

**Radio Shack<sup>®</sup>**



***TK!Solver<sup>™</sup>***  
***from***  
***Software Arts***

READ THIS AGREEMENT CAREFULLY. USE OF THIS PRODUCT CONSTITUTES YOUR ACCEPTANCE OF THE TERMS AND CONDITIONS OF THIS AGREEMENT.

**LIMITED WARRANTY**

This Software program is licensed on an "AS IS" basis, without warranty. The original CUSTOMER's exclusive remedy, in the event of a manufacturing defect in the diskette, is its repair or replacement within thirty (30) calendar days of the date of the Radio Shack sales document received upon the license of the Software. The defective diskette shall be returned to a Radio Shack Computer Center, a Radio Shack retail store, participating Radio Shack franchisee or Radio Shack dealer along with the sales document.

EXCEPT AS PROVIDED HEREIN, RADIO SHACK AND SOFTWARE ARTS, INC. MAKE NO EXPRESS WARRANTIES, AND ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE IS LIMITED IN ITS DURATION TO THE DURATION OF THE WRITTEN LIMITED WARRANTIES SET FORTH HEREIN.

Some states do not allow limitations on how long an implied warranty lasts, so the above limitation(s) may not apply to CUSTOMER.

**DISCLAIMER OF LIABILITY**

RADIO SHACK AND SOFTWARE ARTS, INC. SHALL HAVE NO LIABILITY OR RESPONSIBILITY TO CUSTOMER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY "SOFTWARE" LICENSED OR FURNISHED BY RADIO SHACK AND/OR SOFTWARE ARTS, INC., INCLUDING, BUT NOT LIMITED TO, ANY INTERRUPTION OF SERVICE, LOSS OF BUSINESS OR ANTICIPATORY PROFITS OR INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES.

Some states do not allow the limitation or exclusion of consequential damages, so the above limitation(s) or exclusion(s) may not apply to CUSTOMER.

**SOFTWARE LICENSE**

RADIO SHACK and SOFTWARE ARTS, INC. grant to CUSTOMER a non-exclusive, paid-up license to use this Software on *one* computer subject to the following provisions:

A. Except as otherwise provided in this Software License, applicable copyright laws shall apply to the Software.

B. Title to the medium on which the Software is recorded (diskette) is transferred to CUSTOMER, but not title to the Software.

C. CUSTOMER shall not use, make, manufacture or reproduce copies of the Software except for use on *one* computer and except as is specifically provided in this Software License. CUSTOMER is expressly prohibited from disassembling the Software.

D. CUSTOMER is permitted to make six additional copies of the Software *only* for back-up or archival purposes or if such additional copies are required in the operation of *one* computer with the Software, but only to the extent the Software documentation allows back-up copies to be made.

E. All copyright notices shall be retained on all copies of the Software.

F. CUSTOMER is permitted to transfer unmodified Software and all copies of the Software which have been made to a single third party provided CUSTOMER has paid for the Software to be transferred. In any event the provisions of this License and Warranty Agreement shall be applicable to a third party receiving the Software from CUSTOMER.

THIS LICENSE AND WARRANTY AGREEMENT CONSTITUTES THE ENTIRE AGREEMENT AND UNDERSTANDING BETWEEN THE PARTIES AND SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATION RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

The warranties granted herein give the *original* CUSTOMER specific legal rights, and the *original* CUSTOMER may have other rights which vary from state to state.



## READ ME FIRST

All computer software is subject to change, correction, or improvement as the manufacturer receives customer comments and experiences. Radio Shack has established a system to keep you immediately informed of any reported problems with this software, and the solutions. We have a customer service network including representatives in many Radio Shack Computer Centers, and a large group in Fort Worth, Texas, to help with any specific errors you may find in your use of the programs. We will also furnish information on any improvements or changes that are "cut in" on later production versions.

To take advantage of these services, you must do three things:

- (1) Send in the postage-paid software registration card included in this manual immediately. (Postage must be affixed in Canada.)
- (2) If you change your address, you must send us a change of address card (enclosed), listing your old address exactly as it is currently on file with us.
- (3) As we furnish updates or "patches", and you update your software, you must keep an accurate record of the current version numbers on the logs below. (The version number will be furnished with each update.)

Keep this card in your manual at all times, and refer to the current version numbers when requesting information or help from us. Thank you.

### APPLICATIONS SOFTWARE VERSION LOG

01.00.00

_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

### OP. SYSTEM VERSION LOG

_____
_____
_____
_____
_____
_____
_____
_____



# Read Carefully

In order for us to notify you of modifications or updates to this program you **MUST** complete this card and return it immediately. This card gets you information only and is **NOT** a warranty registration. Register one software package per card only. The registration card is postage paid—it costs you nothing to mail.

Two change of address cards have been included so that you may continue to receive information in the event that you move. Copy all address information from the Registration Card onto them prior to sending the Registration Card. They must show your "old address" exactly as you originally registered it with us.

## Software Registration Card

Cat. No. 26-1630

Version 01.00.00

Name \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ Phone (    ) \_\_\_\_\_ - \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_



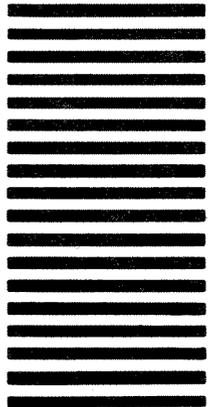
NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 138 FORT WORTH, TEXAS

POSTAGE WILL BE PAID BY ADDRESSEE

**Software Registration  
Data Processing Dept.  
P.O. Box 2910  
Fort Worth, Texas 76113-9965**



# Change of address

NEW ADDRESS

Name \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_ Phone (    ) \_\_\_\_\_ - \_\_\_\_\_  
State \_\_\_\_\_ Zip \_\_\_\_\_

OLD ADDRESS

Name \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_ Phone (    ) \_\_\_\_\_ - \_\_\_\_\_  
State \_\_\_\_\_ Zip \_\_\_\_\_

# Change of address

NEW ADDRESS

Name \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_ Phone (    ) \_\_\_\_\_ - \_\_\_\_\_  
State \_\_\_\_\_ Zip \_\_\_\_\_

OLD ADDRESS

Name \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_ Phone (    ) \_\_\_\_\_ - \_\_\_\_\_  
State \_\_\_\_\_ Zip \_\_\_\_\_

**PLACE  
STAMP  
HERE**

**Software Registration  
Data Processing Dept.  
P.O. Box 2910  
Fort Worth, Texas 76113-9965**

**PLACE  
STAMP  
HERE**

**Software Registration  
Data Processing Dept.  
P.O. Box 2910  
Fort Worth, Texas 76113-9965**

### Changes and Corrections

This Changes and Corrections page contains changes and corrections to the TK!Solver® Instruction Manual.

#### Page 5-11

The last line on the bottom of this page should contain a blue **A**.

#### Page 6-14

The values on the Variable subsheet are actually rounded to 11 significant digits, not 10 as stated.

#### Page 12-17

The unit for the output value of the variable *s* should be “mi/h,” not “mi/hr.”

#### Page 14-2

After you load MUNITs into the MORTGAGE model, you must re-display the Variable Sheet in a single window before saving the model. Do **not** re-display the Rule Sheet as instructed.

By supplying these changes and corrections, Software Arts, Inc. is neither waiving any of its rights nor expanding the scope of any of its obligations under the license agreement for the TK!Solver program.



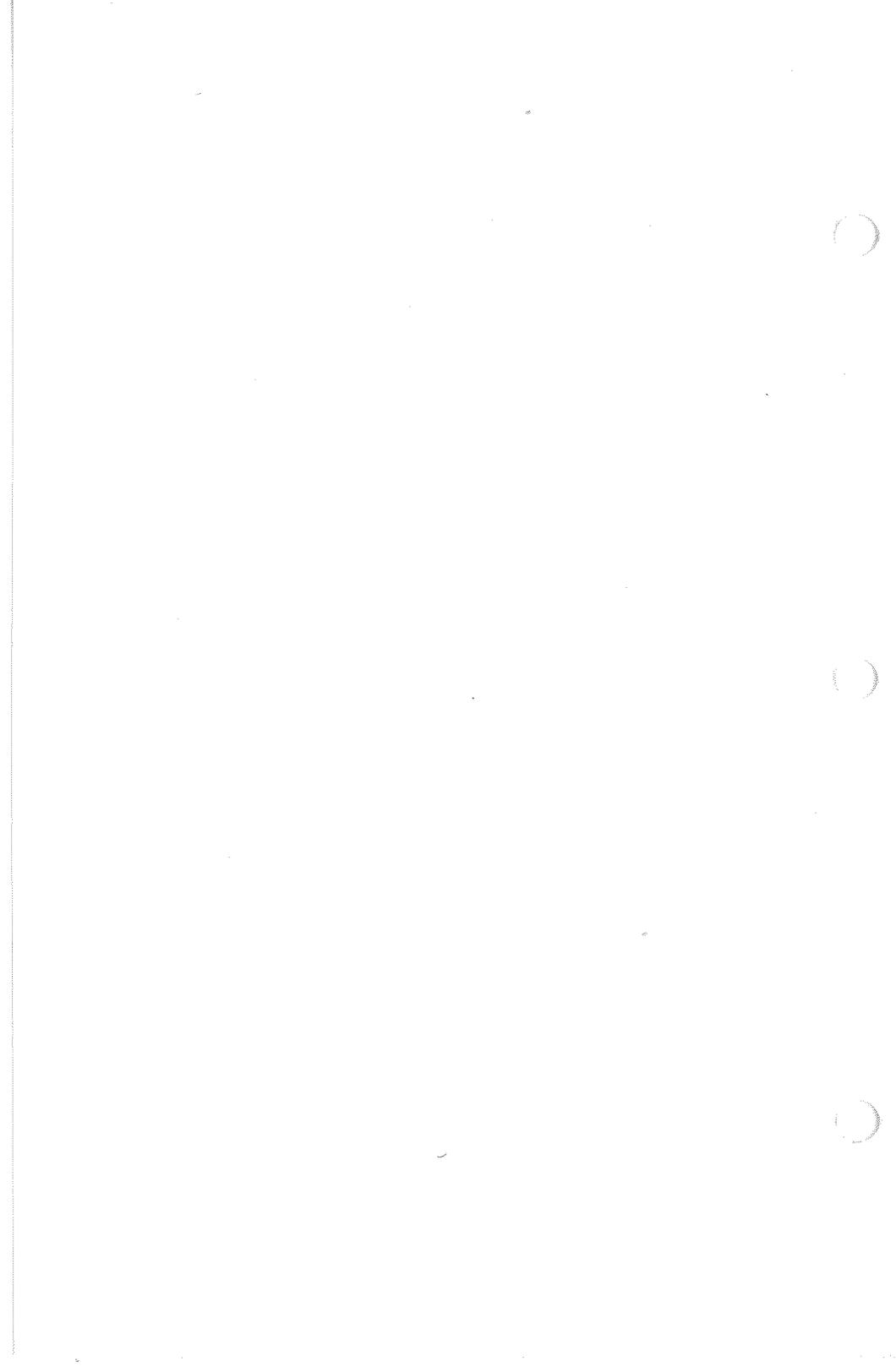
### Converting to 50HZ Operation

The TRS-80 Model 4 Operating System (TRSDOS) is designed for use with a 60-hertz ac power supply. Some countries use 50-hertz ac power. System and applications software purchased from Radio Shack are on 60-hertz disks unless marked "50HZ." If you are in a 50-hertz area, you should convert your disks. (Note: If you attempt to set your system to the power supply it's already set for, the program will abort. This will not result in any problems; it simply means there is no need to go through this procedure.) Only the disks that are used in drive 0 need to be converted.

To convert disks to 50 hertz, a program called HERTZ is run. This program is on all TRSDOS disks and most application software disks. If HERTZ is not on the disk, either place a TRSDOS disk of the same version in drive 1 (multiple-drive system) or copy the program from a TRSDOS disk to the application disk (single-drive system). **Your hardware must have been modified before you apply this patch.**

To perform the procedure, do the following:

- Place the disk to be converted in drive 0.
- At the prompt "TRSDOS Ready", type **DO = HERTZ**. Press ENTER.
- At the prompt for a selection of 5 or 6, type 5 for 50 hertz. If you wish to convert to 60 hertz you type 6.
- Once the patch is applied, the system will automatically reboot itself. The disk has now been permanently converted to 50 hertz.
- If you copied HERTZ to the disk, remove it with the command **REMOVE HERTZ/JCL**.







READ THIS AGREEMENT CAREFULLY. USE OF THIS PRODUCT CONSTITUTES YOUR ACCEPTANCE OF THE TERMS AND CONDITIONS OF THIS AGREEMENT.

**LIMITED WARRANTY**

This Software program is licensed on an "AS IS" basis, without warranty. The original CUSTOMER's exclusive remedy, in the event of a manufacturing defect in the diskette, is its repair or replacement within thirty (30) calendar days of the date of the Radio Shack sales document received upon the license of the Software. The defective diskette shall be returned to a Radio Shack Computer Center, a Radio Shack retail store, participating Radio Shack franchisee or Radio Shack dealer along with the sales document.

EXCEPT AS PROVIDED HEREIN, RADIO SHACK AND SOFTWARE ARTS, INC. MAKE NO EXPRESS WARRANTIES, AND ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE IS LIMITED IN ITS DURATION TO THE DURATION OF THE WRITTEN LIMITED WARRANTIES SET FORTH HEREIN.

Some states do not allow limitations on how long an implied warranty lasts, so the above limitation(s) may not apply to CUSTOMER.

**DISCLAIMER OF LIABILITY**

RADIO SHACK AND SOFTWARE ARTS, INC. SHALL HAVE NO LIABILITY OR RESPONSIBILITY TO CUSTOMER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY "SOFTWARE" LICENSED OR FURNISHED BY RADIO SHACK AND/OR SOFTWARE ARTS, INC., INCLUDING, BUT NOT LIMITED TO, ANY INTERRUPTION OF SERVICE, LOSS OF BUSINESS OR ANTICIPATORY PROFITS OR INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES.

Some states do not allow the limitation or exclusion of consequential damages, so the above limitation(s) or exclusion(s) may not apply to CUSTOMER.

**SOFTWARE LICENSE**

RADIO SHACK and SOFTWARE ARTS, INC. grant to CUSTOMER a non-exclusive, paid-up license to use this Software on *one* computer subject to the following provisions:

A. Except as otherwise provided in this Software License, applicable copyright laws shall apply to the Software.

B. Title to the medium on which the Software is recorded (diskette) is transferred to CUSTOMER, but not title to the Software.

C. CUSTOMER shall not use, make, manufacture or reproduce copies of the Software except for use on *one* computer and except as is specifically provided in this Software License. CUSTOMER is expressly prohibited from disassembling the Software.

D. CUSTOMER is permitted to make six additional copies of the Software *only* for back-up or archival purposes or if such additional copies are required in the operation of *one* computer with the Software, but only to the extent the Software documentation allows back-up copies to be made.

E. All copyright notices shall be retained on all copies of the Software.

F. CUSTOMER is permitted to transfer unmodified Software and all copies of the Software which have been made to a single third party provided CUSTOMER has paid for the Software to be transferred. In any event the provisions of this License and Warranty Agreement shall be applicable to a third party receiving the Software from CUSTOMER.

THIS LICENSE AND WARRANTY AGREEMENT CONSTITUTES THE ENTIRE AGREEMENT AND UNDERSTANDING BETWEEN THE PARTIES AND SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATION RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

The warranties granted herein give the *original* CUSTOMER specific legal rights, and the *original* CUSTOMER may have other rights which vary from state to state.



---

# ***TK!Solver<sup>®</sup> Program Introduction***

---

***Software Arts<sup>™</sup>***

*Custom manufactured in the U.S.A. for Radio Shack,  
a division of Tandy Corporation,  
Fort Worth, Texas 76102*

800-0012-00-D

TK!Solver program copyright © 1984 Software Arts, Inc.  
All rights reserved. Licensed for distribution to Tandy  
Corporation.

TK!Solver documentation copyright © 1984 Software  
Arts, Inc. All rights reserved. Licensed for  
distribution to Tandy Corporation.

TRSDOS, VERSION 6, copyright © 1983 Logical Systems Inc.  
All rights reserved. Licensed to Tandy Corporation.

TK, TK!, TK!Solver, and the stylized exclamation  
point are registered trademarks of  
Software Arts, Inc.

SATN and DIF are trademarks or registered trademarks  
of Software Arts Products Corp.

Software Arts is a trademark of Software Arts, Inc.,  
and Software Arts Products Corp.

Radio Shack, TRS-80, and TRSDOS are registered  
trademarks of Tandy Corporation.

**Software Arts, Inc.**

27 Mica Lane  
Wellesley, MA 02181

Printed in U.S.A.

The TK!Solver program is a product of Software Arts, Inc.,  
which is solely responsible for its contents.

## TK!Solver Introduction

The TK!Solver® program is an entirely new concept in personal computer software. It is a tool for professionals who commonly use equations for analysis, design, and planning and who now depend on calculators to solve their problems. With just a few simple TK!Solver commands you can quickly solve equations, convert units of measurement, plot graphs, and make tables.

Even if you have never used a computer before, in a short time you can be using the TK!Solver program to solve problems. You can use standard textbook formulas or your own. You can use the functions that have been built into the program and you can create your own. Once you have stated your problem, simply enter the known values, then solve the problem with a single keystroke.

This Introduction contains information about the contents of the TK!Solver package. The first section tells you what the different parts of the package are and how to use each of them. The following sections give a general overview of the program, how it works, and how to use it.

## The TK!Solver Package

This package contains your TK!Solver program and many pieces of documentation, including important license information. We have included a wide range of documentation because we want to accommodate all users of the TK!Solver program, whatever their experience with personal computers. You may want to use all of the documentation or only part of it.

We recommend that everyone read the Introductory Guide first. It will quickly acquaint you with the TK!Solver program so that you can begin using it right away.

Your TK!Solver package contains:

In the binder, separated by dividers:

- Introductory Guide
- Introduction (what you are reading now)
- Instruction Manual
- Reference Manual (tabbed)

# TK!Solver Program

---

In the plastic sleeves:

- Reference Card
- The TK!Solver Program Diskette
- TK!Solver Reference Poster
- Software Arts Registration Card

## The Introductory Guide

The Introductory Guide is a sampler of many of the commands and features of the TK!Solver program, and we recommend it be your first step in learning the program. While you are using the program you may find that you understand much of what the program is doing, even if you have never used a computer before. After looking through the Introductory Guide, you will probably be familiar enough with the program to begin using it right away.

```
(6c) Comment: cost of trip                                64 /1

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
    28        mileage          mi/gal     car's milea
   350         miles          mi         miles of tr
         gas      12.5         gal       amount of g
    1.25       price          $/gal     price of ga
         cost     15.625      $         cost of tri

===== RULE SHEET =====
S Rule
- ----
* cost=gas*price
* miles/gas=mileage
```

A simple TK!Solver model that can calculate the cost of gasoline for a trip

### **The Instruction Manual**

The Instruction Manual is a complete tutorial of the TK!Solver program. Beginning with the essential operating system commands that you will need to use your personal computer, the Instruction Manual moves through a set of instructions designed to teach you all the features and commands of the TK!Solver program. Many examples are used throughout the manual; some are included on the TK!Solver program diskette, and some you create as exercises while you work through the manual.

### **The Reference Manual**

The Reference Manual, tabbed so that you can easily find what you are looking for, is a complete and detailed reference for the TK!Solver program. The Reference Manual is especially useful after you have learned the fundamentals of the program and want to understand and explore the implications of the commands, features, and mathematical powers available.

The Reference Manual has cross-references, a general program description, and detailed descriptions of all the program commands and features. The Commands section of the manual includes a chart of the TK!Solver commands. You may want to use it as a reference while you are learning the program. The Mathematics section details the mathematical conventions and solution processes of the program. The Hardware Reference section lists the specifications and requirements for your computer system. Other sections include the Appendices, which contain lists of prompts, status messages, and error messages; the Glossary; and the Index.

### **The Reference Card**

The Reference Card is a brief review of the commands and major features of the program. Once you are familiar with the program, you will probably want to keep the Reference Card beside your computer while you are working.

### **The Program Diskette**

Your TK!Solver package includes one diskette which contains the complete TK!Solver program. The diskette also contains all of the TRS-80® Model 4 files needed to boot the operating system (TRSDOS®).

You may make six backup copies of the TK!Solver program and one backup copy on a hard disk for use on a single computer. Any other copying of the program diskette or individual TK!Solver files is a violation of copyright law and the TK!Solver license agreement.

### **The TK!Solver Reference Poster**

The Reference Poster graphically displays the TK!Solver commands and sheets and lists the built-in functions.

### **The Software Arts Registration Card**

When you fill out and send in the Software Arts Registration Card, you will receive a sample issue of SATN<sup>®</sup>, the journal for users of Software Arts programs. It is intended to help you make more effective use of those programs through tips, tutorials, and examples of practical applications.

### **What the TK!Solver Program Does**

The TK!Solver program helps you solve problems in just a few easy steps:



- Formulate the problem as a TK!Solver model. A TK!Solver model poses a problem using sets of equations to define the relationships among variables.
- Enter the model in the TK!Solver program.
- Enter the known values.
- Type the Action command (!).

The TK!Solver program solves for as many unknown values as it can determine.

Once you have created your model, you can solve a variety of problems, depending on the values known. If the program cannot solve a model, it displays messages to help you diagnose the reason.

Although this version of the program cannot easily do matrix operations or numerical techniques for integration, and is unable to do symbolic calculus, it can find real-number solutions for most sets of algebraic equations.

### How the TK!Solver Program Works

Before you begin using the TK!Solver program, you may find it helpful to know a little about the structure of the program and how it works.

If you were writing down problems to be solved, you would use a sheet of paper. In the program, sheets of paper are replaced by TK!Solver sheets, the screen displays where you record your model. When you have created a model by entering information on the sheets, you type the Action command (!) and use one of the program's solvers, mechanisms used by the program to do calculations, to solve for unknown variable values.

Altogether, there are eight sheets in the program, each containing different information. A model could use all of them or only two. The amount of information you provide depends on your model.

These are the TK!Solver sheets:

- The Rule Sheet contains the equations that define the model.
- The Variable Sheet contains all the variables with their characteristics, such as units and values.
- The Unit Sheet defines conversions between units of measurement.
- The Global Sheet sets limits and requirements for the program, the solvers, and your printer.
- The List Sheet contains all the lists of values in the model.
- The User Function Sheet contains all the functions you have created.
- The Plot Sheet contains the information needed to plot a graph of values.
- The Table Sheet contains the information needed to produce a table of values.

In addition, you can look at more information by displaying subsheets. The three kinds of subsheets show details about variables, lists, and functions created by the user.

# **TK!Solver Program**

---

When you are using the TK!Solver program, you can do several things. You can type and enter information to create a model; you can load a model from a diskette; and you can type TK!Solver commands to manipulate the sheets and the information on them. The TK!Solver program has an Editor, which enables you to change your model easily, and an online Help Facility for information about commands and features.

## **DIF**

The TK!Solver program supports the DIF™ format, a standard storage format developed by Software Arts that enables different programs to exchange data.

## **Conclusion**

Now that you have some idea of what the TK!Solver program is about, it's time to start using it. If you haven't read the Introductory Guide, we recommend you do so before going any further. In just a short time, you will find that the TK!Solver program is the most useful problem-solving tool you've ever used.





---

# ***TK!Solver<sup>®</sup> Program Instruction Manual***

---

***Software Arts<sup>™</sup>***

*Custom manufactured in the U.S.A. for Radio Shack,  
a division of Tandy Corporation,  
Fort Worth, Texas 76102*

800-0012-00-E

TK!Solver program copyright © 1984 Software Arts, Inc.  
All rights reserved. Licensed for distribution to Tandy  
Corporation.

TK!Solver documentation copyright © 1984 Software  
Arts, Inc. All rights reserved. Licensed for  
distribution to Tandy Corporation.

TRSDOS, VERSION 6, copyright © 1983 Logical Systems Inc.  
All rights reserved. Licensed to Tandy Corporation.

TK, TK!, TK!Solver, and the stylized exclamation  
point are registered trademarks of  
Software Arts, Inc.

SATN and DIF are trademarks or registered trademarks  
of Software Arts Products Corp.

Software Arts is a trademark of Software Arts, Inc.,  
and Software Arts Products Corp.

Radio Shack, TRS-80, and TRSDOS are registered  
trademarks of Tandy Corporation.

**Software Arts, Inc.**

27 Mica Lane  
Wellesley, MA 02181

Printed in U.S.A.

The TK!Solver program is a product of Software Arts, Inc.,  
which is solely responsible for its contents.

## Table of Contents

### Part I Introduction

Chapter 1 Using the Instruction Manual .....	1-1
Chapter 2 Using Your TRS-80 Model 4 .....	2-1
Chapter 3 Loading the TK!Solver Program .....	3-1

### Part II Building and Using a Model

Chapter 4 Starting Up .....	4-1
Chapter 5 Moving Around .....	5-1
Chapter 6 Building a Model .....	6-1
Chapter 7 Solving a Model .....	7-1
Chapter 8 Units .....	8-1
Chapter 9 Global Settings and Printing .....	9-1

### Part III The Solvers

Chapter 10 The Direct Solver .....	10-1
Chapter 11 Iterative Solutions .....	11-1
Chapter 12 Using TK!Solver Iteration .....	12-1

### Part IV Lists

Chapter 13 Creating Lists .....	13-1
Chapter 14 The List Command .....	14-1
Chapter 15 Plots and Tables .....	15-1
Chapter 16 User Functions .....	16-1
Chapter 17 Summation .....	17-1



## Part I Introduction

This manual guides you through a step-by-step process designed to teach you how to use the TK!Solver® program. As you work through the chapters, you will learn how to use the TK!Solver commands to build, refine, and solve many TK!Solver models. When you finish this manual, you will have used all the features of the TK!Solver program and applied them to sample models.

A TK!Solver model formulates a problem using sets of equations to define the relationships among variables. TK!Solver models give you the facility not only to solve problems, but also to plot graphs, build tables, and define your own functions.

If you have read through the Introductory Guide, you have some familiarity with the program and what it does. If you have not gone through the Introductory Guide, you may want to do so before beginning to explore in depth the many features of the program.

This manual is divided into four parts.

- **Part I, Introduction**, is a guide to using the Instruction Manual and your operating system and shows you how to load the program.
- In **Part II, Building and Using a Model**, you will begin to use the program and build a model. You will use many models as examples while you are learning to apply the program, but the model that you build in Part II is utilized throughout the manual. As you learn to use the TK!Solver program, this model will be modified and expanded to illustrate applications of all the features of the TK!Solver program.
- **Part III, The Solvers**, discusses the program's solution mechanisms and how to use them.
- **Part IV, Lists**, shows you how to create and use lists of values for solutions, graphs, and tables, and how to define your own functions.

# **TK!Solver Instruction Manual**

---

---

## Contents of Part I:

- Chapter 1 Using the Instruction Manual
- Chapter 2 Using Your TRS-80® Model 4
- Chapter 3 Loading the TK!Solver Program

**Chapter 1 Using the Instruction Manual**

- The Instruction Manual
- Glossary
- Instruction Manual conventions

This Instruction Manual will show you how to use the TK!Solver program. As you work through the chapters, you will use all the commands and features of the program, applying them to examples as you go along. As you become familiar with the TK!Solver program, you will find more and more ways to exercise the freedom that it gives you to explore mathematical problems.

**The Instruction Manual**

The Instruction Manual is structured to take you through the program, from the simpler features to the more complex ones. You will actually use each feature, creating and solving workbook examples with the program. While you are working, you will use an Instruction diskette that you will create in Chapter 3. This diskette will contain information that you need for the examples and will also give you a place to store the sample models you will create.

While you are learning the program, you may want to use the Reference Card as a check list of the commands and features you have learned. You may also want to look at the Command Chart that is in the Commands section of the Reference Manual. It shows the structure of all the TK!Solver commands and may be an aid while you are learning the commands.

Listed below are some common terms used in this manual. A comprehensive glossary appears in the Reference Manual.

**Glossary**

**Boot** To boot means to load the program called the disk operating system, which you use to run other programs on your computer. (The word “boot” originates from the phrase “pull up by the bootstraps.”)

<b>Byte</b>	The memory capacity of a computer is measured in units called “bytes.” Your personal computer is equipped with a given number of bytes. A kilo-byte (1K byte) equals 1024 bytes.
<b>Command</b>	A command is a keystroke, character, or set of characters that tells the program to perform a specific task.
<b>Connection</b>	A connection, or <b>device specification</b> , is a two-letter abbreviation that you use to identify a piece of hardware to the operating system.
<b>Cue</b>	The cue points to the current character position in a line of text and is one character wide. It may appear inside the cursor.
<b>Cursor</b>	The cursor highlights the current position on the screen. It may vary in size according to your position on the screen. The cursor shows the screen position; the cue shows the position within that line.
<b>Default</b>	A default is the value used or the action performed by a program if you do not specify any other value or action.
<b>Device</b>	A physical device is a piece of hardware, such as a printer. A logical device, or <b>device specification</b> , is a two-letter abbreviation that you use to identify a physical device to the operating system.
<b>Devspec</b>	A devspec is a shorter name for <b>device specification</b> .
<b>DIF File</b>	A file using the data interchange format standard to permit exchange of data between programs.
<b>Diskette</b>	A diskette looks like a small phonograph record in a permanent black plastic sleeve. Files are stored on it magnetically. When a diskette is in the disk drive, the computer can read information from it and write information onto it.
<b>Extension</b>	An extension is a suffix for a filename that helps identify the contents of the file. Extensions are separated from a filename by a slash (/), are one to three characters long, and must begin with a letter.

<b>File</b>	A file is a collection of information. Files can contain groups of numbers, lines of text, or programs. Files are stored on either diskettes or hard disks. The number of files that may be stored on a diskette depends on the size of the files (how much information they contain) and how much space there is on the diskette.
<b>Filename</b>	A filename is the name used to identify a specific file on a diskette or hard disk. Filenames are assigned by either a program or a user.
<b>Format</b>	To format a diskette is to prepare it magnetically to accept information.
<b>Hard Disk</b>	A hard disk is a sealed container that holds an aluminum disk coated with a magnetic surface that is used to store information. Files are stored on it and are read from it in the same way as a diskette. The storage capacity of the hard disk, however, is much greater than that of a diskette. Hard-disk memory storage is represented in megabytes. A megabyte equals approximately one million bytes.
<b>Load</b>	To load a file is to copy the contents of the file from the diskette, where it is stored, to the computer's memory, where it can be used.
<b>Model</b>	Data entered into the TK!Solver program which formulates a problem using sets of equations to define the relationships among variables.
<b>Program</b>	A program is a set of instructions for the computer. These instructions may be in one file or in several files.
<b>Prompt</b>	A prompt is a message that appears on the screen asking for a specific choice or for more information. Before the program can continue, you must respond to the prompt by typing the information requested.
<b>Run</b>	To run a program causes the computer to follow an instruction or set of instructions.

## Instruction Manual Conventions

There are several conventions used throughout this text. They are designed to help you identify what you are supposed to type and when you are supposed to type it.

- Uppercase and Lowercase Characters

All commands and filenames in this text are shown in uppercase characters. This convention helps you to identify the order of the commands and how they should be typed. When you type a command or filename in the TK!Solver program, you can use uppercase or lowercase letters, or a combination of both.

- Symbols

In the text, symbols are used to indicate special keys and commands. Table 2-1 in Chapter 2 shows the symbols used in the Instruction Manual.

- Single-Drive and Multiple-Drive Systems

If instructions are different for single-drive and multiple-drive systems, the instructions begin with two lines across the page (for two drives) or one line across the page (for one drive) and end with a box (□).

If there are any special directions for a hard disk, they are given in a separate section that also ends with a box (□).

- Special Type and Color

In the text, the way the words look tells you what you are supposed to do.

*/SL*

Type as shown.

*For Help, type ?* What you see on the computer screen.

*extension*

Enter according to the conventions of TRSDOS.

*extension*

Displayed according to the conventions of TRSDOS.

- Figures

The screen figures you see in this text may be slightly different from what you see on the screen.

- Notes

Throughout the text there are special **Notes** giving general information about the program that will make it easier to use and understand.

The next chapter is about the operating system and some special things you should know before you begin to use the TK!Solver program.

## **Chapter 2 Using Your TRS-80® MODEL 4**

- What you need
- Cursor
- Cue
- Error signal
- Notes about the keyboard
- Files, drives, and diskettes
- Using your computer
- TRSDOS® programs
- File-naming conventions

This chapter describes how to use the Radio Shack® TRS-80® Model 4 personal computer with the TK!Solver program. It includes a list of the hardware required to run the program, the TK!Solver names and descriptions for the keys used by the program, and information about the Model 4 operating system (TRSDOS) as it is used with the TK!Solver program.

In some instances, the directions for a single-drive system are different from those for a multiple-drive system. Directions for multiple-drive systems are marked with a double line; those for single-drive systems are marked with a single line.

If you have a hard disk, follow the directions that correspond to the number of disk drives you have with your hard disk. However, the directions in this manual do not take full advantage of your hard disk. For detailed instructions on using the TRSDOS commands with a hard disk, see the *TRS-80 Model 4 Hard Disk Start-Up Manual*.

### **What You Need**

To use the TK!Solver program, you need:

- The Radio Shack TRS-80 Model 4 computer with at least 128K and one or more disk drives. Two disk drives are recommended.
- The TK!Solver program diskette. The program diskette contains both TK!Solver program files and TRSDOS utilities and system files.
- Extra 5-1/4-inch diskettes to store information.

**Note:** Before a blank diskette can be used to store information, it must be formatted. Formatting is a process that prepares the diskette to accept information. For instructions, see either the section in this chapter titled “Formatting a Diskette” or the *TRS-80 Model 4 Disk System Owner’s Manual*.

Optional items are:

- A Radio Shack parallel printer.
- One or more hard disks.

### **Cursor**

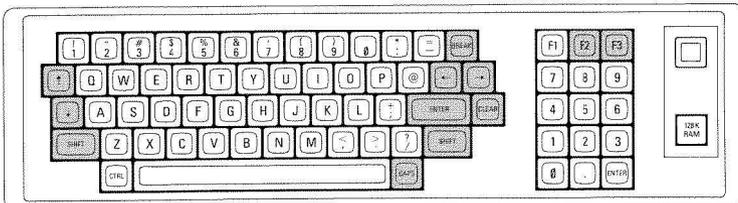
The cursor is a rectangular block of light. Its size changes according to its position on the screen.

### **Cue**

The cue is an underscore or block of light. It is one character wide and marks the current position in a line.

### **Error Signal**

The error signal is a beep.

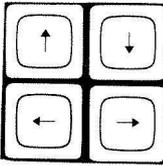


**Figure 2-1**

### **Notes About the Keyboard**

Figure 2-1 is a diagram of the Model 4 keyboard. The highlighted keys are the special keys used for the TK!Solver program. The following descriptions specify how these keys work in the TK!Solver program. They may function differently in TRSDOS and in other programs.

## Arrows



The **Arrow** keys move the cursor up, down, left, and right. They are represented in this text by the symbols  $\uparrow$ ,  $\downarrow$ ,  $\leftarrow$ , and  $\rightarrow$ . If information has just been typed, an Arrow key also enters the information in the current cursor position before moving the cursor. If you are loading a file using the Storage command,  $\rightarrow$  begins Directory Scrolling. Directory Scrolling is explained in Chapter 4.

The Arrow keys work differently when you are using the TK!Solver Editor. The Editor is explained in Chapter 7.

## Backspace



In the TK!Solver program, the key labeled **F2** is the **Backspace** key. It is located in the top row of keys in the numeric keypad. The Backspace key deletes the character to the left of the cue. It is represented in this text by the symbol  $\leftarrow$ .

## Break



The **Break** key cancels whatever you type. It does not, however, cancel a command that has already been executed. This key is represented in the text by the symbol  $\otimes$ . When you don't know what else to do,  $\otimes$  is a panic button that may help.

## Clear



The Clear key is located on the right side of the keyboard and is marked **CLEAR**. Hold down the Clear key to type a command or display special symbols on the screen.

In the TK!Solver program, use the Clear key to type the following special characters:

To type:	Hold down:	and Press:
$\wedge$ (exponentiation symbol)	Clear	;
$\backslash$ (backslash)	Clear	/
$\_$ (underscore)	Clear	Enter

● Edit



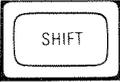
The **Edit** key for the TK!Solver program is the key labeled **F3**. It is located in the top row of keys in the numeric keypad. The Edit key calls the TK!Solver Editor. This key is represented in the text by the symbol ☒. The TK!Solver Editor is discussed in detail in Chapter 7.

Enter



The **Enter** key enters information or completes a command. It is represented in the text by the symbol ◆.

Shift



There are two **Shift** keys, one on either side of the last row of alphabetic keys. As on a typewriter, if a Shift key is held down while a character is typed, the character is typed in uppercase. A Shift key is also used to type the special characters that appear on the top half of the number keys.



To lock in uppercase letters, press the **CAPS** key on the right side of the space bar. To return the keyboard to both uppercase and lowercase letters, press the CAPS key again.

Table 2-1 summarizes the symbols used in this text.

**Text**

**Symbol**

**Key**

⊗

Break

◆

Enter

←

Left arrow

→

Right arrow

↑

Up arrow

↓

Down arrow

⬅

Backspace (F2 key)

☒

Edit (F3 key)

^

Exponentiation (Hold down CLEAR, press ;)

\

Backslash (Hold down CLEAR, press /)

—

Underscore (Hold down CLEAR, press ENTER)

● **Table 2-1**

## **Files, Drives, and Diskettes**

If you have been using your computer for a while and are familiar with it, you may want to skip to the section in this chapter called "File-Naming Conventions" and then go on to Chapter 3.

Before proceeding, please note that this manual assumes you will load the TK!Solver program from drive 0. If you load and run the program from a hard disk, drive 0 is assumed to be the hard disk.

### **Files**

Information on a diskette or hard disk is stored in groups called files. Files are like the files in a filing cabinet, identified by the names written on the folders. Your computer stores information magnetically on a diskette or hard disk and identifies the individual files by their filenames.

Just like a file folder, files on a diskette or hard disk can contain anything. They can contain programs or parts of programs, TK!Solver models, the text of a report to be printed on a printer, or lists of numbers generated by a program as output.

### **Disk Drives**

If you have one disk drive on the system unit, it is called drive 0. If you have two drives, the bottom drive is drive 0 and the top drive is drive 1. TRSDOS begins to search for a file in drive 0. If the file is not stored on a diskette in drive 0, TRSDOS will search each additional drive. It is strongly recommended that you load the TK!Solver program from drive 0 and leave the program diskette in that drive. For more information about loading the program, see Chapter 3.

### **The Hard Disk**

The hard disk is an optional storage device with a memory capacity represented in millions of bytes (megabytes). When you install the hard disk, you assign a drive number to it. In this manual, if you load and run the TK!Solver program from the hard disk, the hard disk is assumed to be drive 0.

### **Diskettes**

Whenever a diskette is not in a drive, it should be kept in its protective sleeve to shield it from the dust in the air and the natural

oil on your fingers. Always hold the diskette along one of the closed edges and be careful not to touch it directly on the oval cutout. Dust, moisture, and oil can damage the diskette and destroy the information on it, as can exposing the diskette to high temperatures.

Anything with an electric motor produces an electromagnetic field. Exposure to any magnetic field can erase information on a diskette. Be careful not to store diskettes where they will be exposed to these fields, such as on top of your computer, against a power cord, or next to a telephone or electrical outlet.

#### Inserting a Diskette

To insert a diskette into a disk drive, open the drive latch by gently pulling up on the bottom of the latch. Insert the diskette into the drive so that the oval cutout on the diskette enters the drive first and the label enters last and is on top. See Figure 2-2.



Figure 2-2



Figure 2-3

If you are using a diskette without a label, slide it in so that the oval cutout enters first and the square notch is on the left. Close the latch as shown in Figure 2-3.

## Using Your Computer

### The Disk Operating System (TRSDOS)

The Model 4 Disk Operating System, or TRSDOS, is the control program that gives the computer its ability to load and run other programs. A part of TRSDOS is stored on the TK!Solver program diskette.

TRSDOS must be booted before any other program can be loaded or run. After TRSDOS has been booted, the system is ready to accept TRSDOS commands. However, if you have one or more disk drives and no hard disk, you must have the following TRSDOS System files on a diskette in drive 0 at all times:

- SYS0/SYS through SYS4/SYS
- SYS6/SYS
- SYS7/SYS
- SYS10/SYS through SYS13/SYS

These System files and the TRSDOS utilities to format and back up diskettes are included on the TK!Solver program diskette.

If you have a hard disk, the TRSDOS System files listed above can be stored on the hard disk.

### Booting TRSDOS

Turn on the computer. The ON/OFF switch is located on the right, under the computer keyboard case. Insert the TK!Solver program diskette into drive 0. To boot the operating system from this diskette, press the **RESET** key, the recessed button on the right side of the keyboard.

**Note:** In this manual, TRSDOS must be booted from drive 0. If you have stored TRSDOS on a hard disk, see the *TRS-80 Model 4 Hard Disk Start-Up Manual* for information about how to load it.

As TRSDOS loads, you will see some messages on your screen. Answer the messages as directed. Consult the *TRS-80 Model 4 Disk System Owner's Manual* for details.

You should see the TRSDOS system prompt:

TRSDOS Ready

The system prompt tells you that TRSDOS is ready to accept commands.

## **TRSDOS Programs**

### **TRSDOS Commands**

The following four sections show you how to use four TRSDOS programs that are included on the TK!Solver program diskette. These programs, `FORMAT`, `BACKUP`, `DIR`, and `COPY`, are ones you will need when you begin to use the TK!Solver program to create and save your own models. If you want to try the instructions given below, you will need two blank diskettes for your examples.

**Note:** The directions below assume that you will use the TRSDOS programs included on the TK!Solver diskette. For information on using these programs from a hard disk, refer to the *TRS-80 Model 4 Hard Disk Owner's Manual*.

**Note:** When following the directions below, make sure that you type the commands exactly as shown. TRSDOS will not accept additional spaces in some command lines.

### **Formatting a Diskette**

Computers find information on a diskette by looking for electronic markings that identify the information. Before a diskette can be used to save information, the diskette must be given these markings by the system. This process is called formatting the diskette.

Formatting a diskette erases any information that is on it. You can recycle old diskettes by reformatting them and preparing them to accept new information. Do **not** reformat your TK!Solver program diskette. If you do, you will erase the programs on it.

You cannot format a diskette while running the TK!Solver program. If you want to save the models you are creating, you must decide how many diskettes you want to have available for storing information and format them **before** you load the TK!Solver program.

---

### Multiple-Drive Systems

If you have not booted TRSDOS from the TK!Solver program diskette, do so. Insert a blank diskette into drive 1 and type:

```
FORMAT :1 ♦
```

The system responds with a series of prompts that are displayed one at a time. Respond to each prompt by pressing ♦. The prompts are:

```
Diskette name ?  
Master password ?  
Single or Double density <S,D> ?  
Number of cylinders ?
```

By pressing ♦ after each of these prompts, you tell the system to use a default. For a list of these defaults, see the *TRS-80 Model 4 Disk System Owner's Manual*. The system now responds:

```
Formatting cylinder 01 (02, 03, ...39)
```

The next message is:

```
Verifying cylinder 01 (02, 03, ...39)
```

When the system has checked each cylinder on the diskette, it displays the messages:

```
Directory will be placed on cylinder 20  
Initializing SYSTEM information .....  
Formatting complete  
TRSDOS Ready
```

When you see the system prompt, remove the formatted, blank diskette from drive 1. Repeat this entire procedure to format a second blank diskette. When you have formatted both diskettes, label one "TK!Solver Backup" and the other "Instruction Diskette." You will use these two formatted diskettes later in this chapter. □

---

**Single-Drive System**

If you have not booted TRSDOS from the TK!Solver diskette, do so. After the system prompt, type:

```
FORMAT :0 ◆
```

The system responds with a series of prompts which are displayed one at a time. Respond to each prompt by pressing ◆. The prompts are:

```
Diskette name ?  
Master password ?  
Single or Double density <S,D> ?  
Number of cylinders ?
```

By pressing ◆ after each prompt, you tell the system to use a default. For a list of these defaults, see the *TRS-80 Model 4 Disk System Owner's Manual*. You should now see the message:

```
Load destination diskette <ENTER>
```

When the disk drive light has gone out, remove the TK!Solver diskette from drive 0. Insert a blank diskette into the drive, making sure the oval cutout enters the drive first. Close the latch. Press ◆.

The system responds:

```
Formatting cylinder 01 (02, 03, ...39)
```

The next message is:

```
Verifying cylinder 01 (02,03, ...39)
```

When the system has checked each cylinder on the diskette, it displays the messages:

```
Directory will be placed on cylinder 20
Initializing SYSTEM information .....
Load SYSTEM diskette <ENTER>
```

Remove the blank, formatted diskette and replace it with the TK!Solver diskette. Press **◆**. The message is:

```
Formatting complete
```

Repeat this entire procedure to format a second blank diskette. When you have formatted both diskettes, label one “TK!Solver Backup” and the other “Instruction Diskette.” You will use these two formatted diskettes later in this chapter. □

### **Backing up a Diskette**

It is a good idea to make copies of the diskettes on which you have stored your models. If an accident destroys the information on a diskette, the models that were on that diskette will be available on the backup diskette.

For this example, you will make a single backup copy of the TK!Solver program by copying the program diskette onto the blank diskette you formatted earlier. The TK!Solver diskette is the SOURCE diskette and the formatted diskette labeled “TK!Solver Backup” is the DESTINATION diskette.

**Note:** If you want to run the TK!Solver program from a hard disk, you must copy the entire TK!Solver program diskette onto the hard disk. This procedure is discussed in Chapter 3.

---

### **Multiple-Drive Systems**

If you have not booted TRSDOS, do so. Insert the diskette labeled “TK!Solver Backup” into drive 1. Type:

```
BACKUP :0 :1 (X) ◆
```

You should see the message:

```
Insert SOURCE disk <ENTER>
```

Make sure the TK!Solver diskette is in drive 0, then press **◆**. You should see the message:

```
Destination disk ID is different:
  Name=DATADISK Date=date
Are you sure you want to backup to it
  <Y,N> ?
```

This message is telling you that the destination diskette name DATADISK is different from the name given to the TK!Solver diskette. Even though the names are different, you can still back up the TK!Solver diskette onto the DATADISK. Answer this prompt by typing:

```
Y◆
```

The system responds:

```
Loading cylinder 01 (02, 03, ...)
```

This is replaced by the messages:

```
Dumping cylinder 01 (02, 03, ...)
Verifying cylinder 01 (02, 03, ...)
```

When the backup procedure is complete, the system displays the message:

```
Source disk is write protected; MOD flags
  not updated
Insert SYSTEM disk <ENTER>
```

Press **◆**. TRSDOS responds:

```
Backup complete
```

The diskette in drive 1 should now contain a backup copy of the TK!Solver program diskette. □

## Single-Drive System

If you have not booted TRSDOS, do so. Type:

```
BACKUP :0 :0 (X) ♦
```

The system reponds:

```
Insert SOURCE diskette <ENTER>
```

Make sure the TK!Solver program diskette is in the drive, then press ♦. The next message is:

```
Insert DESTINATION disk <ENTER>
```

Remove the TK!Solver diskette from drive 0 and replace it with the formatted diskette labeled "TK!Solver Backup." Press ♦. You should now see the message:

```
Destination disk ID is different:
```

```
  Name=DATADISK Date=date
```

```
Are you sure you want to backup to it  
<Y,N> ?
```

This message is telling you that the destination diskette name DATADISK is different from the name given to the TK!Solver diskette. Even though the names are different, you can still back up the TK!Solver diskette on the DATADISK. Answer this prompt by typing:

```
Y ♦
```

You should now see the message:

```
Insert SOURCE disk <ENTER>
```

Remove the destination diskette and replace it with the TK!Solver diskette. Press ♦. The system reads the diskette and then displays the message:

```
Loading cylinder 01 (02, 03, ...10)
```

```
Insert DESTINATION disk <ENTER>
```

Remove the TK!Solver diskette and replace it with the destination disk. The message is:

Dumping cylinder 01 (02, 03, ...10)

This is replaced by the messages:

```
Verifying cylinder 01 (02, 03, 04, 05...10)
Insert SOURCE diskette <ENTER>
```

Each time you are prompted for the SOURCE diskette, insert the TK!Solver program diskette into drive 0 and press **◆**. When you are prompted for the DESTINATION disk, insert the backup diskette into drive 0 and press **◆**.

When you have finished backing up the TK!Solver diskette, you should see the message:

```
Source disk is write protected; MOD flags
      not updated
Insert SYSTEM disk <ENTER>
```

Press **◆**. TRSDOS responds:

```
Backup complete
```

Your destination diskette now contains a backup copy of the TK!Solver program diskette. **□**

### **Creating Storage Diskettes**

If you have only one disk drive and no hard disk, you must create storage diskettes by first making a backup of the TK!Solver diskette and then removing files which are unnecessary for operating the TK!Solver program.

To create storage diskettes for use on a single-drive system, continue this procedure which allows you to remove files unnecessary for operating the TK!Solver program.

Be sure a backup of the TK!Solver diskette is in the drive, then type the two commands below, pausing a moment between them while the files are removed:

```
PURGE BACKUP/CMD:0 (Q=N,I)◆
PURGE FORMAT/CMD:0 (Q=N,I)◆
```

This deletes the two TRSDOS programs `BACKUP` and `FORMAT`, allowing more space for model storage.

Instruction Files, available on the TK!Solver program diskette for use with the Instruction portion of this manual, may be deleted from your storage diskette by typing:

```
PURGE /TK:0 (Q=N) ◆
```

If you load the TK!Solver program from a complete backup of the TK!Solver program diskette, the `TK/CMD` file will not be required for normal operation from a storage diskette. To release more diskette space for model storage, you may remove the `TK/CMD` file by typing:

```
PURGE TK/CMD:0 (Q=N) ◆
```

As an option even more storage space may be made available by deleting the TK!Solver Help File. The TK!Solver Help File provides immediate descriptions and explanations to assist you in employing the TK!Solver program as an effective computational tool. For this reason, removal of the Help File is not recommended to the novice user. To delete the Help File, type:

```
PURGE TK/HLP:0 (Q=N) ◆
```

Removal of these files from a backup copy of the TK!Solver diskette should provide sufficient model storage space to single-disk-drive users without jeopardizing the normal operation of the TK!Solver program. It is important to remember that deletion of these files eliminates the capabilities which these files provide. Removal of the `TK/CMD` file will prevent you from loading the TK!Solver program from your storage diskette. If you remove `TK/CMD` from your storage diskette, be sure to load the TK!Solver program from a complete program diskette, then replace this diskette with the diskette upon which your model is stored.

### **Listing the Contents of a Diskette**

To find out which files are on a diskette, use the TRSDOS program `DIR` to list the contents of the diskette. For example, replace any diskette you may have in drive 0 with the backup copy of the TK!Solver program diskette you just created. Type:

DIR ♦

The contents of the “TK!Solver Backup” diskette should be displayed on your screen.

**Note:** If you have diskettes inserted in any other active drives, the DIR command will also display the contents of those diskettes. Otherwise, the system displays the message [NO DISK] next to the drive number.

To list the TRSDOS System files that are on a diskette, add the SYS parameter to the command. For example, with the TK!Solver backup diskette still in the drive, type:

DIR (SYS) ♦

You should see the TRSDOS System files that are contained on the diskette as well as the other TK!Solver files.

### **Specifying a Drive**

If you have multiple drives on your computer, you may want to list the contents of a diskette that is not in drive 0.

To designate a specific drive when using a TRSDOS command, type the command, a colon (:), and then the number identifying the drive. When you use this syntax, the program goes to the specified drive and performs the command.

The syntax is:

*command :drive number* ♦

For example, insert the TRSDOS diskette into drive 0 and the backup copy of the TK!Solver program diskette into drive 1. Type:

DIR :1 ♦

You should see only the contents of the “TK!Solver Backup” diskette on your screen.

## **Copying a File**

You may need to copy an individual file from one diskette to another to use as a backup copy or somewhere else. You can copy a file from one diskette to another formatted diskette. You can also copy a file from a hard disk to a formatted diskette or from a diskette to a hard disk.

If you plan to copy the TK!Solver program onto a hard disk, you do not need to copy the TK/OVL and TK/HLP files onto storage diskettes. The TK!Solver program uses these files directly from the hard disk.

---

---

## **Multiple-Drive Systems**

If you have not already booted TRSDOS, do so. Insert the diskette labeled "Instruction Diskette" into drive 1.

Make sure the source diskette is in drive 0, then type the following command:

```
COPY <filename>:0 :1 ◆
```

where <filename> is the name of the file you want to copy.

The system responds:

```
Copying: <filename>:0 to <filename>:1
```

When the copy procedure is finished, you should see the system prompt on the screen. □

---

## **Single-Drive System**

If you have not already booted TRSDOS, do so. Make sure the source diskette is in drive 0, then type the following command:

```
COPY <filename>:0 :0 (X) ◆
```

The system responds:

```
Insert SOURCE disk <ENTER>
```

You will be asked several times to insert the SOURCE disk, the SYSTEM disk, and the DESTINATION disk. When you are prompted for either the SOURCE or SYSTEM disk, insert the source diskette into drive 0 and press **◆**. When you are prompted for DESTINATION diskette, insert the formatted diskette you labeled “Instruction Diskette” and press **◆**.

