The MODUS Quarterly

Issue # 6

November 1986

Modula-2 News for MODUS, the Modula-2 Users Association.

CONTENT

Directors of MODUS, the Modula-2 Users Association:

Randy Bush
Pacific Systems Group
601 South 12th Court
Coos Bay, OR 97420
(503) 267-6970

Tom DeMarco
Atlantic Systems Guild
353 West 12th Street
New York, NY 10014
(212) 620-4282

Swend Erik Knudsen
Rechenzentrum
ETH Zürich
Clausiusstrasse 55
CH-8092 Zürich
01 / 256 34 87

Heinz Waldburger
ERDIS SA
CH 1800-Vevey 2
(021) 52 61 71

Jean-Louis Dewez
Laboratoire de
Micro Informatique, CNAM
2, Rue Conté
F-75003 Paris
01 / 42 71 24 14

Administration and membership:

USA: George Symons
MODUS
PO Box 51778
Palo Alto, CA 94303
(415) 322-0547

Europe:

Aline Sigrist
MODUS Secretary
ERDIS SA
P.O.Box 35
CH-1800 Vevey 2

Editor, MODUS Quarterly:

Richard Karpinski
6521 Raymond Street
Oakland, CA 94609
Weekdays (415) 666-4529 (12-7 pm)
Anytime (415) 658-3797 (ans. mach.)
TeleMail M2News or RKarpinski
BITNET dick@ucsfcca
Compuserve 70215,1277
USENET ...!ucbvax!ucsfcgl!cca.ucsf!dick

>> Problems? Missing an issue? <<

Contact your membership coordinator (see above).

Publisher:

George Symons (see above)

Publication schedule:

Deadline	Issue
15 Jan	Feb
15 Apr	May
15 Jul	Aug
15 Oct	Nov

Submissions for publication:

Send CAMERA READY copy to the editor.
Dot matrix copy is often unacceptable.
Machine readable copy is preferred: 60 lines, 70/84 characters.

TeleMail address: M2News

Please indicate that publication of your submission is permitted.
Correspondence not for publication should be PROMINENTLY so marked.

Modula-2 Users' Association
MEMBERSHIP APPLICATION

Name : _____

Affiliation : _____

Address : _____

Address : _____

State : _____ **Postal Code:** _____ **Country:** _____

Phone : (___)____-_____ **Electronic Addr :** _____

Option: ___ Do NOT print my phone number in any rosters
or: ___ Print ONLY my name and country in any rosters
or: ___ Do NOT release my name on mailing lists

Application as: **New Member** ___ or **Renewal** ___

Implementation(s) used : _____

** Membership fee per year (20 USD or 45 SFr) **
Members of the US group who are outside of North America, add $10.00.

In North and South America, please send check or money order (drawn in US dollars) payable to Modula-2 Users' Association at:

Modula-2 Users' Association
P.O. Box 51778
Palo Alto, California 94303
United States of America

Otherwise, please send check or bank transfer (in Swiss Francs) payable to Modula-2 Users' Association at:

Aline Sigrist
MODUS Secretary
ERDIS SA
P.O.Box 35
CH-1800 Vevey 2

The Modula-2 Users' Association is a forum for all parties interested in the Modula-2 Language to meet each other and exchange ideas. The primary means of communication is through the Newsletter which is published four times a year. Membership is for an academic year, and you will receive all newsletters for the full year in which you join. Mid-year applications receive that year's back issues. Modula-2 is a new and developing language; this organization provides implementors and serious users a means to discuss and keep informed about the standardization effort, while discussing implementation ideas and peculiarities. For the recreational user, there is information on the status of the language, along with examples and ideas for programming in Modula-2. For everyone, there is information on current implementations and the other resources available for obtaining information on the language.

Prior issues partial content: (these are still available)

Modula-2 News # 0 October 1984

Revisions ... to Modula-2, Wirth
Spec. of Standard Modules, Hoppe
Modula-2 bibliography, Brown
Modus Membership list
Modula-2 Implementation Questionaire

Modula-2 News # 1 January 1985

Letter to Editor, Layman
Letter to Editor, Bush
Gleaves' Modula-2 text, DeMarco
MODUS Paris meeting, Blunsdon
Report of M2 Working Group, Souter
Library Rationale by Randy Bush
Library Definition Modules
Library Documentation by Jon Bondy
Validation of Modula-2 Impl, Siegel

MODUS Quarterly # 2 April 1985

Letter on Library, Anderson
Letter to Editor, Emerson
Comments on Modula-2, Emerson
Opaque Types, French & Mitchell
Dynamic Instantiation, Sumner
Linking Modula-2, Symons
Library Comments, Peterson
Modula Compilers, Smith
Coding War Games, DeMarco
M2, Alt. to C, Djavaheri/Osborne

MODUS Quarterly # 3 July 1985

Letter re opaque types, Endicott
Letter on language issues, Hoffmann
Modula-2 in "Real Time", Barrow
RajaInOut: safer I/O, Thiagarajan
Contentious Problems, Cornelius
Expressions in Modula-2, Wichmann
Scope Problems: Modules, Cornelius
Corrections to compiler list

MODUS Quarterly # 4 November 1985

MODUS Meeting Report by Bob Peterson
A Writer's View of Conf, Sam'l Bassett
Concerns of a Programmer, Dennis Cohen
Mods to Standard Lib, Nagler & Siegel
Std Lib and Ext'n to Modula-2, Odersky
Std Lib for Unix by Morris Djavaheri
Impl of Std Lib for PC's, Verhulst
M-2 Compilation and Beyond, Foster
Modula-2 Processes, Roger Henry

MODUS Quarterly # 5 February 1986

Export Module Identifier, Cornelius
Multi-dimensional open arrays, Wirth
DIV, MOD, /, and REM, Niklaus Wirth
Multi-dimensional open arrays, Steiger
NULL-terminated strings, Poulsen
ISO Ballot Results re BSI Modula-2
Draft BSI I/O Library, Eisenbach
Portable Language Rationale, Hopper +
ETH-Z Modula-2 for Macintosh, Jewell
NewStudio: for Macintosh, Davidson +

MODUS Administrators supply single copies at $5 US or 12 Swiss Francs.

Hints for contributors:

Send CAMERA READY copy to the editor (dot matrix copy is usually unacceptable). Machine readable copy is preferred. Present facilities permit printing from electronic mail and floppy disks (Sage, IBM PC, Macintosh) using troff, Script and PostScript formatting systems. Working papers and notes about work in progress are encouraged. MODUS Quarterly is not perfect, it is current.

Please indicate that publication of your submission is permitted. Correspondence not for publication should be PROMINENTLY so marked.

Richard Karpinski, Editor
6521 Raymond Street
Oakland, CA 94609
(415) 476-4529 (12-7 pm)
(415) 658-3797 (ans. mach.)

TeleMail M2News or RKarpinski
BITNET dick@ucsfcca
Compuserve 70215,1277
InterNet dick@cca.ucsf.edu
UUCP ...ucbvax!ucsfcgl!cca.ucsf!dick

Editorial

Waste, corruption, and fraud. Unconscienceable delay. Clearly the editor should be replaced. I support this grassroots movement to install a more responsible and time-conscious editor (or at least an assistant editor to hold the editors feet to the coals as necessary to insure the timely production of MODUS Quarterly).

Commercials? Advertisements? Do you want to see ads or other commercial material in your quarterly? Mr. Hedelman has been trying for about a year to get this announcement into MQ. Both he and I have gotten mixed messages about whether it can be permitted. Since my most recent advice from the publisher is to go ahead, we now include:

LILITH, the amazing Modula Machine, FOR SALE, $2600 or best offer
contact Harold Hedelman at (415) 868-2636

Beginning with this issue, European MODUS members will receive their copies via reprinting in Europe. Since the copy can be airmailed both printing tasks can proceed largely in parallel. Perhaps the competitive possibilities can be exploited to improve the timeliness of the process on both sides of the water. We shall see.

This issue has several papers from participants in the British Standards Institution working group on Modula-2. They seem to be particularly interested in your comments on their choices and directions in creating standard Modula-2. Do not disappoint them. We would like to get your views as well. We presume that your ideas will be of interest to many MODUS Quarterly readers.

The most popular article from the last issue was the NewStudio discussion of development in Modula-2 for the Macintosh. Accordingly, Mr. Davidson will receive a year's subscription to MODUS Quarterly. There was actually a contest this time about the best suggestion. One reader suggested that we spell "contributors" correctly, but I found Mr. Stan Osborne's suggestion that he help me get issue #6 out even more helpful. Mr. Osborne gets the prize subscription.

Do you care? Will you make a suggestion? Where is your article? How about a letter? Surely there was something in the last year of MODUS Quarterly that you have an opinion about. Why not say it?

There seems to be enough material to put together another issue right away. If Stan keeps poking me effectively enough, the last issue of the subscription (and calendar) year, MQ #7, will be printed and mailed in December of this year. Perhaps, given sufficient contributions for publication, we can resume the normal schedule for the third subscription year.

Stan also has an idea for issue #8. Perhaps he will elaborate it further in the next issue, but he basic idea is to request that you send in comparative discussions about the various Modula-2 systems which are available for your computer. He is particularly interested in showing the genealogy of the compilers. He claims that the evolution and development of these compilers is a worthwhile topic for discussion since things are happening more and more rapidly all the time. How about rumors of impending products? Are you interested?

Dick Karpinski
yr faithless, irresponsible, laggard (but organized) editor

20/215 Bridge Road,
Glebe,
NSW 2037,
Australia.

22nd November, 1985.

Editor,
Modula-2 News.

Dear Sir,

1. There are three points I would like to raise. Firstly, I agree with your comments with regards to opaque made in the July 1985 issue. As a user of the UNIT construct in Microsoft PASCAL, where opaque types are not available, I am aware of the annoyance caused by the need to recompile user programs when a type definition changes. The current representation of opaque types is elegant and simple and worth the minor inconvenience of having to dereference the pointer in an implementation module.

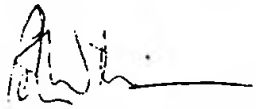
2. The second point I would like to raise concerns the proposed standard library IO modules. In the current proposal the only module with knowledge of the structure of the File type is Files. This means that the only way for variables of this type to be manipulated without breaching the integrity of the language is via the procedures provided in that module. These procedures are insufficient for the modules Binary, Text and FilePositions to do their jobs. A solution to the problem would be to declare the File type in Files along the following lines

```
TYPE
  File = POINTER TO RECORD
    (* System dependent info goes here along with a warning to treat the type
     * as if it were opaque
     *)
  END;
```

This unfortunately means that changes to the definition of File would require all user modules to be recompiled. Therfore a better solution (although it leads to a rather large module) is to combine the modules Files, Binary, Text and FilePositions into a single module and keep File as an opaque. A basic principle of opaque types is that the module that declares them must provide procedures for performing all sensible operations on them.

3. Thirdly, I would like to add my name to the list of users who would like the restrictions on the SET type relaxed, at least to the extent that SET OF CHAR is legal.

Yours sincerely,



(Peter Williams)

8570 N. Mulberry Drive
Tucson, Arizona 85704
May 27, 1986

Barry Cornelius
Department of Computer Science
University of Durham
Durham DH1 3LE
United Kingdom

Dear Dr. Cornelius:

With reference to your recent article in the Modus Quarterly concerning the visibility of exported identifiers, I would like to express my opinions and vote.

I have an extremely strong favoritism to your so-called "proposal 1". The reason for this relates to the fact that I find visualizing (local) modules much more comfortable using the analogy of record structures instead of enumerated types. What I mean by this is that a module which makes identifiers available to its environment ought to have those identifiers "attached" to the module as visible components of it. Another proposal which permits, in effect, free-floating identifiers around the module seems inappropriate for Modula-2, and not consistent with other concepts in the language.

Thus, when an (entire) module is imported or exported, I strongly feel that its exported identifiers should be visible in qualified form, whether or not those identifiers are exported using the "EXPORT QUALIFIED" format. This works quite consistently with compilation units as well as local modules.

As an additional note, adopting this proposal takes us upon an interesting side road. EXPORT QUALIFIED makes identifiers visible from a module, but one can consider this as not actually exporting the identifiers into the surrounding environment! This is because EXPORT QUALIFIED makes that identifier only usable when the prefixing module identifier is present, for both cases of simply using the identifier, as well as further exporting the module as a whole. Again, this is very similar to the way one works with a record structure -- a record's field identifiers are treated the same way as identifiers made visible with EXPORT QUALIFIED. EXPORT without the QUALIFIED reserved word also makes the identifier visible, but this is now the only case whereby the identifier is actually extruded into the environment.

Would you please pass onto the BSI committee one further item unrelated to the topics above. After several years of managing application development efforts in Modula-2, I have found that at no time (repeat, no!) has any definition module we have written not exported one of its identifiers. Thus, I would urge you to have the standard conform to Wirth's current specification of disallowing an export list in definition modules. For implementations of one-pass compilers, the inclusion of a FORWARD directive eliminates the problem of having to put a procedure header in a definition module just so it can be used in a circular reference.

You asked in your article which compiler we currently use. We have switched to a relatively new product called Modula-2PC from PCollier Systems Inc. here in Tucson. This compiler supports your "proposal 1" as well as the FORWARD directive. We have been quite happy with the implementation of both items.

Sincerely yours,

E. R. Videki

TO → cc: Richard Karpinski, Modus Quarterly

Dear Richard,

Why the Plain Vanilla Linkers?

Almost all Modula-2 implementations which provide static linkage of object code do so in a particularly simple-minded manner. The code for all procedures, from modules which have been imported directly or indirectly, is linked into a monolithic load image. This has the unfortunate consequence of discouraging the creation of comprehensive library modules, since all the unwanted procedures for any application end up expanding the size of the load file, and hence slowing down its loading. Better would be to link only those procedures which are found to be callable, as a result of a sort of elementary flow analysis on the program.

This is not hard to do. Sufficient rules are:

- Every module initialization part is a callable "procedure";
- Every procedure called by a callable procedure is callable;
- Procedure constants assigned to procedure variables are callable.

During the linkage process, a global "calls" digraph is formed, and a straightforward transitive closure algorithm deduces which procedures are directly or indirectly callable. Only these are linked.

The M23 distribution of M2RT11, which we are now shipping, contains a linker which works on exactly these lines. Experience shows that only about 50% of total procedures are actually linked for typical user application programs.

The theory is pretty obvious, so there is little excuse to not do it.

Yours faithfully,

K John Gough,
School of Computing Studies,
Queensland Institute of Technology,
GPO Box 2434, Brisbane,
AUSTRALIA.

Michael D. Coren
1628 Arran Way
Dresher, PA 19025
June 14, 1986

Richard Karpinski, Editor
MODUS Quarterly
6521 Raymond Street
Oakland, CA 94609

Dear Mr. Karpinski,

I am writing this letter to submit my vote for the best article in the MODUS quarterly, issue #5, February, 1986. As one who has used the Modula Corporation's MacModula-2 for writing Macintosh-style applications, I can vote for none other than Davidson, Herrmann, and Hoffer's *NewStudio*™ article.

I, too, became involved in Modula-2 through the MacModula-2 system. About a year ago, I was looking to purchase a Macintosh development system, but I didn't want to learn C or FORTH. I knew Pascal, but the only Pascal compiler available at that time was a UCSD system that cost $300 (which is now being distributed by another company for $79). MacModula-2 seemed to fit the bill nicely: It was relatively inexpensive, provided access to the high-level toolbox managers, and Modula-2 is close enough to Pascal that learning the language was not too time-consuming.

Yes, the compiler is slow, but that's the only slow thing about the system. The compiled code (interpreted M-code) is fast; a program I wrote to do some astronomical calculations in MacModula-2 runs about five or six times faster than the same program compiled using a recently

introduced Pascal compiler, which compiles to native code! Furthermore, the documentation for this system is excellent, containing DEFINITION MODULE listings for all of the library MODULEs, and sprinkled with numerous examples.

While this system is far from perfect--the compiler is slow, compile time errors are sometimes flagged and marked as much as 8 lines away from the actual error, and, in many cases, the implementors changed some toolbox interfaces for the sake of using Modula-2 data types (i.e. BITSETs), even though they are clearer and easier to manipulate from their Pascal definitions--its advantages strongly outweigh its shortcomings, and I can recommend this system with a clear conscience to anyone wishing to get involved with writing software for the Macintosh.

Sincerely,

Michael D. Cote

Significant Changes to the Language Modula-2

M2WG-N94: Issue 2: 30th April 1986

Barry Cornelius

Department of Computer Science
University of Durham
Durham DH1 3LE United Kingdom

This paper gives a list of some of the changes being proposed by the Modula-2 Working Group of the British Standards Institution. The list mainly consists of changes that are likely to affect existing Modula-2 programs. The changes are provisional: formally, no decision has any effect until ISO ballots on a draft proposal. However, please mail me if you think that any of these changes are undesirable.

