UO-LISP V1

CPU Family
Z80

Operating System
TRSDOS

PRODUCT PACKAGES

The following lists the product packages available. Check the included price list for current prices.

UO-LISP V1
includes
(A) Interpreter
(B) Compiler
(C) Structure Editor
(D) Utilities
(E) RLISP
(F) Reference Manual
(G) Tutorial Guide (CP/M based)

Little Meta V1
includes (H) Little Meta

PRODUCT DESCRIPTION

UO-LISP Version 1 (V1) relies upon the TRSDOS operating system running on the TRS-80 type computer from Radio Shack (TANDY). UO-LISP VI utilizes standard TRSDOS system calls to implement its operating system interface.

The following paragraphs describe the UO-LISP VI product parts. Review the Requirements, Important Notes, and Product Packages sections before ordering your copy of UO-LISP V1.

(A) INTERPRETER

The UO-LISP VI Interpreter supports the following data types: dotted-pairs, identifiers, integers, strings, and function pointers. It also supports GLOBAL, alist, and local binding mechanisms. Basic functions support manipulation of lists, property lists, and identifiers; other functions support function definition and progs, as well as arithmetic, Boolean, predicate, conditional, and I/O operations. The Interpreter also supports incore compilation, precompiled libraries, garbage collection, error handling, and low level TRS-80 graphics. The Interpreter does not support floating point and session freeze.

(B) COMPILER

The compiler is a LISP program that converts LISP functions into assembly code (LAP) and then assembles the LAP instructions into absolute Z80 machine instructions. The compiled code can be saved on disk in relocatable binary form called fast load files. The LAP instruction set is available to the user and allows implementation of fast assembly language routines, and functions to perform I/O to nonstandard devices. If space and speed are critical, the compiler's optimizing section can be enabled. The optimizer reduces the amount of space required for compiled functions.

(C) STRUCTURE EDITOR

The LISP structure editor is a set of LISP functions that enable the user to: 1) enter and test functions while remaining in the LISP system, 2) save and restore sets of functions and commands on disk, 3) modify functions and test them before saving.

(D) UTILITIES

This collection of utilities include the following: 1) the **LAPP** (LISP Assembly Pretty Print) module displays compiled code in a readable format with hexadecimal output, and assembly code with appropriate spacing, 2) the **PRETTY PRINT** package formats LISP S-Expressions, 3) the **TRACE** package monitors the entrance to and exit from functions.

(E) RLISP

If you find programming in bare bones LISP with all its parentheses a little troublesome, the RLISP high level language, provides a palatable syntax without sacrificing the power or flexibility of LISP. The language features WHILE, REPEAT, AND FOR loops, IF... THEN... ELSE clauses, and infix operators for the common arithmetic and list operations.

(F) REFERENCE MANUAL

The TRS-80 system manual, over 100 pages, documents the Interpreter, the LISP structure editor, the compiler and optimizer, the high level language RLISP, and numerous other support facilities like the TRACE package.

(G) TUTORIAL SUPPORT

Included is a LISP tutorial guide. Note, however, that this tutorial guide was written for the CP/M based UO-LISP V2. This means that the tutorial guide will refer to functions, packages, and software source code available only with the CP/M based version V2. We include this tutorial because you will find that 90% of the guide is useful for learning LISP and AI programming techniques. We expect to properly retrofit this tutorial to UO-LISP V1 in the near future.

(H) LITTLE META

The Little META Translator Writing System is a LISP tool that permits you to specify the syntax of a programming language, how it is to be interpreted, and how to convert it into LISP or assembly code. Little META is a great tool for the Computer Science Professional interested in writing compilers, translators, syntax scanners, preprocessors, or front ends to expert and intelligent systems. It requires the TRS-80 based UO-LISP V1 Interpreter for operation. The package includes a 20 page appendix to reference manual.

REQUIREMENTS

TRS-80 MODEL I

Hardware: Minimum 32K RAM, and two disk drives.
Software: Model I TRSDOS 2.3
Distribution Format: Model I

TRS-80 MODEL III

Hardware: Minimum 32K RAM, and two disk drives.
Software: Model III TRSDOS 1.3
Distribution Format: Model III

TRS-80 MODEL 4 & 4P

Hardware: Minimum 32K RAM, and two disk drives.
Software: Model III TRSDOS 1.3
Distribution Format: Model III

IMPORTANT NOTES

1) TRSDOS is not distributed with UO-LISP V1. If you do not have Model III TRSDOS 1.3 for your Model 4 you may obtain it from your local Radio Shack dealer.

