Volume 3 Number 2

Summer/Fall 1988

microware®

PIPELINES

Covering the Full Spectrum of OS-9 News and Applications

SPECIAL "BRIDGE" ISSUE:

ALSO IN THIS ISSUE:

PIPELINES

Summer/Fall 1988
Volume 3 Number 2

Pipelines is published quarterly by:
Microware Systems Corporation
1900 N.W. 114th Street
Des Moines, Iowa 50322
515-224-1929

Editor:
David F. Davis

Contributors:
Dave West, Richard Russell, Ric Yeates, Jim Guisinger, Andy Ball, James Jones, Bruce J. Brunette (Introl Corporation)

Photography
David F. Davis

Art Director
Tom West

A NOTE TO OUR READERS

As with every issue of **Pipelines**, we would like to solicit your comments, suggestions and thoughts. Also, we invite you to send for publication any interesting OS-9 application articles or useful utility programs. Please address all correspondence to the Editor of **Pipelines**, c/o Microware Systems Corporation. All submissions must include your name, address and telephone number.

NEW OS-9 SOURCE BOOK NOW AVAILABLE

Good news for all OS-9 users! The 1988 Edition of the **OS-9 Sourcebook** is now available. This comprehensive publication lists manufacturers of third-party hardware and software products supporting OS-9. Included in the sourcebook are references for turnkey systems and board-level products, application and development software and OS-9 User's Group/Public Domain software. To receive your free copy of the sourcebook, contact Microware today.

MICROWARE BUILDS "BRIDGES" TO

The theme of this issue of Pipelines is "bridges". While our new UniBridge and PCBridge products have been received with considerable enthusiasm by the OS-9 community, they have much more significance than may be apparent.

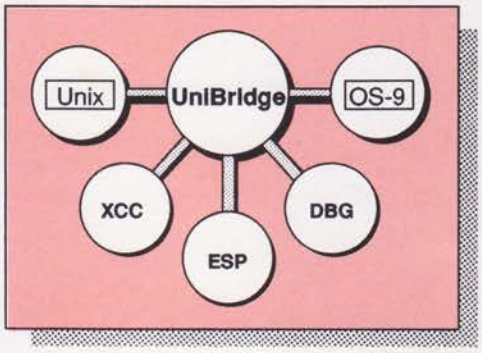
Although OS-9 is unique in that it excels in both development and target system roles, some users prefer a development environment based on UNIX or PC-DOS due to equipment on hand or other tools connected with their project. UniBridge and PCBridge make this a viable option.

But consider the bigger picture. There are de facto standard operating systems for desktop computers and engineering workstations in DOS and UNIX, respectively. Regardless of how you or I feel about these standards, they are a fact of life. But there is a missing third standard which still hasn't been established: a standard real-time operating systems. Our plan is to make OS-9 that standard in the coming years. In order to do so, we have to build effective bridges to other standard operating systems, other processors, and other stan-

OS-9 UniBridge Connects UNIX to Real Time

Microware proudly announces the availability of OS-9 UniBridge... The complete UNIX to real-time connection. UniBridge is an advanced software package for C language development and Ethernet communication that connects UNIX to OS-9. With UniBridge VMEbus system integrators and designers can now develop real-time applications using popular UNIX-based workstations. This allows OS-9 systems to be used with popular SUN, DEC VAX, HP and Motorola UNIX workstations for distributed development and real-time supervisory control.

Software engineers can use UniBridge to connect the rich development environment of UNIX to the powerful real-time capabilities of OS-9. The UniBridge package contains all of the sophisticated tools needed to make the connection between a UNIX host and OS-9 target. UniBridge includes:



UniBridge...The Complete UNIX to Real-Time Connection

OS-9/XCC UniBridge contains both UNIX and OS-9 resident C Compilers for 68000 or 68020 microprocessors with full 68881 support. Users can compile on the host or target to produce compact, re-entrant, position independent object code for real-time execution. Since the OS-9 C Compiler utilizes UNIX C standard libraries, C programs can be compiled with OS-9/XCC to operate on OS-9 resident systems without program conversion. UNIX standard libraries also allow OS-9 C programs to be easily ported to the host environment.