In this paper, the notation "PIM" is used to refer to the book "Programming in Modula-2" by Niklaus Wirth. The Pascal Standard (BS6192/ISO7185) is referred to by the notation "PS". The definitions of the terms 'ordinal type' and 'exception' are given in the paper "Type Conversions in Modula-2" [2].

The section numbers used in this paper correspond to those used in the "Report on The Programming Language Modula-2" (as given in PIM).

3. Vocabulary and representation

WG104: It was agreed that underscores be permitted (as a 'break' character) in an identifier subject to:
- underscores are not permitted as the first or last character of an identifier,
- underscores are significant.

WG007: MaxInt, MinInt, etc., are to be dropped as MAX and MIN make them redundant.

WG086: It was agreed to remove:

```
octalDigit {octalDigit} "C"
```

from the language, since CHR(~~~) can now be used in any context in which ~~~C could be used - see WG099 in §5.

WG106: Two proposals on "strings" have been discussed in detail. Briefly, these proposals are:
(i) assignment of a string constant of length f to a variable of type ARRAY [0..t-1] OF CHAR is permitted if f<=t. If f<t, each of the last (t-f) elements of the array variable is assigned the value which is defined by the constant StringTerminator in the module SYSTEM.
(ii) a new string data type is introduced.
It is possible that either or both of these proposals will be accepted. The views of the Modula-2 community are to be sought - see the paper "All About Strings" [3].

WG012: It was agreed to add NIL as a reserved word.

5. Constant declarations

WG099: It was agreed to take the definition of constant expression given in a paper by Don Ward (M2WG-N52). Informally, this definition is:

Each operand in a ConstExpression must be a constant. A constant is:

+ a literal (either a number, a single character or a string literal),
+ an identifier denoting a constant value (either NIL, or from a CONST declaration, or from an enumeration type),
+ a set denotation whose elements, if present, are ConstExpressions,
+ a call of ABS, CAP or ODD with a parameter which is a ConstExpression,
+ a call of SIZE, MIN or MAX,
+ a type conversion of the form:
  typename(ConstExpression)
  [see WG102 under VAL in $10.2].

WG088: Numeric literals and constant identifiers are of the types ZZ and RR as defined in the paper "Expressions in Modula-2" [1]. Consequently, a literal which is potentially of type LONGCARD does not have a different denotation from one which is potentially of type CARDINAL. [Such literals are actually of type ZZ.]

WG126: As a result of WG099, it is possible to specify explicitly the type of a constant expression as illustrated by the following example:

```
TYPE  OneInTen = [ 1 .. 10 ] ;
CONST IntegerOne = INTEGER(1);
      CardinalOne = CARDINAL(1);
      OneInTenOne = OneInTen(1);
```

WG015: The element of a set literal must be a ConstExpression if the set literal is a part of a ConstExpression. The element of a set literal can be an Expression if the set literal is a part of an Expression.

6.6. Set types

WG131: It was agreed that the Standard be worded so that an implementation must provide SET OF CHAR.

WG034: It was agreed that if a program uses BITSET then BITSET must be imported from SYSTEM. [See also WG074 in $8.2.]

WG035: It was agreed that the constants N and W are to be exported from SYSTEM. However, more meaningful names are to be used.

WG132: It was agreed that a BITSET value maps onto the bits in one word. It was also agreed that the underlying representation of a value which is of a set type other than BITSET is not to be defined.

6.8. Procedure types

WG089: It was agreed that:
- \+ NIL is compatible with an object which is of a procedure type;
- \+ tests of (in)equality can be used between objects which are of a procedure type;
- \+ there should be provision for procedure constants.

8. Expressions

WG041: It was agreed that operands of an expression will be evaluated from left to right.

8.1. Operands

WG073: Although a function procedure may return a value which is of a PROCEDURE, POINTER, RECORD or ARRAY type (see WG061 in $10), it was agreed that these results cannot be "applied", "dereferenced", "selected" or "indexed".

8.2. Operators

WG074: It was agreed that the type preceding a set expression is no longer optional; so there is no longer a default to the type BITSET. [See also WG034 in $6.6.]

8.2.1. Arithmetic operators

WG080: It was agreed that out-of-range errors that occur during expression evaluation will yield an 'exception'. For example: if there is a cardinal expression and its result is positive, but an intermediate result is negative, then an 'exception' will occur.

9.1. Assignments

WG050: It was agreed that the right hand side of an assignment is to be evaluated before the left hand side.

9.2. Procedure calls

WG051: It was agreed that the parameters of a procedure call are to be "evaluated" from left to right.

9.5. Case statements

WG056: There should be an explicit statement that it is an 'exception' if the selecting expression of a CASE statement is not found as a label, and there is no ELSE clause. The definition could be modelled on PS 6.8.3.5, with minor modifications included for Modula-2's ELSE clause.

9.8. For statements

WG003: It was agreed that PS 6.8.3.9 would be used as a starting point for the specification of FOR statements.

10. Procedure declarations

WG061: It was agreed that a function procedure may return a value of any type. Wirth has said that the restriction on function result types in PIM was an implementation restriction of his compilers: the language itself has no restriction.

WG113: It was agreed that FORWARD is not to be made part of the language.

10.1. Formal parameters

WG114: It was agreed that the types of a formal VAR-parameter and that of its corresponding actual parameter must be identical (i.e. not merely compatible). This rule may be relaxed when the formal parameter is of type WORD or ADDRESS.

10.2. Standard procedures

ABS, CAP, and ODD

WG075: The agreed definitions for the above functions appear in the paper "Expressions in Modula-2" [1].

ORD, CHR, FLOAT and TRUNC

WG115: The function procedures ORD, CHR, FLOAT and TRUNC are to be removed from the language. Calls of these functions can be replaced as follows:

```
ORD(value) can now be done by CARDINAL(value)
CHR(value) can now be done by CHAR(value)
FLOAT(value) can now be done by REAL(value)
TRUNC(value) can now be done by CARDINAL(value - 0.5)
```

For brief details of the use of a typename to coerce a value into some other type, see WG102 under VAL in §10.2. For other details, see the paper "Type Conversions in Modula-2" [2].

HIGH

WG116: It was agreed that the parameter to HIGH must be a variable that is specified as an open array parameter. Thus, HIGH always delivers a result of type CARDINAL. MAX applied to the index type of an array can be used for an array which is not an open array parameter.

WG117: It was agreed that when a multi-dimensional open array parameter is passed to HIGH, HIGH delivers the high index bound of the first dimension of the array.

MAX and MIN

WG118: It was agreed that the standard functions MIN and MAX take any 'ordinal type' as a parameter. They return the type's minimum and maximum values. Consequently, if T is a subrange type, then MIN(T) and MAX(T) yield the lower and upper bounds of the subrange. Note that MIN and MAX cannot be applied to the type REAL. However, there will be some other mechanism to deliver the various characteristics of the type REAL.

SIZE and TSIZE

WG119: I believe that no firm decision was reached as to whether the parameter passed to SIZE has to be a type or a variable or either. Similarly, for TSIZE. However, it was agreed that both SIZE and TSIZE should return 'storage units' where the size of a storage unit depends on the implementation. It was also agreed that BitsInAStorageUnit is a constant identifier that is required to be exported from the module SYSTEM.

VAL

WG120: It was agreed that SYSTEM would contain a function procedure called CAST which would do unchecked type transfers.
So, given:

```
VAR  v1:t1;  v2:t2;
```

the call of CAST in:

```
v2:= CAST(t2, v1);
```

would return the value v1 interpreted as if it were of type t2. For full details, see the paper "Type Conversions in Modula-2" [2].

WG102: It was agreed that:

```
typename(expression)
```

would be used for type conversions. [Such conversions are "safe" - they are sometimes called "coercions".]
The restrictions on "typename" and the type of "expression" are defined in the paper "Type Conversions in Modula-2" [2].

WG121: Because of the construct provided in WG102, it was agreed to remove VAL from the language.

DEC and INC

WG122: It was agreed that:
+ the form of DEC with two parameters is removed from the language;
+ the parameter to DEC and INC must be a value which is of some 'ordinal type';
+ both DEC applied to the first value of a type and INC applied to the last value of a type cause an 'exception'.

NEW and DISPOSE

WG085: It was agreed to adopt the position adopted by the 3rd edition of PIM for NEW and DISPOSE, i.e., they are no longer part of the standard Modula-2 environment. This is because the fact that they magically led to calls of ALLOCATE and DEALLOCATE is considered undesirable.

11. Modules

WG069: It was agreed to follow Wirth's latest ideas about the scop. of standard identifiers which are as follows:

> The standard identifiers can be considered as being defined in a universe. If an identifier is not found within a program according to the scope rules, then that universe is searched as a last resort. Note that this allows any standard identifier to be redeclared as a fresh identifier anywhere.

This is in some ways similar to Pascal. It is assumed that in the above definition, the word "program" should read "module".

14. Compilation units

WG109: It was agreed that it is invalid to use the procedure heading:

```
PROCEDURE p(i:INTEGER);
```

in a definition module if the procedure is declared as:

```
PROCEDURE p(j:INTEGER);
```

in the implementation module: the same name has to be used for the two occurrences of the formal parameter. This attempts to ensure that the two formal parameter lists correctly correspond, especially when there are two consecutive parameters of the same type.

WG084: Wirth's proposal ("Revision 2.1") to discard the export list of a definition module is not to be adopted: M2WG has agreed to retain the original syntax and semantics, i.e., objects which are to be exported from a definition module have to be listed in an export list.

WG110: It was agreed that:

+ an opaque type must be declared as a pointer type in the implementation module;
+ an object which is of an opaque type may not be dereferenced in a client module.

It was also agreed that:

+ assignment between objects of an opaque type is legal
+ comparison for (in)equality is legal for objects of an opaque type

within a client module (as well as in the implementation module). Assignment is needed to be legal as it implicitly takes place when a variable of an opaque type is passed as a value parameter to a procedure within a client module.

REFERENCES

1. "Expressions in Modula-2", Brian Wichmann, "MODUS Quarterly" Issue 3, pp. 35-42.

2. "Type Conversions in Modula-2", Brian Wichmann, "MODUS Quarterly" Issue 6.

3. "All About Strings", Barry Cornelius, "MODUS Quarterly" Issue 6.

ELECTRONIC MAIL ADDRESSES

Any comments on these proposals can be sent to me at any of the following electronic mail addresses:

Barry_Cornelius%DURHAM.MAILNET@MIT-MULTICS.ARPA
Barry_Cornelius@uk.ac.durham.mts
bjc@uk.ac.nott.cs

All About Strings

M2WG-N95: Issue 2: 30th April 1986

Barry Cornelius

Department of Computer Science
University of Durham
Durham DH1 3LE United Kingdom

1. Introduction

This paper discusses some of the problems with Modula-2's definition of strings and introduces the two proposals that were discussed at the February meeting of the Modula-2 Working Group of the British Standards Institution. The M2WG would be interested to receive your comments on these proposals.

2. Strings according to the definitions in the Report

2.1 What is meant by 'string'?

§3 of the Modula-2 Report defines the term 'string' to mean 'a sequence of characters enclosed in quote marks'. Hence the word 'string' is used to refer to a literal such as "FRED". Throughout §2 of this paper 'string' will be used with this meaning.

§3 of the Report also says that a string consisting of n characters is of type:

```
ARRAY [ 0 .. n-1 ] OF CHAR
```

So, a string like "FRED" is of the type:

```
ARRAY [ 0 .. 3 ] OF CHAR
```

2.2 Simple uses of 'strings'

By the normal compatibility rules (defined in §6.3 and §9.1 of the Report), a string like "FRED" is assignment compatible with any variable which is also of its type. Hence, the following is allowed:

```
VAR s : ARRAY [ 0 .. 3 ] OF CHAR;
...
s := "FRED";
```

However, Modula-2 places restrictions on the types of objects which may be compared using the relational operators. In fact, §8.2.4 of

the Report says:

> the ordering relations apply to the basic types, INTEGER, CARDINAL, BOOLEAN, CHAR, REAL, to enumerations, and to subrange types.

Thus, unlike Pascal, Modula-2 does not permit strings to be used in comparisons. For example:

```
VAR s : ARRAY [ 0 .. 3 ] OF CHAR;
...
IF s = "FRED" THEN ...
```

is not permitted. However, it is usual for implementations to include a module providing operations on strings:

```
FROM Strings IMPORT CompareStr;
VAR s : ARRAY [ 0 .. 3 ] OF CHAR;
...
IF CompareStr(s, "FRED") = 0 THEN ...
```

We will now look at some of the problems that occur with the current definition of strings.

2.3 Problem 1: Assigning 'short strings' to 'long variables'

The section of the Report concerned with Assignment Statements ($9.1) gives the following feature of Modula-2:

> A string of length f can be assigned to a string variable of length t > f. In this case, the string value is extended with a null character (0C).

[The Report actually uses n1 and n2 rather than f and t; f and t mean 'from' and 'to'.] Thus:

```
VAR s : ARRAY [ 0 .. 3 ] OF CHAR;
...
s := "NW";
```

is legal and is equivalent to:

```
VAR s : ARRAY [ 0 .. 3 ] OF CHAR;
...
s[0] := "N";
s[1] := "W";
s[2] := 0C;
```

Note that the Report leaves the value of s[3] undefined.

There are three points to be made about this feature of Modula-2.

Firstly, this feature permits array assignments that leave component(s) of the array undefined. This appears very strange to those of us who believe in the approach taken by the BS/ISO/ANSI/IEEE Pascal standard. Formulating a rigorous definition where (components of) variables are permitted to have undefined values is possible but for the reasons spelled out in a paper by Derek Andrews (M2WG-N72), it would not be very easy.

The second point is that the underlying representation of a variable that contains a string is given in detail. And it is not a very convenient representation: a variable that contains a string is an array and the string ends at the first occurrence of 0C or at the end of the array if there is no 0C. It has been argued that this amount of detail about the representation of strings should not be given, and that the chosen representation is neither convenient for computers where strings are normally null-terminated nor convenient when the length of the string is stored.

Finally, the value 0C is inappropriate in some character sets.

2.4 Problem 2: What are the types of "A" and ""?

In §3 of the version of the Report given in the first and second editions of Wirth's book "Programming in Modula-2", there appears the sentence:

> A single-character string is of type CHAR, a string consisting of n>1 characters is of type
>
> ```
> ARRAY [ 0 .. n-1 ] OF CHAR
> ```

This means that:

```
s := "ABC";
```

is legal whereas:

```
s := "A";
```

is not if s is declared to be of type ARRAY [0 .. 3] OF CHAR. And, although the syntax allows the literal "", the Report remains quiet about its type. However, presumably:

```
s := "";
```

is allowed.

There are similar problems when passing such values to value parameters. Consider a procedure whose heading is:

```
PROCEDURE DoOperationsUsingStringValue(str:ARRAY OF CHAR);
```

Although the calls:

```
DoOperationsUsingStringValue("FRED");
DoOperationsUsingStringValue("NW");
DoOperationsUsingStringValue("");
```

are permitted, it is not legal to do:

```
DoOperationsUsingStringValue("A");
```

Instead one has to duplicate the code of the string handling procedure in another procedure (presumably called DoOperationsUsingCharValue).

In the third edition of Wirth's book, the sentence from the Report quoted above has been replaced by:

> A string consisting of n characters is of type
> ARRAY [0 .. n-1] OF CHAR

The third edition also includes a new sentence in §9.1:

> A string of length 1 is compatible with the type CHAR.

So "A" is now considered to be of the type ARRAY [0 .. 0] OF CHAR rather than CHAR. This means that most of the above problems have been removed. However, this new definition seems to imply that the type of "" is:

```
ARRAY [ 0 .. -1 ] OF CHAR
```

The type of "" is particularly important when "" is passed as an actual parameter to a value parameter which is an open array parameter. For example, if "A" is passed to DoOperationsUsingStringValue, then the value parameter, str, is of type ARRAY [0 .. 0] OF CHAR and thus HIGH(str) would produce 0. But what happens when "" is passed?

3. The current position of the BSI's Modula-2 Working Group

At the February meeting of the M2WG, there were (at least) two opposing points of view:

- those who thought the holes in the language definition in the area of strings could be patched up;
- those who thought that Modula-2's attempt at strings was appalling and that the language had to have a new predefined type for strings.

There was no real consensus. Instead, we decided to follow up both proposals in further detail and then to seek the views of the Modula-2 community. The two proposals are presented in §4 and §5 of this paper.

4. Proposal 1: Patch up the existing definition

A: If a program assigns a string literal to an array-of-char-variable and the literal is "shorter" than the variable, then the variable will be padded THROUGHOUT on the right with the StringTerminator value.

B: The constant StringTerminator will be defined in a module, probably, SYSTEM. For some implementations, it will have the value CHR(0).

C: The empty string literal is of type 'empty-char' which is compatible with the types ARRAY [0 .. ~~~] OF CHAR.

