THE DEFINICON 68020 COPROCESSOR

A plug-in board that provides 32-bit computing power and math processing for 16-bit machines

Have you ever wished that your personal computer had the speed of a VAX or that the number 64K had never existed? Well, the Motorola 68020 32-bit microprocessor has the computing power to come close to fulfilling both those dreams. BYTE has arranged for Definicon Systems to design a special version of its 68020-based scientific microcomputer system that allows 68020 hardware and software development at a price all can afford. (See the text box "How to Get Your DSI-020 Coprocessor System" on page 122 for complete information.)

Trevor Marshall, Christopher Jones, and Sigi Kluger are engineers with Definicon Systems Inc. They can be contacted at 31324 Via Colinas #108/9, Westlake Village, CA 91362.

The DSI-020 is a coprocessor board (see photo 1) that uses the 12.5-megahertz versions of the 68020 CPU and the 68881 FPU (floating-point unit). Provision has been made for the addition of the 68851 PMMU (paged-memory-management unit) when it becomes available. The board has 1 megabyte of high-speed parity-checked dynamic RAM that requires only one CPU wait state to operate correctly. There are two high-speed (38,400 bps) serial ports (figure 1) and an interrupt-driven timer to ease the task of porting UNIX.

All the facilities of a host IBM PC or compatible are available to the coprocessor, including networking and access to any other peripherals connected to the PC's bus. System calls for full bit-mapped graphics are provided in the kernel. All BDOS and BIOS software interrupts in the PC are available to the 68020 programmer.

OVERVIEW

The DSI-020 board is composed of a number of relatively independent functional blocks. The CPU cluster (figure 2) consists of the 68020, 68881, and 68851. The 32-bit bus address decoding circuitry is shown in figure 3 and includes the FPU20 and DEC20 PALs, the HOLD20 bus request arbiter PAL, the 32-bit bus DMA sequencer (74F161 and DPORT20 PAL), the DSACK20 PAL, and the BE20 byte enable PAL (figure 4). The DSACK20 PAL's main function is to provide signals to the 68020 to indicate that a memory access cycle is complete and whether the size of the transfer was 8, 16, or 32 bits.

A delay line and associated headers delay the CPU's address strobe (AS) signal long enough for all the decoding to have been completed before the $\overline{\text{RASIN}}$ signal is generated to initiate the DP8409 dynamic RAM controller's access cycle. The DP8409 (figure 5) directly drives the multiplexed address and $\overline{\text{RAS}}$ lines of the RAMs. The $\overline{\text{CAS}}$ signal is multiplexed with the Byte Enable signals before being applied to the RAM array, shown in figure 6. The refresh counter is driven from the CPU's clock oscillator, divided by 128. The DP8409 attempts to do a hidden refresh, but if this is not possible, a refresh request ($\overline{\text{RFIO}}$) will be sent to the HOLD20 PAL every 10.24 microseconds. When this is acknowledged (with $\overline{\text{RFSHACK}}$) the DP8409 will perform a $\overline{\text{RAS ONLY}}$ refresh cycle.

Hidden refreshes occur whenever any device other than the RAM itself is addressed. In particular, accesses to the 68881 will cause a hidden refresh to occur. This considerably reduces any overhead that might otherwise be incurred using this DMA refresh technique.

BY TREVOR MARSHALL, CHRISTOPHER JONES, AND SIGI KLUGER

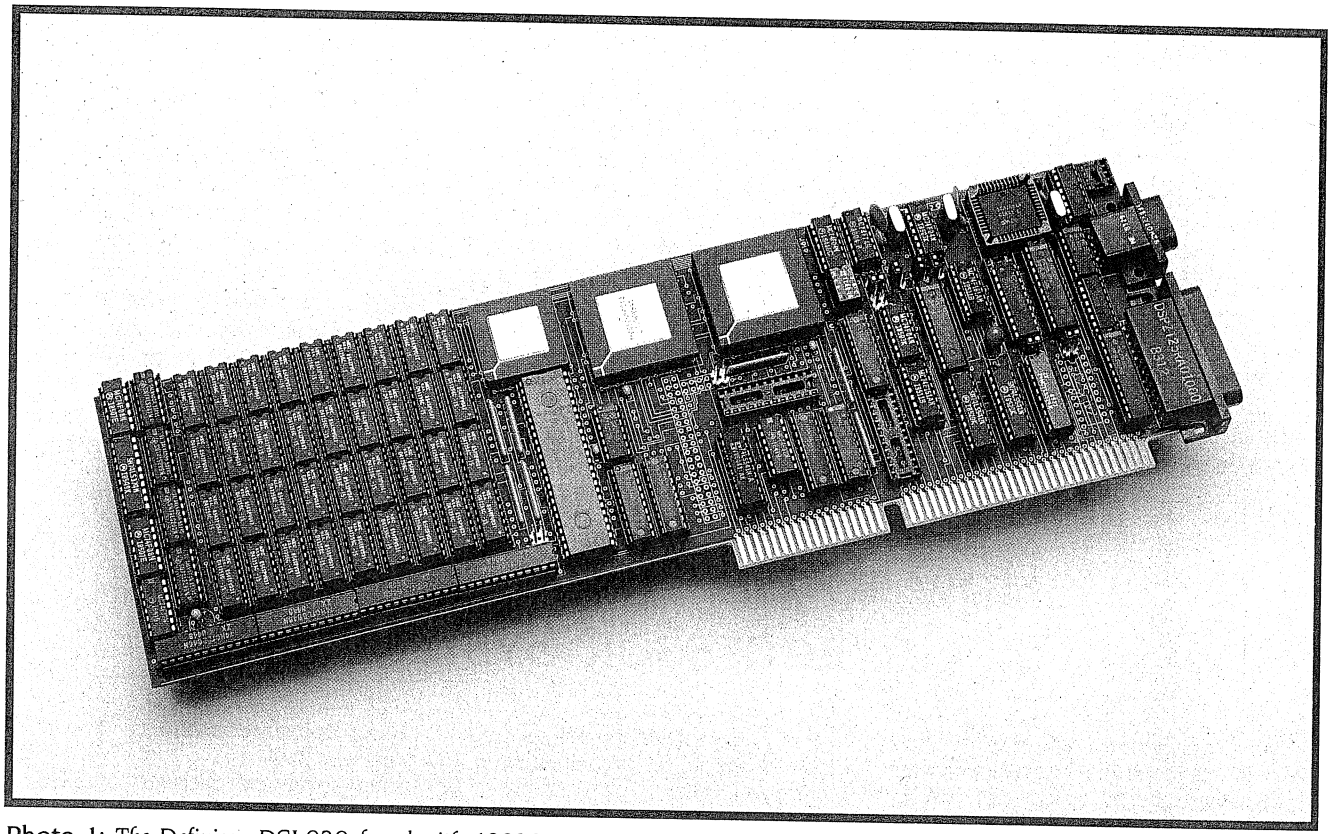


Photo 1: *The Definicon* DSI-020 *board with* 68020 CPU, 68881 FPU, *and* 1 *megabyte of memory.* A 68851 PMMU *is shown with this board although it is not commercially available at this time.*

The parity error signal generated as shown in figure 7 is latched and can be read by the host PC at bit 0 of the BFLAGS port (see table 1 and figure 4). Pulse the Refresh Inhibit line (RFSHINH) briefly to reset this latch.

THE 68851 PMMU

The principal function of the PMMU is the translation of logical addresses from the software running in the CPU to physical addresses within the range of the available hardware facilities. This allows programs to run at addresses that are not available in physical memory. An operating system that swaps infrequently used memory blocks to disk and so enlarges the effective addressing range of programs is called a virtual memory system (VMS). Although Definicon has written a VMS for the DSI-32, the kernel for the DSI-020 does not yet implement virtual memory. The mapping from logical to physical addresses is stored in tables (in memory) that the PMMU can search. It has an internal cache so that the most recently used translation table entries are available without the necessity for an additional bus cycle. The PMMU implements a hierarchical protection mechanism with up to eight access levels. It also provides a breakpoint acknowledge facility to support the 68020 breakpoint instructions.

The 68851 internally decodes its own chip select function.

THE 68881 COPROCESSOR

The 68881 is the first high-speed floating-point accelerator from Motorola. It is addressed when the 68020 places 111 on its status lines (FC0, FC1, and FC2), 0010 on lines A19 through A16, and 001 on lines A15 through A13 and performs a memory cycle. The special coding on the status lines indicates, however, that the cycle is actually for the coprocessor interface. Chip select for the 68881 is derived in the FPU20 PAL.

The major feature of the 68881 is that many frequently used mathematical functions have been microcoded into the silicon. This considerably enhances the performance of the 68020/68881 combination on engineering and scientific computations. It also accounts for its deceptively high performance on some function-sensitive benchmarks, such as the Savage benchmark (*Dr. Dobb's Journal*, March 1984, page 92).

THE HOST PC BUS INTERFACE

The IBM PC XT bus was designed as a single master bus, so there is no

(continued)

PHOTOGRAPHED BY PAUL AVIS

The 8088 is used to relieve the 68020 coprocessor of many disk and console I/O tasks.

method for a slave coprocessor to take control of the bus, tell the 8088 CPU to go to sleep, and drive the peripherals directly. The XT relies on the 8088's DMA controller to take back control of the bus every 15 microseconds or so to perform a dynamic RAM refresh cycle. This forces a coprocessor system such as the DSI-020 either to replace the 8088 or to work in close cooperation with it.

With the design of the PC AT, provision was made for a slave processor to control the bus (via the DRQ and MASTER signals), but a limitation of 15 microseconds remains on the total length of time a slave can keep control before memory refresh fails! This limit obviously makes it very difficult to provide safe direct disk or other stream I/O tasks even using these newer capabilities. It is for these reasons that the DSI-020 is designed to transfer data 1 byte (or word) at a time to the host environment.

During the design of the DSI-32 (August 1985 BYTE, page 120) we found that the total replacement of the 8088 was relatively inefficient and that it is capable of relieving the coprocessor of many of the tedious disk and console I/O tasks. The DSI-020 follows the same design criteria. The MS-DOS system on either a PC AT or a PC XT is used to provide file I/O, networking, and other peripheral tasks, allowing the 68020 kernel to concentrate on the things it does best. The kernel provides the interface and translation between the UNIX-derived 68020 system software and the facilities provided by MS-DOS itself. We allowed the optional use of the 16-bit data transfer facility of the new AT bus. The majority of the improvement resulting from the use of

HOW TO GET YOUR DSI-020 COPROCESSOR SYSTEM

Definicon Systems will provide the elements of the DSI-020 system at the following special prices for BYTE readers.

Note: These special prices have been provided by Definicon, reducing its margins to a bare minimum. In particular, no margin has been allowed for accounting overhead and so *no purchase orders* can be accepted for these special BYTE products.

Terms of payment are VISA/MasterCard/American Express only. There is a 30-day, no-questions-asked, money-back guarantee. Goods must be returned in "as new" condition in original packaging for full credit.

Software support available for all products is limited to a diagnosis and correction of your software problem. Under no circumstances will Definicon's technical support team attempt to teach you to program in any of these languages!

The DSI-020 has been tested in almost all the machines compatible with the IBM XT and AT. If it does not work in yours, Definicon reserves the right either to correct the problem (if it is with the DSI-020) or refund your money.

Although the DSI-020 will operate in a system with only floppy disks, a fixed disk is essential for meaningful program development.

DSI-020 basic hardware: $995
1 megabyte of RAM, 12.5-MHz 68020, and 12.5-MHz 68881
No parity chips or serial ports
MS-DOS interface software, simplified assembler, linker, and assembly-level debugger
The DSI-020 hardware is supplied fully assembled and tested.

Optional hardware upgrade: $99
Chips for the two serial ports, parity-detection circuitry, and the serial number EEPROM
Supplied fully tested if ordered at the same time as the basic hardware.

SVS Library Manager: $49
Forms object modules into libraries that can be searched by the linker rather than included as a whole.

QUELO Macro Assembler and Utilities: $155

SVS BASIC PLUS: $169
BASIC language interpreter similar to DEC's BASIC PLUS language

Living Software BASIC to C Converter: $299
Includes run-time package compiled for compatibility with either SVS C or LLL C.

SVS C: $349
Kernighan and Ritchie definition with most UNIX extensions

LLL C: $349
Kernighan and Ritchie definition with most UNIX extensions

LLL PASCAL: $399
ANSI level 1 PASCAL

SVS PASCAL: $399
ANSI level 0 PASCAL plus extensions

SVS FORTRAN: $459
ANSI FORTRAN-77 language with extensions

All the above items may be ordered only directly from
Definicon Systems Inc.
31324 Via Colinas #108
Westlake Village, CA 91362
(818) 889-1646

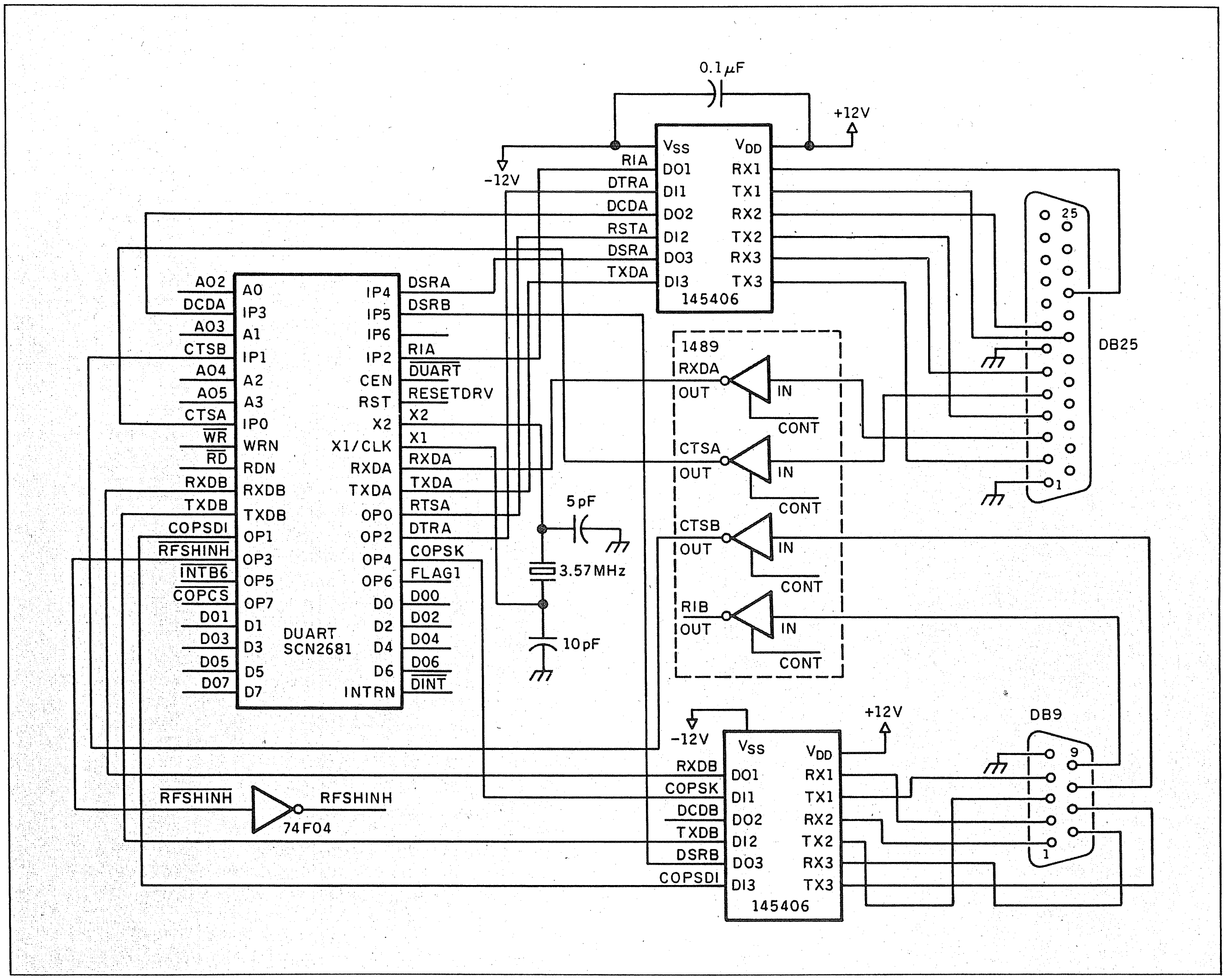


Figure 1: DUART *and* RS-232C *drivers.*

an AT host comes, however, from the faster disk subsystem and not from the wider bus.

