CALCLAB: An Interactive Environment for Scientific Numerical Problem Solving

Yusuf N. Mauladad

AI TR87-59 August, 1987

This research was supported by a grant from the Army Research Program under grant number ARO DAAG29-84-K-0060. Computer equipment was furnished by a grant from Xerox Corporation.
CALCLAB: AN INTERACTIVE ENVIRONMENT FOR SCIENTIFIC
NUMERICAL PROBLEM SOLVING

by

YUSUF NAZIR MAULADAD, B. S. E. E.

THESIS

Presented to the Faculty of the Graduate School
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCES
THE UNIVERSITY OF TEXAS AT AUSTIN
August 1987
Acknowledgements

I would like to thank Dr. Gordon Novak, my advisor, for providing the guidance and advice without which this work would not have been possible. I would also like to express my appreciation to Dr. Vincent Hwang for serving as my second reader. My thanks as well to my friends and associates from whom I have learned so much. Most of all, however, I would like to thank my parents to whom I am grateful for encouragement and support of all my endeavors.

This research was sponsored by the U. S. Army Research Office under contract DAAG29-84-K-0060. Computer equipment was furnished by a grant from Xerox Corporation.

Yusuf Nazir Mauladad

The University of Texas at Austin
July, 1987
Abstract

CalcLab is an interactive environment for solving numerical problems in areas such as as physics. Objects or situations are described by sets of variables. Physical principles and constraints are expressed as mathematical equations. Variables and equations can be connected to form models to solve numerical problems.

A primary goal of this research has been to design and implement a visual environment that is easy and intuitive to use. The user interface requires no programming language expertise. Several features have been incorporated to aid the user by automatically performing simple tasks. Another goal has been to enable the user to build a library of solution models that can be reused or modified. Such models can also be combined with others to solve more complex problems.

CalcLab is implemented in GLISP and runs on Xerox 1108 AI Workstations. It has been used to solve physics problems of the kind typically encountered by freshman physics students.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>iv</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Survey of Related Work</td>
<td>12</td>
</tr>
<tr>
<td>3. Program Overview</td>
<td>17</td>
</tr>
<tr>
<td>4. Examples</td>
<td>23</td>
</tr>
<tr>
<td>5. User Interface</td>
<td>42</td>
</tr>
<tr>
<td>6. Knowledge Representation</td>
<td>49</td>
</tr>
<tr>
<td>7. Automatic Operations</td>
<td>52</td>
</tr>
<tr>
<td>9. Future Extensions</td>
<td>81</td>
</tr>
<tr>
<td>Appendix A: Additional Example</td>
<td>86</td>
</tr>
<tr>
<td>Appendix B: CalcLab GLISP Declarations</td>
<td>96</td>
</tr>
<tr>
<td>Appendix C: CalcLab GLISP Functions</td>
<td>103</td>
</tr>
<tr>
<td>Bibliography</td>
<td>140</td>
</tr>
<tr>
<td>Vita</td>
<td>144</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Introductory Example

CalcLab is an interactive environment for scientific numerical problem solving. This section is a brief introduces CalcLab briefly and demonstrates how the program may be used to model and solve numerical problems.

![Diagram](image)

**Figure 1.1 Banked Track Problem**

Figure 1.1 shows the CalcLab display while modeling the problem of an automobile traveling on a banked circular track of radius 400 m. Given the weight of the automobile, and its velocity, the
problem requires the computation of the minimum required banking of the track. Should the values of the weight, velocity, or radius be changed, the value of the banking will be automatically updated to reflect the changes.

Suppose now that it is wished to find the the maximum velocity at which another automobile, that weighs 25000 N, could travel on the same banked track. The model is extended, and the display after the second automobile is added is shown in Figure 1.2.

![Figure 1.2 Addition of Second Automobile](image)

Finally, it is decided to investigate what happens if the velocity of the first automobile is changed. The model is automatically updated to reflect any change. The result of such a change is shown in Figure 1.3.
Figure 1.3 Propagation of Change

Many operations are performed automatically by CalcLab. These include deriving the formulas used to compute values, propagating the effects of changes, and establishing conversions between units so that values may be input and displayed in the units of the user’s choice. Computations are set up with minimum user interaction. In addition all interaction with the program is done through a menu-driven graphical interface that makes the use of CalcLab intuitive and easy to use. No programming language expertise is required of the user. Furthermore, CalcLab provides the ability to rapidly prototype models. Models can be saved for later use, modified, or combined with other models to solve more complex problems.
1.2 An Environment for Scientific Numerical Problem Solving

Most numerical problems in scientific areas, such as physics, are solved by using physical principles to compute the values of variable quantities that describe a physical object or situation. These principles, typically expressed as mathematical equations, represent the relationships between various variables, and frequently also involve one or more constants. For example, the gravitational attraction between two objects can be expressed by the equation

\[ F = G \frac{m_1 m_2}{d^2} \]

where \( F \) is the force of attraction between two masses, \( m_1 \) and \( m_2 \), which are a distance \( d \) apart. \( G \) is the universal gravitational constant. Given the masses of two objects, and the distance between them, this equation can be used to compute the force of attraction between them.

Consider, however, the case where instead of the masses of the two objects, only their weights on the surface of the earth are known. The weight of an object on earth \( W \), is related to its mass \( m \) by the equation

\[ W = m g \]

where \( g \), a constant, is the acceleration due to gravity on the surface of the earth. This equation can be used to compute the masses of the objects. Once the two masses have been computed they can then be used to compute the force of attraction between the two objects. The problem can be conceptually visualized as the interconnection between objects and physical principles shown in Figure 1.4.
Figure 1.4 Interconnection between Objects and Physical Principles

Suppose now that there is a new problem to be solved. The force of attraction $F$, between two objects a distance $d$ apart, and the weight of one of the objects, $W_1$, are known. The problem requires the computation of the weight of the other object, $W_2$. $W_2$ can be computed using the same equations, or the same interconnection network.

The two equations and the various variables as interconnected in Figure 1. can be considered a model for solving the numerical problems outlined above. CalcLab is an environment intended for building such models to solve problems that can be adequately modeled using mathematical equations.

An equation such as $W = m g$ can be interpreted in several ways: $W = m g$ as the formula for computing the weight of an object given its mass, or $m = W / g$ as the formula for computing the mass of an object given its weight. When an equation is added to the system, CalcLab automatically performs the necessary algebraic manipulation to
derive a formula for each variable in the equation in terms of the other variables and constants used in the equation.

In the model considered above, there are two masses \( m_1 \) and \( m_2 \). These are two separate "instances" of the variable mass. CalcLab allows the user to make more than one instance of any variable, and the value of an instance may be computed using instances of the other variables in a formula.

To compute the value of an instance of a variable, the user may direct the program to construct a computation. If there is more than one applicable formula, the program presents the available options from which the user may select one. The computation is then constructed by the program. If the computation uses other instances, whenever the value of one of those instances changes, the instance computed by the computation is automatically updated. Thus the user can experiment by changing the values of instances to reflect various scenarios, and immediately see the effects of the changes. Since an instance used in a computation may itself be computed by another computation, computations can be cascaded to build increasingly complex models.

An important element in scientific numerical problem solving is the use of units of measurement. Mass, for example, can be measured in kg, grams, or some other units. Units of measurement are significant, and it is important to be able to convert between them. When a unit of measurement is specified the user is prompted for the dimensions of the unit. These dimensions are in terms of the fundamental quantities of mass, length, time and charge. Using the dimensions of units CalcLab automatically establishes conversions between units whenever possible. Since it is possible that the user may wish to use conversions that CalcLab is unable to establish automatically, the program also allows the user to
define conversions between units. Values for instances may therefore be input or displayed in the units of the user's choice, provided such units are already defined.

In the problems considered above, there are two objects, each described by an instance of mass and an instance of weight. It is convenient to be able to manipulate all the instances pertaining to a single object or situation collectively. CalcLab allows the user to do this by grouping instances together in frames. Equations may represent principles that apply universally, or they may represent the relationship particular to an object or situation. Therefore, CalcLab allows equations to be restricted to particular frames. Thus, if the user had defined the frame point-mass, containing an instance of mass and an instance of weight, then the model for the problem above would contain two point-mass frames, and the equation $W = mg$ would be restricted to point-mass frames.

CalcLab frames are similar in nature to the canonical object frames described by Novak [Novak3]. These canonical object frames are idealizations or abstractions of certain features of some objects. Physical principles are defined in terms of these frames which approximate the behavior of certain aspects of real objects or situations.

CalcLab frames might also be considered interpretations or “views” of objects or situations [Minsky]. Since there may be a number of ways to view some object or situation, more than one frame might be defined to model such an object or situation. For example, a planet might be viewed as a point-mass in one problem, or as a sphere in another. Therefore, a point-mass frame could be used to represent a planet in one problem, and a sphere frame used to represent a planet in another. It is also possible to use both types of frames in the same model.
Using CalcLab, the user specifies the outlines of a model in terms of equations, frames, constants, and units of measurement. But as the knowledge base available to the program grows, the user is required to specify fewer details. This is because any models created can be retained, and later modified, reused, or used as parts of other models. For example, if we already had a model for solving the problems outlined above, and at some later time it was desired to compute the velocity of one mass orbiting another mass in space, we could build a new model from the existing one by adding the equation

\[
F_c = \frac{m v^2}{r}
\]

where \( F_c \) is the centripetal force acting on a mass \( m \) traveling in a circle of radius \( r \) with velocity \( v \). An instance of \( F_c \) can then be constrained to an instance of \( F \) and an instance of \( d \) can be constrained to an instance of \( r \). An instance of \( v \) can then be computed. In this case only a new equation needed to be added to the existing model to construct a model for the newer more complex problem.

After several models have been created, the available knowledge of the domain may have grown substantially. Therefore, it is conceivable that CalcLab could also be used in applications involving two kinds of users. The first kind would be the more sophisticated expert model builders, and the second kind would be the relatively naive users of those models.

One of the major difficulties in the domain of numerical problem solving is the difficulty involved in expressing the problem to the computer. CalcLab addresses this problem by extensively employing the following properties of direct manipulation interfaces:
1. Continuous representation of the objects of interest.
2. Physical actions instead of complex syntax.
3. Rapid incremental reversible operations whose impact on the object of interest is immediately visible. [Shneiderman]

CalcLab uses a menu-driven graphical interface for the construction and use of models. Instances of variables, frames, and computations that are part of a model are continuously displayed to make their relationships evident. Users select the action to perform by pointing with the mouse and selecting options from a menu. This visual environment facilitates ease of understanding and use of the models.

CalcLab allows the user to concentrate on problem solving without requiring the use of a traditional procedural control structure. The display indicates a complete image of the current state. Users can see the consequences of their actions immediately, and if the result is not what is desired they can undo their actions and try a different approach. All the "programming" is done graphically in a form that matches the way people think about the problem. There are no hidden operations, and as a consequence some classes of syntax errors are eliminated. For example, it is not possible to point to a nonexistent object [Hutchins].

CalcLab is based on the notion of exploratory programming [Sheil]. It enables the user to experiment in the process of developing problem solving models and may be viewed as an automatic programming system following Barstow's definition:

An automatic programming system allows a computationally naïve user to describe problems using the natural terms and concepts of a domain with informality, imprecision and omission of details. [Barstow1]
1.3 CalcLab Compared to Electronic Spreadsheets

CalcLab has many similarities with commercially available spreadsheet programs such as VisiCalc. The use of a display-oriented interface, the use of formulas to compute values, and the automatic propagation of changes are common to both. Spreadsheets, however, are used to tabulate rows and columns of numbers, and therefore have facilities to manipulate data in that form. CalcLab is intended for different applications, and therefore does not have such facilities.

On the other hand CalcLab has many features that are not found in the common spreadsheet programs. For example, the CalcLab display continuously and graphically shows the instances, computations, and the connections that have been made between them. Additionally, CalcLab allows the user to collect instances that describe a single object or situation into frames. Apart from the convenience of manipulating related instances collectively, frames also provide a way to organize information in a more intuitive way. Furthermore, CalcLab allows the input and display of values using units of measurement, a necessary feature in scientific use, and CalcLab automatically establishes the conversions between convertible units. Another feature of CalcLab that is not found in electronic spreadsheets is the automatic derivation of formulas from equations, thus freeing the user from another usually tedious task. Finally, CalcLab interacts with the user through an intuitive interface. Instead of using a complex command syntax as required for spreadsheet programs, CalcLab lets the user initiate actions by using the mouse and selecting options from menus.
1.4 CalcLab Features

In summary, the main features of CalcLab are:

- The ability to use models as a representation language for understanding and solving numerical problems in the domain.

- A visual environment that is intuitive and easy to use with an interface that allows the user to concentrate on the domain problem without requiring programming language expertise.

- An interface with which the user can communicate in language natural to scientific numerical problem solving.

- An environment that assists the user by performing many tasks automatically.

- The ability to rapidly prototype models which can easily be experimented with and tested.

- A mechanism for the development of a library of reusable models that may be used to solve a range of similar problems, or combined to solve more complex problems.
2. Survey of Related Work

CalcLab is intended to make the task of numerical problem solving easier. Therefore, in addition to aiding the user by automatically performing tedious or repetitive tasks, it also communicates with the user through an interface that is intuitive and easy to use. A further understanding of the motivation for CalcLab may be gained by surveying some related works that share similar goals or design philosophy.

Sutherland’s Sketchpad [Sutherland] program for drawing and editing pictures is the first landmark in this direction. This is not only because of the historical significance of his work in the field of computer graphics, but also because of the novel ideas he discussed, such as the power of graphical interfaces, visualizing the display as “sheets of paper”, interaction using a pointing device, and the graphical description of abstractions [Hutchins].

Sketchpad uses drawings as a means of communication between the computer and the user. A light pen is used to point to drawings on the display and to position them. Additional control is provided by push buttons, and no programming language knowledge is required in the use of the program. Drawings are composed of straight line segments, circle arcs and previously defined symbols. The program stores the topologies of drawings, so that if a symbol is moved, all the lines attached to it move as well. Drawings can be edited and copied, and highly repetitive structures developed by reproducing an original. Furthermore, conditions (geometric constraints) can be applied between lines to form more complex topologies. Sketchpad has also been considered for use as an input program for the acquisition of
topological data to be used by computation programs such as circuit simulators.

ThingLab [Borning] incorporates many of Sutherland's ideas. ThingLab is a simulation laboratory that provides an environment for constructing graphical models of experiments in geometry and physics. It is implemented in Smalltalk-76, and uses objects to represent parts of a model. Relations among such parts are expressed as constraints that must always hold. These constraints not only describe the models but also serve as commands to the system indicating what conditions must be satisfied during simulation.

The ThingLab interface is intended to allow the user to easily describe and model complex simulations. Therefore, the interface makes extensive use of menus and a pointing device in its interaction with the user. ThingLab extends the domain covered by Sketchpad to include models that are not purely graphical, and also allows non-numerical constraints.

Unlike Sketchpad, however, the model builder working with the ThingLab environment must be familiar with the underlying programming language to be able to build the models. Furthermore, ThingLab initially contains no domain specific knowledge. That is accumulated as objects containing that knowledge are specified to the environment.

The concept of interfaces controlled by pointing devices as introduced by Sutherland and expanded by Borning has begun to see extensive use in programming environments, an example of which is the Xerox 1108 Interlisp-D system. The "mouse" is used as a pointing device for a large format display. This, coupled with extensive interactive graphics facilities, produces a high bandwidth of communication with
the user [Sheil]. Interlisp-D is an exploratory programming environment. It is intended for incremental development of programs by allowing the user to experiment with incompletely specified programs and develop them dynamically in an interactive fashion.

These capabilities of an exploratory programming environment are used by the "phi-naught" system [Barstow3] which is implemented in Interlisp-D on Xerox 1108 workstations. This is an automatic programming system that generates code in target languages such as LISP and FORTRAN. This program is intended for the development of models for use in the domain of oil well log interpretation. Oil well log interpretation is a quantitative data interpretation task. Numerical relationships between measured oil well log data and the desired information are used to develop models for this task [Barstow1]. These numerical relationships are expressed as mathematical equations. The program is driven by a menu-driven icon-oriented interface, and employs standard notations and concepts of this domain to facilitate its use by non-programmers.

The work in this thesis is perhaps most closely related to the idea of electronic spreadsheets, a typical example of which is the VisiCalc program. At the time it was introduced, VisiCalc was considered a significant breakthrough because it enabled the computational power of the computer to be used in tasks such as financial analysis [Norman]. With this program spreadsheets can easily be put together through a display-oriented interface that drives the entire system. When changes are made their implications appear automatically in the relevant spots. Programs such as VisiCalc which run on popular microcomputers have greatly simplified the task of developing financial models [Thomas].

The task of tabulating rows and columns of numbers is a common task in business. The tedium involved in performing this task
and the frustration involved in redoing the task if a mistake occurs or changes must be made make automation of this process very useful. VisiCalc enables the computer to perform the precise and repetitive tasks and frees the user for more creative efforts. Instead of requiring the use of an abstract programming language, communication is performed on a page in front of the user. Information is organized as a grid of "cells" referred to by their coordinates. The relations between individual elements, or cells, can be described by simple arithmetic formulas. A formula computes the value of a cell using the values of other cells, and many formulas can be cascaded to form more complex models. Formulas can be "replicated" easily to compute the values of all similar cells. Powerful functions like SUM, AVERAGE and IF enable more complex models to be developed.

Despite their widespread acceptance and significant utility, programs like VisiCalc have some shortcomings. For example, the control syntax involves cryptic commands that are very obscure and not intuitively meaningful. The conventions chosen for commands are suitable for computer programmers but not for naive users, and therefore initial use of the programs is very error prone. Furthermore, relationships between elements must be expressed as explicit formulas, and there is no facility for the program to generate these formulas automatically from equations. The user must therefore still go through the tedious task of explicitly specifying the formula to be used rather than having these produced automatically from more general relationships like equations.

Another commercial product based on display-oriented interface control is the recently introduced LabVIEW software construction environment. This program, which runs on Macintosh computers, is intended for engineering and scientific test and measurement applications [Vose]. Programs are constructed by using a
mouse and menus to manipulate icons which represent software modules called "virtual instruments".

A virtual instrument consists of a graphically displayed front panel and a block diagram. The front panel is used for the input, output, and interactive operation of the block diagram, which is the actual executable code for the virtual instrument. The front panel is intended to make the software module look and feel like an instrument. The block diagrams are used as a graphical programming language because they are a notation commonly used by scientists and engineers. For the system to be useful in sophisticated applications, it is necessary that software modules be able to combine with each other to form larger software modules. Therefore the icons representing the virtual instruments can be "wired" to other icons using the mouse and menus to build larger and more complex instruments.
3. Program Overview

This chapter provides an overview of the CalcLab program. The implementation is described, and the basic concepts and the operations that may be performed are outlined. Additional descriptions are provided in Chapter 5, User Interface, Chapter 6, Knowledge Representation, and Chapter 7, Automatic Operations. Further details are specified in Chapter 8, CalcLab User's Manual.

3.1 Implementation

CalcLab runs on the Xerox 1108 AI Workstations running the Interlisp-D environment. This Lisp environment provides facilities for interactive graphics using a large bit-mapped display and a "mouse" pointing device.

CalcLab is written in GLISP [Novak1], a high level language based on Lisp, which compiles into Lisp and contains Lisp as a sublanguage. GLISP also provides abstract data types and object-oriented programming. Programs are compiled relative to a knowledge base of object descriptions.

Appendix B: CalcLab Declarations, contains the descriptions of the objects used in CalcLab. These descriptions define the actual data structures for the objects, as well as the messages to which the objects can respond. Also in this appendix are the declarations for the global variables and compile time constants used by CalcLab. The CalcLab source code, or GLISP functions, are contained in Appendix C: CalcLab
Functions. For a detailed language description the GLISP User's Manual [Novak2] may be consulted. Within the GLISP functions many Interlisp-D functions are used for implementing the window based graphics, performing the basic input and output operations, and controlling interaction with the mouse and menus.

3.2 Basic CalcLab Concepts

3.2.1 Equations

Equations are mathematical relationships of the form LHS = RHS, where LHS and RHS are expressions involving variables, predefined constants, numbers, and a number of common mathematical and trigonometric operators. An equation is used to derive formulas for each of the variables used in the equation. An example of an equation is:

\[ \text{Energy} = (\text{mass} \times (c \times c)) \]  
(3.1)

where \(c\), a previously defined constant, is the speed of light. Using this equation, CalcLab derives a formula for the variable Energy in terms of mass and \(c\), and a formula for the variable mass in terms of Energy and \(c\).

3.2.2 Variables

A variable is a quantity that has a real number as its value. Many variables also have some units of measurement. For example, the variable mass as used in Equation 3.1 would have the standard unit of measurement kg.
3.2.3 Units

A unit is a unit of measurement for one or more variables. A unit is described using dimensions of mass, length, time and charge. Velocity, for example, may have the unit of measurement km/hr, which has the dimensions $0, 1, -1, 0$ for mass, length, time, charge. The length dimension of km/hr is in km, and the time dimension of km/hr is in hr.

3.2.4 Instances

An instance of a variable is a box in the CalcLab Work Area which is labeled with the name of the instance (derived from the name of the variable) and which contains the value of that instance. There may be more than one instance of a variable. For example, there may be one instance mass0010 of the variable mass with the value 20 kg and another instance mass0238 with the value 9.1 mg in the work area at the same time.

The value of an instance can be computed using a formula for the variable that it is an instance of. Given Equation 3.1 above, the value of mass0238 may be computed using an instance of Energy in the computation. Various operations may be performed on an instance, including making computations, by clicking the left mouse button on the instance, and selecting one of the options from the pop-up menu that appears.

3.2.5 Computations

Computations are used to compute the value of an instance using the values of other instances. A computation is made from a formula derived from one of the known equations. Alternatively,
computations may be made from an expression supplied by the user. Whenever the value of one of the other instances used in a computation changes, the value of the instance being computed is automatically updated. Computations may be cascaded to model complex situations. Clicking the left mouse button on a computation brings up a pop-up menu from which the user may select options to delete or display the computation.

3.2.6 Templates

A template defines a way to collect together the variables that describe a single object or situation. For example, a “triangle” template may contain the variables area, base and height. Equations that apply only to certain objects or situations may be restricted to the appropriate template. An example of an equation that would be restricted to the triangle template is:

\[
\text{area} = (0.5 * (\text{base} * \text{height}))
\]  

(3.2)

3.2.7 Frames

A frame is a labeled box in the Work Area containing some number of instances. A frame is made from a template, and the instances it contains are instances of the variables contained in the template it is made from. A frame is a useful way to manipulate all the instances pertaining to a single object or situation collectively. Various operations may be performed on a frame by clicking the left mouse button on the frame and selecting one of the options from the pop-up menu that appears.
3.2.8 States

The current state is everything known to the program at the time. This includes all variables, equations, templates, frames, instances, computations, constants, and units. A state can be retained for use at some later time. A previously retained state can be loaded for use again, or loaded onto the current state to combine the information in both. Retained states may be saved in or loaded from files. Saving states in a file provides a secure way to archive them.

3.3 CalcLab Operations

Apart from the operations that can be initiated using the pop-up menus of instances, computations and frames, there are other operations that may be initiated from the Main Menu. These are:

ADD-NEW-EQUATION  Add a new equation to the current state.
ADD-NEW-TEMPLATE  Add a new template to the current state.
ADD-TO-TEMPLATE   Add a variable to the current state.
CHANGE-ERROR      Change the program's sensitivity to changes in values.
DEFINE-CONSTANT   Define a new constant in the current state.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINE-CONVERSION</td>
<td>Define a conversion between two units.</td>
</tr>
<tr>
<td>DEFINE-UNIT</td>
<td>Define a new unit in the current state.</td>
</tr>
<tr>
<td>DELETE-EQUATION</td>
<td>Remove an equation from the current state.</td>
</tr>
<tr>
<td>DISPLAY-CONSTANT</td>
<td>Display the value of a constant.</td>
</tr>
<tr>
<td>DISPLAY-TEMPLATE</td>
<td>Display the variables that have been added to a template.</td>
</tr>
<tr>
<td>LOAD-FILE</td>
<td>Load a file containing saved states making the states available to the program as retained states.</td>
</tr>
<tr>
<td>LOAD-STATE</td>
<td>Load a retained state onto the current state.</td>
</tr>
<tr>
<td>MAKE-ALL-FILES</td>
<td>Save retained states in files.</td>
</tr>
<tr>
<td>MAKE-NEW-FRAME</td>
<td>Make a frame in the Work Area.</td>
</tr>
<tr>
<td>MAKE-NEW-INSTANCE</td>
<td>Make an instance in the Work Area.</td>
</tr>
<tr>
<td>QUIT/RESTART</td>
<td>Terminate the current session and start a new one.</td>
</tr>
<tr>
<td>REDRAW-ALL</td>
<td>Redraw the Work Area.</td>
</tr>
<tr>
<td>RETAIN-STATE</td>
<td>Retain the current state for future use.</td>
</tr>
</tbody>
</table>
4. Examples

This chapter illustrates the use of CalcLab as an environment for developing models for solving numerical problems. The procedures used to build models for some example physics problems are described. The examples included here have actually been modeled using CalcLab. The outlines presented here do not detail the specific CalcLab operations required for each step. Chapter 8, CalcLab User's Manual, should be consulted for more specific details regarding the various operations outlined in this chapter.

One of the significant features of CalcLab is the reusability of the models developed using the program. Models may be retained as states, and states may be later loaded to build other models. The problems in this chapter are solved using such retained states. These include CONSTANTS (various physics constants), MECH-UNITS (commonly used mechanics units), THERM-UNITS (commonly used heat and energy units), 3VECTOR-ADDITION (a model for addition of three vectors), VOLUMES (geometrical formulas for spheres etc.), and CURRENT-CHARGE (equations involving current and charge). The models developed in this chapter can, of course, be retained for use in solving other future problems.

It should be noted that there may be more than one method to solve the example problems described below, and the models developed here are only one way to solve these problems. Additionally, in the models described here some assumptions or approximations are implicit. For instance, acceleration might be assumed uniform, or small energy losses may be considered insignificant. Note also that the positions of instances, frames and computations, and the suffixes
generated for their names will probably be different if the reader replicates the procedures described.