You should finish the copy procedure with the TK!Solver diskette in the drive. □

### **File-Naming Conventions**

When you use the TK!Solver Storage command to save a file, you will be asked to give the file a name. TRSDOS imposes certain limitations on how files are named. These limitations are:

- Filenames must be one to eight characters long.
- Filenames must begin with a letter.
- Filenames may contain any uppercase lowercase letter and any number from 0 through 9.
- Filenames cannot contain any other characters.

Filenames also have extensions that identify the type of file you are naming. Extensions are one to three characters preceded by a slash (/). The first character of an extension must be a letter.

If you save a file using the TK!Solver program, the program automatically attaches the correct extension. If you are naming files to save and load using the TK!Solver Storage command, you do **not** have to include the extension as part of the filename, although you may. The TK!Solver program looks for the appropriate extension.

The TK!Solver extensions are:

- /TK** for TK!Solver models and parts of models
- /DIF** for files in DIF format
- /PRF** for print files, files which contain images for the printer

**Note:** If you create a print file using the TK!Solver program and want to see what is in the file, go to TRSDOS and use the LIST command. See the *TRS-80 Model 4 Disk System Owner's Manual* for information about using this command.

### **Conclusion**

This chapter has described some features of your computer and TRSDOS. The next chapter discusses how to load and run the TK!Solver program, how to use the diskettes while reading the Instruction Manual, and how to quit the program.

**Chapter 3 Loading the TK!Solver Program**

- The TK!Solver diskette
- The TK!Solver files
- Creating the Instruction diskette
- Loading the TK!Solver program
- The TK!Solver program and the hard disk
- The Quit command (/Q)

The TK!Solver program diskette contains several files, some of which are program files and some of which are for use with this manual. This chapter tells you about the TK!Solver program diskette and the files on it. It also gives you instructions for creating the Instruction diskette you will use with this manual and for putting the TK!Solver program on a hard disk. You will also learn how to load and quit the TK!Solver program from both a diskette and a hard disk.

If your computer has two or more drives, follow the directions under the heading “Multiple-Drive Systems.” If your computer has one disk drive, follow the directions under the heading “Single-Drive Systems.” If the number of drives is not specified, the text applies to both single- and multiple-drive systems. If your computer has a hard disk, follow the directions under the heading “The TK!Solver Program and the Hard Disk.”

**The TK!Solver Diskette**

You may make six backup copies of the TK!Solver program diskette and one backup copy on a hard disk for use on a single computer. Any other copying of the TK!Solver program diskette or the TK!Solver program files is unauthorized and is a violation of copyright law and the TK!Solver license agreement.

**The TK!Solver Files**

The Overlay File, TK/OVL, contains a part of the TK!Solver program, and the Help File, TK/HLP, contains the text for the Help Facility. Because these two files are not read into memory when the program is loaded, they must be available to the program when it is being used.

If you have these files on a storage diskette, you will be able to use the storage diskette in place of the program diskette after you have loaded the program.

**Note:** If you plan to load and run the TK!Solver program from a hard disk, you do not have to copy TK/OVL and TK/HLP onto any storage diskettes. These files are available to the program from the hard disk. However, this manual assumes that if you copy the TK!Solver program to a hard disk, TRSDOS is also on the hard disk.

---

### Single-Drive Systems

If you have one drive on your system and no hard disk, you **must** have the file TK/OVL on every storage diskette. If you want to use the TK!Solver Help Facility, you also need to have TK/HLP. Each storage diskette must also contain the following TRSDOS System files:

- SYS0/SYS through SYS4/SYS
- SYS6/SYS
- SYS7/SYS
- SYS10/SYS through SYS13/SYS

The easiest way to create extra storage diskettes is to back up the Instruction diskette that you will make in the next section. The Instruction diskette will contain TK/OVL, TK/HLP, the Instruction Files, and the TRSDOS System files listed above. You can then remove the Instruction Files from the storage diskettes, using the TRSDOS PURGE command. To create storage diskettes on a single-drive system, see the section titled “Creating Storage Diskettes.” □

**Note:** For the rest of this manual, you need only the storage diskette labeled “Instruction Diskette” that you formatted and used in Chapter 2.

**Multiple-Drive Systems**

If you keep your backup copy of the TK!Solver program diskette in drive 0, you can use storage diskettes without the files TK/OVL and TK/HLP in the other drives; you do not have to copy these files onto any storage diskettes if you prefer not to. □

**Note:** For the rest of this manual, you need only the storage diskette labeled “Instruction Diskette” that you formatted and used in Chapter 2.

**Using the TK!Solver Files**

Load the TK!Solver program from a backup copy of the program diskette according to the instructions given below. You will be able to continue using the program as long as TK/OVL and the TRSDOS System files are available on a diskette in drive 0 or are on a hard disk.

If you quit the program and return to TRSDOS, you must load the TK!Solver program again. If you turn off the computer, you must boot TRSDOS and then load the program.

**Creating the Instruction Diskette**

This section shows you how to create the Instruction diskette you will need for this manual. After you load the program, you will replace the program diskette with the Instruction diskette and continue with the manual.

**Note:** If you plan to copy the TK!Solver program onto a hard disk, you do not need to create the Instruction diskette. Copies of the Instruction Files will be stored on the hard disk.

The Instruction diskette is the storage diskette you will use while working through this manual. It will contain the files TK/OVL and TK/HLP and the eight Instruction Files that you will use while you are learning the program. It will also contain some TRSDOS System files.

To make the Instruction diskette, you will back up the entire TK!Solver program diskette and then remove the TRSDOS programs BACKUP/CMD and FORMAT/CMD and the TK!Solver program file TK/CMD from it. When you have finished creating your Instruction diskette, we recommend that you make a backup copy of the Instruction diskette before beginning Chapter 4.

---

### Multiple-Drive Systems

If you have not booted TRSDOS from the TK!Solver program diskette, do so. In drive 1, insert the diskette you created in Chapter 2, labeled "Instruction Diskette." Type:

```
BACKUP :0 :1 (X) ♦
```

The system responds:

```
Insert SOURCE diskette <ENTER>
```

Press ♦. The entire TK!Solver diskette should now be copied to the Instruction diskette. When the backup procedure is finished, you are ready to remove the TRSDOS programs BACKUP/CMD and FORMAT/CMD and the TK!Solver program file TK/CMD from the Instruction diskette. By removing these files, you increase the storage space on the Instruction diskette.

Type the three commands below, pausing between each one as the files are removed:

```
PURGE BACKUP/CMD:1 (Q=N, I) ♦
```

```
PURGE FORMAT/CMD:1 (Q=N, I) ♦
```

```
PURGE TK/CMD:1 (Q=N) ♦
```

When the system prompt is displayed after removing TK/CMD, your Instruction diskette is ready to use.

Use the DIR command with the SYS parameter to list the contents of the diskette in drive 1. You should see the eight Instruction Files, the files TK/OVL and TK/HLP, and eleven TRSDOS System files. □

**Single-Drive Systems**

If you have not booted TRSDOS, do so. Type:

```
BACKUP :0 :0 (X) ♦
```

The system responds:

```
Insert SOURCE diskette <ENTER>
```

Make sure the TK!Solver program diskette is in drive 0, then press ♦. The system now copies some of the files into memory and then prompts:

```
Insert DESTINATION diskette <ENTER>
```

Remove the TK!Solver diskette and replace it with the diskette labeled “Instruction Diskette” that you created in Chapter 2. Press ♦.

Keep switching diskettes until all the files are copied. You will have to switch the diskettes many times before the copy is completed.

When the last file is copied, the TK!Solver diskette should be in the drive. You are now ready to remove the TRSDOS commands BACKUP/CMD and FORMAT/CMD and the TK!Solver program file TK/CMD from the Instruction diskette. By removing these files, you increase the amount of storage space on the Instruction diskette.

Remove the TK!Solver diskette from the drive and replace it with the Instruction diskette. Type the following three commands, pausing between each one as the files are removed:

```
PURGE BACKUP/CMD:0 (Q=N,I) ♦  
PURGE FORMAT/CMD:0 (Q=N,I) ♦  
PURGE TK/CMD:0 (Q=N) ♦
```

When the system prompt is displayed after removing TK/CMD, your Instruction diskette is ready to use.

Use the DIR command with the SYS parameter to list the contents of the Instruction diskette. You should see the eight Instruction Files, the files TK/OVL and TK/HLP, and eleven TRSDOS System files. □

### **Creating Storage Diskettes**

If you are using a single-drive system, the easiest way to create storage diskettes is to back up the Instruction diskette you just created and remove the Instruction Files. This will give you more room on the storage diskette for your own model files.

To create a storage diskette, first format a blank diskette. This will be the DESTINATION diskette. Then, insert the Instruction diskette in the drive. This will be the SOURCE diskette. Type:

```
BACKUP :0 :0 (X) ◆
```

Follow the messages as before, inserting the SOURCE disk and DESTINATION disk when prompted. When you have completed the backup, you are ready to remove the Instruction Files from the storage diskette.

Insert the storage diskette into the drive and type:

```
PURGE /TK:0 (Q=N) ◆
```

When TRSDOS has purged all the Instruction Files from your storage disk, it is ready to use. If you want, you can then back up this disk to make additional storage diskettes.

### **Loading the TK!Solver Program**

This section explains how to load the TK!Solver program. Use the backup copy of the program diskette, which you created in Chapter 2, to load the program. This saves wear on your original program diskette. Directions for saving the TK!Solver program on a hard disk and loading it from the hard disk are given later in this chapter.

The instructions in this manual assume that you will load and use the TK!Solver program diskette and Instruction diskette from drive 0. If you use only drive 0, you do not have to designate a particular drive while you are learning to use the program.

If you have not already booted TRSDOS from the TK!Solver diskette, do so. After the TRSDOS system prompt, type:

```
TK ♦
```

The screen should look like Figure 3-1.

```
Version 01.00.00
of the TK!Solver (R) Program

This program is a product of
SOFTWARE ARTS, INC.

Copyright (c) 1984 Software Arts, Inc.
All Rights Reserved.
Licensed for Distribution to Tandy Corporation

Except as specifically authorized,
copying of any part of this program
is prohibited.

***** Press ENTER to Start *****
```

**Figure 3-1**

**Note:** The version number may be different for your program.

This screen shows the TK!Solver copyright notice, the program version number, and the message:

```
Press ENTER to Start
```

Press ♦.

While the program loads, the message is:

```
Loading the TK!Solver program
Copyright (c) 1984 Software Arts, Inc.
```

When the program is loaded, the screen should look like Figure 3-2. The TK!Solver program is now ready to be used.

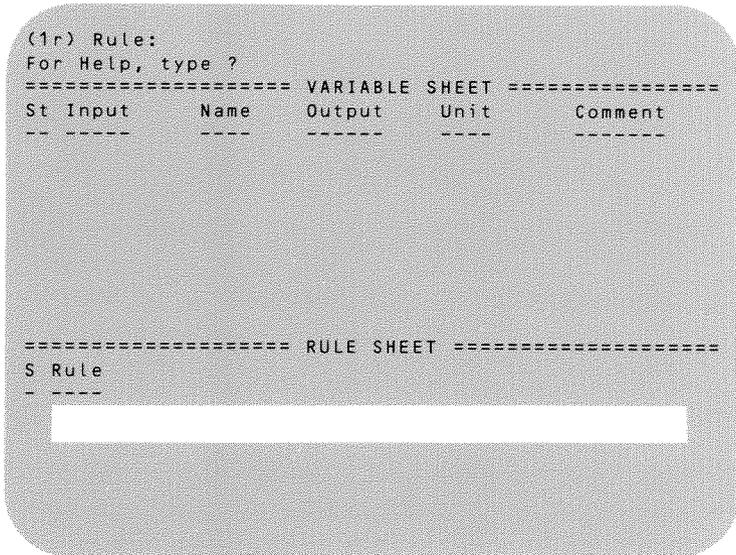


Figure 3-2

**Note:** If you try to load the program and you get the message:

```
A NON-RECOVERABLE PROGRAM ERROR
has occurred. Please write down
the information listed below and
refer to the Reference Manual
for further instructions.
```

you have probably set up a MEMDISK in Bank 0 (zero) of memory before you loaded the program. To load the TK!Solver program successfully, you must disable MEMDISK.

If you need to stop working while you are using the manual, use the Save option of the Storage command, explained in the next chapter, to save your work in a file on the Instruction diskette before you quit the program.

When you are ready to continue with the manual, boot TRSDOS and load the TK!Solver program again. Then use the Load File option of the Storage command, explained in the next chapter, to load the file you were working with and continue with the manual.

### **The TK!Solver Program and the Hard Disk**

If you plan to load and run the TK!Solver program from a hard disk, the instructions in this manual assume that you have installed your hard disk and that the system recognizes the hard disk as drive 0. If you use only the hard disk, you do not need to designate a particular drive while you are learning to use the program. All files you save while using this manual should be saved on the hard disk.

#### **Copying the TK!Solver Program onto the Hard Disk**

To copy the TK!Solver program onto the hard disk, insert the BOOT diskette (a backup of the Hard Disk Installation diskette) into drive 0. Follow the instructions in the *TRS-80 Model 4 Hard Disk Start-Up Manual* to boot TRSDOS from the hard disk.

When you see the **TRSDOS Ready** prompt, remove the TRSDOS BOOT diskette from the bottom disk drive and replace it with the TK!Solver diskette. Type:

```
BACKUP :diskette drive number :hard disk number (NEW) ♦
```

The **diskette drive number** is the number of the bottom disk drive which contains the TK!Solver diskette. The **hard disk number** is the drive number of the hard disk where you want the program to be stored.

**Note:** When you install the hard disk and store TRSDOS on it, the hard disk becomes drive 0 and the bottom disk drive is renumbered. The number of the bottom disk drive is determined by the number of logical drives you assign to the system during hard-disk installation.

All the files on the TK!Solver program diskette should now be copied from the TK!Solver diskette to the hard disk. When the backup procedure is complete, you should see the TRSDOS system prompt. Remove the TK!Solver diskette and keep it as a backup copy. You are now ready to load the TK!Solver program from the hard disk.

### **Loading the TK!Solver Program from Hard Disk**

If you have not already done so, boot TRSDOS from the hard disk. When you see the system prompt, type:

TK ♦

The screen should now look like Figure 3-1. This screen shows the TK!Solver copyright notice, the program version number, and the message:

Press ENTER to Start

Press ♦.

While the program loads, the message is:

```
Loading the TK!Solver program
Copyright (c) 1984 Software Arts, Inc.
```

When the program is loaded, the screen should look like Figure 3-2. The TK!Solver program is now ready to be used.

**Note:** If you try to load the program and you get the message:

```
A NON-RECOVERABLE PROGRAM ERROR
has occurred. Please write down
the information listed below and
refer to the Reference Manual
for further instructions.
```

you have probably set up a MEMDISK in Bank 0 (zero) of memory before you loaded the program. To successfully load the program, you must disable MEMDISK.

If you need to stop working while you are using the manual, use the Save option of the Storage command, explained in the next chapter, to save your work in a file on the hard disk or on a diskette before you quit the program. As long as you have booted TRSDOS from the hard disk, all files you save will automatically be stored on the hard disk.

When you are ready to continue with the manual, boot TRSDOS from the hard disk and load the TK!Solver program again. Use the Load File option of the Storage command, explained in the next chapter, to load the file you were working with, and continue with the manual.

### **The Quit Command (/Q)**

Quitting the TK!Solver program stops the program, clears the computer's memory, and returns you to TRSDOS system level. The Quit command begins with a slash (/).

Type:

/

The Prompt/Error line at the top of the screen shows the list of commands that begin with a slash. The options are:

B	Blank	P	Print
C	Copy	Q	Quit
D	Delete	R	Reset
E	Edit Field	S	Storage
I	Insert	W	Window
L	List	!	Solve
M	Move		

The command **Q**, for Quit, is the first command discussed in this manual. The others will be discussed as they arise. Type:

Q

The prompt is:

Quit: Y N

Because the Quit command stops the program and clears the computer's memory, it requires a confirmation so that you do not quit the program accidentally and lose your work. Typing Y clears the TK!Solver program from the screen. Typing N cancels the Quit command.

Type:

Y

The TK!Solver program is cleared.

**Note:** If you want to quit the program and you do not need to return to TRSDOS, simply remove any diskettes that are in the drives and turn off the computer.

To continue, load the TK!Solver program from TRSDOS by typing:

TK ♦ ♦

When the program has been loaded and your screen looks like Figure 3-2, remove the program diskette from drive 0 and replace it with the Instruction diskette. If you loaded the program from the hard disk, you do not need any diskette in the drive. You are now ready to begin Part II.

### Conclusion

This chapter has told you about the TK!Solver program diskette and files, shown you how to create your Instruction diskette, and given you instructions for loading and quitting the TK!Solver program from both the diskette and the hard disk. In the next chapter you will begin to use the TK!Solver program.

## **Part II Building and Using a Model**

In Part II of the Instruction Manual you will begin to work with models. A model is a representation of a theoretical situation within a given environment. TK!Solver models are mathematical models that use sets of equations to define the relationships between variables.

Part II shows you how to build, use, and refine a TK!Solver model. It starts with the preliminary commands you need to move around in the program, and then guides you while you build a model named TRAVEL. TRAVEL begins as a simple model and is used throughout the rest of the Instruction Manual. As you use it, you will see how a model can be built, used, changed, developed, revised, and expanded to express a variety of problems and how to find solutions to these problems. Part II gives you all the basic commands and features you need to begin using the TK!Solver program.

---

### Contents of Part II:

- Chapter 4 Starting Up
- Chapter 5 Moving Around
- Chapter 6 Building a Model
- Chapter 7 Solving a Model
- Chapter 8 Units
- Chapter 9 Global Setting and Printing

## Chapter 4 Starting Up

- The TK!Solver sheets
- The TK!Solver screen
- The Help Facility
- The Storage command (/S)—Load File (L), Save Model (S), and Delete File (D) options
- Directory Scroll feature

Before you begin to build and solve models using the TK!Solver program, you should know how to save and load files so that you can begin and end your sessions with the program when you wish.

Chapter 3 showed you how to load and quit the program. This chapter describes what you see on your screen, and shows you how to use the Help Facility, and how to load, save, and delete a file.

---

### To begin this chapter:

If you did not quit the program at the end of the last chapter, you are ready to begin Chapter 4.

If you quit the program at the end of Chapter 3, load the TK!Solver program according to the instructions given in Chapter 3.

The screen should look like Figure 4-1.

```
(1r) Rule:
For Help, type ?
===== VARIABLE SHEET =====
St Input      Name      Output    Unit      Comment
-----

```

```
===== RULE SHEET =====
S Rule
-----

```

Figure 4-1

## The TK!Solver Sheets

If you were writing a model without the TK!Solver program, you would probably begin by writing down your equations on a sheet of paper and then painstakingly solve them using a calculator. The TK!Solver program provides you with screen displays called Sheets which you use like sheets of paper to record not only your equations, but the properties or characteristics of each variable including values and units, conversion definitions for units of measure, and a variety of other features.

Each sheet is made up of columns and rows or labeled lines or a combination. The two sheets that are currently on your screen are the Variable Sheet, at the top, and the Rule Sheet, at the bottom. Both are composed of columns and rows.

The information that you put on the sheets goes into spaces called fields. The cursor is the screen marker that moves from field to field to show you where you are on the sheet.

You use the TK!Solver commands to move around and between the sheets, to enter items of information into the fields, creating a model, and to solve your models.

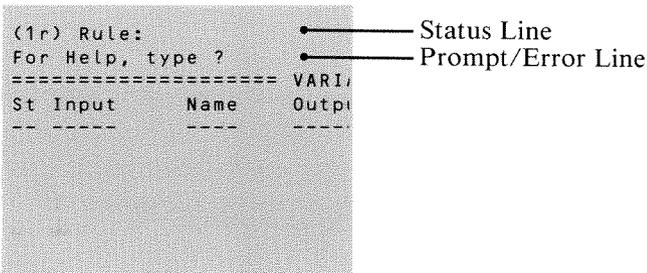
## The TK!Solver Screen

### The Message Area

At the top of the screen are two lines called the Message Area. The first line, the Status line, displays the program status. The second line, the Prompt/Error line, displays messages.

### The Status Line

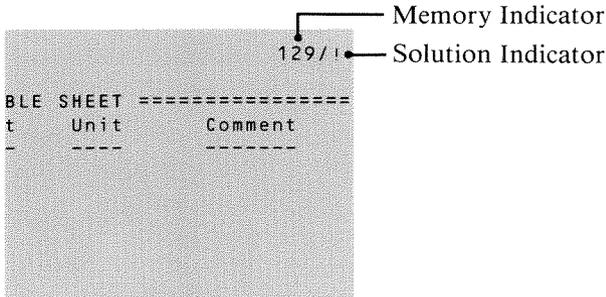
At the far left of the Status line, in parentheses, are a letter and number that tell you the field position of your cursor. It currently says **1r**. This is the Position Indicator. The number is the row number of the field currently containing the cursor (in the example, the first row on the Rule Sheet); the letter represents the column label (Rule).



To the right of the Position Indicator is a space for the label and contents of the current field. Since you have not entered any information yet, the screen shows only the label **Rule** followed by a colon (:). If there were an item in the field, the item would appear on the Status line following the label. If the item is larger than the field, you can usually see the entire item by moving the cursor to the field and looking at the Status line.

**Note:** Values are shown on the Status line rounded to the program's full accuracy of 12 significant digits.

On the far right side of the Status line is a number called the Memory Indicator. The Memory Indicator tells you how many K (1024) bytes of memory you have left. This number depends on the memory capacity of your computer and may not match the number shown in the Figure.



If the amount of memory goes below 1K bytes, the Memory Indicator shows **LOW**. If this happens, you still have memory left and can save your model. As a precaution, you should save your model when you see this message.

When you have run out of memory, the Memory Indicator shows the message **OUT**. There is still usually enough memory left for you to save your model, and you can delete parts of your model to create space in memory, but if you try to add to the model or solve it after the **OUT** message appears, it may not work.

The Memory Indicator is followed by a slash to separate it from the Solution Indicator. There is no Solution Indicator on your screen yet. When you enter your first equation, an exclamation point (!) will appear after the slash to show that your current model is unsolved. The Solution Indicator appears after any modification to the model that will affect the solution. It remains displayed until the model is solved without errors.

### **The Prompt/Error Line**

Below the Status line is the Prompt/Error line. The program uses this line to prompt for a typed response and to display program status messages and error messages. You will see many status messages and prompts as you work through this manual.

## The Help Facility

When the program is loaded, the first message to appear on the Prompt/Error line is the Help message:

For Help, type ?

The Help Facility reminds you of features and commands you have learned but may not remember. It is not intended as a substitute for the Instruction or Reference Manuals.

The Help message is just a reminder about how to call the Help Facility and disappears when you type any key. It does not have to be displayed to use the Help Facility.

If you need help, type a question mark (?).

Type:

?

The message that appears on the Prompt/Error line is the Help prompt:

Help: ? or topic:

The marker that you see on the Prompt/Error line is the cue. It marks your position on the line and is there to indicate that you must type a response. The options are:

**Option:**

**Shows:**

?

Instructions for using the Help Facility.

TOPICS ♦

A list of available topics.

*Any topic* ♦

Information on that topic. If you enter anything other than a topic or ?, the program tries to find it among the topics.

♦

The previous TK!Solver sheet.

Display the instructions for using Help. Type:

?

The screen shows instructions for using the Help Facility. After a time, there is a message at the top of the screen:

`Next page: Y N`

This tells you that there is more than one page of information about ? in the file.

Look at the next page by typing **Y**.

This screen shows more information about ?, but it is not about the Help Facility. The Help Facility gives you all the information it has about the topic, even if it is more than you want to know.

The Help prompt appears again at the top of the screen. When the Help prompt appears, it means that the current page of Help text is the last or only page of text on the topic, and you can ask for a new topic, or leave the Help Facility.

Now look at the list of Help topics by responding to the Help prompt with:

`TOPICS ♦`

This is a list of some of the topics covered in the Help File. This time, in response to the message at the top of the screen, type:

`N`

The Help prompt appears again.

To get information about a particular topic, type and enter it. For example, type:

`QUIT`

Press **♦** to enter the topic. While the program is finding the topic, it displays the message:

`Searching the Help file`

The search may take a few minutes. When the program has completed the search, it displays a page of text describing how to quit the program.

If the topic you have requested does not exist, a message on the Prompt/Error line tells you that it is not a topic and asks you if you want more help. Typing Y returns the Help prompt and typing N returns you to the previous sheet.

Return to the previous TK!Solver sheet by typing:



The screen should again look like Figure 4-1, except that there is no Help message.

### Using a Storage Diskette-- The Storage Command

The Storage command saves and loads files. If you want to keep your model, be sure to save it in a file before you quit the program.

The Storage command is one of the options you saw on the list of commands beginning with a slash (/).

Type the Storage command:

/S

It has seven options. These are shown on the Prompt/Error line:

Storage: L S V U F # D

The options are:

L	Load File
S	Save Model
V	Save Variables
U	Save Units
F	Save Functions
#	Save/Load DIF™ File (lists only)
D	Delete File

Three of these options are discussed in this chapter: L (Load File), S (Save Model), and D (Delete File). The other options are discussed in later chapters.

### **Loading a File**

The Load File option (L) of the Storage command locates a file on a diskette, loads the contents into the TK!Solver program, and displays the model on the screen.

Type:

L

The prompt is:

Load: Filename:

The cue on the Prompt/Error line prompts you to enter the name of the file you want to load.

Specify a file by typing the filename and entering it with **◆**. If you make a mistake while typing the filename, remember that **◀** erases the character to the left of the cue. The Instruction Files include a file named GRAVITY that is a sample model.

Type:

GRAVITY ◆

A message on the Prompt/Error line tells you that the program is loading the file.

The model in the file GRAVITY appears on the screen. The screen should look like Figure 4-2.

```
(1r) Rule: v=g*t

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
              v          ft/s        velocity
              t          s           time
              s          ft          distance
              g          ft/s^2      grav accel

===== RULE SHEET =====
S Rule
-----
* v=g*t
* s=.5*g*t^2
* g=980.665  "in cm per second squared
```

Figure 4-2

Look at the right end of the Status line. Because a new model has been loaded but not solved, the ! has appeared.

### Saving a File

The Save Model option (S) of the Storage command saves your current model under a filename that you assign. While you are working, it is a good idea to save your model from time to time so that it is not lost accidentally. If you do not save the model you are using when you quit the program, that version of the model will be erased.

**Note:** The Instruction Files are always available on the TK!Solver diskette.

The Save Model (S) and Load File (L) options of the Storage command will be especially useful to you as you are working through this manual. When you want to stop working for a while, save your current model in a file. Then, when you are ready to return to the Instruction Manual, load the file and continue.

To save a file on a diskette, the diskette must be formatted. If you have created an Instruction diskette, it has been formatted and can be used to save files. Don't reformat the Instruction diskette or you will erase the files on it.

The instructions below show you how to save the current model, GRAVITY, in a new file on the Instruction diskette.

Type the Storage command with the Save Model option:

```
/SS
```

The prompt is:

```
Save: Filename:
```

A file can have any name you choose, but it must follow the file naming conventions of your system as detailed in Chapter 3.

Name the new file TEST. Type:

```
TEST ♦
```

A message on the Prompt/Error line tells you that the program is saving the file.

**Note:** If your Memory Indicator shows **OUT**, the program may require an hour or more to save a file. You can avoid this by saving your model when the **LOW** message is displayed. Systems with low memory capacity will run out of memory sooner than systems with more.

The program saves the GRAVITY model under the filename TEST on the Instruction diskette. Notice that the model on your screen has not changed.

If you modify your model and want to save the new version, you can save it with a new filename, so that you have copies of both versions of the model, or with the same filename, so that the old version is erased and you have a copy of the new version only.

If you save a file using a filename that already exists on the diskette, the program asks for confirmation of Y or N before saving the new model using the old filename.

Type:

```
/SSTEST ◆
```

The prompt is:

```
Overwrite filename: TEST extension: Y N
```

Typing Y deletes the old file and saves the new file, and typing N cancels the Storage command. Type N .

**Note:** Refer to Chapter 2 for the specific extensions used for your system.

## Deleting a File

The Delete File option (D) of the Storage command deletes from the diskette the file you name. Once a file has been deleted, it cannot be loaded again; it has been erased.

Try the Delete File option by deleting your example file TEST. This time, instead of typing the filename, try using the Directory Scroll feature.

Type:

```
/SD
```

When you are prompted for the filename, type:

⇒

The filename that appears on the Prompt/Error line is the first filename in the TK!Solver directory.

When you scroll through the TK!Solver directory, the filenames roll past on the Prompt/Error line, as though a scroll of information were being unwound in front of you. The Directory Scroll feature of the Storage command scrolls through the names of the TK!Solver files. It scrolls in one direction only.

**Note:** If your system has a hierarchical directory structure, the names of the subdirectories for your current directory scroll past after the filenames. See the Hardware Reference Manual for more information.

The Directory Scroll feature can be used with any of the Storage command options and is especially useful if you can't remember the name of a file, or if you want to see what is on a diskette without quitting the program to use the operating system.

Press **⇒** again, and the filename of the second file in the directory is displayed.

Repeat pressing **⇒** until the filename **TEST** *extension* appears on the Prompt/Error line. This is the file you want to delete. If you scroll past it, press **⊗**, type the Storage command (**/S**) and the Delete File option (**D**) again, and scroll through the files until you find **TEST** *extension*. This is the file to be deleted. Press **◆** to choose it as the file you want.

**Note:** If you scroll past the last filename, the command stops and the prompt disappears.

**Note:** If you specify a filename, but do not specify an extension, the program responds by displaying the first occurrence of that filename with its extension. If the extension is not the correct one, press **⇒** and the program will find the next occurrence of that filename with its extension. When the program finds the correct file, press **◆**.

In all cases, the program requests an explicit confirmation for the delete so that you do not delete a file by mistake. In the example, the prompt is:

```
Delete file: TESTextension: Y N
```

Typing Y deletes the file, and typing N cancels the Storage command. Type:

```
Y
```

A message on the Prompt/Error line tells you that the file is being deleted.

Although the file has been deleted, the model is still in the program and remains on the screen. Deleting a file does not affect the program. You can continue using the model after the file is deleted and you can save it again, if you choose.

## Conclusion

This chapter has shown you how to use the Help Facility, and how to save, load, and delete files. The next chapter discusses how to move around in the TK!Solver program.

## **Chapter 5 Moving Around**

- Windows
- The Arrow keys
- The Switch command (;)
- The Goto command (:)
- The Select command (=)
- The Window command (/W)
- The Rule Sheet
- The Reset command (/R)—All option (A)

Models are created in the TK!Solver program by typing and entering information on the sheets. In order to do this, you must be able to move the cursor around the sheets and to display the different sheets on your screen.

This chapter is about moving around in the program. It gives instructions for moving the cursor, selecting the sheets you want to see on the screen, and clearing the program in preparation for a new model.

You will also be introduced to the Rule Sheet, which contains the equations for your model and is used to define the relationships between the variables.

---

### **To begin this chapter:**

You should have the TK!Solver program loaded, and the Variable and Rule Sheets for the GRAVITY model displayed, as at the end of Chapter 4.

If you quit the program after Chapter 4, load the TK!Solver program and then load the file GRAVITY from the Instruction diskette.

The screen should look like Figure 5-1.

The remaining figures in this manual are shown without the cursor. If you want to know where the cursor is, the Position Indicator in the figure tells you the field that contains the cursor.

(1r) Rule:  $v=g*t$

===== VARIABLE SHEET =====				
St Input	Name	Output	Unit	Comment
	v		ft/s	velocity
	t		s	time
	s		ft	distance
	g		ft/s^2	grav accel

===== RULE SHEET =====

S Rule

-----

- \*  $v=g*t$
- \*  $s=.5*g*t^2$
- \*  $g=980.665$  "in cm per second squared

Figure 5-1

## The Windows

Below the Message Area on your screen are two windows through which TK!Solver sheets can be seen. The Variable Sheet is in the top window, and the Rule Sheet is in the bottom window. You can display any of the TK!Solver sheets in either window, or one sheet in both windows. You can also manipulate the screen so that there is only one window, then split it again so that there are two.

When you load the TK!Solver program, the first sheets on your screen are the Variable Sheet in the top window and the Rule Sheet in the bottom window.

**Note:** When you load a model, the model loads looking as it did when you saved it. For example, if when you saved the model the Unit Sheet was in a single window, the model loads showing the Unit Sheet in a single window.

## Moving the Cursor

### The Arrow Keys and the Switch Command

The cursor shows your current position on the screen. It is moved around a sheet with the Arrow keys. For example, pressing **←** once moves the cursor one field to the left, and pressing **↓** twice moves the cursor two rows down.

The Switch command (;) moves the cursor between windows.

Try moving the cursor around both sheets. As you move it, look at the Status line at the top of the screen and notice that the Position Indicator, the label, and the information on the Status line all change according to the cursor position.

Move the cursor as far left, right, and up as you can. Notice that the cursor stops and you get an error signal at these boundaries. Now move the cursor down. As you move the cursor beyond the information displayed in the window, the sheet scrolls up to display more of the sheet. These two TK!Solver sheets scroll up and down; some others scroll for a limited range.

### The Goto Command

The Goto command (:) provides a quick method for moving the cursor to a specific field within a sheet.

For this example, first move the cursor into the Variable Sheet, then type the Goto command:

```
:
```

The prompt is:

```
Goto: Destination or search:
```

The cue is on the Prompt/Error line so that you can tell the program where you want the cursor to go.

To specify a destination, use the row number followed by the first letter of the column label for the field you want to go to.

For example, move the cursor to the Comment field in row 2 by typing:

2C ♦

Notice that the Position Indicator on the Status line represents the cursor position in the same format that is used for the Goto command.

If you type the letter without the number, the cursor moves to the specified column within the current row. Move it to the Name field in row 2 by typing:

:N ♦

Similarly, if you type the number without the letter, the cursor moves to the specified row within the current column. Move the cursor to row 7 in the Name column by typing:

:7 ♦

The Position Indicator is **7n**.

The Goto command is especially useful if you want to move the cursor to the top of the sheet or the last row containing an item. The top of the sheet is line 1. Move to the top of the sheet by typing:

:1 ♦

The cursor is now in the first field of the Name column.

If you type an asterisk (\*) instead of a number, the cursor moves to the last row on the sheet containing an item. Move the cursor to the last item in the Comment column by typing:

:\*C ♦

The Goto command can also be used to search for a specific item within a column. Typing a quotation mark (") tells the program that you want to search for an item. After the ", type the item you want to find as it appears on the screen, and press ♦. The Goto command searches only within the current column.

**Note:** The item you enter for the search must be exactly what you can see on the screen or the program will not be able to find it. If the item is longer than the width of the field, enter only that portion of the item that is visible.

The Goto command (:) can be used anywhere on the screen, but to see how it works, move the cursor so that the field you are going to search for is off the screen. Type:

```
:20 ◆
```

Now find the word “time” in the Comment column by typing:

```
:"time ◆
```

The cursor moves to the field containing the word **time**.

## **The Sheets and the Windows**

### **The Select Command**

The Select command (=) changes the sheet in the window containing the cursor. The cursor is currently in the top window. Select a different sheet for that window by typing:

```
=
```

The message is:

```
Sheet:  V R U G L F P T
```

These options represent the titles of the TK!Solver sheets:

<b>Option</b>	<b>Sheet</b>
V	Variable Sheet
R	Rule Sheet
U	Unit Sheet
G	Global Sheet
L	List Sheet
F	User Function Sheet
P	Plot Sheet
T	Table Sheet

Select the Unit Sheet for the top window by typing:

```
U
```

The screen should look like Figure 5-2 with the Unit Sheet in the top window. The Unit Sheet is discussed in the next chapter.

(1f) From: ft

```
===== UNIT SHEET =====
From      To      Multiply By  Add Offset
-----
ft        cm        30.48006
in        cm        2.540005
m         cm        100
km        cm        100000
ft/s^2    cm/s^2    30.48006
ft/s      cm/s      30.48006
in/s      cm/s      2.540005

===== RULE SHEET =====
S Rule
-----
* v=g*t
* s=.5*g*t^2
* g=980.665  "in cm per second squared
```

Figure 5-2

## The Window Command

The Window command (/W) changes the screen from two windows to one, showing the sheet containing the cursor, or from one window to two, showing both the sheet currently containing the cursor and another that you choose. To see how it works, type:

/W

With one exception, the options are the same as those for the Select command (=). The message is:

Window: 1 V R U G L F P T

The only difference is the option **1**. When the screen is split into two windows, this option is the only one you can use with the Window command (/W). It changes the screen from two win-

dows to one window showing the sheet containing the cursor. The cursor was in the Unit Sheet when you typed /W, so the Unit Sheet will be the sheet displayed in the one window. Type:

1

The screen should look like Figure 5-3. There is now one window containing the Unit Sheet for the GRAVITY model.

```
(1f) From: ft

===== UNIT SHEET =====
From      To      Multiply By  Add Offset
-----
ft        cm        30.48006
in        cm        2.540005
m         cm        100
km        cm        100000
ft/s^2    cm/s^2    30.48006
ft/s      cm/s      30.48006
in/s      cm/s      2.540005
```

**Figure 5-3**

Select the Rule Sheet for the window by typing the Select command:

=R

Now change the screen back to two windows with the Window command. The top window will contain the current sheet, the Rule Sheet. Type:

/W

The remaining options for the Window command represent the titles of the TK!Solver sheets. When you split the screen into two windows, the program needs to know which sheet you want in the bottom window. Display the Variable Sheet in the bottom window by typing:

V

The screen should look like Figure 5-4. These are the same sheets you saw at the beginning of the chapter, but now they are in different windows—the Rule Sheet in the top window and the Variable Sheet in the bottom window.

```
(1) Input:
===== RULE SHEET =====
S Rule
- ---
* v=g*t
* s=.5*g*t^2
* g=980.665 "in cm per second squared

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
          v          ft/s      velocity
          t          s         time
          s          ft        distance
          g          ft/s^2    grav accel
```

Figure 5-4

## The Rule Sheet

Now that you know how to move around, let's take a more detailed look at the Rule Sheet.

The Rule Sheet contains the equations that establish the relationships between the variables. It has two columns, the Status column on the left and the Rule column on the right.

### The Status Column

The Status column contains output fields. This means that only the program can generate information in this column; you cannot enter anything there, although you can move the cursor to the column to look at status messages.

In the GRAVITY model, the Status fields for all of the equations contain asterisks (\*). The \* is the symbol the program uses to tell you that the rule, or equation, is unsatisfied. Move the cursor over one of the asterisks and look at the Status line at the top of the screen. The message is:

```
Status: * Unsatisfied
```

When a model is solved, the Status fields for the satisfied equations are blank and the message on the Status line is **Satisfied**.

The other character the program uses in the Status column is the greater than symbol (>). If you made an error in the model, when you try to solve it the program generates a > next to the equation found to be in error. You can look at the error message just as you looked at the one for the unsatisfied equation by moving the cursor to the Status field containing the symbol and looking at the Status line.

### The Rule Column

The Rule column contains the rules. All rules must be equations composed of expressions. TK!Solver expressions have the same form as algebraic expressions.

You can enter one rule per row. Variable names must begin with a letter or one of the special characters @, #, \$, %, or \_ and may contain letters, numbers, and those special characters.

If you want to put a comment on the Rule Sheet, you can enter it following a rule or instead of a rule by typing a quotation mark (") and then the comment. When the program is solving a model, it ignores anything in a row after a quotation mark.

Because most keyboards do not have all the standard symbols used for mathematics, the TK!Solver program uses computer conventions for standard arithmetic operators. These operators are:

Operator	TK!Solver Symbol
+	+
-	-
×	*
÷	/
exponent	^

**Note:** Multiplication must always be specified with \*. The TK!Solver program does not assume multiplication. For example, the equation  $ab + ac = a(b + c)$  must be entered as  $a*b + a*c = a*(b + c)$ . The program considers  $ab$  and  $ac$  to be variable names and  $a()$  to be a function.

**Note:** Scientific notation using E may be used instead of the exponent ^, if desired.

**Note:** When solving an equation, the TK!Solver program first determines whether values are positive or negative and then solves the equation from left to right in this order of precedence: exponentiation, multiplication and division, addition and subtraction.

There are many functions built into the TK!Solver program. Some are standard mathematical functions, such as *sine* and *tangent*, and others are functions that have been created for the TK!Solver program, such as *pi* and *dot*. The Reference Manual contains a complete list of the built-in functions available and the syntax of each.

You may also define your own functions. This program feature is described in Chapter 16.

**Note:** The program does **not** distinguish between uppercase and lowercase characters when looking at the names of TK!Solver built-in functions; you can enter the TK!Solver names for the built-in functions in both uppercase and lowercase characters. The program distinguishes between uppercase and lowercase for all other names and functions in the program.

When using functions in TK!Solver rules, the syntax is:

$f(x)$

For example:

$\sin(x)$

### **Resetting the Program**

The next chapter shows you how to begin building a model. The model you will use is a new one that is used throughout the manual to show you a variety of TK!Solver features and commands. Before loading the new model, you must reset the program and clear the GRAVITY model from the screen.

The Reset command (/R) resets all or part of the program without quitting it. Type:

/R

The message is:

Reset: V S A

The three Reset options are:

<b>Option</b>	<b>Action</b>
Variable Value (V)	Resets all values on the Variable Sheet
Sheet (S)	Resets the sheet currently containing the cursor.
All (A)	Resets the entire program.

The Variable Values (V) and Sheet (S) options are explained in Chapter 7. For this example, use the All option (A) to reset the program, clearing all the sheets. Type:

The message is:

```
Reset all: Y N
```

So that you don't reset the program by accident and lose your model, each Reset option requires a confirmation of Y before the program is reset. Typing N cancels the Reset command. Type:

```
Y
```

The TK!Solver program has been reset and is cleared of the GRAVITY model. The screen looks as it did when you first loaded the program, with a blank Variable Sheet in the top window and a blank Rule Sheet, containing the cursor, in the bottom window. You can now enter or load a new model.

### **Conclusion**

This chapter has explained how to move around the TK!Solver program and introduced the Rule Sheet. The next chapter shows how to enter items and begin building a model.

## **Chapter 6 Building a Model**

- The TRAVEL model
- Types of fields
- Entering equations
- Automatic variable entry
- Merging files
- The Variable Sheet
- The Variable subsheet
- The Dive command (>)
- The Return command (<)
- The Unit Sheet

To use the TK!Solver program to solve a model, you must first enter the model on the TK!Solver sheets. Which sheets you use depend on which features of the program your model requires. Most models will use at least the Rule and Variable Sheets because the Rule Sheet holds the equations that define variable relationships, and the Variable Sheet holds information about the variables.

Any model that requires conversions for units of measurement also needs the Unit Sheet. Unit conversions are defined on the Unit Sheet, and the program needs these conversion definitions to convert values from one unit to another.

The Rule Sheet was introduced in Chapter 5. This chapter introduces the Variable and Unit Sheets and gives instructions for building a sample model named TRAVEL. TRAVEL is a simple model and is used through the rest of this manual to show you how to build a model and expand it using the various TK!Solver features.

---

### **To begin this chapter:**

The program should be reset with the Variable Sheet in the top window and the Rule Sheet in the bottom window, both blank, as at the end of Chapter 5.

If you quit the program after the Chapter 5, load the program.

The screen should look like Figure 6-1.

```
(1r) Rule:
For Help, type ?
===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----

```

```
===== RULE SHEET =====
S Rule
-----
```

Figure 6-1

## The Travel Model

The TRAVEL model is a simple model that finds the length and cost of a trip by car. It defines the relationships between mileage, speed, distance, gas price, and the cost of a trip. The model uses three equations. While you are working through this chapter, you will enter the equations on the Rule Sheet, the characteristics of the variables on the Variable Sheet, and unit conversion definitions on the Unit Sheet.

### Types of Fields

As you saw in the examination of the Rule Sheet, you cannot enter information in all fields. Output fields, such as in the Status column on the Rule Sheet, can contain only information generated by the program. The TK!Solver program has two other kinds of fields, entry fields and option fields.

In option fields, you can type only one of a limited set of characters. The characters that can be typed as options and what they mean differ according to the field and the sheet. Option fields are discussed individually as part of the sheet descriptions.

The items that you type and enter go into entry fields. The cursor marks your current position on the screen. If the cursor is in an entry field, when you type a character that character appears in the field. When you have finished typing the item that you want in the field, you enter it by pressing either  $\blacklozenge$  or an Arrow key.

For example, the Rule column on the Rule Sheet consists of entry fields. If the cursor is not in the first Rule field, move it there, then type the characters *mlg*.

A cue has appeared in the field, like the one that appeared on the Prompt/Error line when you were prompted for a filename in the Storage command (/S). The cue marks your position in the field.

### Entering Equations

The TRAVEL model has three equations. They are:

$$mlg = d / gas$$

$$s = d / t$$

$$cost = pr * gas$$

As you can see, the characters that you just typed form the first variable name in the first equation. This equation states that mileage is equal to distance divided by the amount of gas. The variable *mlg* stands for mileage, *d* for distance, and *gas* for amount of gasoline. Finish typing the equation so that it looks like the equation below. If you make a typing mistake, remember that  $\blacktriangleleft$  erases the character to the left of the cue.

$$m \text{ l } g = d / gas$$

(Equation 1)

When you have typed the equation correctly, enter it by pressing  $\blacklozenge$ . The cue disappears, and the equation is displayed on the Status line at the top of the screen.

The screen should look like Figure 6-2. Notice that each of the variable names in the equation has been entered on the Variable Sheet in the column labeled **Name**.

```
(1r) Rule: mlg=d/gas

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
              mlg
              d
              gas

===== RULE SHEET =====
S Rule
-----
* mlg=d/gas
```

Figure 6-2

Move the cursor over the asterisk (\*) on the Rule Sheet and look at the top of the screen. The message on the Status line says the equation is **Unsatisfied**.

Move the cursor over and down into the second field in the Rule column and enter Equation 2. The variable *s* stands for speed, and *t* for time. The variable *d* is the same as in Equation 1. Type:

```
s=d/t ◆
```

(Equation 2)

The screen should look like Figure 6-3.

This is a good time to save the model in a file. Using the Storage command (/S), save the model on the Instruction diskette using the filename TRAVEL. Type:

```
/SS
```

At the prompt type:

```
TRAVEL ◆
```

When the message disappears, the file has been saved.

```
(2r) Rule: s=d/t

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
              mlg
              d
              gas
              s
              t

===== RULE SHEET =====
S Rule
-----
* mlg=d/gas
* s=d/t
```

Figure 6-3

### Automatic Variable Entry

The two new variable names from the second equation were inserted automatically on the Variable Sheet just as the first ones were. This automatic feature is controlled through the Global Sheet. Call the Global Sheet into the bottom window by typing:

```
=G
```

The Global Sheet is made up of labeled lines, each of which is associated with one field. The first field, labeled **Variable Insert ON**, is the switch that controls the automatic entry of variables and is an option field. If it is set to **Yes**, variable names are automatically entered on the Variable Sheet in the Name column when they are entered on the Rule Sheet as part of an equation.

When the program is loaded, this field is set to **Yes**. If you want to insert variables by hand, you can change this field to **No** by typing N. Typing Y returns it to **Yes**.

**Note:** This manual assumes throughout that the Variable Insert ON field is set to **Yes**.

Other fields on the Global Sheet set specifications for the solvers and the printer. The Global Sheet is discussed in more detail in Chapters 9 and 12. Call the Rule Sheet back into the bottom window with the Select command (=). Type:

```
=R
```

### Merging Files

It is possible to load more than one file at a time, merging them to create a single model. The last equation for TRAVEL is an Instruction File named COST. Load COST into the existing model using the Storage command (/S). Type:

```
/SL
```

At the prompt, type:

```
COST ◆
```

The screen should look like Figure 6-4. Notice that the new file added information on the Variable Sheet as well on the Rule Sheet. Only the two new variable names were added to the existing list. When files are merged, the program does not duplicate variables, and it coordinates the current information with the new information. Items on the Rule Sheet and Unit Sheet, however, are duplicated if they appear in both the current model and the merged file. You will see an example of this later in this manual.

**Note:** The information in the file COST has been merged with the TRAVEL model, but the file COST itself is unaffected.

Having added the equations and variables from the file COST to the model, save the file again using the same name, TRAVEL. This time the program asks if you want to overwrite the existing file named TRAVEL. Because you are saving a newer version of the model, you do want to overwrite the old file. Respond to the message by typing **Y**. This causes the program to delete the old information in the file TRAVEL and replace it with the new information.

```
(1r) Rule: mlg=d/gas

===== VARIABLE SHEET =====
St Input      Name      Output    Unit      Comment
-----
              mlg
              d
              gas          gal      amount of g
              s
              t
              cost        $        cost of tri
              pr          $        price of ga

===== RULE SHEET =====
S Rule
-----
* mlg=d/gas
* s=d/t
* cost=gas*pr
```

Figure 6-4

Let's take a closer look at the Variable Sheet while entering some information for the TRAVEL model.

### The Variable Sheet

The Variable Sheet contains characteristics of the variables. Each row describes one variable, and each column lists a particular characteristic of the variables. There are six columns on the Variable Sheet, and, therefore, six characteristics shown for each variable.

#### The Status Column — Output Fields

The first column on the left is the Status column. Each Status field is both an output and an option field. If the value of a variable causes an error during solution, the program generates the symbol  $\rangle$  in the Status field. When the cursor is moved over the error symbol, an error message is displayed on the Status line at the top of the screen.

The use of Status fields as option fields is explained at the end of this section.

## The Input Column

The column to the right of the Status column is the Input column. The Input fields are entry fields that contain the input values for the variables. Values must be numbers or expressions that the program can evaluate.

Values on the Variable Sheet are shown rounded to fit the field width. If you move the cursor to a field containing a value, the value is shown on the Status line to the full accuracy of 12 significant digits. All calculations use full accuracy. For the range of values accepted by your system, see the Hardware Reference section of the Reference Manual.

Symbolic values must begin with an apostrophe ('). For a more detailed explanation of values used with the TK!Solver program, see the Reference Manual.

The fields in the Input column can act as a calculator to evaluate expressions. For example, move the cursor to the Input field next to *mlg* and type:

40 ♦

The value **40** is entered in the field and appears on the Status line at the top of the screen.

Now change the value in the Input field by typing:

20-5 ♦

The program evaluates the expression in the Input field and produces the value **15** as the mileage. In the TK!Solver program, any field that can contain input values can also evaluate expressions.

Now move the cursor down to the next Input field. Suppose you used 5 gallons of gas and want to determine the distance travelled. Enter an expression. Type:

*mlg*\*5 ♦

Because *mlg* has a value, the program can evaluate the expression and enters the value **75** in the Input field.

### The Name Column

The next column to the right is the Name column. Variables are associated with the rules through the variable names. The Name column contains the seven variable names for the model TRAVEL. Name fields are entry fields. Variable names must begin with a letter or one of the special characters @, #, \$, %, or \_\_, and may contain letters, numbers, and those special characters.

### The Output Column

The Output column, to the right of the Name column, is where the program displays the values determined when the model is solved.

### The Unit Column

The Unit column is next and contains the units of measurement for the variables. Although only one unit is displayed for each variable on the Variable Sheet, there are actually two units for each variable, a Calculation unit and a Display unit.

The first unit name you enter for each variable in its Unit field becomes the Calculation unit. The Calculation unit is the unit assumed for the variable when it appears in a rule. All calculations and unit conversions done internally use the variable value in the Calculation unit. It is essential to set Calculation units if you are using rules that require compatible units and you want to display the values in different units.

For example, look at the first equation on the Rule Sheet:

$$mlg = d / gas$$

If *mlg* is to be calculated in miles per gallon, you must calculate *d* in miles and *gas* in gallons, otherwise the program cannot calculate the values correctly. If the Calculation unit for *d* is in feet and for *gas* is in liters, the program can solve the model, but the output values would not correctly reflect the equation. As long as the Calculation units are compatible, the Display unit can be feet for *d* and liters for *gas* or any other units whose conversion has been defined on the Unit Sheet.

The Display unit is the unit you see on the screen. It may be the same as the Calculation unit. To change the Display unit, you simply enter a new unit name. The new unit entered becomes the Display unit, but the Calculation unit remains the one you originally entered for the variable. You will see how to change a Calculation unit later in this chapter.

**Note:** The program stores all values in the Calculation unit. If you use a variable name as part of an expression to be evaluated, the program uses the value of the Calculation unit, not the Display unit. When your values are displayed in mixed units, you should make cautious use of the calculation feature of entry fields.

The TRAVEL model requires several different units for the different variables. First, enter the Calculation units for the variables in the model.

The Calculation unit for *mlg* is mi/gal. Move the cursor to the Unit field for the variable *mlg* and enter the Calculation unit by typing:

mi / gal ♦

Enter the remaining Calculation units in the same way, giving *d* the unit mi, *s* the unit mi/h, and *t* the unit h.

**Note:** Be careful to enter only one unit name per field. If you make a typing error, use either ◀ or ⊗ to correct your error. The first unit name you enter, even if it is in error, becomes the Calculation unit. The section below about the Variable subsheet shows you how to change a Calculation unit. If you accidentally enter an incorrect Calculation unit, you will be able to correct it while working on that section.

Suppose you want the Display unit for *mlg* to be km/l. With the cursor in the Unit field for the variable *mlg* type:

km / l ♦

When you entered the new Display unit, the program displayed a question mark next to the value in the Input field. Unit conversions must be defined on the Unit Sheet before the program can perform a unit conversion, and you have not yet filled in the Unit Sheet. The program displays a question mark next to the value to

tell you that the unit conversion has not been defined and it does not have enough information to make the conversion from the Calculation unit, mi/gal, to the Display unit, km/l. Leave the units as they are for now. They will be used later in an example.

### The Comment Column

The Comment column provides a space for you to keep notes and comments about the variable in that row. It is ignored by the solvers.

