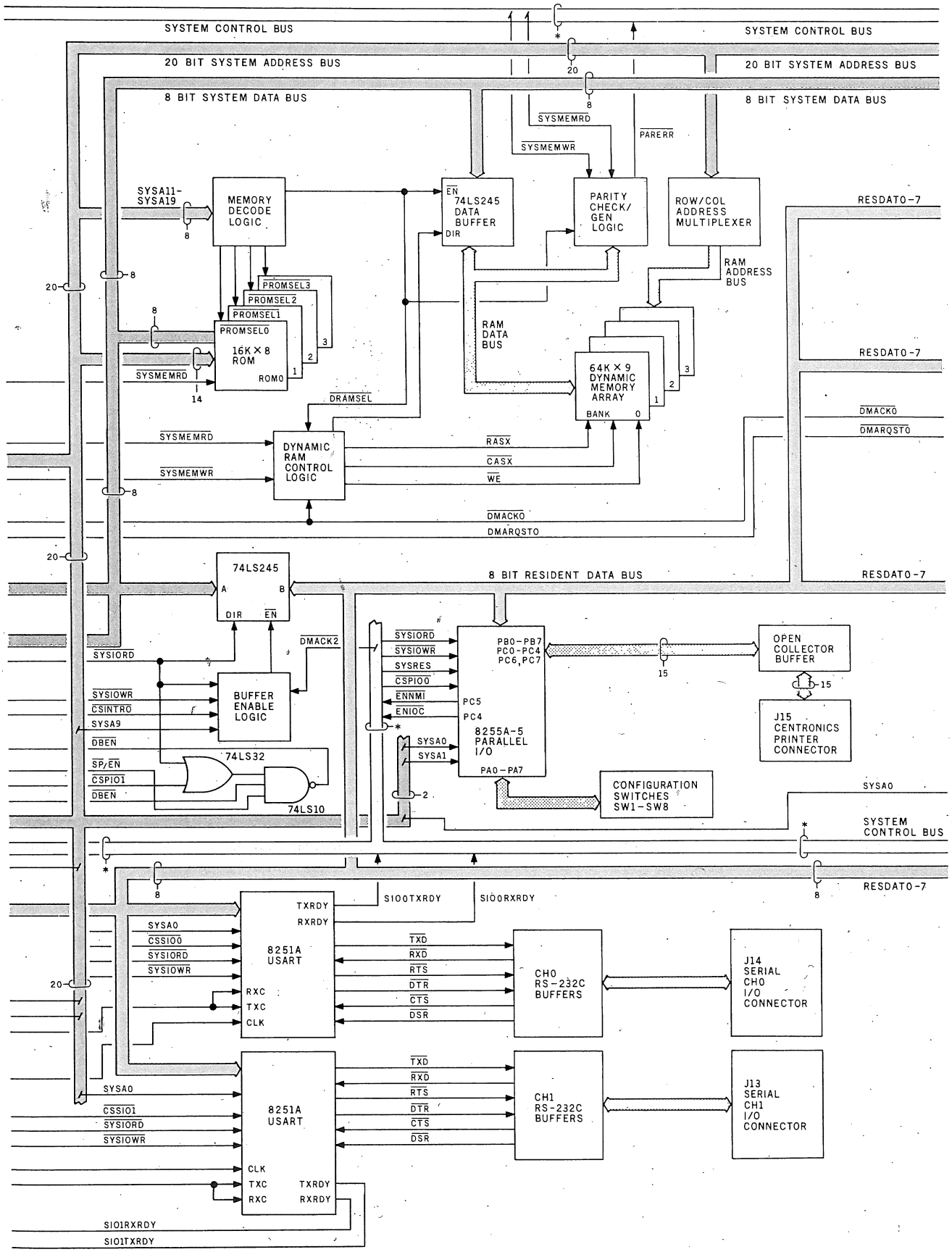


Figure 2: *Complete, detailed flow diagram of the MPX-16 system. (The diagram is continued on page 86.)*


SYSTEM CONTROL BUS
20 BIT SYSTEM ADDRESS BUS
8 BIT SYSTEM DATA BUS
SYSMEMRD
SYSMEMWR
PARERR
SYSA11-SYSA19
MEMORY DECODE LOGIC
EN 74LS245 DATA BUFFER DIR
PARITY CHECK/ GEN LOGIC
ROW/COL ADDRESS MULTIPLEXER
RESDAT0-7
RAM ADDRESS BUS
RAM DATA BUS
PROMSEL3
PROMSEL2
PROMSEL1
PROMSEL0
16K X 8 ROM
ROM0
64K X 9 DYNAMIC MEMORY ARRAY
BANK 0
DRAMSEL
DYNAMIC RAM CONTROL LOGIC
RASX
CASX
WE
DMACK0
DMARQST0
74LS245
8 BIT RESIDENT DATA BUS
SYSIORD
SYSIOWR
SYSRES
CSPIO0
ENNMI
ENIOC
PB0-PB7
PC0-PC4
PC6,PC7
PC5
PC4
8255A-5 PARALLEL I/O
PA0-PA7
SYSA0
SYSA1
OPEN COLLECTOR BUFFER
J15 CENTRONICS PRINTER CONNECTOR
CONFIGURATION SWITCHES SW1-SW8
DMACK2
BUFFER ENABLE LOGIC
CSINTR0
SYSA9
DBEN
SP/EN
CSPIO1
74LS32
74LS10
SIO0TXRDY
SIO0RXRDY
8251A USART
TXRDY
RXRDY
TXD
RXD
RTS
DTR
CTS
DSR
RXC
TXC
CLK
CSSIO0
CSSIO1
CH0 RS-232C BUFFERS
CH1 RS-232C BUFFERS
J14 SERIAL CH0 I/O CONNECTOR
J13 SERIAL CH1 I/O CONNECTOR
SIO1RXRDY
SIO1TXRDY

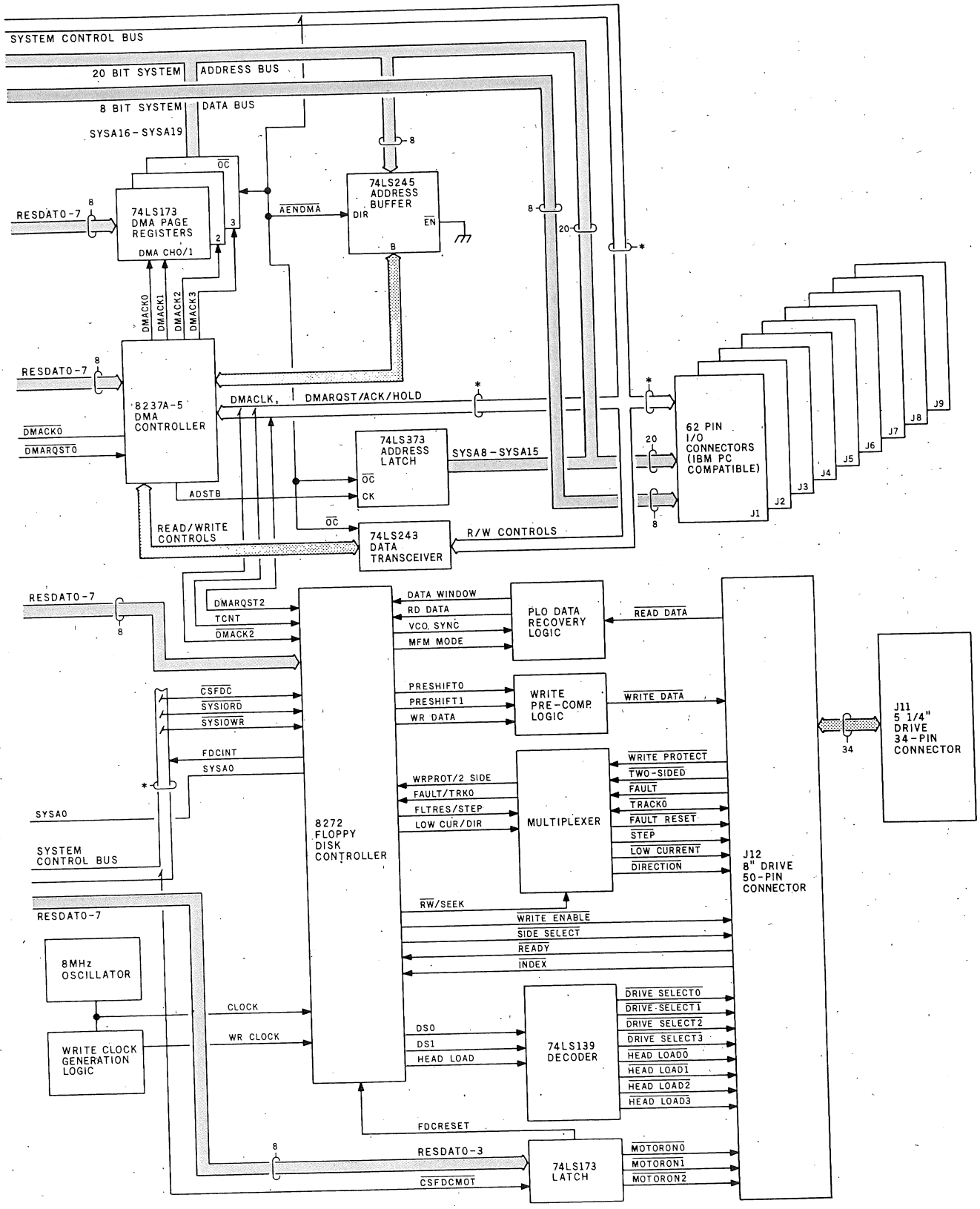


Figure 2: *Continued from page 85.*

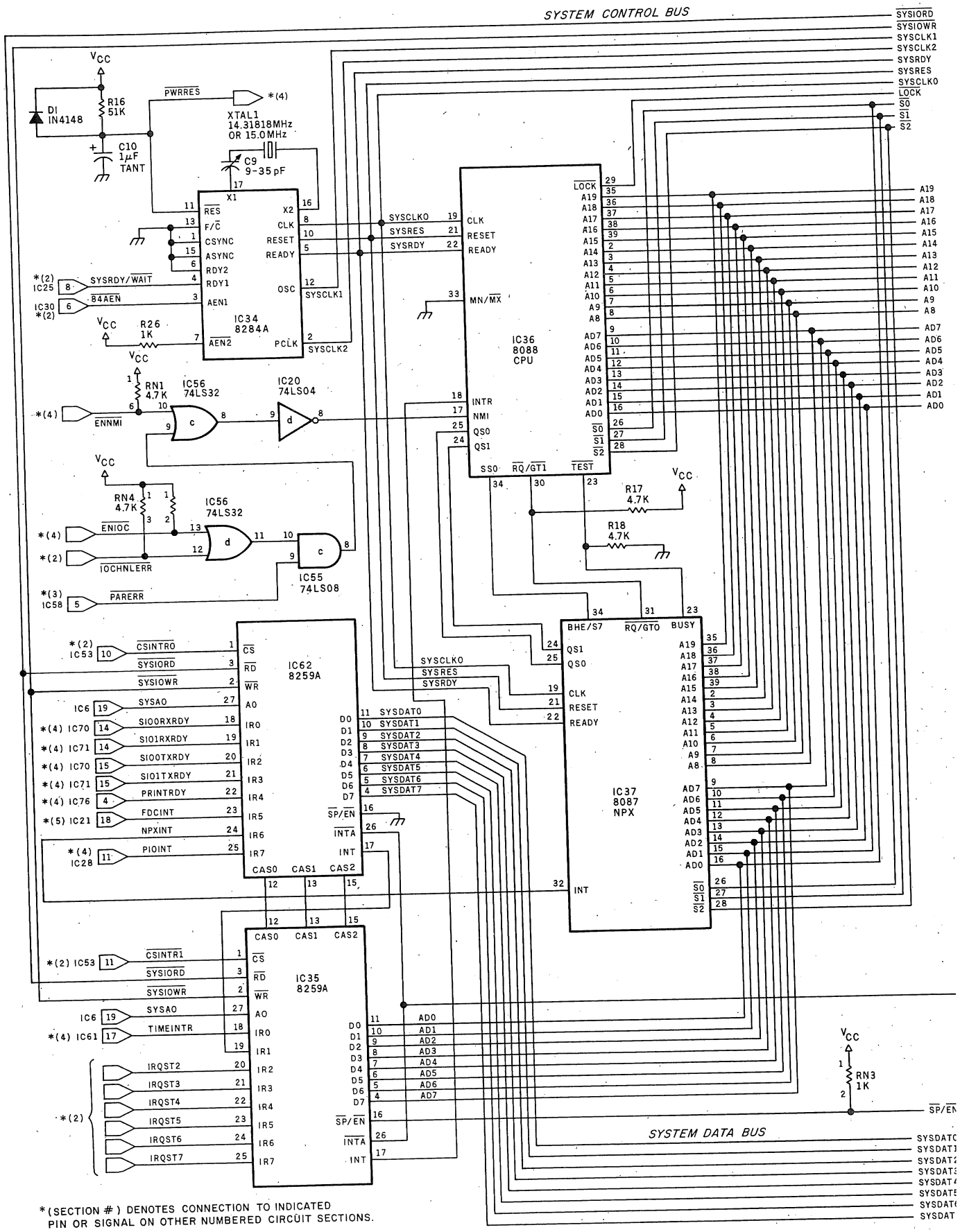
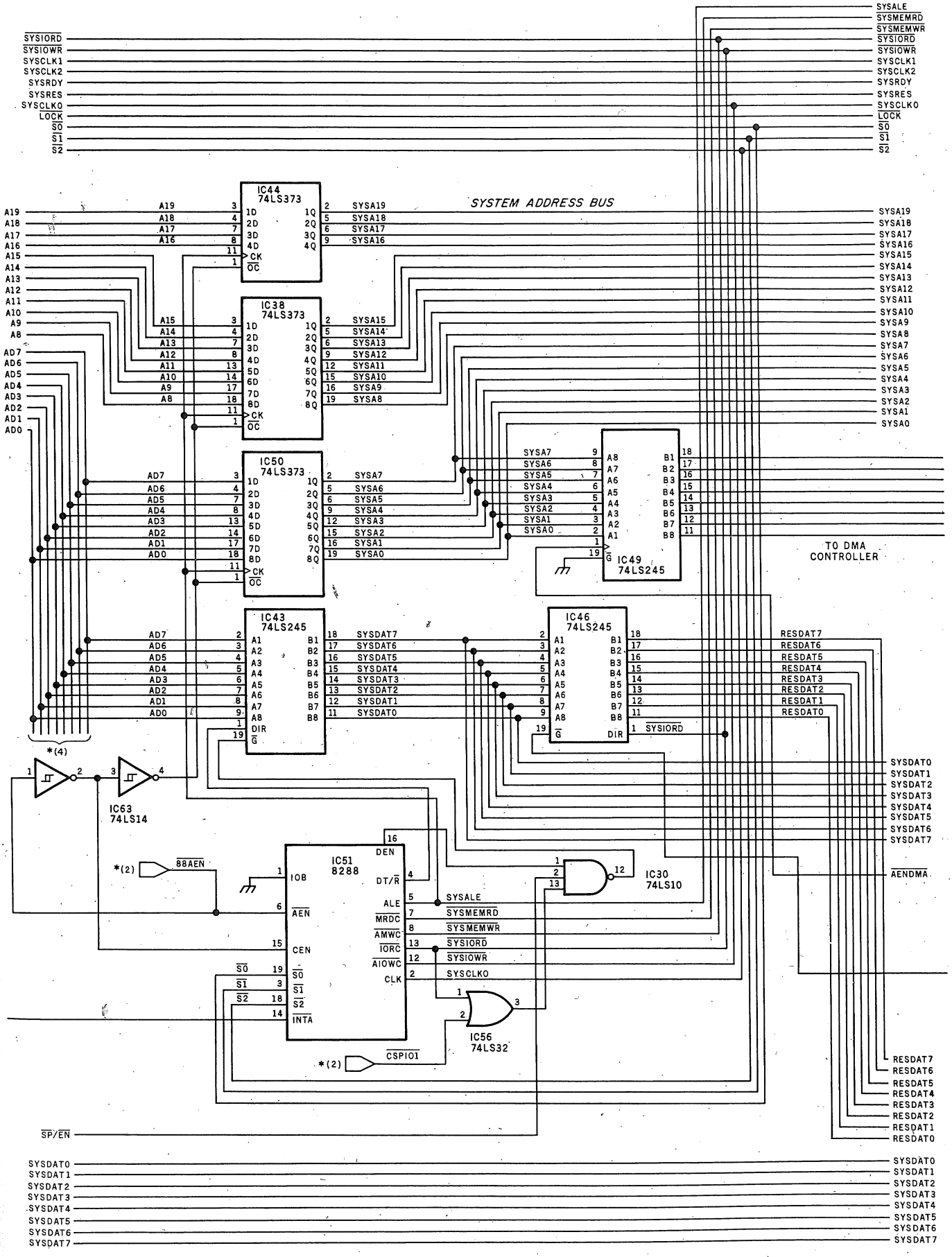


Figure 3a: *Half of section 1 of the schematic diagram of the MPX-16; the second half of section 1 appears as figure 3b on the next t pages. Sections 2 through 5 of the schematic will appear in December's and January's articles. Many connections to other sections the schematic are indicated in this figure by the notation *(n), where n is the section number; IC numbers in the other sections*



given where appropriate. Here are shown the main processor, numeric coprocessor, interrupt controllers, clock generator, bus controller, bus latches, bus transceivers, and miscellaneous components. A table of power connections will be published in the December 1982 BYTE.

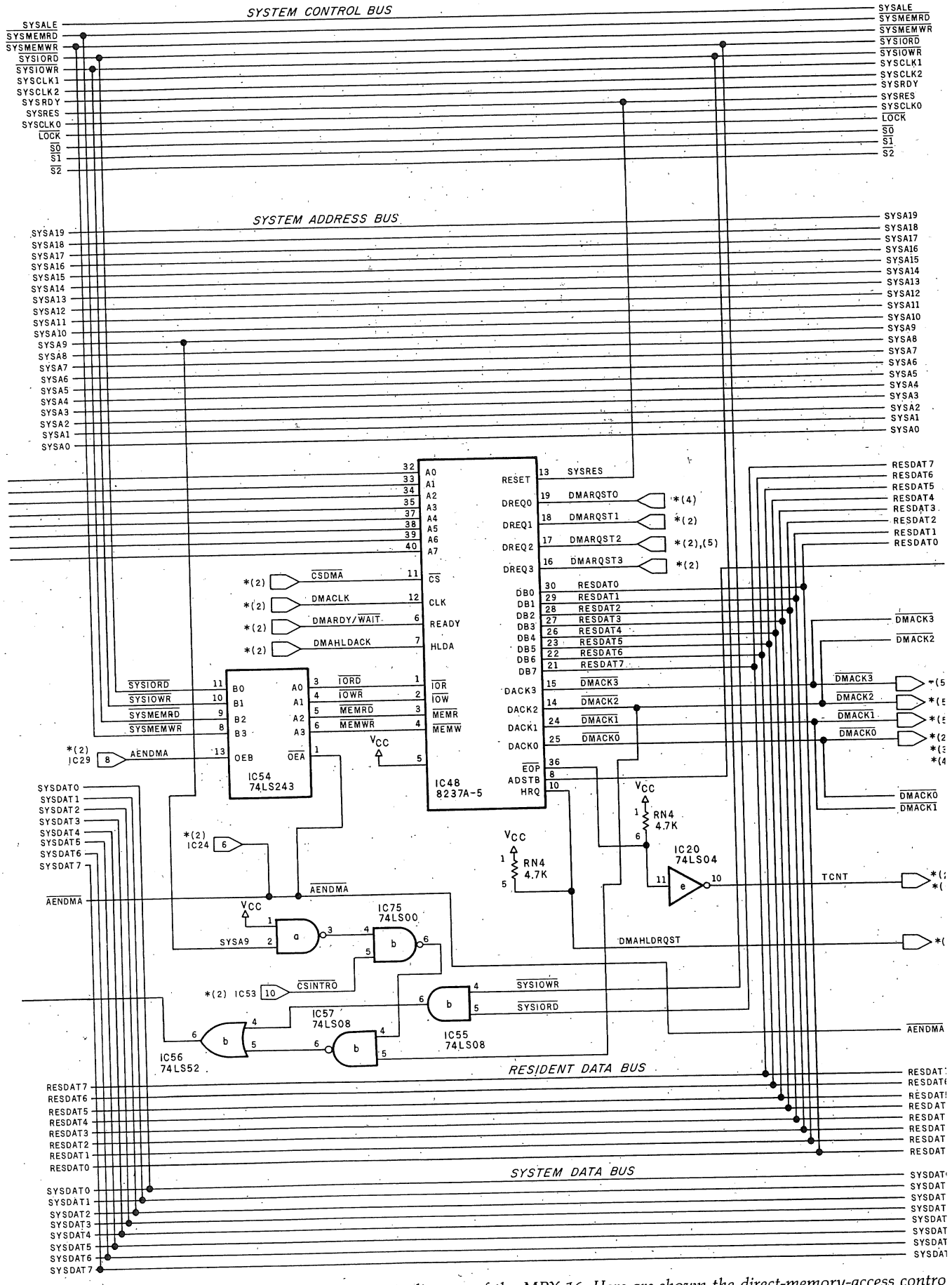
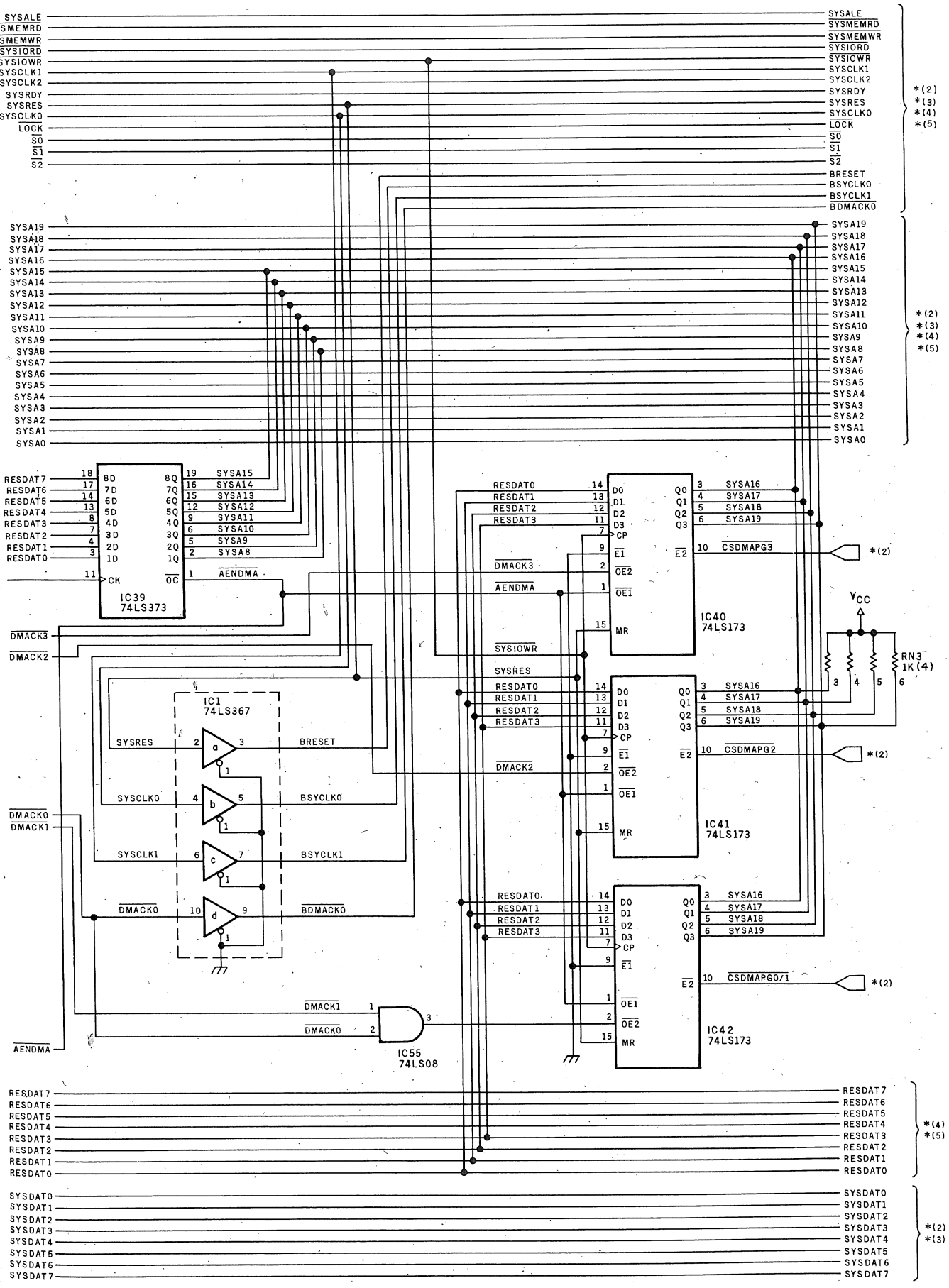


