USUS NewsLetter

Serving the Pascal, Modula-2, and Portable Programming Community Vol. 5 No. 3 December 1991

I N T H I S I S S U E

Copyright 1991, USUS INC. All Rights Reserved.

The USUS NewsLetter is published 6 times per year by USUS, the UCSD Pascal System User's Society, P.O. Box 1148 La Jolla, California 92038. The NewsLetter is a direct benefit of membership in USUS.

Tom Cattrall Editor
William Smith Publisher

THE PROPOSED STANDARD FOR MODULA-2
by Pat Terry

USUS members may be curious as to the state of the proposed standard for Modula-2.

This "unofficial" note aims to give some idea of the rough state of the language that is likely to be proposed in the next Draft of the Standard, as agreed at the most recent meeting of WG13/SC22, held in July 1991 in Tuebingen, Germany.

The note is (deliberately) imprecise, but should give you a feeling of the surprises one may expect to see when the draft is published, probably later this year. In what follows I have focussed on the "added" features, rather than the "clarifications", of which there are many (for example, exactly what is meant by compatibility in various contexts).

Devotees of Modula-2 as it was originally described in PIM (Programming in Modula-2, Wirth's textbook), where it was a rather "small" language, may be surprised at some of what follows. Indeed, the trend towards extending the language desire for extensions is not shared by all of the Standards group, and in particular not by the US members.

Comments, criticisms and queries will be welcomed. You can e-mail them to me, at either of the addresses:

Pat.Terry@f4.n7104.z5.fidonet.org

or

pdterry@m2xenix.psg.com

or, on FidoNet as Pat Terry of 5:7104/4.

You can also send comments to the chairman of the US TAG (Task Action Group), Randy Bush, whose name will be familiar to aficionados of the excellent old Volition Modula-2 and Oregon Software Modula-2 implementations. Randy can be contacted by e-mail at

randy@m2xenix.psg.com

or, by regular mail, at

Pacific Systems Group,
9501 SW Westhaven Drive
PORTLAND
OR 97225

Any errors and misleading statements in what follows are unintentional, and are my responsibility; they should not reflect on the other members of the group.

The draft of the proposed standard document runs to about 500 pages, and there are still bits to be added to the text. Much of this bulk is attributable to the fact that the language is specified primarily using a mathematical language "VDM-SL" (Vienna Definition Language - Specification Language"). And part of the delay in standardizing Modula-2 has arisen because VDM-SL is not yet standardized or stable either!

On to the changes:

New reserved words:

```
REM   FORWARD   PACKEDSET   EXCEPT
FINALLY    RETRY
```

/ and REM provide alternative ways of doing whole number division and remaindering. PACKEDSET is a way of specifying "packed" sets (that is, bitsets), and EXCEPT, FINALLY and RETRY are there to deal with exception handling and termination, probably the biggest and most contentious additions to a once fairly simple language in this area.

New pervasive (standard identifiers):

```
COMPLEX   CMPLX    ENTER    IM    INT
INTERRUPTIBLE   LEAVE LENGTH    LFLOAT
LONGCOMPLEX   PROT   PROTECTION      RE
UNINTERRUPTIBLE
```

These are discussed in more detail below.

New lexical elements:

(! and !) as equivalents for [and]
(: and :) as equivalents for { and }
@ as equivalent for ^ (pointer dereferencing)

These are all to handle non-ASCII machines.

<* compiler directives *> in <* brackets *>

This attempts to get away from comments that are not really comments and to ensure portability to a greater extent than before.

__Underlines_allowed_anywhere_____in_identifiers

This should be attractive to devotees of this style.

While the Committee has not gone so far as to introduce a STRING type (the issue was discussed), some extra support for string operations has been added. For example

+ may be used to concatenate string constants:

```
"this" + " is fun"
```

New types:

The new types COMPLEX and LONGCOMPLEX give the possibility for doing complex arithmetic very much as in FORTRAN. Standard identifiers

```
CMPLX(real, imaginary)  construct complex
                        from real
RE(complex)             extract real part
IM(complex)             extract imaginary
                        part
```

and a basic complex library have been proposed.

The type PROTECTION allows for more "portable" control over interrupts. Five standard identifiers are introduced for this:

function PROT() returns current level of protection

INTERRUPTIBLE, UNINTERRUPTIBLE define two of the levels (an implementation may add others)

Procedures ENTER(protection) and LEAVE(protection) allow for bracketing statements to guard against interrupts.

As already mentioned, there is a new type constructor. The PACKEDSET keyword allows for construction of sets required to be "packed" so that each element is represented by one bit. For example

```
TYPE
  Permissions = PACKEDSET OF
    (mayOpen, mayClose, mayTryCinstead);
```

and the standard type BITSET is in this category:

```
TYPE
  BITSET = PACKEDSET OF
    [0 .. SomeImplementationDefinedLimit];
```

Other sets are not required to be packed, and the upper limit on set size may be considerably larger for "ordinary" sets than for "bitsets".

Other new pervasive identifiers have been added to clean up

type conversions:

```
INT(real)          convert real to INTEGER
LFLOAT(wholenumber) convert wholenumber
                (INTEGER/CARDINAL) to LONGREAL
```

Other conversion functions like this have also been "relaxed". So, for example,

```
FLOAT(wholenumber)     convert wholenumber
                (INTEGER or CARDINAL) to REAL
```

The general purpose type converter, VAL, has been considerably relaxed. Note that VAL does not cast or coerce, as it does in some present implementations.

The last new pervasive is LENGTH:

LENGTH(string) returns length of its string parameter

Type Casting:

Type casting is now to be done as SYSTEM.CAST(TYPE, value), not TYPE(value) as it was in PIM. This is to ensure that the potentially non-portable use of this feature is properly highlighted. The TYPE(value) syntax is to be removed.

Various other fundamental changes:

Various of these have been made. Some of these would take too long to explain in detail here:

The order of initialization of circularly importing modules is now defined.

Recursive procedure types are allowed for example:

```
TYPE  RecProc = PROCEDURE (Real, RecProc)
```

Coroutines have been moved from SYSTEM to another system module, named COROUTINES. Several extra features have been added; and these are no longer directly compatible with PIM. The motivation is that interrupt handling, in particular, on the PIM model was inadequate.

Multi-dimensional open array parameters are allowed.

A new set of rules for FOR statements and their control variables has been formulated. Basically, thou shalt define control variables locally and thou shalt not alter them; this is explained more clearly in about 16 pages of formal specification!

FORWARD declarations are allowed, and must be acceptable to multipass compilers.

Termination and exception handling have been built into the language through the keywords FINALLY and EXCEPT and RETRY, along with a supporting system level module. This is a very big change. Some idea of the feature can be gleamed from the following examples:

```
IMPLEMENTATION MODULE Wotsit;

  IMPORT EXCEPTIONS (*system module*);

  VAR
    LocEx : EXCEPTIONS.EXCEPTION;

  PROCEDURE Action ();
    VAR
      Which : EXCEPTIONS.EXCEPTION;
    BEGIN
      Something;
      IF Wrong THEN
        EXCEPTIONS.RAISE(LocEx) END;
      SomethingElse
    EXCEPT
      HandleException;
    END Action;

  BEGIN (*Module body*)
    InitialisationCode;
  FINALLY
    TerminationCode
  EXCEPT
    HandleTheException
  END Wotsit.
```

Second example:

```
FROM EXCEPTIONS IMPORT
  ExceptionValue, EXCEPTION,
  RAISEGENERALEXCEPTION;
FROM LibModule IMPORT
  LibExceptionValue, LibException, Fly,
  ReplaceRubberBand;

PROCEDURE KeepFlying();

  PROCEDURE TryFlying();
    BEGIN
      Fly
    EXCEPT
      IF LibExceptionValue() =
         BrokenRubberBand
      THEN
        ReplaceRubberBand;
        RETRY
      END
      (* Re RAISE exception *)
    END TryFlying;

  BEGIN
    (* Statements in normal execution *)
    TryFlying;
    (* ..... *)
  EXCEPT
    CASE ExceptionValue() OF
      | NotLanguageException:
```

```
            RAISEGENERALEXCEPTION("Unknown
                          library exception")
        | IndexException
            (* Take recovery exception if
               possible *)
            RETRY
        (*  Other cases *)
        ELSE
          (* Re Raise exception *)
      END
  END KeepFlying;
```

Value constructors are allowed for arrays and records, as well as for sets:

```
TYPE
  ArrayType = ARRAY [0 .. 10] OF REAL;
VAR
  x : ArrayType;
BEGIN
  (* lots of code *)
  x := ArrayType{5.3 BY 5, 6.7,
                 8.2*sqrt(y), 0.0 BY 3}
```

Full generality is allowed, which this simple example does not show. In particular, note that it is not intended just for constants or for variable initialization.

The SYSTEM module typically is required to export

```
(*types*)
    LOC, BYTE, WORD, ADDRESS, MACHINEADDRESS

(*ADDRESS manipulation*)
    ADDADR, SUBADR, DIFFADR, ADDRESSVALUE

    SHIFT, ROTATE

    CAST (*type casting*)
    TSIZE, ADR (*old favorites*)
```

LOC is the "smallest addressable location" (typically a byte). MACHINEADDRESS is for use in specifying absolute addresses in variable declarations:

```
VAR
  Screen
   [MACHINEADDRESS {0B800H,0H} ] : BigArray;
```

Library modules:

The modules of the "standard" library will be "optional" - however, implementations that claim to be standard will not be allowed to have modules of these names that do not conform exactly to the features and semantics of the "standard" ones. Note also that several of these will make direct use of the exception handling features, and so will be rather unlike anything seen up till this time. Only the briefest of summaries can be given here:

I/O Library

This is different in very many respects from all of the ones proposed in earlier drafts of the Standard. Indeed, every draft has seen virtually a complete redesign on what has gone before.

I/O operations (reading, writing)

StdChans	10 procedures
ProgramArgs	3 procedures
IOTypes	1 type
TextIO	8 procedures
WholeIO	4 procedures
RealIO	5 procedures
LongIO	5 procedures
RawIO	2 procedures
IORes	1 procedure and 1 type

The following are much the same, but for default channels

STextIO	8 procedures
SWholeIO	4 procedures
SRealIO	5 procedures
SLongIO	5 procedures
SRawIO	2 procedures
SIORes	1 procedure and 1 type

Device modules (opening, closing, positioning)

DevConsts	3 types and 7 constants
StreamFile	3 procedures, 3 types and 5 constants
SeqFile	7 procedures, 3 types and 5 constants
RndFile	10 procedures, 5 types and 6 constants
TermFile	3 procedures, 3 types and 5 constants

Interfaces

IOChan	17 procedures and 3 types
IOLink	7 procedures and 17 types

Storage

Storage	5 procedures, 1 type and 2 constants

Concurrent processing

Processes	16 procedures and 5 types (Not the PIM module)
Semaphores	5 procedures and 1 type

String handling

CharClass	6 procedures
Strings	21 procedures and 3 types
ConvTypes	2 types
WholeConv	8 procedures
RealConv	7 procedures
LongConv	7 procedures
WholeStr	4 procedures
RealStr	5 procedures
LongStr	5 procedures

System clock

SysClock	5 procedures and 8 types

Mathematics

RealMath	12 procedures, 2 constants and 1 type
LongMath	12 procedures, 2 constants and 1 type
ComplexMath	15 procedures, 1 type and 3 constants
LongComplexMath	15 procedures, 1 type and 3 constants
LowReal	15 procedures and 15 constants
LowLong	15 procedures and 15 constants

<><><><><><><><> The VOGON News Service <><><><><><><><><>

Edition : 2336 Tuesday 4-Jun-1991 Circulation : 8466

VNS TECHNOLOGY WATCH: [Mike Taylor, VNS Correspondent] [Littleton, MA, USA]

COMPUTERWORLD 1 April

CREATORS ADMIT UNIX, C HOAX

In an announcement that has stunned the computer industry, Ken Thompson, Dennis Ritchie and Brian Kernighan admitted that the Unix operating system and C programming language created by them is an elaborate April Fools prank kept alive for over 20 years.

Speaking at the recent UnixWorld Software Development Forum, Thompson revealed the following:

"In 1969, AT&T had just terminated their work with the GE/Honeywell/AT&T Multics project. Brian and I had just started working with an early release of Pascal from Professor Nicklaus Wirth's ETH labs in Switzerland and we were impressed with its elegant simplicity and power. Dennis had just finished reading 'Bored of the Rings', a hilarious National Lampoon parody of the great Tolkien 'Lord of the Rings' trilogy. As a lark, we decided to do parodies of the Multics environment and Pascal. Dennis and I were responsible for the operating environment.

"We looked at Multics and designed the new system to be as complex and cryptic as possible to maximize casual users' frustration levels, calling it Unix as a parody of Multics, as well as other more risque allusions. Then Dennis and Brian worked on a truly warped version of Pascal, called 'A'. When we found others were actually trying to create real programs with A, we quickly added additional cryptic features and evolved into B, BCPL and finally C. We stopped when we got a clean compile on the following syntax:

```
for(;P("),R--;P("|"))for(e=C;e-- ;P("_"+(*u++/8)%2))P("| "+(*u/4)%2);
```

"To think that modern programmers would try to use a language that allowed such a statement was beyond our comprehension! We actually thought of selling this to the Soviets to set their computer science progress back 20 or more years. Imagine our surprise when AT&T and other US corporations actually began trying to use Unix and C! It has taken them 20 years to develop enough expertise to generate even marginally useful applications using this 1960's technological parody, but we are impressed with the tenacity (if not common sense) of the general Unix and C programmer. In any event, Brian, Dennis and I have been working exclusively in Pascal on the Apple Macintosh for the past few years and feel really guilty about the chaos, confusion and truly bad programming that

have resulted from our silly prank so long ago."

Major Unix and C vendors and customers, including AT&T, Microsoft, Hewlett-Packard, GTE, NCR, and DEC have refused comment at this time. Borland International, a leading vendor of Pascal and C tools, including the popular Turbo Pascal, Turbo C and Turbo C++, stated they had suspected this for a number of years and would continue to enhance their Pascal products and halt further efforts to develop C. An IBM spokesman broke into uncontrolled laughter and had to postpone a hastily convened news conference concerning the fate of the RS-6000, merely stating 'VM will be available Real Soon Now'. In a cryptic statement, Professor Wirth of the ETH institute and father of the Pascal, Modula 2 and Oberon structured languages, merely stated that P. T. Barnum was correct.

In a related late-breaking story, usually reliable sources are stating that a similar confession may be forthcoming from William Gates concerning the MS-DOS and Windows operating environments. And IBM spokesman have begun denying that the Virtual Machine (VM) product is an internal prank gone awry.

{COMPUTERWORLD 1 April}
{contributed by Bernard L. Hayes}

Object-Oriented Programming (OOP) Resources

David T. Craig
736 Edgewater, Wichita, Kansas 67230

26 September 1990

Object-oriented programming (OOP) and OOP libraries are becoming more and more prevalent in the field of computer programming. To assist in the introduction and learning of this exciting development I have compiled a set of documents that provide an excellent foundation for the understanding of OOP from a Pascal perspective.

Since 1983 Apple Computer has been involved with OOP. Apple first developed the language Clascal (Classes and Pascal) for its Lisa computer. Complimenting this unique language was an extensive set of class libraries called the Lisa Toolkit. Clascal evolved in 1985 into Object Pascal for the Macintosh computer and the Toolkit evolved into MacApp. Note that Niklaus Wirth, the designer of Pascal and Modula, assisted Apple in the design of Object Pascal.

Apple provided the following documents for Clascal, the Toolkit, and Object Pascal:

- **An Introduction to Clascal (55 pages)**

This is an excellent discussion of the merits and mechanics of an object-oriented programming methodology. I find this document to be more readable than Apple's later Object Pascal tutorials.

- **Object Pascal Report (9 pages)**

Complete description of Object Pascal and how it compares to Pascal. Note that this report was placed by Apple into the public domain.

- **Object Pascal vs. Lisa Clascal (4 pages)**

Fascinating discussion of the changes Apple made to Clascal to create Object Pascal. Provides rationales for each change.

- **Toolkit Reference Manual and Sources (≈ 1000 pages)**

Detailed reference for all the Toolkit classes and the Toolkit Clascal sources. If you really want to see what goes into developing a complete class library this is for you. Toolkit sources are also available on Macintosh diskettes.

Many other secondary resources from magazines exist which introduce Clascal, the Toolkit, and Object Pascal. I have photocopies of articles covering these topics.

Copies of these documents have been sent to the USUS Administrator. If others in USUS have an interest in these contact either the administrator or David Craig. I will gladly copy whatever you need as long as you pay for the photocopies (≈ 10¢ per page) and the postage cost.

Thirty Years Watching The Same Mistakes

Felix E. Bearden and Angela Markwalter

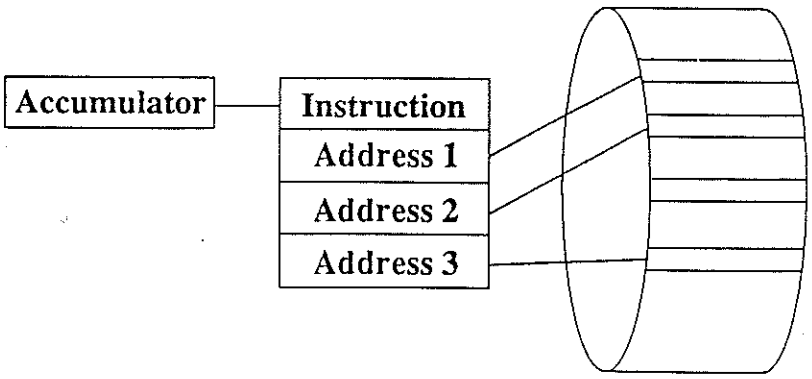
There was a town in south Texas called "Nowhere" that was having a very serious problem with rats, the rodent variety. At a town council meeting where the problem was discussed in very ecological terms, a suggestion was made to acquire the tom cat from the neighboring town of "Somewhere" because of his reputation as a ratter and a lover. The consensus was that if the tom were brought in there would be both an increase in cats and, most likely they would be cats with rat catching skills. The tom was acquired, and sure enough, after 6 months there was a definite drop in the rat population. After 8 months the measurable population dropped to about one half. After a year, the rat population was essentially eliminated.

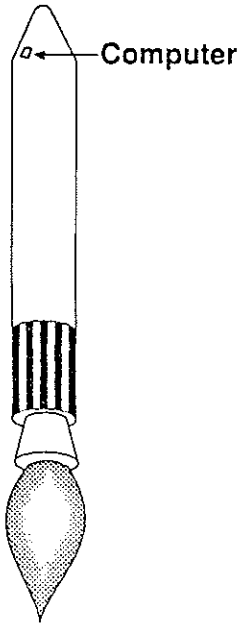
Unfortunately, "Nowhere" had a new problem. Cats. Again the town council met. The tom had a reputation around south Texas and having been passed from town to town there was no rat problem anywhere and his reputation as a lover also was well known so no other town wanted him. One suggestion that was made was that they terminate the taboo tactile tabby. But gentler hearts prevailed and the local vet was given the job of neutering the tender tom. At the next months council meeting the cat counters reported that the number of cat pregnancies had leveled off. However, at the next meeting, the tom trackers reported that the cat pregnancies had increased by an alarming 87 percent.

Again the vet was dispatched to catch and examine the tom. At the next meeting the vet reported that he had checked the tom and had found that the operation was successful but that he believed that he had identified the problem.

The tom had become a "consultant".

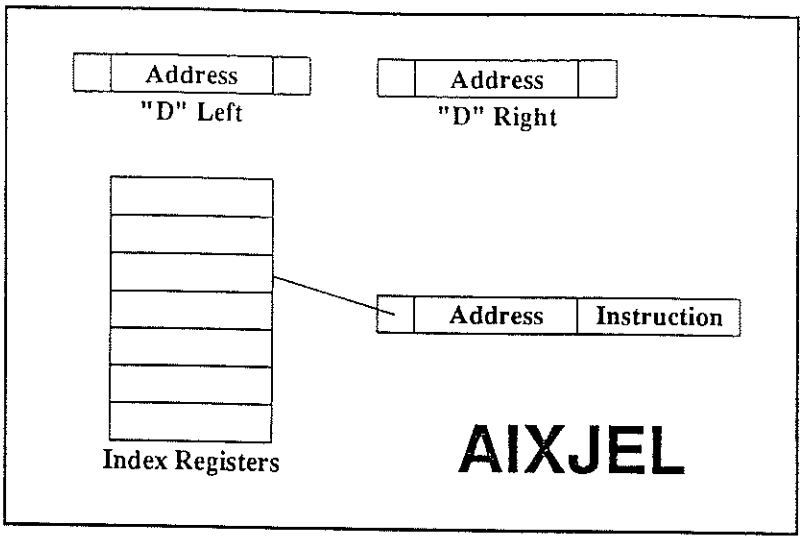
In 1959, I took a correspondence course on computers from the Philco Technological Institute. I did pretty well on the course— even the part where I had to program a three address machine. Do any of you even know what that is? Why three addresses? This computer had a drum memory, the first address was the location of the second operand (the first was in the accumulator), the second address was the location into which the results of the operation would be stored, and the third was the location of the next instruction to be executed. The trick to writing effective programs on this machine was to optimize the location of operands and instructions on the drum to reduce the effect of rotational latency. And THAT my friends was the original RISC processor.





When I finished the course I was teaching the electrical and electronic systems in the Jupiter Missile system. As you may recall, this was an inertially guided missile. What you may not know is that it had an analog computer that could compute course corrections faster than any of the existing digital computers of its day, and, more importantly, it fit neatly inside the missile. And THAT was a powerful computer. (I would advise you not to stand in its way, or wake, when it is launched). Even though there were a lot of analog computers in use at the time, in fact there

was one hybrid analog/digital computer, and it was less expensive, the digital computers finally won the contest in the market place.