The Comment column for TRAVEL contains comments about the three variables, *gas*, *cost*, and *pr*, that were loaded from the file COST.

Move the cursor to the Comment field for *mlg* and type the comment:

```
mi leage
```

Instead of pressing **◆** to enter the comment, press **␣**. The comment is entered and the cursor moves down to the next field. When you are in an entry field, the Arrow keys both enter information and move the cursor.

Fill in the rest of the comments that are missing:

Variable	Comment
d	distance
s	speed
t	time

The screen should look like Figure 6-5.

### The Status Column — Option Fields

Now that you are more familiar with the Variable Sheet, let's return to the discussion of the Status fields and their use as option fields. A Status field on the Variable Sheet has five options available that affect the variable in that row.

To enter an option, simply move the cursor to the Status field in the appropriate row and type the letter representing the option you want. Options do not require **◆**.

```
(5c) Comment: time

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
   ?15        mlg          km/l        mileage
   75         d            mi          distance
                gas          gal         amount of g
                s            mi/h        speed
                t            h          time
                cost         $          cost of tri
                pr          $          price of ga

===== RULE SHEET =====
S Rule
-----
* mlg=d/gas
* s=d/t
* cost=gas*pr
```

Figure 6-5

Options available for the Status column on the Variable Sheet are:

Option	Result
Input (I)	Moves the output value to the Input field. Deletes Guess option (G).
Output (O)	Moves the input value to the Output field. Deletes Guess option (G).
List (L)	Associates the variable with a List of values. (See Chapter 13.) Can coexist with Guess option (G).
Guess (G)	Designates the input value as the first guess for an iterative solution. (See Part III.) Can coexist with List option (L).
Blank (B)	Blanks the variable value.

Use the Blank option (B) to erase the Input value for *mlg*. Move the cursor to the Status field for *mlg* and type:

B

Blank out the Input value for *d* by the same method.

**Note:** The program remembers the last value that was blanked. That value can be returned to the screen if you type the Input (I), Output (O), or Guess (G) option in the Status field when the Input and Output fields are blank.

### The Variable Subsheet

Several of the TK!Solver sheets have subsheets that show an expanded amount of information. The Variable Sheet is one of these. For each variable there is a subsheet that lists all of the characteristics of that variable. It is another way of looking at the variable.

### The Dive Command (>)

Subsheets are called with the Dive command (>). This command replaces the Variable Sheet with the subsheet for the variable in the row containing the cursor.

For example, display the Variable subsheet for the variable *mlg* by moving the cursor into the row containing *mlg* and typing:

>

The screen should look like Figure 6-6. The Variable Sheet is replaced with the subsheet for the variable *mlg*.

```
(s) Status:
===== VARIABLE: mlg =====
Status:
First Guess:
Associated List:
Input Value:
Output Value:
Display Unit:          km/l
Calculation Unit:     mi/gal
Comment:              mileage

===== RULE SHEET =====
S Rule
- - - -
* mlg=d/gas
* s=d/t
* cost=gas*pr
```

Figure 6-6

Instead of columns and rows of fields, this sheet consists of labeled lines, each of which has one field associated with it. There are only eight fields on the sheet.

The Status, Input, Output, and Comment fields show the same information as those on the Variable Sheet. The only difference in these fields is that the values displayed on a Variable subsheet are shown rounded to 10 significant digits.

The First Guess field sets a default first guess for iteration. TK!Solver iteration is discussed in Chapters 10 through 12.

The Associated List field names the list of values associated with the variable. If there is an associated list, the Status field is set to L. Lists are discussed in Part IV.

There are two unit fields, one for the Display unit and one for the Calculation unit. As you can see, they are different for *mlg*. It is unlikely you would want to change the Calculation unit once your model is built, but if you wanted to change it, it is best to dive to the subsheet and change the unit name in the Calculation Unit field.

**Note:** If you made a mistake entering any of your Calculation units for TRAVEL, you can correct them by diving to the subsheet for that variable and changing the Calculation unit there.

**Note:** If you set the Calculation unit on the subsheet, the program does not assume a Display unit as it does when the Calculation unit is set on the Variable Sheet.

The Position Indicator is different on the Variable subsheets than on the Variable and Rule Sheets. It consists of a single letter identifying the field. The Goto command (:), therefore, requires only the identifying letter to locate a field.

For example, move the cursor to the Display Unit field by typing:

```
:D ↵
```

If you change an item on this sheet, the change is reflected on the Variable Sheet. To make the Display and Calculation unit the same for *mlg*, change the Display unit by typing:

```
mi/gal ♦
```

### The Return Command (<)

The Return command (<) moves you from the subsheet back to the previously displayed sheet. Return to the Variable Sheet to look at the change you have made by typing the Return command:

```
<
```

Notice that the cursor is in the same field as it was when you typed the Dive command (>). If you had wanted to move from the subsheet to another sheet, you could have done so by using the Select command (=) instead of the Return command (<).

The Unit field for *m/g* displays **mi/gal**.

### The Unit Sheet

The Unit Sheet is where you enter the definitions that enable the program to convert values from one unit of measurement to another. Although units have been entered on the Variable Sheet, the program cannot convert values until you define the conversions on the Unit Sheet.

A Unit Sheet with the conversions already defined for TRAVEL is in a file on the Instruction diskette. When this file has been loaded and merged with TRAVEL, the model will be ready to use.

Use the Select (=) and Window (/W) commands to display the Unit Sheet in a single window. Type:

```
=U
/W1
```

**Note:** You don't have to display a particular sheet when you load a file, but you have been asked to display the Unit Sheet here so that you can see what happens when you load the file.

The file is named TUNITS. Load it from the Instruction diskette using the Load option (L) of the Storage command (/S). Type:

```
/SL
```

At the prompt, type:

```
TUNITS ♦
```

The screen should look like Figure 6-7. Use the Storage command (/S) with the Save Model option (S) to save TRAVEL again so that your saved model contains the units.

```
(1f) From: mi

===== UNIT SHEET =====
From      To      Multiply By  Add Offset
-----
mi        km        1.609
mi/h      km/h      1.609
mi/gal    km/l      .4251
gal       l         3.785
h         min       60
C         F         1.8       32
```

Figure 6-7

The TK!Solver program uses the Unit Sheet to convert values wherever a unit conversion is desired. All unit conversions must be defined on this sheet.

All fields on the Unit Sheet are entry fields. The Multiply By field is the conversion factor that relates the units in the From and To fields. The From field names the unit from which the factor converts and the To field names the unit to which the factor converts. The Add Offset field contains the offset factor for conversion if one is required; otherwise it remains blank.

You may name your units anything you choose. Unit names can be created from all printable characters, including blanks, but blanks at the beginning or end of a unit name are deleted when the name is entered.

The conversion factors are also up to you. If you define the conversion incorrectly, the conversions made by the program will be incorrect.

Use the Window command (/W) to split the screen and look at the Variable Sheet to see which units have had conversions defined. Type:

```
/WV
```

As you can see, all units have at least one conversion defined on the Unit Sheet except for \$.

Suppose a trip is taking you from America to Germany, and you want to know the conversion of price and cost from American dollars to Deutschmarks. Assume for this example that the rate of exchange is .44 DM to the American dollar.

Move the cursor into the Unit Sheet, then move it to the last row containing an item by typing:

```
:*◆
```

Move the cursor down into the first blank row, then type \$ in the From field. Use  $\Rightarrow$  to enter the unit and move the cursor. Enter the unit **DM** in the To field and the conversion factor **.44** in the Multiply By field.

```
(7m) Multiply By: .44
```

```
===== UNIT SHEET =====
From      To          Multiply By  Add Offset
-----
mi         km           1.609
mi/h      km/h         1.609
mi/gal    km/l         .4251
gal       l            3.785
h         min          60
C         F            1.8          32
$         DM           .44

===== VARIABLE SHEET =====
St Input   Name      Output    Unit      Comment
-----
           mlg       gal       mi/gal    mileage
           d         mi        mi         distance
           gas      gal       gal        amount of g
           s         mi/h      mi/h      speed
           t         h         h          time
           cost    $         $          cost of tri
           pr         $         $          price of ga
```

Figure 6-8

You have defined the conversion for the units \$ and DM and the screen should look like Figure 6-8. The Unit Sheet and unit conversions are discussed in more detail in Chapter 8.

Save TRAVEL again. Type:

```
/SS
```

At the prompt, type:

```
TRAVEL ♦
```

Confirm the overwrite by typing **Y**.

### Conclusion

You have just built the model TRAVEL using information from a variety of sources, and the model is ready to be used. You may want to quit the TK!Solver program and make a backup copy of your file TRAVEL using your operating system. The next chapter shows you how to solve some problems using the model.

## Chapter 7 Solving a Model

- Solving a problem
- The Action command (!)
- Underdefining a model
- Changing the values
- The Storage command (/S)—Save Variables option (V)
- The Blank command (/B)
- The Reset command (/R)—Sheet option (S)
- The Delete command (/D)
- The Reset command (/R)—Variable Values option (V)
- Overdefining a model
- Changing the model
- The Insert command (/I)
- The Edit Field command (/E)

Now that you have created a model, you can solve a variety of problems concerning the variables in the model. You can also edit the model by adding, deleting, or changing rules, and by changing the input values and variables.

In this chapter, you will begin using the TRAVEL model to solve several simple problems. The chapter then shows you how to change the model to solve a more complex set of problems.

---

### To begin this chapter:

At the end of Chapter 6, the screen was split with the Unit Sheet for TRAVEL in the top window and the Variable Sheet in the bottom window. Use the Select command (=) to display the Rule Sheet in the top window so that the screen is split with the Rule Sheet in the top window and the Variable Sheet in the bottom window.

If you saved the TRAVEL model and quit the program at the end of the last chapter, load the program, then use the Storage command (/S) to load the file TRAVEL. The Unit Sheet should be in the top window and the Variable Sheet in the bottom window. Use the Select command (=) to put the Rule Sheet in the top window.

The screen should look like Figure 7-1.

```
(1r) Rule: mlg=d/gas

===== RULE SHEET =====
S Rule
-----
* mlg=d/gas
* s=d/t
* cost=gas*pr

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
              mlg              mi/gal     mileage
              d              mi         distance
              gas             gal        amount of g
              s             mi/h       speed
              t              h         time
              cost          $         cost of tri
              pr              $         price of ga
```

Figure 7-1

## Solving a Problem

This is the first problem you will solve using the TRAVEL model:

If your car gets 27 miles to the gallon, and the price of gasoline is \$1.25 a gallon, how far can you travel on \$10 worth of gasoline?

On the Variable Sheet, enter the value 27 for the variable *mlg* by moving the cursor to the Input field beside the variable and typing:

27 ♦

Now enter the values 10 in the Input field for *cost* and 1.25 in the Input field for *pr*. Notice that the values entered are in the Display units.

## The Action Command (!)

The exclamation point (!) is the Action command, and it works differently on different sheets. If the cursor is in any field in the Variable or Rule Sheet and the Action command (!) is typed, the

program attempts to solve the model. You must hold down the Shift key while typing the exclamation point. Solve the model by typing:

!

The screen should look like Figure 7-2. Output values have been solved for the two variables *d* and *gas*. With the given input values, you could go 216 miles on 8 gallons of gasoline.

```
(7i) Input: 1.25
===== RULE SHEET =====
S Rule
-----
mlg=d/gas
* s=d/t
cost=gas*pr

===== VARIABLE SHEET =====
St Input      Name      Output    Unit      Comment
-----
    27        mlg                mi/gal    mileage
                d          216       mi        distance
                gas         8         gal       amount of g
                s                mi/h      speed
                t                h         time
    10        cost                $         cost of tri
    1.25      pr                $         price of ga
```

Figure 7-2

### Underdefining a Model

The values for *s* and *t* were not solved, and, as you can see by looking at the Rule Sheet, the one equation containing those variables remains unsatisfied. The model is underdefined because there are not enough input values given for the program to completely solve the model. The values for *s* and *t* remain blank.

Enter the value 55 in the Input field for the variable *s* and type the Action command (!) again. This time a value is solved for *t*, and all the equations are satisfied.

You may have noticed that while the program was solving the model, a message appeared on the Prompt/Error line:

```
D i r e c t   S o l v e r
```

The Direct Solver is one of the TK!Solver program's methods of solution. The program can also solve problems by an iterative method using the Iterative Solver. The concept of iteration and the use of the Iterative Solver are discussed in Part III.

Move the cursor over the Output field for  $t$  and look at the Status line. Notice that on the Variable Sheet, the value for  $t$  is rounded to fit the width of the column while on the Status line the value is shown to the full accuracy of 12 digits.

### Changing the Values

There are three ways to change an item in an entry field:

- Enter a new item, replacing the existing item in the field.
- Blank the existing item.
- Edit the existing item.

Suppose you now have \$50 to spend instead of \$10. Change the value for *cost* to 50 by moving the cursor to the Input field for *cost* and entering the new value. When you press  $\blacklozenge$  or an Arrow key, the new value replaces the old one. Any item in an entry field can be changed in this way.

When you changed the value, the Solution Indicator on the right end of the Status line reappeared. When a model is changed, the program does not automatically solve it again. You must type the Action command (!) again to solve it.

Solve the model with the new value by typing:

```
!
```

The screen should look like Figure 7-3 with all the output values changed. The Solution Indicator on the Status line disappears when the model is solved.

(6i) Input: 50

```

===== RULE SHEET =====
S Rule
-----
mlg=d/gas
s=d/t
cost=gas*pr

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
      27      mlg      1080      mi/gal    mileage
              d        40        mi        distance
              gas       40        gal       amount of g
      55      s        19.636364  mi/h      speed
              t
      50      cost
      1.25    pr

```

Figure 7-3

### The Storage Command—Variables Option (V)

The next sections of this chapter show you how to blank fields and delete rows. It uses the Variable Sheet for examples, so before you begin blanking parts of your model, save the current Variable Sheet. Then you can load it later and use it again.

You can save the variables using the Save Variables option (V) of the Storage command (/S). Your cursor can be on any sheet, but only the Variable Sheet is saved. Because subsheets are just another way of looking at the information on the Variable Sheet, any subsheets that are associated with the Variable Sheet are saved along with the Variable Sheet. Type:

```
/SV
```

At the prompt, enter the filename TVAR. The variables for TRAVEL are now saved in the file TVAR.

## The Blank Command

You have used the Blank option (B) in the Status column to blank values on the Variable Sheet. The Blank option (B) is used to blank values only on the Variable Sheet. There are two other ways to blank an entry field that can be used on any sheet: the Reset command (/R) that you used in Chapter 5, and the Blank command (/B).

The Blank command (/B) is used to blank a specific field or column of fields. For example, to blank the variable name for *s*, move the cursor to the Name field for *s* and type:

```
/B
```

The prompt is:

```
Blank: Point to last field (4n)
```

If you wanted to blank a column of fields, you would use the cursor to point to the last field you wanted to blank. Blank just the name *s* by typing:



The name disappears from the screen.

Now blank a column of fields. Move the cursor to the top of the Input column and type:

```
/B
```

At the prompt, use the cursor to point to the last item in the Input column. Notice that as you move the cursor, the number and letter in parentheses at the end of the prompt message changes to reflect the cursor position. The Position Indicator on the Status line does not change, but reflects the original cursor position. When you have moved the cursor to the last item in the Input column, press .

The screen should look like Figure 7-4. All of the input values were blanked, and the cursor returns to the field in which you originally typed the Blank command (/B).

(1i) Input:

```
===== RULE SHEET =====
S Rule
-----
mlg=d/gas
s=d/t
cost=gas*pr
```

```
===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
           mlg           mi/gal      mileage
           d           1080        mi         distance
           gas          40          gal        amount of g
           t           19.636364   h          speed
           cost         $           cost of tri
           pr           $           price of ga
```

Figure 7-4

The previous values are still available, however. Move the cursor to the Status field for the variable *cost* and type **I**. The last value reappears in the Input field. The Status field options Input (I), Output (O), and Guess (G) all return the most recent value.

You can use the Goto command (:) with the Blank command. Move the cursor to the top of the Unit column and type **/B**. At the prompt, point to the bottom of the Unit column by typing:

```
: * ◆
```

A message returns requiring a confirmation of the cursor position. Press **◆** to confirm, and the entire Unit column is blanked.

**Note:** If you use the Blank command (**/B**) in a Unit field on the Variable Sheet, both the Calculation and Display units are blanked.

The Blank command (**/B**) does not work on output fields.

## The Reset Command—Sheet Option (S)

There is another way to reset the screen. In a previous chapter, you used the All option (A) of the Reset command (/R) to reset the program, clearing all the sheets. The Sheet option (S) resets only the sheet containing the cursor.

With the cursor on the Variable Sheet, type:

```
/RS
```

Confirm the command by typing **Y**.

The Variable Sheet is reset and is now blank, but the Rule Sheet remains as it was. With the Select command (=), call the Unit Sheet into the bottom window and notice that it also has not changed. The Sheet option (S) of the Reset command (/R) affects only the sheet containing the cursor; the rest of the model is unaffected.

Let's look at what happens if two files containing the same information are merged. With the TRAVEL model still loaded, load the file TRAVEL again.

Use the cursor to scroll through the Variable and Unit Sheets and notice that the variables have not been duplicated, but the units have.

Use the Select command (=) to call the Rule Sheet into the top window and scroll through it. The rules have duplicated as well. Rules and units are loaded as they were saved with no check for duplication.

The values for the variables were blanked because the model you loaded had blank variable values. When you merge models, the items on the Variable Sheet in the new model are merged with those in the old model, but if any items are in conflict, such as variable values, the item in the model being loaded takes precedence.

## The Delete Command

Because we are using a simple model, the duplicated rules and units can be easily identified. You may sometimes want to keep duplicated rules and units on your sheets, but if not, you can delete them using the Delete command (/D).

The Delete command (/D) is similar to the Blank command (/B), but instead of blanking fields, the Delete command (/D) deletes entire rows on a sheet. In this example, you will delete the duplicated rules and units.

The first three equations on the Rule Sheet are the original TRAVEL equations, so it is the last three that we want to delete. With the cursor in the Rule Sheet, move the cursor to the fourth row by typing:

```
: 4 ♦
```

Type the Delete command:

```
/ D
```

The prompt is:

```
Delete: Point to last row (4r)
```

As with the Blank command (/B), you can delete one row by typing ♦ or you can move the cursor to the end of a block of rows and delete the entire block. Type:

```
: * ♦
```

When the prompt reappears, press ♦ to confirm the cursor position, and the three duplicated equations are deleted.

**Note:** When a rule is deleted on the Rule Sheet, the variables used in that rule are **not** deleted, but remain on the Variable Sheet.

Use the Select command (=) to call the Unit Sheet into the top window and use the Delete command (/D) in the same way to delete all the items on the sheet from row 8 through the last row on the sheet.

Fill in the variables by loading the file you saved earlier named TVAR. Use the Load option (L) of the Storage command (/S). When the file has been loaded, your screen should look like Figure 7-5.

(8m) Multiply By:

===== UNIT SHEET =====				
From	To	Multiply By	Add Offset	
mi	km	1.609		
mi/h	km/h	1.609		
mi/gal	km/l	.4251		
gal	l	3.785		
h	min	60		
C	F	1.8	32	
\$	DM	.44		

===== VARIABLE SHEET =====					
St	Input	Name	Output	Unit	Comment
	27	mlg		mi/gal	mileage
		d	1080	mi	distance
		gas	40	gal	amount of g
	55	s		mi/h	speed
		t	19.636364	h	time
	50	cost		\$	cost of tri
	1.25	pr		\$	price of ga

Figure 7-5

## The Reset Command—Variable Values Option (V)

Below is a new problem designed to show you what happens if you overdefine a model. Before trying the new problem, reset all the variable values used in the first problem using the Variable Values option (V) of the Reset command (/R). You do not have to have the cursor in the Variable Sheet when you type the command. Type:

/RV

Confirm the command by typing **Y**.

## Overdefining a Model

Display the Rule Sheet in the top window and the Variable Sheet in the bottom.

If you travel 100 miles at 60 mi/hr using 4 gallons of gasoline, what is your mileage for 5 hours?

On the Variable Sheet, enter these values in the Input fields for the variables:

Input Value	Variable Name
100	<i>d</i>
4	<i>gas</i>
60	<i>s</i>
5	<i>t</i>

Solve the model by typing:

!

The screen should look like Figure 7-6. As you can see, although the program was able to find the value for *mlg*, something is wrong. The program produced error symbols in the Status fields for one of the rules and three of the variables, and the Solution Indicator is still on.

(5i) Input: 5

```
===== RULE SHEET =====
S Rule
- ---
  mlg=d/gas
) s=d/t
* cost=gas*pr
```

```
===== VARIABLE SHEET =====
St Input      Name      Output    Unit      Comment
-----
           mlg       25        mi/gal    mileage
) 100         d         -         mi        distance
) 4           gas      -         gal       amount of g
) 60         s         -         mi/h      speed
) 5           t         -         h         time
           cost     -         $         cost of tri
           pr        -         $         price of ga
```

Figure 7-6

Move the cursor to a variable Status field containing an error symbol. The error message is **Overdefined**, as it is for the other two. The input values are in conflict.

One problem is in the second equation, as the Status field for that equation on the Rule Sheet indicates. Move the cursor into that Status field and look at the error message. It says that the rule is **Inconsistent**. All three of the variables in that equation have input values, and these values are not consistent with the relationship defined by the equation.

To correct this error, one of the input values marked **Overdefined** should be blanked. Blank the value for  $t$  using either the Status field Blank option (B) or the Blank command (/B).

Now solve the model.

This time the model was solved, but the third equation is still unsatisfied because the model is now underdefined. To correct this, enter the input value **1.35** for the variable  $pr$ .

Now solve the model again. This time it is completely solved. The screen should look like Figure 7-7.

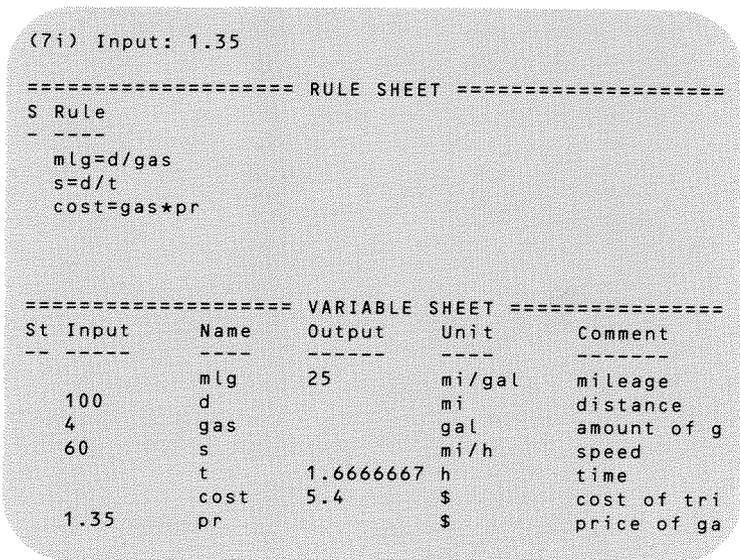


Figure 7-7

If the model contains a math error, the TK!Solver program shows error messages on the Prompt/Error line that can help you to find your mistakes and correct them.

### **Changing the Model**

TK!Solver models can be altered in a variety of ways and for many purposes. A simple model, such as TRAVEL, can be extended by adding equations. Models can also be made more complex by modifying existing equations to account for factors previously unstated.

Let's modify the TRAVEL model to calculate the energy requirements of a hypothetical car more accurately. Gas provides the energy to move a car. The amount of energy (*gas*) is presently calculated based on distance (*d*) and an assumed gas mileage (*mlg*) value. It might be more accurate to calculate gas and gas mileage as a function of the energy inherent in the gasoline. This requires another equation in the model.

### **The Insert Command**

Instead of adding the new equation to the bottom of the list of rules, we can choose to insert it at the top using the Insert command (/I). This command inserts a blank row at the row containing the cursor, moving everything at or below the cursor down one row.

Move the cursor to the top of the Rule Sheet and type:

```
/I
```

A blank row is inserted at the top of the Rule Sheet.

Enter the new rule with the variable *power* representing the energy inherent in gasoline:

```
gas=t*power
```

The new variable *power* should be on the Variable Sheet. Give it the Calculation unit **hp** (horsepower) and the comment **energy component**.

Move the cursor to the first blank row after the last equation on the Rule Sheet, and enter the equation shown below. It defines

*power* as a function of mileage (*mlg*). Later in this chapter you will edit this equation to compute power based on drag considerations.

$$d/(mlg*t)=power$$

Because you have added a new rule that defines *gas* using the new concept of power, the value for *gas* will be derived when the model is solved. You must, therefore, blank the input value for *gas* on the Variable Sheet; otherwise the model is overdefined.

When you have blanked the value for *gas*, solve the model with the value 225 hp for *power* and the values that are there from the last problem ( $d=100$ ,  $s=60$ ,  $pr=1.35$ ). The screen should look like Figure 7-8.

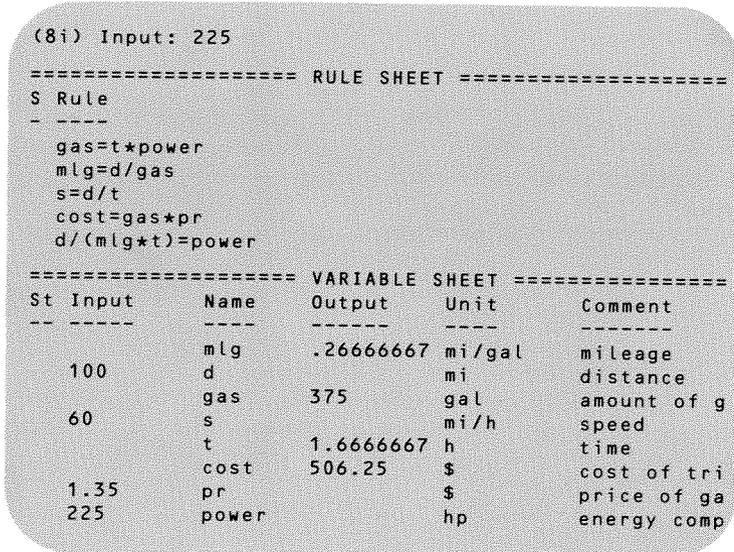


Figure 7-8

As you can see, the model shows that at a speed of 60 mph you would get mileage of approximately .267 mi/gal. Change the speed to 100 and solve, then to 45 and solve. Obviously, the new concept of power requires a more refined equation than the current one.

As the solutions above showed, the equation for power must be made more accurate. Increased accuracy is obtained by determining drag characteristics, which are unique for every car. To increase the accuracy of the model, you can alter the equation to mimic the behavior of a hypothetical car.

### The Edit Field Command

If you want to change an item in an entry field, you can use the Editor to do this. The Edit Field command (/E) calls the Editor to the field containing the cursor.

**Note:** If you are editing a value, the Editor gives you access to the full accuracy of 12 digits.

First, move the cursor to the field containing the rule  $d/(mlg*t) = power$  and call the Editor by typing:

```
/E
```

The message on the Prompt/Error line tells you that the Editor is in use:

```
( E d i t )
```

The cue appears in the field, just as it appears when you are typing information, but its behavior is slightly different when you are using the Editor.

The Arrow keys, instead of entering what you have typed and moving the cursor, only move the cue.  $\Rightarrow$  and  $\Leftarrow$  move the cue one character to the right and left.  $\Updownarrow$  moves the cue to the beginning of the field, and  $\Downarrow$  moves it to the end of the item.

Move the cue around with the arrows and notice that it seems to create a blank space within the field. This blank space is actually not a space at all but marks a position between characters. If you type a character, it appears to the left of the cue.  $\blacktriangleleft$  deletes the character to the left of the cue.

The equation given below is a model for the power used by a hypothetical car driven at a given speed. It has been derived using two assumptions:

- The amount of internal friction, represented by the variable  $f$ , is constant.
- Wind resistance, represented by the variable  $w$ , is a cubic function of the speed of the car.

With thought, an entire class of models could be built around this equation, expanding the TRAVEL model to deal with a wide variety of situations.

Use the Arrow keys and ◀ to edit the equation so that it looks like the one below.

$$f + (w * s^3) = power$$

Enter the new equation with ♦.

**Note:** If you attempt to enter an equation that contains an error, whether you are entering it for the first time or from the Editor, there is an error signal, and an error message appears on the Prompt/Error line. The program calls the Editor and the cue is moved to the point where the error was detected so that it can be corrected. You will stay in the Editor until you either enter a valid rule or press ⊗.

Move the cursor into the Variable Sheet and down until the sheet scrolls up and you can see the two new variables that have been added. The values we will give these variables are hypothetical, but they could be obtained experimentally.

Give  $f$  the Calculation unit **hp** and  $w$  the Calculation unit **hp\* $h^3$ /mi $^3$** . Comment  $f$  as **internal friction** and  $w$  as **wind resistance**. For this example, enter the input values of **.75** for  $f$  and **8.5E-6** for  $w$ .

**Note:** The number entered in scientific notation is evaluated.

Because you have edited the equation to determine power and entered values for the variables, blank the input value for  $power$  using the Blank option (B) in the Status field or the Blank command (/B).

Move the cursor to the top of the sheet so that you will be able to see the results when they appear, and solve the model. The screen should look like Figure 7-9.

(1i) Input:

```

===== RULE SHEET =====
S Rule
-----
gas=t*power
mlg=d/gas
s=d/t
cost=gas*pr
f+(w*s^3)=power

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
              mlg       29.516665   mi/gal    mileage
100           d         29.516665   mi        distance
              gas       3.3879167   gal       amount of g
45           s         45          mi/h      speed
              t         2.2222222   h         time
              cost     4.5736875   $         cost of tri
1.35         pr        1.35        $         price of ga
              power    1.5245625   hp        energy comp

```

Figure 7-9

If you try solving the model with different values for  $s$ , you will discover that the model now determines mileage much more accurately. The added equations made the model larger and improved its performance considerably.

Save the new model using the filename `TRAVEL2`. You now have two versions of the `TRAVEL` model, the simple one in the file `TRAVEL`, and the more complex one in the file `TRAVEL2`.

**Note:**  is the Edit command and is another way of calling the Editor. It can be used to modify existing field contents, just as the Edit Field command (`/E`) is, but you can use it anywhere, including on items that have not yet been entered. For example,  allows you to change a filename on the Prompt/Error line before it is entered. You will use  later in this manual.

## Conclusion

This chapter has shown you how to enter values and how to solve and modify a model. The next chapter discusses units and the Unit Sheet.

## Chapter 8 Units

- Changing the Display unit
- Conversion levels
- The Move command (/M)
- The Copy command (/C)
- The Solve command (/!)
- Saving the Unit Sheet
- The Storage command (/S)—Save Units option (U)

The TK!Solver program can convert values between units. In order to do these conversions, the program refers to the Unit Sheet on which conversions between specific units are defined.

This chapter shows you how to set up the Unit Sheet so that it has the widest possible application within the program.

---

### To begin this chapter:

At the end of Chapter 7, the Variable and Rule Sheets for TRAVEL2 were displayed.

If you quit the program at the end of Chapter 7, load the TK!Solver program, then load the file TRAVEL2.

Display the Variable Sheet in a single window so that you can see the whole sheet. The screen should look like Figure 8-1.

### Changing the Display Unit

When you entered the units on the Variable Sheet, you entered only one unit for each variable, so the unit that you see displayed for each variable is both the Display unit and the Calculation unit. If you want to check this, use the Dive command (>) to look at the Variable subsheet for each variable, and then return to the Variable Sheet with the Return command (<).

The unit conversions defined for this model are on the Unit Sheet. Use the Window command (/W) to display it in the bottom window.

On the Variable Sheet, the unit for *mlg* is shown as **mi/gal**. On the Unit Sheet a conversion has been defined for mi/gal and km/l, so those units can be converted from one to the other.

(1) Input:

```

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
              mlg       29.516665   mi/gal    mileage
100           d         29.516665   mi        distance
              gas       3.3879167   gal       amount of g
45           s         3.3879167   mi/h      speed
              t         2.2222222   h        time
              cost    4.5736875   $        cost of tri
1.35         pr         4.5736875   $        price of ga
              power   1.5245625   hp       energy comp
.75          f         1.5245625   hp       internal fr
.0000085    w         1.5245625   hp*h^3/mi wind resist

```

Figure 8-1

Move the cursor into the Unit field for *mlg* on the Variable Sheet and type:

```
km / l ♦
```

The value for *mlg* is converted from mi/gal to km/l according to the defined conversion. Now use the Dive command (>) to look at the subsheet for *mlg*. The Display unit has been changed to km/l, but the Calculation unit remains as mi/gal.

Use the Return command (<) to display the Variable Sheet. Other units and values on the Variable Sheet can be changed according to conversions defined on the Unit Sheet. Change the unit for *d* from mi to km, for *t* from h to min, and for *cost* from \$ to DM.

If a unit conversion has not been defined on the Unit Sheet, the program cannot make the conversion, although it tries. If you enter a unit with no conversion defined on the Unit Sheet, a question mark appears beside the value, and the value displayed is in the Calculation unit.

For example, there is no conversion defined between mi and ft. Change the Display unit for *d* to ft. The unit is changed when you enter it, but the value now has a question mark beside it. This tells

you that the program could not make the conversion. The value displayed is the conversion to mi, the Calculation unit for *d*. Leave the unit as it is. This will be used later in an example.

## Conversion Levels

The TK!Solver program uses the Calculation unit as a base for conversions. It converts units directly and, to a limited extent, indirectly, using the conversions defined on the Unit Sheet. Some experimentation with the Unit Sheet will illustrate how this works.

Move the cursor into the Unit Sheet and use the Window command (/W) to display it in one window. In this example, you will define a set of linear units of measurement. So that you can more easily see what is happening in the example, you will enter this set of conversion definitions in a place on the sheet separate from those that are already there.

## The Move Command

The Move command (/M) moves the row currently containing the cursor to a specified row, and changes the locations of the other rows appropriately.

There is one direct conversion for linear measurement already on the sheet, the conversion from mi to km. Move the cursor to the row containing that conversion, then move the row to a new location on the Unit Sheet by typing:

```
/M
```

The prompt is:

```
Move: Point to destination (1m):
```

Use the cursor to point to the place you want the current row to be. The number and letter in parentheses changes according to the cursor position while the Position Indicator shows the original cursor position. As with the Blank (/B) and Delete (/D) commands, you can use the Goto command (: ) to point to the destination. Type:

```
: * ◆
```

Now complete the Move command by pressing ◆. The screen should look like Figure 8-2.

(7m) Multiply By: 1.609

===== UNIT SHEET =====			
From	To	Multiply By	Add Offset
-----			
mi/h	km/h	1.609	
mi/gal	km/l	.4251	
gal	l	3.785	
h	min	60	
C	F	1.8	32
\$	DM	.44	
mi	km	1.609	

Figure 8-2

Use the Insert command (/I) to insert a blank row between the row you just moved and the one above it. This will visually separate your new set of conversions from the existing set.

Enter three more conversion definitions in the three rows below the one defining mi to km:

From	To	Multiply By
mi	ft	5280
ft	yd	1/3
yd	in	36

The screen should look like Figure 8-3.

Notice that when you entered 1/3 in the Multiply By field, it was evaluated. The Multiply By column on the Unit Sheet can evaluate expressions, like the Input column on the Variable Sheet.

Select (=) the Variable Sheet and notice that the value for  $d$  has been correctly converted to ft. Change the unit for  $d$  to **mi** again. The program can now correctly convert ft to mi and mi to ft.

(11m) Multiply By: 36

===== UNIT SHEET =====			
From	To	Multiply By	Add Offset
---	---	-----	-----
mi/h	km/h	1.609	
mi/gal	km/l	.4251	
gal	l	3.785	
h	min	60	
c	F	1.8	32
s	DM	.44	
mi	km	1.609	
mi	ft	5280	
ft	yd	.3333333333	
yd	in	36	

Figure 8-3

Change the unit for  $d$  to **yd**. Again the conversion is correctly made. Using the conversions defined on the Unit Sheet, the program can make the conversion from mi to ft and ft to yd. This is not a direct conversion from mi to yd, but an indirect conversion made through the unit ft in two steps.

Now change the unit for  $d$  to **in**. As you can see by looking at the value for  $d$ , this time the program was unable to make the conversion. This indirect conversion required three steps to convert from the Calculation unit to the Display unit—mi to ft, ft to yd, then yd to in. The TK!Solver program can make indirect conversions through two steps and no more, and it was unable to make this conversion.

To avoid problems of this sort, define all similar unit conversions either from a common unit or to a common unit. This prevents conversions that require more than two steps.

For example, if all four of the conversions for linear units of measurement were either from mi or to ft, any one of the units could have been converted directly or indirectly to any of the others in

no more than two steps. Then, if a new unit is added, such as m (meter), there is no confusion about which unit to use to define the conversion.

Display the Unit Sheet in the window. Following the directions below, you will use ft as the common unit and set up a new set of conversions for linear measurements, adding a conversion for m (meters).

First, use the Delete command (/D) to delete the conversions for mi to km and ft to yd. The conversions remaining are the ones for mi to ft and yd to in.

There eventually will be five conversions in this set, for mi, yd, in, km, and m. All will be defined according to their conversion to ft.

### **The Copy Command**

The To field will be the same for all the conversions you are about to define. Using the Copy command (/C) you can copy the item “ft” into the other five To fields you are going to use. The Copy command (/C) copies the item in the field currently containing the cursor to a specified field.

**Note:** The Copy command (/C) is valid only on entry fields, but not on all entry fields. For example, you cannot copy a variable name in a Name field on the Variable Sheet to another field in that column because you cannot define two variables to have the same name.

In this example, you will copy the To field item **ft** five times. Move the cursor to the To field containing the unit **ft** and type:

```
/C
```

The prompt is:

```
Copy: Point to destination (8t)
```

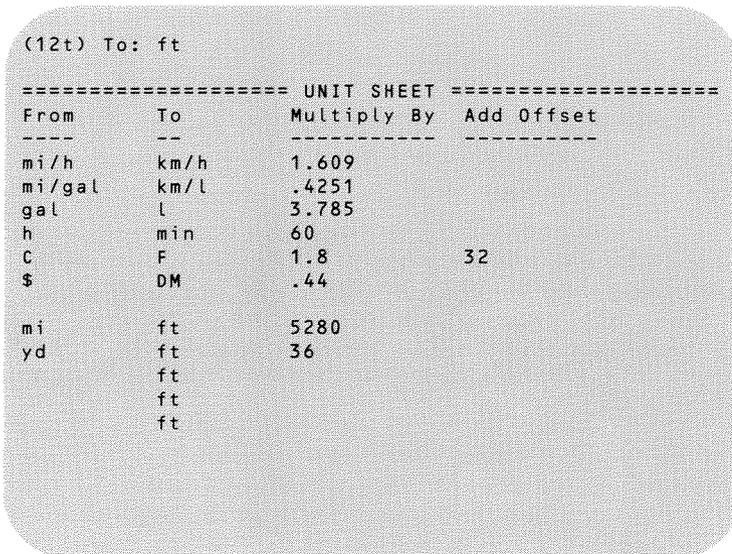
As with the Move (/M), Delete (/D), and Blank (/B) commands, you are asked to use the cursor to point to the location to which you want the field contents copied. The indicator in parentheses changes, as it did for the Move command (/M), showing the current cursor position.

Because the unit in the To field in the next row should be changed to ft, move the cursor down into the next To field and press  $\diamond$ . The new item **ft** replaced the old item **in**.

Do this three more times to fill in the To field for the rest of the conversions:

/C  $\diamond$   $\diamond$

The screen should look like Figure 8-4.



(12t) To: ft

From	To	Multiply By	Add Offset
mi/h	km/h	1.609	
mi/gal	km/l	.4251	
gal	l	3.785	
h	min	60	
C	F	1.8	32
\$	DM	.44	
mi	ft	5280	
yd	ft	36	
	ft		
	ft		

Figure 8-4

**Note:** The Copy command (/C) can be used with the Switch command (;) to copy fields between windows. It can also be used with the Goto command (:).

Finish entering the conversions to reflect Table 8-1. When you enter the expressions shown here, the program will evaluate them.

From	To	Multiply By
mi	ft	5280
yd	ft	3
in	ft	1/12
m	ft	1/.3048
km	ft	1/.3048*1000

Table 8-1

Split the screen with the Window command (/W), and display the Variable Sheet in the bottom window. Change the unit for  $d$  to **m**, to **yd**, to **km**, to **in**. The program is able to make all of the conversions.

Reset the values on the Variable sheet with the Variable Values option (V) of the Reset command (/R), so that you can enter the input values for a new problem.

**Note:** Remember to check the Display unit **before** entering the value. Values are assumed to be in the Display unit, so you may need to change the Display unit before you enter the value.

With internal friction of .5 Btu/s and wind resistance of .0000106 hp\*h<sup>3</sup>/mi<sup>3</sup>, what is your mileage if you travel 500 miles at 60 miles per hour?

The Unit Sheet contains no conversion for hp to Btu/s, so you must enter the conversion definition on the Unit Sheet before you enter the value for  $f$ . Move the cursor into the Unit Sheet and enter the conversion at the bottom:

From	To	Multiply By
hp	Btu/s	.706243

Save TRAVEL2 again with the new unit conversions you have added.

Now enter the new Display units for  $f$  and  $mi$ , and the values for all the input variables in the problem shown above. This time, instead of using the Action command (!) to solve the model, try the Solve command (/!).

## The Solve Command

The Solve command (/!) calls the solver to attempt solution of the model from any sheet. It does exactly what the Action command does when the cursor is on the Variable or Rule Sheet, but the Solve command (/!) does it from anywhere in the program.

If you wanted to solve the model now with the cursor on the Unit Sheet, you would find that the Action command (!) gives an error signal. Try it. Move the cursor into the Unit Sheet and type !.

The Action command (!) performs a variety of actions depending on which sheet the cursor is in, but it solves the model only when the cursor is in either the Variable or Rule Sheet.

Solve the model now by using the Solve command. Type:

```
/!
```

The model is solved; the screen should look like Figure 8-5.

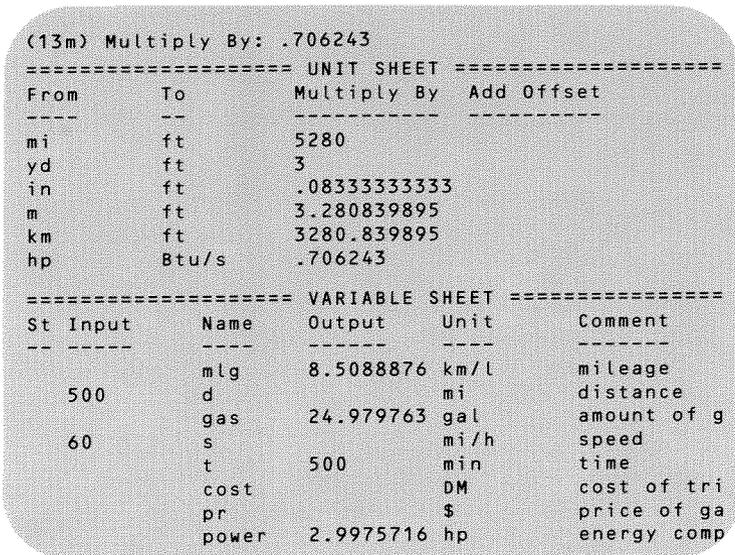


Figure 8-5

If your output values are significantly different from those shown, check that the units on the Variable Sheet are the same as those given in the problem.

**Note:** The Solve command (/!) works on the Variable and Rule Sheets as well as all the others.

Reset the program with the All option (A) of the Reset command (/R).

### Saving the Unit Sheet

When you were building the model TRAVEL in Chapter 6, you loaded a file named TUNITS containing the Unit Sheet for the TRAVEL model. The Save Units option (U) of the Storage command (/S) saves a file that contains only the information on the Unit Sheet.

In this section you will create and save a file of unit conversions to be used in a later chapter for the example model MORTGAGE.

### Creating the Unit Sheet

Display the Unit Sheet in either window, and enter the three conversion definitions needed for the model MORTGAGE:

From	To	Multiply By
years	months	12
percent	monthly rate	$1/12 * 1/100$
percent	decimal__form	.01

As you can see, the names for these units are not the ones used with TRAVEL. The name of the unit is up to you, and it may contain blanks, as does monthly rate, underscores, as does decimal\_\_form, and any other printable characters. Blanks at the beginning and end of a unit name are deleted when the name is entered.

You can define any unit according to any conversion you choose. The accuracy of the conversion is up to you, within the TK!Solver program limitation of 12 significant digits.

### The Storage Command—Save Units Option (U)

The Save Units option (U) of the Storage command (/S) saves a file containing only the information on the Unit Sheet, ignoring all other information. The Unit Sheet does not have to be on the screen either to save or load the unit file.

Save a file of the Unit Sheet you have just created by typing:

/SU

At the prompt, enter the filename `MUNITS`, for mortgage units. These units are now ready for the `MORTGAGE` model.

### **Conclusion**

This chapter has discussed the Unit Sheet, how to set it up, and how it relates to the Variable Sheet. The next chapter discusses the Global Sheet and how to use your printer with the TK!Solver program.

## **Chapter 9 Global Setting and Printing**

- The Global Sheet
- The Printer fields
- The Print command (/P)

The Global Sheet is used for specifications that affect the model as a whole. The fields on the Global Sheet are divided into three sections, according to their use. This chapter describes the Global Sheet and how to use the group of fields on it that apply to your printer.

Once you have set this group of fields to your requirements, you can use the Print command (/P) to print all or part of the contents of a sheet. You can use the Print command (/P) with a printer, or you can use it to write print files, files that are formatted for printing.

If you do not have a printer, you may want to read only the first section of this chapter, about the Global Sheet, and skip the rest.

---

### **To begin this chapter:**

Use the Reset command (/R) to reset the program, and load the TRAVEL2 model.

Display one window containing the Global Sheet by typing:

```
/W1  
=G
```

The screen should look like Figure 9-1.

### **The Global Sheet**

The fields on the Global Sheet are divided into three sections. The first contains only one field, the Variable Insert ON field discussed in Chapter 6. The second section is a group of five fields concerning the Solvers. These fields and their use are discussed in Part III. The third group of fields is associated with the printer.

```
(v) Variable Insert ON: Yes

===== GLOBAL SHEET =====
Variable Insert ON:          Yes
Intermediate Redisplay ON:  Yes
Automatic Iteration ON:     Yes
Comparison Tolerance:       .000001
Typical Value:              1
Maximum Iteration Count:    10

Page Breaks ON:             Yes
Page Numbers ON:            Yes
Form Length:                66
Printed Page Length:        60
Printed Page Width:         80
Left Margin:                 0
Printer Device or Filename: PRN
Printer Setup String:
Line End (CR/LF or CR):    CR/LF
```

Figure 9-1

All of the fields on the Global Sheet have default values, but you can change any of them to suit the needs of your model and the requirements of your system. When you save your model, the Global Sheet is saved with any changes you have made. If you reset the Global Sheet with the Reset command (/R), the Global Sheet is reset with the default values.

For example, change the first field to **No** by typing **N** in that field. Change the next two fields from **Yes** to **No** in the same way. Now type the Reset command (/R) using the Sheet option (S):

```
/RS
```

Type **Y** to confirm the command.

Each of the three fields returns to **Yes**, the default value.

### **The Printer Fields**

The Print command (/P) either prints a specified portion of the current sheet on a printer or writes it in a print file on a diskette.

If you do not have a printer, you may want to skip the rest of this chapter and go on to Part III. If you do not have a printer and would like to try the Print command (/P) anyway, you can use these instructions to create a print file and then look at the contents of the file by using the appropriate system command, shown in Chapter 2.

If your system and your printer have constraints or requirements (such as the use of control characters to specify the printing format) you must allow for them when using the printer with the TK!Solver program. See the user manual for your printer to determine if there are constraints and what they are.

Before trying out an example of the Print command (/P), you should look through the group of Printer fields on the Global Sheet to see that they are set up correctly for your printer and the paper you are using.

#### **Page Breaks ON**

The first field in the group of Printer fields is labeled **Page Breaks ON** and is an option field with the options of **Yes** or **No**. The default is **Yes**.

If this field is **Yes**, the printer breaks the text at the bottom of a page of paper and starts again at the top of the next page, skipping one line at the top of the page and one at the bottom. This causes the printer to skip over the perforations in continuous feed paper.

If this field is **No**, the printer prints continuous text with no breaks. The program assumes that form and page length are the same, and so ignores the next three fields.

#### **Page Number ON**

The Page Number ON field is another option field with the options of **Yes** or **No** and has a default value of **Yes**. If this field is **Yes**, the page number is printed at the bottom of the page with one line space separating it from the preceding text. If the field is **No**, page numbers are not printed. If page breaks are not used, page numbers are not printed.

## Form Length

The Form Length field tells the program the length of the paper and is counted by lines. The default is **66** lines, the length of standard letter size 8-1/2 x 11 paper. This field requires an integer.

## Printed Page Length

The Printed Page Length field tells the program the length for the printed text, including the page number, if that option is used. The default is **60** which leaves six lines for top and bottom margins on standard letter size 8-1/2 x 11 paper. These six lines are divided between top and bottom margin according to how you initially line up the paper in the printer. This field requires an integer.

## Printed Page Width

The Printed Page Width field tells the program the width of the printed text on the page and is counted by characters. The default is according to your system. Standard letter size 8-1/2 x 11 paper is 80 characters wide. This field requires an integer.

## Left Margin

The Left Margin field tells the program the width of the left margin counted by characters. The default is **0**. The width of the right margin can be determined by adding this width to the printed page width and subtracting that number from the form width.

For example, if your form is 80 characters wide, and you set the Printed Page Width to 65 and the Left Margin to 10, the right margin will be 5 characters wide. This field requires an integer.

## Printer Device or Filename

The Printer Device or Filename field tells the program whether to print the text on the printer or in a print file. The default is the printer.

The characters used to designate the printer are specific to your system and are the default for this field.

If you are writing a file, enter the filename in this field, including any specifications necessary. Filenames must follow the rules for your system. The program automatically adds an extension to the filename, identifying the file as a print file. Refer to Chapter 2 for the extension used for your system.

### Printer Setup String

The Printer Setup String field is specific to your printer. There is no default value because each printer has its own specifications, and you may not need a setup string at all. Look in the Hardware Reference or consult your printer manual to find the setup string for your printer.

Characters that don't print, such as the Escape character, are called control characters. They send special commands to the system. If your printer's setup string requires control characters, a caret (^) tells the program that the next character represents a control character. Table 9-1 shows how to enter some of them. Refer to your printer manual for any others that you need.

- ^C = The next character is a control character.
- ^E = This is an escape character.
- ^H = The next two characters are hex digits.
- ^^ = This character is a caret.

#### Table 9-1

If you enter characters that are not interpreted by your printer as control characters, the program will print them at the top of the page, but no special allowance is made for them in the Form Length and Printed Page Length settings.

### Line End

The Line End field is specific to your system. Some systems require different settings for a print file than for a printer.

This is an option field. The default for this field is the same line termination as the one used by your system's editor. If your system requires you to change the termination specifications for different types of printing, the TK!Solver program requires the same changes. Consult your system and printer manuals.

The options are:

Option:	Type:
Carriage Return & Line Feed	&
Carriage Return	C
Line Feed	L
System-defined	S

**Note:** If the default for your system is S, you cannot change the option.

### The Print Command

When you have checked all the print fields on the Global Sheet, you are ready to use the Print command (/P).

The Print command (/P) either prints a specified portion of the current sheet on a printer or writes it in a file on a diskette, depending on what is specified in the Printer Device or Filename field on the Global Sheet. If it is copied in a file, the program gives the filename an extension that identifies it as a print file. The extension used for your system is given in Chapter 2.

You must specify the block of text that you want to print. The Print Command prints the rows specified with the column labels.

Connect your printer and turn it on. When you line up the paper, set it so that the print head rests on the line where you want the printing to begin. The printer prints one page length, according to the setting on the Global Sheet, then skips the number of lines that are the difference between the form length and page length. This gives you a margin at the bottom of the first page and top of the second; the lengths of the top and bottom margins depend on the difference between the form and page length and where you set the print head at the beginning.

Suppose you want to print the first six variables on the Variable Sheet. Use the Select command (=) to display the Variable Sheet for TRAVEL2. Position the cursor on the first row that you want printed, in this example the first row on the sheet, and type:

```
/P
```

The prompt is:

```
Print: Point to last line (1i)
```

The indicator in parentheses changes as you move the cursor, as it does for the Move (/M) and Copy (/C) commands. Move the cursor to the last row that you want printed. In this example use the Goto command (: ) to point to row 6. Type:

```
:6 ♦
```

Now finish the Print command by pressing ♦. As the printer prints the text, the screen shows you what is being printed. When it is finished, the previous display returns.

You can point to the portions of the sheet you want to print from bottom to top instead of top to bottom. The Print command (/P) always prints the rows from top to bottom, and always prints the column labels.

To display the contents of a print file on your screen, see the instructions for your system in Chapter 2.

**Conclusion**

This chapter has discussed the Global Sheet and shown you how to use the printer and the Print command (/P). This is the end of Part II. Part III explains the two methods of solution used by the TK!Solver program to solve problems.



## **Part III The Solvers**

The TK!Solver program uses two methods to solve problems, the Direct method and the Iterative method. The method you use depends on the problem you want to solve. Each solver uses the given input values to find all the variables possible within one solution of the problem.

The TK!Solver program can solve most systems of algebraic equations, although its ability to do matrix and numerical techniques for integration is limited, and it cannot do non-numeric calculus.

The program always attempts a solution using the Direct Solver first. If the problem is such that a Direct solution is not possible, you can set a guess value for one or more variables and use the Iterative Solver to attempt a solution.