The DSI-020 automatically senses whether a PC XT 8-bit bus or a PC AT 16-bit bus is available and adjusts to either 8- or 16-bit transfers as appropriate. When an XT is sensed, by the absence of a ground level on pin D18 of the interface (figure 8), the 74HCT245 buffer (figure 9) connects the top half of the 16-bit bus to the lower half of the bus so that the XT can see it. When an AT bus is present, the DSI-020 will assert $\overline{\text{MEMCS16}}$ to the host whenever it is addressed at an even boundary. This conditions the AT for a possible 16-bit data transfer. If for some reason you are using an XT work-alike in which the 18-pin AT connector fingers are physically obstructed by components on the motherboard, it is possible to have the extra connector removed.

A jumper block has been provided to disable the AT sensing circuitry. This allows the use of the DSI-020 in an AT using the XT bus mode.

THE DMA CYCLE

When the 68020 kernel wants the 8088/286 to perform a task, it asserts interrupt 2 (optionally INT 7). The 8088/286 addresses the card. The AT20 PAL (figure 4) generates a $\overline{\text{RAMSEL}}$ signal, which in turn generates an $\overline{\text{IOCHRDY}}$ signal to the host. This forces the 8088/286 to wait until the 68020's 32-bit bus becomes available. The DPORT20 also issues the $\overline{\text{HOLD86}}$ request to the HOLD20 PAL. This PAL arbitrates the DMA request with refresh requests from the dynamic RAM controller and produces a BUS REQUEST ($\overline{\text{BR}}$) to the 68020. On seeing this request the 68020 waits until it has completed the current 32-bit bus cycle and then releases its bus and asserts BUS GRANT ($\overline{\text{BG}}$). The HOLD20 PAL then issues $\overline{\text{HLDA86}}$, which releases the 74F161A DMA sequencer. The DPORT20 PAL then takes over the 32-bit bus and generates signals that simulate some of the control signals normally generated by the 68020. The dynamic RAM controller responds to these signals, placing the required byte (or word) of data on the 32-bit

(continued)

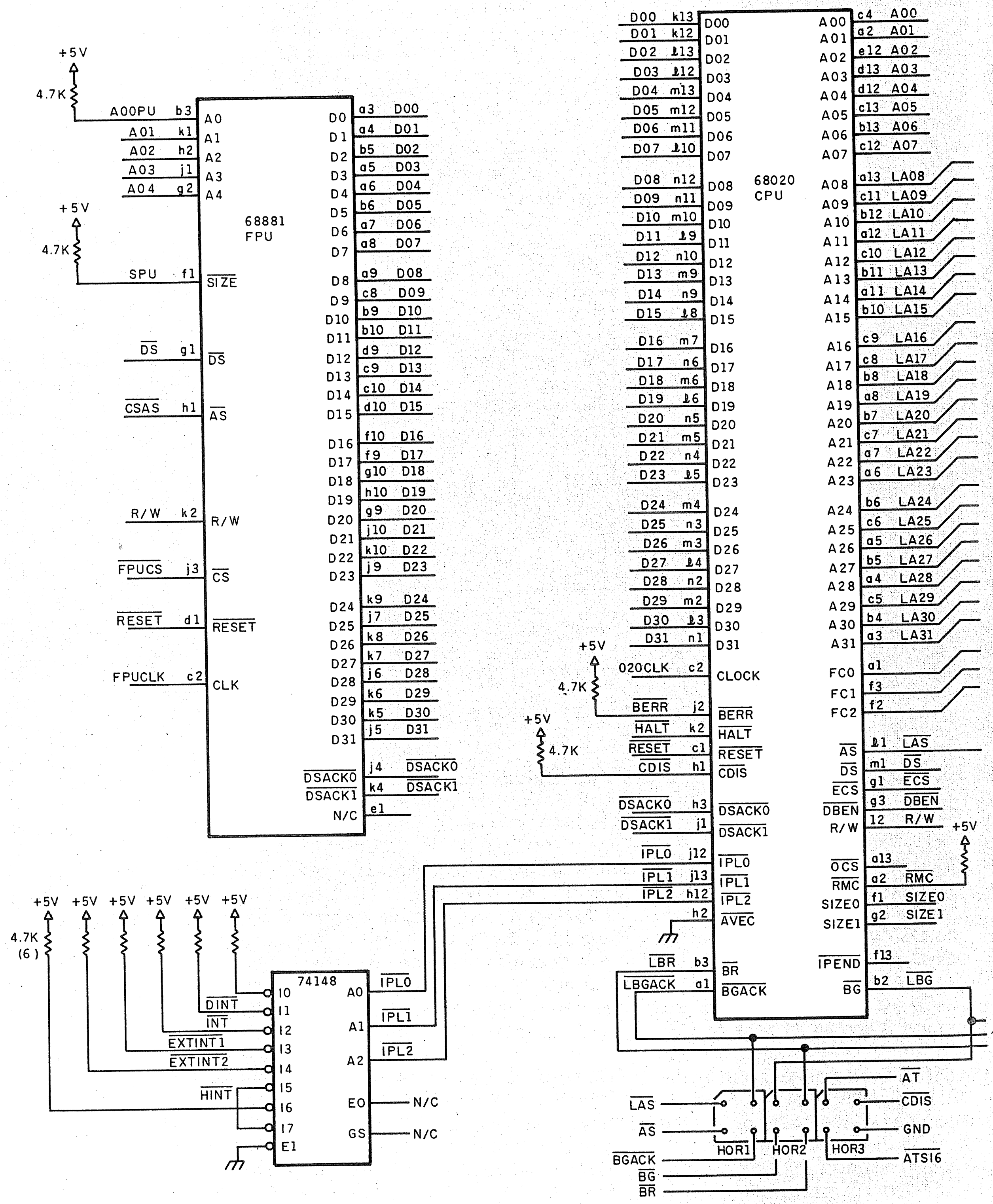
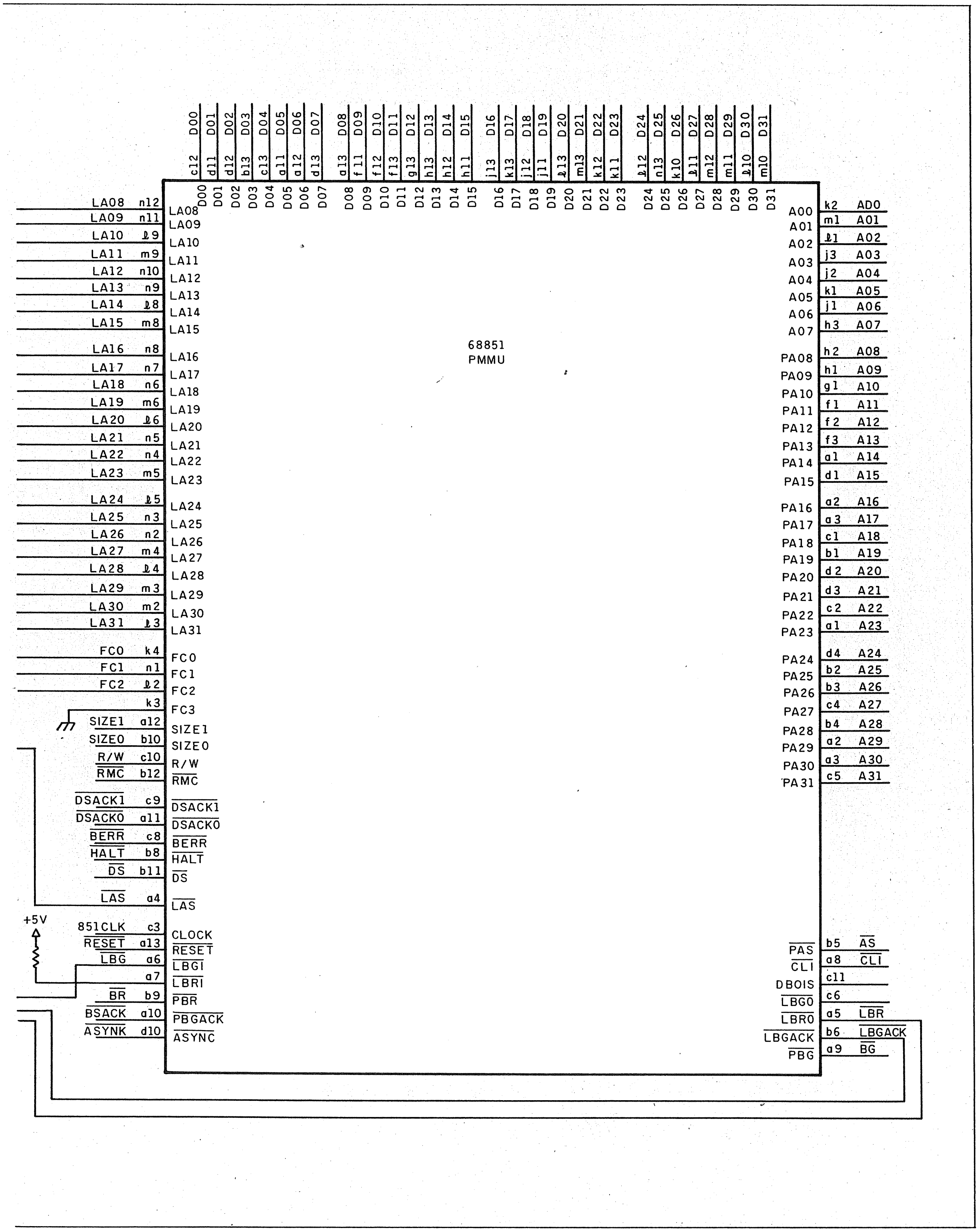


Figure 2: *The* CPU, *floating-point* (FPU), *and paged-memory-management* (PMMU) *processors.*


68851
PMMU
+5V
851CLK
CLOCK
RESET
LBG
BR
BSACK
ASYNK
AS
CLI
LBR
LBGACK
BG

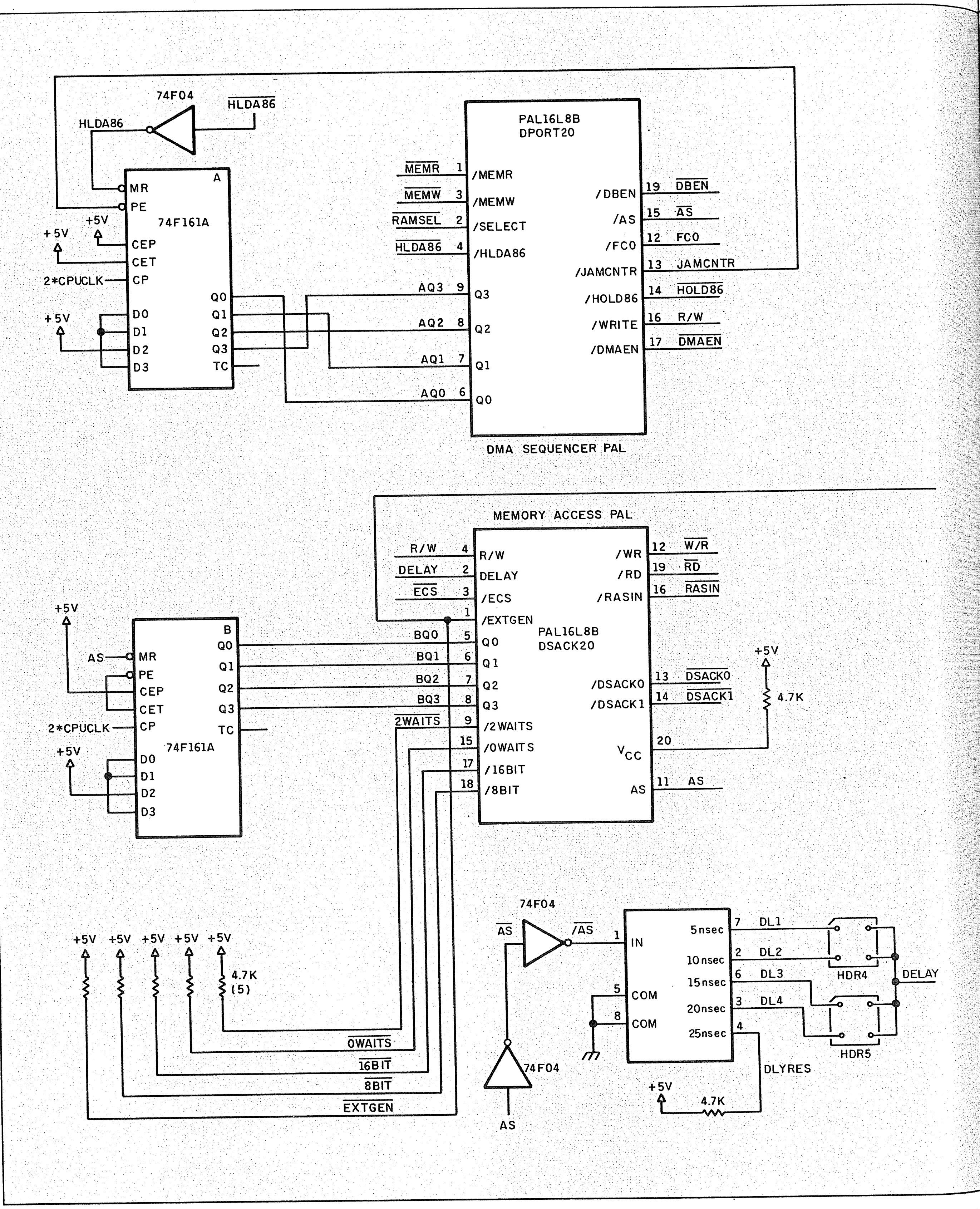
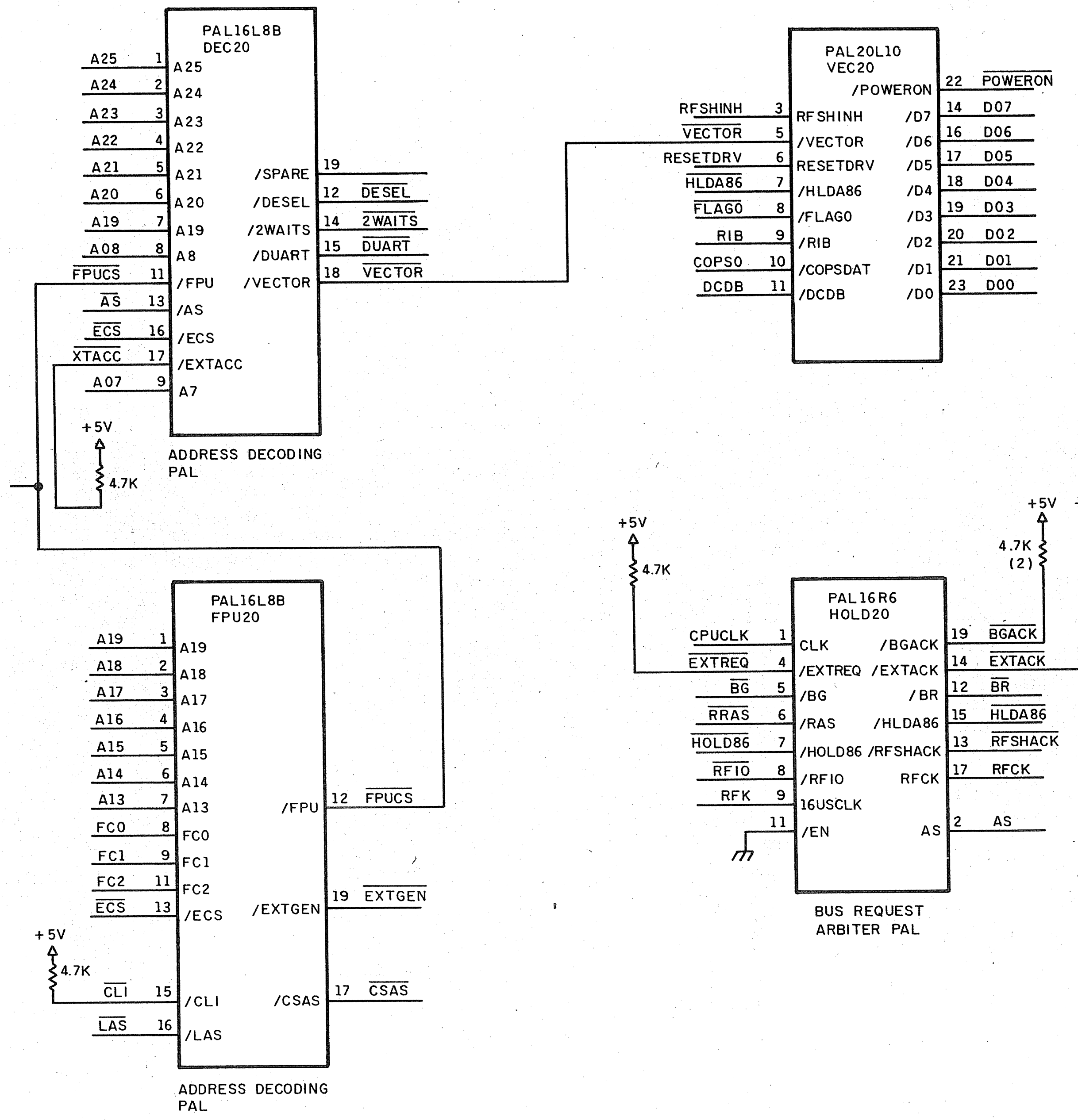


