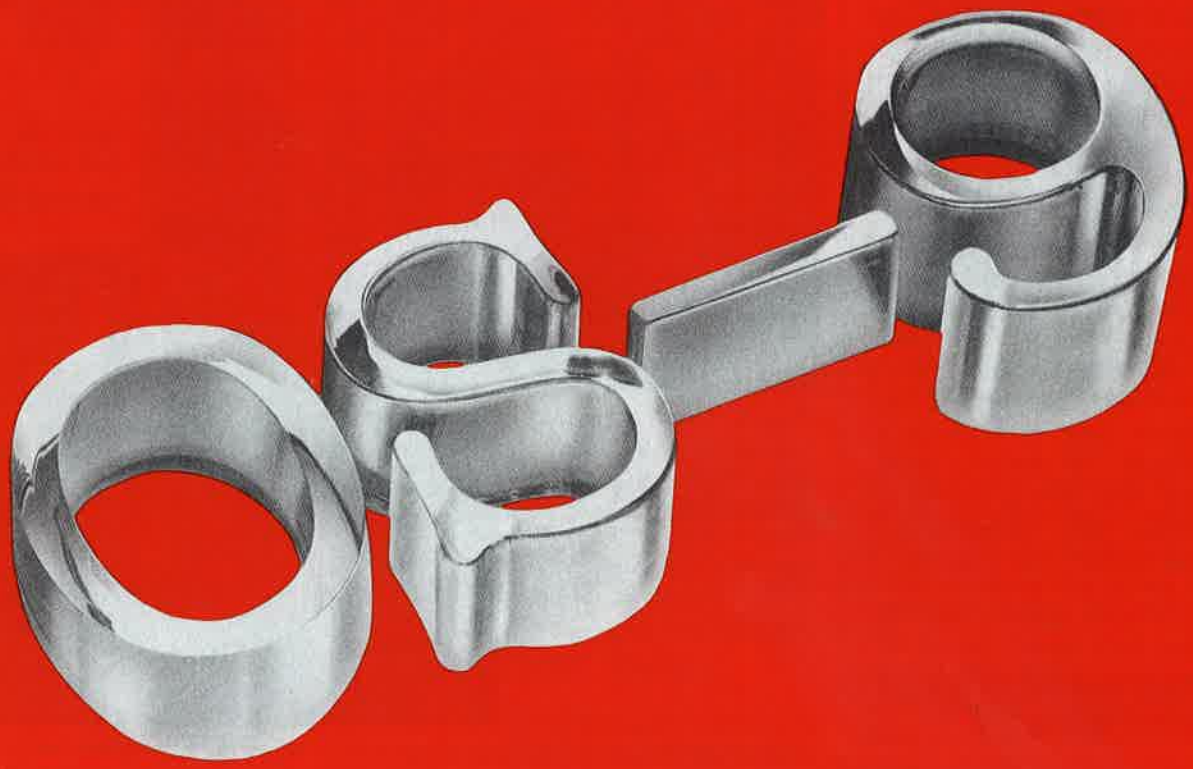
microware® PIPELINES

Covering the Full Spectrum of 6809 and 68000 Applications of OS-9

Volume 2 Number 2 Summer 1986



In This Issue:

PIPELINES
Summer 1986
Volume 2 Number 2

Pipelines is published quarterly by:
Microware Systems Corporation
1866 N.W. 114th Street
Des Moines, IA 50322
(515)224-1929

Contributing Editors:
Andy Ball
Phyllis Casel
Ken Kaplan
Kim Kempf
Dave West

Photography: George Edwards

Do you have news for Pipelines? Articles submitted for publication should include author's name, address, telephone number and date. Comments, suggestions and letters of "gripes" as well as "praise" are solicited. Please send to Microware, c/o Phyllis Casel. If there is someone you know who would like to be on our mailing list, please call or mail in their name and address. *

HARDWARE and SOFTWARE SOURCEBOOKS

Updated versions of the OS-9 Software Sourcebook and Hardware Vendor List are available free of charge. These lists are comprehensive guides containing application software and hardware running under OS-9/6809 and OS-9/68000. The catalogs were compiled as a service to OS-9 users and include information obtained and condensed from the supplier's catalogs and advertisements. Microware has attempted to make sure, but cannot guarantee, that all products listed are presently available. If you would like to receive a copy of either of these guides please contact the Marketing Department. *

GROUND BREAKING CEREMONIES

For the second time in less than three years, Microware has broken ground for the construction of a new building to accomodate its growing number of systems software personnel. On June 13, 1986, informal ground breaking ceremonies were held to celebrate Microware's continued growth and commitment to the future.