UNIBRIDGE
Please turn to Page Three

OTHER DEVELOPMENT PLATFORMS

dards for communications, applications interfaces, and so on.

UniBridge, PC-Bridge and our Ethernet software products are the first steps in this direction. And you'll be seeing much more from us over the next couple of years in the areas of kernels, graphics, and networking.

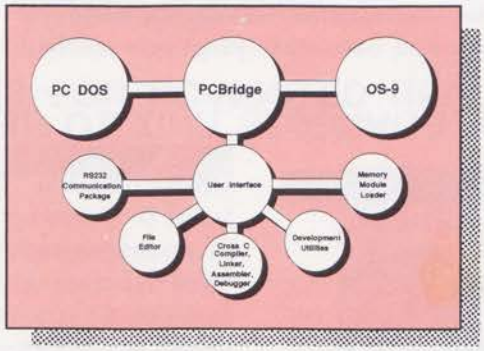
Having a better mousetrap alone is not enough. One thing we need in order to make OS-9 the undisputed standard real time operating system is your enthusiastic support. This is the main reason for the incredible momentum OS-9 has accumulated in the last couple of years. Every time we look into the subject of "how do people find out about OS-9?" The number one answer is always "personal referral from friends or colleagues". So spread the word, write the magazines, and keep the networks buzzing! It makes a difference and we really appreciate it.

—Ken Kaplan
President,
Microware

PCBridge Links PC's to Real Time

PCBridge is an easy-to-use PC-hosted development system for OS-9/680x0 applications. Through PCBridge, MS-DOS users gain access to the OS-9 Operating System. PCBridge provides a C cross compiler, assembler and linker, and a set of program development utilities. These utilities include terminal emulation, text and binary file transfers between MS-DOS and OS-9, file manipulation utilities and session logging. The development utilities are distributed between the host and target systems.

PCBridge provides a platform for distributed applications, building synergistically upon the real-time aspects of OS-9 and an easy-to-use PCBridge interface under MS-DOS. For example, OS-9 can be used for such tasks as real-time data acquisition, image processing, factory and robotics control. PCBridge can link these types of systems for process monitoring, status and control to information management applications on PC-DOS. A total distributed application can be developed and integrated using PCBridge. The system uses a front-end PC/XT/AT which is linked to the target OS-9 system via a high-speed serial line. The user interface is pop up menu-driven, with the user selecting a function (Edit, Compile program, load memory module, etc.) indicating necessary parameters, and the PC and OS-9 systems cooperate to perform the operation without further user intervention. Selections are made via keyword, keypad cursor keys, or a mouse.



PCBridge...A PC-Hosted Development System for OS-9/680x0

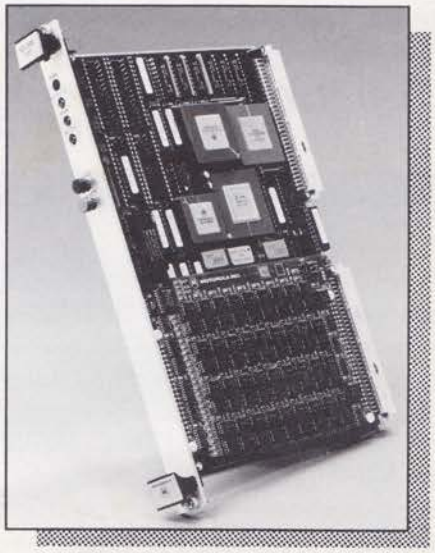
PCBRIDGE
Please turn to Page Four

New Development Pak From Microware

A new Development Pak for the Motorola VME 147 6U single board computer is now available from Microware. The new Dev Pak provides full support for the MVME 147's MC68030 microprocessor and MC68882 floating point coprocessor operating at 20 or 25 MHz. Because of many on-board features, the MVME 147 is capable of replacing up to five standard VME modules with an substantial increase in performance.