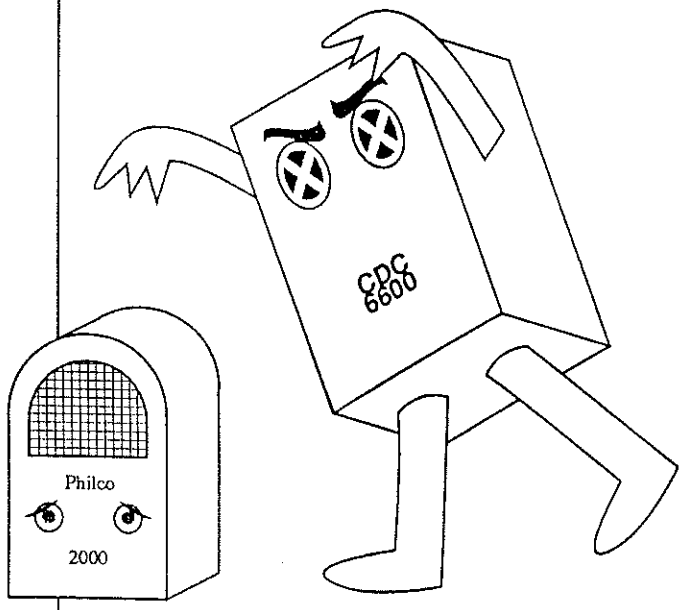


You don't need a higher language!

I must have done fairly well on my course, because I was invited to go to school to learn how to maintain the "Philco 2000". Are you surprised to know that Philco built a computer in addition to radios and washing machines? Well, they did, and did a fair job of it too. This machine was so powerful that some (all hardware engineers) claimed that a higher level language was not necessary. Well, it did have a great instruction set. How about AIXJEL? The description is, Add the contents of the address part of the instruction to an index register and if the result is equal or less than the value in the address part of the left half of the "D" register then jump to the address contained in the address part of the right half of the "D" register. Or, in other words, it is the termination instruction in a "FOR" loop. Another feature: floating point was standard. In fact on the later models, the floating point processor did floating point in hexadecimal instead of digital. Remember now, the IBM 360 was still a dream (at least to an engineer). And I don't know whether they even teach this approach any more, but the arithmetic unit was asynchronous! That's right, the instruction time depended on the operands rather than the worst case timing of synchronous machines.

The machine was a great machine. Two higher level languages were available. COBOL, and FORTRAN. The only language I used was FORTRAN. I don't think it ever met the standard, but worse, it had a typical non-thinking compiler implementor's limitation. Even though the word length was 48 bits, INTEGERs were only 16 bits. Somebody thought it was "neat" to do INTEGER computation in the index registers and only floating point in the main accumulator. With such a great floating point unit, why would anybody use an integer for anything other than indexing anyway? After spending a million dollars hand tuning the application programs on a 6600, Control Data Corporation (CDC) replaced the Philco 2000 at the Westinghouse facility doing atomic research and Philco 2000's started to be replaced by newer, more popular machines. Note: I didn't say more powerful. By the way, the 6600 was a multi-processor machine with a less powerful instruction set. One of the biggest problems for the software engineer was that HE was responsible for avoiding memory contentions and for synchronizing the processors. (At the time there were a number of papers on compilers which were supposed to generate code for parallel processing to automatically take care of these problems but I don't know if one was ever completed. If so, none ever became very popular.) The competing Philco 2000 was a pipe-lined processor and could be processing up to eight instructions at a time, AND had built-in contention logic and was priced lower.

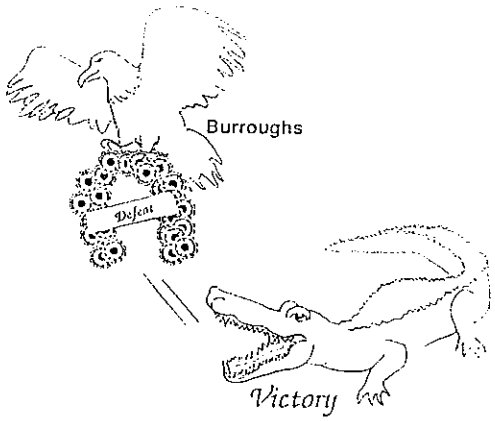
The Winner: The CDC 6600



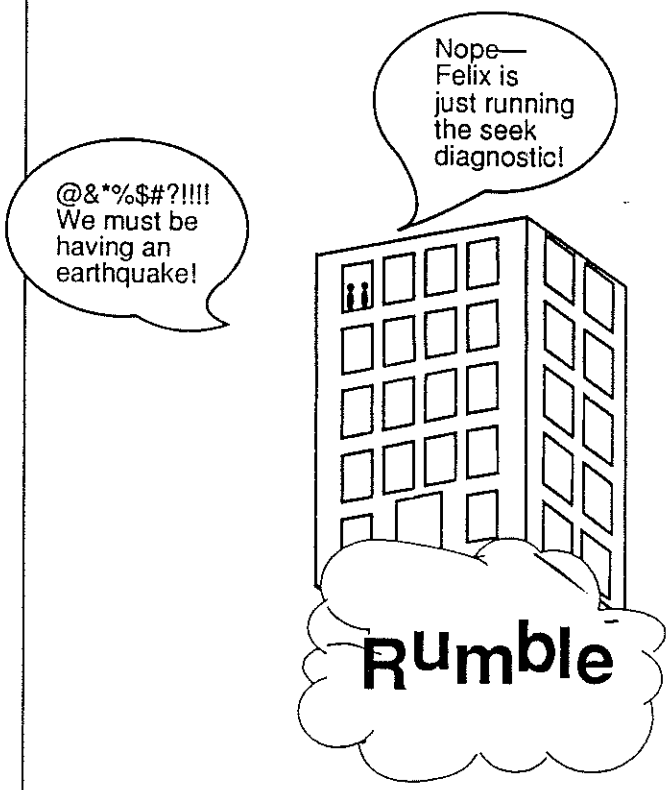
During that same period, I became aware of high level languages. In fact, as a member of the Association for Computing Machinery (ACM) I saw many references to a language called "ALGOL". Even early in my career I didn't like to reinvent the wheel so the collection of algorithms from the ACM was something I kept close. I had hand converted a number of the procedures to assembly language but I never used a real live ALGOL compiler. After Peter Naur presented an elegant description of ALGOL 60 to the world, a formal ALGOL working committee under the International Federation for Information Processing (IFIP) published a 140 page report[1] describing a language permitting user extensions such as defining data types, data structures, and operators that apply to these new types and structures. Were they on the right track? It looked like it. Unfortunately the report had "almost no exposition and the language description was inelegant."[2] I'm not sure that the report caused the demise of ALGOL, but I'm certain it didn't breath life into the language either.

In the "colonies" as you like to say, the popular languages were FORTRAN, COBOL, and IBM's entry, PL/I. Even though the Burroughs 5000, 6000, and 7000 series computer design was strongly influenced by ALGOL, Burroughs had to supply FORTRAN to its users because of their demand.

The Winner: FORTRAN (at least for the moment)



Philco announced that they were no longer going to pursue the computer market. With that announcement went a superior machine, while others, namely the IBM 70xx, and CDC 6x00 gained market domination. My experience at Philco was good. In the development lab I saw the early attempts at building a tape drive using a low mass capstan drive instead of the pinch roller mechanism that stretched so many tapes. And I saw an early moving head disk drive. The disks were about eight feet in diameter and the heads were moved by hydraulic cylinders. It worked fairly well except that it shook the whole building when doing a seek.



You really don't need one!

By this time, I was a hooked hacker (a person who loved to program computers, but had no formal training in structure and design). Minicomputers were becoming popular because of their simplicity and low cost. They were particularly popular for measurement and control applications because they could be placed closer to the application, be dedicated to a particular application and usually run unattended. I decided that I would join the minicomputer revolution at a company called Scientific Data Systems (SDS). Now there was a group of hackers. I think that the main challenge of the programming staff was uniqueness of programs. The approaches were novel. Of course they had to be when your only input/output device was a Teletype. One person, Richard Resnick, wrote an assembler that solved the forward reference problem in a one pass assembler by writing the output on paper tape backwards. Now that is taking last-in first-out to the extremes, but it worked. One of my attractions to SDS was that they were developing an elegant processor called the Sigma-7. The computer, apparently modeled after the "Atlas" computer[3], was the culmination of all of what we understood was needed for a machine designed for multi-programming (multiple registers, paged mem-

ory, and high speed state switching to name just a few of its features), but doomed by the buyout of Xerox Corporation. I can't say much about their high level languages, the only language I used was the "Meta-Symbol."[4] As assemblers go, it was about as elegant as they come. In order to write a program, you had a library of procedures to describe both the programmer's language and the target machine for which you were generating code. The assembler was so good that *"you didn't really need a higher level language"* or so I was told. They may have had FORTRAN or COBOL, but I never used it or saw it.

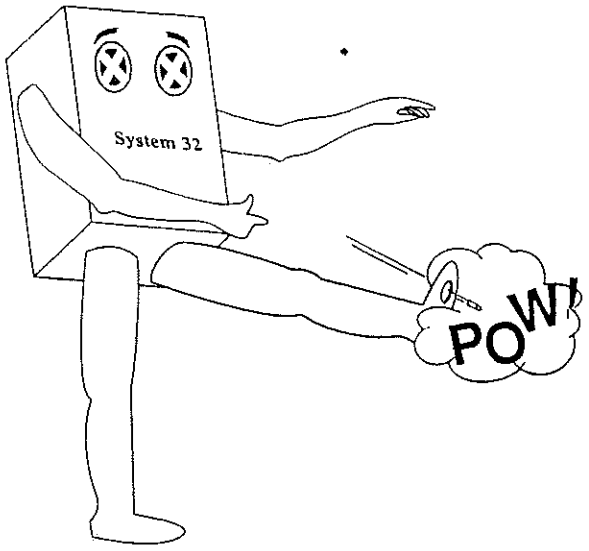
Software Engineering

Everybody was talking about a "new" concept called ***"Modular Programming"****. Because "software development costs were so high and so unpredictable, programmers were encouraged to learn about and practice techniques that came from the concepts."*

Because I had experience with the Sigma-7, I got tired of the rat-race in California, my wife was sensitive to the instability of the various geological faults (not to mention the citizens), and moving to Florida at the same salary represented a 12% raise, I left the state of nuts and earthquakes and came to the land of sunburns and retirement homes. Systems Engineering Laboratories (SYSTEMS) gave me the opportunity to contribute to the design of still another 32 bit minicomputer. Because some of the engineers were from California and knew that the new processor would probably compete with the Sigma-5 and Sigma-7, the architecture was similar. Some of the bad features of previous SYSTEMS machines were omitted, like the dubious "TURN OFF MEMORY"[5], which had an OpCode of 0, and this was in a day when computers didn't make the distinction between program and data.

However, because of production price constraints and schedule, some of the better features were omitted. Most of you are accustomed to easily booting your computers. On earlier SYSTEMS computers, we keyed in the bootstrap loader (about eleven instructions) using the front panel switches. Of course, the time it took to load in the bootstrap was insignificant compared to the time it took to load the program on paper tape, on a Teletype, particularly if the tape got tangled, and tore. We finally started using Milar tape, strong enough to pull the reader right off the Teletype when it got tangled up. But this was a new 32 bit processor, designed to have a hard disk connected. The programmers won the battle for a machine that automatically booted, but the engineers won the war. The bootstrap was on the "here-is" response drum on the Teletype, a real problem when the so called "dumb" CRT terminals hit the market.

Finally, some importance was given to the development of a compiler. With the power of big computer in the frame and cost of a minicomputer, SYSTEMS decided to offer a FORTRAN compiler on the "SYSTEMS 32". They contracted with a supplier to provide a fairly good compiler that met the FORTRAN 60 standard but had a number of IBM Level 5 compiler extensions. Unfortunately, the leadership changed at Systems. The then male president (later to become a woman) decided to invest in supplying multi-terminal data entry stations for large IBM installations and subscribed to some questionable management techniques which encouraged the better engineers to leave and form two other companies. The Systems 32 with a decent compiler was a good solution to a large number of process control problems but without people who knew how to sell and support the computer in that market we saw more good effort go to waste.



One of the companies that was formed from SYSTEMS believed that "twenty-four bit computers were going to take over the world!" I went to work for the other one, Modular Computer Systems (ModComp). There I had the opportunity to design and and work with the team of four who implemented the FORTRAN compiler for their first machine (then revised for the second, third, and forth machines). FORTRAN was gaining acceptance in the "Process Control" market and even though few application programmers were using it, it was being listed as a requirement on mini-computer requests for quotes (RFQ). ModComp, in its first year proved that hoards of programmers approach is not necessary to develop software. Ten programmers produced three operating systems, all of the cpu and device diagnostic programs, a macro assembler, a text editor, a FORTRAN system and two software libraries. Among the interesting stories there is where our first major customer, ALCOA, sent a team of engineers and programmers to ModComp to tour our facility, which was at that time in a deserted grocery store, and watch us

run a benchmark on the prototype. We carefully explained that we had modified their programs because we hadn't implemented the "DATA" and "COMMON" directives yet. Would you believe that they bought our system? And were one of our best customers. Maybe they knew we needed all the help we could get.

In 1976, a group of process control engineers and programmers meeting as a working group of the FORTRAN Committee of the International Purdue Workshop on Industrial Computer Systems were successful in establishing a standard "Procedures for Executive Function, Process Input-Output, and Bit Manipulation."[6] Even though this standard seemed to have little effect on the mini-computer industry, it did signal that higher level languages were being used more by industry who now saw the advantages of higher level languages and standard software interfaces.

Software Engineering

*Everybody was talking about a "new" concept called "**Structured Programming**". Because "software development costs were so high and so unpredictable, programmers were encouraged to learn about and practice techniques that came from the concepts."*

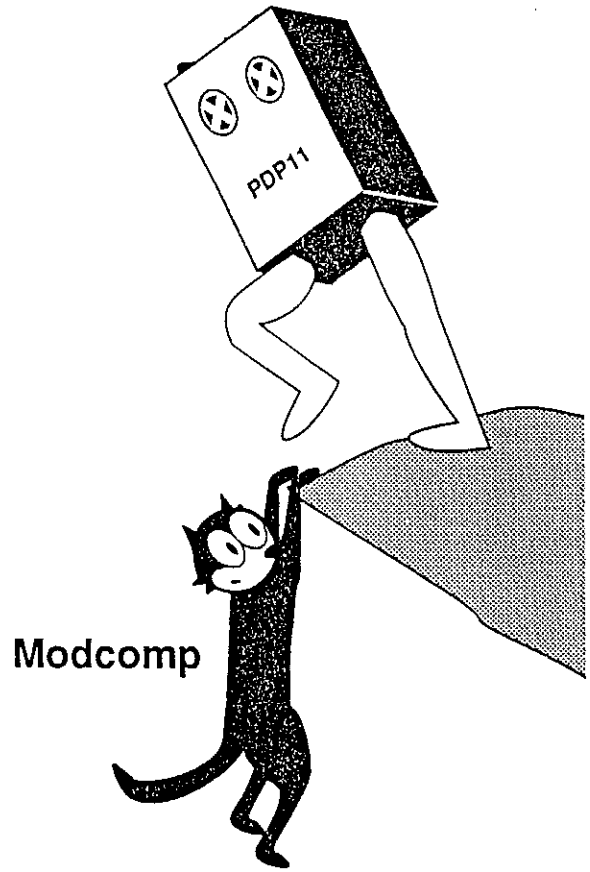
After we had gotten well into the design of the ModComp III, we saw the announcement of the new entry of Digital Equipment Corporation into the measurement and control market place. Something called a PDP-11. Somehow we were able to obtain preliminary marketing and engineering data on the PDP-11 and decided that it couldn't possibly be a threat to our ModComp III. Compared to the ModComp series of computers the PDP-11 was a RISC processor.

It is rumored that there is a marketing technique taught in some American colleges of business where the salesmen and developers are gathered into a conference room and determine the worst feature of a new product and what is going to cause the most sales resistance. Then the marketing department develops a campaign selling that feature as the most unique and beneficial feature in the produce. I can't swear that this technique was employed at DEC, but remember the advertising? Remember the "UNIBUS?"

According to the mailings I get from time to time, ModComp is still hanging in there. I left after it appeared that ModComp was not going to invest in another compiler like Pascal or even "C" or a systems programming language that we were proposing. I must note that after I had left and returned for a visit some time later, they had a Pascal Compiler that a customer had either donated or sold them. Ten years after I started design of the FORTRAN compiler, and I can testify that is wasn't the best design, it was still sold as a product (a much enhanced and improved version, it even had DATA and COMMON).

We had been right about the design of the PDP-11, later models of it had additional higher speed buses to enhance performance. Similar changes in the architecture of the ModComp II, IV and CLASSIC were not required.

BUT the winner: The PDP-11!



I left south Florida to move back closer to home. I was born in Birmingham, Alabama, but returned to Atlanta, Georgia which was about 180 miles away. I went to work for an engineering firm that used a matrix management system (that is a system where no person having the title of manager can really manage what the people in his department are doing but is still accountable for the success or failure of a project). Even though Pascal and "C" were available on a number of machines (I had used Primer on Pascal by David Gries as a teaching text) I was instructed that our applications would be written in FORTRAN because and I quote "there are more programmers that write in that language than in any other."

But really, you don't!

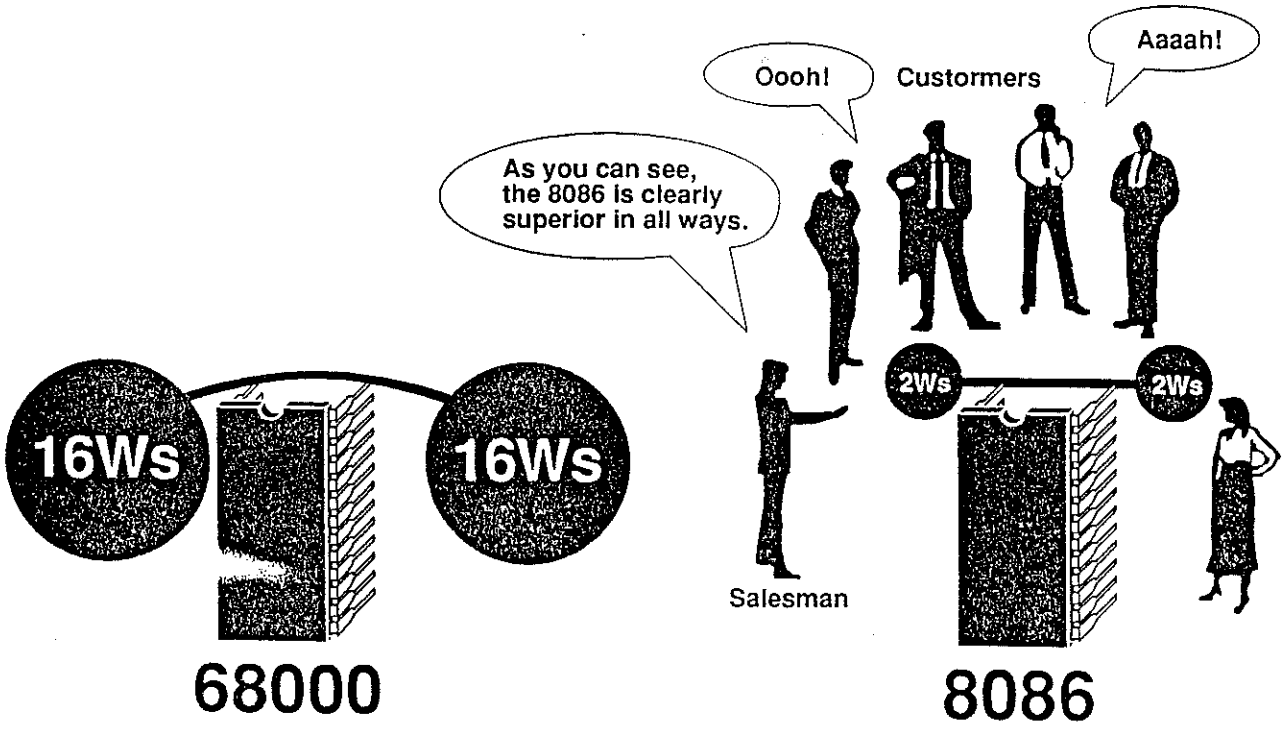
While I was still at ModComp I heard something about large scale integration (the ModComp II and IV used bit slice technology) and microprocessors. In Atlanta I watched as an engineer built a microprocessor controller, programmed both its operating system and application in assembly language and installed it in a local steel plant. Why was he using a microprocessor? "Because it is much cheaper and the instruction set is so simple that we don't need a higher level language to program it." After we totaled the development costs, the support costs, and maintenance costs we found that the microprocessors life-cycle cost estimates were going to be about the same or more than a mini-computer. Here we go again.

I was happier at doing than managing so I formed FBAI. As president I could be a "doer" and as a "doer" I found many opportunities where large organizations needed something done but couldn't get it done in their own organization (usually because the staff considered the job a prostitution of their abilities).

One of my opportunities was to do a study for Bailey Controls (they had unwittingly supplied the control system for the infamous Three Mile Island). The study resulted in a report[7] on the various 16 bit microprocessors available at the time, their high level language support, the performance of their high level language programs, and a proof project that a control system could be converted to a high level language from assembly language measuring the difference in performance and implementation. Part of the study involved running the Bailey Controls loaded version of the "Whetstone" benchmark on most of the available microprocessors of the day in addition to several minicomputers to evaluate performance of the language/machine. We had a measurement of 1,042 whetstones on the VAX 11/780 run using FORTRAN included for reference. The microprocessors for which a higher level language was available and which were made available for test were TI990/101M, INTEL 8086, and the Western Digital WD/90. The HP-85, an 8 bit processor was tested because of its availability— in other words, I had one. The Motorola 68000 was in beta test and no development software was available. Of particular note, the WD/90's operating system was version 2.0 of the p-System and the compiler was version 3.0 of the UCSD Pascal Compiler. Performance of these machines ran from 21.27 whetstones to 1.23. The slowest 16 bit processor: the 8086 at 2.06. The fastest: the WD/90 at 21.27. I understand that one of the remaining WD/90 processors is owned either by PECAN US or Eli Wilner a principal of PECAN US.