D: A string literal of length n (n>0) is of type ARRAY [0 .. n-1] OF CHAR.

E: A string literal of length 1 is compatible with the type CHAR.

F: If a formal parameter is a value parameter which is an open array parameter and the actual parameter is a string literal then the formal parameter will have:

- o the same type as the actual parameter if the string literal is not empty;
- o the type ARRAY [0 .. 0] OF CHAR and the value of the element will be StringTerminator if the actual parameter is an empty string literal.

G: There will be a module in the Standard library which will provide a comprehensive set of procedures for operations on string objects.

5. Proposal 2: Add a new predefined string type

A: A new string type will be added to the language. In essence, this will be similar to that available in UCSD Pascal or in Atholl Hay's proposal for ISO Standard Extended Pascal (M2WG-N65).

B: The Modula-2 language will contain ways (e.g., operators or standard procedures) to provide the operations of:

assignment,
relational operators,
string <--> char type transfers,
string <--> character array type transfers,
denotation for the empty string,
denotation for string literals,
operation to return the length of a string,
operation to concatenate two strings,
operation to return a substring from a string,
operation to find the start of a substring within a string.

C: Other operations will be provided by a module in the Standard library since such procedures may be written in terms of the language.

D: If possible, this new string type's definition will be similar to that being provided in ISO Standard Extended Pascal.

6. Postscript

Please send your comments on these two proposals to me as soon as possible. You may find the following electronic mail addresses useful:

Barry_Cornelius%DURHAM.MAILNET@MIT-MULTICS.ARPA
Barry_Cornelius@uk.ac.durham.mts
bjc@uk.ac.nott.cs

Type Conversions in Modula-2

B A Wichmann, 1986-03-25
National Physical Laboratory
Teddington, Middx, TW11 OLW, UK
Issue 5: M2WG-N67

Version for MODUS Newsletter.

Type conversions in Modula-2 appear to be somewhat poorly defined and do not necessarily function as a user might expect. For instance, the use of a type name to perform a numeric conversion actually just retypes the underlying representation of the bit pattern. An example would be CARDINAL(-1). Since -1 is not a value of type CARDINAL, the bit pattern of all ones (on a 2's complement machine), becomes the largest value of type CARDINAL, ie, MAX(CARDINAL). According to Wirth, the proper solution is to use VAL for such conversions but this is rather clumsy having two parameters rather than one.

The BSI standardization group has decided that this situation is unacceptable since type conversions are now dependent upon the physical representation. Any such dependence upon the physical representation should be in module SYSTEM and require explicit import to denote its unsafe nature. This note details a provisional decision made by the group. Formally, no decision has any effect until ISO ballots on a draft proposal. However, if you think that this proposed change is undesirable, then please convey this to me or another member of the group.

Terminology

By an unsafe conversion we mean one that depends upon the underlying physical representation (such as use of the type name in the current language). By a safe conversion we mean one with a clear abstract semantics (such as VAL currently). By a whole number type we mean an integer or cardinal type. (We speak of a cardinal type in case there is more than one.) By ordinal type we mean a whole number type, type CHAR or an enumeration type (including BOOLEAN). By an exception we mean a run-time event beyond which the semantics of the program is undefined. (Implementations may produce a warning on an exception, provide some non-standard recovery or just continue processing.)

Proposed change to conversions

There are two categories of conversions: numeric conversions and others. For numeric conversions there is a clear underlying value which should be preserved while changing type - hence the possibility of a safe conversion exists.

1. Numeric conversions

If T is an whole number or real type, and Expr is an expression of an whole number or real type, then T(Expr) is a numeric conversion of the value Expr to type T. If the mathematical value of Expr does not lie within MIN(T)..MAX(T), then T(Expr) gives rise to an exception. If Expr is a constant expression, then T(Expr) is also a constant expression, provided Expr lies in the range MIN(T)..MAX(T). (The notion of being a

constant is needed to specify other parts of Modula-2.) For conversions involving real types, the nearest value is taken, provided the result is in range.

Conversions to/from any cardinal type, any enumerated type and type CHAR are permitted (hence being a generalization of ORD and CHR) with the following exceptions: (1) if a conversion is from an enumerated type T to an enumerated type S, then the base type of T must equal the base type of S, (2) no conversions are allowed (directly) between an enumerated type and type CHAR.

Given the following declarations:

```
TYPE
   Day = (Mon, Tue, Wed, Thr, Fri, Sat, Sun);
   WeekDay = [Mon..Fri];
VAR
   INT: INTEGER;
   CARD: CARDINAL;
```

We have the following:

```
REAL(100)                  =   100.0
INTEGER(1.4)               =   1
BOOLEAN(0)                 =   FALSE
CHAR(10)                   =   CHR(10)    (* 10 is CARDINAL *)
CHAR(INT)                  =   illegal
WeekDay(TRUE)              =   illegal
WeekDay(INT)               =   illegal
WeekDay(CARDINAL(INT))     =   legal, exception unless 0<=INT<=4
WeekDay(Sat)               =   exception
Day(5)                     =   Sat  (* 5 is CARDINAL *)
BOOLEAN('A')               =   illegal
REAL(TRUE)                 =   illegal
CARDINAL(-1.0)             =   exception
```

This can be further illustrated by a table:

Expr	REAL	INTEGER	CARDINAL	CHAR	BOOLEAN	WeekDay	Day
REAL()'	Y	Y	Y	I	I	I	I
INTEGER()	E	Y	E	I	I	I	I
CARDINAL()	E	E	Y	Y	Y	Y	Y
CHAR()	I	I	E	Y	I	I	I
BOOLEAN()	I	I	E	I	Y	I	I
WeekDay()	I	I	E	I	I	Y	E
Day()	I	I	E	I	I	Y	Y

Where Y = allowed, no exception possible
E = allowed, exception possible
I = illegal

The following functions are now strictly speaking redundant:

```
CHR(card)          =   CHAR(card)
FLOAT(x)           =   REAL(x)
TRUNC(x)           =   CARDINAL(x - 0.5)
ORD(enum)          =   CARDINAL(enum)
```

None of the above functions are retained. The reason for this is that the functionality is provided by the enhanced semantics using the type name and that the effect of all the above can be obtained by a user by simply writing the procedure needed:

```
PROCEDURE FLOAT(X: CARDINAL): REAL;
   BEGIN
   RETURN REAL(X);
   END FLOAT;
```

Note that if FLOAT or TRUNC were to be retained, then problems would arise due to the possibility of several real types. Similarly, one may wish to convert a character to a short integer rather than CARDINAL. These problems are avoided by use of the type name.

Note that the procedure VAL is no longer required and hence is not retained. (One could retain it as a long-hand for a call of the numeric conversion above, but there seems no point to that.)

2. Unsafe conversions

A large amount of low-level programming can be undertaken in a 'high-level' language if unsafe conversions are permitted. On the other hand, it should be possible to identify such conversions easily because changes are likely to be needed in porting code. Hence BSI has decided that such low-level facilities should be in module SYSTEM. The procedure that is exported by SYSTEM for this purpose is:

```
CAST ( type: "Type";
            value: OfAnyType   ): type;
```

Hence the value given by the second parameter is regarded as being of the type named as the first parameter. (The parameter convention is similar to the old VAL.) There will be some restriction on the two types involved, for instance:

```
TSIZE(OfAnyType) = TSIZE(type)
```

Note that the application of CAST to pointer types could lead to insecurities outside the local context of its use.

As an example, consider the provision of a function to OR the bits of two INTEGER parameters. We assume here that INTEGER and BITSET occupy the same amount of space - a WORD. Note that BITSET also has to be imported from SYSTEM due to its machine dependence.

```
FROM SYSTEM IMPORT CAST, BITSET;

PROCEDURE ORbits (x, y: INTEGER): INTEGER;
   BEGIN
   RETURN CAST(INTEGER, (CAST(BITSET,x)+CAST(BITSET,y)) )
```

```
END ORbits;
```

As expected, this function has dangerous consequences. For instance, the semantics depends upon the representation of integers: ORbits(-1,2) gives -1 on a 2's complement machine but -2 on a 1's complement machine. The dangers inherent in this are the reason why BSI is suggesting that CAST is imported from SYSTEM.

3. Change Analysis

This note makes a 'change' to Modula-2 and hence it is necessary to see what this implies. We must therefore analyse existing Modula-2 code which in turn relies upon existing implementations. Since multiple numeric types are not currently implemented, the problem in practice involves only type CARDINAL and INTEGER. Again, on existing implementations, we have twos complement arithmetic with unsigned values for CARDINAL. We therefore have the following table:

```
I: INTEGER;
C: CARDINAL;                                ($   ANSWER   $)
```

CARDINAL(I)	Old M-2	New M-2
I in 0..MAX(INTEGER)	I	I
I in MIN(INTEGER)..-1	compiles unsafe result	exception run-time trap?
INTEGER(C)		
C in 0..MAX(INTEGER)	C	C
C in MAX(INTEGER)+1..MAX(CARDINAL)	compiles unsafe result	exception run-time trap?

Hence conversion which rely upon the underlying representation will now be trapped at run-time with good implementations rather than produce machine-dependent results. If an implementation chooses not to detect the run-time error, then the same result will be obtained. I believe that this is an acceptable change. In practice in performing a conversion, the change here is likely to be smaller than that encountered between existing Modula-2 implementations. In any case, the problem arises currently because the result of the 'unsafe result' above is not defined.

Some decisions on numerical issues

B A Wichmann, 86-07-10
M2WG: N104, Issue 1

Introduction

This note records decisions made at the BSI Modula-2 meeting on the 16th June 1986. It may be necessary to revise these decisions if widespread objections are made at the public meeting in July.

Type CARDINAL

The paper by Niklaus Wirth on the Ceres compiler created much discussion on the definition of type CARDINAL. The issues were finely balanced between removal of CARDINAL (for simplicity) or retention (for compatibility). In the end it was decided to retain CARDINAL but to remove the assignment compatibility with INTEGER. The reason for removing this assignment compatibility is that it could not easily be retained once LONGCARD and LONGINT etc are introduced.

This change might appear to be too radical. However, programs conforming to current practice will fail to compile with the new standard due to the lack of assignment compatibility. The error message would be very explicit and hence re-editing the program by adding the necessary type conversion would be very simple. One could even envisage a compiler performing this textual conversion automatically. The revised program is more strict in its use of numeric types, making subsequent changes easier. Note that since CARDINAL and INTEGER are distinct types, there is no change in the rules for VAR parameters.

Different numeric types

It was decided to accept the proposal in N79 that all implementations must provide three cardinal, integer and real types. These types will have standard names. The type names CARDINAL, INTEGER and REAL will be retained, but no decision has been made on the identifiers for the other two cardinal, integer and real types. All three CARDINAL types (say) will be regarded as distinct, but an implementation would be free to implement them by means of one physical type. Note also that the range of CARDINAL could be such that it can be implemented using signed integer arithmetic. Hence the minimal hardware support for (efficient) Modula-2 is one integer and one real type. The advantage of having three fixed types is that no generic mechanism is needed - three explicit versions are produced if required.

The type of subranges

The problem is to decide the base type of a subrange as in:

```
V: TypeId [ M : N ];
```

This is equivalent to:

```
V: [ TypeId(M) : TypeId(N) ];
```

Hence the use of TypeId in the first form is strictly unnecessary, but is retained for compatibility with existing code. We now only need to consider the case:

```
V: [ M : N ];
```

where M and N are constant expressions (of an ordinal type(s)). If both expressions have the same base type, then that is the base type of V. If they have different base types, then the program is illegal. The only remaining cases are when type ZZ is used (which cannot be a base type). This occurs when literals are used.

If one expression is of type ZZ, and the other of a whole number type, then the base type is the other type. Hence the only case still to resolve is when both expressions involve literals only and are of type ZZ. Then, if M >=0, the base type is CARDINAL, otherwise the base type is INTEGER. Note that a subrange declaration can be illegal if the declared range is not within the range of the base type.

The naive user can forget these rules, inserting TypeId when in trouble.

Examples

Range	Base type/ illegal
CARDINAL[1: 3]	CARDINAL
CARDINAL[INTEGER(1): 3]	CARDINAL
[INTEGER(1):CARDINAL(3)]	illegal
[TRUE:TRUE]	BOOLEAN
[1: 3]	CARDINAL
[1:INTEGER(3)]	INTEGER
[-1: 3]	INTEGER
CARDINAL[-1:3]	illegal
INTEGER[-1: MAX(CARDINAL)]	probably illegal since MAX(CARDINAL) > MAX(INTEGER)

The last example illustrates a portability problem, but this is no worse than that occuring with the use of literals.

Improving the quality of Definition Modules

Arthur Sale

Department of Information Science – University of Tasmania
GPO Box 252C HOBART Tasmania Australia 7001

An important aspect of Modula-2's definition modules is that they should contain *all* the information that a user of the module needs. However, frequently the writers of modules do not seem to realise this and require the user to read documentation or even parts of the implementation module in order to understand what services the module provides. Where such information is provided, it is frequently in an imprecise form.

To improve this situation, definition modules should be written so that each exported object is well defined. Each kind of object is discussed below. The case of procedures is the most important.

Constants
The derivation of the value should be clearly stated together with any relationship to other constants or exported objects. If the constant is one which the user may modify this should be indicated.

Types
The uses of the type should be stated. If the procedures implement operations on values of the type, then the user's attention should be directed to the relevant procedures. For transparent types which contain enumeration types an explanation of the constant values is probably also required.

Variables
If an exported variable introduces an insecurity (in other words if the user of a module modifies it value, the module is not guaranteed to conform to its definition) then the decision to export it is a poor one. If no insecurity is introduced because the module itself never uses the value of the variable, then there is no reason why a function procedure should not be exported instead (except for the rather spurious reason that a variable access might be faster than a function call). This has the advantage that an inadvertent alteration of the value that was set by the module cannot occur. Variable export should never occur in high quality modules and thus no convention is needed.

Procedures
It is now well accepted that statements and procedures can be semantically defined by means of *pre-* and *postconditions*. The convention should be to provide a precondition and a postcondition for each procedure or function procedure, specified as formally and precisely as the application permits, together with an informal description of the procedure's action [Gries 1981, Sale 1986]. The informal description may be omitted if the action is obvious from the pre- and postconditions but these conditions should never be omitted. For programmers who are unfamiliar with the theory of *predicate transformers* [Dijkstra 1976], the meaning is as follows:

> *If the precondition is satisfied when the procedure is activated, then after the activation the implementation of the procedure is guaranteed to satisfy the postcondition. The precondition is conventionally assumed to be the weakest possible precondition (ie no extra constraints are included in it that are not essential to this guarantee).*

References

Dijkstra, E W. *A Discipline of Programming*. Prentice-Hall, Englewood Cliffs, 1976.
Gries, D. *The Science of Programming*. Springer-Verlag, New York, 1981.
Sale, A H J. *Modula-2: Discipline & Design*. Addison-Wesley, London, 1986.