Throughout Part II you used the Direct Solver to solve the models TRAVEL and TRAVEL2, although how this method works was not explained. Part III discusses Direct and Iterative methods of solution, what each one is, and how they work. Chapter 12 also shows you how to use the Iterative Solver with the TK!Solver program and the conditions under which its use is necessary to solve a problem.

---

Contents of Part III:

- Chapter 10 The Direct Solver
- Chapter 11 Iterative Solutions
- Chapter 12 Using TK!Solver Iteration

## Chapter 10 The Direct Solver

- How the Direct Solver works
- A single equation
- A set of equations
- The Direct Solver and the TK!Solver program
- Changing an equation to avoid iteration
- Models requiring iteration

The Direct method of solution solves models by substituting known values to determine unknown values. This substitution enables the Direct Solver to solve for a variable regardless of its position in the equation. The Iterative Solver incorporates the Direct method of solution.

This chapter is about the Direct Solver and how it works.

---

### To begin this chapter:

Reset the program with the Reset command (/R).

### How the Direct Solver Works

The Direct Solver solves a model containing several equations by the same method it solves a model containing a single equation. A model containing a single equation must meet the following conditions to be solved by the Direct method:

- 1 All variables in the equation, except the one being solved for, must have input values.
- 2 Because of internal constraints, the variable being solved for must appear only once in the equation.

For example, the Direct Solver **cannot** solve for  $x$  in these equations:

$$x + x = y$$

(Direct solution is possible if written as  $2*x = y$ )

$$x * x = y$$

(Direct solution is possible if written as  $x^2 = y$ )

- 3 The variable being solved for cannot be the argument of a function without a unique inverse.

For example, the Direct Solver **cannot** solve for  $x$  in these equations:

$$\text{int}(x) = y$$

$$\text{abs}(x) = y$$

$$\text{sgn}(x) = y$$

### A Single Equation

Using these conditions, look at the equation:

$$a + b = c$$

Condition 2 has been met because none of the variables appears twice in the equation. And because the equation contains no function at all, condition 3 is not applicable.

To meet condition 1, two of the three variables must be given input values. If, for example, the values of  $a$  and  $c$  are known, this model can be solved for the value of  $b$ .

### A Set of Equations

For the Direct Solver to solve a set of equations, each equation in the set must meet the above conditions sometime during the solution process. As each equation is satisfied, the output values are used as input values for other unsatisfied equations in the model.

Example 1 is a model containing three equations. This model will be used to demonstrate how the Direct Solver works. Before entering this model on the TK!Solver sheets, let's examine how the Direct Solver would solve this model for two input values.

Example 1

$$a + b = c \quad \text{Equation 1}$$

$$d - e = b \quad \text{Equation 2}$$

$$d = (a + b)/2 \quad \text{Equation 3}$$

If the value of  $a$  is 13 and of  $c$  is 4, then in:

Equation 1, the values for  $a$  (13) and  $c$  (4) are known.

Equation 2, no values are known.

Equation 3, only the value for  $a$  (13) is known.

Equation 1 meets all three conditions for solution by the Direct Solver, so it can be solved for  $b$  (-9).

When Equation 1 is solved and the value for  $b$  (-9) is known, Equation 3 meets the conditions for Direct solution, so it can be solved for  $d$  (2).

When Equations 1 and 3 are solved, the values for  $b$  (-9) and  $d$  (2) are known. Then Equation 2 meets the conditions for Direct solution and can be solved for  $e$  (11).

The model can be completely solved by the Direct Solver if input values are given for  $a$  and  $c$ .

## The Direct Solver and the TK!Solver Program

Now try Example 1 using the TK!Solver program. Enter the three equations on the Rule Sheet, and then enter the value 13 for  $a$  and 4 for  $c$ .

While the model is being solved, notice the message on the Prompt/Error line that tells you the Direct Solver is working. Solve the model by typing the Action command (!) or the Solve command (/!). The screen should look like Figure 10-1.

```
(3i) Input: 4

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
    13         a          -9
           b
    4          c          2
           d
           e          11

===== RULE SHEET =====
S Rule
-----
a+b=c
d-e=b
d=(a+b)/2
```

Figure 10-1

What would the Direct Solver do if one of the equations failed to meet the conditions for Direct solution? To try this, edit Equation 3 so that it looks like this:

$$d = (a + b) / (2 * d)$$

Solve the model.

The asterisks in the Status fields for Equations 2 and 3 tell you that these equations are unsatisfied. The Direct Solver solved Equation 1, but neither Equation 2 or 3 satisfied the three conditions necessary for Direct solving. Equation 3 could not be solved because the unknown variable  $d$  occurs twice, violating the second condition for Direct solution, and Equation 2 failed because it contained two variables without known values, violating the first condition.

Use the Reset command (/R) to reset the values for this model, then save it in a file named DTWO. It will be used as an example in Chapter 12.

### Changing an Equation to Avoid Iteration

Equations that cannot be solved by the Direct method of solution can often be solved using the Iterative method, but the need for iteration can frequently be avoided if you rewrite the equations so that they meet the three conditions. There are many ways to reformulate models.

For example, in this problem you can rewrite Equation 3 so that the Direct Solver can solve the model for some sets of known values.

One way to rewrite the equation is to use the built-in square root function. This function has the syntax:

$$SQRT(x)$$

The equation below shows the same relationship between the variables as Equation 3, but because it uses the variable  $d$  only once, it meets the second condition for Direct solution, the conditions that it previously violated. Edit Equation 3 so that it looks like this:

$$d = SQRT((a + b) / 2)$$

**Note:** Names of the built-in functions can be typed in either uppercase or lowercase characters, or a combination.

**Note:** The TK!Solver program always returns a positive square root, so the third condition, requiring a function to have an inverse, has also been met.

Give  $a$  a value of 13 and  $c$  a value of 4 and solve the model. The screen should look like Figure 10-2.

```
(3i) Input: 4
===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
   13         a         -9
              b
   4          c         1.4142136
              d
              e         10.414214
=====
S Rule
-----
a+b=c
d-e=b
d=SQRT((a+b)/2)
```

**Figure 10-2**

Another way to change an equation is to use a factored form. For example, if the original equation is in this form:

$$d + b*d = a$$

it could be changed to:

$$d*(1 + b) = a$$

The equation now contains the variable  $d$  only once.

Other techniques for reformulating models are discussed in Chapter 12.

**Models Requiring Iteration**

In any model, the Direct Solver solves all equations that meet the three conditions for Direct solution. The modified version of Example 1 is a model that cannot be completely solved using the Direct Solver. If the model contains an equation that cannot be solved by the Direct Solver, the asterisk remains in the Status field of that equation after the Direct Solver has stopped.

Below are three examples of models that require Iterative solutions. Each example lists the known (input) and unknown (output) variables and the reason the model requires iteration.

**Example 2:**

$$-8 = 3*a - 2*b + c - d + 4*e$$

$$-27 = 2*a - b - c + 3*d + 5*e$$

$$15 = a + b + c - 2*d - 2*e$$

$$-2 = 3*a + 3*b - c - d + 2*e$$

$$13 = a + 2*b + c - 4*d + 3*e$$

Known Variables: None

Unknown Variables: All variables

Need for Iteration: Not enough input values.

(Example 2 is used again in Chapter 12 and is stored in the Instruction File LINEAR.)

**Example 3:**

$$y = x^2 - 2*x - 5$$

Known Variables:  $y$

Unknown Variables:  $x$

Need for Iteration: Variable being solved for occurs more than once.

### Example 4:

$$b = \log(a)^2 - \log(a)$$

$$a = \sin(b)^3$$

Known Variables: None

Unknown Variables: All variables

Need for Iteration: Not enough input values and variable being solved for occurs more than once.

### Conclusion

This chapter has shown you how the Direct Solver works. The next chapter discusses the concept of iteration. If you are familiar with this concept you may want to go on to Chapter 12.

## Chapter 11 Iterative Solutions

- The concept of iteration
- The divide and average method
- The first guess
- Watching iteration work

As was shown in the last chapter, some equations cannot be solved using the Direct method of solution. Often these equations can be solved using an Iterative method of solution. The TK!Solver program has an Iterative Solver that can be used to attempt a solution when the Direct Solver is unable to solve the model. To use the Iterative Solver to its best advantage, you should first understand what an Iterative solution is.

This chapter is about the concept of iteration and how it can be used to solve problems. If you know what iteration is, you may want to skip this chapter and go on to Chapter 12, which explains how to use the program's Iterative Solver.

---

### To begin this chapter:

Reset the program using the Reset command (/R).

### The Concept of Iteration

Iteration is a problem solving procedure that uses successive approximations to find an answer. The procedure repeats a set of operations, each repetition more closely approximating the answer.

You have probably used an Iterative method before, but never called it by that name. The game of Twenty Questions follows an Iterative procedure. You begin with a guess, and each guess gains you some information that improves the next guess. Eventually this guessing leads you to narrow in or converge on an answer. All Iterative methods, no matter how they are defined, follow this same procedure.

In mathematics, the first step in an Iterative procedure is to guess the answer to the problem. By manipulating this guess, a new guess is generated that more closely approximates the answer. This process is repeated until a solution is reached.

There are many different methods of mathematical iteration, and they differ primarily in the way the guess is manipulated to find the next approximation of the answer. Some methods are more generally applicable while others have only restricted use. The method used by the TK!Solver program is widely applicable.

The process of making a guess, manipulating it mathematically, and generating a new guess constitutes one step or iteration. The number of steps needed to solve a problem varies greatly, and depends on the particular Iterative method being used and on how close the first guess is to the correct answer. Sometimes, no matter how many steps are executed, an Iterative process does not converge to a solution.

The procedure may be unable to converge to an answer for one of several reasons. The most common reasons are that the system of equations does not contain a solution that satisfies all the equations simultaneously or that the first guess was poor.

### **The Divide and Average Method**

The following example illustrates one Iterative method and shows how iteration works, each step moving closer to the answer. This particular Iterative method, known as the Divide and Average method, was developed by Heron of Alexandria in the first century A.D. It is used to determine square roots.

Each iteration of the Divide and Average method is composed of these operations:

- 1 Guess a value for the square root of a number  $n$ .
- 2 Divide the number  $n$  by the guess.
- 3 Find the average of the guess and the quotient calculated in the second step.
- 4 Use this average as a new guess.
- 5 Repeat the process until the average converges to an answer.

Using these four steps of the Divide and Average method, you can determine a square root by working through several iterations. To do this with pencil and paper requires much cumbersome long division. Instead, let's set up the problem as a TK!Solver model. Then you can see how iteration works using the Divide and Average method, without doing any of the calculations yourself.

Step 1 is to choose the first guess, and you must do this yourself. But the model you are about to build will do the rest of the problem for you.

In step 2 you divide the number you are trying to find the square root of by your guess. In the equation below,  $n$  represents the number,  $g$  the guess, and  $q$  the quotient, the result of step 2. Enter this equation on the Rule Sheet:

$$n/g=q$$

In step 3 you find the new guess,  $ng$ , by averaging  $g$  and  $q$ . On the Rule Sheet, enter the equation for step 3.

$$ng=(g+q)/2$$

The screen should look like Figure 11-1.

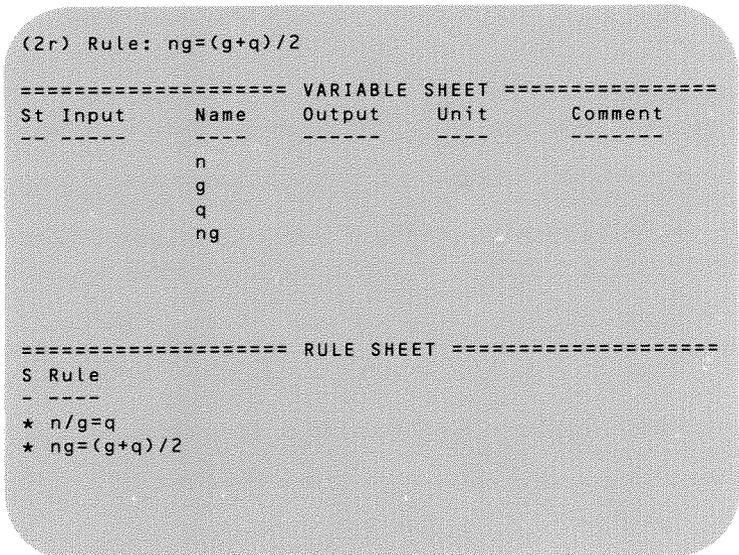


Figure 11-1

### The First Guess

The problem is to find the square root of 1296. First, decide on a first guess.

If you know the square root of a number less than 1296 and one greater than 1296, a good guess might be a value between those two. The square roots of 900 and 1600 are 30 and 40, respectively. Then, since

$$900 < 1296 < 1600$$

$$\sqrt{900} < \sqrt{1296} < \sqrt{1600}$$

$$30 < \sqrt{1296} < 40$$

The square root of 1296 is between 30 and 40. Because 35 is midway between these two known square roots, it is a good value for the first guess.

The model needs two input values, the value of the first guess and the number whose square root you are trying to find. On the Variable Sheet, enter 1296 as the input value for  $n$  and 35 as the input value for  $g$ .

The model has been set up so that each solution creates a new value for  $ng$ , which is then used as the value for  $g$  in the next iteration.

### Watching Iteration Work

With the cursor in the Input field for  $g$ , follow the instructions below:

Type:	Result:
!	Solves the model with $q = 37.028571$ and $ng = 36.014286$ .
ng ♦ !	Enters $ng$ as the new guess and solves with $q = 35.985720$ and $ng = 36.000003$ .
ng ♦ !	Enters $ng$ as the new guess and solves with $q = 35.999997$ and $ng = 36$ .
ng ♦ !	Solves the model. All values are now 36. The square root of 1296 is 36.

As you can see in Table 11-1, the consecutive values for *ng* decrease towards 36.

**ng**  
36.014286  
36.000003  
36

**Table 11-1**

This method, like most Iterative methods, converges on the answer. The TK!Solver program's Iterative Solver is much more powerful and easier to use than the Divide and Average method. The principle is the same, but the Iterative Solver uses more sophisticated methods, and the TK!Solver program does the work.

**Conclusion**

This chapter has explained the concept of the Iterative solution. The next chapter explains how the Iterative Solver works and how the TK!Solver program uses iteration.

## Chapter 12 Using TK!Solver Iteration

- Using iteration in the TK!Solver program
- Unsuccessful guesses
- Slow convergence
- Divergence and oscillation
- Changing the number of iterations
- Comparison tolerance
- Input values
- Redundant equations
- Automatic iteration

One of the Solvers used by the TK!Solver program is the Iterative Solver. When you have a problem that requires an Iterative solution, you can set one or more values as a guess and use the Iterative Solver to solve the model.

This chapter shows you how to use the Iterative Solver and explains how it works.

---

### To begin this chapter:

Reset the program and load the model named DTWO, saved on the Instruction diskette in Chapter 10. The screen should look like Figure 12-1.

### Using Iteration in the TK!Solver Program

To use the program's Iterative Solver, all you have to do is set a first guess for the desired variable, and type the Solve command (/!). The better your first guess, the more quickly the Iterative Solver can solve the model. The DTWO model that you created in Chapter 10 is used here to show you what the program looks like when you use the Iterative Solver.

This model requires iteration because the variable  $d$  appears twice in the third equation. Because it appears twice,  $d$  is the value that must be found by iteration.

Give  $a$  the value 13 and  $c$  the value 4. Use the value 12 for your first guess for  $d$ . Enter 12 in the Input field, then set it as a guess by moving the cursor to the Status field for the variable  $d$  and typing the Status field option:

```
(3r) Rule: d=(a+b)/(2*d)

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
              a
              b
              c
              d
              e

===== RULE SHEET =====
S Rule
- ----
* a+b=c
* d-e=b
* d=(a+b)/(2*d)
```

Figure 12-1

This is one of the Status field options discussed in Chapter 6. It sets the variable as a guess variable, and the input value you give that variable is used as the first guess. Notice that on the Status line the content of the Status field is spelled out as **Guess**. You can remove the Guess setting by typing the Guess option (G) again.

**Note:** If you do not enter a value in the Input field before you type the Guess option (G) in the Status field, the program will automatically put into the Input field either the last value for that variable or, if there was no previous value, a 0.

With 12 set as a first guess for  $d$ , solve the model by typing the Action command (!) or the Solve command (/!).

**Note:** The message that appears at the top of the screen during Iterative solution is explained later in this chapter.

The screen should look like Figure 12-2.

```
(4s) Status:

===== VARIABLE SHEET =====
St Input      Name      Output    Unit      Comment
-----
   13         a          -9
           b
   4          c
           d      1.4142136
           e      10.414214

===== RULE SHEET =====
S Rule
-----
a+b=c
d-e=b
d=(a+b)/(2*d)
```

Figure 12-2

## The BOX Model

Although iteration is easy to use, you must know more about the Iterative Solver and its method of solution before you can use it to the best advantage. The following example illustrates both the use of the Iterative Solver and the concepts behind its use.

### The Problem

A company produces rectangular boxes and wants to know on what basis to price them. The price can be based on either the surface area or the volume of the boxes. The values representing both the surface areas and the volumes are dependent on the length, width, and height of the boxes.

To choose the attribute on which to base box prices, you must first determine the dimensions of the box for which the values for the volume and surface area are equal. The range of box sizes the company produces based on volume is between 1 and 10. If volume increases more rapidly than surface area, the price of the boxes might be based on volume, and vice versa.

The model you are about to enter is named BOX and can solve this problem using the Iterative Solver.

Reset the program and enter Equation 1, the equation for volume, on the Rule Sheet. The volume of the box is represented by  $V$ , the length by  $l$ , the width by  $w$ , and the height by  $h$ .

$$V = l * w * h \quad \text{Equation 1}$$

The two equations below are equivalent equations for surface area. The variable  $SA$  represents surface area.

$$SA = 2 * l * w + 2 * l * h + 2 * w * h \quad \text{Equation 2}$$

$$SA = 2 * (l * w + l * h + w * h)$$

Equation 2 is shown in two forms. The second one is the factored form of the first. Using factored forms of equations often reduces the processing time needed by the TK!Solver program.

Enter Equation 2 in the factored form (the second form) on the Rule Sheet.

To solve this problem for all rectangular boxes would involve the investigation of many different possibilities. We are using this model as an example of iteration, so we will consider only the special case of cubes for which all three dimensions are equal.

Since the problem is to determine when a cube's volume and surface area are equal, you need three more equations to complete your model.

Equation 3 sets the length and width equal, and Equation 4 sets the width and height equal. These equations restrict the model to cubes. Equation 5 sets the volume and surface area equal. Enter Equations 3 through 5 on the Rule Sheet.

$$l = w \quad \text{Equation 3}$$

$$w = h \quad \text{Equation 4}$$

$$V = SA \quad \text{Equation 5}$$

The BOX model is now complete. The screen should look like Figure 12-3.

```
(5r) Rule: V=SA
===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
              V
              l
              w
              h
              SA

===== RULE SHEET =====
S Rule
-----
* V=l*w*h
* SA=2*(l*w+l*h+w*h)
* l=w
* w=h
* V=SA
```

**Figure 12-3**

Enter the value 10 for *l* and solve the model.

The program solves the model, but there is a > in the Status fields for *V*, *SA*, and the equation  $V = SA$ . Move the cursor to the Status field for  $V = SA$  and look at the message on the Prompt/Error line:

```
(5s) Status: > Inconsistent
```

The message tells you that the equation is inconsistent with the given values. The model is overdefined and was solved with different values for *V* and *SA*. They are not equal although the equation says they should be.

Often such an inconsistency can be overcome by setting one of the inconsistent variables as a guess and solving the problem iteratively. When a variable is designated as a guess, the program uses the Iterative Solver to solve a model. A variable is designated as a guess when the **G** appears in the Status field of the variable.

Since you have already entered 10 as a value for *l*, use it as your guess. In the Status field for the variable *l*, type:

```
G
```

The model is now ready for the program to attempt an Iterative solution. As the Iterative Solver works, a message on the Prompt/Error line will show the number of iterative steps that have been tried out of the approximate number of iterative steps needed to solve the problem. The estimate of the total number of iterative steps needed may change during the iteration process.

When the Solver has finished, both the G and the message will disappear, and the value for  $l$  will be moved to the Output field.

In this problem, the answer will converge to 6 after seven steps of iteration. Table 12-1 illustrates what will happen when you solve the model with a guess of 10 for  $l$ .

<b>Step</b>	<b>Value</b>
1	10
2	7.7778074
3	6.5577657
4	6.0810962
5	6.0021081
6	6.0000015
7	6

**Table 12-1**

Now solve the model and watch the values for  $l$  change as in Table 12-1.

The model was solved iteratively showing you that cubic boxes of 6 units to a side have equal surface area and volume.

Because this example has been built primarily to demonstrate iteration in the TK!Solver program, there is no further analysis of the BOX model in this manual. However, the BOX model will be used later in this chapter. Save the model using the filename `BOX`.

### **Unsuccessful Guesses**

When using iteration, you should give some thought to the guess you use because results can vary significantly depending upon the nature of the guess.

In problems that have multiple solutions, the Iterative Solver can find only one of the solutions for a given first guess. To find the other solutions to the problem, you must try different first

guesses. One method for doing this is to edit the equation so that both sides are divided by  $(x-r)$ , with  $x$  representing the variable and  $r$  representing the answer found by the first guess value.

Although there is no assured method of avoiding an unsuccessful guess, the more reasonable the guess is, the more likely you are to get the desired results.

Every guess leads either to a solution of the model or to one of three less satisfactory results. The three unsuccessful results are:

- A slow convergence.
- Iterations that diverge, moving away from the correct answer instead of towards it.
- Iterations that oscillate between two values.

## Slow Convergence

The BOX model can be used to demonstrate the first type of unsuccessful result, a slow convergence. Any number could have been used as a guess for the value of  $l$  in the BOX model, and another number might not have been as successful a guess.

For example, try to solve the model using a first guess of 4 for  $l$ . You do not have to reset the values left from the previous solution. Change the value of  $l$  to 4, and type a **G** in the Status field. While the model solves, compare the successive output values with Table 12-2, which shows the values of  $l$  for each step of iteration.

Step	Value	Step	Value
1	4	6	10.716623
2	44	7	8.2081384
3	30.066811	8	6.7724745
4	20.813598	9	6.1434953
5	14.701069	10	6.0064065

**Table 12-2**

Solve the model.

When the Iterative Solver finishes, the screen should look like Figure 12-4.

```
(2i) Input: 6.0000137581

===== VARIABLE SHEET =====
St Input      Name      Output    Unit      Comment
-----
G 6.0000138  l
                w
                h
                SA

===== RULE SHEET =====
S Rule
-----
* V=l*w*h
* SA=2*(l*w+l*h+w*h)
* l=w
* w=h
* V=SA
```

Figure 12-4

The value of  $l$  is still in the Input field, and the G is still in the Status field. The message for the Iterative Solver told you that there were 10 steps of iteration tried. After doing 10 iterative steps, the Iterative Solver stopped, although the values were converging correctly towards 6. The guess of 4 resulted in a slow convergence.

When a model fails to converge, you can type the Solve command (/!) again to restart the solving process. Solve the model again. This time the model is solved in one more iteration.

### Divergence and Oscillation

The other two types of unsuccessful guesses are easy to identify from their results. As an example, reset the program, and enter this equation:

$$-2 * x^2 + 3 * x - 2 = y$$

Give  $y$  a value of 0. Give  $x$  the value of 2 and set it as your first guess.

Solve the model and watch the values for  $x$ .

The successive values for  $x$  follow no clear pattern and the message on the Prompt/Error line shows high approximations for successful iteration. Whenever the estimate for the total number of iterations needed is consistently over 20 within one solving, it is possible that your first guess will not be successful.

It is possible that the first guess will never lead to a solution. If this is the problem, you should try a different guess and attempt to solve the model again. Try a new guess of .22 and solve the model again.

This model still fails to solve, because there is something wrong with the model. It could be one of two things, either the equations are inconsistent or the solution is not a real number.

In this case, the solution set consists of complex numbers. This version of the TK!Solver program does not perform complex arithmetic.

### Changing the Number of Iterations

Reset the program and load the BOX model again. When you solved the BOX model earlier, it required more than 10 iterations to solve from the first guess of 4 after the Iterative Solver was restarted.

The default for the maximum number of iterations is 10, but the Iterative Solver stops before that if it converges to an answer. The maximum number of iterations is set on the Global Sheet.

Change the screen to one window and display the Global Sheet. There are five fields in the second group of fields on the sheet. The settings for these fields control the global defaults for the Solvers.

The last field in the group of Solver fields is the Maximum Iteration Count field. It sets the maximum number of iterations performed by the Iterative Solver. Currently it is set to 10, the default count. You can set it to any number you choose, but it must be an integer.

Earlier, you ran the Iterative Solver twice to solve the BOX model from a first guess of 4. If the Maximum Iteration Count had been 20 instead of 10, the Iterative Solver would have had to run only once.

Move the cursor to this field and enter the value 20. Later you will solve the BOX model again using this setting.

When you were running the Iterative Solver, the intermediate values for each step of the iteration were displayed on the sheet. The first field in the group of Solver fields, labeled **Intermediate Redisplay ON**, controls this intermediate display. Since you have solved the BOX model several times already, the intermediate values will not be interesting for another solving, so change the value here to **No** by typing **N**.

Changing this field to **No** can speed up the time needed to solve a model. If you type **Y**, the value changes back to the default value of **Yes**, but leave the field at **No** for this example. Now when the model is solved, the Solver message will still change to show the iterations, but the intermediate values will not be displayed.

Display the Variable Sheet and solve the BOX model with a first guess of 4 for  $I$ . Although the intermediate values are not displayed, each iteration step is noted on the Prompt/Error line as the model is solved. When the Iterative Solver finishes, the Solution Indicator disappears.

### Comparison Tolerance

How does the program know when the Iterative Solver has reached a solution? The first solution of the BOX model, using 10 as a guess, required only seven iterations, although the number of iterations was set at 10. While the program solves the model, it checks both values and equations to see if they are equal according to the Comparison Tolerance specified on the Global Sheet.

Display the Global Sheet again. The third field in the group of Solver fields is the Comparison Tolerance field. The default value is 0.000001.

As the Iterative Solver works, it compares each successive guess value with the current guess value. If the absolute value of their difference divided by the absolute value of the larger of the two values is greater than the Comparison Tolerance, a new guess is generated. If the ratio is less than or equal to the Comparison Tolerance, and if each set of expressions in each equation and all values of a variable are judged equal by the Comparison Tolerance, the problem is considered solved.

This concept for guess values can be stated algebraically. Letting  $ng$  stand for new guess,  $pg$  for previous guess, and  $CT$  for Comparison Tolerance, the Iterative Solver stops whenever the inequality below is true:

$$\frac{|pg - ng|}{\max(|ng|, |pg|)} \leq CT$$

Table 12-3 shows the values for this inequality for each iteration of the BOX model when it was solved with a first guess of 10.

$ pg $	$ ng $	$\frac{ pg-ng }{\max( ng ,  pg )}$
7.7778074	6.5577657	.15686191
6.5577657	6.0810962	.07268779
6.0810962	6.0021081	.01298912
6.0021081	6.0000015	.00035098
6.0000015	6.0000000	.00000025

**Table 12-3**

The Iterative Solver stopped when the inequality obtained a value less than the Comparison Tolerance default value:

$$.00000025 < .000001$$

As the Comparison Tolerance increases in value, the precision of the solution decreases. Conversely, as the Comparison Tolerance is made smaller, the solution becomes more precise.

**Note:** The Comparison Tolerance also has an influence on the Direct Solver. The Direct Solver uses the Comparison Tolerance to determine equality. If the ratio of the absolute value of the difference between two expressions to the expression having the greater value is less than the Comparison Tolerance, the values are treated as equal by the Direct Solver.

### Input Values

The number of input values required for a model varies with both the required output and the method of solution. The number of input values needed for the Direct solution of a model can be

found from the following equation. The variable  $V$  represents the number of variables,  $E$  the number of equations, and  $NV$  the number of input values needed to solve the model.

$$V - E = NV$$

Reset the program and load the DTWO model again. This example uses the DTWO model to experiment with the relationship expressed by the equation above.

In the DTWO model there are 5 variables and 3 equations. In the equation above, if  $V$  is 5 and  $E$  is 3, then  $NV$  is 2. You need two input values to solve the DTWO model.

Try solving the DTWO model giving input values to different pairs of variables.

There are ten possible pairs of variables for input, but only four of these pairs result in a complete solution of the model. The relationship expressed in the equation above tells you only how many input values are needed for solution, not which variables must have the values.

Input values for the following variable pairs provide solutions to the model:

$a, d$

$b, d$

$b, e$

$d, e$

All other solutions of the model require iteration and, therefore, at least one input guess value as well as the two regular input values are needed.

Let's use a specific example. Reset the variable values, then enter the values 13 for  $a$  and 4 for  $c$ .

When you solved this model at the beginning of the chapter, you set the guess values for  $d$  because it appears more than once in the third equation and is the variable necessitating the iterative solution. A guess value is really an output value rather than an input value, so you must still set a guess for  $d$ .

Set  $d$  as the guess variable again by typing **G** in the Status field. If you have not yet entered a value, the program automatically gives  $d$  the value 0 when it is set as a guess. Enter a guess value of 6 in the Input field for  $d$  and solve the model.

The screen should look like Figure 12-5.

```
(4i) Input:
=====
St Input      Name      Output      Unit      Comment
-----
    13         a          -9
    4          b          1.4142136
          c          10.414214
          d
          e
=====
===== RULE SHEET =====
S Rule
-----
a+b=c
d-e=b
d=(a+b)/(2*d)
```

Figure 12-5

## Redundant Equations

An Iterative solution can be avoided in many models by adding a redundant equation to the model. A redundant equation is determined by an algebraic manipulation of one or more equations of the model.

An algebraic manipulation of the third equation of the DTWO model yields a very useful redundant equation. By multiplying each side of the equation by  $2*d$ , the equation is transformed to:

$$2*d^2 = (a + b)$$

By substituting for  $c$  from the first equation, the result is:

$$2*d^2 = c$$

Add this equation to the Rule Sheet. To blank the old output values, use the Blank option (B) in the Status fields for the variables  $b$ ,  $d$ , and  $e$ ; then solve the model with the current input values of 13 for  $a$  and 4 for  $c$ .

The screen should look like Figure 12-6.

(5s) Status:

===== VARIABLE SHEET =====					
St	Input	Name	Output	Unit	Comment
-----					
	13	a			
		b	-9		
	4	c			
		d	1.4142136		
		e	10.414214		

===== RULE SHEET =====	
S	Rule
-----	
	a+b=c
	d-e=b
	d=(a+b)/(2*d)
	2*d^2=c

Figure 12-6

Notice that the solution is the same as the last one, which was solved by iteration. This redundant equation allows the Direct Solver to solve the model for all pairs of input variables except  $a, e$  and  $c, d$ .

If your model contains redundant equations, the number of variables requiring input values is determined by the equation below in which  $NV$  represents the number of variables needing input values,  $V$  is the total number of variables,  $TE$  is the total number of equations, and  $RE$  is the number of redundant equations.

$$NV = V - TE + RE$$

**Note:** In any system of equations, some equations may be unintentionally redundant.

### Automatic Iteration

You can use the Iterative Solver without setting a guess on the Variable Sheet by setting the program to use Automatic Iteration. If the variables are set for Automatic Iteration, the program automatically uses the Iterative Solver if the Direct Solver fails to solve the model. The fields that you must set to do this are on the Global Sheet and the Variable subsheets.

Display the Global Sheet in the top window.

The second field in the group of Solver fields is the Automatic Iteration ON field. The default value for this field is **Yes**. If the field is set to **Yes**, the Direct Solver automatically calls the Iterative Solver if:

- 1 The Iterative Solver is needed to solve the model,  
**and**
- 2 The variables that require Iterative solutions have values in the First Guess fields on their Variable subsheets to set default first guesses.

The model you will use to demonstrate the Automatic Iteration feature is the simultaneous linear equation model that you saw in Chapter 10. It is in an Instruction File named **LINEAR**.

Reset the program and load the file **LINEAR**.

Display the Global Sheet in the bottom window and check it to see if the Automatic Iteration ON field is set to **Yes**. If it is not, type **Y** in that field, then return to the Rule Sheet.

To use Automatic Iteration, you must enter the first guess for each variable on its Variable subsheet. Move the cursor to the row for the variable *a* and use the Dive command (**>**) to display the Variable subsheet for *a*.

The First Guess field is the second field on the sheet. Enter the value **1** in this field. This value will be the default first guess value for *a* if the Iterative Solver is used.

Return to the Variable Sheet, dive into the Variable subsheets for each of the other variables, and enter **1** in the First Guess field on each subsheet. The **LINEAR** model is now set to perform Automatic Iteration.

The message on the Prompt/Error line will be first the Direct Solver message, and then, when the Direct Solver is unable to solve the model, the Iterative Solver message.

Solve the model. The screen should look like Figure 12-7.

```
(5i) Input:

===== VARIABLE SHEET =====
St Input      Name      Output    Unit      Comment
-----
              a         -1
              b          2
              c          4
              d         -3
              e         -2

===== RULE SHEET =====
S Rule
-----
-8=3*a-2*b+c-d+4*e
-27=2*a-b-c+3*d+5*e
15=a+b+c-2*d-2*e
-2=3*a+3*b-c-d+2*e
13=a+2*b+c-4*d+3*e
```

Figure 12-7

The Iterative Solver did not use the guess for the variable  $e$ , because once the values of the other variables were known,  $e$  was uniquely determined.

Although the guess for  $e$  was not used, it did not affect the solution. You can set as many variables as you want for Automatic Iteration. The program will use the minimum number needed to solve the problem.

Most models can be set to use Automatic Iteration, but its usefulness varies from model to model. If a model is solved over and over again within a narrow range of input values, Automatic Iteration would be useful because the first guesses can all be provided on the Variable subsheet and therefore need be entered only once.

For other models, it may be too difficult to set first guesses that are general enough for Automatic Iteration to be beneficial.

### Iteration and TRAVEL2

Now let's return to the TRAVEL2 model and solve a problem using the Iterative Solver. Reset the program, and load the model TRAVEL2. Display the Variable Sheet in a single window, and reset the variable values.

Here is the problem:

If you have a maximum of \$25 to spend on fuel for a 500 mile trip, and gasoline costs \$1.25 a gallon, at what speed must you drive your car to get the mileage you need to complete the trip? The value for internal friction is  $1/3$  Btu/s and the value for wind resistance is  $9.5 \times 10^{-6} \text{ hp} \cdot \text{h}^3 / \text{mi}^3$ .

**Note:** This problem is here strictly as an example to show the Iterative Solver working with TRAVEL. It gives one speed needed to complete the trip and does not maximize mileage.

Before trying the Iterative Solver, try solving the model with the Direct Solver. If any units need to be changed, be sure you change the unit before you enter the input value. Enter the input values for the problem given above: \$25 for *cost*, 500 mi for *d*, \$1.25 for *pr*,  $1/3$  Btu/s for *f*, and  $9.5 \times 10^{-6} \text{ hp} \cdot \text{h}^3 / \text{mi}^3$  for *w*. You want the output value for *s* to be in mi/hr, *mlg* in mi/gal, and *t* in h. Solve the model.

The model was solved, but there is no solution for *s*, the value we are trying to find, or for *t* or *power*. Select the Rule Sheet and you can see that several equations were not satisfied.

Since the Direct Solver seems unable to find the value for *s*, try using the Iterative Solver. Display the Variable Sheet again and move the value for *mlg* into the Input field by typing **I** in the Status field for *mlg*.

Because the value for *s* is the value you are trying to find, it is a good variable to set as a guess. Enter a value of 45 for *s* and set it as a guess. Now solve the model.

The screen should look like Figure 12-8. The model is completely solved.

```
(4s) Status:
===== RULE SHEET =====
S Rule
-----
mlg=d/gas
s=d/t
cost=gas*pr

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
25            mlg              mi/gal     mileage
500           d              mi          distance
              gas           20         gal        amount of g
              s           57.900465  mi/h       speed
              t           8.6355093  h          time
25            cost          $          cost of tri
1.25          pr              price of ga
.33333333    power        2.3160186  hp         energy comp
.0000095     f              Btu/s      internal fr
              w              hp*h^3/mi  wind resist
```

Figure 12-8

## Conclusion

This chapter has explained how the TK!Solver program uses iteration and how the Iterative Solver works. This is the end of Part III. Part IV discusses lists, how to display and print graphs and tables of values, and how to define and use your own functions.



## **Part IV Lists**

TK!Solver lists are lists of values, either numeric or symbolic. If you want to solve a single model for several different values of one or more of the variables, instead of entering one value at a time, you can enter a list of input values that can be used to solve for a list of output values. Lists also designate domains and ranges for functions, and are used to plot graphs and build tables.

This part of the Instruction Manual is about lists of values and how they are created and used.

---

### Contents of Part IV

- Chapter 13 Creating Lists
- Chapter 14 The List Command
- Chapter 15 Plots and Tables
- Chapter 16 User Functions

## Chapter 13 Creating Lists

- The List Sheet
- The List subsheet
- The Fill List feature
- Symbolic values
- Saving a DIF file
- Deleting a list

This chapter is about the kinds of lists you can create and how to create them. As you work through this chapter, you will create four lists that you will use in examples in Chapters 14 and 16.

This chapter also discusses the use of DIF format files and how to save files in DIF format using the DIF option (#) of the Storage command (/S).

---

### To begin this chapter:

Reset the program.

### The List Sheet

Lists and their characteristics are shown on the List Sheet. Display the List Sheet in one window by typing:

```
/W1  
=L
```

The screen should look like Figure 13-1.

(1n) Name:

```

===== LIST SHEET =====
Name      Elements  Unit      Comment
-----

```

Figure 13-1

The List Sheet has the same relationship to the lists in a model as the Variable Sheet has to the variables. The List Sheet shows you the names and characteristics of each list in the model.

There are four columns on the List Sheet. The Name column shows the names of the lists; the Elements column tells you how many values, or elements, are in each list; the Unit column shows the Display unit for the list; and the Comment column is for comments, notes, or descriptions. All of the columns contain entry fields except the Elements column, which contains output fields.

The first three lists that you will create in this chapter will be used with a new model named MORTGAGE. This model will be described in more detail in Chapter 14 when you begin using it.

The first list to be created for MORTGAGE is named *i* and is a list of interest rates. In the first field in the Name column, enter the name of the list, *i*.

List names have the same syntax as variable names. They must begin with a letter and may contain letters, numbers, and the characters @, #, \$, %, and \_. If you try to enter a list name containing another character, such as a + or a blank space, the program will not accept it.

Because the Elements column contains output fields, it will be filled in automatically when you enter the values for the list. Skip over it for now and move to the first field in the Unit column.

Lists have Storage units and Display units, similar to Calculation units and Display units for the variables. Units are entered for a list on the List Sheet just as they are entered for a variable on the Variable Sheet. The first unit you enter is both the Storage unit and the Display unit for the list. Subsequent entries change the Display unit, but not the Storage unit. Conversions between Storage and Display units, as between Calculation and Display units, must be defined on the Unit Sheet. When conversions are not defined, all values are assumed to be in Storage units.

Because no unit conversions have been defined in this model, we will give the list  $i$  one unit for both the Storage unit and the Display unit. Enter the unit name **percent** in the first Unit field.

Comments are always ignored by the program during solutions. Enter the comment **interest rate** in the Comment field for the list  $i$ .

The screen should look like Figure 13-2.

```
(1c) Comment: interest rate

===== LIST SHEET =====
Name      Elements  Unit      Comment
-----
i          -----  percent  interest rate
```

Figure 13-2

## The List Subsheet

Like variables, lists have subsheets, and, like a Variable subsheet, the List subsheet is called with the Dive command ( $\triangleright$ ). To fill in the values for a list, you must call the subsheet and enter the values there. Dive to the subsheet for  $i$  by placing the cursor anywhere on the row containing the description of the list  $i$  and typing:

$\triangleright$

The screen should look like Figure 13-3.

```
(1v) Value:
===== LIST: i =====
Comment:          interest rate
Display Unit:     percent
Storage Unit:     percent
Element Value
-----
```

Figure 13-3

The List subsheet is titled with the name of the list, in this case  $i$ .

The List subsheet looks different from any of the sheets you have worked with so far. The top of the subsheet consists of three labeled lines with one field per line, and the rest of the subsheet consists of two labeled columns.

The first field contains the comment you entered on the List Sheet. The second and third fields show the Display unit and Storage unit.

All values entered in the list are assumed to be in the Display unit if the unit conversion has been defined on the Unit Sheet. The Storage unit is the unit used as a base for conversions of the list values, both to the Display unit and, during solution, to the Calculation unit of the associated variable. The program can make these conversions only if the unit conversions have been defined on the Unit Sheet.

However, unit conversion is only part of the purpose of the Storage unit. If a list is saved in a DIF file, the value saved is in the Storage unit. When the DIF file is loaded, the value is converted from the Storage unit to the Display unit. Again, the program is unable to make these conversions if they have not been defined on the current Unit Sheet. DIF files and their use are discussed in the last section of this chapter.

The Element column contains output fields. When you enter a value in a Value field, the program generates a number in the corresponding Element field. This element number is positional and allows you to locate a particular value in a list. In the next chapter you will learn how to use the element number to get values from a list and put values into a list.

Starting with the cursor in the first Value field, enter the following list of values by typing each value and then pressing  $\downarrow$ .

### Values for i

16.375

17

$16 + 1/2$

18

17.45

$16 + 4/5$

The screen should look like Figure 13-4. Notice that when the values were entered, the expressions were evaluated and element numbers appeared in the Element column.

```

(6v) Value: 16.8

===== LIST: i =====
Comment:                interest rate
Display Unit:           percent
Storage Unit:           percent
Element Value
-----
1      16.375
2      17
3      16.5
4      18
5      17.45
6      16.8

```

Figure 13-4

**Note:** If you try to enter an expression that includes a variable defined on the Variable Sheet, the program evaluates the value using the Calculation unit and not the Display unit.

Return to the List Sheet with the Return command ( $\zeta$ ), and notice that the Element field for  $i$  has been filled in with a **6**, the number of values in the list. The list  $i$  is now defined and ready to be used.

### The Fill List Feature

The next list to be created for the MORTGAGE model is a list of values describing the possible payment periods, or term, for a mortgage. Move the cursor into the Name field of the next row, and name this list **term**. Enter the Display unit **years** and the comment **payment period**, then dive into the subsheet.

The values for **term**, in years, are to run from 5 to 50 in intervals of 5. You could enter all the values one at a time, but the values can be entered automatically using the Fill List feature.

The Fill List feature can be used if a list is to be numeric with the numbers evenly spaced. Given the first and last values in a list, the Fill List feature interpolates the intervening values and enters them. It can be used only on a sheet containing list values and is called by the Action command (!).

In the first field in the Value column, enter the value 5. Move the the cursor to the 10th element by typing:

:10 ◆

Enter the value 50.

The screen should look like Figure 13-5.

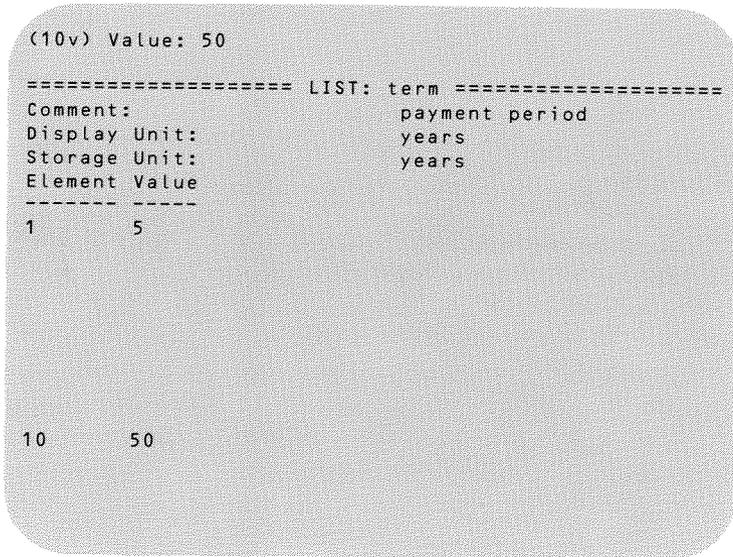


Figure 13-5

Now press the Action command (!). The message is:

Fill List: Y N

This request for confirmation is to prevent you from accidentally writing over values that may already be in the list.

Type **Y**. The screen should look like Figure 13-6 with the values for *term* filled in.

```
(10v) Value: 50

===== LIST: term =====
Comment:                payment period
Display Unit:           years
Storage Unit:          years
Element Value
-----
1          5
2          5
3          5
4          5
5          5
6          5
7          5
8          5
9          5
10         5
```

Figure 13-6

The list *term* is now defined. The last list to be created for MORTGAGE is a list of bank names.

### Symbolic Values

Lists may contain symbolic as well as numeric values. Return to the List Sheet with the Return command (<), and create a new list named **B** with the comment **bank names**. Because this is a list of symbolic values, it does not need a unit.

Dive into the subsheet.

An apostrophe (') tells the program that what follows is a symbolic value. Symbolic values have the same restrictions as TK!Solver variable and list names. They must begin with an alphabetic character and may contain letters, numbers, and the characters @, #, \$, %, and \_\_. They may not contain blank spaces or operators.

The first bank on the list is named the First National Bank. Because you cannot enter blank spaces for a symbolic value, use underscores instead. With the cursor in the first Value field type:

```
'First_National
```

Here is an example of how to use the Edit command (⌘). If you forget to type an apostrophe, ⌘ calls the Editor. The next bank is named Main Street Bank. Move the cursor to the second Value field and type but do not enter:

```
Main_Street
```

**Note:** If you tried to enter the value as shown above, you would get an error message because the apostrophe (') is missing, and the program would automatically call the Editor. The cue would move to the place where the first error is suspected.

Type ⌘ to call the editor. Move the cursor to the beginning of the field with ⌘ and add the missing apostrophe ('). Enter the value with ⌘.

**Note:** This command can also be used to edit an item on the Prompt/Error line, such as a filename you may have misspelled.

Complete the list by entering the remaining four bank names in the order given below. Enter them, typing underscores in place of the blanks, with an apostrophe before each name to tell the program that it is a symbolic value.

### Values for B

```
Family Savings  
County Bldg and Loan  
City Cooperative  
Little River Federal
```

The screen should look like Figure 13-7.

```

(6v) Value: 'Little_River_Federal

===== LIST: B =====
Comment:                bank names
Display Unit:
Storage Unit:
Element Value
-----
1      'First_National
2      'Main_Street
3      'Family_Savings
4      'County_Bldg_an
5      'City_Cooperati
6      'Little_River_F

```

Figure 13-7

The lists for MORTGAGE are now all entered. The last list to be created is a list of mileages for the model TRAVEL2.

### Creating the List for TRAVEL2

Return to the List Sheet using the Select command (=) and move the cursor down to the next blank row.

Except for the list name, all the characteristics of a list can be entered on the List subsheet. Name the list *mlg* and dive into the subsheet.

Enter the comment *mileage* and both the Display and Storage unit as *km/1*.

The values for *mlg* are from 4.251 to 19.1295 and there are eight elements in the final list. Enter 4.251 as the first value and 19.1295 as the eighth, then use the Action command (!) to fill the list.

Return to the List Sheet. The screen should look like Figure 13-8. All of the information has been filled into the row for *mlg*.

(4n) Name: mlg

```
===== LIST SHEET =====  
Name      Elements  Unit      Comment  
-----  
i         6          percent  interest rate  
term     10         years    payment period  
B        6          bank     bank names  
mlg      8          km/l     mileage
```

Figure 13-8

## Saving a DIF File

DIF is a standard format for storing information. It allows different programs, all using the same storage format, to use the information in a single file, and avoids the necessity of saving the same information over again in different formats for different programs.

If you save a list in a DIF file, you can load the values into another program that uses the DIF format. Conversely, if you have a DIF file of values from another program, those values can be loaded into the TK!Solver program as a list.<sup>1</sup>

Normally when you are using a list with different models, you use the Save Model option (S) and the Load File option (L) of the Storage command (/S) to save and load the file containing the list. The DIF option (#) should be used only when lists are to be

<sup>1</sup> For information about the DIF format and programs that use it, write to:  
DIF Clearinghouse  
PO Box 638  
Newton Lower Falls, MA 02162

shared between programs, when the values are going to be used in another program, or when values from another program are going to be used with the TK!Solver program. However, as an example of how to save a DIF file, the following section shows you how to save the list *mlg* in a DIF file. In the next chapter you will load it again from the DIF file.

The DIF option (#) of the Storage command (/S) can be used to save and load only lists. If you are going to save a list in a DIF file, you must be on the List Sheet. The cursor must be on the row containing the list you want to save before you type the command. Move the cursor to the row describing the list *mlg* and type the Storage command with the DIF option:

```
/S#
```

The message is:

```
D I F   S t o r a g e :   S   L
```

These choices are Save (S) and Load (L). Type:

```
S
```

The prompt is:

```
D I F   S a v e :   P o i n t   t o   l a s t   l i s t   ( 4 n )
```

If you want to save more than one list, you could use the cursor to point to a block of lists on the List Sheet and they would all be saved in the file.

Because *mlg* is the only list to be saved, press .

The prompt is:

```
D I F   S a v e :   F i l e n a m e :
```

The cue is on the Prompt/Error line so that you can type in the filename, just as for any other Storage command. Enter the filename TLIST.

The program names the file TLIST and adds the DIF extension so that any program that uses DIF files can identify it. The information saved in the DIF file consists of the values in the list expressed in the Storage unit. Both Display and Storage unit names are saved, but the conversion is not. You must be sure that the conversion is defined on the Unit Sheet before you load the DIF file into a model.

### Deleting a List

You can delete a list by deleting the row on the List Sheet. The list *mlg* has been saved for later, so delete the row using the Delete command (/D).

Dive into the subsheet for that row. The list is gone.

The lists for MORTGAGE have been created, and the list for TRAVEL2 has been created, saved in a DIF file, and deleted from the current List Sheet. The current information can now be merged with the model MORTGAGE.

Do this by loading the Instruction File MORTGAGE. Save the new version of MORTGAGE again using the same filename.

### Conclusion

This chapter has shown you how to create lists. The next chapter explains the List command (/L) and how to use it.

## **Chapter 14 The List Command**

- The List command (/L)
- The Solve option (!)
- Creating a list through the Variable Sheet
- Changing list associations
- Loading a DIF file
- The Block option (B)
- The Get (G) and Put (P) options

One of the ways to use lists of values is with the List command (/L). The List command has several options that use lists in association with variables. Once a list has been associated with a variable on the Variable Sheet, the values in the list can be used for input or output values of the variable and can be manipulated from the Variable Sheet.

This chapter is about the List command, what it is, and how to use it.

---

### **To begin this chapter:**

At the end of the last chapter, the MORTGAGE model was loaded.

If you quit after the last chapter, load the program, then load the MORTGAGE model that you saved at the end of Chapter 13.

Display the Variable Sheet in a single window. The screen should look like Figure 14-1.

### **The MORTGAGE Model**

In this chapter you will use a new model named MORTGAGE. The MORTGAGE model is set up to show the relationships between variables for a major credit purchase. It can be used to determine several values, including monthly payment amount, term of payment, down payment percentage, and interest rate. The model has nine variables in five equations.

(11) Input:

St	Input	Name	Output	Unit	Comment
		H		dollars	House price
		d		dollars	Down paymen
		dp		percent	Down paymen
		p		dollars	Mortgage
		n		years	Term
		i		percent	Interest ra
		A		dollars	Monthly pay
		T		dollars	Total of pa
		Tip		dollars	Total inter

Figure 14-1

Display the Unit Sheet for the MORTGAGE model in the bottom window. You can see that it contains no unit conversion definitions. Because some of the variables in the model have different Display and Calculation units, these unit conversions must be defined before values are entered in the model.

If unit conversions have not been defined, values entered on the Variable Sheet are assumed to be in Calculation units. If you enter a value before defining the necessary conversions on the Unit Sheet, some or all of your values may be incorrect.

The units file that you created in Chapter 8 contains the units you need for MORTGAGE. Load the file MUNITS. Redisplay the Rule Sheet in the bottom window and save the new version of MORTGAGE.

## The List Command

The List command (/L) has several options that enable you to manipulate lists of values by associating them with variables on the Variable Sheet. These options include the means to use lists or partial lists as input and output values with either the Direct or Iterative Solver.

The four options are:

Option	Effect
Solve (!)	Solves the model for all the values in the associated list or lists.
Block (B)	Solves the model for a given block of values within a list.
Put (P)	Puts the current value of each variable associated with a list at a designated element in the associated list.
Get (G)	Gets a value from each associated list from a designated element in the list and displays it as the current value of the variable.

### The Solve Option

The Solve option (!) of the List command (/L) uses either the Direct or Iterative Solver, according to the needs of the problem, and uses lists for the input and output values.

The problem given below is a typical application of the MORTGAGE model.

A house is offered on the market for \$95,000. A bank charging 17.5% interest requires 20% down payment. How is the monthly payment amount for this mortgage affected by the term of the loan?

You can use the Solve option (!) of the List command (/L) to solve this problem, giving a list of terms as input values and solving for a list of relative monthly payment amounts.

### Preparation for the Solve Option

To use the Solve option (!) you must do two things. You must tell the program which variables are to be associated with lists, and you must tell it which of those lists should be used for input values, as opposed to those that will receive output values.

In this example, two variables will be associated with lists. The variable  $n$  requires an input list because it represents the term. The changes in the value for the term changes the value of the monthly payment amount, represented by the variable  $A$ . Therefore,  $A$  will be associated with an output list.