Figure 3b: *Second half of section 1 of the schematic diagram of the MPX-16. Here are shown the direct-memory-access contro control-signal latches, registers, and transceivers; and various logic gates. Note the large number of bus lines for addresses, data, control signals. Sections 2 through 5 of the schematic will appear in December's and January's articles. Many connections to o*



*sections of the schematic are indicated in this figure by the notation *(n), where n is the section number; IC numbers in the other sections are given where appropriate.*

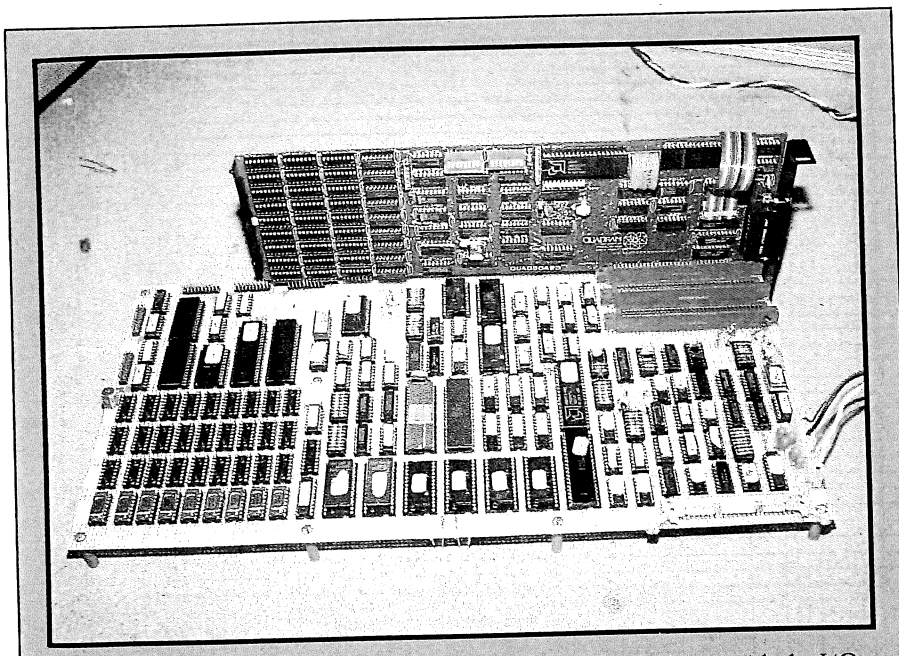


Photo 3: *Prototype of the MPX-16 being tested for compatibility with the I/O-expansion bus of the IBM Personal Computer. A Quadram Quadboard (an expansion card for the IBM PC that contains 256K bytes of memory, a serial port, a parallel port, and a real-time clock) is inserted in one of the MPX-16's slots. It works!*

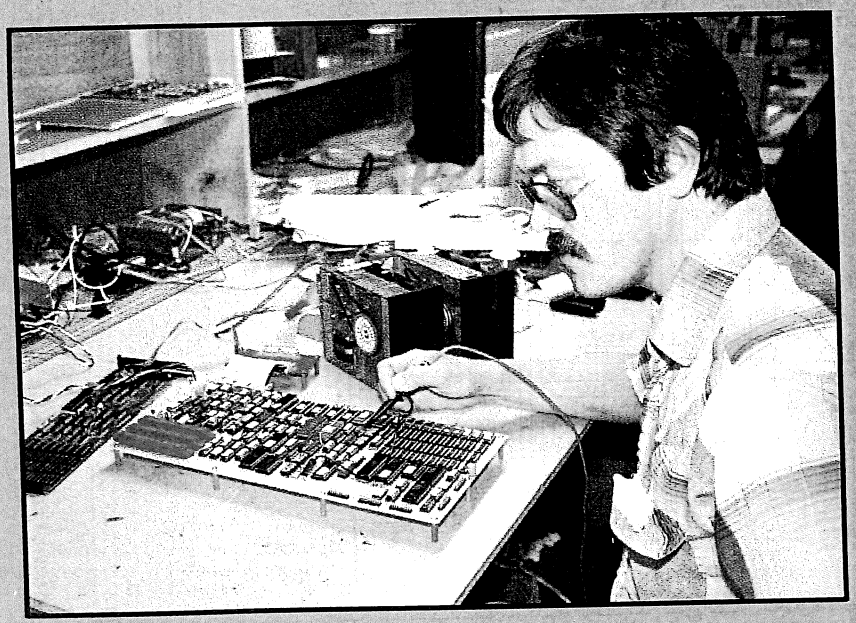


Photo 4: *Engineer Jim Norris of Owl Electronic Laboratories tests the MPX-16 prototype.*

level block diagram, while figure 2 on pages 84, 85, and 86 contains a full flow diagram for all parts of the system.

We'll look at each constituent subsystem separately, beginning with the processor and coprocessor; arrangement of data, address, and control signal buses; clock signals; the NMI (nonmaskable interrupt); and the DMA (direct memory access) subsystem. Section 1 of the schematic diagram, which appears as figures 3a and 3b on pages 88, 89, 90, and 91, contains most of these subsystems, although I do mention some things that will show up in schematic-diagram sections to be published in parts 2 and 3 of this series.

Intel 8088 Processor

The new 16-bit microprocessors are more powerful than their 8-bit predecessors. Not only do they operate at faster speed, but the 16-bit chips manipulate numerical quantities in larger chunks, directly address more memory, and offer the programmer expanded instruction sets. But along with the greater capability comes a new set of computer-design considerations.

An alternative to complete commitment to 16 bits is embodied in the heart of the MPX-16: the powerful Intel 8088 microprocessor. The 8088 uses a 16-bit internal architecture and instruction set and possesses a 1-megabyte memory-addressing capability and a 64K-byte I/O-addressing capability, but communicates through an 8-bit external data bus (sort of like putting its data flow through a funnel). The 8088 has a common internal architecture and complete software compatibility with the pure-16-bit Intel 8086 microprocessor. As a result, the 8088 provides an excellent way for designers, engineers, hobbyists, and students to ease into the world of 16-bit computing by taking advantage of its 8-bit-compatible bus structure.

The 8088 can be used in low-cost systems that employ a few multiplexed-bus support chips such as the Intel 8155 (2K-bit static RAM with I/O ports and timer), 8755A (16K-bit EPROM with I/O ports), and 8185

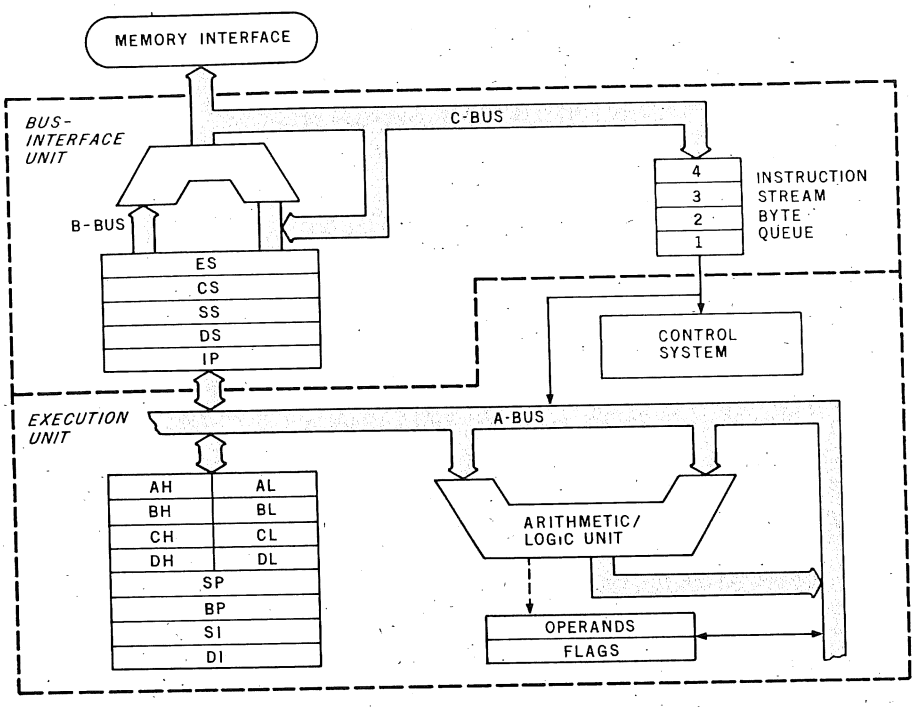


Figure 4: *Functional block diagram of the Intel 8088 microprocessor. Its pipelined architecture, shared with the 8086, increases speed by overlapping the execution of instructions with memory-access operations.*

(1K-byte static RAM). That was the approach I took in my previous article series (see reference 2, listed on page 114). But the power of the 8088 can best be exploited when it serves as the nucleus of a fully expanded system, using its full address space and coprocessing capabilities.

The 8088 microprocessor can be set up to interact with other components in the system in either the maximum or minumum mode. Certain control and status signals differ between the two modes. The selection is made by connecting the MN/$\overline{MX}$ pin on the 8088's package to ground or to +5 V (volts). In the minimum mode, the 8088 functions as a standalone processor, interacting with peripheral devices somewhat like the 8-bit 8085 processor. In the maximum mode, other integrated circuits perform certain specialized functions such as bus control, numeric data processing, and input/output control. In the MPX-16, the 8088 is configured for maximum-mode operation.

Architecture of the 8088

The internal architectures of the 8088 and 8086 processors are identical. A diagram of their internal structure is shown in figure 4. The 8088 contains two logical functional divisions—the bus-interface unit (BIU) and the execution unit (EU)—with a logical pipeline between them that provides an instruction queue.

The 8088 uses *instruction queuing* to increase computing speed. A 4-byte instruction queue holds contents of the four bytes in memory that consecutively follow the instruction being performed by the execution unit. These four bytes of instructions or data are brought into the processor before they are to be executed; therefore, when the EU is ready to execute the next instruction, frequently it or the data required is contained already in the queue. Only when the EU needs to access nonconsecutive addresses (or during a few combinations of especially fast-executing instructions) will time be consumed for memory fetches. By not tying up the memory bus as often as its nonqueuing 8-bit predecessors, the 8088 makes the bus available for use by other powerful support devices. The overall result is increased efficiency and faster processing.

The *execution unit* is where the actual processing of data takes place inside the 8088. It is here that the familiar arithmetic and logic unit (ALU) is located, along with the registers used to manipulate data, store intermediate results, and keep track of the pushdown stack. The EU accepts instructions that have been fetched by the BIU, then processes the instructions. It next returns operand addresses to the BIU, processes the operands, and then passes them back to the BIU for storage in memory.

The role of the *bus-interface unit* is to maximize bus-bandwidth utilization (that is, to speed things up by making sure that the bus is used to its full capacity). The BIU carries out this assignment in two basic ways: first, by fetching instructions before they are needed by the EU and storing them in the instruction queue, and second, by taking care of all operand fetch and -store operations, addre

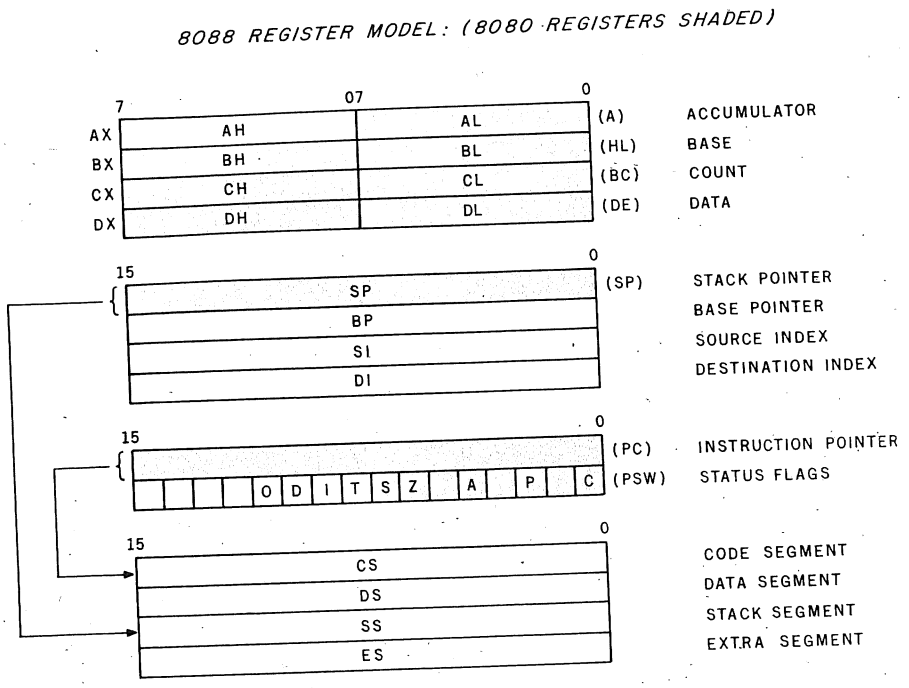


Figure 5: *Programmer's model of the 8088's fourteen 16-bit registers. The shaded registers are the 8080-register subset, that is, the registers that are common to the 16-bit 8088 and its 8-bit predecessors.*

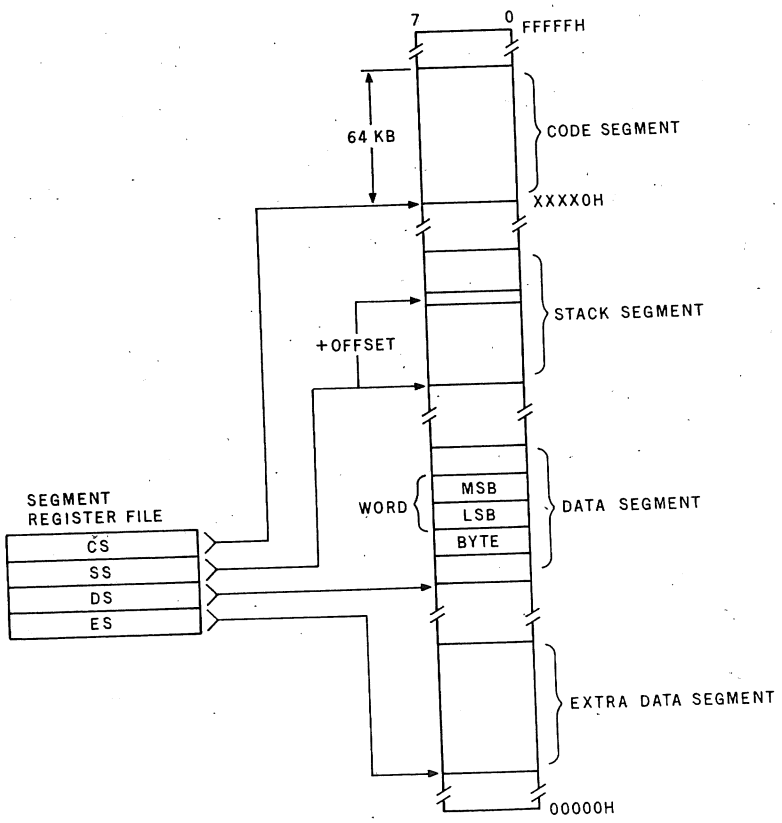


Figure 6: *Memory organization in the 8088. Memory segmentation is used to address up to 1 megabyte (1,048,576 bytes) in segments of 64K bytes. The 8088 creates a 20-bit address by combining a 16-bit offset value with a segment-boundary value stored in one of the segment registers.*

relocation, and bus control. (These actions of the BIU leave the EU free to concentrate on processing data and carrying out instructions.)

Figure 5 shows the programmer's model of the 8088's fourteen 16-bit registers. The shaded registers are the 8080-register subset, that is, the registers that are common to the 8088 and its 8-bit predecessors.

The general registers, also called the HL group because they can be subdivided into high and low bytes, include the accumulator (AX), base register (BX), count register (CX), and data register (DX). The two bytes in any of the general-purpose registers can be operated on separately; for instance, the AX register can be addressed as a 16-bit register, AX, or the high-order byte can be addressed as the register AH and the low-order byte as AL. The same holds true for BX, CX, and DX.

Another group of registers is the pointer and index (or P and I) group. This set contains the stack-pointer (SP), base-pointer (BP, an extra pointer into the stack), source-index (SI), and destination-index (DI) registers. Generally speaking, these registers hold offset addresses used for addressing within a segment of memory. They can also participate, along with the general-register group, in the arithmetic and logical operations of the 8088.

The 8088 uses *memory segmentation* to address this large memory space efficiently; it deals with memory as a set of four 64K-byte segments simultaneously defined (possibly overlapping) within the memory-address space, which is organized as a linear array of 1,048,576 bytes, addressed as hexadecimal 00000 through hexadecimal FFFFF. The 8088 creates a 20-bit address by combining a 16-bit offset value with a segment-boundary value stored in one of the segment registers. Figure 6 shows how this works.

Each of the 16-bit segment registers, the code-segment (CS) register, the stack-segment (SS) register, the data-segment (DS) register, and the extra-data-segment (ES) register, contains a value that can be combined with the 16-bit offset address speci-

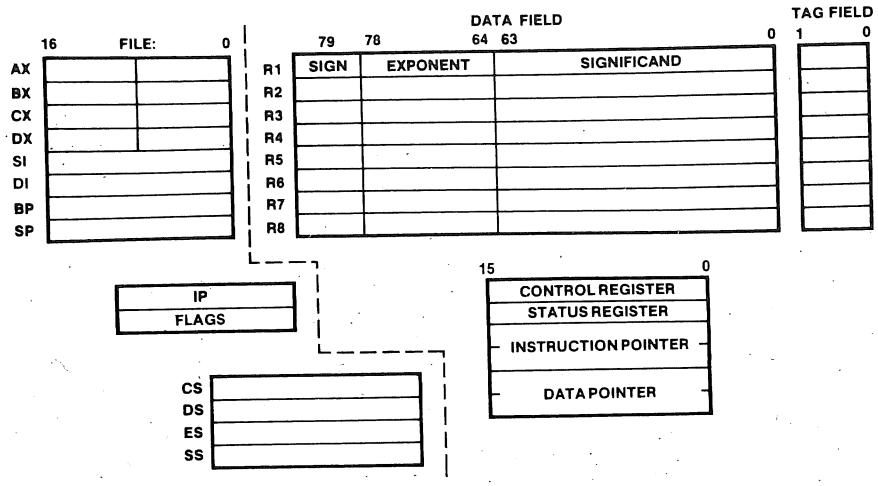


Figure 7: *Programmer's model of the 8088/8087 coprocessor combination. The 8087 adds eight 80-bit registers to the architecture and 68 operations to the instruction set. The 8088/8087 combination can operate on BCD (binary-coded decimal) numbers up to 18 digits long without round-off errors and perform arithmetic on 64-bit integers. (Figure provided courtesy of Intel Corporation.)*

fied by the instruction operand to form the 20-bit address. For instance, the 16-bit value in the code-segment register first has four low-order zero bits appended; it is then added to the low-order 16 bits of the offset address. When the 8088 fetches an instruction or data byte from memory, it comes from the location at the absolute address thereby formed.

The memory is thus divided into four segments: the code segment, where instructions are stored; the stack segment, where the pushdown stack is located; the data segment, where data to be operated on is found; and the extra segment, a 64K-byte data area assignable for any data-storage use. Which code-segment register is used to form the address varies according to what processor instruction is being executed.

The 8088 has both relative and absolute control-branching instructions. When all branch instructions within a given segment of memory are specified in relation to the instruction pointer and the program segment does not modify the value of the code-segment register, that program segment can be relocated dynamically anywhere within the entire address space simply by moving the code, updating the value of the code-segment register, and resuming execution.

The 8087 Numeric Processor

The Intel 8087 numeric processor extension (NPX) is an integrated circuit designed for use with the 8086 or 8088 (serving as the central-processing unit, or CPU) to form a high-performance numeric-data-processing system (called the iAPX 86/20 NDP or iAPX 88/20 NDP in Intel jargon). Its use is optional in the MPX-16.

The 8087 is designed to coordinate its functions with other processors in a coprocessing or multiprocessing environment. As a coprocessor, the 8087 adds 68 machine instructions to the system; these operate on its eight 80-bit floating-point registers, which function alongside the 8088's register set. The 8087 is designed to handle very large numbers; its internal temporary-storage format for floating-point quantities is 80 bits: 1 bit for sign, 15 bits for exponent, and 64 bits of mantissa. A programmer's model of the resulting architecture is shown in figure 7.

Capable of executing arithmetic, trigonometric, exponential, and logarithmic instructions, the 8087 conforms to the proposed IEEE (Institute of Electrical and Electronics Engineers) floating-point standard

and provides 100 times the performance of the 8088 working alone. It's a real number cruncher.

The 8087 uses its own instruction queue to monitor the 8088's instruction stream, operating only on those instructions intended for it. When a numeric instruction appears in the code, it is treated as an "escape" from the normal sequence by both the 8088 CPU and 8087 NPX; the NPX processes it while the CPU finishes its current task. Concluding that, the CPU either does nothing (if the NPX requires no further data) or it calculates an address and reads a byte of memory which is used by the NPX (the CPU ignores this value in its own computations).

The 8087 is only an extension processor and cannot run by itself. It needs a separate CPU to operate the data, address, and control buses, which provide it with instructions and operands. Once the NPX has started its operation, the CPU may continue executing the main program while the NPX "crunches." This parallel operation of the NPX and CPU can continue until the NPX needs to reference memory. Only then will the processor give the NPX access to the bus (the main processor may, however, continue to process instructions from its instruction queue). A special request/grant line, $\overline{RQ/GT0}$, is used to pass control of the buses shared between the NPX and the CPU. The relationship between the CPU and the NPX is similar to the master/slave scheme used in less complicated computers, while the protocol is somewhat like hold and hold-acknowledge signals, although more complicated. (Additional processors or coprocessing devices can be attached to the NPX/CPU combination through another signal line, $\overline{RQ/GT1}$, although no provision for this has been made in the MPX-16.)

The amount of time that the processor actually waits to get back on the bus is very small. If it were not for a few stolen memory cycles, the coprocessor's operation would be essentially invisible to the host processor; it's a small price to pay for the great increase in performance for numeric computation. As a comparison, even though it's quite a powerful microprocessor, the pure-16-bit 8086 takes about 20 milliseconds to compute a square root, using a floating-point subroutine. Eliminating the subroutine and using the 8087 instead, the result can be calculated in less than 40 *micro*seconds (the speed-up is similar for the 8088). Such speed is an undeniable asset to high-level languages such as BASIC and Pascal. They not only run faster, but the memory space devoted to floating-point subroutines is saved.

MPX-16 Bus Structures

The MPX-16 system supports two major signal-bus structures, the processors' local bus and the global system bus, as you can see clearly in the simplified block diagram of figure 1 and somewhat less clearly in the detailed diagram of figure 2. Most of the signals in the MPX-16 pass on one or more of the several buses.

The *local bus* is shared by the 8088

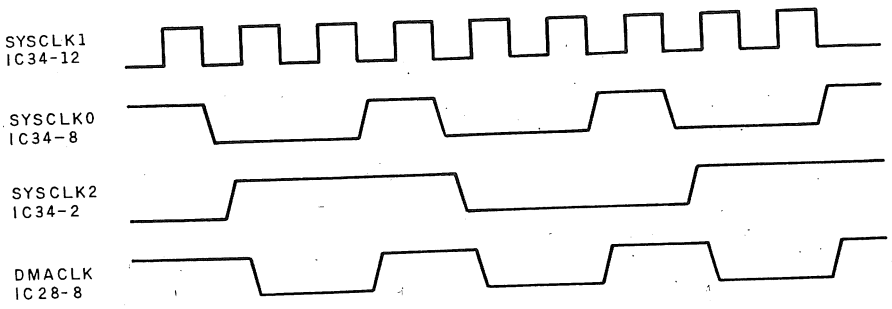
CPU and the 8087 NPX (if it is installed), either of which can be the local-bus master. The system bus can be driven either by the local bus under the control of the 8288 bus controller or by the 8237A-5 DMA controller.

The local bus consists of 8 multiplexed address and data lines (AD0 through AD7), 12 address lines (A8 through A19), and 3 status and control lines (S0, S1, and S2), which are connected with the global system bus through three-state buffers. Several other signals, including system clock signals and reset lines, are directly common with the system bus. The local bus can be controlled by either the CPU or the NPX, both of which have on-chip arbitration logic to determine which processor has control of the bus.