Example

The following definition module is an example of this style of programming. It is extracted from a CAD package and was not especially written for this Note. The pre- and postconditions are written in an informal style but with some care that they do define the semantics of the exported procedures.

```
DEFINITION MODULE BooleanFunction;

(*  This module provides a uniform interface for the acquisition of a multi-output Boolean function, and for the
    disposition of such a function. Different implementations may supply different remote forms on storage
    devices, such as standard cube format, UI-MINI format, or others. The module is intended for use with logic
    minimization, analysis and partitioning tools, and for input to VLSI programmed logic array (PLA)
    generators. *)

(*  © Copyright 1986 Prof A H J Sale
    Department of Information Science, University of Tasmania,
    GPO BOX 252C, HOBART Tasmania, Australia 7001. *)

  (*  The following EXPORT statement is not needed in Modula-2 systems that conform to the
      1984 revisions. *)

  EXPORT QUALIFIED
      MaxInOut,
      InOutRange, Ternary, F, T, X, CubeAndOutType,
      AttachFunction, DetachFunction, Attached, More,
      SupplyInOutNumbers, SupplyNextSpec,
      WriteFunction, CloseFunction, WriteAllowed,
      WriteInOutNumbers, WriteNextSpec;

  CONST
      (*  MaxInOut must be >= (number of inputs + number of outputs).
          Recompile module if the value is changed. *)
      MaxInOut = 99;

  TYPE
      InOutRange = [0..MaxInOut];

      (*  As a logic component, the following correspondences hold:  F = 0, T = 1, X = don't care. *)
      Ternary = (F,T,X);

      (*  This type is the format of an inputs (or logic cube) and outputs specification. The first (number of inputs)
          components are the input specification, while the next (number of outputs) components are the
          corresponding output specification. *)
      CubeAndOutType =
        ARRAY InOutRange OF Ternary;

(*  PROCEDURES ASSOCIATED WITH INPUT OF A LOGIC FUNCTION. *)

PROCEDURE AttachFunction(Name: ARRAY OF CHAR);
(*  Precondition: Any previous call to this function has been followed by a call to DetachFunction.
    Postcondition: The function identified by the parameter is regarded as attached to the module. Attached returns
      TRUE unless the attachment was unsuccessful. *)
```

```
PROCEDURE DetachFunction;
(*      Precondition: A function is attached (ie a successful call to AttachFunction has not yet been followed by
          a call to this procedure.
        Postcondition: The function is detached.  Attached returns FALSE. *)

PROCEDURE Attached( ): BOOLEAN;
(*      Precondition: None.
        Postcondition: Returns TRUE if and only if a function is attached. *)

PROCEDURE More( ): BOOLEAN;
(*      Precondition: A function is attached (see above).
        Postcondition: Returns TRUE if there remain specs to be supplied. *)

PROCEDURE SupplyInOutNumbers(VAR InputVars, Outputs: CARDINAL);
(*      Precondition: A function has been attached.
        Postcondition: The number of input variables and the number of outputs is returned in the parameters. *)

PROCEDURE SupplyNextSpec(VAR Data: CubeAndOutType);
(*      Precondition: A function has been attached AND More returns TRUE.
        Postcondition: The next spec is returned in the parameter.  The result of the function procedure More will
          depend on whether this is the last or not. *)

(*   PROCEDURES ASSOCIATED WITH OUTPUT OF A LOGIC FUNCTION. *)

PROCEDURE WriteFunction(Name: ARRAY OF CHAR);
(*      Precondition: Any previous call to this function has been followed by a call to CloseFunction.
        Postcondition: The writing of a function is permitted, to be identified by the parameter.
          WritingAllowed returns TRUE unless it is not possible to write a function (eg no room).  *)

PROCEDURE CloseFunction;
(*      Precondition: The writing of a function is permitted (ie a successful call to WriteFunction has not yet
          been followed by a call to this procedure).
        Postcondition: No further writing to the function is permitted, and it is regarded as no longer associated
          with the module. *)

PROCEDURE WriteAllowed( ): BOOLEAN;
(*      Precondition: None.
        Postcondition: Returns TRUE if and only if writing to a function is permitted. *)

PROCEDURE WriteInOutNumbers(InputVars, Outputs: CARDINAL);
(*      Precondition: WriteAllowed returns TRUE and neither WriteInOutNumbers nor WriteNextSpec have been
          called since the call to WriteFunction.
        Postcondition: The parameters are accepted as the number of input variables and the number of outputs and
          written to the function. WriteAllowed returns TRUE if the write was successful. *)

PROCEDURE WriteNextSpec(Data: CubeAndOutType);
(*      Precondition: WriteAllowed returns TRUE and exactly one call to WriteInOutNumbers has been made
          since the call to WriteFunction.
        Postcondition: The parameter is accepted as the next spec and is written to the function.
          WriteAllowed returns TRUE if the write was successful.*)

END BooleanFunction.
```

A PROGRAMMING ENVIRONMENT FOR MODULA-2

First, I want to thank you for a wonderful source of information on Modula-2; I really look forward to every issue of MODUS Quarterly.

INTRODUCTION

This article is about a dream, and the realization of it. Back in 1983, I started working on my first Modula-2 compiler. It was not an easy task, and even if I finally got something which can be called a result, I was not at all satisfied with my compiler in particular, and the process of creating and maintaining large software projects in general. I thought on Modula-2 as an excellent car with no fuel; the language needed an environment.

This was in july 1984, and the planning took nearly six months (I was, and still am, a student). When designing PEM (short for Programming Environment for Modula-2), I wanted it to support not only the process of entering Modula-2 statements, but in fact the whole software life-cycle. I learned this: no design is final! There is always something that could have been done in a more elegant way. I found the following to be very important in a programming environment:

* the ability for tools to exchange information
* that the tools are controlled in a similar way
* the ability to have several tools resident and jump from one to another without the need for actually terminating them.
* the ability to let more than one tool be visible at the same time -- windows.

Reaching the end of the planning cycle, these were the most important conclusions:

1) The system should be operating system independent, and should neither include nor support any conventional file system (opposed to many LISP systems). Instead, PEM should operate at a higher abstraction database level with projects, documents, libraries etc.

2) The system should consist of a kernel library (holding low-level drivers, resource management, disk I/O, application and window management), an administrative part to support the first stages of the software life-cycle, and a development part for Modula-2/EBNF related tasks.

3) The system should be controlled by a Core Editor (similar to an operating system's command interpreter). The core editor should have its own Modula-like interpretive "batch"-language. I named it CEL, short for Core Editor's Language. CEL should support declaration of constants and variables, and procedure declarations (for batch operations). The statements supported should be a small subset of Modula-2 statements, including procedure calls for entering commands or performing batch operations. The Core Editor should support context sensing and switching, making it unnecessary to shift up the command level for access to each new tool. Instead, an active tool should have commands installed in the core editor, which are rerouted to - and executed by - the tool itself. Commands entered should have the same syntax as Modula-2 procedure calls, and the scheme should be implemented via procedure variables.

4) A DIANA-like representation for Modula-2 modules was needed, combined with incremental compilation (see PATCHING AND SPLICING). The system should support large-scale systems (75,000 lines and more), because this is where Modula-2 really comes to use. Division of systems into subsystems should be supported, allowing one programming site to be "merged" with another without any fuzz.

5) The system should support the whole software life-cycle. (Here, I examined each step of the software life-cycle, thinking of the tools which are required).

THE STRUCTURE OF PEM I:

KERNEL, CORE EDITOR, NIKLAUS, COMMUNICATION, DESK TOOLS

Since PEM is operating system independent, it must be an operating system in itself or make use of an underlying one. The current implementation is compiled with LogiTech's system on an IBM PC AT with 3,6 Mbytes of RAM, and here I have both PC-DOS and LOGITECH's "standard" library, but on the NS32332 Modula-2 workstation I'm designing, PEM will represent the lowest abstraction level one can work with. The disk structure will be a direct implementation of the PEM database, and there will be no "files" or "directories" in the manner of traditional operating systems like UNIX and PC-DOS.

PEM is divided into eight main parts:

1) THE KERNEL (modules for task-management, memory-swapping, resource management, window management, drivers etc.)

2) THE CORE EDITOR

3) DESK TOOLS

4) THE ADMINISTRATIVE PART

5) NETWORK MANAGEMENT/COMMUNICATION

6) THE CONFIGURATION EDITORS

7) NIKLAUS (an expert system for design analysis, program refinement and debugging, now a part of the Modula-2 Object Editor)

8) THE DEVELOPMENT PART (EBNF & MODULA-2)

The kernel is the low-level part of PEM, representing an interface to (or being the actual) operating system. Some modules are available to applications (= tools), and others are only available trough the entering of commands in the core editor.

The core editor is the heart of the system. Applications being active have their own commands installed, and these commands are rerouted to the applications themselves. This means that all applications are controlled (receives commands trough) the core editor. However, applications are free to make use of pop-up menus and menu-bars for the sake of user frendlyness. With the exisiting PEM applications, you usually can have it both ways.

The kernel also incorporates a mailbox concept, allowing applications to exchange information. Applications can "subscribe" for different types of data, and they will receive that information (through a 'reader' procedure variable) whenever another application calls the MailBox module to send some. There are two ways of sending information; you can send to one specific application or to all "subscribers" of a type of information.

The desk tools are quite "standard"; an SideKick-like calculator also supporting octal arithmetic, a phone/address list, a notepad and an appointment calendar combined with an alarm clock. An ASCII table is also included among the desk tools.

The administrative part of PEM contains a Document Manager (word processor), a Project Manager, a relational database system, a spread sheet and a simple drawing tool. It supports the planning process and the process of writing documentation.

The network/communication section of PEM is neither yet implemented nor clearly specified; it will be fully implemented when PEM is ported to the NS32332 workstation I'm designing. (The IBM PC AT is too weak a machine to cope with advanced networking and PEM). Today, it includes a terminal emulator which takes up to 9600 baud). The only facility which will be added to this section on the AT, is the ability to use a Hayes-compatible modem for auto-dialing (and perhaps remote UNIX access).

The configuration editors complete the system's impression of great flexibility. You can reconfigure virtually anything, and it is possible to keep different configurations for each project and/or user (I must stress that PEM is a single-user system, but several users can share the same disk and/or work on the same project. Also, on the future workstation, PEM machines can be connected in a LAN).

NIKLAUS (hope it's ok, prof. Wirth!), is an expert system I've included in the Modula-2 Object Editor. Among other things it will detect many logical errors as you enter them, and it will also "learn" the most common error situations occuring as you use the system.

THE STRUCTURE OF PEM II

THE DEVELOPMENT PART -- EBNF TOOLS AND MODULA-2 OBJECT EDITOR

In PEM all kinds of information is supported with an object editor especially made for editing it. This is also for Modula-2 and EBNF -- and even for CEL (Core Editor's Language). In this section I will concentrate on the tools/applications operating on Modula-2 and EBNF.

EBNF is a notation which is used to describe syntax (or, if you want to, data formats). It was used to define the syntax of Modula-2, and can certainly be used for documentation purposes when writing software. Apart from the EBNF Object Editor, which is used to edit EBNF declarations, there is an EBNF Pretty- printer, an EBNF Data Editor and an EBNF Data Verifier.

The Modula-2 tools in the PEM development part make out 70% of the system. One of the most important among these, is the Modula-2 Object Editor. It is very powerful, and its design was of course heavily influenced by the DIANA-like representation of Modula-2 source under PEM. The main features of the Modula-2 (multi-window) Object Editor are:

* Built in advanced Modula-2 Code Browser
* Automatic statement transforming
* A fill-in capability in import/export lists
* Statement wrapping and unwrapping
* Built in templates for all declarations and statements
* Built in syntactic and semantic checking (compilations always successful)
* Intelligient DWIM (Do What I Mean) capability on identifiers and Modula-2 source in general.
* Dependency-tree analysis with automatic activation of the compilation system for recompiling all modules depending of the definition module you've changed. The editor can also lead you trough all the places in the respective clients, informing you what you must change (adding a parameter, for example). The lead-change process is also available within modules. Also note that you can make queries trough the core editor like "what if I changed the declaration of the type WindowInfo ?", and the editor will show you all the modules in which you would have to make changes and recompile.
* the ability to exchange information/commands with other Modula-2 tools (all tools have this capability)
* a powerful local/global rename facility
* optionally, the user can be forced to follow Jirka Hoppe's (quite popular) naming convention.
* NIKLUAS, a Modula-2 Code Export System has been integrated into the editor, detecting logical errors as you enter them. It will also "learn" about new error types and situations and build and maintain a database used when spotting logial errors.

The Code Browser is advanced in the way it is cooperating with the rest of the Modula-2 Object Editor and in its flexibility. You can command it (trough a Core Editor command, of course) to spy on your editing activities. When you enter "ReshapeWindow(w)" and want to know more about the declaration of it, the code browser will (in this case where two more parameters are required) warn you and display the part of DEFINITION MODULE WindowHandler where this procedure is defined.

The fill-in capability means that you can specify
"FROM WindowHandler IMPORT <all object which I refer to>",
"IMPORT <all modules which I refer to>" and
"EXPORT <all objects from this local module which I refer to".
On name conflicts, the user will be notified and forced to take some action, of course.

The DWIM facility on identifiers means that if you try to enter "i := 15", and the only visible identifier looking like "i" is "I", the editor will give a "Hmm. Neither visible nor declared. Did you mean 'I' (y/n)". The DWIM facility also lets you say to the editor "Write the code to assign elements [5..16] in array 'x' to elements [8..19] in array 'y'". This works for other statement types as well.

Statement transforming means that the editor can transform one statement into another having the same effect. Examples are: IF to CASE, WHILE to REPEAT to LOOP.. etc. This facility can also be used to change x := x + n type of statements TO INC(x,n).

The PEM Object Editor is a structure-based editor. You can enter the lengthy "P.R.O.C.E.D.U.R.E" and the editor will recognize it, but you can also press <ALT>-<P> (on the AT), and get

```
PROCEDURE <ident> [parameters] [return-type] ";"

END <ident>;
```

Then you just fill in! This really speeds up typing, but at the same time it is NOT a strait-jacket editor. When you're editing you can decide a great deal of the programming style, where you want blank lines, how you want comments to look, etc. The main part of the programming style, however, must be altered with one of the configuration editors.

In a proper programming environment, a version manager to keep track of different versions and module histories is a must. PVAHM, the Pem Version And History Manager, is closely integrated with the Modula-2 Object Editor to provide "undo/recall earlier version" type of operations.

A library manager has been included to perform operations on libraries of Modula-2 modules. Libraries can be deleted, merged, split, created, renamed etc. One can also move libraries between projects or make them common to all projects. The library manager can produce module dependency (import) trees and software bus diagrams for documentation.

A Modula-2 pretty printer is included (closely integrated with the Modula-2 Object Editor) for giving hard-copies of Modula-2 modules. Also included are a Modula-2 absolute linker and a loader. At the moment dynamic linking is not supported. This is due to the fact that PEM is such a big system that I have not been able to test compiled programs on the machine running it. However it works beautifully for embedded system's applications, and here, dynamic linking is almost never needed since the software is to be stored in read-only memory. On the future workstation (and in the AT XENIX version) though, I will support dynamic linking and the kernel will support dynamic module residency like OS-9 and MEDOS. PEM is not an ordinary programming environment; it is meant to include code generators for several processors, and therefore, software being debugged is more likely to be down-loaded to an external CPU. Dynamic linking will therefore only be supported on the future NS32332 workstation running PEM.

Among the tools for debugging and analysis are a post-mortem dump analyzer and a symbolic run-time debugger with a very nice facility: you can test a procedure by calling it by name! Example:

```
"WriteString(window,"this is a test")".
```

An execution profiler for bottle-neck analysis and timing is also included, closely cooperating with the simulator and the debugger.

The great thing about the Modula-2 tools is the way they can cooperate, bringing the software development rate to new heights.

AN EXAMPLE

In this example I will demonstrate how PEM is revolutionary among Modula- 2 environments.

Imagine we are debugging the MODULE WindowDemo. We are in the debugger window where the source code is displayed. We have just found a bug in the procedure OpenAllWindows, we have "forgotten" to open the windows called "commandWindow" and "menuWindow". We also want a "hello" on top of the latter. Therefore, we must add some code...

```
/-------------(in debugger's source code window)---------------\

MODULE WindowDemo; (* Frode L. Odegard, MAR 1986 *)

 FROM DiskWindows   IMPORT LoadAndCreateWindow;
 FROM WindowHandler IMPORT ....

 .....
 PROCEDURE OpenAllWindows;
 BEGIN
  LoadAndCreateWindow(22,windowOne); (* 22/23 = window ref. no *)
  LoadAndCreateWindow(23,windowTwo)  (* windowOne/Two is result*)
 END OpenAllWindows;

 ....
END WindowDemo.

\----------------------(end source code window)----------------/
```

Now, we activate the editor, adding some lines:

```
/--------(Modula-2 Object Editor Main Source Code Window)-------\

MODULE WindowDemo; (* Frode L. Odegard, MAR 1986 *)

 FROM DiskWindows   IMPORT LoadAndCreateWindow;
 FROM WindowHandler IMPORT ....
 FROM WindowIO      IMPORT WriteString;

 .....
 PROCEDURE OpenAllWindows;
 BEGIN
  LoadAndCreateWindow(22,windowOne); (* 22/23 = window ref. no *)
  LoadAndCreateWindow(23,windowTwo); (* windowOne/Tow is result*)
  LoadAndCreateWindow(93,commandWindow);
  LoadAndCreateWindow(37,menuWindow);
  WriteString(menuWindow,"hello")
 END OpenAllWindows;
 ....
 END WindowDemo.

\--------(End Modula-2 Object Editor Main Source Window)--------/
```

We activate the compilation system specifying SPLICE and PATCH options. SPLICE means that incremental compilation is performed; only the new edits are recompiled, the new code is spliced into the PEM database. To put it simple, the machine code for the new statements becomes a sort of a subroutine called from the insertion point. The machine code for the new edits is put at the end of the module. PATCH means that the new code is patched directly into the workspace of the debugger. The reference tables for the debugger are also updated. The communication between the tools mentioned in this example is implemented trough the earlier described mailbox concept.

By now we can switch to the debugger again, as nothing had happened at all! With one bug less on our source code. And the best thing is that we never really left the debugger... It was there, all the time.

With PEM, patching elevates from a black art to a feature.

IMPLEMENTATION DETAILS

The language implemented will follow the BSI standard.

REALITIES

The first commercial release of PEM will be spring 1987 on the IBM PC AT. A minimum of 20Mbyte (40 is better) hard disk storage and 3.6 MBytes RAM is currently required, but I hope to solve the RAM problem by introducing memory swapping. A new DOS 4.0 or 5.0 supporting the iAPX 286 chip's 16 MByte capability and with proper virtual memory technique would no doubt be of great help.