### 4.1 Example Problem 1

An arrow while being shot from a bow was accelerated over a distance of 2.0 ft. If its speed at the moment it left the bow was 200 ft/s what was the average acceleration imparted by the bow? [Resnick]

Since this problem uses units of mechanics, the previously retained state MECH-UNITS, which contains definitions of various units from mechanics, is loaded first. Next, a general model for solving problems involving uniform acceleration is developed. This is done by adding a new template “moving-object”. Next, the following equations, restricted to moving-object, are added:

\[
\text{acceleration} = \left( \frac{\text{final-velocity} - \text{initial-velocity}}{\text{time}} \right) \quad (4.1)
\]

\[
\left( 2 \times \text{distance} \right) \times \text{acceleration} = \\
\left( \left( \text{final-velocity} \times 2 \right) \times \left( \text{initial-velocity} \times 2 \right) \right) \quad (4.2)
\]

\[
\text{distance} = \\
\left( 0.5 \times \left( \text{final-velocity} + \text{initial-velocity} \right) \right) \times \text{time} \quad (4.3)
\]

The standard units for acceleration, final-velocity, initial-velocity, time and distance are m/s2, m/s, m/s, s and m. These, and other units, and the conversions between appropriate units, are known to the environment after MECH-UNITS was loaded. Therefore, they need not be defined but are simply selected from the pop-up menus when the equations are added.
At this point the current state is a general problem solving model for objects moving under uniform acceleration, and this model is saved for future use by retaining the current state under the name UNIFORM ACCELERATION.

To actually solve the above problem, a moving-object frame is made in the work area. This moving-object frame is an idealized “view” of the arrow in the problem. 2 ft and 200 ft/s are input for the distance and final-velocity instances of the frame. Next, the computation of the acceleration is initiated using a known equation, of which there is a choice of two (Equation 4.1 and Equation 4.2). In this case Equation 4.2 is selected from the pop-up menu that appears. The computation is constructed automatically by the program using the required instances in the frame. In order to display acceleration in ft/s2, the units that it is displayed in must be changed using its pop-up menu. The resulting work area is shown in Figure 4.1.

Figure 4.1 Work Area for Example Problem 1
This example illustrates the utility of having retained states containing predefined units, with the conversions between units automatically established, for use in building problem solving models. This enables the input and display of values in the units of choice. Additionally, a general model for uniform acceleration problems has been developed and retained for later use.

4.2 Example Problem 2

An electron with an initial velocity of 1.0E4 m/s is electrically accelerated through 1 cm after which its velocity is 4.0E6 m/s. What was its acceleration, assumed constant? [Resnick]

This problem may be solved using the model developed for Example 1. This is done by loading the state UNIFORM-ACCELERATION, and then making a new moving-object frame. This frame is now a view of the electron, and the known values for initial-velocity, final-velocity and distance are input. The computation of the acceleration is initiated using a known equation in terms of final-velocity, initial-velocity, and distance (Equation 4.2). The required instances are automatically selected from the frame. The resulting Work Area is shown in Figure 4.2.

This example shows how a previously developed model may be used to solve problems other than those which the model was originally developed for.
4.3 Example Problem 3

At the instant the traffic light turns green, an automobile starts with a constant acceleration of 6 ft/s². At the same instant a truck, traveling with a constant speed of 30 ft/s, overtakes and passes the automobile. (a) How far beyond the starting point will the automobile overtake the truck? (b) How fast will the car be traveling at that instant? [Resnick]

In this problem there are two objects moving with uniform acceleration, so the previously retained UNIFORM-ACCELERATION state (from Example 1) is loaded, and two moving-object frames are made in the Work Area. One of these represents the automobile, and is renamed "automobile", the other represents the truck, and is renamed "truck". Known values, 6 ft/s² for automobile acceleration and 30 ft/s for truck initial-velocity, are then input.
The point of interest is when the automobile passes the truck, a point at which they both have the same values for distance and time. Therefore, the truck time is constrained to use the value of the automobile time, and the automobile distance is constrained to use the truck distance. As a result of using these constraints there is now a cycle in the current state.

The unknown quantities for the automobile are the automobile time and automobile final-velocity. The time is computed using Equation 4.1 and the final-velocity is computed using Equation 4.2, with all the required instances being automatically selected from the automobile frame. The unknown quantities for the truck are the truck distance and truck final-velocity. The distance is computed using Equation 4.3 and the final-velocity is computed using Equation 4.1, with all the required instances being automatically selected from the truck frame. The Work Area after all these computations have been set up is shown in Figure 4.3.

At this stage the current state is a stable initial state representing the situation when the truck passes the automobile. In order to find another stable state that satisfies all constraints, a guess will have to be input for time or distance, and if the state stabilizes with non-zero values for distance and time, that will represent a time at which the automobile passes the truck. 30 s is input for truck time, and the state immediately becomes unstable. In this condition the changing values for time and distance produce further changes in the values for time and distance because of the cycle in the current state. After a few seconds the state becomes stable again as the values for time and distance converge to the solution. This stable final state is shown in Figure 4.4.
Figure 4.3 Work Area for Example Problem 3 (initial)

Figure 4.4 Work Area for Example Problem 3 (final)
This example illustrates how CalcLab may be used to model a problem involving simultaneous equations by establishing cyclic constraints between the entities involved, and then perturbing the state and waiting for values to converge to a solution. Thus, problems may also be modeled and solved using this relaxation technique. There are some cases in which this approach may not work because the state starts to oscillate, or the underlying Interlisp system overloads. For some cases the allowed error has to be changed so that a stable, though less accurate, state is reached ultimately, or reached sooner.

4.4 Example Problem 4

A charged thundercloud is producing rain drops of average diameter 3 mm, each carrying a charge of 3E-15 C. How heavy must be the rainfall (in cm per hour) for the electric current density to the earth to be 5E-11 A/m^2? [Bennet]

This problem is an example of how concepts from different areas of physics need to be brought together to solve a problem. CalcLab is an environment well suited for this. Before starting to develop a model for this problem MECH-UNITS, CURRENT-CHARGE, and VOLUMES, previously retained states containing definitions of mechanics units, current and charge models, and geometric relationships for solids are loaded.

First the following equation is added:

$$\text{rainfall-rate} = \left(\frac{\text{total-volume}}{\text{area}}\right) / \text{time}$$

(4.4)

The standard unit for rainfall-rate is m/s. A new unit, cm/hr, is defined with dimensions 0, 1, -1, 0, for mass, length, time, charge, with the
length dimensions in cm and time dimensions in hrs. Rainfall-rate can now be displayed in cm/hr.

It is necessary to be able to describe a raindrop. This is done by making a new "sphere" frame (previously defined in VOLUMES) in the Work Area. A value of 3 mm is input for the diameter. The radius and volume computations are created automatically when initiated. An instance of charge/unit is made and the value 3E-15 C input for it. An instance of current-density is made and the value 5E-11 A/m2 input for it.

Once all the known numbers have been input an instance of rainfall-rate is made and a computation for it initiated using the only known equation. When prompted for total-volume, area and time to use in the computation, new instances of each are made. Next the total-volume is computed. This is done using the already known (from VOLUMES) equation:

\[ \text{total-volume} = (\text{unit-volume} \times \text{quantity}) \quad (4.5) \]

When prompted, the sphere volume is selected for unit-volume and a new instance is made for quantity. Next the quantity is computed using the previously added (from CURRENT-CHARGE) equation:

\[ \text{charge} = (\text{charge/unit} \times \text{quantity}) \quad (4.6) \]

When prompted, a new instance is made for charge and the instance of charge/unit selected for charge/unit. Next charge is computed using (from CURRENT-CHARGE) the previously added equation:

\[ \text{current} = (\text{charge} / \text{time}) \quad (4.7) \]
When prompted, a new instance is made for current and the instance of time is selected for time. Next the current is computed using the previously added equation:

\[
\text{charge-density} = \left( \frac{\text{current}}{\text{area}} \right) \tag{4.8}
\]

When required, the displayed instances of charge-density and area are selected.

The model is now complete, but there is no answer because time and area still have the value zero. The answer is independent of the values of time and area, so any values such as 1 min and 1 mi2 for time and area are input. The required answer for rainfall-rate is now available, as shown in Figure 4.5.

![Figure 4.5 Work Area for Example Problem 4](image-url)
This example is a good illustration of a case where the ability to quickly define a new unit (cm/hr) is needed. It also demonstrates the utility of using previously defined knowledge, in this case the sphere frame and restricted equations along with Equation 4.5-8.

4.5 Example Problem 5

In an experiment with a high-energy beam, hydrogen atoms each of mass 1.67E-27 kg strike a target with a velocity of 2E7 m/s. If 1E15 atoms arrive each second, and the target is a lump of brass of 0.5 kg thermally insulated, find how long it will take for the temperature of the brass to rise by 100 K. (Specific heat capacity of brass = 0.38E3 J/kg/K) [Bennet]

To solve this problem first the states MECH-UNITS and THERM-UNITS are loaded. Then a new template "heated-object" is added to which is restricted the added equation:

\[
\text{total-energy} = \\
( ( \text{mass} \times \text{temp-change} ) \times \text{spec-heat-cap} )
\] (4.9)

The units J, kg, and K are chosen as standard units for total-energy, mass and temp-change. For spec-heat-cap a new unit, J/kg/K, is defined, with dimensions 2 for length in m and -2 for time in s.

Next a new template "moving-object" is added to which is restricted the equation:

\[
\text{kinetic-energy} = (0.5 \times (\text{mass} \times (\text{velocity} \times 2)))
\] (4.10)

The units J and m/s are selected as standard units for kinetic-energy and velocity.
Two more equations are added to build a complete model of this problem. The first is:

$$\text{net-energy} = (\text{energy/particle} \ast \text{quantity}) \quad (4.11)$$

The unit J is chosen for net-energy and for energy/particle, with no units for quantity. The second is:

$$\text{quantity} = (\text{rate} \ast \text{time}) \quad (4.12)$$

The unit s is chosen as the standard unit for time. A new standard unit, /s, is defined for rate. This unit has time dimension -1 in s.

The model is built by first making a heated-object frame representing the lump of brass and a moving-object frame representing a hydrogen atom. The assumption/approximation that the mass of the hydrogen atoms is negligible when compared to the lump of brass is implicit here.

All the given information is input first. An instance of rate is made and the value 1E15 /s input for it. 0.5 kg, 100 K, and 0.38E3 J/kg/K are input for the heated object mass, temp-change, and spec-heat-cap. 1.67E-27 kg and 2E7 m/s are input for the moving object mass and velocity.

The problem requires the computation of time, so an instance of time is made for which is a computation is initiated using the only known equation for time. When prompted a new instance is made for quantity and the existing instance of rate is selected for rate.
Next, quantity is computed by selecting Equation 4.11 from the pop-up menu. When prompted, heated-object total-energy is selected for net-energy and moving-object kinetic-energy is selected for energy/particle. Finally, heated-object total-energy and moving-object kinetic-energy are computed (automatically) from their known equations. The model and solution are complete, as shown in Figure 4.6.

![Figure 4.6 Work Area for Example Problem 5](image)

This example shows how the different entities involved in a problem can be neatly organized and displayed using CalcLab so that their interactions are evident. The model built for this problem can be reused for similar problems by simply changing the values of the known quantities with the result being computed automatically.
4.6 Example Problem 6

A tennis ball is dropped onto the floor from a height of 4.0 ft. It rebounds to a height of 3.0 ft. If the ball was in contact with the floor for 0.01 s, what was its average acceleration? [Resnick]

This example illustrates the use of a previously defined constant and the use of one type of frame for two different phases of motion. Initially the previously retained states MECH-UNITS, containing mechanics units definitions, and CONSTANTS, containing constant definitions, are loaded.

Next new template “free-fall” is added. Then the equation:

\[(\text{vel-at-bottom}^2) = (2 \times (g \times \text{height}))\]  

is added, restricted to the free-fall template. \(g\), the acceleration due to gravity at the surface of the earth, is a constant already defined in CONSTANTS. The standard units for vel-at-bottom and height are m/s and m respectively.

Also added, not restricted to any template, is the equation:

\[\text{acceleration} = (\left(\text{final-vel} - \text{initial-vel}\right) / \text{time})\]  

with the standard units m/s², m/s, m/s, and s for acceleration, final-vel, initial-vel, and time.

Since there is motion in two directions, downwards then upwards, a “reverse-direction” template is added to which is restricted the equation
downwards = (- upwards) \hfill (4.15)

After this is done two new free-fall frames are made in the work area and one is renamed "down" and the other is renamed "up." The values 4.0 ft and 3.0 ft are input for down height and up height respectively. A reverse-direction frame is made, and upwards for this frame is constrained to use the value of the up vel-at-bottom. The reverse-direction downwards is computed using a known equation, a computation that the program makes automatically. An instance of time is needed, so one is made and the value .01 s input for it.

Figure 4.7 Work Area for Example Problem 6

Computations for vel-at-bottom for both frames are performed using a known equation, again computations that the program makes automatically. The problem is to find the acceleration, so an instance of acceleration is made, and a computation using a
known equation initiated. When prompted, downwards is selected for final-vel, down vel-at-bottom selected for initial-vel, and the time instance is selected for time. The required results are now available, acceleration may be displayed in ft/s², and the work area appears as shown in Figure 4.7.

4.7 Example Problem 7

A train moving at an essentially constant speed of 60 mi/hr moves eastward for 40 min, then in a direction 45 degrees east of north for 20 min, and finally westward for 50 min. What is the average velocity of the train during this run? [Resnick]

Figure 4.8 3VECTOR-ADDIION State Loaded

The above problem can be solved by visualizing each phase of motion as a two-dimensional vector, adding the three vectors, and using
the resultant to compute the average velocity. A previously retained state containing a model for the addition of three vectors, 3VECTOR-ADDITION, is loaded, after which the work area appears as shown in Figure 4.8. In addition to this state, the state MECH-UNITS is also loaded.

To describe each phase of the train’s motion, a new template "motion" is added. The equation

\[
\text{avg-velocity} = (\text{distance} / \text{time})
\]  

(4.16)

is then added, restricted to motion. Next, four motion frames are made, renamed phase-1, phase-2, phase-3 and resultant, and then positioned above the appropriate vector as shown in Figure 4.9.

Figure 4.9 Motion Frames Renamed and Positione
The resultant time is the sum of the times for each individual phase. A computation is initiated for resultant time, but instead of using a known equation, the expression \((\text{time} + (\text{time} + \text{time}))\) is specified for use in the computation. When prompted, phase-1 time, phase-2 time, and phase-3 time are selected for each time in the expression.

![Figure 4.10 Work Area for Example Problem 7](image)

For each phase, the magnitude of the associated vector is constrained to use the value of the distance of that phase. The resultant distance is constrained to use the value of \(\text{vector1} + 2 + 3.\text{out}\) magnitude. The speed 60 mi/hr is input for the avg-velocity of each phase of motion. 0, 45, and 180 degrees are input for the angle of the appropriate vectors for the three phases. Computations using a known equation are initiated for the distance of each phase, and these are made automatically by the program. A computation is initiated for the
resultant avg-velocity, which is also made automatically. Then the avg-velocity units are changed to mi/hr. The problem is solved, and the work area appears as shown in Figure 4.10.
5. User Interface

CalcLab employs a direct manipulation interface to interact with the user. All possible options at any time are displayed in a menu from which the user may select the desired option with the mouse. This approach has a number of advantages. The user does not have to remember all the possible operations and their names since they are always displayed. Additionally the user does not have to remember any complex command syntax since the appropriate menu is presented to the user at each step. Since all possible options are displayed, and the user may only select from those displayed, the possibility that the user will make an error by selecting a nonexistent option or mistyping an option is eliminated.

All instances, computations and frames that are part of the current state are continuously displayed, and to perform an operation on one of them the user simply clicks the left mouse button on it and selects the desired option from the pop-up menu that appears.

Whenever the program requires a response from the user, and any possible responses are known to the program, a pop-up menu of those responses is presented, from which the user may select one. The option to define a new response, if appropriate is also included in the menu. For example, when adding a variable to a template, the user is presented with a menu of all the variables from which the user may select the one to add. This menu also contains the option for the user to define a new variable to be added to the template. If this option is selected, the user is then asked to actually type in the name of the variable. When typed input is required of the user, a window where the user may type that input is presented on the display.
Once CalcLab is loaded and initialized, all interaction between the user and CalcLab occurs using the CalcLab windows and the menus that appear in them. The CalcLab windows, through which most of this interaction takes place, is shown in Figure 5.1. The four CalcLab windows shown in this figure are described individually below.

Figure 5.1 CalcLab Windows
5.1 CalcLab Work Area

The Work Area is used to display all the instances, computations and frames that are part of the current state. Figure 5.2 shows the Work Area in use. It contains a heated-object frame, a moving-object frame, various instances, and the computations that have been made interconnecting them. Since the Work Area has limited space it is necessary to be able to shrink frames and instances. Figure 5.3 shows the Work Area containing the same instances, computations and frames as shown in Figure 5.2, but with the heated object frame and quantity instance shrunk, and with the frames and instances moved towards one side of the Work Area to conserve space.

![Figure 5.2 Work Area](image-url)
5.2 CalcLab Main Menu

The Main Menu is a menu of all the operations that may be performed other than those that may be selected from the pop-up menus for the instances, computations, and frames. The Main Menu appears as shown in Figure 5.4. The user may select the desired operation from these options clicking the left mouse button on it.

Figure 5.4 CalcLab Main Menu
5.3 CalcLab Status and Prompts

The Status and Prompts window is used to show the user the action being performed when one is initiated, and to indicate when it terminates. Additionally, this window is used to prompt the user for input. Figure 5.5 shows the Status and Prompts window after the file EXAMPLES has been loaded and the state BENNET-5 has been loaded.

![CalcLab Status and Prompts](image)

Figure 5.5 CalcLab Status and Prompts

5.4 CalcLab Display

The Display is used to display constants, computations, and the variables in templates. Figure 5.6 shows this window after the variables in the template heated-object, and the computation for an instance of time, have been displayed.
5.5 Pop-Up Menus

In addition to the Main Menu, operations may be selected from the pop-up menus of instances, computations and frames. These menus appear when the left mouse button is clicked on the instance, computation, or frame on which the operation is to be performed. Figure 5.7 shows the instance rate0355 and its pop-up menu. Figure 5.8 shows the computation CMP0366 and its pop-up menu. Figure 5.9 shows the frame moving-object0346 and its pop-up menu. For further details on the operations performed by options in all of these menus Chapter 8, CalcLab User's Manual, should be referred to.
Figure 5.7 Instance and Pop-Up Menu

Figure 5.8 Computation and Pop-Up Menu

Figure 5.9 Frame and Pop-Up Menu
6. Knowledge Representation

The knowledge available to CalcLab consists of computations, constants, equations, formulas, frames, templates, instances, links, units and variables. These constitute the current state of the system and are all represented by GLISP objects as specified in Appendix B: CalcLab Declarations. Each of these GLISP objects is described by the actual data structure used for the object and the messages that the object can respond to. The following is a brief description of the more important components of these objects.

A computation (COMPUTATION) consists of several components. INSTANCECOMPUTED is the name of the instance that it is used to compute. INSTANCESNEEDED is a list of the names of all the instances that are used in the computation. EXPRESSION is the actual Lisp expression in terms of these instances that is used to compute the required value. CONSTANTS is a list of the names of all the constants used in the computation. LINKSLIST is a list of the names of all the links connecting instances to this computation. In addition there is information regarding the display position and size of the computation in the Work Area.

A constant (CONSTANT) is described by two things: its value and a brief description string that is printed in the Display window when the constant is displayed. Equations (EQUATION) too are described by two things: the character string of the original equation added, and a list of the names of all the formulas derived from the equation. Formulas (FORMULA) are described by: VARIABLES, which is a list of the name of all the variables used in the formula, EXPRESSION, which is the actual Lisp expression for the formula in terms of these variables,
CONSTANTS, which is a list of the names of all the constants used in the formula, and FRAMEDEF, which is the name of the template that this formula is restricted to if it was derived from an equation restricted to a frame template.

A frame (FRAME) is a data structure representing a stereotyped situation, and has slots that are filled by instances of data [Minsky]. These slots (INSTANCES) contain the names of the instances that are contained in the frame. In addition, a frame contains information regarding its position and size in the Work Area. Frames are made from templates, and templates (FRAMEDEF) consist of a list of the names of the variables in template (VARIABLES) and a list of the names of all the frames that have been made from the template.

An instance (INSTANCE) has many component parts. It has a value, LINKSLIST, which is a list of the names of all the links associated with it, COMPUTATIONS, which is a list of all the computations that are used to compute its value, USEDBY, which is a list of all the computations it is used in to compute the values of other instances, FRAME, which is the name of the frame (if any) that it is a part of, VARIABLETYPE, which is the name of the variable that it is an instance of, and additional information regarding the units it is to be displayed in and information regarding its position and size in the Work Area.

A link (LINK) consists of the names of the instance and computation it connects, and the coordinates of its end points. A unit (UNIT) is represented by a dimension vector (mass, length, time, charge), a list of conversions for units that it can be converted to, and MKSCCF, which is the factor by which it must be multiplied to convert it to mks (meter-kilogram-second-Coulomb) units. The last object is a variable (VARIABLE) which is described by a list of the names of formulas derived for it and the name of the standard unit for its measurement.
When a state is retained, all these objects are copied into a list under the name of the state. Thus it is possible to separate and store knowledge in different states. To enable easy archiving, and to make a state safe from accidental smashing, states may be saved in files of the user's choosing. Thus knowledge is represented by objects at the lowest level, organized into states to form separate models, and at the next higher level stored in files for later use.
7. Automatic Operations

Much of the power of CalcLab as an environment for numerical problem solving comes from the operations it performs automatically. These operations free the user from the need to perform tedious or repetitive tasks and allows the user to concentrate on the problem at hand. Four significant automatic operations of CalcLab are described in this chapter.

7.1 Algebraic Manipulation

Algebraic manipulation is performed to derive formulas for variables from the equations that use them. Although there are many equations from which CalcLab cannot derive the formulas for all the variables in those equations, the program will still derive formulas for as many variables as it can, leaving it to the user to specify expressions to use for those variables for which formulas could not be derived. The algebraic manipulation functions operate on commonly used mathematical and trigonometric functions. The automatic derivation of formulas frees the user from having to do so manually.

7.2 Algebraic Simplification

Frequently, expressions are entered by the user that can be simplified to produce more efficient expressions. Therefore CalcLab simplifies the formulas that it derives to remove any redundancies that were part of the original equation, or that were produced by the
algebraic manipulation to derive the formulas. While it is performing this simplification the program also checks for potential error conditions such as division by zero or square root of a negative number. In this way the user can be alerted to possible error conditions before they occur, and the models fixed to remove those problems before they become too deeply embedded and therefore too hard to detect.

7.3 Unit Conversion

When a new unit is defined, in terms of its dimensions, the program attempts to automatically establish conversions between the new unit and any already known units with the same dimensions. For example, if the units m/s, mi and hr are already defined, when the new unit mi/hr is defined, all the user has to specify is the dimensions for the unit, and the fact that its length dimension is in mi and its time dimension is in hr. The conversion between mi/hr and m/s will be established automatically, and the user will be free to input and display instance values using either mi/hr or m/s where previously only one could be used. Since unit conversion is usually a tedious task it is very useful to have it done automatically.

7.4 Making Computations

The making of computations is initiated by the user, but in many cases the computation is made completely by the program. For example, if the instance to be computed is part of a frame, and there is only one known equation that can be used to make the computation, and all the other required instances are also part of the frame, then the computation is automatically made using that equation with the instances in the frames being automatically selected. This kind of
condition arises quite frequently, so there is significant utility in this ability.

If the user indicates that a computation is not to be made using a known equation, then the user is asked for an expression to use to compute the instance. This expression may contain the names of instances or variables or both. The program automatically prompts the user to select a specific instance for each occurrence of a variable in the expression. It is possible to specify the expression \((\text{time} + \text{time})\), for example, in which case the program will prompt the user to select an instance of time for both occurrences of time in the expression.

When an instance is selected for a computation or a USE constraint, CalcLab automatically performs the necessary checking to make sure the instance selected has compatible units or dimensions with the one required. This frees the user from having to check for the selection of incompatible instances and helps prevent the inadvertent selection of an inappropriate instance.

8.1 Equations

Equations are used to derive the formulas used in constructing computations. Equations may be added to the current state using the ADD-NEW-EQUATION option. When an equation is added, for each variable in the equation a formula in terms of the other variables is derived. When an instance of a variable needs to be computed, a computation for the instance is created using one of the formulas for the variable. An equation may be restricted to a particular template, in which case formulas derived from the equation can only be used to construct computations for variables in that frame template.

Equations are mathematical relationships of the form LHS = RHS, where LHS and RHS are expressions involving variables, predefined constants, numbers, and two kinds of operators:

1. Binary Infix Operators:
   +   (plus)
   -   (difference)
   *   (times)
   /   (quotient)
   **  (exponent)

2. Unary Prefix Operators:
   -   (minus)
   LOG
   ANTILOG
   COS
   ARCCOS
   SIN
   ARCSIN
   TAN
   ARCTAN
   SQRT
Some examples of legal expressions are:

\[
3 \\
\text{width} \\
(\text{width} \times \text{height}) \\
(X^{**2}) \\
((3 \times (\text{COS Y})) + (X^{**3}))
\]

If it were wished, for example, to add the equation weight = mass \times g, where \( g \) is the constant "acceleration due to gravity", the following steps need to be taken:

1. Select the DEFINE-CONSTANT option to add the constant \( g \), with value 9.81, and description "acceleration due to gravity".

2. Select the ADD-NEW-EQUATION option and for LHS input weight, and for RHS input \((\text{mass} \times g)\).

Note that there are some equations from which the formulas for some variables cannot be derived. If the program is unable to derive a formula for a variable the user may specify an expression to use to compute an instance of that variable. COS, ARCCOS, SIN, ARCSIN, TAN, ARCTAN all assume angular measurement in degrees.

8.2 Variables

A variable is an entity that has a real number as its value. A variable may also have a standard unit of measurement, which is the unit assumed for a variable in all equations that use that variable.

A variable may be added to a state in one of two ways; either by adding an equation containing the variable (ADD-NEW-EQUATION
option), or by adding the variable to a frame template (ADD-TO-TEMPLATE option).

A state may contain more than one instance of a variable, each of which has its own unique value and its own unique name formed from the name of the variable.