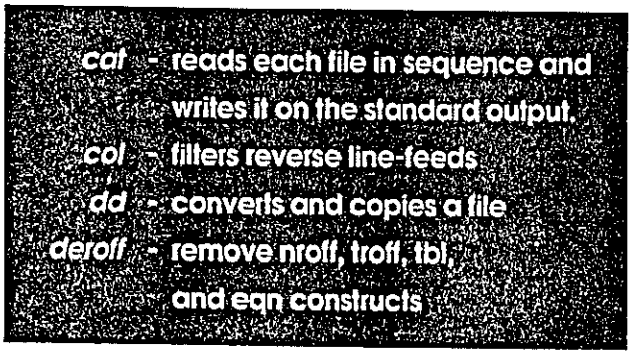
The winner: the Intel 8086! We had been done unto by big blue.

As a supplier in a free market, we are not always able to convince our customers of what is right. When the PC's started outnumbering their competition by 100 to 1 it is



not easy to convince all of your customers that your "Sage IV" is a better solution to their computer needs. Therefore, FBAI finally relented and purchased an IBM portable to support a number of customers that wanted me to do applications on the PC. Having already purchased the p-System for the HP-86, however, I was able to start the development of an application which was ultimately placed on a PC running a p-System which was purchased from SoftTech. The p-System was then and still is superior to the operating system for the 80x86 machines known as DOS— "Dumb Operating Stuff".

But the winner in the market place: "Dumb Operating Stuff."



During this period another operating system was under development at Bell Labs[8]. An operating system that was supposed to do every thing an operating system was intended to do. Many articles were written extolling the elegance of the structure and features of this new operating system. An operating system that took the Modula concept of opaque to the control language. No? "*Cat* reads each *file* in sequence and writes it on the standard output..", "col - filter reverse line feeds","dd - convert and copy a file", and I love this one, "deroff - remove nroff, troff, tbl and eqn constructs"[9]. In all fairness, even though that is not my purpose here, there are a number of directives that do make sense. While installing a system for one of my customers, I accidently typed in "date" and was surprized when the date was displayed on my terminal. It was in a funny format though, year, month, and day, instead of day, month, and year as we are used to seeing it. Most of us who are involved with control by computer understand that time is an important consideration. Things happen in milliseconds to which we must respond. There are also times we want to suspend our application program for short periods of time, and we can in this all things to all people operating system, we can "sleep", for a number of SECONDS. (NOTE: some of the later releases for the 80386 permit you to sleep for milliseconds).

HP 9915 Transporter-Robot-Carousel controller	UNIX Cell Controller
9915 Processor ROM 16 Kbytes of RAM 0 disk	80386 processor 4 Mbytes of ROM 80 Mbytes of disk
Controlls	**Controlls**
4 carousels 2 robots 2 level transporter 6 dispatch stations 1 return station	2 carousels 1 robot 1 level transporter 2 dispatch stations 1 return station

This new operating system is a model of efficiency. I just installed KitTrac, an inventory accounting, Automatic Storage and Retrieval System (ASRS) control and plant-wide material management node, in a PS-2 with a 1 MByte of memory and 30 MByte disk drive controlling a system with two carousels, a MOBOT robot , and a single level gravity conveyor system with two dispatch stations and one return station. Each device had its own motion controller with asynchronous interfaces. A "cell" controller was supplied which queued commands from KitTrac and managed the operation of the mechanical part of the ASRS. Using this wonderful new operating system running on a Compaq 386, the application required only 4 MBytes of RAM and 80 MBytes of disk storage. This may not sound so bad except an associate of mine (designed and wrote) and I (optimized) a program that solved the same control problem using an HP9915 (the controller version of the HP-85) in under 16 Kbytes and using the little cartridge tape drive that was available on the HP-9915 (actually, the program was stored on ROM and could run without the use of the tape drive). We controlled four carousels, two robots, and a two level powered transporter with 6 dispatch stations and one return station. AND our system was faster.

In order to achieve this efficiency, it was necessary to invent a high level language that looked like the machine for which it was written. Even though the popular story is that "C" was chosen for the name of this language because it was the second letter of BCPL[10], it has always been my suspicion that the language was named for the grade that Kernighan and Ritchie received in their "compiler design" class. Yes, I have had the misfortune of writing a program using that compiler after having used UCSD Pascal. The program was the implementation of a controller using a Z80 microprocessor for counting sponges and measuring the amount of liquid in them. "C" was the best choice available in this application. Fortunately, the preprocessor permitted me to substitute "BEGIN" for begin brace"{", "END" for end brace "}", and a few other choice changes so that I could read the program better, and I was using "ASE' with a rewritten

ASE Pascal Assistant[11] to assist me in keeping the syntax correct, so, I came out of the experience with few scars. When I got the SoftTech p-System, I found a membership application to USUS which I filled in and joined. With the CompuServe material in hand I logged on and found MUSUS which was promoting the upcoming "Stride Faire". I went, and was introduced to a machine that had the p-System as its main operating system. I somehow had missed the "Sage II" and "Sage IV". I thought I was in heaven. After all these years I found a bunch of folks doing it right. I was so enthusiastic that I sold a customer on buying two of my KitTrac systems on the then new Stride 440. I ported the system to the Stride in a matter of days. We installed the system and it ran so well for three years that the customer failed to properly back up his data-base. The Bernouli removable drive had a major crash and I was only able to recover about 30% of the data. My customer prior to the crash had decided that all microprocessor systems would be replaced with 80286 and 80386 based processors because "they were more available in the market" and asked me to convert from the 440s to PS-2 Model 80s. After warning him that he may not expect the same speed or reliability, I took his money and did what he asked.

In the fall of 1988 A. Robert (Bob) Spitzer, M.D. invited me to work with him on a grant using a neural network to decompose and analyze electromyograms (EMGs)[12]. The offer looked attractive not because of the money, but because I could work with the Stride, Modula-2 (Scenic Soft version), and neural networks. The data acquisition system (now called NSI-Rivka) needed to collect, display, and store data at rates above 50KHz. We needed to write handlers for special devices like AtoD converters and bitmap processors. UCSD Pascal, limited to a 64K data space, limited in handler development to assembly language, and limited in processing speed by the interpreter finally yielded to a native code generating Modula-2 with a rich set of system development features. To support NSI-Rivka we have developed the NSI-Daniel library which permits you to get as close to the hardware as you need to accomplish your objective. The "Artificial Neural Network" (ANN) system which is now being used to analyze a number of electrophysiological signals (EMG, EKG, EEG, and signals in the gastrointestinal tract) was implemented and is still being developed as a research tool using the Daniel library.

Sure, there are some things I dislike about Modula-2. The CARDINAL data type can only be justified by the fact that it gives you double the amount of positive INTEGERS in most applications. In our compiler, indexing may be done with only CARDINAL values which are limited to 16 bits. In artificial neural networks, as well as seismic processing and a number of other applications, arrays tend to have dimensions that are greater than 65576. We currently resort to pointers and ADDRESS arithmetic to handle these large arrays. There are even things I don't like about DANIEL. But I wouldn't be a good salesman if I told you those.

The SuperStride 740, in keeping with its predecessors, is a great machine. Fully compatible with the VME bus, designed for multi-processor applications (we are currently quoting a system with two 740s, one running UNIX and the other running an expanded version of NSI-Rivka), super fast, based on a NON-RISC processor, the 68030) it is clearly the leader technologically in its field. However, the survival of the Stride has been in doubt for the last several years. "Nobody has ever been fired for recommending IBM."

I want to see the Stride survive. If I had the investment capital, I would put it into Millennium, who now sells the Stride, to promote sales, development, and thus its survival. Failing that, I promote it when I can and help them any way I can. After receiving the source of the Modula-2 compiler for the Stride, I called a person in development (whom I will not name) to ask if they needed something fixed in the compiler to help them out. I was told that they "were not interested in Modula-2 because they could find "C" programmers six for a shilling."[13] I regret that I didn't say "You get what you pay for."

In November, 1989, ISO/IEC JTC1/SC22 Languages released a document requesting the review of a document entitled: DP10514: Information Processing Systems-Programming Language - Modula-2, and Letter Ballot. This document, printed on both sides of 8 1/2 by 11 paper, and barely fitting into a 1 inch three ring binder was the "Third Working Draft Modula-2 Standard". The standard is presented mostly in a variant[14] of a non-standard language called VDM-SL(Vienna Development Method-Specification Language), a language even more difficult to learn than "American". If the IFIP committee's 140 page report can sink a language such as ALGOL, then this "Standard" should blow Modula-2 out of the water.

The winner: probably "C".

Software Engineering

*Everybody is talking about a "new" concept called **"Object Oriented Programming System"**. Because "software development costs were so high and so unpredictable, programmers were encouraged to learn about and practice techniques that come from the concepts."*

"So What's your point?" as my daughter would say.

Are we doomed to use second best technology because it appears that the products available depend more on

marketing and positive economic feedbacks[15] than on excellence in technology? I think not. Your presence at this meeting is assurance that there are some who are always seeking excellence in their work and demanding excellence in the products they purchase.

Is there something we can do? You are doing some of it. You are supporting USUS-UK and you are buying UCSD p-Systems and using Pascal.

What more can we do?

1. Support each other in our conviction that what we are doing is indeed in pursuit of excellence.

2. Be more vocal. Let the world know that you believe in excellence and this is the best way to achieve it.

3. Keep your vendors informed as to what you expect of them. PECAN put a lot of effort into supplying "C" in the Power System. I feel that their development dollars could be better spent in bringing Modula-2 up to standard or writing a highly optimized native code generator for p-code. But I never told them.

4. Educate your co-professionals. If they do not know why your approach is better, patiently explain why, patiently explain again, then hit them in the head with the two-by-four.[16]

5. Listen to your co-professional. You may learn something. Even if his language of choice is "C". Remember that the obstacles he faces are higher than your own.

6. Educate yourselves! Give new ideas a chance before you discard them as uselsess. I'm currently involved in an "OOPS" study group and even though (at least so far) the concepts are not that far removed from "Modular programming" and "Structured Programming" I am finding that studying "OOPS" is forcing me to look at software design differently. Revisit old ideas. There were things an analog computer could do that were faster than digital processing. I suspect there still are. And one of these days engineers will see the advantage of the :"Balanced Ternary Number System"[17] Then we will all have to talk about "TRITS" instead of "BITS". Then Kernigan and Ritchie will have the opportunity to invent a new programming language "D" to run on a 12 trit machine using the operating system "VOJY."[18]

Thank you for permitting me to share some of my experiences and thoughts with you.

[1]van Wijngaarden, A. (Ed.). Report on the algorithmic language ALGOL 68. Numerische Mathematik 14 (1969),79-218

[2]Rosen, S., Programming Systems & Languages 1965-1975,Communications of the ACM, Volume 15, Number 7

[3]Kilburn,t., Edwards,D.B.G., Lanigan,M.J., and Sumner,F.H. One Level Storage Storage System, IRE Trans.Electronic Comput. EC-11 (Apr. 1962, 223-235.

[4]SYMBOL and META-SYMBOL REFERENCE MANUAL, Scientific Data Systems (Feb. 1966)

[5]This instruction was actually designed to be used in the "Power Failure Interrupt procedure" and actually turned off the memory so that it wasn't destroyed as power was lost in memory.

[6]Revised ISA Standard S61.1-1976, Industrial Computer System FORTRAN Procedures for Executive Function, Process Input-Output, and Bit Manipulation.

[7]Bearden,F. Report On High Level Language Evaluation, Bailey Controls Company

[8]Ritchie, D.M., Thompson, K.L., The UNIX Time-sharing System. CACM, July 1974

[9]Bell Telephone Laboratories, Incorporated, UNIX Programmers Manual, Revised and Expanded Version, Holt, Rinehart and Winston

[10]Jones, D., ANSI/ISO 'C' Standard, USUS(UK)LTD Newsletter, December 1990

[11]Karpinski, R., Pascal Assistant, a Fancy ASE Macro Module, USUS News and Report Number 14, July, 1985

[12]Spitzer, A.R., Hassoun, M., Wang, C., Bearden, F., Signal Decomposition and Diagnostic Classification of the Electromyogram using a Novel Neural Network Technique, Proceeding of the Fourteenth Annual Symposium on Computer Applications in Medical Care

[13]He actually said "a dime a dozen" but six for a shilling sounds better here.

[14]Section 0 Page 1, Third Working Draft Modula-2 Standard (D106)

[15]Arthur, W.B., Positive Feedbacks in the Economy, Scientific American, February 1990

[16]Refers to the story where one farmer who owned the mule with the reputation of being the best plowing mule in the county sold the mule to his neighbor. The farmer instructed his neighbor that he must have a felt lined stall, must be fed Jim Dandy Mule Feed at exactly 6 AM and 6PM, and that the harness must have brightly shined brass fittings. Having fullfilled all these requirements, the neighbor harnessed the mule in preparation to plow up his field. The mule wouldn't move. The neighbor called the farmer and expressed his distress. The farmer asked if everything had be done to specification. After checking the stall and harness he told the neighbor that he had done everything correctly. He then picked up a two-by-four board and soundly rapped the mule about the head after which the mule plowed up the field in half the time required by the neighbors other mule. The neighbor, nonplussed, asked the farmer "If I had done all of the things required of me, why did you have to hit the mule to get him to plow." The farmer replied "You have to get his attention, first."

[17]Knuth, D., Vol 2, Art of Computer Programming - Seminumerical Algorithms, Addison Wesley

[18]VOJY - UNIX promoted by one letter

MODULA-3 NEWS

Number 1, June 1991

Published by:
Pine Creek Software
Suite 300
305 South Craig Street
Pittsburgh, PA 15213

Editor: Samuel P. Harbison

Note: This is the on-line version of the Modula-3 News. It is missing photographs and graphics that are present in the printed version. For a copy fo the printed version, contact harbison@bert.pinecreek.com.

In This Issue:

Photograph of Modula-3 designers omitted.

Caption: Modula-3 language designers (left to right): Luca Cardelli, Bill Kalsow, Greg Nelson, Mick Jordan, and Jim Donahue. Jordan and Donahue were formerly with Olivetti.

Xerox PARC Adopts Modula-3

Xerox's Palo Alto Research Center (PARC), one of the leading computer science research institutions in the world, has adopted Modula-3 for use in many of their new projects.

Mark Weiser, Principal Scientist and Head of the Computer Science Laboratory at PARC, reports that CSL is on a multi-year track to switch their programming onto Modula-3 from Cedar (which Modula-3 strongly resembles). PARC intends to build a good Modula-3 programming environment along the way, based on their current Cedar environment. PARC has a strong commitment to language interoperability, so they will be intercalling Modula-3, Scheme, C, and Cedar. PARC will use their own portable, multi-threaded, garbage-collecting run-time, PCR, rather than SRC's run-time environment. PCR is in the public domain, and is already in wide use inside Xerox for research and products. The Computer Science Laboratory plans to make some supporting parts of the new environment freely available.

PARC has been the site of many important developments in computer science, including personal workstations, bitmapped graphic displays, the Ethernet network, and the Smalltalk object-oriented programming language. A focus of the Computer Science Laboratory today is the world of ubiquitous computing, and the infrastructure for supporting hundreds of independent wireless computers per person per office.

EDITOR'S CORNER

Will Modula-3 be Successful?

A while ago, I visited a professor of computer science who was familiar with Modula-3. He said that Modula-3 was probably the best language he'd ever seen. It was a shame that it wasn't likely to go anywhere because the momentum behind C++ was too great. I didn't want to hear this; I was just about to quit my comfortable job as vice president of a compiler company and become, in effect, a Modula-3 evangelist.

Will Modula-3 be just another good idea that languishes in journal articles and conference proceedings? It doesn't have to be. But, if we want to make Modula-3 successful,

we have to work at it. The first problem is exposure: a lot of people don't know that Modula-3 exists. Tell them about it. Pass on a copy of this newsletter or a Modula-3 brochure. Give them a brief summary of the features and the "feel" of the language.

Once their interest is piqued, they'll ask a tougher question: What are the risks and benefits of choosing Modula-3? Some of the risks are obvious: a new language, no PC-hosted compilers, no third-party libraries, and only one book. And, especially, no large user base to build confidence. (Software developers do travel in herds.) However, there are also some non-risks. Although Modula-3 is new, it's not an academic exercise; its features have been proven in other languages in commercial environments. Also, you don't need an up-front investment to try out Modula-3: all the necessary software is free from DEC.

But more important, consider the benefits of using Modula-3. Increased quality, reliability, and maintainability of your software. Better productivity. Lower training costs. Choosing the wrong language--one that lacks modern features, is unsafe, or is very complex--could easily cost a company millions of dollars over the lifetime of a large software product. If you had to debug 50,000 lines of code that Someone else wrote, would you prefer that code to be Modula-3 or C++?

Persistence is the key. Keep spreading the word. And, above all, keep writing those Modula-3 programs and tools.

Modula-3 Information

Harbison, S., "Modula-3," BYTE 15(12), November 1990. An introductory article.

Nelson, G. (ed), Systems Programming with Modula-3, Prentice Hall, 1991. Includes the final language reference and papers on the I/O library, threads, and the Trestle window system.

King, K., "What's New with Modula-2?" Dr. Dobb's Journal 16(6), June 1991. A good update on the Modula-2 family of languages, including Oberon and Modula-3; no specifics on Modula-3.

Harbison, S., Modula-3. A Modula-3 textbook that will appear in the fall of 1991 from Prentice Hall.

Usenet newsgroup comp.lang.modula3. (If you do not have Usenet access, you can send mail to m3-request@decwrl.dec.com and request them to forward postings to you.)

For more information, including scheduling seminars on Modula-3 for your school, user group, or business, contact Pine Creek Software.

Modula-3 on PC's

In line with its policy of encouraging the development of Modula-3 tools, Pine Creek Software has had discussions with several software vendors about putting Modula-3 on personal computers. To lower costs and development time, we have proposed modifying an existing programming environment--perhaps one for Object Pascal, Modula-2, or Ada. To date, no development plans have been agreed upon. Software vendors and developers interested in this project should contact Pine Creek.

Foreign Partners

Pine Creek Software is seeking representatives in Europe, Asia, Africa, and Australia to distribute Modula-3 software, documentation, and Modula-3 News. These organizations would be the primary source of Modula-3 information in their areas.

Overseas interest in Modula-3 is strong. The Modula-2 language has seen more success in Europe than in the United States. If you might be interested in helping promote Modula-3, please contact Pine Creek Software.

SPwM3 Appears

Prentice Hall began shipping Greg Nelson's Systems Programming with Modula-3 in May. SPwM3 is the first book on Modula-3 and is now the de facto reference for the language and core libraries.

The book, a collection of papers and original material, includes the updated language reference, discussions of threads and the I/O library, and a Trestle tutorial. The final chapter, "How the language got its spots," is an enlightening and amusing account of Modula-3 language committee deliberations.

The book (ISBN 0-13-590464-1) costs $25.00 and can be obtained at bookstores, from Prentice Hall, or from Pine Creek Software.

PRODUCT WATCH: Olivetti-Derived ToolKit Announced

On May 11, 1991, Mick Jordan of DEC SRC announced

the availability of the Modula-3 toolkit (m3tk), an evolution of the otherwise defunct Olivetti implementation of Modula-3. The code-generation and run-time aspects of the original implementation have been removed and the remaining Modula-3 code has been configured as a toolkit of reusable components, compatible with SRC Modula-3. The system is made available under the same licence terms as the SRC compiler. In addition, the original Olivetti sources are covered by a separate (less restrictive) copyright notice.

The extensible toolkit is designed to support the creation of Modula-3 program development tools and is structured around a compiler front-end (syntax and semantics) which uses a public Abstract Syntax Tree (AST) to represent program source. Among the several tools in m3tk is the beginning of an integrated, incremental program development environment. It includes the Modula-3 compiler front end, a tool to scan the file system for source file changes and recompile changed (and dependent) units, a pre-linker to analyze a program for completeness, a primitive browser, and a Makefile generator.

The toolkit is stored at gatekeeper.dec.com as the file /pub/DEC/Modula-3/m3tk/dist-1.0.tar.Z. Building information is contained in the file m3tk-install.notes, and on gatekeeper.dec.com as

/pub/DEC/Modula-3/m3tk/README.

A description of the system can be found in: Mick Jordan, "An Extensible Programming Environment for Modula-3" Proceedings of the Fourth ACM SIGSOFT Symposium on Software Development Environments, Software Engineering Notes, 15, 6, Dec 1990.

USER REPORT: Steve Harrison, Advanced Technology Development, DEC

"We are the graphics software arm of the RISC workstations group at DEC. Some time ago, I got interested in Jorge Stolfi's work on the ZZ-Buffer--a ray tracing acceleration method. Jorge works at DEC SRC, and coded his initial implementation in the Modula-2+ language, which runs only on their own proprietary hardware. I wanted to experiment with the algorithm, but on a platform available to non-SRC folks like myself. I began to translate the code to C, but then I discovered Bill Kalsow's Modula-2+ to Modula-3 converter. The converter does a reasonable job for most simple things. Then Jorge and I spent many happy weeks doing the rest of the translation and uncovering teething problems with the then-new SRC Modula-3 compiler.

"The SRC Modula-3 compiler is much better now than it was in the early days. I must say that Eric Muller and Bill Kalsow at SRC have done a first-rate job of fixing bugs, and making other changes we asked for. Bill and Eric continue to make significant improvements to the compiler now that they have some reasonable input.

"I use Mick Jordan's compiler front end tools [see Product Watch, above] for Makefile generation, quick syntax analysis and program checking. I recommend Mick's tools to anyone serious about developing Modula-3 programs.

"All in all, I'm very pleased with Modula-3, the SRC compiler--and in the way my own work on graphics algorithms is going!"