The CPU acts as a host processor to the NPX coprocessor. For example, when the 8087 NPX requires use of the local bus to return the result of an operation, it notifies the CPU by placing a series of handshaking signals on its bidirectional request/grant arbitration line $\overline{RQ}/\overline{GT0}$. A ready/wait control line is used to lengthen bus cycles on the local bus, which may be necessary to meet the access-time requirements of slow memory and peripheral devices, or to accommodate a DMA cycle already in progress on the system bus.

The *system bus* consists of 20 system address lines, 8 bidirectional system data lines, and several system control lines. The system data bus drives the system-board memory arrays and the I/O-expansion connectors and is buffered again to produce a "resident" data bus to which most of the on-board peripheral devices are attached. The system control bus consists of all timing signals, bus-cycle-control signals, interrupt-request lines, DMA-request/acknowledge lines, and system-bus-arbitration-control lines.

Control of the system bus is determined by a sequential-logic system-bus-arbitration circuit. The bus is always being controlled either by one of the two coprocessors via the local bus and the 8288 bus controller (with the $\overline{88AEN}$ control line active), or by the 8237A-5 DMA controller (with $\overline{AENDMA}$ active). The simple bus-arbitration circuit isolates the local bus from the system bus whenever system-bus access is given to the DMA controller for direct access to memory by one of the peripheral devices. For the DMA controller to gain access to the system bus in response to its HOLD request, a "locked" 8088 instruction (which must have continuous bus access for the 8088) must not be in execution, and the local bus must be in an idle state. The $\overline{LOCK}$ signal is also active during interrupt-acknowledge sequences, preventing the occurrence of a DMA cycle in the middle of the acknowledge sequence. Since neither of the coprocessors is involved in this bus-request/grant-arbitration sequence, a low input to the RDY1 line on the 8284 clock generator is used to force continuous wait states to be inserted in the local-bus timing cycle



	TYPICAL CLOCK PERIODS (ns)							
	SYSCLK0		SYSCLK1		SYSCLK2		DMACLK	
	CLK_L	CLK_H	CLK_L	CLK_H	CLK_L	CLK_H	CLK_L	CLK_H
15.0MHz CRYSTAL	133.3	66.7	33.3	33.3	200	200	105	95
14.31818MHz CRSTAL	139.7	69.8	34.9	34.9	209.5	209.5	111.5	98

Figure 8: *Timing diagram showing the relationship of the four major system clock signals. The table shows clock-high and clock-low periods for both the 14.31818-MHz and 15.0-MHz crystals in nanoseconds.*

until access to the system bus has been restored to the local bus.

Reset and Clock-Generator Circuits

The power-on reset pulse and all major system clock signals for the MPX-16 system and I/O-expansion slots are generated by an Intel 8284 clock generator and driver (IC34 in figure 3a). The 8284 is designed to provide the optimum clock signal, with a 33 percent duty cycle, at the voltage levels and transition times required by the 8088 CPU (IC36) and the 8087 NPX (IC37).

The 5-MHz 8088 used in the MPX-16 must operate with a clock rate between 2 and 5 MHz. (The 8088-2 version can run at 8 MHz.) The standard MPX-16 operates at a frequency of 4.77 MHz, which is derived from a 14.31818-MHz crystal oscillator. This crystal frequency provides compatibility with IBM Personal Computer color-graphics adapters, which use the 14.31818-MHz OSC clock output of the 8284 to produce a 3.58-MHz color-burst signal. The variable trimmer capacitor C9 is used to make minor adjustments in the clock frequency. An optional 15-MHz crystal can be substituted to operate the MPX-16 at its maximum clock rate of 5 MHz.

The 8284 divides the 14.31818-MHz oscillator frequency by 3 to provide the 33-percent-duty-cycle CPU clock, SYSCLK0. This clock signal is used by many parts of the MPX-16, including the 8087, the 8288 bus controller, the system-bus-arbitration circuit, and the I/O-expansion channels. SYSCLK0 is also used to provide a clock signal (DMACLK) for the 8237A-5 DMA controller. Some Schmitt-trigger inverter sections (IC24, which will appear next month in section 2 of the circuit) lengthen the level-high duration of SYSCLK0 so that the clock requirements of the 8237 will be met. Deriving the DMACLK signal from SYSCLK0 has the obvious advantage of maintaining synchronization between the local bus masters (the 8088 and 8087) and the alternate system-bus master, the DMA controller (the 8237).

In addition to the processor clock signal, the 8284 provides a peripheral-device clock signal, which is one-half the frequency of the processor clock and has a 50 percent duty cycle. The oscillator clock, SYSCLK1, is not used on the MPX-16 circuit board but is routed to the I/O-expansion connectors.

The peripheral clock, SYSCLK2, is used to drive the timer input of the 8155H-2 component (IC47, which will appear in January in section 4 of the circuit). The 8155's timer output is used to generate periodic memory-refresh requests for the dynamic memory on the system board, using the DMA controller (which we'll discuss further presently). The relationship of the four major system clock signals is illustrated in figure 8, which also contains a table of clock-high and clock-low periods for both the 14.31818- and 15.0-MHz crystals.

The 8284 clock generator is also used to generate the power-on reset pulse, SYSRES, which is active high. When power is first turned on, the rising supply voltage activates the Schmitt-trigger input pin $\overline{RES}$ on the 8284, which has approximately 0.25 V of hysteresis; as a result, the SYSRES pulse remains active until a voltage level of 1.05 V is reached on the $\overline{RES}$ input. The resistance/capacitance time constant set by R16 and C10 provides the necessary minimum reset pulsewidth of 50 μs (microseconds). The 1N4148 diode D1 provides a discharge path for C10 when power has been removed.

Nonmaskable-Interrupt Logic

The nonmaskable interrupt (NMI) input of the 8088 CPU is used for handling parity errors in the system-board memory and I/O-channel errors, which are typically also parity errors occurring in expansion-memory modules.

Although the NMI signal is nonmaskable once it gets to the 8088, logic is provided to externally mask the signals that would normally generate an NMI, if desired. Two input lines, PC4 and PC5, of the 8255A-5 PPI (programmable peripheral interface, IC60, to appear in section 4 of the circuit) are used as active-low enable signals. The $\overline{ENNMI}$ signal either enables or prevents NMI signals from reaching the 8088.

One source of interrupts is the $\overline{PARERR}$ signal, which is generated by the system-board circuit that calculates parity values for memory and detects errors. The second source of interrupts is the $\overline{IOCHNLERR}$ signal, which comes from the I/O-expansion slots. The latter signal can be masked by the $\overline{ENIOC}$ control line in such a

way that only system-board parity errors will cause an interrupt. The NMI input of the 8088 CPU is edge-sensitive but must remain active-high during two consecutive CPU clock cycles to guarantee recognition of the interrupt.

The NMI condition is predefined to be a "type 2" interrupt for the 8088, the highest-priority hardware interrupt. Since the vector (control-branching) location has been pre-assigned to locations hexadecimal 00008 through 0000B, no interrupt-acknowledge sequence is needed in a program.

DMA Controller and Bus Arbitration

Direct memory access has long been known as a way to improve the performance and I/O speed of a computer system by allowing I/O devices to directly transfer data to or from system memory without processor intervention, but until recently it has rarely been found in microcomputer systems. However, more widespread use of DMA has been made possible by semiconductor manufacturers, which have developed new ICs that make DMA much more easily provided. The MPX-16 employs one such integrated circuit, the Intel 8237A-5 DMA controller (IC48 in figure 3b on page 90).

Four independent channels of 20-bit-address direct memory access are supported by the MPX-16 system. Two of the DMA channels are available on the I/O-expansion bus to support high-speed data transfers between external peripheral devices and memory. A third channel, used by the floppy-disk-drive controller, is connected to the I/O-expansion bus for compatibility with the IBM Personal Computer.

The fourth DMA channel is used to provide the periodic refresh signal for the on-board dynamic-memory array, as well as any expansion memory boards, in which each row address of the dynamic-memory chips must be accessed. During system initialization, the TIMER OUT output of the 8155H-2 is set up to trigger a dummy DMA transfer approximately every 15 μs. The DMA channel is pro-

grammed for a memory-read cycle; it automatically increments the row-address counter for memory after each refresh cycle.

When no DMA requests are pending, the DMA controller is in an idle state (S1) and can be programmed by the CPU. If a DMA channel requests service for a peripheral device and that channel has been enabled by the system software, the DMA controller sends the signal DMAHLDRQST (DMA hold request) to the system-bus-arbitration circuit and enters the active state S0. The 8237 remains in the S0 state until it has received the signal DMAHLDACK (DMA hold acknowledge) from the bus-arbitration circuit, indicating that it has been granted control of the system bus.

At this time the system-bus-arbitration circuit isolates the local bus from the system bus by activating the control signal $\overline{88AEN}$. When this signal becomes inactive again, the 8288 bus controller (IC51) places the system-bus command-line buffers into a high-impedance state and disables the 74LS245 data transceiver IC43. In addition, the $\overline{88AEN}$ signal places the system-bus-address latches, IC50, IC38, and IC44, into a high-impedance state so that the local-bus master can drive the local bus during a DMA cycle without affecting operations on the system bus.

After one system-clock cycle following the arrival of the hold-acknowledge signal, the $\overline{AENDMA}$ control signal from IC24 enables the DMA bus-interface components. One of the 74LS373 latches, IC43, drives system address lines SYSA8 through SYSA15. The eight low-order address lines, SYSA0 through SYSA7, are driven by lines A0 through A7 on the 8237 through a 74LS245 transceiver, IC49 (shown in figure 3a on page 89). The data-flow-direction input of IC49 is controlled by the $\overline{AENDMA}$ signal such that data flow is from IC49 *to* the system address bus when a DMA transfer is in progress.

During processor memory transfers, the $\overline{AENDMA}$ signal is high, and address information flows *from* the system address bus through IC49 to the 8237. The four high-order system-bus-address lines, SYSA16 through SYSA19, are driven by three 4-bit latches (IC40, IC41, and IC42). These latches are loaded by either the operating system or application software and allow each DMA channel to operate in a separate 64K-byte section of memory if desired. Since DMA channel 0 is used for memory refresh and only the eight low-order address lines are significant, the latch for DMA channel 1 is used to drive the upper four address lines for both channels 0 and 1. The three address latches are enabled when both the $\overline{AENDMA}$ signal is active-low and the appropriate acknowledge signal is active.

Once the transfer of a single byte has been completed, the DMA controller turns off the DMAHLDRQST line. As a result, the DMAHLDACK signal goes inactive almost immediately. On the next clock cycle, the system-bus-interface components and 8288 bus controller are reactivated by a low state on the $\overline{88AEN}$

line, and the DMA bus-interface components are disabled by a high state on the $\overline{\text{AENDMA}}$ line. After the $\overline{\text{88AEN}}$ signal goes active-low, the 8288 does not drive the control bus until at least 105 ns (nanoseconds) and not more than 275 ns have elapsed, if a local-bus master has a bus cycle pending. A section of the 74LS10 three-input NAND gate IC30 and some flip-flops (in section 2) guarantee this by delaying the $\overline{\text{84AEN}}$ signal by two clock periods from the time the $\overline{\text{88AEN}}$ line goes low.

The DMAHLDRQST signal goes inactive after the transfer of each byte, even if the channel requesting service has not dropped the request. This provides at least one machine cycle between successive DMA transfers.

To Be Continued

Since it may take you a month to digest this much information, I'll stop the first installment of this series here. (Besides, I don't want to take up the whole magazine, though I could easily do it in describing this complex project.)

Next Month and Thereafter:

In Part 2, I'll concentrate on the MPX-16's memory section, interrupt logic, and I/O-expansion bus (including a detailed definition of each signal). The third installment will discuss the serial and parallel I/O ports, floppy-disk-drive controller, and operating-system BIOS, plus any other facts needed to summarize the project. ■

To receive a complete list of Ciarcia's Circuit Cellar project kits available from the Micromint, circle 100 on the reader service inquiry card at the back of the magazine.

Editor's Note: *Steve often refers to previous Circuit Cellar articles as reference material for each month's current article. Most of these past articles are available in reprint books from BYTE Books, McGraw-Hill Book Company, POB 400, Hightstown, NJ 08520.* Ciarcia's Circuit Cellar, Volume I, *covers articles that appeared in BYTE from September 1977 through November 1978.* Ciarcia's Circuit Cellar, Volume II, *contains articles from December 1978 through June 1980.* Ciarcia's Circuit Cellar, Volume III, *contains the articles that were published from July 1980 through December 1981.*

Acknowledgments

Thanks to Jim Norris, George Martin, and Linda Spencer of Owl Electronic Laboratories for their contributions to the design.

Thanks to Mark Dahmke and Gordon Heins for their help with the documentation.

Thanks to Bill Morello and his staff at Techart Associates for their careful work in drawing the schematic diagrams.

References

1. Cantrell, Thomas. "An 8088 Processor for the S-100 Bus." Part 1, September 1980 BYTE, page 46; Part 2, October 1980 BYTE, page 62; Part 3, November 1980 BYTE, page 340.
2. Ciarcia, Steve. "Ease into 16-Bit Computing." Part 1, March 1980 BYTE, page 17. Part 2, April 1980 BYTE, page 40. Reprinted in *Ciarcia's Circuit Cellar, Volume II.* Peterborough, NH: BYTE Books, 1981, page 171.
3. Ciarcia, Steve. "The Intel 8086." November 1979 BYTE, page 14. Reprinted in *Ciarcia's Circuit Cellar, Volume II.* Peterborough, NH: BYTE Books, 1981, page 120.
4. *Component Data Catalog.* Santa Clara, CA: Intel Corporation, 1981.
5. *iAPX 88 Book.* Santa Clara, CA: Intel Corporation, 1981.
6. *iAPX 86,88 User's Manual.* Santa Clara, CA: Intel Corporation, 1981.

Build the Circuit Cellar MPX-16 Computer System
Part 2

A continued description of an 8088-based system that shares its principles of operation with the IBM Personal Computer.

Steve Ciarcia
POB 582
Glastonbury, CT 06033

This article is the second of three describing the design and operation of my most ambitious construction project to date: the Circuit Cellar MPX-16 computer system. I've written these articles with the intent of giving you a grasp of the basic functional parts of a complicated piece of electronic equipment and how these parts work together.

Because the MPX-16 is somewhat more complex than the projects I normally write about, I've had to simplify the presentation of many details to fit them into the magazine, but if you're interested in building an MPX-16, you can get all the details you need from the *MPX-16 Technical Reference and User's Manual*, which comes with the printed-circuit board available from The Micromint (see the text box on page 78). This book includes timing diagrams and listings of the MPX-16's special software.

Last month I presented an overview of the system and a discussion of the coprocessors and bus structures. This month, I'd like to continue by explaining memory, interrupts, the expansion bus, and I/O (input/output) decoding. But first, here's a recap of the MPX-16's features.

System Features

The Circuit Cellar MPX-16 computer system fundamentally consists of a single 9- by 12-inch five-layer printed-circuit board (containing 120 integrated-circuit packages), to which various peripheral devices are attached. It has nine expansion slots and is completely compatible with the I/O-expansion bus of the IBM Personal Computer.

The MPX-16 uses the Intel 8088 microprocessor and the optional 8087 numeric coprocessor; the main circuit board has room for 256K bytes of user memory and contains two serial and three parallel I/O ports, a floppy-disk controller, and EPROMs (erasable programmable read-only memories) containing the BIOS (basic I/O system) module of Digital Research's CP/M-86 16-bit disk operating system. The MPX-16 can be readily expanded to provide a full 1 megabyte of user memory and several megabytes of hard-disk mass storage. A more detailed list of characteristics appears in table 1.

The MPX-16 was initially designed to run CP/M-86, but eventually Microsoft's MS-DOS operating system will be available for it, making it possible to run most software written for the IBM Personal Computer on the MPX-16, except software that uses unique features of the IBM PC. The principal difference is this: with the present BIOS, the MPX-16 communicates with the user through a serially interfaced display terminal instead of a memory-mapped video display. (You could theoretically install an IBM Color Graphics Display Adapter and a serial IBM-type keyboard for exact hardware emulation.)

The MPX-16 is well suited for use as a low-cost 8088-based computer for integration into a complete hardware/software package, chiefly because it combines so many functions on a single printed-circuit board. Putting together the hardware of a complete system, you need only add a power supply, a serial video-display or printing terminal, and one floppy-disk drive (either 5¼- or

IBM and IBM Personal Computer are trademarks of International Business Machines Corporation.
CP/M-86 is a trademark of Digital Research Inc.

1. designed to use a 5-MHz Intel 8088 microprocessor, which combines a 16-bit architecture with an 8-bit bus interface and has 20-bit addressing capability for up to 1 megabyte of system memory, operating in maximum mode to support multiprocessing
2. optional Intel 8087 math coprocessor
3. onboard space for four 64K-byte banks of dynamic RAM for a total of up to 256K bytes, with parity generation and error detection
4. sockets for up to 64K bytes of JEDEC 24- or 28-pin standard ROM or EPROM devices
5. two RS-232C serial I/O ports
6. two 8-bit general-purpose parallel I/O ports with handshaking control lines
7. one Centronics-compatible parallel printer port
8. four programmable timers (one for a real-time clock, two for data rates, one for memory-refresh requests)
9. four independent DMA (direct memory access) channels
10. sixteen levels of vectored, prioritized interrupt control
11. single- or double-density floppy-disk controller for controlling up to four 5¼-inch or 8-inch single- or double-sided drives
12. five 62-pin I/O-expansion-channel connectors (hardware compatible with the IBM Personal Computer) with space for four more
13. five-layer 9- by 12-inch printed-circuit board
14. BIOS for CP/M-86 in EPROM

Table 1: *Major characteristics of the MPX-16 computer system.*

Start Address			
Decimal	Hexadecimal	Bank	Function
0	00000	0	64K to 256K bytes of R/W memory on system board
64K	10000	1	
128K	20000	2	
192K	30000	3	
256K	40000	4	up to 704K bytes of expansion memory in I/O channel
320K	50000	5	
384K	60000	6	
448K	70000	7	
512K	80000	8	
576K	90000	9	
640K	A0000	10	
704K	B0000	11	
768K	C0000	12	
832K	D0000	13	
896K	E0000	14	
960K	F0000	R	64K bytes of system ROM/EPROM

Figure 1: *Map of memory-address-space allocation in the MPX-16, in 64K-byte increments.*

8-inch). Turn on the power, insert a CP/M-86 disk, and go. And by the time you read this, an enclosure for the circuit board should be available. Many applications need nothing more.

System Memory

The stars of the show in November's article were the Intel 8088 microprocessor and the 8087 numeric processor extension (NPX), with supporting roles played by the Intel 8284 clock generator/driver, the 8288 bus controller, and the 8237A-5 DMA (direct memory access) controller. This month we look at some less glamorous but equally necessary components, starting with a type of component so prosaic as to be called a commodity by the semiconductor industry: the memory.

The MPX-16 system circuit board contains both read-only and read/write memory. In addition to the possible 64K bytes of ROM, the MPX-16 circuit board contains sockets for up to 256K 9-bit words (an 8-bit byte plus a parity bit) of dynamic RAM (random-access read/write memory). Furthermore, to augment the onboard memory, as much as 704K bytes of expansion RAM or ROM can be added in the I/O-expansion slots using readily available memory-expansion boards such as the Quadram Quadboard or the Seattle Computer RAM-Plus card. A memory map of the 8088's 1-megabyte (1,048,576-byte) address space in 64K-byte increments is shown in figure 1. Two of the five sections of the schematic diagram are included in this article; section 2 appears as figure 2 on pages 48, 49, and 50; section 3 as figure 3 on pages 52, 53, and 54. A table of integrated circuits, giving their type, location, and power connections, appears on pages 56 and 60 as table 2.

ROM Configuration

Four integrated-circuit sockets, designated IC82 through IC85 in section 3 of the schematic diagram, are provided for holding ROM (read-only memory) chips, which most often are EPROM devices. These four JEDEC- (Joint Electron Device Engineering Council) standard 28-pin sockets can contain several sizes of EPROMs, any of the various "byte-wide" (8-bit word size) devices such as the 2716 (16K bits or 2K bytes), the 2732 (4K bytes), the 2764 (8K bytes) or the 27128 (16K bytes). EPROMs with 24-pin packages, such as the 2716s and 2732s, are plugged into the lower 24 pins of the sockets, with certain jumper connections set accordingly.

For proper operation, the MPX-16 circuit board must contain a ROM or EPROM device in the highest address space (socket IC85) and a bank of RAM in the lowest address space because the 8088 processor fetches its first instruction after a power-up reset from location hexadecimal FFFF0 (usually a jump instruction branching to an initialization routine) and uses interrupt vectors in the range hexadecimal 00000 to 003FF.

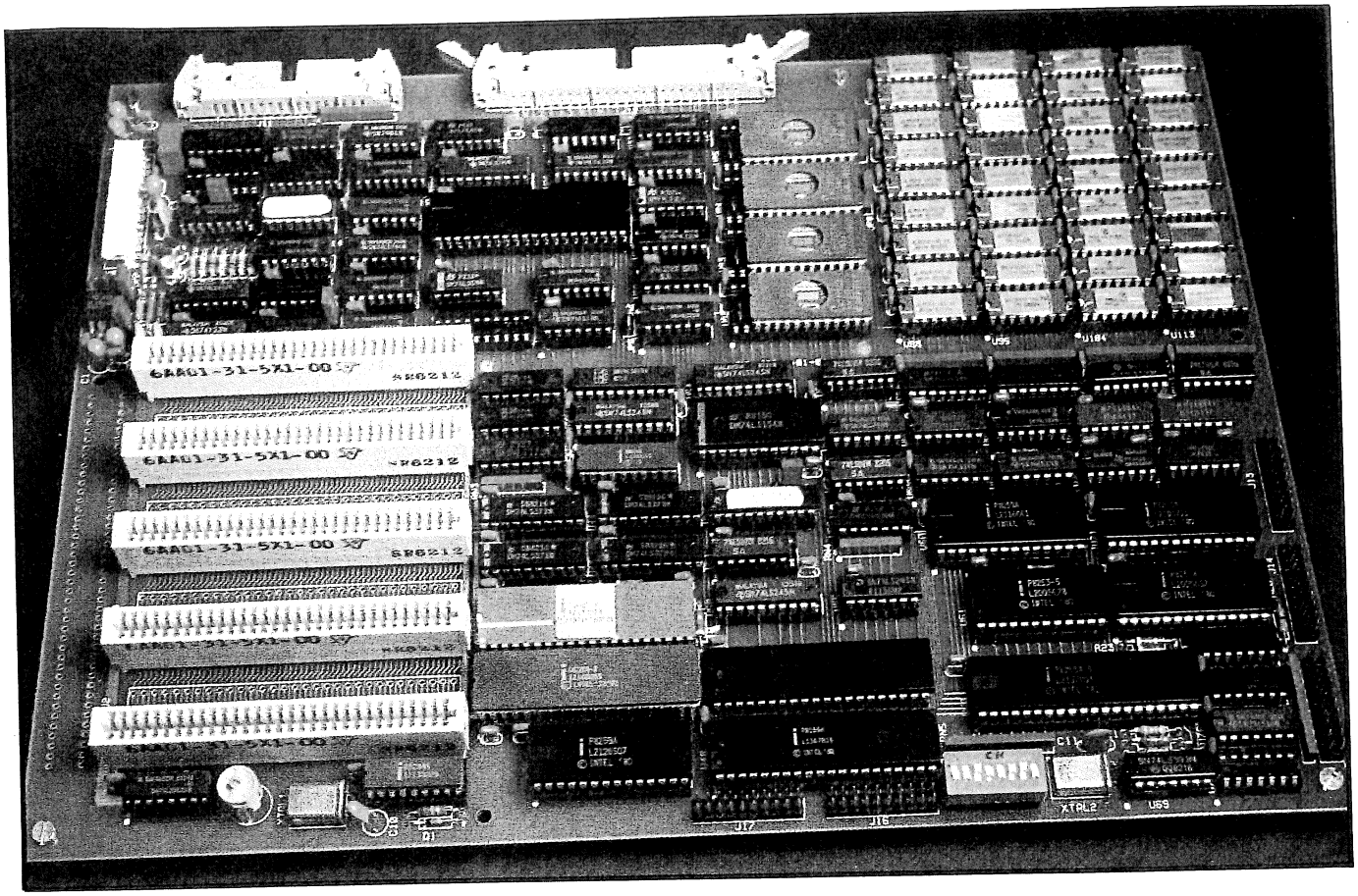


Photo 1: *The Circuit Cellar MPX-16 single-board computer system, which uses the latest technology to provide lots of low-cost computing power. The five-layer printed-circuit board contains 120 integrated circuits including most common peripheral-device interfaces; furthermore, any peripheral-device card intended for use with the IBM Personal Computer can be plugged into one of the I/O-expansion slots. There are nine slot positions, but only five sockets are installed initially.*

The capacity of the ROMs (or EPROMs) used on the system board must be compatible with the configuration of onboard jumpers JP1 through JP6 and with the program stored in the 32-word by 8-bit address-decoding PROM (programmable ROM) device IC45, an HM7603. The PROM program and jumper arrangements supplied with the system board are intended for type-2732 EPROMs. A different decoding PROM is needed for other memory-device types so that the four ROM sockets may be decoded into a contiguous address space in each case. (A PROM-programming table is included in the MPX-16 documentation.) The ROM-decoding logic and memory organization are respectively shown in sections 2 and 3 of the schematic diagrams.

