Volume 3 Number 1 Spring 1988

microware®

PIPELINES

Covering the Full Spectrum of OS-9 News and Applications

IN THIS ISSUE:

◆ OS-9 VERSION 2.2...Page 2

◆ NEW DEVELOPMENT PAKS FROM MICROWARE...Page 3

◆ ON THE C SIDE...Pages 4, 5 & 6

◆ NEW ITALIAN IMPLEMENTORS...Page 8

PIPELINES
Spring 1988
Volume 3 Number 1

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

Editor:
David F. Davis

Contributing Editors:
Andy Ball Dave West James Jones

Photography
David F. Davis

Art Director
Tom West

A NOTE TO OUR LOYAL 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. Address submissions to the Editor of Pipelines, c/o Microware Systems Corporation. Please include your name, address and telephone number.**

JOB OPPORTUNITIES AT MICROWARE

Due to continued growth, Microware is expanding its programming and marketing staff and currently accepting resumes for job openings. Programming applicants must have practical experience with OS-9 creating device drivers and installing system software on new systems. Marketing candidates should have a computer related background, good communications skills and a superior track record in sales. Send your resume (including salary requirements) today to the Personnel Director at Microware. No phone calls please!**

OS-9 VERSION 2.2

In February, Microware released a new 2.2 Version of the Industrial, Personal and Professional OS-9 Operating System. The new version includes an update of selected operating system modules, plus an enhanced version of Microware's C Compiler. This release does not require the re-assembly or modification of any system-level software modules or application programs. Most OEM's have competed integration and testing of V2.2 and are currently shipping this new version of OS-9.

The module updates includes additional functionality to support three exciting new Microware software products (previewed in the Fall, 1987 edition of Pipelines): C Source Level Debugger, Ethernet TCP/IP package and Memory Protection Module (SPU). (Please see the C Side in this issue for an informative tutorial covering the new C Source Level Debugger.) Of course, the release contains corrections for errors reported in the 2.1 Version of OS-9/680x0.

OS-9 Version 2.2 now available from Microware

New features added to C include bit-fields, enumerated data types and the ability to return structures and unions. The use of bit fields can reduce program memory requirements by allowing data to be stored in a more efficient sized variable. Enumerated data types have also been included. Their use makes programs more understandable and error free by permitting constants to be referenced by a symbolic name. And finally, the ability to return structures and unions is useful for manipulation of variables. With the addition of these three new features, Microware's Version 3.0 C Compiler now represents an even more powerful C Language tool for software development.

All changes included in the new OS-9 Version 2.2 are documented fully in Release Notes that accompany the update disks. End users may refer to these notes for a more exact description of changes made in the operating system. Anyone having questions regarding V2.2 should contact their Microware representative or their system supplier for additional information.**

ADA CROSS COMPILER

Microware announces the availability of the Systeam Ada Cross Compiler for VAX/VMS systems from our West German distributor, Dr. Rudolf Keil, GmbH. The Systeam Ada conforms to the Department of Defense 4.4 Ada specification. Microware now can offer its customers a tool to pursue U.S. government contracts which specify software projects to be developed under the Ada programming language.

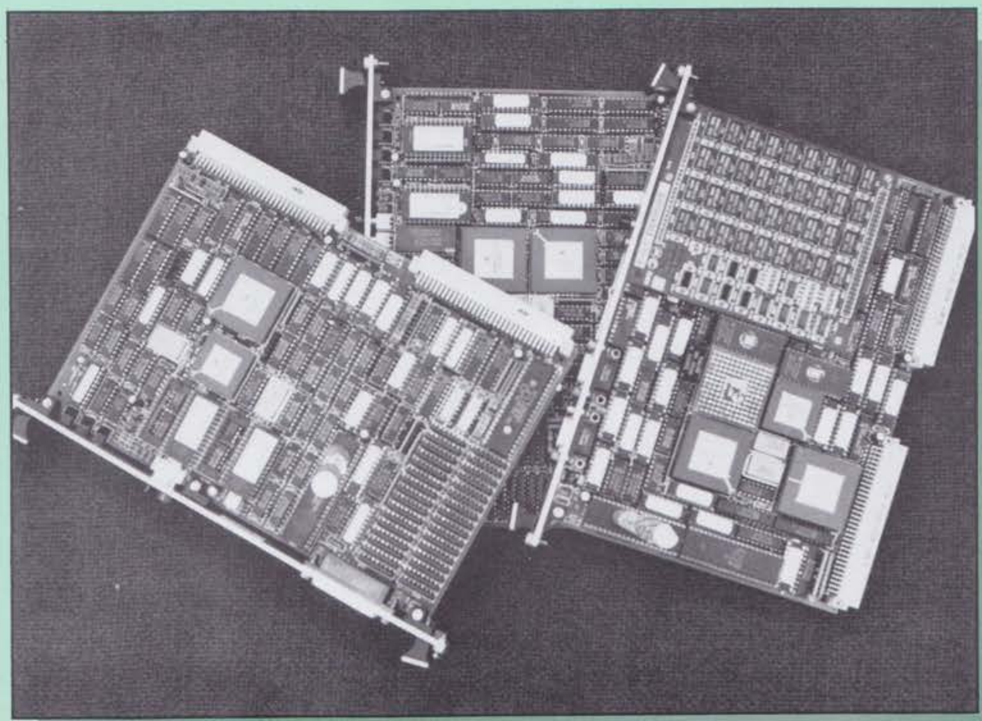
The Ada Cross Compiler is hosted on VAX/VMS and produces object code for a Motorola MC68020 CPU and MC68881 FPU running under OS-9. It translates full Ada as specified in the 1983 Ada Reference Manual.