I WISH TO THANK:

* Dick Karpinski and Randy Bush (for extensive phone-help)
* Niklaus Wirth (for M-2 and for answering important questions)
* Paul Levy and Michael Devlin (Rational Inc.) (for the core editor concept)
* My Father, Bjarne L. Odegard, for lending me an IBM PC AT during the two years of development (still going on).

Yours Sincerely,

Frode L. Odegard

ACADEMIC MODULA-2 SURVEY

24 August, 1986

A little while ago, I asked for information about where and how Modula-2 was or was about to be used in academic courses. I am slowing acccumulating responses. I am pretty sure that there is a lot more activity than this out there. When I get some more, I will again repost.

If you are currently using Modula-2 in a class, or are about to do so, I would appreciate hearing from you. I will update and repost this survey as I get additional information. I also appreciate getting corrections and amplifications. I certainly need all the help that I can get.

Please send it to me in the following format:
(a) institution,
(b) course number, name
(c) level of course (freshman, soph, junior, senior, grad, etc.),
(d) text used,
(e) compiler used,
(f) hardware configuration,
(g) contact person,
(h) miscl.

If you have information about a Modula-2 compiler, please tell me:
(a) Source: person (if relevant), organization, address
(b) Machine
(c) Operating system

... Larry Mazlack PMSILJM.UCCCVM1.BITNET

institution	course #, name	levl	text	compiler	configuration	contact
Bethel College	COS 331 Data St & Algorithms	j	Stubbs & & Webre Ford & Weiner	Powell	UNIX4.2/VAX750	Glen Wieb Eric Gossett
U. British Columbia	CPSC 210 data struc & softw design	s	Beidler& Jackowitz + Kruse	ETHZ MacLogimo v 1.2	Mac 512	Vincent Manis
U. California at Berkeley	CS162 Operating Systems	j,sn	Wirth	Powell	UNIX/VAX750	
U.Cal,Berkeley Extension	X440 Advanced Prog w M2	m	Ford/Wein Wirth	Logitech	IBM PC	John Eckstrom
U.Cal, Davis	ECS 110 Data St ECS 140 Prog Ln	j,sn j,sn	Stubbs& Webre + Standish Wirth	Powell	UNI/VAX750	Bruce Martin
Camosun College	Comp270 Data Commun	s	Wirth + data comm	Logitech	IBM PC/XT	Darrell Wick

institution	course #, name	levl	text	compiler	configuration	contact
Canisius College	CSC250 Fundamen	s	Gleaves	Hamburg	VMS/VAX8600	Ron Curtis
	CSC311 Architec	j	Gleaves	Powell	UNIX4.2/VAX750	
	CSC333 Op Sys	j	none	IBM PC,AT	DOS/Logitech	Jim Leone
	CSC350 Soft Eng	sn	none	Powell	UNIX4.2/VAX750	
U. Colorado at Colorado Springs	CS110 Intr Prog	f	Sincovec & Weiner	Logitech v1.1 -----> v2.0	Zenith ZW-2xx (IBM AT compatible)	Richard Sincovec
	CS145 Data Str & Algorth	f				
	CS245 Data & Al	s				
U. Delaware	CIS135 Modula-2	f	Ford&Wien	Powell	UNIX4.2/VAX785	Fritz
	CIS180/1 Int CS	f	Ford&Wien	Powell	UNIX4.2/VAX785	Myers
	CIS361 Op Sys	j	none	Logitech	DOS/IBM PC	Ball
	CIS471 Compiler	sn	Wirth	Powell	UNIX/VAX	Gohkal
	CIS663 Op Sys	g	Wirth	Powell	UNIX/VAX	Sethi
	CIS672 Compiler	g	WIrth	Powell	UNIX/VAX	Gohkale
Georgia Tech	ICS 3410 Prg Ln	j	King	Logitech	IBM PC	Leaver wort
	ICS 4430 Intro Op Sys	sn	King			Ahamad
	ICS 4830 SW Eng	sn	King			Kim King
	ICS 6430 Op Sys	g	King			Dasgupta
Hampshire College	OCS215 Data Str	s	open	Hamburg	VMS/VAX750	Richard Muller
Johns Hopkins University	52.354 Op Sys	j,s, g	Wirth + OS text	Powell	UNIX 3.4BRL computer=???	Anna Hopkins
Imperial Coll (London, Eng.)	CS 120 Prog 1 -- 2 quarters	f	Wirth	Tartan	UNIX/VAX750	S.Eisen-bach
Indianapolis U, Bloomingon	C431,2 Compiler	sn,g	Wirth	Hamburg	VMS/VAX780	Chingman Jim Lo Bill Wang
	C435,6 Op Sys	sn,g				
	C445,6 Inf Sys	sn,g				
	C481 Cmp Graphc	sn,g				
Indiana U - Purdue U at Indianapolis	CSCI 450 Softwr	sn,g	Wirth	Hamburg	VMS/VAX780	Chingmin Jim Lo
	CSCI 461 Prg Ln	sn,g				
	CSCI 502,503 Compilers	sn,g				
	CSCI 503 Op Sys	sn,g				
Kansas State U	CMPSC405 Prg Ln	j	Wirth	Modula Corp	Zenith Z-150 (IBM clone) on an Ethernet	Harvard Townsend
	CMPSC420 Op Sys	j,sn	Gleaves			
	CMPSC820 Op Sys	g	none			
U. Lowell	CS305 Comp Arch	j	Tannen-baum Hammacher	Powell	Ultrix V1.0 --->UNIX4.3 VAX750	David Landskov
U. Lund (Lund, Sweden)	DA 403 Data Str	s,jn	Astor (in Swedish)	Hamburg	VMS/VAX780	Christiar Collberg
MIT	66.035 Com Lang Engnring	j,sn	open	???	DEC 20	John Guttag

institution	course #, name	levl	text	compiler	configuration	contact
Ohio U.	CS458 Op Sys II	sn,g	Walker:M2 Beck:Sys	??? ???	VMS/VAX750	Klaus Eldridge
Oxford U Prog Rsch Grp Oxford,England	Math & Computation	f,sn	open	Hamburg SUN	VMS UNIX4.2/SUN	Jonathen Bowen
Pennsylvania State	CS497A Data Str	g	Wirth--> Ford, Wiener	Powell	UNIX4.2/VAX780	Parker
U. Portland	CS 261 CS-I ** CS 351 Data St	s ?	? ?	Logitech	VMS/VAX780	Garland Bayley
U. Reading (United Kingdom)	C310 Int Op Sys c410 Op Sys Fdm	j j	Wirth,Joy Wir,Joyce	U New So Wales	Unix V7 PDP 11/44	A. Pell P. Martin
Rockefeller U	prog skills for lab computers	mi	Christian	Logitech	IBM PC	Kaare Christian
San Francisco State U	CS 415 Op Sys	sn	Ben-Ari, Deitel, Wirth	Logitech	IBM PC-XT-AT	Stan Osborne
SUNY Buffalo	CS 113 Intro ** CS 114 Intro2 CS 305 Prog Sys CS 505 Prog Sys	f f sn g	open Weiner&Si Gleaves Gleaves	open open Powell Powell	VMS/VAX clustr VMS/VAX UNIX4.2/VAX780 UNIX4.2/VAX780	Deepak Kumar
Technisce U. Berlin West Berlin	Modula2 Kompakturs	j	Gleaves Wirth	Cambridge adapted to UNXIXV	UNIX V, rel 1 UnSoft	Anfried Ossen
U. Texas at Austin	CS 372 Op Sys	j-sn	M2:notes OS:Peter-	Powell	UNIX4.2/VAX780	Jeff Brumfield
U. Utah	simulation CS 562 Parallel Comp Arc	j,sn sn,g	Wirth Wirth	Powell Powell	UNIX4.3/VAX780 VAX860a6	Richard M Fujimoto
U. Wisconsin at Madison	CS 564 Intro DB **may be no more due to compiler problems**********	sn,g	Wirth	Powell	UNIX/VAX750	Michael J Carey
U. Zurich (Switzerland)	contrl simple experiments	s,g	Wirth	ETHZ RT11 os: RT-11	DEC PRO-380	Patrik Eschle

** ---> under development

Level:
- f = freshman
- s = sophmore
- j = junior
- sn= senior
- g = graduate
- mi= miscellaneous

The information that I have on the compilers is:

(a) Cambridge:

VAX/UNIX4.2BSD Also cross-compiled for 68000up
Peter Robinson/Piete Brooks
(pr%cl.cam.ac.uk@cs.ucl.ac.uk & pb%cl.cam.ac.uk@cs.ucl.ac.uk)
Cambridge U Computing Laboratory
Corn Exchange Street
Cambridge CB2 3QG
England
(tel: +44-223-352435)

EMS sys iii (UNIX V, rel 1)
IBM4381 (VM370)
for UNIX version:
Arnfried Ossan
OSSEN at DBOTUI11.BITNET
AO at DBOTU16.BITNET
for VM370/IBM:
Thomas Havernoll
HABERNOL at DBOTU11.BITNET

(b) CERN: runs on VAX ccross-compiled for 68000 up
IBM3081, ND Nord 500, CDC Cyber 835
op sys: UNIX4.2BSD, VMS3.5, IBM MVS, SINTRAN 3, NOS/BE 1.5

David Foster/Horst von Eiccken/Julian Blake
(david@cernvax.uucp)
Data Handling Division
CERN
CH-1211
Geneva 23
Switzerland

(c) Djavaheri Brothers: runs on MC68000 UNIX systems
currently all academic users are on SUN equipment

Stan Osborne
(415) 341-1768
ucbvax!dual!dbi!stan

(d) ETHZ RT11: a modification of ETHZ that runs on DEC PRO-380
(PDP-11 architecture)

PatriK Esche
Physics, U Zurich
K538911@CZHRZUA.BITNET

(e) Hamburg: runs on VAX/VMS
often packaged with the Wirth book. Available from:
J W Schmidt/H Echhardt/J Koch/M Mall/P Putfarkeni

J.W. Schmidt
University of Frankfurt
Lehrstuhl DBIS
Box 11 19 32
D-6000 Frankfurt/Main 11
Federal Republic of Germany
rbiffm@eckhardt.UUCP ...!mcvax!unido!rbiffm!eckhardt

(f) Logitech: runs on IBM PC's, VAXs (op sys=VMS). Available from:
Logitech, Inc
805 Veterans Blvd
Redwood City, CA 94063

(g) Modula Corp: runs on Macs, IBM PC,XT, and clones, MS-DOS 2.0 and above
Modula Corp
950 N. University Ave
Provo Utah 84604
(801) 375-7400

(h) Powell: runs on VAXs, either UNIX or ULTRIX. Available from:
Michael L. Powell
Digital Equipment Corp.
Western Research Laboratory
100 Hamilton Ave
Palo Alto, CA 94301

(i) SUN: runs on SUN-3/UNIX4.2BSD
SUN Microsystems Inc.
2550 Garcia Avenue
Mountain View, California 94043
(tel: +1-415-960-1300)

(j) Tartan: runs on VAX/UNIX4.2BSD, RT PC/AIX, RT PC/ACIS
(AIX is UNIX sys 5 clone, ACIS is 4.2 clone)
Scott Wilson
Tartan Labs
461 Melwood Ave
Pittsburgh, PA 15213
(412) 621-2210

(k) UNIX V7: runs on PDP11s
Jeffrey Tobias
Department of Computer Science
University of New South Wales

I'll cheerfully update the compiler information, as I get it.

I'll also try to reply with a confirmation when you send me something. However, make sure to include your E-mail address. Using "R" has erratic results.

Larry Mazlack

UUCP	{tektronix,dual,sun,ihnp4,decvax}!ucbvax!ucbernie!mazlack
New style	PMSILJM.UCCCVM1.BITNET
ARPA | CSNET	mazlack%ernie@berkeley.ARPA
BITNET	PMSILJM.UCCCVM1.BITNET
telephone	(415) 528-0496
snail	CS Dept, 571 Evans, U. California, Berkeley, CA 94720

ETH EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE ZÜRICH
Institut für Informatik

Zürich, 1. Dec. 1985/NW

Compilers for Modula-2

For the last several years, the Institut für Informatik of ETH has distributed under license agreement two Modula-2 compilers: compiler M2RT11 for the PDP-11 under the RT11 operating system and a derivative, compiler M2M, generating M-code for the Lilith computer. The Institute is now **discontinuing distribution.**

At the same time, a new compiler is being released. This is a fast single-pass compiler, currently available for Lilith in source and object form. Versions for the 68000 and 32000 processors will be made available in source form in January. The compilers are distributed by

Modula Corporation
950 N. University Ave.
Provo, UT 84604 / USA

Further Implementations of Modula-2

The following Modula-2 compilers have been reported as being available. For further information please contact the indicated sources directly.

PDP-11 RT11 (revised M2RT11)	School of Computing Studies Queensland Institute of Technology G.P.O. Box 2434 Bridsband, Queensland 4001 / Australia
PDP-11 RSX-11 (adapted M2RT11)	Dr. Josef A. Muheim BBC AG Abt. ESL / Werk Turgi CH-5401 Baden / Switzerland
VAX - VMS (ported M2RT11)	Fachbereich für Informatik Universität Hamburg Schlüterstr. 70 D-2000 Hamburg 130 / Germany
VAX - UNIX	Mr. P. Robinson Computer Laboratory University of Cambridge Corn Exchange Street Cambridge CB2 3QG / England
Intel 8086 (ported M2RT11)	LOGITECH, Inc. 165 University Avenue Palo Alto, CA 94301 / USA LOGITECH SA CH-1143 Apples / Switzerland
Motorola 68000 (ported M2RT11)	Dr. H. Burkhart Institut für Elektronik, ETH Zürich CH-8092 Zürich / Switzerland

M2RT11 / IBM PC / MacIntosh	Modula Research Institute 950 N. University Ave. Provo, UT 84604 / USA
IBM PC	Interface Technologies Corporation 3336 Richmond, Suite 200 Houston, Texas 77098 / USA
VAX	Dr. Michael Powell DEC Western Research Laboratory Hamilton Ave. Palo Alto, CA 94301 / USA

For further information we also refer to the Bulletin of the Modula-2 Users' Association, which can be ordered from

Modula-2 Users' Association
P.O. Box 51778
Palo Alto, California 94303 / USA

or

Aline Sigrist
MODUS Secretary
ERDIS SA
P.O.Box 35
CH-1800 Vevey 2

MODUS USA Members

Abrahamsen, Barry 98103
Acreman, Chris 78752
Adelaide, The Univ. of AUSTRALIA
American Financial Resource 46268
Ames, Richard M. 93010
Anderson, Donald W. 92093
Anderson, Eric B. 55805
Anderson, Terry L. 99324
Andrea, John A. no address
Andrews, Guy 12538
Auld, William G. 98168
Azlin, Lawrence A. III 94530
Babb, David R. 61008
Baldwin, R. UK
Bannister, Preston L. 92714
Barrett, M.J. 66219
Bauer, August A. (CDP,CCP) 94116
Baumgarten, Henry E. 68588
Bean, Robert E. P.E. 79923
Becker, Todd 48126
Beeby, John 94030
Beidler, John 18510
Beringer, F. M. 10538
Bielak, Richard 11209
Billstrom, David 97202
Bondy, Jon 19003
Bonham, Verlene 89520
Bonham, William 89520
Book, Erwin 90066
Breeden, Thomas 22903
Brenes, Roman A. 48207
Brewster, Larry T. 65401
Brown, Geoffrey 78752
Brown, Kenneth 94939
Brown, Winsor 92647
Buddernberg, Rex A. 93943
Bull, Everett L. Jr. 91711
Bullock, John D. Jr. 60076
Bush, Randy 97420
Butler, Kenneth J. 15213
Butler, Morgan 93030
Butterfield, Bruce A. 90024
Buttrill, S.E. "Bud" Jr. 94301
Calif. Polytechnic St. Univ. 93407
Carlson, Roy E. 95051
Catlin, Charles B. 54956
Cattrall, Tom 97101
Chamberlin, Douglas 02181
Chang, Te-Chnan 94720
Chastain, Peter 94088
Clark, Randy 95061
Cohen, Dennis Robert 91206
Coleburn, Randy C. 30338
Collins, Geof 80303
Coren, Michael D. 19025
Cornelius, Barry ENGLAND
Cottingham, Bob 77098
Crandall, Keith 94720
Crawford, Albert L. 77843
Crawford, David 99502
Crouse, Gary 80236
Culham, Eugenia Canada
Dangler, Terry 26505
Daniel, Donald 93555
Davidson, Andrew H. 90232
Davis, Jack R. 37922
Davis, Thomas 98103
De Jong, Edward 94610
DeMarco, Tom 04856
deRIE, Jan 97219
Djavaheri, Morris 94404
Dodge, George H. 98033
Doglione, Arthur 85257
Dwyer, Mather B. 02138
Eastman, Steve J. 94305
Eckstrom, John 94596
Eiselen, Everett T. 95030
Eland, Dave R. no address
Emerson, Mark L 90064
Emery, Chuck CANADA
Endicott, Steve 92801
Ernst, Steve 98239
Esters, Scott 94101
Evans, John 78746
Fengler, Alf J. 06798
Feustel, David 14851
Fisher, Robert C. 33183
Fletcher, Donald 10028
Flory, Prof. David 07666
Folland, Gordon 48042
Folliott, Bruce Wm. Canada
Foote, George B. Jr. 01741
Francis, Jack 80909
Frank, Daniel M. 53711
Fujimoto, Richard M. 84112
Fuller, Jonah C., D.B.A. 94131
Fuller, Richard 29673
Gabbey, Richard C. 92631
Gerber, Philip 60048
Gianola, Maurizio 94063
Glassberg, Carl no address
Gotterer, Malcolm H. 33156
Graffunder, Kurt 55418
Gramling, Frank CER/USO 29602
Greiner, Perry L. 95126
Gross, Peter J. 98027
Gulbenkian, John 94598-2734
Hakansson, Sven 97005
Hapner, Mark W. 95126
Haray, Tuyet no address
Harvey, George W. Ph.D. 94903
Hawes, Jack 85201
Hedelman, Harold 94109
Hendricksen, Charles S. 98053
Heynneman, R.A. 95050
Hijazi, Nabil 22124
Hofkin, Mary K. 92038
Holder, John W. 48640