The ROM-address-space-decoding logic for the system board is enabled whenever all three high-order system address bits, SYSA17 through SYSA19, are high, causing the output of a NAND gate (IC30) to go low. If five PROM-address bits SYSA11 through SYSA15 or SYSA12 through SYSA16 (depending on the jumper configuration) address one of the programmed locations, the selected ROM-chip-enable line (one of $\overline{\text{PROMSEL0}}$ through $\overline{\text{PROMSEL3}}$) is also driven low, selecting that memory device. The $\overline{\text{ROMSEL}}$ signal at IC28 pin 5 (a two-input OR gate in section 2) also enables a wait-state-generation circuit if jumper JP7 is connected. After one of the $\overline{\text{PROMSEL}}x$ lines has been driven active-low, a $\overline{\text{SYSMEMRD}}$ (system memory read, active-low) signal from the system bus master will initiate the memory-read cycle and generate a single wait state if JP7 is connected. Valid data from the ROMs is available on the data bus after $\overline{\text{SYSMEMRD}}$ goes low.

Memory chips are less glamorous than microprocessors, but just as necessary.

Normally, the MPX-16 requires ROM or EPROM devices with an access time of 350 ns (nanoseconds) or faster. The optional wait-state feature afforded through JP7 allows use of slower ROM devices with 450-ns access times. If faster devices are used, then JP7 should not be installed and the MPX-16 can operate with no wait states.

The EPROMs on the standard MPX-16 system board contain a power-on self-test routine and I/O drivers, including the CP/M-86 BIOS

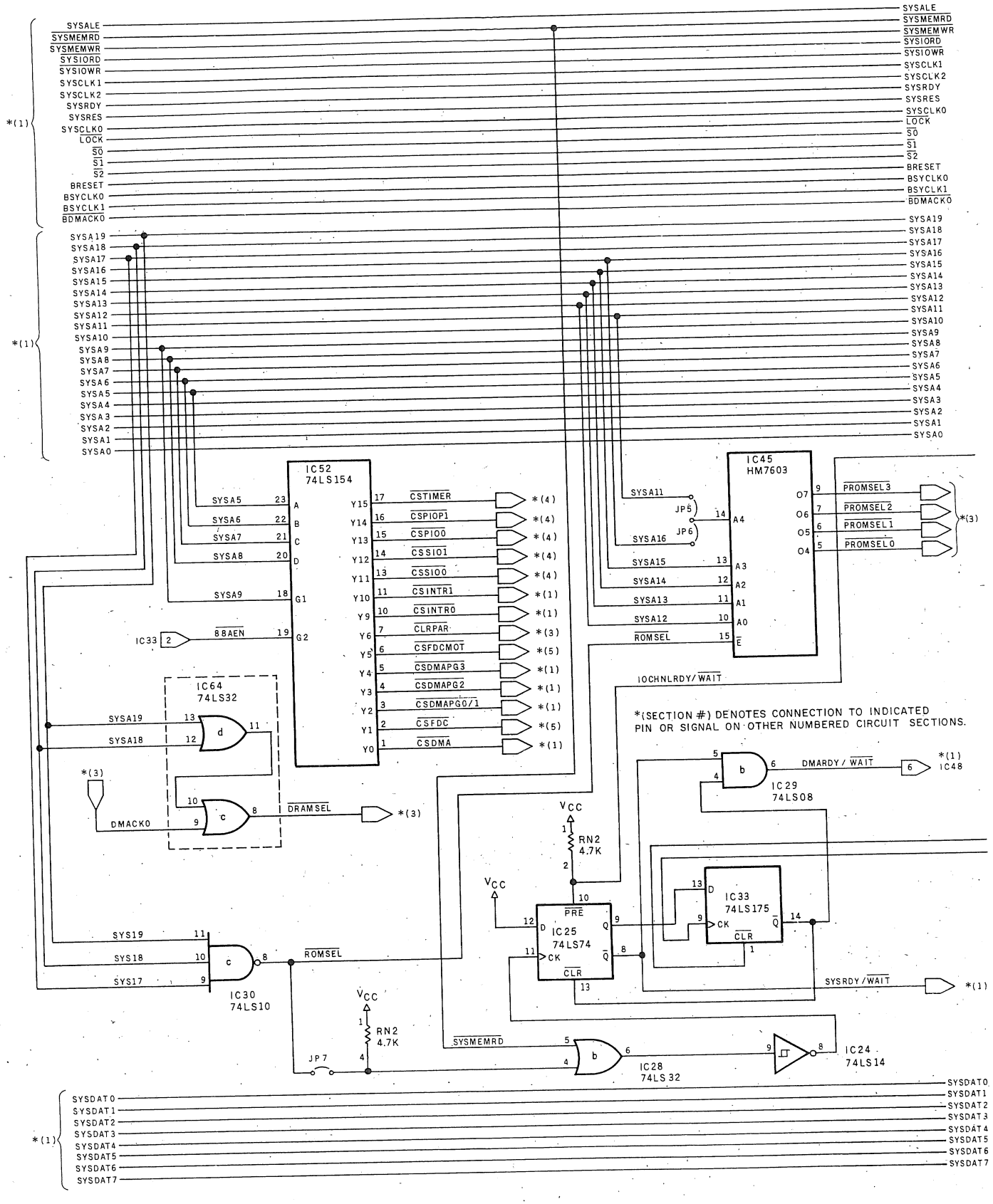
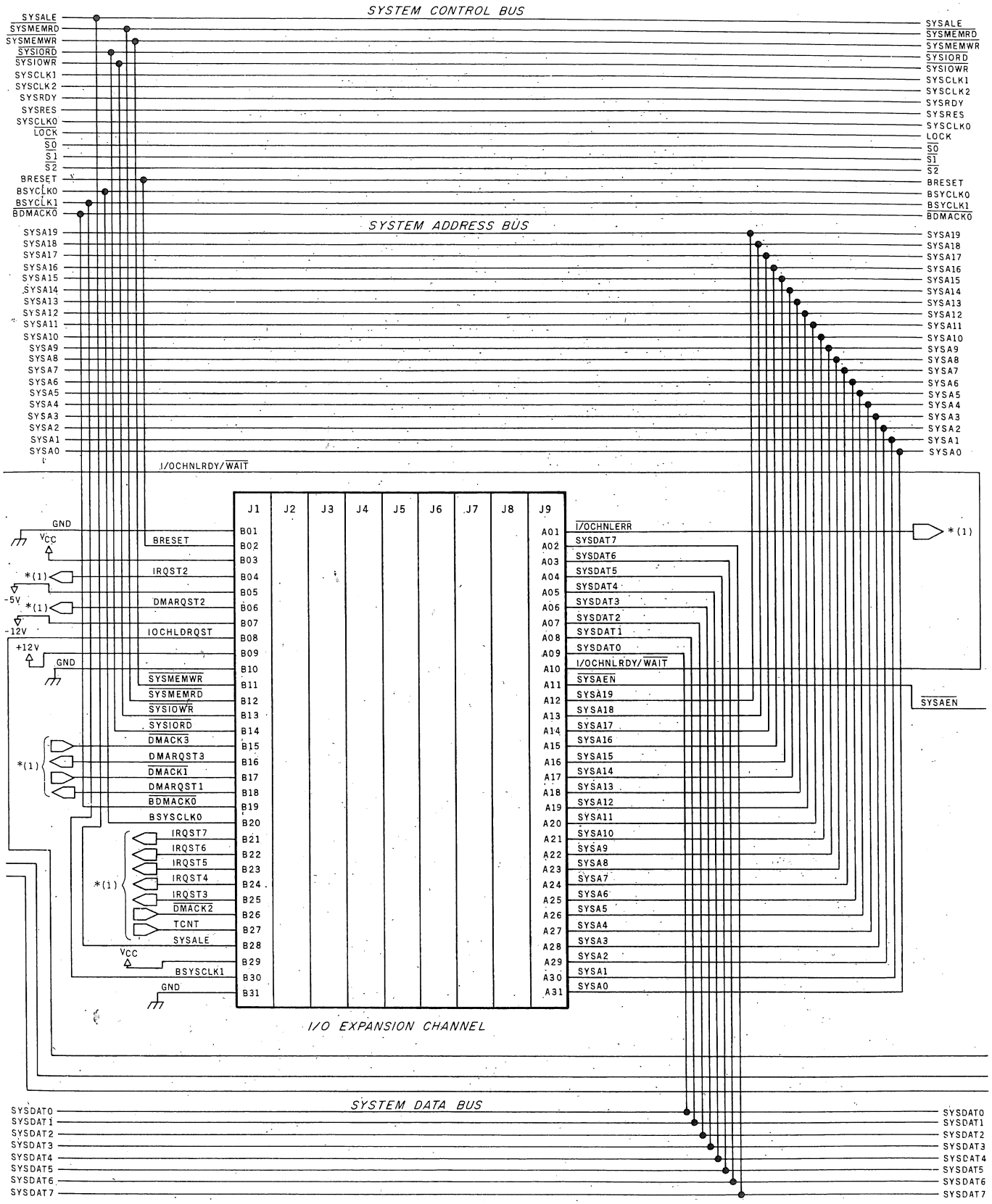


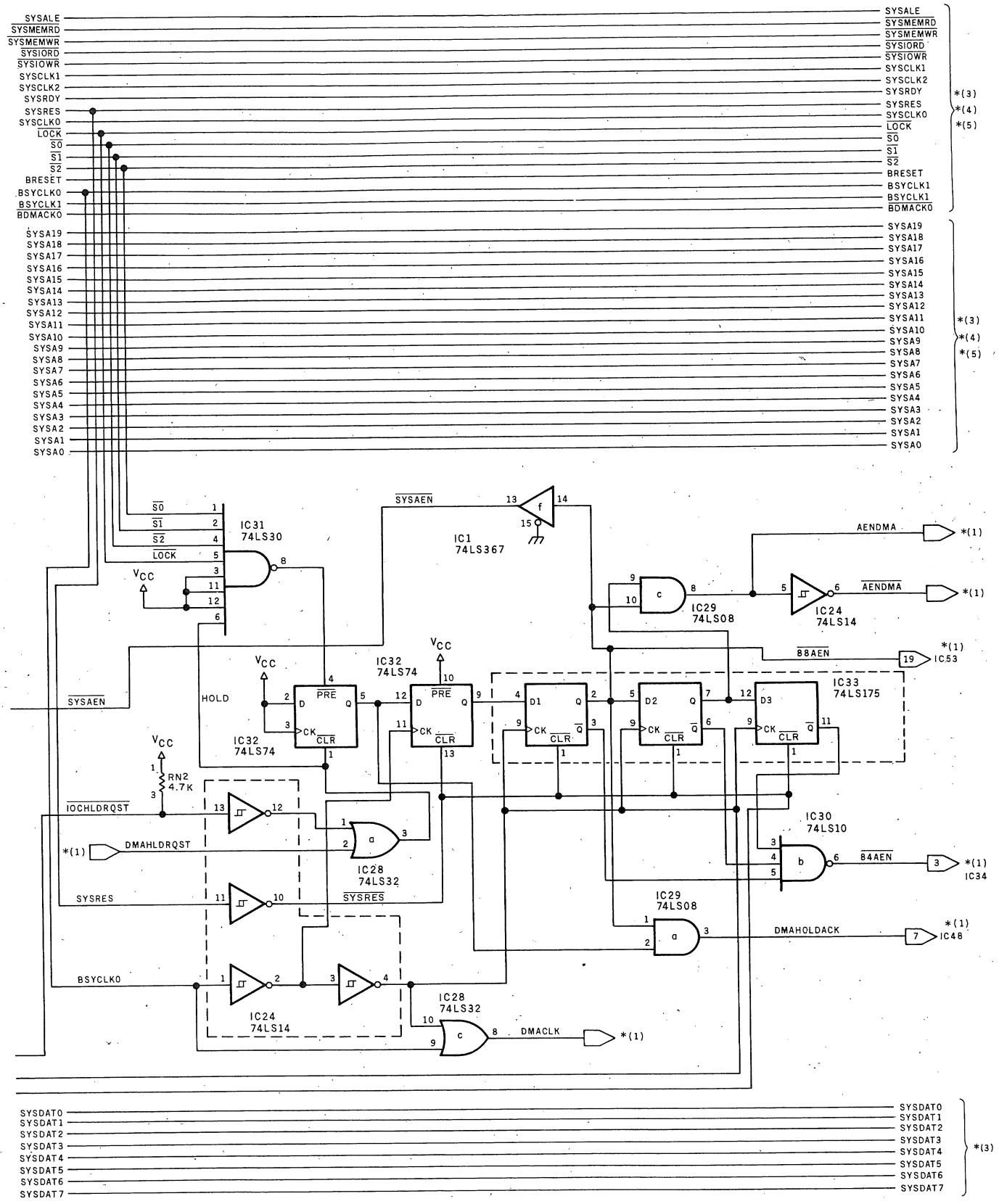
Figure 2: *Section 2 of the schematic diagram of the MPX-16 computer's main circuit board. (Section 1 was printed in last month's ar ticle; section 3 appears here in figure 3; sections 4 and 5 will appear in next month's article.) The notation *(n) indicates that a give signal line connects to a component or another line shown in schematic section n.*

A table of all the MPX-16's integrated circuits appears as table 2 on pages 56 and 60, giving each device's number, type, se



tion/figure location, and power connections. Connections to the I/O-expansion-channel slots are of course made to each individual slot. Possible substitutes for the HM7603 are the 74S288, the 82S123, and the AM27S09, although it's best to use the HM7603. (The diagram is continued on page 50.)

Figure 2: *Continued from page 49.*



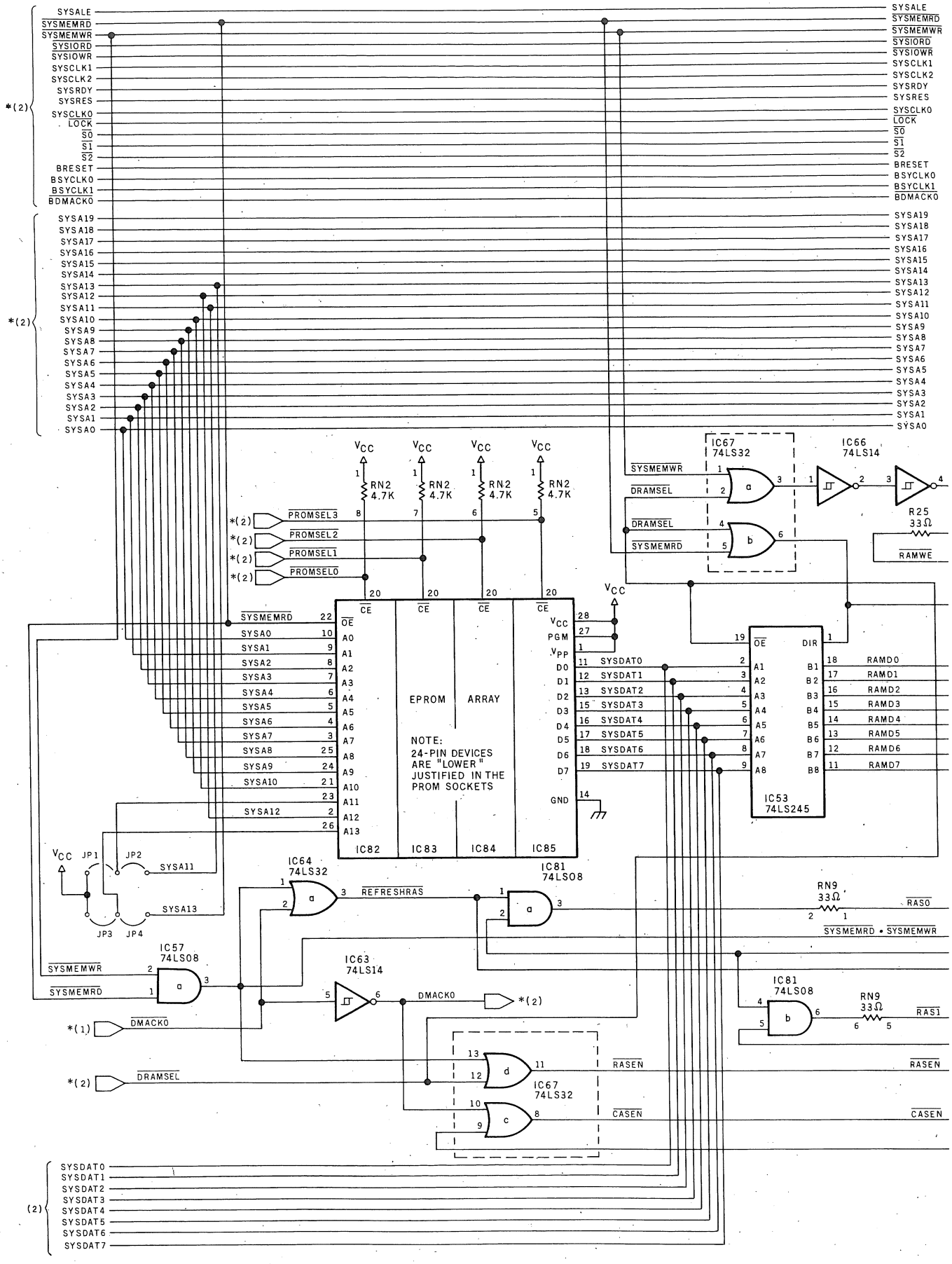
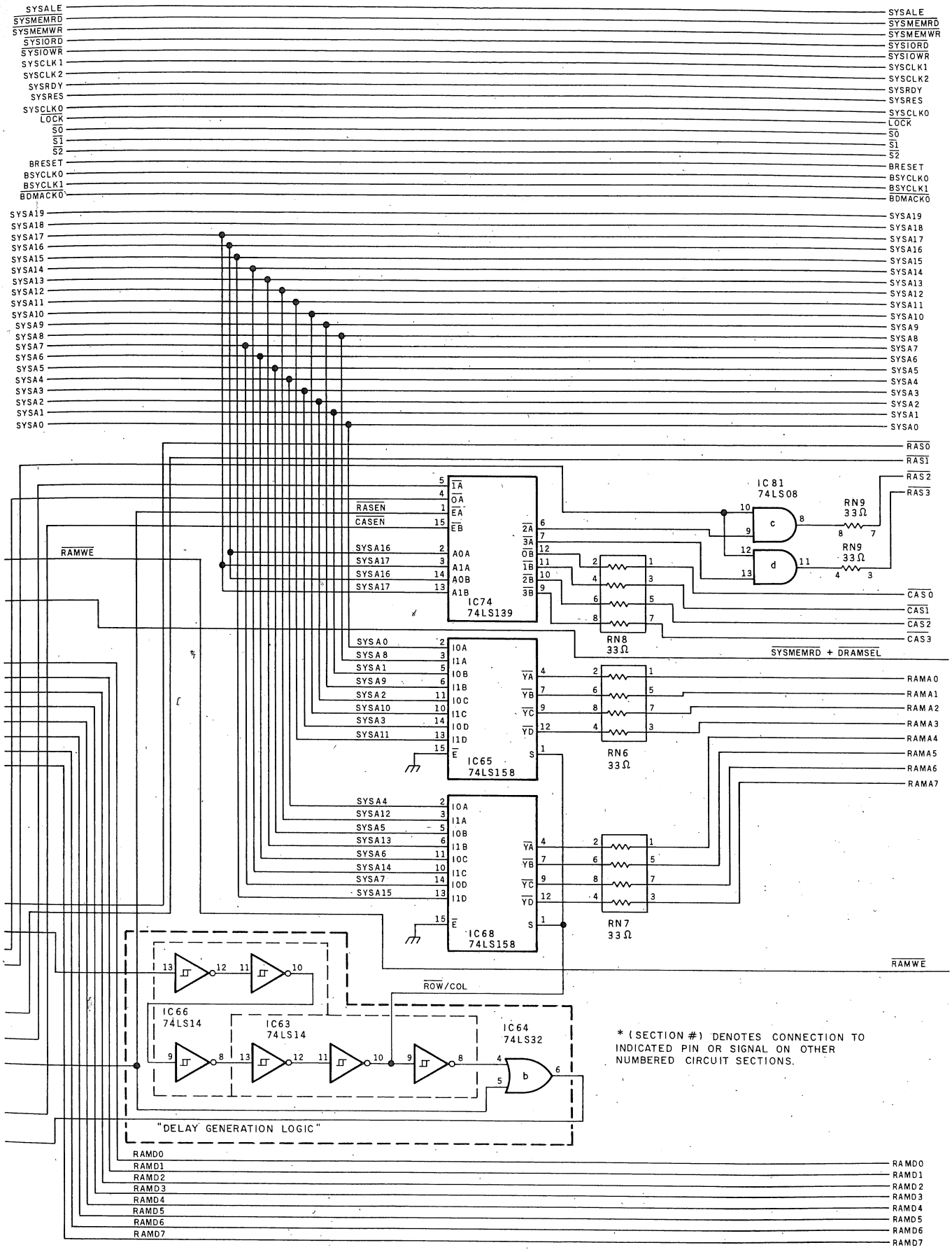


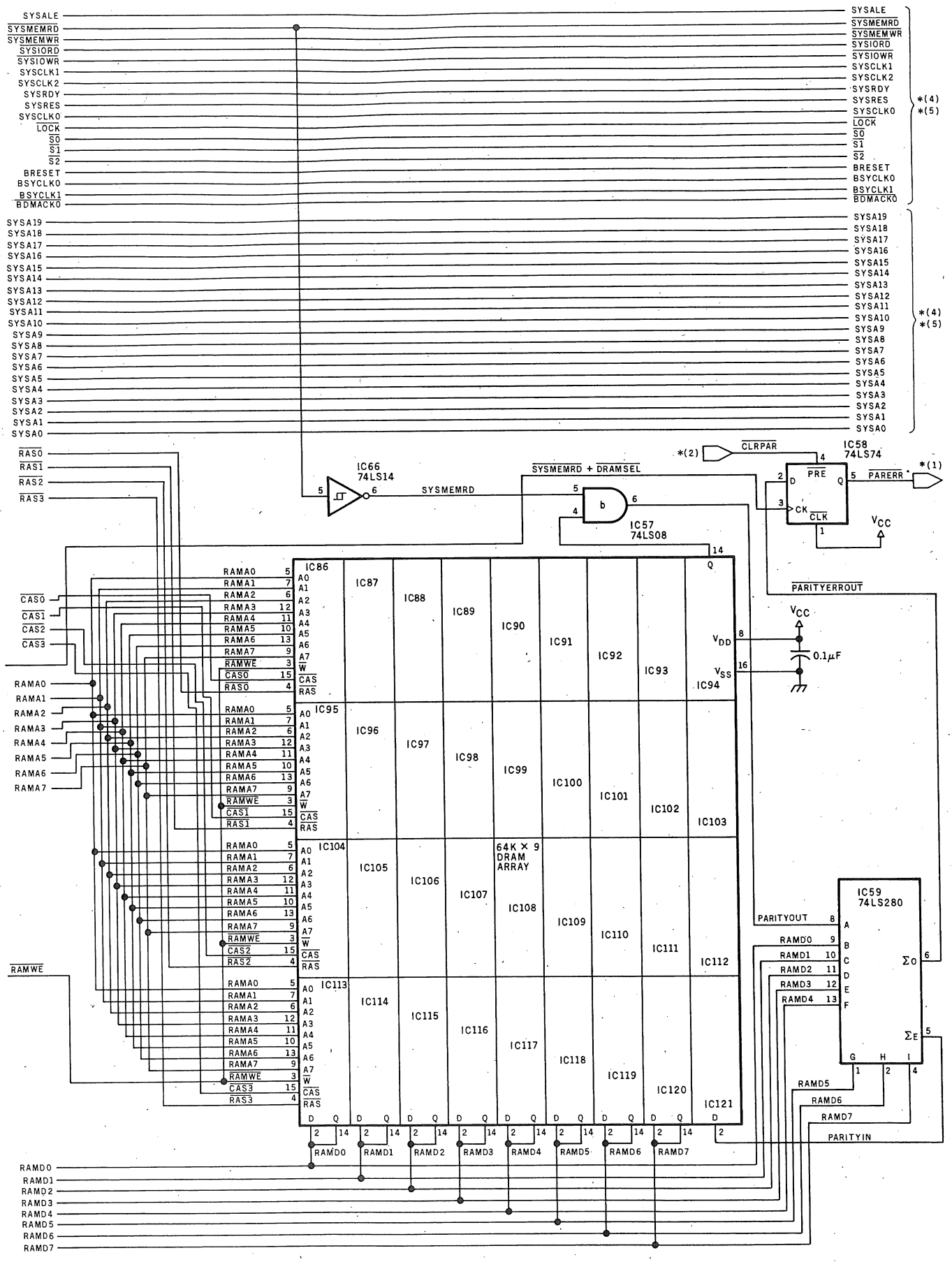
Figure 3: *Section 3 of the schematic diagram of the MPX-16 computer's main circuit board. The notation *(n) indicates that a give signal line connects to a component or another line shown in schematic section n.*

Connections shown on the edges of the dynamic-memory array on page 54 are of course made to each individual chip. Bypa



capacitors, not shown, should be installed adjacent to most integrated circuits between +5 V and ground. A table of all the MPX-16's integrated circuits appears as table 2 on pages 56 and 60, giving each device's number, type, section/figure location, and power connections. (The diagram is continued on page 54.)