Figure 3: *Bus address decoding PALs.*


PAL16L8B
DEC20
A25 1 A25
A24 2 A24
A23 3 A23
A22 4 A22
A21 5 A21
A20 6 A20
A19 7 A19
A08 8 A8
FPUCS 11 /FPU
AS 13 /AS
ECS 16 /ECS
XTACC 17 /EXTACC
A07 9 A7
/SPARE 19
/DESEL 12 DESEL
/2WAITS 14 2WAITS
/DUART 15 DUART
/VECTOR 18 VECTOR
+5V
4.7K
ADDRESS DECODING PAL
PAL20L10
VEC20
/POWERON 22 POWERON
RFSHINH 3 RFSHINH
VECTOR 5 /VECTOR
RESETDRV 6 RESETDRV
HLDA86 7 /HLDA86
FLAG0 8 /FLAG0
RIB 9 /RIB
COPS0 10 /COPSDAT
DCDB 11 /DCDB
/D7 14 D07
/D6 16 D06
/D5 17 D05
/D4 18 D04
/D3 19 D03
/D2 20 D02
/D1 21 D01
/D0 23 D00
PAL16L8B
FPU20
A19 1 A19
A18 2 A18
A17 3 A17
A16 4 A16
A15 5 A15
A14 6 A14
A13 7 A13
FC0 8 FC0
FC1 9 FC1
FC2 11 FC2
ECS 13 /ECS
CLI 15 /CLI
LAS 16 /LAS
/FPU 12 FPUCS
/EXTGEN 19 EXTGEN
/CSAS 17 CSAS
+5V
4.7K
ADDRESS DECODING PAL
+5V
4.7K
+5V +5V
4.7K (2)
PAL16R6
HOLD20
CPUCLK 1 CLK
EXTREQ 4 /EXTREQ
BG 5 /BG
RRAS 6 /RAS
HOLD86 7 /HOLD86
RFIO 8 /RFIO
RFK 9 16USCLK
11 /EN
/BGACK 19 BGACK
/EXTACK 14 EXTACK
/BR 12 BR
/HLDA86 15 HLDA86
/RFSHACK 13 RFSHACK
RFCK 17 RFCK
AS 2 AS
BUS REQUEST ARBITER PAL

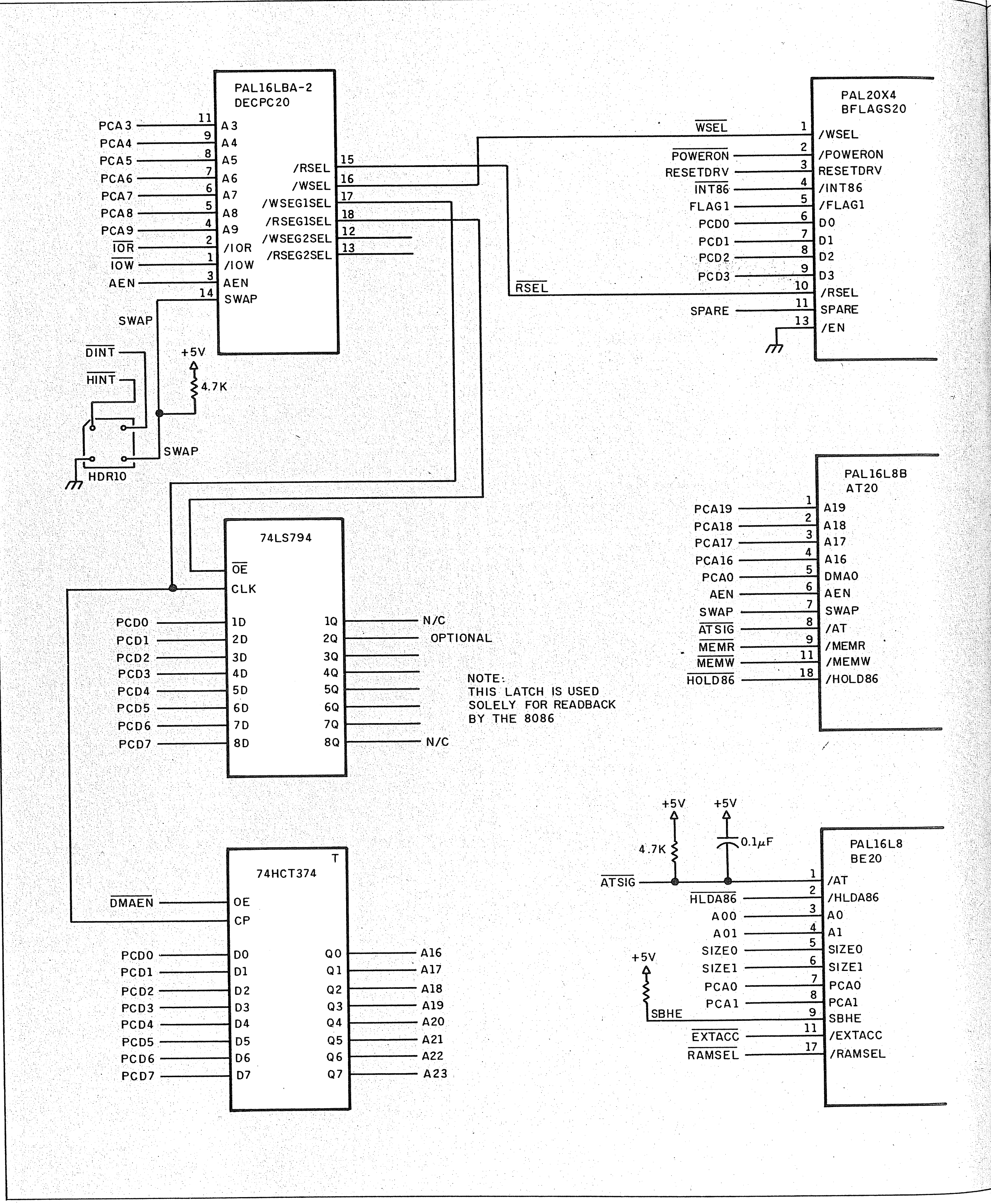
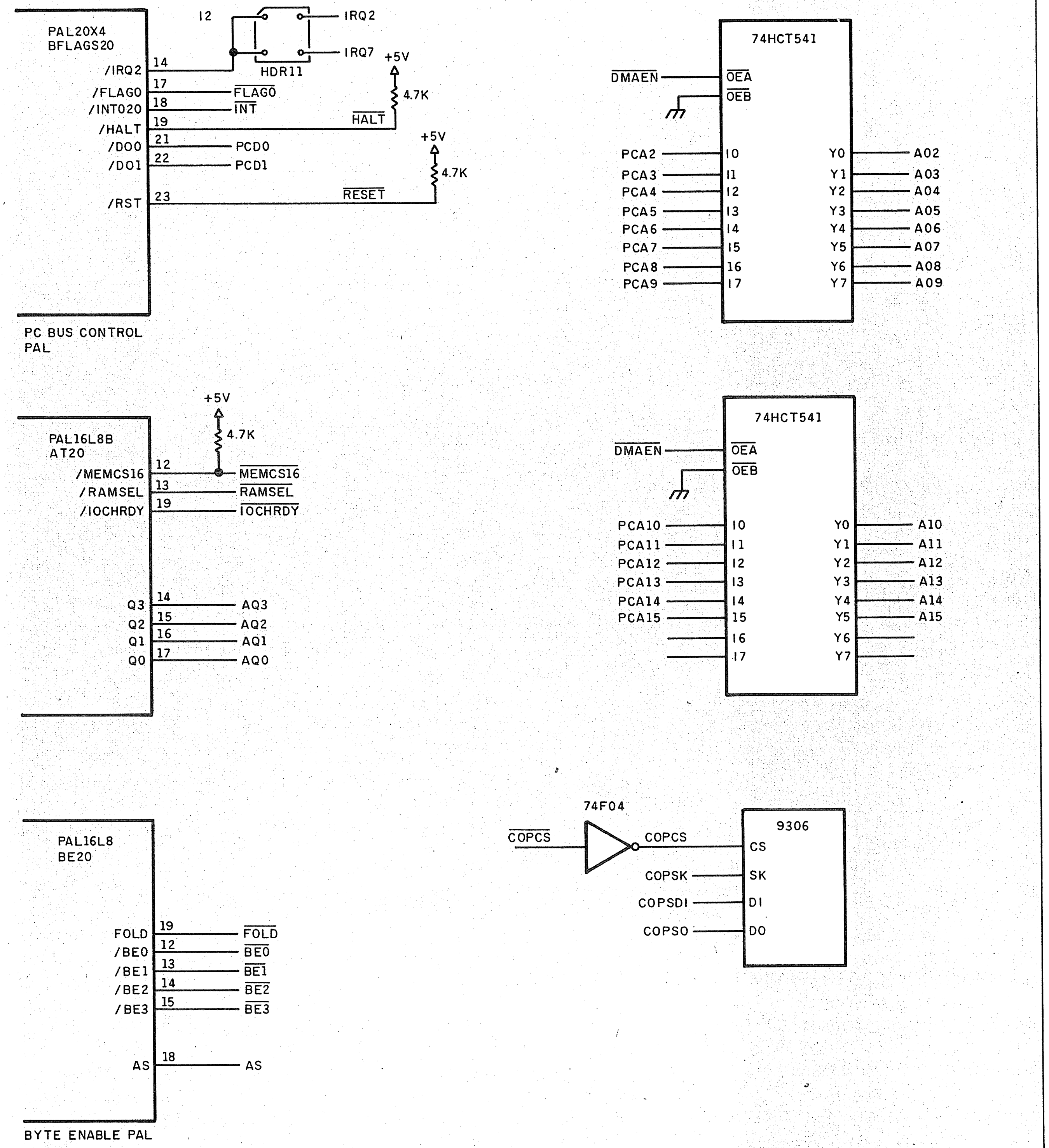


Figure 4: PC *interface circuitry and* PALs.


PAL20X4
BFLAGS20
/IRQ2
/FLAG0
/INT020
/HALT
/D00
/D01
/RST
14
17
18
19
21
22
23
12
HDR11
IRQ2
IRQ7
FLAG0
INT
HALT
PCD0
PCD1
RESET
+5V
4.7K
PC BUS CONTROL PAL
PAL16L8B
AT20
/MEMCS16
/RAMSEL
/IOCHRDY
MEMCS16
RAMSEL
IOCHRDY
Q3
Q2
Q1
Q0
AQ3
AQ2
AQ1
AQ0
PAL16L8
BE20
FOLD
/BE0
/BE1
/BE2
/BE3
AS
BE0
BE1
BE2
BE3
BYTE ENABLE PAL
74HCT541
DMAEN
OEA
OEB
PCA2
PCA3
PCA4
PCA5
PCA6
PCA7
PCA8
PCA9
I0
I1
I2
I3
I4
I5
I6
I7
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
A02
A03
A04
A05
A06
A07
A08
A09
PCA10
PCA11
PCA12
PCA13
PCA14
PCA15
A10
A11
A12
A13
A14
A15
74F04
COPCS
9306
CS
SK
DI
DO
COPSK
COPSDI
COPSO