Microware's MVME 147 Dev Pak features an "030" optimized kernel and complete support for the MC68882 FPCP. Serial and parallel I/O is handled through 4 RS-232C serial ports and one Centronics parallel printer port. Hard and flexible disk support is provided by the OMTI 5400 controller. On-board Ethernet support will be available in an MVME 147 Dev Pak update scheduled for 4-quarter release.

As with all Microware Development Paks, the MVME 147 Dev Pak includes a C Compiler, Assembler, Linker, Debugger, Microware Basic, a Screen Editor, complete documentation and free 90-day "Hotline" support. Contact Microware or an authorized Microware distributor today.



The MVME 147 Single Board Computer

Questions and Answers about UniBridge From the Design Team

To help people understand the use of UniBridge, the UniBridge Design Team has assembled this information to address the most asked questions.

Q In what type of environments and for what applications would UniBridge be best suited?

A UniBridge was designed for real-time applications that require UNIX-based software development and supervisory control. UniBridge provides UNIX programmers with an interactive gateway to OS-9 based real-time applications. Because UniBridge also includes resident C Language development tools, users have the option to unlink from the UNIX host and develop or maintain code directly on the OS-9 target.

Q What are the software/hardware requirements for UniBridge?

A UniBridge operates on any OS-9/ 680x0 VMEbus system supported by Version 2.2. Either Industrial OS-9 w/RBF, Personal or Professional OS-9 may be used. The ESP module requires the CMC Ethernet Node Processor and a standard Ethernet LAN network. The UNIX host must be either a Sun 3 (SUN/OS), DEC VAX (BSD 4.2), HP 9000 S 300 (HP/UX) or Motorola Delta series (UNIX System V/68).

The UniBridge Design Team

Sitting (l-r): Andy Nicholson, Steve Guntly, Richard Russell, Kathie Flood and Rob Beaver. Standing: Kim Kempf and Steve Johnson

Q What is UniBug?

A UniBug is the UniBridge version of the C Source Level Debugger. This special version allows programs to be debugged from the UNIX host (via Telnet) while they are executed on the OS-9 target system. Its operation is almost identical to the C Source Level Debugger.

Q How does the System State Debugger work?

A The System State Debugger can be used for debugging both system and user state programs resident on the OS-9 host/target. SysDbg is a very powerful tool for testing and debugging resident target system hardware device drivers and device descriptors. The debugger is only operational when the OS-9 target is disconnected from UNIX host.

Q On what media is UniBridge distributed?

A Cartridge or 9-track tape is provided for the UNIX host, and 3 1/2" or 5 1/4" OS-9 formatted diskettes are used on the OS-9 target.

UNIBRIDGE CONNECTS UNIX TO REAL TIME

UNIBRIDGE

Continued from Page One

OS-9/ESP The OS-9/ESP Ethernet Support Package provides complete Ethernet TCP/IP communication between the host environment and OS-9 based systems. OS-9/ESP incorporates both FTP and Telnet protocols for efficient file transfer and remote login capabilities. Users can easily access OS-9 from UNIX, UNIX from OS-9 and OS-9 from OS-9 for distributed software development and supervisory real-time execution. OS-9/ESP features a C-compatible Berkeley 4.2 socket library combined into an Internet database as a single OS-9 data module.

OS-9/SRCDBG OS-9/SRCDBG combines both a full-featured C Source Level Debugger and System State Debugger to provide a rich environment for testing and debugging OS-9 C Language programs. The C Source level Debugger features a C expression interpreter and an extensive command set which allows the user to debug any OS-9 C program at the source level. Users have the ability to invoke debugger control and communication, data manipulation and system commands to significantly decrease software development time.

UniBridge comes with complete professional documentation and free 90-day "Hotline" support. For additional information on UniBridge, contact Microware today .

REMODELED PRODUCTION FACILITY SPEEDS ORDER PROCESSING

This summer marks the completion of an extensive remodeling project in the Microware Production and Warehouse facility. Microware's steady growth necessitated the upgrade to the Production Department to effectively meet the needs of our customers.