The Modula-3 Mark

The Modula-3 mark below was commissioned by Pine Creek Software as a distinguishing symbol for Modula-3 related products and services. If you would like to use the mark, send a description of your intended use to Pine Creek Software and we will forward the necessary licensing forms and artwork.

Modula-3 mark omitted

Pine Creek Software

If you'd like to reach us at Pine Creek Software, use any of these postal and electronic addresses:

Postal: Suite 300, 305 South Craig Street,
Pittsburgh PA 15213, USA
Phone & FAX: +1 412 681 9811
Internet: harbison@bert.pinecreek.com
AppleLink: D6463
CompuServe: 73577,2217
BIX: samharbison
GEnie: S.HARBISON

SRC MODULA-3

Version 1.6 Released

Version 1.6 of SRC Modula-3 was released at the end of March after a three-month beta test period. This version includes many bug fixes and includes new support for Sun-3, Encore, and Acorn computers. [See "SRC Modula-3 Hosts" for a complete list of hosts.] There are new interfaces to UNIX™ and type-safe offline storage

"pickles". Version 1.6 does not support the most recent set of language changes. *See "Which Modula-3?"*

SRC Modula-3 is presently the only available implementation of Modula-3. It includes a Modula-3-to-C translator; a prelinker; a "cc-like" m3 command; interfaces for X11R4, UNIX, I/O, and other useful facilities; profiling and coverage support; and documentation. Modula-3 programs can be debugged using the standard UNIX source debuggers. The SRC Modula-3 release also includes a test suite and all source code (mostly Modula-3). The software is provided "as-is," but it is currently being actively maintained and upgraded by a group at DEC SRC.

SRC Modula-3 may be obtained by anonymous ftp from the Internet site gatekeeper.dec.com in directory /pub/DEC/Modula-3/m3-1.6. UUCP and Easynet access via DECWRL are also available.

Porting Problems, Slow Migration of SRC Modula-3 to PCs

Many people have been interested in porting SRC Modula-3 to the various versions of UNIX on PC's. Although the PC's have adequate hardware resources, some characteristics of the SRC software have hindered the ports.

The problems come from the fact that the PC versions of UNIX are all based on System V UNIX, but the SRC software relies on some of the Berkeley UNIX features found in most of the workstation versions of UNIX. For example, System V traditionally limits file names to 14 characters, whereas Berkeley UNIX allows much longer names. The SRC software uses long names, and many file names would no longer be distinct if shortened to accommodate System V. A second problem is that the build scripts rely on symbolic links to connect various files and subdirectories within the directory hierarchy. System V supports only "hard links," which cannot be used to connect directories.

Changing the file names and build scripts would be manageable in a small software release, but SRC Modula-3 1.6 consists of about 2,900 files and 1,400 links in 350 directories (many of which are part of the included test suite). Most directories have their own build scripts.

SRC Modula-3 Hosts

Here is the complete list of hosts to which SRC Modula-3 1.6 has been ported. You should not expect problems porting to other models or releases of the hardware and software.

Computer	OS
VAX 8800	Ultrix 3.1
DECstation 5000	Ultrix 3.1
Sparcstation-1	SunOS 4.0.3
Sun-3	SunOS
Apollo DN4500	Domain/OS 10.2
HP 9000/300	HP-UX 7.0
IBM S/6000	AIX 3.1
IBM RT	IBM/4.3 (AOS 4.3)
IBM PS/2	AIX 3.1
Encore Multimax	UMAX 4.3 (R4.1.1)
Acorn R260	RISC iX 1.21

SRC Modula-3 does not run under DOS or OS/2, nor has it been ported to the various versions of PC UNIX.

Next: Version 2.0

The next release of SRC Modula-3--probably to be numberd version 2.0--is under development at SRC. Bill Kalsow, one of the SRC Modula-3 authors, reports that release 2.0 will contain the final Modula-3 language changes, the Trestle window package, and improved code-generation and linking. This will be a major upgrade and a lengthy testing period is expected. No release date was announced.

Who ya gonna call?

Comments of general interest about SRC Modula-3 should be posted on comp.lang.modula3. Mail sent to m3@decwrl.dec.com will also be posted on comp.lang.modula3.

To contact the developers directly, send mail to:
m3-request@decwrl.dec.com.

MODULA-3 TIPS

Default Initialization of References

Q I know that Modula-3 guarantees that variables never have "illegal" values. Does this mean that the following reference variable will be initialized to NIL?

```
VAR P : REF REAL;
```

A No. The language says that P will be initialized to

some value of its type; NIL is such a value and is quite convenient for compilers. However, a compiler could initialize P to a pointer to the value -34.88E20. As a matter of style, you should always write those initializations your program depends on:

```
VAR P : REF REAL := NIL;
```

The Modula-3 designers felt that requiring the default initialization of references to be NIL would, in effect, encourage a poor programming style.

Object Initialization

Q How can I define an initialization procedure for my object types?

A Modula-3 does not support automatic initialization procedures (constructors) for object types. Any client can invoke NEW on an object type, even if the type is opaque. The fields and methods of objects returned by NEW are initialized according to any initializers in the type declaration (or they are given default values). If you need custom initialization, the accepted convention is to provide an explicit method--usually named init--for the object type:

```
TYPE Class = Parent OBJECT (* new fields *) ...
METHODS init(any: Any): Class := ClassInit; ...
END;

PROCEDURE ClassInit(self: Class; any: Any):
Class =
BEGIN
self := Parent.init(self, ... ); (* if neces-
sary *)
(* Initialize new fields in self *)
RETURN self;
END ClassInit;
```

A client is expected to allocate and initialize an object this way:

```
VAR newInstance := NEW(Class).init(...);
```

The init method calls its parent's init function (if any), initializes its own fields, and returns the original object. The init function is a new method, not an override of Parent.init. This means the new init can have any signature. It's important that init not allocate the object, so that descendant types can use this same style for their initializations.

Ensuring That Objects Are Initialized

Q How can I ensure that clients call my object's init method?

A In your object type, include a boolean field, initCalled, with an initial value of FALSE. In the init method, set initCalled to TRUE. In your other methods, check this field to see if the object was initialized. If the object type is opaque, clients won't be able to forge the value of the initCalled field.

Object Finalization

Q How can I define a finalization procedure (destructor) for my objects.

A You can't, but maybe you don't have to. Destructors are often used to release an object's storage. In Modula-3, this is taken care of by the garbage collector. For end-of-program cleanup, there are library packages that let you register procedures to be called when the Modula-3 program is about to terminate.

Which Modula-3?

As software is upgraded and new books appear, there may seem to be several variations of the Modula-3 language. However, these variations are only temporary, and soon everyone will converge on a single standard.

Until recently, the Modula-3 language was defined by DEC SRC Report 52, Modula-3 Report (Revised), November 1989. However, in December 1990 Modula-3 was changed, as described in a memo from Greg Nelson, "Twelve Changes to Modula-3." This memo introduced generics, the EXTENDED floating-point type, new floating-point interfaces, and some smaller changes. The updated, "official" language is described in the first Modula-3 book, Nelson's Systems Programming with Modula-3 (Prentice Hall, 1991). SPwM3 is now considered the official reference for the language.

Unfortunately, the current versions of the SRC Modula-3 compiler and the Modula-3 ToolKit support only the older language described in SRC Report 52. Release 2.0 of SRC Modula-3 will support the official language. If you avoid generics, the EXTENDED type, and leave out the OVERRIDE keyword in object type declarations, you should be able to use SRC Modula-3 without too much difficulty.

Changes to the Modula-3 language are approved by the Language Committee, which consists of Luca Cardelli, Jim Donahue, Mick Jordan, Bill Kalsow, and Greg Nelson.

Subscribe to Modula-3 News

Modula-3 News is sent free of charge to people interested in Modula-3. To get your own subscription, send your name and address to Pine Creek Software.

SHORT NOTES

GNU MODULA-3 PROJECT
Prof. Eliot Moss is heading a research project at the University of Massachusetts at Amherst to create a Persistent Modula-3 system that would transparently merge objects in a long-term store with newly created objects in a running Modula-3 program. A second aspect of the project is an improved garbage collector that makes use of compiler-generated information to locate precisely all references to heap-allocated data. Moss' group is building a new Modula-3 compiler based on the GNU software from the Free Software Foundation. Once developed, the Modula-3 system would be distributed with other GNU software.

Upstaging Modula-2? In his June Dr. Dobb's article updating Modula-2, Kim King says that Modula-2 "is even in danger of being upstaged by its own offspring, Oberon and Modula-3."

BIG BLUE MODULA-3? The IBM Rochester (Minn.) Laboratory has been using Modula-3 heavily. They performed the ports of SRC Modula-3 to the IBM RISC System/6000, the IBM RT PC, and IBM PS/2 under AIX.

TRESTLE BETA IMMINENT A beta test version of the Trestle window system will shortly be available from DEC SRC. A tutorial on Trestle, which is written in Modula-3, can be found in Nelson's Systems Programming with Modula-3. The Trestle Reference Manual by Manasse and Nelson is now available as DEC SRC Research Report 68.

MODULA-3 DOES WINDOWS There are an increasing number of X Windows-related interfaces available in Modula-3. In addition to the X11R4 interfaces included with SRC Modula-3, there are the Trestle interfaces (see above) and an interface to TK/TKL (a window system developed at Berkeley). There also seems to be interest in Modula-3 interfaces to OSF/Motif and to ATK, the Andrew Tool Kit.

OOPSLA '91 TUTORIAL The OOPSLA '91 conference will feature a half-day tutorial on Modula-3. Sam Harbison will present the intermediate-level tutorial, which is aimed at programmers who have had some exposure to object-oriented programming concepts but who know nothing about Modula-3. OOPSLA '91 will be held October 6-11 in Phoenix.

End of Modula-3 News, Number 1, June 1991

From the NY Times

The annual Spring Comdex computer show in Atlanta earlier this month meant a booming business for the Bulletstop, an indoor firing range in suburban Marietta where customers can rent firearms and bullets to shoot anything they please, as long as it is already dead and fits through the doors. The Bulletstop gave Comdex visitors a chance to vent their frustrations by venting PC's, printers, hard disks, monitors and manuals with lead.

Paul LaVista, the owner, said about 10 groups of high-tech types came in during the Comdex show. "I'm not a computer whiz, but one group brought in what looked like a hard disk and blasted it," he said. "Another bunch brought in some kind of technical manual. The thing was enormous, about 2,000 pages. They rented three machine guns -- an Uzi, an M3 grease gun and a Thompson -- and when they were done it looked like confetti."

"It must have been quite a show," LaVista said of Comdex. "Doctors and computer types usually have a lot of pent-up anxiety, but these folks were dragging when they came in. When they left they were really up. The range looked like a computer service center after a tornado."

LaVista said PC's were popular targets year-round. "People are frustrated with them," he said. A year ago seven or eight men carried in a giant old Hewlett-Packard printer. "I ran an extension cord to it, and just as it started to whirr and spit out paper, they blasted it," he said.

PBS - an alternative to the p-System.

Part 1

By Stephen Pickett

The p-System was initially conceived in the late 1970's as a strategy for escaping the confusion arising in the marketplace, from the presence of a number of different styles of microprocessor. Its instigators (a group at UCSD led by Kenneth Bowles) were beginning to recognize the principle that in the near future, the cost of building software would start to exceed that of the computer hardware it was designed to run on. In the circumstances, they used and later enhanced a very stable design originating at least in part from ETH in Switzerland, the brainchild of Niklaus Wirth, the creator of Pascal. Their implementation became known as UCSD Pascal, and was hailed by many as a landmark, mainly owing to its ability to run on a microcomputer with 64K of memory. Looking back, this is still an incredible achievement measured by the megabytes of programs and hardware resources used in today's software development systems.

Over the last ten years I have been personally involved in designing and implementing enhancements to this architecture. This has been, on the whole, a rewarding task, especially when a large program written 5 or even 10 years ago on an Apple II or a 5MHz IBM PC can run under Windows, or OS/2, or DesqView. Perhaps in a future paper I can describe some of these improvements. However, they all suffer from the limitations inherent in the p-System itself which no one apparently foresaw when designing the original system. This has meant to me that, for as long as I can remember, people "outside" the circle of believers have been saying "the p-System is dead". Too bad they're wrong - a number of successful software projects are still being written and maintained in this so-called "dead" environment.

Notwithstanding the above, I have for a number of years been looking to design a more realistic replacement for the p-System. It is taken for granted that the resultant system will have certain "magic" properties that its owner can rely upon:

1. Small code file size
2. Re-usability of object modules (the INTERFACE concept still hardly understood "out there")
3. Infinite code space.
4. Stability and ease of porting of interpreted environment.

However the driving forces determining system design are different in the nineties:

1. No absolute requirement for compiled object code to be portable to multiple processor types.
2. Most advanced operating systems have many features originally provided by the p-System.
3. Run-time compatibility with other languages and language systems is crucial.
4. Memory, instead of being in short supply, is so plentiful that we cannot usefully address it all.

How in practice might we go about this? A good start would be to list those areas which need to be changed substantially in a new design. Each one of these constitutes an obstacle provided by the present p-System - each one, if overcome, an advantage for new system when it is built.

Obstacles posed by existing p-System: (potential advantages if removed in new system)

1. Fixed directory structure and (now) non-standard fle system.

When UCSD Pascal was created (later the p-System) there was no standard to violate - MSDOS did not exist. The limit of 77 files and no subdirectories (.SVOLs were added later) seemed adequate on the original microcomputer media, namely floppy diskettes. Subsequently, the world has adopted several standards, none of which even slightly corresponds to that of the p-System.

The CP/M (later MSDOS) restriction of 8.3 naming convention for file names (8 characters with a 3 character suffix) turns out to be a slight compatibility barrier for programs which assume they can use volume:filename.xxxx (the typical p-System filename). However this is more a problem for application developers to make their programs configurable enough to store the volume names in a variable, rather than hard-wired. Some solutions for mapping of names (eg SYSTEM.MISCINFO becomes SYSTEMMI.SCI, and FOOBAR.TEXT becomes FOOBAR.TXT) have also been implemented in existing p-Systems which are MSDOS-aware. Finally, the makers of DOS are more aware of foreign file systems and foreign file names, so perhaps it will be easy to create a foreign file system based on p-System directories at

some date in the future.

Nevertheless, the pressure is to be compatible with the existing drivers and disk layouts of common hosting systems, and PBS' goal is to help programs to be as independent of disk/driver/directory details as possible. In practice all this means is that volume name variables need to be long enough to hold a full pathname.

2. Dependencies on p-System operating System (PSOS) - much of it undocumented

A serious problem with the continued portability of p-code is the dependence of the p-code itself on structures assumed to be present at run-time in the memory space of the p-System operating system. Not only is this a restriction which causes code to fail to run if any of the said structures are changed, but the runtime architecture becomes non-re-entrant in a totally unnecessary way. An example of this is the way in which a constant may be passed as a parameter by pushing on the stack a pointer to the EREC of the segment from which it came. This is done automatically by the compiler during code generation!

3. Lack of separation of I/O component of interpreter

The p-System interpreter was traditionally a single file, with I/O and other interpreted functions built in. As a consequence there was no clear separation by interface of the I/O based components from the remaining functionality required in the runtime. In addition, the requirement for a multi-tasking interpreter complicated the implementation with details of the operating system which are, in an ideal world, better left untouched.

4. Multi-tasking foundation basically useless - no re-entrancy except for p-code programs

The multi-tasking provided by the p-System is quite slow, and has many hooks built in to the presence of the operating system and all the data structures in it, as referred to in 2 and 3. Much better, therefore, to rely on the multi-tasking executive present (DesqView, Windows, OS/2) of the host system and keep the interpreter clean - the only real requirement being re- entrancy. The only multi-user p-Systems implemented up to now have had a complete PSOS and interpreter loaded into memory for each task anyway.

5. Limited addressability of Data (64K minus Globals)

PBS provides two partial solutions to the addressability problem, which has been the most pressing problem posed by the p-System since developers started getting stack overflows during compilation. One is that with most of the PSOS (and its variables) disappeared from the stack-heap, about 8K bytes is freed up for actual programs. This guarantees that no existing p-System program runs out of memory. The second is simply that provision is made for calls to routines that can address memory outside the 64K data segment, although in general it is not possible to return an address from such a routine to the existing pcode as there is no way for the 16-bit-addressed p-System to deal with more than 16 bits of address information.

As hinted in the foregoing discussion, PBS was conceived as an answer to all of these problems, but with the intent of full compatibility of existing compiled pcode applications (some of them upwards of 2 million lines of UCSD Pascal source code). The resulting system has considerable advantages over the more widely-distributed products from Borland and Microsoft, with the special quality that, from the program and programmer's point of view, it is still close enough to the p-System that the universe looks the same as it always did. I was amused to note in a recent press release that Microsoft just announced a pseudo-code-generating version of their C compiler. The article referred to "the interpreted code, which Microsoft calls P-code". So the wheel turns full circle.

What actually started the project off was the necessity to provide a p-code interpreter that functioned correctly in 286/386 protected mode. For many of the reasons outlined above, there are too many things which the p-System does that are incompatible with the presence of a "real" operating system such as OS/2. In fact most of the interpreter and PSOS would simply be illegal in protected mode. On checking out OS/2 (Windows 3 protected mode didn't exist at that time), I discovered that it does many of the things the p-System has been doing for years (and more as well). A client of mine (who uses the p-System) actually went to a seminar, where OS/2 was described, and was amazed to hear a voice pipe up from across the room "that sounds like the UCSD p-System all over again". In fact the idea quickly became to generate p-code for OS/2 that conforms to the rules of the OS/2 environment and thereby get OS/2 to do much of the work the PSOS had been doing.

Next time I'll start to go into details of how we went about this sizeable task, and review the two critical components of PBS - the Interpreted DLL (IDLL) code file format, and the software which creates it, the Microtopia code server.

USUS Software Library Catalog

Summer 1991 Version
by
Keith Frederick

I'm proud to announce the introduction of the new USUS Software Library. Keeping good on our pledge to extend support to other high level languages, we have made a major upgrade to our Software Library. The new library contains over forty megabytes of new software (in addition to our 30+ volume library of UCSD Pascal software). The new software library includes the following sections:

- Source Code and Utilities for:
- Ada
- Modula-2
- Modula-3
- Oberon
- Pascal
- Turbo Pascal
- UCSD Pascal

[C and C++ Sections are still in development]

- Other Language Implementations

Includes complete implementations of languages that USUS does not currently support, such as SmallTalk, ProLog, Forth, etc. These are provided so that members can experience these languages at low cost. Often the language implementation comes with complete source (noted in the catalog if so).

- Documentation & Specification

Includes docs and specs on transfer protocols, graphic formats, networking protocols, etc.

- Demo & Evaluation Disks

Vendor provided disks for evaluating their software. These are provided at minimal cost and allow the user to try out the software before buying.

In addition to the commented file listings, the USUS Software Library Catalog contains the necessary price lists, order forms, donation forms, a section on the USUS PowerTools software, a definitions section, contents page, etc. In all, the document is about sixty pages (and growing!). Due to the size, we are making the Catalog available by three methods: disk, hardcopy or from our CompuServe forum, CODEPORT. See the order form in this issue for ordering information.

The new USUS Software Library Catalog is but one of a series of new services and benefits that will be introducing in the coming months, so stay tuned!

Keith Frederick
USUS Secretary & Administrator

Ada Software Library

Ada Volume 001

NAPP.....NASA's Ada Pretty Printer. No Source Code.
PAGER2.....Source Tools for creating, scanning, and extracting from paged files.
BD3.....Source illustrating modeling using Ada's tasking mechanism.
ADA-MET1.....Program (and source) to measure complexity of Ada source code.

Ada Volume 002

ADA-LRM1.....As below
ADA-LRM3.....As below
ADA-LRM4.....Disk based Ada Language Reference Manual. Needs Volume 003 to be complete.

Ada Volume 003

ADA-LRM2.....Disk based Ada Language Reference Manual. Needs Volume 002 to be complete.

Ada Volume 004

ADATU200.....Disk Based Interactive Ada tutorial.

Ada Volume 005

SPELL.....Complete Spellchecker and dictionary.

Ada Volume 006

ADACLI.....Ada Command Line Interface.
ASYNENTR.....Generic package for asynchronous entry callls.
BIT.....Bit manipulation routines for INTEGERs.
BPTREE.....Binary Plus Tree Generic Package.
CAS.....Source code analysis routine.
CLP.....Commannd Line Processor in Ada.
COUNTADA.....Counts number of Ada statements in an Ada fragment.
CPA.....Allows common pools in Ada.
CSET.....Character identification routines.
CSTRINGS.....Routines to manipulate null-terminated strings.
CUSTIO.....Low level character I/O routines.
DIPLOT.....Device independent 2-dimensional plotting package.
DLIST.....Doubly-linked list routines.
DSTR.....Dynamic string manipulation routines.
DUNIT.....Dimensional units routines.
ENV.....Environment interface package.
FGET.....Perform character I/O: GETC, UNGETC, GETCH, GET_CHAR.
FILECOMP.....Compare two ASCII files.
FLISTER.....Linked list routines.
FOF.....Report generator; formatted output generator.

Ada Volume 007

PARSER.....Generic parser like UNIX ARGC/ARGV.
PERMUTAT.....Display all permuations for an array.
PRIOR.....Prioritized Queue routines.
QSORT.....Quicksort routine in Ada.
RAN2.....Random Number Generator.
RAN3.....Another Random Number Generator.
RANDOM.....Yet another Random Number Generator.
RESERVE.....Determine if a word is an Ada reserved word routine.
SAFEIO.....Error checking input/output routines.
SDEPDG.....System dependency package.
SEARCH.....Binary and sequential seartching routines.
SLIST.....Single linked list routines.
SORTARRY.....Several array sorting routines.
STACK.....Abstract Stack manipulation routines.
STRCOMP.....Sophisticated string comparison package.
STRINGER.....String manipulation routines.

