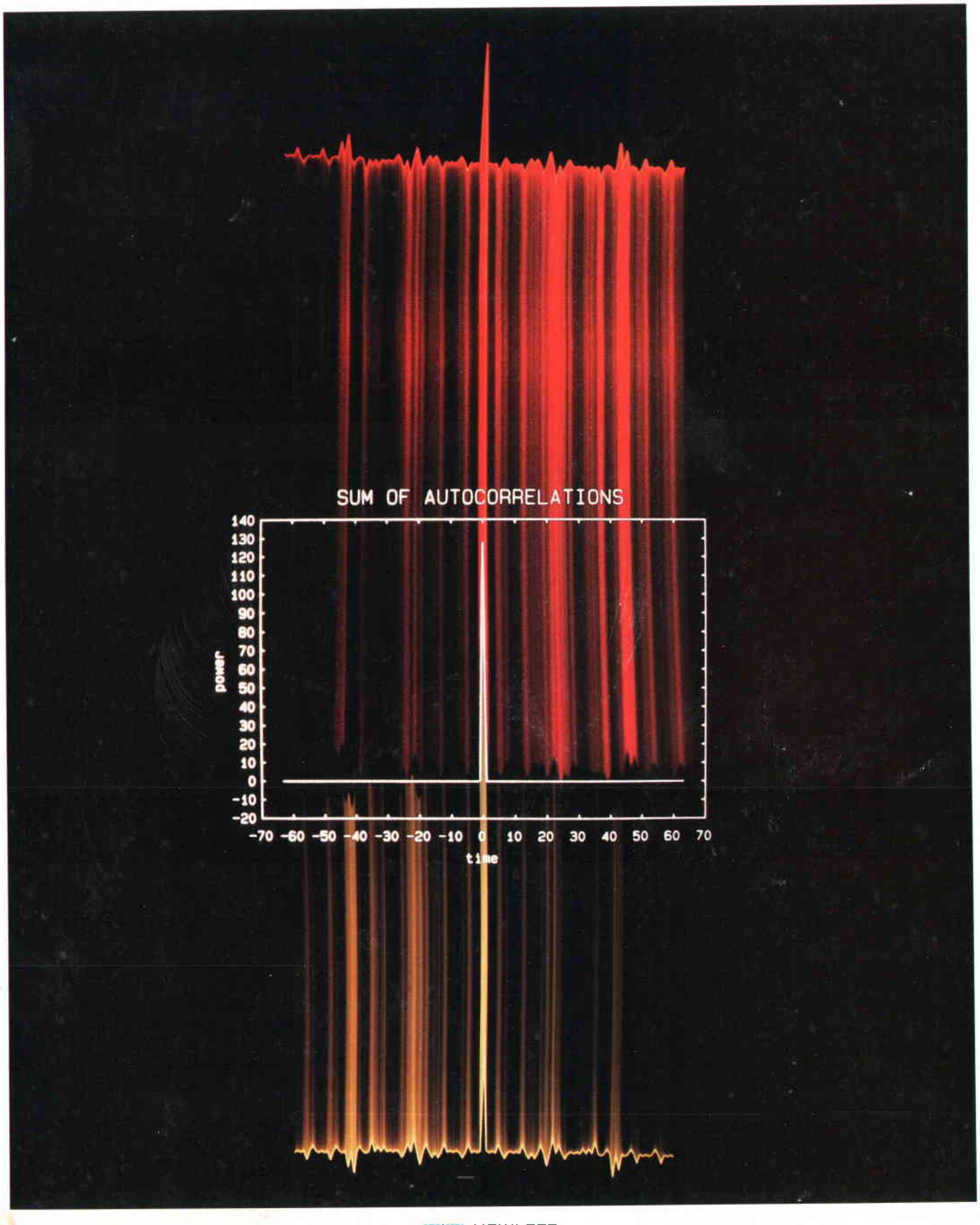
HEWLETT-PACKARD JOURNAL

DECEMBER 1988



HEWLETT PACKARD

HEWLETT-PACKARD JOURNAL

December 1988 Volume 39 • Number 6

Articles

Editor, Richard P. Dolan • Associate Editor, Charles L. Leath • Assistant Editor, Hans A. Toepfer • Art Director, Photographer, Arvid A. Danielson
Support Supervisor, Susan E. Wright • Administrative Services, Typography, Anne S. LoPresti • European Production Supervisor, Michael Zandwijken

Expanded Memory for the HP Vectra ES Personal Computer

This memory subsystem provides high-performance expanded memory and extended memory support for HP Vectra Personal Computer applications while maintaining compatibility with industry standards.

by Gary W. Lum, Milton J. Lau, and Wesley H. Stelter

THE VECTRA ES EXPANDED MEMORY CARD is a memory subsystem consisting of a hardware card and a software driver for the HP Vectra ES Personal Computer. The key design objectives for this subsystem were to:

- Conform to industry standards, including LIM EMS 3.2 and 4.0 (see box, page 61)
- Provide a high-performance expanded memory system to support Microsoft® Windows and the NewWave environment
- Provide a high-performance memory system to support extended memory applications (see box, page 62, for an explanation of extended and expanded memory).

The design challenge was to provide superior performance and yet remain compatible. Both LIM EMS 4.0 and 3.2 define a protocol for using expanded memory, but EMS 3.2 includes support of function calls 10 and 11, which dictate a specific memory architecture, and several major applications make use of them. These software functions define not only the memory cards' I/O port addresses, but the I/O data bit assignments as well. The memory card had to conform to these definitions—a high-performance memory system that was not register-compatible with the LIM EMS 3.2 specification would not meet our objectives. How does one contribute within such a strict definition of compatibility?

The project team began by addressing the level of compatibility required by end-users. Clearly, full support of LIM EMS 4.0 was a requirement for Microsoft Windows, and because some major industry applications use LIM EMS 3.2 functions 10 and 11, supporting these was also necessary. Extended memory support was required for support of new operating systems, such as OS/2. None of these applications specifies a particular mechanical configuration of a memory card, or a maximum CPU or memory clock speed. Research showed that customers wanted at least 1M bytes of additional memory and required any upgrades to be user-installable. Industry trends showed that a maximum configuration of 6M to 8M bytes would be competitive in the next generation of memory cards. Therefore, we required full compatibility with LIM EMS 3.2 and 4.0, but had some freedom in the mechanical configuration, memory configuration, and electrical performance of the card.