The user interface is easy to use. The cross compiler is well integrated into the underlying host operating system. It can recognize about 500 different English language error messages. Messages are self-explanatory and accurately positioned.

NEW DEVELOPMENT PAKS FROM MICROWARE

MVME 133A, 134 & 135 NOW AVAILABLE

New Development Paks for Motorola VME 133A, 134 and 135 CPU's are now available from Microware. The MVME 133A, 134 and 135 are popular 68020 CPU's designed for software development and industrial applications. The new Development Paks have been customized to support both European (MVME 319) and American (MVME 320) configurations. An optional MVME 050 system controller can be added to provide two additional serial ports and one parallel port in addition to other features.


The MVME 133A, 134 and 135 32-bit monoboard microcomputers

The MVME 133A includes complete device driver or trap handler support for the on-board MC68881 FPU, RS-232 serial ports and 20 Mhz clock. These features make this microcomputer a popular choice for numerically intensive applications. The MVME 134 has a MC68851 Paged Memory Management Unit supported by Microware's OS-9/SSM Security Pak making it an ideal engine for large multi-user systems. The significant feature of the MVME 135 is a VSB private memory bus which OS-9 supports for multi-processor systems and systems incorporating high-speed DMA controllers. As with all Microware/Motorola Development Paks, these versions of OS-9 can support up to 16 megabytes of RAM. Microware has also recently released an Optional MVME Driver package for all Motorola VME Development Paks. The Optional Driver package includes object code for the MVME 335 serial/parallel device drivers and the MVME 350 magnetic tape device drivers.

Contact Microware or an authorized Microware distributor for complete Development Pak pricing and distribution information.**

NOW AVAILABLE FOR OS-9

The cross compiler can translate up to 1800 lines per minute on a VAX 8500 and requires 1.5 MBytes of hard disk storage space. The amount of virtual memory required for a given application depends on the size and number of compilation units to be compiled and typically varies between 4 and 7 MBytes.

The cross compiler is delivered on magnetic tape or floppy disk and full documentation is supplied with every cross compiler. To order the Systeam Ada Cross Compiler, or for more information, contact Dr. Keil at the address listed below:

Dr. Rudolf Keil, GmbH
Gerhart-Hauptmann-Str. 30
6915 Dossenheim, West Germany
Phone: 06221-862091
Telex: 461166
FAX: 8221-861954

ELECTRONIC MAIL INCREASES PRODUCTIVITY

Microware announces the availability of an Electronic Mail (EMail) communications program for multi-node networks and development systems. EMail uses your favorite OS-9 editor (uMACS, SCRED, etc.) to create or modify messages. EMail features distributed mailing lists (entire groups accessed at one time) or consecutive mailing lists (from terminal to terminal). Mailing lists can also be defined in terms of other lists. EMail features on-line help and complete, easy-to-understand documentation. Contact Microware for additonal information.**

ON THE C SIDE - USING SRCDBG

The following is an example of using srcdbg to debug a C program. Only the spelling mistakes have been edited, to spare the author embarrassment. (Indeed, the program turned out to have two errors instead of the one intentionally put there!)