2) So far non-TRSDOS systems (like NEWDOS, DOS+, & LDOS) that are TRSDOS-like in nature, available on Model's I & III, have shown to be compatible with UO-LISP V1.

3) UO-LISP V1 does not run under Model 4 TRSDOS 6.anything.

NORTHWEST COMPUTER ALGORITHMS
P.O. Box 90995
Long Beach, California 90809
(213) 426-1893

CPU Family
Z80

UO-LISP V2

Operating System
CP/M

PRODUCT PACKAGES

The following lists the product packages available. Check the included price list for current prices.

UO-LISP V2

includes
- (A) Interpreter
- (B) Compiler
- (C) Editors
- (D) Extended Numbers
- (E) Fast Load Libraries
- (F) RLISP
- (G) LISPTEX
- (H) Reference Manual

UO-LISP V2 with TUTORIAL SUPPORT

includes
- (A) Interpreter
- (B) Compiler
- (C) Editors
- (D) Extended Numbers
- (E) Fast Load Libraries
- (F) RLISP
- (G) LISPTEX
- (H) Reference Manual
- (K) Tutorial Support

Little Meta V2

includes (I) Little Meta

Learn Lisp V2

includes (J) Learn Lisp System

PRODUCT DESCRIPTION

UO-LISP Version 2 (V2) relies on the CP/M operating system running on Z80 type computers. UO-LISP V2 utilizes standard CP/M system calls to implement its operating system interface.

The following paragraphs describe the product parts. Review the Requirements, Important Notes, and Product Packages sections before ordering your copy of UO-LISP V2.

(A) INTERPRETER

The UO-LISP Interpreter supports the following data types: integers, dotted-pairs, code-pointers, strings, vectors, and identifiers. It also supports GLOBAL, FLUID (alist), and local binding mechanisms. Basic functions support manipulation of lists, property lists, and identifiers; other functions support function definitions and progs as well as arithmetic, Boolean, predicate, conditional, and I/O operations. All functions are redefinable. The evaluator supports EXPR, FEXPR, and MACRO type functions. The Interpreter also supports in-core compilation, precompiled libraries, garbage collection, back trace on error, session freeze and restart, up to six open files, and user installable I/O routines.

(B) COMPILER

The compiler is a LISP program that converts LISP functions into assembly code (LAP) and then assembles the LAP instructions into machine code. Compiled code is stored in core or saved on disk in relocatable binary form called fast load files. The LAP instruction set is available to the user and allows the implementation of fast assembly language routines, and functions to perform I/O to nonstandard devices. If space is critical, the compiler's optimizing section can be enabled reducing the amount of space required for compiled functions. Where speed is essential the optimizer can be tuned to produce fast code. Also included with the compiler is a Z80 assembler, that implements all of the Z80 instructions and supports intermixing LISP and Z80 assembly source codes.

(C) EDITORS

There are three editors: character, structure, and screen. The **CHARACTER EDITOR** is used in conjunction with the structure editor. It allows character level editing of LISP expressions and atoms. The **STRUCTURE EDITOR** allows the user to manipulate function definitions, GLOBAL and FLUID variables, identifiers, and property lists. The multi-window **LSED SCREEN EDITOR** allows the user to edit files of LISP source code with the file image displayed in the top window. The user may point at any LISP expression and have it evaluated with the results appearing in the bottom window. Two files may be edited simultaneously. Cursor controls and edit commands are control-key sequences entered from the keyboard. LSED comes configured with a standard set of commands to which the user may add, change or bind to alternate keys. This editor is capable of editing text as well as LISP source code.

(D) EXTENDED NUMBERS

The extended numbers fast load library contains a BIGNUM package and a FIXED point precision package. **BIGNUM** permits operations with integers up to 2000 digits in length. **FIXED** is designed with financial calculations in mind and may be used where floating point numbers would normally be used.