Given these constraints, the ES expanded memory card definition evolved into a card that provides the following features:

- 100% compatibility with LIM EMS 4.0 and 3.2
- Daughter board configuration that does not take up a standard Vectra I/O slot.
- Up to 8M bytes of expanded or extended memory
- User-installable memory upgrades
- All 8M bytes of memory running at the same memory speed as the Vectra ES motherboard memory
- EMS 4.0-specific hardware for superior expanded memory performance.

Each of these features adds capability to the Vectra ES Computer, yet does not compromise the required compatibility models. A proprietary motherboard/memory-card interface was defined that makes the 8M-byte memory array appear to the 80286 CPU as motherboard memory. Since the new LIM EMS 4.0 function calls do not specify a hardware architecture, proprietary high-performance circuitry was designed to assist these calls. Surface mount logic ICs and the latest DRAM technology allow both the new logic and an 8M-byte memory array to fit on a single card.

Hardware Architecture

The ES expanded memory card is partitioned into three major blocks (Fig. 1): a single 8M-byte DRAM array, extended memory logic, and expanded memory logic. The extended and expanded logic blocks are two independent memory systems. Separate extended and expanded mem-

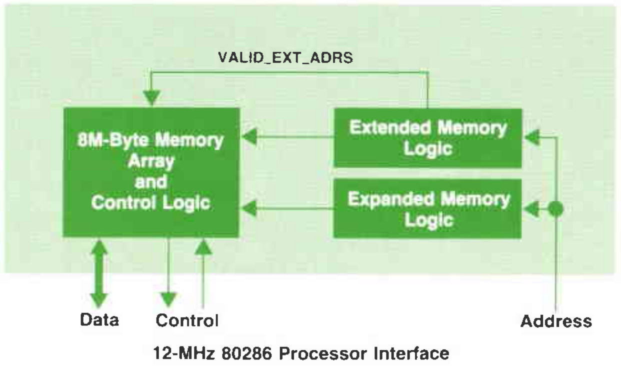


Fig. 1. *Block diagram of the Vectra ES expanded memory card.*

ory addresses are generated each memory cycle, and one is chosen depending on the type of cycle (defined by the state of the highest microprocessor address lines).

The 8M-byte DRAM array is organized as four independent 2M-byte banks (the reason for this is explained later). Each bank is 18 bits wide—16 bits of data and 2 parity bits—which allows a minimum memory configuration of 2M bytes, with upgrades in 2M-byte increments. On-board logic handles all timing and refreshing for the memory array. Each bank can be assigned to either expanded or extended memory via DIP switches, so the user can specify a combination of both kinds of memory.

The extended memory logic is straightforward (Fig. 2). Comparators and DIP switches define a starting address and an ending address on any 1M-byte boundary. These signals generate VALID_EXT_ADRS, which is used in a lookup table to determine which of the four physical memory banks is used. Any memory bank not assigned in the lookup table is considered to be expanded memory, allowing a combination of both extended and expanded memory. DIP switches define all extended memory parameters, since extended memory must be defined at system power-up, before any software driver can configure the card.

The expanded memory logic, shown in Fig. 3, is considerably more complex. Four logically independent banks of eight map tables each,* bank-switching logic, and a set of I/O registers make up this block. The I/O registers are used to configure the bank-switching logic and update map table contents. Once the map tables are programmed, they begin managing the expanded memory pages whenever memory addresses are presented to the board.

Since the card is register-compatible with EMS 3.2, both the I/O port locations and the bit definitions were predefined. To provide EMS 4.0 with the full mapping capability it requires (it allows programs to address up to 32M bytes of expanded memory), a map table of 64 entries was defined. Each entry corresponds to a 16K-byte block in the MS®-DOS 1M-byte address space. With this structure, any open 16K-byte block can be mapped with expanded memory, which gives applications the ability to manipulate much larger amounts of expanded memory than the 64K bytes available in EMS 3.2. When expanded memory is used to backfill user memory, an operating environment can manage several concurrent applications by simply mapping in or out additional pages, rather than moving programs off to disc. Special I/O decoding is used to place the EMS 3.2-compatible I/O ports at their required addresses, while making the additional 60 EMS 4.0-compatible map table entries available via proprietary I/O ports.

*As explained later, there are eight tables per bank so the card can run eight tasks before requiring software mapping.

Map Table Operation

A map table entry consists of a page enable bit and a 7-bit page offset. The page enable bit determines whether the current table entry is active. It is necessary when multiple expanded memory cards are present in a system. The page offset selects 1 of 128 16K-byte expanded memory blocks, which limits each card to a maximum of 2M bytes. The basic operation of the map table is shown in Fig. 4. When a memory address is presented to the map table, the appropriate entry is selected. If the page enable bit is set, then the corresponding expanded memory page is accessed; otherwise, the bank-switching logic will not respond. In a system that has two or more expanded memory cards, only one page enable bit can be set for a given 16K-byte block; otherwise, memory contention occurs.

Because we wanted to support 8M bytes, we needed a way to select one of the possible 512 16K-byte expanded memory blocks. The most straightforward approach was to define the page offset to be 9 bits, which would select 1 of 512 expanded memory blocks.