8.3 Units

When a variable is added to a state the user is asked if it has a standard unit of measurement. If it does, then the user is asked to select the standard unit from those already known. If a new unit is to be defined, the user can select NEW-STANDARD-UNIT at this point. Whenever a new unit is defined the user is prompted to select the dimensions of Mass, Length, Time and Charge for the unit. This is done by selecting one of -3, -2, -1, 0, 1, 2, 3, as the power of each dimension of that unit.

The basic units m (meter), kg (kilogram), s (second) and C (Coulomb), are predefined as the mksc units for CalcLab.

If a variable can be measured in units other than the standard unit, then the other units may be added using the DEFINE-UNIT option.

When a new unit is defined, and the user has specified the dimensions of that unit, the user is asked to select the units for each dimension from other known units. If the dimension is not in one of those units the user can specify a conversion factor from that dimension to one of the basic units. CalcLab uses this information to define automatic conversions between units. If the user wishes to define a
conversion between units that was not automatically defined, the user can do so using the DEFINE-CONVERSION option.

For example, in order to define the unit m/s, after selecting the DEFINE-UNIT option, select 0, 1, -1, and 0 for its mass, length, time, and charge dimensions. Then select m for the mass dimension unit and s for the time dimension unit. To subsequently define the unit km, select 0, 1, 0, 0 for its dimensions, then indicate that the length dimension of km is none of the known units. The program will ask if the length dimension converts to m, which it does, and then the conversion factor of 1000 from the length dimension of km to m can be typed in. To then define the unit km/s, select 0, 1, -1, 0 as the dimensions, then select km for the length dimension and s for the time dimension. CalcLab will automatically define a conversion between km/s and m/s, and if m/s is specified as the standard unit for some variable, say velocity, then values for instances of velocity may be specified in km/s or m/s. Additionally, values of those instances can be displayed in km/s or m/s as desired.

8.4 Instances

Instances are made by selecting the MAKE-NEW-INSTANCE option. An instance of a variable is a box in the Work Area. This box is labeled with the name of the instance, and contains the value of the instance (initially zero). When the left mouse button is clicked on an instance, a pop up menu as shown in Figure 8.1 appears. The user can select one of the following options from the pop-up menu:
Figure 8.1 Instance and Pop-up Menu

**DELETE**

This option deletes the instance from the state. If there are any computations associated with the instance they are deleted too.

**RENAME**

This option allows the user to change the name of the instance.

**USE**

This option is used to constrain the value of the instance to the value of some other instance. The user is prompted to select the other instance by clicking the left mouse button on it and selecting the SELECT option from the pop-up menu. The two instances are connected through a USE pseudo-computation.

**COMPUTE**

This option is used to construct a computation for the value of the instance. The user is asked whether a known equation is to be used, or a user-specified expression is to be used. If a known computation is to be used, then the user is asked to select the equation to be used from a menu. If only one equation applies, then it is automatically selected. As the
computation is constructed, the user is asked to select needed instances of the variables that are used in the computation. These can be selected by clicking the left mouse button on them and selecting the SELECT option from the pop-up menu. If the equation used is restricted to a particular frame, and the instance to be computed is part of the frame, then if a needed instance is also part of the frame, it will be automatically selected. If the user indicates that a known equation is not to be used, then the user is prompted to input an expression that can be used to compute the value of the instance to be computed. The user may use any known constants, variables or instances in this expression, using the form used for the LHS or RHS of equations.

**SHRINK**

This option shrinks the size of the box so that only the name of the instance is displayed. This helps save space in the Work Area.

**EXPAND**

This option reverses SHRINK.

**MOVE**

This option is used to move the box for the instance to a new location within the Work Area.

**CHANGE-UNITS**

This option is used to display the value of the instance in one of the other units to which its standard unit can be converted.

**INPUT-VALUE**

This option is used to input a new value for the instance from the user. When prompted, the user should type in the value, but not the units, if any. If there is a choice of units a pop-up menu will appear from which the user can select the appropriate unit. Note that if there is a computation or a USE for the instance this new value might be overwritten when there is a change in the value of some other instance.

**SELECT**

This option is used to select the instance for use in a USE or a computation.
Occasionally, the window prompting the user to select an instance will freeze in reverse video. In this case the user should first select the instance, then press the space bar to continue.

Note that an instance does not need to be made until it is needed. An instance can be made using the MAKE-NEW-INSTANCE option at the time that the program prompts the user to select an instance. The MAKE-NEW-INSTANCE option automatically selects the instance made for use in any computation or USE currently being constructed.

8.5 Computations

Computations are used to compute the value of an instance using the values of other instances. They are created by selecting the COMPUTE option from the pop-up menu of the instance to be computed. Whenever the values of the other instances used in the computation change, the value of the instance being computed is automatically updated. Note that a USE behaves like a computation, and can be similarly manipulated. If the left mouse button is clicked on a computation, a pop-up menu as shown in Figure 8.2 appears, from which the user can select one of the following options:

![Figure 8.2 Computation and Pop-up Menu](image)
DELETE  This option deletes the computation from the state.

DISPLAY  This option is used to display, in the Display, the equation that this computation is based on, and the LISP expression used to do the actual computation.

8.6 Templates

It is sometimes convenient to collect all the variables that describe a single object or process into a single frame template. A template can be initialized by using the ADD-NEW-TEMPLATE option. Variables can then be individually added to the template using the ADD-TO-TEMPLATE option. Alternatively, when adding an equation that is restricted to a previously initialized template using the ADD-NEW-EQUATION option, the user can specify which variables used in the equation, if any, are to be added to that template.

When the MAKE-NEW-FRAME option is selected the program makes a frame in the Work Area window. This frame contains instances of all the variables that are part of the frame template from which the frame is made. The frame is given a unique name formed from the name of the template. A state may contain more than one frame made from a single frame template.

The DISPLAY-TEMPLATE option may be used to see which variables have been added to a template.
8.7 Frames

Frames are collections of instances. They are useful as descriptions of objects or processes that are described by more than one variable. The instances in a frame correspond to the variables in the frame template from which the frame is made. Frames are made by selecting the MAKE-NEW-FRAME option. Frames are displayed in the Work Area as boxes, labeled with the name of the frame, and enclosing boxes for instances in the frame. By clicking the left mouse button on the label for the frame, a pop-up menu as shown in Figure 8.3 appears, from which the user can select one of the following options:

![Frame and Pop-up Menu]

**Figure 8.3 Frame and Pop-up Menu**

**DELETE**

This option is deletes the frame from the current state. If the frame contains instances that are currently used in some computations, then those computations must first be deleted before the frame may be deleted.
RENAME

This option allows the user to change the name of the frame.

SHRINK

This option shrinks the frame so that only its label is displayed. This helps conserve space in the Work Area window.

EXPAND

This option reverses SHRINK.

MOVE

This option is used to move the box for the frame to another location in the Work Area.

8.8 States

A state is everything known to a program at a certain time. States consist of variables, equations, frame templates, instances, frames, computations, constants and units. The current state is the state being used by the program at the current time. States can be retained for later use using the RETAIN-STATE option. A retained state can be loaded in to replace the current state, or added to the current state, using the LOAD-STATE option. States can be saved in and retrieved from files using the MAKE-ALL-FILES and LOAD-FILE options.

8.9 Loading and Initialization

CalcLab runs in the Interlisp-D environment on the Xerox Al Workstations. To load CalcLab from a floppy disk, first insert the floppy disk in the drive, then type

(LOAD 'FLOPPY)CALCLAB.DCOM)

in the Interlisp-D Executive window at the flashing caret. After CalcLab has been loaded it is initialized by typing
in the Interlisp-D Executive window at the flashing caret. The CalcLab windows will then appear, and CalcLab is ready for use. If the Interlisp-D Executive window is obscuring any of the CalcLab windows, it can be "shaped" into the space just below the CalcLab Work Area using the mouse. The resulting screen appears as shown in Figure 8.4.

Figure 8.4 Initial CalcLab Screen
8.10 CalcLab Windows

Status and Prompts
This window shows the user what the program is currently doing, or has just completed doing. It also displays errors detected by the program, and may display prompts for user input.

Display
This window displays information that the user might wish to see.

Main Menu
This contains the main menu from which the user can select any one of the options by clicking the left mouse button on it.

Work Area
This window displays the instances, frames, and computations that are part of the current state. By clicking the left mouse button on one of these the user can initiate actions from their pop-up menus.

It is possible to "close" all the CalcLab windows by clicking the right mouse button and selecting the close option. If this should happen accidentally, type

(DRAWALL)

in the Interlisp-D Executive window to restore CalcLab.

8.11 Input to CalcLab

The program can require input from the user in one of two ways. It either presents a pop-up menu from which the user should make a selection by clicking the left mouse button on the desired option. Or it asks the user to type in a response into an input window.
When the program requires the user to type in input it prompts the user with a message followed by `>>>`. In order to type into an input window, the user should click the left mouse button in the input window as a result of which a flashing caret appears where the user should type in. The response can then be typed in. Unless it is a parenthesized expression, the user must follow the response with a carriage return. The user may switch between uppercase and lower-case input using the LOCK key.

There are certain words that are reserved for internal use by the program, and if one of those is entered, an error message is printed, and the user is prompted for input again. The symbol `%` is not acceptable as input.

If the TTY window for MOUSE input window occludes any needed part of the Work Area, the window can be "shaped" into a more convenient location.

When the value of an instance is asked for, only the value needs to be typed in. If the value is in any units of measurement for the instance the program will automatically ask the user to select the units from a pop-up menu. Values may be typed using scientific notation. For example, `1.6 \times 10^{-19}` may be typed in as `1.6E-19`.

Occasionally, while the program needs the user to select an instance for a USE or computation, the input window will freeze in reverse video. In this case the user should first perform the selection requested, then press the space bar to continue.

The program can be made to abort whatever action it is performing by simultaneously pressing the PROP'S and D keys.
8.12 Error Messages

When a message of the form --- message --- appears in the Status and Prompts window, the program has detected an error and aborted whatever was being done.

8.13 Main Menu Options

The Main Menu appears as shown in Figure 8.5. The user may select any of the Main Menu options by clicking the left mouse button on the desired one. The Main Menu options are described below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD-NEW-EQUATION</td>
<td>This is used to add a new equation to the current state. The user is asked to select the template that this equation is to be restricted to. If the equation is not to be restricted to any template the NOT-RESTRICTED option should be selected. The user is then asked to type in the LHS of the equation, and then to type in the RHS of the equation. LHS and RHS are the left hand side and right hand side of the equation, and must have the correct expression syntax for equations. The program then processes the equation one variable at a time. For each new variable the user is asked to indicate if the variable has a standard unit of measurement. If the...</td>
</tr>
</tbody>
</table>
user selects YES, the user will be asked to select the unit from those already known. If the user wishes to define a new unit, then the user should select NEW-STANDARD-UNIT at this point. New units are defined as explained for the DEFINE-UNIT main menu option. If the equation being added is to be restricted to a frame template, the user will also be asked if the variable should be added to that template. If any constants are used in the equation, they must be previously defined for the current state using the DEFINE-CONSTANT main menu option or by loading a previously retained state containing those constants, otherwise they will be treated as variables.

ADD-NEW-TEMPLATE

This is used to initialize a new template in the current state. The user is asked to type in the name of the template to be added. To add variables explicitly to the template the ADD-TO-TEMPLATE option may be selected. Alternatively, ADD-NEW-EQUATION can be used to add variables to a template if the equation being added is restricted to that template and contains the variables to be added.

ADD-TO-TEMPLATE

This is used to add a variable to a frame template. The user is asked to select the template to which the variable is to be added. Then the user is asked to select the variable to be added from those already part of the current state. If a new variable is to be added, the user should select NEW-VARIABLE at this point. The user is then asked to indicate whether the variable has a standard unit of measurement. If the user selects YES the user is asked to select that unit from those known or define a new unit by selecting NEW-STANDARD-UNIT from the pop-up menu.

CHANGE-ERROR

This is used to change the value of a the allowed error. Allowed error is the minimum fraction of an instance’s value which is considered sufficient to change the value of an instance when added or
subtracted from it. This option can be used to change the sensitivity of the program to vary small changes in the value of instances. Smaller values of the allowed error will make the program more sensitive. Larger values can be used to make the current state converge faster if cycles exist. The user is asked to type in the new value for the allowed error.

**DEFINE-CONSTANT**

This is used to define a new constant to the current state. The user is asked to type in the name of the constant to be added. Then the user is asked to type in the value of the variable. Finally the user is asked to type in a brief description string (enclosed in double quotes) for the variable.

**DEFINE-CONVERSION**

This is used to define a conversion between two units already part of the current state if one has not automatically been defined. The user is asked to select the first unit, then to select the second unit. Then the user is asked to type in the number by which a quantity in the first unit must be multiplied to convert it to a quantity in the second unit.

**DEFINE-UNIT**

This is used to define a new unit to the current state. The user is asked to type in the name of the new unit. Then user is asked to select the Mass, Length, Time, and Charge dimensions for the unit. In order to automatically define conversions between this unit and others, for each non-zero dimension of the unit, the user is asked to select one of the known one dimensional unit power units as an equivalent. If that dimension of the unit is not equivalent to any of those, then the user should select NONE-OF-THOSE. In this case the user is asked if that dimension converts to one of the basic units, and if so the user is prompted for the factor for conversion. For example the length dimension of miles converts to m by multiplication by 1609. After this is done for all non-zero dimensions of the unit to be defined, CalcLab automatically defines any possible
Conversions between this unit and any others. If one of the non-zero dimensions is not in terms of any known units, or does not convert to a basic unit, then CalcLab cannot automatically define conversions for the new unit being defined. If the user desires conversions that were not automatically defined, then this can be done using the DEFINE-CONVERSION main menu option.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE-EQUATION</td>
<td>This is used to remove an equation from the current state. All computations created from an equation prior to the deletion of the computation are unaffected. All variables used in the deleted equation are also unaffected. The user is asked to select the equation to delete.</td>
</tr>
<tr>
<td>DISPLAY-CONSTANT</td>
<td>This is used to display a constant. The user is asked to select the constant to display, after which the constant description string and value are displayed in the Display.</td>
</tr>
<tr>
<td>DISPLAY-TEMPLATE</td>
<td>This is used to display the variables that have been added to a frame template. The user is asked to select the frame template to display. The variables that have been added to the frame template are then displayed as a list in the Display.</td>
</tr>
<tr>
<td>LOAD-FILE</td>
<td>This is used to load a file which contains one or more states that the user wishes to use. The user is asked to type in the name of the file to be loaded. The user should not type in the .CL extension to the file name. It is assumed that the file to be loaded exists on the hard disk. If it does not it needs to be copied to the hard disk first. If a state is to be saved in a file that was made in a previous session, then that file should be loaded before the state to be saved is retained.</td>
</tr>
<tr>
<td>LOAD-STATE</td>
<td>This is used to load a retained state onto the current state. Retained states are those created by RETAIN-STATE or LOAD-FILE during the current session. The user is</td>
</tr>
</tbody>
</table>
first asked if the current state is to be saved first. Then the user is asked if the current state should be deleted before loading. Then the user is asked to select the state to be loaded. If the current state is deleted the loaded state becomes the current state. Otherwise the loaded state and the current state together become the current state. Note that it is possible to overwrite existing templates if the loaded state uses the same names for different templates.

MAKE-ALL-FILES
This is used to make files within which retained states can be stored for future use. Retained states are made using the RETAIN-STATE option. The program makes all files specified for retained states. Note that MAKE-ALL-FILES is the only way to secure retained states for future use.

MAKE-NEW-FRAME
This is used to make a frame in the Work Area. When a frame is made, instances of all the variables in the template it is made from are included in the frame. The user is asked to select the template from which the frame is to be made. Then the user is asked to indicate the location for the frame in the Work Area by clicking the left mouse button at that location.

MAKE-NEW-INSTANCE
This is used to make an instance in the Work Area. The user is asked to select the variable that this is an instance of. Then the user is asked to indicate, by clicking the left mouse button, where the instance is to be located in the Work Area. If an instance is made while the program is waiting for the user to select an instance for use in a computation, the instance created is automatically considered selected.

QUIT/RESTART
This option is used to terminate the current CalcLab session and start a new one. The user is asked if the current state needs to be retained before terminating the session.

REDRAW-ALL
This option is used to redraw the Work Area. This option is typically used to clear
73

up the Work Area after a lot of manipulation has been done and the graphics need redrawing.

This option is used if the current state is to be retained for future use. If it is intended to save this state for use in later sessions then a file in which the state should be stored should also be specified. The user is asked to type in what name the state should be retained under. Then the user is asked to indicate whether the instances, frames, and computations in the Work Area are also to be retained as part of the state. Then the user is asked if the state is to be saved in a file, in which case the user is asked to type in the file name (without .CL extension). Note that this does not actually save the state in the file. That should be done by selecting the MAKE-ALL-FILES option. If a state is to be saved in a file that was made previously, then the file should be loaded with LOAD-FILE before this option is used.

8.14 Saving and Loading from Files

If the user wishes to use states that have already been saved in a file, the user should load the file using the LOAD-FILE option. Then, the desired states may be individually loaded using the LOAD-STATE option. To save the current state in a file the user should use the RETAIN-STATE option, then use the MAKE-ALL-FILES option before concluding the session.

Files made by CalcLab are automatically given a .CL extension to their names to distinguish them from other names. The user should not specify this extension when using the RETAIN-STATE or LOAD-FILE options.
CalcLab works with files that reside on the workstation hard disk. To copy a file from floppy disk to the hard disk type

(COPYFILE '(FLOPPY)filename.CL '(DSK)filename.CL)

in the Interlisp-D Executive window. To copy a file from the hard disk to a floppy disk type

(COPYFILE '(DSK)filename.CL '(FLOPPY)filename.CL)

in the Interlisp-D Executive window.

8.15 Hardcopy

A hardcopy of any CalcLab window can be produced by clicking the right mouse button on the window and selecting Hardcopy from the pop-up menu. The hardcopy will be printed at the default printer.

8.16 Example

Consider the following problem:

Compare your average speed in the following two cases. (a) You walk 240 ft at a speed of 4.0 ft/s and then run 240 ft at a speed of 10 ft/s along a straight track. (b) You walk for 1.0 min at a speed of 4.0 ft/s and then run for 1.0 min at 10 ft/s along a straight track. [Resnick]
The user might wish to try this example in order to gain familiarity with CalcLab. In order to solve this problem the following sequence of steps may be taken:

1. Define a template for the action of traveling:
   
   Select ADD-NEW-TEMPLATE.
   When prompted type in the name travel.

2. Add the equation distance = speed * time, restricted to travel, and also add variables distance, speed and time to travel.

   Select ADD-NEW-EQUATION.
   Select travel as the template to restrict the equation to.
   When prompted for the LHS of the equation, type in distance.
   When prompted for the RHS of the equation, type in (speed * time).
   For each variable in the equation, when prompted to:
     Indicate if the variable has a standard unit of measurement-
     Select YES for each of distance, speed and time.
     Select m for distance and s for time.
     Select NEW-STANDARD-UNIT for speed.
   Type in the name of new unit-
     Type in m/s for speed.
   Select dimensions for new unit-
     Select MASS = 0, LENGTH = 1, TIME = -1, CHARGE = 0, for m/s.
     Select m for the LENGTH dimension and s for the TIME dimension of m/s.
   Make each variable a part of travel-
     Select YES.

3. Define the minutes, ft, and ft/s units:

   Select DEFINE-UNIT.
   When prompted type in min.
   Select MASS = 0, LENGTH = 0, TIME = 1, CHARGE = 0.
   Select NONE-OF-THESEx for the TIME dimension units.
   The time dimension converts to s.
   Type in 60 when prompted for the conversion factor.
   Conversion between min and s is automatically defined.

   Select DEFINE-UNIT.
When prompted type in ft.
Select MASS = 0, LENGTH = 1, TIME = 0, CHARGE = 0.
Select NONE-OF-THOSE for the LENGTH dimension units.
The length dimension converts to m.
Type in 0.3048 when prompted for the conversion factor.
Conversion between ft and m is automatically defined.

Select DEFINE-UNIT.
When prompted type in ft/s.
Select MASS = 0, LENGTH = 1, TIME = -1, CHARGE = 0.
Select ft for the LENGTH dimension units.
Select s for the TIME dimension units.
Conversion between ft/s and m/s is automatically defined.

4. Make three travel frames; one for the first stage, one for the second stage, and one for the overall travel:

Select MAKE-NEW-FRAME.
Select travel.
Position the frame in the work area.
Click the left mouse button on the label for the frame created and select RENAME.
Type in stage-1.
Select MAKE-NEW-FRAME.
Select travel.
Position the frame in the work area.
Click the left mouse button on the label for the frame created and select RENAME.
Type in stage-2.
Select MAKE-NEW-FRAME.
Select travel.
Position the frame in the work area.
Click the left mouse button on the label for the frame created and select RENAME.
Type in overall.

Figure 8.6 shows the Work Area after the three frames have been made. Note that the user will probably have different suffixes for the instance names, and that the locations of the frames in the work area depend on where the user places them.
5. Connect the overall travel to stage-1 travel and stage-2 travel using the relationships; overall distance = stage-1 distance + stage-2 distance, and overall time = stage-1 time + stage-2 time:

Click the left mouse button in the distance instance of overall. Select COMPUTE, then select NO for "using known equation?" When prompted type in (distance + distance). When prompted select the distance instance from stage-1, then the distance instance from stage-2. A computation is made linking the three distances.

Click the left mouse button in the time instance of overall. Select COMPUTE, then select NO for "using known equation?" When prompted type in (time + time). When prompted select the time instance from stage-1, then the time instance from stage-2. A computation is made linking the three times.

Figure 8.7 shows the Work Area after the frames are connected.
Figure 8.7 Work Area after the Frames are Connected

6. Solve part (a) of the problem by first inputting the necessary values, and then making the computations:

Click the left mouse button on the stage-1 distance.
Select INPUT-VALUE.
When prompted type in 240.
Select ft from the pop-up menu.
Click the left mouse button on the stage-2 distance.
Select INPUT-VALUE.
When prompted type in 240.
Select ft from the pop-up menu.
Click the left mouse button on the stage-1 speed.
Select INPUT-VALUE.
When prompted type in 4.
Select ft/s from the pop-up menu.
Click the left mouse button on the stage-2 speed.
Select INPUT-VALUE.
When prompted type in 10.
Select ft/s from the pop-up menu.
Click the left mouse button on the stage-1 time.
Select COMPUTE.
Select YES to "using known equation?"
The time for stage-1 will be automatically computed.
Click the left mouse button on the stage-2 time.
Select COMPUTE.
Select YES to "using known equation?"
The time for stage-2 will be automatically computed.
Click the left mouse button on overall distance.
Select CHANGE-UNITS.
Select ft from the pop-up menu.
Click the left mouse button on overall speed.
Select CHANGE-UNITS.
Select ft/s from the pop-up menu.
Click the left mouse button on the overall speed.
Select COMPUTE.
Select YES to "using known equation?"
The overall speed will be automatically computed.

This is the answer for part (a).

7. Solve part (b) of the problem by first inputting the necessary values, and then making computations:

Click the left mouse button on the stage-1 time.
Select INPUT-VALUE.
When prompted type in 1, then select min from the pop-up menu.
Click the left mouse button on the stage-2 time.
Select INPUT-VALUE.
When prompted type in 1, then select min from the pop-up menu.
Click the left mouse button on the stage-1 distance.
Select COMPUTE.
Select YES to "using known equation?"
The distance for stage-1 will be automatically computed.
Click the left mouse button on the stage-2 distance.
Select COMPUTE.
Select YES to "using known equation?"
The distance for stage-2 will be automatically computed.
The overall speed is automatically computed.

This is the answer for part (b).

Figure 8.8 shows what the Work Area looks like at this point.
Figure 8.8 Work Area after all Computations are made.
9. Future Extensions

CalcLab is an environment well suited for scientific numerical problem solving. It has many features which make it an effective tool for problem modeling and problem solving. It can, however, be extended in a number of ways, and outlined below are some possible enhancements that would increase its power and utility.

One of the limitations of CalcLab is the restricted syntax that is used to specify equations and expressions. One possible future enhancement, therefore, is the addition of an input parser to process a less restricted syntax for equations. The parser would translate the input to the Lisp form used internally by CalcLab. The program would then be able, for example, to directly input equations such as:

\[ W + U = 3 \times X + 2 \times Y + Z^{2 \times 2} \]

This would make the specification of equations easier and much more convenient than using the current Lisp-like expressions which require many parentheses and have a relatively more rigid syntax.

The usability of CalcLab could also be enhanced by the addition of general functions that operate on a variable number of quantities. These could include functions that compute the sum or average of a number of instances. The power of the program could be further increased by incorporating the ability to handle procedural concepts and conditionality. If this ability were available the user would be able to easily model more complicated situations. As an example the following information could then be directly specified.
IF airpspeed > 600 mi/hr
    THEN max-range = 0.5 * cfactor * fuel-amount
    noise = 4 * nfactor
ELSE  max-range = 0.9 * cfactor * fuel-amount
    noise = .003 * nfactor * airpspeed

Such an extension could be implemented by enhancing the notion of formulas to include sets of expressions, rather than just a single expression. When a formula is selected, the expressions for that formula would be executed in sequence. Some of these expressions could be statements that describe the conditions that must be satisfied before the rest of the expressions can be executed. The computation building functions would have to be modified to perform the necessary condition checking and sequentially process the expressions for the formula selected.

CalcLab currently has no facility for the specification for descriptive information to accompany the models that are built using it. Such information would be very useful in understanding models, especially by domain-naive users who might require some explanation of the models before using them. Therefore, another possible enhancement could be the expansion of the knowledge representation to include more descriptive information about the various entities. In addition, the ability to add graphical descriptive information would be useful since many problems in areas such as physics are easier to comprehend and model if some graphical description is available.

This kind of enhancement could be implemented by associating a bitmap with the current state. The user could "paint" a graphical description in the bitmap when building a model. When the state is retained the bitmap would be retained with it. When a state is loaded, the associated bitmap would be loaded with it and made
available for display to the user. Since the facilities for painting and manipulating bitmaps are already available in the underlying Interlisp-D system, such an enhancement would be relatively straightforward to implement.