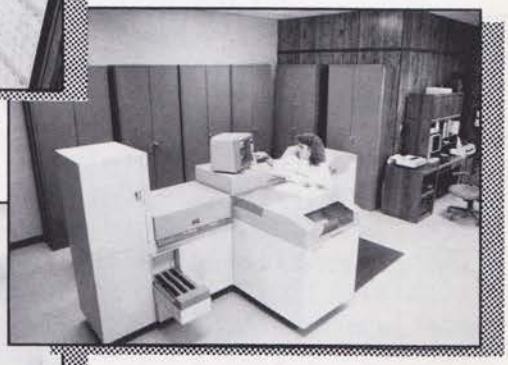
The facility now features a revised floor plan to better facilitate material handling, thus improving order processing and shipping efficiency. A state-of-the-art disk duplicating machine speeds software production and eliminates many errors associated with software reproduction. The production staff has also been increased to help guarantee that most standard orders, as well as special orders, will be shipped within 24 hours.

Microware's Production staff is proud of and excited about their redesigned facility. Jim Guisinger, Manager of Production, summed up his staff's attitude when he stated, "Our revamped Production Department is a reflection of Microware's continuing commitment to provide customers with first-class service and defect-free merchandise. We are striving to make sure that all Microware customers receive a quality product in a timely manner without hassles or delays."

The Microware Production staff (l-r): Lori Lockard, Jim Guisinger, Pete Burns and Heather Hall.

The new disk duplicator cuts software reproduction time.

A revised floor plan makes more efficient use of work space.



The Xerox 9900 copier is used for in-house publishing of data sheets and manuals.

The Production Department has a complete bindery and storage space for all Microware literature.

PCBRIDGE PROVIDES FOR PC-HOSTED REAL-TIME DEVELOPMENT

PCBRIDGE

Continued from Page One

The target OS-9 system can be almost any configuration, including ROM-based with limited RAM and no disk. Applications can be developed on the PC and loaded into the target system across a high-speed serial line for execution or testing.

PCBridge is distributed on either 3 1/2" or 5 1/4" diskettes, and come with complete professional documentation and free 90-days "Hotline" support. For additional information on PCBridge, contact Microware today.

PCBridge operates on IBM and IBM-compatible hardware

UPDATES FROM MICROWARE

C Source Level Debugger, FORTRAN, ESP

MICROWARE UPDATE POLICY

Microware offers customers who purchased Version 1.0 of any end-user software product an update to the next version free of charge. Contact Microware within 90-days of a new version's release for complete update information.

C SOURCE LEVEL DEBUGGER

A new release of the Source Level Debugger is now available. Version 2.0 has been optimized for even higher performance and includes many new powerful features. These features include: Debugging multiple module programs (i.e. trap handlers and subroutine modules); Assembly level debugging; Complete access to processor registers in C Language expressions; conditional break points, break counts and command scripts; and complete stack backtrace capabilities. All modifications made in Version 2.0 have been fully documented in release notes included with the updated software.

The complete list of Version 2.0 C Source Level Debugger Commands:

```
asm(.)      a[ssign]   b[reak]      c[hange]
c[h]c       c[h]d      c[h]x        con[text]
di[asm]     dil[ist]   d[ump]       fi[nd]
fo[rk]      f[rame]    g[o]         gostop(gs)
i[nfo]      k[ill]     li[nk]       l[ist]
lo[cals]    l[o]g      mf[ill]      ms[earch]
n[ext]      o[ption]   p[rint]      re[ad]
r[eturn]    se[tenv]   shell($)     s[tep]
sy[mbol]    t[race]    unse[tenv]   w[atch]
```

New C Source Level Debugger Commands added to Version 2.0:

```
b[reak] [<location_expr>] [:wh[en] <C_expr>]
[:co[unt] <num>]
```

The user may set conditional break points, break counts and breaks at source or assembly language locations.

```
c[h]c [<scope_expression>]
```

This enables the user to set break points etc. without the use of scope/line number expressions.

```
c[h]x <pathlist>
```

Changes the current execution directory for SrcDbg.

```
con[text] [<scope_expression>]
```