However, this would make the I/O registers incompatible with LIM EMS 3.2. Therefore, we decided upon a more complex addressing scheme, which makes the card look like four 2M-byte expanded memory cards, as shown in Fig. 5. Each of the four memory banks is defined as a logical LIM EMS 3.2 board, each capable of addressing its respective 128 16K-byte expanded memory blocks. The bank-switching logic simulates the parallel memory decoding of four boards and generates a single address to the 8M-byte

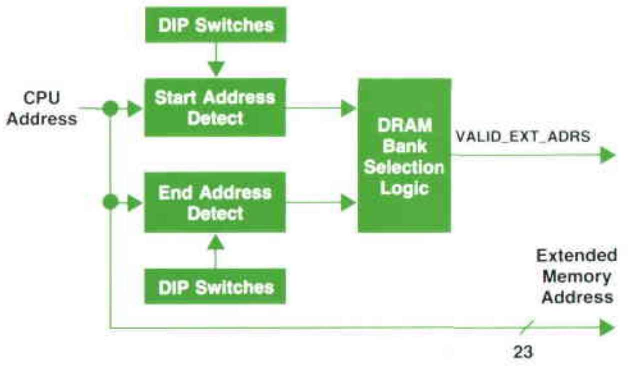


Fig. 2. *Extended memory logic.*

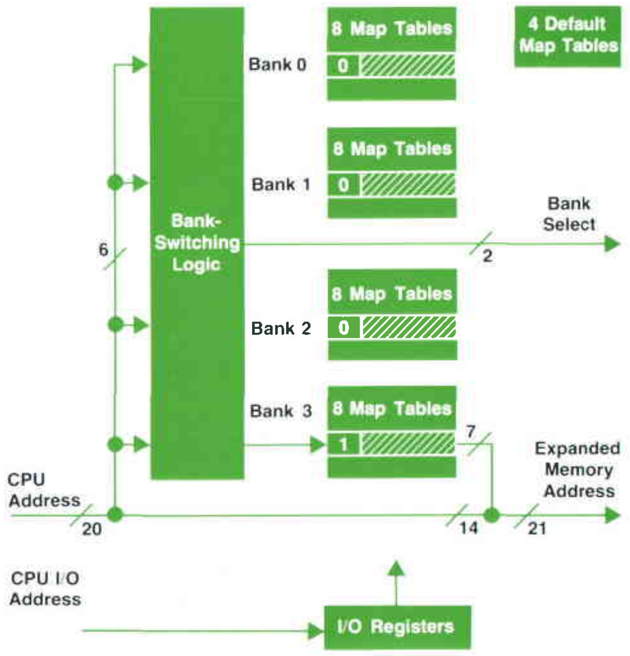


Fig. 3. *Expanded memory logic.*

memory array. To any expanded memory application, the ES expanded memory card looks like four independent 2M-byte memory cards.

To improve EMS 4.0 performance within the EMS 3.2 compatibility model, new circuitry was developed. The resulting capability is a superset of EMS 3.2, incorporating multiple map tables, independent control of these map tables, and a microprocessor interface that allows 12-MHz memory operation anywhere within the 16M-byte address range. Each of these capabilities is usable by software drivers that know of their existence, such as the Vectra ES expanded memory manager. Applications that write directly to I/O registers are not affected by the additional circuitry.

Multiple map tables work on the same principle as microprocessor register sets. Rather than storing in memory an image of an active register set when invoking a new task, additional copies of the register sets are available, and can be selected using a single instruction. Using data from the HP Windows development group, it was decided to implement eight map tables, allowing the card to run eight tasks before requiring software mapping. Unique to bank 0 on the expanded memory card are four additional default map tables. These nonvolatile map tables are used at power-on, and store different configurations of backfilled memory. DIP switches are used to select the appropriate backfill configuration. All map tables are controlled by two proprietary I/O registers, MTACTIVE and MTACCESS. MTACTIVE defines which map table is used at execution time. MTACCESS defines which map table is accessible, and therefore updated, through the appropriate (and EMS 3.2-compatible) I/O registers. The independent control allows a current task to continue executing while a new task is being set up.

The Vectra ES motherboard memory is a 16-bit-wide array operating at 12 MHz, with one wait state (249-ns cycle time). The objective of the ES expanded memory card