Another limitation of CalcLab is the fixed size Work Area which limits the amount of information that can be clearly displayed at one time. This limitation can be overcome by allowing a much larger Work Area of which only a portion is visible at any time. The Work Area could be scrolled horizontally or vertically to see the area of interest. This ability could be still further enhanced by allowing the user to zoom out to see the "bigger picture" or zoom in to focus in on a particular section of the Work Area.

In order to implement these extended Work Area display facilities a graphics package would be required to generate the graphics. A window over the Work Area would be defined, and the position of the window would be changed to affect the scrolling action. The size of this window would be enlarged or reduced to produce the effects of zooming in or zooming out.

CalcLab does not allow the user to specify a range of allowable values for variables. In many problems, however, there is a definite range of allowable values for variables, and the addition of the ability to specify such ranges would be very helpful. In addition to more realistic modeling, this would help the user detect inconsistencies in models being built and could also be used to investigate impractical or impossible scenarios. The program could easily be modified to monitor the values of instances every time a new value is input or a new value is computed. If the new value were to fall outside the allowable range, the user would be alerted to this condition.
CalcLab lacks the ability to process or generate time sequences of data. In many practical problems the temporal behavior of some variables is significant, and therefore the ability to model time variable stimuli and generate time variable output is very useful. Thus, one possible future enhancement is the inclusion of a timing mechanism based on an internal clock. This mechanism could be used to model time variant situations and thus enable CalcLab to be used to solve a whole new range of problems.

One way to implement this enhancement would be to build an external driver capable of setting and monitoring the values of instances in CalcLab. The driver would generate time variant stimuli, as specified by the user, by setting a new value for the appropriate instances at each time step. The results of setting those values would also be monitored at each time step, and these could be displayed in either tabular or graphical form. No changes to the actual CalcLab program would be necessary since the driver would be external to CalcLab.

CalcLab allows the user to construct cycles and use relaxation as a method to solve problems. The program does not, however, actually detect any cycles that exist in the current state. These cycles may give rise to situations where the system starts oscillating, and does not converge to a solution in a reasonable amount of time. Therefore, the ability to detect such cycles and warn the user when oscillation occurs would be very useful. Analytical methods could be used to analyze the current state and determine if it is acyclic. If cycles are detected, further feedback analysis could be performed to determine whether the system is stable or not. If instability is possible, instead of using the relaxation method, analytical methods would then be applied to solve the problem.
CalcLab also has limitations on the complexity of equations it is algebraically able to manipulate. Consequently there are many equations from which CalcLab cannot derive the formulas for all the variables used. Hence, the ability to algebraically manipulate or solve more complex equations would prove very useful. In addition, CalcLab is limited to first order systems. But there are many problems which involve second order or higher order systems. The enhancement of CalcLab to be able to handle differential equations would therefore increase the power of the program considerably. One way to enable CalcLab to process more complex mathematical equations is to interface it to an existing mathematical symbolic manipulation package such as Macsyma. If this is done, whenever an equation that the program is unable to process itself is encountered, the appropriate routine in the symbolic manipulation package would be called to perform the required algebraic manipulation and mathematical derivation.

The ability to solve systems using numerical methods would be a useful enhancement too. When the program is unable to algebraically manipulate equations to construct computations, it could use these numerical techniques to produce solutions. The method of regula falsi and the Newton-Raphson method are some of the techniques that might be made available. Implementing this would require an interface to the routines for these numerical techniques. This interface would have to transform the problem into the appropriate form for the routines. It would also have to determine which of the various available methods would have to be applied to find a solution.

In conclusion, there are many enhancements that could be made to CalcLab. As a tool CalcLab has several significant features which make it very suitable for scientific numerical problem solving. By extending the program, the power and utility already available in the program can be applied to an even wider range of use.
Appendix A: Additional Example

This example further illustrates the use of CalcLab by describing how a set of related numerical problems are modeled and solved using the program. The problems involve satellites and rockets and are solved using a knowledge base of equations and frames.

Before the equations are added, two previously retained states must first be loaded. These are MECH-UNITS, which contains definitions and conversions for many commonly used units of mechanics, and CONSTANTS, which contains the definitions of some common constants such as G (universal gravitational constant), g (gravitational acceleration at the surface of the earth) and PI (22/7). The knowledge base contains a number of physical principles represented by equations that are not restricted to any template. These equations are:

\[
\text{force-attraction} = \left( \left( G \times \text{mass1} \right) \times \left( \text{mass2} / \left( \text{distance} \times \text{distance} \right) \right) \right) \quad \text{(A.1)}
\]

\[
\text{force-centripetal} = \left( \left( \text{mass} \times \left( \text{velocity} \times \text{velocity} \right) \right) / \text{orbit-radius} \right) \quad \text{(A.2)}
\]

\[
\text{velocity} \times \text{period} = \left( 2 \times \left( \text{PI} \times \text{orbit-radius} \right) \right) \quad \text{(A.3)}
\]

\[
\text{weight-on-earth} = \left( \text{mass} \times g \right) \quad \text{(A.4)}
\]

\[
\text{escape-velocity} = \left( \sqrt{2 \times \left( G \times \left( \text{planet-mass} / \text{orbit-radius} \right) \right)} \right) \quad \text{(A.5)}
\]
acceleration = \left( \frac{\text{final-velocity} - \text{initial-velocity}}{\text{time}} \right) \quad (A.6)

\text{force} = (\text{mass} \ast \text{acceleration}) \quad (A.7)

After these equations are added, a "satellite" template is added, and to the satellite template are added the variables mass, orbit-radius, velocity and period. Then a "rocket" template is added and to it is restricted the added equation:

\text{total-mass} = (\text{payload} + \text{mass}) \quad (A.8)

with total-mass, payload, and mass added to the template when the equation is added. Also added to the rocket template are the variables weight-on-earth and the new variable max-thrust.

When the equations are added, the following are selected as the standard units of measurements for the variables indicated:

N for force-attraction, force-centripetal, force, weight-on-earth and max-thrust.

kg for mass1, mass2, mass, planet-mass, total-mass and payload.

m for distance and orbit-radius.

m/s for velocity, escape-velocity, final-velocity and initial-velocity.

s for period and time.

m/s² for acceleration.
Note that certain approximations and assumptions are made, some implicitly, in modeling the problems to follow. For example, it is assumed that all satellites have circular orbits, and that the mass of a rocket is constant during the period of interest, with the max-thrust continuously available during this time. Also the energy losses to friction etc. are considered negligible.

All computations made in the following description are made using known equations. The astronomical data used was obtained from [Resnick].

\begin{table}
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{Nature} & \textbf{Value} \\
\hline
\text{Earth's Mass} & 5.98E24 kg \\
\text{Earth's Diameter} & 149E6 km \\
\hline
\end{tabular}
\caption{Earth's Physical Properties}
\end{table}

The first problem is to find the period of the earth as it revolves around the sun. Since the earth can be viewed as a satellite of the sun, a new satellite frame to model the earth is made. This frame is renamed Earth. Then 5.98E24 kg and 149E6 km are input for the mass
and orbit-radius of Earth. Next a new instance of mass is made, to model the sun, and it is renamed Sun-mass. 1.9707E30 kg is input for this. The resulting Work Area is shown in Figure A.1. To model the fact that there is gravitational attraction between the earth and sun, a new force-attraction instance is made. A computation for this instance is initiated, and since there is only one applicable equation, it is automatically selected. When prompted for mass1, Earth mass is selected, and when prompted for mass2, Sun-mass is selected. When prompted for distance, Earth orbit radius is selected.

![Figure A.2](image)

The unit of display of the Earth period is changed to days, and a computation initiated for Earth period. The equation and instances required for this are automatically selected. But the Earth velocity, which is used in this computation is not yet computed, so a computation is initiated for Earth velocity. There is a choice of equations here, and
Equation A.2 is selected. When prompted for an instance of force-centripetal, force-attraction is selected since these forces are equivalent for orbit to be maintained. The other instances are selected automatically, and the desired result is produced. The Work Area at this stage is shown in Figure A.2.

The next problem is to find the period of the moon around the earth. To do this a new satellite frame is made, and renamed Moon. Moon mass is 7.347E22 kg, and Moon orbit-radius is 38E4 km, and these values are input. Since the problem of the moon orbiting the earth is similar to the problem of the earth orbiting the sun, the problem is modeled in the same way. A force-attraction instance is made, and a computation initiated for it. The only applicable equation is automatically selected, and when prompted for mass1, mass2 and distance, Moon mass, Earth mass, and Moon orbit-radius respectively are selected.
The unit of Moon period is changed to days, and a computation for period initiated. The equation and instances to use are automatically selected. Next a computation for Moon velocity is initiated, using Equation A.2, and when prompted for force-centripetal the newly made force-attraction is selected. The resulting Work Area, with both problems modeled simultaneously, is shown in Figure A.3.

The third problem is to find the period of a satellite "Satellite-A," of mass 1000 kg in a orbit of radius 5000 miles around the earth. A simple way to model this situation is to replace the data for the moon by the data for Satellite-A. To do this, first rename Moon to Satellite-A, then input 1000 kg and 5000 mi for the mass and orbit-radius of Satellite-A. The unit of Satellite-A period is changed to mins, and thus, using the model developed for the previous problem, this problem is
modeled and solved as shown in Figure A.4.

The next problem involves a "X-rocket," which weighs 40000 lb on earth, and has max-thrust 10000 lb. This rocket is to be used to pull Satellite-A out of the earth’s gravitational field. The problem is to find the required burn time for the rocket, which is assumed to be already in orbit with the satellite.

First, a new rocket frame is made, renamed X-rocket, and 40000 lb and 10000 lb input for its weight-on-earth and max-thrust. A computation is initiated for X-rocket mass. Equation A.4 is used with required instances selected automatically. Then X-rocket total-mass is computed, with automatic selection of the equation and instances. Next the X-rocket payload is constrained to use the Satellite-A mass. The Work Area at this point is shown in Figure A.5.
The problem involves finding the burn time, so an instance of time is made, and its unit changed to mins. A computation is initiated for it, with the only applicable equation. When prompted for acceleration, a new instance of acceleration is made. When prompted for final-velocity, a new instance of escape-velocity is made. When prompted for initial-velocity, Satellite-A velocity is selected.

Next, acceleration must be computed, using Equation A.7, and when prompted for force and mass, X-rocket max-thrust and X-rocket total-mass respectively are selected. After this escape velocity is computed, with the only applicable equation. When prompted for planet-mass, Earth mass is selected and when prompted for orbit-radius, Satellite-A orbit-radius selected. The required burn time is now available as shown in Figure A.6.
From Figure A.6, it may be noticed that the Work Area is becoming quite congested. If it is now known that the only variable entity in future problems is Satellite-A, and only the burn times for the rocket have to be found, everything except Satellite-A and time may be shrunk, and the various instances and frames moved to conserve Work Area space. The result of doing this is shown in Figure A.7.

Finally, the new specifications for Satellite-A of mass 4000 kg and orbit-radius 4500 miles are made. These values are input for Satellite-A, and the required burn time automatically computed as shown in Figure A.8.
Appendix B: CalcLab GLISP Declarations
CALCULATE DECLARATIONS

(GLISOBJECTS COMPUTATION)
(GLISOBJECTS CONSTANT)
(GLISOBJECTS EQUATION)
(GLISOBJECTS FORMULA)
(GLISOBJECTS FRAME)
(GLISOBJECTS FRAMEDEF)
(GLISOBJECTS INSTANCE)
(GLISOBJECTS LINK)
(GLISOBJECTS UNIT)
(GLISOBJECTS VARIABLE)
(GLISCONSTANTS COMPREHEND INDEX WIDTH DEFAULT ERROR INSTANCE HEIGHT INSTANCE WIDTH MAXCHARS RESER WORDSH SWINGHEIGHT WINDOWHEIGHT WINDOWSOURCE WINDOWWIDTH)
(GLISGLOBAL ALLCOMPUTATIONS ALLCONSTANTS ALLOCCATIONS ALLFORMULAS ALLFRAMES ALLFRAMEDEF

(* CalcLab GLISP Declarations)

(GLISOBJECTS COMPUTATION)

(LIST (INSTANCE COMPUTED ATOM)
(INSTANCES NEEDED (LISTOF ATOM))
(EXPRESSION ANYTHING)
(X INTEGER)
(Y INTEGER)
(DX INTEGER)
(DY INTEGER)
(LINKLIST (LISTOF ATOM))
(NAME ATOM)
(EQUATIONSTRING STRING)
(CONSTANTS (LISTOF ATOM)))

DOC (* A computation is a procedure used to evaluate a particular instance)
(INSTANCE COMPUTED is the name of the instance whose value is computed)
(INSTANCES NEEDED is a list of the names of instances that are needed in the computation)
(EXPRESSION is a Lisp expression in terms of the instances needed, that evaluates to the value to be computed)
(X,Y are the x,y coordinates of the box for the computation, with width DX and height DY)
(LINKLIST is a list of the names of all links to the box)
(NAME is the name of the computation)
(EQUATIONSTRING is the equation on which this computation is based)
(LIST (INSTANCES (LISTOF ATOM))
  (X INTEGER)
  (Y INTEGER)
  (DX INTEGER)
  (DY INTEGER)
  (NAME ATOM)
  (EXPANDED BOOLEAN)
  (FRAMEDEF ATOM))

DOC (* (A frame is a collection of instances related to each other)
   (INSTANCES is a list of the names of all instances in the frame)
   (X, Y are the coordinates for the box for the frame, with width DX and height DY)
   (NAME is the name of the frame)
   (EXPANDED is true if the frame is expanded)
   (FRAMEDEF is the name of the frame definition for the frame)
   (DRAW draws the box for the frame))
)

[train DRAWFRAME])
)

[GLISOBJECTS]

(FRAMEDEF

(LIST (VARIABLES (LISTOF ATOM))
  (FRAMES (LISTOF ATOM)))

DOC (* (A frame definition is a frame definition, or a template used for creating frames)
   (VARIABLES is a list of the names of variables used in the definition)
   (FRAMES is a list of the names of frames created from the definition))
)

[GLISOBJECTS]

(INSTANCE

(LIST (VALUE NUMBER)
  (X INTEGER)
  (Y INTEGER)
  (DX INTEGER)
  (DY INTEGER)
  (LINKSLIST (LISTOF ATOM))
  (COMPUTATIONS (LISTOF ATOM))
  (USEDBY (LISTOF ATOM))
  (VARIABLETYPE ATOM)
  (STANDARDUNIT ATOM)
  (DISPLAYUNIT ATOM)
  (DISPLAYFACTOR NUMBER)
  (NAME ATOM)
  (FRAME ATOM)
  (EXPANDED BOOLEAN))

DOC (* (VALUE is the value of the instance)
   (X and Y are the x,y coordinates of the display box for the instance)
   (DX and DY are the width and height of the box for the instance)
   (LINKSLIST is a list of the names of all links to the box)
   (COMPUTATIONS is a list of the names of computations that are used to compute the value
    of the instance)
   (USEDBY is a list of the names of the computations that use the instance to compute the
    values of other instances)
   (VARIABLETYPE is the variable that this is an instance of)
   (STANDARDUNIT is the standard unit of measurement for the instance)
   (DISPLAYUNIT is the unit that the instance is to be displayed in)
   (DISPLAYFACTOR is the number the DISPLAYUNIT must be multiplied by to convert to the
    STANDARDUNIT)
   (NAME is the name of the instance)
   (FRAME is the name of the frame this instance is a part of)}


(PSEUDO-DISK)CALCULATE.LSP:1  8-Jul-87 15:38:56

(EXPAND) is true if the instance is expanded
(COMPUTE) causes the evaluation of the instance
(DISPLAY) displays the value of the instance
(REDisplay) displays the instance with different units
(INPUTVALUE) inputs a value for the instance
(SETVALUE) sets the value of the instance
(SELECT (CAUSES) selection of this instance
(CREATECOMPUTATION) creates a computation for this instance
(DRAW) draws the box for the instance
(DELETE) deletes the instance
(MOVE) moves the box for the instance
(SHRINK) shrinks the box for the instance
(EXPAND) expands the box for the instance
(REDRAW) redraws the instance

MSG

{{COMPUTE COMPUTEINSTANCE}
(DISPLAY DISPLAYINSTANCE)
(REDisplay REDisplay)
(INPUTVALUE INPUTVALUE)
(SETVALUE SETVALUE)
(SELECT SELECTINSTANCE)
(CREATECOMPUTATION CREATECOMPUTATION)
(CONSTRUCTCOMPUTATION CONSTRUCTCOMPUTATION)
(DRAW DRAWINSTANCE)
(DELETE DELETEINSTANCE)
(MOVE MOVEINSTANCE)
(SHRINK SHRINKINSTANCE)
(EXPAND EXPANDINSTANCE)
(REDRAW REDRAWINSTANCE) } }

[GLISPGOBJECTS]

(LINK)

(List (FROM INTEGER)
(TO INTEGER)
(INFORMATION) (ATOMIC) (COMPUTATION))

DOC

(\* (A link is a line connecting the box of an instance to the box of a computation)
(FROM X INTEGER)
(TO INTEGER)
(INFORMATION) (ATOMIC) (COMPUTATION)
(DRAW DRAWLINK)
(ERASE ERASELINK)
(DELETE DELETELINK) )

[GLISPGOBJECTS]

(UNIT)

(List (DIMENSIONS (LIST OF INTEGER))
(CONVERSIONS (LIST OF ANYTHING))
(MSCC NUMBER))

DOC

(\* (A unit is a unit of measurement for some variable (s))
(DIMENSIONS is the dimension vector for the unit. It has the form (MASS LENGTH TIME CHARGE))
(CONVERSIONS is a list of conversions for the unit. For each unit that this unit can
convert CONVERSIONS contains a list of the name of that unit and the number by which this unit must be multiplied to convert it to MKSC units))

\}

\)

[GLISPOBJECTS]

[GLISPVARIABLE]

(LIST (FORMULAS (LISTOF ATOM))
  (STANDARDUNIT ATOM))

DOC ("The value of an instance of a variable can be computed using formulas
  (FORMULAS is a list of the names of the formulas that can be used)
  (STANDARDUNIT is the standard unit of measurement for the variable)"

\]

[GLISPCONSTANTS]

(COMPOPTIONS 15 INTEGER )

(COMPWIDTH 80 INTEGER )

(DEFAULTERROR .00005 NUMBER )

(.GetInstanceHeight 30 INTEGER )

(InstanceWidth 110 INTEGER )

(MAXCHARS 15 INTEGER )

(RESERVEDWORDS (QUOTE (ALLCOMPUTATIONS ALLCONSTANTS ALLEQUATIONS ALLFORMULAS ALLFRAMES ALLFRDEFS
  ALLOVARS ALLINSTANCES ALLLINKS ALLOWEDERROR ALLSTATES
  ALLUNITS ALLVARS BasicUnit CheckUnit Compute-Error
  Constant ConstName FillFilename Filename Frame
  FrameDef FrameName Frame-CHosen FrameName Instance
  Instance-Name InstName ItemName LHS ListName Loadlist
  LoopItem LoopVariable MenuList NewFormula NewName
  New-Standard-UNIT New-VARIABLE Nil None-of-These
  Not-Restricted RHS SelectedInstance StanUnit StanUnits
  StanUnit2 StanName T TestFrame TestInstance Unit Unit2
  UnitName Var-CHosen Variable NAME VarName D.W
  M.W P.W W.W)) (LISTOF ATOM) )

(SHRUNKHEIGHT 15 INTEGER )

(WINDOWHEIGHT 650 INTEGER )

(WINDOWNORIGIN 0 INTEGER )

(WINDOWWIDTH 1024 INTEGER )

[GLISPGLOBALS]

(ALLCOMPUTATIONS (LISTOF ATOM) )

(ALLCONSTANTS (LISTOF ATOM) )

(ALLEQUATIONS (LISTOF ATOM) )

(ALLFORMULAS (LISTOF ATOM) )

(ALLFRAMES (LISTOF ATOM) )

(ALLFRDEFS (LISTOF ATOM) )
(PSEUDO-DSK)CALCLABDEC.LSP:1 8-Jul-87 15:38:58

(ALLGVARS   (LISTOF ATOM)   )
(ALLINSTANCES   (LISTOF ATOM)   )
(ALLLINKS   (LISTOF ATOM)   )
(ALLOWEDEERROR   NUMBER   )
(ALLSTATES   (LISTOF ATOM)   )
(ALLUNITS   (LISTOF ATOM)   )
(ALLVARS   (LISTOF ATOM)   )
(FILESTOMAKE   (LISTOF ATOM)   )
(SELECTEDINSTANCE   INSTANCE   )

(* (ALLCOMPUTATIONS is a list of the names of all known computations) (ALLCONSTANTS is a list of the names of all known constants) (ALLEQUATIONS is a list of the names of all equations) (ALLFORMULAS is a list of the names of all known formulas) (ALLFRAMES is a list of the names of all frames) (ALLFRAMEDEFs is a list of the names of all frame definitions) (ALLGVARS is a list of the names of all the globally available variables) (ALLINSTANCES is a list of the names of all known instances) (ALLLINKS is a list of the names of all known links) (ALLOWEDEERROR is the amount of a value that is considered insignificant for converging values of instances) (ALLSTATES is a list of the names of all known states) (ALLUNITS is a list of all known standard units) (ALLVARS is a list of the names of all variables known) (FILESTOMAKE is a list of the names of files that have not yet been made) (SELECTEDINSTANCE is the instance selected by the user))

(PUTPROPS CALCLABDEC.LSP COPYRIGHT ("YUSUF MAULADAD" 1987))
(DECLARE: DONTCOPY
t (FILEMAP (NIL)))
STOP
Appendix C: CalcLab GLISP Functions
(* Copyright (c) 1987 by Yuzuf Maulud. All rights reserved.

(PRESETCOMPRINT CALCLABCONS)

(RPAQ) CALCLABCONS (* CalcLab GLISP Functions)

(FNS ADEQUATION ADTOFRAME ADDUNIT ADDEQUATION AUTOCONVERSION BUILDCOMPUTATION CALCLAB CHANGEERROR
 CHOOSEINSTANCE COMPATIBLE COMPUTE-ANTLOG COMPUTE-ARCCOS COMPUTE-ARCSIN COMPUTE-ARCTAN
 COMPUTE-COS COMPUTE-DIFFERENCE COMPUTE-ERROR COMPUTE-EXPT COMPUTE-FORMULA COMPUTE-INIT
 COMPUTE-LOG COMPUTE-MENU COMPUTE-MENUS COMPUTE-PLUS COMPUTE-QUOTIENT COMPUTE-SQRT COMPUTE-SIN
 COMPUTE-TAN COMPUTE-TIMES COMPUTECTION COMPUTEUNIT COMPUTEUNIT COMPUTERATE CONSTRUCTCOMPUTATION
 CONTAINMP CREATECOMPUTATION CREATEINSTANCE DEFINECONSTANT DEFINECONVERSION DEFINEFRAME
 DELETECOMPUTATION DELETEFRAME DELETEINSTANCE DELETELINK DISPLAYCONSTANTS
 DISPLAYFRAME DISPLAYINSTANCE DRAWALL DRAWBOX DRAWCOMPUTATION
 DRAWFRAME DRAWINSTANCE DRAWLINK EQUATEINSTANCE ERASELINK EXPANDFRAME EXPANDINSTANCE
 EXTRACTCONSTANTS EXTRACTVARS FINDUNIT INPUTVALUE INSIDEBOX LOADSTATE LOADSTATEFILE
 MAKEALLFILES MAKECOMPUTATION MAKEFRAME MAKEINSTANCE MAKESTATEFILE MOVECOMPUTATION
 MOVEFRAME MOVEINSTANCE PRINTINO PRINTMP PROCESSDIEMENSION PROGRESSUNIT PROMPTREAD
 REDisplay REINITIALIZE RENAMEFRAME REMAKEINSTANCE SAVESTATE SELECTINSTANCE
 SELECTUNIT SVALUE SHINKFRAME SHINKINSTACE SIMP-ANTLOG SIMP-ARCCOS SIMP-ARCT
 SIMP-COS SIMP-DIFF SIMP-ERROR SIMP-EXPT SIMP-FOO SIMP-INIT SIMP-LOG
 SIMP-MINUS SIMP-PLUS SIMP-SIN SIMP-SQRT SIMP-TAN SIMP-TIMES SIMPIFY SUBSTFIRST
 TESTANDEVAL TESTFORSELECTION TRANSLATE UNITS-INIT YESNO MENU)

(* CalcLab GLISP Functions)

(DEFINE)

(ADDEQUATION
 [GLAMBDA NIL

 (* edited: "8-Jul-87 15:11")
 (* Adds an equation to those already known)
 (PROG (EQUATIONLHS EQUATIONVARS NEWFORMULA EXISTINGVAR VARIABLE STANUNIT LHS RHS EQUATIONCONSTS
 MENULIST FREDCHOSEN FRED:FRAMEDEF EQNAME FORMULALIST FORMULAXPRESSION)
 (* EQUATIONLHS = LHS is an equivalent form of the the equation added; EQUATIONVARS are the
 variables used in the equation; NEWFORMULA is the name of the new formula created for each
 variable in EQUATIONVARS; EXISTINGVAR is a variable that already exists; STANUNIT is the
 standard unit of measurement for the variable; LHS and RHS are the LHS and RHS of the equation
 (EQUATIONCONSTS are the known constants used in the equation); MENULIST is a list of menu
 options; FREDCHOSEN is the name of the frame template that this equation belongs to; FRED:FRAMEDEF
 is the frame template chosen; EQNAME is the name of the new equation; FORMULALIST is a list of
 all formulas derived from the new equation; FORMULAXPRESSION is the expression derived
 for a formula)