Figure 3: *Continued from page 53.*



IC Number	Type	Schematic Section	+5 V	GND	+12 V	-12 V
VR1	LM7905	5 (3-2)	(voltage regulator)			3
IC1	74LS367	1 (1-3b), 2 (2-2)	16	8		
IC2	74LS123	5 (3-2)	16	8		
IC3	74LS157	5 (3-2)	16	8		
IC4	74LS124	5 (3-2)	16	8		
IC5	74LS175	5 (3-2)	16	8		
IC6	74LS173	5 (3-2)	16	8		
IC7	74LS393	5 (3-2)	14	7		
IC8	74LS10	5 (3-2)	14	7		
IC9	74LS74	5 (3-2)	14	7		
IC10	M1116-8M	5 (3-2)	14	7		
IC11	74LS153	5 (3-2)	16	8		
IC12	74LS14	5 (3-2)	14	7		
IC13	74LS74	5 (3-2)	14	7		
IC14	74LS74	5 (3-2)	14	7		
IC15	74LS74	5 (3-2)	14	7		
IC16	74LS74	5 (3-2)	14	7		
IC17	74LS175	5 (3-2)	16	8		
IC18	7406	5 (3-2)	14	7		
IC19	spare socket					
IC20	74LS04	1 (1-3ab), 5 (3-2)	14	7		
IC21	8272	5 (3-2)	40	20		
IC22	74LS240	5 (3-2)	20	10		
IC23	7407	5 (3-2)	14	7		
IC24	74LS14	2 (2-2)	14	7		
IC25	74LS74	2 (2-2)	14	7		
IC26	74LS139	5 (3-2)	16	8		
IC27	7407	5 (3-2)	14	7		
IC28	74LS32	2 (2-2), 4 (3-1)	14	7		
IC29	74LS08	2 (2-2)	14	7		
IC30	74LS10	1 (1-3a), 2 (2-2)	14	7		
IC31	74LS30	2 (2-2)	14	7		
IC32	74LS74	2 (2-2)	14	7		
IC33	74LS175	2 (2-2)	16	8		
IC34	8284A	1 (1-3a)	18	9		
IC35	8259A	1 (1-3a)	28	14		
IC36	8088	1 (1-3a)	40	1,20		
IC37	8087 (option)	1 (1-3a)	40	1,20		
IC38	74LS373	1 (1-3a)	20	10		
IC39	74LS373	1 (1-3b)	20	10		
IC40	74LS173	1 (1-3b)	16	8		
IC41	74LS173	1 (1-3b)	16	8		
IC42	74LS173	1 (1-3b)	16	8		
IC43	74LS245	1 (1-3a)	20	10		
IC44	74LS373	1 (1-3a)	20	10		
IC45	HM7603-5	2 (2-2)	16	8		
IC46	74LS245	1 (1-3a)	20	10		
IC47	8155H-2	4 (3-1)	40	20		

Table 2: *Integrated circuits in the MPX-16. Here are shown each device's number, type, section/figure location, and power connections.*

The location of each chip in the five-part schematic diagram is listed by schematic section; the characters in parentheses show in which article the section appeared and which figure the device appears in. Some integrated circuits containing multiple gates appear in more than one schematic section. *(The table is continued on page 60.)*

IC Number	Type	Schematic Section	+5 V	GND	+12 V	-12 V
IC48	8237A-5	1 (1-3b)	31	20		
IC49	74LS245	1 (1-3a)	20	10		
IC50	74LS373	1 (1-3a)	20	10		
IC51	8288	1 (1-3a)	20	10		
IC52	74LS154	2 (2-2)	24	12		
IC53	74LS245	3 (2-3)	20	10		
IC54	74LS243	1 (1-3b)	14	7		
IC55	74LS08	1 (1-3ab), 5 (3-2)	14	7		
IC56	74LS32	1 (1-3a)	14	7		
IC57	74LS08	3 (2-3)	14	7		
IC58	74LS74	3 (2-3), 4 (3-1)	14	7		
IC59	74LS280	3 (2-3)	14	7		
IC60	8255A-5	4 (3-1)	26	7		
IC61	8253-5	4 (3-1)	24	12		
IC62	8259A	1 (1-3a)	28	14		
IC63	74LS14	1 (1-3a), 3 (2-3)	14	7		
IC64	74LS32	2 (2-2), 3 (2-3)	14	7		
IC65	74LS158	3 (2-3)	16	8		
IC66	74LS14	3 (2-3)	14	7		
IC67	74LS32	3 (2-3)	14	7		
IC68	74LS158	3 (2-3)	16	8		
IC69	74LS393	4 (3-1)	14	7		
IC70	8251A	4 (3-1)	26	4		
IC71	8251A	4 (3-1)	26	4		
IC72	1489	4 (3-1)	14	7		
IC73	1489	4 (3-1)	14	7		
IC74	74LS139	3 (2-3)	16	8		
IC75	74LS00	1 (1-3b)	14	7		
IC76	74LS14	4 (3-1)	14	7		
IC77	7407	4 (3-1)	14	7		
IC78	7407	4 (3-1)	14	7		
IC79	1488	4 (3-1)		7	14	1
IC80	1488	4 (3-1)		7	14	1
IC81	74LS08	3 (2-3)	14	7		
IC82	EPROM	3 (2-3)	28,1*	14		
IC83	EPROM	3 (2-3)	28,1*	14		
IC84	EPROM	3 (2-3)	28,1*	14		
IC85	EPROM	3 (2-3)	28,1*	14		
IC86	4164	3 (2-3)	8	16		
↓	↓	↓	↓	↓		
IC121	4164	3 (2-3)	8	16		

* depends on type of EPROM used

Table 2: *Continued from page 56.*

and a floppy-disk bootstrap-loader routine.

RAM Configuration

The onboard user-programmable memory of the MPX-16 consists of one to four 64K-byte banks of nine type-4164 64K-bit dynamic RAM devices. Within the 8088 processor's 1-megabyte address space, the MPX-16 must have at least the lowest 64K-byte bank of RAM (bank 0) installed from hexadecimal addresses 00000 to 0FFFF so that interrupt-routine pointers can reside in the locations from hexadecimal 00000 to 003FF. The RAM chips are required to have an access time of no more than 200 ns and a cycle time of 335 ns. Single-bit parity generation and

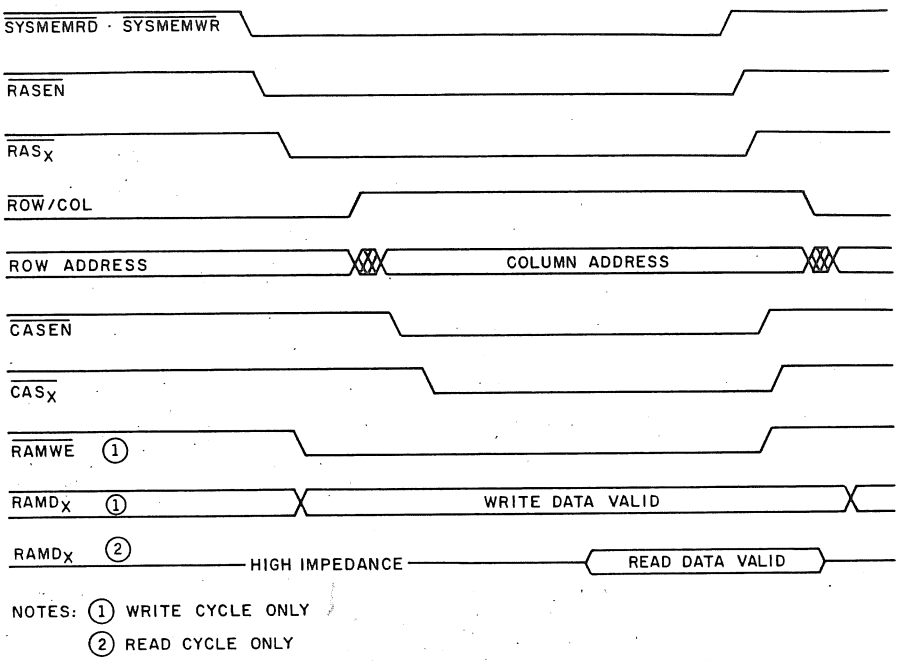


Figure 4: *Timing diagram for the memory operation of the MPX-16.*

error detection are provided for all of the 256K-byte onboard memory.

The RAM address-decoding logic is shown in section 2 of the schematic diagram (figure 2), and the read/write control logic, address multiplexers, RAM array organization, and parity-generation/error-detection logic are shown in section 3 (figure 3). The onboard RAM address space is selected when two conditions are met: the two high-order address bits SYSA18 and SYSA19 are both low and a memory-refresh cycle is not in progress (shown by DMACK0, the DMA-channel-0-acknowledge signal, being low). Because of this decoding scheme and the fact that the MPX-16 power-on self-test routine automatically clears memory and determines its size, the full 256K bytes of onboard RAM should be installed before you put in additional RAM in the I/O-expansion slots.

Dynamic Memory Refresh

Because dynamic RAM devices are used for the MPX-16's programmable memory, a memory-refresh circuit is necessary to prevent data stored in them from being lost. The 64K-bit dynamic RAMs require that all 256 rows be addressed every 4 ms (milliseconds) to maintain the integrity of the data (the columns need not be individually addressed); one row must be addressed for refreshing approximately every 15 μs (microseconds). To eliminate having a separate bus-arbitration circuit for this purpose, memory refresh is carried out by executing a DMA (direct memory access) read cycle in a "RAS-only" manner—that is, using only the row-address-strobe inputs of the memory chips. Because refresh is controlled by the DMA circuit, there can never be a conflict between the refresh operation and the processor's memory references.

The $\overline{\text{DMACK0}}$ signal goes active-low to indicate to the rest of the system that a refresh cycle is in progress. This signal disables the RAM-decoding circuitry, prevents the generation of a CAS (column-address strobe) signal, and enables the $\overline{\text{REFRESHRAS}}$ input at IC64 pin 2 (in section 3, figure 3). When the system bus master, the 8237A DMA controller (IC48 in section 1, printed last month), drives the $\overline{\text{SYSMEMRD}}$ or $\overline{\text{SYSMEMWR}}$ (system memory write) line low, the output at IC 64 pin 3 also goes low. This causes the outputs of the four two-input gates (sections of IC81: positive AND gates used as negative ORs), whose other input comes from IC74, to go low. These outputs form the RAS inputs for each of the four RAM banks. (The 33-ohm series resistors in the RAS control lines are there to reduce ringing on the lines, which might latch a new row address during the middle of the memory cycle.) The DMA controller is set up by the system-initialization software to automatically increment the address counter after each refresh-memory cycle.

Memory Operation

A diagram of typical timing cycles for normal memory-read and write operations is shown in figure 4. For either type of memory cycle, the read/write-control logic is enabled when the $\overline{\text{DRAMSEL}}$ signal is low, indicating that two conditions have both been fulfilled: a valid address (lower than hexadecimal C0000) has been latched on the system bus and the DMACK0 signal (from IC63, pin 6) is low (indicating that a refresh cycle is not in progress).

A memory cycle is initiated when the output of an AND gate (IC57 pin 3 in section 3) goes low, indicating that either the $\overline{\text{SYSMEMRD}}$ or the $\overline{\text{SYSMEMWR}}$ control signal has been driven low by the system bus master. The $\overline{\text{RASEN}}$ (RAS enable) signal at IC67 pin 11, produced from the output of IC57 ORed with $\overline{\text{DRAMSEL}}$, enables the 1-of-4 (2- to 4-line) decoder IC74 to select one of the four lines $\overline{\text{RAS0}}$, $\overline{\text{RAS1}}$, $\overline{\text{RAS2}}$, or $\overline{\text{RAS3}}$ (row-address-input enable for each of the four banks—which one is selected depends on the logic levels of the SYSA16 and SYSA17 address lines) and sets up the row address on the multiplexed memory-address lines RAMA0 through RAMA7. A chain of Schmitt-trigger inverter sections, IC63 and IC66, delays the active-low output from IC57 pin 3 by five gate-delay periods, holding the row-address condition until the type-4164 memory chips have had sufficient time to latch the address bits.

When the $\overline{\text{ROW/COL}}$ signal goes high (column addressing active), the

multiplexers change the contents of RAMA0 through RAMA7 to the column address derived from the system-address-bus lines SYSA8 through SYSA15. The CASEN signal enables the B outputs of the 1-of-4 decoder IC74, which drives the CAS-control line for one of the memory banks.

The data-input and data-output lines of each RAM chip are tied together onto a common bidirectional memory-data line. The entire RAM array is isolated from the system data bus by bus transceiver IC53, which is enabled by the DRAMSEL signal during nonrefresh memory cycles, allowing data to pass between the RAM array and the system data bus.

The direction of data flow is controlled by the output of IC67 pin 6, a logical OR of DRAMSEL and SYSMEMRD. During memory-read cycles, this signal is low, causing the data on the memory data bus to be transferred to the system data bus. During memory-write cycles, the direction signal is high and the data flow is from the system data bus to the memory data bus.

Parity Checking

Until the introduction of the IBM Personal Computer, memory with parity checking was rare in personal computers but had been used for years in larger computers. IBM did well to copy this feature of larger machines, since the constant decreases in memory prices have made it more and more cost-effective. The MPX-16 also incorporates parity memory for increased system reliability and user confidence. Parity generation and checking in the MPX-16 are provided by a 74LS280 parity generator (IC59) and a type-D flip-flop (IC58), shown in figure 3 on page 54.

During a memory-write cycle, the PARITYOUT signal presented to pin 8 of IC59 is low, because the output of IC57 (an AND gate) is disabled by the low state of the active-high SYSMEMRD signal. The parity bit computed by IC59 from the eight RAM data lines is written into the parity-bit memory chip (the ninth one of each bank) for the bank being addressed.

When a memory-read cycle occurs, the output of IC57 is enabled, and the parity bit that was previously written for each byte is routed to IC59 and used to check for an error in the parity value. When the rising edge of the signal from IC67 pin 6 (DRAMSEL OR SYSMEMRD) is detected by the flip-flop IC58, it latches parity value.

When no parity error is present, the odd-parity output (Σ O) of the 74LS280 will be a logic high state. When an error does occur, the odd-parity output will be low. The PARERR signal from IC58 is sent to the NMI (nonmaskable interrupt) logic and will remain set until the next memory-read cycle for which no parity error occurs, or until the flip-flop is preset by a low state on the CLRPAR (clear parity) line, IC58

pin 4. Software called through the interrupt vector then notifies the user of a memory error.

Interrupt Advantages

The versatility of any computer system is enhanced if its processing can be interrupted by outside events so that it doesn't have to continually keep track of what is going on in the outside world. The MPX-16 supports an interrupt system with 16 levels of interrupt priority, for a high degree of versatility in dealing with the external environment.

Perhaps the major advantage to using interrupts is the increase in throughput resulting from their use in handling the system I/O functions. Instead of the processor's spending a great deal of time checking to see if I/O devices are ready to transfer data or waiting for them to be ready, in an interrupt-driven system the processor can continue executing its application program, only suspending execution to attend to an I/O device when the device signals that it is actually ready for data transfer.

Although it can be tougher to debug, interrupt-driven software is generally more compact and efficient than that which must explicitly check I/O devices by polling or waiting. But we don't have space here to discuss the software aspects at length.

MPX-16 Interrupt Logic

The interrupt structure of an 8088-based system revolves around an interrupt-vector lookup table located low in system memory from location hexadecimal 00000 through 003FF. Each interrupt vector in the table consists of 4 bytes that point to the address of an interrupt-service routine. Up to 256 interrupt vectors, numbered from decimal 0 to 255, can be used to specify starting addresses of interrupt routines anywhere in the 8088's 1-megabyte address space. Each of the interrupt vectors is assigned an interrupt-type number that points to its location in the lookup table. The type number multiplied by 4 equals the offset of the vector from location 00000.

The highest priority interrupt is the

nonmaskable-interrupt (NMI) input at pin 17 of the 8088 microprocessor, IC36. This signal is an internally synchronized edge-triggered input which causes a predefined "type-2" interrupt that "vectors" (passes control) to the location identified by the eighth position in the table. Although the 8088's NMI input is not directly maskable by software, the MPX-16 contains extra hardware that can mask the interrupt signal before it gets to the 8088, given proper setup by the software. The NMI input is used to report system memory-parity errors and errors from the I/O-channel expansion slots.

The next 15 levels of interrupts are implemented by two Intel 8259A programmable interrupt controllers (PICs), IC35 and IC62 in section 1 of the schematic diagram, which was printed in last month's article. One of the programmable interrupt controllers, IC35, serves as the master and resides on the multiplexed local bus shared with the processors. The other, IC62, is a slave device to IC35 and resides on the system bus. The master/slave configuration is set up during the initialization process by software.

All of the peripheral devices residing on the system board, such as the serial and parallel I/O-port controllers, are supported by interrupt-request lines on the 8259A PICs. Interrupt requests from the PICs drive the INTR input of the 8088 (pin 18). This signal is a level-triggered input that can be internally masked by a software instruction. Interrupts requested by the INTR input do not have predefined vector types as does the nonmaskable interrupt. In the case of the 8259A PIC, a consecutive block of eight interrupt types, one for each of the eight interrupt-request input pins, is programmed into the device by the system software as part of the initialization process when the power is turned on.

Handling Interrupts

When an interrupt signal is received on the 8088's INTR pin, the processor enters an interrupt-acknowledge cycle that is used to determine the interrupt type. First the processor preserves what it was doing when interrupted: the state of the machine is saved by pushing the contents of the flag register, code-segment register, and instruction pointer onto the stack. In addition, the interrupt flag is cleared, disabling further interrupts from occurring until the processor is ready for them. (If nested interrupts are desired, the interrupt-service routine must reenable the processor to receive interrupts, while ensuring that the most crucial tasks are not delayed until too late. The programming is not easy.)

In the next step, the 8288 bus controller (IC51) issues two interrupt-acknowledge pulses on the $\overline{\text{INTA}}$ line. The first pulse signals the 8259A PICs that the interrupt request is being granted. When the second $\overline{\text{INTA}}$ pulse is issued, the 8-bit code for the interrupt type is placed onto the data bus. The value of the interrupt type is multiplied by 4 (simply by being shifted left 2 places) to determine the

Priority Level	Source	Signal Name	Description
0	NMI	PARERR or IOCHNLERR	memory-parity or I/O-channel errors
1	master	TIMEINTR	real-time clock
2	slave	SIO0RXRDY	serial-channel-A receive ready
3	slave	SIO1RXRDY	serial-channel-B receive ready
4	slave	SIO0TXRDY	serial-channel-A transmit ready
5	slave	SIO1TXRDY	serial-channel-B transmit ready
6	slave	PRINTRDY	printer-port ready
7	slave	FDCINT	floppy-disk-controller interrupt
8	slave	NPXINT	numeric-processor-extension (8087) interrupt
9	slave	PIOINT	parallel-I/O-port interrupt
10	master	IRQST2	I/O-channel interrupt
11	master	IRQST3	I/O-channel interrupt
12	master	IRQST4	I/O-channel interrupt
13	master	IRQST5	I/O-channel interrupt
14	master	IRQST6	I/O-channel interrupt
15	master	IRQST7	I/O-channel interrupt

Table 3: *Interrupt signals in the MPX-16, listed in order of priority. Priority-0 errors go through the 8088's NMI input, while the rest go through either the master or the slave 8259A interrupt controller.*

address of the interrupt vector. Program control is then transferred to the address contained in the 4 bytes of the interrupt vector. Note that the first 2 bytes are used as the new instruction pointer (lower 16 bits of the address) and the second 2 bytes are used to form the new code-segment register (upper 16 bits). When the interrupt-service routine has completed execution, control is returned to the main program via an IRET instruction, which pops the original flag and address information off the stack into the active registers. The main program then resumes execution where it left off, with the interrupts reenabled.

Interrupt Priorities

The organization of the system-board interrupt-priority scheme is shown in table 3. The highest priority hardware interrupt, as we've seen, is the NMI, which is caused by memory-parity or I/O-channel errors. The highest priority maskable interrupt is from the IR0 input of the master 8259A PIC, which is generated by the real-time clock. The next eight interrupts in priority come from peripheral devices attached to the slave 8259A PIC, which is in turn attached to the IR1 input of the master 8259A. The last six interrupts come from the I/O-expansion-channel connectors. These interrupts drive the IR2 through IR7 inputs of the master 8259A.

Two other points concerning the 8259A PICs should be noted. Although a priority has been assigned to each interrupt-request input of the 8259A PICs, these can be changed by the system software. In addition, the 8259A PICs can even be used to implement a polled I/O system. (These devices provide considerable flexibility for handling I/O servicing at a relatively low hardware cost.) And finally, all of the interrupt-service routines in the MPX-16 system can be be invoked via a software-interrupt instruction that specifies the interrupt type. This can be useful in starting an I/O device and in debugging the interrupt routines.

I/O-Expansion Channels

The MPX-16 system board supports an I/O-expansion channel that represents an extension of the system bus. Peripheral devices are connected through several 62-pin card-edge connectors like those used by peripherals designed for the IBM Personal Computer. The MPX-16 computer system

Pin	Signal Name		Pin	Signal Name	
A01	I/OCHNLERR		B01	GND	
A02	SYSDAT7		B02	BRESET	
A03	SYSDAT6		B03	V_{cc}	
A04	SYSDAT5		B04	IRQST2	
A05	SYSDAT4		B05	−5 V DC	
A06	SYSDAT3	system	B06	DMARQST2	
A07	SYSDAT2	data	B07	−12 V DC	
A08	SYSDAT1	bus	B08	IOCHLDRQST	
A09	SYSDAT0		B09	+12 V DC	
A10	IOCHNLRDY/WAIT		B10	GND	
A11	SYSAEN		B11	SYSMEMWR	system-status control
A12	SYSA19		B12	SYSMEMRD	
A13	SYSA18		B13	SYSIOWR	
A14	SYSA17		B14	SYSIORD	
A15	SYSA16		B15	DMACK3	
A16	SYSA15		B16	DMARQST3	DMA control
A17	SYSA14		B17	DMACK1	
A18	SYSA13		B18	DMARQST1	
A19	SYSA12	system	B19	BDMACK0	
A20	SYSA11	address	B20	BSYSCLK0	
A21	SYSA10	bus	B21	IRQST7	
A22	SYSA9		B22	IRQST6	
A23	SYSA8		B23	IRQST5	
A24	SYSA7		B24	IRQST4	
A25	SYSA6		B25	IRQST3	
A26	SYSA5		B26	DMACK2	
A27	SYSA4		B27	TCNT	
A28	SYSA3		B28	SYSALE	
A29	SYSA2		B29	V_{cc}	
A30	SYSA1		B30	BSYSCLK1	
A31	SYSA0		B31	GND	

Table 4: *Pin/signal relationships in the I/O-expansion connectors. These assignments are compatible with those in the expansion slots of the IBM Personal Computer. Many of the system control signals are buffered before being fed to these connectors.*

can potentially contain 1 megabyte of memory and still have spare expansion slots for special-purpose I/O modules, which might include videotex decoders, process-control or data-acquisition interfaces, or local-network interfaces.