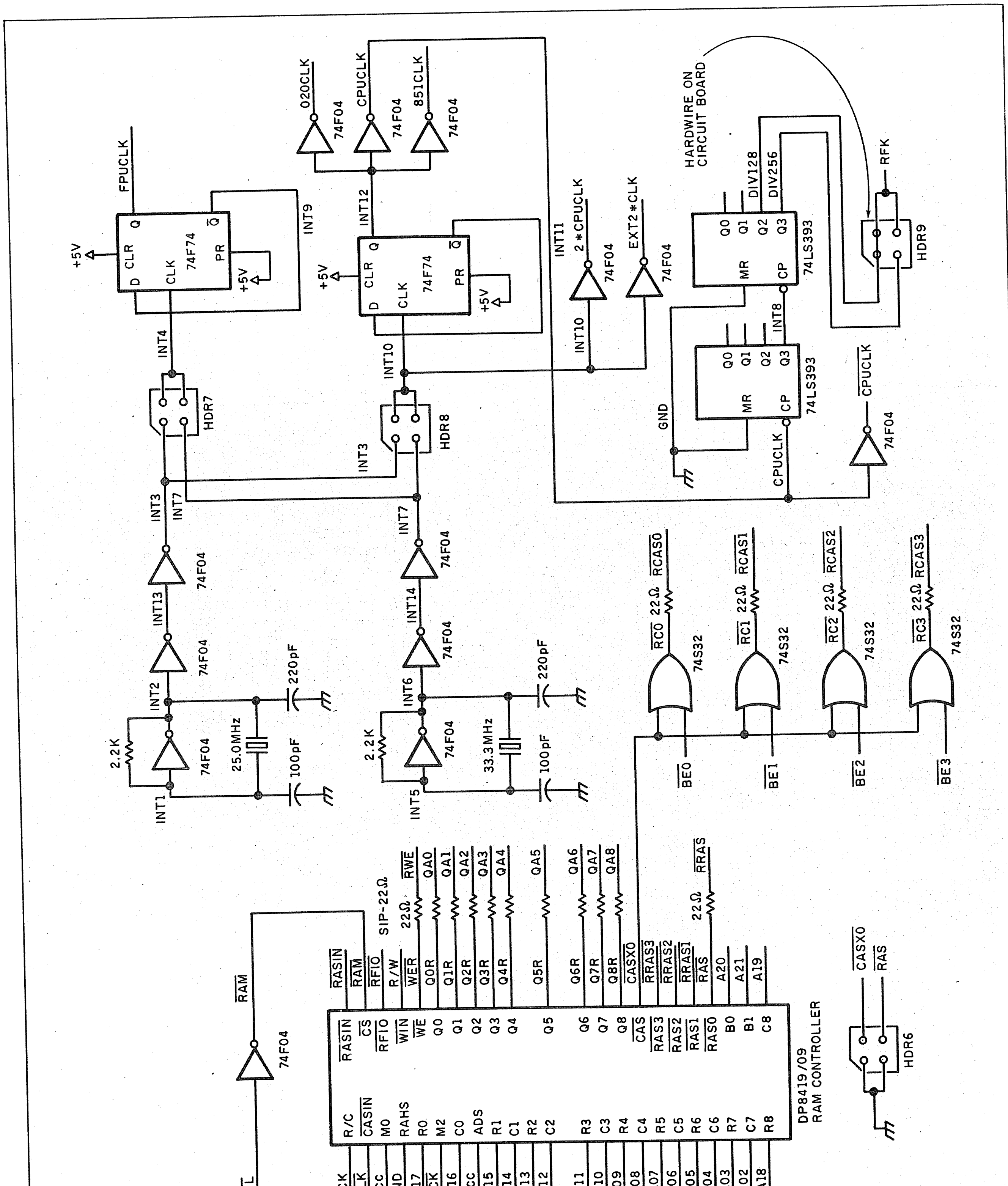


Figure 5: RAM *controller circuitry.*

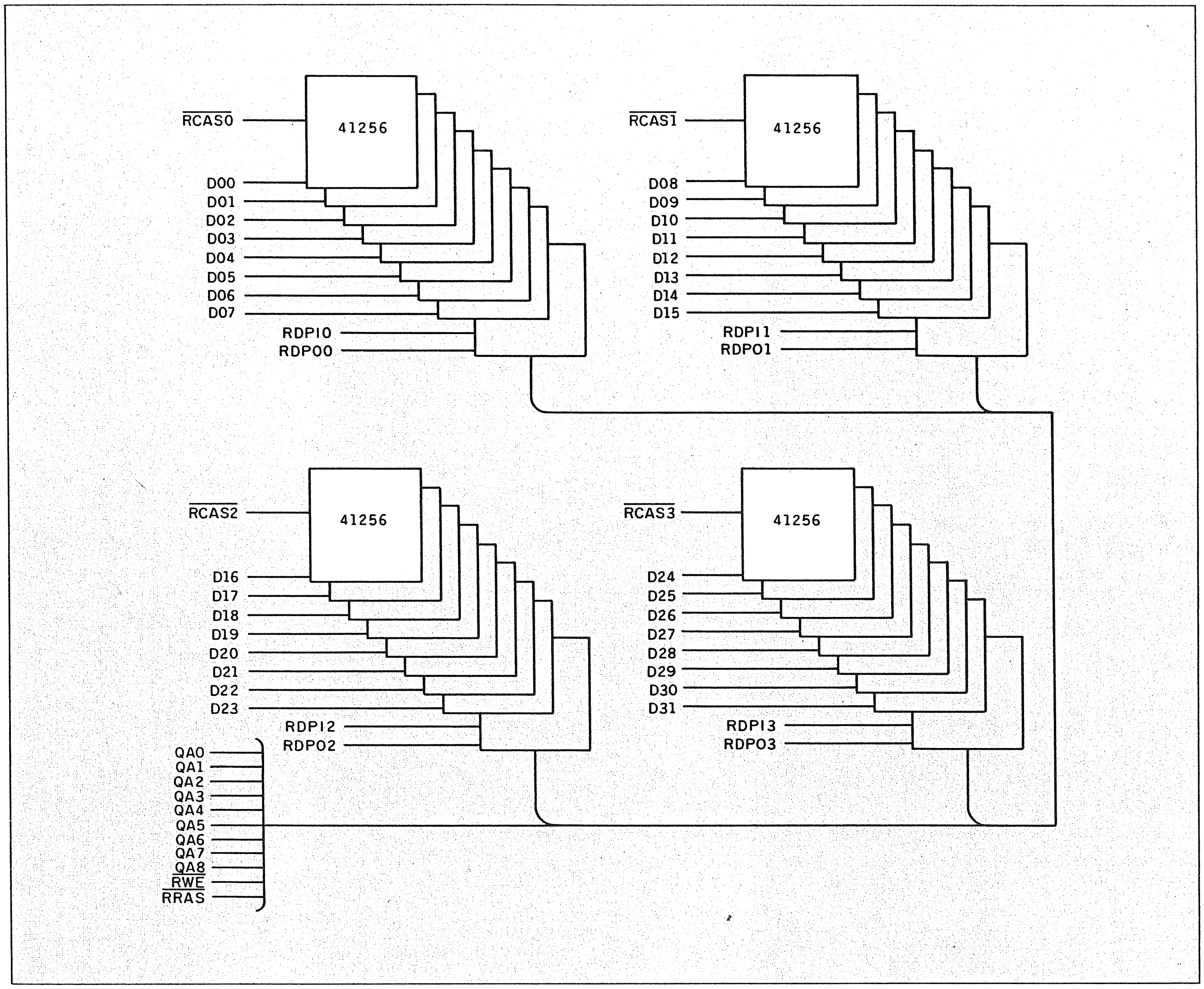


Figure 6: *Memory array.*

bus so that it can be read by the host computer's data bus.

Despite the number of discrete operations during a DMA transfer, the entire cycle is usually performed in less than a microsecond. Since the 68020 continues to process instructions within its cache, DMA transfers usually do not significantly affect execution speed. This allows a graphics application to continuously update the display adapter's screen without slowing the formation of the next image.

WHY INTEL DOESN'T TALK TO MOTOROLA

A major difficulty arises when interfacing an Intel-like processor (such as the 8088 or 80286) with any of the Motorola processors. The Intel architecture defines that, starting at the base address (usually 0) the first byte (at address offset 0) holds data bits 0–7 (see table 2), the second byte (at address offset 1) holds data bits 8–15, and the third and fourth bytes hold data bits 16–31.

Motorola, however, has defined that the first byte holds bits 24–31 and so on in exactly the reverse order. This has the advantage that when you examine a hexadecimal dump of Motorola memory the data is exactly as you read it, the most significant byte first, least significant last, and no mental reordering of bytes is necessary. The disadvantage is, of course, that the DSI-020 design had to provide for this byte reordering, so that data from the PC's screen or disk files would appear to the 68020 rearranged in the correct byte order.

This feature is provided within the hardware design, as software fixes proved far too slow for disk I/O. If you look carefully at figure 4, you will note that A0 and A1, the two least significant (LS) address lines, are not used directly but go to a PAL, the Byte Enable (BE20) PAL, before they interface with any of the memory or peripherals. In addition, PCA0 and PCA1, the LS address lines from the PC, also go to this PAL. Table 2 gives the design equations for this PAL. It can

(continued)

be seen that it takes the lower address lines, the $\overline{\text{HLDA86}}$ signal (to indicate if the cycle is a processor cycle or a DMA cycle), and the SIZE outputs from the processor and decodes them to four independent Byte Enable outputs (BE0–BE3). These are used to enable the appropriate data lines to the data bus. If a full 32-bit word is being fetched by the CPU, all four will be active; if only 8 bits, then only one will be active.

One incompatibility still exists, however. If a WORD WRITE is performed by an IBM PC XT into the DMA memory, the least significant and most significant bytes will be swapped. Consequently, you must be careful to use byte data transfers or remember to reverse the order of the bytes ob-

(continued)

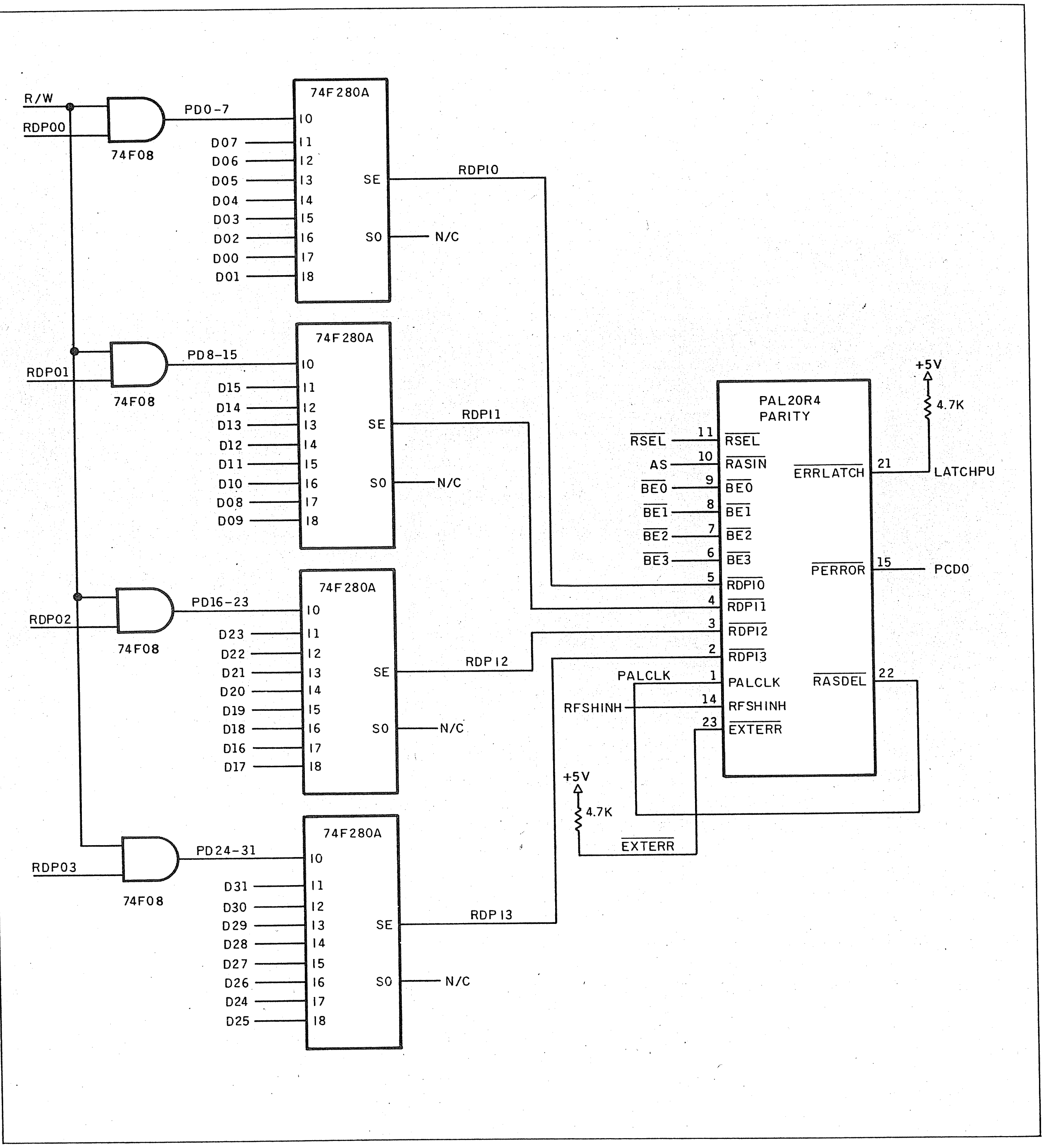


Figure 7: *Parity generation and detection circuits.*

Inquiry 382 for End-Users.
Inquiry 383 for DEALERS ONLY. →

Table 1: DSI-020's *memory-addressing technique.*

The DSI-020's memory is all available to the host 8088/286. It appears as a 64K-byte "window" that can be steered through the 16-megabyte total address space by latching a "segment select" value at the segment port.

There are two distinct address spaces at which DSI-020 can be operated:

HDR10 position	Memory Base	Segment Port	BFLAGS Port
Ground	E0000 Hex	160 Hex	150 Hex
V_{cc}	D0000 Hex	2B0 Hex	2A0 Hex

Use the ground position for XT clones only; the V_{cc} position can be used with either XT or AT clones.

The BFLAGS control port is used for PC bus I/O control.
When READ by the PC:
Bit 0, if low, indicates that a parity error has occurred.
Bit 1 is a copy of the FLAG1 output of the DUART.
When WRITTEN by the PC (all outputs to the 68020 are latched)
Bit 0, when high, asserts $\overline{\text{HALT}}$ to the 68020.
Bit 1, when high, asserts $\overline{\text{INT020}}$ to the 68020.
Bit 2, when high, resets the 68020.
Bit 3 is FLAG0, which may be read by the VEC20 PAL.

A number of status bits can be read from the VEC20 PAL:

Bit 0	is used by diagnostics to check the state of the Refresh Inhibit signal.
Bit 1	shows the state of the RS-232C RIB signal.
Bit 2	reads the data from the COPS EEPROM serial number chip.
Bit 3	shows the state of the RS-232C DCDB signal.
Bit 4	allows the 68020 to read the FLAG0 signal from the 8086.

PC Scheme–
a simple, modern LISP
for under $100.

Texas Instruments presents PC Scheme, the $95* solution to your symbolic processing needs.

Whether you're an experienced LISP user, or just beginning to discover the power of symbolic programming languages, PC Scheme is the right product for you. It runs on IBM® Personal Computers, as well as the TI Professional Computer family, including the Business-Pro™ computer.

Powerful features include an optimizing incremental compiler for ease of programming and fast execution; an EMACS-like editor; extensions for debugging, graphics, and windowing; DOS-CALL capability; and a programming system for the development of object-oriented applications—all designed to work efficiently on personal computers.

To order, or for more information, call toll-free:

1-800-527-3500

*Suggested list price.
Business-Pro is a trademark of Texas Instruments Incorporated.
IBM is a registered trademark of International Business Machines Corporation.

TEXAS INSTRUMENTS

Creating useful products
and services for you.

©1986 TI 261765-02

The 68020 *coprocessor runs* UNIX *software tools without a* UNIX *kernel.*

tained by a word transfer when writing PC interface software for the DSI-020.

THE BIDIRECTIONAL LATCHES

The 68020 instruction cycles are very much faster than those of the 8088/286. This means that at the end of a bus cycle, after the 68020 has been released from the DMA task, the PC bus interface still requires the DMA data to be valid. Thus, on a PC bus READ cycle, the DMA data byte has to be latched while RAM data is valid so that it can be made available to the host asynchronously. This is the function of the 74HCT646 latches (see figure 9).

The $\overline{\text{RASIN}}$ (strobe) signal is activiated by the DPORT20 PAL at the end of the DMA cycle to latch the data. The RAM array is further stabilized by filter capacitors on the board (figure 10). When the PC writes to the DSI-020, the latches remain transparent for the total cycle, however.