 (PRNTMP "Adding equation ...")
 [FOR FREDFRAME IN ALLFREDIFS (MENULIST ++
 (LIST FREDFRAME (LIST (QUOTE QUOTE)
 (QUOTE FREDFRAME))
 (MENU -
 (LIST (QUOTE NOT-RESTRICTED)
 (NL))
 (FREDCHOSEN - (MENU (CREATE MENU (TITLE "Restrict to which template?"))
 (ITEMS MENULIST)
 (MENUPOSITION - (QUOTE (700 . 600))
 (IF FREDCHOSEN (FRED = (EVAL FREDCHOSEN)))
 (LHS = (PROMPTREAD "What is the LHS (left hand side) of the equation??")
 (RHS = (PROMPTREAD "What is the RHS (right hand side) of the equation??")
 (EQUATIONLHS = (LIST (QUOTE DIFFERENCE)
 (TRANSLATE LHS)
 (TRANSLATE RHS))
 (* LHS = RHS translates to EQUATIONLHS = (DIFFERENCE LHS RHS) = 0)
 (EQUATIONCONSTS = (EXTRACTCONSTANTS EQUATIONLHS))
 (EQUATIONVARS = (EXTRACTVARS EQUATIONLHS))
 (FOR CONNAME IN EQUATIONCONSTS (EQUATIONVARS --
 (QUOTE CONNAME))
 (QUOTE EQUATIONCONSTS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUATIONVARS))
 (QUOTE EQUATIONLHS))
 (QUOTE EQUA...
(FOR LOOPVARIABLE IN EQUATIONVARS
  (FOR EACH VARIABLE IN EQUATIONVARS
    (SIMP (COMPUTE-FORMULA LOOPVARIABLE EQUATIONVARS 0))
  (IF (CONTAINS (QUOTE COMPUTE-ERROR)
    (NEWFORMULA = NIL)
  ELSE (NEWFORMULA = (GENSYM (QUOTE FORMULA)))
    (* Make a new formula)
  (FORMULALIST =)
  (SET NEWFORMULA (A FORMULA WITH VARIABLES =
    (REMOVE LOOPVARIABLE EQUATIONVARS)
    EXPRESSION = FORMULALEXPRESSION
    EQUATIONSTRING =
    (CONCAT LHS "=" RHS)
    CONSTANTS = EQUATIONCONS
    FRAMEDEF = FRDEFCHosen
    FORVARIABLE = LOOPVARIABLE)
  (ALLFORMULAS =)
  (NEWFORMULA)
  (IF (MEMBER LOOPVARIABLE ALLVARS)
    (* This variable already known)
  (IF (AND FRDEFCHosen (NOT (MEMBER LOOPVARIABLE
     FRDEF:VARIABLES))
    (YESNOMENU (CONCAT *Make " LOOPVARIABLE
    " a part of " FRDEFCHosen " frames?"
    (QUOTE (280 . 170)
    (FRDEF:VARIABLES = LOOPVARIABLE)
    ELSE (IF (NOT (MEMBER LOOPVARIABLE ALLVARS))
    (ALLVARS = LOOPVARIABLE))
  (EXISTINGVAR = (EVAL LOOPVARIABLE))
  (IF NEWFORMULA (EXISTINGVAR:FORMULAS =)
    (NEWFORMULA)
  ELSE (* Make new variable)
    (STANUNIT = NIL)
  (IF (YESNOMENU (CONCAT
    "Is there a standard measurement unit for "
    LOOPVARIABLE "?"
    (QUOTE (280 . 170))
    (STANUNIT = (SELECTSTANUNIT LOOPVARIABLE))
  (IF NEWFORMULA (SET LOOPVARIABLE
    (A VARIABLE WITH FORMULAS =
    (LIST NEWFORMULA)
    STANDARDUNIT = STANUNIT))
  ELSE (SET LOOPVARIABLE
    (A VARIABLE WITH STANDARDUNIT =
    STANUNIT))
  (ALLVARS = LOOPVARIABLE)
  (IF (AND FRDEFCHosen (NOT (MEMBER LOOPVARIABLE
    FRDEF:VARIABLES))
    (YESNOMENU (CONCAT *Make "
    LOOPVARIABLE
    " a part of " FRDEFCHosen
    " frames?"
    (QUOTE (280 . 170)
    (FRDEF:VARIABLES = LOOPVARIABLE)
  ELSE (IF (NOT (MEMBER LOOPVARIABLE ALLVARS))
  (ALLVARS = LOOPVARIABLE)
  (SIMP (GENSYM (QUOTE EQUATION)))
  (ALLEGATIONS =)
  (SET EQUATION (AN EQUATION WITH EQUATIONSTRING =
    (CONCAT LHS "=" RHS)
    FORMULAS = FORMULALIST))
  (PRINTNP (IF FRDEFCHosen (CONCAT LHS "=" RHS " added for " FRDEFCHosen)
  ELSE (CONCAT LHS "=" RHS " added")

  (ADDTOFRAME
(GLAMBA NIL (* edited: "8-Jul-87:14:36")
("Add a variable to the frame template")
(NEWLIST FREDCHOOSE VARCHOOSE FRDEF:FRAME DEF STANUNIT)
(* (NEWLIST is the list of menu items) (FREDCHOOSE is the name of frame definition chosen)
(VARCHOOSE is the variable chosen) (FRDEF is the frame definition chosen) (STANUNIT is the standard unit of measurement of any new variable added))

(PRINTNP "Adding variable to template ")
[FOR FRDEFNAME IN ALLFRDEFS (NEWLIST ++
 (LIST FRDEFNAME (LIST (QUOTE SETQ)
 (QUOTE FRDEFCHOOSE)
 (LIST (QUOTE FRDEFCHOOSE)
 (FRDEFNAME))

[IF NEWLIST (MENU (CREATE MENU (TITLE = "Add to which template?")
 (ITEMS = NEWLIST)
 (MENUPOSITION = (QUOTE (520 . 800)
 (IF (FREDCHOOSE (NEWLIST = NIL)
 [FOR VARCHOOSE
 IN (SORT ALLVARS)
 (NEWLIST ++
 (LIST VARCHOOSE (LIST (QUOTE SETQ)
 (VARCHOOSE))
 (NEWLIST ++
 (LIST (QUOTE NEW-VARIABLE)
 (VARCHOOSE))
 [VARCHOOSE = (MENU (CREATE MENU (TITLE = "Add which variable?")
 (ITEMS = NEWLIST)
 (MENUPOSITION = (QUOTE (520 . 800)
 (IF (NOT VARCHOOSE)
 (VARCHOOSE = (PROMPTREAD "What is the name of the new variable?")
 (IF (YENOMENU (CONCAT "Is there a standard measurement unit for 
 VARCHOOSE ?")
 (QUOTE (520 . 180)))
 (STANUNIT = (SELECTSTANUNIT VARCHOOSE)))
 (SET VARCHOOSE (A VARIABLE WITH STANDARDUNIT = STANUNIT))
 (ALLVARS ++
 VARCHOOSE))
 (IF (AND VARCHOOSE FREDCHOOSE)
 (FRDEF = (EVAL FREDCHOOSE))
 (IF (MEMBER VARCHOOSE FRDEF:VARIABLES)
 (PRINTNP "--- already added ---")
 ELSE (FRDEF:VARIABLES ++
 VARCHOOSE)
 (PRINTNP (CONCAT VARCHOOSE * added to * FREDCHOOSE))
 ELSE (PRINTNP "--- bad selection ---"))

(ADDUNIT (GLAMBA (VARIABLENAME) (* edited: "8-Jul-87:15:01")
(* Adds a new unit to those already known. If VARIABLENAME is the name of the variable for which this is being added, if called from SELECTSTANUNIT. If VARIABLENAME is not NIL, then the name of the unit added is returned.)

(PROG (UNITNAME DIMENSIONVECTOR) (* (UNITNAME is the name of the unit to be added)
 (DIMENSIONVECTOR is the dimension vector for the unit to be added))

(PRINTNP "* Defining unit ")
 (UNITNAME = (PROMPTREAD (IF VARIABLENAME (CONCAT
 "What is the unit to be defined for "
 VARIABLENAME * ?")
 ELSE "What is the unit to be defined?")
 (IF (AND (NOT VARIABLENAME)
 (MEMBER UNITNAME ALLUNITS)
 (PRINTNP (CONCAT "*** UNITNAME * already exists ***")
 ELSE VARIABLENAME (WHILE (MEMBER UNITNAME ALLUNITS)
 (PRINTNP (CONCAT "*** UNITNAME * already exists ***")
 (UNITNAME = (PROMPTREAD "What is the unit to be defined?")))
 (IF (NOT (MEMBER UNITNAME ALLUNITS)
 [FOR DIMENSION
IN (QUOTE (MASS LENGTH TIME CHARGE))
(DIMENSIONVECTOR =>
(MENU (CREATE MENU (TITLE =>
(CONCAT
"Select dimensions of "
DIMENSION " for "
UNITNAME))
(ITEMS => (QUOTE ((-3 -3)
(-2 -2)
(-1 -1)
(0 0)
(1 1)
(2 2)
(3 3)
)
(MENUROWS = 1)
(CENTERFLAG = 1)
(MENUBORDERIZE = 1)
(MENUPosition = (QUOTE (240 . 170)))
(ITEMWIDTH = 40)
(PRINTMP " Warning! Dimensionless unit")
(PROGRESSUNIT UNITLEN DIMENSIONVECTOR)
(# VARIABlename (RETURN UNITNAME))
))

(AUTOCONVERSION
(GLAMBA (UNIT1)
(* edited: "4-Jun-8722-40")
(* This function automatically defines conversions
between the unit named UNIT1 and all other units which
have the same dimension vector)
(* (UNITOBJ1 is the unit named UNIT1)
(UNITOBJ2 is the unit named UNIT2))

(PROG (UNITOBJ1:UNIT UNITOBJ2:UNIT)

(UNITOBJ1 = (EVAL UNIT1))
(FOR UNIT2 IN ALLUNITS (UNITOBJ2 = (EVAL UNIT2))
(AND (UNITOBJ1:MKSCCF (UNITOBJ2:MKSCCF))

(PRINTMP (CONCAT " Conversion defined between "
UNIT1 " and "
UNIT2))
))}

(BUILDCOMPUTATION
(GLAMBA (FORINST EXPRCOMP EXPRLIST EXPRCONS EXPRSTRING XPOS YPOS)
(* edited: "17-Feb-8720.39")
(* Builds a computation for the instance FORINST using the expression EXPRCOMP which is in terms of instances
EXPRLST and constants EXPRCONS. EXPRSTRING is a string for the expression and XPOS, YPOS are the coordinates
where the computation box is drawn)

(PROG (COMPNAME COMP:COMPUTATION LINKNAME LM:LINK INST:INSTANCE)
(* (COMPNAME is the name of the computation created) (COMP is the actual computation) (LINKNAME is the name of a
link) (LINK is the link) (INST is an instance))

(COMPNAME = (GENSYM "-COMP"))
(SET COMPNAME
(A COMPUTATION WITH INSTANCEDOURED = FORINST NAME = COMPNAME X = XPOS Y = YPOS
DX = COMPPATH DY = COMPPATH constants = EXPRCONS EQUATIONSTRING =
EXPRSTRING INSTANCEDOURED = EXPRCONS EXPRESSION = EXPRCOMP))
(ALLCOMPUTATIONS =>
COMPNAME)
(COMP = (EVAL COMPNAME))
(SEND COMP DRAW)
(FOR INSTNAME IN EXPRLIST (INST = (EVAL INSTNAME))
(INST:USEDBY =>
COMPNAME))

(FOR INSTNAME}
(CALCLAB (GLAMBDA NIL)
  (COPYRIGHTFLG = NIL)
  (KEYACTION (QUOTE LOCK))
  (QUOTE (LOCKTOGGLE . LOCKTOGGLE)))
  (DEFAULTTYPREGION = (QUOTE (200 10 800 180)))
  (ALLVARS = NIL)
  (ALLUNITS = NIL)
  (ALLINSTANCES = NIL)
  (ALLCOMPUTATIONS = NIL)
  (ALLFORMULAS = NIL)
  (ALLVARS = NIL)
  (ALLCONSTANTS = NIL)
  (ALLDEFINED = NIL)
  (ALLVARIABLES = NIL)
  (CLOSEDERROR = DEFAULTERROR)
  (CLOSEDSTATES = NIL)
  (FILESTATES = NIL)
  (COMPUTERUN) (* initialises COMPUTEUN properties)
  (SIMPLIFYUN) (* initialises SIMPLIFYUN properties)
  (UNITS-UNIT)
  (M.W = (CREATEW (CREATEREGION WINDOWORIGIN 0 WINDOWWIDTH WINDOWHEIGHT)
                   "CalcLab Work Area")
           "CalcLab Work Area")
  (M.W = (CREATEW (CREATEREGION WINDOWORIGIN (00 + WINDOWHEIGHT)
                   WINDOWWIDTH 601)
           "CalcLab Main Menu")
           "CalcLab Main Menu")
  (P.W = (CREATEW (CREATEREGION WINDOWORIGIN (151 + WINDOWHEIGHT)
                   (WINDOWWIDTH / 2)
                   195))
           "CalcLab Status and Prompts")
           "CalcLab Status and Prompts")
  (DSPSCROLL (QUOTE ON))
  (D.W = (CREATEW (CREATEREGION (WINDOWWIDTH / 2)
                   (WINDOWHEIGHT / 2)
                   105))
           "CalcLab Display")
           "CalcLab Display")
  (DSPSCROLL (QUOTE ON))
  (D.W)
  (PRINTER (CONCAT "CalcLab: You are at: " 8 July 1987")
  (WINDOWPROP D.W (QUOTE BUTTONEVENTFN))
  (QUOTE TESTFORSELECT))
  (ATTACHWINDOW M.W D.W (QUOTE TOP))
  (QUOTE CENTER))
  (ATTACHWINDOW P.W D.W (QUOTE TOP))
  (QUOTE LEFT))
  (ATTACHWINDOW D.W D.W (QUOTE TOP))
  (QUOTE RIGHT))
  (ADDMENU (CREATE MENU [ITEMS = (QUOTE [[QUIT/RESTART (REINITIALIZE)
                                   (REDRAW-ALL (DRAWALL))
                                   "Redraw the work area")
                                   (CHANGE-ERROR (CHANGEERROR)
                                   "Change the allowed error")
                                   (DISPLAY-TEMPLATE (DISPLAYFRAME)"
"Show the variables in a template"
(DISPLAY-CONSTANT (DISPLAYCONSTANTS))

"Show description and value of constant"
(DEFINE-CONSTANT (DEFINECONSTANT))

"Add a new constant to the current state"
(LOAD-FILE (LOADSTATEFILE))
"Load a file containing retained states"
(LOAD-STATE (LOADSTATE))
"Load a previously retained state"
(MAKE-NEW-INSTANCE (MAKEXINSTANCE))

"Make a new instance in the work area"
(ADD-NEW-TEMPLATE (DEFINEFRAME))
"Define a new template"
(DELETE-EQUATION (DELETEEQ))
"Delete an existing equation"
(DEFINE-CONVERSION (DEFINECONVERSION))

"Define a conversion between two units"
(MAKE-ALL-FILES (MAKEALLFILES))
"Make all unread files"
(RETAIN-STATE (SAVESTATE))
"Retain the current state"
(MAKE-NEW-FRAME (MAKEFRAME))
"Make a new frame in the work area"
(ADD-TO-TEMPLATE (ADDTOFRAME))
"Add a variable to a template"
(ADD-NEW-EQUATION (ADDEQUATION))

"Add a new equation to the current state"
(DEFINE-UNIT (ADDUNIT))
"Add a new unit to the current state"

(CENTERFLS + 1)
(MENUCOLUMNS + 1)
(MENUBORDERSIZE + 1)
(ITEMWIDTH + 160))

W.W.))

(CHANGEERROR
(GLAMBDA NIL

(" edited: "4-Jun-8722:52"
(" Changed the value of the allowed error in
converging values of instances

(PRINTNP "Changing value of allowed error ...")
(PRINTNP (CONCAT "Old value of allowed error is " ALLOWEDERROR))
(ALLOWEDERROR = (PROMPTREAD "Enter new value for allowed error"))
(PRINTNP (CONCAT "Allowed error is now " ALLOWEDERROR))
(PRINTNP " changed" T)))

(CHOSENINSTANCE
(GLAMBDA (FORCOMPUTATION OFVARIABLE FROMINSTANCES)

(" edited: "28-Jun-8701:32"
(" Chooses an instance of OFVARIABLE that is chosen by the user for FORCOMPUTATION. Returns a list with the name of
the instance and the conversion factor for its use. FROMINSTANCES is a list of instances from which automatic
selection should be tried first

(PROG (TESTINST:INSTANCE OFVAR:VARIABLE (CONVFACOR 1))

(" TESTINST is the instance being tested for automatic selection)(OFVAR is the actual OFVARIABLE)
(CONVFACOR is the factor by which the chosen instance must be multiplied to be compatible)

(SELECTEDINSTANCE = NIL)
(OFVAR = (EVAL OFVARIABLE))
(FOR INSTNAME IN FROMINSTANCES
TESTINST = (EVAL INSTNAME))
(# (TESTINST:VARIABLETYPE = OFVARIABLE)
(SELECTEDINSTANCE = TESTINST)
(PRINTNP (CONCAT " selected from same frame"))

(IF (NOT SELECTEDINSTANCE)

(TEMP)"
(PRINTNP (CONCAT "Select instance compatible with " OFVAR))

)
(TERPRI)
    (WHILE (NOT SELECTEDINSTANCE)
        (PRINT (* "Select an instance compatible with " OFVARIABLE "..."))
        (TERPRI)
    )
    (WAITFORENTER 500)
    (IF SELECTEDINSTANCE
        (CONVFACTOR (COMPATIBLE SELECTEDINSTANCE STANDARDUNIT OFVAR STANDARDUNIT))
        (UNIFACTORLIST the list of all unifname-conversion-factor pairs for STANUNIT1)
        (CONDFACTOR the conversion factor)
    )
    (RETURN (LIST SELECTEDINSTANCE NAME CONVFACTOR))

(COMPUTABLE)
    (LAMBDA (STANUNIT1 STANUNIT2)
        (* edited: "11-Feb-87 14:02")
        (* Checks if STANUNIT1 and STANUNIT2 are compatible units. Returns NIL if they are not. If they are returns the factor by which STANUNIT2 must be multiplied to be compatible to STANUNIT1)
    )
    (FOR UNIPAIR IN UNIFACTORLIST IF ((CAR UNIPAIR) = STANUNIT2)
        (CONDFACTOR (CADR UNIPAIR))
    )
    (RETURN CONVFACTOR)

(COMPUTE-ANTilog)
    (LAMBDA (VAR FOR RHS ARG)
        (COMPUTE-FORMULA VAR FOR ARG (QUOTE LOG) RHS))
    (* edited: "4-Feb-87 14:37")

(COMPUTE-ACOS)
    (LAMBDA (VAR FOR RHS ARG)
        (COMPUTE-FORMULA VAR FOR ARG (QUOTE COS) RHS))
    (* edited: "29-Jan-87 11:32")

(COMPUTE-ASIN)
    (LAMBDA (VAR FOR RHS ARG)
        (COMPUTE-FORMULA VAR FOR ARG (QUOTE SIN) RHS))
    (* edited: "27-Jan-87 10:48")

(COMPUTE-ATAN)
    (LAMBDA (VAR FOR RHS ARG)
        (COMPUTE-FORMULA VAR FOR ARG (QUOTE TAN) RHS))
    (* edited: "27-Jan-87 10:49")

(COMPUTE-COS)
    (LAMBDA (VAR FOR RHS ARG1)
        (COMPUTE-FORMULA VAR FOR ARG1 (QUOTE COS) RHS))
    (* edited: "28-Dec-86 17:52")

(COMPUTE-DIFFERENCE)
    (LAMBDA (VAR FOR RHS ARG1 ARG2)
        (IF (CONTAINSP ARG1 VAR FOR)
            (COMPUTE-FORMULA VAR FOR ARG1 (QUOTE PLUS) RHS ARG2))
            ELSE (COMPUTE-FORMULA VAR FOR ARG2 (QUOTE DIFFERENCE) ARG1 RHS))
    (* edited: "12-Apr-86 10:03")

(COMPUTE-ERROR)
    (LAMBDA (VAR)
        (* edited: "8-Jul-87 15:08")
        (* Called when a formula cannot be derived for the variable VAR. An error message is printed)
    )

(PROG NIL)
    (PRINT (* "--- unable to derive formula for " VAR "---"))
(RETURN (QUOTE COMPUTE-ERROR))

(COMPUTE-EXPT
  (LAMBDA (VAR-FOR RHS ARG-1 ARG-2)
    (IF (CONTAINSP ARG-1 VAR-FOR)
      (LIST (QUOTE EXPT)
        (COMPUTE-FORMULA VAR-FOR ARG-1 (LIST (QUOTE EXPT) RHS)
        (LIST (QUOTE QUOTIENT)
          I ARG-1))
      ELSE (COMPUTE-FORMULA VAR-FOR ARG-2 (LIST (QUOTE QUOTIENT)
        (LIST (QUOTE LOG)
          RHS)
        (LIST (QUOTE LOG) ARG-1)))
    )
  )

(COMPUTE-FORMULA
  (LAMBDA (VAR LHS RHS)
    (* edited: "27-Jun-8721:23")
    (* Returns an expression for VAR given that LHS = RHS
     (* LHS and RHS are legal Lisp expressions)
     (* RHS should not contain VAR)
    (PROG (OPERATOR ARG1 ARG2)
      (IF (NULL LHS)
        (RETURN (COMPUTE-ERROR VAR))
      ELSEIF (ATOM LHS)
        (IF (EQ LHS VAR)
          (RETURN RHS)
      ELSEIF (CONTAINSP RHS VAR)
          (RETURN (COMPUTE-ERROR VAR))
      ELSEIF (CONTAINSP RHS VAR)
          (RETURN (COMPUTE-ERROR VAR))
      ELSE (SETQ OPERATOR (CAR LHS))
          (SETQ ARG1 (CADR LHS))
          (SETQ ARG2 (CAAR LHS))
          (RETURN (APPLY* (GETPROP OPERATOR (QUOTE COMPUTEFN))
            VAR RHS ARG1 ARG2))
    )
  )

(COMPUTE-INIT
  (LAMBDA NIL
    (* edited: "4-Feb-8714:46")
    (* Initializes all the COMPUTEFN properties
     (* Each operator in an equation has an algebraic
      manipulation function whose name is on its COMPUTEFN
      property)
    (DEFLIST (QUOTE ((DIFFERENCE COMPUTE-DIFFERENCE)
      (PLUS COMPUTE-PLUS)
      (TIMES COMPUTE-TIMES)
      (FQUOTIENT COMPUTE-QUOTIENT)
      (SORT COMPUTE-SORT)
      (MINUS COMPUTE-MINUS)
      (EXPT COMPUTE-EXPT)
      (SIN COMPUTE-SIN)
      (COS COMPUTE-COS)
      (TAN COMPUTE-TAN)
      (ARCSIN COMPUTE-ARCSIN)
      (ARCCOS COMPUTE-ARCCOS)
      (ARCTAN COMPUTE-ARCTAN)
      (LOG COMPUTE-LOG)
      (ANTILOG COMPUTE-ANTILOG)))
    (QUOTE COMPUTEFN))))

(COMPUTE-LOG
  (LAMBDA (VAR-FOR RHS ARG)
    (COMPUTE-FORMULA VAR-FOR ARG (LIST (QUOTE ANTILOG)
      RHS)))

(COMPUTE-MINUS
  (LAMBDA (VAR-FOR RHS ARG1)
    (COMPUTE-FORMULA VAR-FOR ARG1 (LIST (QUOTE MINUS)
      RHS)))

(COMPUTE-PLUS
  (LAMBDA (VAR-FOR RHS ARG1 ARG2)
    (IF (CONTAINSP ARG1 VAR-FOR)
      (COMPUTE-FORMULA VAR-FOR ARG1 (LIST (QUOTE DIFFERENCE)
        )
      ELSEIF (CONTAINSP ARG2 VAR-FOR)
      ELSEIF (CONTAINSP RHS VAR)
      ELSEIF (CONTAINSP RHS VAR)
      ELSEIF (CONTAINSP RHS VAR)
      ELSEIF (CONTAINSP RHS VAR)
      ELSEIF (CONTAINSP RHS VAR)
      ELSEIF (CONTAINSP RHS VAR)
(PSEUDOS-ASKCALC1AB.LSP:1 8-Jul-87 15:38:11)

ELSE (COMPUTE-FORMULA (VAR-FOR ARG2 (LIST (QUOTE DIFFERENCE)
    RHS ARG2))

(COMPUTE-QUOTIENT
[GLAMBDA (VAR-FOR RHS ARG1 ARG2)
  (IF (CONTAINS ARG1 VAR-FOR)
    (COMPUTE-FORMULA (VAR-FOR ARG1 (LIST (QUOTE TIMES)
      RHS ARG2))
    ELSE (COMPUTE-FORMULA (VAR-FOR ARG2 (LIST (QUOTE QUOTIENT)
      ARG1 RHS))))

(COMPUTE-SQRT
[GLAMBDA (VAR-FOR RHS ARG1)
  (COMPUTE-FORMULA (VAR-FOR ARG1 (LIST (QUOTE TIMES)
    RHS RHS)))]

(COMPUTE-SIN
[GLAMBDA (VAR-FOR RHS ARG1)
  (COMPUTE-FORMULA (VAR-FOR ARG1 (LIST (QUOTE ARCSIN)
    RHS)))]

(COMPUTE-TAN
[GLAMBDA (VAR-FOR RHS ARG1)
  (COMPUTE-FORMULA (VAR-FOR ARG1 (LIST (QUOTE ARCTAN)
    RHS)))]

(COMPUTE-TIMES
[GLAMBDA (VAR-FOR RHS ARG1 ARG2)
  (IF (EQUAL ARG1 ARG2)
    (COMPUTE-FORMULA (VAR-FOR ARG1 (LIST (QUOTE SQRT)
      RHS))
    ELSE (CONTAINS ARG1 VAR-FOR)
      (COMPUTE-FORMULA (VAR-FOR ARG1 (LIST (QUOTE QUOTIENT)
        RHS ARG2))
      ELSE (COMPUTE-FORMULA (VAR-FOR ARG2 (LIST (QUOTE QUOTIENT)
        RHS ARG1))]