A variable is designated as an input variable by giving it an arbitrary input value. Because  $n$  will use a list of input values, you must give it an input value so that the program recognizes it as an input variable.

Although it is not necessary, it is usually a good idea to solve the model once before using the List command (/L), just to be sure the problem is correctly stated. Enter the values given in the problem above and the arbitrary value of 25 for the variable  $n$  to designate it as an input variable.

**Note:** Commas (,) are not accepted in the Input field, so don't try to enter the comma as part of the house price.

Solve the model. The screen should look like Figure 14-2.

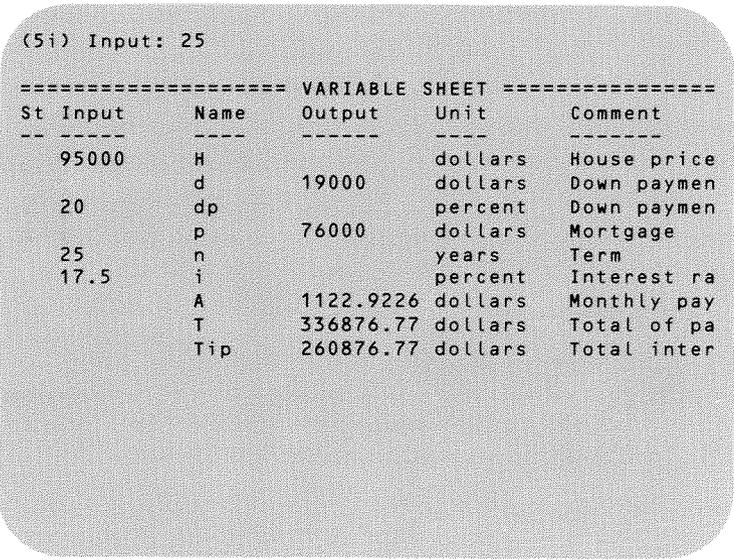


Figure 14-2

If the model solved correctly, the program knows which values are input and which are output. Next, identify the variables to be associated with lists.

An **L** in the Status field of a variable associates that variable with a list. An **L** can be entered in a Status field by two methods. We will use one method for  $A$  and the other for  $n$ .

Move the cursor to the Status field for the variable  $A$  and type:

L

Now dive to the Variable subsheet for  $A$ .

The **L** that you entered on the Variable Sheet appears in the Status field.

The third field on the subsheet is the Associated List field and contains the name of the list that is associated with the variable, in this case  $A$ . If this field is blank when you type an **L** in the Status field on the Variable Sheet, the program automatically uses the variable name for the name of the associated list.

If you want to look at the associated list, you can dive from the Variable subsheet to the List subsheet. It does not matter where the cursor is on the Variable subsheet. Dive to the List subsheet for  $A$ .

List  $A$  is currently empty. The variable  $A$  is an output variable, so list  $A$  will be filled, during solution, with output values.

Notice that the Calculation and Display units for variable  $A$  have become the Storage and Display units, respectively, for list  $A$ . Units are automatically generated for a list only when it is a new list created by setting a list association on the Variable Sheet.

The Return command ( $\langle$ ) calls the previous sheet. Return to the Variable Sheet by typing the Return command twice ( $\langle\langle$ ).

The other method of associating a variable with a list will be used for  $n$ . Move the cursor to the row containing the variable  $n$  and dive to the Variable subsheet.

In Chapter 13 you created a list of payment periods named *term*, and then saved it as part of the MORTGAGE model. This is the list you want to associate with the variable  $n$ .

Move the cursor to the Associated List field and enter the list name *term*. Notice that the Status field immediately shows **L**. Whenever a name is entered in the Associated List field, the program automatically enters an **L** in the Status field of the variable. If there is no list with that name, an empty list is created.

Review the values in the list *term* by diving to the List subsheet for *term*. As you can see, it is the same list you created in the last chapter.

Use the Select command (=) to display the Variable Sheet and notice that there is an **L** in the Status field for *n*.

**Note:** If you reset the variable values with the Variable Value option (V) of the Reset command (/R), the list associations are left intact.

### Using the Solve Option

You have designated your input variables and set your two list associations, so you are now ready to use the List command (/L) with the Solve option (!). Because the Solve option calls one of the Solvers, the messages that appear when the Solvers are running also appear when this option is used. The intermediate values for this solving will be the values in the lists, so they are of interest. Check the Global Sheet to be sure that the Intermediate Redisplay ON field is set to **Yes**, then return to the Variable Sheet.

Type the List command:

```
/L
```

The message is:

```
List: ! B P G
```

Type the Solve option:

```
!
```

The values that appear on the Variable Sheet are the input and output values in the lists. While the Direct Solver solves using each element in the input list, the message is:

```
List Solver: element number, Direct Solver
```

The element number is incremented as each element of the input list is used. The element number would be shown even if the Intermediate Redisplay ON field was set to **No**, although the input and output values would not be shown.

Because the Solve option (!) can run with either the Direct or Iterative Solver, this message also shows which solver is being used. When the message clears, the Direct Solver has finished. The original values are redisplayed at the end of solution.

Type the Dive command twice ( $> >$ ) to display the List subsheet for list  $n$ . Use the Window command ( $/W$ ) with the List Sheet option ( $L$ ) to split the screen and display the List Sheet in the bottom window. Then dive to the List subsheet for list  $A$  in the bottom window.

The output values in list  $A$  are in the same order as the input values in list  $n$ . If you scroll down to look at the last few elements in both lists, you can see that the last few values in list  $A$  are all very close. When you compare the corresponding elements in list  $n$  and list  $A$ , you can see that the impact of varying the length of the payment period almost vanishes around 35 years, although the overall cost is quite different.

**Note:** If the program is unable to solve for a value, that element in the output list will be left blank.

### Creating a List Through the Variable Sheet

In Chapter 13 you created lists using the List Sheet. Since you can display a List subsheet by diving from the Variable Sheet, you can also create a list by diving down from the Variable Sheet. Lists created through the Variable Sheet are automatically recorded on the List Sheet.

The list you are about to create will be used with a second MORTGAGE example. This problem examines how different down payments affect the monthly payment amount.

Display the Variable Sheet in a single window. Set the variable  $dp$  as a list variable by typing  $L$  in the Status field, then dive down to the List subsheet for the new list  $dp$ .

Enter the comment **down payment %**. The Display and Storage units have been assumed from the variable Display and Calculation units. Use the Fill List feature to enter 10 values from 5 to 50.

The screen should look like Figure 14-3.

```
(10v) Value: 50

===== LIST: dp =====
Comment:                down payment %
Display Unit:           percent
Storage Unit:           decimal_form
Element Value
-----
1          5
2         10
3         15
4         20
5         25
6         30
7         35
8         40
9         45
10        50
```

Figure 14-3

Return to the Variable Sheet.

## Changing List Associations

Of the two variables associated with lists from the previous solution of the model, only one is needed for the new problem, the list associated with the variable  $A$ . Because the variable  $dp$  is associated with the input list, the list association for variable  $n$  is not required for this problem.

The List option (L) in the Status column acts as a switch both to associate and disassociate a list. If a variable has a list associated with it, the association can be broken by typing **L** in the Status field. Move the cursor to the Status field for the variable  $n$  and type **L**. The **L** that was there disappears. The association has been broken, but the list remains intact.

The variable  $A$  is still associated with a list of output values. When you solve the model again, the new output values will replace the old ones.

If you want to save the old values, you must rename the list. The current list  $A$  is needed later in the manual, so the list must be renamed. Display the List Sheet using the Select command (=).

Change the name of list *A* to *A1* by editing the current list name or entering the new name in the name field for *A*. Now the list *A1* contains the old values and list *A* is empty.

Call the Global Sheet and set the Intermediate Redisplay ON field to **No**. This speeds up the solution process. Redisplay the Variable Sheet.

The model has been solved once with the variable *dp* as an input variable and the variable *A* as an output variable, so you do not have to designate them again as input and output. The value for the variable *n* can remain as it is.

Type the List command, `/L` with the Solve option, `!`.

The message is the same as it was the last time you ran the Solve option, but the list values are not shown during solution. Because the program does not have to stop to show each value, the time needed to reach the solution is much less.

### **The List Command with TRAVEL2**

The next example uses the TRAVEL2 model and the DIF file TLIST to show you the Block option (B) of the List command (`/L`).

Save the newest version of the MORTGAGE model with the new lists, then reset the program and load the TRAVEL2 model.

### **Loading a DIF File**

In Chapter 13 you saved a list for TRAVEL2 in a DIF file. Normally, information would be loaded from a DIF file only if it were used in another program. The DIF file TLIST is used in this chapter to show you how to load a DIF file.

Load the DIF file by typing the Storage command (`/S`) with the DIF option (`#`) and then choosing Load (`L`):

```
/S#L
```

The message is:

```
DIF Load: Filename:
```

As for the other options of the Storage command (/S), the cue is left on the Prompt/Error line so that you can enter the filename. Type:

```
TLIST ♦
```

While the file loads, the message is:

```
Loading from TLISTDIF extension
```

When the message disappears, the file is loaded.

Display the List Sheet. The list *mlg* has been loaded.

Display the Variable Sheet in the top window and the Rule Sheet in the bottom window, then save the new version of the TRAVEL model in a file named TRAVEL3.

**Note:** Lists are saved with all their characteristics, but if a list already on the List Sheet has the same name as one that is being loaded, the two lists are not merged. The new list is loaded with the Name field blank, even though the list name is stored in the DIF file.

**Note:** Because list names must follow the TK!Solver program's convention for names, list names beginning with a number, saved from other programs, are not recognized as list names by the TK!Solver program.

### The Block Option

The Block option (B) of the List command (/L) works exactly as the Solve option (!) except that it operates on a specified block of elements within the lists instead of on the full lists.

The model must be prepared for the Block option just as it is prepared for the Solve option. First designate the input variables by giving them arbitrary values, then associate lists with the appropriate variables.

This example reintroduces the hypothetical car that was used in Chapter 7. The solution to the problem requires iteration. We will use automatic iteration. Because it is not possible to choose a first

guess value that solves the model for every value in the list *mlg*, the Block option (B) of the List command (/L) will be used to solve for the greatest possible range of values.

The experimental car now has an internal friction factor of 0.65 Btu/s and wind resistance of  $3.5 \times 10^{-6} \text{ hp} \cdot \text{h}^3 / \text{mi}^3$ . Given a trip of 500 miles, what speeds can be traveled to obtain the mileages in the list *mlg*?

Because the problem is concerned with the effect of speed on mileage using the factors for internal friction and wind resistance as stated in Chapter 7, the equation on the Rule Sheet dealing with cost,  $\text{cost} = \text{gas} * \text{pr}$ , is extraneous to this application. Mask it by moving the cursor to that rule, calling the Editor, and adding a quotation mark (") before the first character. The program now treats this equation as a comment and the program will use the four remaining equations.

On the screen, display one window containing the Variable Sheet. Because *cost* and *pr* are in the masked rule only, there are eight variables used in this example, and the model requires four input values for solution.

First determine which variables will be associated with lists and which must be designated as input variables. Since the example explores the relationship of mileage to speed, the variable *mlg* will use the list *mlg* for input values and the variable *s* will require a list for output values.

If your Variable Sheet has values, reset them using the Variable Values option (V) of the Reset command (/R). Enter the values given in the problem along with the arbitrary value 25 mi/gal for *mlg*, so that it is marked as an input variable. The screen should look like Figure 14-4.

The list *mlg* contains values for the variable *mlg* that can be used to solve for a list of values for the variable *s*.

Associate the variable *mlg* with the list *mlg*. Because they have the same name, you have only to type **L** in the Status field for the variable *mlg*.

```
(4s) Status:

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
    25      mlg      mi/gal     mi/gal     mileage
    500      d        mi         mi         distance
          gas      gal        gal         amount of g
          s        mi/h       mi/h        speed
          t        h          h          time
          cost     $          $          cost of tri
          pr        hp         hp         price of ga
          power    hp         hp         energy comp
    .65      f        Btu/s      Btu/s      internal fr
    .0000035 w        hp*h^3/mi  hp*h^3/mi  wind resist
```

**Figure 14-4**

Now dive to the Variable subsheet for the variable *mlg*. As you can see, the list association is set. Look at the units. Both Calculation and Display units for the variable *mlg* are in **mi/gal**. Dive to the List subsheet for the list *mlg*. Here both the Display and Storage units are **km/l**.

Table 14-1 below shows the units for the variable *mlg* and the associated list *mlg*.

<b>Variable <i>mlg</i></b>	<b>List <i>mlg</i></b>
Display unit	Display unit
mi/gal	km/l
Calculation unit	Storage unit
mi/gal	km/l

**Table 14-1**

When an existing list is associated with a variable, either the Calculation and Storage units must be the same, or there must be a conversion defined between them on the Unit Sheet; otherwise the values used and produced by the List command (/L) will be inaccurate.

The conversion between mi/gal and km/l is defined on the Unit Sheet, and the program can make the conversion.

In this example, all necessary unit conversions have been defined, so the model can be solved in its current state. But while you are working on this problem, it would be more convenient if the Display units were the same for the variable and the list.

Change the Display unit for the list to **mi/gal**. Now, even though they are stored in the DIF file as km/l, in this model the values for the list *mlg* are shown in **mi/gal**.

Return to the Variable Sheet and associate the variable *s* with the list *s*.

Because the variable *s* is a guess variable for iteration, it requires a guess value for each element that will be in the output list. There are two ways to set these values. You can either use Automatic Iteration, or you can enter a set of first guesses in the list *s*. A list of first guesses would be overwritten automatically as the output values were generated.

This example sets the guess value for the variable *s* using Automatic Iteration. The program will use the First Guess value on the Variable subsheet for *s* as the first guess in an Iterative solution for each value in the output list.

In some cases this may not be a successful guess for all of the elements. If you use this method of setting a first guess for an Iterative solution of lists, you must be sure to examine the results carefully to be sure that they are sensible results.

Check the Global Sheet to be sure the Automatic Iteration ON field is set to **Yes**, and return to the Variable Sheet.

Dive to the subsheet for the variable *s* and enter **20** in the First Guess field. The model is now ready for solution.

In this example, we will use the Block option (B) to solve the model for the second through sixth elements in the list *mlg* rather than for the whole list of values.

Return to the Variable Sheet and type:

/LB

The prompt is:

```
List Block: First element:
```

This prompt asks you for the element number of the first value in the block of values that you want to use. Since we are interested in the values for elements 2 through 6, type:

```
2 ♦
```

The next prompt is:

```
List Block: Last Element:
```

The last element to be used is element 6, so type:

```
6 ♦
```

While the model is being solved for these five elements, the message is:

```
List Solver: element, Iterative Solver,  
step out of estimate
```

Notice that this is the same message that appears when you use the Solve option. The Block option is exactly the same as the Solve option, but the Solver is applied only to a specific section of the list.

Because the program runs the Iterative Solver for each element in the specified block, it takes time to complete the solution.

When the message disappears the Iterative Solver has solved the model for the five elements, but no value for  $s$  is shown on the Variable Sheet.

Dive to the List subsheet for  $s$ . Your screen should look like Figure 14-5. These values are the set of speeds required for the specified mileages in this hypothetical car.

Notice that the element numbers of the values in the list  $s$  correspond to the element numbers of the input values in the list  $mlg$ , even though the list contains no other values.

```

(1v) Value:

===== LIST: s =====
Comment:
Display Unit:          mi/h
Storage Unit:          mi/h
Element Value
-----
2          13.94790435
3          18.87821891
4          24.25812781
5          30.62755156
6          40.12852363

```

Figure 14-5

### The Get and Put Options

The Get (G) and Put (P) options of the List command (/L) can be used to look at a particular instance of a model. The Get option (G) tells the program to get a specified element from each associated list and show the values on the Variable Sheet. The Put option (P) of the List command (/L) puts the current values of all variables associated with lists into the lists at a specified element.

Return to the Variable Sheet.

The Block solution of the TRAVEL3 model did not provide a value for the first element of the list *s*. The Get (G) and Put (P) options of the List command (/L) can be used to find this value.

To see the value for the first element in the list *m/g*, use the List command (/L) with the Get option (G). The cursor can be anywhere on the sheet. Type:

```
/LG
```

The prompt is:

```
Get from element:
```

This prompt asks for the element number of the value you want to see. We are interested in the first element, so type:

1 ♦

The program gets all first elements in associated lists and displays them.

The values shown for the variables *mlg* and *s* are those stored as the first element in their respective lists. The list *s* has no value at the first element. Set the variable *s* as a guess variable with a value of 10. The **L** remains in the Status field. A variable may be both a guess variable and associated with a list.

Solve the model. The screen should look like Figure 14-6.

```
(4i) Input:
===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
L  10         mlg              mi/gal    mileage
      500         d              mi        distance
              gas         50         gal       amount of g
L              s         9.2311630  mi/h     speed
              t         54.164356  h        time
              cost        $          cost of tri
              pr         $          price of ga
              power    .92311630  hp       energy comp
      .65         f         Btu/s     internal fr
      .0000035  w         hp*h^3/mi  wind resist
```

**Figure 14-6**

When the value has been found, you can replace the null value in the first element of list *s* with the new value by using the Put option (P).

Use the List command (/L) with the Put option (P). Type:

/LP

The prompt is:

Put to element:

This value is to be the first element, so type:

1 ♦

Dive to the List subsheet for *s* and look at the first element. The value has been changed. Because the value of the first element in the list *mlg* was not changed on the Variable Sheet, it was not changed in the list.

Save TRAVEL3.

**Note:** The purpose of this example is to demonstrate the use of the List command (/L). It explores only one of several possible sets of answers that can be obtained using this model. You can explore the problem yourself by changing the first guess value to determine maximums and minimums.

### **Conclusion**

This chapter described the List command (/L) and how to use it. The next chapter describes how to use lists to plot graphs and print tables of values.

## Chapter 15 Plots and Tables

- The Plot Sheet
- The display scale
- The Table Sheet

When you have solved a problem using lists for input and output values, you may want to compare the values in two or more of the lists. The TK!Solver program provides two methods for comparing values: plots and tables. TK!Solver plots and tables can be displayed on your screen, printed on a printer, or written in print files.

This chapter tells you how to use the Plot and Table Sheets to produce plots and tables of list values.

---

### To begin this chapter:

Reset the program.

### The NPOWER Model

The model used to show you how to produce plots and tables is named NPOWER and is on your Instruction diskette. Load the file NPOWER. The screen should look like Figure 15-1.

NPOWER is a simple model that finds the square and the cube of  $n$ . Using this model, you will create lists associated with the variables  $a$ ,  $b$ , and  $c$ , and then compare them.

The NPOWER model includes a list named  $n$  that you can see if you look at the List Sheet. This list will be associated with the variable  $n$ . First, set the variable  $n$  as an input value by giving it the arbitrary value of 1. Then associate all the variables, including  $n$ , with lists by typing **L** in each of their Status fields on the Variable Sheet.

When the model is solved using the values in the list  $n$  as input values, lists of output values will be created for the variables  $a$ ,  $b$ , and  $c$ .

```
(1r) Rule: a=n

===== VARIABLE SHEET =====
St Input      Name      Output    Unit      Comment
-----
              a
              n
              b
              c

===== RULE SHEET =====
S Rule
-----
* a=n
* b=n^2
* c=n^3
```

Figure 15-1

Solve the model using the List command (/L). When it has finished, save the new version of the NPOWER model, with the three new lists *a*, *b*, and *c*, writing over the old version of the model.

### The Plot Sheet

The information that you enter on the Plot Sheet tells the program which values you want to plot and how you want the plot to look. Display the Plot Sheet in a single window by typing:

```
/W1
=P
```

The Plot Sheet, like the List subsheet, has labeled lines at the top of the sheet, each with one field, and columns filling the rest of the sheet.

The first row on the sheet is the Screen or Printer field. It is an option field with the default **Screen**. If this field is set to **Screen**, the plot is shown only on your screen. If you have a printer, you can type a **P** in this field to send the plot to the printer or to a print file. The specifications for the printer are set on the Global Sheet as described in Chapter 9.

If the Screen or Printer field is set to **Printer**, the plot is shown on your monitor while it is being printed. Because it is scaled for the printer, the plot may not be readable on the screen.

The second field on the sheet is the Title field. Anything that you enter in this field is printed as a title at the bottom of the plot.

The third field is the Display Scale ON field with the default of **Yes**. If this field is set to **Yes**, the numbers used for scaling the plot are shown along both axes. If it is set to **No**, no scale values are shown.

The last field at the top of the sheet is the X-Axis field. The program uses the values in the list named here as the  $x$ -coordinates. You can enter any legal name in this field, but if it is not the name of a list, or if the list is blank, you will receive an error message saying that no values are available for the plot.

There are two columns filling the rest of the sheet. The left column is the Y-Axis column. The program uses the values in the lists named here as the  $y$ -coordinates for the plot of each point. More than one list can be used for  $y$ -coordinates.

The Character field for each Y-Axis list tells the program which single character to use when plotting with that set of  $y$ -coordinates. You can use any printable character for a plot, but if you want to use a command character, such as /, =, or :, you must use either  or /E to type it in the field. You can type only one character in each field of the Character column. ♦ is not required.

Plot the values in the list  $n$  against the other three lists by giving the Plot Sheet the following settings:

- The plot should be displayed on your screen, so if the first field on the sheet is not set to **Screen** move the cursor to that field and type **S**.
- In the second field, enter the title **Plot of  $n$** .
- The Display Scale field should be set to **Yes**. If it isn't, move the cursor to that field and type **Y**.
- The list to be used for the  $x$ -axis is the original list,  $n$ . Move the cursor to the X-Axis field and enter the list name  **$n$** .
- The plot will relate the values in  $n$  to the corresponding values in the lists  $a$ ,  $b$ , and  $c$ . Move the cursor down to the first Y-Axis field and enter the list name  **$a$** .

The default plot character is an asterisk. It appears automatically in the first Character field. The plot of  $n$  to  $a$  will be shown using asterisks.

- Now move the cursor down and enter the list names  $b$  in the second Y-Axis field and  $c$  in the third. Asterisks are automatically entered in their Character fields as well.
- Because we want to see how the graphs for the three lists differ, each plot should use a different character. Move the cursor to the Character field for  $b$  and type the character  $\#$ . The Character field accepts only one character and does not require  $\diamond$ . Move the cursor to the Character field for  $c$  and enter the character  $o$ .

Look at the plot by typing the Action command ( $\uparrow$ ).

The screen should look like Figure 15-2. This is the plot of  $n$  related to  $a$ ,  $b$ , and  $c$ , represented by the characters set on the Plot Sheet. Notice the title at the bottom of the plot.

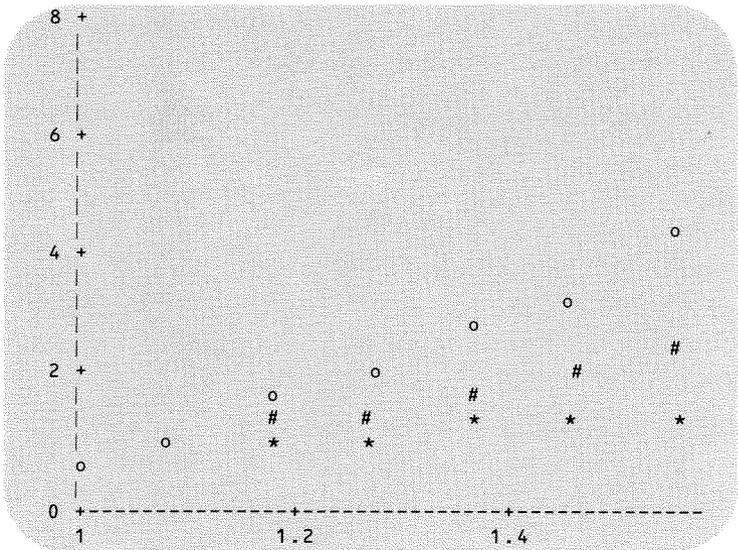


Figure 15-2

**Note:** The program plots one character per point. If points overlap, the last one takes precedence.

**Note:** If the plot is a complex one the program may require a minute or more to produce it.

### The Display Scale

The values shown on the  $x$ - and  $y$ -axes of the plot are generated by the program. Based on the highest and lowest values in the X-Axis list, it produces evenly spaced values that are shown as the scale for the  $x$ -axis.

The values for the  $y$ -axis are produced in a similar manner, except that all the Y-Axis values are combined, and the highest and lowest values of the combined lists are the ones used for the scale.

You can turn off the display of the scale values by setting the Display Scale ON field to **No**.

When you have finished looking at or printing the plot, return to the Plot Sheet by pressing  $\blacklozenge$ .

### The Table Sheet

Another way of comparing lists of values is by looking at them in a table. The information that you enter on the Table Sheet tells the program which values you want shown as a table and how you want the table to look. Display the Table Sheet by typing:

=T

Like the Plot Sheet, the Table Sheet has labeled lines at the top of the sheet, each with one field, and columns filling the rest of the sheet.

The first field on the Table Sheet is the Screen or Printer field and is exactly like the one on the Plot Sheet. It is an option field with the default **Screen**. If the first field is changed to **Printer**, the table appears on the screen while it is being printed on the printer or written to a print file.

The second field on the Table Sheet is the Title field. Like the Title field on the Plot Sheet, it is for the title of the table. The title is shown at the top of the table.

The third field sets the table to be either vertical, with the values in each list shown from top to bottom, or horizontal, with the values in each list shown from left to right. This field is an option field with the default of **Vertical**.

The List column contains the names of the lists to be shown in the table.

The Width column tells the program how many characters wide the columns in the table are to be. The width must be an integer. The default is **10**.

The First column contains the element numbers of the first value in each list that is to be in the table. Because the number represents an element, the number must be an integer. The default is **1**.

The Header column contains the header or title that appears at the beginning of each list in the table.

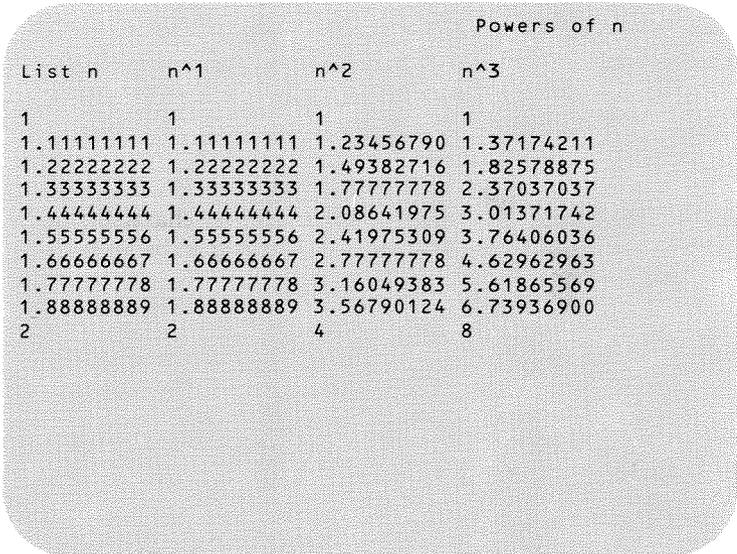
When you enter a list name, the Width and First fields are filled in with the default settings. You can change them according to your needs.

The Header field may remain blank.

Set up the Table Sheet for the four lists in *NPOWER* using the following settings:

- Leave the first and third fields at their defaults, **Screen** and **Vertical**.
- Give the table the title **Powers of n**.
- In the List column, enter the names of the four lists—**n**, **a**, **b**, and **c**.
- Leave the column width at **10** characters and the first element as **1**.
- In the Header column, enter a header for each list—**List n** for **n**, **n<sup>1</sup>** for **a**, **n<sup>2</sup>** for **b**, and **n<sup>3</sup>** for **c**.

Look at the table by typing the Action command (!). The screen should look like Figure 15-3.



Powers of n			
List n	n^1	n^2	n^3
1	1	1	1
1.111111111	1.111111111	1.23456790	1.37174211
1.222222222	1.222222222	1.49382716	1.82578875
1.333333333	1.333333333	1.77777778	2.37037037
1.444444444	1.444444444	2.08641975	3.01371742
1.555555556	1.555555556	2.41975309	3.76406036
1.666666667	1.666666667	2.77777778	4.62962963
1.777777778	1.777777778	3.16049383	5.61865569
1.888888889	1.888888889	3.56790124	6.73936900
2	2	4	8

**Figure 15-3**

Press  $\blacklozenge$  to return to the Table Sheet.

Now make the table horizontal. Move the cursor to the Vertical or Horizontal field and type H.

Type the Action command (!). The screen should look like Figure 15-4.

A horizontal table may use more space across the screen than a vertical table. There are ten values in each of these lists, but if you count the number of values on the screen, some are missing.

To see the rest of the table, type  $\blacklozenge$ . If the table requires a third page, type  $\blacklozenge$  again. This is necessary only when displaying the table on the screen. The printer automatically prints out the required number of pages for the entire table.

Again, press  $\blacklozenge$  to return to the Table Sheet.

**Note:** If you merge two or more files that have common information on the Plot or Table Sheet, the common information is duplicated.

		Powers of n				
List	n	1	1.11111111	1.22222222	1.33333333	1
n <sup>1</sup>	1	1.11111111	1.22222222	1.33333333	1.33333333	1
n <sup>2</sup>	1	1.23456790	1.49382716	1.77777778	1.77777778	2
n <sup>3</sup>	1	1.37174211	1.82578875	2.37037037	2.37037037	3

Figure 15-4

### Plot and Table with TRAVEL3

Now that you've worked with the Plot and Table Sheets, try using them with the lists in TRAVEL3.

Reset the program and load the TRAVEL3 model.

The two lists currently associated with TRAVEL3 are the list for mileage, *mlg*, and the list for speed, *s*.

Display the Plot Sheet in a single window and set up a plot titled **Mileage related to Speed** with the scale values turned off. Specify the *x*-axis as the list *mlg* and the *y*-axis as the list *s*. Type the Action command (!).

The screen should look like Figure 15-5.

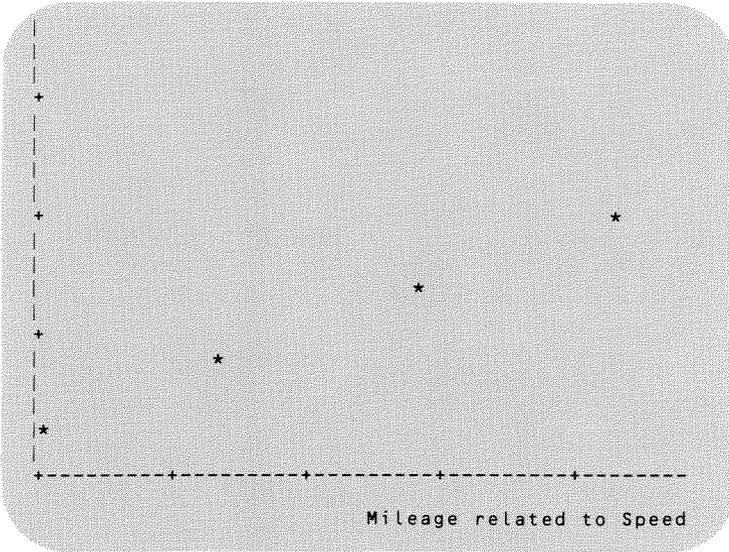
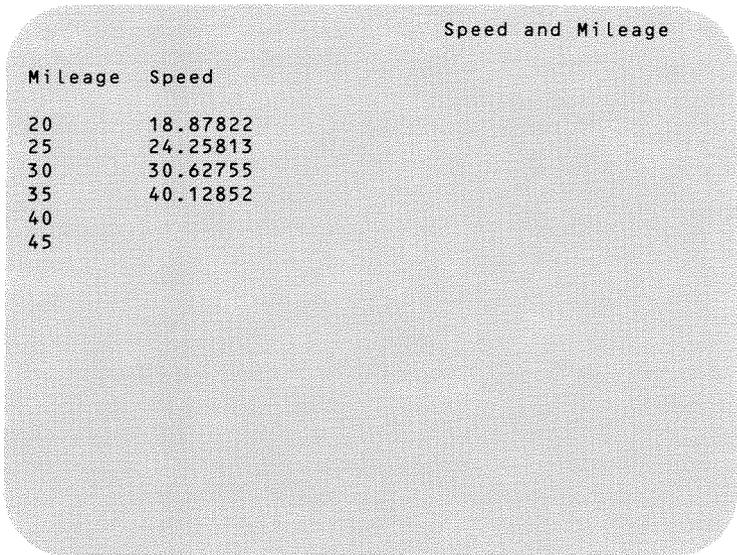


Figure 15-5

Return to the Plot Sheet by typing  $\diamond$  .

Display the Table Sheet and set up a vertical table, titled **Speed and Mileage**, showing the same two lists, *mlg* first and then *s*, in columns 8 characters wide beginning with the third value on each list. Give *mlg* the heading **Mileage** and *s* the heading **Speed** . Type the Action command (!).

The screen should look like Figure 15-6.



Mi Leage	Speed
20	18.87822
25	24.25813
30	30.62755
35	40.12852
40	
45	

**Figure 15-6**

Return to the Table Sheet by pressing  $\blacklozenge$ . If you save TRAVEL3 now, the plot and table settings will be saved as part of the model.

### **Conclusion**

This chapter has shown you how to create plots and tables using the Plot and Table Sheets. The next chapter describes how to create and use your own functions.

## Chapter 16 User Functions

- Functions
- Function syntax in the TK!Solver program
- The User Function Sheet
- Creating a function
- The User Function subsheet
- The Storage command (/S)— Save Function option (F)
- Applying TK!Solver user functions
- Interpolation

The TK!Solver program provides many built-in functions, such as SIN, PI, and APPLY, but you can also use your own functions. You can create functions, using lists for both the domain and the range, by entering particulars on the User Function Sheet. Three types of mapping are available: Table, Linear, and Step. Once a function has been created, it can be used in expressions and for plots and tables.

This chapter describes what a function is and explains how to use the User Function Sheet to define your own functions and to define other relationships between lists. It also shows you how to make plots and tables with a user function. See the Reference Manual for details about the TK!Solver built-in functions.

---

### To begin this chapter:

Reset the program.

### Functions

This section of the chapter contains a brief discussion about what a function is. If you know what functions are, you may want to skip the next few paragraphs.

A function is a relationship between two sets of values. The two sets are called the domain and the range. Each element in the domain is paired with one, and only one, element in the range. This relationship is arbitrary and may be as simple as  $a = b$ .

For example, look at the sets of values below. The relationship between the sets of values in Example 1 defines a function, but the relationship shown in Example 2 does not.

**Example 1**

domain	range
Set A	Set B
1.....2	
2.....5	
3.....6	
4.....6	

**Example 2**

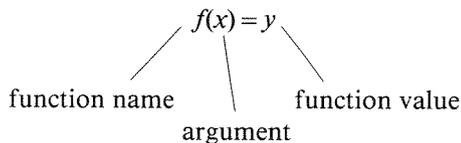
domain	range
Set X	Set Y
1.....8	
1.....4	
2.....9	
2.....5	

In Example 1, each element of the domain, Set A, has a unique match with an element in Set B, the range. Elements in the range may be associated with more than one value in the domain.

In Example 2, each element of the domain, Set X, is matched with more than one element in the range, Set Y. Example 2 is not a function because each element in the domain is not matched uniquely with an element in the range.

**Function Syntax and Evaluation**

There are many different types of functions and many different ways to express them. Below is a common syntax for expressing functions. This is the syntax used by the TK!Solver program:



The function name identifies the function, and, like variable and list names, it must begin with a letter and may contain only letters, numbers, and the special characters @, #, \$, %, and \_.

The argument is a value from the domain; the function value is a value from the range. The argument and function values can be any TK!Solver expression, including symbolic values. If you use a symbolic value, it must begin with an apostrophe (') so that the program recognizes it as a symbolic value.

The TK!Solver program uses this syntax for both built-in functions and user functions.

A function typically evaluates the argument (domain value), returning the function value (range value). It sometimes can be useful to invert this relationship and evaluate the function value, returning the related domain value. When an inversion is possible, the function is said to have an inverse. If values in the range are associated with more than one domain value, this inverse relationship cannot occur.

## The User Function Sheet

Change the screen to one window, and then select the User Function Sheet by typing:

```
= F
```

The screen should look like Figure 16-1.

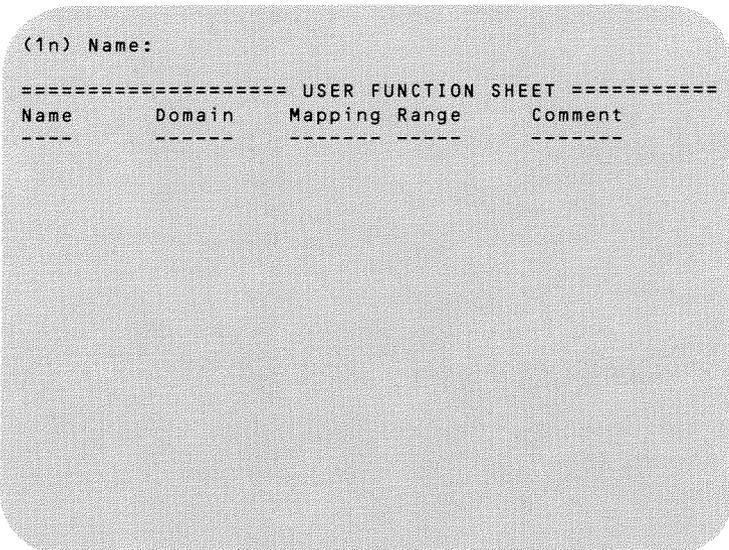


Figure 16-1

The User Function Sheet lists the user functions and displays some of their characteristics.

The Name field displays the name of the user function, and the Domain field displays the name of the list used for the domain values. The Mapping field, an option field, shows the type of relationship between the domain and the range used in the function. The Range field contains the name of the list used for the range values.

### Creating a Function

You can create different types of functions using the User Function Sheet. The examples in this section are of three typical types of functions:

- A discrete function—one that has a finite number of arguments and values.
- A linear function—one that takes the form  $a * x + b = y$ .
- A step function—one that associates an interval of domain values with a constant.

First you will create the discrete function *disf*. The function *disf* will relate the symbolic and alphabetic representations of a set of numbers.

Enter the function name **disf** in the first field of the Name column. The Mapping field in this row automatically displays the value **Table**. **Table** is the default value for this field and is displayed when a function is named. The Mapping field has three options—Table, Step, and Linear—entered by typing the initial letter of the mapping. These options are discussed as they are encountered throughout this chapter. The default option, Table, is the one used for the function *disf*.

The Table option matches elements within the two associated lists, element for element. Only the explicit values in the domain and range lists can be argument and function values. There is no interpolation.

Enter **a** as the name of the Domain List and **number** as the name of the Range list. Enter **discrete function** in the Comment field.

The screen should look like Figure 16-2.

```
(1c) Comment: discrete function

===== USER FUNCTION SHEET =====
Name      Domain    Mapping Range    Comment
-----
disf      a           Table  number    discrete funct
```

Figure 16-2

## The User Function Subsheet

Like the Variable and List Sheets, the User Function Sheet has subsheets. Dive to the User Function subsheet for *disf* by typing the Dive command (>).

A User Function subsheet is similar in appearance to a List subsheet, although its purpose is different. The four fields at the top of the sheet contain items from the Comment, Domain, Mapping, and Range fields on the User Function Sheet. The rest of the sheet contains the element numbers and values for the two lists that make up the domain and range of the function.

The columns labeled **Domain** and **Range** contain the values used for the function. Lists can be created by entering values here, but the lists must be named before values can be entered. (You named the lists when you entered them on the User Function Sheet.) As on a List subsheet, the Element column contains the element number of the corresponding values. The program generates the element number automatically when a value is entered in one of the lists. As you can see, the cursor moves automatically to the first field in the column labeled **Domain**.

On a User Function subsheet, the Action key (!) calls the Fill List feature. Enter a list of five values from **10** to **50** for the domain list *a*.

The values for the range of the function *disf* are symbolic values, so you must type an apostrophe before the value. Enter these values for the range list *number*:

Element	Range Value
1	'twenty
2	'forty
3	'sixty
4	'eighty
5	'one__hundred

Before going on, you need to save this function, because it is used later in an example. To do this, use the Save Function (F) option of the Storage command (/S). This option saves the current User Function Sheet and User Function subsheets. Type:

```
/SF
```

As for all the other Storage command (/S) options, you are prompted for a filename:

```
Function Save: Filename:
```

Enter the filename `DISF`, and the program saves the function.

Return to the User Function Sheet. Delete the function *disf* from the User Function Sheet by deleting the row that contains the functions. Use the Delete command (/D).

The two other functions to be used as examples are stored on the Instruction diskette in the file `FUNC`. Load the file `FUNC`. The screen should look like Figure 16-3. The names of the two new functions, *linearf* and *stepf*, indicate the type of functions they are.

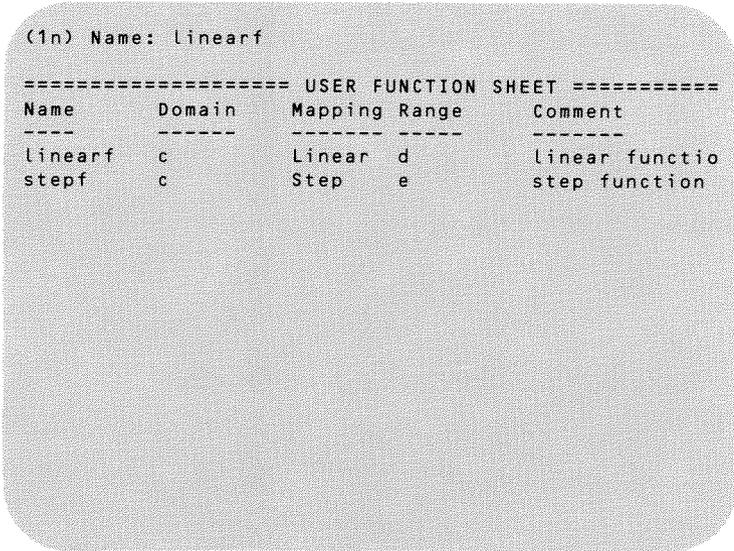


Figure 16-3

## Tables and Plots of Functions

One way to describe the relationship between a domain and range expressed as a function is by looking at tables and plots of the functions. You can do this with the Table and Plot Sheets. This section looks at the two new functions, *linearf* and *stepf*, using tables and plots.

Step functions associate an interval of domain values with a constant. Linear functions associate range and domain values using the equation below, with  $x$  representing the domain value,  $y$  the range value, and  $a$  and  $b$  the constants defining the relationship between the domain and range:

$$a * x + b = y$$

To show the plot of a function or make a table of its values, use the lists that are the domain and range of the function. Notice that the two sample functions use the same list for the domain.

Display the Table Sheet in the window in place of the User Function Sheet, then enter the title **Functions** and the domain and range lists for the two functions as shown below. Because they share a common domain list, you need to enter that list only once.

List	Width	First	Header
c	10	1	Domain
d	15	1	Linearf Range
e	15	1	Stepf Range

Display the table by typing the Action key (!). Your screen should look like Figure 16-4.

Functions		
Domain	Linearf Range	Stepf Range
10	20	20
12	24	20
14	28	20
16	32	20
18	36	20
20	40	40
22	44	40
24	48	40
26	52	40
28	56	40
30	60	60
32	64	60
34	68	60
36	72	60
38	76	60
40	80	80
42	84	80
44	88	80

Figure 16-4

As you can see, for every value in the domain there is a different value under **Linearf Range**, but the values under **Stepf Range** are constant for intervals of several elements in the domain.

Type  $\blacklozenge$  to see the second page, then return to the Table Sheet by typing  $\blacklozenge$  again.

Both linear and step functions take their names from the graphs which represent them. Let's look at the plot for the linear function first.

Display the Plot Sheet and enter the title **Linear Function**. Enter the lists c for the x-axis and d for the y-axis, then display the plot by typing the Action key (!).

As you can see, the plot is essentially a straight line, hence a linear function. Because of the way the TK!Solver program displays graphs, there are slight variations from expected plots, such as the slight bends seen in the example.

Type  $\blacklozenge$  to return to the Plot Sheet and set the plot for *stepf* using the same method. Change the title to **Step Function**, delete the list *d* from the Y-axis field, and enter the list name *e* instead. Type the Action key. This time the plot shows plateaus that look like the outline of a set of steps.

Type  $\blacklozenge$  to return to the Plot Sheet.

If you want to compare two or more functions on one plot, you can do so only if they use the same domain. This can be done with the two sample functions by adding the list *d* to the Plot Sheet in a Y-axis field and changing the plotting character for one of them.

### Using User Functions

This section shows you how to use User Functions, with the three functions *disf*, *linearf*, and *stepf* as examples. Display the User Function Sheet and load the file DISF so that all three functions are again present on the User Function Sheet. The list used for the domain in function *disf* can be used to show the functions *linearf* and *stepf* more clearly.

The domain for the three functions is 10 through 50, but the lists used, *a* and *c*, have slightly different forms; *a* has only five values and *c* has 21. For this example, we will use the shorter form.

Change the domain for both *linearf* and *stepf* to *a* by changing the list name in their domain fields. Dive to the subsheet for *linearf*, move the cursor to the Range column, and dive again so that the List subsheet for *d* is displayed. Use the Delete command (**/D**) with the Goto command (**:**) to delete the values in the list, and use the Action command (**!**) to enter five new elements with the values **20** to **100**.

Type the Return command (**<**) twice to return to the User Function Sheet, then repeat the above procedure to change the range values for *stepf*. You have not actually changed the extent of the range for either function because the mapping remains the same.

Now all the lists have five elements, the three functions have the same domain, and *linearf* and *stepf* have the same range. The model given below uses all three functions.

Display the Variable and Rule Sheets on a split screen with the Variable Sheet in the top window and the Rule Sheet in the bottom window. Enter the following three rules:

```
disf(a)=b
linearf(a)=d
stepf(a)=e
```

Give the variable *a* a value of 20 and solve the model. The screen should look like Figure 16-5. The output values for the variables *d* and *e* are the same, even though the mapping of the two functions is different. In their current forms, there is no difference between the functions *linearf* and *stepf* when their domain values are limited to explicit values in the lists. In the section below on interpolation, you will see these two functions behaving differently.

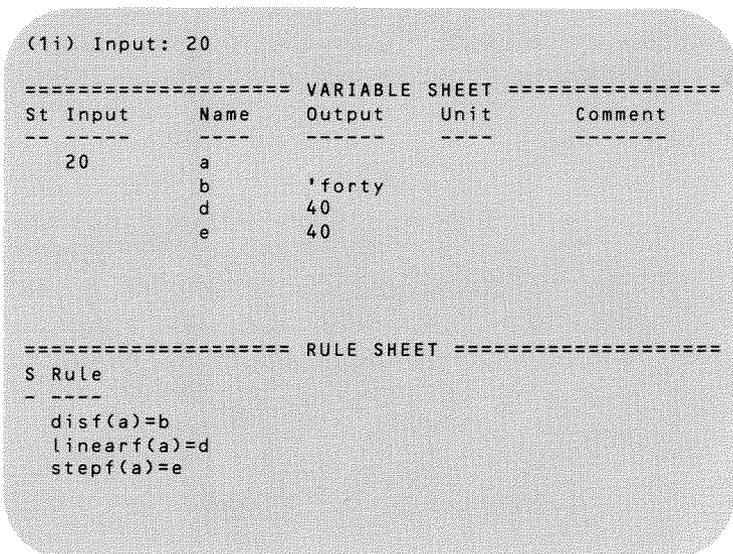


Figure 16-5

User functions can also be solved as inverses. Reset the variable values. Because all three functions have the same domain, you can solve for all their inverses simultaneously.

Enter the value 'sixty for variable  $b$ , and the value 60 for both variables  $d$  and  $e$ . Solve the model. The domain for all is the same, so there is no conflict in finding  $a$  and all three equations have been satisfied.

## Interpolation

Interpolation is a way of estimating a value between two known values of a function's range or domain. The Mapping field on the User Function Sheet shows the type of interpolation used for the function.

The Table option is for functions that do not require interpolation. Using this option, the function operates with only the values that have actually been entered on the domain and range lists, using them like a table of values.

The Linear option interpolates proportionally, finding values between the given values.

The Step option interpolates like a step function, returning a constant for a given interval of domain values.

This example shows interpolation. Reset the variable values.

The three functions use the three mapping options. Because *disf* uses table mapping, it will not be used for interpolation, so delete the first rule, the one using *disf*, from the Rule Sheet.

Give variable  $a$  an input value of 25 and solve the model. The screen should look like Figure 16-6.

The value for  $d$  was found using linear interpolation by solving this proportion for  $x$ :

$$\begin{array}{l} \text{Domain Values} \\ \hline 25 \\ 30 \end{array} = \begin{array}{l} \text{Range Values} \\ \hline x \\ 60 \end{array}$$

The values used in this equation are the corresponding elements in the domain and range.

```
(1i) Input: 25

===== VARIABLE SHEET =====
St Input      Name      Output    Unit      Comment
-----
      25       a
              b
              d         50
              e         40

===== RULE SHEET =====
S Rule
-----
  linearf(a)=d
  stepf(a)=e
```

Figure 16-6

The value for variable *e* was found using step mapping and is the range value for the domain interval from 20 to 30.

TK!Solver interpolation assumes that no more than two contiguous values are used for interpolation. This means that when you set up functions, you must enter the range and domain values in ascending or descending order, otherwise the interpolation will not be what you might expect.

For more details, please consult the Reference Manual.

### Applying TK!Solver User Functions

Now let's look at the MORTGAGE and TRAVEL models to see how functions can be applied. Reset the program.

#### User Functions with MORTGAGE

In Chapter 13 you created two lists for the MORTGAGE model to be used with functions. These lists were *i*, containing interest rates, and *B*, containing bank names. In this example, these two lists are used to create a function that compares the interest rates associated with each bank.

Load the MORTGAGE model and reset the variable values. The Variable Sheet should be in a single window.

You are shopping for a 25-year mortgage on a \$97,000 house with 15% down, and you want to compare the interest rates offered by several banks.

The list associations have remained. In this example, the variables  $i$  and  $A$  use lists. Disassociate the list for  $dp$  by typing **L** in the Status field.

In this example, the list  $i$  will be associated with the variable  $i$ . When a list is used for either the domain or range of a user function, and that list is associated with a variable, and the variable and list both have units, the Calculation unit of the variable and the Display unit of the list must be the same.

Split the screen, displaying the List Sheet in the bottom window, and dive to the List subsheet for the list  $i$ . Change the Display unit to **monthly rate** so that it is the same as the Calculation unit for the variable  $i$ .

With the User Function Sheet in the top window, create the function **bank** with list **B** as the domain and list **i** as the range using Table mapping. This function takes bank names as arguments and returns interest rates.

Because the program identifies the function by the function name, the arguments and function values used in a rule do not have to be the list names of the domain and range but can be any variable or list name. The following rule uses the function *bank* with the argument *bank\_\_name*. With the Rule Sheet in the bottom window, enter the new rule:

```
bank(bank__name)=i
```

Change the display to a single window containing the Variable Sheet so that you can see all the variables. Enter the values from the problem shown above along with the symbolic value **'First\_\_National** for *bank\_\_name*.

The variables  $i$  and  $A$  will have output values in lists  $i$  and  $A$ , respectively. The variable *bank\_\_name* is associated with the input list  $B$ . You must dive to the Variable subsheet for *bank\_\_name* to name the associated list,  $B$ .

Redisplay the Variable Sheet. The screen should look like Figure 16-7. Solve the model using the List command (/L).

```
(10i) Input: 'First_National

===== VARIABLE SHEET =====
St Input      Name      Output      Unit      Comment
-----
      97000    H         dollars    House price
              d         dollars    Down paymen
      15       dp        percent    Down paymen
              p         dollars    Mortgage
      25       n         years      Term
L          i         percent    Interest ra
L          A         dollars    Monthly pay
              T         dollars    Total of pa
              Tip        dollars    Total inter
L 'First_Na bank_na
```

Figure 16-7

Look at the results by making a table of the lists *i*, *A*, and *B*. Save the model MORTGAGE.

### User Functions with TRAVEL

In the last few chapters, the different versions of the TRAVEL model have been used for a hypothetical car. Now we will modify the TRAVEL2 model to accommodate three different types of car—economy, midsize, and sedan. The functions show how the type of car affects the gas mileage.

On a 500-mile trip, how do the mileage and cost compare for the three different types of cars if they are driven at 40 miles per hour?

You will use the List command (/L) for this problem. Reset the program and load the file TRAVEL2, then load the file TFUNC that contains the four functions needed for this problem. Reset the variable values.

Display the User Function Sheet in the top window. The first three functions, *econo*, *mid*, and *sedan*, associate speed and mileage for each type of car.<sup>1</sup> The remaining function, *car*, selects the type of car used to solve the model.

This problem requires two new rules to determine mileage as a function of car type. Display the Rule Sheet in the bottom window, then add the two new rules to those already on the sheet:

```
car(type)=ty
apply(ty,s)=mlg
```

The first equation uses the function *car* with Table mapping to identify the type of car you want to solve the model for. In the top window, dive to the User Function subsheet for *car*.

As you can see, the range for *car* comprises the names of the three functions that determine mileage. The value for the variable *type* must be 'E', 'M', or 'S', so that the function *car* can select the function to be used for the solution.

The second new equation uses the built-in function *apply* to associate the car type with the function that has been selected to determine mileage. The built-in function *apply* has the syntax:

*APPLY(function name, domain value)*

It searches for the function named and evaluates that function for the given domain value, returning the appropriate range value.

In the equation above, the value for the variable *ty* is the name of the function to be evaluated. The variable *s* is the argument of the selected function.