Fully qualifies a symbol in terms of scope. Informs the user exactly which symbol is going to be referenced in an expression.

```
fi[nd] [<name>]
```

Displays all scope expressions found for <name>. Informs user of all occurrences of a name.

```
f[rame] [[+ | -] <number>]
```

Changes stack frame to <number>. Frame with no arguments displays current call stack frame information. User now can access local variables in the function call stack.

```
g[o] [<location_expr>] [:dis[play]]
```

Go now provides a way to run the program to a certain spot without the user setting and removing a break point.

```
l[o]g <pathlist> | : off
```

Writes SrcDbg commands to <pathlist>. ": off", closes the log file. The user can now save a series of commands and re-execute them at a later time.

```
lo[cals]
```

Displays the values of all local symbols. This provides a quick way of referencing local variables.

```
o[ption] { <options>}
```

Options:

fpu	toggle fpu register display.
fregs	toggle fpu display between hex and decimal.
rom	toggle rom (soft) and ram (hard) breakpoints.
source	toggle source display during assembly level displays.
watch	toggle location display after watch expression changes.
dbg	toggle reading of ".dbg" files.
stb	toggle reading of ".stb" files.
prompt	toggle prompt output.
echo	toggle command line output.

These provide the user with greater control of the source debugger and its displays.

```
re[ad] [<pathlist>]
```

Reads SrcDbg commands from <pathlist> and enables the debugger to read command scripts. These may be created by the user with an editor or with the "log" command.

```
se[tenv] <environment_name>
<environment_definition>
```

Sets a shell type environment variable. The user can now change the environment for the use of the debugged program or the debuggers' environment itself.

UPDATES FROM MICROWARE

C Source Level Debugger, FORTRAN, ESP

`sy[mbol] [<C_expr>]`

Displays the result of the expression as a symbolic expression. This command is useful in showing what symbol a pointer variable is referencing.

`unse[tenv] <environment_name>`

Deletes environment variable. Provides further control over the environment.

New Assembly Level Commands:

`c[hange] [<C_expr>]`

This command provides an easy way to change byte(s), word(s), or longword(s) in memory.

`gostop | gs[<number>]`

Executes <number> of machine instructions in the current subroutine. Similar to the "next" command but at assembly level.

`li[nk] <module_name>`

Links to <module_name> and places module address in ".r7". A user can load/link a memory module and then use ".r7" in C Language expressions to access it.

`dil[ist] [<location_expr>][: [<count>]]`

Displays C source with disassembly. This gives the user a way to see the assembly code that is mapped to their C language code.

`di[sasm] [ [<C_expr>] [: [<count>]] ]`

Disassembles memory at the result of <C_expr>. Provides a means to display assembly code.

`d[ump] [ [<C_expr>] [: [<count>]`
`[<format>]] ]`

Displays memory at the result of <C_expr>. This formatted memory dump command has user controls on the display format.

`mf[ill] <begin> : <end> : <value>`

Fills memory with <value>. This command provides an easy way to fill memory with a desired bit pattern.

`ms[earch] <begin> : <end> : <value>`
`[: <mask>]`

Searches memory for <value>. Provides an easy way to search memory for a desired bit pattern.

`t[race] [<number>]`

Provides a way to step through assembly language code an instruction at a time.

OS-9/68000 FORTRAN 77 COMPILER VERSION 1.2

Microware has released FORTRAN 77, Version 1.2. This new edition update includes corrections for known problems in the "fort" executive and both compiler phases, "fortp1" and "fortp2". In addition, the fortp2 user error messages are now displayed with more meaningful descriptions. For example the error message "can't write temporary file" will be displayed when -t=/r0 is used and the RAM disk fills up. The D floating point notation format is now also supported. All modifications made in Version 1.2 have been fully documented in release notes included with the updated software. The Fortran update is available for both 68000 and 68020 target systems.

ETHERNET SUPPORT PACKAGE (ESP) - VERSION 1.1

Microware announces the release of ESP Version 1.1. This new edition update incorporates bug fixes and enhancements to improve the reliability and performance of the ESP software.