This program is inspired by a passing comment in Kernighan and Pike (**The Unix Programming Environment**, Prentice-Hall) that most programs designed to handle text make an assumption about maximum line lengths and hence fail when handed cases that violate that assumption. Here we attempt to avoid this assumption by allocating space to hold the text of a line from the heap, using realloc() to grab more space when it is needed. The original program reads as follows:

```
#include <stdio.h>

#define TRUE      1
#define FALSE     0
#define HUNK      256

typedef struct {
    char    *text;
    int     len;
    int     buflen;
}   flexline;

main()
{
    char            *flexgets();
    flexline        line;

    while (flexgets(&line, stdin) != (char *) NULL)
        fputs(line.text, stdout);
}

char *
flexgets(line, fp)
register flexline       *line;
register FILE           *fp;
{
      register int      c;
      int               addc();

      line->len = 0;

      while ((c = getc(fp)) != EOF) {
          if (!addc(line, c))
              return NULL;
          if (c == '\n')
              break;
      }

      if (line->len == 0 || !addc(line, '\0'))
          return NULL;
      else
          return line->text;
}

int addc(line, c)
register flexline       *line;
register int            c;
{
    void    *realloc();

    if (line->len > line->buflen) {
        if ((line->text = realloc(line->text, line->buflen += HUNK)) == NULL)
            return FALSE;
    }
    line->text[line->len++] = c;
    return TRUE;
}
```

The above source code was placed in the file sample.c. We now join the shell, already in progress:

```
$ cc -gqt=/r0 sample.c
$ srcdbg
srcdbg: fork sample <sample.c
```

[I was lazy and just read stdin for an initial demo, so I redirect stdin to avoid having to type input at the program.]

```
Reading symbol file:
sample.c
File: "sample.c"
Context: sample.c\main
  13: main()
      ^
```

ON THE C SIDE - USING SRCDBG
(Continued)

```
srcdbg: g
Execution Error - Error #000:102 (E$BusErr)  A bus trap error occurred.
class = 0111  violation address = 00000000  instruction = 4E55
File: "sample.c"
Context: sample.c\addc\$blk0
  57:   return TRUE;
               ^
```

[Since the program counter points at the next instruction by the time an exception occurs, sometimes you may have to look at the preceding statement to find the true location of the problem. That seems to have happened here; "return TRUE" is pretty innocuous. We remind ourselves of the text of addc():]

```
srcdbg: list stet
  46: addc(line, c)
  47: register flexline *line;
  48: register int      c;
  49: {
  50:   void    *realloc();
  51:
  52:   if (line->len > line->buflen) {
  53:       if ((line->text = realloc(line->text, line->buflen += HUNK)) == NULL)
  54:           return FALSE;
  55:   }
  56:   line->text[line->len++] = c;
  57:   return TRUE;
  58: }
[End of file]
```

[Looks good to me--if you've filled the buffer, try to grab some more memory, then insert the new character at the correct position. Let's look at the contents of *line.]

```
srcdbg: print (void *) line->text
0x0
srcdbg: print line->buflen
2272338
```

[Clearly something's wrong. *line wasn't initialized. In fact, I intended to make it a global variable and let C's zeroing of uninitialized globals do the job for me. Fortunately, we can keep debugging without having to recompile, like this:]

```
srcdbg: fork sample <sample.c
Reading symbol file:
sample.c
File: "sample.c"
Context: sample.c\main
  13: main()
      ^

srcdbg: step
  18:   while (flexgets(&line, stdin) != (char *) NULL)
               ^

srcdbg(STEP): print (void *) line.text
0x0
srcdbg: assign line.text = 0; assign line.len = 0; assign line.buflen = 0
```

[There. We'll add the initialization to the code later. Onward!]

```
srcdbg: g
Execution Error - Error #000:102 (E$BusErr)  A bus trap error occurred.
class = 0111  violation address = 00000000  instruction = 4E55
File: "sample.c"
Context: sample.c\addc\$blk0
  57:   return TRUE;
               ^
```

[What?! Let's look at that again...]

```
srcdbg: fork sample <sample.c
Reading symbol file:
sample.c
File: "sample.c"
Context: sample.c\main
  13: main()
      ^

srcdbg: step
  18:   while (flexgets(&line, stdin) != (char *) NULL)
               ^

srcdbg(STEP): assign line.text = 0; assign line.len = 0; assign line.buflen = 0; step
  22: char *
           ^

srcdbg(STEP): step
  30:   line->len = 0;
        ^
```

ON THE C SIDE - USING SRCDBG
(Continued)

```
srcdbg(STEP): step
  32:   while ((c = getc(fp)) != EOF) {
                ^
srcdbg(STEP): step
  33:       if (!addc(line, c))
                ^
srcdbg(STEP): step
  46: addc(line, c)
      ^
srcdbg(STEP):
  52:   if (line->len > line->buflen) {
            ^
srcdbg(STEP):
  56:   line->text[line->len++] = c;
        ^
```

[Aha! We're not allocating more memory--or any memory at all. Looks like the ubiquitous "off by one" error; this statement should bomb. We verify this...]

```
srcdbg(STEP):
Execution Error - Error #000:102 (E$BusErr)  A bus trap error occurred.
class = 0111  violation address = 00000000  instruction = 33C0
File: "sample.c"
Context: sample.c\addc\$blk0
  56:   line->text[line->len++] = c;
        ^
srcdbg: list 52 56
  52:   if (line->len > line->buflen) {
  53:       if ((line->text = realloc(line->text, line->buflen += HUNK)) == NULL)
  54:           return FALSE;
  55:   }
  56:   line->text[line->len++] = c;
srcdbg: quit
```