(COMPUTE-COMPUTATION
[GLAMBDA (self:COMPUTATION)
  (*edited: "25-Apr-8733:10")
  ("Computes the instance computed by this computation using the computation")
  (PROG (COMPUTATIONEXPRESSION INSTNEEDED:INSTANCE INSTNCOMPUTED:INSTANCE CONST:CONSTANT VALUE)
    (*COMPUTATIONEXPRESSION is the Lisp expression used to compute the INSTNCOMPUTED.
     INSTNEEDED is the current needed instance being checked.
     INSTNCOMPUTED is the instance to be computed.
     CONST is the current constant being checked.
     VALUE is the value computed.)
    (COMPUTATIONEXPRESSION = EXPRESSION)
    (FOR INSTANCENAME IN INSTANCESNEEDED (INSTNEEDED + (EVAL INSTNNAME))
      (COMPUTATIONEXPRESSION = (SUBST (THE VALUE OF INSTNEEDED)
        INSTNNAME COMPUTATIONEXPRESSION))
    (FOR CONSTNAME IN CONSTANTS (CONST + (EVAL CONSTNAME))
      (COMPUTATIONEXPRESSION = (SUBST (THE VALUE OF CONST)
        CONSTNAME COMPUTATIONEXPRESSION))
    (INSTNCOMPUTED = (EVAL INSTNCOMPUTED))
    (SEND self DRAW)
    (VALUE = (TESTSTATE COMPUTATIONEXPRESSION))
    (IF VALUE (SEND INSTNCOMPUTED SETVALUE VALUE)
      ELSE (PRINTN "--- no value computed ---")]

(COMPUTE-INSTANCE
[GLAMBDA (self:INSTANCE)
  (*edited: "10-Jun-8716:32")
  ("Computes the value of the instance using its computation, which is created if it is not already created.
  (IF COMPUTATIONS PRINTN "--- Delete existing computation first ---")
  ELSE (IF YESOMENU "Make computation using known equation?" (CONS (self:X +
    INSTANCEDWIDTH)
      (self:Y + 120))
    ELSE (SEND self CREATECOMPUTATION)
      ELSE (SEND self CONSTRUCTCOMPUTATION))]
(CONSTRUCTCOMPUTATION

(GLAMBA (self:INSTANCE)

("* edited: "28-Jun-87 01:30"
(* Constructs a computation for the instance based on
an expression for the instance entered by the user)

(PROG (EXPRCONSTS EXPRINSTNS EXPRSTRING XPOS YPOS ERRORFLAG INSTEPR EXPRITEMS)

("* EXPRCONSTS is a list of constants in the expression
EXPRINSTNS is a list of instances in the expression
EXPRSTRING is the string for the expression
XPOS, YPOS are the display coordinates for the computation
ERRORFLAG is set if an error occurs
INSTEPR is the expression for computing the instance in terms of other
instances
EXPRITEMS is a list of instances and variables in the original expression)

(PRINTNP (CONCAT "Constructing user specified computation for: " self:NAME "."
(PRINTNP (CONCAT "Enter expression that evaluates to: " self:NAME))
(EXPRSTRING = (CONCAT self:NAME " * " INSTEPR))
(INSTEPR = (SIMPLIFY (TRANSLATE INSTEPR)))
(EXPRINSTNS = (EXTRACTCONSTS INSTEPR))
(EXPRITEMS = (EXTRACTVARS INSTEPR T))
(FOR CONST IN EXPRCONSTS (EXPRITEMS ++ CONST))
(FOR LOOPITEM IN EXPRITEMS (IF (MEMBER LOOPITEM ALLINSTANCES)
EXPRINSTNS ++ LOOPITEM)
ELSE (MEMBER LOOPITEM ALLVARS)
(PRINTNP (CONCAT "Need an instance of: 
LOOPITEM))
(SELECTEDINSTANCE = NIL)
(WHILE (NOT SELECTEDINSTANCE)
(TERPRI)
(PRINTNP (CONCAT "Select instance of: 
LOOPITEM "."))
(WAITFORINPUT 500)
(IF SELECTEDINSTANCE
(SELECTEDINSTANCE = LOOPITEM)
(PRINTNP (CONCAT "--- Incompatible instance selected --- Select instance compatible with
LOOPITEM
EXPRINSTNS ++ SELECTEDINSTANCE=NAME))
(INSTEPR = (SUBSTITUTE SELECTEDINSTANCE=NAME
LOOPITEM INSTEPR))
ELSE (PRINTNP (CONCAT "--- LOOPITEM
is unknown --- computation construction aborted ---")
(ERRORFLAG = T))
(IF (NOT ERRORFLAG)
(XPOS = (self:X + self:DX + 6)
(YPOS = (self:Y + 8))
(self:COMPUTATIONS ++
(BUILDCOMPUTATION self:NAME INSTEPR EXPRINSTNS EXPRCONSTS
EXPRSTRING XPOS YPOS)))
(PRINTNP (CONCAT EXPRSTRING " added")

(CONTAINS

(GLAMBA (EXPRESSION VAR)

("* edited: "8-Nov-86 22:45")
("* Returns NIL if the EXPRESSION does not contain
any occurrence of the variable VAR)

(IF (NULL EXPRESSION)
NIL
ELSE (ATOM EXPRESSION)
(EXPR STRING VAR)
ELSE (OR (CONTAINS (CADR EXPRESSION)
VAR)
(CONTAINS (CADR EXPRESSION)
VAR))

(CREATECOMPUTATION

(GLAMBA (self:INSTANCE)

("* edited: "6-Apr-87 01:33")
("* Creates a computation for the instance)
(PROG (INSTANCEVARIABLE VARIABLE FORMULA ADOPTITION FORMULACHOSSEN FORMULAIMPLIMENFORMULAIMPLIMEN
 INFRAME FRAME FROMINSTANCES XPOS YPOS DEFAULTFORMULA)

(* INSTANCEVARIABLE is variable of which the instance is an instance (FORMULA ADOPTITION is a formula for
 INSTANCEVARIABLE:VARIABLE) FORMULACHOSSEN is the formula chosen by the user for the computation)
(INFRAME is the frame for the instance if one exists) FROMINSTANCES is the list of instances in INFRAME other
than self风景XPOS, YPOS are the coordinates where the computation should be created (DEFAULTFORMULA is the name
of the default formula)

(INSTANCEVARIABLE = (EVAL VARIABLETYPE))
(IF FRAME (INFRAME = (EVAL FRAME))
(FROMINSTANCES = (REMOVE self:NAME INFRAME:INSTANCES))

FOR FORMULANAME
IN THE FORMULAS OF INSTANCEVARIABLE
(FORMULA ADOPTITION = (EVAL FORMULANAME))
(IF FORMULA ADOPTITION FRAMEDEF (IF (AND FRAME (FORMULA ADOPTITION FRAMEDEF =
\n INFRAME:FRAMEDEF))
(MENULIST =
\n (LIST (THE EQUATIONSTRING OF FORMULA ADOPTITION)
\n (LIST (QUOTE QUOTE)
\n FORMULANAME))
\n (DEFAULTFORMULA + FORMULANAME))
ELSE (MENULIST =
\n (LIST (THE EQUATIONSTRING OF FORMULA ADOPTITION)
\n (LIST (QUOTE QUOTE)
\n FORMULANAME)))
\n (DEFAULTFORMULA + FORMULANAME)))
(IF MENULIST IF (LENGTH MENULIST) = 1)
\n (FORMULACHOSSEN = (EVAL DEFAULTFORMULA))
\n (PRINTNP (CONCAT "Automatically using ">
\n FORMULACHOSSEN: EQUATIONSTRING))
\n ELSE (FORMULACHOSSEN = (EVAL MENU (CREATE MENU (TITLE =
\n "Using which equation?"
\n [ITEMS = MENULIST]
\n (XPOS = ((self:X = self:DX)+ 5))
\n (YPOS = (self:Y + 8))
\n (self:COMPUTATIONS =
\n (SEND FORMULACHOSSEN MAKECOMPUTATION self:NAME FROMINSTANCES
\n XPOS YPOS)))
\n ELSE (PRINTNP "--- no formula exists ---")

(CREATEINSTANCE
\n (GLAMBDA \n (VARIABLENAME FRAMENAME XPOS YPOS) (* edited: "7-Jun-87 14:22")
\n (* creates an instance of the variable whose name is VARIABLENAME, FRAMENAME is the name of the frame that
\n this instance belongs to, XPOS, YPOS are the coordinates for the instance, returns the name of the instance created)

(PROG (INSTANCENAME INSTANCECREATED INSTANCE ACTUALVARIABLE:VARIABLE INST-X INST-Y LOCATION)
\n (* (INSTANCENAME is the name of the created instance) INSTANCECREATED is the actual instance created)
\n (ACTUALVARIABLE is the actual variable whose name is VARIABLENAME) (INST-X, INST-Y are the coordinates for the instance
\n box) (LOCATION is the location of the box))
\n (INSTANCENAME = (GENSYM VARIABLENAME))
\n (# XPOS (INST-X = XPOS)
\n (INST-Y = YPOS)
\n ELSE (PRINTNP "Specify location for the instance")
\n (LOCATION = (GETEXPOSITION INSTANCEWIDTH INSTANCEHEIGHT NIL NIL W.W))
\n (INST-X = (CAR LOCATION))
\n (INST-Y = (CDR LOCATION)))
\n (ACTUALVARIABLE = (EVAL VARIABLENAME))
\n (INSTANCECREATED =
\n (SET INSTANCENAME
\n (AN INSTANCE WITH X = INST-X Y = INST-Y DX = INSTANCEWIDTH DY =
\n INSTANCEHEIGHT VARIABLETYPE = VARIABLENAME FRAME =
\n FRAMENAME STANDARDUNIT = (THE STANDARDUNIT OF
\n ACTUALVARIABLE)
\n NAME = INSTANCENAME DISPLAYUNIT = (THE STANDARDUNIT OF
\n ACTUALVARIABLE)
\n DISPLAYFACTOR = 1.0 EXPANDED = T)))
\n )
(PROG (CONSTNAME CONST:CONSTANT)
   (CONSTNAME = (PROMPTREAD "What is the name of the constant?"))
   (IF (MEMBER CONSTNAME ALLCONSTANTS)
      (PRINT "* Overwriting previous definition")
      ELSE (SETF CONSTNAME (A CONSTANT))
   (ALLCONSTANTS =
      (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CONS (CON...
(DEFINEFRAME
 (GLAMBA NIL
 (PROG (FRODEFFAME)
   (PRINTNP "Adding template ...")
   (FRODEFFAME = (PROMPTREAD "What is the name of the template to be added? ")
   (SET FRODEFFAME (A FRAMEDEF))
   (ALLFREDFS ++
   FRODEFFAME)
   (ALLFREDFS = (SORT ALLFREDFS))
   (PRINTNP (CONCAT FRODEFFAME " template added")
   T)))

(DELETECOMPUTATION
 [GLAMBA (self:COMPUTATION)
 (PROG (DLINK:LINK DINST:INSTANCE)
   (DINST = (EVAL INSTANCECOMPUTED))
   (DINST:COMPUTATIONS =
   self:NAME)
   (FOR INSTNAME IN INSTANCESNEEDED (DINST = (EVAL INSTNAME))
   (DINST:USEDBY ++
   self:NAME))
   (FOR LINKNAME IN self:LINKSLIST (DLINK = (EVAL LINKNAME))
   (SEND DLINK DELETE LINKNAME))
   (ALLCOMPUTATIONS =
   self:NAME)
   (OSFILL (CREATEREGION self:X self:Y (self:DX + 1)
   (self:DY + 1))
   0
   (QUOTE REPLACE)
   W:W)
   (PRINTNP (CONCAT self:NAME " deleted")))

(DELETEEQN
 [GLAMBA NIL
 (PROG (EQN:EQUATION EQNNAME FORMULA VAR:VARIABLE MENULIST)
   (EQN is an equation whose name is EQNNAME) (FORM is a formula that needs to be deleted)
   (VAR is a variable which can no longer use FORM) (MENULIST is a list of menu items))
   (FOR EQUATIONNAME IN ALLEQUATIONS (EQN = (EVAL EQUATIONNAME))
   (MENULIST =
   (LIST EQN:EQUATIONNAME (LIST (QUOTE EQNNAME) (QUOTE EQN:MENU)
   EQNNAME = (MENU (CREATE MENU (TITLE = "Delete which equation?")
   (ITEMS = MENULIST)
   (MENUPosition = (QUOTE (700 . 600)
   (EQN = (EVAL EQNNAME))
   (FOR FORMULANAME IN EQN:FORMULAS (FORM = (EVAL FORMULANAME))
   (VAR = (EVAL FORM:FORVARIABLE))
   (VAR:FORMULAS =
   FORMULANAME)
   (ALLFORMULAS =
   FORMULANAME))
   (ALLEQUATIONS =
   EQNNAME)
   (PRINTNP (CONCAT EQN:EQUATIONNAME " deleted")))

(DELETEFRAME
 [GLAMBA (self:FRAME)
 (PROG (INST:INSTANCE COMP:COMPUTATION FRO:FRAMEDEF ERRORFLAG)
   (* INST is an instance in the frame) (COMP is a computation connected to the frame) (FRODEF is the frame definition
   for the frame deleted) (ERRORFLAG is set when an instance that cannot be deleted is found))
   (FOR INSTNAME IN INSTANCES (INST = (EVAL INSTNAME))
   (# USEDBY ERRORFLAG = T)))
(IF ERRORFLAG (PRINTNP "--- frame contains instances that are used in computations --- cannot delete ---")
    (FOR INSTNAME IN INSTANCES (INST = (EVAL INSTNAME))
        (FOR COMPNAM IN COMPUTATIONS (COMP = (EVAL COMPNAM))
            (SEND COMP DELETE))
        (DSPFILL (CREATEREGION self:X self:Y (self:DX + 2) (self:DY + 2))
            (QUOTE REPLACE)
            W.W)
        (FRDEF = (EVAL self:FRAMEDEF))
        (FRDEF:FRAMES -- self:NAME)
        (ALLFRAMES -- self:NAME)
        (PRINTNP (CONCAT self:NAME " deleted")))

(DELETEINSTANCE)
(GLAMBDA (self:INSTANCE)
    (* edited: "27-Jun-87 12:00")
    (* Deletes the instance)
    (PROG (COMP:COMPUTATION INSTFRAME:FRAME)
        (* (COMP is the computation of the instance if it has one, which is also deleted) (INSTFRAME is the frame of this instance is a part of))
        (IF USEDFR (PRINTNP "--- cannot delete --- used in computation(s) ---")
            (ELSE FOR COMPNAM IN COMPUTATIONS (COMP = (EVAL COMPNAM))
                (SEND COMP DELETE))
            (ALLINSTANCES -- self:NAME)
            (DSPFILL (CREATEREGION self:X self:Y (self:DX + 1) (self:DY + 1))
                (QUOTE REPLACE)
                W.W)
        (IF FRAME (INSTFRAME = (EVAL FRAME))
            (INSTFRAME:INSTANCES -- self:NAME)
            (PRINTNP (CONCAT self:NAME " deleted")))

(DELETELINK)
(GLAMBDA (self:LINK LINKNAME)
    (* edited: "15-Dec-86 15:57")
    (* Deletes the link whose name is LINKNAME)
    (PROG (DISTN:INSTANCE)
        (* DISTN is an instance from which the link must be removed)
        (DISTN = (EVAL FROMINSTANCE))
        (DISTN:LINKSLIST -- LINKNAME)
        (SEND self:ERASE)
        (ALLLINKS -- LINKNAME)
        (SEND DISTN DRAW)))

(DISPLAYCONSTANTS)
(GLAMBDA NIL
    (* edited: "8-Jun-87 10:33")
    (* Displays the description of a constant)
    (PROG (ITEMSLIST CONST:CONSTANT)
        (FOR CONSTATE IN (SORT ALLCONSTANTS)
            (CONST = (EVAL CONSTNAME))
            (ITEMSLIST =
                (LIST CONSTATE (QUOTE PRINTIND)
                    (CONCAT CONSTATE " : " CONST:DESCRIPTION
                        CONST:VALUE)
                (MENU (CREATE MENU (TITLE + "Display which constant")
                    (ITEMS = ITEMSLIST)
                    (MENUPosition = (QUOTE (850 . 800))))

(DISPLAYCOMPUTATION)
(GLAMBDA (self:COMPUTATION)
    (* edited: "20-Apr-87 10:56")
(DISPLAYFRAME
  [GLAMBA NIL
   (PROG (MENUITEM FRDEF:FREDEF)
      [FOR FRDEFNAME IN ALLFRDEFS (FRDEF + (EVAL FRDEFNAME))
       (MENUITEM ++
         (LIST FRDEFNAME (LIST (QUOTE PRINTING)
                              (CONCAT FRDEFNAME
                                   " consists of "
                                   FRDEF:VARIABLES)])
       )]
   )
   (MENU (CREATE MENU (TITLE "Display which frame template")
           ( ITEMS = MENUITEMS)
           (MENUPOSITION = (QUOTE (520 . 600)))
   )
   )

(DISPLAYINSTANCE
  [GLAMBA (self:INSTANCE)
   [PROG (DISPLAYSTRING)
      (IF EXPANDED (DSPFILL (CREATEREGION (X + 2)
                                   (Y + 2)
                                   (DX - 4)
                                   (DY - SHRUNKHEIGHT - 4))
         (QUOTE REPLACE)
         W.W)
      (MOVE TO (X + 4)
               (Y + 4)
               W.W)
      (IF (X + DX) > WINDOWWIDTH)
         (PRINTNP "--- insufficient space for box ---")
      ELSE (DISPLAYSTRING + (CONCAT (TIMES VALUE DISPLAYFACTOR)
                                     (IF STANDARDUNIT (CONCAT " " DISPLAYUNIT)
                                     ELSE ""))
      (IF ((CHARS DISPLAYSTRING) > MAXCHARS)
         (PRINI (SUBSTRING DISPLAYSTRING 1 MAXCHARS)
                W.W)
      ELSE (PRINI DISPLAYSTRING W.W))
   )
   )

(DRAWALL
  [GLAMBA NIL
   [PROG (DLINK:LINK DFRAME:FRAME DINST:INSTANCE)
      (DRENDER W.W)
      (CLEAR W.W)
      (FOR LKNNAME IN ALLLINKS (DLINK + (EVAL LKNNAME))
       (SEND DLINK DRAW))
      (FOR FRDNAME IN ALLFRAMES (DFRAME + (EVAL FRDNAME))
       (SEND DFRAME DRAW))
      (FOR INSTNAME IN ALLINSTANCES (DINST + (EVAL INSTNAME))
       (SEND DINST DRAW)
       (SEND DINST DISPLAY))
   )
   )

(DRAWBOX
  [GLAMBA (X Y DX DY LINESIZE LABEL)
   [MOVE TO (X Y W.M)
   (DRAWTO X (Y + DY)
            LINESIZE (QUOTE REPLACE)
            W.M)
   (DRAWTO (X + DX)
   )
   )
(DRAWTO X + DX)
  Y LINESIZE (QUOTE REPLACE)
  W.W)
(DRAWTO X Y LINESIZE (QUOTE REPLACE)
  W.W)
(MOVETO X (Y + DY) SHRUNKHEIGHT)
  W.W)
(DRAWTO X + DX)
  (Y + DY - SHRUNKHEIGHT)
  LINESIZE
  (QUOTE REPLACE)
  W.W)
(IF ((X + DX)
  > WINDOWWIDTH)
  PRINTLN "----- insufficient space ----")
ELSE (DSPFILL (CREATEREGION (X + 2)
  (Y + DY - SHRUNKHEIGHT + 2)
  (DX - 4)
  (SHRUNKHEIGHT - 2))
  0
  (QUOTE REPLACE)
  W.W)
(MOVETO (X + 3)
  (Y + DY - SHRUNKHEIGHT + 5)
  W.W)
(IF ((NCHARS LABEL)
  > MAXCHARS)
  (PRIN1 (SUBSTRING LABEL 1 MAXCHARS)
  W.W)
ELSE (PRIN1 LABEL W.W))

(DRAWCOMPUTATION
 (SLAMBDA (self:COMPUTATION)
  (* edited: "7-Jun-8713:42")
  (* Draws the box for the computation)
  (DSPFILL (CREATEREGION X Y (DX + 1)
  (DY + 1))
  0
  (QUOTE REPLACE)
  W.W)
  (DRAWBOX X Y DX DY 1 NAME)))

(DRAWFRAME
 (SLAMBDA (self:FRAME)
  (* edited: "9-Jan-8715:41")
  (* Draws the frame)
  (DRAWBOX X Y DX DY 2 NAME)))

(DRAWINSTANCE
 (SLAMBDA (self:INSTANCE)
  (* edited: "9-Jan-8716:52")
  (* Draws the box for the instance)
  (IF (OR EXPANDED (NOT FRAME))
   (DRAWBOX X Y DX DY 1 NAME))))

(DRAWLINK
 (SLAMBDA (self:LINK)
  (* edited: "8-Jul-8715:20")
  (* Draws the link after computing it)
  (* FROMINST is the instance at which to start)
  (* TOCOMP is the computation at which to end)
  (PROG (FROMINST:INSTANCE TOCOMP:COMPUTATION)
   (* FROMINST + (EVAL FROMINST:INSTANCE))
   (* TOCOMP + (EVAL TOCOMP:COMPUTATION))
   (FROMX + (FROMINST:X + (FROMINST:DX / 2))
   (TODY + (TOCOMP:Y + (TOCOMP:DY / 2)))
   (MOVE TOFX FROMX FROMY W.W)
   (DRAW TOX TOY 1 (QUOTE REPLACE)
   W.W)
   (SEND FROMINST DRAW)
   (SEND FROMINST DISPLAY)
   (SEND TOCOMP DRAW))))
{PSEUDO-DSK} \text{callab.lsp:1} 8-Jul-87 15:38:19

(EQUATEINSTANCE
   \text{GLAMBA (self:INSTANCE)}
   \text{(*) edited: "7-Jun-87 14:20"}
   \text{(*) Sets up a computation so that the instance always}
   \text{has the value of some other compatible instance)}
   \text{COMPNAME}
   \text{(*) COMP is the computation created) (INSTLINK is a link to the computation) (OTHERINST is the other instance that}
   \text{this instance is equated to) (INSTCHOSEN is the name of OTHERINST) (LINKNAME is the name of a link)}
   \text{INSTUNITPAIR is a list of the name of the instance chosen and the conversion factor for its use)}
   \text{COMPNAME is the name of the pseudo-computation created)
   \text{(PRINT "Setting up USE")}
   \text{(COMPNAME = (gensym "USE"))}
   \text{(INSTUNITPAIR = (CHOSENINSTANCE COMPNAME self:VARIABLETYPE))}
   \text{(INSTCHOSEN = (CAR INSTUNITPAIR))}
   \text{(SET COMPNAME (A COMPUTATION WITH INSTANCECOMPONED self:NAME NAME = COMPNAME I =}
   \text{(self:X = self:DX + 5))}
   Y = (self:Y + 5)
   Dx = COMPWIDTH Dy = COMPHEIGHT EXPRESSION =
   \text{(IF ((CADR INSTUNITPAIR) = 1))}
   \text{INSTCHOSEN}
   \text{ELSE (QUOTE TIMES)}
   \text{INSTCHOSEN}
   \text{INSTUNITPAIR (CADR INSTUNITPAIR))}
   \text{EQUATIONSTRING = INSTCHOSEN INSTANCESNEEDED = (LIST INSTCHOSEN)))}
   \text{(self:COMPUTATIONS ++ COMNAME)}
   \text{(ALLCOMPUTATIONS ++ COMNAME)}
   \text{(COMP = (EVAL COMNAME))}
   \text{(SEND COMP DRAW)}
   \text{(LINKNAME = (GENSYM (QUOTE LINK)))}
   \text{(SET LINKNAME (A LINK WITH FROMINSTANCE self:NAME TOCOMPUTATION = COMNAME))}
   \text{(ALLLINKS ++ LINKNAME)}
   \text{(INSTLINK = (EVAL LINKNAME))}
   \text{(COMP:LINKLIST ++ LINKNAME)}
   \text{(self:LINKLIST ++ LINKNAME)}
   \text{(SEND INSTLINK DRAW)}
   \text{(OTHERINST = (EVAL INSTCHOSEN))}
   \text{(LINKNAME = (GENSYM (QUOTE LINK)))}
   \text{(SET LINKNAME (A LINK WITH FROMINSTANCE INSTCHOSEN TOCOMPUTATION = COMNAME))}
   \text{(ALLLINKS ++ LINKNAME)}
   \text{(INSTLINK = (EVAL LINKNAME))}
   \text{(COMP:LINKLIST ++ LINKNAME)}
   \text{(OTHERINST:LINKLIST ++ LINKNAME)}
   \text{(SEND INSTLINK DRAW)}
   \text{(SEND COMP COMPUTE))}

\text{(ERASELINK}
   \text{GLAMBA (self:LINK)}
   \text{(*) edited: "7-Jun-87 13:44"}
   \text{(*) Erases the link)
   \text{(MOVE TO FROM X W.W)}
   \text{(DRAW TO TOY 3 QUOTE ERASE)}
   \text{(W.W))}

\text{(EXPANDFRAME}
   \text{GLAMBA (self:FRAME)}
   \text{(*) edited: "7-Jun-87 13:45"}
   \text{(*) Expands the box for the frame)
   \text{(*) INST is an instance in the frame)
   \text{COUNTER is a positioning counter for the instances)
   \text{(PROG (INST INSTANCE COUNTER:INTEGER)
               (DSPFILL (CREATEREGION self:X self:Y (self:DX + 2)) (self:DY + 2))
               (QUOTE REPLACE)
               W.W)
               (self:DX = (\text{LENGTH self:INSTANCES})}
(PSEUDO-DISK)CALCLAB.LSP:1 8-Jul-87 15:36:19

* 
(INSTANCEHEIGHT + 2))
(GLFW (self:Y + self:DY)
> WINDOWHEIGHT
(ENDIF)
(self:Y = (self:Y - self:DY))
(GLFW EXPANDED = T)
(COUNTER + 3)
(PFOR INSTNAME IN INSTANCES (INST = (EVAL INSTNAME))
(INST:X = (self:X + 3))
(INST:Y = (self:Y + COUNTER))
(SEND INST EXPAND)
(COUNTER ++
(INSTANCEHEIGHT + 2)))
(SEND self DRAW)))

(EXPANDINSTANCE
(GLAMBA (self:INSTANCE))

(DSPFILL (CREATE_REGION X Y (DX + 1)
(DY + 1))

0 (QUOTE REPLACE
W, W)

(DY = INSTANCEHEIGHT)
(EXPANDED = T)
(SEND self MOVE X Y)))

(EXTRACTCONSTANTS
(GLAMBA (EXPRESSION))

(*edited: "7-Jun-87 13:45")
(* expands the box for the instance)

(*edited: "17-Dec-87 14:36")
(* returns a list of all known constants in
expression)

(*edited: "25-Mar-87 10:53")
(* returns a list of all the variables in expression, assuming that it is a legal lisp expression.
Duplicates are not eliminated if ALLOWDUPS is T)

(FINDUNITS
(GLAMBA (ATUNIT))

(*edited: "3-Jun-87 21:14")
(* returns a list of unitname-conversionfactor pairs
for all units that ATUNIT can convert to.)