THE OPERATING SYSTEM

The basic operating system for programming or operating the DSI-020 is MS-DOS (or PC-DOS). Definicon has implemented an interface between the 8088/80286 and the 68020 that allows the 68020 to run UNIX-quality development tools without requiring either a UNIX kernel or a UNIX file system. The system calls are accessed via the 68020 TRAP #14 instruction. The text box "DSI-020 Kernel Functions" on page 142 provides a list of

(continued)

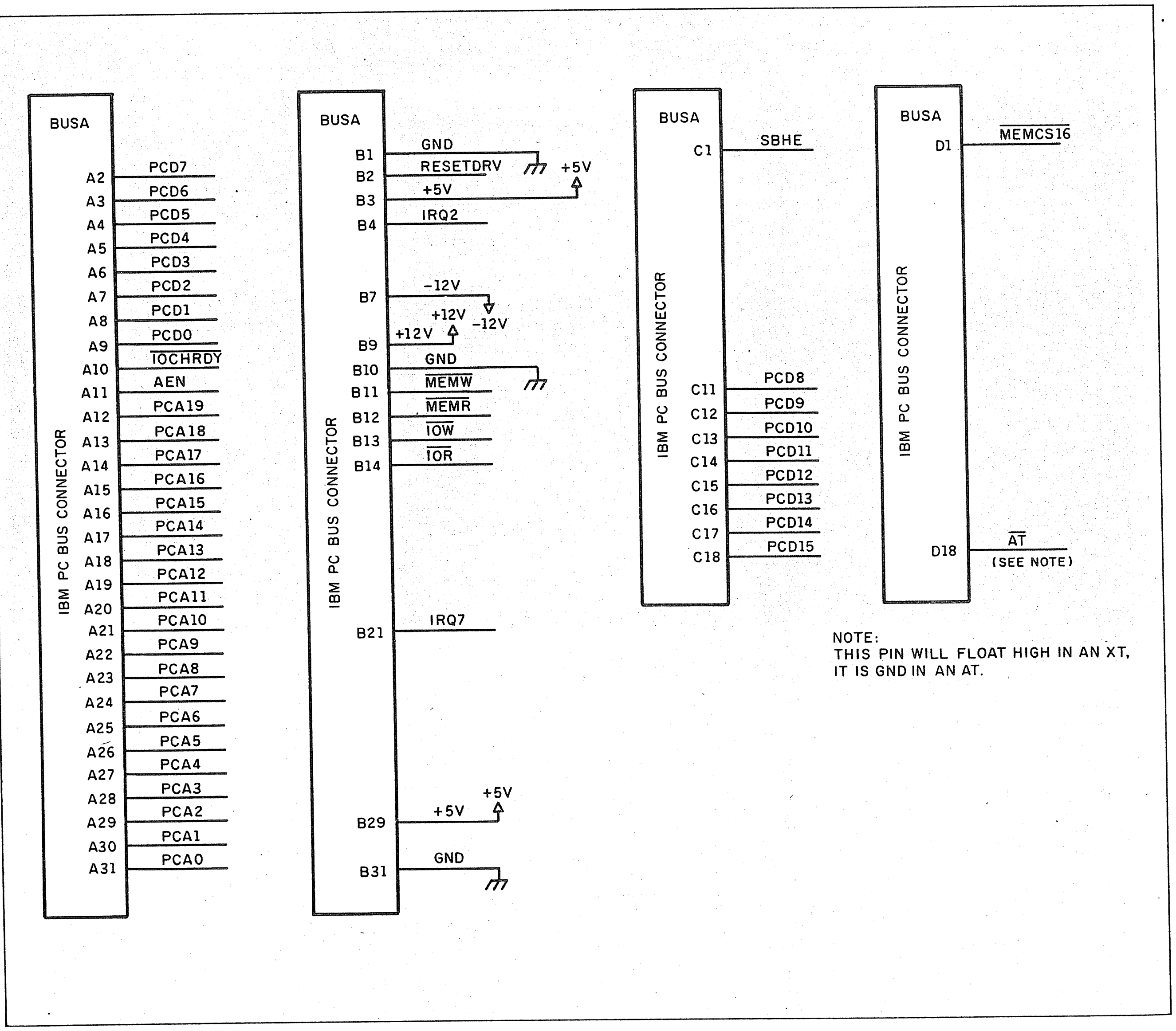


Figure 8: PC *bus connections.*

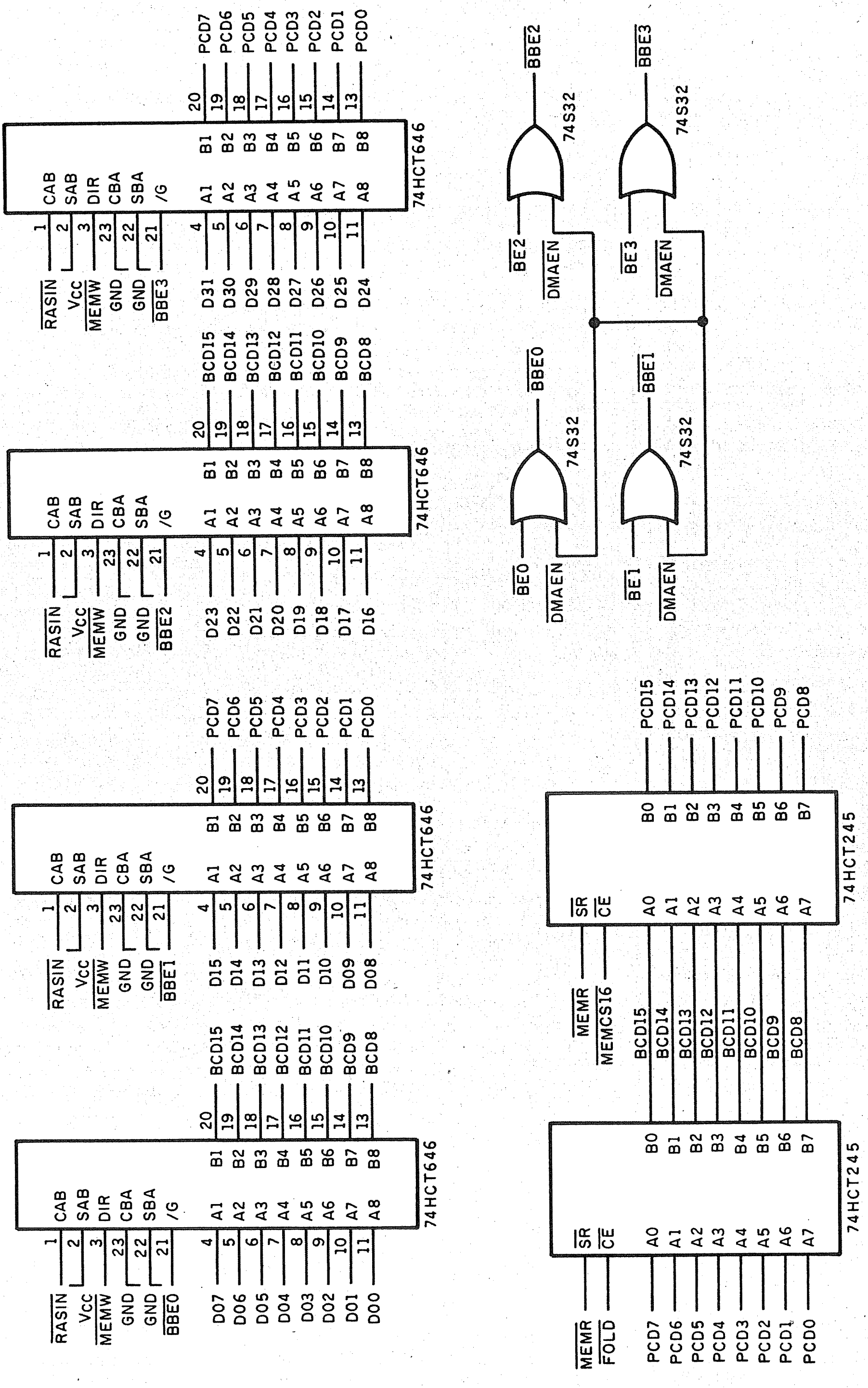


Figure 9: PC *bus interface latches.*

Inquiry 230

MetaComCo™ Languages

Atari ST™ • Amiga™ • QL™

LATTICE® C
The well known LATTICE® C compiler, full featured portable implementation.

MCC™ PASCAL
An ISO/ANSI standard Pascal. A fast single pass compiler, ideal for commercial, educational and personal use.

MCC™ ASSEMBLER
A professional standard macro assembler with many powerful features.

CAMBRIDGE LISP
LISP interpreter/compiler providing a complete LISP development environment.

BCPL
A powerful compiler offering the convenience of a high level language with the flexibility of an assembler.

APL
Keyword and symbolic versions of this important language from MicroAPL.

MCC™ BASIC
68000 BASIC interpreters and compilers from MetaComCo.

MENU+™
An easy to use program environment with pull-down menus. Menu+ comes free with Atari software. Upgrades ready for registered users.

TOOLKIT
Provides useful tools and utilities. From the people who wrote AmigaDOS.

Product	Atari™ ST	Commodore™ Amiga	Sinclair™ QL
Lattice C	$149.95	$149.95	$129.95
Pascal	$99.95	$99.95	$99.95
Assembler	$79.95	*	$49.95
Lisp	TBA	$199.95	$79.95
BCPL	TBA	TBA	$79.95
APL	$199.95	TBA	$129.95
Basic	†	*	—
MENU+	$29.95	TBA	—
TOOLKIT	TBA	$49.95	—

* = available from Commodore
† = available from Atari
Cambridge LISP is available for CP/M-68K™
QL Lisp includes interpreter only

METACOMCO™

The Quality Source for 68000 Software
5353 #E Scotts Valley Dr., Scotts Valley, CA 95066

Contact your local dealer or call:

Tel: US 800-AKA-META

Cal 800-GET-META, Eur (UK) 44-272-428781

Telex (UK) 444874 METACO G

Add 6.5% tax if CA resident.

Lattice is a registered trademark of Lattice Inc.
Amiga and AmigaDOS is a trademark of Commodore-Amiga Inc.
Atari ST is a registered trademark of Atari Inc.
CP/M-68K is a trademark of Digital Research Inc.

the available system calls. Note that functions 5 through 15 emulate the basic UNIX I/O system. Thus, STDIN (the keyboard) is treated as a FILE with a handle of 0, while STDOUT (the CRT) is addressed as a FILE with a handle of 1. These functions cannot interfere with the operation of the host PC, and it is recommended that you use them in preference to the enhanced interface calls (1,16–25). These enhanced calls, if invoked with faulty syntax, are sufficiently powerful to interfere with the operation of the MS-DOS system itself.

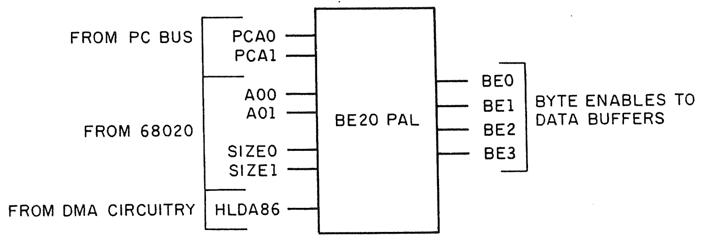
Note that any peripheral available to the host CPU is also available to the 68020. Bit-mapped graphics adapters are programmed directly using functions 16 through 19, networking is implicitly supported by the DOS file system, and specialized peripherals, such as A/D converters, can be accessed

(continued)

Table 2: *Comparison of Motorola and Intel memory arrangement and the hardware logic interface required for translating the data.*

	ADDRESS 0 LSB . . MSB	ADDRESS 1 LSB . . MSB	ADDRESS 2 LSB . . MSB	ADDRESS 3 LSB . . MSB
INTEL DATA BITS	D0 . . . . . D7	D8 . . . . D15	D16 . . . D23	D24 . . . D31
MOTOROLA DATA BITS	D24 . . . D31	D16 . . . D23	D8 . . . . D15	D0 . . . . . D7
BYTE ENABLE	BE3	BE2	BE1	BE0

	ADDRESS 0	ADDRESS 1	ADDRESS 2	ADDRESS 3
	LSB.....MSB	LSB.....MSB	LSB.....MSB	LSB.....MSB
INTEL DATA BITS	D0..........D7	D8........D15	D16......D23	D24......D31
MOTOROLA DATA BITS	D24......D31	D16......D23	D8.........D15	D0.........D7
BYTE ENABLE	BE3	BE2	BE1	BE0

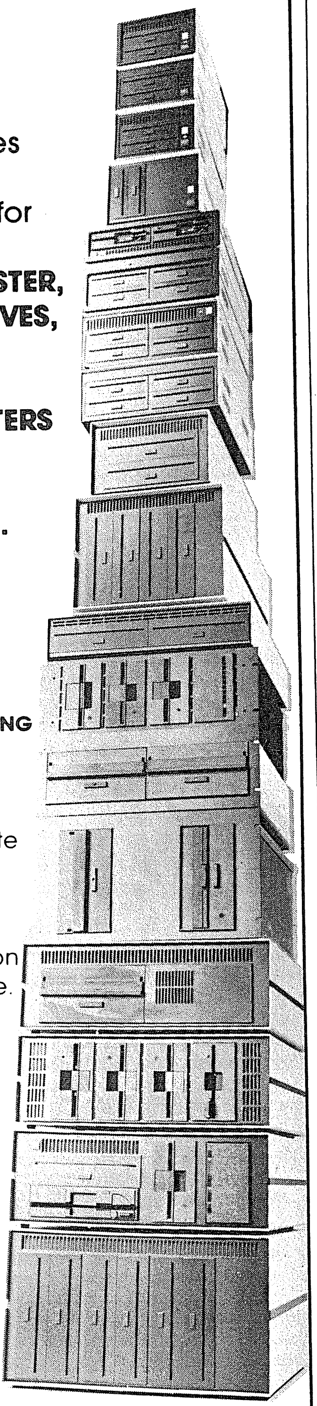


PAL EQUATIONS:

$$\begin{array}{lll}
\text{BE0} = & \text{HLDA86} * \text{PCA0} * \text{PCA1} & \text{:This is the PC bus decode term} \\
+ & \overline{\text{HLDA86}} * \text{A00} * \text{SIZE0} * \text{SIZE1} & \text{:These next four implement the decode} \\
+ & \overline{\text{HLDA86}} * \overline{\text{SIZE0}} * \overline{\text{SIZE1}} & \text{:schematic from page 5-17 of the} \\
+ & \overline{\text{HLDA86}} * \text{A00} * \text{A01} & \text{:}\textit{Motorola MC68020 User's Manual}. \\
+ & \overline{\text{HLDA86}} * \text{A01} * \text{SIZE1} & \\
 & & \\
\text{BE1} = & \text{HLDA86} * \text{PCA1} * \overline{\text{PCA0}} & \text{:The remaining equations are} \\
+ & \overline{\text{HLDA86}} * \overline{\text{A00}} * \text{A01} & \text{:constructed in a similar fashion.} \\
+ & \overline{\text{HLDA86}} * \overline{\text{A01}} * \overline{\text{SIZE0}} * \overline{\text{SIZE1}} & \\
+ & \overline{\text{HLDA86}} * \overline{\text{A01}} * \text{SIZE0} * \overline{\text{SIZE1}} & \\
+ & \overline{\text{HLDA86}} * \overline{\text{SIZE0}} * \text{A00} * \overline{\text{A01}} & \\
 & & \\
\text{BE2} = & \text{HLDA86} * \text{PCA0} * \overline{\text{PCA1}} & \\
+ & \overline{\text{HLDA86}} * \overline{\text{SIZE0}} * \overline{\text{A01}} & \\
+ & \overline{\text{HLDA86}} * \overline{\text{A01}} * \text{A00} & \\
+ & \overline{\text{HLDA86}} * \text{SIZE1} * \overline{\text{A01}} & \\
 & & \\
\text{BE3} = & \text{HLDA86} * \overline{\text{PCA0}} * \overline{\text{PCA1}} & \\
+ & \overline{\text{HLDA86}} * \overline{\text{A00}} * \overline{\text{A01}} &
\end{array}$$

Inquiry 180

DRIVE INN



Enclosures & power supplies for **FLOPPY, WINCHESTER, TAPE DRIVES, SINGLE BOARD COMPUTERS & S-100 SYSTEMS.**

8 inch
5 inch
3 inch

CUSTOMIZING AVAILABLE

Call or write for free catalogs and application assistance.