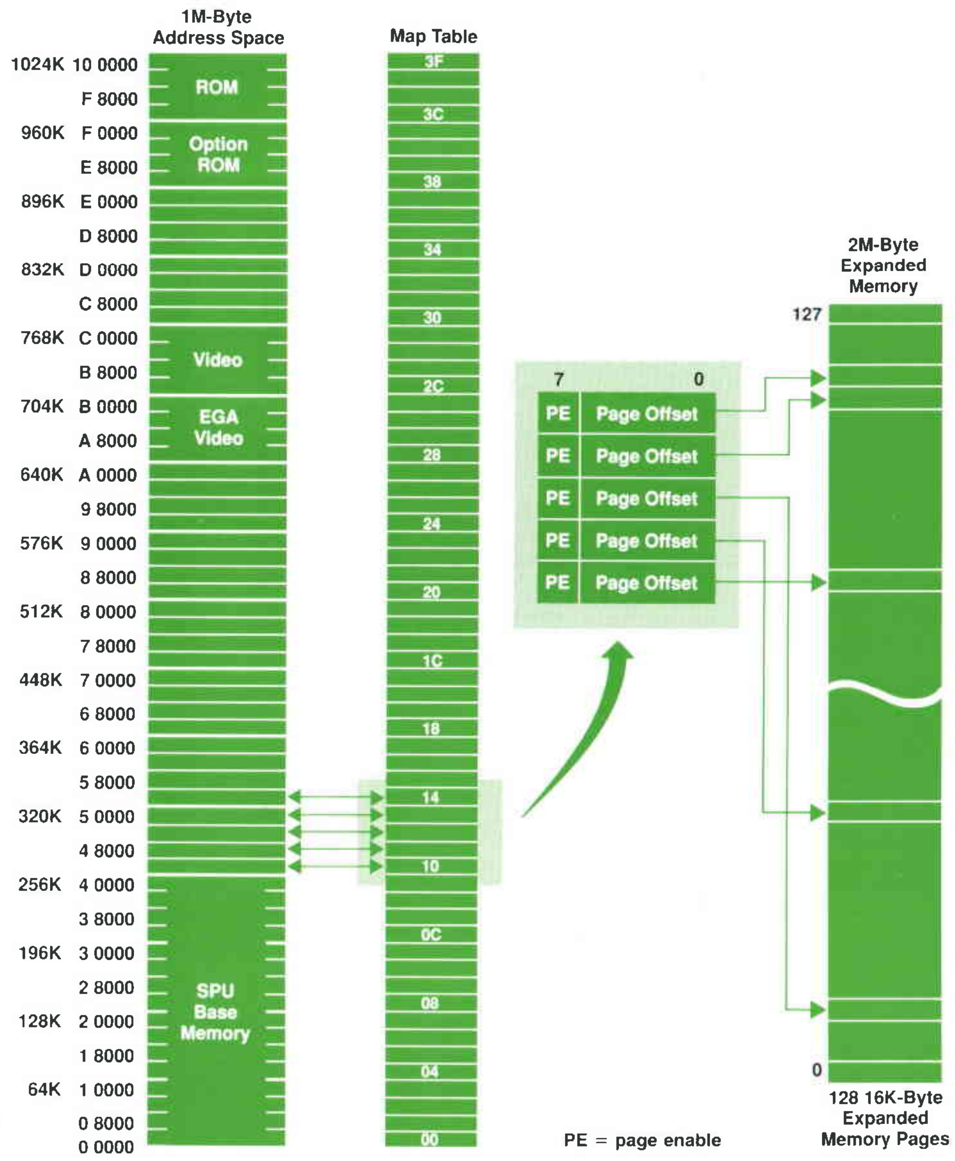


Fig. 4. *Map table operation.*

was to operate its 8M-byte memory array at the same speed and memory width. Because of timing limitations in the ES/12's industry-standard backplane, all memory accesses are limited to 8 MHz with one wait state. Furthermore, backplane memory can only be accessed as 16-bit words if an entire 128K-byte address range is programmed for 16-bit accesses, again because of timing restrictions on the industry-standard backplane. Because much of the 640K-to-1M-byte DOS address range runs as 8-bit accesses, a 128K-byte block may not exist, and consequently all backplane memory accesses may only be 8-bit. Quick calculations showed that 8-bit accesses at 8 MHz with one wait state would not meet our performance goals. It became clear that the ES expanded memory card could not operate over the standard backplane, and a new proprietary connector was defined which allows access to the local 80286 buses and control signals. The particular signals and buses used are:

- The local 80286 data bus to provide 20 ns of additional data setup time for a write to the expanded memory card.
- The local 80286 address bus to provide decoding 40 ns earlier than available over the backplane. It is mainly access to this bus that makes 16-bit, 12-MHz memory operation possible.
- Memory commands (MEM_READ, MEM_WRITE) directly from the 80286 to generate memory timing signals (RAS, CAS). This allows standard 100-ns DRAM to be used.

The use of local 80286 buses required new circuitry to be added to the ES/12 motherboard. Because both motherboard memory and the expanded memory card reside on the local buses, direction control of data and address buffers needed to be modified when memory accesses were initiated by either DMA devices or backplane masters.

Hardware Implementation

The Vectra ES expanded memory card is roughly the size of a standard Vectra I/O card. Two proprietary edge connectors are intentionally offset from standard backplane connectors so users are mechanically prevented from installing the ES expanded memory card in a standard I/O slot.

The 8M-byte memory is implemented using 1M-byte single in-line memory modules (SIMMs). Each SIMM has nine 1M × 1-bit, 100-ns CMOS DRAMs mounted on a small printed circuit board. The supporting circuitry is implemented in discrete TTL and PALs. High-speed (25-ns) SRAM is used to implement the 2048 map table entries required for expanded memory. The nonvolatile map tables are stored in high-speed PROM. Surface mount technology is used wherever possible. Roughly 85% of the components are surface mounted.

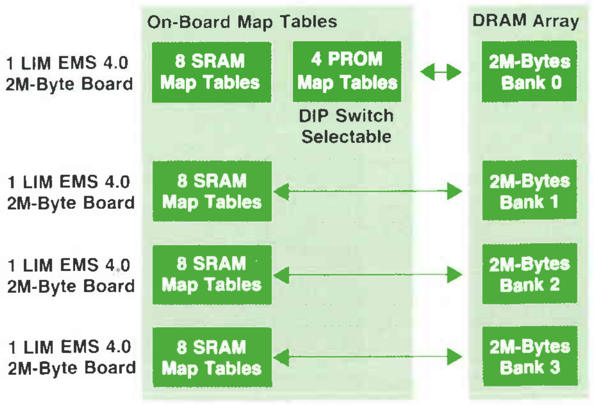


Fig. 5. *An application's view of the ES expanded memory card.*

Software Driver Architecture

The Vectra ES expanded memory manager is a software driver. It must be loaded before the expanded memory card can be used. It is designed to be 100% compatible with the LIM 3.2 and LIM 4.0 specifications.

All code was developed in 80286 assembly language under the MS-DOS environment. The code is written as a standard character device driver and is installed during normal initialization of MS-DOS. Although it does not perform any character I/O operations, it is identified with a character device name, EMMXXXX0, and remains as a typical terminate, stay resident program. The size of the resident code is dynamic and varies with the amount of expanded memory available in the memory pool. This achieves efficient use of conventional memory to allow other huge applications to run. After initialization, the typical size of the memory manager is about 8K bytes when managing 4M bytes of expanded memory.

The memory manager consists of three separate pieces of code:

- Initialization routines
- LIM 3.2 memory management functions
- LIM 4.0 memory management functions.