(*UFLIST is a list of unitname-conversionfactor pair lists)(CHECKEDNAMES is a list of the names of all units
that have been checked)(PREUFLIST is a list of unitname-conversionfactor pairs that have not yet been checked
for inclusion in UFLIST)(CURRNAME is the name of the unit currently being checked)(CURRFAC is the conversion
factor for the unit currently being checked)(CURROBJ is the unit named CURRNAME)

(PREUFLIST ++
(LIST ATUNIT 1))
(PSEUDO-DSK)CALCLAB.LSP:1  8-Jul-87 15:36:19

(P) WHILE PRAUFLIST (CURRNAME = (CAAR PRAUFLIST))
     (CURRFACOR = (CADR (CAR PRAUFLIST)))
     (PRAUFLIST = (CDR (CAR PRAUFLIST)))
     IF (NOT (MEMBER CURRNAME CHECKEDNAMES))
         (CHECKEDNAMES = CURRNAME)
         (UFLIST = (LIST CURRNAME CURRFACOR))
         (CURROBJ = (EVAL CURRNAME))
         FOR UFPAIR IN CURROBJ:CONVERSIONS (PRAUFLIST)
             (LIST (CAR UFPAIR)
                 (TIMES CURRFACOR (CADR UFPAIR))
         (RETURN UFLIST))))

(INPUTVALUE
     (GLambda (self:INSTANCE) ("* edited: "6-Jun-8713:24")
         (* Inputs a value for this instance)
     (PROG (NUMBERININPUT MENULIST UNITFACTORLIST USELECTED)
         ("* NUMBERININPUT is the value input by the user")
         (MENULIST is a list of menu items)
         (UNITFACTORLIST is a list of
         unit-name-conversion-factor pair[s])
         (USELECTED is the unit-name-conversion-factor pair selected)
         (WHILE (NOT (NUMBERP NUMBERININPUT))
             NUMBERININPUT = (PROMPTREAD (CONCAT "Enter the new value for "
                 "NAME")
             IF STANDARDUNIT (UNITFACTORLIST = (FINDUNITS STANDARDUNIT))
                 FOR UFPAIR IN UNITFACTORLIST (MENULIST =
                     (LIST (CAR UFPAIR)
                         (QUOTE QUOTE)
                     UFPAIR)
                 (IF (LENGTH MENULIST)
                     > 1)
                     USELECTED = (MENU (CREATE MENU (TITLE = "Specify units")
                         (ITEMS = MENULIST)
                         (MENUPOSITION = (CONS (self:x +
                             INSTANCEWIDTH)
                         (self:y + 10))
                         (DISPLAYUNIT = (CAR USELECTED))
                         (DISPLAYFACTOR = (CADR USELECTED))
                         (NUMBERININPUT = (QUOTIENT NUMBERININPUT (CADR USELECTED)]
                     (SEND self setVALUE NUMBERININPUT))))

(INSIDEBOX
     (GLambda (x y dx dy xcoord ycoord)
         ("* edited: "5-Dec-8511:06")
         (* Returns the position inside the
         box defined by x, y, dx, dy)
     (AND (XCOORD >= x)
         (XCOORD <= (x + dx))
         (YCOORD >= y)
         (YCOORD <= (y + dy))

(LOADSTATE
     (GLambda NIL
         ("* edited: "27-Jun-87:22:10")
         (* Loads a previously saved state)
     (PROG (LOADLIST MENULIST OBJ1:UNIT OBJ2:UNIT LOADNAME VOBJ:VARIABLE)
         ("* LOADLIST is the list to be loaded")
         (MENULIST is a list of menu items)
         (OBJ1 and OBJ2 are units)
         (LOADNAME is the name of the list to be loaded)
         (VOBJ is a variable being loaded)
         (IF (YESNOMENU "Retain current state before loading new state?" (QUOTE
             (160 . 600)))
             (SAVESTATE))
         (IF (YESNOMENU "Delete current state before loading new state?" (QUOTE
             (160 . 600)))
             (ALLVARS = NIL)
             (ALLFORMULAS = NIL)
             (ALLINSTANCES = NIL)
             (ALCOMPUTATIONS = NIL)
             (ALLINKS = NIL)
             (ALLCONSTANTS = NIL)
             (ALLUNITS = NIL)
             (ALLPRODEFS = NIL)
(DEFINE (LOADNAME NAME))

(EXIT)

(MENULIST ++)

(EXIT)

(EXIT)

(EXIT)

(EXIT)

(EXIT)

(EXIT)

(EXIT)

(EXIT)

(EXIT)
(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)

(LOADSTATEFILE)
(PRINTNP "Making computation ...")
(COMPUTATIONNAME = (GENSYM "=CNP")
(COMPUTATIONEXPRESS = self:EXPRESSION)
(SET COMPUTATIONNAME
   (A COMPUTATION WITH INSTANCECOMPUTED = INSTANCENAME NAME = COMPUTATIONNAME X =
      XPOS Y = YPOS DX = COMPWIDTH DY = COMPHEIGHT CONSTANTS = self:CONSTANTS
      EQUATIONSTRING = self:EQUATIONSTRING))
(ALLCOMPUTATIONS ++
   COMPUTATIONNAME)
(REALCOMPUTATION = (EVAL COMPUTATIONNAME))
(SEND ACTUALCOMPUTATION DRAW)
(LINKNAME = (GENSYN (QUOTE LINK)))
(SET LINAME (A LINK WITH FROMINSTANCE = INSTANCENAME TOCOMPUTATION =
   COMPUTATIONNAME))
(ALLLINKS ++ LINKNAME)
(ACTUALLINK = (EVAL LINKNAME))
(CHOSENINST = (EVAL INSTANCENAME))
(REALCOMPUTATION:LINKSLIST ++
   LINKNAME)
(CHOSENINST:LINKSLIST ++
   LINKNAME)
(SEND ACTUALLINK DRAW)
(FOR VARIABLENAME IN VARIABLES
   (* For each variable used by the formula)
      (INSTINITPAIR = (CHOSENINSTANCE COMPUTATIONNAME VARIABLENAME FROMINSTANCES))
      (INSTANCECHOSEN = (CAR INSTINITPAIR))
      (NEEDEDLIST ++
         INSTANCECHOSEN)
      (IF ((CAAR INSTINITPAIR) = 1) (COMPUTATIONEXPRESS = (SUBST INSTANCECHOSEN
         VARIABLENAME COMPUTATIONEXPRESS))
         ELSE (COMPUTATIONEXPRESS = (SUBST (LIST (QUOTE TIMES)
         INSTANCECHOSEN
         (CAADR
         INSTINITPAIR))
         VARIABLENAME COMPUTATIONEXPRESS)))
(LINKNAME = (GENSYN (QUOTE LINK)))
(SET LINAME (A LINK WITH FROMINSTANCE = INSTANCECHOSEN TOCOMPUTATION =
   COMPTULATIONNAME))
(ALLLINKS ++
   LINKNAME)
(ACTUALLINK = (EVAL LINKNAME))
(CHOSENINST = (EVAL INSTANCECHOSEN))
(REALCOMPUTATION:LINKSLIST ++
   LINKNAME)
(CHOSENINST:LINKSLIST ++
   LINKNAME)
(SEND ACTUALLINK DRAW)
(REALCOMPUTATION:INSTANCESNEEDED = NEEDEDLIST)
(REALCOMPUTATION:EXPRESSION = COMPUTATIONEXPRESS)
(PRINTNP "Computation created")
(SEND ACTUALCOMPUTATION COMPUTE)
(RETURN COMPUTATIONNAME)))

(MAKEFRAME
   (GLANEDR NIL
   (*) edited: 8-Jun-87 10:39")
   (*) Makes a frame using a frame definition)
   (PROG (MENUFILE FRDFCHOSEN INSTANCESUSED FRAMENAME FRAMEHEIGHT LOCATION XPOS YPOS
   COUNTER:INTEGER FRAMENAME:FRAME INST:INSTANCE FRDIF:FRAMEDEF)
   (*) (MENUFILE is a list of all menu items) (FRDFCHOSEN is the name of the frame definition from which the frame is
   made) (INSTANCESUSED is a list of the names of all instances used in the frame) (FRAMENAME is the name of the frame
   created) (FRAMEHEIGHT is the height of the frame) (LOCATION is the location of the box for the frame)
   (XPOS, YPOS are the coordinates of LOCATION) (COUNTER is a counter to determine the locations of instances of the
   frame) (FRAMENAME:FRAME is the frame made) (INST is an instance of the frame) (FRDIF is the frame definition from
   which the frame is made)
(MAKEINSTANCE)
  (GLAMDA NIL)

(MAKESTATEFILE)
  (GLAMDA (STATETOSAVE)
    (PROG (MENULIST FILENAME)
      (* (MENULIST is a list of menu items) (FILENAME is the name of the file in which the state is to be saved))
      (FILENAME - (PROMPTREAD "Save in which file?  (Do not specify extension")*)
      [IF (BOUNDP (PACK* FILENAME (QUOTE CONS)))
        [IF [NOT (INFILECONS? STATETOSAVE (QUOTE VAR)
              (PACK* FILENAME (QUOTE CONS))
            (ADDTOFILE STATETOSAVE (QUOTE VAR)
              (PACK* FILENAME (QUOTE .CL))]
        ELSE (SET (PACK* FILENAME (QUOTE CONS)))
      (STATETOSAVE))

(SETSLOTS FILENAME)
  (GLAMDA (STATETOSAVE)
    (PROG (MENULIST)
      (* (MENULIST is a list of menu items))
      (NAME "Create an instance of which?")
      (ITEMS - MENULIST)
      (MENUPOSITION - (QUOTE (350 . 600))
        (IF FREDCHOSEN (FRAMENAME - (GENSYM FREDCHOSEN))
          (FRED - (EVAL FREDCHOSEN))
          (FRED:FRAMES -- FRAMENAME)
          (FRAMEHEIGHT - (LENGTH FRED:FRAMES)
            (+ INSTANCEHEIGHT + 2) + 10))
          (LOCATION - (GETBBOXPOSITION INSTANCEWIDTH + COMPWIDTH + 5)
            FRAMEHEIGHT NIL NIL W.W))
          (XPOS - (CAR LOCATION))
          (YPOS - (CDR LOCATION))
          (COUNTER + 3)
          (FOR VARNAME
            (REVERSE FRED:FRAMES)
            (INSTANCESUSED - (CREATEINSTANCE VARNAME FRAMENAME XPOS + 3)
              (YPOS + COUNTER)))
          (COUNTER + 1)
          (INSTANCEHEIGHT + 2))
          (FRAMEMADE - (MEMBER FRAMENAME (INSTANCESUSED))
            (NAME "expanded")
            (FRAMEHEIGHT NAME - FRAMENAME EXPANDED = + 1 FRAMEDEF FREDCHOSEN))
          (FRAMEMADE - (EVAL FRAMEMADE))
          (ALLFRAMES -- FRAMENAME)
          (SEND FRAMEMADE DRAM)
          (PRINT (CONCAT FRAMENAME " created"))
        ELSE
          (PRINT "--- aborted ---")
      (PROG (VARIABLELIST MENULIST)
            (VARIABLELIST - (SORT ALLGVAR))
            (FOR LOOPVARIABLE IN VARIABLELIST (MENULIST ++
              (LIST LOOPVARIABLE)
              (LIST (QUOTE CREATEINSTANCE)
                (LIST (QUOTE QUOTE)
                  LOOPVARIABLE)
              (LIST (QUOTE QUOTE)
                loopvariable)
              (LIST (QUOTE QUOTE)
                (MENUPosition = (QUOTE (350 . 600)))
              ))
          )
      (PROG (MENULIST)
        (* (MENULIST is a list of menu items)))
        (NAME "Create an instance of which?")
        (ITEMS - MENULIST)
        (MENUPOSITION - (QUOTE (350 . 600)))
      )
    )
  )

(VALUES)
  (GLAMDA (VAR)
    (VALUES VAR)
    (VALUES (QUOTE CLOSURE))

(VALUES)
  (GLAMDA (VAR)
    (VALUES VAR)
    (VALUES (QUOTE CLOSURE))

(PACKFILE)
  (GLAMDA (NAME EXT)
    (VALUES NAME)
    (VALUES (QUOTE CLOSURE))

(PACKFILE)
  (GLAMDA (NAME EXT)
    (VALUES NAME)
    (VALUES (QUOTE CLOSURE))

(PACKFILE)
  (GLAMDA (NAME EXT)
    (VALUES NAME)
    (VALUES (QUOTE CLOSURE))
{(PSEUDO-DSK)CALCLAB.LSP:1  8-Jun-87 15:36:19

(LIST (LIST (QUOTE VARS)
  STATETOSAVE)
  (IF (NOT (MEMBER FILENAME FILESTORAGE))
    (FILESTORAGE ++
     FILENAME))
  (PRINTNP (CONCAT FILENAME "CONS updated"))

(MOVECOMPUTATION
  [GLAMBDA (self:COMPUTATION XPOS YPOS)
     (* edited: "7-Jun-87 13:46")
     (* Moves the box for the computation to the location
       selected by the user)
     (* XPOS, YPOS are the coordinates that the computation
       is to be moved to)
     (* LINKNAME is the name of ACTUALLINK)
  ]

(MOVENAME
  [GLAMBDA (self:FRAME)
     (* edited: "7-Jun-87 14:24")
     (* Moves the box for the frame)

  ]

(MOVEINSTANCE
  [GLAMBDA (self:INSTANCE XPOS YPOS)
     (* edited: "7-Jun-87 14:24")
     (* Moves the box for the instance to the position selected by the user.
       XPOS, YPOS are the coordinates where the instance is to be moved)

  ]

 Antar

 Antar

 Antar

 Antar

 Antar
(DSPFILL (CREATEREGION self:X self:Y (self:DX + 1) (self:DY + 1))
  0 (QUOTE REPLACE) NIL W.W)
(FOR LINKNAME IN self:LINKSLIST (ACTUALLINK = (EVAL LINKNAME))
  (SEND ACTUALLINK ERASE))
(IF XPOS (self:X - XPOS) (self:T Y + YPOS)
  ELSE (PRINTNP (CONCAT "Specify new location for " self:NAME))
  (NEWLOCATION = (GETBOXPOSITION (self:DX + COMWIDTH + 5) self:DY NIL NIL W.W))
  (self:X = (CAR NEWLOCATION))
  (self:Y = (CDR NEWLOCATION))
  (IF ((self:X + INSTANCEWIDTH + COMWIDTH + 12) > WINDOWWIDTH)
    (self:X = (WINDOWWIDTH - INSTANCEWIDTH - COMWIDTH - 12))
    (SEND self DRAW)
    (SEND self DISPLAY)
    (FOR COMPNAME IN COMPUTATIONS (COMP = (EVAL COMPNAME))
      (IF EXPANDED (SEND COMP MOVE (self:X + self:DX + 5) (self:Y + 8))
        ELSE (SEND COMP MOVE (self:X + self:DX + 3) self:Y)))))
(FOR LINKNAME IN self:LINKSLIST (ACTUALLINK = (EVAL LINKNAME))
  (SEND ACTUALLINK DRAW))

(PRINTIND
  (SLAMBDA (MESSAGE) (* edited: "7-Jul-87 13:42")
    (* Displays message in the D window)
    (TERPRI D.W)
    (PRIN1 MESSAGE D.W)))

(PRINTNP
  (SLAMBDA (MESSAGE NOTTERPRI)
    (* edited: "27-Jun-87 22:17")
    (* Prints MESSAGE in the P window.
      IF NOTTERPRI is T then the message will not be on a new line)
    (IF NOTTERPRI (PRIN1 " " P.W)
      ELSE (TERPRI P.W))
    (PRIN1 MESSAGE P.W)))

(PROCESSDIMENSION
  (SLAMBDA (TYPE VECTOR BASICUNIT UNITNAME)
    (* edited: "8-Jul-87 15:05")
    (* This function processes one dimension of the unit being added. The name of the unit being added is UNITNAME. The dimension being processed is TYPE and VECTOR is the dimension vector for TYPE. BASICUNIT is the basic MKS unit for the TYPE.)
    (PRG (UNITOBJ:UNIT MENLIST FACTOR)
      (* (UNITOBJ is the unit named CHECKUNIT) (MENLIST is a list of menu options) (FACTOR is the number by which the TYPE dimension of the unit named UNITNAME must be multiplied to convert to BASICUNIT))
      [FOR CHECKUNIT IN ALLUNITS (UNITOBJ = (EVAL CHECKUNIT))
        (IF (UNITOBJ:DIMENSIONS = VECTOR)
          (MENLIST = (LIST CHECKUNIT UNITOBJ:MKS))
          (LIST (QUOTE NONE-OF-THOSE) NIL))]
      [FACTOR = (MENU (CREATE MENU (TITLE = (CONCAT TYPE " dimension of " UNITNAME " is in which?"))
        (ITEMS = MENLIST)
        (MENUPOSITION = (QUOTE 250 . 170))
        (IF (NOT FACTOR)
          (IF (YESNOMENU (CONCAT "Does the " TYPE " dimension of " UNITNAME " convert to " BASICUNIT ")
            (QUOTE 250 . 170))))
          (FACTOR = (PROMPTREAD (CONCAT "How much must the " TYPE " dimension of ") UNITNAME)"
(PROCESSUNIT)
[GLAMBDA (UNITNAME DVECTOR)]

(PROG ((CONVFATOR 1)
(CONVERTING T)
UNITOBJ:UNIT)

("CONVFATOR is the number by which this unit must be multiplied to covert it to MKSC units"
CONVERTING is a flag cleared if the unit is discovered non convertible to MKSC units (UNITOBJ is the unit named UNITNAME))

(IF ((CAR DVECTOR)
  (= 0)
  (CONVERTING + (PROCESSDIMENSION (QUOTE MASS)
  (QUOTE (1 0 0 0))
  (QUOTE kg)
  UNITNAME))

(IF CONVERTING (CONVFATOR * (TIMES CONVFATOR (EXP1 CONVERTING (CAR DVECTOR))))

(IF (AND CONVERTING ((CADDR DVECTOR)
  (= 0))
  (CONVERTING + (PROCESSDIMENSION (QUOTE LENGTH)
  (QUOTE (0 1 0 0))
  (QUOTE m)
  UNITNAME))

(IF CONVERTING (CONVFATOR * (TIMES CONVFATOR (EXP1 CONVERTING (CADDR DVECTOR))))

(IF (AND CONVERTING ((CADDDDR DVECTOR)
  (= 0))
  (CONVERTING + (PROCESSDIMENSION (QUOTE TIME)
  (QUOTE (0 0 1 0))
  (QUOTE s)
  UNITNAME))

(IF CONVERTING (CONVFATOR * (TIMES CONVFATOR (EXP1 CONVERTING (CADDDDR DVECTOR))))

(ALLUNITS --
UNITNAME)
(PRINTISP (CONCAT "unit " UNITNAME " defined")
T)

(IF CONVERTING (SET UNITNAME (A UNIT WITH DIMENSIONS = DVECTOR MKSCF = CONVFATOR)
(AUTOCONVERSION UNITNAME)
ELSE (SET UNITNAME (A UNIT WITH DIMENSIONS = DVECTOR MKSCF = NIL)))

(PROMPTREAD)

(GLAMBDA (PROMPT))

("Prints PROMPT then inputs S-expression and returns it. If the S-expression entered is one of the RESERVEDWORDS then the user is asked to re-enter input")

(PROG (INPUTREAD)
(TERPRI)
(PRIN1 PROMPT)
(TERPRI)
(PRIN1 " >> ")
(INPUTREAD = (READ T NIL NIL))
(WHILE (MEMBER INPUTREAD RESERVEDWORDS)
(PRIN1 " -- reserved word ignored --")
(TERPRI))
(PSEUDO- DSK) CALCLAB.LSP: 1  8-Jul-87 15:36:19
(PRIN1 PROMPT)
(TERPRI)
(PRIN1 "")
(INPUTREAD + (READ T NIL NIL)))
(RETURN INPUTREAD))

(REDISPLAY
[GLAMDA (self:INSTANCE)
  ("edited: "6-Jun-8713:25")
  ("Changes the units in which the instance is displayed")
  (PROG (MENULIST UNIFACTORLIST UFSELECTED)
    ("MENULIST is the list of possible items (UNIFACTORLIST is a list of the unit to be displayed and the conversion factor (UFSELECTED is the unirname-conversion factor pair selected))."
    (IF STANDARDUNIT (UNIFACTORLIST - (FINDUNITS STANDARDUNIT))
      [FOR UPAIR IN UNIFACTORLIST (MENULIST =
        (LIST (CAR UPAIR)
          (LIST (QUOTE QUOTE)
            UPAIR))
      ]
      [USELECTED = (MENU (CREATE MENU (TITLE = "Display in which units?")
        (ITEMS = MENULIST)
        (MENUPosition = (CONS (self: X INSTANCEWIDTH)
          (self: Y 80))
      (DISPLAYUNIT = (CAR USELECTED))
      (DISPLAYFACTOR = (CADR USELECTED))
      (SEND self DISPLAY))

    (REINITIALIZE
    [GLAMDA NIL
      ("edited: "8-Jun-8710:31")
      ("Restarts CalcLab")
      (SavESTATE))
      (MAKEALLFILES)
      (CLOSE W.
        (CALCLAB)))
  )

(RENAMEFRAME
[GLAMDA (self:FRAME)
  ("edited: "6-Apr-8701:43")
  ("Changes the name of the frame")
  ("FRNAME is the new name for the frame")
  ("FRNAME is an instance in the frame")
  ("FRDEF is the name of the frame definition")
  (NEWNAME + (PROMPTREAD (CONCAT "What is the new name for "
    self:NAME ".")))
  (if (NOT (MEMBER NEWNAME (APPEND ALLCONSTANTS ALLFRAMES ALLFRDFS ALLINSTANCES
    ALLSTATES ALLUNITS ALLVARS))
    (ALLFRAMES = (SUBST NEWNAME self:NAME ALLFRAMES))
    (FOR INSTNAME IN INSTANCES (FRAMEINST + (EVAL INSTNAME))
      (FRAMEINST:FRAME + NEWNAME))
    (FRDEF = (EVAL self:FRAMEDF))
    (FRDEF:FRAMES = (SUBST NEWNAME self:NAME FRDEF:FRAMES))
    (self:NAME = NEWNAME)
    (SET NEWNAME self)
    (SEND self DRAW)
    ELSE (PRINT "--- this name already being used ---")

(RENAMEINSTANCE
[GLAMDA (self:INSTANCE)
  ("edited: "6-Apr-8701:45")
  ("Changes the name of the instance")
  ("NEWNAME is the new name")
  ("INSTFRAME is the frame containing the instance")
  ("INSTCOMP is a computation linked to the instance")
  ("LINK is a link connected to the instance")
  (NEWNAME + (PROMPTREAD (CONCAT "What is the new name for "
    self:NAME ".")))
  (if (NOT (MEMBER NEWNAME (APPEND ALLCONSTANTS ALLFRAMES ALLFRDFS ALLINSTANCES
    ALLSTATES ALLUNITS ALLVARS))
    (ALLINSTANCES = (SUBST NEWNAME self:NAME ALLINSTANCES))
    (IF FRAME (INSTFRAME = (EVAL FRAME))
      (INSTFRAME:INSTANCES = (SUBST NEWNAME self:NAME INSTFRAME:INSTANCES))
    (FOR COMPUTATION (INSTCOMP = (EVAL COMPUTATION))
      (INSTCOMP:INSTANCECOMPUTED = NEWNAME))
  )
(SAVESTATE
  (GLAMBDA NIL
    (* edited: "8-Jul-87 15:36:19")
    (* Saves the current state in a list that can be later loaded)
    (PROG (LISTNAME VLIST FLIST ILIST CLIST LLIST CONSLIST UNITLIST FRDEFLIST FRAMELIST EQONLIST)
      ("LISTNAME is the name of the list in which the state is saved")
      (VLIST is the sub-list containing variables)
      (FLIST is the sub-list containing formulas) (ILIST is the sub-list containing instances)
      (CLIST is the sub-list containing computations) (LLIST is the sub-list containing links)
      (CONSLIST is the sub-list containing constants) (UNITLIST is the sub-list containing units)
      (FRDEFLIST is the sub-list containing frame definitions)
      (FRAMELIST is the sub-list containing frames) (EQONLIST is the sub-list containing the names of all equations)
      (PROMPTREAD "Retain under what name??")
      (WHILE (NO MEMBER LISTNAME APPEND ALLCONSTANTS ALLFRDEFS ALLVARS ALLUNITS
               ALLINSTANCES ALLFRAMES)
        (PRINT "--- name already in use --- ignored ---")
        (LISTNAME = (PROMPTREAD "Retain under what name??"))
      (IF (BOUNDP LISTNAME)
        (PRINTNP (CONCAT "Overwriting older " LISTNAME))
        (IF (NOT (MEMBER LISTNAME ALLSTATES))
          (ALLSTATES =
            LISTNAME))
        (FOR ITEMNAME IN ALLVARS (VLIST =
          LIST ITEMNAME (EVAL ITEMNAME))
        (FOR ITEMNAME IN ALLFORMULAS (FLIST =
          LIST ITEMNAME (EVAL ITEMNAME))
        (FOR ITEMNAME IN ALLCONSTANTS (CONSLIST =
          LIST ITEMNAME (EVAL ITEMNAME))
        (FOR ITEMNAME IN ALLUNITS (UNITLIST =
          LIST ITEMNAME (EVAL ITEMNAME))
        (FOR ITEMNAME IN ALLFRDEFS (FRDEFLIST =
          LIST ITEMNAME (EVAL ITEMNAME))
        (FOR ITEMNAME IN ALLALEQUATIONS (EQONLIST =
          LIST ITEMNAME (EVAL ITEMNAME))
        (IF (YESNOMENU "Also save work area?" (QUOTE (300 . 170)))
          (FOR ITEMNAME IN ALLINSTANCES (ILIST =
            LIST ITEMNAME (EVAL ITEMNAME))
          (FOR ITEMNAME IN ALLCOMPUTATIONS (CLIST =
            LIST ITEMNAME (EVAL ITEMNAME))
          (FOR ITEMNAME IN ALLLINKS (LLIST =
            LIST ITEMNAME (EVAL ITEMNAME))
          (FOR ITEMNAME IN ALLFRAMES (FRAMELIST =
            LIST ITEMNAME (EVAL ITEMNAME))
          (SET LISTNAME (HCOPYALL (LIST VLIST FLIST ILIST CLIST LLIST CONSLIST UNITLIST FRDEFLIST FRAMELIST ALLVARS ALLUNITS ALLALIST ALLFRAMES ALLNAMES) (DATE)
            (PRINTNP (CONCAT LISTNAME " retained")
            (IF (YESNOMENU (CONCAT LISTNAME " to be saved in file?"))
              (QUOTE (300 . 170)))
              (MAKESTATEFILE LISTNAME)))
        (SELECTINSTANCE
          (GLAMBDA (self:INSTANCE)
            ("edited: "27-Jun-87 22:18")
            (* Sets SELECTEDINSTANCE to this instance)
            (PRINTNP (CONCAT self) (" selected")
            (QUOTE 1)))
        (SELECTSTANUNIT
          (GLAMBDA (VARIABLENAME)
            ("edited: "8-Jul-87 15:00")
            NULL))
      (END)
(* This function asks the user to specify the standard unit for the variable VARIABLENAME. It returns the name of the unit. The user can either select one of the already known units or add a new unit as the standard unit.*)