Ada Volume 008

STAB.....Block structured language tool to manipulate symbols.
TBD.....Tool to aid in the design of Ada software.
TESTLOG.....Unit to allow logging of execution for testing purposes.
TOD.....Time of day routines.
VDT100.....Routines to interface with VT100 terminal.
VLENGTHI.....Variable length record manipulation routines.

Ada Volume 009

A970.....Routines for a TVI 970 terminal
ASC.....Another Ada statement counter.
CALC.....Online calculator that also handles variables.
CBREAK.....Program to separate and combine text files.
CONSTRCT.....Ada source code project manager.
CREATETB.....Table builder and formatter.
ED.....Line based text editor.
ED2.....Another line based text editor.
FCHECK.....Yet another Ada statement counter.
MIMS.....Mobile Information Management System.

Ada Volume 010

PRP.....Interactive progoram to evaluate signal performance and noise of a set of RF propagation links. Allows 300 nodes, each with 15 receivers and 15 transmitters.

Ada Volume 011

WSMGS.....Three-dimensional Map Generation System.

Ada Volume 012

WP.....Complete word processor and formatter. Needs Ada Volume 013 to be complete.

Ada Volume 013

WP.....Additional Source for Ada Word Processor in Ada Volume 012 Needs Ada Volume 013 to be complete.

C Software Library

Under construction.

C++ Software Library

Under construction.

Modula-2 Software Library

Modula-2 Volume 001

M2CMP20.....Modula-2 Compiler. Source to compiler not included.
M2DOC20.....Documentation for Compiler above.
M2EXA20.....Utilities for Modula-2 compiler above.
M2LIB20.....Modula-2 Libraries for compiler above.
M2UTL20.....More Utilities for Modula-2 compiler above.

Complete Modula-2 Compiler, by Fitted Software Tools, with libraries and integrated editor, make utility, linker, makefile generator, and execution profiler.

Modula-2 Volume 002

MOD2SRC.....Source code that goes along with Tutorial below.
MOD2TXT.....16-chapter Modula-2 Tutorial. Print utlity included.
AFAI.....Log, square root, and reciprocal routines.
BALTREE.....Inserting/Deleting in an AVL-balanced tree.
BINGCD.....Compute Greatest Common Divisor or 2 natural numbers.
BTREE.....Insert/Delete elements in a B-Tree.
CMPISQRT.....Computer largest integer <= to the square root of a given integer.
CMPPOWER.....Raise integer to a positive power.
COMPLEX.....Complex number multiplication example.
CROSS1.....Generate cross reference table for all words in a text.
CROSS2.....As above but use hash-table instead of binary tree.
CRUNCH.....Eliminate extra white spaces between words.
DIVIDE.....Divide natural numbers by only using addition and subtraction.
EDIT.....Change text to be fully justified each line (flushed left and right).
FIBONACCI.....Compute fibonacci numbers two different ways.
FRACTION.....Compute table of exact fractions.
GCDLCM.....Computer Greatest Common Divisor and Least Common Multiple.
HARMONIC.....Compute harmonic function: H(n) = 1 + 1/2 + 1/3 +....+1/n.
KNIGHTST.....Find path of a knight on a chess board.
LIST.....Search routine to locate record in a list.
MARRIAGE.....Finds solution to the stable marriage problem(!).
MERGESOR.....Natural merge sort routine, uses 3 files and 2 phases.
OPTIMALT.....Finds optimally structured binary search tree for 'n' keys.
PALINDRO.....Finds integers whose squares are palindromes.
PERMUTE.....Compute 'n!' permutations.
PLO.....Skeleton compiler. Checks syntax according to given grammar.
POLYSORT.....Polyphase sort routine.
POSTFIX.....Convert infix expression to postfix.
POWER2.....Compute table of positive and negative powers of 2.
PRIMER.....Compute table of primes.
PRINTERP.....Plot function f(x)=exp(-x) * cos(2*pi*x) on the screen.
QUEENS8.....Find setting of 8 queens on 8x8 chess board where no queen checks another.
RECURREN.....Compute functions as truncated sums & determine recurrence relations.
SELECTIO.....Determine optimal selection of objects given object qualities.
SIEVE.....Sieve of Erastosthenes.
SORT.....Routines for Bubble, Shaker, and Quicksort sorts.
STRLIB.....Library module to handle string manipulations.
SUM10000.....Compute sum 1 - 1/2 + 1/3 - 1/4 +...-1/10000 four ways.
SUMOFCUB.....Find smallest positive int that can be represented as sum of two cubes.
TOPSORT.....Topological sort routine.
TREE.....Another Insert/Delete routine for binary trees.
WORDLENG.....Read a text and count # of words of length 1..20 and over 20.

Modula-3 Software Library

Under construction.

Oberon Software Library

Oberon Volume 001

OBERONM.....Oberon-M v1.1 compiler, docs, sample source. No compiler source. MS-DOS. Requires an 80186 or higher cpu.
MATRIX.....Advanced Matrix handling routines.

Pascal Software Library

Pascal Volume 001

NRPAS13.....Numerical Recipes software. Over 200 routines for scientific computation: integration, linear algebra, differential equations, and a lot more.
QPARSER.....Tools for developing compilers and translators. Provides grammar- directed "front-end" and several tools for developing the semantics "back-end" of a compiler.

Pascal Volume 002

MYSTIC.....Mystic ISO Pascal compiler & editor. MS-DOS. No source to compiler.
PASCSRC.....Source code to below Pascal tutorial.
PASCTXT.....14 Chapter Pascal tutorial.
SURPAS.....SURPAS Pascal compiler, editor, and run-time package. No source to compiler. MS-DOS.
CAL.....Displays Gregorian calendar for any month and year.
CPMCOP.....Transfer file from CP/M disk in Unit 5 to Pascal Disk in Unit 4.
XYPLOT.....Generate 2-D plots of X,Y data pairs.

Turbo Pascal Software Library

Turbo Pascal Volume 001

DATES.....Keep a list of memos, displays calendars.
DATEIT3A.....Several date manipulation routines.
DATETIME.....Date and time utility.
ANSICRT.....ANSI alternative to CRT unit.
ASYNC4.....Async4 communication routines, interrupt handler.
ATKYBD.....Set AT keyboard delay and typematic.
BLOAD.....Load BASIC BSAVE'd files.
BTREE4.....Unit (no source) for B-Tree indexing, data & file management.
CHAIN.....Chain facility for Turbo Pascal 4 and 5 users.
COLORDEF.....CONST file with constants for all text color combinations.
CRTPATCH.....Patch units compiled with CRT unit to allow use of TP5.0 TCRT Unit.
EXECWIN.....Keep child process output within specified window.
EXTEND.....Extends the number of open files DOS will allow.
LPT.....Printer Unit.
PSCREEN.....Save and display packed text screens/windows.
TESTEMU.....Test reinitialize emulator.

TPCLONE.....Routines for cloning typed constants into a program.
TPENV.....Routines for manipulating the DOS environment.
TPKEYS.....Keyboard installation program.
TPSPOOL.....Simple print spooler.
TPSTACK.....Unit to monitor stack and heap usage.
TPSWITCH.....Switching screens on dual monitor systems and writing to both screens.
TPTIMER.....Allow high resolution timing of events; good for measuring benchmarks.
COMMCALL.....Access COM1 port with interrupt handler.
COMSET.....Access COM1 and COM2 from Turbo Pascal.
CONCR4.....Concurrent Programminng Executive.
CONVERTB.....Convert Turbo Pascal data files from CP/M to MS-DOS.
CONV_P18.....Convert all TP reserved words to uppercase; includes 29 string-related functions implemented in assembly language.
CRCASM.....Speed optimized routine for cyclic-redundency check.
DIRSEL4.....Menu routine for selecting files.
ERRTRACE.....Unit for TP4 to display error traceback information.
EXEUTIL.....Tools to pack TP EXE header and patching stack/heap size without recompiling.
INLIN219.....Assembler designed to produce inline assembly code for TP3 and TP4. Comes with disassembler.
KTOOLS30.....Text screen menu and windowing routines.

Turbo Pascal Volume 002

LCOMMTP.....Routines,for TP4, to access full capabilities of PC's async comm ports.
LTCOMM50.....As above, but for TP5.
MAKEWIND.....Procedures for pop-up windows.
METAWIND.....Complete demo of MetaWindows graphic software.
MNDLBROT.....Generate and display Mandlebrots.
MOUSTOOL.....Tool interface utilities, TP5.

Turbo Pascal Volume 003

OAS.....Shareware version of the Open Architecture Screen Interface System.
OKI390.....Printer utility for Okidata 390-391 printers.

Turbo Pascal Volume 004

OPROSM.....Complete list of declarations for all documented routines in Object Professional.
PAS-SCI.....Dozens of scientifc pascal routines.
PASED11.....No Source. Programmer's editor for Pascal; specifically tailored for Turbo Pascal users.
PASLIB.....Misc. routines for screen handling (supports dual monitors), windows, and date handling.
POSBM.....String search routines.
PP50.....Pascal Pretty Printer.
PROCPARM.....Allows procedures to be Pascal parameters.
PULL15.....multi-level pull-down menu utilities, TP3. Needs QWIK30 & WNDW30 includes.
PULL20.....as above, for TP4. Comes with QWIK40 & WNDW40 units.

Turbo Pascal Volume 005

PULL55.....As Pull20 in vol. 004, but for TP5.5.
AMOUSE55.....Code to manage microsoft mouse.
EPB13.....Ed's Pascal Beautifier.
COMM_TP4 .....TP4 serial communication routines.
INDX18EU.....Indexed file utility.
INTRFC61.....Program to dump TPU files.
MOUSEFIX.....TSR; fixes bug in mouse driver when used with TP6.
NKTOOLS.....Misc. tools; I/O logging, text file device drivers, math and string routines.
PASMSG.....No source. Message filter for Turbo Pascal.
PCKSELFM.....Method for creating self-modifying EXE files that keep integrity under checking schemes.
PPP.....Pretty Pascal Printer for TP5

Turbo Pascal Volume 006

PASTUT24.....Turbo Pascal tutor with accompanying source.
PULL5X.....As PULLXX in Vol 005 and Vol. 004 but for TP5.X.
PULTP4.....Pulldown and pop-up menu system.
PASTOOLS.....ARGC, ARGV, and environment routines.
DSKRD-WR.....Absolute disk read/writes; TP6.
EVAL.....Evaluate Infix expressions.
FASTWR.....Fast writing to video memory.
GRAFDUMP.....TP4 or better Epson graphics screen dump.
HEXCOM.....Make COM file from HEX file.
INTERUPT.....Turbo Pascal interrupt handler code.
IO5150.....Device driver for IBM's 5150 PC serial port.
PCDISK.....Change volume lables and misc. other vol. and dir access routines.
PRINTDIR.....Directory show routine with options.
RAW-LPT.....Change file handler to process characters in raw mode.
RENAME.....File and directory renaming.
SELECTOR.....Several CHAR routines; many based on C routines in <ctype.h>.

Turbo Pascal Volume 007

QWIK30.....For TP3.
QWIK41A.....For TP4.
QWIK42.....For TP4.
QWIK55.....For TP5.5.
QWIK5X.....For TP5.X.
QWIK42B.....For TP4.
QWIK is made up of fast screen writing routines.
SLLIST.....Single linked list routine.
SEARCH.....String search routines.

Turbo Pascal Volume 008

OOPSIO.....No source, except DEF files. Group of I/O objects.
SFM.....Super File Manager; cross between X-Tree and Norton Utilities.
3D_MANDL.....Generate 3-D Mandelbrots.
TPRAT5.....Microsoft mouse driver for TP5.
QUARTIC.....Solve quartic equations.
PAINTTP.....Paint program, source code illustrates OOP.
STRG57.....String processing routines for TP5.5
STRG61A.....String processing routines for TP6.
PASENG.....Directory search engine.
SYS60A.....Replacement unit for TP6.0 to increase speed.

Turbo Pascal Volume 009

T301AS.....RS-232 support routines.
TJOOP11.....Source code illustrating OOP and polymorphism.
TOTDEMO.....TechnoJock's windowing and menu routines, v1.0; demo but fully usable.

Turbo Pascal Volume 010

SYST55C.....Replacement for SYSTEM.TPU in TP5.5.
STAY42.....Demos/Templates for creating "Stay Resident" programs.
THELP.....Permanent Resident Help Utility for Turbo Pascal.
TICKTOCK.....High Resolution Timing illustration.
TOAD_IAC.....Inter-Applications Communications fiddler.
TOADADD.....Add numeric strings to each other or numeric strings to ints.
TOADLONG.....LONGINT functions and procedures for TP3.
TOADLN5.....Much improved READLN for strings.
TLIST23.....Pascal Source Code lister.
THREED.....Toolkit for drawing and manipulating 3-D objects.
TDEBUG.....No Source. Source code debugger for TP3.
TAVID12.....Fast direct video routines in TASM.
T-REF.....Sophisticated Source lister and cross referencer.
TASKER4.....Non-preemptive multi-tasking.

Turbo Pascal Volume 011

TP-TSR.....TSR demo package, showing how to write sophisticated resident apps.
TP4MENU1.....Complete program shell foor developing user interfaces.
TP5MENU1.....As above, for TP5.
TPFORT12.....Access Microsoft FORTRAN routines from TP.
TPIO22.....Data entry controller.
TPMATH.....Several math functions: trig, complex, Bessel, matrices, etc.
TPPOP16.....Package containing tools to write TSR's.
TPPOPUPS.....Pop-up window and menu bar routines.
TPSPOOL.....Print spooler.
TPSTR121.....TP Rexx Strings unit, implemented in Assembly.
TPW32.....Quick multi-level windowing routine.
TPW60.....As above, or TP6.

Turbo Pascal Volume 012

TP6XMS.....Use Extended Memory from TP6.
TPA22.....Integrated compile-time assembler for TP4 or TP5.
TPENHKBD.....Activate or simulate Enhanced Keyboard.
TPFAST30.....Fast routines for bit functions, strings, screen handling, keyboard handling, files, etc.
TPTC17.....Turbo Pascal to C Translator and related files.
TPTC17SC.....As above.
TPTC17TC.....As above.
TPTCINFO.....As above.
TPZSFZ.....ZMODEM send/receive code.

Turbo Pascal Volume 013

TPJOYSTK.....Joystick routine for TP.
TPMUSIC.....Play music in the background.
TPU2ASM.....No source. Symbolic disassembler for TP units.
TSHELL12.....TP preprocessor shell; allows use of C-like preprocessor statements.
TSPEECH.....Speech driver and TP include file.
TSW.....Turbo Screen Works. Design and manage screens.
TTY.....Dumb terminal for COM1.
TURBO_TK.....TechnoJock Turbo Toolkit for TP4.
TRIDV183....."Door" creating utility for BBS's.
SVGABGI.....Super VGA BGI file and include.
VMATH10.....Vector and matrix procedures and functions.
UPCONV14.....Convert indentifiers/reserved words to different format.
ITP.....Source code for Inside Turbo Pascal 1990. Intermediate to advance topics.

Turbo Pascal Volume 014

TURBOGEN.....Creates TP source code to handle user I/O, file I/O, and error checking. User paints screen. Requires Turbo Database Toolbox.
TSPA2340.....No source except DEF files. Units for bit manipulation, run-time error handling, system/file information acquisition, string manipulations, etc.
TSPA2350.....As above, for TP5.
TSPA2355.....As above, for TP5.5.
TSPA2360.....As above, for TP6.
ZINDENT7.....Indenting and formatting functions for source code.

Turbo Pascal Volume 015

WINDOW.....Window BIOS demo and extender.
WINDOW34.....Multi-level random access windowing package.
WNDW40.....As above, for TP4.
WNDW42.....As above, for TP4.2.
WNDW55.....As above, for TP5.5.
WNDW_MSJ.....Simple text windowing package.
TWOSCRN.....Use two screens/monitors in TP.
DIALER.....Simple modem dialer.
LZH.....LZH algorithm in TP, compressing and decompressing.
HIGRAF.....High level graphic routines for scientific graphics.

Turbo Pascal Volume 016

TJOCKDOC.....Documentation for TechnoJock's Toolbox.
TJOCKSRC.....Source Code for TechnoJock's Toolbox.
TJOCKDEM.....Demos for TechnoJock's Toolbox.
Windowing, menu, user input, string formatting, directory liisting, etc. routines.

Turbo Pascal Volume 017

TUTORPAS.....Tutor on how to build a compiler, uses TP as a learning tool.
PNL001.....(Turbo) Pascal Newsletter, issue #1
PNL002.....(Turbo) Pascal Newsletter, issue #2
PNL003.....(Turbo) Pascal Newsletter, issue #3
PNL004.....(Turbo) Pascal Newsletter, issue #4
PNL005.....(Turbo) Pascal Newsletter, issue #5
PNL006.....(Turbo) Pascal Newsletter, issue #6

The Turbo Pascal Newsletters contain articles and accompanying source code to a variety of topics. Contains both beginner and advanced subjects.

UCSD Pascal Software Library

ALL THE SOFTWARE IN THE UCSD PASCAL SOFTWARE LIBRARY MAY NOT BE DISTRIBUTED TO NON-USUS MEMBERS.

UCSD Pascal Volume 001

COMBINE.TEXT 8 A simple little thing to combine 2 or more text files.
CPM.DOC.TEXT 8 Documentation of 8080/Z-80 interfaces and programs.
CPMCOPY.TEXT 14 An all-Pascal CP/M file copier.
CRC16.TEXT 6 Assembly-language CRC generator/checker for MODEM.TEXT.
CRT.I.O.TEXT 12 Very powerful, crash-proof data entry UNIT for CRT menus.
DISKREAD.TEXT 8 Assembly-language direct track/sector disk reader.
FORMAT.DOC.TEXT 30 Documentation (from Pascal News) for FORMAT.
FORMAT.TEXT 20 Large, wonderful Pascal program prettyprinter.
FORMAT1.TEXT 34 Part of FORMAT.TEXT (subfile).
FORMAT2.TEXT 28 Part of FORMAT.TEXT (subfile).
GETCPM.TEXT 8 Reads CP/M files --> UCSD-format disks.
GETCPM2.TEXT 8 Another version of GETCPM.TEXT.
GOTCHA.DOC.TEXT 28 Read all about UCSD's hidden gotchas for 8080/Z-80 users.
INITVAR.TEXT 18 Part of PRETTY.TEXT (subfile).
INOUTREADY.TEXT 10 Assembly-language routines: read/write to i/o port, etc.
INTRODUCTN.TEXT 26 A statement of purpose--why we are here, how we work.
L.TEXT 12 A short but effective text printer with several options.
MODEM.TEXT 44 1st of 2 D.C. Hayes Modem drivers (S-100 version).
MODEM1.TEXT 38 The second " " " " " .
PRETTY.TEXT 36 The second Pascal prettyprinter, from the Pascal News.
PRETTY.DOC.TEXT 10 Documentation for both FORMAT and PRETTY.
RWCPM.TEXT 6 Assembly-language direct disk reading/writing.
SIMP.TEXT 20 Program to produce random text; sounds "professional."
TYPESET.TEXT 12 Takes text from editors & right-justifies it.
READCPM.TEXT 6 Assembly-language direct disk read.
UNITS.DOC.TEXT 12 Re UNITS, SEGMENTS, & EXTERNAL routines.
COMMENT01.TEXT 10 Reviewer's comments about files on this disk.
VOL01.DOC.TEXT 8 You're reading it.

This volume was assembled by Jim Gagne from material collected by the Library Committee.

UCSD Pascal Volume 002a

512.DOC.TEXT 22 Documentation for 512-byte sectoring routines on 2A & 2B.
ACOUSTIC.TEXT 4 Use an acoustic modem with Pascal Transfer Program (PTP).
BOOTASM.TEXT 4 Assemble a file with UCSD assembler and save it on CP/M.
BOOTCPM.TEXT 4 Start up under UCSD and then boot up CP/M.
CPMIO.DOC.TEXT 14 How to alter the CP/M interpreter for fancy disk action.
DCHAYES.IO.TEXT 10 Use a Hayes modem with Pascal Transfer Program (PTP).
DELETE.LF.TEXT 10 Transfer a textfile to UCSD, then dump ASCII linefeeds.
DFOCO.DOC.TEXT 46 Documentation for DFOCO.ASM on Volume 2B.
H14.DRIVER.TEXT 28 Print out a text file on the Heath printer at full speed.
H19.DOC.TEXT 4 Notes on optimizing the Heath terminal for UCSD Pascal.
H19.GOTOXY 4 Textfile to compile your own GOTOXY for the Heath H19.
H19.MISCINFO 1 SYSTEM.MISCINFO for the Heath terminal.
HAZEL.MISCINFO 1 SYSTEM.MISCINFO for the Hazeltine terminal.
HEXOUT.TEXT 4 Pascal routine to print out integers in hexadecimal.
KBSTAT.TEXT 4 Another keyboard status routine, this time for PTP.TEXT.
LINECOUNTR.TEXT 8 Count the lines of a textfile.
MOVRAM.TEXT 4 Assembly-language routine for BOOTASM.
NEW.GOTOXY.TEXT 4 Let GOTOXY handle your CRT screen, too. Sample.
PE1100.GOTOXY 4 Textfile GOTOXY for the Perkin-Elmer 1100 (Fox) terminal.
PERUSE.PG.TEXT 4 Look over a textfile on your CRT one page at a time.
POLICY.DOC.TEXT XX How the Users' Group Library runs.
PRIME1.TEXT 4 Pascal routine to find prime numbers.
PRIME2.TEXT 4 Another prime-number generator.
PTP.DOC.TEXT 22 Documentation for the Pascal Transfer Program.
PTP.TEXT 96 The Pascal Transfer Program. Requires ASE to edit.
PUNCH.TAPE.TEXT 6 Send data from the UCSD system to the Heath paper punch.
RANDOMBYTE.TEXT 4 Assembly-language routine to access Z-80's R register.
READ.TAPE.TEXT 6 Complement of PUNCH.TAPE.
SHELLMSORT.TEXT 6 Sort a disk-based ASCII list.