The 8,000 sq. ft. facility which will house the administrative and marketing departments, is located immediately north of corporate headquarters. Occupancy is planned for January 1987. The present facility will continue to be the domicile of the Research and Development staff.

Above: Microware staff members look on as Ken Kaplan digs the first shovel of dirt.

Below: A view of the new building during construction.

OS-9 SELECTED FOR EUROPEAN EDUCATION STANDARD

Three major European electronics companies including Thomson, the French electronics and defense group, Ing. C. Olivetti of Italy and Acorn Computer Group PLC, the UK educational computer company, have joined forces to develop and promote a common European standard for educational and personal microcomputers incorporating OS-9/68000.

Thomson has been searching for a major European partner to support its recent entry into the personal computer market. Thomson has maintained that it is essential to form an alliance under a European standard for personal computers. The alliance comes at a time when Thomson is increasing its commitment to the personal computer market.

Thomson's personal computer sales have been strong and are expected to top 1 billion French Francs in 1986. They have also made strong advances in the educational sector of the personal computer market and will be supplying the bulk of the 120,000 educational computers purchased this year by the French government for local schools. *

NEW FACES

Front Row (l-r): Ken Mizuno, Kim Gegner and Christy Longstaff. Back Row: Richard Russell, Robert Wolf and Paula Henderson.

Microware's continued growth is reflected in the recent addition of six new employees to the staff.

Ken Mizuno, software engineer, is a representative of Microware Japan. He is one of the original designers of the OS-9/68000 Network File Manager (NFM). Ken transferred to the United States to continue work on NFM and CD-I software tools.

Kim Gegner, receptionist, is a Des Moines native and 1983 graduate of Johnston High School, Des Moines, IA. Kim formerly worked for Norwest Bank as a research and data entry clerk.

Christy Longstaff, assistant cook, attended DMACC (Des Moines Area Community College), in Ankeny, Iowa and is a 1980 graduate of Valley High School in West Des Moines, IA.

Richard Russell, software engineer, attended Southwest Texas State University in San Marcos, and received a Bachelor's degree in Computer Science in December, 1985. During this time he became familiar with OS-9 and used his home system with Microware's 6809 C compiler to design compilers and other programs. Richard is working towards his Master's degree in Computer Science.

Robert Wolf, electronics technician, is currently working towards his A.A.S. in Hi-Tech Electronics at Des Moines Area Community College. His vocational training includes: circuit analysis, digital design and machine-language programming. Robert also developed an application package for cooperative buying clubs.

Paula Henderson, production clerk, is a Junior at the University of Northern Iowa, majoring in Psychology. Paula is assisting in the Production Department for the summer. *

OS-9/68000 FOR ATARI 520/1040 ST

OS-9's multi-tasking, multi-user power is now available for the Atari 520/1040 ST from TLM Systems, Inc.

The Atari ST line is complete with a two-button mouse, 3.5 inch disk drive, high-resolution monochrome monitor or RGB color monitor. At the heart of the ST's is the 68000 microprocessor which runs at eight MHz. Supporting the processor are four custom designed chips handling graphics, high-speed disk access, system timing and memory control. Along with the main micro-processor and the four custom chips are 512 Kbytes of RAM for the 520ST and one Mbyte of RAM for the 1040ST, 192 Kbytes of ROM and a multi-voice sound chip. The 95-key keyboard is divided into four areas: QWERTY typing area, 10 programmable function keys, editing section and four cursor control keys, plus a numeric keypad for spreadsheet, accounting and other projects.