Hooper, John W. 92084
Hooper, Robert C. 89509
Hooper, Robert L. 94566
Huber, Frank D. 55343
Hugelman, Rodney D. 61801
Hyman, Robert A. MN 55105
Ivey, Dr. Jerry L. 83401
Iwasawa, Teiichiro no address
Jackowitz, Paul 18510
Jett, Roy Jr. 45241
Jewell, Christopher T. 95117
Johnson, Mark CANADA
Johnston, W Scott. 66601
JongeVos, Hans 97219
Karpinski, Richard 94609
Kay, Greg no address
Keeney Brian E. PAC 159 20521
Kelly, Tom CANADA
Kelly-Bootle, Stan 94941
Kerth, Norm 97229
Khalsa, Jai Gopal Singh 02568
King, K.N. 30332
Kish, Bill 73505
Kleider, Alexander M.D., P.C. 51104
Knight, Robert R., III 08648
Kohlmayr, Gerhard F. 06033
Korndorfer, Peter C. 07006
Korostoff, Edward 19151
Laity, Irving A. Jr. 35601
Lampton, Don 40214
Lanter, Sean 98072
Lawson, Bruce A. 06120
Layman, Andrew 94102
Lebsack, Don 50158
Leckebush, Johannes West Germany
Lesnick, Richard 94114
Leveque, F.H. UNITED KINGDOM
Lillo, Trotzki 20852
Lins, Chuch 92117
Littlejohn, Micahel B. 90505
Liuny, Miron 53706
Lo, Chingmin 46223
Loh, Jeff 91367
Lovejoy, Alan 94606
Lovell, Charles, A. 91505
Lustgraaf, Paul 50011
Madland, Ken 98112
Maguire, Brian 11103
Maples, Keith 35803
Marini, Giacomo 94063
Marinich, Marsha 20004
Martynenko, Walter 19438
Mathis, Darrell 94545
McArthur, Jeffrey S. 84070
McCormick, Dr. John W. 12901
McKinna, Peter no address
McLarty, Spike 97201
Meehan, Michael 35810
Meisel, David D. 14454
Michael, Jan 52333
Mickel, Andy 55414
Miller, Dennis 55106
Miller, James C. 61615
Montandon, Denis 37211
Moore, John B. CANADA
Mortlseder, Alfred no address
Muchnick, Steven S. 94043
Nagler, Robert 94303
Nassi, Isaac R. Ph. 01701
Nelson, Art 02575
Ng, Dr. Roger C. CANADA
Ng, Sauw Tet 92714
Nicholls, William H. 98373
Nolan, Chris 03062
Norem, William Charles 98125
O'Hara, Michael J. 90278
Ohran, Richard 84604
Osborne, Stan 94122
Osterlund, Russell G. Jr. 60074
PMI 97206
Page, George W. 20616
Paull, Dennis 94022
Pearson, John E. 33709
Perry, Jack 98036
Phillips, Bob 97219
Pickelmann, Paul R. 48106
Polunsky, Richard A. 77098
Powell, Edward R. Canada
Powell, Michael 94301
Pratt, Seth 94710
Prohaska, J.R. 94303
Ragan, Dr. Donal M. 85287
Ramsey, Henry W. 98188
Recard, Steven J. 14564
Reed, David A. 94063
Reid, Thomas F. Dr. 22070
Rhoads, David G. Ph.D. 19810
Richter, Adam 15208
Rivera, Romel no address
Roberts Joe, C. 75042
Roberts, Marsha J. 07974
Robertson, Kenneth 20770
Rollins, Gene Dr. 15213
Rondinaro, Mark F. 14853
Rottingen, Glenn 08520
Roubertou, Christian 29072
Russell, K.D. 83415
Russell, Robert 94707
Sallee, Gary F. 92686
Samuelson, Bruce no address
Saunders, Tom CANADA
Savit, Jeffery B. 07043
Schmitt, Jeff 21239
Schnapp, Russell L. 92126
Schubert, Dr. Karl D. 13850
Schwerdfager, Lance CANADA
Sewell, E. Wayne 75042
Shammas, Namir Clement 23060
Shaunfield, George 77083
Shebanow, Andrew 94704

Shields, Lan 27604
Shost, John 08619
Siegel, Jeremy 94306
Sincovec, Richard Dr. 80933-7150
Sloboda, Jan 10032
Smith, Erik 98020
Snodgrass, Richard Dr. 27514
Snyder, Robert W. 60064
Sofsky, Jack 30088
Soucy, Thomas J. 01905
Souter, J. B. United Kingdom
Souvestre, John R. 70006
Space, Lawrence G. 14534
Spillman, Michael D. 98021
Stander, Jeffrey M. 97333
Stanton, Michael BRAZIL
Starkey, Roy 78761
Stepka, Thomas 22043
Stowers, Irving F. 94550
Stuntz, Warren E. 39553
Sullivan, Brian Canada
Sutcliffe, Richard J. CANADA
Taimre, Ilmar AUSTRALIA
Tal, John 48075
Tal, John 48219
Tavan, Richard M. 95070
Taylor, David 94903
Thelen, Bruce 98105
Thiagarajan, Raja 47401
Trei, Peter G. 10015
Tweedy, Jack 98053
Tynor, Steve 30318
Vaughan, Bill 85028
Videki, E.R. II 85704
Von Essen, Edwin M. 93003
Wadsworth, David 17011
Wajih, A. R. M. 77272
Walker, Robert T. 92601
Wall, Darlene M. 98632
Wallace, Mark 90049
Ward, Terry A. 50613
Wardlaw, Stephen Dr. 06504
Warner, Hoyt no address
Weiser, Richard 02053
Weisert, Mike 95003
Wells, Charles 44106
Wexelblat, Alan 78759
Wick, Darrell Dr. CANADA
Wiley, Larry D. 10028
Williams, Clark Mr. 92714
Williams, Robert S. 20901
Wilson, Ann M. 02138
Wilson, John D. 92660
Wilson, Max L. 95139
Wilson, Paul CANADA
Wittie, Prof. Larry D. 11794
Woodcock, Thomas 40208
Worden, David H. 95014
Yates, Leonard 78759
Yonce, Leslie J. no address

MODUS USA Members
listed by address

Nassi, Isaac R. Ph.
41 Carter Dr.
Framingham, MA 01701
Work : 617-877-8508

Foote, George B. Jr.
Advantec
61 Judy Farm Rd.
Carlisle, MA 01741
Work : 617-369-3798

Soucy, Thomas J.
Microputer Services
13 Mildred Street
Lynn, MA 01905
Work : 617-599-8014

Weiser, Richard
Gamewell Corp.
10 Gamewell Rd.
Medway, MA 02053
Work : 617-533-4331

Wilson, Ann M.
Intermetrics, Inc.
733 Concord Ave.
Cambridge, MA 02138

Dwyer, Mather B.
Intermetrics, Inc.
733 Concord Ave.
Cambridge, MA 02138

Chamberlin, Douglas
Chamberlin Computer Services
P.O. Box 195
Wellesley, MA 02181

Khalsa, Jai Gopal Singh
MicroStrategies
Box 2278
Vineyard Haven, MA 02568
Work : (617) 645-2260

Nelson, Art
Crow Computer
State Rd.
West Tisbury, MA 02575
Work : 617-693-3007

Nolan, Chris
Digital Equipment Corp.
110 Spti Brook Rd.
(ZK2-3/N30)
Nashua, NH 03062
Work : 603-881-7013

DeMarco, Tom
Atlantic Systems Guild
P.O. Box 553
Rockport, ME 04856
Work : 212/620-4282

Kohlmayr, Gerhard F.
Mathmodel
80 Founders Road
Glastonbury, CT 06033
Work : 203-633-5659

Lawson, Bruce A.
The Hartford Graduate Center
275 Windsor Street
Hartford, CT 06120
Work : 203-548-2489

Wardlaw, Stephen Dr.
Yale Univ. school of medicine
UNHH-6026CB
20 York Street
New Haven, CT 06504
Work : 203-785-2436

Fengler, Alf J.
Philips Medical Systems, Inc.
80 Park Road
Woodbury, CT 06798

Korndorfer, Peter C.
U.S. Trust co. of New York
326 Central Ave.
West Caldwell, NJ 07006
Home : 212-420-6397

Savit, Jeffery B.
Savvy Computing
2 Edgecliff Road
Upper Montclair, NJ 07043
Work : (201) 744-4093

Flory, Prof. David
Fairleigh Dickinson Univ.
Physics
1000 River Road
Teaneck, NJ 07666
Work : (201) 692-2258
Home : 2282

Roberts, Marsha J.
95 Southgate Rd.
Murray Hill, NJ 07974
Home : 201-464-2580

Rottingen, Glenn
Stevens Institute of Tech.
56-21 Grandview Terrace
Hightstown, NJ 08520
Home : (609) 443-3132

Shost, John
Congoleum Corporation (TOC)
861 Sloan Avenue
Trenton, NJ 08619
Work : (609) 587-1000

Knight, Robert R.,III
Multi Solutions, Inc.
123 Franklin Corner Road
Suite 207
Lawrenceville, NJ 08648
Work : 609-896-4100

Trei, Peter G.
Irving Trust Corporation
Irving Trust 101B-9E
One Wall Street
New York City, NY 10015
Work : 212-815-3711

Fletcher, Donald
348 E. 92 St. #3B
New York, NY 10028
Home : 212-289-0328

Wiley, Larry D.
960 Park Avenue
New York, NY 10028
Work : 212-535-3858

Sloboda, Jan
708 W. 171st St. #55A
New York, NY 10032

Beringer, F. M.
871 Fenimore Rd.
Larchmont, NY 10538
Work : 914-834-2270

Maguire, Brian
Random House College Division
32-30 44th St.
Long Island, NY 11103
Work : 718-278-9030

Bielak, Richard
526 79th Street
Brooklyn, NY 11209
Home : 718-680-2965

Wittie, Prof. Larry D.
SUNY at Stony Brook
Computer Science
1426 Lab. Office Building
Stony Brook, NY 11794-4400
Work : (516) 246-8215

Andrews, Guy
High Tech Solutions
582 No Quaker Rd
Hyde Park, NY 12538
Work : 914-229-8649

McCormick, Dr. John W.
State University of New York
Computer Science Department
145 Redcay Hall
Plattsburgh, NY 12901
Work : (518) 564-2785

Schubert, Dr. Karl D.
P.O. Box 199
Vestal, NY 13850

Meisel, David D.
54 Westview Crescent
Geneseo, NY 14454

Space, Lawrence G.
11 Littlebrook Dr.
Pittsford, NY 14534

Recard, Steven J.
REDCOM Lsanotsyotird, Inc.
1 Redcom Center
Victor, NY 14564
Work : 716-377-0390

Feustel, David
Daflo Computer Services, Inc.
P.O. Box 6599
Ithaca, NY 14851

Rondinaro, Mark F.
Cornell University Lab of
Nuclear Studies
Wilson Lab
Ithaca, NY 14853
Work : 607-256-4882

Richter, Adam
115 Conover Road
Pittsburgh, PA 15208

Butler, Kenneth J.
Tartan Laboratories, Inc.
477 Melwood Ave.
Pittsburgh, PA 15213
Work : 412/621-2210

Rollins, Gene Dr.
Carnegie-Mellon University
Computer Science Dept.
Pittsburgh, PA 15213
Work : 412-578-7561

Wadsworth, David
316 SOmerset Dr.
Shiremanstown, PA 17011
Work : 717-763-8642

Beidler, John
University of Scranton
Computer Science Dept.
Scranton, PA 18510
Work : (717) 961-7446

Jackowitz, Paul
University of Scranton
Dept. of Math/Comp. Science
Scranton, PA 18510

Bondy, Jon
P.O. Box 148
Ardmore, PA 19003
Home : (215) 642-1057

Coren, Michael D.
1628 Arran Way
Dresher, PA 19025

Korostoff, Edward
University of Pennsylvania
6421 Overbrook Ave.
Philadelphia, PA 19151
Work : 215-477-3239

Martynenko, Walter
2750 Bittersweet Dr.
Harleysville, PA 19438
Home : 215-234-8639

Rhoads, David G. Ph.D.
David G. Rhoads Associats,Inc
5 East Dale Road
Wilmington, DE 19810
Work : 302-475-0355

Marinich, Marsha
Future Enterprises, Inc.
1331 Pennsylvania Ave.
Suite 1301
Washington, DC 20004
Work : 202-662-7676

Keeney Brian E. PAC 159
Panama-Department of State
Washington, DC 20521
Work : 52-5682

Page, George W.
Chug Capitol Heath Users Grou
R12, Box 148
Bryans Road, MD 20616

Robertson, Kenneth
8651 Greenbelt Rd., #T-2
Greenbelt, MD 20770
Home : (301) 552-3161

Lillo, Trotzki
Olympia Data Systems
12247 Tildenwood Dr.
Rockville, MD 20852
Home : 301-468-9379

Williams, Robert S.
M-Systems
208 Lexington Dr.
Silver Spring, MD 20901
Work : 301-593-0062

Schmitt, Jeff
Towson State University
1000 Litchfield Road
Baltimore, MD 21239
Work : 301-377-7875

Stepka, Thomas
7702 Lunceford Lane
Falls Church, VA 22043
Home : 703-790-8510

Reid, Thomas F. Dr.
The Mitre Corp.
1105 Criton Street
Herndon, VA 22070
Work : 703-883-6556
Home : 703-689-0091

Hijazi, Nabil
Productivity Software, Inc.
10856 Parcel Court
Oakton, VA 22124
Home : (703) 620-6242

Breeden, Thomas
Univ. of Virginia
Dept. of Biomedical Eng.
P.O. Box 3314, Univ. Station
Charlottesville, VA 22903
Work : 804-924-2668
Home : 804-973-5975

Shammas, Namir Clement
Pyramid Software, Inc.
4814 Mill Park Ct.
Glen Allen, VA 23060
Work : (804) 282-2294
Home : 804-270-3802

Dangler, Terry
PC Support
RT. 4, Box 337B
Morgantown, WV 26505
Work : 304-599-6288

Snodgrass, Richard Dr.
University of North Carolina
Department of Computer Scienc
Chapel Hill, NC 27514

Shields, Lan
Netlink Inc
3214 Spring Forest Rd.
Raleigh, NC 27604
Work : 919-878-8612

Roubertou, Christian
Michelin Tire Corp.
P.O. Box 579
Lexington, SC 29072

Gramling, Frank CER/USO
Michelin Tire Co
Box 2846
Greenville, SC 29602
Work : 803-277-9300

Fuller, Richard
Signalogic Services
Rt 4 Box 259-A
Piedmont, SC 29673

Sofsky, Jack
5415 Post Road Pass
Stone Mountain, GA 30088
Home : (404) 469-3044

Tynor, Steve
1069 McMillan St.
Atlanta, GA 30318
Work : 404-875-0252

King, K.N.
GA Institute of Technology
School of Information and
Computer Science
Atlanta, GA 30332
Work : (404)894-2595

Coleburn, Randy C.
Consultant's Choice, Inc.
8601 Dunwoody Place,
Atrium Suite 122
Atlanta, GA 30338
Work : 404-992-8430

Gotterer, Malcolm H.
6400 SW 112 St
Miami, FL 33156
Work : 305-665-5888

Fisher, Robert C.
Creative Management Services
7005 SW 138th Court
Miami, FL 33183
Work : 305-382-4353