(E) FAST LOAD LIBRARIES

The fast load libraries extend the total number of UO-LISP functions to over 400. **MACROS** provide a number of useful LISP constructs such as the FOR, WHILE, REPEAT, DEFSTRUCT, CASE, BACK QUOTE, and READ macros. **PRETTY PRINT** displays LISP expressions indented to show different levels of structure. **TERSE PRINT** abbreviates large S-expresseion by displaying only top level information in deep structures and only the first few elements of long lists. The depth and length can be altered at any time. **TRACE** permits you to watch the evaluation of both interpreted and compiled fuctions. The arguments of a function are displayed before the routine is entered, and the value of the function is displayed before it is exited. TRACE also supports break points and tracing of SETQ type variable assignments. **LDDT**, the LISP Debugger, is a

hexadecimal debugging program. This package can be used to examine and change the contents of any location in storage from LISP. The **LAPP** (LAP Pretty Print) module displays compiled code in a readable format with hexadecimal output, and assembly code with appropriate spacing. **PROFILE** provides automatic execution time profiling by counting the number of times functions are called. **CREF** supplies the user with a cross-reference of all the functions and variables used in a file. The **HISTORY,** saving read loop is a top level LISP and RLISP reader that is useful during the debugging and editing phase of a program. The HLOOP reader maintains a queue of the top level commands entered and allows these to be reexecuted, used as part of other commands, and listed. **GLOBALS** provide access to a number of global quantities that are normally hidden from the LISP user. The **SORT** package sorts items in main storage or on disk. **HUNKS** is a very fast array structure for storing constants. If the function you are calling is not available in your present environment, the **AUTO LOADER** will load the required package. The **SWAPPER** package supports the swapping of fast load files, allowing the user to write programs as large as their disk capacity. The **UTILS** package contains a number of useful low level operations that were not considered to be a necessary part of the Interpreter. It contains functions for performing fast arithmetic and logical operations, a random number generator, and **PEEK** and **POKE.** The **CPM** package contains the basic CP/M operating system interface functions. The **XCPM** package provides communication with the CP/M operating system BDOS disk directory functions. **TERMINAL,** the number of terminals with differing protocols available to the open market place is growing. LISP programs that require special terminal actions (like the screen editor LSED) use a terminal driver as an interface. We provide the LISP source code to a collection of commonly used protocols so you can create your own terminal driver. Please note that this collection of sources is not yet fixed. Sometimes included are Televideo TV920, Lear Siegler ADM22, ADM3A/Kaypro, and H19.

(F) RLISP

If you find programing in bare bones LISP with all its parentheses a little troublesome, the RLISP high level language provides a palatable syntax without sacrificing the power of LISP. The language features WHILE, REPEAT and FOR loops, IF... THEN... ELSE clauses, and infix operators for the common arithmetic and list operations.

(G) LISPTEX

LISPTEX is an extremely powerful document formatter that accepts text and formatting commands. The command set can be extended by calling and defining LISP functions within the text or in library files.

(H) REFERENCE MANUAL

The CP/M based reference manual, over 350 pages, is a comprehensive document with hundreds of examples plus appendices on new facilities. The manual covers all the base functions and facilities of the Interpreter, the editors, the optimizing compiler and assembler, the high level language RLISP, and the numerous fast load libraries described above.

(I) LITTLE META

The Little Meta Translator Writing System is a LISP tool that permits you to specify the syntax of a programming language, how it is to be interpreted, and how to convert it into LISP or assembly code. META is a great tool for the computer Science Professional interested in writing compilers, translators, syntax scanners, preprocessors, or front ends to expert and intelligent systems. It requires the CP/M based UO-LISP V2 for operation. The package includes a 20 page appendix to the main reference manual.

(J) LEARN LISP

A special UO-LISP environment; includes Interpreter, structure editor, Pretty Print, TRACE utility, on-line help, a 90 page tutorial, and a 117 page reference manual. Learn LISP is a special system for those learning LISP for the first time or for those who have some LISP knowledge but really haven't taken the time to cover all the basic LISP functions and programming techniques. The tutorial section is intended to help the beginning LISP user learn LISP fundamentals and good AI programming techniques. Learn LISP. Includes a pedogogical rule - based expert system. NOTE: This package does not allow expansion to use the compiler and fast load libraries.

(K) LISP TUTORIAL SUPPORT

This package is sold only as an addition to the main system UO-LISP V2. It provides the beginner with the Learn LISP system described above. The learners reference manual, however, is replaced by the main reference manual.