[So--the > should be >=. We edit the file using an editor, add the initialization, change the test and recompile.]

```
$ cc -gqt=/r0 sample.c
$ srcdbg
srcdbg: fork sample <sample.c
Reading symbol file:
sample.c
File: "sample.c"
Context: sample.c\main
  13: main()
      ^
srcdbg: g
#include <stdio.h>

#define TRUE      1
#define FALSE     0
#define HUNK      256
...ect.
"sample" exited normally
```

[Success! All in a day's work for srcdbg...]

```
srcdbg: q
$
```

This simple program doesn't show all the features of srcdbg; we didn't need to look at the call stack or set breakpoints or watchpoints to find the bugs. It does, however, show some of the high-level features of srcdbg.

- we always looked at the C source code, not the instructions of the compiled program

- we could refer to variables by name, rather than having to figure out their locations on the stack or in the process's data area. (We did this for fields within structures, which debug won't do for us, and we could have done it for static variables, which debug can't find.)

NEW EMPLOYEES

Here are the latest additions to our growing staff.

Jim Guisinger is Microware's new Manager of Inventory/Purchasing. Jim previously worked for a Des Moines-area business systems company as a branch manager.

Dianne Mosley, Administration Assistant for R&D, formerly worked as an executive secretary for a Des Moines area publisher in their personnel Department.

Tom West has joined the Marketing Department in the new position of Art Director. Tom formerly worked for a local advertising agency and has a degree in Commercial & Advertising Art/Graphic Design.

Marv Howard is a new voice on the Microware Hotline. He has a degree in Computer Science from Iowa State University where he worked his way through school as a programmer and mainframe systems engineer for several local companies.

Pictured (l-r): Jim Guisinger, Dianne Mosley, Tom West and Marv Howard.

(Not Pictured)
Mike Burgher, Technical Services Manager--Western Region, previously worked for Continental Oil and Control Data for 12 years. Mike will be helping Microware customers in the Western Region define their software requirements and developing OS-9 tools for varied hardware.

Scott Watters, Software Engineer--Western Region, is a graduate of Sacramento State University. His duties will include programming, porting OS-9 to a variety of hardware and providing technical support to customers in the Western Region.

Last Call for "HELP!"

We Need to Clean-Up Our Mailing List.

As we mentioned in the last issue of Pipelines, the publication's mailing list has grown to dwarf the Tokyo Telephone Directory. We are now faced with the decision of either charging for Pipelines to cover the cost of postage, or reviewing our mailing list. Since we want everybody to receive Pipelines free of charge, we have decided on the latter course of action. To assure that you continue to receive Pipelines (without interruption and without cost) **YOU MUST FILL OUT AND RETURN THE INFORMATION COUPON BELOW** (If you have already sent in your coupon from the last issue of Pipelines, please disregard this notice, and **"thanks!"**). Send the completed form to: Editor, Pipelines c/o Microware Systems Corporation, 1900 N.W. 114th Street, Des Moines, Iowa 50322. If there is someone you know who would also like to be on our mailing list, please include their name and address, too.

Name ______________________ Title ______________

Company ______________________ Phone (   ) ______________

Address/P.O. Box ______________________________

City/State/Zip ______________________________

MICROWARE WELCOMES ITALIAN AUTHORIZED IMPLEMENTORS

Microware and Eximar S.r.l. of Rome, Italy, have signed an Authorized Implementors agreement for OS-9 technical sales and service in Italy. As Authorized Implementors, Eximar will be working with Microware and its Distributors to promote Microware products throughout Italy. Their primary focus will be to provide a complete menu of technical assistance including education, software support, integration and installation services.

Eximar's president, Carlo Narcisi, previously served in the Italian armed forces and represented Motorola GEG for 15 years prior to forming Eximar in 1983. Eximar's Technical Services Manager, Ercole Maccioni, also worked for Motorola Microsystems prior to joining the firm in 1984. Ercole and Roberto Evangelista (pictured at left) came to Microware for technical training and product marketing meetings the week of April 18th.

Eximar services the Italian defense electronics industry on behalf of selected American high-technology companies. They also provide installation and integration services for commercial customers in the the process control and board-level markets. Readers interested in more information may contact Eximar at the following address:

Eximar S.r.l.
Via Dei Ciceri, 59 * 00175 - Rome, Italy
Phone: (6) 762221 * Telex: 623358

Eximar's Ercole Maccioni and Roberto Evangelista

microware®

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

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