Because this problem uses these two rules, the two old rules used to determine mileage should be deleted. Delete the rules:

```
gas=t*power
f+(w*s^3)=power
```

Change the screen to one window and display the Variable Sheet so that you can see all the variables. Notice that although the

<sup>1</sup> The values used for these functions are taken from United States Environmental Protection Agency publication 460/3-80-010 on Passenger Car Fuel Economy, September 1980.

rules containing the variables *power*, *f*, and *w* were deleted, the variable names are still on the Variable Sheet. If you want to save this version of the TRAVEL model, name it TRAVEL4.

Because we want to examine how the mileage for the three types of cars affects cost, we will solve the model using the List command (/L).

Enter the input values for the variables shown below and solve the model:

Value	Variable
500 mi	<i>d</i>
40 mi/h	<i>s</i>
\$1.29	<i>pr</i>
'M	<i>type</i>

Associate the variables *ty*, *mlg*, and *cost* with output lists. Associate the variable *type* with the input list *t*, which contains the values 'E', 'M', and 'S', by changing the name of the associated list on the subsheet for the variable *type*.

Display the Variable Sheet again. The screen should look like Figure 16-8.

(11s) Status: List

===== VARIABLE SHEET =====					
St	Input	Name	Output	Unit	Comment
-----					
L		mlg		mi/gal	mileage
	500	d		mi	distance
		gas		gal	amount of g
	40	s		mi/h	speed
		t		h	time
L		cost		\$	cost of tri
	1.29	pr		\$	price of ga
		power		hp	energy comp
		f		hp	internal fr
		w		hp*h^3/mi	wind resist
L	'M	type			
L		ty			

Figure 16-8

Solve the model using the List command (*/L*). To see the results, set the Table Sheet to show a table of the lists *ty*, *mlg*, and *cost*.

To see how speed affects the results, solve the model with a different value, or a list of values, for *s*. Remember to rename the current lists if you want to save the current values.

### Conclusion

This chapter has discussed user functions, what they are, and how to use and apply them. The next chapter is a summation of the TK!Solver Instruction Manual.

## **Chapter 17 Summation**

This is the last chapter in the Instruction manual. You have used and created several models throughout this manual. While working with these models, you have used most of the features of the TK!Solver program. The models you develop may not evolve into complex models like the TRAVEL model did, but the quality of a model is based on its utility, not its size or complexity. The TRAVEL model, for example, grew to accommodate and demonstrate the various features of the program. You may find instead that many of your models require no more than the Variable, Rule, and Unit Sheets.

TK!Solver models are more than a mathematical statement of a problem. These models introduce a new dimension to the problem-solving process. Before the TK!Solver program was available, it was difficult to formulate problems that required a solution for more than one variable or instance. TK!Solver models allow you to easily manipulate all the variables of the problem, solving immediately for a variable or combination of variables.

A model begins with the statement of a problem in terms of the relationships of variables involved in that problem. These relationships can be explored in many ways—by changing input values, by changing the relationships of the variables, by refining and expanding the rules, by changing units, by experimenting with and comparing lists of values—by altering and stipulating a variety of conditions.

The equations used in building models for the TK!Solver program can be entered as they are formulated. They do not need any special formulation, nor do they need to be entered in a specific order, and you can change the units associated with any variable without having to alter an equation.

As you use the TK!Solver program with your own models to solve your own problems, you will find shortcuts and develop special ways of using the program that will mold it to your specific needs.

The building and expansion of TRAVEL is an example of how a model can be created to express a problem and refined to explore the possibilities of other problems related to the same variables.

The first version of the TRAVEL model that you used was a simple one expressing relationships between speed, distance, time, and cost. This model was expanded by adding two new equations to look more closely at the way mileage was affected by mechanical factors in a hypothetical car. By adding unit conversion definitions, input and output values could be used in any chosen unit.

More use was made of the TK!Solver features as you explored the effects of changes made to the hypothetical car. The use of lists gave the advantage of multiple answers to multiple input possibilities. The use of functions enabled you to interrelate variables so that the model could be used with a wide range of input values.

The TK!Solver program helps you to explore solutions to problems in a more flexible environment than has ever been possible. Now that you have learned how to use the program, you can begin to define and construct your own models. The TK!Solver package includes a Reference Card, a Help Facility, and a Reference Manual, all of which can be used to find answers to questions that may arise while you are using the program.

To complement the TK!Solver program, Software Arts, Inc. has developed a line of TK!SolverPacks, packages of models for specific applications. These are available for many fields, interests, and needs.

Each TK!SolverPack contains a variety of complete models for a specific application, and you can modify and personalize these models to fit your requirements. Now that you are familiar with the TK!Solver program, you may want to use the TK!SolverPack that best applies to your specific needs. TK!SolverPacks are available from software dealers.

Software Arts, Inc. supports users of the TK!Solver program with the publication TK!SATN, the journal for TK!Solver users. For more information, write to: TK!SATN Information, Software Arts, Inc., 27 Mica Lane, Wellesley, MA 02181.

**Index**

- ! 4-4, 7-2, 13-7, 15-4, 15-7, 16-6
- : 5-3
- ; 5-3
- = 5-5
- ? 4-5
  
- Action command (!) 7-2, 13-7, 15-4, 15-7, 16-6
- arithmetic symbols 5-10
- arrows 5-3
- automatic iteration 12-15
  
- Blank command (/B) 7-6
- blank option, Variable Sheet 6-12
- blank row 7-9
- block solving 14-10, 14-13
  
- Calculation unit 6-9, 8-1
- change windows 5-3
- clear program 5-11
- clear sheet 7-9
- clear values 7-10
- command 1-2
  - Action (!) 7-2, 13-7, 15-4, 15-7, 16-6
  - Blank (/B) 7-6
  - Copy (/C) 8-6
  - Delete (/D) 7-8
  - Dive (>) 6-13
  - Edit  chapter 2, 7-17
  - Edit Field (/E) 7-16
  - Goto (: ) 5-3
  - Insert (/I) 7-13
  - List (/L) 14-2, 14-10, 14-13, 14-15
  - Move (/M) 8-3
  - Print (/P) 9-6
  - Quit (/Q) chapter 3
  - Reset (/R) 5-11, 7-8, 7-10
  - Return (<) 6-15
  - Select (=) 5-5
  - Solve (/!) 8-9
  - Storage (/S) 4-7, 4-8, 4-9, 4-11, 7-5, 8-10, 13-12, 14-9, 16-6
  - Switch (;) 5-3
  - Window (/W) 5-6
- comparison tolerance 12-11
- conversions, unit 6-15, 8-1
- Copy command (/C) 8-6
- create function 16-4
- create list 13-2
- cue 1-2, chapter 2
- cursor 1-2, chapter 2, 5-3
  
- Delete command (/D) 7-8, 10-1
- delete file 4-11
- DIF file 13-5, 13-11, 14-9
- Direct solution, conditions for 10-1
- Direct Solver 10-1
- directory scroll 4-11
- display new sheet 5-5
- display print file chapter 2
- Display units 6-9, 8-1, 13-3
- Dive command (>) 6-13
- domain 16-2
  
- Edit command  chapter 2, 7-17
- Edit Field command (/E) 7-16
- Editor chapter 2, 7-16, 7-17
- enter, equations 6-3
- enter, variables 6-5
- entry field 6-3
- equations 5-8
- error signal chapter 2
- extensions chapter 2
  
- field 4-2
  - blank 7-6
  - copy 8-6
  - types 6-2
- file 1-2
  - DIF 13-5, 13-11, 14-9
  - merge 6-6, 7-8
  - print 9-6

- filename extensions chapter 2
- fill list 13-7
- functions 5-10, 16-1
  - argument 16-2
  - create 16-4
  - inverse 16-3
  - mapping 16-4
  - name 16-2
  - save file 16-6
  - syntax 16-2
  - type 16-4
  - use 16-2
  - value 16-2
- Global Sheet 6-5, 9-1, 12-9
- glossary 1-2
- Goto command (:) 5-3
- graph 15-2
- guess, first 11-3, 12-2
- guess option (G) 12-2
- Help Facility 4-5
- Help File 3-1
- highlight 1-2
- input values 6-8
- Insert command (/I) 7-13
- interpolation 16-11
- inverse 16-3
- iteration 12-7, 12-8
  - automatic 12-15
  - concept 11-1
  - conditions for 10-6
  - divergence and oscillation 12-8
  - slow convergence 12-7
- Iterative Solver 12-1
- key symbols chapter 2
- List command (/L) 14-2
  - block option (B) 14-10, 14-13
  - get option (G) 14-15
  - put option (P) 14-15
  - solve option (!) 14-3
- list
  - create 13-2
  - name 13-2
  - fill 13-7
- List Sheet 13-1
- List subsheet 13-4
- literal values 13-8
- load DIF file 14-9
- load file 1-3, 4-7, 4-8
- Memory Indicator 4-4
- merge files 6-6, 7-8
- Message Area 4-3
- Move command (/M) 8-3
- move cursor 5-3
- name
  - function 16-2
  - list 13-2
  - unit 6-16, 8-10
  - variable 6-9
- operators 5-10
- option field 6-2
- output field 6-2
- Overlay File chapter 3
- overwrite file 4-10
- Plot Sheet 15-2
- Position Indicator 4-3
- precedence 5-10
- Print command (/P) 9-6
- print file 9-6
- printer fields 9-3
- Prompt/Error Line 4-3, 4-4
- prompts 1-3
- Quit command (/Q) chapter 3
- range 16-2
- Reset command (/R) 5-11
  - All option (A) 5-11
  - Sheet option (S) 7-8
  - Variable values option (V) 7-10
- Return command (<) 6-15

- row
  - delete 7-9
  - insert 7-13
  - move 8-3
- rules 5-8
- Rule Sheet 5-8
- save file 4-7, 4-9
  - DIF 13-12
  - function 16-6
  - model 4-7
  - units 8-10
  - variables 7-5
- scientific notation 5-10
- screen 4-3
- scroll 4-11
- search 5-4
- Select command (=) 5-5
- Sheet 4-2, 5-5
  - Global 6-5, 9-1, 12-9
  - List 13-1
  - List subsheet 13-4
  - Plot 15-2
  - Rule 5-8
  - Table 15-5
  - Unit 6-15, 8-2
  - User Function 16-3
  - User Function
    - subsheet 16-5
  - Variable 6-7
  - Variable subsheet 6-13, 12-15
- Solution Indicator 4-4
- Solve command (!) 8-9
- solver
  - Direct 10-1
  - Iterative 12-1
- special commands chapter 2
- special keys chapter 2
- split screen 5-6
- Status field options, Variable
  - Sheet 6-12, 7-7
- Status Line 4-3
- Storage command (/S) 4-7
  - Delete file option (D) 4-11
  - Load DIF file option (#L) 14-9
  - Load file option (L) 4-8
  - Save DIF file option
    - (#S) 13-12
  - Save function option
    - (F) 16-6
  - Save model option (S) 4-9
  - Save units option (U) 8-10
  - Save variables option
    - (V) 7-5
- Storage unit 13-3, 13-5
- subsheet
  - Dive down 6-13
  - List 13-4
  - Return up 6-15
  - User Function 16-5
  - Variable 6-13, 12-15
- Switch command (;) 5-3
- symbolic values 13-8
- Table Sheet 15-5
- unit conversions 6-15, 8-1
- unit names 6-16, 8-10
- units 6-9, 6-14, 8-1, 13-3, 14-5
- units, save 8-10
- Unit Sheet 6-15, 8-2
- unsolved model 4-4
- User Function Sheet 16-3
- User Function subsheet 16-5
- values 6-8, 13-8
- variable insert 6-5
- variable names 6-9
- Variable Sheet 6-7
- Variable Sheet Status
  - field 12-1
  - blank option 6-12
  - guess option (G) 6-12
  - input option (I) 6-12, 7-7
  - list option (L) 6-12, 14-4
  - output option (O) 6-12
- Variable subsheet 6-13, 12-15
- window 5-2
- Window command (/W) 5-6







---

# ***TK!Solver™ Program Reference Manual***

---



***Software Arts™***

Copyright © 1983 Software Arts, Inc.  
All rights reserved.

**Software Arts**

27 Mica Lane  
Wellesley, MA 02181

Printed in the U.S.A.

TK, TK!, TK!Solver, and TK!SolverPack  
are all trademarks of Software Arts, Inc.

Software Arts is a trademark of Software  
Arts, Inc. and Software Arts Products  
Corp.

The TK!Solver program is a product of  
Software Arts, Inc. which is solely  
responsible for its contents.

**Table of Contents**

<b>Contents</b> .....	i
Manual Summary .....	i
Manual Conventions .....	iii
Alphabetic Cross-reference .....	v
Functional Cross-reference .....	viii
<b>Part I: Program Overview</b> .....	I-1
Program Startup .....	I-1
TK!Solver Screen .....	I-2
Movement Through the Program .....	I-10
Program Environments .....	I-12
Field Types .....	I-16
Error Conditions .....	I-22
<b>Part II: Sheets</b> .....	II-1
Variable Sheet (V) .....	II-3
Variable Subsheets (V>) .....	II-8
Rule Sheet (R) .....	II-11
Unit Sheet (U) .....	II-13
Global Sheet (G) .....	II-16
List Sheet (L) .....	II-23
List Subsheets (L>) .....	II-26
User Function Sheet (F) .....	II-30
User Function Subsheets (F>) .....	II-34
Plot Sheet (P) .....	II-38
Table Sheet (T) .....	II-42
<b>Part III: Commands</b> .....	III-1
Left Arrow (←) .....	III-3
Right Arrow (→) .....	III-3
Up Arrow (↑) .....	III-3
Down Arrow (↓) .....	III-3
Enter (◆) .....	III-4
Break (⊗) .....	III-5
Backspace (◀) .....	III-5
Action (!) .....	III-6
Blank (/B) .....	III-10
Copy (/C) .....	III-12
Delete (/D) .....	III-14
Dive (>) .....	III-16
Edit (☒) .....	III-18

# Table of Contents

---

Edit Field (/E) .....	III-21
Goto (:)	III-23
Help (?)	III-26
Insert (/I)	III-29
List (/L)	III-30
Move (/M)	III-34
Print (/P)	III-36
Quit (/Q)	III-38
Reset (/R)	III-39
Return (<)	III-41
Select (=)	III-42
Solve (/!)	III-44
Storage (/S)	III-45
Switch (;)	III-51
Window (/W)	III-52
<b>Part IV: Mathematics</b> .....	IV-1
Equation Components .....	IV-1
Expression Evaluation .....	IV-6
Model Solution .....	IV-7
Model Development Overview .....	IV-13
Model Development Using Sheets .....	IV-14
Model Development Using TK!Solver Features .....	IV-16
<b>Part V: Built-In Functions</b> .....	V-1
<b>Part VI: Hardware Reference</b> .....	VI-1
<b>Appendices</b> .....	A-1
TK!Solver Files .....	A-1
Unit Name Conventions .....	A-4
Prompt Messages .....	A-7
Status Messages .....	A-17
Error Messages .....	A-22
<b>Glossary</b> .....	G-1
<b>Index</b> .....	Index-1

**Manual Summary**

The TK!Solver™ Reference Manual contains ten divisions: Contents, Program Overview, Sheets, Commands, Mathematics, Built-in Functions, Hardware Reference, Appendices, Glossary, and Index. The following summary indicates the topics discussed in each section.

**Contents**

An introduction to the TK!Solver Reference Manual, including Table of Contents, Manual Summary, Manual Conventions, and Alphabetic and Functional Cross-references to program features.

**Part I: Program Overview**

A general description of the program and its features, including Startup, Screen, Movement, Program Environments, Field Types, and Error Conditions.

**Part II: Sheets**

A detailed description of the Sheets and Subsheets in the program.

**Part III: Commands**

A detailed description of the Special Keys and Commands in the program.

**Part IV: Mathematics**

A detailed discussion of the Mathematical features of the program, including Equation Components, Expression Evaluation, Model Solution, and Model Development.

**Part V: Built-in Functions**

A detailed list of the Built-in Functions available with the program.

### **Part VI: Hardware Reference**

A detailed description of the parts of the program that are computer dependent, including hardware requirements, keyboard, disk drives, printers, system notes, file specifications, global settings, printer setup strings, and size constraints.

### **Appendices**

Additional information about the TK!Solver program, including TK!Solver Files, Unit Name Conventions, Prompt Messages, Status Messages, and Error Messages.

### **Glossary**

A definition of terms used in all of the TK!Solver documentation.

### **Index**

An alphabetical index to the Reference Manual.

## Manual Conventions

Several conventions are used in the TK!Solver Reference Manual to identify special kinds of text:

- Important terms are indicated in boldface the first time they are used.
- The typeface “OCR-B” is used to indicate exact program output:

`This sentence is set in OCR-B.`

- A class of program output or user input is indicated in italic.
- Variable names and list names are indicated in italic.

Special conventions have been used to refer to the use of keys that vary from keyboard to keyboard. These keys, or combinations of keys, are represented by both symbols and names in the manuals: the symbol is listed below on the left; the name is on the right. See the Hardware Reference section for which key or combination of keys to use on your keyboard.

- ← Left Arrow
- Right Arrow
- ↑ Up Arrow
- ↓ Down Arrow
- ◆ Enter
- ⊗ Break
- ◀ Backspace
- ☒ Edit

User input in the Reference Manual is usually indicated in uppercase. On computers that allow uppercase and lowercase, either case may be used. Program output is described as it appears on computers using both cases.

## **Manual Conventions**

This manual is described for video screens that display 80 characters wide by 24 lines high. Some adjustments are made by the program to accommodate for other size screens, but these changes are not described within this manual since they do not greatly affect use of the program other than error messages and the Help Facility text. However, depending on the computer used, screen pictures in the manuals may differ from what appears on the user's screen.

Program data referred to as entered by the user throughout the documentation can also be loaded into the program from storage files.





**Alphabetic Cross-reference**

An alphabetic index of commands and sheets:

**Commands**

<b>Action</b> (!)	Performs the action appropriate to the current sheet, including model solution, list value fill-in, plot or table display.	<b>III-6</b>
<b>Blank</b> (/B)	Erases the current field item, or a column of items beginning with the current item.	<b>III-10</b>
<b>Copy</b> (/C)	Copies the current field item to the specified destination.	<b>III-12</b>
<b>Delete</b> (/D)	Deletes the current row, or a block of rows beginning with the current row.	<b>III-14</b>
<b>Dive</b> (>)	Displays a dependent subsheet for the current sheet or subsheet in the current window.	<b>III-16</b>
<b>Down Arrow</b> (↓)	Moves the cursor down one row.	<b>III-3</b>
<b>Edit</b> (☑)	Edits a user response on the Prompt/Error line or an unentered item in the current entry field.	<b>III-18</b>
<b>Edit Field</b> (/E)	Edits the entered item in the current entry field.	<b>III-21</b>
<b>Goto</b> (:)	Moves the cursor to the specified field on the current sheet.	<b>III-23</b>
<b>Help</b> (?)	Calls the Help Facility.	<b>III-26</b>
<b>Insert</b> (/I)	Inserts a blank row at the current row.	<b>III-29</b>
<b>Left Arrow</b> (←)	Moves the cursor left one field.	<b>III-3</b>
<b>List</b> (/L)	Solves a model for lists of values associated with variables, or stores or retrieves list values.	<b>III-30</b>

## Cross-references

<b>Move</b> (/M)	Moves the current row to the specified destination.	<b>III-34</b>
<b>Print</b> (/P)	Prints the specified rows from the current sheet onto an output device or into a file.	<b>III-36</b>
<b>Quit</b> (/Q)	Exits the TK!Solver program.	<b>III-38</b>
<b>Reset</b> (/R)	Clears variable values, the current sheet, or all data from the program.	<b>III-39</b>
<b>Return</b> ( $\langle$ )	Returns from the display of a subsheet to the previous sheet or subsheet.	<b>III-41</b>
<b>Right Arrow</b> ( $\Rightarrow$ )	Moves the cursor right one field.	<b>III-3</b>
<b>Select</b> (=)	Displays the specified sheet in the current window.	<b>III-42</b>
<b>Solve</b> (/!)	Solves the currently defined model from any sheet.	<b>III-44</b>
<b>Storage</b> (/S)	Saves, loads, or deletes files containing models or portions of models.	<b>III-45</b>
<b>Switch</b> (;)	Moves the cursor between windows.	<b>III-51</b>
<b>Up Arrow</b> ( $\Uparrow$ )	Moves the cursor up one row.	<b>III-3</b>
<b>Window</b> (/W)	Displays a single window or splits the screen into two windows.	<b>III-52</b>

**Sheets and Subsheets**

<b>Global Sheet (G)</b>	Contains global settings and program defaults.	<b>II-16</b>
<b>List Sheet (L)</b>	Contains a table of all lists.	<b>II-23</b>
<b>List Subsheet (L&gt;)</b>	Contains a detailed description of one list defined on the List Sheet.	<b>II-26</b>
<b>Plot Sheet (P)</b>	Contains information necessary to plot list values.	<b>II-38</b>
<b>Rule Sheet (R)</b>	Contains equations for a model.	<b>II-11</b>
<b>Table Sheet (T)</b>	Contains information necessary to display a table of list values.	<b>II-42</b>
<b>Unit Sheet (U)</b>	Contains definitions for conversions between units of measurement.	<b>II-13</b>
<b>User Function Sheet (F)</b>	Contains a table of all functions defined by the user.	<b>II-30</b>
<b>User Function Subsheet (F&gt;)</b>	Contains a detailed description of one function defined on the User Function Sheet.	<b>II-34</b>
<b>Variable Sheet (V)</b>	Contains data describing the variables of a model.	<b>II-3</b>
<b>Variable Subsheet (V&gt;)</b>	Contains a detailed description of one variable defined on the Variable Sheet.	<b>II-8</b>

### Functional Cross-reference

An index of commands and sheets arranged by their functions:

#### Movement Commands

<b>Left Arrow</b> (←)	Moves the cursor left one field.	<b>III-3</b>
<b>Right Arrow</b> (→)	Moves the cursor right one field.	<b>III-3</b>
<b>Up Arrow</b> (↑)	Moves the cursor up one row.	<b>III-3</b>
<b>Down Arrow</b> (↓)	Moves the cursor down one row.	<b>III-3</b>
<b>Goto</b> (:)	Moves the cursor to the specified field on the current sheet.	<b>III-23</b>
<b>Select</b> (=)	Displays the specified sheet in the current window.	<b>III-42</b>
<b>Switch</b> (;)	Moves the cursor between windows.	<b>III-51</b>
<b>Dive</b> (>)	Displays a dependent subsheet for the current sheet or subsheet in the current window.	<b>III-16</b>
<b>Return</b> (<)	Returns from the display of a subsheet to the previous sheet or subsheet.	<b>III-41</b>
<b>Window</b> (/W)	Displays a single window or splits the screen into two windows.	<b>III-52</b>

#### Modification Commands

<b>Move</b> (/M)	Moves the current row to the specified destination.	<b>III-34</b>
<b>Insert</b> (/I)	Inserts a blank row at the current row.	<b>III-29</b>
<b>Delete</b> (/D)	Deletes the current row, or a block of rows beginning with the current row.	<b>III-14</b>

<b>Copy</b> (/C)	Copies the current field item to the specified destination.	<b>III-12</b>
<b>Blank</b> (/B)	Erases the current field item, or a column of items beginning with the current item.	<b>III-10</b>
<b>Edit Field</b> (/E)	Edits the entered item in the current entry field.	<b>III-21</b>
<b>Edit</b> (/E)	Edits a user response on the Prompt/Error line or an unentered item in the current entry field.	<b>III-18</b>
<b>Reset</b> (/R)	Clears variable values, the current sheet, or all data from the program.	<b>III-39</b>

**Solution Commands**

<b>Action</b> (/!)	Performs the action appropriate to the current sheet, including model solution, list value fill-in, plot or table display.	<b>III-6</b>
<b>Solve</b> (/!)	Solves the currently defined model from any sheet.	<b>III-44</b>
<b>List</b> (/L)	Solves a model for lists of values associated with variables, or stores or retrieves list values.	<b>III-30</b>

**General Program Commands**

<b>Quit</b> (/Q)	Exits the TK!Solver program.	<b>III-38</b>
<b>Help</b> (/?)	Calls the Help Facility.	<b>III-26</b>
<b>Storage</b> (/S)	Saves, loads, or deletes files containing models or portions of models.	<b>III-45</b>
<b>Print</b> (/P)	Prints the specified rows from the current sheet onto an output device or into a file.	<b>III-36</b>

## Cross-references

### Sheets

<b>Variable Sheet (V)</b>	Contains data describing the variables of a model.	<b>II-3</b>
<b>Rule Sheet (R)</b>	Contains equations for a model.	<b>II-11</b>
<b>Unit Sheet (U)</b>	Contains definitions for conversions between units of measurement.	<b>II-13</b>
<b>Global Sheet (G)</b>	Contains global settings and program defaults.	<b>II-16</b>
<b>List Sheet (L)</b>	Contains a table of all lists.	<b>II-23</b>
<b>User Function Sheet (F)</b>	Contains a table of all functions defined by the user.	<b>II-30</b>
<b>Plot Sheet (P)</b>	Contains information necessary to plot list values.	<b>II-38</b>
<b>Table Sheet (T)</b>	Contains information necessary to display a table of list values.	<b>II-42</b>

### Subsheets

<b>Variable Subsheet (V))</b>	Contains a detailed description of one variable defined on the Variable Sheet.	<b>II-8</b>
<b>List Subsheet (L))</b>	Contains a detailed description of one list defined on the List Sheet.	<b>II-26</b>
<b>User Function Subsheet (F))</b>	Contains a detailed description of one function defined on the User Function Sheet.	<b>II-34</b>





### Program Overview

#### Program Startup

The way to start the TK!Solver program depends on the particular computer and operating system being used. The startup for your system is described in the TK!Solver Instruction Manual and in the Hardware Reference section of this manual.

The TK!Solver program first displays a full-screen startup message with a copyright notice and program version numbers. Initiate use of the program by pressing the Enter key. The message "Loading the TK!Solver program" appears at the top of the screen while the program is being loaded.

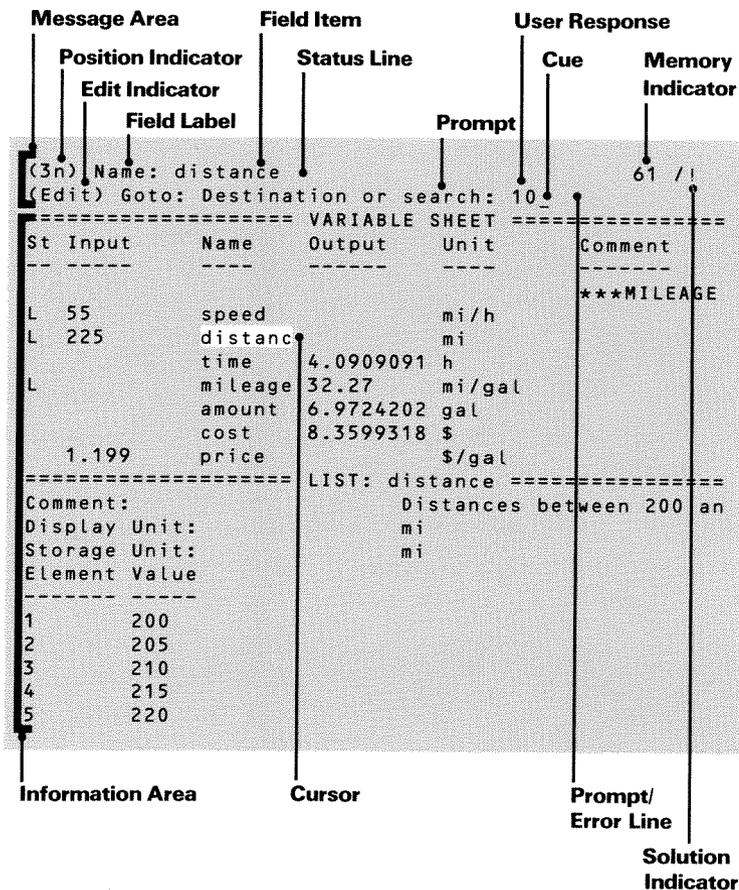
At program startup, two windows are displayed. They contain the two most frequently used sheets: the Variable Sheet, and the Rule Sheet. The top line on the screen, called the Status line, displays program status messages. The second line of the screen, called the Prompt/Error line, displays user messages and begins the program with a message about how to call the Help Facility. This message disappears with the first keystroke. The cursor, which marks the current field, begins the program on the first row of the Rule Sheet in the first Rule field.

Any time after this display appears, you can either use the program or exit it by typing the Quit command (/QY).

## Screen

### The TK!Solver Screen

The TK!Solver program screen is divided into two areas: the Message Area and the Information Area.



### Message Area

The Message Area is used by the program to provide status information. It consists of two lines: the Status line and the Prompt/Error line.

#### Status Line

The Status line provides five kinds of status information:

The **Position Indicator** is a number and letter (or only a letter) in parentheses that indicates the address of the field marked by the cursor and changes as the cursor moves. The number, if any, represents the row; the letter stands for the label of the field. (Each field in the program has a letter name used in referring to its address.)

At program startup, **(1r)** is displayed, indicating the Rule field on the first row of the Rule Sheet.

The **Field label** on the Status line displays the label of the current field.

At program startup, it displays **Rule:**.

The **Field item** on the Status line displays the contents of the field currently marked by the cursor. Field items that are longer than the screen fields they occupy can be viewed more fully on the Status line. This includes the full display of value items that are rounded off or expressed in scientific notation so that they can display within their fields. The end of a field item even longer than the space available on the Status line can only be viewed by using the Editor on the current field and moving the cue to the end of the item.

The explanation for the letters and characters displayed in Status fields on the Variable and Rule sheets can be viewed on the Status line by moving the cursor to the field in question and looking at the complete definition on the Status line.

At program startup, no field item is displayed.

The **Memory Indicator** is a number followed by a slash that indicates the amount of remaining memory and is displayed in K (1024) bytes. When the remaining memory is less than 1 K, the Memory Indicator displays **LOW**. Only minimal actions are allowed: some actions requiring additional memory fail. When there is no remaining memory, the Memory Indicator displays **OUT**. At this point, most actions except saving a file fail, until something is deleted.

At program startup, a number is displayed representing the amount of memory available for use after the program is loaded.

## Screen

The **Solution Indicator** is an exclamation point (!) that indicates the solution status of the model. It appears whenever a modification is made that may invalidate the current solution. It disappears when the Solve command is typed, any equation in the model is satisfied, and there are no errors in the model.

### Prompt/Error Line

The Prompt/Error line displays the prompts, status messages, and error messages provided by the program. Certain prompts require the user to type on the Prompt/Error line.

At program startup, the Prompt/Error line displays “For Help, type ?”. The Help reminder disappears when the first key is pressed.

The Prompt/Error line displays six kinds of information:

The **Edit Indicator** is the word “Edit” in parentheses that appears whenever the user is in the Editor on the current field or on the Prompt/Error line itself. It disappears when the Enter key completes the entry or the Break key cancels it.

**Prompts** are produced by the program to indicate the next user action. Most prompts offer a selection of options; some require entering information on the Prompt/Error line. (See the appendix on Prompts for a listing of each prompt in the program.)

The **Pointer Indicator** is a number and letter (or letter only) in parentheses that appears at the end of prompts that request the user to point to another field. It supplies the current cursor position during pointing and changes whenever the Arrow keys or the Goto command (: ) move the cursor. While the Pointer Indicator is being used, the Position Indicator does not change. The Pointer Indicator disappears when the Enter key completes the command.

**User responses** are required by certain commands. Some commands offer a choice of options. If an option is typed, it does not appear on the Prompt/Error line. Some commands require entries to be made on the Prompt/Error line itself. The prompts for these commands are ended by a colon. The cue appears on the Prompt/Error line indicating where to type. Any information

typed on the Prompt/Error line must be completed with the Enter key. A user response that has not yet been completed by the Enter key is called an “entry” in this manual.

The Editor can be called to edit a partial entry on the Prompt/Error line by using the Edit command  any time before the final Enter key is pressed.

**Status messages** are displayed during certain program processes that require time to perform, such as using the solvers, or deleting, loading, or saving files. (See the appendix on Status Messages for a listing of each status message in the program.)

**Error messages** are produced by the program for a variety of reasons, including output errors or attempts to enter an invalid command, entry, or option. They are usually accompanied by a computer error signal.

Entry errors automatically place the user in the Editor and disappear when a valid item is entered. Command and option error messages disappear with the next valid keystroke.

See the appendix on Error Messages for a listing of each error message in the program.

### **Information Area**

The Information Area is the part of the screen that displays the sheets on which to set up models and generate solutions.

The Information Area can display either one or two **windows** at once, allowing a view of two sheets simultaneously, or a full screen view of a single sheet. Two different views of the same sheet may also be displayed at one time; any change made to one view of a sheet is automatically made on the other. Changes made to fields that correspond on more than one sheet, such as a field on both the Variable Sheet and one of its subsheets, are made simultaneously on both sheets whether or not both are currently displayed.

**Sheets** are locations, viewed through windows, for entering and displaying the data for a model. There are eight main sheets:

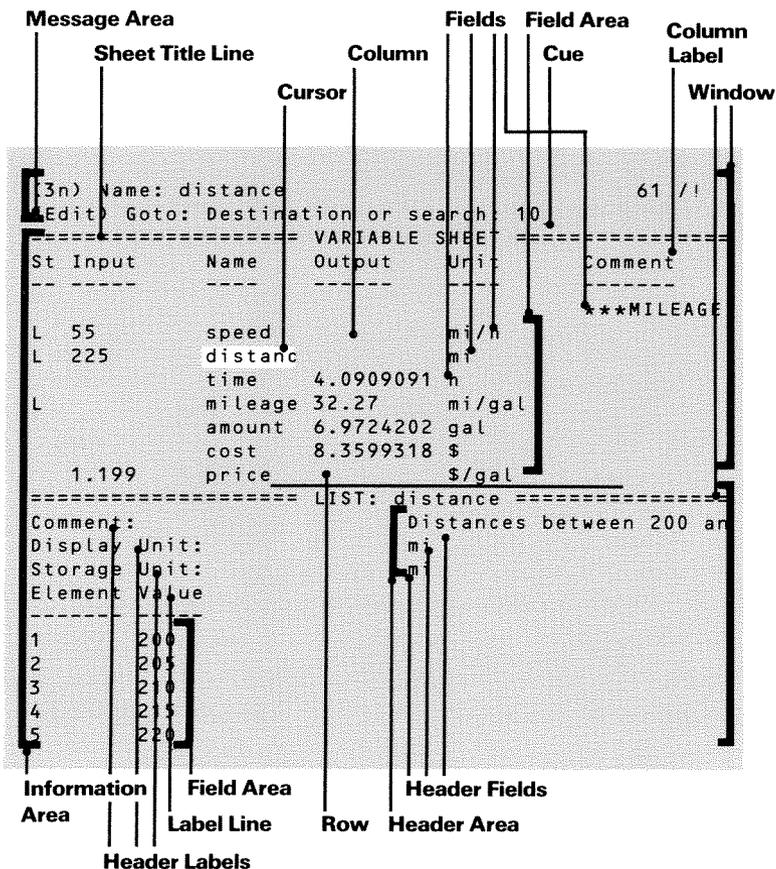
- Variable Sheet
- Rule Sheet
- Unit Sheet

## Screen

- Global Sheet
- List Sheet
- User Function Sheet
- Plot Sheet
- Table Sheet

**Subsheets** are related to particular sheets and provide detailed information about entities on the corresponding sheet. A subsheet can be reached by placing the cursor on a row on a sheet and diving into the subsheet related to that row. (See the Dive command for more information.) There are three kinds of subsheets:

- Variable subsheets
- List subsheets
- User Function subsheets



The parts of a sheet or subsheet are:

### **Sheet Title Line**

The Sheet Title line defines the top of a sheet. It contains the name of the sheet and a row of equal signs (=).

A Sheet Title line for a subsheet differs from one for a sheet in that the name of the described variable, list, or function is included following the subsheet title.

### **Fields and Labels**

Fields are positions on a sheet where the cursor can be moved and information entered or displayed. Each field is marked by a label. There are three types of fields: entry fields, option fields, and output fields (see the section on Field Types). Each field in the program has a unique address that appears as the Position Indicator on the Status line whenever the cursor is located on that field.

Fields have a display size, which is the size of the field appearing on the screen, and an item size, which is the actual length of the contents of the field.

For convenience, the Reference Manual describes fields as being organized into one of two areas: Header Areas and Field Areas.

**Header Areas** contain general information about the contents of a sheet or use of the program. For example, the Header Area on the Global Sheet contains information the program needs when solving equations.

A Header Area contains a block of rows, each consisting of a Header label on the left, and a Header field on the right. Each label describes the type of information contained in only one field. A field address in a Header Area consists of a single letter, which stands for the label.

Several TK!Solver commands do not work in Header Areas. For example, rows cannot be inserted, deleted, or moved. Many entries in Header fields are required, and cannot be blanked.

The Header Area is above the Field Area when both occur on the same sheet. It disappears when a Field Area is scrolled down, reappearing when the cursor is moved back up to the Header Area.

The Global Sheet and a Variable subsheet contain only Header Areas. The Plot and Table sheets, and List and User Function

## Screen

subsheets, contain Header Areas as well as Field Areas. The other sheets do not contain Header Areas.

**Field Areas** contain fields organized in rows under a line of column labels. One label describes every field below it. Column labels are always displayed, even if the Field Area is scrolled down. All TK!Solver commands operate in Field Areas. A field address in the Field Area is a row number and a single letter, which stands for the column label.

### Rows and Columns

A row is a line of labeled fields. A column is a vertical list of fields under the same label.

### Items

An item is the information typed into a field by the user or displayed there by the program. Characters typed into an entry field are only entered by pressing the Enter key or an Arrow key.

There are three types of items:

- Entry item: one or more characters typed and entered by the user into an entry field.
- Option item: a single character typed by the user into an option field, interpreted and expanded as appropriate by the program.
- Output item: information displayed by the program in an output field.

An item may be longer than the display size of the field it occupies. Maximum character item length is 200 characters; real numbers are accurate up to 10 digits; maximum integer length is dependent on your computer. (See the Hardware Reference section for more information.)

Values may be rounded off or expressed in scientific notation in order to fit into the display size of various fields. However, all calculations are performed using the full accuracy.

Items that exceed their display size may be viewed more fully on the Status line by moving the cursor to the field. Items longer than the area on the Status line can only be viewed in the Editor, by scrolling to the end of the item with the Arrow keys.

#### Other Displays

There are four other displays that preempt all or part of the screen in the TK!Solver program:

- The Help Facility replaces the Information Area with a display labeled “HELP” at the top, where topics about the use of the TK!Solver program can be viewed.
- Plot displays replace the entire screen with a one-page graph.
- Table displays replace the entire screen with a single- or multi-page table of list values.
- Print displays replace the entire screen with lines printed to a file or printer device by the Print command (/P).

## Movement

### Movement Through The Program

The TK!Solver program has two moveable pointers that mark the current point of attention: the cursor moves around the screen; the cue moves within text.

#### Cursor

The cursor marks the current field. Each sheet has an initial cursor position, the place marked by the cursor the first time a sheet is accessed. When a sheet is re-accessed the previous cursor position and view of the sheet are displayed. The cursor always returns to the initial cursor position on a subsheet.

The cursor is moved from field to field by the Arrow keys:

- ◀ Left one field
- ▶ Right one field
- ↑ Up one row
- ↓ Down one row

It can also be moved by the Goto command (:); the Switch command (;), which moves it between windows; and as a result of some commands that require pointing to a new field.

If an attempt is made to move the cursor past the right or left side of the screen, or up past the top of a sheet or subsheet or down below its end, the computer produces an error signal. The bottom of most sheets and subsheets is defined at row 32000, but the number of items that can be entered is limited by the amount of memory available. Variable subsheets and the Global Sheet have fixed last lines.

If the cursor is moved down beyond the current screen or window display, the current sheet scrolls down a line at a time to display a new view. The view may be scrolled back up by moving the cursor past the top of the current display.

#### Cue

The cue marks the position within the text in the current field or on the Prompt/Error line. It creates an artificial space and marks an insertion point.

It appears whenever characters are typed or edited within a field, or when the user is prompted to type on the Prompt/Error line.

All characters typed appear to the left of the cue; the Backspace key deletes the character to the left of the cue. The cue can be moved around within the text and disappears, along with the space it creates, after the Enter key is pressed.

When editing, the cue is moved around a field or the Prompt/Error line by the Arrow keys:

- ◀ Left one character
- ▶ Right one character
- ⤴ To the beginning of the text
- ⤵ To the end of the text

If an item is longer than the display size of the field or the display area of the Prompt/Error line, its view is changed as the cue moves. When the Enter key is pressed to enter the item in a field the cue disappears and the beginning of the item is displayed at the left edge of the field.

### Program Environments

There are several environments or modes in the TK!Solver program in which particular user actions are valid. They are categorized for convenience only in the Reference Manual.

As the cursor moves around the program, it marks an entry field, an option field, or an output field. Modes in the TK!Solver program are dependent on where the cursor is positioned and what is typed. (See the section on Field Types for more information about valid field entries.)

There are three main program modes:

- Command mode: on an entry, option, or output field
- Entry mode: on an entry field
- Option mode: on an option field

There are two additional environments: the Editor and the Help Facility. The Editor can be called from Command, Entry, or Option mode to edit the text currently being typed with the Edit command . The Help Facility can only be called with the Help command (?) from Command mode.

In addition to the active environments, there are times when the cursor is temporarily inactive during program actions that require time to perform. A status message usually indicates such an action, which can only be interrupted by pressing the Break key.

### Command Mode

Command mode allows single-key commands and extended commands to be typed any time the user is not in another mode. The user is automatically placed in Command mode if the first character typed is one of the following commands:

- Edit command
- ? Help command
- / a Slash command
- = Select command
- : Goto command
- ; Switch command  
(when there are two displayed windows)
- > Dive command  
(from the Variable, List, User Function, Plot and Table sheets, and Variable and User Function subsheets)

- < Return command  
(from Variable, List, and User Function subsheets)
- ! Action command  
(from the Variable, Rule, Plot, and Table sheets; from List and User Function subsheets)

Certain commands require more than one keystroke: the Select command (=), the Goto command (:), the Slash commands (/), and the Help command (?). Of these, the Select command and some of the Slash commands prompt for a command option.

The Goto command (:), the Help Facility (?), the Storage command (/S), and the List command (/L) prompt for user responses on the Prompt/Error line. Commands that require typing on the Prompt/Error line in response to a prompt end in a colon. The cue appears on the Prompt/Error line and any characters typed must be entered with the Enter key. The Editor can be called on the Prompt/Error line by typing the Edit command  any time before the Enter key is pressed. Valid user responses are prescribed by the particular command, or the computer file system in the case of file specifications.

### Entry Mode

Entry mode allows characters to be typed and entered into an entry field. The user is automatically placed in Entry mode any time the cursor is positioned on an entry field and any character other than a command is typed. In Entry mode, the cue appears and all characters typed are inserted to its left; pressing the Backspace key deletes one character to the left of the cue.

Any characters typed into an entry field are accepted until the Enter key is pressed, at which time the item is checked for validity and entered. A valid item is determined by the type of entry field. If the item is determined to be invalid when the Enter key is pressed, the user is automatically placed in the Editor. The Break key can be used any time before the Enter key is pressed to cancel an uncompleted item. If the field contained a previous item, it is restored when the Break key is pressed.

Any characters typed after the Enter key or Break key is pressed, other than a command character, begin a new entry in the current field.

## Environments

To enter a command character or a space as the first character of an entry field, call the Editor and use the special character as a regular entry character.

### Option Mode

Option mode allows choices that perform certain actions to be typed into option fields. Options are entered without using the Enter key. The user is placed in Option mode any time the cursor is positioned on an option field and a character that is not a command character is typed. When a valid option is typed, the option is displayed in the field as appropriate and any immediate action taken. If an invalid option is typed, an error message on the Prompt/Error line indicates the valid option choices for that field.

### The Editor

The Editor allows modification of the characters in a field or on the Prompt/Error line. The user is automatically placed in the Editor any time an entry error is made on an Entry field. The Editor can also be called from Command mode with the Edit Field command (/E), or from Entry mode or on the Prompt/Error line with the Edit command . (See the Command section for further information on the Edit commands.)

The word “(Edit)” appears on the Prompt/Error line any time the Editor is being used.

The Editor is exited by pressing either the Enter key, to check the item for validity and complete it, or the Break key, to cancel the current unentered item and restore any previous item to the field, if one exists.

#### The Help Facility

The Help Facility provides on-line reminders about program features. It is called by typing the Help command (?). The Help topic prompt appears on the Prompt/Error line. There are three possible responses to this prompt any time it appears:

- Typing a second ? to get more information on using the Help Facility.
- Typing a topic name followed by the Enter key to locate all text on the specified topic.
- Pressing the Enter key to exit the Help Facility.

A list of valid topics can be displayed by typing the topic name "TOPICS" followed by the Enter key.

A single topic name may be associated with a number of blocks of text. If the text takes more than one page, a prompt appears for the next page. Typing **Y** continues to the next page; **N** returns immediately to the Help topic prompt. When the final text block has been displayed for a topic, the Help topic prompt re-appears. A prompt must appear, indicating the end of a page or that the last section of text has been displayed, before typing further.

A topic name can be corrected on the Prompt/Error line by using the Edit command  any time before the Enter key is pressed. The Editor is exited and the search for the topic begun by pressing the Enter key.

If the specified topic is not available, a prompt appears offering more help. Typing **Y** displays the Help topic prompt again; typing **N** exits the Help Facility.

Help can be exited any time the Help topic prompt is displayed by pressing the Enter key, or interrupted at any point by pressing the Break key. The previous display and cursor position are returned.

### Field Types

There are three types of fields in the TK!Solver program: entry fields, option fields, and output fields.

#### Entry Fields

Entry fields accept items typed in and entered by the user. The user is automatically placed in Entry mode whenever the cursor is positioned in an entry field and any character other than a command is typed. Text typed into a field is verified and completed by pressing the Enter key. If an item is not valid for the current field, pressing the Enter key produces an explanatory error message and places the user in the Editor.

Some entries automatically insert items in other fields. When the Variable Insert ON field on the Global Sheet is set to **Yes**, any previously undefined variable name entered in a Rule field is automatically inserted at the bottom of the Name column on the Variable Sheet. If a previously undefined list name is entered in any list name entry field, it is automatically inserted at the bottom of the Name column on the List Sheet. (Automatic list name insertion is independent of the Automatic Insert ON field on the Global Sheet.)

Once automatically inserted, a variable name or list name remains, even if the field in which it was entered is blanked or edited.

The maximum allowable length for a name, unit, comment, rule, filename, or setup string entry is 200 characters, although the display size is limited to the size available in a field or on the Status line. The maximum length for a variable name, symbolic value, or expression entered into a value, numeric value, or integer value field is also 200 characters. Text entries are case sensitive, meaning that distinction is made between uppercase and lowercase on computers that use both cases.

Numbers entered in value and numeric value entry fields have a maximum of 12 significant digits. The numeric range for values is system dependent. See the Hardware Reference section for the range for your system. Although the internal representation of number entries used in calculations is constant, numbers are displayed differently to fit in different display sizes: the Field item on the Status line is displayed up to the full 11 or 12 significant

digits; value and numeric value fields are displayed up to 10 significant digits. Numbers are rounded off or expressed in scientific notation upon entry to fit smaller display sizes. In greater display sizes, zeros are used to indicate the number of entered digits beyond the significant digits.

Integer entry limits are determined by the computer being used and the valid values for the particular field. (See the Hardware Reference section and the Sheet section for details.)

Value, numeric value, and integer entries can be entered as numbers or as expressions, if all variables have defined values. (See the Mathematics section for more information about valid expressions.)

Integer and numeric value entry fields contain default values provided by the program which can be changed but cannot be blanked.

Entry fields accept nine kinds of entries:

#### **Name Entry**

A name field is used to assign a name to a single value, a list of values, or a user-defined function. A relationship can be created between a list and a variable. A function can be formed by entering two list names in appropriate fields.

A name entry cannot begin with a number but thereafter can contain any combination of letters, digits, and the following characters up to 200 characters:

@ # \$ % \_

Spaces are not allowed.

A variable name can be entered in the Name field on the Variable Sheet; a user function name can be entered in the Name field on the User Function Sheet; all other name entry fields take list names.

#### **Unit Name Entry**

A unit name field is used to assign a unit of measurement name to a variable or list, or to define the conversion relationship between two unit names.

The conversion relationship between two unit names is assigned on the Unit Sheet. One unit name is assigned to each unit field on

## Field Types

the Unit Sheet, and each row defines one unit conversion relationship.

A pair of unit names can be assigned for each variable and each list. Variables have Calculation units and Display units; lists values have Storage units and Display units. Subsheets display the pair of unit names separately. The Unit fields on the Variable and List sheets show only a Display unit name.

The first unit name entered in a Unit field on the Variable Sheet assigns both the Calculation and the first Display unit for the corresponding variable; the first unit name entered in the Unit field on the List Sheet assigns both the Storage unit and the first Display unit for the corresponding list. Any subsequent unit name entry in the same field changes the Display unit only. (See the Unit Sheet in the Sheet section and Units in the Mathematics section for more information about unit conversions.)

A unit name entry can contain any combination of displaying characters, including internal spaces, up to 200 characters. Leading and trailing spaces are deleted upon entry.

### **Comment Entry**

A comment field is used to enter a comment or to display text. Comments are not used by the program in calculations.

A comment entry can contain any printable characters and spaces up to 200 characters. The first character of a comment cannot be a command character or space unless it is edited in.

Comment entries on the Plot and Table sheets are used as display text.

### **Value Entry**

A value field is used to assign a numeric or symbolic value to a variable, a list element, or a user function list element.

A value entry may be entered as a real number, a symbolic value, or any valid expression with defined values. The absolute value range for a real number entry is system dependent. See the Hardware Reference section for the range for your system. A symbolic value can be entered like a name entry, preceded by an apostrophe.

#### **Numeric Value Entry**

A numeric value field is used to assign parameters for the Solvers.

A numeric value entry may be entered as a real number or any valid expression with defined values. A numeric value field is assigned a positive real number value within the range for your system. See the Hardware Reference section for the range for your system. If a negative number is entered, its absolute value is used.

Numeric value entries are entered in the Comparison Tolerance and Typical Value fields on the Global Sheet. They are required entry fields with default values that can be changed but cannot be blanked.

#### **Integer Entry**

An integer field is used to assign parameters for program features. All integer entries are required and have default values.

An integer entry must be entered as a positive integer or any valid expression with defined values that evaluates to a positive integer. The maximum valid integer in a field is determined by the computer being used, but all integer fields have practical limits dependent upon their use. (See the Hardware Reference section or the Sheet section for more information.)

#### **Rule Entry**

A rule field is used to assign a statement relating variables, expressed as an equation between two expressions.

A rule entry must contain an equal sign and must conform to the conventions for valid rules outlined in the Mathematics section. Any characters following a quotation mark (") in a Rule field are interpreted as a comment. A rule field may contain up to 200 characters.

A rule entry can only be entered in the Rule fields on the Rule Sheet.

#### **Filename Entry**

A filename entry is used to specify the output device name or the file specification for a file used by the program.

A filename entry can be entered up to 200 characters, but the length and validity of a file specification is usually determined by the operating system of the computer. (See the Hardware Reference section for more information.)

## Field Types

A filename entry can be entered on the Global Sheet in the Printer Device or Filename field, or on the Prompt/Error line in response to a prompt for a filename in the Storage command.

### Setup String Entry

A setup string field is used to enter a string that specifies printing options for the attached printer.

A setup string entry can include displaying characters which represent control and escape sequences that direct the printer to print titles, alter type size, underscore or boldface, issue form feeds, etc. The first character cannot be a command character unless edited in. A setup string entry field can accept up to 200 characters, but valid setup strings are dependent on the computer and printer being used. (See the Global Sheet in the Sheets section, the Hardware Reference section, and the manuals that accompany your computer and printer for more information.)

A setup string entry can only be entered in the Printer Setup String field on the Global Sheet.

### Option Fields

Option fields accept a single character, without the Enter key, selected by the user from a limited set of possibilities and interpreted by the program. If an invalid option is typed, an error message displays listing the valid options.

There are two kinds of option fields: those in which the selected character is used to perform a task (such as the **I** or **L** options in the Status fields of the Variable Sheet); and those in which the character is expanded into a full definition (such as the **T** option in the Mapping fields of the User Function Sheet).

Some option fields have default values that can be changed but they cannot be blanked. Field names ending in "ON" require **Y** or **N** options. Field names containing "or" offer a choice between two requirements.

The Status fields on the Variable Sheet are special option fields that are also used by the program to display error codes. The Character fields on the Plot Sheet may contain any displaying character or be blanked to display nothing.

#### **Output Fields**

Output fields are used by the program to display information. The cursor can be moved to any output field except Element fields on List and User Function subsheets, but the user cannot type in output fields. The Status fields on the Variable Sheet are both option and output fields. The Value field on List subsheets and the Domain and Range fields on User Function subsheets are both Value Entry and Output fields.

On List and User Function subsheets, element numbers are automatically assigned in a column to the left of values as they are entered.

### **Error Conditions**

Error messages for the TK!Solver program appear on the Prompt/Error line at the top of the screen, or, in the case of solution errors, in the Status columns of the Variable or Rule sheets.

Error messages are sometimes provided on the bottom of the screen for hardware problems, such as a missing disk or printer. These errors usually include an instruction about what to do.

See the Appendix section for a list of each error message provided by the program and the appropriate actions to take.

The Break key can be used to cancel an extended command, discard a partially entered item, or interrupt an action. The computer produces an error signal and cancels the activity.