REQUIREMENTS

Hardware: Z80 CPU, with 64K RAM and two disk drives.
Operating System: Standard CP/M 2.2 or higher.

IMPORTANT NOTES

1) UO-LISP V2 derives its speed and power from using most of the Z80 resources, that is the RST's and X & Y registers. Nonstandard CP/M's may use these same resources in a manner conflicting with UO-LISP's usage. Check the latest table of configurations for supported and not-supported systems before placing your order for UO-LISP V2.

2) Several common terminal drivers are shipped in source code form with UO-LISP V2 in support of LSED. If we do not support your terminal type you may have to write your own terminal driver to run LSED.

NORTHWEST COMPUTER ALGORITHMS
P.O. Box 90995
Long Beach, California 90809
(213) 426-1893

CPU Family
8086

UO-LISP V3

Operating System
PC-DOS or
MS-DOS

PRODUCT PACKAGES

The following lists the product packages available. Check the included price list for current prices.

UO-LISP V3

includes
- (A) Interpreter
- (B) Compiler
- (C) Editors
- (D) Extended Numbers
- (E) Fast Load Libraries
- (F) RLISP
- (G) LISPTEX
- (H) Reference Manual

UO-LISP with TUTORIAL SUPPORT

includes
- (A) Interpreter
- (B) Compiler
- (C) Editors
- (D) Extended Numbers
- (E) Fast Load Libraries
- (F) RLISP
- (G) LISPTEX
- (H) Reference Manual
- (K) Tutorial Support

Little Meta V3

includes (I) Little Meta

Learn LISP V3

includes (J) Learn Lisp System

PRODUCT DESCRIPTION

UO-LISP Version 3 (V3) relies upon the PC-DOS or MS-DOS operating system running on an IBM-PC or compatible computer. UO-LISP V3 utilizes standard PC-DOS or MS-DOS system calls to implement its operating system interface.

The following paragraphs describe the product parts. Review the Requirements, Important Notes, and Product Packages sections before ordering your copy of UO-LISP V3.

(A) INTERPRETER

The UO-LISP V3 Interpreter supports the following data types: integers, dotted-pairs, code-pointers, strings, vectors, and identifiers. It also supports GLOBAL, FLUID, and local binding mechanisms. Basic functions support manipulation of lists, property lists, and identifiers; other functions support function definitions and progs, as well as arithmetic, Boolean, predicate, conditional, and I/O operations. All functions are redefinable. The evaluator supports EXPR, FEXPR, and MACRO type functions. The Interpreter also supports in-core compilation, precompiled libraries, garbage collection, back trace on error, session freeze and restart, up to six open files, and user installable I/O routines.

(B) COMPILER

The compiler is a LISP program that converts LISP functions into assembly code (LAP) and then assembles the LAP instructions into machine code. Compiled code is stored in core or saved on disk in relocatable binary form called a fast load file. The LAP instruction set is available to the user and allows implementation of fast assembly language routines, and functions to perform I/O to nonstandard devices. If space and speed are critical, the compiler's optimizing section can be enabled reducing the amount of space required for compiled functions. An 8086 assembler will be available in a future update.

(C) EDITORS

There are three editors: character, structure, and screen. The **CHARACTER EDITOR** is used in conjunction with the structure editor. It allows character level editing of LISP expressions and atoms. The **STRUCTURE EDITOR** allows the user to manipulate function definitions, GLOBAL and FLUID variables, identifiers, and property lists. The multi-window **LSED SCREEN EDITOR** allows the user to edit a file of LISP source code with the file image being displayed in the top window. The user may point at any LISP expression and have it evaluated with the results appearing in the bottom window. Two files may be edited simultaneously.

Cursor controls and edit commands are control-key sequences entered from the keyboard. LSED comes configured with a standard set of commands to which the user may add, change, or bind to alternate keys. This editor is capable of editing text as well as LISP.

(D) EXTENDED NUMBERS

The extended numbers fast load library contains a BIGNUM package and a FIXED point precision package. **BIGNUM** permits operations with integers of up to 2000 digits in length. **FIXED** is designed with financial calculations in mind and may be used where floating point numbers would normally be used.