The ST comes with an array of built-in ports. The left side houses a 128K ROM cartridge slot. The right side houses two joystick ports. The rear panel contains the power in, MIDI in, MIDI out, video, parallel printer, RS-232 serial, floppy disk and hard disk ports. The ST's printer port supports an industry standard parallel interface.

Communication with other computer systems, data bases and special function peripherals is supported by the RS232 port using Microware's COM package. Disk drives and hard disks are accessible through a high-speed DMA channel linked to both the floppy disk and hard disk ports.

TLM licensed Microware's Basic, C and Pascal compilers which are now available from TLM for this system. For further information, contact Nick Costanzo of TLM Systems, Inc., 4704 W. Jennifer, Suite #105, Fresno, CA 93711 Phone: (209)276-2333. *

PC-DOS Disk Copy Utility

The PC-DOS Disk Utility Program (PC-Disk), is a special utility program with complete facilities for the creation and manipulation of IBM's PC-DOS (MS-DOS) disks for OS-9/68000. The program permits standard five inch floppy disks to be used to transfer data either from OS-9 to PC-DOS or to OS-9 from PC-DOS. It provides a complete set of functions for accessing, creating and manipulating disks in the format used by the IBM PC and compatible computers. The program can read, write and delete subdirectories, list directory contents and format new disks. All file names and operation directives are entered via the OS-9 command line making it possible to embed the use of this program into other programs and command scripts. File input and output may be directed to and from the standard input and output files of OS-9, easing its use in command line pipe sequences.

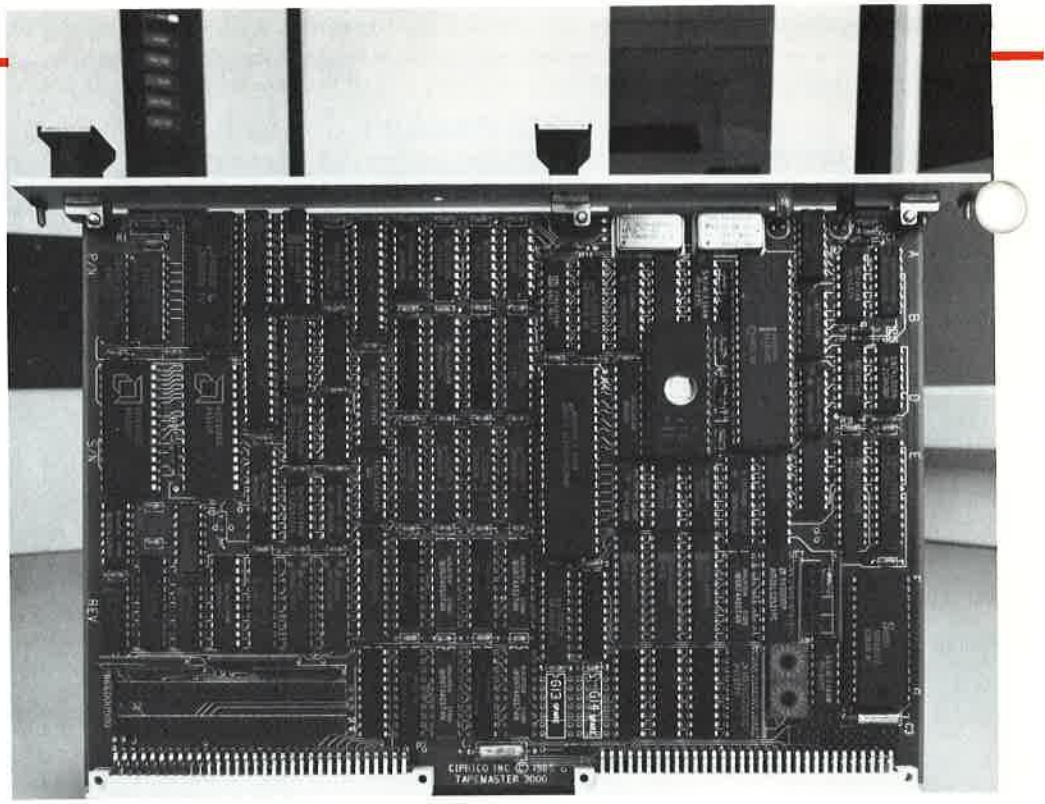
Written so that it can be used either directly from the OS-9 Shell or embedded in an applications specific program, PC-Disk offers the following capabilities:

* Initial Formatting of PC-DOS disks.
* Reading PC-DOS files.
* Writing PC-DOS files.
* Directory Listing of PC-DOS disks.
* Creation of PC-DOS subdirectories.
* Deletion of PC-DOS files.
* Deletion of PC-DOS subdirectories.