INTEGRAND
RESEARCH CORPORATION
8620 Roosevelt Ave. • Visalia, CA 93291
209/651-1203

TELEX 5106012830 (INTEGRAND UD)
EZLINK 62926572

We accept BankAmericard/VISA and MasterCard

using the I/O facilities (functions 20 and 21). In addition, any MS-DOS program that installs itself as an 8088/80286 interrupt process can be accessed via function 25. Most mice and many hardware drivers can be accessed in this way. COM1:, COM2:, and PRN: are accessed by opening files called COM1, COM2, or PRN.

Next month the DSI-020 software support will be discussed in detail.

PERFORMANCE

While almost every reader took the trouble to look at the benchmarks published in the DSI-32 article, apparently only a handful actually believed we had not specifically made them up to favor the 32032! Consequently, the benchmark used to characterize the DSI-020 performance was not written by Definicon. This program is generally regarded as an "industry-standard" benchmark. This has the added advantage that you will be able to find a plethora of data tabulating the relative performance of other computers. Also, a special 16.7-MHz version of the DSI-020 was assem-

(continued)

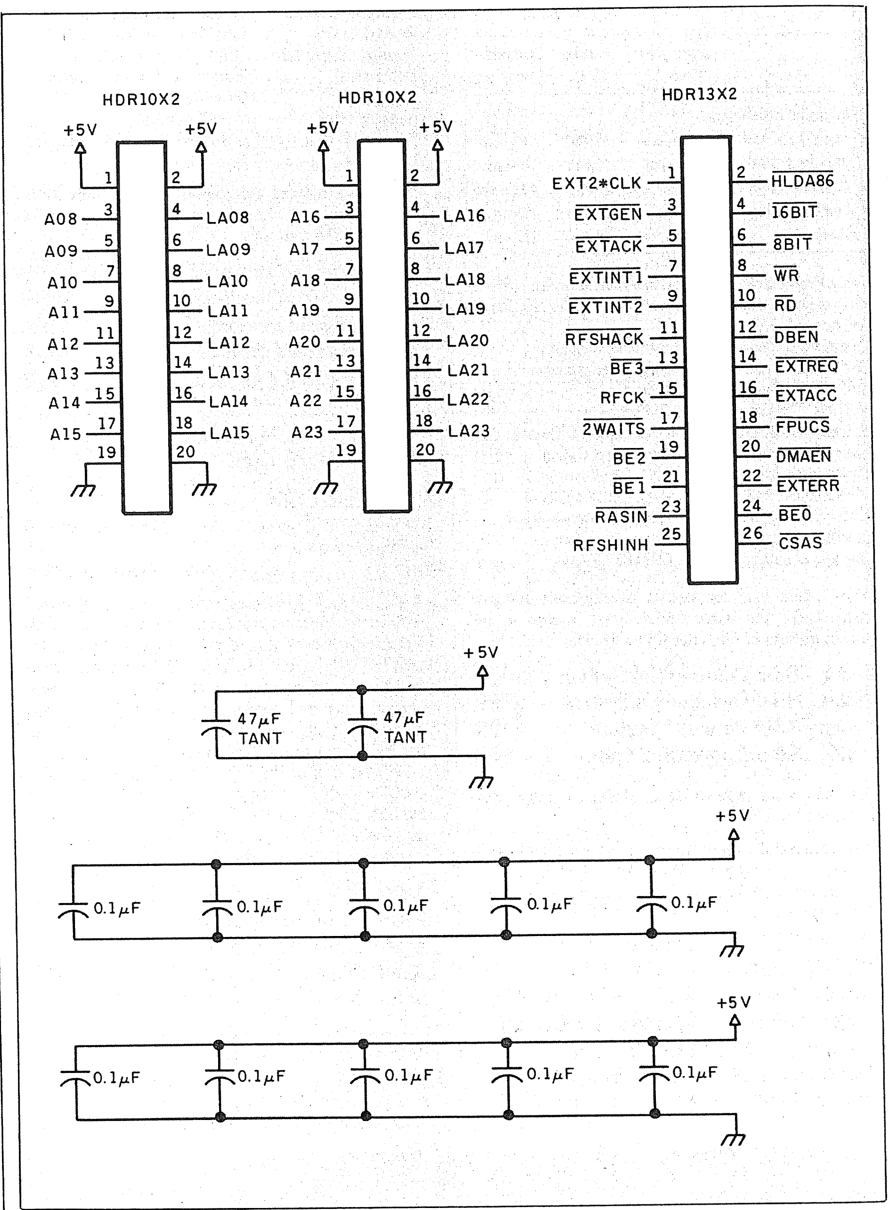


Figure 10: *Headers and filter capacitors for the* DSI-020.

DSI-020 Kernel Functions

Register D0 is always used to communicate the function code to the kernel and will not be listed in the "entry parameters."

A suffix of .L, .W, or .B after a register shows the data size expected by the kernel.

The "no error" status code returned in D0 is 0000.

Address registers are always passed as 32-bit values.

Function 1—BDOS Call

Used to communicate an MS-DOS system request (INT 21H) to the host. This function should be used only by experienced programmers.

Entry parameters:	A0	DX argument
	D1.B	BDOS function code (AH argument)
Return parameter:	D0.B	AL return code, if any, else undefined

This function was implemented for system software development. BDOS functions that allow multibyte I/O (such as string in/out and file I/O operations) will not work properly if crossing 64K page boundaries.

Function 5—Open File

Entry parameters:	A0	pointer to ASCIIZ filename with optional MS-DOS path specification
	D1.W	open mode, one of: 0 = read file 1 = write file 2 = read and write
Return parameters:	D0.W	file ID (handle), or FFFF = error and
	D1.W	error code

Function 6—Close File

Entry parameter:	D1.W	file handle
Return parameters:	D0.W	status. If error,
	D1.W	error code

Function 7—Create File

Entry parameter:	A0	pointer to ASCIIZ filename
Return parameters:	D0.W	file handle, or FFFF = error and
	D1.W	error code

Function 8—Read File

Entry parameters:	D1.W	file handle
	A0	disk transfer address (DTA)
	D2.L	number of bytes to read
Return parameters:	D0.L	actual number of bytes read, or 0000 = error and
	D1.W	error code

Function 9—Write File

Entry parameters:	D1.W	file handle
	A0	DTA
	D2.L	number of bytes to write
Return parameters:	D0.L	actual number of bytes written, or 0000 = error and
	D1.L	error code

Function 10—Delete File (Unlink)

Entry parameter:	A0	pointer to ASCIIZ filename
Return parameters:	D0.W	status (0000 = file deleted) or
	D1.W	error code

Function 11—Rename File

Entry parameters:	A0	pointer to old ASCIIZ filename
	A1	pointer to new ASCIIZ filename
Return parameters:	D0.W	status (0000 = file renamed) or
	D1.W	error code

Function 12—Seek to Byte in File

Entry parameters:	D1.W	file handle
	D2.L	byte to seek to
	D3.W	base, one of: 0 = from beginning of file 1 = from current location of FP 2 = from EOF
Return parameters:	D0.L	current position, or FFFFFFFF = error and
	D1.W	error code

Function 13—Determine Current Byte Offset

Entry parameter:	D1.W	file handle
Return parameters:	D0.L	current posistion from BOF, or FFFFFFFF = error and
	D1.W	error code

Function 14—Get Command Line Arguments

Entry parameter:	none	
Return parameters:	D0.W	number of commands
	A0.L	pointer to array of pointers to arguments

Function 15—Terminate

Entry parameter:	none
Return parameter:	none (does not return)

Function 16—Move Memory Block from 8086

Entry parameters:	A0	source address in 8086 space (20 significant bits)

	A1	destination address in 68020 space
	D1.L	word count
Return parameter:	none (memory is moved)	

FUNCTION 17—MOVE MEMORY BLOCK TO 8086

Entry parameters:	A0	source address in 68020 space
	A1	destination address in 8086 space (20 significant bits)
	D1.L	word count
Return parameter:	none (memory is moved)	

FUNCTION 18—MOVE MEMORY FROM 8086 BACKGROUND)

Same as function 16, except that function call does not wait for completion before returning.

FUNCTION 19—MOVE MEMORY TO 8086 (BACKGROUND)

Same as function 17, except that function call does not wait for completion before returning.

FUNCTION 20—READ 8086 PORT

Entry parameter:	D1.W	8086 port address
Return parameter:	D0.B	value read from port

FUNCTION 21—WRITE 8086 PORT

Entry parameters:	D1.W	8086 port address
	D2.B	byte to be output
Return parameter:	none	

FUNCTION 23—ALLOCATE 8086 MEMORY

Entry parameter:	D1.L	number of bytes to allocate
Return value:	D0.L	pointer:
		high word = DS
		low word = OFFSET
		00000000 if no memory available

FUNCTION 24—DEALLOCATE 8086 MEMORY

Entry parameter:	D1.L	pointer (DS:OFFSET) to RAM
Return value:	none	

FUNCTION 25—EXECUTE 8086 INTERRUPT

Entry parameters:	D1.B	interrupt vector number
	A0	pointer to register structure Sequence: DX, CX, BX, AX
Return value:	Register structure filled with result	

FUNCTION 27—GET SYSTEM DATE

Entry parameter:	none		
Return value:	D0.L	bits:	
		31..16	year (1980 base)
		15..08	month
		07..00	day

FUNCTION 28—GET SYSTEM TIME

Entry parameter:	none		
Return value:	D0.L	bits:	
		31..24	hour
		23..16	minute
		15..08	second
		07..00	1/100

FUNCTION 29—CHAIN TO PROGRAM (EXEC( ))

Entry parameters:	A0	pointer to full command line
	A1	desired load address
Return value:	none	— new program is loaded and executed if it is a valid .E20 file

FUNCTION 30—GET/SET RS-232C CONTROLS

This function is used to control the on-board DUART.

Entry parameters:	D1.B	00=get controls, FF=set controls
	D2.B	00=set stop bits
		01=set parity
		02=set character length
		03=set baud rate
	D3.L	set value and port
		bits 23..16 = logical port number
		bits 07..00 = value
Return value:	D0.L	controls if get, error if set
		bits 19..18 = number of stop bits
		bits 17..16 = parity (0=N, 1=EVEN, 2=ODD)
		bits 15..08 = character length
		bits 07..00 = baud rate

FUNCTION 31—INPUT CHARACTER

This function inputs a byte from the local DUART.

Entry parameter:	D1.B	logical port number
Return value:	D0.W	status word:
		bit 15 set if no character available
		bits 8..0 contain character if bit 15=0

FUNCTION 32—OUTPUT CHARACTER

This function outputs a character to the local DUART.

Entry parameter:	D1.B	character to be output
Return value:	none (character is output, waiting if necessary)	

FUNCTION 33—GET BOARD SERIAL NUMBER

Entry parameter:	none	
Return value:	D0.L	board serial number

All functions listed are subject to change due to improvements or bug fixes at any time without notice. Type the file IO-READ.ME on your distribution disk for the latest definitions.

Table 3: DSI-020 *Whetstone benchmarks.*

1. Single-Precision Whetstones
Performance is quoted as the number of Whetstones per second

DSI-020 12.5 MHz	DSI-020 16.7 MHz	IBM PC RT	VAX-11/780
770K	1028K	200K	1152K

2. Double-Precision Whetstones
Performance is quoted as the number of Whetstones per second

DSI-020 12.5 MHz	DSI-020 16.7 MHz	IBM PC RT	VAX-11/780
676K	902K	185K	798K

Note: All machines have floating-point hardware or coprocessors.

bled to show the peak performance currently available using the 68020 microcomputer family. Although the chip set made available by Motorola is a 12.5-MHz version, it is our experience that the careful hacker will note the additional crystal oscillator on the schematic and hone his system's performance to the limit. One tip: You will need to use 100-nanosecond RAM and carefully watch the FPU's operation at these faster speeds, however.

The Whetstone benchmark first appeared in ALGOL in *Computer Journal* (February 1976, pages 47–49). This version was translated to FORTRAN by D. Frank in 1979. Both single- and double-precision results are given. The Whetstone tests mathematical functions such as LOG and EXP in addition to linear arithmetic. Table 3 shows the results obtained with the DSI-020 at 12.5 MHz and 16.7 MHz, the IBM PC RT, and a VAX-11/780.

You can draw your own conclusion from the data presented in table 3, but one thing is clear: The days when a desktop microcomputer could be regarded as a development tool with limited capability are gone. The question that must be asked now is "how much further can technology go?" A VAX-11/780 dissipates about 6 kilowatts of electricity and requires a small army of technicians to service it. The DSI-020 dissipates 10 watts, and its peripherals and operating procedures (MS-DOS) can be comprehended and serviced by most all the population. Time alone will tell. ■

Eighth Annual Conference and Exposition

INTERACTIVE VIDEODISC

In Education and Training

August 20-22, 1986
J. W. Marriott Hotel
Washington, D. C.

Presentations Cover:

Current Applications
Emerging Technology
Methodology Implementation
Learner Productivity Improvement

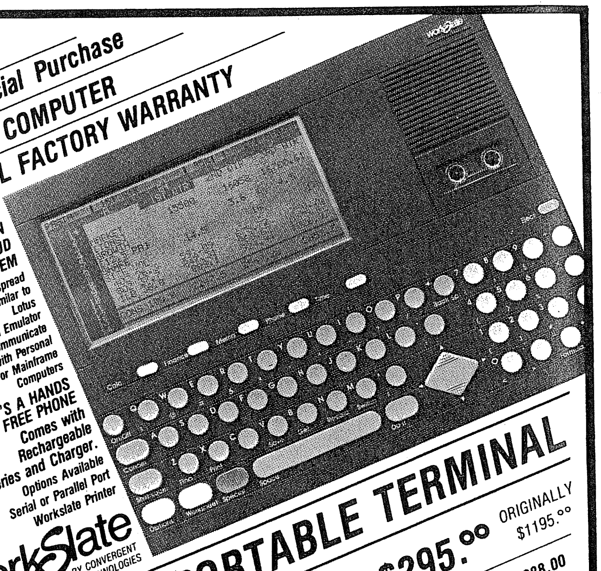
Pre-conference tutorials are scheduled for August 18 and 19.