(E) FAST LOAD LIBRARIES

The fast load libraries extend the total number of UO-LISP functions to over 400. **MACROS** provide a number of useful LISP constructs such as the FOR, WHILE, REPAEAT, DEFSTRUCT, CASE, DEFMACRO, BACK QUOTE, and READ macros. **PRETTY PRINT** displays LISP expressions indented to show different levels of structure. **TERSE PRINT** abbreviates large S-expressions by displaying only top level information in deep structures and only the first few elements of long lists. The depth and length can be altered at any time. **TRACE** permits you to watch the evaluation of both interpreted and compiled functions. The arguments of a function are displayed before the routine is entered, and the value of the function is displayed before it is exited. Trace also supports break points and tracing of SETQ type assignments. The **LAPP** (LAP Pretty Print) module displays compiled code in a readable format.

PROFILE provides automatic execution time profiling by counting the number of times functions are called. **CREF** supplies the user with a cross-reference of all the functions and variables used in a file. The **HISTORY,** saving read loop is a top level LISP and RLISP reader that is useful during the debugging and editing phase of a program. The HLOOP reader maintains a queue of top level commands and allows these to be reexecuted, used as part of other commands, and listed. The **SORT** package sorts items in main storage or on disk. **TERMINAL,** the number of terminals with differing protocols available to the open market place is growing. LISP programs that require special terminal actions (like the screen editor LSED) use a terminal driver as an interface. We provide the LISP source code for a standard IBM-PC terminal driver. If your computer is not 100% PC compatible you may have to modify the termainal driver for LSED to work properly.

(F) RLISP

If you find programming in bare bones LISP with all its parentheses a little troublesome, the RLISP high level language provides a palatable syntax without sacrificing the power of LISP, The language features WHILE, REPEAT and FOR loops, IF . . . THEN . . . ELSE clauses, and infix operators for the common arithmetic and list operations.

(G) LISPTEX

LISPTEX is an extremely powerful document formatter that accepts text and formatting commands. The comand set can be extended by calling and defining LISP functions within the text or in library files.

(H) REFERENCE MANUAL

The PC-DOS and MS-DOS based reference manual, over 350 pages, is a comprehensive document with hundreds of examples plus appendices on new facilities. The manual covers all the base functions and facilities of the Interpreter, the editors, the optimizing compiler, the high level language RLISP, and the numerous fast load libraries described above.

(I) LITTLE META

The Little Meta Translator Writing System is a LISP tool that permits you to specify the syntax of a programming language, how it is to be interpreted, and how to convert it into LISP or assembly code. META is a great tool for the Computer Science Professional interested in writing compilers, translators, syntax scanners, preprocessors, or front ends to expert and intelligent systems. It requires the PC-DOS or MS-DOS based UO-LISP V3 for operation. The package includes a 20 page appendix to the main reference manual.

(J) LEARN LISP

A special UO-LISP environment; includes Interpreter, structure editor, Pretty Print, TRACE utility, on-line help, a 90 page tutorial, and 117 page reference manual. Learn Lisp is a special system for those learning LISP for the first time or for those who have some LISP knowledge but really haven't taken the time to cover all the basic LISP functions and programming techniques. The tutorial section is intended to help the beginning LISP user learn LISP fundamentals and good AI programming techniques. Learn LISP includes a pedagogical rule-based expert system. NOTE: This package does not include the compiler and fast load libraries described above.

(K) LISP TUTORIAL SUPPORT

This package is sold only as an addition to the main system UO-LISP V3. It provides the beginner with the Learn LISP system described above. The learners reference manual, however, is replaced by the main reference manual.

REQUIREMENTS

Hardware: 8086/8088 type CPU (IBM-PC or compatible) with minimum of 128K RAM, two floppy disk drives (320K) or one floppy disk and a hard disk.

Operating System: PC-DOS 1.1 or higher, or MS-DOS 2.0 or higher.

IMPORTANT NOTES

We provide the LISP source code for a standard IBM-PC terminal driver. If your computer is not 100% PC compatible you may have to modify the terminal driver for LSED to work properly.