The standard MPX-16 system board has five expansion connectors installed in alternating positions, effectively located on 1-inch center-to-center spacings. An additional four connectors can be installed between them, if needed; the resulting nine connectors will be on half-inch center-to-center spacing. Spacing on 1-inch centers is usually required for disk controllers and I/O boards. Memory boards, on the other hand, will generally fit in half-inch spacing.

The I/O-expansion channel has been designed to be pin-for-pin hardware-compatible with the IBM Personal Computer (model 5150). The IBM PC bus was chosen, as I explained last month, to take advantage of the expected proliferation of IBM-PC-compatible peripheral-adapter modules and expansion memories. However, because the MPX-16 system board already supports most of the peripheral I/O functions that would ordinarily be added to the IBM computer, the I/O-expansion slots are available for new uses.

Table 4 lists the signal connections to the pins of the I/O-expansion connectors. All signal lines in the I/O channel are compatible with LS-TTL (low-power Schottky-diode-clamped transistor-transistor logic) signals. Brief descriptions of each group of lines follow.

Oscillator Clock (BSYSCLK1): This is a buffered version of the main system timing clock. It runs at a frequency of either 14.31818 MHz or 15.0 MHz, depending on which crystal is installed. It has a 50 percent duty cycle.

System Clock (BSYSCLK0): This is a buffered version of the system processor clock. It runs at a frequency that is one-third that of SYSCLK1. It has a 33 percent duty cycle (high for one-third of the cycle, low for two-thirds).

System Reset (BRESET): This is a buffered version of SYSRES, which is active on power-up. It is synchronized to the falling edge of the SYSCLK0 waveform and is used for initialization of all system hardware components.

Address Latch Enable (SYSALE): This signal is used to indicate the presence of a valid address on the system bus. The falling edge of SYSALE is normally used to latch the address. This signal is generated by the 8288 bus controller during bus cycles initiated by one of the local-bus masters. The system-address enable signal, SYSAEN, should be used to enable this signal in the I/O channel.

System Address Enable (SYSAEN): This line, when active-low, indicates that one of the system coprocessors (either the 8088 or the 8087) has control of the system bus. When SYSAEN is high, the 8237A-5 DMA controller has control of the system bus and drives the system address, system memory, and I/O-read/write lines.

I/O Channel Ready (IOCHNLRDY/WAIT): This line is normally high. When a slow I/O device or expansion memory board decodes a valid address, this line should be driven low, causing the flip-flops IC25 and IC33 to insert wait states into the bus cycle until the slow device has completed its cycle. (To avoid conflict with memory refresh, this line should never be held low for more than 1 or 2 μs.)

System Memory Read (SYSMEMRD): This control line is used to gate the memory-device data buffers onto the system data bus during memory-read cycles initiated by either the processor or DMA controller.

System Board Peripheral Device	Base Address (hexadecimal)
8237A-5 DMA controller	000
8272 floppy-disk controller	020
DMA page registers 0 and 1	040
DMA page register 2	060
DMA page register 3	080
floppy-disk-drive motor-on register	0A0
parity-error flip-flop clear	0C0
spare (reserved)	0E0
spare (reserved)	100
8259A interrupt controller—slave	120
8259A interrupt controller—master	140
console serial I/O port	160
auxiliary serial I/O port	180
8255A-5 parallel I/O	1A0
8155H-2 parallel I/O and timer	1C0
8253-5 counter-timers	1E0

Table 5: *Base addresses of the I/O-device-control registers.*

System Memory Write (SYSMEMWR): This control is used to store the data present on the system data bus into the selected memory location during memory-write cycles initiated by either the processor or DMA controller.

System I/O Read (SYSIORD): This control line is used to gate the selected I/O device to accept the data present on the system data bus during I/O-read cycles initiated by either the processor or the DMA controller.

System I/O Write (SYSIOWR): This control line tells the selected I/O device to accept the data present on the system data bus. It is active in I/O-write cycles initiated by either the processor or DMA controller.

I/O-Channel (Parity) Error (I/OCHNLERR): This signal, when enabled by the system software, will cause an interrupt via the NMI input of the 8088 processor. It is normally used to alert the processor to a parity error in memory devices residing in the I/O channel.

System Address Bus (SYSA0 through SYSA19): These lines form a 20-bit system address bus, which can address up to 1 megabyte of memory. SYSA0 represents the least significant address bit (LSB), and SYSA19 represents the most significant address bit (MSB). These lines can be driven either from the processor or from the DMA controller and are considered to be active-high.

The MPX-16 computer system can potentially contain 1 megabyte of memory and still have spare expansion slots.

System Data Bus (SYSDAT0 through SYSDAT7): These lines form the 8-bit system data bus and can be driven by the processor, memory devices, or I/O devices. They are bi-directional and are considered to be active-high. SYSDAT0 is the LSB, SYSDAT7 the MSB.

I/O Channel Interrupt Requests (IRQST2 through IRQST7): These lines are prioritized interrupt-request lines, with IRQST2 having the highest priority and IRQST7 the lowest priority. The lines are edge-triggered and active-high; however, the request signal must be maintained in the high state until the interrupt request has been acknowledged. The interrupt-service routine written for each particular device in use must usually do this.

DMA Requests (DMARQST1 through DMARQST3): These lines are prioritized DMA-request lines, with DMARQST1 having the highest priority and DMARQST3 the lowest priority. The lines are active-high and must be held high until the corresponding $\overline{\text{DMACK}x}$ line goes active-low. DMARQST2 is used by the system-board floppy-disk controller and is included in the I/O channel only for compatibility with the IBM Personal Computer. These lines are typically used by peripheral devices such as disk controllers to request DMA service.

DMA Acknowledge Lines ($\overline{\text{DMACK1}}$ through $\overline{\text{DMACK3}}$): These lines are used to acknowledge DMA requests generated by the DMARQSTx lines.

DMA Acknowledge 0 ($\overline{\text{BDMACK0}}$): This is a buffered $\overline{\text{DMACK0}}$ line and signifies that a DMA-controlled dynamic-memory-refresh cycle is in progress.

DMA Terminal Count (TCNT): This signal is active-high when any of the four DMA channels reaches a terminal count. The corresponding DMA-acknowledge line should be used in conjunction with the TCNT signal.

Peripheral Power: +5 volts (V) DC ±5%, logic ground, +12 V DC ±5%, −12 V DC ±10%, and −5 V DC ±10% power connections are all provided in each expansion connector.

I/O-Decoder Logic

The MPX-16 computer system contains a variety of onboard, high-performance peripheral devices: direct support for all of the major I/O functions needed to form a complete microcomputer system, as listed in table 1 on page 44.

All of the system-board I/O peripherals are addressed or selected by the 4-to-16 decoder IC52 (shown in

section 2, figure 2 on page 48). This decoding logic maintains addressing compatibility with IBM Personal Computer peripherals by using the system-address-bus line SYSA9 to determine whether the peripheral device being selected is on the main circuit board or off it. A low state on the SYSA9 line enables one of the strobe inputs of the decoder; the other strobed input is enabled if one of the local bus masters has control of the system bus, indicated by a low state on $\overline{88AEN}$. When an I/O-device interface chip is selected by this decoded address and either the $\overline{SYSIORD}$ or $\overline{SYSIOWR}$ line is active, an I/O bus cycle is performed. During DMA cycles the I/O decoder is disabled.

The base address for each of the system-board I/O devices is shown in table 5 on page 76. The total number of address-space locations used by each peripheral device varies; this will be discussed in more detail next month in part 3.

Next Month:

If you've followed everything in this second installment on the Circuit Cellar MPX-16 computer system, you're doing well. In the January article I'll fill you in on the serial and parallel I/O ports, counters, floppy-disk controller, and operating-system BIOS, among other topics.■

Acknowledgments

Thanks to Jim Norris, George Martin, and Linda Spencer of Owl Electronic Laboratories for their contributions to this project.

Thanks to Mark Dahmke and Gordon Heins for their help with the documentation.

Thanks to Bill Morello and his staff at Techart Associates for their careful work in drawing the schematic diagrams.

Editor's Note: *Steve often refers to previous Circuit Cellar articles as reference material for each month's current article. Most of these past articles are available in reprint books from BYTE Books, McGraw-Hill Book Company, POB 400, Hightstown, NJ 08520.*

Ciarcia's Circuit Cellar, Volume I, *covers articles that appeared in BYTE from September 1977 through November 1978.* Ciarcia's Circuit Cellar, Volume II, *contains articles from December 1978 through June 1980.* Ciarcia's Circuit Cellar, Volume III, *contains the articles that were published from July 1980 through December 1981.*

To receive a complete list of Ciarcia's Circuit Cellar project kits available from the Micromint, circle 100 on the reader service inquiry card at the back of the magazine.

References

1. Cantrell, Thomas. "An 8088 Processor fo the S-100 Bus." Part 1, September 198(BYTE, page 46. Part 2, October 198(BYTE, page 62. Part 3, November 198(BYTE, page 340.
2. Ciarcia, Steve. "Build the Circuit Cella MPX-16 Computer System, Part 1." November 1982 BYTE, page 78.
3. Ciarcia, Steve. "Ease into 16-Bit Com puting." Part 1, March 1980 BYTE, pag 17. Part 2, April 1980 BYTE, page 40. Re printed in *Ciarcia's Circuit Cellar, Volum II.* Peterborough, NH: BYTE Books, 198 page 171.
4. Ciarcia, Steve. "The Intel 8086. November 1979 BYTE, page 14. Re printed in *Ciarcia's Circuit Cellar, Volum II.* Peterborough, NH: BYTE Books, 198 page 120.
5. *Component Data Catalog.* Santa Clar CA: Intel Corporation, 1981.
6. *iAPX Book.* Santa Clara, CA: Intel Corpor tion, 1981.
7. *iAPX 86,88 User's Manual.* Santa Clar CA: Intel Corporation, 1981.
8. Morgan, Christopher L. and Mitche Waite. *8086/8088 16-bit Microprocesso Primer.* Peterborough, NH: BYTE Book 1982.

The following items are available from:

The Micromint Inc.
561 Willow Ave.
Cedarhurst, NY 11516
(516) 374-6793
(for technical information)
(800) 645-3479
(for orders only)

1. MPX-16 single-board computer system: assembled, tested, and burned-in. Includes 64K bytes of RAM, Digital Research CP/M-86 operating system on 8-inch or 5¼-inch floppy disk, CP/M-86 BIOS in EPROM, MPX-16 Technical Reference and User's Manual. *Requires power supply and floppy-disk drive.*
Single-quantity price.....$1895

2. MPX-16 single-board computer system, as above, but with 256K bytes of RAM installed.
Single-quantity price.....$2135

3. MPX-16 single-board computer system, with 64K bytes of RAM.
In OEM quantities of 100 $1200 each

4. Complete MPX-16 disk-based system: includes MPX-16 single-board computer, assembled, tested, and burned-in, with 256K bytes of RAM installed, CP/M-86 operating system on 5¼-inch floppy disk, CP/M-86 BIOS in EPROM, power supply, one 5¼-inch single-sided floppy-disk drive, connecting cables, MPX-16 Technical Reference and User's Manual. *Enclosure sold separately.*
Single-quantity price.....$2895

5. Unpopulated (blank) printed-circuit board for the MPX-16 computer system: five-layer, screened, and solder-masked. Includes CP/M-86 BIOS in EPROM, MPX-16 Technical Reference and User's Manual.
Single-quantity price.....$300

6. Digital Research CP/M-86 User's Manual *(three-volume set), sold separately.....$40*

7. MPX-16 Technical Reference and User's Manual, *sold separately....$50*

8. Enclosure for MPX-16 circuit board.....call for price

* * *

When it becomes available for the MPX-16, Microsoft's MS-DOS operating system may be optionally substituted for CP/M-86.

The MPX-16 is available to OEMs in large quantities either as a circuit board or as a complete system with floppy-disk drives and enclosure. Call the Micromint for prices and delivery information.

For orders within the continental United States, please include $10 for shipping; overseas orders please include $30. Residents of New York please include 7 percent sales tax.

Build the Circuit Cellar MPX-16 Computer System
Part 3

The final installment describing the design of the MPX-16, which is I/O-compatible with the IBM Personal Computer.

Steve Ciarcia
POB 582
Glastonbury, CT 06033

This month's article is the last of three on the construction of the Circuit Cellar MPX-16 computer, which is built around the Intel 8088 microprocessor. In part 1, I presented an overview of the system and a discussion of the coprocessors and bus structures. Last month, in part 2, I described the memory, interrupt mechanism, expansion bus, and I/O- (input/output) decoding sections. This month I'd like to finish by describing the serial and parallel I/O, counters and timers, the floppy-disk interface, and an overview of certain parts of the CP/M-86 operating system.

Because the MPX-16 is somewhat more complex than the typical Circuit Cellar project, I've had to simplify or abbreviate my treatment of many details to fit the articles into only three issues of BYTE; to learn some nuances of the individual system parts, you should consult the references I have listed on page 82. (More detailed information on the MPX-16, including timing diagrams and listings, is available in the *MPX-16 Technical Reference and User's Manual*, available from The Micromint.) But these articles contain enough information for you to understand the basic functions of all the subsystems and how they work together. And most of what you can learn applies also to the IBM Personal Computer and other similar machines. We'll continue the presentation after we review the major features of the MPX-16.

Most of what you can learn about the MPX-16 applies also to the IBM Personal Computer.

IBM and IBM Personal Computer are trademarks of International Business Machines Corporation.

CP/M-86, CP/M-80, and MP/M are trademarks of Digital Research Inc.

MPX-16 Features

The Circuit Cellar MPX-16 computer system, shown in photo 1 on page 56, fundamentally consists of a single 9- by 12-inch five-layer printed-circuit board (containing 120 integrated circuits), to which various peripheral devices are attached. Its I/O-expansion bus is completely compatible with that of the IBM Personal Computer but has nine expansion positions instead of five.

The MPX-16 uses the Intel 8088 microprocessor and the optional Intel 8087 numeric coprocessor; the main circuit board has room for 256K bytes of user memory and contains two serial and three parallel I/O ports, a floppy-disk controller, and EPROMs (erasable programmable read-only memories) containing the BIOS (basic input/output system) module of Digital Research's CP/M-86 16-bit disk operating system. The MPX-16 can be expanded by plugging in various circuit boards and interfaces

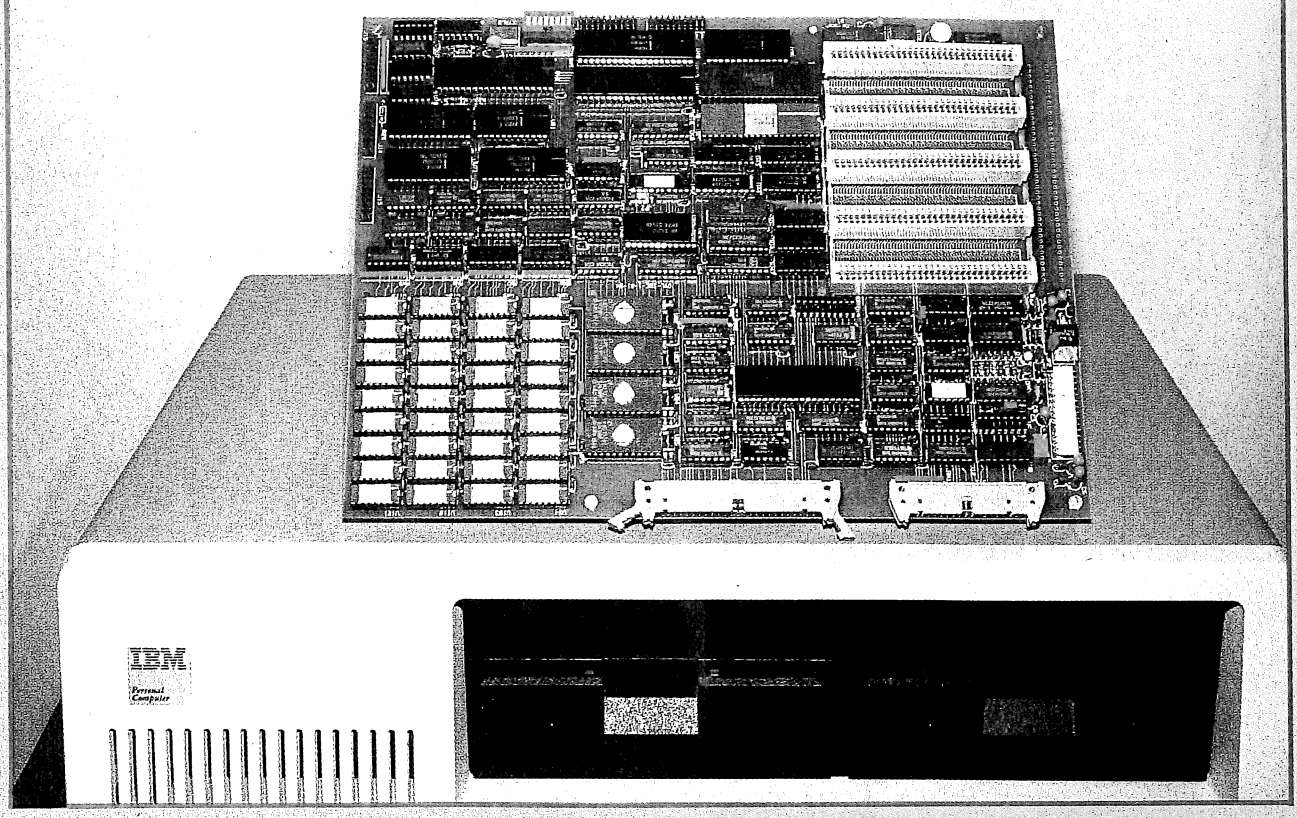


Photo 1: *The MPX-16 has been designed to be compatible with the IBM Personal Computer in that peripheral devices made for use with the IBM PC can be plugged into the I/O-expansion bus of the MPX-16.*

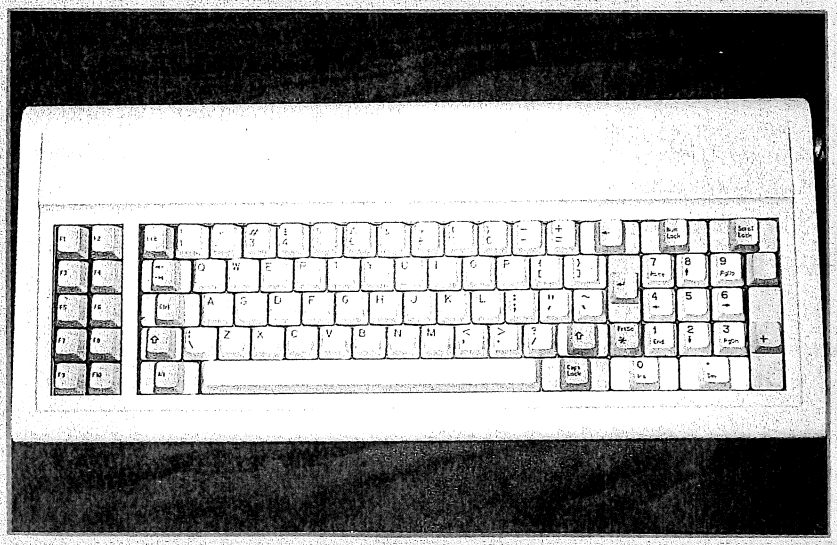


Photo 2: *This keyboard, made by Key Tronic Corporation (Building 14, Spokane Industrial Park, Spokane, WA 99214), is nearly an exact copy of the keyboard of the IBM Personal Computer.*

to provide a full megabyte of user memory and additional external mass storage. A more detailed list of characteristics appears in table 1 on page 59.

The MPX-16 was initially designed to run CP/M-86, but eventually Microsoft's MS-DOS operating system will be available for it, making it possible to run most software written for the IBM Personal Computer on the MPX-16, except software that uses unique features of the IBM machine. The principal difference is this: with the present operating-system BIOS, the MPX-16 communicates with the user through a serially interfaced display terminal instead of through a memory-mapped video display. In theory, you could plug an IBM Display Adapter into one of the expansion slots and connect a serial keyboard (such as the Key Tronic model shown in photo 2) for exact

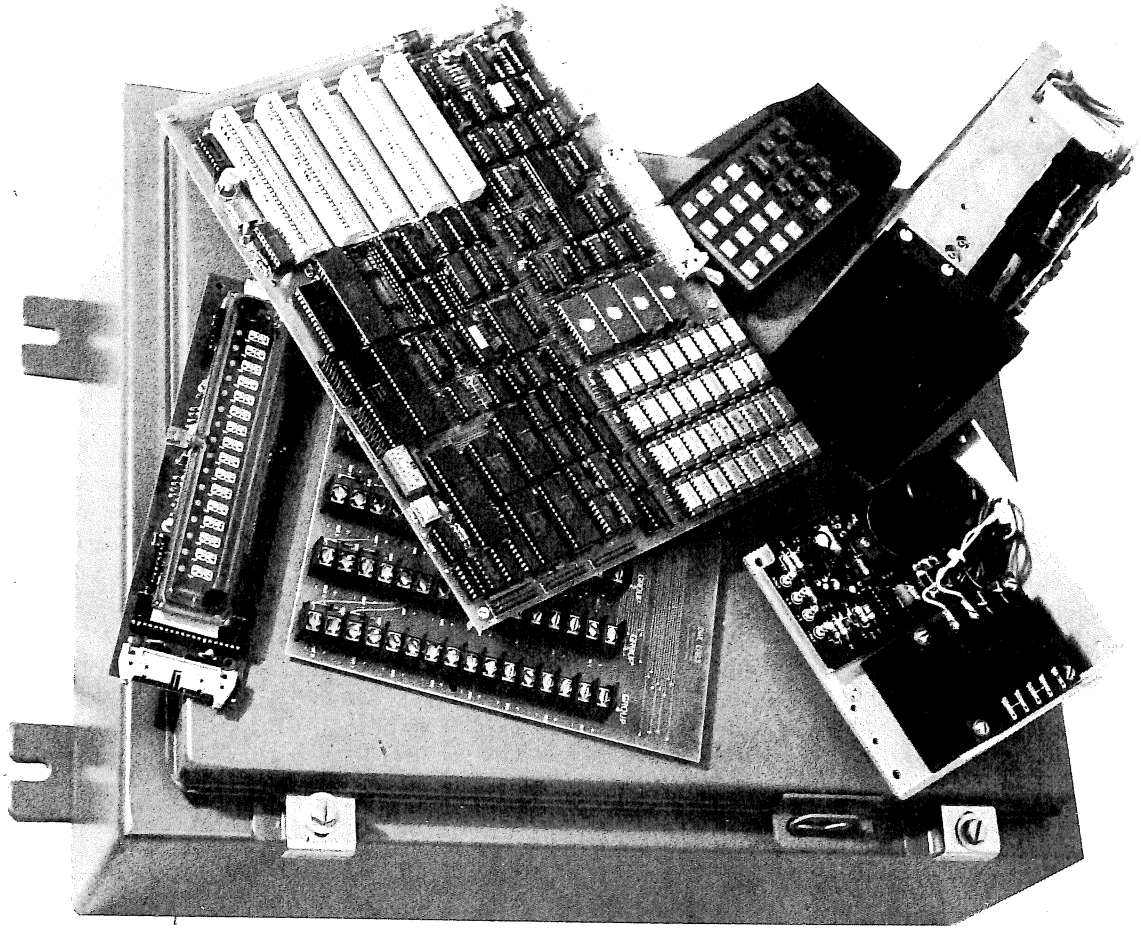


Photo 3: *Blasts and flying fluids won't faze an MPX-16 computer protected by a Hoffman heavy-duty NEMA 12 enclosure. (Photo courtesy of Owl Electronic Laboratories Inc.)*

hardware emulation.

The MPX-16 is well suited for use as a low-cost 8088-based computer for integration into a complete hardware/software package chiefly because it combines so many functions on a single printed-circuit board. Putting together the hardware of a complete system, you need only add a power supply, a serial video-display or printing terminal, and one floppy-disk drive (either 5¼- or 8-inch). By the time you read this, an enclosure for the circuit board should be available. Many applications need nothing more.

Photo 3 shows the MPX-16 along with all the other components needed to create an industrial control system, including a NEMA 12 (a National Electrical Manufacturers Association specification) enclosure, which should protect it from any environment you'd want to operate it in.