Included among exhibitors:

3M	New Media Graphics
DEC	Online Computer Systems
NCR	MetaMedia Systems
WICAT	Applied Science Associates
Pioneer	GWF Associates
Sony	Creativision
U.S. Video	LaserVideo

For further information contact::
SALT **Society for Applied Learning Technology**
50 Culpeper Street, Dept. B
Warrenton, Virginia 22186
703-347-0055
Outside Virginia call toll free: 1-800-457-6812

Special Purchase
LAP COMPUTER
FULL FACTORY WARRANTY



BUILT IN 300 BAUD MODEM
Built in spread sheet, similar to Lotus
Terminal Emulator to Communicate with Personal or Mainframe Computers
IT'S A HANDS FREE PHONE
Comes with Rechargeable Batteries and Charger.
Options Available Serial or Parallel Port Workslate Printer

workSlate BY CONVERGENT TECHNOLOGIES

THE MOST PORTABLE TERMINAL

OUR BEST PRICE EVER **$295.°°** ORIGINALLY $1195.°°

This state of the art computer comes with two instructional cassettes that talk you through the learning process. This means a human voice tells you which keys to press as you are learning to manipulate information on the screen. This computer also has built-in hardware that lets you hook in to services like Dow Jones/News Retrival, The Source, MCI Mail, and many other services, including thousands of electronic bulletin boards. • **Dimensions:** 8½"x11¼"x 1", 3 lbs. **Display:** LCD, 46 characters per line, 16 lines. **Keyboard:** 60 keys, Typewriter-style keyboard and calculator pad. **Power:** NiCad-rechargeable battery, AC adaptor/rechanger or 4AA alkaline batteries. Software: Built-in handles worksheets for Financial analysis, phone, calendar and communications management. Internal Memory up to 12 8½" x 11" pages Microcasettes store up to 10 worksheets or 45 minutes of audio information.

INCLUDES FREE SOFTWARE WORTH $88.00
Information and Consultant Services Software.

OTHER SOFTWARE PACKAGES • WITH PURCHASE ONLY

Cash Management	$15.00	Insurance Analyzer	$15.00
Financial Statements	(2) $15.00	Real Estate	(2) $15.00
Inventory Analysis	$15.00	Electronic Mail	$25 00
Marketing Management	$15.00	Information Services	$25.00
Sales Reporter	$15 00	Loan Analysis	$15 00
Travel	$15.00	Portfolio Analysis	(2) $15 00

15 Day Money Back Guarantee

CEC 1206 S. Amphlett Blvd. #3
San Mateo, CA 94402
415/342-4058
800/228-3411

Inquiry 312 Inquiry 101

THE DEFINICON 68020 COPROCESSOR

Software to give you 32-bit capabilities in a 16-bit machine

Editor's note: This is part 2 of a two-part article describing Definicon Systems Inc.'s DSI-020 coprocessor board for IBM XTs, ATs, *and work-alikes. Part* 1 (July BYTE) *described the hardware and operating system kernel. Part* 2 *will focus on the software available for the board. For full information on the Motorola* 68020 *CPU chip, see* "The MC68020 32-*bit Microprocessor" by Paul F. Groepler and James Kennedy* (*November* 1984 BYTE).

Last month we presented the schematics and hardware theory of operation for the DSI-020 coprocessor. Hardware, however, doesn't function too well without software tools; so this month, after looking at the remaining hardware topics, we'll discuss the available software support.

EFFECTIVENESS OF THE 68020'S INSTRUCTION CACHE

A jumper has been provided so that the internal instruction cache memory of the 68020 can be manually disabled. Running a program with and without this jumper quickly demonstrates how effective an instruction cache can be.

Listing 1: *Memory test program.*

```
         ORG      $00004000
;
;set to 4030 if test on longword boundary
;
TSTORG   EQU      $00004032
;
START:   MOVE.L   #RWADDR,A0
         MOVE.L   #$5555AAAA,D0
BEGIN:   CLR.L    D1
LOOP:    MOVE.L   D0,(A0)   ;data memory access
         CMP.L    (A0),D0   ;data memory access
         BNE.L    ERROR
         SUBQ     #1,D1
         BNE.L    LOOP
         BRA.L    BEGIN
;
ERROR:   STOP     #0
;
         ORG      TSTORG
;
RWADDR:  DC.L     0
         END
```

On short programs, such as the memory test in listing 1, the speed advantage with the cache is about 40 percent. The reason for this enhancement becomes evident if the scope photograph of the bus activity with the cache on (photo 1) is compared to that with the cache off (photo 2). Both photos were taken of the 68020 $\overline{AS}$ address strobe with a time base of 1 microsecond per centimeter. A high level on the $\overline{AS}$ signal indicates no bus activity. If the $\overline{AS}$ signal is low for 75 percent of the total time, the bus is fully utilized. Note that bus access occurs only during instructions that actually reference data memory when the cache is on, while the bus has to also supply all the program code fetches when the cache is off.

By comparison, on floating-point-intensive programs, such as the Whetstone benchmark, the difference is only a few percent. When the number of CPU wait states is increased (for slow memory) the speed enhancement due to the cache becomes even greater. If you wish to investigate this phenomenon further, pull the $\overline{2WAITS}$ input of the DSACK20 PAL low to simulate the effect of having slower memory on the system. The program execution in photo 1 was optimized in that the data fetches are aligned on a long-word boundary, while in photo 3 data fetches are not aligned on a long-word boundary, and thus two 16-bit fetches are required for each 32-bit access. In photo 2 (cache off) you can clearly see the execution of each instruction. Note that the two

BY TREVOR MARSHALL, CHRISTOPHER JONES, AND SIGI KLUGER

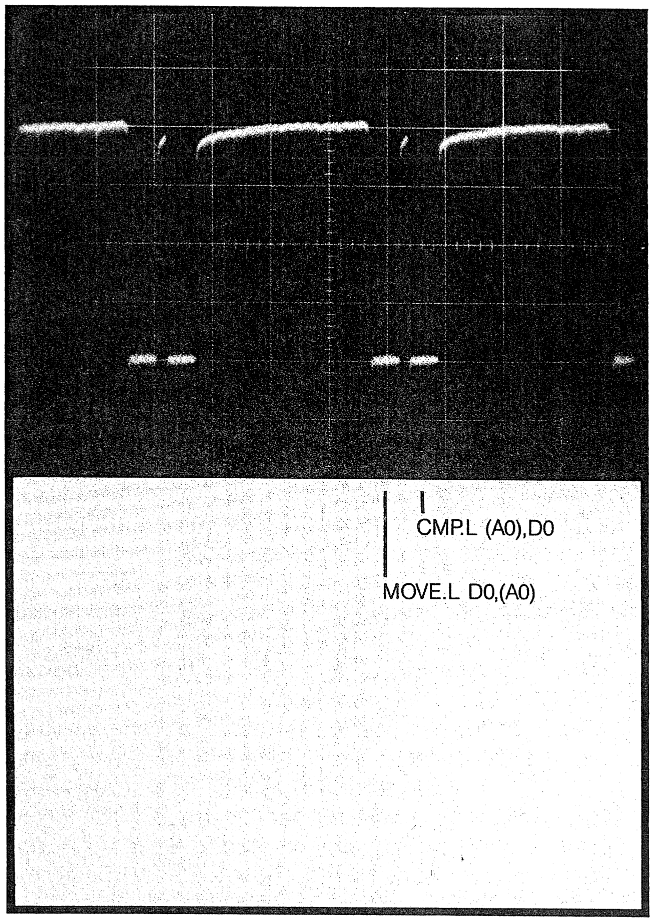


Photo 1: *The two fetches referencing data locations are displayed. All op-code fetching is from the cache.*

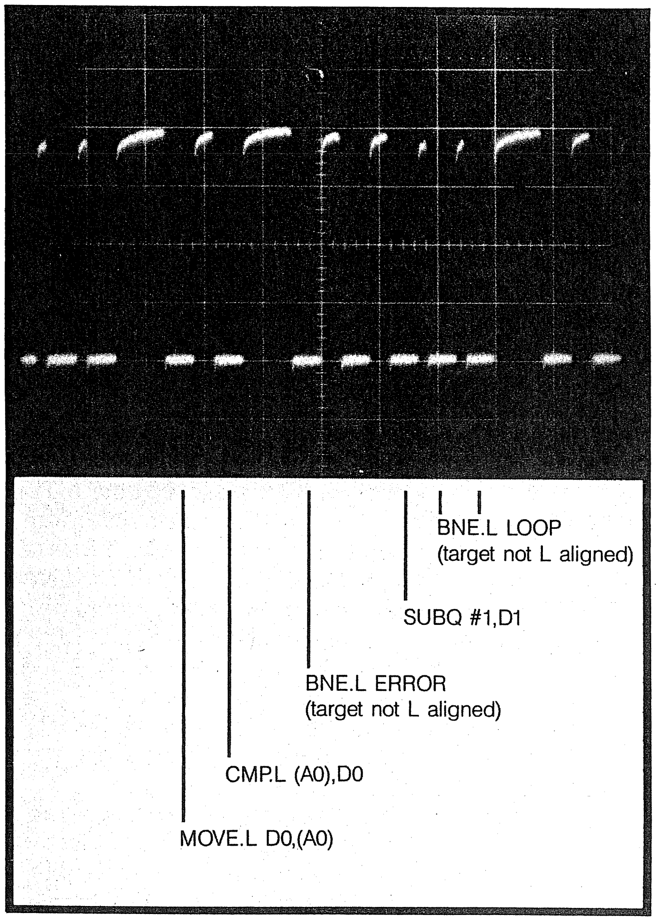


Photo 2: *With the cache disabled, all data fetches are seen on the scope. Note that two fetches occur for each of the conditional branches to non-long-word-aligned addresses.*

conditional branches reference non-long-word-aligned targets and require two cycles.

THE EEPROM CONTAINING THE DSI-020 SERIAL NUMBER

To get some of the larger software houses to port their products to the DSI-020, we had to provide a means to ensure that the software was indeed running on the board for which it was sold. The on-board 9306 COPS EEPROM can be interrogated by the kernel using function 33. It will return the serial number as a 32-bit integer. The most significant byte of the serial

(continued)

Trevor Marshall, Christopher Jones, and Sigi Kluger are engineers with Definicon Systems Inc. They can be contacted at 31324 Via Colinas #108/9, Westlake Village, CA 91362.

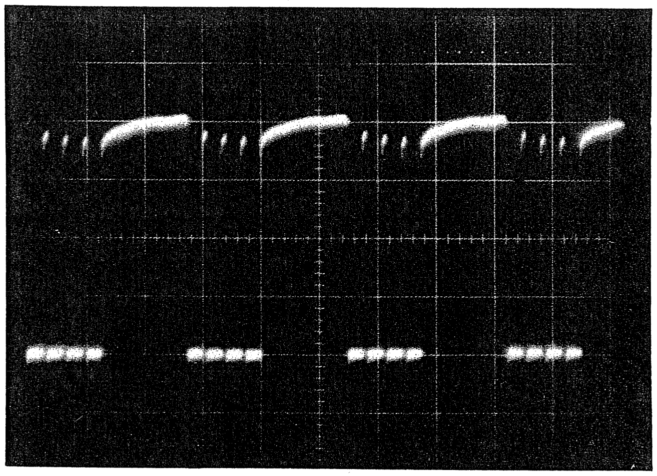


Photo 3: *This photo shows two memory fetches for each of the data locations due to the data not being aligned on a long-word boundary. Instead of fetching 32 bits of data, two 16-bit transfers are executed.*

The kernel software, a simple assembler, a linker, and a debugger are bundled with the DSI-020 *hardware.*

number carries information about the hardware revision level.

Some bytes of this EEPROM are available for storing software parameters (such as passwords).

OPERATING SYSTEM

The basic operating system for programming or operating the DSI-020 is MS-DOS, and all communication between the user program and the MS-DOS host occurs via the 68020 TRAP #14 instruction. Parameters are transferred in registers. Data registers are used for values, address registers for pointers. All functions save only the registers they use, but data registers used to pass values may return with their contents altered. Abnormal conditions, such as trying to execute an invalid instruction, division by zero, attempts to use privileged instructions, or undefined function calls, will cause an error message on the host's console and immediate termination to MS-DOS or the debugger.

On entry to a program the kernel sets up the stack at the top of RAM and preloads the following registers:

A0.L	program entry point (4000 hexadecimal)
A1.L	start of heap
A2.L	top of heap/bottom of stack
4(A7)	ARGV pointer
8(A7)	ARGC

ARGC is the count of command-line arguments that were used when the program was invoked. ARGV points to a list of ARGC pointers that address the command-line arguments, saved as ASCIIZ strings (zero-terminated ASCII strings).

For HEX or IMAGE programs the entry point must be 4000 (hexadecimal). The remainder of your program can be located anywhere in the user program space. Files in DSI-LINK format (.E20) may be located at any address above 3FFF.

You can stop the program either by executing a TERMINATE service request (function 15) or by returning to the kernel using the RTS instruction. The TERMINATE service request always returns 00, while RTS supplies a value in D0.B to MS-DOS as return code. Since the return code may be tested in a conditional MS-DOS batch stream, the RTS method is preferred, provided you can maintain the stack integrity.

Your program may set up its own stack pointer, which can be located anywhere within the user program space. Alternately, you can use the system-provided default stack space allocated to start at the end of RAM and grow downward.

The interface kernel uses only 16K bytes of the 68020's address space, allowing your application to use almost the full megabyte of available RAM. Table 1 shows the memory map utilization. Note that the 68020 uses the first 400 hexadecimal bytes for its trap vector locations, and the rest of the address space to 4000 hexadecimal is used for disk buffers and the kernel code itself. After a programmed or power-on reset, the 68020 picks up its initial stack pointer from address 0 and its initial program counter value from address 4. This value points to the 2010 kernel code, which loads your program from the disk, relocates it to 4000, places the stack at the top of RAM, points HEAPLOW at the top of the program code, and begins execution.

The kernel software, a simplified assembler, a linker, and an assembly-level debugger are bundled with the DSI-020 coprocessor hardware. The simple assembler supports all the 68000 op codes, plus the 68881 floating-point instructions. However, many of the newer 68020 addressing modes are not supported. In addition, the syntax of those newer addressing modes that are supported follow the 68000, rather than the 68020, conventions. The debugger provides facilities similar to those of DOS's Debug or CP/M's DDT within the 68020 environment.

The kernel can load and execute disk files that have been created using Motorola S record format, absolute image format, or the DSI-020 linker. Programs created in S record or image format must load and begin execution at 04000 hexadecimal.