Two new header files appear in the DEFS directory. `errno.h` is identical to the `errno.h` supplied with the 3.0 C Compiler. It includes the error number definitions for the socket errors. `sgstat.h` is the get/setstat struct file and contains additional definitions for the get/setstat options call to the ENP10 driver.

The ETC directory includes a new error message file "errmsg.short" which incorporates the error messages, mentioned in the `errno.h` description, into a file to use as the "/dd/sys/errmsg" file on the OS-9 target system.

A file "enp.gate" describes how to change the "enp.a" device descriptor to direct packets to a gateway machine for internet routing. This can be found in the ENP directory.

The following hardware and software is minimally require to install and run OS-9/ESP:

1. ENP-10+ Board with V4.1 K1 Kernel ROMs
2. OS-9 System running V2.2 or later
3. Ethernet LAN system

For complete list of all changes made in this edition update, please refer to the full release notes provided on the shipment floppy.

ON THE C SIDE

The _ss_ssig() C Library Function

The C Compiler's _ss_sig() function is a useful way to watch for data without wasting CPU time by polling. The _ss_sig() function sets up a signal to be sent to the calling process when an interactive device has data ready.

This program prints a start message and then accepts characters as input. Please note that you must have OS-9/680x0 Version 2.1 or 2.2. For more complete understanding of what's happening, read the documentation for intercept(), _ss_ssig(), _gs_rdy(), _ss_rel(), and F$SigMask.

If the keyboard quit character is typed, the quit signal is received and the program exits. A keyboard abort (^C) causes the program to print a message and then go back to sleep. Text characters are read and printed.

After processing the keyboard abort, special processing is necessary (this processing would also be necessary for quit if we weren't going to exit). The keyboard abort character must be read and the dataready flag is cleared. Whenever the keyboard abort or quit characters are pressed, two signals are sent - the driver's interrupt service routine sends the abort or quit signal, and the data ready signal. Because the signals are queued, both signals are processed before our program actually executes again. Thus, keyboard quit or keyboard abort sets the dataready flag as well as the abort or quit flag.

Please note the use of setsigmask(). This allows signals to be blocked until we are ready to process them. The sleep() call clears the signal mask, ensuring that no signals occur too early.

```
#include <stdio.h>
#include <errno.h>

#define TRUE 1
#define FALSE 0

#define SSIGVAL 20 /* a signal code - don't confuse it with others */

/* global variables */
int kabort, quit, dataready;

/************* sighand: signal handling routine ************* /
sighand(sigcode)
register int sigcode;
{
    switch (sigcode) {
        case 2:
            quit = TRUE:
            break;
        case 3:
            kabort = TRUE;
            break;
        case SSIGVAL:
            dataready = TRUE;
            break;
        default:
            /* ignore others */
            break;
    }
} /* end of sighand */

/************* main: main routine ************* /
main(argc, argv, envp)
int argc;
char **argv, **envp;
{
    register int count;
    char buf[100];
```