SMARTREMOT.TEXT 22 Set up your machine as a smart remote terminal.
TIMING.DOC.TEXT 10 How to tune your disk drives for fast 512-byte sectors.
TVI912C.GOTOXY 4 Another GOTOXY text, this time for the TelVideo 912.
UPDATE.DOC.TEXT 18 Latest news on the UCSD Pascal Users' Group Library.
COMMENT02B.TEXT 20 Documentation for the second disk of this volume.
COMMENT02A.TEXT 12 Notes on all the programs in Volume #02A and #02B
WRITER.DOC.TEXT 4 Documentation for WRITER.
WRITER.TEXT 22 A quick but nifty text or source file printer.
VOL02A.DOC.TEXT 10 You're reading it.

XX = Programs withdrawn or not on the disk.

This volume was assembled by Jim Gagne from material collected by the Library Committee.

UCSD Pascal Volume 003

BLACKJACK.TEXT 20 The famous game. Allows negative funds.
CHASE.TEXT 22 Get away from robots, but don't get zapped by fence.
DEBTS.TEXT 26 Home finance program keeps track of bills.
OTHELLO.TEXT 12 Another famous game.
OTHELL1.TEXT 16 An include file.
OTHELL2.TEXT 16 An include file.
OTHELLINIT.TEXT 16 Subfiles for Othello
POLICY.DOC.TEXT XX How the USUS Library works.
PROSE.DOC1.TEXT XX Documentation for Prose. Copied from Pascal News
PROSE.DOC2.TEXT XX #15. Read before trying program.
PROSE.TEXT XX Also from Pascal News #15. Author is J.P Strait.
PROSE.0.TEXT XX Include files for Prose.
PROSE.A.TEXT XX
PROSE.B.TEXT XX
PROSE.C.TEXT XX
PROSE.D.TEXT XX
PROSE.E.TEXT XX
PROSE.F.TEXT XX
PROSE.1.5.CODE XX Object version. Will run under UCSD V.I.4 & V.I.5
REQUESTS.TEXT XX Programs and routines needed in this library.
SNOOPY.TEXT 14 A calendar program featuring the WW 1 flying ace.
STORE.DATA 2 A sample data file for DEBTS.TEXT
UNIVERSAL.TEXT XX Suggestions for a UNIT that will help remove hardware dependencies from Pascal Programs.
VOL03.DOC.TEXT 6 You're reading it.

XX = Programs withdrawn or not on the disk.

This volume was assembled by Jim Gagne from material collected by the Library Committee.

UCSD Pascal Volume 004

DBBUILDER.TEXT 38 Part of Keneth Bowles' database seed.
DBUNIT.TEXT 4 The major data accessing routines, allowing records of variable size and user-defined linkage & nesting.
DBUNIT.1.TEXT 18 Subfile of DBUNIT
DBUNIT.2.TEXT 32 " " "
DBUNIT.3.TEXT 34 " " "
DBUNIT.4.TEXT 30 " " "
KB.DATABASE.DOC 74 A detailed class manual to show you how to use it.
KB.DBDEMO.TEXT 4 Demo program to further document the system.
KB.SCUNIT.TEXT 16 Screen control unit with some very nice screen i/o.
KB.STARTER.TEXT 30 Help set up the data structures.
KB.TESTDB 32 A test database data file, used by DBDEMO.
COMPARE.TEXT 34 From the Pascal News No. 12; prints out textfile diff's.
COMPRESS.TEXT 8 Compress leading/strip trailing blanks; shrink files.
INDEX.TEXT 24 Expanded index to Jensen & Wirth--now you can find it.
USUS.NEWS.TEXT XX Learn all about the UCSD System Users' Society.
WUMPUS.TEXT 28 The game of Wumpus, elegantly implemented.
TEACH.WUMPUS 10 Documentation on the wunders of Wumpus.
WUMP.CAVE0.TEXT 4 One of several cave configurations you can select from
WUMP.CAVE1.TEXT 4 within the game; if you get bored with one, try
WUMP.CAVE2.TEXT 4 another.
WUMP.CAVE3.TEXT 4
WUMP.CAVE4.TEXT 4
WUMP.CAVE5.TEXT 4
COMMENT04.TEXT 12 Reviewer's comments about files on this disk.
VOL04.DOC.TEXT 8 You're reading it.

XX = Programs withdrawn on not on the disk.

This volume was assembled by Jim Gagne from material collected by the Library Committee.

UCSD Pascal Volume 005

ADDRS.DOC.TEXT 10 Doc for STRUCT, UPDATE, and GETSORT address database.
CRTINPUT.TEXT 20 A tuned-up string, boolean & textfile input package.
DIR.TEXT 16 See the directory, double-column & alphabetized, with file date & size, plus a list of unused areas.
DISKREAD.TEXT 26 Similar to UCSD's PATCH, lets you alter disks directly.
FMT.1.5.CODE 27 Frank Monaco's text formatter program, Version I.5
FMT.2.0.CODE 26 Same for versions II.0.
FMT.EXAMP.TEXT 16 Sample text for FMT - "before" version of READ.DISKR.
GETNUMBER.TEXT 30 Fancy, sophisticated integer & decimal input routines.
GETSORT.TEXT 12 Part of the mailing list "database" system.
HEXDECOCT.TEXT 18 Convert integers any way you want: HEX, DECimal, OCTal.
ID2ID.TEXT 36 From the PASCAL NEWS No. 15, converted by Frank Monaco. Lets you change one or more identifiers in Pascal

source to others of your choice.
MAKEMASKS.TEXT 18 Allows you to edit SUPER CRT masks, put them on disk.
MONACO.DOC.TEXT 14 Documentation for some of the files on this disk.
PEEK.POKE.TEXT 8 Thought you couldn't PEEK or POKE? Shows you how.
QUICKSORT.TEXT 4 Example of fast disk sorting algorithm.
READ.DISKR.TEXT 24 Documentation for DISKREAD and example of FMT at work.
READ.FMT.TEXT 82 Documentation for FMT.x.x.CODE and example of output.
SCREENCNTL.TEXT 4 Example of SEPARATE UNITS, with some nice routines.
SOFT.TOOLS.DOC 18 Doc for CRTINPUT, GETNUMBER, MAKEMASKS, & SCREENCNTL.
SP.TEXT 14 Allows your line printer to follow FORTRAN conventions.
STRUCT.TEXT 8 Part of the mailing address "database" system.
UNIT.GOOD.TEXT 22 Should be called WONDERSTUFF; solves those nagging terminal dependencies for good, plus allows you to read the directory from any disk, get date, etc.
UPDATE.TEXT 22 Part of the mailing address "database" system.
VOL05.DOC.TEXT 8 You're reading it.

This volume was assembled by Jim Gagne from material collected by the Library Committee.

UCSD Pascal Volume 006

PTP.BUSH.TEXT 154 An original Pascal Transfer Program, by Mark Gang, edited by Randy Bush. Baud rate is selected by inter-modem dialogue; choice of radix-41 or true binary file transfer; improved algorithms speed up data transfer.
BAUD.A.TEXT 6 The suffix "A.TEXT" indicates 8080 assembly-
CTS.A.TEXT 4 language files used by PTP; the function of each
DIALER.A.TEXT 6 routine is described by "PTP-FILES.TEXT". Note that
DTONEDET.A.TEXT 4 these routines are highly processor- and modem-
DTRON.A.TEXT 4 specific.
HANGUP.A.TEXT 4
KBSTAT.A.TEXT 4
MODEMINI.A.TEXT 4
MREAD.A.TEXT 4 Rewrite these files for your machine.
MRECSTAT.A.TEXT 4
MWRITE.A.TEXT 4
RI.A.TEXT 4
RINGING.A.TEXT 4
SH.A.TEXT 4
SYSNAME.TEXT 4 User ID (just a few characters) used by PTP.
PTP-FILES.TEXT 8 PTP documentation - which files are which.
PTP-INST.TEXT 6 PTP documentation - how to set it up.
PTP-USE.TEXT 20 PTP documentation - how to use.
FORMAT.TEXT 6 A corrected and updated FORMAT (from USUS Vol.
FORMAT.1.TEXT 16 1), so that now it works reasonably well. Most bugs
FORMAT.2.TEXT 24 are gone (except the program still has trouble with
FORMAT.3.TEXT 18 extended comments and may terminate prematurely).
FORMAT.4.TEXT 20 Format options are now MENU SELECTED!! Finally, it
FORMAT.5.TEXT 22 accepts nearly all valid UCSD Pascal syntax. It is
FORMAT.6.TEXT 24 handy for reformatting Pascal source to fit on CRT screens of different sizes.
FMT.64MASK.DATA 5 Mask for FORMAT menu for 64-column screens.
FMT.64MENU.TEXT 4 Source for above data; needs MAKEMASKS from Vol. 5.
FMT.MASK.DATA 5 This menu mask is for 80-column CRTs and Apples.
FMT.MENU.TEXT 6 Source for the above data; needs MAKEMASKS from Vol5.
FMT.NEWDOC.TEXT 20 Documentation on these changes to FORMAT.
FORMAT.DOC.TEXT 30 Copy of original FORMAT documentation from Vol. 1.
BANNER.TEXT 20 This program was donated by David Mundie. Prints a banner vertically on print-out paper, with up to 6 lines or 7-inch letters.
VOL06.DOC.TEXT 8 You're reading it.

This volume was assembled by Jim Gagne from material collected by the Library Committee.

UCSD Pascal Volume 007

FASTREAD.TEXT 8 Dan Dorrough's unit (used in MAP) that speeds up readln for strings by a factor of about 10.
MAP.TEXT 38 The precompiler from PUG News #17 that allows Pascal
MAP-A.TEXT 42 macros, fancy constant expressions, nested INCLUDE
MAP-B.TEXT 32 files, conditional compilation, and more, converted by Dan Dorrough of PCD Systems.
MAP.DOC.TEXT 18 Documentation from PUG News.
PRXREF.TEXT 30 David Lewis' superb Pascal cross-referencer and source
PRXREF.TBL.TEXT 34 lister with several nice features: follows INCLUDE
PRXREF.OPT.TEXT 24 files, adds line numbers that match those of the UCSD
PRXREF.INI.TEXT 24 compiler, and marks procedures/functions in the xref
PRXREF.UTL.TEXT 28 list.
PRXREF.PFI.TEXT 38
KEYHIT.TEXT 4 Assembly language keyboard poller (from USUS).
PRX.DOC.TEXT 68 Clear and thorough (!) documentation for PRXREF.
PROC.REF1.TEXT 44 Procedure/function cross-referencer from PUG News #17. It died when tried on a large program with many cross-references, but works fine with smaller sources. Use is obvious.
PROC.REF2.TEXT 22 Pat Horton's own procedure cross-referencer.
VOL07.DOC.TEXT 6 You're reading it.

This volume was assembled by Jim Gagne from material collected by the Library Committee.

UCSD Pascal Volume 008

ARCHIVER.TEXT 10 Save & retrieve disk images.
CHAIN.TEXT 6 Chain to another program.
CHAIN.1.TEXT 4 Demo programs...
CHAIN.2.TEXT 4
COPYBLOCKS.TEXT 6 Copy blocks to a file by block number.
CRMBLEV1.2.TEXT 14 Break up long files for editing with the UCSD editor.
D.TEXT 20 Revised disk directory lister.
DISKSORT.TEXT 12 Revised sample sort.
ERROR.DATA.TEXT 4 Messages used by FILEUNIT.
EXHALEV2.1.TEXT 6 Send data to remote port.
INHALEV2.1.TEXT 8 Receive data from remote.
FAST.SEEK.TEXT 10 Greatly speed SEEK procedure.
FILEUNIT.TEXT 32 General-purpose file handling routines.
GLOBAL.II0.TEXT 30 GLOBALS for UCSD system II.0.
GLOBAL.III.TEXT 36 .....III.0.
LINECOUNT.TEXT 10 Counts lines in text files or entire volumes.
LISP.TEXT 28 Public-domain LISP interpreter.
LISTER.TEXT 6 List text files.
MAILER.DOC.TEXT 10
MAILER.TEXT 22 Mailing list facility; sounds NICE.
MODEMV2.2.TEXT 8 Rework of program on Volume 2A.
MULDIV.Z80.TEXT 6 Part of Z80.SEEK.
PERUSEV4.6.TEXT 14 Look over a text file forwards & back; FAST!
RECOVER.TEXT 6 Find program source text after zapping a directory.
REM.TERM.TEXT 34 Hardware-independent communications utility.
REM.UNIT.TEXT 32 One implementation of new USUS standard remote unit.
SCREEN.TEXT 6 Western Digital screen control unit.
SCREENUNIT.TEXT 10 Terminal-independent screen control from Volume 5.
UNITS.DOC.TEXT 22 Documentation for FILEUNIT.
WRITERV7.2.TEXT 34 Nice text printer, updated from Volume 2A.
Z80.SEEK.TEXT 8 Fast seek routine specific to Z-80's.
COMMENT08.TEXT 20 Reviwer's comments about files on this disk.
VOL08.DOC.TEXT 8 You're reading it.

This volume was assembled by Jim Gagne from material collected by the Library Committee.

UCSD Pascal Volume 009

ADV.TEXT 34 Source for ADVENTURE.
ADV.DOC.TEXT 20 Read this documentation on setting up the program.
ADV.MISCINFO 4 Tells ADV your screen dimensions.
ADVINIT.TEXT 22 Run this program to set up ADV's data files.
ADVS1.TEXT 42 These are the text files used by ADVINIT.
ADVS2.TEXT 8
ADVS3.TEXT 22
ADVS4.TEXT 10
ADVS5.TEXT 14
ADVS6.TEXT 38
ADVS7.TEXT 6
ADVS8.TEXT 4
ADVS9.TEXT 4
ADVS10.TEXT 4
ADVS11.TEXT 4
ADVSUBS.TEXT 42
ADVVERB.TEXT 42
CASTLES.TEXT 36 A board game for two or more players, in which you
CASTLES.DOC 6 and your opponents are warlords plundering each other, raising armies, etc.
SPACEWAR.TEXT 20 Fast action for two players shooting it out in their space ships...it'll require work to get it running on your machine.
STARTREK.TEXT 6 A Pascal version of the classic game.
STAR.PART1.TEXT 24
STAR.PART2.TEXT 22
STAR.PART3.TEXT 22
COMMENT09.TEXT 10 Reviewer's comments about files on this disk.
VOL09.DOC.TEXT 6 You're reading it.

This volume was assembled by Jim Gagne from material collected by the Library Committee.

UCSD Pascal Volume 010

BENCHMARK.TEXT 28 Jon Bondy's benchmark with some added goodies
BENCHMARK1.TEXT 18 An include file
NEW.BFS.TEXT 22 Interesting, but what is it?
KRUSKAL.TEXT 22 ditto
KRUSKAL.1.TEXT 28 An include file
CATALOG.TEXT 26 A file manager data base.
CATALOG.1.TEXT 24 An include file
CATALOG.2.TEXT 10 An include file
BTREE.GET.TEXT 28 A large implementation of a B-tree algorithm.
BTRE.FIND2.TEXT 22 Include files
BTREE.DEL1.TEXT 22
BTREE.DEL2.TEXT 32
BTREE.PRNT.TEXT 26
BTREE.DOIT.TEXT 30
BTREE.DCLR.TEXT 10
BTREE.STD.TEXT 8
BTRE.FIND1.TEXT 22
BTREE.INIT.TEXT 12
BTRE.FILE.TEXT 30 The main program and the documentation for B-tree.
COMMENT10.TEXT 6 Reviewer's comments about files on this disk.
VOL10.DOC.TEXT 6 You're reading it

This volume was assembled by George Schreyer from material collected by the Library Committee.

UCSD Pascal Volume 011

MAIL.DOC.TEXT 100 Documentation for MAIL.
MAIL.E.G.TEXT 10 A sample source document
MAIL.LETT.TEXT 4 A sample form letter
MAIL.INFO.DATA 4 A sample form
SCREENOPSX.TEXT 12 A Screen Control unit for version II.0
SCREENOPSA.TEXT 14 A Screen Control unit for Apple
MAIL.TEXT 18 The main program
MAIL1.TEXT 16 an include file

MAIL2A.TEXT 32
MAIL2B.TEXT 32
MAIL3.TEXT 16
MAIL4.TEXT 26
MAIL5.TEXT 14
MAIL6.TEXT 20
MAIL7.TEXT 22
MAIL8.TEXT 12
MAIL9.TEXT 10
MAILINITEG.TEXT 4 A sample data form
MAIL.FORM.TEXT 6 A sample form
MAIL.READ.TEXT 12 Documentation on the files in MAIL
MAIL.INIT.TEXT 32 Converts a text file into a MAIL data file
CHASE.TEXT 22 A reworked version of the game on Volume 3
BLACKJACK.TEXT 22 A reworked version of the game on Volume 3. This
BJACK.1.TEXT 16 one splits pairs, doubles down, and has insurance.
COMMENT11.TEXT 4 Reviewer's comments about files on this disk.
VOL11.DOC.TEXT 8 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library Committee.

UCSD Pascal Volume 012

WINDOWS.TEXT 20 A screen window unit. Slow but nice. Needs IV.0.
W.SEGS.TEXT 28 an include file
W.IO.TEXT 20 ditto
W.IMPLN.TEXT 32 ditto
WFILER.TEXT 26 A demonstration program for WINDOWS. Acts like the Filer
W.DOC.TEXT 22 Documentation for WINDOWS.
OFFLOAD.TEXT 24 A command line interpreter which replaces the USCD command prompt. Needs IV.0
OFF.INFO.TEXT 4 an include file for OFFLOAD
OFF.START.TEXT 8 ditto
OFF.READ.TEXT 8 instructions for using OFFLOAD
OFF.DOC.TEXT 26 Documentation for OFFLOAD
HELP.DISK.TEXT 4 A help file for OFFLOAD
HELP.KEYS.TEXT 4 ditto
HELP.OFF.TEXT 4 ditto
HELP.UTIL.TEXT 6 ditto
NEW.PAGE.TEXT 4 A data file for OFFLOAD
NEW.TEXT 4 ditto
PARAM.INFO.TEXT 4 ditto
MAKE.PAGE.TEXT 4 A utility for OFFLOAD
PRINT.MEM.TEXT 4 Part of the benchmarks. Displays memory available.
PRINT.HEAP.TEXT 6 Analyzes heap usage. Needs IV.0 and timer support.
BENCH.USUS.TEXT 22 Jon Bondy's benchmark with some added goodies.
BENCH.PCW.TEXT 12 Personal Computer World benchmark. Needs timer support.
BENCH.SWAP.TEXT 4 Segment swap benchmark. Needs IV.0 and timer support.
BENCH.BYTE.TEXT 6 The infamous Byte benchmark
CPROC.TEXT 16 Another command line interpreter. Needs IV.0
VOLS.SMAC 4 Data for CPROC
STARTUP.TEXT 20 A startup program which sets the prefix and date.
AUGMENT.TEXT 42 A program which adds timing info to a Pascal source file so that timing data can be obtained.
ANALYZE.TEXT 24 Analyzes the results of AUGMENT. Needs timer support.
R.ANALYZE.TEXT 24 Ditto except uses reals.
DISK_COPY.TEXT 8 A disk copy and verification program.
LMFORMAT.TEXT 18 A simple Pascal source formatter.
COMMENT12.TEXT 16 Reviewer's comments about files on this disk.
VOL12.DOC.TEXT 8 You're reading it.

This volume was assembled by George Schrier from material collected by the Library Committee.

UCSD Pascal Volume 013

DECLARE.TEXT 22 Include files of RUNON
INITC.TEXT 26 ditto
DOPAGE.TEXT 34 ditto
READNU.TEXT 30 ditto
READLN.TEXT 16 ditto
MAIN.TEXT 22 RUNON main program. A nice fast text formatter.
SYSGEN.TEXT 4 Compile this file to make RUNON.
RUNON_DOC.TEXT 4 RUNON documentation in un-formatted form.
INTRO_DOC.TEXT 10 an include file of the documentation.
HOWTO_DOC.TEXT 12 ditto
DOT_DOC.TEXT 26 ditto
DEFALT_DOC.TEXT 4 ditto
SPEC_DOC.TEXT 10 ditto
ERR_DOC.TEXT 12 ditto
TECH_DOC.TEXT 16 ditto
TAXNAMES.TEXT 18 Generates form line names for FIT.
TAXTABLE.TEXT 24 Generates an out-of-date tax table for FIT.
TAXCALC.TEXT 20 an include file for FIT. The Federal Income Tax Program
TAXSTART.TEXT 8 ditto
TAXRW.TEXT 10 ditto
TAXPRINT.TEXT 16 ditto
TAXEDIT.TEXT 22 ditto
FIT.TEXT 20 The main program of FIT.
STARTUP.TEXT 22 A version II.0 startup program with date and prefix set
PDATE.TEXT 4 a unit for STARTUP
SPIN.TEXT XX an external procedure for Startup H-89 only.
ERRORDATA 1 This data file should have been on Volume 8.
SCREDIT.TEXT 36 A screen form generator, simple but it works.
SCRGEN.TEXT 18 Converts output of SCREDIT to Pascal soruce
TYPES.TEXT 6 an include file for SCREDIT and SCRGEN
COMMENT13.TEXT 6 Reviewer's comments about files on this disk.
VOL13.DOC.TEXT 8 You're reading it.

XX = Programs withdrawn or not on the disk.

This volume was assembled by George Schreyer from material collected by the Library Committee.