NORTHWEST COMPUTER ALGORITHMS

P.O. Box 90995

Long Beach, California 90809

(213) 426-1893

NORTHWEST COMPUTER ALGORITHMS

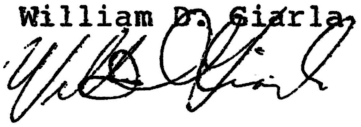
WE'RE STILL ALIVE

I bet you thought we had died and were flushed down the bit bucket. As you can see we are still alive and LISPing away. So why the hugh delay on the newsletters?

WELL.... we've been working on a very important project. We took on the computer algebra system called REDUCE. We put on the blinders and produced 27,000 lines of LISP. REDUCE now runs on the 8085 family of computers in less than 640k. More about REDUCE in our next issue of the newsletter.

So what about the newsletters???? Worry not... You will receive at least 4 newsletters for your paid subscription, plus a few more because we are so late.

William D. Clarke

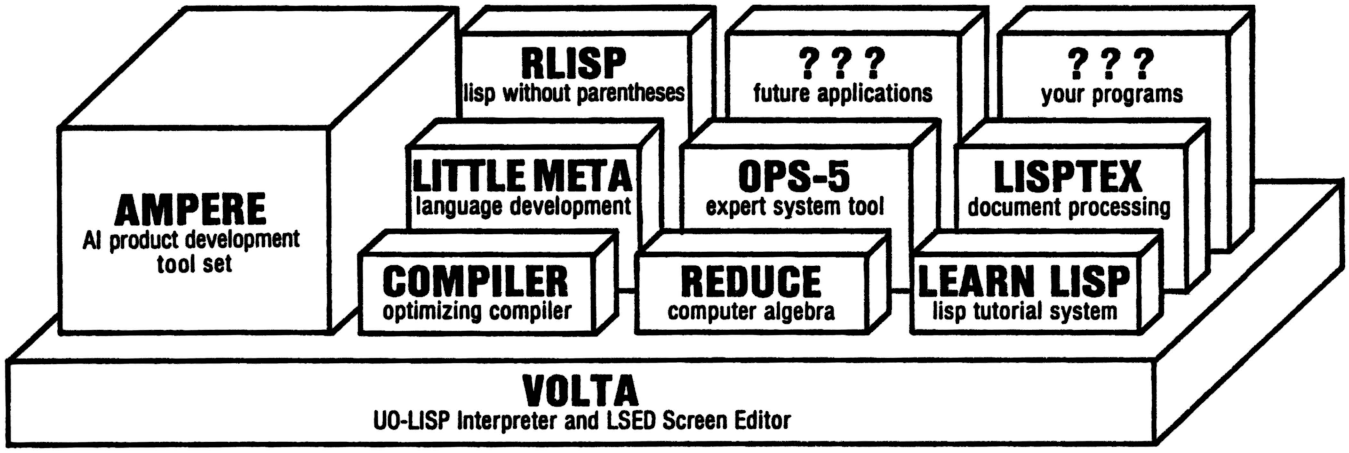


P.O. Box 1747 • Novato, California 94948 • (415) 897-1302

REDUCE

An interactive software system designed for general mathematical computations of interest to physicists, mathematicians and engineers. 27,000 lines of LISP in 640K. The most powerful computer algebra system for Hewlett Packard VECTRA PCs, IBM PCs and compatible computers. Requires 512K, hard disk recommended.

Differentiation
Integration
Factorization
Matrix Algebra
Linear Equations
High Energy Physics



UO-LISP

It's NOT COMMON, it's FAST!!!

INDUSTRIAL
UO-LISP is strong. Take for example the computer algebra system REDUCE. Until the powers of UO-LISP were applied to REDUCE it ran only on main frame and powerful mini computers like the CRAY X-MP and the SUN Workstation. UO-LISP now runs REDUCE on the 8086 family of micro computers by compiling its 27,000 lines of LISP source into less than 640K of efficient native code. Run REDUCE on a technical PC such as the **HP VECTRA** and you have a **PC BREAK-THROUGH** for scientists, engineers and mathematicians.

LOWEST PRICED LISP FOR YOUR PC
VOLTA is the electric base plate for the UO-LISP family of products. You get a Standard LISP interpreter with a full set of data types: integers, big numbers, floating point, strings, atoms, pairs and vectors; a full featured screen editor modeled after EMACS with parenthesis balancing, debugging tools, more than 200 Standard LISP functions and system utilities all documented in a compact function reference guide.