ON THE C SIDE

The _ss_ssig() C Library Function

(Continued)

```
    intercept(sighand); /* set up signal handler first and foremost */
    kabort = quit = dataready = FALSE;
    printf("SSig example, ^E to quit\n");

    /* loop where we wait for data to come in and then read it */
    while (TRUE) {
        if (setsigmask(1) == -1) {  /* so we don't get signals before we're ready */
            exit(_errmsg(errno, "can't set signal mask - "));
        }
        if (_ss_ssig(0, SSIGVAL == -1) {  /* set up signal on stdin */
            exit(_errmsg(errno, "can't set up signal - "));
        }
        sleep(0); /* sleep until signal */
        if (quit) exit(0); /* what kind of signal */
        if (kabort) {
            kabort = FALSE;
            dataready = FALSE;
            printf("got your abort, going back to sleep\n");
            if (read(0, buf, 1) != 1) {  /* read ^c */
                exit(_errmsg(errno, "can't get abort char - "));
            }
        } else if (dataready) {
            dataready = FALSE;
            if ((count = _gs_rdy(0)) == -1) {  /* oh, oh! */
                exit(_errmsg(errno, "no data! - "));
            } else {  /* read data */
                printf("reading %d characters\n", count);
                /* assume less data than buffer will hold */
                if ((count = read(0, buf, count)) < 0) {
                    exit(_errmsg(errno, "error reading - "));
                } else if (count == 0) {
                    printf("eof encountered, exiting program\n");
                    exit(0);
                } else {
                    buf[count] = NULL;
                    printf("\ndata read = '%s'\n", buf);
                }
            }
        }
    } /* end of while */
} /* end of main */

/************ setsigmask: set the signal mask *************/
#asm
setsigmask move.l d0, d1 put new mask value in proper place
    clr.l d0 must be zero
    os9 F$SigMask do it
    bcc.s good
bad move.l d1,errno(a6) save error code
    move.l #-1,d0 set error return
    bra.s done
good clr.l d0 everything is ok
done rts
#endasm
/* end of setsigmask */
```

OS-9/68000 VERSION 2.2
EDITION NUMBERS

Microware provides this edition number information to assist OS-9 users in determining whether or not they have the latest edition of a software product.

You can use the ident utility to check your software against this list. If you don't have the latest editions, call your system manufacturer or Microware for information on updating your software.

OS-9/68000 VERSION 2.2

Operating System Modules

Module	Edition
kernel	#40
rbf	#55
scf	#26
pipeman	#29
nfm	#41
sbf	#4
ssm851	#21
ssm451	#25

System Utilities:

Module	Edition
attr	#19
backup	#13
binex	#15
build	#13
cfp	#20
cmp	#13
code	#13
compress	#13
copy	#23
count	#13
date	#16
dcheck	#17
deiniz	#13
del	#14
deldir	#17
dir	#25
dsave	#23
dump	#19
echo	#15
exbin	#16
expand	#14
fixmod	#13
format	#9
free	#13

System Utilities (continued):

Module	Edition
frestore	#8
fsave	#9
grep	#14
help	#2
ident	#16
iniz	#13
l68	#53
link	#13
list	#13
load	#14
login	#21
makdir	#13
make	#23
mdir	#17
merge	#13
mfree	#13
ndir	#9
nmon	#4
os9gen	#17
pd	#14
pr	#20
printenv	#4
procs	#19
qsort	#16
r68	#55
r68020	#78
rdump	#11
rename	#19
save	#13
setime	#20
shell	#47
sleep	#11
tape	#4
tee	#10
tmode	#17
touch	#11
tr	#16
tsmon	#15
unlink	#13
xmode	#14

C Compiler:

Module	Edition
cc	#36
cpp	#30
c68	#214
c68020	#214
o68	#19

Fortran 77 Compiler:

Module	Edition
fort	#21
fortp1	#24
fortp2	#159
fortp220	#159

Microware Basic:

Module	Edition
Basic	#22
RunB	#22

Pascal Compiler:

Module	Edition
pp68	#12
pc68	#12
pt68	#12

Trap Handlers:

Module	Editoon
cio	#6
cio020	#6
math	#13
math881	#6

Programming Tools:

Module	Editon
com	#7
debug	#35
mail	#11
querymail	#10
spl	#20
splman	#20
splprt	#20
srcdbg	#24
sysdbg	#54

Editors:

Module	Edition
edt	#13
maketerm	#8
scred	#58
uMacs	#16

NEW EMPLOYEES AT MICROWARE

Microware is on the move! Here are the latest additions to our family.

Rob Beaver, Technical Support Technician, has a background in C and Assembly Language programming. Rob previously worked at NCR in Dayton, Ohio where he was also involved in Technical Support.

Steve McClellan, Software Engineer, has a degree in Computer Information Systems from Drake University. He spent the last five years working for a Des Moines-based communications company. Steve is now part of the Microware CD-I team.

Production Assistant, Heather Hall, is a graduate of the University of Kansas where she received a Bachelors degree in Journalism. Heather's duties include shipping, receiving and software duplication.

