A high-level programming environment suitable for artificial intelligence research, program development and delivery systems.

**TEK COMMON LISP PROGRAMMING LANGUAGE**

- A full Common Lisp implementation
- Optimized for the Tektronix 4400 Series of Artificial Intelligence Systems
- Offers a rich set of features for rapid prototyping of AI concepts
- Includes a resident run-time compiler for highly optimized machine code
- Extensive debugging information available for compiled code

Common Lisp was conceived by a large committee of academicians and AI researchers as a language that would incorporate the very best features of other Lisp dialects. Tek Common Lisp is a full implementation of this language (as specified in "Common LISP. The Language" by Guy Steele), configured to run on the Tektronix 4400 Series of Artificial Intelligence Systems. As such, it offers a much richer set of data types and more complex program structures than other Lisp dialects currently in use.

**A New Standard**

Common Lisp is considered by many artificial intelligence experts to be a new industry standard for AI programming environments. The reasons for this consensus are reflected in the general parameters established for the language:

**COMMONALITY:** Common Lisp focuses the features of several different implementations of Lisp into a common dialect.

**PORTABILITY:** Applications written in Common Lisp are easily ported to any Common Lisp implementation.

**EXPRESSIVENESS:** Common Lisp is a very rich language that employs the most valuable constructs from other Lisp dialects.

**EFFICIENCY:** Common Lisp has features designed to facilitate the production of fast, high-quality compiled code.

**COMPATIBILITY:** Since Common Lisp is derived from a number of popular dialects, code from other Lisp dialects should readily map into Common Lisp.

**Tek Common Lisp Features**

Tek Common Lisp has been specifically optimized and enhanced for performance on the Tektronix 4400 Series. It provides AI researchers and software developers with a personal Lisp programming environment previously available only on dedicated Lisp machines:

- Powerful optimizing compiler with built-in debugging features
- Lexically scoped interpreter and compiler
- Full featured package system for symbol name differentiation
- Rich collection of numerical primitives and built-in functions
- Built-in garbage collector and dynamic storage management
- Complete implementation of arrays, vectors and strings
- Flexible and full-featured interactive user interface
- Powerful and flexible debugging aids
- Powerful facilities for structures and macros
- Lexical closures
- User-extensible data type facility
- Built-in user-extensible parser and hash-table facility

Copyright ©1985, Tektronix, Inc. All rights reserved.
Tek Common Lisp offers the pro-
and the rapid prototyping of concepts.

Tek Common Lisp goes beyond the specifications of the language to provide:
- On-line documentation
- User-definable error handler
- Powerful and robust foreign function interfaces to C and FORTRAN programs*
- Full-featured built-in Flavors system for object-oriented programming*

Data Types and Type Declarations
An extensive set of data types allows the development of sophisticated structures for complex AI systems. In addition, it is possible to declare specific types for variables, which aids debugging and promotes efficient compiled code. The user has the choice of working without type declarations for flexibility in prototyping, or using type declarations to optimize code. Among the types available in Tek Common Lisp are:

- Arrays
- Bit-vectors
- Hash-tables
- Lists
- Floating point numbers
- Complex arithmetic characters

Dynamic Storage
Tek Common Lisp offers the programmer the advantages of dynamically allocated storage and automatic garbage collection. Data structures can grow as necessary and are reclaimed when no longer in use. This encourages experimentation and an open-ended programming style—an essential element for AI development and the rapid prototyping of concepts.

Lexical Scoping
Lexical scoping means that entities (such as variable bindings) may be referenced only within the scope of the program portions in which they are established. Both the interpreter and compiler of Tek Common Lisp are lexically scoped. In many earlier Lisp dialects, there were inconsistent semantics between compiled code, which used lexical scoping, and interpreted code which used dynamic scoping. This consistency between the interpreter and compiler is seen as the solution to a long-standing problem.

Lexical Closure
A closure is a type of functional object that is used to save the values of variables between different invocations of the closure, and to protect this data from inadvertently being overwritten by other Lisp functions. It is useful for implementing advanced control structures and data access mechanisms.

Packages
One problem with earlier Lisps was the use of a single name space for all symbols. This often caused accidental name collisions in large Lisp systems with modules written by different programmers. Tek Common Lisp uses the package feature to solve this difficulty. Packages are used to organize symbols into different name spaces, promoting program modularity and avoiding name conflicts.

Debugging Information
Often, compiled Lisp code provides very little debugging information to the user. Correcting errors in compiled code can be tedious, requiring the user to load source code in order to discover the problem. Tek Common Lisp offers extensive debugging information for both interpreted and compiled code. Users can pinpoint errors while running compiled code, resulting in efficient and productive use of time and resources.

Ordering Information
4400P33 Opt. 04 Tek Common Lisp Programming Language for 4404
4400P33 Opt. 05 Tek Common Lisp Programming Language for 4405
4400P33 Opt. 06 Tek Common Lisp Programming Language for 4406

Customer Software Services
Tek Common Lisp includes a one-year software update service which can be renewed annually.

*Available January 1986