DESIGNER'S WORKSTATION
AMPERE is a complete LISP development workstation for the HOT designer creating large scale AI systems for micros. AMPERE includes VOLTA, COMPILER, LITTLE META, LISPTEX, RLISP, program development tools and a 300 page programmers reference manual.

MODULAR FAMILY OF PRODUCTS
Plug LISPTEX into VOLTA and process text. Plug LEARN LISP into VOLTA and learn lisp while exploring a rule based expert system and other AI techniques. Plug the COMPILER into VOLTA and create fast run time libraries.

REQUIREMENTS
IBM-PC or compatible with a minimum of 256K and PC-DOS 2.0 or higher. CP/M also supported.

CALL OR WRITE TODAY and we'll send you brochures, a product price list with bundle discounts, timing tests and order forms. Product prices range from $50.00 to $500.00. Order the UO-LISP configuration that best meets your needs.

NORTHWEST COMPUTER ALGORITHMS
P.O. Box 1747, Novato, California 94948
(415) 897-1302

REDUCE EXAMPLES

The following REDUCE examples are typeset with comments in the Helvetical Bold font, user typed input in the Times Roman font and algebra system responses in the Courier font and slightly indented. User typed input may be in upper or lower case.

Integer arithmetic is exact

```
A := (for i := 1:50 product i);
```

```
   A := 30414093201713378043612608166064768844377641568960512000000000000
```

and so is rational arithmetic.

```
A/2^60;
```

```
   2161051298920808821692148751911927380176169433593 75/8192
```

Floating point numbers can also be as accurate as you wish.

```
on bigfloat, numval;

precision 25;
```

```
   25
```

This looks like an integer ...

```
e**(pi*sqrt(163));
```

```
   26253 74126 40768 744.0
```

... but higher precision shows the truth.

```
precision 35;
```

```
   35
```

```
e**(pi*sqrt(163));
```

```
   26253 74126 40768 743.9 99999 99999 92500 7
```

```
off bigfloat;
```

Polynomials can be factored.

```
on factor;

(x^20-1);
```

```
   8    6    4    2          4    3    2          4    3    2
  (X  - X  + X  - X  + 1)*(X  + X  + X  + X + 1)*(X  - X  + X  - X + 1)*

   2
  (X  + 1)*(X + 1)*(X - 1)
```

```
off factor;
```

Differentiation, Integration, Factorization

REDUCE 3.2

Differential and Integral calculus examples.

```
int(1/(x^8-1),x);
```

$$\begin{aligned}&(SQRT(2)*LOG(- SQRT(2)*X + X^2 + 1) - SQRT(2)*LOG(SQRT(2)*X + X^2 + 1)\\&\quad - 2*SQRT(2)*ATAN((- SQRT(2) + 2*X)/SQRT(2)) - 2*SQRT(2)*ATAN((SQRT(\\&\quad 2) + 2*X)/SQRT(2)) + 2*LOG(X - 1) - 2*LOG(X + 1) - 4*ATAN(X))/16\end{aligned}$$

The answer is available in the REDUCE work space variable called ws.

```
df(ws,x);
```

$$1/(X^8 - 1)$$

Other examples ... ...

```
df( (x*(cos(log(x)) + sin(log(x))))/2, x);
```

$$\cos(\log(x))$$

```
int(cos(log(x)),x);
```

$$(X*(COS(LOG(X)) + SIN(LOG(X))))/2$$

```
int(1/cos(x),x);
```

$$- LOG(TAN(X/2) - 1) + LOG(TAN(X/2) + 1)$$

```
int((a+b*x)^n,x);
```

$$((A+B*X)^N*(A+B*X))/(B*(N+1))$$

```
df( (a+b*x)^n * (a+b*x) / b*(n+1), x);
```

$$(A+B*X)^N$$

... again using the variable ws.

```
df(exp exp exp exp x,x);
```

$$E^{(E^{(E^{(E^X)})} + E^{(E^X)} + E^X + X)}$$

```
int(ws,x);
```

$$E^{(E^{(E^{(E^X)})})}$$

Inversion of Hilbert matrices.

```
mata := mat( (1/(x-1), 1/(x-2), 1/(x-3)),
             (1/(x-2), 1/(x-3), 1/(x-4)),
             (1/(x-3), 1/(x-4), 1/(x-5)) )$

1/mata;
```