(PROG (MENULIST SELECTEDUNITNAME)
  (* MENULIST is a list of menu options
     (SELECTEDUNITNAME is the name of the unit specified by
     the user))

  (FOR UNITNAME
    IN (SORT ALLUNITS)
      (MENULIST =
        (LIST UNITNAME (LIST (QUOTE QUOTE) UNITNAME)
         (MENULIST =
           (LIST (QUOTE NEW-STANDARD-UNIT) NIL))
        (SELECTEDUNITNAME = (MENU CREATE MENU TITLE
          (COMCAT "Select standard unit for " VARIABLENAME)
          (ITEMS MENULIST)
          (MENUPOSITION = (QUOTE 310 . 170))
        (IF (NOT SELECTEDUNITNAME)
          (SELECTEDUNITNAME = (ADDUNIT VARIABLENAME)))
        (RETURN SELECTEDUNITNAME)))))

(SETVALUE
  (GLAMBA (self:INSTANCE NEWVALUE)
    (* Set the value of the instance to NEWVALUE. If NEWVALUE is a different value than the older value then all computations that use the instance are recomputed.*)

    (PROG (COMP:COMPUTATION)
      (* COMP is the computation that is recomputed)
      (IF OR (NEWVALUE > (VALUE + (TIMES VALUE ALLOWEDERROR)))
        (NEWVALUE < (VALUE - (TIMES VALUE ALLOWEDERROR))
          (VALUE = NEWVALUE)
          (SEND self DISPLAY)
          (FOR COMPNAME IN USEDBY (COMP = (EVAL COMPNAME))
            (SEND COMP COMPUTE)))
    )

(SHRINKFRAME
  (GLAMBA (self:FRAME)
    (* SHRINKFRAME shrinks the frame)

    (PROG (INST:INSTANCE)
      (DSPFILL (CREATEREGION self:X self:Y (self:DX + 2)
        (self:DY + 2))
        (QUOTE REPLACE)
        self:X
        self:Y
        (self:DX = SHRUNKHEIGHT)
        (self:EXPANDED = NIL)
        (FOR INSTNAME IN INSTANCES (INST = (EVAL INSTNAME))
          (INST:EXPANDED = NIL))
        (FOR INSTNAME IN INSTANCES (INST = (EVAL INSTNAME))
          (INST:X = self:X)
          (INST:X = (self:X + 3))
        (SEND INST SHRINK)
        )
    )

(SHRINKINSTANCE
  (GLAMBA (self:INSTANCE)
    (* SHRINKINSTANCE shrinks the box for the instance)

    (IF (NOT FRAME)
      (DSPFILL (CREATEREGION X Y (DX + 1)
        (DY + 1))
      (QUOTE REPLACE)
      X
      Y
      (DY = SHRUNKHEIGHT)
      (EXPANDED = NIL)
      (SEND self MOVE X Y))
    )

(SIMP-ANTLOG
  (GLAMBA (ARG)
    (IF (NUMBERP ARG)
      (* SIMP-ANTLOG is the number of the argument*)
      (QUOTE ARG))
    )

(* SIMP-ANTLOG is the number of the argument*)

(* Shrinks the box for the instance*)

(* SHRINKFRAME shrinks the frame*)

(* Set the value of the instance to NEWVALUE. If NEWVALUE is a different value than the older value then all computations that use the instance are recomputed.*)

(* COMP is the computation that is recomputed*)

(* SHRINKFRAME shrinks the frame*)

(* SHRINKINSTANCE shrinks the box for the instance*)

(* SIMP-ANTLOG is the number of the argument*)
(PSEUDO-DSK)CALCLAB.LSP:1  8-Jul-87 15:38:19

  (ANTilog ARG)
  ELSE (LIST (QUOTE ANTilog) ARG)))

(SIMP-ARCC)
  (GLANDBA (ARG))
  (IF (NUMBERP ARG)
    (ARCCOS ARG)
    ELSE (LIST (QUOTE ARCCOS) ARG)))))

(SIMP-ARC)
  (GLANDBA (ARG))
  (IF (NUMBERP ARG)
    (ARCSIN ARG)
    ELSE (LIST (QUOTE ARCSIN) ARG)))))

(SIMP-ARCT)
  (GLANDBA (ARG))
  (IF (NUMBERP ARG)
    (ARCTAN ARG)
    ELSE (LIST (QUOTE ARCTAN) ARG)))

(SIMP-COS)
  (GLANDBA (ARG))
  (IF (NUMBERP ARG)
    (COS ARG)
    ELSE (LIST (QUOTE COS) ARG)))

(SIMP-DIFF)
  (GLANDBA (ARG1 ARG2))
  (IF (AND (NUMBERP ARG1) (NUMBERP ARG2))
    (DIFFERENCE ARG1 ARG2)
    ELSEIF (ARG1 = 0)
      (LIST (QUOTE MINUS) ARG2)
    ELSEIF (ARG2 = 0)
      ARG1
    ELSE (LIST (QUOTE DIFFERENCE) ARG1 ARG2)))

(SIMP-ERROR)
  (GLANDBA (MESSAGE))
  (PRINT (CONCAT "--- error detected while simplifying ---" MESSAGE))
  (RINGBELLS))

(SIMP-EXPT)
  (GLANDBA (ARG1 ARG2))
  (IF (AND (NUMBERP ARG1) (NUMBERP ARG2))
    (EXPT ARG1 ARG2)
    ELSEIF (ARG1 = 0)
      0
    ELSEIF (ARG2 = 0)
      1
    ELSEIF (ARG1 = 1)
      ARG1
    ELSE (LIST (QUOTE EXPT) ARG1 ARG2)))

(SIMP-QUO)
  (GLANDBA (ARG1 ARG2))
  (IF (ARG2 = 0)
    (SIMP-ERROR "--- divide by zero ---")
  ELSEIF (AND (NUMBERP ARG1) (NUMBERP ARG2))
    (DIVIDE ARG1 ARG2)
  ELSE (LIST (QUOTE QUO) ARG1 ARG2)))

("edited: "23-Jan-8714:03")

("edited: "23-Jan-8714:02")

("edited: "11-Feb-8715:10")

("edited: "23-Jan-8714:01")

("edited: "23-Jan-8713:48")

("edited: "23-Jan-8714:13")

("Prints MESSAGE to indicate an error detected in simplification.")

("edited: "23-Jan-8714:00")

("edited: "23-Jan-8713:55")
(PSEUDO-DISKCALCLAB.LSP:1 5-Jul-87 15:30:19)

(FQUOTIENT ARG1 ARG2)
ELSEF (ARG1 = 0) 0
ELSEF (ARG2 = 1) ARG1
ELSE (LIST (QUOTE FQUOTIENT)
ARG1 ARG2)))

(SIMP-NINIT)
(GLAMBDA NIL

(DEFLIST (QUOTE ((DIFFERENCE SIMP-DIFF)
(PLUS SIMP-PLUS)
(TIMES SIMP-TIMES)
(FQUOTIENT SIMP-FQUO)
(SQRT SIMP-SQRT)
(MINUS SIMP-MINUS)
(EXPT SIMP-EXP)
(SIN SIMP-SIN)
(COS SIMP-COS)
(TAN SIMP-TAN)
(ARCOSIMP-ARCS)
(ARCOSIMP-ARCC)
(ARCTAN SIMP-ARCT)
(LOG SIMP-LOG)
(ANTILOG SIMP-ANTILOG)))

(QUOTE SIMPLIFYFN))))

(SIMP-LOG)
(GLAMBDA (ARG)

(IF (NUMBERP ARG)
(IF (ARG < 0)
(SIMP-ERROR "--- log of non-positive number ---")
ELSE (LOG ARG)))
ELSE (LIST (QUOTE (LOG)
ARG))))

(SIMP-MINUS)
(GLAMBDA (ARG)

(IF (NUMBERP ARG)
(MINUS ARG)
ELSE (LIST (QUOTE MINUS)
ARG))))

(SIMP-PLUS)
(GLAMBDA (ARG1 ARG2)

(IF (AND (NUMBERP ARG1)
(NUMBERP ARG2))
(PLUS ARG1 ARG2)
ELSEF (ARG1 = 0)
ARG2
ELSEF (ARG2 = 0)
ARG1
ELSE (LIST (QUOTE PLUS)
ARG1 ARG2)))

(SIMP-SIN)
(GLAMBDA (ARG)

(IF (NUMBERP ARG)
(SIN ARG)
ELSE (LIST (QUOTE SIN)
ARG))))

(SIMP-SQRT)
(GLAMBDA (ARG)

(IF (NUMBERP ARG)
(IF (ARG < 0)
(SIMP-ERROR "--- square root of negative number ---")
ELSE (SORT ARG))
ELSE (LIST (QUOTE SQRT)
ARG))))

(SIMP-TAN)
(GLAMBDA (ARG)

("edited: "20-Apr-8719:06")
("initiates the SIMPLIFYFN properties")

("edited: "20-Apr-8719:06")

("edited: "23-Jan-8713:58")

("edited: "23-Jan-8713:50")

("edited: "23-Jan-8714:00")

("edited: "15-Feb-8714:27")

("edited: "23-Jan-8714:01")
(PSEUDO-DSK)CALCLAB.LSP:1 R-Jul-87 15:36:19

(IF (NUMBERP ARG)
  (TAN ARG)
  ELSE (LIST (QUOTE TAN)
            ARG)))

(SIMP-TIMES
 (GLAMBA (ARG1 ARG2)
 (IF (AND (NUMBERP ARG1)
           (NUMBERP ARG2))
    (TIMES ARG1 ARG2)
    ELSEF (ARG1 = 0)
            0
    ELSEF (ARG2 = 0)
            0
    ELSEF (ARG1 = 1)
            ARG2
    ELSEF (ARG2 = 1)
            ARG1
    ELSE (LIST (QUOTE TIMES)
               ARG1 ARG2)))))

(SIMPLIFY
 (GLAMBA (EXPRESSION))

  (PROG (OPERATOR ARG1 ARG2)

    (IF (NULL EXPRESSION)
         (RETURN NIL)
    ELSEF (ATOM EXPRESSION)
         (RETURN EXPRESSION)
    ELSEF (NULL (CAR EXPRESSION))
         (RETURN NIL)
    ELSE (OPERATOR = (CAR EXPRESSION))
         (ARG1 = (SIMPLIFY (CADR EXPRESSION)))
         (ARG2 = (SIMPLIFY (CADDR EXPRESSION)))
         (RETURN (APPLY* (GETPROP OPERATOR (QUOTE SIMPLIFYFN))
                         ARG1 ARG2)))

(SUBSTFIRST
 (GLAMBA (NEW OLD EXPRESSION))

  (IF (NOT EXPRESSION)
       NIL
  ELSEF (EQUAL EXPRESSION OLD)
       NEW
  ELSEF (CONTAINS? (CADR EXPRESSION) OLD)
       (LIST (CAR EXPRESSION)
              (SUBSTFIRST NEW OLD (CADDR EXPRESSION))
              (CADDR EXPRESSION))
  ELSE (LIST (CAR EXPRESSION)
           (CADDR EXPRESSION)
           (SUBSTFIRST NEW OLD (CADDR EXPRESSION))
          )

(TESTANDEVAL
 (GLAMBA (EXPRESSION))

  ("This function evaluates the Lisp expression EXPRESSION. If the expression would cause division by zero, then it prints an error message and returns NIL. Otherwise, it returns the value of the expression.")

  (PROG (OPERATOR ARG1 ARG2)

    (IF ((LENGTH EXPRESSION) = 3)
         (OPERATOR = (CAR EXPRESSION))
         (ARG1 = (TESTANDEVAL (CADR EXPRESSION)))
         (ARG2 = (TESTANDEVAL (CADDR EXPRESSION)))
         (IF (AND ARG1 ARG2)
             (IF (AND OPERATOR = (QUOTE FQUOTIENT))
                  (ARG2 = 0)
                  (PRINT "--- attempt to divide by zero ---"))))

  ("OPERATOR is the name of the top level function"
   (ARG1 and ARG2 are the arguments to the operator))

  (RETURN)
(RETURN NIL)
ELSE (RETURN (APPLY* OPERATOR ARG1 ARG2)))
ELSE (RETURN NIL))
ELSEIF ((LENGTH EXPRESSION) > 2)
(OPERATOR = (CAR EXPRESSION))
(ARG1 = (TESTANDVAL (CADR EXPRESSION)))
(IF ARG1 (RETURN (APPLY* OPERATOR ARG1))
ELSE (RETURN NIL))
ELSE (RETURN EXPRESSION))

(TESTFORSELECTION)
("GLAMBA (WINDOW"
(* Determines if an instance or computation was selected in WINDOW, and if one was, determines what action to take
with the instance or computation)

(PROG (XCOORD YCOORD CURPOSITION TESTINSTANCE:INSTANCE TESTCOMPUTATION:COMPUTATION
TESTFRAME:FRAME)

(* (XCOORD is the x coordinate of the cursor position) (YCOORD is the y coordinate of the cursor position)
CURPOSITION is the cursor position) (TESTINSTANCE is the instance being tested) (TESTCOMPUTATION is the
computation being tested) (TESTFRAME is the frame being tested))

(CURPOSITION = (CURSORPOSITION NIL W.W))
(XCOORD = (CAR CURPOSITION))
(YCOORD = (CDR CURPOSITION))

FOR FRAMENAME
IN ALLFRAMES (TESTFRAME = (EVAL FRAMENAME))
(IF (INSIDEBOX TESTFRAME:X ((TESTFRAME:Y + TESTFRAME:DY) - 15)
TESTFRAME:DX 10 XCOORD YCOORD)
(XCOORD = WINDOWWIDTH)
(YCOORD = WINDOWWIDTH)
(MENU (CREATE MENU TITLE = (CONCAT *For* FRAMENAME))
(ITEMS = (LIST (LIST (QUOTE DELETE)
(QUOTE TESTFRAME))
"Delete this frame from the current state")

LIST (QUOTE SAVE)
(QUOTE TESTFRAME)
"Save state before removing this frame")

IF TESTFRAME:EXPANDED
LIST (QUOTE SHRINK)
(QUOTE TESTFRAME)
"
Shrink this frame to conserve work area space")
ELSE LIST (QUOTE MOVE)
(QUOTE TESTFRAME)
"
"Move this frame to another location in the work area")

FOR INSTANCENAME
IN ALLINSTANCES (TESTINSTANCE = (EVAL INSTANCENAME))
(IF (INSIDEBOX TESTINSTANCE:X TESTINSTANCE:Y TESTINSTANCE:DX TESTINSTANCE:DY
XCOORD YCOORD)
(XCOORD = WINDOWWIDTH)
(YCOORD = WINDOWWIDTH)
(IF (AND TESTINSTANCE:FRAME (NOT TESTINSTANCE:EXPANDED))
ELSE MENU (CREATE MENU TITLE = (CONCAT *For* INSTANCENAME))
(ITEMS =
"Expand this frame and make its instances visible")
LIST (QUOTE MOVE)
(QUOTE TESTFRAME)
"
"Move this frame to another location in the work area")
(LIST
  (LIST (QUOTE DELETE)
        (LIST (QUOTE DELETEINSTANCE)
              (QUOTE TESTINSTANCE))

"Delete this state from the current state")
  (LIST (QUOTE RENAME)
        (LIST (QUOTE RENAMENAME:INSTANCE)
              (QUOTE TESTINSTANCE))

"Change the name of this instance")
  (IF (NOT TESTINSTANCE:FRAME)
      (LIST (QUOTE MOVE)
            (LIST (QUOTE MOVEINSTANCE)
                  (QUOTE TESTINSTANCE))

"Move this instance to another location in the work area")
  ELSE (LIST (QUOTE MOVE)
           (LIST (QUOTE PRINT))

"This instance cannot be moved outside its frame")
  (IF TESTINSTANCE:EXPANDED
      (LIST (QUOTE SHRINK)
            (LIST (QUOTE SHRINKINSTANCE)
                  (QUOTE TESTINSTANCE))

"Shrink the instance to conserve work area space")
  ELSE (LIST (QUOTE EXPAND)
        (LIST (QUOTE EXPANDINSTANCE)
              (QUOTE TESTINSTANCE))

"Expand the instance and display its value")
  (LIST (QUOTE USE)
        (LIST (QUOTE EQUATEINSTANCE)
              (QUOTE TESTINSTANCE))

"Constrains the value of this instance to the value of some instance to be selected by the user")
  (LIST (QUOTE COMPUTE)
        (LIST (QUOTE COMPUTEINSTANCE)
              (QUOTE TESTINSTANCE))

"Makes a computation for this instance")
  (LIST (QUOTE CHANGE-UNITS)
        (LIST (QUOTE REDISPLAY)
              (QUOTE TESTINSTANCE))

"Display the value of this instance in some other units")
  (LIST (QUOTE INPUT-VALUE)
        (LIST (QUOTE INPUTVALUE)
              (QUOTE TESTINSTANCE))

"Input value of this instance from keyboard")
  (LIST (QUOTE SELECT)
        (LIST (QUOTE SELECTINSTANCE)
              (QUOTE TESTINSTANCE))

"Select this instance for use as the currently needed instance in the computation being created"
  (FOR COMPUTATIONNAME
       IN ALLCOMPUTATIONS
            TESTCOMPUTATION = (EVAL COMPUTATIONNAME))
  (IF (INSDEREAUX TESTCOMPUTATION:Y TESTCOMPUTATION:X TESTCOMPUTATION:DZ
              TESTCOMPUTATION:DY XCOORD YCOORD)
      (MENU (CREATE MENU (TITLE = (CONCAT "For " COMPUTATIONNAME))))
```
(PSEUDO-DSK)CALCLAB.LSP:1 8-Jul-87 15:36:19

(ITEMS = (LIST (QUOTE DELETE)
            (LIST (QUOTE
                (DELETECOMPUTATION)
                (QUOTE TESTCOMPUTATION)
            ))
        )

"Delete this computation from the current state"
(ITEMS = (LIST (QUOTE DISPLAY)
            (LIST (QUOTE
                (DISPLAYCOMPUTATION)
                (QUOTE TESTCOMPUTATION)
            ))
        )

"Display the equation and actual expression of this computation"

(TRANSLATE
  (LAMBDA (EXPRESSION)
    (* edited: "17-Apr-871:01"
      (* Returns EXPRESSION with infix operators replaced by
         prefix Lisp operators)
      (* OPERATOR is the infix operator)

    (PROG (OPERATOR)
      (IF ((LENGTH EXPRESSION) = 3)
          (OPERATOR = (CADR EXPRESSION))
          (IF (OPERATOR = (QUOTE +))
              (RETURN (LIST (QUOTE PLUS))
                (TRANSLATE (CAR EXPRESSION))
                (TRANSLATE (CADR EXPRESSION))
            )
          ELSEIF (OPERATOR = (QUOTE -))
              (RETURN (LIST (QUOTE DIFFERENCE))
                (TRANSLATE (CAR EXPRESSION))
                (TRANSLATE (CADR EXPRESSION))
            )
          ELSEIF (OPERATOR = (QUOTE *))
              (RETURN (LIST (QUOTE TIMES))
                (TRANSLATE (CAR EXPRESSION))
                (TRANSLATE (CADR EXPRESSION))
            )
          ELSEIF (OPERATOR = (QUOTE /))
              (RETURN (LIST (QUOTE QUOTIENT))
                (TRANSLATE (CAR EXPRESSION))
                (TRANSLATE (CADR EXPRESSION))
            )
          ELSEIF ((CAR EXPRESSION) = (QUOTE **))
              (RETURN (LIST (QUOTE EXPT))
                (TRANSLATE (CAR EXPRESSION))
                (TRANSLATE (CADR EXPRESSION))
            )
          ELSEIF ((CAR EXPRESSION) = (QUOTE quot))
              (RETURN (LIST (QUOTE EXPT))
                (TRANSLATE (CAR EXPRESSION))
                (TRANSLATE (CADR EXPRESSION))
            )
          ELSEIF ((LENGTH EXPRESSION) = 2)
              (OPERATOR = (CAR EXPRESSION))
              (IF (OPERATOR = (QUOTE sqrt))
                  (RETURN (LIST (QUOTE SQRT))
                    (TRANSLATE (CAR EXPRESSION))
                  )
              ELSEIF (OPERATOR = (QUOTE Ŕ))
                  (RETURN (LIST (QUOTE MINUS))
                    (TRANSLATE (CADR EXPRESSION))
                  )
              ELSEIF (OPERATOR = (QUOTE sin))
                  (RETURN (LIST (QUOTE SIN))
                    (TRANSLATE (CADR EXPRESSION))
                  )
              ELSEIF (OPERATOR = (QUOTE cos))
                  (RETURN (LIST (QUOTE COS))
                    (TRANSLATE (CADR EXPRESSION))
                  )
              ELSEIF (OPERATOR = (QUOTE tan))
                  (RETURN (LIST (QUOTE TAN))
                    (TRANSLATE (CADR EXPRESSION))
                  )
              ELSE (RETURN (LIST (CAR EXPRESSION))
                (TRANSLATE (CADR EXPRESSION))
            )
          ELSE (RETURN EXPRESSION))
    )
  )
)
Bibliography

[Barstow1] Barstow, D.
"Domain Specific Automatic Programming"
*IEEE Transactions on Software Engineering*, SE-11 (11), November 1985, pp. 1321-1336

[Barstow2] Barstow, D., H. Shrobe
"From Interactive to Intelligent Programming Environments"
in Barstow, D., H. Shrobe, E. Sandewall (eds.)

[Barstow3] Barstow, D.
"A Perspective on Automatic Programming"
*The AI Magazine*, Spring 1984, pp. 5-27

[Bennet] Bennet, G.

[Borning] Borning, A.
"ThingLab: A Constraint-Oriented Simulation Laboratory"
Stanford Computer Science Department Report STAN-CS-79-746, Stanford University, July 1979
[diSessa]  diSessa, A.
"Notes on the Future of Programming: Breaking the Utility Barrier"
in Norman, D., S. Draper (eds.)

[Hutchins]  Hutchins, E., J. Hollan, D. Norman
"Direct Manipulation Interfaces"
in Norman, D., S. Draper (eds.)

[Minsky]  Minsky, M.
"A Framework for Representing Knowledge"
in Haugland, J. (ed.)

[Norman]  Norman, D.
"Cognitive Engineering"
in Norman, D., S. Draper (eds.)

[Novak1]  Novak, G.
"GLISP: A LISP-Based Programming System with Data Abstraction"
The AI Magazine, Fall 1983, pp. 37-47
[Novak2] Novak, G.
GLISP User's Manual
Technical Report TR-83-25 (Preliminary Version),
Department of Computer Sciences, University of Texas
at Austin, December 1983

[Novak3] Novak, G.
"Representations of Knowledge in a Program for
Solving Physics Problems"
Proceedings of the Fifth International Joint Conference
on Artificial Intelligence, MIT, Cambridge, August 1977,
pp. 286-291

[Resnick] Resnick, R., D. Halliday
Physics, Part I, John Wiley & Sons, 1966

[Sheil] Sheil, B.
"Power Tools for Programmers"
in Barstow, D., H. Shrobe, E. Sandewall (eds.)
Interactive Programming Environments, McGraw-Hill,
1984, pp. 19-30

[Shneiderman] Shneiderman, B.
"Direct Manipulation: A Step Beyond Programming
Languages"
IEEE Computer, 16 (8), August 1983, pp. 57-69

[Sutherland] Sutherland, I.
"Sketchpad: A Man-Machine Graphical Communication
System"
Technical Report No. 296, Lincoln Laboratory, MIT,
January 1963
[Thomas] Thomas, T.
Financial Decision Making with VisiCalc and SuperCalc,
Prentice Hall, 1985

[Vose] Vose, G., G. Williams
"LabVIEW: Laboratory Virtual Instrument Engineering
Workbench"
BYTE, September 1986, pp. 84-92

[Winograd] Winograd, T.
"Frame Representations and the Declarative/Procedural
Controversy"
in Bobrow, D., A. Collins (eds.)
Representation and Understanding: Studies in

[Xerox] Xerox Corporation
Interlisp-D Reference Manual,
Xerox Corporation, October 1985
Vita

Yusuf Nazir Mauladad, the son of Yasmine and Nazir Ahmed Mauladad, was born in Sargodha, Pakistan, on March 8, 1962. He was schooled first at St. George’s Primary, Nairobi, and then at Aitchison College, Lahore, where he was awarded the Higher School Certificate by the University of Cambridge in December 1979. In January, 1981, he entered The University of Texas at Austin from where he received the Bachelor of Science in Electrical Engineering with High Honors in December 1983. He entered the Graduate School of The University of Texas at Austin in January, 1984.

Permanent Address: P. O. Box 40462
Nairobi
Kenya

This thesis was typed by the author using a Xerox Viewpoint workstation.