When used in conjunction with other OS-9/68000 utility programs, the program can also be used for:

* Copying between two PC-DOS disks.
* File Comparison between two PC-DOS disks.
* Hexadecimal Display of PC-DOS file contents.
* Writing all files in a PC-DOS directory to OS-9.
* Writing all files in an OS-9 directory to a PC-DOS disk.

With the exception of formatting, all operations can be performed on both the standard 40 track, double-sided, double-density, nine sector disk format commonly used by PC-DOS as well as the older eight sector format. Disks can be formatted only in the nine sectors per track format. Also, automatic double stepping is provided so that PC-DOS disks can be read or written on 80 track floppy disk drives. PC-Disk is configured for use on the Hazelwood UniQuad system. For Non-UniQuad systems, documentation on the utility/driver interface is included to assist users in making the modifications necessary in their driver. PC-Disk is supplied on a single five inch floppy disk along with a comprehensive manual that includes copious examples. For more information contact: Hazelwood Computer Systems, 907 E. Terra, O'Fallen, MO 63366, (314)281-1055. *



TAPE SUPPORT NOW AVAILABLE

Microware is now offering an archive/tape support package for OS-9/68000 that includes a Sequential Block File Manager (SBF), device drivers and backup/restore utilities.

The backup/restore utilities provide a convenient and user-friendly interface for archiving important files that can be used with either RBF (Random Block File Manager) or SBF. Full or partial directory structures can be incrementally archived to floppies, hard disk or magnetic tape. The restoration utility includes an interactive environment that allows the user to mark and restore entire directory structures and/or specific files and directories within them.

Sample VME and Multibus drivers for Ciprico Tapemaster 1000/3000 tape controllers are included to serve as models for OEM's who are adapting the archive/tape support package to a given tape controller. The Tapemaster controllers are intelligent, high performance half-inch tape drive controllers for the VME and Multibus. They can control up to eight formatted start/stop or streaming, GCR, PE or NRZI tape drives conforming to the Pertec interface standard.

The archive/tape support package is available immediately from Microware as an additional option to new or existing OEM licenses. For more information contact Andrew (Drew) Crane at Microware. Additional information on Tapemaster controllers is available from Stu Reile at Ciprico, 2800 Campus Park Drive, Suite 110, Plymouth, MN 55441 (612)559-2034. *

ON THE C SIDE

In this edition of Pipelines a continuance of helpful hints has been included to assist both the pro and novice programmers to better understand Microware's implementation of the C language. To receive a copy of the C strategies that appeared in previous editions of Pipelines, contact Phyllis Casel at Microware.

Stream I/O Buffering. The stdio functions can provide three different buffering methods: block buffering, line buffering and no buffering. Block buffering is normally used when the I/O device is a disk. This buffering method stores up a number of characters (256 or 512) and delivers them to/from the device with one I/O call. It is much more efficient to transfer full blocks than a small number of characters. Line buffering is used when communicating with a line-oriented device such as a terminal or printer. Line buffering delays the actual output until the newline characters are written to the stream. Conversely, input characters are not delivered to the program until the newline character is read from the stream. The buffering facility can also be disabled so each character is delivered to or from the device as soon as it is available.