Pearson, John E.
4075 52 St. N.
St. Pete, FL 33709
Home : 813-527-4281

Laity, Irving A. Jr.
2202 Penny Lane SE
Decatur, AL 35601
Work : 205-355-0027

Maples, Keith
2503 Monteview Dr.
Huntsville, AL 35803
Work : 205-532-1827

Meehan, Michael
3908 Shamrock Dr.
Huntsville, AL 35810
Home : 205-859-9078

Montandon, Denis
2713 Ellington Drive
Nashville, TN 37211

Davis, Jack R.
Philips Home Interactive
9117 Carlton Circle
Knoxville, TN 37922
Work : 615-588-5800
Home : 615-691-6973

Stuntz, Warren E.
2347 Sandalwood Dr.
Gautier, MS 39553

Woodcock, Thomas
University of Lousiville
1364 South First St.
Lousiville, KY 40208

Lampton, Don
Army Research Institute
4513 S. 6th
Louisville, KY 40214

Wells, Charles
Case Western Reserve Univ.
Dept. of Mathematics & Stat.
Cleveland, OH 44106
Work : (216)932-5931

Jett, Roy Jr.
9484 Southgate Dr.
Cincinnati, OH 45241
Home : 513-891-6409

Lo, Chingmin
Dept. of Computer Science,
1201 E. 38th St.
Indianaplois, IN 46223
Work : 317-923-1321 X353

American Financial Resource
3500 DePauw Blvd. Suite 1100
Indianapolis, IN 46268
Work : 317-875-7499

Thiagarajan, Raja
4423 East Trailridge Road
Bloomington, IN 47401

Folland, Gordon
General Motors
GM Proving Ground
40-ESA
Milford, MI 48042
Work : 313-685-6329

Tal, John
Rollins Medical/Dental System
360 Clausen Bldg.
23100 Providence Dr.
Southfield, MI 48075

Pickelmann, Paul R.
U. Of Michigan
Rm 3252
Box 1248
Ann Arbor, MI 48106
Work : 313-763-5224
Home : 313-995-9691

Becker, Todd
6619 Berrie
Dearborn, MI 48126

Brenes, Roman A.
Lafayette Pavilion Apts #1106
One Lafayette Plaisance
Detroit, MI 48207

Tal, John
18477 Lenore
Detroit, MI 48219

Holder, John W.
Holder & Assoiciates, Inc.
2907 Camberley Lane
Midland, MI 48640
Work : 517-636-7373

Lustgraaf, Paul
Iowa State University
32 Carver Hall
Ames, IA 50011
Work : 515-294-1832

Lebsack, Don
BEST Consultants
409 Debra Drive
Marshalltown, IA 50158

Ward, Terry A.
Academic Computing Services
2202 Oxford Lane
Cedar Falls, IA 50613
Work : 319-273-6816

Kleider, Alexander M.D., P.C.
1925 Summit
Sioux City, IA 51104
Work : 712-255-8089

Michael, Jan
RR4 Box 64
Solon, IA 52333
Work : 319-365-4631
Home : 319-643-2858

Liuny, Miron
University of Wisconsin, Madi
Computer Science Dept.
1210 West Dayton St.
Madison, WI 53706
Work : 608-262-0856

Frank, Daniel M.
Prairie Computing
1802 Keyes Ave.
Madison, WI 53711
Work : 815/455-1356
Home : 815/255-0002

Catlin, Charles B.
408 Beaulieu Road
Neenah, WI 54956
Home : 414-729-6338

Hyman, Robert A.
1754 Jefferson Ave.
St. Paul, MN 55105
Home : 612-635-5025

Miller, Dennis
Macalester College
1050 Pacific St.
St. Paul, MN 55106
Home : 612-776-5280

Huber, Frank D.
CPT Corporation
8100 Mitchell Rd.
Eden Prairie, MN 55343
Work : (612) 937-8000

Mickel, Andy
Apollo Computer, Inc.
106 SE Arthur Avenue
Minneapolis, MN 55414
Work : 612-835-4541

Graffunder, Kurt
Honeywell MN 65-2400
3660 Marshal NE
Minneapolis, MN 55418
Work : 612-782-7377

Anderson, Eric B.
930 N. 12th Ave. E.
Duluth, MN 55805

Gerber, Philip
P.O. Box 1123
Libertyville, IL 60048
Work : 312-362-8418

Snyder, Robert W.
Abbott Laboratories
Bldg. AP-9 Dept. 471
Abbott Park, IL 60064
Work : 312-937-8001
Home : 312-490-1179

Osterlund, Russell G. Jr.
Zurich Insurance Co.
842 Carriage Lane #8
Palatine, IL 60074

Bullock, John D. Jr.
Shure Brothers, Inc.
7622 Kedvale Ave.
Skokie, IL 60076
Work : 312-866-2336

Babb, David R.
Law Offices
P.O. Box 69
Belvidere, IL 61008
Work : 815-544-2720

Miller, James C.
2722 W Eugenie
Peoria, IL 61615

Hugelman, Rodney D.
University of Illinois
117 Transportation Bldg.
104 South Mathews Ave.
Urbana, IL 61801
Work : 217-333-7735

Brewster, Larry T.
1956 San Fernando Court
Rolla, MO 65401

Barrett, M.J.
American Fire Sprinkler
15521 W. 110th St.
Lenexa, KS 66219
Work : 913-541-8200

Johnston, W Scott.
Dupont
P.O. Box 481
Topeka, KS 66601
Work : 913-379-0571

Baumgarten, Henry E.
University of Nebraska-Lincol
Department of Chemistry
Lincoln, NE 68588
Work : 402-472-3301

Souvestre, John R.
J. Parnell & Assoc.
4435 Veterans Blvd. #221
Metairie, LA 70006
Work : 504-454-3734

Kish, Bill
Integral Consulting
2109 N.W. Rowena Terrace
Lawton, OK 73505
Home : (405) 248-8640

Roberts Joe, C.
1529 Meadowcrest
Garland, TX 75042
Home : 214-276-8157

Sewell, E. Wayne
3822 Hillsdale Lane
Garland, TX 75042
Home : 214-494-4761

Shaunfield, George
13315 Verbena
Houston, TX 77083

Cottingham, Bob
Cottingham Software Inc.
3701 Kirby, Suite #712
Houston, TX 77098
Work : 713-523-0033

Polunsky, Richard A.
Interface Technologies Corp.
3336 Richmond, Suite 200
Houston, TX 77098
Work : 713-523-7255

Wajih, A. R. M.
Softcad, Inc.
P.O. Box 722254
Houston, TX 77272
Work : 713-270-0184
Home : (713) 879-1797

Crawford, Albert L.
Texas A&M University
Dept. of Computer Science
Zachary Engineering Center
Bryan, TX 77843
Work : 409-845-5534

Evans, John
309 Ridgewood Road
Austin, TX 78746
Home : (512) 327-0876

Brown, Geoffrey
University of Texas
5802 Duval
Austin, TX 78752
Home : 512-450-0737

Acreman, Chris
Acreman Engineering
300 E. Huntland Dr. Ste. 215
Austin, TX 78752
Work : 512-454-1792

Wexelblat, Alan
MCC
Echelon Bldg #1, Ste. 200
9430 Research Blvd.
Austin, TX 78759
Work : 512-834-3586

Yates, Leonard
Star Value Software
12218 Scribe Dr.
Austin, TX 78759
Home : 512-837-5498

Starkey, Roy
Hitec Data Systems
P.O. Box 15404
Austin, TX 78761

Bean, Robert E. P.E.
Willis R. Bean & Associates,
P.O. Box 3536
El Paso, TX 79923
Work : 915-533-6223

Crouse, Gary
4629 South Xavier St.
Denver, CO 80236
Work : 303-866-5579

Collins, Geof
AESI
75 Manhattan Dr. Ste. 302
Boulder, CO 80303
Work : 303-449-2910

Francis, Jack
CPA Software and Systems
317 N. Bonfoy Ave.
Colorado Springs, CO 80909
Work : 303-471-2677

Sincovec, Richard Dr.
Univ. of Colorado
Computer Science Dept.
P.O. Box 7150
Colorado Springs, CO 80933-7150
Work : 303-593-3325

Ivey, Dr. Jerry L.
3209 Tipperary Lane
Idaho Falls, ID 83401
Work : 208-526-9066

Russell, K.D.
EG&G Idaho, Inc.
P.O. Box 1625 TSB
Idaho Falls, ID 83415
Work : 208-526-9592

McArthur, Jeffrey S.
Stargazer Enterprises Inc.
76 Cottage Ave.
Sandy, UT 84070
Work : 801-255-0392

Fujimoto, Richard M.
Univesity of Utah
3160 Merrill Engineering Bldg
Computer Science Dept.
Salt Lake City, UT 84112

Ohran, Richard
Modula Corporation
950 North University Avenue
Provo, UT 84604
Work : 801/375-7400

Vaughan, Bill
Bill Vaughan & Co.
3033 E. Sierra St.
Phoenix, AZ 85028

Hawes, Jack
630 W. McLellan Road
Mesa, AZ 85201
Work : 602-964-3853
Home : 602-964-1024

Doglione, Arthur
Logical Models
2828 N. Seventy-Fourth Pl.
Scottsdale, AZ 85257
Work : 602-949-8155

Ragan, Dr. Donal M.
Arizona State University
Dept. of Geology
Tempe, AZ 85287
Work : (602) 965-6939

Videki, E.R. II
IBM Corporation
8570 N. Mulberry Drive
Tucson, AZ 85704

Hooper, Robert C.
630 Harbin Lane
Reno, NV 89509

Bonham, Verlene
Stride Micro
P.O. Box 30016
Reno, NV 89520
Work : 702-322-6868

Bonham, William
Stride Micro
P.O. Box 30016
Reno, NV 89520
Work : 702-322-6868

Butterfield, Bruce A.
UCLA/Instructional Devel.
10962 Le Conte Ave.
Los Angeles, CA 90024
Work : 213-825-7771

Wallace, Mark
Information Engineering
11693 San Vicente Bl. Ste.168
Los Angeles, CA 90049

Emerson, Mark L
Hughes Aircraft
11809 Gateway Bl.
Los Angeles, CA 90064
Work : 213-615-7382
Home : 213-477-7916

Book, Erwin
3169 Colby Avenue
Los Angeles, CA 90066
Home : 213-397-2385

Davidson, Andrew H.
New Line 7
P.O. Box 1211
Culver City, CA 90232
Work : 213-277-7217

O'Hara, Michael J.
Hughes Aircraft Company
2404 Harriman Lane
Redondo Beach, CA 90278
Work : 213-371-2365

Littlejohn, Micahel B.
Software Engineering Ass. Inc
23864 Hawthorne Blvd.Ste #200
Torrance, CA 90505

Cohen, Dennis Robert
215 N. Kenwood, #102
Glendale, CA 91206

Loh, Jeff
6301 Glade ASve. K203
Woodland Hills, CA 91367
Home : 818-883-0110

Lovell, Charles, A.
Lovell Systems
347 N. Lima St.
Burbank, CA 91505
Work : 818-845-3882

Bull, Everett L. Jr.
Pomona College
Department of Mathematics
Claremont, CA 91711
Work : 607-256-7443

Hofkin, Mary K.
Instar Solutions
P.O. Box 2975
La Jolla, CA 92038
Work : (619) 274-5288

Hooper, John W.
1202 Warmlands Avenue
Vista, CA 92084

Anderson, Donald W.
UC, San Diego
B-028 Office of Academic Comp
La Jolla, CA 92093
Work : 619-452-2628

Lins, Chuch
Chuck Lins Sofware
5063 Clairemont Mesa Blvd. #6
San Diego, CA 92117

Schnapp, Russell L.
7671 Northrop Place
San Diego, CA 92126
Home : 619-578-7014

Walker, Robert T.
P.O. Box 622
Atwood, CA 92601

Gabbey, Richard C.
2248 Bedford Dr.
Fullerton, CA 92631
Work : 714-993-5463

Brown, Winsor
15332 Fieldston Lane
Huntington Beach, CA 92647
Home : (714) 891-6043

Wilson, John D.
Optimal Data Corp.
4940 Campus Dr.
Newport Beach, CA 92660
Work : 714-833-3300

Sallee, Gary F.
Sallee Software
19912 Fernglen Drive
Yorba Linda, CA 92686
Work : 714-970-2864

Bannister, Preston L.
FileNet Corporation
10 Thunder Run #34C
Irvine, CA 92714
Work : (714)786-4735

Ng, Sauw Tet
13 Woodfern
Irvine, CA 92714

Williams, Clark Mr.
MCDonnell Douglas
Computer Systems Company
1562 Reynolds Ave.
Irvine, CA 92714

Endicott, Steve
Endicott Consulting Inc.
1408 James Way
Anaheim, CA 92801
Work : 714-535-2416

Von Essen, Edwin M.
Telos Computing
6038 Fremont St.
Ventura, CA 93003
Work : 805-644-3546

Ames, Richard M.
U.S. Navy
2704 E. Via Del Nogal
Camarillo, CA 93010-2234
Work : 805-482-6513

Butler, Morgan
1931 North H. St. #70
Oxnard, CA 93030
Work : 805-388-2801

Calif. Polytechnic State Uni
Computer Systems Laboratory:
Department of Computer Scienc
San Luis Obispo, CA 93407
Work : 805-546-2147

Daniel, Donald
2005 Dibb
Ridgecrest, CA 93555
Home : 619-446-2186

Buddernberg, Rex A.
Naval Postgraduate School
SMC #1309
Monterey, CA 93943
Work : 408-646-9608

Paull, Dennis
Paull Associates Inc.
146 Main Street #205
Los Altos, CA 94022
Work : 415-948-9275

Beeby, John
4 Corte Nueva
Milbrae, CA 94030

Muchnick, Steven S.
Sun Microsystems, Inc.
2550 Garcia Ave. MS 5-40
Mountain View, CA 94043

Marini, Giacomo
Logitech, Inc
805 Veterans Boulevard
Redwood City, CA 94063

Reed, David A.
Charles Evans & Associates
301 Chesapeake Dr.
Redwood City, CA 94063
Work : (415)369-4567

Gianola, Maurizio
Logitech, Inc
805 Veterans Bld
Redwood City, CA 94063

Chastain, Peter
ESL, Inc.
495 Java Dr.
P.O. Box 3510
Sunnyvale, CA 94088

Esters, Scott
Keystone Software
P.O. Box 11325
San Francisco, CA 94101

Layman, Andrew
BREAKTHROUGH SOFTWARE
537 Jones Street, #8613
San Francisco, CA 94102
Work : (415)-898-1919
Home : (415) 386-1614

Hedelman, Harold
110 Clarendon
San Francisco, CA 94109
Work : 415-868-0290

Lesnick, Richard
P.O. Box 14633
San Francisco, CA 94114
Home : 415-863-0361

Bauer, August A. (CDP,CCP)
Software Engineering Conslt.
P.O. Box 16266
San Francisco, CA 94116
Work : 415-469-8504

Osborne, Stan
SFSU Computer Science Dept.
1678 48Th Avenue
San Francisco, CA 94122
Work : 415-661-6868

Fuller, Jonah C., D.B.A.
P.O. Box 31610
San Francisco, CA 94131
Work : (415) 953-2776

Buttrill, S.E. "Bud" Jr.
1417 Parkinson
Palo Alto, CA 94301
Work : 415-572-1401
Home : 415-321-8338

Powell, Michael
Digital Equipment Corporation
Western ReaBsearch Laboratory
100 Hamilton Avenue
Palo Alto, CA 94301
Work : 415-853-6620

Nagler, Robert
1077B Tanland Dr.
Palo Alto, CA 94303
Work : (415) 322-0547

Prohaska, J.R.
Teknowledge
P.O. Box 51862
Palo Alto, CA 94303
Work : 619-327-6600

Eastman, Steve J.
Resarch Libraries Group, Inc.
Jordan Quad., ACACIA
Stanford, CA 94305
Work : 415-329-3528

Siegel, Jeremy
4183 Park Blvd.
Palo Alto, CA 94306
Home : 213-858-2827

Djavaheri, Morris
Djavaheri Bros
P.O. Box 4759
Foster City, CA 94404-0759
Work : 415-341-1767

Azlin, Lawrence A. III
Mt. Davidson Software
5508 MacDonald Ave.
El Cerrito, CA 94530

Mathis, Darrell
Xerox Corp.
26250 Industrial Blvd.
Hayward, CA 94545-2992
Work : 415-887-4743

Stowers, Irving F.
Search Technology
5275 Diane Lane
Livermore, CA 94550
Home : 415-455-9173

Hooper, Robert L.
Hooper Consultants
4534 Eull Court
Pleasanton, CA 94566
Work : 415-485-0175

Eckstrom, John
Eckstrom Consulting Services
P.O. Box 5121
Walnut Creek, CA 94596
Work : 415-944-0401

Gulbenkian, John
Gulbenkian Assoc.
431 Rock Oak Road
Walnut Creek, CA 94598-2734
Work : (415) 932-4250

Lovejoy, Alan
Senior Systems Analyst:
473 Hanover Ave
Oakland, CA 94606
Work : 415-622-0135
Home : 415-763-0654