Initializing the expanded memory card requires five tasks. The first is scanning the 1M-byte address range for

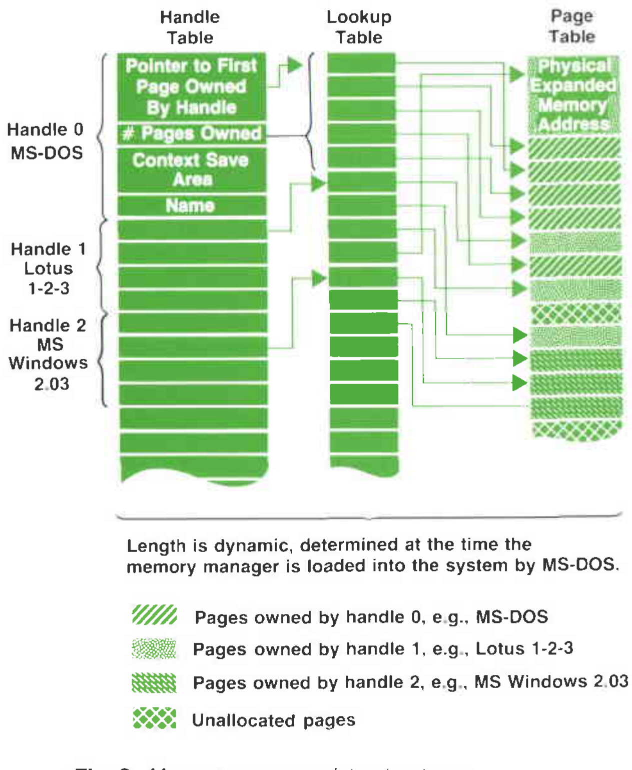

Handle Table
Lookup Table
Page Table
Pointer to First Page Owned By Handle
# Pages Owned
Context Save Area
Name
Physical Expanded Memory Address
Handle 0 MS-DOS
Handle 1 Lotus 1-2-3
Handle 2 MS Windows 2.03
Length is dynamic, determined at the time the memory manager is loaded into the system by MS-DOS.
Pages owned by handle 0, e.g., MS-DOS
Pages owned by handle 1, e.g., Lotus 1-2-3
Pages owned by handle 2, e.g., MS Windows 2.03
Unallocated pages

Fig. 6. *Memory manager data structures.*

unused 16K-byte blocks. These empty spaces are areas in which pages of expanded memory can be mapped. The spaces need not be contiguous and every available space from the end of base memory through option ROM is scanned. The algorithm uses a nondestructive read/write scheme to check for RAM and the industry-standard AA55 checksum for ROM. In cases where known areas of memory space are being used for memory mapped I/O, an exclude option on the driver command line prevents searching these blocks.

The second initialization task is setting up the map tables. To allow expanded memory to be mapped into various 16K-byte holes, the hardware needs to be told what pages can be mapped. This is achieved by writing the expanded memory page number into a map table. On power-up the default map table is read from PROM and copied into one of the eight SRAM map tables. During this procedure, the expanded memory pages used to backfill MS-DOS memory from 256K bytes to 640K bytes are marked and excluded from the the expanded memory pool.

The third initialization task is identifying memory types. Conventional memory on the expanded memory card is identified by reading the default map table. The remaining memory is either extended or expanded, and is determined by using an overlapping algorithm. An ID byte is written on every 16K-byte interval in extended memory. Then, using expanded memory, every 16K-byte page is read. If the expanded memory page contains the extended memory ID byte, then the page is extended memory. Pages being used as extended memory are marked so they do not get allocated into the memory pool. The read/write test for extended memory is nondestructive so that any extended memory applications such as VDISK.SYS will not be destroyed.

The fourth initialization task is a memory test. The RAM test is used to verify the memory's read/write ability. Any 16K-byte page that contains a bad memory location is marked and not entered in the memory pool. This procedure is also necessary during power-on to initialize the DRAM parity checking circuitry.

Data Structure

The final initialization task is data structure setup. The memory manager uses three data structures to control the expanded memory pool (see Fig. 6). These are the handle table, the lookup table, and the page table.

The handle table is used to store information about handles and the pages they own. A handle is merely a token used by the memory manager to identify an entry in this table. When applications request pages, a handle is returned back to them so that when future requests to manipulate the pages are made, the owner of the pages can be identified. One handle is returned for every request to allocate pages from the memory manager. Each entry consists of a pointer to the first page owned by the handle, the number of pages owned by the handle, the context save area, and the name of the handle (8 bytes of ASCII).

The lookup table contains the N consecutive logical page translation mappings that belong to a handle. Each element is considered a logical page and contains a pointer to a page table entry or a null entry if unused.

The page table contain two fields. The first field is used to indicate that the page has been allocated and the second field contains the physical address of the expanded memory page being used. The mapping of a page is finished when this address is written into an entry in the currently active map table.

The handle table is adjustable and can be programmed by the user to allow up to 64 handles. The lookup and page tables are dynamically determined by the amount of expanded memory available. The more expanded memory added to the card, the larger the data structure. This approach eliminates wasted space for unused entries because of nonexistent expanded memory. Since the majority of memory manager requests are allocating, reallocating, deallocating, and mapping expanded memory, a data structure is needed that handles the requests in an efficient manner. At first a linked list was considered because it provides a fast means of reallocating and deallocating memory pages; however, it would not be efficient for mapping pages. An important realization was that almost all memory manager activities after an application has been allocated expanded memory involve mapping those pages so the application can use them. Therefore, a lookup table was chosen over a linked list because it is much faster for translating and mapping logical pages. Speed in mapping pages

LIM EMS 3.2 and 4.0

Lotus Development Corporation, Intel Corporation, and Microsoft Corporation jointly defined the original *LIM Expanded Memory Specification* (EMS) in 1984. This specification, at revision 3.2, allows programs with an EMS 3.2 driver to use up to 8M bytes of expanded memory for data storage. LIM EMS 3.2 defines a rather limited bank-switching scheme, allowing only four 16K-byte blocks of expanded memory to be present at any one time. These four 16K-byte blocks, or pages, must be contiguous, meaning that a single 64K-byte open space must be available for bank switching. This 64K-byte page window must reside above user memory, typically at segments C800H, CC00H, D000H, and D400H. RAM discs, print spoolers, and certain programs (such as Lotus® 1-2-3® or AutoCAD™) make use of this additional storage. Early versions of LIM EMS include software function calls that allow an application to write directly to the I/O ports of the expanded memory card. These calls are not documented in revision 3.2, but they are supported.

In August 1987, LIM EMS 4.0 was announced. With 4.0, several limitations of version 3.2 were removed. EMS 4.0 allows programs to address up to 32M bytes of expanded memory, code as well as data to be stored in expanded memory, and the functionality of multitasking, which allows multiple applications, RAM discs, and spoolers to run simultaneously in expanded memory. Any open 16K-byte block in the 1M-byte address space can be filled with expanded memory.