UCSD Pascal Volume 014

COPVOL.TEXT 18 Jon Bondy's disk copier (will copy Z-80 type boot blocks).
COPVER.ASM.TEXT 8 an external procedure for COPVAL.
COPFILE.TEXT 12 Jon Bondy's file copier.
COMPFILE.TEXT 10 A binary file comparison program.
GAME.TEXT 26 A game with a maze and demons by Jon Bondy.
GAME1.TEXT 18 an include file of GAME.
DEFAULT.GPAT 4 a data file for GAME.
CROSSES.GPAT 4 ditto
SPARSE.GPAT 4 ditto
GAME.ASSEM.TEXT 4 an external procedure for GAME (keypress).
SCANNER.TEXT 14 A nifty program which looks through a disk for a string.
KBSTAT.TEXT 4 A keypress routine for an H-89.
BONDYSTUFF.TEXT 14 Jon's documemtation.
HOME_LOAN.TEXT 10 A simple program to calculate simple loans.
BANNER.TEXT 22 Prints banners in BIG letters.
STOCK.TEXT 22 A Stock Market game.
STOCK.DATA.TEXT 6 a data file for STOCK.
STOCK.DOC.TEXT 6 documentation for STOCK.
SRCCOM.TEXT 18 A nice source comparison program.
FASTREAD.TEXT 8 a unit for SRCCOM.
REFERENCE.TEXT 24 A simple but effective procedural cross referencer.
REFER.INC.TEXT 22 an include file of REFERENCE.
8080CONV.TEXT 26 Converts 8080 instructions to Z-80 instructions.
LOOK.UP.TABLE 15 A data file for 8080CONV.
TABLE.TEXT 18 The text of the data in case you have to recreate it.
REFORM.TEXT 8 A utility for 8080CONV.
CALENDAR.TEXT 12 A perpetual calendar (requires an H-19).
DAYOFWK.TEXT 8 Calculates the day of the week for any date.
LISTINFO.TEXT 36 Generates a report of your *SYSTEM.MISCINFO.
SORTS1.TEXT 6 These four programs are demos of four different sorts.
SORTS2.TEXT 6
SORTS3.TEXT 6
SORTS4.TEXT 6
HEXDUMP.TEXT 12 Dumps blocks in hex.
ROMAN.TEXT 10 Converts decimal dates to Roman numerals.
COMMENT14.TEXT 8 Reviewer's comments about files on this disk.
VOL14.DOC.TEXT 8 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library Committee.

UCSD Pascal Volume 015

HSM.UROOT.TEXT 22 A RemoteUnit for the Hayes SmartModem (uses UNITSTATUS).
HSM.UINC1.TEXT 16 an include file of HSM.UROOT.TEXT.
STD.UNIT.TEXT 24 A RemoteUnit for a dumb modem (uses UNITSTATUS).
TERM.MAIN.TEXT 20 Bob Peterson's terminal emulator program.
TERM.LOG.TEXT 14 an include file of TERM.MAIN.TEXT.
HSM.UINC2.TEXT 14 ditto
TERM.EMUL.TEXT 10 ditto
TERM.INIT.TEXT 22 ditto
TERM.UTIL.TEXT 22 ditto
CONTENTS.TEXT 14 Documentation for TERM.MAIN and Bob's RemoteUnitsx.
SMTREMV5.TEXT 26 A terminal emulator specific to the LSI-11.
IOUNIT.TEXT 8 a unit for SMTREMV5.TEXT.
TOMUS4.C2.TEXT 24 Mike Ikezawa's terminal emulator.
REMUNIT.L3.TEXT 28 Mike's RemoteUnit (specific to an LSI-11).
SET_BREAK.TEXT 4 an external procedure for REMUNIT.L3.TEXT.
CLR_BREAK.TEXT 4 ditto
TOMUS3.C1.TEXT 10 comments for TOMUS4.C2.TEXT.
TOMUS3.A.TEXT 22 Documentation for TOMUS4.C2.TEXT.
COMM.TEXT 24 Jon Bondy's terminal emulator.
REMTALK.TEXT 24 Transfer files between two closely coupled UCSD computers.
TELETALKER.TEXT 24 Randy Bush's Communications program. Uses a RemoteUnit.
A.///.REMU.TEXT 72 Arley Dealey's Remote Unit for the Apple ///.
COMMENT15.TEXT 10 Reviewer's comments about files on this disk.
VOL15.DOC.TEXT 8 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library Committee.

UCSD Pascal Volume 016

P.TEXT 24 A simple text formatter which is easy to use.
P.INC.TEXT 16 an include file for P.
INV.TEXT 8 An inventory management program from Pat Horton.
ISSUE.TEXT 10 an include file for INV.
BASPROC2.TEXT 12 ditto
ADD.TEXT 8 ditto
REPORT.TEXT 20 ditto
BASPROC.TEXT 18 ditto
INV.DOC.TEXT 34 documentation for INV.
HORTON.DOC.TEXT 4 Pat Horton's comments on files he submitted.
Z80.SEEK.TEXT 8 A fast Z80 seek procedure. Should have been on Volume 8.
CHECKBOOK.TEXT 26 Jim Gagne's checkbook balancer.
USUS.INV.TEXT 36 A USUS disk order entry program.
INVCS.MASK.DATA 5 a data file for USUS.INV.
8.INCH.TEXT 8 Prints 8" disk labels.
APPLE.LABL.TEXT 6 Prints Apple disk labels.
CRTINPUT.TEXT 22 A unit used by Jim's programs.
GETNUMBER.TEXT 30 Another unit used by Jim's programs.
ASE.HEADER.TEXT 6 The declarations for the ASE Header Page.
BUNIT.TEXT 12 Mike Adams's B-tree unit.
BDEBUG.TEXT 6 an include file of BUNIT.
BHKEEP.TEXT 6 ditto
BINTERN.TEXT 24 ditto
BIO.TEXT 8 ditto
BMAIN.TEXT 22 ditto
BDRIVER.TEXT 12 A main program which uses and demos BUNIT.

BDOC1.TEXT 30 Excellent documentation for Mike Adams's B-tree.
BDOC2.TEXT 30 More documentation on the b-tree.
BDOC3.TEXT 16 Even more documenation.
COMMENT16.TEXT 8 Reviewer's comments about files on this disk.
VOL16.DOC.TEXT 8 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library Committee.

UCSD Pascal Volume 017

SYSTEM.A.TEXT 48 UCSD Pascal Version I.3 operating system.
SYSTEM.B.TEXT 44 an include file of the operating system.
LINKER.TEXT 8 The Linker.
FILER.TEXT 50 The Filer.
YALOE.TEXT 44 Yaloe (I.3 didn't have a screen editor).
GLOBALS.TEXT 22 Globals for the system.
COMP.A.TEXT 40 First file of the Compiler.
COMP.B.TEXT 34 an include file.
COMP.C.TEXT 44 ditto
COMP.D.TEXT 52 ditto
COMP.E.TEXT 46 ditto
COMP.F.TEXT 34 ditto
BOOTER.TEXT 4 The bootstrap copier.
XFER.TEXT 6 A single disk file transfer program.
VOL17.DOC.TEXT 6 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library Committee.

UCSD Pascal Volume 018

SEGMAP.1.TEXT 34 Arley Dealey's version independant segment mapper.
SEGMAP.2.TEXT 28 an include file of segmap.
ODMSCU.TEXT 10 One Damn More Screen Control Unit (for Segmap).
LIFE.TEXT 32 The game of LIFE.
LIFE.INC.TEXT 8 an include file for LIFE.
BLACKBOX.TEXT 38 A guessing game based on particle physics.
BLACK.DOC1.TEXT 6 a help file for BLACKBOX.
BLACK.DOC2.TEXT 6 ditto
TELE.TEXT 18 Keeps a data base of telephone traffic.
SORTUNIT.TEXT 18 A three-way key sort unit.
SORT2.TEXT 12 A sort program.
DEBUG.A.TEXT 22 The UCSD I.3 debugger.
DEBUG.B.TEXT 44 an include file of the debugger
BENCHMARKS.TEXT 10 An overview of the benchmarks on this disk.
PWROF2.TEXT 8 A Pascal benchmark program.
8QUEENS.TEXT 6 ditto
NUMBERIO.TEXT 8 ditto
ANCEST.S.TEXT 6 ditto
PRIMES.TEXT 6 ditto
QUICKSORT.TEXT 6 ditto
QUR.TEXT 6 A simple benchmark to measure "system" speed.
STARS.TEXT 6 A simple I/O benchmark.
COMPKILLER.TEXT 6 A benchmark designed especially to crash your compiler.
WHETSTONE.TEXT 12 The famous WHETSTONE benchmark.
WHET.DOC.TEXT 16 Some notes on WHETSTONE, taken from MUSUS.
SIEVE.TEXT 8 The infamous Byte Benchmark, as standard as possible.
LONG_INT.TEXT 12 Tests long integer operations.
INTRINSICS.TEXT 18 Tests system intrinsics.
SEGMASHER.TEXT 8 A segment swap speed tester.
REPORT.DOC.TEXT 6 Some simple instructions for running the benchmarks.
REPORTFORM.TEXT 8 A form for recording the results of the benchmarks.
BENCH.USUS.TEXT 26 Jon Bondy's benchmark (again).
BONDY_FORM.TEXT 8 A form for recording the results of BENCH.USUS.
COMMENT18.TEXT 6 Reviewer's comments about files on this disk.
VOL18.DOC.TEXT 8 You're reading it.

This volume was assembled by George Screyer from material collected by the Library Committee.

UCSD Pascal Volume 019

DEC & RX01 Specific

MODEM.PAS.TEXT 6 A simple routine to redirect I/O.
DEC.INDEX.TEXT 6 An index of the DEC specific software already in the USUS Library.
2K.KEY.TEXT 12 Some patches for the 2k Key.
PATCHES.TEXT 30 A summary of the patches to the UCSD DEC
PATCH.CONT.TEXT 20 interpreters to make them work right.
SYSTEM.INTERP 18 The interpreter for I.3.
SYSTEM.PASCAL 110 The operating system for I.3.
XFER.CODE 3 A single drive file transfer program for I.3.
BOOTER.CODE 2 The I.3 bootstrap copier.
SETUP.CODE 13 The I.3 Setup utility.
MAKE_I.3.TEXT 4 An RT-11 command file to assemble and link the I.3 interpreter.
RX11.TEXT 26 The I.3 floppy driver.
EIS.TEXT 4 A conditional assembly definition.
MACROS.TEXT 14 Macro definitions for the I.3 interp.
MAINOP.TEXT 64 Part of the guts of the I.3 interp.
TRAPS.TEXT 36 ditto
PROCOP.TEXT 64 ditto
LP11.TEXT 10 The printer driver for I.3.
RXBOOT 2 The boot block for I.3.
PVM.MAC.TEXT 16 A RAMDISK handler for II.0.
ZAPRAM.TEXT 6 A utility to initialize RAMDISK on boot.
COMMENT19.TEXT 8 Reviewer's comments about files on this disk.
VOL19.DOC.TEXT 6 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library Committee.

UCSD Pascal Volume 020

UNLPATCH.1.TEXT 26 The University of Nebraska Lincoln disk patch utility
UNLPATCH.2.TEXT 24 it displays its data in octal.

UNLPCH.DOC.TEXT 10 doc for above.
AUTOPSY.TEXT 10 Divides a file into small enough pieces for the sytem editor.
SCREEN.H19.TEX. 10 A screen control unit for Autopsy
SCREEN.TEXT 6 A terminal independant version of SCREEN.H19.
LWRCASE.TEXT 8 Converts a file to lower case but leaves literals intact.
UPRCASE.TEXT 8 same but to upper case.
HOME_LOAN.TEXT 14 A simple minded simple loan calculator.
SIGFIG.19.TEXT 14 Get another "significant" figure, or maybe even more!
OTHELLO.TEXT 28 Steve Brecher's OTHELLO game, originally on Volume 3
OTHELLO.1.TEXT 26 an include file. This game is a real killer!
BASE.TEXT 6 A numeric base converter, works nice.
H19UTIL.TEXT 24 A screen control unit for BASE. Modify it for other terminals.
NUMBER2.TEXT 12 A unit for BASE.
FASTREAD.TEXT 10 Another version of a fast string read routine.
ESCORT.DOC.TEXT 8 Documentation for the Jonos ESCORT BIOS
E.BOOT.TEXT 14
EBIOS.TEXT 8
BIOS.CONST.TEXT 8
BIOS.SERPT.TEXT 22
BIOS.DISKS.TEXT 26
BIOS.PHONE.TEXT 4
BIOS.DATA.TEXT 10
SXFR.SVCS.TEXT 26
FORMATTER.TEXT 8
TRANSPORTR.TEXT 42
BOOTMAKER.TEXT 10
EBIOS-GENR.TEXT 4
EBOOT-GENR.TEXT 4
FMT-LINK.CODE XX
FMT-GENR.TEXT 4
TRANS-GENR.TEXT 4
FORMATTER.CODE XX
SXFR.SVCS.CODE XX
TRANS-MGR.CODE XX
BOOTR-GENR.TEXT 4
BOOT.WRITE.CODE XX
BOOTMAKER.CODE XX
E.BOOT.CODE XX
E.LOAD.BOOT 1
EBIOS.CODE XX
E.LOAD.BIOS 4
BOOT.WRITE.TEXT 16
COMMENT20.TEXT 8 Reviewer's comments about files on this disk.
VOL20.DOC.TEXT 8 You're reading it.

XX = Programs withdrawn or not on the disk.

This volume was assembled by George Schreyer from material collected by the Library Committee.

UCSD Pascal Volume 021

This Volume is specific to Apple machines

PTP.HAYES.TEXT 32 An Apple][version of PTP.BUSH from Vol. 6.
PTP.HAYE1.TEXT 28 Should be useful for anyone with Version II
PTP.HAYE2.TEXT 30 based system. As is, it is an Apple/Hayes Micromodem
PTP.HAYE3.TEXT 28 version ready to run. The modem drivers must be
PTP.HAYE4.TEXT 26 changed for another modem.
PTP.HAYE5.TEXT 26
MDMDRVR.TEXT 16 External modem support, Hayes Micromodem][.
PTP.APPLE.CODE 39 Compiled, linked, ready for Apple/Hayes.
SYSNAME.TEXT 4 Sample user ID (mine).
PTP-USE.TEXT 20 Original PTP documentaion.
PTP.DOC.TEXT 14 Doc. for this version.
CHAREDIT.TEXT 24 Create new graphic character set - Apple][.
DOSCAT.TEXT 12 Read catalog on Apple DOS disk.
DOSTRANS.TEXT 8 Program using DOSstuff to transfer text from DOS.
DOSUNIT.TEXT 22 Unit to read Apple DOS disks from Pascal programs.
DOSTR.TEXT 24 Documentation for the above.
ACPMCOPY.TEXT 12 Apple disk version of program from Vol 1 to fetch CP/M files. All in Pascal, so it's slow, but it works.
GETFUNCS.TEXT 14 Input strings, reals, integers, Boolean from CONSOLE:.
STAT.DOC.TEXT 10 Documentaion for a series of statistical programs
ANOVA2.TEXT 12 by Phil Elder.
ANOVA1.TEXT 10 Stat programs.
SEG.TEXT 8
DSTAT.TEXT 10
TIND.TEXT 8
TDEP.TEXT 8
CORR.TEXT 8
CH12.TEXT 8
CH11.TEXT 8
VOL21.DOC.TEXT 8 You're reading it

This volume was assembled by George Schreyer from material collected by the Library Committee.

UCSD Pascal Volume 022

Graphics Programs for the Terak (LSI-11) Computer

GRAPH.DOC.TEXT 40 Documentation for units and programs on this disk.
POST.DOC.TEXT 24 Documentation for POST_ENTRY.TEXT.
REAL_INPUT.TEXT 8 Unit to input real numbers from the console.
REVIEW.TEXT 14 Unit to facilitate running these programs with a "dumb" Hiplot plotter instead of the Terak.
POST_ENTRY.TEXT 28 Unit to input functions from the console or a file and to evaluate these functions.
SCRN_STUFF.TEXT 8 Screen control unit for these programs.
PLOTTER.TEXT 16 Unit to drive "dumb" Hiplot plotter.
GRAPHICS.TEXT 28 Fundamental graphics unit for both Terak screen and "dumb" Hiplot plotter.
FACT_STUFF.TEXT 14 Math unit for factorial, log factorial, and related calculations.

FUNC.TEXT 24 Plot functions entered from console.
POLAR.TEXT 22 Plot polar functions ent.red from console
DISTRIB.TEXT 24 Calculate and plot normal, Poisson, and binomial distributions.
SINES.TEXT 14 Plot sine functions of various types.
HISTOGRAM.TEXT 20 Plot histograms using data from file or console.
HISTOGRAM.DATA 2 Sample data file for the above.
CURVE_FIT.TEXT 28 Polynomial curve fitting and plotting.
CONTOUR.TEXT 28 Plot contours of 3-dimensional surfaces.
TRIANGLE.TEXT 40 Constructs triangle with minimum input and plots result.
TRAVERSE.TEXT 42 A calculating and plotting program for surveyors.
IVP.TEXT 42 "Solves" differential equations by Euler and 4th order Runge-Kutta techniques and plots solution.
VOL22.DOC.TEXT 8 You're reading it.

This volume was assembled by Henry Baumgarten from material collected by the Library committee.

UCSD Pascal Volume 023

STARGAME.TEXT 28 A simple, but captivating game.
RNDTEST.TEXT 16 A random number generator and some simple tests
.RNDDOC.TEXT 14 Documentation for RNDTEST.
IOUNIT.TEXT 16 A unit to quickly read and write characters and strings.
IOTEST.TEXT 8 A program to test and benchmark IOUNIT.
IODOC.TEXT 10 Documentation for IOUNIT and IOTEST.
SPELLER.TEXT 24 A Pascal spelling checker.that uses IOUNIT
DICT.TEXT 78 A small literal dictionary for SPELLER.
SPELLDOC.TEXT 18 Documentation for SPELLER and DICT.
DF.DOCUM.TEXT 26 The Display Filer Documentation.
DF.IV.0.TEXT 86 Display Filer for IV.0.
DF.IV.1.TEXT 96 Display Filer for IV.1.
VOL23.DOC.TEXT 8 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library committee.

UCSD Pascal Volume 024

ADVX1.TEXT 48 A data file.
ADVX2.TEXT 8
ADVX3.TEXT 22
ADVX4.TEXT 10
ADVX5.TEXT 16
ADVX6.TEXT 42
ADVX7.TEXT 6
ADVX8.TEXT 4
ADVX9.TEXT 4
ADVX10.TEXT 4
ADVX11.TEXT 4 The last data file.
ADVXCONS.TEXT 4 An include file of Adventure.
ADVXINIT.TEXT 24 Creates the Adventure data file from the data source files.
ADVXVERB.TEXT 44 An include file of Adventure.
ADVXINIT4.CODE 8 A version 4 code file of ADVXINIT.
ADVXINIT2.CODE 9 A version 2 code file if ADVXINIT.
ADV.MISCINFO 4 Specifies screen size.
ADVX2.CODE 50 An un-linked vers. n 2 code file of ADVX
ADVXSUBS.TEXT 20 An include file of Adventure.
ADVXSEGS.TEXT 28 An include file of Adventure.
ADVX.TEXT 38 The main program of Adventure.
ADVX.DOC.TEXT 22 Documentation of Adventure.
VOL24.DOC.TEXT 6 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library committee.

UCSD Pascal Volume 026

Universal Data Entry Documentation and Code Files
-->Version IV.x ONLY<--

Sources on UCSD Pascal Volume 025

UD.INTRDOC.TEXT 16 UDE Documentation.
UDE.1.TEXT 32
UDE.2.TEXT 26
UDE.3.TEXT 32
UDE.4.TEXT 34
UDE.5.TEXT 32
UDE.6.TEXT 8
SD/DEFINE.CODE 30 UDE Sub-programs.
SH.SCREEN.CODE 21
UD.SORT.CODE 45
UD/COPY.CODE 41
UD/LIST.CODE 39
UD/MAINT.CODE 23
UD/SORT.CODE 25
UD/UDE.CODE 3 UDE Main Program.
SH/SCREEN.UNIT 21 A necessary unit.
USERLIB.TEXT 4 Install this as your USERLIB.TEXT.
UD/LIST.SCRN 8 A couple of data files.
UD/SORT.SCRN 8
README.1ST 8 Read this FIRST!!!
VOL26.DOC.TEXT 6 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library committee.

UCSD Pascal Volume 027
FreeForm (a 3-D spreadsheet) Sources
Documentation and code files on UCSD Pascal Volume 028

--> IV.x ONLY <--
4 word reals recommended

FF.FREEFRM.TEXT 22 The main program of FreeForm
FF.COPY2.TEXT 18 an include file
FF.DATA1.TEXT 20 an include file
FF.FORMS3.TEXT 22 an include file
FF.DATA3.TEXT 20 an include file
FF.DATA2.TEXT 20 an include file
FF.DATA5.TEXT 22 an include file
FF.FORMS5.TEXT 26 an include file
FF.FORMS1.TEXT 24 an include file
FF.FORMS2.TEXT 14 an include file
FF.FORMS4.TEXT 16 an include file
FF.MISC2.TEXT 20 an include file

FF.BASICS1.TEXT 30 an include file
FF.COPY1.TEXT 26 an include file
FF.BASICS3.TEXT 24 an include file
FF.BASICS2.TEXT 24 an include file
FF.MISC1.TEXT 32 an include file
FF.DATA4.TEXT 32 an include file
README.1ST.TEXT 8 Read this FIRST!
VOL27.DOC.TEXT 6 You're reading it

This volume was assembled by George Schreyer from material collected by the Library committee.

UCSD Pascal Volume 028

FreeForm (a 3-D spreadsheet)
Documentation and Run Modules
(Sources on UCSD Pascal Volume 027)

--> Version IV.x ONLY <--
4 word reals recommended

FF.A.TEXT 20 FreeForm Documentation.
FF.B.TEXT 16 ditto
FF.C.TEXT 18 ditto
FF.D.TEXT 18 ditto
FF.E.TEXT 18 ditto
FF.F.TEXT 8 ditto
FF.G.TEXT 14 ditto
FF.I.TEXT 18 ditto
FF.J.TEXT 26 ditto
FF.K.TEXT 20 ditto
FF.4WORD.CODE 125 FreeForm for 4 word reals; IV.x only.
FF.2WORD.CODE 124 FreeForm for 2 word reals; IV.x only.
README.1ST.TEXT 8 Read this first!
VOL28.DOC.TEXT 6 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library committee.