Parallel I/O Interface

The MPX-16 System Board supports four independent parallel I/O ports; of these, two are dedicated to single purposes and two are available as general-purpose I/O ports. The two dedicated ports use the Intel 8255A-5 programmable peripheral interface (PPI), which appears as IC60 in section 4 of the schematic diagram, figure 1 on pages 60 and 61. The other two ports are implemented using the Intel 8155H-2 chip, IC47 in figure 1, which contains two I/O ports, a 14-bit counter/timer circuit, and 256 bytes of read/write memory. (This memory is not used in the MPX-16. I've written about the 8155 before; see reference 3.) The relationship of the parallel I/O subsystems with the global system bus structures can be seen in the system block diagram (see figure 2 in part 1, November 1982 BYTE, pages 84 through 86). Most notably, the 8155 communicates over the local address/data bus shared with the processors, while the 8255 receives its data through the buffered resident data bus.

One of the dedicated ports is used during system initialization to read the settings of DIP (dual-inline pin) switches SW1 through SW8, which form an 8-bit system-configuration value. The eight lines of the configuration switches drive the port-A lines of the 8255. These lines are initialized by the power-up software initialization routine as input lines in the 8255's operating-mode 0 (basic input/output). The operating system can read the switch settings via an input instruction from I/O address hexadecimal 1A0. Data bits 0 to 7 in the value obtained contain the respective settings of SW1 to SW8.

The second dedicated parallel port in the 8255 is normally set up as a

1. designed to use a 5-MHz Intel 8088 microprocessor, which combines a 16-bit architecture with an 8-bit bus interface and has 20-bit addressing capability for up to 1 megabyte of system memory, operating in maximum mode to support multiprocessing
2. optional Intel 8087 math coprocessor
3. onboard space for four 64K-byte banks of dynamic RAM for a total of up to 256K bytes, with parity generation and error detection.
4. sockets for up to 64K bytes of JEDEC 24- or 28-pin standard ROM or EPROM devices
5. two RS-232C serial interface ports
6. two 8-bit general-purpose parallel I/O ports with handshaking control lines
7. one Centronics-compatible parallel printer port
8. four programmable timers (one for a real-time clock, two for data rates, one for memory-refresh requests)
9. four independent DMA (direct memory access) channels
10. sixteen levels of vectored, prioritized interrupt control
11. single- or double-density floppy-disk controller for controlling up to four 5¼-inch or 8-inch drives
12. five 62-pin I/O-expansion-channel connectors (hardware compatible with the IBM Personal Computer) with space for four more
13. five-layer 9- by 12-inch printed-circuit board
14. BIOS for CP/M-86 in EPROM

Table 1: *Features of the MPX-16 computer system.*

Centronics-compatible printer port. This second port can also be used as a general-purpose 15-bit parallel interface with 10 output lines and 5 input lines. Fourteen of the I/O lines are connected to the port-B and port-C lines of the 8255. All 15 lines are buffered and connected to the 20-pin Bergstik connector J15. The 10 output lines from port B and bits 6 and 7 of port C drive sections of the open-collector buffers IC77 and IC78. The 5 input lines are buffered by IC77 and IC76, with pull-up resistors on the input lines to allow for use of open-collector drivers on the other end. Signal-return paths are provided on pins 14 through 18 of J15.

The two nondedicated parallel ports, which communicate to the outside world through the two 20-pin Bergstik connectors J16 and J17, are implemented with the 8155H-2, IC47. These two identical I/O ports, each with 11 I/O lines (three of which are used for handshaking control), are initialized by the software initialization routine as one 8-bit output port (J16) and one 8-bit input port (J17). Because these ports are meant to be used for varying purposes, the application software of the user will typically reinitialize the 8155 to suit the application. This is accomplished by writing a new control word into the 8155's command/status register located at I/O address hexadecimal 1C0.

Serial Interface

The MPX-16 system board contains two independent RS-232C asynchronous serial I/O ports (also known as serial channels). These are primarily intended to be used in connecting the system to video-display terminals, but they may be attached to any compatible RS-232C devices. One of the serial channels (CH0) has been defined as the console I/O port for the CP/M-86 operating-system software. The second serial port (CH1) is available for user-defined applications.

The two RS-232C serial ports are implemented with Intel 8251A USARTs (universal synchronous/asynchronous receiver/transmitters), as shown in figure 1. An 8251A is capable of transmitting and receiving simultaneously at different data rates; however, the MPX-16 system requires that the same rate be used for both transmitting and receiving. A split-speed application may be supported by using both serial ports, programmed to operate at different rates.

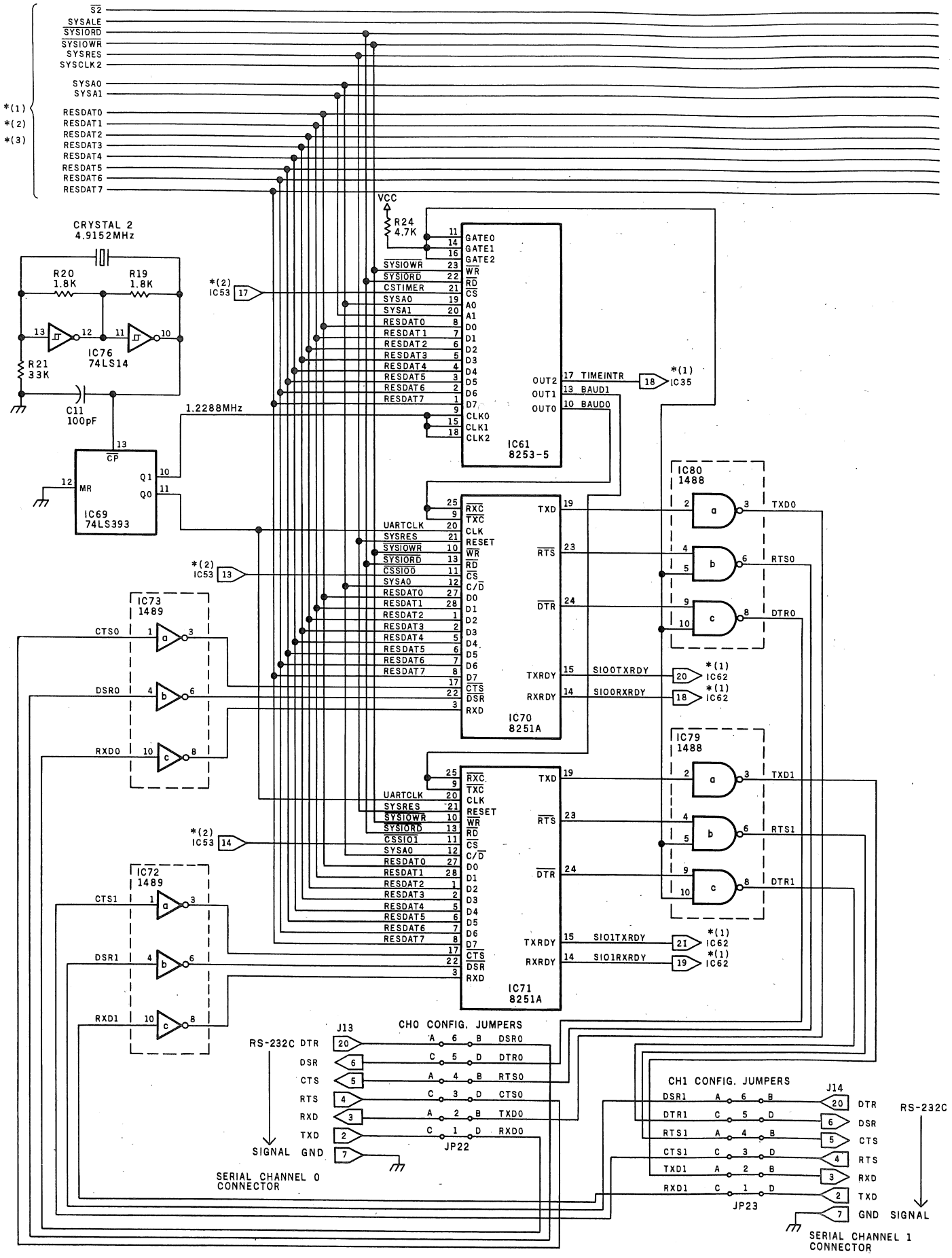
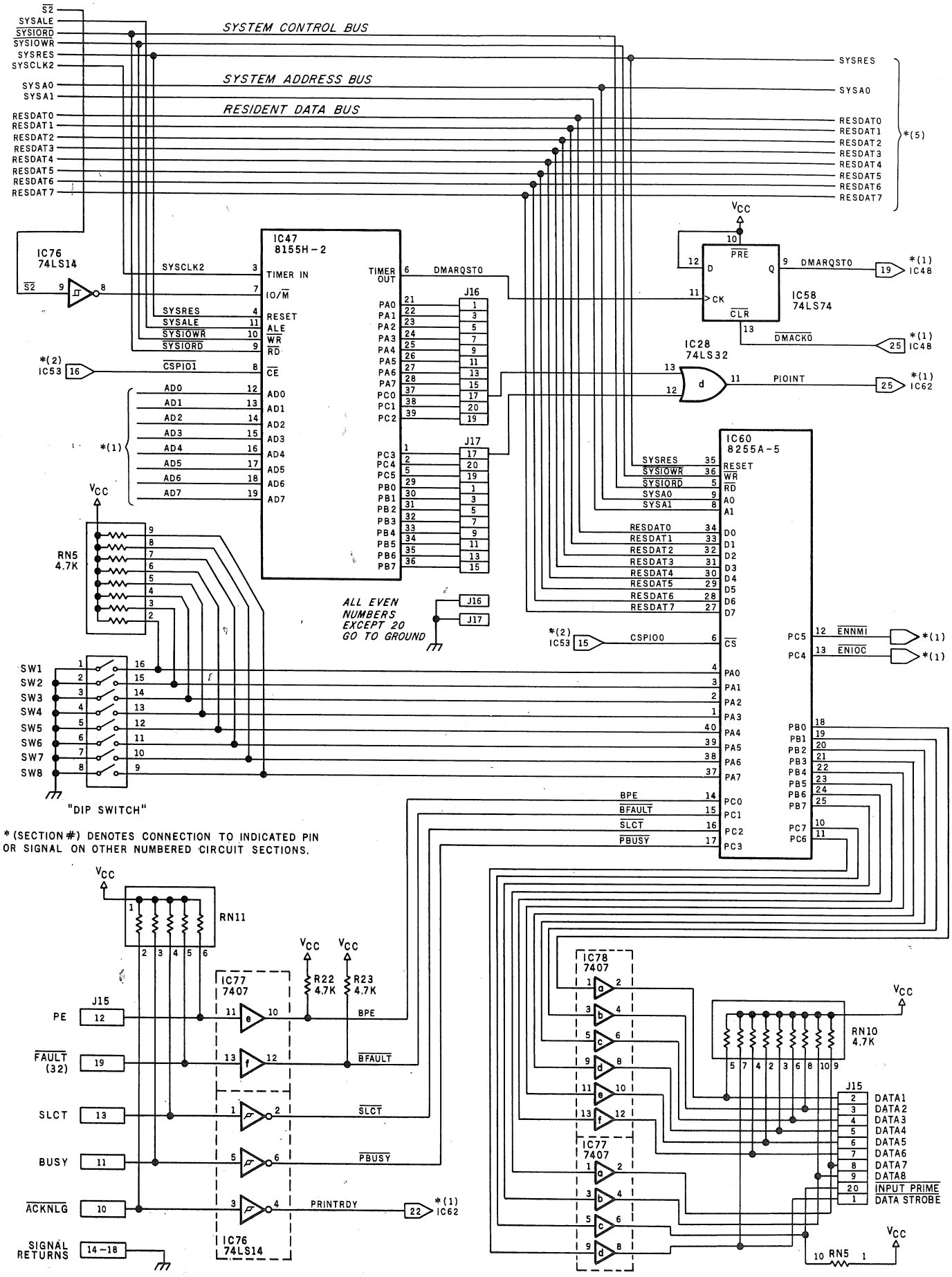


Figure 1: *Section 4 of the schematic diagram of the MPX-16 computer. Section 1 appeared in November's article; sections 2 and 3 appeared in December's article. Connections to other sections of the schematic are shown by the notation *(n), where n is the number of the other section.*



Here are shown the interface circuits for the serial and parallel I/O ports: the 8251A USARTs and the 8255A-5 and 8155H-2 parallel-interface components.

Both transmitter-ready and receiver-ready interrupt-request signals are generated during communication sequences. These signals are fed into interrupt-request lines IR0, IR1, IR2, and IR3 of the slave 8259A programmable interrupt controller, IC62 (which appeared in section 1 of the schematic diagram in November's article). The channel-0 interrupts have priority over the channel-1 interrupts, and the receiver-ready interrupt requests have priority over the transmitter-ready requests.

Both types of request signals are active-high. The receiver-ready interrupt request, which signals the main processor that a character has been received and converted to a parallel format, is obtained from the 8251A USART's RXRDY output line. Similarly, the transmitter-ready interrupt request, which signals the processor that the 8251A is ready to transmit another character to a peripheral device, is taken from the TXRDY output line of the 8251A. (Each USART also provides four control lines that can be used for modem control.)

Counter/Timers

Four independent counter/timers are found on the MPX-16 system board. All four are used for dedicated system functions and generally should not be used for other purposes. Three of these counter/timer circuits are part of the Intel 8253-5 programmable interval timer (PIT), IC61. The fourth one is the timer section of the 8155H-2, IC47, which was discussed above. All of the counter/timers are visible in section 4 of the schematic diagram, figure 1.

The 8253-5 PIT contains three independently programmable 16-bit counter/timer circuits capable of clock rates of up to 2 MHz (megahertz). These counters can be operated in any of six different modes: terminal-count-interrupt generator, programmable one-shot, rate generator, square-wave generator, software-triggered strobe, and hardware-triggered strobe.

On the MPX-16 system board, all three counter/timers of the 8253 PIT are programmed by the power-up-initialization software routine to

operate in mode 3 (square-wave generator). The input clock signal that drives all three of the 8253's counter-clock-input lines is obtained from a simple crystal-controlled oscillator circuit consisting of a 4.9152-MHz crystal, a couple of inverter gates, a few resistors, and a capacitor. The output of this circuit, a 4.9152-MHz square wave, is then divided down by a 74LS393 binary counter to form a 2.4576-MHz USART clock and a 1.2288-MHz clock to drive the 8253 PIT counters.

The first counter circuit of the 8253 PIT is used as a software-programmable data-rate generator, producing a signal called BAUD0. Similarly, the second counter circuit is used to produce the data-rate signal BAUD1. The data rate for both serial channels is set at power-up for 9600 bps (bits per second) using a data-rate multiplier factor of 16. The system software then automatically initializes the data rate for the console serial channel (channel 0) when the user types a Return character in ASCII (American Standard Code for Information Interchange). The first character must be Return for proper data-rate initialization. If the input data rate of the console terminal is not 9600 bps, the program reinitializes the counter-1 circuit of the 8253 to match the new data rate.

So that system crashes will not occur, the memory-refresh signal must never be altered by application software.

The third counter/timer circuit of the 8253 PIT is intended for use as a real-time clock for either time-of-day or software-timing-delay applications. This clock is initialized at power-up by software, preset for a 10-ms (millisecond) period (100 Hz). This clock output drives the IR0 line of the master 8259A interrupt controller, IC35, and forms the highest-priority maskable system interrupt. This timekeeping capability can be very useful in interrupt-driven, real-time process-control applications.

The fourth counter/timer on the MPX-16 system board is the timer section of the 8155H-2, IC47. This timer is driven by the SYSCLK2 (2.386-MHz) clock signal to produce the square-wave signal REFRQST, which has a period of 15.1 μs (microseconds). The REFRQST output signal activates the periodic refresh operation required by the dynamic RAMs (random-access read/write memories). This vital signal must never be altered by the user's application software; if it is, system crashes may occur.

Floppy-Disk Drive Controller

The MPX-16 system supports up to four floppy-disk drives. Versatility is provided by jumper-selectable features of the MPX-16's floppy-disk controller interface: either 5¼-inch or 8-inch drives may be used and up to four drives may be attached to the

A. Power Lines

All power to the disk drives is supplied from an external power supply through separate power cables. A typical 5¼-inch floppy-disk drive will require approximately +5 V (volts) DC at 0.5 A (amps) and +12 V DC at 1 A. A typical 8-inch drive will require +24 V DC at 1.3 A, +5 V DC at 0.8 A, −5 V DC at 0.05 A and 115 V AC at 0.3 A.

B. Output Lines

DRIVESEL *x*: The four drive-selection lines, numbered 0 through 3, are provided to enable the selected drive to respond to input signals and consequently to output data and/or status information. Each individual drive must be configured to respond to one of the four drive-select signals. This is usually accomplished via a programmable shunt header or a DIP switch. A drive is selected by a logic low state on the select line assigned to it.

DIRECTION: This control line defines the direction of motion of the selected drive's read/write head during a step operation. A high state (equivalent of logic 1) will cause the head to move out, toward the outer edge of the disk. A low state (logic 0) will cause the read/write head to move in, toward the center of the disk.

STEP: This control line causes the selected drive to move its read/write-head carriage one position in the direction controlled by the direction-select line. Each step is initiated by the low-to-high transition of the STEP pulse. Direction changes must occur at least 1 μs before the trailing edge of the step pulse.

WR ENABLE: The write-enable, or write-gate, signal enables the writing of data onto the disk when it is active-low. When this line is inactive-high, the read-data logic and head-step logic circuits are enabled.

HEADLOAD *x*: The four head-load lines, numbered 0 through 3, are alternative output lines which usually require the user to install or configure the drive unit to accept them. The head-load line can be used to load and unload the read/write head from the disk's surface. If desired, the heads may be kept loaded to avoid the 50-ms head-load time. Typically a drive will be configured so that the read/write head loads when either the drive-select line or the motor-on control line becomes active.

MOTOR ON *x*: Three output lines, numbered 0, 1, and 2, are provided for motor-on/motor-off control. The MOTOR ON 0 line on pin 16 of J11 and J12 is the standard floppy-disk interface signal. The MOTOR ON 1 and MOTOR ON 2 lines are available as alternative output control lines. When the MOTOR ON line of the floppy-disk drive (if available) is driven active-low, the drive motor will be turned on, allowing reading or writing on the drive. Typically, a 1-second delay is required after activating the motor control line prior to reading or writing. To maximize motor life, the motor for the drive is usually turned off after 2 seconds if no commands have been issued to the drive.

SIDESELECT: This output control line is used to select which side of a two-sided floppy disk is to be used for reading or writing. This line is provided for future system expansion; it is not supported by the current MPX-16 system software. A logic high on this line designates the read/write head on side 0, and a logic low indicates selection of the side-1 read/write head. A typical delay of 100 μs is required before reading or writing after switching sides.

LOW CURRENT: This output control line is an active-low signal used only by 8-inch drives. It causes a reduced current flow through the read/write head when writing data on tracks 43 to 76. When tracks 0 through 42 are selected, the low-current signal is high, causing a greater current flow.

FAULT RESET: This is an active-low output signal which can be used to reset a disk drive's fault logic, if the drive has some.

WR DATA: The write-data output line contains the serial data information to be written onto the disk. This signal is enabled by the WR ENABLE control line. Each positive transition on the WR DATA line causes the current through the read/write head to be reversed, thus writing a data bit onto the disk.

C. Input Lines

READY: The active-low READY input line can be used to indicate the status of the disk drives when the circuitry in the drive supports such a function. This signal typically indicates that the drive motor is rotating at the correct speed and that two index holes have been detected after a disk has been inserted into the drive. If drive-ready indication is not supported by the drive being used, the jumper to ground must be installed. The READY signal is conditioned by a 150-ohm pull-up resistor and a Schmitt-trigger inverter.

INDEX: The INDEX interface line is an active-low signal that occurs once for each revolution of the disk. This signal indicates the logical beginning of a track. It is conditioned by a 150-ohm resistor and a Schmitt-trigger inverter.

TRACK0: This input line is active-low when the drive's read/write head is positioned over track 0 of the disk (the outermost track) and the access logic circuitry is driving current through phase 1 of the stepper motor's windings. This signal is at a logic 1 at all other times. The TRACK0 signal is conditioned by a 150-ohm pull-up resistor and a Schmitt-trigger inverting buffer.

TWOSIDED: The active-low TWOSIDED input signal, for 8-inch drives, indicates that a double-sided disk is contained in the drive when low, and a single-sided disk is in the drive when high. This signal is terminated by a 150-ohm pull-up resistor and a Schmitt-trigger inverting buffer. This signal is not supported by the current system software but is available for future use as two-sided drives become more widely used.

WRITE PROTECT: This active-low input signal indicates that the disk inserted on the selected drive has been write-protected, and thus no write operations can be performed. On 8-inch drives, the write-protect notch is left uncovered to write-protect the disk; conversely for 5¼-inch drives, the write-protect notch on the disk must be covered to write-protect the disk. This input line is terminated by a 150-ohm pull-up resistor and a Schmitt-trigger inverting buffer.

FAULT: When available, on 8-inch drives, this input line indicates that a fault condition has been detected by the drive-control logic and that further operations on the drive should not be permitted. Thus active-low input is terminated by a 150-ohm pull-up resistor and a Schmitt-trigger inverting buffer.

RD DATA: The read-data input signal contains serial data and clock-bit information read from the disk when the WR ENABLE control line is high (inactive). This line provides an active-low pulse of approximately 200 ns for each flux reversal detected by the drive electronics, whether a data bit or a clock bit. This raw data signal is conditioned by a 150-ohm pull-up resistor and a Schmitt-trigger inverter.

Table 2: *Descriptions of the floppy-disk-drive interface signals found in the MPX-16 system. Both 8-inch and 5¼-inch drives are supported by the floppy-disk controller.*

system. Three drive-motor-control lines and four head-load-control lines are available; both 34-pin and 50-pin connectors, with industry-standard signal/pin assignments, are provided for 5¼-inch and 8-inch drives, respectively. A description of the functions of each interface signal is given in table 2 on page 66.

Either single- or double-density recording may be selected under software control. The normal disk format is compatible with the IBM 3740 format (in the 8-inch size) or with the IBM Personal Computer (in the 5¼-inch size—what might be called the IBM 5150 format), but this can be changed via a software modification. Single-density recording uses the FM (frequency modulation) technique, while double-density operation uses the MFM (modified frequency modulation) technique. (See reference 7 for an explanation of FM and MFM as applied to floppy disks.)

The heart of the floppy-disk interface is an Intel 8272 single-chip floppy-disk controller, or FDC (IC21). This device appears in section 5 of the schematic diagram, figure 2 on pages 70 and 71, along with the rest of the floppy-disk interface logic.

The Intel 8272 was designed to be pin- and function-compatible with the NEC (Nippon Electric Company) μPD765 floppy-disk controller. These controllers support 15 software commands, processor-interrupt generation, DMA (direct memory access) data transfers, and generation of several control signals that can be used to reduce the amount of hardware support logic required to employ double-density recording formats. The 8272 FDC, in conjunction with the 8237A DMA controller, IC48, forms an efficient disk-interface subsystem.

There are six basic functional sections in the disk interface: clock-signal-generation logic, motor-on/off logic, drive-control logic, data-write logic, processor-interface logic, and data-recovery logic for reading the disk.

Clock-Signal Generation

The 8272 FDC requires two external clock signals as input: a 4- or 8-MHz square-wave clock and a data-write clock, with a pulse duration of 250 ns (nanoseconds), that is pulsed at one of three frequencies.

The square-wave clock input at pin 19 of the FDC is derived from an 8-MHz crystal oscillator, IC10. If 8-inch drives are to be used, jumper JP16 must be installed and JP17 removed. This routes the 8-MHz clock directly to pin 19. When 5¼-inch drives are to be used, JP27 must be installed and JP16 removed, applying a 4-MHz signal to pin 19, instead.

The repetition rate of the 250-ns data-write clock pulse is 1 MHz, 500 kHz (kilohertz), or 250 kHz, depending on the disk-drive type and disk format. Multiplexer IC3 selects the correct clock frequency for the desired recording density. When the MFM signal coming from the 8272 is in a logic low state, single-density frequencies are selected. When MFM is high, the double-density frequencies are selected.