EMS 4.0 can run on expanded memory cards designed only for EMS 3.2; only a new driver is required. Such a system, however, will not make use of all the power of EMS 4.0, because the new function calls can be accessed only by hardware cards that have specific EMS 4.0 circuitry, such as backfill capability or multiple map tables. Map tables are the tool used by the expanded memory manager driver to manage the bank switching of expanded memory.

is a priority, since operating systems, device drivers, and interrupt subroutines also make calls to the memory manager. Consider the following algorithm to map a logical page using a linked list data structure:

```
while ( lookup→logical_page != logical_page ) do {
        lookup = lookup→next /* traverse */
}
/* map the page */
out( MAP_TABLE, page_table[lookup→logical_page])
```

The translation from logical page to physical address would have required an average of 64 traversals down the linked list data structure for a 2M-byte card. This number would double for every 2M bytes added to the system.

Using the lookup table data structure, the algorithm is reduced to:

```
/* map a page */
out( MAP_TABLE, page_table[ lookup[ logical_page ] ] )
```

The translation from logical page to physical address requires only two reads.

Expanded versus Extended Memory

Three kinds of memory can exist in an MS-DOS personal computer: conventional, extended, and expanded. The physical memory devices used are identical in all cases. Only the access mode differs.

Conventional memory resides in the linear address space of an Intel microprocessor (8088, 8086, 80286, or 80386) running MS-DOS. This memory can be accessed directly by the 20 address lines of an 8088/6. Of this 1M-byte address space, the lowest 640K bytes is allocated as conventional memory (also referred to as user or base memory). Memory space between 640K bytes and 1M bytes is designated by the computer architecture for video, hard disc, option ROMs, and other system utilities, but there are normally areas in this 384K-byte memory that are not used.

Extended memory is the linearly mapped memory above the 1M-byte memory space. In a virtual-mode 80286 or 80386, up to 15M bytes of extended memory can be accessed. Because MS-DOS does not support virtual mode, extended memory is limited to specially written utilities such as RAM discs and disc caches. An 8088/6 cannot access any extended memory.

Expanded memory is a method that allows any Intel microprocessor running MS-DOS to access more than 1M bytes of memory. This is accomplished by bank switching, in which 16K-byte blocks (or pages) of expanded memory are mapped into some or all of the empty spaces that occur in the 1M-byte memory space. An expanded memory specification (EMS) describes the specific software calls that manipulate the bank switching.

Most personal computers allow a portion of their base memory (which normally resides on the motherboard) to be disabled, usually from 256K bytes to 640K bytes. Although the disabled memory is inaccessible, it allows the more powerful expanded memory to replace, or backfill, the disabled base memory. This capability is particularly useful for supporting LIM EMS 4.0.

Expanded Memory Access

The LIM 4.0 functions provide an interface for operating systems and/or applications to access expanded memory. Each function is optimized for speed. The memory manager is also reentrant so device drivers and interrupt service routines can use expanded memory. An application requesting service from the memory manager would follow these steps:

1. Request from the memory manager M logical pages of expanded memory. The memory manager will scan the page table entries for unallocated pages. When they are found the lookup table is updated with M consecutive entries pointing to the allocated pages. The number of pages requested and a pointer to the first logical page are stored in the handle table. A handle representing an entry to the handle table is returned to the application.
2. Request the page frame address. There are always at least four consecutive 16K-byte page frame windows available for mapping in the 1M-byte address space (to maintain compatibility with LIM 3.2). The memory manager will return the segment address of the first 16K-byte page frame.
3. Request to save the current page map. Since other applications may have already mapped expanded memory into the page frame, the current mapping must be preserved. The mapping is saved in the context save area of the handle table.
4. Request to map a page. The application will give the memory manager a handle, a logical page, and a physical page. From the handle and logical page, a physical address of the expanded memory page will be translated. The map table entry given by the physical page will be written with this physical address, thus completing the mapping.
5. At this time the application can now read, write, and/or execute code at the page frame address where the page is currently mapped.
6. Request to unsave a page mapping. When the application has completed its task with the mapped page it was using, it must replace the original page mapping before leaving. The memory manager will replace the old mappings that were stored in the context save area of the handle table.
7. Request to deallocate pages. The memory manager will deallocate pages belonging to the application by resetting the allocated bit for each page in the page table. The lookup table entries will then be cleared and adjusted for the deallocated pages. Each active handle will have its pointer to the first logical page entry adjusted to the lookup table. Other applications can then reuse the expanded memory pages returned to the pool.

Performance Data

The data collected on ES expanded memory card compatibility and performance has been excellent. We have not found any application or utility using LIM EMS 3.2 or 4.0 that does not run on the ES expanded memory card. In comparisons of a Vectra ES/12 with the ES expanded memory card and an ES/12 with a conventional expanded memory card, the following data has been taken:

Load Time from Disc to Memory
(seconds)

Application	ES Expanded Memory Card	Conventional Card
Windows Paint	4.3	8.1
Lotus® 1-2-3®	3.9	5.8

Given these results, we believe we have met our initial design objectives. The performance differences in the above data are attributable to the ES card's 16-bit memory access anywhere in the address space, 12-MHz operation anywhere in the address space, use of the eight hardware map tables, and optimized driver design.

Conclusions

The development of products for the industry-standard marketplace is deceivingly difficult. Because "industry compatible" is not always a cleanly written specification, part of the design process is determining what aspects of industry compatibility need to be considered. The challenge for product developers is to identify where contributions can be made within the nebulous compatibility model and then provide added value without compromising that model. The ES expanded memory card, with its high-speed memory system, large memory size, and adherence to industry standards, is an example of added value within the industry-standard marketplace.

Acknowledgments

The development of the ES expanded memory card represents the contributions and commitment of many people and functional areas. We would like to thank Rich Archuleta and Tuan Nguyen for the initial conception of the product, Chrissy Klander for design and test contributions, and Greg Woods for test code development. The HP Grenoble Personal Computer Division software design team, headed by Yves Loheac, deserves special recognition for their development of the installation program.