UCSD Pascal Volume 029

A script driven communications package and a weaver's helper
(a USUS REMUNIT is needed for CONVERS, see UCSD Pascal Volume 015)

DRAW4A.TEXT 32 A simple pattern weave analyzer.
DRAW4A.1.TEXT 34 an include file.
DRAW8A.TEXT 32 A more complex pattern weave analyzer.
DRAW8A.1.TEXT 36 an include file.
DRAWDN.DOC.TEXT 26 Documentation for the weaver's design package.
OSMISC_II0.TEXT 12 Misc routines for CONVERS for version II.0.
OSMISC_IV.TEXT 14 Same for IV.x.
TEXTIO_II0.TEXT 26 Text file routines for CONVERS for version II.0.
TEXTIO_IV.TEXT 14 Same for IV.x.
SCRNOP_II0.TEXT 14 An ersatz SCREENOPS for II.0.
CONV_TEST.TEXT 6 A test script.
CONVDOC.TEXT 70 Documentation for CONVERS.
INSTALL.TEXT 6 Installation notes for CONVERS
CONVERS.TEXT 112 CONVERS itself.
TERMINAL.TEXT 8 A dumb terminal emulator which can stand alone.
VOL29.DOC.TEXT 6 You're reading it.

This volume was assembled by George Schreyer from material collected by the Library committee.

UCSD Pascal (Modula-2) Volume 033

Modula-2 Stuff: Original Volition pShell & a BTree Package
The pShell was donated to the Library by Volition Systems.
The BTree package was submitted by Joseph Folse.

The following files make up the pShell package.

ARGS.D.TEXT 4 Definition Module for Command line arguments.
ARGS.I.TEXT 4 Implementation Module Command line arguments.
PSHELL.DOC.TEXT 26 General pShell documentation.
SHELL.NOTE.TEXT 12 Apple-specific pShell documentation.
SH.TEXT 32 The shell itself.
LS.TEXT 12 Directory listing module.
GREP.TEXT 6 Simple string finder.
SORT.TEXT 10 A sort module.
CAT.TEXT 6 Textfile concatenator (or lister).
CP.TEXT 6 File copier.
MV.TEXT 4 File renamer.
MORE.TEXT 4 Screen at a time lister.
DATE.TEXT 6 Show the current system date.
MEM.TEXT 4 Memavail.
ECHO.TEXT 4 Echo command line arguments.
MC.TEXT 4 Invoke the Modula-2 compiler.
F.TEXT 4 Invoke the filer.
ED.TEXT 4 Invoke the editor.
RM.TEXT 4 Remove files.

The following files make up the BTree package:

MBTREED.TEXT 10
MBTREE.TEXT 8
MBTBIO.TEXT 8
MBTDEBUG.TEXT 8
MBTHKEEP.TEXT 6
MBTINTERN.TEXT 24
MBTMAIN.TEXT 22
MBDOC1.TEXT 30
MBDOC2.TEXT 30
MBDOC3.TEXT 16
MBTRETST.TEXT 14
ABTDEFD.TEXT 8
ABTDEF.TEXT 4
ABTBIOD.TEXT 4
ABTBIO.TEXT 4
ABTDEBUGD.TEXT 4
ABTDEBUG.TEXT 4
ABTHKEEPD.TEXT 4
ABTHKEEP.TEXT 4
ABTINTERND.TEXT 4
ABTINTERN.TEXT 4
ABTMAIND.TEXT 6
ABTMAIN.TEXT 4
ABTREED.TEXT 10

ABTREE.TEXT 8
ABTRETST.TEXT 14
BTREE.DOC.TEXT 8 Main documentation file.
VOL33.DOC.TEXT 8 You're reading it.

This volume was assembled by Dennis Cohen from material collected by the Library committee.

UCSD Pascal Volume 034

PASMAT.TEXT 112 Source to a fairly flexible Pascal prettyprinter.
PASMAT.DOC.TEXT 24 The documentation for Pasmat.
UCSDXREF.TEXT 40 Identifier crossreferencer which flags assignments and declarations.
PREF.TEXT 26 Arthur Sale's Procedure Referencer in UCSD Pascal.
PREF.I.TEXT 44 An include file for PREF.
TCLISP.TEXT 70 A "tiny" lisp in UCSD.
XLISP.DOC.TEXT 50 The documentation for the C version from which it was developed.
ABSTRACT.TEXT 12 More documentation.
README.TEXT 8 Still more documentation.
TCLISP.CODE 13 The executable (IV.1).
HANOI.TEXT 4 An example program.
VOL34.DOC.TEXT 6 You're reading it.

This volume was assembled by Dennis Cohen from material collected by the Library committee.

UCSD Pascal Volume 035

PDOSTRANS.TEXT 36 Originally on Vol31: which was withdrawn. Transfer IBM PCDOS text files to p-System text files.
PDOSOPS.TEXT 52 Part of the above.
LIFE.TEXT 28 Jai Khalsa's game of LIFE...as discussed on MUSUS and in the USUS Bulletin #13. An example of fast two-dimensional list processing. Also originally on Vol31:
LIFE.DOC.TEXT 36
LNPR.TEXT 22 A group of Modula 2 programs submitted by P.D.Terry.
LNPRINC.TEXT 18 A lot of worthwhile material you can use.
LNPRDATA.TEXT 4
LNPR.DOC.TEXT 16
OPENFILE.TEXT 6
TERM.DOC.TEXT 8
TERM.TEXT 36
FIXDIR.TEXT 22
FIXINC.TEXT 28
FIX.DOC.TEXT 10
LISTER.TEXT 12
LISTER.DOC.TEXT 4
SETFX80.TEXT 16
SETFX.DOC.TEXT 8
EASYDEF.TEXT 12
EASYMOD.TEXT 26
TERMDEF.TEXT 12
TERMMOD.TEXT 28
README.TEXT 6 Listing of the programs in this group.
VOL35.DOC.TEXT 6 Your reading it.

This volume was assembled by David Rhoads from material collected by the Library committee.

UCSD Pascal Volume 036

TEMPL.SUB.TEXT 38 The main program of Carl Helmers enhanced "REPORT" program for use with Tom Swan's PDBS data base. Provides additional formating capability.
MAKE_TEMPL.TEXT 14 A quick way to start creating a template
P3_LINES.TEXT 4 These are demo files explained in the documentation
A3_LINES.TEXT 4
AFORM_LTR.TEXT 6
PFORM_LTR.TEXT 6
ALIST_HEAD.TEXT 4
ALIST.HEAD 4
ALIST.TEXT 4
PLIST.TEXT 4
AGENERIC.TEXT 4
PGENERIC.TEXT 4
STATUS.TEXT 4
PA_DBS_FI.TEXT 4
A_PDBS.FIL.PDBF 3
TEMPLATES.TEXT 52 The documentation file.
DOCGGENCONV.TEXT 6 More on documentation.
STARTREK.TEXT 70 Revision of original program on VOL9: Specific to IV.2.2 and STRIDE with WY50. Puts all action into windows instead of continuous scroll.
STREK.TEXT 10 Broken up for older p-System Editors. (Was not done
STREK1.TEXT 30 with "TEXTSPLIT"...see program below.)
STREK2.TEXT 36
STREK.DOC.TEXT 6 Explanation of the files and how to compile for the Sage.
RIPPLES.TEXT 6 Repeatedly draws a pattern on a 24 x 80 terminal.
COUNTWORDS.TEXT 4 Simple program for writers who get paid by the word.
BIGJUL.TEXT 18 Screen display of weekdays for any period of time starting 01/01/1769 and into the future.
BIGJULDOC.TEXT 6 Author's comments and explanations.
TEXTSPLIT.TEXT 18 Break big text files into small ones of any size you specify. This is the best of any we have seen so far.
HOME_LOAN.TEXT 20 Generates amortization table on printer or console.
VOL36.DOC.TEXT 6 You're reading it.

NOTE: This volume directory is not idenctical to that published in the Winter 1987-88 Newletter. S2.TEXT and its supporting files have been removed because they were identical to STARTREK.TEXT. Programs starting with RIPPLES and ending with HOME_LOAN have been added as replacements.

This volume was assembled by William Reed from material collected by the Library committee.

UCSD Pascal Volume UK3

An ADA Syntax Checker
Submitted by USUS(UK)

ADADOC.TEXT 20 Documentation of the ADA Syntax Checker.
ADA.TEXT 34 The main program of the ADA Syntax Checker.
COINT.TEXT 14 an include filex.
LAINIT.TEXT 8 ditto
LANEXT.TEXT 40 ditto
LUERROR.TEXT 14 ditto
LUINIT.TEXT 16 ditto
PARSER.TEXT 40 ditto
TYPES.TEXT 18 ditto
STGET.TEXT 12 ditto
TEXTDAT.TEXT 136 A data file.
FILECHECK.TEXT 4 A utility program.
ADA.CODE 33 A code file of the ADA Syntax Checker that won't run under IV.0.
GENDAT.TEXT 4 Generates the file TEXTDAT.TEXT.
GENDAT.CODE 2
ADATEST.TEXT 21 A non-UCSD text file of a sample ADA test case.
NEWADATEST.TEXT 24 A version of ADATEST.TEXT converted to UCSD formatd but without proper indentation.
CONTENTS.TEXT 6 The original UK contents file.
VOLUK3.DOC.TEXT 6 You're reading it.

This volume was assembled by USUS(UK) from material collected by their Library committee.

UCSD Pascal Volume UK4

An APL Interpreter and other stuff Submitted by USUS(UK)

APL.TEXT 32 The APL Interpreter.
APLPARSE1.TEXT 18 an include file.
APLPARSE2.TEXT 66 ditto
APLPROCS.TEXT 34 ditto
APLPARSE3.TEXT 6 ditto
APLHEAP.TEXT 12 ditto
APLINIT.TEXT 16 ditto
APLCHERS.TEXT 6 ditto
APLPARSE0.TEXT 16 ditto
SORT.DOC.TEXT 42 Documentation of the Sort/Merge Utility.
SORT.MERGE.TEXT 26 The main Sort/Merge program.
SORT.READ.TEXT 6 Some more notes on the Sort/Merge files.
SORT.DUMUN.TEXT 8 A example of the required user-supplied for Sort/Merge.
SORT.TXTUN.TEXT 18 A more general example of SORT.DUMUN.TEXT.
SPBSSTUFF.TEXT 40 A unit full of useful goodies.
SPBSSDOC1.TEXT 32 Documentation for SPBSSTUFF.
SPBSSDOC2.TEXT 32 Documentation for SPBSSTUFF.
SPBSSTUFF.CODE 16 A code file for IV.0 of SBBSSTUFF.
CONTENTS.TEXT 10 The UK's file of the contents of this disk, there's more detail in this one about this disk.
VOLUK4.DOC.TEXT 6 You're reading it.

This volume was assembled by USUS(UK) from material collected by their Library committee.

V. USUS Software Library Part II

The second part of the USUS Library contains implementations of computer languages other than Ada, Pascal, Modula-2, Modula-3, and Oberon. These are provided simply as a resource to USUS members to view characteristics of other languages, read about them, and try them out without needing to purchase commericial implementations. Also, along the same idea, we have a complete set of demonstration disks and evaluation disks from several companies that provide programming utilities. Lastly, Part II of the Software Library contains documentation files and specification files for a host of topics related to programming. Noted examples include documentation on graphic file formats and communication protocols.

Because the Demo and Evaluation volumes are not homogeneous in size, the pricing structure is different. In the listing, there will be an individual price for each of the volumes. Furthermore, USUS will only provide the volumes as received from the original company. Thus, we will only provide the disk format that is given to us. We will mail only exact duplicates of the originals.

When ordering from the Other Language Implementations Library, on the order form where it asks "Language" write in "OTHER." Likewise when ordering from the Demo and Evaluation Disk Library, write "DEMO & EVAL" for "Language" in the order form. And, for the Documentation and Specification Library, write "DOC & SPEC" for the "Language."

If you have any questions, leave a message on CODEPORT or write.

Demo and Evaluation Disk Library

MULTIEDIT

VEDIT

SHIELD

Documentation and Specification Library

Doc & Spec Volume 001

4KXMODEM.....Specs for 4K extension to XMODEM.
BPROTO.....VIDTEX and CompuServe B protocol specifications.
CMODEM.....C-MODEM transfer protocol spec.
MEGALINK.....MegaLink file transfer protocol spec.
MIT-SLFP.....MIT's Serial Line Framing Protocol spec.
MODEM7.....TOPS20 MODEM7 implementation spec.
PROTOCOL.....A primer on transfer protocols.
WXMODEM.....Specs on XMODEM, XMODEM CRC, and WXMODEM.

X-PC.....X.PC protocol specification.
XPKTPROT.....X-Packet protocol specs for Packet Radio transfers.
YMODEM8.....XMODEM and YMODEM specifications.
ZMODEM8.....ZMODEM specification.
PROTCISA.....CompuServe A protocol spec.
FAST.....Fast transfer protocol spec for error-free lines.
AEPRO.....ASCII Express extensions reference to Christensen protocol.
SLIP.....SLIP protocol specification.
UUCP.....UUCP protocol specification.
ASYNCH-FA.....Asynchronous Fascimile Control Standard specification.
XMODEM-C.....Calculating XMODEM CRC documentation.
GIF.....Graphic Interchange File Format Documentation.
TIFF-40.....Tag Image File Format Rev. 4 documentation.
FIPS.....Federal Information Processing Standards Publication Catalog.

Doc & Spec Volume 002

BASIS17.....Documentation for ANS FORTH. In RTF (Rich Text Format). Due to the file sizes, this volume is only available in 640k, 720k, 800k, 1.2 meg and 1.44 meg disk sizes.

Other Language Implementations Library

Other Language Implementations Volume 001

SMDOCS.....Documentation to SmallTalk implementation below.
SMEXE.....MS-DOS SmallTalk executable.
SMPROGS.....Sample SmallTalk programs for above.
SMTALK.....C source code to the SmallTalk implementation above.
ZEN.....Zen Forth programming language. MS-DOS.

Other Language Implementations Volume 002

ASIC20.....BASIC compiler, editor, docs. No source to compiler.
XLISPDOC.....Documentation for X-LISP implementation below.
XLSP21EX.....MS-DOS X-LISP executable.
XLSP21TC.....C source code to X-LISP implementation above.

Other Language Implementations Volume 003

SC88.....Small-C compiler, outputs 8088 assembly source. Comes with C source for the compiler and libraries. Compiler is for MS-DOS.
COBOL650.....ANSI 6.50 COBOL compiler and docs. MS-DOS. No source to compiler.
PROLOG19.....Implementation of a Prolog interpreter. MS-DOS. Documentations and a great deal of sample source. No source for the interpreter.

Other Language Implementations Volume 004

VSNOBOL4.....Implementation of Snobol 4 for MS-DOS. Documentation and sample source. No source for the Snobol 4 implementation itself.
FORTHV2.....Implementation of FORTH-83 for MS-DOS. Documentation and sample source. FORTH source to portions of system.
ABC.....The ABC language, easy to learn interpretive language. MS-DOS.

Notice to International Members

If you are an overseas member and would rather have Computer Language Magazine be sent to you via airmail (as opposed to surface mail, which is the default) there is an additional $25 per year fee. To receive CLM airmail, send a check or money order for $40 (made out to USUS, Inc.) to our La Jolla address and we will notify the CLM staff to change your subscription to airmail delivery.

USUS SOFTWARE LIBRARY ORDER FORM

Ship To:

Name:_____

Address:_____

_____

City:_____ State/Province:_____

Country:_____ Postal Code:_____

Phone: _____ (in case there are questions about your order)

Member Number:_____ (required for UCSD Pascal Software Library Volumes)

Volumes You Want:

Language	Volume Numbers
_____	_____
_____	_____
_____	_____

Destination, and payment amount (check one):

Per Volume: ☐$6.00 USA ☐$6.00 Canada/Mexico ☐$8.00(airmail) International

Times _____ volumes ordered, Total Amount Enclosed: _____

Please send a check or money order for the appropriate amount. The check or money order must be drawn from a US Bank or Office, be in US dollars, and made payable to USUS Inc. Send the order directly to the USUS librarian:

Stephen Pickett
PO Box 1279
Point Roberts, WA 98281-1279

Indicate the desired format and operating system:

Disk Size (3.5" or 5.25") and Capacity:_____ Disk Operating System:_____ Computer System:_____

Formats Supported Include:

Apple][280 blocks
MSDOS hosted p-system (360K, 1.2M, 720K, 1.44M)
P-system (specified by number of blocks in the disk format): 640, 720, 800 (NCI/RECS), 1280 (Stride), 1440 (3.5"), 1440 (5.25"), 1600 (NCI 3.5"), 1600 (NCI 5.25"), 2400, 2880
Some 8" formats (ask)
MSDOS native (360K, 1.2M, 720K, 1.44M)

IMPORTANT: Please read and sign the following agreement.

Software Order Agreement

I acknowledge that neither USUS nor any of its representatives nor the author makes any warranty with respect to the contents of any Software Library disks, particularly, no warranty of fitness for any purpose.

For any volumes from the USUS UCSD Pascal Software Library (volumes 001 .. UK4), I also acknowledge that I will not permit the Programs I receive pursuant to the foregoing order to be published for profit in whole or in part or to be transferred to any person who is not a member of USUS, without the express written consent of the author identified in the files of the Library. Further, lacking such author's consent, I will ensure that the following items have been submitted to the designated individual (for now the USUS Treasurer, USUS Librarian, or the USUS administrator) before transferring any Program(s) to another USUS member: a) the current Software Order Agreement signed by the software recipient; and b) $1 per volume or fraction of a volume received.

Signed _____ Date _____

From the editor

by Tom Cattrall

Finally, the newsletter is back. The long time since the last issue has been entirely my fault. Due to a heavy work load, some business trips, and some bouts of bad health, I just hadn't made the time to work on the newsletter. I'm hoping to get back onto a reasonable schedule now. I have enough material to get another issue out soon.

Expanded Library

Keith Frederick has spent a lot of time collecting material from various computer systems around the world. The result is a huge increase in our library volumes. You can see the result in the new library listing starting on page 24 of this issue. In his comments he doesn't say anything about his part in this, so I will: Keith did all of this work on his own initiative. Nobody else helped him. We all should be grateful for his efforts in bringing so much new material to our library.

USUS Board Elections

It is time for another election of directors to the USUS board. If you are interested, or know someone that is, please get in contact with me. The duties consist of attending meetings on the Compuserve forum and providing guidance at other times.

News from CodePort (MUSUS)

The forum on Compuserve is keeping a slow but steady pace. We have added a section for APL and an assistant sysop (Woody Butler) that is a long time user of APL. Messages on Modula-2, Oberon, and Pascal, along with APL make up the bulk of discussions. It seems to me that more people frequent the forum for the library files than for the discussions. The number and quality of files submitted to the online libraries continues to grow.

Submission Guidelines

Submit articles to me at the address shown on the back cover. Electronic mail is probably best, disks next best, and paper copy is last. If your article has figures or diagrams, I can use encapsulated Postscript files in any of the disk formats listed below. If you can't produce encapsulated Postscript, then paper copy is probably the only practical method for submitting graphics.

You can send E-Mail to my Compuserve ID: 72767,622, or indirectly from internet: 72767.622@compuserve.com. For disks, I can read Sage/Stride/Pinnacle format disks. Also, any MS-DOS 5.25 or 3.5 disks, and 3.5" Amiga disks. If anyone wants to send Mac format disks I could probably get someone to translate them into something I can use. Whatever you send, please mark on the disk what format it is. That will save me a lot of guesswork.
Text should be plain ascii rather than a word processor file. It can have carriage returns at the end of all lines or only at the ends of paragraphs. What you send doesn't have to look pretty. I will take care of that. My spelling checker will take care of spelling errors too. If you want special formatting use the following conventions:

1. _Underline_, put an underline character at each end of the section to underline.

2. *Bold*, put a star at each end of the section to **bold**.

3. ^Italics^, put a caret at each end of the section to be set in *italics*.

4. ??Special requests??, such as ??box next paragraph?? should be surrounded with "?? ??".

NewsLetter Editor : Tom Cattrall
Amity Software Inc.
7600 Seawood Road SE
Amity, Oregon 97101
503/835-1613
Compuserve : 72767,622
Internet : 72767.622@compuserve.com
tomc@techbook.com

NewsLetter Publisher : William Smith

USUS Board of Directors

Felix Bearden	74076,1715
Tom Cattrall	72767,622
Keith Frederick	73760,3521
Gary Gibb	72230,1601
Stephen Pickett	71016,1203

USUS Officers

President:		
Treasurer:	Bob Clark	72747,3126
Secretary:	Keith Frederick	73760,3521

USUS Staff

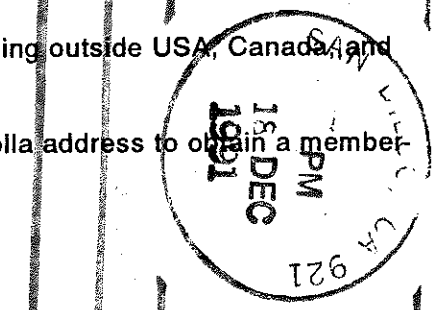
Administrator:	Keith Frederick	73760,3521
Legal Advisor:	David R. Babb	72257,1162
Librarian:	Stephen Pickett	71016,1203
MUSUS Sysop:	Harry Baya	76702,513

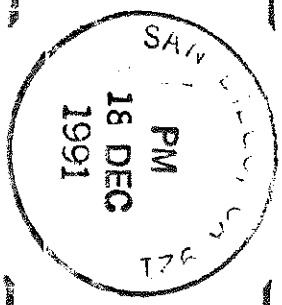
USUS Membership Information

Student Membership	30 / year
Regular Membership	45 / year
Professional Membership	$ 100 / year

$15 special handling outside USA, Canada and Mexico.

Write to the La Jolla address to obtain a membership form.





USUS
P.O. BOX 1148
LA JOLLA, CA 92038

ADDRESS CORRECTION REQUESTED
FIRST CLASS MAIL

[redacted]