### **Entry Errors**

When an attempt is made to enter an invalid item in an entry field, the computer produces an error signal, an error message appears on the Prompt/Error line, the user is placed in the Editor, and the cue appears to the right of the first recognized error in the field. The error message disappears when the Break key is pressed to cancel the entry, or when the Enter key is pressed to enter a valid item.

### **Option Errors**

When an attempt is made to type an invalid option in an option field, the computer produces an error signal, and an error message appears on the Prompt/Error line providing the valid options. The error message disappears with the next valid keystroke.

### **Command Errors**

When an attempt is made to type invalid command keystrokes, or a command at an invalid time, the computer produces an error signal, and the command prompt, if any, disappears from the Prompt/Error line.

#### Unit Errors

When an attempt is made to convert the value of a variable to another unit with no unit conversion defined, a question mark (?) appears to the left of the value in the corresponding value field.

If units are used for a variable or a list, both the Calculation unit, or Storage unit, and the Display unit must be defined. An error is indicated if one of the two units has been independently blanked on a subsheet: a question mark appears to the left of any unit name that has no defined Calculation or Storage unit; a question mark appears to the right of any unit name with no defined Display unit.

The question mark in a value field or a unit field disappears when a proper unit conversion definition is provided or all units are blanked for the variable values in question.

#### Solution Errors

An unsolved model is indicated by the Solution Indicator in the upper righthand corner of the screen and certain symbols in the Status fields of the Variable and Rule sheets.

The Solution Indicator (!) appears whenever a change is made that may affect the solution of a model, such as entering a new value or editing a rule. It disappears when any equation is solved and there are no errors in the model.

Before a solution, an asterisk (\*) appears in the Status column of the Rule Sheet to the left of all unsatisfied rules. An unsatisfied rule is one that has not yet been evaluated successfully by the solvers.

After a solution has been attempted and an error found, the greater than symbol (>) appears in the Status column of the Rule Sheet identifying equations in error. The program recognizes an equation containing a calculation error (such as division by zero) and sets of equations that are inconsistent with each other. In a set of conflicting equations, usually the first equation in conflict with a previously evaluated rule is marked. Whenever the TK!Solver program encounters inconsistent or contradictory data in an equation, it stops solving.

## Error Conditions

When solution fails due to an error and rules are marked, the greater than symbol (>) also appears in the Status column of the Variable Sheet to indicate the variables involved in the failure. Variables are marked for overdefinition or being involved in a calculation error.

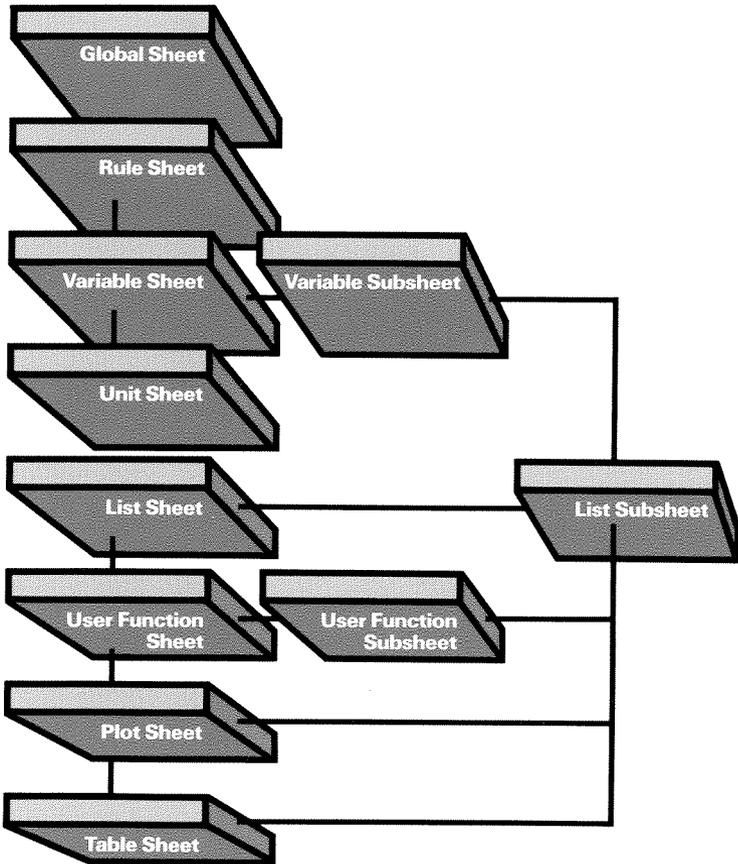
An explanatory message is provided for each Status field symbol indicating the type of failure. It can be viewed by moving the cursor to the Status field in question and looking at the explanation of the symbol on the Status line.

Error symbols in Status fields are displayed until the error condition no longer exists. Asterisks remain until a rule is satisfied or determined to contain an error by one of the solvers.





**Sheets**



### **Sheets and Subsheets**

There are eight sheets and three subsheets in the TK!Solver program.

This section describes the sheets and subsheets available with the TK!Solver program in the following order:

Variable Sheet  
Variable Subsheets  
Rule Sheet  
Unit Sheet  
Global Sheet  
List Sheet  
List Subsheets  
User Function Sheet  
User Function Subsheets  
Plot Sheet  
Table Sheet

The description of each sheet or type of subsheet contains:

- its purpose
- its picture, containing data from the mortgage model used in the Instruction Manual
- how to access it and the subsheets associated with it
- its initial cursor position
- its fields

Each field listing contains:

- its label, followed by its label letter in parentheses (label abbreviations on the screen are also spelled out here)
- its field type, in brackets
- any default value for the field
- a description of the field
- a description of any field options

See the Overview section for more information about terminology used to describe sheets and the conventions for valid field types.

**The Variable Sheet**

The Variable Sheet contains data describing the variables of a model. Each row can contain the description for one variable. There is a Variable subsheet displaying more information about each variable; items entered into fields on the Variable Sheet are updated simultaneously on corresponding fields on Variable subsheets.

```
(1) Input: 95000 61 /
```

===== VARIABLE SHEET =====					
St	Input	Name	Output	Unit	Comment
-----					
	95000	H		dollars	House price
		d	19000	dollars	Down paymen
	20	dp		percent	Down paymen
		p	76000	dollars	Mortgage
	25	n		years	Term
	17	i		percent	Interest ra
		A	1092.7254	dollars	Monthly pay
		T	327817.62	dollars	Total of pa
		Tip	251817.62	dollars	Total inter

**Access:**

The Variable Sheet appears in the top window at program startup. It can be displayed in the current window by typing =V, or in a new window by typing /WV from a one-window display.

A Variable subsheet for a variable can be reached by typing the Dive command (>) with the cursor on any field on the variable row.

The List subsheet for a list associated with a variable can be reached by placing the cursor on any field in the variable row and typing the Dive command (>) twice, once into the Variable subsheet, and a second time into the List subsheet for the associated list. The cursor may be positioned on any field on the Variable subsheet.

## Variable Sheet

### Initial Cursor Position:

The Input field on the first row

### Fields:

Each variable is described by a row containing six fields, located in columns under the labels in the Label line at the top of the sheet. The Variable Sheet fields are:

#### St [Status] (s)

[Option]

Defines the status of the variable. It is a two-character field. The valid options are:

#### I Input Option

Designates a variable as an input variable. If an output value is currently displayed for the variable, it is moved to the Input field. If a guess value is displayed, the “G” setting in the Status field is removed and the value considered an input value. If no value is displayed, a zero value is assigned or the previous input, guess, or output value is displayed in the field. Although a list setting “L” is unaffected, a list is designated as an input list with the input option.

#### O Output Option

Designates a variable as an output variable. If an input value is currently displayed for the variable, it is moved to the Output field. If a guess value is displayed, the “G” setting in the Status field is removed and the value moved to the Output field. If no value is displayed, a zero value is assigned or the previously input or output value is displayed in the field. Although a list setting “L” is unaffected, a list is designated as an output list with the output option.

#### L List Option

Associates a variable with a list name. An “L” is displayed in the first character position of the Status field. The variable name is assigned by default to the Associated List field on the Variable subsheet for that variable and a list by that name is created on the List Sheet if it does not already exist. The default list name may be changed in the Associated List field on the Variable subsheets. Lists are inserted at the

bottom of the List Sheet in the order they are associated and created. If there are Calculation and Display units for a variable when the list association is made, they are assigned as the Storage and Display units for the list. (If units are assigned to a variable after the list association is already made, they are not carried over.) Typing **L** a second time removes the “L” from the Status field and disassociates the list, leaving it listed on the List Sheet but no longer connected with the variable.

**G**    **Guess Option**

Designates a variable value as a starting guess value for a variable for use with the Iterative Solver. A “G” is displayed in the second character position of the Status field and the word “Guess” displayed on the Status line. If an output value is currently displayed for the variable, it is moved to the Input field. If an input value is displayed, it is considered a guess value. If no value is displayed for the variable, a zero value is assigned or the previous blanked input, guess, or output value is displayed in the field. The previous input value may be changed. Typing **G** a second time removes the “G” from the Status field and leaves the value in the Input field as an input value. A list setting “L” is not changed. If a list is associated, a “G” in the Status field designates the elements in the list as guess values. The results of the Iterative solution are written into the list for each element.

**B**    **Blank Option**

Blanks any values for a variable from the Input or Output fields and any “G” option from the Status field. A list setting “L” is not changed. Blanked values can be recovered by typing **I**, **O**, or **G** in the Status field. Blanked values are saved when a model is saved.

[Output]

If solution fails, the program displays an error symbol (>) in the second character position of the Status field of each variable involved in an error condition. If the cursor is moved to the Status field containing the error symbol, its explanatory message is displayed on the Status line at the top of the screen.

## Variable Sheet

### **Input (i)**

[Value Entry]

Accepts an input value for a variable. An input value can be any valid numeric value or expression, or a symbolic value. Entering an input value erases any value from the Output field for that variable entity.

If Calculation and Display units are assigned to the Unit field for the variable, changing the Display unit immediately converts the input value to a value in the new Display unit. If the conversion between the Calculation unit and Display unit is undefined, the value is displayed in the Calculation unit, preceded by a question mark (?).

If a numeric value has been blanked, it can be recovered by typing **I** or **G** in the Status field.

See the Overview and Mathematics sections for more information about valid value entries.

### **Name (n)**

[Name Entry]

Accepts a unique variable name. A variable must be named if it is to be associated with a list.

If the Variable Insert ON field on the Global Sheet is set to “Yes”, any variable names entered in a Rule field on the Rule Sheet that do not previously exist are automatically inserted at the bottom of the Name column. Once inserted on the Variable Sheet a variable remains even if the rule in which it was entered is deleted or edited.

### **Output (o)**

[Output]

Displays the output value for a variable if a solution produces a value or if a value is moved there by typing **O** in the Status field.

If different Calculation and Display units are assigned to the variable, the output value appears in the Display unit. If the conversion between the designated Calculation unit and Display unit is undefined, the value is displayed in the Calculation unit, preceded by a question mark (?).

If the value has been blanked, it can be recovered by typing **O** in the Status field.

**Unit (u)**

[Unit Name Entry]

Assigns the Calculation and Display unit names for the variable. If the Unit field is blank, the first unit entered becomes both the Calculation and Display units for the variable. Any subsequent entry changes only the Display unit. The Blank command (/B) can be used on the Variable Sheet to blank both the Calculation and Display units for a variable simultaneously.

The unit conversion must be defined explicitly on the Unit Sheet, or the two units must be defined in terms of a common third unit. If the conversion is defined, the numeric value in the Input or Output field (and Input Value or Output value, and First Guess fields on a Variable subsheet) is immediately converted to its value in the Display unit whenever the Display unit name is changed.

If the conversion between the Calculation unit and the Display unit is not defined on the Unit Sheet, any value in the Input or Output field is expressed in the Calculation unit and preceded by question mark (?).

Any variable associated with units must have both a Calculation and Display unit assigned at all times, although they can be identical. If the Calculation unit name has been blanked from the Variable subsheet without also blanking the Display unit, the unit name appears preceded by a question mark. If the Display unit has been independently blanked, the unit name appears followed by a question mark.

For more information about unit conversion, see the Mathematics section.

**Comment (c)**

[Comment Entry]

Accepts a comment or description of the variable. Any characters entered into this field are ignored by the program during solving. Comments can be especially useful if a model is to be saved for later use.

## Variable Subsheet

### Variable Subsheets

Each Variable subsheet contains a detailed description of one variable defined on the Variable Sheet. There is one Variable subsheet for each variable, whether named or not, on the Variable Sheet. Items entered in fields on a Variable subsheet are updated simultaneously on corresponding fields on the Variable Sheet.

```
(s) Status: L 61 /
===== VARIABLE: i =====
Status: L
First Guess:
Associated List: i
Input Value: 17
Output Value:
Display Unit: percent
Calculation Unit: monthly rate
Comment: Interest rate
```

#### Access:

A Variable subsheet is accessed by placing the cursor anywhere on the row on the Variable Sheet for the variable to be examined and typing the Dive command (>). Typing the Return command (<) returns to the Variable Sheet.

If a list name has been assigned to the Associated List field, the associated List subsheet can be reached by diving again with the cursor positioned on any field on the Variable subsheet.

#### Initial Cursor Position:

The Status field

**Fields:**

The variable defined on a Variable subsheet is described by eight header fields, with a label and a field on each line. Fields appearing on both a Variable subsheet and the Variable Sheet behave in the same way unless otherwise noted. The Variable subsheet fields are:

**Status: (s)**

[Option/Output]

Same as the Status field on the Variable Sheet, except that the “G” option is spelled out as “Guess”.

**First Guess: (f)**

[Value Entry]

Provides a starting guess value for automatic use of the Iterative Solver. The first guess value is used if the Automatic Iteration ON field on the Global Sheet is set to “Yes” and the Direct Solver fails to find a value for the variable.

A first guess value can be any valid numeric value or expression. If Calculation and Display units are assigned for the variable, changing the Display unit name immediately converts the guess value to the new Display unit. Automatic iteration is most useful if the problem values are within a narrow range.

**Associated List: (a)**

[Name Entry]

Associates the variable with a list for use by the List Solver. When **L** is typed in the Status field on the Variable Sheet or a Variable subsheet, the variable name is assigned as the list name by default. If the list name does not previously exist, it is inserted at the bottom of the Name column on the List Sheet. The default list name can be changed to any other list name by typing the new name into the Associated List field. Entering a list name in the Associated List field automatically generates an “L” in the Status field for both the Variable Sheet and subsheet.

An association may be cleared by blanking the Associated List field or by typing a second “L” in the Status field on either the Variable subsheet or the Variable Sheet. Once inserted on the List Sheet, a list remains even if the list association is removed.

## Variable Subsheet

### **Input Value: (i)**

[Value Entry]

Same as the Input field on the Variable Sheet except that the value is displayed with up to 12 significant digits.

### **Output Value: (o)**

[Output]

Same as the Output field on the Variable Sheet except that the value is displayed with up to 12 significant digits.

### **Display Unit: (d)**

[Unit Name Entry]

Same as the latest unit name entry in the Unit field on the Variable Sheet. Displays the current Display unit for the variable. The Display unit can be assigned on the Variable Sheet by changing the unit name in the Unit field, or on the Variable subsheet by typing directly into the the Display Unit field. The Display unit on a Variable subsheet should not be blanked without also blanking the Calculation unit. Each time the Display unit is changed and the conversion is defined on the Unit Sheet, the corresponding input and output and first guess values are converted.

### **Calculation Unit: (u)**

[Unit Name Entry]

Same as the first unit name entry in the Unit field on the Variable Sheet. Displays the current Calculation unit for the variable. The Calculation unit can be assigned on the Variable Sheet as the first unit name entered in the Unit field, or on the Variable subsheet by typing directly into the the Calculation Unit field. The Calculation unit on a Variable subsheet should not be blanked without also blanking the Display unit.

### **Comment: (c)**

[Comment Entry]

Same as the comment field on the Variable Sheet. It is not used in solutions.

## The Rule Sheet

The Rule Sheet contains the equations for a model. Each row can contain one equation and its status, as well as a comment.

```
(1r) Rule: H=p+d                                     61 /
===== RULE SHEET =====
S Rule
-----
H=p+d
d=H*d

A=p*(i/(1-(1+i)^-n))
T=A*n
Tip=T-p


```

### Access:

The Rule Sheet appears in the bottom window at program startup. It can be displayed in the current window by typing =**R**, or in a new window by typing /**WR** from a one-window display.

### Initial Cursor Position:

The Rule field on the first row

### Fields:

Each rule is described by a row containing two fields located in columns under the labels in the Label line at the top of the sheet. The Rule Sheet fields are:

**S [Status] (s)**

[Output]

Used by the program to provide a message about the status of the rule in the corresponding row. An explanation of the character

## Rule Sheet

displayed in a Status field can be viewed on the Status line at the top of the screen by moving the cursor to the Status field in question.

- \* An asterisk appears in the Status field whenever a rule is entered or modified, and indicates that the equation is unsatisfied.

The Status field is blank if the Rule field contains only a comment, or when solution is successful and no error conditions arise.

- › An error symbol appears in the Status field if an error is found during solution. Errors can involve function arguments, division by zero, inconsistent variable values within a rule, or contradictory rules.

Further explanation of error messages is provided in the appendix on Error Messages.

### **Rule (r)**

[Rule Entry]

Accepts equations according to the conventions for valid rules. (See the Mathematics section for more information concerning valid rules.)

A rule field can contain an equation, a comment, or both. All text after a quotation mark (") in a Rule field is regarded as a comment.

A quotation mark in front of a rule excludes a rule from being evaluated by the solvers.

If the Variable Insert ON field on the Global Sheet is set to "Yes" (the default setting), any variables not already on the Variable Sheet are automatically inserted at the bottom of the Name column on the Variable Sheet when a rule is entered. Once assigned, the variable names remain even if they are removed from the equations on the Rule Sheet.

Error messages appear on the Prompt/Error line for any invalid rule entries. User functions must be defined on the User Function Sheet before they can be used in rules.

See the Mathematics section for more information concerning rule evaluation.

## The Unit Sheet

The Unit Sheet contains definitions for conversions between units of measurement used in a model. Each row on the Unit Sheet describes a conversion between two units of measure.

```
(4m) Multiply By: 61 /
===== UNIT SHEET =====
From      To      Multiply By  Add Offset
-----
percent   decimal_f .01
percent   monthly_r 8.3333333E-4
years     months   12
```

A unit conversion must be defined on the Unit Sheet before it can be used on the Variable or List Sheets or subsheets. Once a conversion is defined, it can be used in either direction. Two units can be used in conversions even though their conversion is not explicitly defined on the Unit Sheet, if they are each defined in terms of a third common unit.

The management of units is left to the user. It is recommended that all variables on the Variable Sheet be assigned unit names at the same time, prior to value assignment. Since rule evaluation is performed in Calculation units, it is important to assign Calculation units for variables so that the results of solutions are consistent among related variables. Check the Display unit, and change it if necessary before you enter any input value. It is necessary to assign units to a variable before associating that variable with a list, if unit conversions are being used. The Storage unit of a list

## Unit Sheet

should have a conversion defined on the Unit Sheet with the Calculation unit of its related variable or else be identical. If a function argument value is assigned to a variable, the Display units for the domain and range lists must be the same as the Calculation unit for the variable.

For further information on unit conversions, see the Mathematics section.

### **Access:**

The Unit Sheet can be displayed in the current window by typing = **U** or in a new window by typing /**WU** from a one-window display.

### **Initial Cursor Position:**

The From field on the first row

### **Fields:**

Each unit conversion is described by a row containing four fields, located in columns under the labels in the Label line at the top of the sheet. The Unit Sheet fields are:

#### **From (f)**

[Unit Name Entry]

Names the first unit. A single value of the first unit equals the specified multiple of the second unit.

See the Overview section for valid unit name entries.

#### **To (t)**

[Unit Name Entry]

Names the second unit. The second unit is defined in terms of a single first unit.

#### **Multiply By (m)**

[Value Entry]

Defines the number of second units necessary to equal a single first unit. A Multiply By factor entry can be any valid numeric value or expression. If the field is left blank, one second unit equals one first unit.

**Add Offset (a)**

[Value Entry]

Defines any difference between the zero points of the scales used for the two units. This offset value must be in terms of the second unit. For example, converting from degrees Centigrade to degrees Fahrenheit requires an offset of 32. The addition to align the scales is performed after the multiplication. An offset item can be any valid numeric value or expression. If the field is left blank, the offset is assumed to be zero.

## Global Sheet

### The Global Sheet

The Global Sheet contains global settings and program defaults. Each row contains one global setting labeled on the left. Different computer systems have different default values; for more information about the global settings for your system, see the Hardware Reference section.

```
(v) Variable Insert ON: Yes                               61 /
===== GLOBAL SHEET =====
Variable Insert ON:   Yes
Intermediate Redisplay ON: Yes
Automatic Iteration ON: Yes
Comparison Tolerance: .000001
Typical Value:       1
Maximum Iteration Count: 10

Page Breaks ON:      Yes
Page Numbers ON:     Yes
Form Length:         66
Printed Page Length: 60
Printed Page Width:  80
Left Margin:         0
Printer Device or Filename: PRN
Printer Setup String:
Line End (CR/LF or CR): CR/LF
```

#### Access:

The Global Sheet can be displayed in the current window by typing =G or in a new window by typing /WG from a one-window display.

#### Initial Cursor Position:

The Variable Insert ON field

#### Fields:

The global settings for the program are described by fifteen header fields, with a label and a field on each line. The Global Sheet is organized in three sections: the first section contains one field affecting the entry of variables; the second section contains five fields affecting the use of the solvers; the third section contains nine fields affecting use of the printer. The printer options

are used by the Print command (/P) and the Plot and Table sheets when their Printer options are selected. The Global Sheet fields are:

**Variable Insert ON: (v)**

[Required Option]

Default: Yes

**Y** Yes Option**N** No Option

Controls automatic insertion of the variables used in rules onto the Variable Sheet. If this field is set to “Yes”, the program automatically enters variable names not already on the Variable Sheet at the bottom of the Name column on the Variable Sheet when an equation is entered on the Rule Sheet. Once inserted, variable names remain on the Variable Sheet even if they are edited or blanked from the Rule Sheet. If this field is set to “No”, variable names must be inserted by hand on the Variable Sheet.

**Intermediate Redisplay ON: (r)**

[Required Option]

Default: Yes

**Y** Yes Option**N** No Option

Controls the display of intermediate values during Iterative and list solution. If the field is set to “Yes”, the results for intermediate values are displayed on the Variable Sheet during solution. If the field is set to “No”, the screen retains the initial values until the end of the solution and is updated at the end of Iterative solution only. The status message with the element number or the iteration step is still displayed on the Prompt/Error line. Solution is faster when the field is set to “No”.

**Automatic Iteration ON: (a)**

[Required Option]

Default: Yes

**Y** Yes Option**N** No Option

Allows automatic iteration to be used when a Direct solution fails. If the field is set to “Yes” and first guess values are assigned on Variable subsheets, the Iterative Solver is called automatically

after the Direct Solver fails to reach a solution. If the field is set to “No” or if first guess values have not been entered, solution is terminated when the Direct Solver fails.

**Comparison Tolerance: (c)**

[Required Numeric Value Entry]

Default: .000001

Defines equality for variables and expressions during solving. Two values are considered equal if the absolute value of their difference divided by the larger of the two values is less than the comparison tolerance. A comparison tolerance can be any positive valid numeric value or expression.

See the Mathematics section for more information.

**Typical Value: (t)**

[Required Numeric Value Entry]

Default: 1

Used in conjunction with the comparison tolerance to determine at what point values are considered near zero. The product of the typical value and the comparison tolerance determines this point. Whenever values are equal to or less than this product, the denominator of the comparison tolerance formula is changed to the typical value. As the typical value increases in size for a constant comparison tolerance, there is a concurrent decrease in the precision of equality determination. A typical value can be any positive valid numeric value or expression.

See the Mathematics section for more information.

**Maximum Iteration Count: (i)**

[Required Integer Entry]

Default: 10

Defines the maximum number of iterations to be tried by the Iterative Solver before it stops. Any integer between 1 and 10000 may be assigned, although values higher than 20 are not recommended.

See the Mathematics section for more information.

**Page Breaks ON: (b)**

[Required Option]

Default: Yes

Y Yes Option

N No Option

Defines page breaks for the Print command and display of tables. If the field is set to “Yes”, the output from the Print command (/P) or the Table Sheet (!) is printed with breaks after each printed page length. Table displays repeat any specified title and headers on each page.

If the field is set to “No”, output is generated as a single continuous sheet. When page breaks are off: page numbers and form length settings are ignored; titles and headers for vertical tables appear at the top of the listing; titles for horizontal tables appear in between each segment of lists, and headers appear on every line.

**Page Numbers ON: (n)**

[Required Option]

Default: Yes

Y Yes Option

N No Option

Controls page numbering for the Print command, and plot and table displays. If the field is set to “Yes”, the output from the Print command (/P) or the Plot and Table Sheets (!) are printed with page numbers on the bottom of each page, preceded by the word “Page”. When page numbers are used, the amount of printed material is decreased by two lines at the bottom of each page. If the field is set to “No”, or page breaks are not in effect, page numbers are not used.

**Form Length: (f)**

[Required Integer Entry]

Default: 66

Defines the total number of lines on each sheet of paper. Any integer between 6 and 1000 may be assigned. The amount of space left between pages is determined by the form length minus the printed page length. If the printed page length is greater than or equal to the form length, the printed page length value is used and no space is left between pages. On perforated paper, leave a margin at the top and bottom of the page by dividing in half the difference between printed page length and the form length and moving the printer head down that number of lines below the perforations.

When page breaks are off, the form length setting is ignored.

### **Printed Page Length: (l)**

[Required Integer Entry]

Default: 60

Defines the number of lines available for printing on each page. Any integer between 6 and 1000 may be assigned. The setting for printed page length must be less than or equal to form length. If the printed page length is greater than the form length, the form length is ignored.

The amount of information printed per page is decreased from the printed page length setting by two lines each for page numbers and titles that were entered on the Table or Plot sheets.

The length of a printed plot can be increased or decreased by adjusting the printed page length, whether page breaks are on or not.

### **Printed Page Width: (w)**

[Required Integer Entry]

Default: 80

Defines the number of character positions available to the program for printing across the width of the page. Any integer between 6 and 200 may be assigned. The page width and the left margin setting together should be less than or equal to the physical width of the paper being used or information on the right end of each line will be lost.

### **Left Margin: (m)**

[Required Integer Entry]

Default: 0

Defines the number of character positions to indent before the first characters of the line are printed. Any integer between 0 and 200 may be assigned. The left margin setting plus the printed page width should be smaller than or equal to the width of the paper or information on the right end of each line will be lost.

### **Printer Device or Filename: (d)**

[Filename Entry]

Default: [Computer-dependent]

Defines the device or filename to which output is printed. The device or filename is used for output from the Print command (/P) or the Plot Sheet and Table Sheets (!) when their “Screen or Printer” fields are set to “Printer”.

Most operating systems have a reserved name used to refer to the attached printing device. (See the Hardware Reference section for the default option and the manual that accompanies your operating system for more information.) Any other file specification directs output to the named file. Valid file specifications are determined by the operating system. (See the Hardware Reference section or your operating system manual for more information.) Error messages are provided when an invalid file specification is used, not when it is entered. (See the appendix on Error Messages for more information about error messages.)

If the field is blanked, the output is displayed on the screen using the printer options set in the other printer definition fields. The Enter key must be pressed to display each new page and return to the previous display after the last page.

**Printer Setup String: (s)**

[Setup String Entry]

Defines a special string sent to direct the printer at the beginning of any printing action. A setup string is normally defined by a series of printable ASCII characters and control code sequences. They are represented in the TK!Solver program by special conventions:

- ^ Indicates the beginning of a control code sequence
- ^^ Sends the caret character itself (^)
- ^R Sends a carriage return character
- ^L Sends a line feed character
- ^C Precedes a Control character
- ^E Represents an Escape character
- ^H Precedes a two-digit hexadecimal code for an ASCII character

For example, Control A is represented by “^CA”, and “^CA” is equivalent to “^H01”. Some printer settings remain in effect until explicitly turned off in a subsequent print action with a newly specified printer setup string.

See the Hardware Reference section for more information concerning setup strings.

**Line End: (e)**

[Required Option]

Default: [Computer-dependent]

**&** Carriage Return & Line Feed

**C** Carriage Return

**L** Line Feed

**S** System-defined

Defines the line end sequence to be used for the printer. Systems that do not use carriage return or line feed have a default defined for that system. If the default is **&**, **C**, or **L**, the **S** option provides the correct code to the printer. If the default option is **S**, it cannot be changed.

This field is not changed by the Reset command (/R) and it is not saved or loaded as part of a model. See the Hardware Reference section for the default for your computer. See your hardware and printer manuals for more information about the sequence required by your printer.

## The List Sheet

The List Sheet contains a table of all lists created in the program. A list is a set of values assigned to positional element numbers. Lists can have blank values. Each row can contain the description for one list. There is a List subsheet displaying more information about each list; items entered in fields on the List Sheet are updated simultaneously on corresponding fields on List subsheets.

```
(1n) Name: i                                     61 /!
```

===== LIST SHEET =====			
Name	Elements	Unit	Comment
-----	-----	-----	-----
i	15	percent	
A	15	dollars	
T	15	dollars	

### Access:

The List Sheet can be displayed in the current window by typing =L or in a new window by typing /WL from a one-window display.

A List subsheet for a list can be reached by typing the Dive command (>) with the cursor on any field on the list row.

### Initial Cursor Position:

The Name field on the first row

### Fields:

Each list is described by a row containing four fields, located in columns under the labels at the top of the sheet. The List Sheet fields are:

## List Sheet

### **Name (n)**

[Name Entry]

Accepts a unique name for a list. A list must be named if it is to be associated with a variable or be considered a domain or range list.

A list name is automatically inserted at the bottom of the Name column, if it does not previously exist and:

- an **L** option is typed into the Status field on the Variable Sheet or Variable subsheet
- a list name is assigned to the Associated List field on a Variable subsheet
- a list name is assigned to the Domain or Range fields on the User Function Sheet
- a list name is entered in the Domain List or Range List fields on a User Function subsheet
- a list name is entered in the X-Axis or any Y-Axis field on the Plot Sheet
- a list name is entered in a List field on the Table Sheet

### **Elements (e)**

[Output]

Used by the program to indicate the number of the last element in a list. An integer representing the number of the last element is automatically assigned when values are entered on a List subsheet or a User Function subsheet.

### **Unit (u)**

[Unit Name Entry]

Assigns the Storage and Display unit names for the list. If the Unit field is blank, the first unit name entered becomes both the Storage and Display unit for the list. Any subsequent entry changes only the Display unit. The Blank command (/B) can be used on the List Sheet to blank both the Storage and Display units for a list simultaneously.

If Display and Storage units for a list are different, their unit conversion must be defined on the Unit Sheet, or the two units must be defined in terms of a common third unit. If the conversion is defined, the numeric values in the Value column on the List subsheet are immediately converted to their values in the Display unit whenever the Display unit name is changed.

If a list is associated with units, both a Storage and Display unit should be defined at all times, although they can be identical. If the Storage unit name has been blanked from the List subsheet without also blanking the Display unit, the unit name appears preceded by a question mark (?). If the Display unit has been independently blanked, the unit name appears followed by a question mark.

If a list is associated with a variable, its Storage unit must have a defined relationship to the Calculation unit of the variable, or the results may be unreliable.

For more information about unit conversion, see the Mathematics section.

**Comment (c)**

[Comment Entry]

Accepts a comment or description of the list. Any characters entered into this field are ignored by the program in reaching solutions.

## List Subsheet

### List Subsheets

Each List subsheet contains a detailed description of one list defined on the List Sheet. There is one List subsheet for each list on the List Sheet, whether named or not. Items entered in fields on a List subsheet are updated simultaneously on corresponding fields on the List Sheet.

```
(1v) Value: 12 61 / 1
===== LIST: i =====
Comment:
Display Unit:          percent
Storage Unit:         monthly rate
Element Value
-----
1      12
2      12.5
3      13
4      13.5
5      14
6      14.5
7      15
8      15.5
9      16
10     16.5
11     17
12     17.5
13     18
14     18.5
15     19
```

A List subsheet is created for each of the two lists on a User Function subsheet. Values entered in the domain and range lists on a User Function subsheet are automatically entered in the Value fields of the two corresponding List subsheets. Changes made to either the List subsheets or User Function subsheet appear on all corresponding sheets and subsheets.

#### Access:

A particular List subsheet is accessed by placing the cursor anywhere on the row on the List Sheet for the list to be examined and typing the Dive command ( $\downarrow$ ). Typing the Return command ( $\leftarrow$ ) returns to the List Sheet.

A List subsheet can also be accessed by typing the Dive command ( $\downarrow$ ) from:

- any field on a Variable subsheet with an associated list
- the Domain List, Range List, Domain, and Range fields of a User Function subsheet
- the X-Axis, Y-Axis, or Character fields on the Plot Sheet
- any row in the Field Area of the Table Sheet

**Initial Cursor Position:**

The first Value field

**Fields:**

The list defined on a List subsheet is described by three header fields at the top of the sheet, with a field label and a field on each line. Each list element is described by a row in the Field Area containing two fields in columns under labels. The values in a list are identified by an element number automatically generated by the program on the left side of the sheet as values are entered. Corresponding fields appearing on a List subsheet and a User Function subsheet or the List Sheet behave in the same way unless otherwise noted. The List subsheet fields are:

**Comment: (c)**

[Comment Entry]

Same as the Comment field on the List Sheet.

**Display Unit: (d)**

[Unit Name Entry]

Same as the latest entry in the Unit field on the List Sheet. Defines the unit the values of a list are currently displayed in. The Display unit can be assigned on the List Sheet by changing the previous unit name in the Unit field or on the List subsheet by typing directly into the the Display Unit field. The Display unit on a List subsheet should not be blanked without also blanking the Storage unit. Each time the Display unit is changed, the values in the Value column are automatically converted.

If the List subsheet is for the domain or range list for a user function, the functional relationship is determined strictly in the Display unit values.

## List Subsheet

### **Storage Unit: (s)**

[Unit Name Entry]

Same as the first entry in the Unit field on the List Sheet. Defines the unit in which the values of a list are stored and the unit converted to Calculation units when associated with variables. The Storage unit can be assigned on the List Sheet as the first unit name entered in the Unit field, or on the List subsheet by typing directly into the the Storage Unit field. The Storage unit on a List subsheet should not be blanked without also blanking the Display unit.

Lists saved with the Save File (/SS) and DIF Save (/S#S) options of the Storage command are saved with both Storage and Display units.

### **Element**

[Restricted Output]

Element numbers are automatically and sequentially assigned as values are entered. If a value is blanked, its element number is not displayed but the row remains reserved for it. If a value is deleted, its element number is reassigned to the next undeleted value. Some built-in functions and certain options of the List command use the element number to refer to a list value. The cursor cannot be moved to the Element column.

### **Value (v)**

[Value Entry/Output]

Assigns a numeric or symbolic value to a list element. A value entry can be any valid TK!Solver expression.

If the conversion is not defined between the Display unit and Storage unit on the Unit Sheet, a question mark (?) precedes values expressed in the Storage unit in the Value column.

Numeric values may be automatically generated by assigning a numeric value to the first element and the last element, and using the Action command (!Y) to calculate the intermediate values. Values may also be filled in as a result of the action of the Solve or Block option of the List command if the list is designated as an output list.

See the Mathematics section for further information concerning use of lists in solutions.

Values in a list corresponding with a domain or range list on a User Function subsheet are updated simultaneously on both List subsheets.

### The User Function Sheet

The User Function Sheet contains a table of all functions defined by the user. A function can be defined as the relationship between two sets of values, the domain and the range. Each row can contain the description for one user function. There is a User Function subsheet displaying more information about each user function; the values for the domain and range are displayed on a User Function subsheet and two separate List subsheets. Items entered in fields on the User Function Sheet are updated simultaneously on corresponding fields on User Function subsheets.

(1r) Range: interest 61 / 1

```
===== USER FUNCTION SHEET =====
Name      Domain      Mapping Range      Comment
-----
bank      name          Table      interest
```

**Access:**

The User Function Sheet can be displayed in the current window by typing =F or in a new window by typing /WF from a one-window display.

A User Function subsheet for a user function can be reached by typing the Dive command (>) with the cursor on any field on the function row.

The List subsheet associated with the domain or range of a user function can be reached from the User Function Sheet by placing the cursor on any field in the row and typing the Dive command

( $\rangle$ ) twice, once into the User Function subsheet, and a second time into the List subsheet for either the domain or range list. The domain List subsheet can be reached from the User Function Subsheet by placing the cursor on the Domain List field or on any value in the Domain column and typing the Dive command ( $\rangle$ ). The range List subsheet can be reached from either the Range List field or any value field in the Range column.

**Initial Cursor Position:**

The Name field on the first row

**Fields:**

Each user function is described by a row containing five fields, located in columns under the labels in the Label line at the top of the sheet. The User Function Sheet fields are:

**Name (n)**

[Name Entry]

Accepts a unique user function name. A user function may remain unnamed. A user function must be defined before it can be used in equations or expressions. The function name cannot duplicate a built-in function name; built-in function names, unlike user function names, are case insensitive. For example, “func1” and “FUNC1” are two different user functions, while “sin” and “SIN” both refer to the same built-in sine function.

**Domain (d)**

[Name Entry]

Accepts a list name to designate the domain of the function. The domain contains the values for the argument of the function. If the list name does not previously exist, it is inserted at the bottom of the Name column on the List Sheet and in the Domain List field on the corresponding User Function subsheet.

Any units for a domain list must be entered on the List Sheet or List subsheet for the domain list. Functional relationships are determined only for Display unit values.

**Mapping (m)**

[Required Option]

Default: Table

**T** Table Option

**S** Step Option

**L** Linear Option

Defines the interpolation relationship between the domain and range values. The Table lookup option is automatically displayed in the field as soon as an item is entered in any field on the row. It may be changed to any of the other options but the field cannot be blanked.

Table lookup and linear interpolation are possible in either direction between the domain and range. Step functions can only be interpolated from the domain to the range.

Values supplied as function arguments for user functions must fall within the bounds of the specified list. Both lists should be the same length because the additional elements in the longer list are ignored.

If the exact value of the function argument for the user function is in the specified list, it maps directly to the corresponding value in the other no matter which mapping option is selected. If the exact value of the function argument is not a value in the specified list:

- table lookup returns an error,
- linear interpolation calculates a value in between the values in the corresponding list,
- the step function chooses the value in the range list corresponding to the previous value in the domain list.

### **Range (r)**

[Name Entry]

Accepts a list name to designate the range of the function. The range list contains the function values. If the list name does not previously exist, it is inserted at the bottom of the Name column on the List Sheet and in the Range List field on the corresponding User Function subsheet.

Any units for a range list must be defined on the List Sheet or List subsheet for the range list. Functional relationships are determined only for Display unit values.

**Comment (c)**

[Comment Entry]

Accepts a comment or description of the user function. Any characters entered into this field are ignored by the program in reaching solutions.

### User Function Subsheets

Each User Function subsheet contains a detailed description of one function defined on the User Function Sheet. There is one User Function subsheet for each function on the User Function Sheet whether named or not. Items entered in fields on a User Function subsheet are updated simultaneously on corresponding fields on the User Function Sheet.

Each User Function subsheet is associated with two List subsheets, one for the domain and one for the range, containing the same values as the User Function subsheet. Values entered in the domain and range lists on a User Function subsheet are automatically entered in the Value fields of the corresponding List subsheets.

Domain and range values should be expressed in Display units. Functional relationships are determined strictly by the Display unit values.

```
(10r) Range: 15.5 61 / 1  
===== USER FUNCTION: bank =====  
Comment:  
Domain List: name  
Mapping: Table  
Range List: interest  
Element Domain Range  
-----  
1 'first_town 12.5  
2 'new_town_b 14  
3 'old_town_b 18  
4 'mid_town_b 13.5  
5 'little_tow 16  
6 'big_town_b 19  
7 'tiny_town_ 14.5  
8 'huge_town_ 18  
9 'small_town 17.5  
10 'town_bank 15.5
```

**Access:**

A particular User Function subsheet is accessed by placing the cursor anywhere on the row on the User Function Sheet for the function to be examined and typing the Dive command (>). Typing the Return command (<) returns to the User Function Sheet.

The List subsheet associated with the domain can be reached by placing the cursor on the Domain List field or on any value in the Domain column, and typing the Dive command (>). The range List subsheet can be reached from the Range List field or any value in the Range column.

**Initial Cursor Position:**

The first Domain field

**Fields:**

The user function defined on a User Function subsheet is described by four header fields at the top of the sheet, with a field label and a field on each line. Each function element is described by a row in the Field Area containing three fields in columns under the Label line. The domain and range values are identified by an element number automatically generated by the program on the left side of the sheet as values are entered in either column. Fields appearing on both a User Function subsheet and the User Function Sheet behave in the same way unless otherwise noted. The User Function subsheet fields are:

**Comment: (c)**

[Comment Entry]

Same as the Comment field on the User Function Sheet.

**Domain List: (f)**

[Name Entry]

Same as the Domain field on the User Function Sheet. If the list name does not previously exist, it is inserted at the bottom of the Name column on the List Sheet and in the Domain field on the User Function Sheet.

**Mapping: (m)**

[Required Option]

Default: Table

**T** Table Option

**S** Step Option

**L** Linear Option

Same as the Mapping field on the User Function Sheet, except that diving to a User Function subsheet for an unnamed function assigns the default Table mapping option automatically.

### **Range List: (t)**

[Name Entry]

Same as the Range field on the User Function Sheet. If the list name does not previously exist, it is inserted at the bottom of the Name column on the List Sheet and in the Range field on the User Function Sheet.

### **Element**

[Restricted Output]

Element numbers are automatically assigned as domain and range values are entered. The cursor cannot be moved to the Element column.

### **Domain (d)**

[Value Entry/Output]

Assigns a numeric or symbolic value to a domain list element. A domain value entry can be any valid numeric value or expression, or a symbolic value.

Numeric values may be automatically assigned by defining the first and last values for the domain list and using the Action command (!Y) to calculate the intermediate values. The cursor must be placed on a field in the Domain column. Values may also be filled in as a result of the action of the solvers /L! and /LB if the list is designated as an output list.

The values for the domain list may be entered in the Domain column on the User Function subsheet or on the corresponding List subsheet. They are automatically entered on the other subsheet.

See the Overview section for further information concerning valid value entries.

**Range (r)**

[Value Entry/Output]

Assigns a numeric or symbolic value to a range list element. A range value entry can be any valid numeric value or expression, or a symbolic value.

Numeric values may be automatically assigned by defining a first and last value for the range list and using the Action command (!Y) to calculate the intermediate values. The cursor must be placed on a field in the Range column. Values may also be filled in as a result of the action of the solvers /L! and /LB if the list is designated as an output list.

The values for the range list may be entered in the Range column on the User Function subsheet or on the corresponding List subsheet. They are automatically entered on the other subsheet.

See the Overview section for further information concerning valid value entries.

## Plot Sheet

### The Plot Sheet

The Plot Sheet contains information necessary to plot values in one list against the values in all other specified lists. Each row in the Field Area can contain the description for one list.

A plot is produced from the Plot Sheet with the Action command (!).

```
(1c) Character: #                               61 /!  
===== PLOT SHEET =====  
Screen or Printer:          Screen  
Title:  
Display Scale ON:          Yes  
X-Axis:                     i  
Y-Axis      Character  
-----  
T          #   
Tip        $
```

#### Access:

The Plot Sheet can be displayed in the current window by typing =P or in a new window by typing /WP from a one-window display.

A List subsheet for lists described on the Plot Sheet can be reached by placing the cursor on the X-Axis field or any field on a Y-axis row and typing the Dive command ()).

#### Initial Cursor Position:

The Screen or Printer field

**Fields:**

The plot defined on the Plot Sheet is described by four header fields at the top of the sheet, with a field label and a field on each line. Each Y-axis list on the plot is described by a row in the Field Area containing two fields in columns under the Label line. The Plot Sheet fields are:

**Screen or Printer: (s)**

[Required Option]

Default: Screen

**S** Screen Option**P** Printer Option

Designates the destination for the plot output. If the field is set to “Screen”, the plot is displayed on the screen in a format compatible with the video screen size. The printer definitions on the Global Sheet are ignored. To return to the previous display after the plot, the Enter key must be pressed.

If the field is set to “Printer”, the plot is formatted using the printer options set on the Global Sheet and sent to the specified printer device or filename. The output is displayed on the screen while it is being sent to the printer or file. The Printed Page Width and Printed Page Length fields on the Global Sheet can be used to increase or decrease the size of the printed plot.

If the Printer Device or Filename field on the Global Sheet is blank, the output is only displayed to the screen, with the specified printer formats. The Enter key is required to return to the previous display after the plot.

The Break key cancels the plot at any point, produces an error signal and returns immediately to the previous display.

**Title: (t)**

[Comment Entry]

Specifies an optional plot title to be displayed at the bottom of a plot. If specified, a title decreases the plot length by two lines. If the title is wider than the screen or the Printed Page Width field on the Global Sheet, it is truncated.

## Plot Sheet

### **Display Scale ON: (d)**

[Required Option]

Default: Yes

**Y** Yes Option

**N** No Option

Controls whether an approximate scale is displayed on the axes of the plot. If the field is set to “Yes”, a scale of numbers is printed along the X-axis and Y-axis to indicate an intermediate set of numbers evenly distributed between the determined end points. The numbers on the X-axis are determined from the largest and smallest values in the X-axis list. The numbers on the Y-axis are determined from the largest and smallest values in all Y-axis lists. If the difference between the largest and smallest values for either axis is greater than 1E120, the word “=ERROR=” displays instead of the scaling numbers on the axes.

The length of each axis and the scale numbers displayed are affected by the size of the plot. The size of the plot is determined by screen size when the “Screen” option is selected, or by the Printed Page Length and Printed Page Width settings on the Global Sheet when the “Printer” option is selected.

If the field is set to “No”, the X-axis and Y-axis are displayed with tic marks and no scaling numbers, allowing a slightly larger image for the plot.

### **X-Axis: (x)**

[Name Entry]

Designates a single list as the values for the horizontal coordinates (abscissa values). If the list name does not previously exist, it is inserted at the bottom of the Name column on the List Sheet. An X-axis list name must be specified to produce a plot.

### **Y-Axis (y)**

[Name Entry]

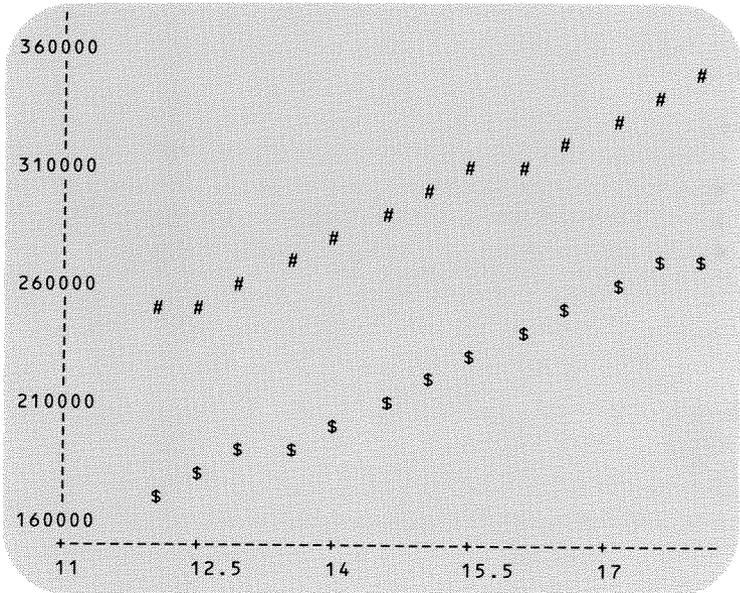
Designates the lists to be used as vertical coordinates (ordinate values), which in conjunction with the abscissa values determine the points of the plot. If the list name does not previously exist, it is inserted at the bottom of the Name column on the List Sheet. If any specified Y-axis list is longer than the X-axis list, only the corresponding number of values is plotted. At least one Y-axis list name must be specified in order to plot values.

**Character (c)**

[Option]

Default: \*

Designates the character to be used to plot the points associated with the Y-axis list specified on the same row. The default character is automatically supplied by the program as Y-axis list names are entered. It can be changed to any other displaying character or blanked to display nothing. A new character is accepted without pressing the Enter key. Command characters (? / = ; > < !) may only be used if edited into the field with the Edit command (). For overlapping plot points, only the character for the last list plotted is displayed.



Plot generated from Plot Sheet shown above.

### The Table Sheet

The Table Sheet contains information necessary to display a table of list values. A table can be defined as a display or printout of designated values from all specified lists. Each row in the Field Area can contain the description for one list.

A table is produced from the Table Sheet with the Action command (!).

```
(1w) Width: 10                                     61 / !
===== TABLE SHEET =====
Screen or Printer:                               Screen
Title:                                           Sample Table
Vertical or Horizontal:                         Vertical
List      Width First Header
-----
i         10   1   Rate
A         18   1   Monthly Payment
T         18   1   Total Payments
```

#### Access:

The Table Sheet can be displayed in the current window by typing =**T** or in a new window by typing /**WT** from a one-window display.

The List subsheet for a list described in the Field Area can be reached by placing the cursor on any field in the row and typing the Dive command (>).

#### Initial Cursor Position:

The Screen or Printer field

**Fields:**

The table defined on the Table Sheet is described by three header fields at the top of the sheet, with a field label and a field on each line. Each list on the table is described by a row in the Field Area containing four fields in columns under the Label line. The Table Sheet fields are:

**Screen or Printer: (s)**

[Required Option]

Default: Screen

**S** Screen Option

**P** Printer Option

Designates the destination for table output. If the field is set to “Screen”, the table is displayed on the screen in a format compatible with the video screen size. The printer definitions on the Global Sheet are ignored. To see a subsequent page of information or to return to the previous display after the last page of a table, the Enter key must be pressed.

If the field is set to “Printer”, the table is formatted using the printer options set on the Global Sheet and sent to the specified printer device or filename. The output is displayed on the screen while it is being sent to the printer or file without waiting for the Enter key to go from page to page, or return to the previous display.

If the Printer Device or Filename field on the Global Sheet is blank, the output is only displayed to the screen, with the specified printer formats. The Enter key is required to see a subsequent page of information or to return to the previous display after the last page of a table.

The Break key cancels the table at any point, produces an error signal, and returns immediately to the previous display.

**Title: (t)**

[Comment Entry]

Specifies an optional title to be displayed at the top of a table. If specified, a title decreases the table page length by two lines. If the title is wider than the screen or the Printed Page Width field on the Global Sheet, it is truncated. The title is printed again at the top of each page or segment of lists.

## Table Sheet

### **Vertical or Horizontal: (v)**

[Required Option]

Default: Vertical

**V** Vertical Option

**H** Horizontal Option

Designates the orientation of the table.

Vertical tables organize the values in columns under headers printed across the top of the page. If a vertical table has more lists than can be printed on a single page, all values for the lists on the first page are printed before printing any other lists.

Horizontal tables display the headers to the left of the values. If a horizontal table has more values in any list than can be printed on a single page, the same number of values for all lists are printed before printing additional values.

Lists of varying lengths may be printed on the same table. The length allowed for all lists is determined by the longest list.

### **List (l)**

[Name Entry]

Assigns the lists of values to be displayed in rows or columns. If the list name does not previously exist, it is inserted at the bottom of the Name column on the List Sheet. If the list has no values, it is printed as a blank column under its header. A blank row inserted in the list of list names is ignored when a table is produced. At least one list must be specified with assigned values to produce a table.

### **Width (w)**

[Required Integer Entry]

Default: 10

Designates the display width for the values in the list. There is a one-character space left automatically between columns.

A default width of 10 is automatically supplied by the program whenever an item is entered in any other field on the row. The width setting may be changed to any integer from 3 to the width of the screen or printer being used (up to 128), but the field cannot be blanked.

Each column on a vertical table prints according to the specified width settings. If more than one width value is specified for a horizontal table, the largest width value is used.

If a width value is supplied on a blank row, the default first element value is automatically entered by the program in the First field. A line containing only a width value and a first element value prints a blank column of the specified width within the table.

The number of decimal places displayed can be roughly approximated by controlling the width of the display field. In small column widths, wide numbers are displayed in scientific notation or rounded off.

**First (f)**

[Required Integer Entry]

Default: 1

Assigns the element of the list to display as the first element on the table. All list values are displayed from the specified element to the end of the list. The element number 1 is automatically supplied by the program whenever an item is entered in any other field on the row. It may be changed to any integer from 1 to the total number of elements in the list, but the field may not be blanked. If a first element value is supplied on a blank row, the default width value is automatically entered by the program.

**Header (h)**

[Comment Entry]

Specifies an optional label to display with values for a list on the same row. On vertical tables, headers are displayed above columns of values and truncated, if necessary, to fit within the specified width for each column. Care should be taken to begin headers for vertical tables with distinguishing information. If the Page Breaks ON field on the Global Sheet is set to "Yes", headers are repeated at the top of each page. If the Page Breaks ON field is set to "No", headers appear only at the top of new lists.

On horizontal tables, headers are displayed to the left of values. The first column of values is displayed to the right of the longest label in the table plus a single space. Care should be taken to make headers for horizontal tables shorter than the width of the

# TK!Solver Reference Manual

## Table Sheet

screen or the printed page width being used, minus at least the width of one column. On horizontal tables, headers are repeated for each segment of lists whether page breaks are on or off.

If no header is specified, the list name is used as the header. A header can be assigned to a row or column with no list name to print a label by itself.