$$MAT(1,1) := (X^5 - 11*X^4 + 47*X^3 - 97*X^2 + 96*X - 36)/4$$

$$MAT(1,2) := (- X^5 + 13*X^4 - 65*X^3 + 155*X^2 - 174*X + 72)/2$$

$$MAT(1,3) := (X^5 - 15*X^4 + 85*X^3 - 225*X^2 + 274*X - 120)/4$$

$$MAT(2,1) := (- X^5 + 13*X^4 - 65*X^3 + 155*X^2 - 174*X + 72)/2$$

$$MAT(2,2) := X^5 - 15*X^4 + 88*X^3 - 252*X^2 + 352*X - 192$$

$$MAT(2,3) := (- X^5 + 17*X^4 - 113*X^3 + 367*X^2 - 582*X + 360)/2$$

$$MAT(3,1) := (X^5 - 15*X^4 + 85*X^3 - 225*X^2 + 274*X - 120)/4$$

$$MAT(3,2) := (- X^5 + 17*X^4 - 113*X^3 + 367*X^2 - 582*X + 360)/2$$

$$MAT(3,3) := (X^5 - 19*X^4 + 143*X^3 - 533*X^2 + 984*X - 720)/4$$

1/mata might be useful in a FORTRAN program.

```
on fort;

ws;

      MAT(1,1)=(X**5-11.*X**4+47.*X**3-97.*X**2+96.*X-36.)/
     . 4.
      MAT(1,2)=(-X**5+13.*X**4-65.*X**3+155.*X**2-174.*X+
     . 72.)/2.
      MAT(1,3)=(X**5-15.*X**4+85.*X**3-225.*X**2+274.*X-
     . 120.)/4.
      MAT(2,1)=(-X**5+13.*X**4-65.*X**3+155.*X**2-174.*X+
     . 72.)/2.
      MAT(2,2)=X**5-15.*X**4+88.*X**3-252.*X**2+352.*X-192.
      MAT(2,3)=(-X**5+17.*X**4-113.*X**3+367.*X**2-582.*X+
     . 360.)/2.
      MAT(3,1)=(X**5-15.*X**4+85.*X**3-225.*X**2+274.*X-
     . 120.)/4.
      MAT(3,2)=(-X**5+17.*X**4-113.*X**3+367.*X**2-582.*X+
     . 360.)/2.
      MAT(3,3)=(X**5-19.*X**4+143.*X**3-533.*X**2+984.*X-
     . 720.)/4.

off fort;
```

Matrix Inverses, Determinants, Transposition

REDUCE 3.2

REDUCE knows about sine, cosine, exponentials and logarithms, and many other functions. If that is not enough users can define their own functions.

```
operator myfn;

myfn(x)+y;
```

```
   MYFN(X) + Y
```

You can give rules for differentiation for example

```
for all x let df(myfn(x),x) = exp(x^2);
```

and now use it to simplify a second derivative,

```
df(myfn(x)^3,x,2);
```

$$6*E^{(X^2)}*MYFN(X)*(E^{(X^2)} + MYFN(X)*X)$$

and some integrals.

```
int(x*myfn(x),x);
```

$$(- E^{(X^2)}*X + 2*MYFN(X)*X^2 + MYFN(X))/4$$

REDUCE can handle matrices as well.

```
matrix rx(3,3), ry(3,3), mata(3,3);
```

Product of rotation matrices

```
rx := mat( ( 1,       0,       0),
           ( 0,    cos(th), sin(th)),
           ( 0,   -sin(th), cos(th)) )$

ry := mat( ( cos(phi),   0, -sin(phi)),
           (        0,   1,        0),
           ( sin(phi),   0,  cos(phi)) )$

rx*ry;
```

```
   MAT(1,1) := COS(PHI)
   MAT(1,2) := 0
   MAT(1,3) :=  - SIN(PHI)
   MAT(2,1) := SIN(TH)*SIN(PHI)
   MAT(2,2) := COS(TH)
   MAT(2,3) := COS(PHI)*SIN(TH)
   MAT(3,1) := COS(TH)*SIN(PHI)
   MAT(3,2) :=  - SIN(TH)
   MAT(3,3) := COS(TH)*COS(PHI)
```