The buffering method used by the stdio functions is determined when the first character is read or written to the stream. The standard paths are set to default conditions when the program is started. The following tables show which buffering method is used as default for the standard paths and devices:

File Mgr	BufSiz	I/O Function	Buffering Method
RBF	512	write/read	Block buffering
non-RBF	256	writeln/readln	Line buffering

Stream	Open For	Buffering Method
stdin	reading	Line
stdout	writing	Block if RBF, Line if not
stderr	writing	Block if RBF, Line if not

The buffering scheme was chosen because it provides the maximum amount of data possible to the device using the minimum number of I/O system calls. The buffering method performed on a stream can be altered by simply changing the stream flags before I/O is performed. For example, to prevent the stderr stream from buffering its output, do the following before writing to stderr:

```
stderr->_flag |= _UNBUF | _RBF;
```

The __UNBUF flag causes the stream to be unbuffered. The __RBF flag causes the write() function to be used for output. The same action applied to the stdin stream causes the characters to be delivered to the program immediately.

Stream File Update Mode. When using a stream in update mode (see the fopen() "+" action), care must be taken to ensure an fseek() is called when the direction of data transfer changes. The effect of the fseek() is to cause the buffer to be flushed or filled so data can be transferred in the opposite direction. This action is required even though the position in the file does not change. For example, to read a record and then write the subsequent record:

```
fread(buf, sizeof buf, 1, fp);  /* input data from file */
fseek(fp, 0, 1);  /*change to writing */
fwrite(nbuf, sizeeof nbuf, 1, fp);  /* write something else */
```

Keep in mind that the record-locking facilities of RBF are not entirely effective on files accessed via the stream functions. Only the RBF end-of-file lock works in this case. The low-level I/O functions must be used for the RBF record-locking routines to be fully effective.

Interactive Stream I/O. Interactive I/O is best performed on the standard input and output streams. These streams are line buffered when referring to a terminal and are tuned for interactive I/O. An input loop can be coded as follows:

```
#include <stdio.h>
main()
{
    char buf[BUFSIZ];
    while (puts("Enter data:"),gets(buf)) {
            printf("The data you entered was: %s\n",buf);
    }
}
```

The while loop contains two function calls, one for the prompt and one to read the input data. Note the usage of the comma operator. In this case the puts() function is called, its value discarded, the gets() function is called and its value used as the test condition for the while. When EOF is reached on the input stream (caused by entering an escape on the terminal), the loop will terminate.

The puts() and gets() functions, by definition use the stdout and stdin streams, respectively. The stdout stream is automatically flushed before input is performed on stdin. This is so any prompt appears before the input request. An alternative approach for interactive I/O is to use a single stream such as stderr. To do this you must set the __READ flag for stderr (as described above), and remember to do an fseek() or fflush() when the direction of data transfer changes. *

PORTING 6809 BASIC09 TO 68000 BASIC

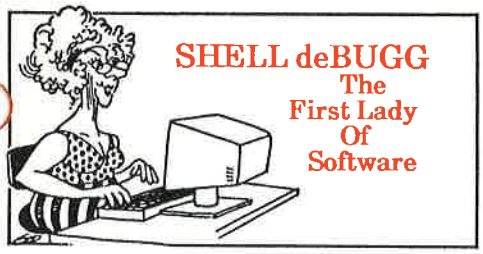
The following Basic09 code can be useful in porting 6809 Basic09 (referred to as 09) software to 68K Basic (referred to as 68K). Since 09 integers are two bytes and 68K integers are four bytes, 09 data files can not be directly used by 68K. Integers read in from the 09 data file under 68K will not be valid since 68K expects a four byte integer. So, when reading integers from 09 data files one must account for the differences in the integer type. This program has been published by the PIPELINES staff, not the support service staff; therefore, ON ERROR do not GOTO Microware's Support Service Hotline.

The conversion can be done as follows:

```
TYPE short=highbyte, lowbyte:BYTE
DIM o9int:short
DIM oskint:INTEGER
DIM path:BYTE

oskint=0
(* read the integer *)
GET #path,o9int

(* do the conversion *)
oskint=o9int.highbyte*256+o9int.lowbyte

(* sign extend the integer if necessary *)
IF LAND(o9int.highbyte,$80)<>0 THEN
 oskint=LOR(oskint,$FFFF0000)
ENDIF
```