Karpinski, Richard
Modula Assured Quality Sftwr.
6521 Raymond Street
Oakland, CA 94609
Work : (415) 666-4529
Home : 658-3797

De Jong, Edward
Beyond Workds
285 Jayne Ave.
Oakland, CA 94610
Work : 415-836-0965

Shebanow, Andrew
HyperSoft
2124 Kittredge St. #174
Berkeley, CA 94704

Russell, Robert
1119 Shattuck Ave.
Berkeley, CA 94707
Home : 415-524-5953

Pratt, Seth
Ivory Consulting Corporation
2550 Ninth St. Suite 201
Berkeley, CA 94710
Work : 415-486-0690

Crandall, Keith
University of California
Dept. of Civil Engineering
Berkeley, CA 94720
Work : 415-642-9066

Chang, Te-Chnan
University of California
Dept of Civil Engineering
Berkeley, CA 94720
Work : 415-642-9066

Harvey, George W. Ph.D.
Pan Pacific Institute
Internationa Health & Nutriti
P.O Box 6249
San Rafael, CA 94903
Home : 472-7346

Taylor, David
MicroPro
33 San Pablo Avenue
San Rafael, CA 94903
Work : 415-499-4098

Brown, Kenneth
Beyond Words
1800 Lincoln Village Circle
Apt. 2323
Larkspur, CA 94939
Work : 415-461-7550

Kelly-Bootle, Stan
Unix Review (Cont. Editor)
25 Parkwood Ave.
Mill Valley, CA 94941
Work : 415-381-5027

Weisert, Mike
Borland Intl.
243 Spreckles Dr. #A
Scotts Valley, CA 95003
Work : 408-438-8400 x 421

Worden, David H.
10264 Parkwood Dr. #4
Cupertino, CA 95014

Eiselen, Everett T.
IBM
101 Milmar Way
Los Gatos, CA 95030

Heynneman, R.A.
Digital Equipment Corp
444 Saratoga Ave., 33B
Santa Clara, CA 95050

Carlson, Roy E.
Software Mechanics
Suite #1
660 Harvard Ave.
Santa Clara, CA 95051
Work : 408-247-8618

Clark, Randy
P.O. Box 7739
Santa Cruz, CA 95061
Home : 408-458-3391

Tavan, Richard M.
20297 Hickory Hill Way
Saratoga, CA 95070
Home : (408) 867-5797

Jewell, Christopher T.
Pacific Systems Group
3900 Moorpark Ave. #37
San Jose, CA 95117
Work : 408-241-3064

Greiner, Perry L.
Greiner Corporation
1550 The Alameda
Suite 207
San Jose, CA 95126
Work : (408) 279-2717

Hapner, Mark W.
846 St. Elizabeth Dr.
San Jose, CA 95126
Home : (408) 370-1256

Wilson, Max L.
IA Systems, Inc.
112 Winsten Ct.
San Jose, CA 95139
Work : 408-281-2157

Hakansson, Sven
Sern Electronics, Inc.
6700 SW 105th #311
Beaverton, OR 97005
Work : 503-646-2411

Cattrall, Tom
7600 Seawood Rd. SE
Amity, OR 97101
Home : 503-835-1613

McLarty, Spike
1522 SW 18th St.
Portland, OR 97201
Work : 503-226-7569

Billstrom, David
Billstrom & Neuhauser
4025 S.E. 32nd Avenue
Portland, OR 97202
Home : 503-232-2071

PMI
4536 SE 50th
Portland, OR 97206
Work : 503-293-7706

Phillips, Bob
Oregon Software, Inc.
6915 SW. Macadam
Portland, OR 97219
Work : 503-245-2202

JongeVos, Hans
Oregon Software
6915 MacAdam Ave.
Portland, OR 97219
Work : 503-225-2202

deRIE, Jan
Oregon Software
6915 S.W. MacAdam Ave.
Portland, OR 97219
Work : 503-245-2202

Kerth, Norm
11521 NW Laidlaw Rd
Portland, OR 97229

Stander, Jeffrey M.
Maritime Infosystems Ltd.
6660 Reservoir Rd.
Corvallis, OR 97333
Work : 503-929-2552

Bush, Randy
Pacific Systems Group
601 South 12th Court
Coos Bay, OR 97420
Work : 503/267-6970
Home : 503/572-5391

Smith, Erik
ScenicSoft, Inc.
12314 Scenic Drive
Edmonds, WA 98020
Work : (206)742-6677

Spillman, Michael D.
Data I/O
23710 48th Ave. S.E.
Bothell, WA 98021
Work : 206-483-1277

Gross, Peter J.
Pacific Rim Real-Time Systems
P.O. Box 1021
Issaguah, WA 98027
Work : 206-392-8941

Dodge, George H.
13701 115th Ave. N.E.
Kirkland, WA 98033

Perry, Jack
Inventory Auditors
5723 198th S.W.
Lynnwood, WA 98036
Work : 206-775-4674

Hendricksen, Charles S.
Cedar Software, Inc.
P.O. Box 327
19607 Redmond Road
Redmond, WA 98053
Work : 206-868-5202

Tweedy, Jack
Cedar Software, Inc.
5405 - 258th Ave. NE
Redmond, WA 98053

Lanter, Sean
Cedar Software, Inc.
P.O. Box 798
Woodinville, WA 98072

Abrahamsen, Barry
2301 North 65th St.
Seattle, WA 98103
Home : 206-524-1929

Davis, Thomas
4026 Evanston N.
Seattle, WA 98103

Thelen, Bruce
Maritime Infosystems
4556 University Way N.E.
Seattle, WA 98105
Work : 206-547-4100

Madland, Ken
1913 - 14th Ave. East
Seattle, WA 98112
Work : 206-323-5604

Norem, William Charles
11304 Riviera Place N.E.
Seattle, WA 98125

Auld, William G.
2118 S. 107 #1
Seattle, WA 98168
Home : 206-246-9143

Ramsey, Henry W.
Unimac, Inc.
P.O. Box 98533
Seattle, WA 98188

Ernst, Steve
Computer Aided Estimating Inc
P.O. Box 98
Coupeville, WA 98239
Work : 206-678-6923

Nicholls, William H.
BGW Systems Inc.
Suite 200
16714 Meridian South
Puyallup, WA 98373

Wall, Darlene M.
Lower Columbia College
DPCS Department
1600 Maple Street
Longview, WA 98632
Work : 206-577-2379

Anderson, Terry L.
Walla Walla College
Dept. of Computer Science
College Place, WA 99324
Work : (509) 527-2276

Crawford, David
8840 Cordell Cir. Unit C-2
Anchorage, AK 99502
Work : 907-562-2203
Home : 907-243-1164

Taimre, Ilmar
Ilmar Taimre Pty Ltd
P.O. Box 365
Glen Waverley
Victoria, 3150
AUSTRALIA
Work : 61 (03) 233-6573

The University of Adelaide
Dept. of Computer Science
GPO BOX 498
Adelaide
South Australia, 5001
AUSTRALIA
Work : 08-228-5833

Stanton, Michael
Dept.DE Informtica, PUC-RJ
Rua Marques De SAO
VICENTE 225
Rio De Janeiro, 22453
BRAZIL
Work : 021-274-4449

Culham, Eugenia
Kwantlen College
P.O. Box 9030
Surrey
British Columbia, V3T 5H8
Canada
Work : 604-588-4411

Sullivan, Brian
Techware Systems Corp.
2325 Burrard St.
Vancouver
British Columbia, V6J 3J2
Canada
Work : 604-734-5294

Powell, Edward R.
Entropy Limited
Box 34126, Station D
Vancouver, BC V6J 4M1
Canada
Work : (604) 228-0011

Wick, Darrell Dr.
Camosun College
3100 Foul Bay Rd.
Victoria, BC V8P 4X8
CANADA
Work : 604-592-1281

Sutcliffe, Richard J.
Trinity Western University
7600 Glover Road
Langley, BC V3A 4R9
CANADA
Home : (604) 888-7511

Schwerdfager, Lance
308-1300 Richmond Road
Ottawa
Ontario, K1K1S7
CANADA
Home : 613-749-5181

Kelly, Tom
Human Computing Resources
267 Glebeholme Blvd
Toronto Ontario , M4J1T1
CANADA
Work : 416-922-1932

Saunders, Tom
29 Indian Trail
Toronto
Ontario, M6K 1Z8
CANADA
Home : 416-767-9967

Wilson, Paul
Westronic Inc.
154-11810 MacLeod Tr. SE
Calgary, Alberta, T2J 2V8
CANADA
Home : 403-271-2378

Johnson, Mark
11428 - 77 Avenue
Edmonton
Alberta, T6G0L8
CANADA
Home : 403-438-0555

Folliott, Bruce Wm.
Alberta Research Council
Atmos.Sci.Dept.,7th Fl.Ter.Pl
4445 Calgary Trail South,
Edmonton, AL T6H 5R7
Canada
Work : 403-438-0555 x-311

Ng, Dr. Roger C.
Sherritt Gordon Mines Ltd.
Analytical Services Dept.
Fort Saskatchewan
Alberta, T8L 2P2
CANADA
Work : 403-998-6644

Emery, Chuck
TDI Computer systems Ltd.
66 Twenty-third Street
Toronto
Ontario, M8V 3N2
CANADA
Work : (416) 259-5081

Moore, John B.
University of Waterloo
Dept. of Management Science
Waterloo, Ontario, N2L 3G1
CANADA

Cornelius, Barry
Dept. of Computer Science
University of Durham
Durham, DH1 3LE
ENGLAND
Work : Durham 64791 x 792

Baldwin, R.
British Oceanics Equip. Hire
Unit 14
Wellheads Crescent Trading Es
Dyce, Aberdeen, AB2 0GA
UK
Work : 0229-29006

Souter, J. B.
British Standards Institute
P.O. Box 375
Linford Wood
Milton Keynes, MK14 6LO
United Kingdom
Work : 0908-315555

Leveque, F.H.
Department of Computing
Imperial College
180 Queens Gate
London, SW7 2BZ
UNITED KINGDOM

Leckebush, Johannes
Redauteur Computer Personlich
Hans-Pinsel Str. 2
Haar Bei Munchen, D-8013
West Germany
Work : 089-4613-237

Prior issues partial content: (these are still available)

Modula-2 News # 0 October 1984

Revisions ... to Modula-2, Wirth
Spec. of Standard Modules, Hoppe
Modula-2 bibliography, Brown
Modus Membership list
Modula-2 Implementation Questionaire

Modula-2 News # 1 January 1985

Letter to Editor, Layman
Letter to Editor, Bush
Gleaves' Modula-2 text, DeMarco
MODUS Paris meeting, Blunsdon
Report of M2 Working Group, Souter
Library Rationale by Randy Bush
Library Definition Modules
Library Documentation by Jon Bondy
Validation of Modula-2 Impl, Siegel

MODUS Quarterly # 2 April 1985

Letter on Library, Anderson
Letter to Editor, Emerson
Comments on Modula-2, Emerson
Opaque Types, French & Mitchell
Dynamic Instantiation, Sumner
Linking Modula-2, Symons
Library Comments, Peterson
Modula Compilers, Smith
Coding War Games, DeMarco
M2, Alt. to C, Djavaheri/Osborne

MODUS Quarterly # 3 July 1985

Letter re opaque types, Endicott
Letter on language issues, Hoffmann
Modula-2 in "Real Time", Barrow
RajaInOut: safer I/O, Thiagarajan
Contentious Problems, Cornelius
Expressions in Modula-2, Wichmann
Scope Problems: Modules, Cornelius
Corrections to compiler list

MODUS Quarterly # 4 November 1985

MODUS Meeting Report by Bob Peterson
A Writer's View of Conf, Sam'l Bassett
Concerns of a Programmer, Dennis Cohen
Mods to Standard Lib, Nagler & Siegel
Std Lib and Ext'n to Modula-2, Odersky
Std Lib for Unix by Morris Djavaheri
Impl of Std Lib for PC's, Verhulst
M-2 Compilation and Beyond, Foster
Modula-2 Processes, Roger Henry

MODUS Quarterly # 5 February 1986

Export Module Identifier, Cornelius
Multi-dimensional open arrays, Wirth
DIV, MOD, /, and REM, Niklaus Wirth
Multi-dimensional open arrays, Steiger
NULL-terminated strings, Poulsen
ISO Ballot Results re BSI Modula-2
Draft BSI I/O Library, Eisenbach
Portable Language Rationale, Hopper +
ETH-Z Modula-2 for Macintosh, Jewell
NewStudio: for Macintosh, Davidson +

MODUS Administrators supply single copies at $5 US or 12 Swiss Francs.

Hints for contributors:

Send CAMERA READY copy to the editor (dot matrix copy is usually unacceptable). Machine readable copy is preferred. Present facilities permit printing from electronic mail and floppy disks (Sage, IBM PC, Macintosh) using troff, Script and PostScript formatting systems. Working papers and notes about work in progress are encouraged. MODUS Quarterly is not perfect, it is current.

Please indicate that publication of your submission is permitted. Correspondence not for publication should be PROMINENTLY so marked.

Richard Karpinski, Editor
6521 Raymond Street
Oakland, CA 94609
(415) 476-4529 (12-7 pm)
(415) 658-3797 (ans. mach.)

TeleMail M2News or RKarpinski
BITNET dick@ucsfcca
Compuserve 70215,1277
InterNet dick@cca.ucsf.edu
UUCP ...ucbvax!ucsfcgl!cca.ucsf!dick

Modula-2 Users' Association
MEMBERSHIP APPLICATION

Name : _____

Affiliation : _____

Address : _____

Address : _____

State : _____ **Postal Code:** _____ **Country:** _____

Phone : (___)____-_____ **Electronic Addr :** _____

Option: ___ Do NOT print my phone number in any rosters
or: ___ Print ONLY my name and country in any rosters
or: ___ Do NOT release my name on mailing lists

Application as: **New Member** ___ or **Renewal** ___

Implementation(s) used : _____

** Membership fee per year (20 USD or 45 SFr) **
Members of the US group who are outside of North America, add $10.00.

In North and South America, please send check or money order (drawn in US dollars) payable to Modula-2 Users' Association at:

Modula-2 Users' Association
P.O. Box 51778
Palo Alto, California 94303
United States of America

Otherwise, please send check or bank transfer (in Swiss Francs) payable to Modula-2 Users' Association at:

Aline Sigrist
MODUS Secretary
ERDIS SA
P.O.Box 35
CH-1800 Vevey 2

The Modula-2 Users' Association is a forum for all parties interested in the Modula-2 Language to meet each other and exchange ideas. The primary means of communication is through the Newsletter which is published four times a year. Membership is for an academic year, and you will receive all newsletters for the full year in which you join. Mid-year applications receive that year's back issues. Modula-2 is a new and developing language; this organization provides implementors and serious users a means to discuss and keep informed about the standardization effort, while discussing implementation ideas and peculiarities. For the recreational user, there is information on the status of the language, along with examples and ideas for programming in Modula-2. For everyone, there is information on current implementations and the other resources available for obtaining information on the language.

Modula-2 Users' Association
MEMBERSHIP APPLICATION

Name : _____

Affiliation : _____

Address : _____

Address : _____

State : _____ **Postal Code:** _____ **Country:** _____

Phone : (___)____-_____ **Electronic Addr :** _____

Option: ___ Do NOT print my phone number in any rosters
or: ___ Print ONLY my name and country in any rosters
or: ___ Do NOT release my name on mailing lists

Application as: **New Member** ___ or **Renewal** ___

Implementation(s) used : _____

** Membership fee per year (20 USD or 45 SFr) **
Members of the US group who are outside of North America, add $10.00.

In North and South America, please send check or money order (drawn in US dollars) payable to Modula-2 Users' Association at:

Modula-2 Users' Association
P.O. Box 51778
Palo Alto, California 94303
United States of America

Otherwise, please send check or bank transfer (in Swiss Francs) payable to Modula-2 Users' Association at:

Aline Sigrist
MODUS Secretary
ERDIS SA
P.O.Box 35
CH-1800 Vevey 2

The Modula-2 Users' Association is a forum for all parties interested in the Modula-2 Language to meet each other and exchange ideas. The primary means of communication is through the Newsletter which is published four times a year. Membership is for an academic year, and you will receive all newsletters for the full year in which you join. Mid-year applications receive that year's back issues. Modula-2 is a new and developing language; this organization provides implementors and serious users a means to discuss and keep informed about the standardization effort, while discussing implementation ideas and peculiarities. For the recreational user, there is information on the status of the language, along with examples and ideas for programming in Modula-2. For everyone, there is information on current implementations and the other resources available for obtaining information on the language.

MODUS
c/o Pacific Systems
PO Box 51778
Palo Alto, CA 94303
USA

Aline Sigrist
ERDIS SA
CH-1800 Vevey 2
(021) 52 61 71

Return Postage Guaranteed