Rick Weiss, Software Engineer, has been doing Assembly Language programming for the last five years primarily in the field of audio synthesizers. He is currently part of the Microware CD-I team.

Pictured (l-r): Rob Beaver, Steve McClellan, Heather Hall, Rick Weiss, Don Sherinian, Pete Burns and Kristi Kramersmeier

Don Sherinian, Director of Corporate Development, is in charge of corporate planning. Don has worked with Microware in the past as a consultant in the area of real estate planning.

Pete Burns is Microware's other new **Production Assistant.** He has previously worked as a social worker in the Des Moines area. Pete's hobbies include playing guitar, writing and motocross.

Account Administrator, Kristi Kramersmeier is a recent graduate of Buena Vista College. Kristi majored in Marketing with a minor in Japanese. Her computer experience includes exposure to Basic, Cobol and Pascal.

ADD SCSI SUPPORT TO YOUR OS-9 SYSTEM WITH THE INTROL 300 VME/SCSI ADAPTOR

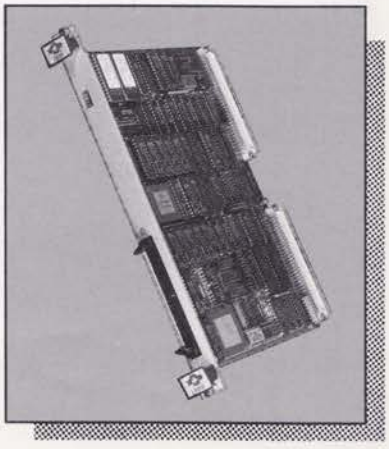
OS-9 users can now add in SCSI disk and tape support to their VMEbus systems by using the Introl 300 VME/SCSI adapter from the Introl Corporation of St. Paul, MN. The Introl 300 controller is delivered with an OS-9 device driver that allows users to plug up to 7 SCSI peripherals into one adapter.

The Introl 300 was designed for high-performance applications and supports synchronous SCSI devices up to 4 Mbyte/sec., asynchronous rates up to 1.5 Mbyte/sec. and VME transfer rates up to 20 Mbyte/sec. In addition to being fast on both ends, the 68000-based Introl 300 has a command overhead under one µs and a very efficient FIFO design.

Other performance features include support of scatter/gather, command queuing and SCSI disconnect/reconnect.

The adapter has also been tested on disk and tape drives from CDC, HP, Fujitsu and others. Optical drives are currently being evaluated as well.

For additional information on the Introl 300, call Laura Price at Introl at 612-613-7600.



The Introl 300 VME/SCSI adaptor

EDUCATION AND TRAINING SEMINARS AT MICROWARE

Microware regularly conducts seminars at our Des Moines, Iowa headquarters and by special request at other Microware facilities. These seminars cover a wide variety of topics designed for the entire spectrum of OS-9 users. Seminar attendees range from OEM representatives to end-users, and come from all over the world.

Microware regularly conducts seminars at both its Des Moines and Santa Clara Offices

Seminars are a combination of lectures, example implementations and product demonstrations. The material is covered during a two-day session. Microware currently offers two types of seminars: Intermediate and Advanced Topics. Specialized seminars are also available at any Microware location, or on-site at your facilities. Prices for these customized seminars will be quoted upon request.

The Intermediate Topics seminar is designed for users relatively new to OS-9 who need to gain product knowledge quickly. Topic areas addressed in this seminar include utility usage, interprocess communications, C Language, debugging and the OS-9 I/O system. Emphasis will be given to understanding the underlying OS-9 concepts so that advanced subjects can be more readily understood.

The Advanced Topics seminar was created for those users who are familiar with OS-9 and require additional implementation information. The subject matter builds upon the material presented in the Intermediate Topics seminar with additional emphasis placed on writing device drivers and customizing your OS-9 system.

For more information on Microware seminars, contact Ric Yeates at the Des Moines office or Drew Crane at the Western Region Office.

microware®

MICROWARE SYSTEMS CORPORATION
1900 NW 114th Street
Des Moines, Iowa 50322

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