Conversely, 68K Basic integers can be converted and written as 6809 Basic09 integers in the following manner: (Note: this is only valid for integers that are between +-32k.)

```
TYPE short=highbyte, lowbyte:BYTE
DIM o9int:short
DIM oskint:INTEGER
DIM path:BYTE

(* get the low byte *)
o9int.lowbyte=oskint
(* move the high byte down *)
oskint=LAND(oskint,$0000FF00)
oskint=oskint/256
(* get the high byte *)
o9int.highbyte=oskint

PUT #path, o9int
```



Dear Shell,

I'll bet you thought you could slip a change in Version 1.2's F$Fork system call past me, and I wouldn't notice. Think again, lady! I noticed that the OS-9/68000 F$Fork system call has changed in a way that is not documented in the Technical Manual. The length of the parameter string (plus one for the end-of-line character), is not the same as it was in Version 1.1. Now, it seems the parameter length is 14 if there are no parameters, or it is equal to the number of characters passed plus 16 or 17 depending on whether or not the parameter string passed has an even number of characters. The fact that the parameter length is no longer passed in register d5 makes handling the parameter area more difficult in assembly language. What is the reason for this change?

-All Worked Up

Dear All Worked Up,

The F$Fork system call has not changed for Version 1.2 of the OS-9/68000 Operating System. The F$Fork and F$Chain system calls simply copy the parameter string of the given length into the parameter area of the new process.

OS-9 treats the parameter area as a sequence of bytes of a specified length. A program is free to pass anything to a new process, as long as the program and the new process agree on the format of the data.

The observed change is actually a change in the version 1.2 **SHELL** and not the F$Fork system call. The **SHELL**, which is written in C, uses the os9exec() C library call to preprocess the arguments that are passed to the new process via the os9fork() C library call. The os9exec() function builds a data structure that, when processed by a C program's startup routine (c.start.r), will yield an ArgV pointer list suitable for passing to the main() function of a C program.

In addition to handling the arguments, the os9exec() function also passes the "enviroment" maintained by **SHELL**. The enviroment consists of strings normally maintained by **SHELL** that a program can access via the getenv() library function. The environment is useful for setting various default cases for program operation.

These changes were documented in the Version 1.2 and 1.3 C Compiler release. The C language enviroment is simply exercising it's ability to pass, through the F$Fork call, a parameter list of it's own choosing. The OS-9/68000 Version 1.2 utilities use the os9exec() function and an enhanced cstart.r to utilize this extended argument passing procedure. This implementation is upwardly compatible with the previous method of passing arguments, although the environment variable facility will not be available unless the program is changed to use the os9exec() function and relinked with the new cstart.r.

The proper technique to handle the argument list is documented in the cstart.r source code (supplied with the C compiler). Assembly language programs which are started by **SHELL** and handle arguments should use the cstart.r method to access the arguments.

Dear Ms. deBUGG,

For the past year I have been under the supervision of a respected psychiatrist whom I felt I could trust with my darkest secrets. Recently I told him of my fantasy concerning data modules. I've always wanted to use them, but was unsure of what to do. Now I've found that my psychiatrist has betrayed my trust and told others about my fantasy. They look at me like I am crazy. Since I can no longer trust him, I need your advice. How can data in a data module be accessed?

- Embarrassed

Dear Embarrassed,

It's a very traumatic experience when one's secrets have been revealed by a trusted confidant. It is perfectly normal to want to use data modules. Data modules are used to store global data and configuration information for user-level programs. The F$DatMod System Request creates a data module with the specified attribute/revision and clears the data portion of the module. The module is initially created with a valid CRC, and entered into the system module directory. Several processes can communicate with each other using a shared data module. Care should be taken not to alter the data module's header and name string to avoid the possibility of the module becoming unknown to the system. Provided that the data module is already in the module directory, it can be accessed from an assembly language program by using the F$Link system call. From a C language program, use the modlink() function. *

* * * * * * * * * * * * * * * * * * *

Send your OS-9 questions to:

Microware Systems Corporation
c/o SHELL deBUGG
1866 N.W. 114th Street
Des Moines, IA 50322