PROGRAMMING THE 68020 KERNEL IN ASSEMBLY LANGUAGE

Listing 2 contains a short program (in simplified 68020 assembly language) that takes an MS-DOS ASCII file of any length and types it on the console. Note that the program uses the (preferred) UNIX-like I/O functions 5, 8, and 9 to access the DOS file system. On entry from the kernel the stack is already initialized and pointers to the command-line parameters have been placed on it. The filename you give is opened for reading using function 5. The kernel returns a "handle" used to identify this file on subsequent accesses. A multibyte read command (80000 hexadecimal bytes) is executed. The return code, which is zero (if nothing has been read) or the actual number of bytes read, is examined. If it is zero, the program terminates. Otherwise, the number of bytes actually read are written to the

(continued)

Table 1: *Memory map of the* DSI-020. *The* DSI-020 *decodes* 16 *megabytes of* 68020 *address space.*

00000000-000003FF	reserved for 68020 reset and trap vectors
00000400-00003FFF	used by the kernel
00004000-000FFFFF	user program RAM space
	user default stack grows down from top of RAM
00F7FE03	2681 DUART base address
00F00000	addresses the VEC20 PAL
	reads EEPROM and RS-232C channel B signals

Listing 2: *Program to type an MS-DOS file on the console. Note the calls to the DSI-020 kernel for file I/O and the alternate console I/O method for the error message.*

```
;THIS PROGRAM TYPES A FILE ON THE CONSOLE USING THE DSI-020 CARD
;USE: LOAD TYPE FILENAME
;
         ORG      $00004000
;
ENTRY:   MOVE.L   8(A7),D0          ; GET ARGC PASSED BY KERNEL INIT
         MOVE.L   4(A7),A1          ; GET *ARGV
         MOVE.L   4(A1),A0          ; GET POINTER TO ARGV[1]
         CMPI.W   #2,D0             ; DO WE HAVE AN ARG?
         BEQ.B    OKAY              ; YES, GO ON
         MOVEQ    #1,D0             ; ELSE SET BDOS REQUEST
         MOVEQ    #9,D1             ; FUNCTION 9
         MOVE.L   #MESS1,A0         ; ERROR MESSAGE TO PRINT
         TRAP     #14               ; DO IT
         CLR.B    D0                ; CLEAR ERROR RETURN CODE
         RTS                        ; RETURN TO MSDOS
;
OKAY:    MOVEQ    #0,D1             ; 0 IN D1 MEANS OPEN FOR READ
         MOVEQ    #5,D0             ; FUNCTION 5 = OPEN FILE
         TRAP     #14               ; GET MSDOS TO DO IT
         CMPI.W   #$FFFF,D0         ; ERROR RETURN CODE?
         BEQ.W    NOTOPN            ; YES, GO FUSS ABOUT IT
         MOVE.W   D0,(HANDLE)       ; ELSE SAVE THE FILE'S HANDLE
LOOP:    MOVE.L   #BUFFER,A0        ; STORE BUFFER POINTER
         MOVE.W   (HANDLE),D1       ; GET FILE HANDLE AGAIN
         MOVEQ    #8,D0             ; READ FILE REQUEST
         MOVE.L   #$80000,D2        ; READ UP TO 524288 bytes
         TRAP     #14               ; DO IT
         CMPI.L   #0,D0             ; END OF FILE ENCOUNTERED?
         BNE.B    RDOK              ; IF NOT GO DISPLAY WHAT WE HAVE
         RTS                        ; ELSE JUST RETURN TO MSDOS
;
RDOK:    MOVE.L   D0,(BYTES)        ; STORE # OF BYTES READ
         MOVE.L   #BUFFER,A0        ; SET UP BUFFER PTR
         MOVEQ    #1,D1             ; WILL WRITE TO STDOUT
         MOVE.L   (BYTES),D2        ; # OF BYTES
         MOVEQ    #9,D0             ; WRITE FILE REQUEST
         TRAP     #14               ; SEND BYTES AND IGNORE RET CODE
         BRA.B    LOOP              ; SINCE STDOUT SHOULD EXIST
;
NOTOPN:  MOVEQ    #1,D0             ; TELL THE OPERATOR HE
         MOVEQ    #9,D1             ; BLEW IT
         MOVE.L   #MESS2,A0         ; AND HAD BETTER LEARN TO SPELL
         TRAP     #14
         RTS                        ; THEN LET DOS MAKE HIM TRY AGAIN
;
MESS1:   DC.W     $0D0A             ; CARRIAGE RETURN AND LINE FEED
         DC.W     'USE: LOAD TYPE FILENAME'
         DC.W     $0D0A
         DC.W     '$'               ; TERMINATES THE DOS OUTPUT STRING
;
MESS2:   DC.W     $0D0A
         DC.W     'CANNOT OPEN FILE',$0D0A,'$'
;
HANDLE:  DC.W     0                 ; USED TO HOLD THE FILE HANDLE RETURNED BY DOS
BYTES:   DC.L     0                 ; THE NUMBER OF BYTES WE ACTUALLY READ
;
BUFFER:  DC.B     0        ; BUFFER START, ENDS AT THE TOP OF RAM
         END
```

It is possible to code specialized multitasking software using the 2681 DUART timer.

console by addressing it as stdout, and the read operation is repeated. The second read attempt after reading the whole file will, of course, cause the program to terminate. Thus, you can display a file of any size up to the disk or operating system limits.

The error message handler, however, uses the alternate console I/O method. Function 1 (BDOS call) is used to send an MS-DOS interrupt 21H to the host with AH = 9. Reference to a DOS manual will show that this causes MS-DOS to output the indicated string until a $ character is reached. Although this technique is often simpler for an experienced programmer, you should not attempt it if you are unfamiliar with MS-DOS assembly language programming. One of the pitfalls of this method is that some DOS functions require a buffer to be placed in 8086 memory to receive the data (this is the function of system calls 23 and 24).

MULTITASKING

The structure of the DSI-020 kernel makes it possible, but difficult, to write specialized multitasking software using the interrupt-driven timer in the 2681 DUART. Most programming applications typically require tasks such as target program execution, editing, and printing to be concurrent. This allows you to set up an execution task, the source code, and a link map in windows on the display. Because the host MS-DOS kernel is not reentrant and the editor and typing functions are usually performed in the host environment, a multitasking host DOS will be necessary. We have found that Concurrent DOS (Digital Research, Pacific Grove, CA), which offers four concurrent tasks windowed onto the display, comes closest to the ideal environment. Unfortunately, it is not MS-DOS and so is not 100 percent compatible. It also requires significant user familiarization before all its features become usable.

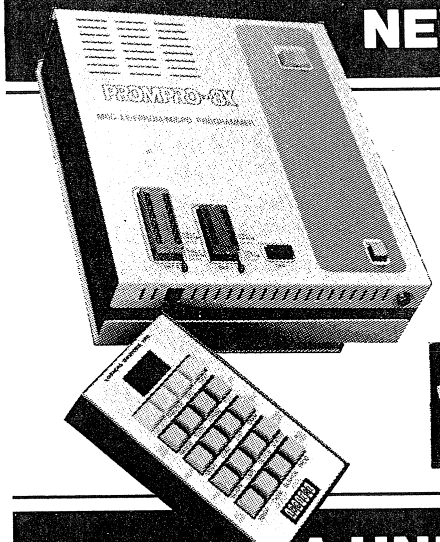
Other multitasking systems that work fine with the DSI-020 include DoubleDOS (SoftLogic Solutions, Manchester, NH), Microsoft Windows, and TopView (IBM). Note that early versions of Windows did not allow a background task to remain active. DoubleDOS is by far the least expensive solution, and although it offers only two tasks plus a print spooler, this is often as many activities as you can concentrate on anyway!

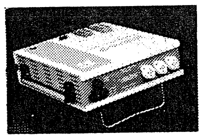
With such a multitasking environment you can operate the DSI-020 as an array processor. An application can be set up in the DSI-020 to, for example, invert some large matrices. You can switch the DSI-020 to a background task and pass parameters to it from a foreground program running on the host computer. It is our experience, however, that this approach is seldom worthwhile. This multiprocessor approach rarely gives equivalent performance to having the entire software package running on the 32-bit processor itself. If it is necessary for you to examine the results using a package such as dBASE, the results can be easily passed to the 8086 in a disk file.

SOFTWARE DEVELOPMENT TOOLS

Two sets of commercial compilers are available for the DSI-020. In addition, Definicon has ported Gordon Brandley's 68000 version of the public domain Palo Alto Tiny BASIC interpreter and a minimal 8080 emulator that can run CP/M-80 software. Tiny BASIC provides the minimal environment to experiment with the 68020,

NEW FROM LOGICAL DEVICES INC:





PROMPRO-8X™ Model II

A stand-alone programmer starting at $895.00 can put you in business to program EE/EPROMs PAL/PLDs,* Single Chip micros,* and Bipolar PROMs,* + EPROM IN-CIRCUIT EMULATION* capability that can speed up your development time considerably and an RS-232 communications port that lets you integrate it with your IBM PC as a total firmware and Logic development station.

All from a company with an excellent reputation for quality and service.

A UNIVERSAL DEVICE PROGRAMMER

Table 2: *Comparison of the* LINPAK *results with two versions of the* DSI-020, *an* IBM RT PC Model 20, *and a* VAX-11/780. *All machines have floating-point hardware or math coprocessors.* VAX-11/780 LINPAK *values courtesy Argonne National Laboratories.*

Double-Precision LINPAK

Performance is quoted in terms of total time taken, in seconds.

DSI-020 (12.0 MHz)	DSI-020 (16.7 MHz)	IBM RT PC	VAX-11/780
10.0	7.38	19.1	4.96

and its performance in no way compares with that of the other DSI-020 software.

All of the specialized compilers are fully optimizing compilers that produce compact and efficient code. Assembly language modules, written using the bundled simple assembler, can be interlinked with the code from any of these compilers.

It should be noted that all of these compilers produce object code directly. In order to see the efficiency of the code being generated you must use a disassembler or Definicon's assembly-level debugger.

Silicon Valley Software (Cupertino, CA) has ported its FORTRAN, Pascal, C, and BASIC to the DSI-020. SVS compilers are currently sold (at much higher prices) by most major workstation manufacturers. The linker and assembler were written by SVS. In addition, Lattice Logic Ltd. (Edinburgh, Scotland) has provided ports of its optimizing Pascal and C compilers for the DSI-020.

The SVS FORTRAN is a 68020 version of the company's fully validated ANSI-77 FORTRAN. It also includes a number of the more common extensions.

The SVS Pascal is an ANSI Level 0 Pascal enhanced with a number of extensions predominantly derived from the UCSD system, including string handling and the ability to compile code modules separately. Thirty-two-bit integers and both single- and double-precision floating-point data types are supported.

LLL Pascal is a 68020 version of Lattice's fully BSI (British Standards Institute) certified compilers, which comply with the standard for ANSI Level 1 Pascal with conformant arrays. The BSI uses the identical standards required by the FSTC (Federal Software Test Center). ANSI Pascal does not support the string data type directly. Strings may, however, be programmed using conformant arrays.

The SVS BASIC interpreter follows the DEC BASIC Plus language definition. SVS will support this interpreter only by paying prompt attention to any bugs that may be reported. It will not be enhanced from its current capabilities.

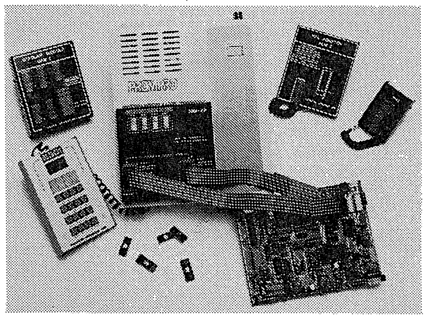
Both SVS C and LLL C are full Kernighan and Ritchie definitions as extended by the UNIX environment. The compatibility of the SVS compiler is sufficient to allow the entire UNIX System V source kernel to be compiled without error.

Definicon has ported the BASIC-to-C converter from Living Software (Milton Keynes, Great Britain) to the DSI-020. This converter takes Microsoft (IBM) BASIC and converts it to a form that can be compiled using either C compiler. The BASIC run-time functions are implemented for the 68020. This converter provides programmers with the fastest possible environment to execute BASIC code. It is not, however, an interpreter. Microsoft BASIC (on the host PC) should be used to debug the program before attempting to convert and run it using the 68020.

Assembly language software devel-

(continued)

LEADER IN DEVICE PROGRAMMING



*some devices require optional adaptors.

FEATURES:
- Stand-Alone/RS-232
- 512K bits RAM buffer
- EPROMs 2716-27512
- Capacity: Two EPROMs (set programming)
- Accepts Intel, Mot. Hex files
- Accepts JEDEC files
- Optional Keypad/Alpha display
- Tilt stand and carrying handle
- Works with Exatron IC handler

also available: **UV ERASERs and GANG programmers.**

All products 100% made in U.S.A.

CALL 1-800-331-7766

Logical Devices Inc. 1321 NW 65th Place
Ft. Lauderdale, FL 33309

	TELEPHONE	TELEX
U.S.A.	(305) 974-0967	383142
Canada East	(514) 694-7710	05-823554
Canada West	(604) 291-8866	04-356533
England	0652-688626	527339
Itly., Ger., Swss	(02) 688-2141	845 79734
Australia	(03) 560-1011	790 35703
	(02) 736-1564	790 82635

WITH AN UNUSUAL PRICE TAG: $895

Inquiry 167 for End-Users. Inquiry 168 for DEALERS ONLY.

opment has been supported by Quelo Inc. (Seattle, WA). Quelo has written a complete software development package that enables you to directly bootstrap a 68020 product. The DSI-020 itself is testimony to the effectiveness of Quelo software. Quelo is offering a specially priced subset of its complete package to BYTE readers, consisting of an assembler, macro preprocessor, and symbol report generator. In addition it is providing LTXCON, a converter to allow modules compiled using Quelo tools to link and run on the DSI-020.

Quelo's assembler package is identical in capability to Motorola's Macro Assembler. It implements completely all the 68020 instructions and addressing modes. The assembler creates standard S record output files in addition to the relocatable output format. This package allows programmers to develop and test other (dedicated) 68020 systems using the DSI-020. Although the program's starting point must be 4000 hexadecimal, all of the memory can be used by a stand-alone program. If the kernel is overwritten, however, no MS-DOS support functions will be available to the user's program.

Editor's note: For prices of the software mentioned in the section above, see part 1 of this article (July BYTE) *or contact* Definicon *Systems.*

CONCLUSION

Since the publication of the first part of this article, we have obtained the LINPAK benchmark results for the DSI-020 board. The LINPAK test suite, from the Argonne National Laboratories, tests the ability of a computer system to perform linear arithmetic, that is, matrix algebra, addition, subtraction, multiplication, and division. Specifically, it uses Gauss-Jordan elimination to solve a series of large matrices. It is written in FORTRAN. Table 2 gives the double-precision results obtained with the DSI-020 at 12.0 MHz, the DSI-020 at 16.7 MHz, the IBM RT PC, and a VAX-11/780. Note: the CPU frequency for the production model DSI-020 is 12.5 MHz, while the benchmark times are for a prototype board (12.0 MHz) and a peak performance version (16.7 MHz).

The DSI-020 is the second 32-bit coprocessor to come from a Definicon design team. As technology marches on, it is often hard to stop and look at where we are and where we are going. The DSI-020, we believe, represents the achievement of a goal we set two years ago: to put VAX-11/780 power on every scientist's desktop.

Reaching that goal, however, is like shooting at a moving target. Already Motorola is talking about 25-MHz 68020s, bringing further memory and peripheral interface problems. It often seems that engineers are good at creating more work for themselves. Let's hope that while we press on with faster and faster technologies, the tools we leave behind get into the hands of the applications programmers who can put them to good use. ■

SideTalk™

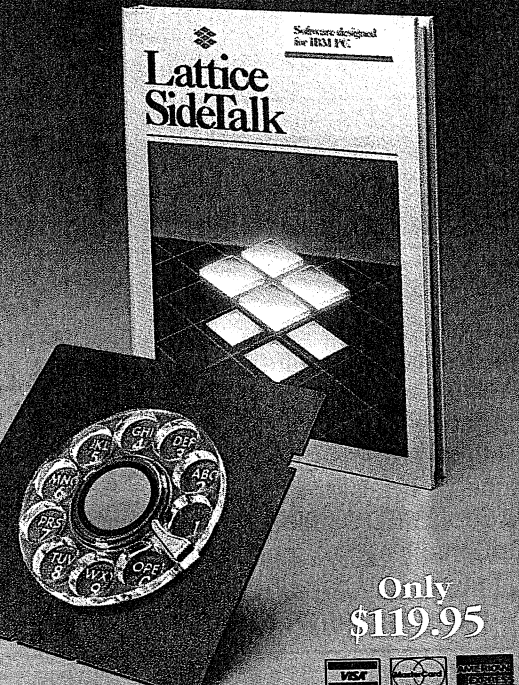
It minds the phone while you do your work

No more interruptions! Because now SideTalk can transfer your files or receive your mail while you're using your computer for other important matters. And SideTalk is programmable with its own BASIC-like language. What's more, you're never more than a keystroke away from all of the power of SideTalk.

When we say SideTalk is the best telecommunications program on the market, we're not just talking out of the side of our mouth.

Lattice

To place your order:
Lattice, Incorporated
Post Office Box 3072
Glen Ellyn, Illinois 60138
312-858-7950
TWX 910-291-2190



Only $119.95

Inquiry 159