References

1. *Lotus/Intel/Microsoft Expanded Memory Specification*, Version 4.0, October 1987.
2. T. Mirecki, "Expandable memory," *PC Tech Journal*, Vol. 4, no. 2, February 1986.
3. J. A. Lefor and K. Lund, "Reaching into expanded memory," *PC Tech Journal*, Vol. 5, no. 5, May 1987.

Authors

December 1988

6 Advanced Optical TDR

Franz Sischka

Born in Stuttgart, West Germany, Franz Sischka earned his Diplom Ingenieur degree (1979) and doctoral degree (1984) from the University of Stuttgart. After completing his studies in 1979, he joined the Institute of Network and System Theory of the University. During this time, he worked on a new and easily calculable modeling scheme for bipolar transistors and on the design of low-power amplifiers. Franz joined HP in 1984 and since has focused his professional interests on fiber optic instruments. He was a project leader for the HP 8145A OTDR.

Michael Fleischer-Reumann

With HP since 1980, Michael Fleischer-Reumann is an R&D project manager in the lightwave section of the Böblingen Instrument Division. He has been project manager or project leader for a number of fiber optics products, including the HP 8152A Optical Average Power Meter, HP 8154B LED Source, and HP 8158B Optical Attenuator, and he contributed to the hardware design of the HP 8112A 50-MHz Pulse Generator. A patent describing pulse generator timing is based on his ideas. In 1986, he assumed resonsibility for OTDRs. Michael has written or coauthored several previous articles for the HP Journal. He was born in Essen, Germany, and received his Diplom Ingenieur degree from the Ruhr University of Bochum. He and his wife live in Gechingen. In his spare time, Michael teaches electronics at a Stuttgart college. He also enjoys backpacking, mountain hiking and bicycling, white-water kayaking, and playing guitar.

14 Complementary Correlation OTDR

Franz Sischka

Author's biography appears elsewhere in this section.

Moshe Nazarathy

Moshe Nazarathy received his BSc and DSc degrees in electrical engineering from the Technion Institute of Technology, Israel. In the ensuing two years, he held a postdoctoral appointment with the Information Systems Laboratory at Stanford University. Moshe came to HP in 1984 and joined the staff of the instrument and photonics laboratory. His design contributions to the HP 8145A OTDR include the signal processing algorithms. A number of other projects he has worked on include fiber optic signal processing circuits and test instrumentation, solid-state lasers, and ultrafast optoelectronic modulation and sampling techniques.

Steven A. Newton

As a project manager at HP Laboratories, Steve Newton directed the design contributions of his organization to the OTDR project and helped invent the instrument's signal processing system. Born in Teaneck, New Jersey, he received his BS degree in physics from the University of Massachusetts (1976), and his MS (1978) and PhD (1983) degrees in applied physics from Stanford University. He came to HP on a part-time basis in 1978 and worked on optical design, optical data storage, and metal vapor lasers. Since he joined the HP Laboratories staff full-time in 1983, Steve has focused on research on single-mode fiber components, circuits, and measurement systems. He is now project manager of the fiber optics team in the instruments and photonics laboratory. He is named inventor or coinventor on seven patents and has authored or coauthored over 30 papers and articles. Steve and his wife live in Belmont, California. His outside interests include sports and music.

22 Optical TDR Components

Siegfried Gross

A project engineer in the fiber optics development section of the Böblingen Instrument Division, Sigi Gross joined HP in 1984. As a member of the team that developed the HP 8145A OTDR, his main responsibility was design of the laser driver circuits. His Diplom Ingenieur degree is from the Berufsakademie Stuttgart. He, his wife, and his small daughter live in Sindelfingen, West Germany. His hobbies include several kinds of sports, choir music, and photography.

Robin Giffard

A native of the island of Guernsey, U.K., Robin Giffard received his undergraduate and graduate degrees in physics from Oxford University. Since he came to HP in 1980 as a development engineer, he has worked on a variety of projects, including trapped-ion frequency standards and laser interferometer circuitry. On the HP 8145A OTDR, his design objectives focused on the receiver chain. Robin is a resident of Los Altos, California, is married, and has two small children. In his off-hours, he enjoys reading and music.

Jürgen Beck

Born in Tübingen, West Germany, Jürgen Beck joined HP's Böblingen Instrument Division in 1984. He first performed experimental investigations of the optical system for the HP 8145A OTDR, using the results for his diploma thesis. After receiving his Diplom Ingenieur degree from the University of Stuttgart, he continued work on the OTDR project, developing the mechanical configuration of the instrument. He later became responsible for the optical system. Jürgen, his wife, and his son live in Tübingen. Among his favorite pastimes are traveling, skiing, sailing, and riding his motorcycle.

29 OTDR Data Processing

Wilfried Pless

After receiving his Diplom Ingenieur degree in 1983 from the Ruhr University in Bochum, West Germany, Willy Pless joined HP as an R&D engineer. His first assignment was with a team developing the HP 8151A Optical Pulse Power Meter, one of HP's first fiber optics instruments. His main responsibility in the development of the HP 8145A OTDR included the signal processing software. Willy was born in Westphalia, is married, and has three children. He is building a house, which monopolizes most of his spare time.

Jochen Rivoir

Jochen Rivoir received his Diplom Ingenieur degree in 1983, after studying electronics engineering with the emphasis on system design at the University of Karlsruhe. He came to HP in 1985. On the HP 8145A OTDR project, Jochen's varied contributions encompass the processor circuit board, the DSPE circuit board, and firmware elements. He enjoys swimming, skiing, singing, and playing guitar, but a newborn daughter demands all his and his wife's attention.

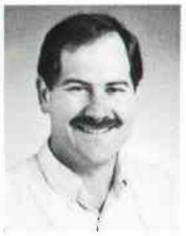
35 OTDR User Interface

Joachim Vobis

Joachim Vobis is a native of Heidelberg, West Germany. He joined HP after receiving his Diplom Ingenieur degree from the University of Karlsruhe. His initial assignments included the design of fast analog circuits for photonic instruments, but he soon transferred to a team of software designers where he assumed responsibility for the entire firmware package for the HP 8145A OTDR. Joachim is married and has a son and two daughters. Among his leisure activities are swimming, hiking, backpacking, and amateur radio.