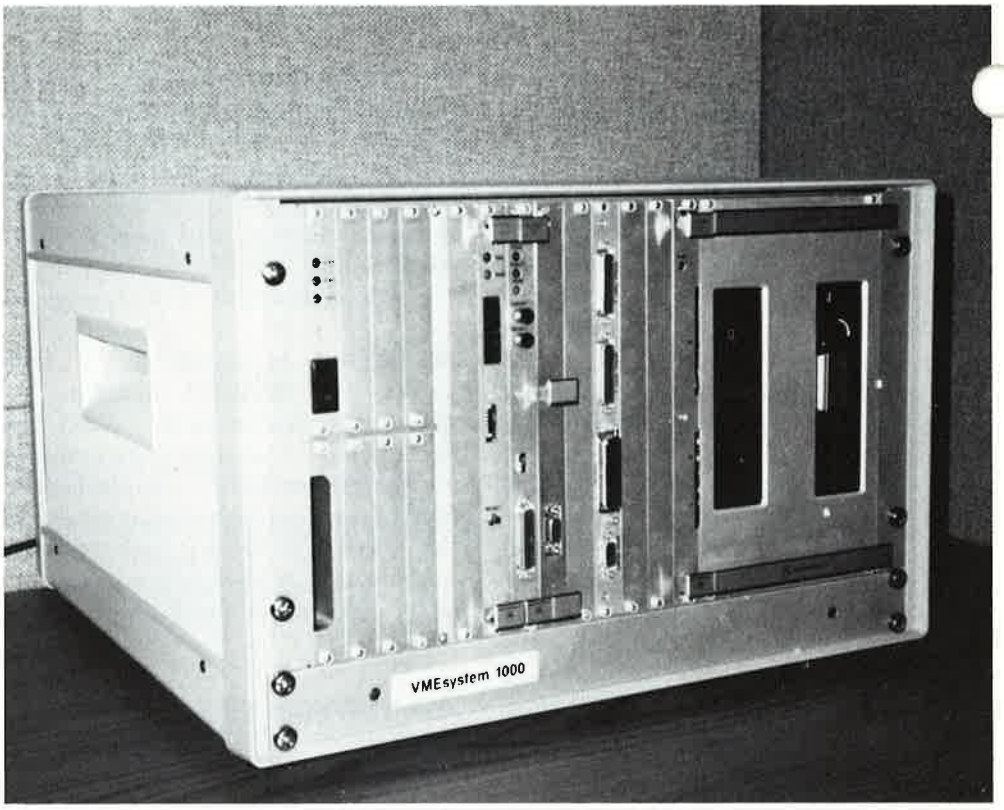
* * * * * * * * * * * * * * * * * * *

OS-9 SEMINAR GOES ON-THE-ROAD

Microware's Annual OS-9 Seminar will be going on the road in an effort to reach more OS-9 users. Dr. Bill Moore, Microware's Training and Education Director, will be holding three regional training seminars throughout the United States in the next 12 months. Bill's seminars will focus on I/O interfaces, programming languages, the latest developments in the OS-9 world and specific interests of OS-9 users. For more information, or to register your interest in bringing the OS-9 training sessions to your area, please contact Bill Moore at Microware. The time and location of the first conference will be announced in the near future. *

MICROWARE ATTENDING BUSCON/86 - EAST

Microware will be exhibiting at Buscon/86 - East in Boston, MA October 1st and 2nd. Manufacturers of buses, boards, controllers and surface mounted devices will be displaying their latest technology. Andrew Crane, Mary Marturello and Bill Moore will be in Microware's booth #208, they invite you to stop by and say hello. *



SYSTEM 1000 DEVELOPMENT PAK

Microware is now offering a Development Pak for the System 1000, a full height VME-Bus development system from Motorola. The System 1000 Development Pak supports the 121 CPU, 050 system controller and 320 floppy/hard disk controller. The Development Pak is configured to boot from a 5 inch 80-track floppy diskette and includes a hard disk driver and sample descriptor for user installation. System 1000 was on display in the Microware booth at N.C.C. 86 in Las Vegas. The Development Pak is immediately available from Microware and authorized Microware distributors. *

MICROWARE®

Microware Systems Corporation
1866 N.W. 114th Street
Des Moines, Iowa 50322

Bulk Rate
U.S. Postage
Paid
Des Moines, IA
Permit #2864