Text continued on page 72

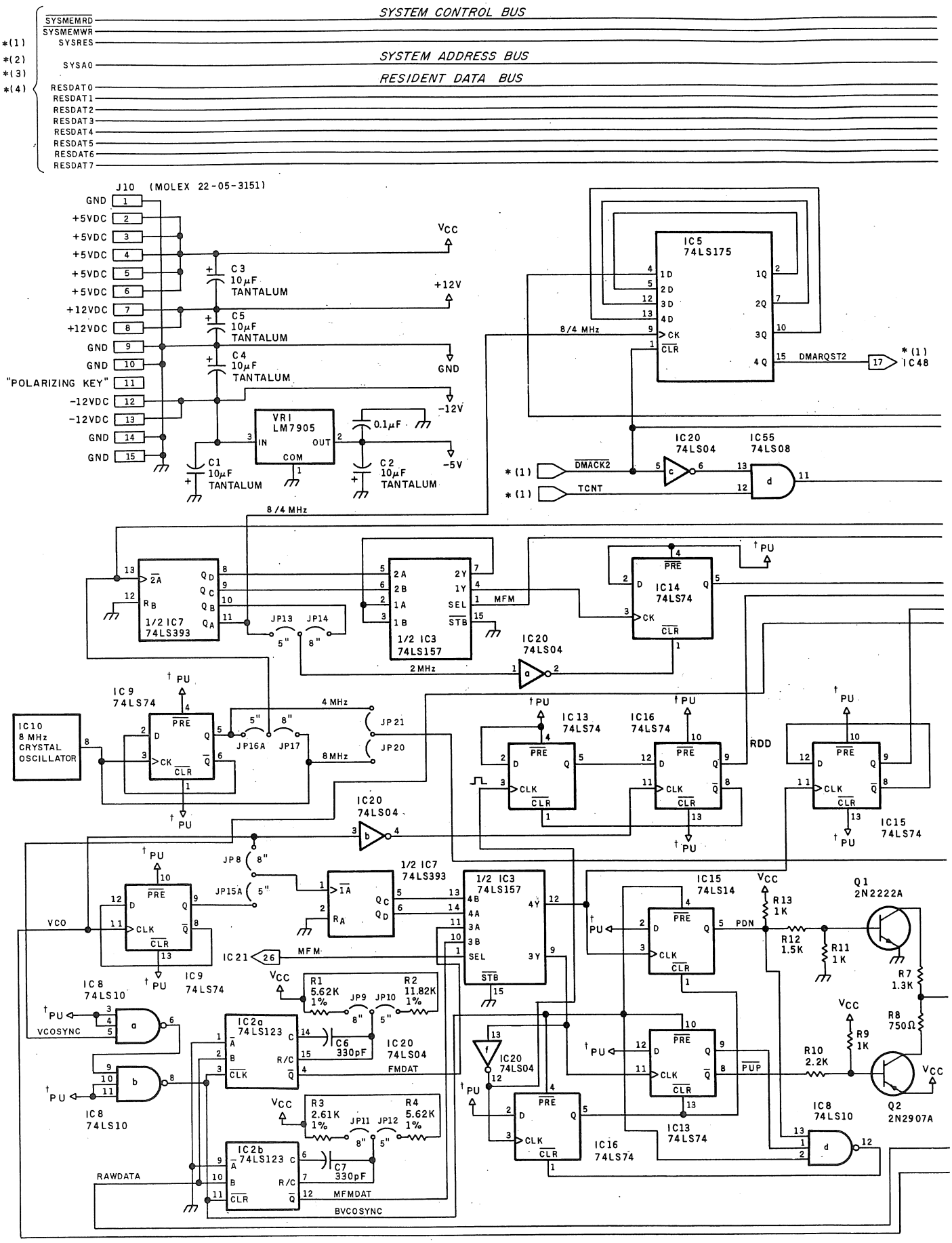
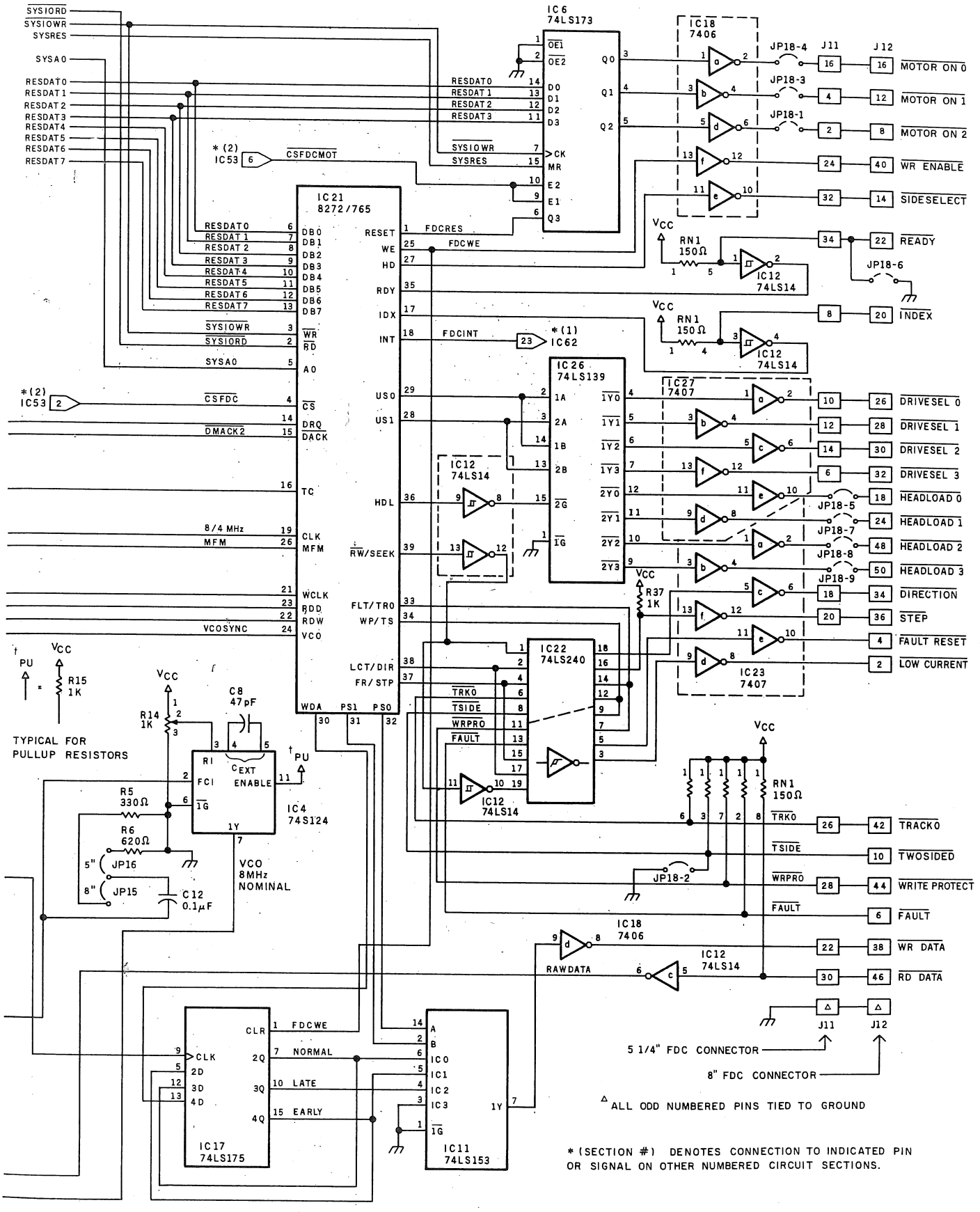


Figure 2: *Section 5 of the MPX-16 schematic diagram. Here are shown the system-board power connections and the floppy-disk controller, including the PLL (phase-locked loop) circuitry used to recover data read from a disk. Connections for both 8-inch and 5¼-inch drives are shown.*



A complete table of the MPX-16's integrated circuits was printed in the part 2 of this series (December 1982 BYTE, pages 56 and 60). The table included a listing of power connections and a cross-reference by schematic section.

Motor Control

The floppy-disk-drive interface provides three separate motor-on/off control lines for the floppy-disk drives: $\overline{\text{MOTOR ON 0}}$, $\overline{\text{MOTOR ON 1}}$, and $\overline{\text{MOTOR ON 2}}$. These signals are generated by a 74LS173 quad D-type register chip, IC6. The 4-flip-flop register is addressed as an I/O device residing on the resident data bus at hexadecimal address 0A0.

The Q0 output of IC6 controls the $\overline{\text{MOTOR ON 0}}$ line. To turn the motor on, a logic 1 is written into Q0, and to turn off the motor a logic 0 is written. The Q1 and Q2 outputs of IC6 similarly control the $\overline{\text{MOTOR ON 1}}$ and $\overline{\text{MOTOR ON 2}}$ lines.

The $\overline{\text{MOTOR ON 0}}$ line is connected to pin 16 on both J11 (the 5¼-inch-drive connector) and J12 (the 8-inch-drive connector). Use of this pin for motor control in floppy-disk interfaces is fairly standard throughout the computer industry. The other two motor-control lines are not standard but are provided to allow additional control, if needed, by wiring the interface cable appropriately. The most common arrangement is for $\overline{\text{MOTOR ON 0}}$ to control drive A, $\overline{\text{MOTOR ON 1}}$ to control drive B, and $\overline{\text{MOTOR ON 2}}$ to control drives C and D. All three control lines have an onboard jumper that can be used to disconnect the signal from the disk-drive connectors.

Drive-Control Logic

The floppy-disk-interface drive-control logic consists of all control signals other than the motor-on/off control signals supplied to or received from the electronic circuitry inside the floppy-disk drives. All of the output signal lines are driven by type-7406 open-collector inverting drivers or type-7407 open-collector noninverting drivers. All input signal lines are conditioned by 150-ohm pull-up resistors and 74LS14 Schmitt-trigger inverter gates. All of the signals, input and output, are active-low.

The $\overline{\text{RW/SEEK}}$ line of the 8272 FDC is used to multiplex eight DC interface signals onto four pins of the 8272. When the FDC is in the seek mode (with $\overline{\text{RW/SEEK}}$ low), pin 19 of the 74LS240 octal inverting buffer IC22 is driven low. This causes the $\overline{\text{TRACK0}}$ and the $\overline{\text{TWOSIDED}}$ signals to be input into pins 33 and 34 of the FDC, and the $\overline{\text{DIRECTION}}$ and $\overline{\text{STEP}}$ signals from pins 38 and 37 to be output to the drives.

When the FDC is in the read/write mode (with $\overline{\text{RW/SEEK}}$ high), pin 1 of the inverting buffer IC22 is driven low. This allows the $\overline{\text{WRITE PROTECT}}$ and $\overline{\text{FAULT}}$ signals to pass into pins 34 and 33 of the FDC and lets the $\overline{\text{FAULT RESET}}$ and $\overline{\text{LOW CURRENT}}$ signals from pins 37 and 38 of the FDC pass to the drive. Note that the four signals that were gated by a low state on the $\overline{\text{RW/SEEK}}$ line are now blocked by the high-impedance state of their buffer sections. A pull-up resistor is provided to ensure that a false $\overline{\text{STEP}}$ command is not issued to the drive units.

The 8272 FDC provides two control signals to select one of four drives, US0 and US1 on pins 29 and 28. These two lines drive the 74LS139 dual 2-to-4-line demultiplexer, IC26, which selects the desired drive by placing a low state on the corresponding $\overline{\text{DRIVESEL}}\,x$ line. The signals from US0 and US1 are tapped off to another section of the demultiplexer to activate the head-load signal at the same time. (The interface may be wired to load all heads together or separately.)

The HD (head-select) output of the 8272, pin 27, is available for applications where two-sided disk drives are available. This signal can be used to select one of the two read/write heads. Initially, the MPX-16 system software supports only single-sided drives and does not use this control signal. A two-sided modification will eventually be incorporated.

Two input pins, the $\overline{\text{READY}}$ and $\overline{\text{INDEX}}$ signals are conditioned by 74LS14 Schmitt-trigger inverters and routed directly to the 8272. The $\overline{\text{READY}}$ line can be jumpered to ground if the attached drives do not provide a status-ready indication. An index pulse occurs once per revolu-

tion of the disk when a soft-sectored floppy disk (the type supported by the MPX-16) is being used in the selected drive.

Data-Write Logic

The data-write logic consists of the 74LS175 quad type-D flip-flop IC17 and the 74LS153 4-to-1 decoder, IC11. The 74LS175 is configured as a shift register clocked by the single/double-density write clock, which provides the precompensation required for double-density recording. The actual value (250 or 125 ns) depends on the particular drive size being used and is selected by jumpers JP20 and JP21.

Data-Recovery Logic

The data-recovery (data-read) logic of the floppy-disk interface, shown on page 70 of figure 2, is fairly complex, due to the subtleties of MFM double-density recording. The MPX-16 uses a PLL (phase-locked-loop) circuit to decode the double-density data. The 8272 floppy-disk controller, IC21, requires two input signals, the RDD and RDW signals at pins 23 and 22, respectively, to be generated from the raw-data signal read from the disk and transmitted to the interface by the drive electronics. The RDD signal consists of one positive pulse for each magnetic-flux reversal read from the disk, which can signify either a clock bit or a data bit. The RDW signal tells the 8272 of the status of the "data window" (a period of time in which a pulse may or may not occur), which is used by the 8272 to determine if the flux reversal is a data bit or a clock bit (see reference 7).

The 8272 provides two output signals, the VCOSYNC and MFM signals, that simplify the implementation of a PLL data-recovery circuit. The VCOSYNC signal goes active-high when valid data is being read from the disk and is used to enable the PLL logic. When a gap area (a place on a floppy disk where no data is recorded—for example, between the disk's identification and data fields) is being read by the read/write head, the VCOSYNC signal goes low to disable the PLL. In addition, the VCOSYNC signal can be high only after the read/write head has been loaded and the head-load time has elapsed. The MFM signal from the 8272, when active-high, indicates that the 8272 has been programmed for double-density operation; when MFM is inactive-low, single-density operation is indicated. This signal, along with the data-recovery logic, allows the recording mode to be software-selected between single- and double-density operation.

The active-high RAWDATA pulses from the disk-drive circuitry trigger two one-shot multivibrator sections, both in IC2, which serve as pulse shapers for the phase-detector logic. Section IC2a shapes the single-density (FM) data pulses, while section IC2b works for double-density (MFM) data. Separate one-shots are provided for the MFM and FM modes so that the recording format can be selected only by software.

The one-shots take the raw data pulses from the drive and stretch or shrink them to a constant length, as required. The duration of the output pulses of the one-shots is determined by resistors R1 through R4 and capacitors C6 and C7. Jumper connections JP9 through JP12 are used to set up the correct pulse duration for 5¼-inch or 8-inch drives. The RC (resistance/capacitance) values are chosen to provide a shaped data pulse width that is one-half the duration of the data window. These values are 2 μs for 5¼-inch and 1 μs for 8-inch FMDAT (single-density data) pulses, and 1 μs and 500 ns for 5¼-inch and 8-inch MFMDAT (double-density data) pulses, respectively.

A type-74S124 voltage-controlled oscillator (VCO), IC4, generates a free-running 8-MHz VCO output frequency used to track the incoming data stream. The VCO frequency is also divided by 2 to produce a 4-MHz clock pulse. Jumpers JP8 and JP15 select the correct VCO frequency for the type of drive in use (8 MHz for 8-inch and 4 MHz for 5¼-inch).

The read-data pulse for the 8272's RDD input is derived from IC13 and IC16. Pin 5 of IC13 (the Q output) goes high when this flip-flop detects the rising edge of each inverted data pulse, which corresponds to the leading edge of the negative-going raw data pulse from the disk drive. On the rising edge of the next inverted 8-MHz VCO-clock pulse, the Q output of IC13 is then clocked into flip-flop IC16, forming the positive RDD pulse required by the 8272.

CP/M-86 BIOS

Digital Research's CP/M-86 operating system is designed to operate in almost any 8086- or 8088-based microcomputer system. This flexibility has been made possible by dividing the operating-system code into functional sections, one of which is accessible to the computer's manufacturer, dealers, and users. This section is the lowest-level portion and is called the *basic input/output system* or BIOS (usually pronounced "by-ahs" or "by-ohs" for short).

The higher-level BDOS (basic disk operating system—"bee-dahs"), the nucleus of CP/M-86, calls on the BIOS to gain access to the physical hardware of the computer system, in our case, the MPX-16. This provides a very machine-independent environment for the BDOS.

Imagine the BIOS as a slave that the BDOS can order around. The BDOS knows what it wants to do (communicate with the disk controller or console serial port, for example) but doesn't know exactly how to talk to the hardware. It does have rapport with the BIOS, though, and can ask the BIOS to communicate with the hardware and return the results.

As a user, you will almost always receive your CP/M-86 computer system with a customized BIOS previously installed by your manufacturer or dealer. But if you buy CP/M-86 directly from Digital Research, it will not contain a BIOS that will work with the MPX-16. To support this project, I have arranged for a customized BIOS to be written, burned into EPROMs, and distributed by The Micromint for use with the MPX-16.

The inner workings of the BIOS and full instructions on how to customize it are too complex to deal with in this article and are covered in great detail in the CP/M-86 documentation, so rather than duplicate that material, I shall attempt to explain in English terms what the various parts of the BIOS do.

BIOS Organization

The BIOS portion of CP/M-86 resides constantly in user memory during normal system operation. When power is first applied to the MPX-16, the 8088 processor comes up executing instructions at the very top of memory, in the space assigned to EPROM in the MPX-16. The first instruction it encounters is an initialization vector that causes control to branch to the initialization routine. This routine first performs diagnostic operations to make sure that the system is working properly, then it copies the BIOS out of its storage locations in the EPROM into addresses low in memory. Control is then transferred to the cold-start vec-

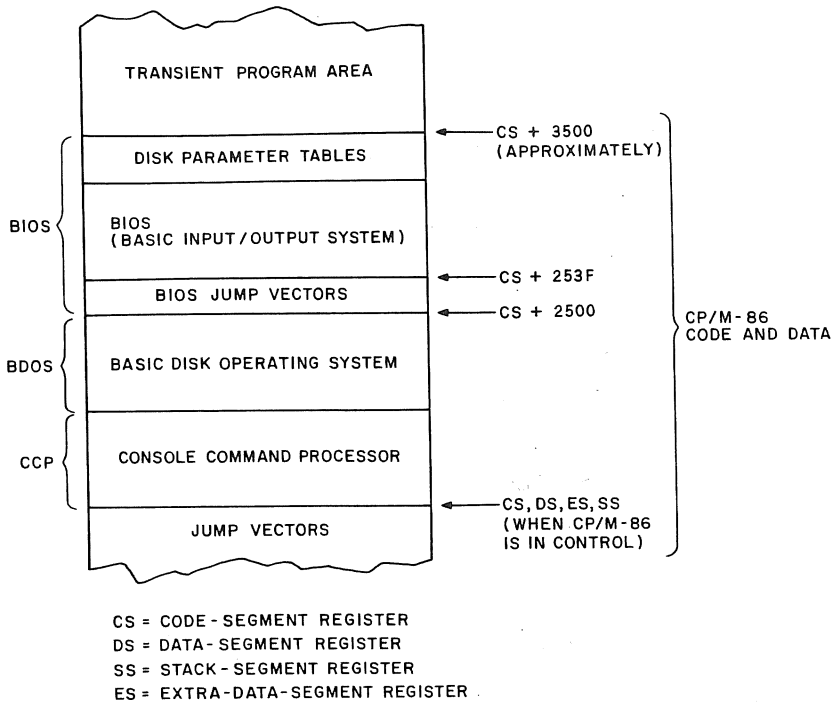


Figure 3: *Memory map of the CP/M-86 operating system as configured for the MPX-16. In 64K-byte systems, the CS, DS, SS, and ES registers will all contain a value of zero, and the segments will overlap. User programs are loaded into the TPA (transient program area).*

Offset from Start of BIOS	Instruction	BIOS Function Number	Description
0000	JMP INIT	0	cold start
0003	JMP WBOOT	1	warm start
0006	JMP CONST	2	console status check
0009	JMP CONIN	3	console character input
000C	JMP CONOUT	4	console character output
000F	JMP LIST	5	list-device character output
0012	JMP PUNC	6	punch-device character output
0016	JMP READER	7	reader-device character input
0018	JMP HOME	8	move to track 0
001B	JMP SELDSK	9	select a disk drive
001E	JMP SETTRK	10	set track number
0021	JMP SETSEC	11	set sector number
0024	JMP SETDMA	12	set DMA-offset address
0027	JMP READ	13	read selected disk sector
002A	JMP WRITE	14	write selected disk sector
002D	JMP LISTST	15	return list-device status
0030	JMP SECTRAN	16	sector translation
0033	JMP SETDMAB	17	set DMA segment address
0036	JMP GETSEGB	18	get MEM region table offset
0039	JMP GETIOB	19	get IOBYTE
003C	JMP SETIOB	20	set IOBYTE

Table 3: *BIOS (basic input/output system) jump vectors for CP/M-86 on the MPX-16. These jump instructions are the 21 entry points to the BIOS. The BDOS module calls these subroutines when it needs to send commands or receive data from the actual hardware (machine-dependent) interfaces, such as disk drives or serial ports. The offset address is from the start of the BIOS, which is located at an address in memory hexadecimal 2500 locations up from the start of the CCP/BDOS code segment.*

tor of CP/M-86, and normal operation begins.

Figure 3 shows a typical memory map for a CP/M-86 installation. The BIOS is made up of several subsections. The first 63 bytes contain 21 jump vectors, each 3 bytes long. Each jump vector is an instruction to transfer control to the address in memory of a routine that performs an assigned low-level function, such as restarting CP/M-86 or getting a console character. These functions are listed in table 3.

As shown in figure 3, the BIOS resides in memory at an address offset by hexadecimal 2500 from the base address of CP/M-86. This offset is constant, but the upper boundary of the BIOS may change, depending on the size and special requirements of the microcomputer hardware. For example, some disk controllers are interrupt-driven, some are set up to use DMA transfers, and some use regular I/O transfers to communicate with the processor. The complexity of the BIOS depends on how many different features like these it must support.

The first two jump vectors, as shown in table 3, are for system reinitialization. The first one is called directly by the CP/M-86 loader program and performs any needed hardware initialization when CP/M-86 is loaded "from cold start" (for the first time after the computer is turned on). The second is called the "warm-start" vector because it is called whenever a program terminates (through BDOS function 0). After the warm-start operation has been completed, control is immediately transferred to the part of CP/M-86 with which the user converses, the console command processor, or CCP.

The next six jump vectors in table 3 transfer control to various character-I/O routines. In all of the routines, a character being sent out to a device must be placed in the CL register, and any character or status information being returned will appear in the AL register. For example, CONST, CONN, and CONOUT pass characters to and from the logical console device in this manner. The next vector (LIST) sends a character to the

logical list device (usually the printer). Further down, we see that function 15 (LISTST) returns the status of the list device.

The reason why the list-status routine is not located adjacent in memory to the list-output routine is simple: when the first version of CP/M-80 was written, no list-status routine existed. It was added later, but to avoid rearranging all the jump vectors, it was added as function 15. In CP/M-86, other jump vectors were added after it. The logical device names Reader and Punch are actually obsolete. They were intended for a paper-tape reader and punch, but these routines are now used to operate various auxiliary input and output devices.

Disk I/O Routines

BIOS functions 8 through 14 and function 16 are used for disk-controller communications. For example, the HOME function causes the currently selected disk to return to track 0 (that is, it causes the read/write head to seek to the outermost track). The SELDSK function activates the disk drive whose address is passed in the CL register and makes it the current disk (this is how the default disk is activated).

The READ and WRITE functions transfer a single record (128 bytes) from the current DMA buffer (set with SETDMA) to or from the currently selected disk (SELDSK) at the current track and sector (SETTRK and SETSEC). The BDOS refers to the disk directory on disk to know where to read or write information when needed.

Disk-Definition Tables

All of the recently introduced operating systems from Digital Research, including CP/M-86 and CP/M-80 version 2.2, are table-driven. This means that all the disk definitions and storage-allocation information is kept in tables in the section of memory occupied by the BIOS, rather than in the BDOS. This allows for flexibility in interfacing disk drives and other peripheral devices to the system. Early versions of CP/M-80 assumed that all disks attached to the system were identical: 8-inch single-density drives. Now, many systems have one to four floppy disks, and perhaps an additional hard disk, for mass storage. A few even have so-called RAM disks (large-capacity semiconductor random-access read/write memories set up to simulate disk drives). Because the modification of the tables is usually performed by an experienced programmer, the user rarely has the need to modify them. (To keep this article from running overlong, I'll let those of you who are really interested look to the CP/M-86 documentation to learn those software mysteries.)

In Conclusion

That's all the information on the MPX-16 we can reasonably cover in three magazine articles, but more information is available for those of you who need it in the *MPX-16 Technical Reference and User's Manual*, available separately from The Micromint.

You've probably noticed a great reliance on Intel components throughout the computer. These are present in the MPX-16 for compatibility, because they are used in the IBM Personal Computer, but I suspect that IBM's design team selected these components because of Intel's foresight in promptly supporting its 16-bit microprocessors with parts that work well together, at reasonable cost, in a complete solution to a computer-design problem.

Overseeing the design of the MPX-16 has been quite an adventure for me these past few months. I hope you've enjoyed reading this epic.

Next Month:

We'll look at a single-line alphanumeric liquid-crystal display for use in a portable computer terminal. ■