39 Plain Paper Printing

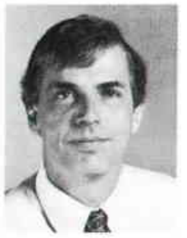
Steven J. Bares



Steve Bares carried the engineering responsibilities for the technical studies of plain papers reported in this issue of the HP Journal and acted as liaison to marketing for all plain-paper market research. He continues to be involved in ink developments for future inkjet printers and in matching HP's inkjet products with the capabilities of the paper industry. Before coming to HP in 1985, Steve worked on experimental and theoretical studies of laser desorption and high-power laser/surface interactions in a postdoctoral appointment at Exxon Research and Engineering Company. His BS degree in chemistry is from the California State University at Humboldt (1979), and his PhD degree in physical chemistry (1983) and his MBA degree are from Oregon State University. Steve has published five papers on laser studies and has submitted a number of patent disclosures describing inkjet inks, dyes, and paper testing systems. Steve was born and raised in California, is married and has two children. He lives in Corvallis, Oregon. His leisure activities include playing guitar, international travel, political history, and racquetball.

45 Host Independent Emulators

Arnold S. Berger



Arnie Berger was hardware project manager and leader of the original design and definition team for the HP 64700 Series Emulators. His past design responsibilities include the storage CRT for the HP 1727A Oscilloscope and the HP 54300A Probe Multiplexer, and he was the project manager for the HP 64120A Card Cage. His professional experience before joining HP in 1979 includes basic research in nondestructive testing as a senior staff scientist at Ford Motor Company and research on crystalline defect interactions in metal at Argonne National Laboratory. Arnie's BS degree (1966) and PhD degree (1971) are in materials science and are both from Cornell University. He has written some 25 papers and oral presentations, two journal articles, and one prior HP Journal article. He is a member of the American Physical Society and an authority on techniques for the measurement of electrical resistivity at cryogenic temperatures. Arnie was born in Brooklyn, New York, is married and has a child. He and his wife, who is a software design engineer in HP's Electronic Design Division, live in Colorado Springs, Colorado. His avocations include bicycling, woodworking, running, and restoring old HP instruments.

52 Emulator Software

William A. Fischer, Jr.

During development of the HP 64700 Emulator, Bill Fischer was project manager in charge of software design. He has since become section manager with responsibility for Intel and Motorola emulation products. Previously, he has worked as technical support engineer and product marketing engineer. In the ten years before he joined HP in 1984, Bill was an engineer at the Hamilton Standard Division of United Technologies, designing automated test equipment fór the space shuttle. He holds a BSEE degree (1973), an MSEE degree (1978), and a master's degree in management (1980), all from Rensselaer Polytechnic Institute. He has authored and coauthored a number of papers and articles, mainly about software project management. Bill was born in Attleboro, Massachusetts, and lives with his wife and four children in Colorado Springs, Colorado. He enjoys running and playing basketball.

57 Expanded Memory

Gary W. Lum

As lead hardware engineer, Gary Lum's contribution to the NewWave development was the expanded-memory card hardware architecture for the HP Vectra ES Personal Computer. He is now project manager for advanced PC development. Since joining HP in 1979, he has worked on the logic design for HP 2627A and HP 2626A terminals, IC design for the HP Touchscreen II Personal Computer, and on HP Vectra PC development. Gary's BS degree in electrical engineering is from the University of California at Berkeley (1979) and his MS degree is from Stanford University (1981). He was born in Syracuse, New York, is married, and lives in Santa Clara, California. He is interested in film and film history, photography, and gardening.

Milton J. Lau

R&D engineer Milton Lau served as software project leader for development of the HP Vectra ES expanded-memory manager and RAM disc, and for debugging the expanded-memory card. Among his past projects were the operating system and firmware for the HP 150 Touchscreen PC and BIOS firmware for the HP Vectra PC. Before joining HP in 1984, he was an R&D engineer for a small company, developing firmware for terminals compatible with HP and Digital Equipment Corporation systems. Milton's BSCS degree is from California State University, San Francisco (1983). He was born in San Francisco, California, and lives in Milpitas, California. His favorite leisure activities include basketball, skiing, and flying remote-controlled gliders.

Wesley H. Stelter

With memory system design and architecture his special interests, Wes Stelter's work experience as development engineer includes design for the HP Vectra and Vectra ES Personal Computers. On the NewWave project, he was responsible for the expanded-memory board for the Vectra ES PC. He attended Cogswell College and received a BSET degree in 1977, the year he also joined HP. Wes was born in San Francisco, where he serves as assistant scoutmaster in the Boy Scouts. He's married, has a child, and makes his home in San Bruno, California. For recreation, Wes enjoys hiking, marksmanship, history, and house reconstruction.

74 Redfield-Kunz Generalization

Ulrich H. Haeberlen

Ulrich Haeberlen's involvement with the discipline of nuclear magnetic resonance goes back to the years 1967 to 1969, when he worked in John Waugh's NMR laboratory at the Massachusetts Institute of Technology. When he later returned to his native Germany, Ulrich joined the Max Planck Institute for Medical Research in Heidelberg to establish a research group focusing on solid-state NMR. He attended the Technische Hochschule in Stuttgart, where he earned his diploma and PhD degree. Ulrich's association with HP dates back to the mid-sixties, as he likes to recall, when an HP RF double-balanced mixer he was using greatly impressed him with its design and performance characteristics.

Alexander Keller

The theoretical and mathematical aspects of molecular motion in solid-state nuclear magnetic resonance have been Alex Keller's focal interests in the past three years. He was born in Germany and studied at the Heidelberg University, where he earned his PhD degree. Although Alex' primary professional concern is with mathematics, there are occasional ventures into experimental physics, one of which produced his contribution to use of the HP 5180A Waveform Recorder described in this issue of the HP Journal.