Sample Table

Rate	Monthly Payment	Total Payments
12	800.4503481	240135.1044
12.5	828.6691435	248600.7431
13	857.154831	257146.4493
13.5	885.8901109	265767.0333
14	914.8583926	274457.5178
14.5	944.0438159	283213.1448
15	973.4312644	292029.3793
15.5	1003.006378	300901.9135
16	1032.755554	309826.6661
16.5	1062.665934	318799.7801
17	1092.725403	327817.621
17.5	1122.922573	336876.7719
18	1153.246757	345974.0271
18.5	1183.687954	355106.3862
19	1214.236823	364271.047

Table generated from Table Sheet shown above.

lif  
ex  
e c

fit  
A.

ru  
es  
hh  
sc  
e.  
w

p  
.  
N

OC

n  
apl  
isc

ivi  
r i



! J  
!A!

S



ost  
pdt

U



06

M

d

w

e

st

qu

es

ru



VA

tit

re

ex

ill

E





## Commands

There are two kinds of commands in the TK!Solver program: special computer keys and commands that use one or more character keys.

Cross-references to both command names and command usage are provided in the Contents section at the beginning of the Reference Manual.

Special keys are described by the names and symbols employed in this manual. For more specific information about the keys to press on your keyboard, see the Hardware Reference section.

Special keys are described for their actions in various environments:

Arrow keys	← → ↑ ↓
Enter key	◆
Break key	⊗
Backspace key	◀

Commands are alphabetized by their command names:

Action	!
Blank (field)	/B
Copy (field)	/C
Delete (row)	/D
Dive (subsheets)	>
Edit	☒
Edit Field	/E
Goto (field)	:
Help	?
Insert (row)	/I
List	/L
Move (row)	/M
Print (sheet)	/P
Quit	/Q
Reset	/R
Return (subsheets)	<
Select (sheet)	=
Solve	/!
Storage	/S
Switch (window)	;
Window	/W

## Commands

Some commands use a single keystroke and act immediately. These commands include the Arrow keys, the Dive (>), Return (<), and Switch (;) commands, and the Solve, Plot, and Table actions of the Action command (!). Of these commands, the Break key can only cancel the options for the Action command.

Some commands, called extended commands, use more than one keystroke. The extended commands in the TK!Solver program are the Select command (=), the Goto command (:), the Fill action of the Action command (!), and any command beginning with a slash (/). The Break key can be used at almost any point to cancel an extended command.

Other commands call program facilities and have their own system of user actions and responses. The Edit command (☒) and the Edit Field command (/E) call the Editor. The Help command (?) calls the Help Facility. The Break key can be used to exit from these environments.

Certain extended commands act immediately and require no further user response. These commands include the Insert (/I) and Solve (/!) commands.

Certain extended commands prompt the user to point to a final cursor position and press the Enter key (◆). These commands include the Blank (/B), Copy (/C), Delete (/D), Move (/M), Print (/P), and DIF Save option (/S#S) of the Storage command.

Certain extended commands prompt for a selection of options on the Prompt/Error line. These commands include the List (/L), Reset (/R), Select (=), Storage (/S), and Window (/W) commands.

Certain commands require confirmation with Y (Yes) or N (No). These commands include the Fill action of the Action (!) command, some places in the Help Facility (?), the Quit (/Q), and Reset (/R) commands, and deleting or overwriting with the Storage command (/S).

Certain commands require user responses to be typed on the Prompt/Error line and completed by pressing the Enter key. These commands include the Goto (:), List (/L), and Storage (/S) commands, and prompts for topics in the Help Facility (?). The Edit command (☒) can be used during these commands prior to

the final Enter key to edit user responses on the Prompt/Error line.

The Storage command (/S) employs Directory Scrolling on the Prompt/Error line to select a filename that exists.

## **Arrow Keys**

### **Command Mode:**

The Arrow keys move the cursor from its current position one field in the direction of the arrow:

←	one field left
→	one field right
↑	one row up
↓	one row down

### **Entry Mode:**

The Arrow keys enter any data typed into the current field and move the cursor to the next field in the direction of the arrow. If the item is invalid for the current field, the user is automatically placed in the Editor. An error message displays on the Prompt/Error line and the cursor remains on the same field.

### **The Editor:**

The Left Arrow key (←) moves the cue one character to the left.

The Right Arrow key (→) moves the cue one character to the right.

The Up Arrow key (↑) moves the cue to the beginning of the text.

The Down Arrow key (↓) moves the cue to the end of the text.

### **Directory Scrolling:**

The Right Arrow key (→) is used to display a filename with the Directory Scrolling feature of the Storage command (/S). It can be used alone, or after a partial file specification to select all files on a diskette that correspond to the particular Storage command option. (See the Storage command and the Hardware Reference section for more information.) Once one filename is displayed, the Right Arrow key can be used to display each subsequent filename until the one to be selected appears or until there are no more matching filenames. In systems with a hierarchical directory structure, the names of subdirectories are displayed after the last filename.

## Enter Key

### ◆ Enter Key

#### **Program:**

The Enter key (◆) starts the program (or stops it in the event of insufficient memory or a disk error) from the initial screen.

The Enter key also restarts the program after a disk or printer error once the program has begun.

#### **Entry Mode:**

The Enter key enters the characters typed into the current field as permanent data, if they constitute a valid item for the current field. If the item is invalid for the current field, the user is automatically placed in the Editor and an error message is displayed on the Prompt/Error line.

#### **Command Mode:**

The Enter key completes the pointing commands including the Blank (/B), Copy (/C), Delete (/D), Move (/M), and Print (/P) commands, and the DIF Save option of the Storage command (/S#S).

The Enter key completes any user response typed or edited on the Prompt/Error line for the Goto (:), List (/L), and Storage (/S) commands and performs the appropriate action.

#### **The Editor:**

The Enter key enters the characters typed or edited in the current field as permanent data, if they constitute a valid item for the current field. If the item is invalid for the current field, the user remains in the Editor and an error message is displayed on the Prompt/Error line.

#### **The Help Facility:**

The Enter key enters any topic name typed or edited on the Prompt/Error line and searches for the appropriate topic. It can also be used alone instead of entering a topic name to leave the Help Facility.

#### **Plots, Tables, or Prints:**

The Enter key displays the following page of a multiple-page table or print displayed only to the screen. The Enter key also returns the user to the previous display after a plot or the last screen of a table or print displayed only to the screen.

**Directory Scrolling:**

The Enter key selects the current filename for use with the options of the Storage command (/S).

⊗ **Break Key**

**Program:**

The Break key (⊗) cancels the program at the initial screen.

The Break key cancels the output action and returns to the program after a disk or printer error once the program has begun.

Pressing the Break key twice cancels the program after a program disk error once the program has begun.

**Entry Mode:**

The Break key restores the previous item any time before the Enter key is pressed. After the Enter key, the Break key has no effect.

**The Editor:**

The Break key restores the most recent item any time before the Enter key enters a new item.

**Command Mode:**

The Break key cancels a command any time before the last keystroke of an extended command.

The Break key interrupts the Help Facility and immediately returns the user to the previous display.

The Break key returns the user to the previous display before the last page of a multiple-page table or print displayed only to the screen.

◀ **Backspace Key**

**Entry Mode, Command Mode, the Editor or the Help Facility:**

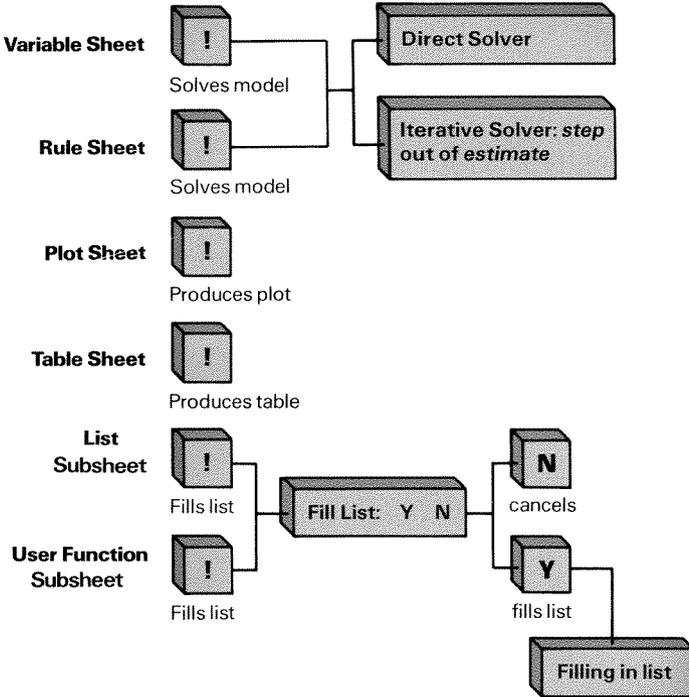
The Backspace key (◀) deletes the character to the left of the cue any time before the Enter key is pressed.

## Action

### ! Action Command

The Action command performs the action appropriate to the current sheet, including model solution, list value fill-in, plot or table display.

#### ! Perform action



#### Description:

##### Solve Action (from the Variable or Rule Sheets)

Typing the Solve action (!) solves the currently defined model by calling the Direct Solver or the Iterative Solver, as required by the model. The cursor may be positioned on any field on the Variable Sheet or Rule Sheet.

The Iterative Solver is called automatically if either:

- a **G** option is set for one or more variables in the Status field on the Variable Sheet, or

- the Direct Solver fails, the Automatic Iteration ON field on the Global Sheet is set to “Yes”, and values are supplied for one or more variables in the First Guess fields of their Variable subsheets.

During solution, a status message including the name of the Solver being used is displayed on the Prompt/Error line:

`D i r e c t S o l v e r`

or

`I t e r a t i v e S o l v e r : s t e p o u t o f e s t i m a t e`

If the Iterative Solver is called, a count of the current iteration step and the estimated number of steps needed appears at the end of the status message as successive approximations are made. If the Intermediate Redisplay ON field on the Global Sheet is set to “Yes” (the default), the values at each iteration step are displayed in the Input and Output fields on the Variable Sheet during solution.

When solution is complete, results are displayed in the appropriate output fields of the Variable Sheet and dependent subsheets. If solution fails, error codes appear in the Status fields on the Rule and Variable Sheets. (See the appendix on Error Messages for more information.)

The Solution Indicator (!) in the upper righthand corner of the Status line disappears when the solvers stop and no errors have been found.

**Fill Action (from List and User Function subsheets):**

Typing the Fill action (!) calculates intermediate values in the current list. The cursor may be on any field on a List subsheet. On a User Function subsheet, the cursor may be on any field in the domain or range list to be filled in.

The block of elements to be filled in is determined by the last element in the list containing a specified value. The block must begin at Element 1, and both values must be numeric. The difference between the first value in the list and the last is evenly distributed over the intervening elements; values are filled in for all intervening elements, whether or not they contained previous values. If unit names have been assigned for the list, the filled in values appear in the Display unit.

## Action

The user is prompted for confirmation:

Fill List: Y N

Typing **Y** calculates and enters the values; typing **N** cancels the Fill action. The status message "Filling in list" is displayed while the values are being inserted.

### **Plot Action (from the Plot Sheet):**

Typing the Plot action (!) displays a plot of the relationship between the values in one list to the values in all others specified on the Plot Sheet. The cursor may be positioned on any field on the Plot Sheet.

The values from the List subsheet specified in the X-Axis field are plotted against the values of all those specified in the Y-Axis column. Plot values are displayed in the Display units for the specified lists.

A one-page plot is displayed to the screen. It is also printed to the printer if the Screen or Printer field is set to **P** and the printer filename is specified for the appropriate printing device in the Printer Device or Filename field on the Global Sheet. When the screen option is set to **S** (the default) or the Printer Device or Filename field is blank, it is necessary to press the Enter key to return to the previous display from the plot.

### **Table Action (from the Table Sheet):**

Typing the Table action (!) displays tables of values from the List subsheets for all lists specified on the Table Sheet. The cursor may be positioned anywhere on the Table Sheet. Table values are displayed in the Display units for the specified lists.

A one or more page table is displayed to the screen. It is also printed to the printer if the Screen or Printer field is set to **P** and the printer filename is specified for the appropriate printing device in the Printer Device or Filename field on the Global Sheet. When the screen option is set to **S** (the default) or the Printer Device or Filename field is blank, it is necessary to press the Enter key to display each page after the first, and then, after the last page, return to the previous display.

#### Usage Notes:

- For more information on using the Solve action, see the descriptions of the Solvers in the Mathematics section.
- For more information on using the Plot and Table actions, see the descriptions of the Plot and Table sheets in the Sheet section.

#### Errors:

- During the Fill action (! from List or User Function subsheets), typing  $\otimes$  cancels the command. The computer produces an error signal, the Fill prompt disappears from the Prompt/Error line, and the command is cancelled. Any values already filled in remain on the screen.
- See the Overview and Mathematics sections and the appendix on Error Messages for more information about solution failures.

#### Related Commands:

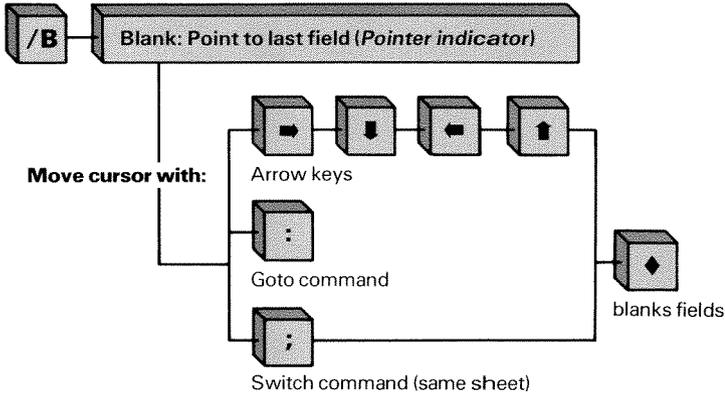
- The Solve command (/!) solves the currently defined model from any sheet.
- The Solve (/L!) and Block (/LB) options of the List command apply the solvers to variables associated with lists.

## Blank (field)

### /B Blank Command

The Blank command erases the current field item, or a column of items beginning with the current item.

#### /B Blank



#### Description:

Typing **/B** ♦ blanks the current field item. The Blank command is completed by pressing the Enter key. The cursor remains on the same field.

Typing **/B** *pointing* ♦ blanks field items in the current column from the starting cursor position to the final cursor position in the same column. The program prompts for the last field to blank and indicates the starting cursor position on the Prompt/Error line:

Blank: Point to last field (*Pointer Indicator*)

The Pointer Indicator is displayed at the end of the Prompt/Error line to indicate the current cursor position as it changes during pointing; the Position Indicator in the upper lefthand corner of the screen continues to display the starting cursor position. The Arrow keys may be used in any combination up or down with the Goto and Switch commands to reach the final cursor position on the same sheet. When the Blank command is completed by pressing the Enter key, the cursor returns to its starting position and the Blank command prompt disappears.

**Usage Notes:**

- The Goto command (:) Destination or Search options may be used to move the cursor towards the final cursor position. The Pointer Indicator is not displayed during the Goto subset of the Blank command.
- The Switch command (;) may be used to switch the cursor from one view of a sheet in one window to another view of the same sheet in the other window. Both windows must contain the same sheet prior to beginning the Blank command. This feature is useful in blanking long columns of items.
- The Blank command cannot be used on required option, required entry, or output fields; or on a subsheet displayed to the screen for a deleted variable, list, or user function.
- Items in the Field Area and Header Area of a sheet cannot be blanked with the same Blank command.
- The Blank command can be used to erase blankable field items in a block of rows containing both blankable and unblankable fields. A warning message is displayed.
- If the Display unit in a Unit field on the Variable or List Sheet is blanked, both the Display unit and the Calculation or Storage unit are cleared.
- If a value on a List subsheet, or both values in the same row for the Domain or Range fields on a User Function subsheet, are blanked, the corresponding element number is not displayed but the space is retained.
- The default value “\*” in the Character field on the Plot Sheet is the only option field that may be blanked by editing in a space, so that the display of the corresponding list may be suppressed on a plot.
- If an item is blanked from an input field on the Variable Sheet or the Input Value field on a Variable subsheet, it is retained in memory and can be returned by typing an “I”, “O”, or “G” option in the Status field.

**Related Command:**

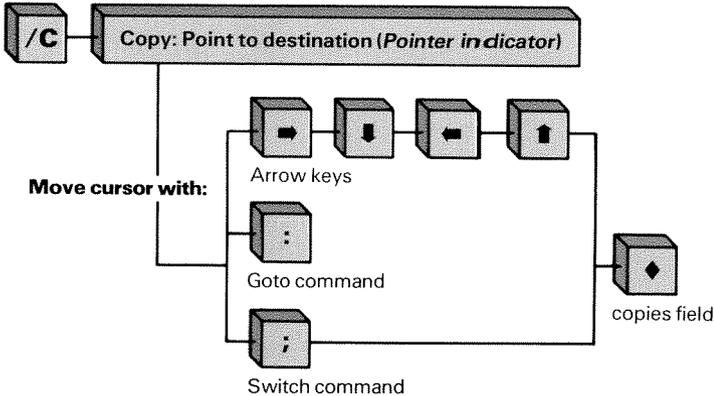
- The Delete command (/D) deletes entire rows.

## Copy (field)

### /C Copy Command

The Copy command copies the current field item to the specified destination.

#### /C Copy field



#### Description:

Typing */Cpointing* ♦ copies the current field item to the indicated final cursor position on any field displayed on the screen. The Copy command copies the complete contents of the field, not only the portion of the item displayed in the field. The Copy command is completed by pressing the Enter key.

The program prompts for the destination for the copy and indicates the starting cursor position on the Prompt/Error line:

Copy: Point to destination (Pointer Indicator)

The Pointer Indicator is displayed at the end of the Prompt/Error line to indicate the current cursor position as it changes during pointing; the Position Indicator in the upper lefthand corner of the screen continues to display the starting position. The Arrow keys may be used in any combination in any direction with the Goto and Switch commands to reach the final cursor position. When the Copy command is completed by pressing the final

Enter key, the cursor remains at the destination and the Copy command prompt disappears.

A field item can be copied from an entry, option, or output field to any entry or option field on the screen. The copied item replaces any previous item in the destination field. If a blank field is copied to another location, the destination field is erased of any previous contents. If the destination is an entry field and the copied item is invalid in the new position, an error message appears on the Prompt/Error line and the user is placed in the Editor. If the destination is an option field and the first character of the copied item is an invalid option, the valid options are displayed on the Prompt/Error line.

**Usage Notes:**

- The Goto command (: Destination or Search options may be used to move the cursor to the final destination. The Pointer Indicator is not displayed during the Goto subset of the Copy command.
- The Switch command (;) may be used to switch windows so that an item can be copied from one sheet or subsheet to another, or from one view of a sheet or subsheet to another. Both sheet views must be displayed on the screen at the beginning of the command.
- The Copy command can be used to copy the contents of a short display field into a longer one to see more information.
- The Copy command cannot copy an item to an output field.

**Related Command:**

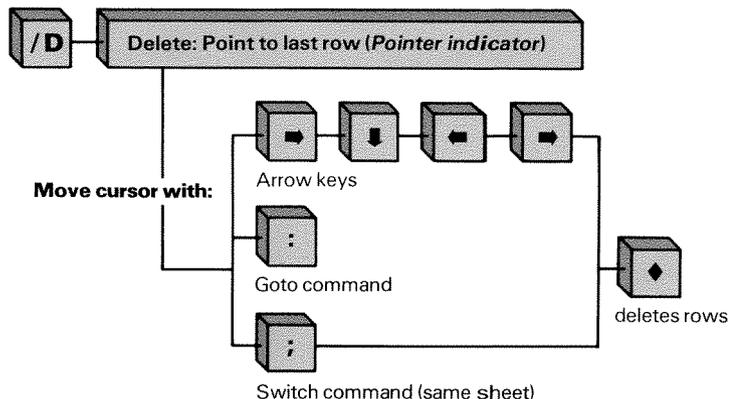
- The Move command (/M) moves the current row to the specified destination.

## Delete (row)

### /D Delete Command

The Delete command deletes the current row, or a block of rows beginning with the current row.

#### /D Delete rows



#### Description:

Typing **/D** ♦ deletes the entire current row. The cursor may be positioned on any field of the row to be deleted. The Delete command is completed by pressing the Enter key. The cursor remains in the same position, and the rows below the deleted row are moved up.

Typing **/D** *pointing* ♦ deletes rows from the current cursor position to the final cursor position on the same sheet. The cursor may be positioned on any field in either row. The program prompts for the last field to be deleted and indicates the starting cursor position on the Prompt/Error line:

Delete: Point to last row (*Pointer Indicator*)

The Pointer Indicator is displayed at the end of the Prompt/Error line to indicate the current cursor position at it changes during pointing; the Position Indicator in the upper lefthand corner of the screen continues to display the starting cursor position.

The Arrow keys may be used in any combination in any direction with the Goto and Switch commands to reach the final cursor position on the same sheet. When the Delete command is completed by pressing the Enter key, the cursor returns to the starting cursor position, the rows below the deleted rows are moved up, and the Delete command prompt disappears. The cursor marks the item below the last deleted item in the same column.

**Usage Notes:**

- The Goto command (;) Destination or Search options may be used to move the cursor towards the final cursor position. The Pointer Indicator is not displayed during the Goto subset of the Delete command.
- The Switch command (;) may be used to switch the cursor from one view of a sheet in one window to another view of the same sheet in the other window. Both windows must contain the same sheet prior to the beginning of the Delete command. This feature is useful in deleting many lines.
- The Delete command can only delete rows in the Field Area of a sheet.
- If a row in the Field Area on a List subsheet or a User Function subsheet is deleted, any subsequent rows in the list are renumbered and moved up.

**Related Command:**

- The Blank command (/B) blanks field items.

## Dive

### > Dive Command

The Dive command displays a dependent subsheet for the current sheet or subsheet in the current window.

#### > Dive



displays subsheet

#### **Description:**

Typing > from certain sheets or subsheets replaces the display in the current window with the appropriate dependent subsheet. The dependent subsheet for a sheet or subsheet contains more detailed information about the variable, list, or function on the current line. When the subsheet appears, the cursor always moves to the initial cursor position for the subsheet.

#### **From the Variable Sheet:**

Typing > with the cursor on any field in a row on the Variable Sheet displays the Variable subsheet for the variable described in that row.

#### **From a Variable subsheet:**

Typing > with the cursor on any field on a Variable Subsheet with an item specified in the Associated List field displays the List subsheet for the associated list.

#### **From the List Sheet:**

Typing > with the cursor on any field in a row on the List Sheet displays the List subsheet for the list described in that row.

#### **From the User Function Sheet:**

Typing > with the cursor on any field in a row on the User Function Sheet displays the User Function subsheet for the user function described in that row.

**From a User Function subsheet:**

Typing  $\rangle$  from a User Function subsheet displays the List subsheet for the domain or range list. To display the associated domain list, the cursor must be positioned on the Domain List field or on any value in the Domain column. To display the associated range list, the cursor must be positioned on the Range List field or on any value in the Range column.

**From the Plot Sheet:**

Typing  $\rangle$  with the cursor on the X-Axis field on the Plot Sheet displays the List subsheet for the X-Axis list.

Typing  $\rangle$  with the cursor on either Y-Axis field in a row in the Field Area of the Plot Sheet displays the List subsheet for the list described in that row.

**From the Table Sheet:**

Typing  $\rangle$  with the cursor on any field in a row in the Field Area of the Table Sheet displays the List subsheet for the list described in that row.

**Usage Notes:**

- The Dive command can be used twice in succession to display a List subsheet from a Variable Sheet or a User Function Sheet through the intermediate subsheet.
- There are no subsheets associated with the Rule, Unit, or Global sheets.
- The Return command ( $\langle$ ) can be used to return to the previous sheet or subsheet, following the same path in reverse.
- It is possible to use the Select ( $=$ ) or Window ( $/W$ ) commands without using the Return command ( $\langle$ ) first.

**Related Commands:**

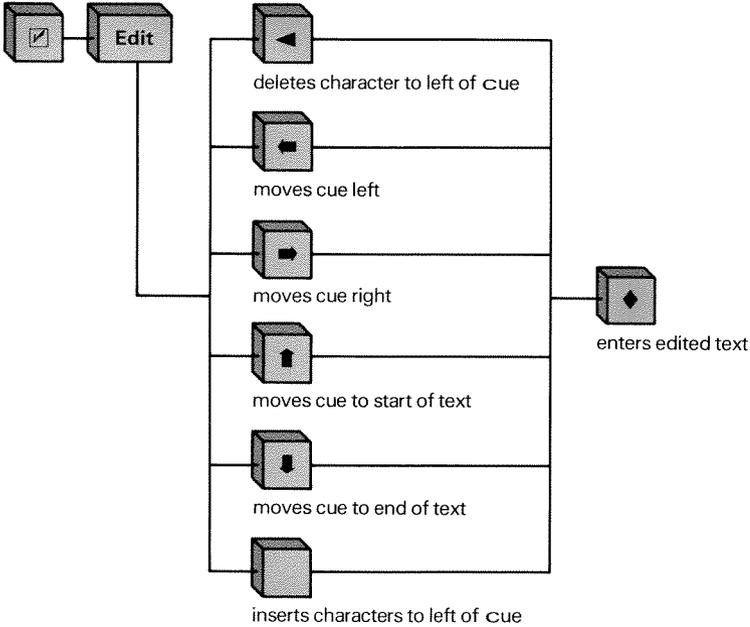
- The Return command ( $\langle$ ) returns the display of the previous sheet or subsheet to the current window after the Dive command ( $\rangle$ ).
- The Select command ( $=$ ) displays the specified sheet in the current window.

## Edit

### ☑ Edit Command

The Edit command edits a user response on the Prompt/Error line or an unentered item in the current entry field.

### ☑ Edit



**Description:**

Typing  calls the Editor in the current field or on the Prompt/Error line. It can be called from any of the commands that require user response on the Prompt/Error line before the Enter key is pressed to complete the entry. It can also be called from Entry mode on the current entry field before the Enter key is pressed to enter the item. It can be used, as well, to edit an option, an entered item, or an empty field. When  is typed, the Edit Indicator [“(Edit)...”] appears on the Prompt/Error line and the cue appears to the left of the first character in the item or the user response on the Prompt/Error line. In the Editor, the Arrow keys move the cue within the entry text:

- ← Moves the cue left one character
- Moves the cue right one character
- ↑ Moves the cue to the beginning of the text
- ↓ Moves the cue to the end of the text

Characters may be typed as in Entry mode. Special keys perform particular actions in the Editor, similar to those in Entry mode:

- ◀ Deletes the character to the left of the cue
- ◆ Exits the Editor and enters the edited changes
- ⊗ Exits the Editor without changes

Pressing the Break key any time before the final Enter key when editing a user response cancels the command. Pressing the Break key when editing an item cancels the Editor session and restores the previous item, if any, to the field.

## Edit

### Usage Notes:

The Edit command  can be used to edit the following user responses:

- Addresses or search items with the Goto command (:).
- Filenames for the Storage command (/S).
- Element numbers with the Block, Put, and Get options of the List command (/L).
- Topic names in the Help Facility.

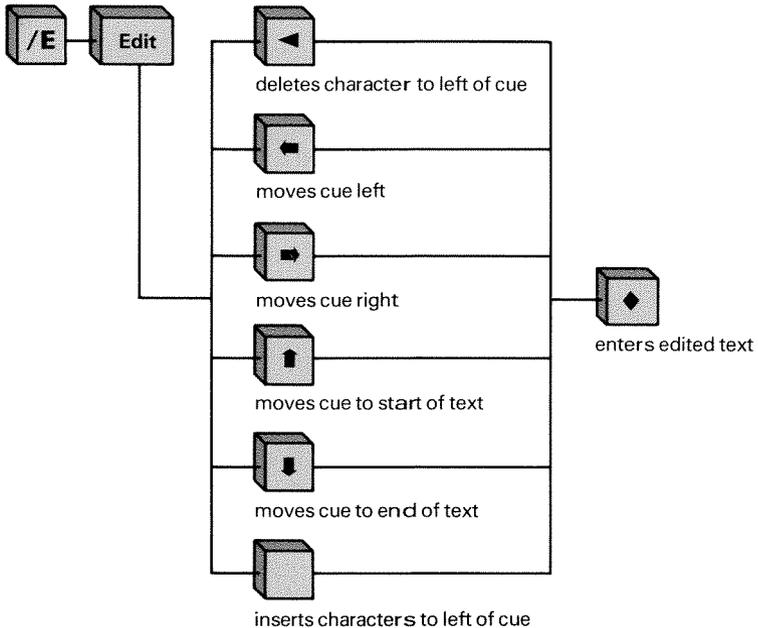
### Related Command:

- The Edit Field command (/E) edits the entered item in the current entry field.

## /E Edit Field Command

The Edit Field command edits the entered item in the current entry field.

### /E Edit field



### Description:

Typing /E calls the Editor on the current entry field item. The Editor edits the complete contents of the field, not only the portion of the item displayed in the field. When /E is typed, the Edit Indicator ["(Edit)..."] appears on the Prompt/Error line and the cue appears to the left of the field item, at the beginning of the field. In the Editor, the Arrow keys move the cue within the text:

- ⇐ Moves the cue left one character
- ⇒ Moves the cue right one character
- ↑ Moves the cue to the beginning of the text
- ↓ Moves the cue to the end of the text

## Edit Field

Characters may be typed as in Entry mode. Special keys perform particular actions in the Editor, similar to those in Entry mode:

- ◀ Deletes the character to the left of the cue
- ◆ Exits the Editor and enters the edited changes
- ⊗ Exits the Editor without changes

If the Break key is pressed any time prior to pressing the Enter key, the previous item is restored to the field.

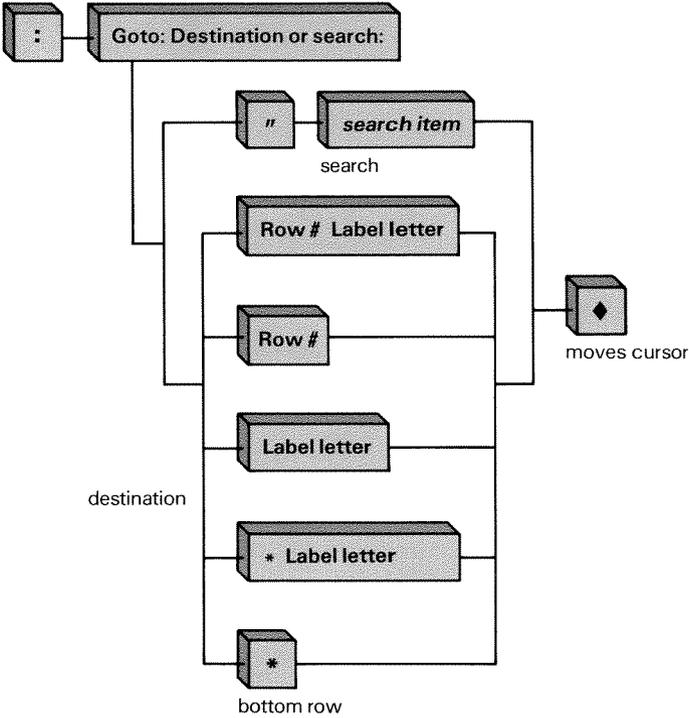
### Related Command:

- The Edit command (☒) edits a user response on the Prompt/Error line or an unentered item in the current entry field.

**: Goto Command**

The Goto command moves the cursor to the specified field on the current sheet.

**: Goto field**



**Description:**

**Destination Option (*address* ◆)**

Typing the Destination option (*:address* ◆) moves the cursor from its current position to the specified destination on the same sheet.

### Goto (field)

An address is similar to the Position Indicator displayed on the Status line except that it need not be specified in full. The following ways of specifying an address are available:

*row number label letter*

*row number*

*\*label letter*

*label letter*

*\**

The complete address for a destination field is usually made up of a row number and label letter. A field in a Header Area is indicated simply by its label letter. (The label letter for every field is given in parentheses after its field label in the Sheet section.)

If only a row number is specified, the cursor moves up or down in the current column of the Field Area to the specified row. From the Header Area, the cursor moves to the first field on the specified row.

An asterisk can be substituted for the row number of the last row on a sheet containing an item (:\* ♦). It may be used with or without a label letter.

If only a label letter is specified for a field in the Field Area, the cursor moves to the designated field on the same row.

The fields on the Global Sheet and a Variable subsheet can be located by typing their label letters or a sequential row number. On the Global Sheet, the blank lines between sections are included in the count.

The user response on the Prompt/Error line is completed by pressing the Enter key. It is possible to edit any characters typed on the Prompt/Error line before the Enter key by pressing .

#### **Search Option ("search item ♦")**

Typing the Search option (:"search item ♦) moves the cursor to the first field in the current column that displays the specified search item. Every character displayed in the field must be typed exactly as it appears to match, except that trailing blanks are ignored. If a match does not occur below the cursor, the column is searched from the top of the column back down to the current cursor position.

If the Search option is used without specifying a search item (:” ◆), the most recently specified item is reused.

The search item on the Prompt/Error line is completed by pressing the Enter key. It is possible to edit any characters typed on the Prompt/Error line before the Enter key by pressing .

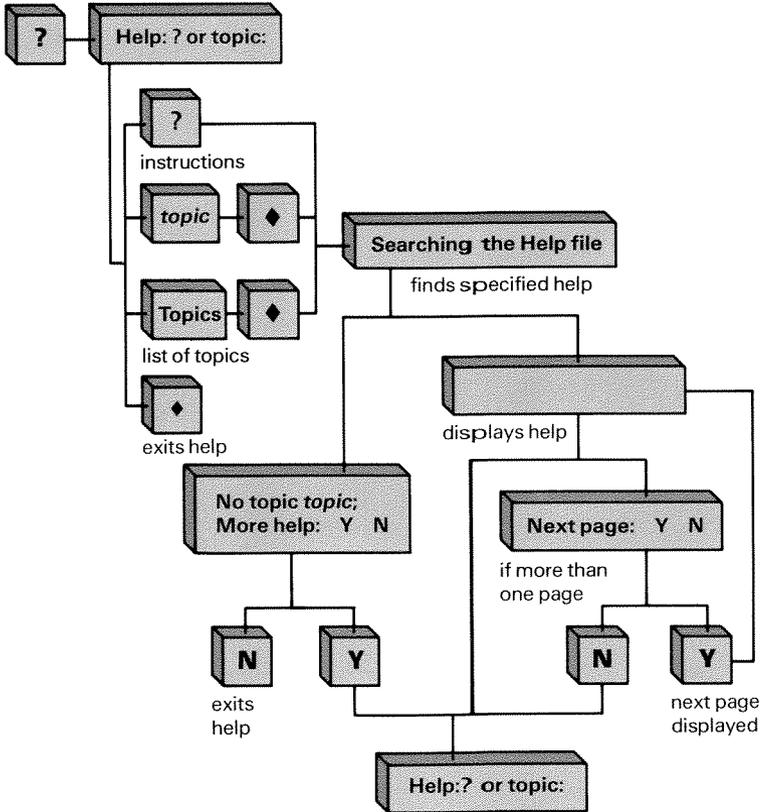
**Usage Notes:**

- The Goto command can be used to point in the Blank (/B), Copy (/C), Delete (/D), Move (/M), and Print (/P) commands, and the DIF Save option of the Storage command (/S#S).

### ? Help Command

The Help command calls the Help Facility.

#### ? Help



#### Description:

Typing ? calls the Help Facility and displays the Help topic prompt on the Prompt/Error line of the current sheet display:

Help: ? or topic:

When a topic name is typed and entered with the Enter key in response to the Help topic prompt, a Status message "Searching the Help file" is displayed on the Prompt/Error line while the program searches for the first piece of related text. If a listing is

found for a topic, the Information Area is replaced with the Help Facility display: a line similar to a Sheet Title line displays the word “HELP” at the top of the window, and the first page for the requested topic displays in the window.

If the current topic displays in a single page, the Help topic prompt redisplay on the Prompt/Error line when the search is complete. If the current topic is more than a single page long, the following prompt appears on the Prompt/Error line:

**Help: Next Page: Y N**

Typing **Y** continues to the subsequent page; typing **N** returns directly to the Help topic prompt for another topic name.

If the specified topic does not exist, the following prompt appears on the Prompt/Error line:

**No topic *topic*; More Help: Y N**

Typing **Y** returns to the Help topic prompt for another topic name; typing **N** exits the Help Facility.

### **Instructions Option (?)**

Typing the Instruction option (?) any time the Help topic prompt appears displays instructions on using the Help Facility, and any other topics related to the character “?”. The Enter key is not used after the ? as it must be after any other topic name.

### **Topic Option (*topic* ♦)**

Typing a Topic option (*topic* ♦) any time the Help topic prompt appears displays all related text for the specified topic name. Typing **TOPICS ♦** displays a list of Help Facility topics.

### **Exit Option (♦)**

Typing ♦ any time the Help topic prompt appears exits the Help Facility and returns to the previous display and cursor position.

Pressing the Break key exits the Help Facility at any time.

## Help

### Usage Notes:

- A topic name can be edited on the Prompt/Error line any time before the Enter key is pressed with the Edit command (⌘).
- Since Help Facility searches require time to perform and may take more than one page to display or may result in no topic, the next prompt should appear on the Prompt/Error line before another topic name is entered.
- If something is typed after the Help topic prompt and corrected to ? using the Backspace key, the Enter key must be pressed to enter the ? topic name.

## **/I Insert Command**

The Insert command inserts a blank row at the current row.

### **/I Insert row**



inserts row at current row

### **Description:**

Typing **/I** inserts a new row at the current row. The cursor may be positioned on any field in the row below the one to be inserted. The cursor stays on the same field and all the rows below the new row move down.

### **Usage Notes:**

- Insert command may only be used in the Field Area of a sheet.

## List

### **/L List Command**

The List command solves a model for lists of values associated with variables, or stores and retrieves list values.

#### **Description:**

The List command manipulates list values associated with variables. List command options are dependent on list information related to the current model and stored in the Variable, Rule, and List sheets, and their related subsheets. (For further discussion, see the Mathematics section.) The cursor may be positioned on any field on any sheet when using the List command.

#### **Solve Option (/L!)**

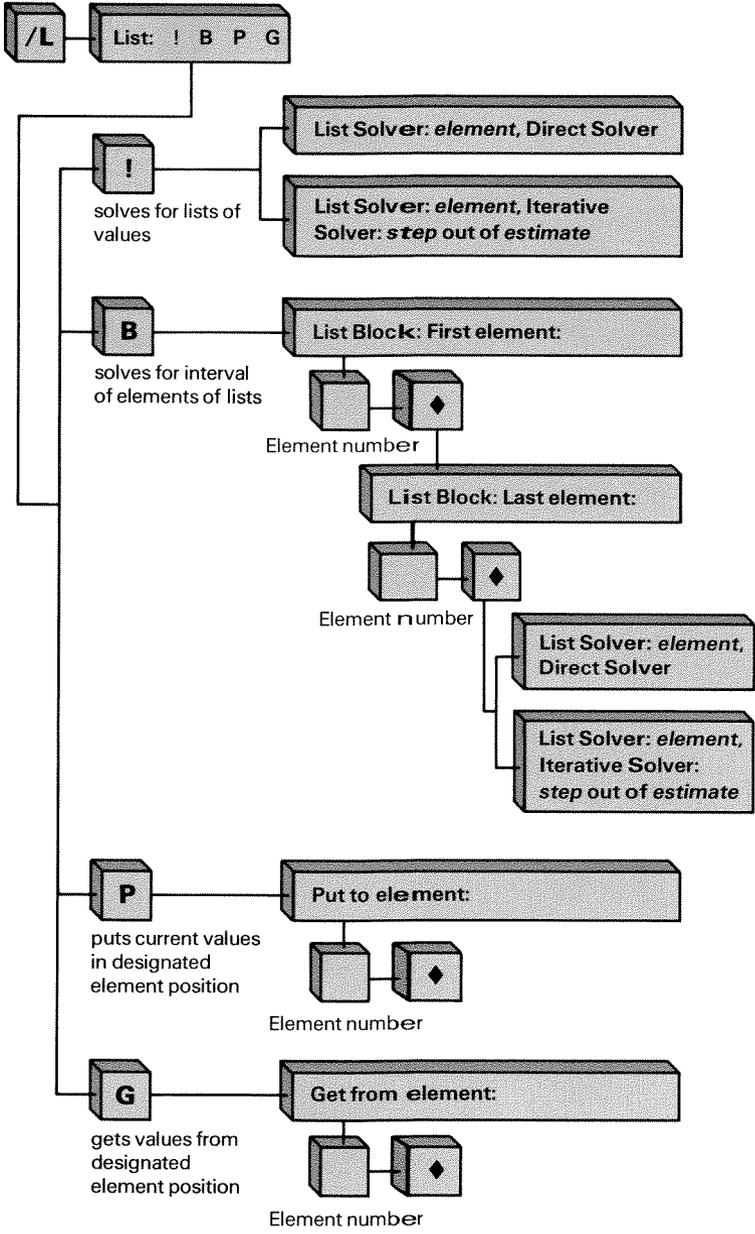
Typing **/L!** applies either of the solvers to a model with at least one associated list. All variables to be associated with lists must display an “L” option in the Status field, and at least one initial input value must be assigned on the Variable Sheet before the List Solver can be called successfully.

The model is solved for all values in every list associated with a variable designated as an input or guess variable. Input lists are designated by assigning any value in the Input field for the associated variable. A list of guess values is designated by setting the “G” option in the Status field for any associated variable. The list of values associated with a guess variable are used as the first guess values for each element in the Iteration solution. The solution value for each element is assigned to the appropriate element number in the list as it is determined. If iteration is required and there are no values in a guess list, the First Guess value on the Variable subsheet for the associated variable is used for each element in the solution.

Output values are assigned during solution to any associated list that is not designated as an input list, and all previous values in those lists are replaced or blanked. The List Solver stops when the longest input list is exhausted; after solution, all output lists have the same number of elements as the longest input list, although some values may be blank. A blank value is produced in an output list for any value that cannot be evaluated in an input list.

If units are used, the Calculation unit of a variable should be the same as, or have a defined conversion to, the Storage unit of its associated list, or the List Solver produces unreliable results.

/L Manipulate lists



## List

A Status message appears on the Prompt/Error line during solution indicating the current element number and the solver being used:

```
List Solver: element, Direct Solver
```

or

```
List Solver: element, Iterative Solver:  
                  step out of estimate
```

If the Intermediate Redisplay ON field on the Global Sheet is set to “Yes”, the input and output values are displayed on the Variable Sheet as they are processed.

### Block Option (/LB)

Typing /LB*element* ♦ *element* ♦ applies either of the solvers to an interval of list values in a model with at least one associated list. The Block option is similar to the Solve option except that it solves for a subset of all lists.

The program prompts for the beginning and ending element numbers:

```
Block: First element:  
Block: Last element:
```

The responses are typed on the Prompt/Error line and completed by pressing the Enter key.

The model is solved for each value within the specified block in every list associated with a variable designated as an input or guess variable, beginning with the first specified element number and ending with the last specified element number, inclusively, for all associated lists. If no first element is specified, the block begins with Element 1. If no last element is specified, the block ends with the last element in the longest input list.

### Put Option (/LP)

Typing /LP*element* ♦ assigns the current input and output values from the Variable Sheet for any list variables to the specified element number for all associated lists. The program prompts for the element number into which the values should be inserted:

```
Put to element:
```

The response is typed on the Prompt/Error line and completed by pressing the Enter key.

If the units for a variable and its associated list differ, the conversion is made from the Calculation unit for the variable into the Storage unit of the list before it is displayed on the List subsheet in Display units.

**Get Option**

Typing `/LGelement` ♦ assigns the values from the specified element number from the associated lists to any list variables on the Variable Sheet. It gets the values from the specified element number for all associated lists and replaces any current input and output values on the Variable Sheet. The program prompts for the element number from which to retrieve list values:

```
Get from element:
```

The response is typed on the Prompt/Error line and completed by pressing the Enter key.

If a variable associated with a list has not been designated an input or output list, the value is assigned to the Output column. If the units for a variable and its associated list differ, the conversion is made from the Storage unit for the list into the Calculation unit of the variable before it is displayed on the Variable Sheet in Display units.

**Usage Notes:**

- The Edit command () can be used to edit the user response to the Block, Put, and Get options any time before the Enter key is pressed.
- An expression that evaluates to an integer can be entered instead of an integer in response to the prompts for element numbers by the Block, Put, and Get options if it can be evaluated immediately.

**Related Commands:**

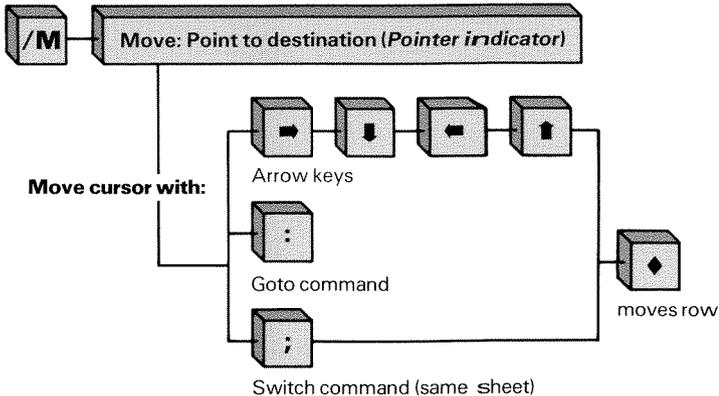
- The Action command (!) performs the action appropriate to the current sheet, including model solution, list value fill-in, and plot and table display.
- The Solve command (/!) solves the currently defined model from any sheet.

## Move (row)

### /M Move Command

The Move command moves the current row to the specified destination.

#### /M Move row



#### Description:

Typing **/M**pointing ♦ moves the current row to the indicated final cursor position on the same sheet. The cursor may be positioned on any field in the row to be moved. The Move command copies the complete contents of the fields on a row, not only the portions of items displayed in the fields. The Move command is completed by pressing the Enter key.

The program prompts for the destination for the move and indicates the starting cursor position on the **P**rompt/**E**rror line:

**Move:** Point to destination (*Pointer Indicator*)

The Pointer Indicator is displayed at the end of the Prompt/Error line to indicate the current cursor position as it changes during pointing; the Position Indicator in the upper left-hand corner of the screen continues to display the starting cursor position. The Arrow keys may be used in any combination in any direction with the Goto and Switch commands to reach the final cursor position on the same sheet. When the Move command is completed by pressing the Enter key, the cursor remains at the destination, other rows are moved down to make space at the destination, or up to fill the space vacated by the move. The Move command prompt disappears.

**Usage Notes:**

- The Goto command (:) Destination or Search options may be used to move the cursor to the final destination. The Pointer Indicator is not displayed during the Goto subset of the Move command.
- The Switch command (;) may be used to switch windows so that a row can be easily moved from one view of a sheet to another. Both windows must contain the same sheet prior to beginning the Move command.
- A row can only be moved in the Field Area of a sheet.
- If a blank row is moved to another location, a blank row is inserted at the destination.

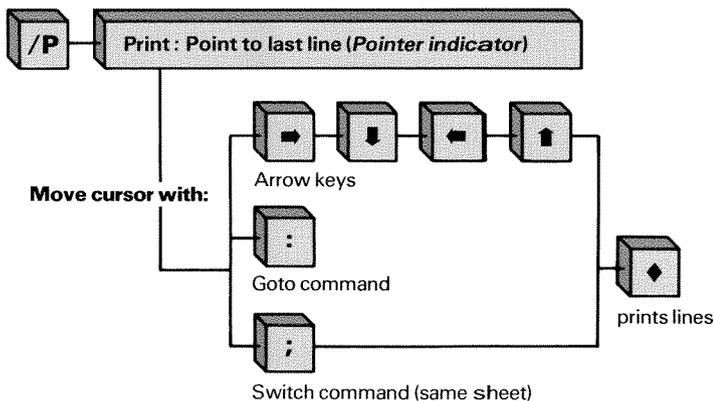
**Related Command:**

- The Copy command (/C) copies the current item to the specified destination.

### /P Print Command

The Print command prints the specified rows from the current sheet onto an output device or into a file.

#### /P Print lines



#### Description:

Typing */P*pointing ♦ prints the block of rows on the current sheet from the starting cursor position to the final cursor position. The cursor may be positioned on any field on the first row to be printed.

The program prompts for the last line to print and indicates the starting position on the Prompt/Error line:

Print: Point to last line (*Pointer Indicator*)

The Pointer Indicator is displayed at the end of the Prompt/Error line to indicate the current cursor position as it changes during pointing; the Position Indicator in the upper lefthand corner of the screen continues to display the starting cursor position. The Arrow keys may be used in combination in any direction with the Goto and Switch commands to reach the final cursor position on the same sheet. When the Print command is completed by pressing the Enter key, the current screen is replaced with a display of the lines being printed.

The specified lines are printed on the printer or to the filename specified in the Printer Device or Filename field on the Global Sheet. (See the Hardware Reference section for valid filenames and printer device names for your computer.) The output is formatted according to the options specified in the other Printer fields on the Global Sheet and displayed simultaneously on the screen. If the Printer Device or Filename field is blank, the lines are displayed only to the screen and it is necessary to press the Enter key to display each page after the first, and then, after the last page, return to the previous display.

Pressing the Break key interrupts the printing action and returns immediately to the previous display.

**Usage Notes:**

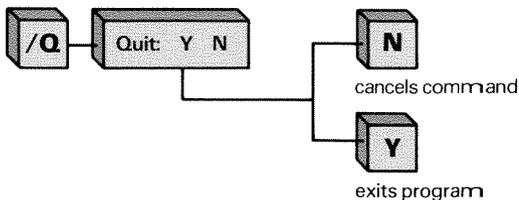
- The Goto command (: Destination or Search options may be used to move the cursor towards the last row to print. The Pointer Indicator is not displayed during the Goto subset of the Print command.
- The Switch command (;) may be used to switch the cursor from one view of a sheet in one window to another view of the same sheet in the other window. Both windows must contain the same sheet prior to beginning the Print command. This feature is useful in printing long sheets.
- The block of rows to be printed can be from both the Header Area and the Field Area.

## Quit

### /Q Quit Command

The Quit command exits the TK!Solver program.

#### /Q Quit program



#### Description:

Typing /QY exits the TK!Solver program.

Typing /QN or /Q⊗ cancels the Quit command. The computer produces an error signal, the Quit prompt disappears from the Prompt/Error line, and the user remains in the program.

#### Usage Notes:

- The program can only be exited from Command mode.
- All current data and user settings are lost unless saved with the Storage command (/S) before quitting.
- Using the Reset command (/RAY) is equivalent to exiting the program (/QY) and restarting it.

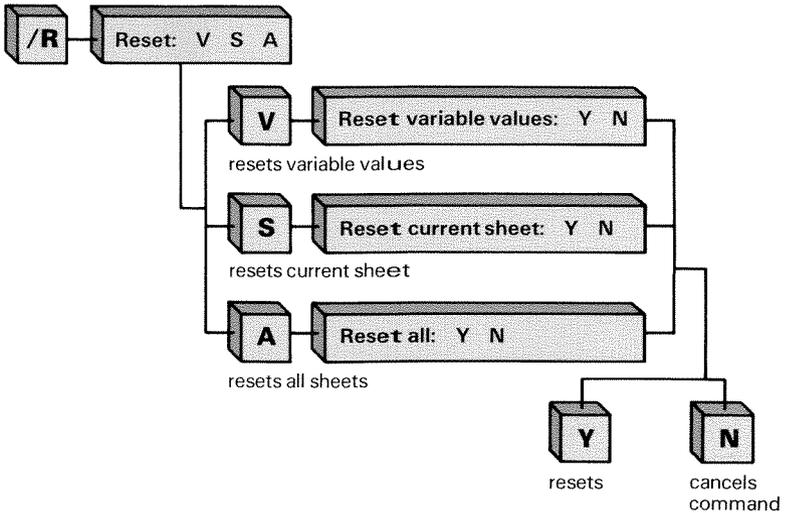
#### Related Commands:

- The Storage command (/S) saves and reloads files containing models or portions of models.
- The Reset command (/R) clears data from sheets or portions of sheets.

## /R Reset Command

The Reset command clears variable values, the current sheet, or all data from the program.

### /R Reset program, sheet, or values



### Description:

#### Variable Option (/RV)

Typing the Variable option (/RV) clears all values from the Input and Output fields and any G settings in the Status field on the Variable Sheet and all of its subsheets. The cursor may be positioned on any sheet. The program prompts for confirmation:

```
Reset variable values: Y N
```

Typing Y clears variable values; typing N cancels the Reset command. A Status message is displayed while the program clears data: "Resetting variable values".

#### Sheet Option (/RS)

Typing the Sheet option (/RS) clears all field items and resets any program defaults on the current sheet; the cursor must be positioned on the sheet to be reset. The cursor is moved to the initial cursor position for that sheet. The program prompts for confirmation:

## Reset

Reset current sheet: Y N

Typing **Y** resets the current sheet; typing **N** cancels the Reset command. A Status message is displayed while the program clears data: “Resetting current sheet”.

### All Option (/RA)

Typing the All option (/RA) clears all field items on all sheets and resets program default values. It is equivalent to re-initializing the program. The cursor may be positioned on any sheet. The program prompts for confirmation:

Reset all: Y N

Typing **Y** resets the program; typing **N** cancels the Reset command. A Status message is displayed while the program clears data: “Resetting everything”.

### Usage Notes:

- The Reset command can be cancelled by pressing the Break key any time before the final **Y** confirmation. Once **Y** has been typed the resetting action cannot be interrupted.

### Related Commands:

- The Storage command (/S) saves and reloads files containing models or portions of models.
- The Quit command (/Q) exits the TK!Solver program and clears all data.

**< Return Command**

The Return command returns from the display of a subsheet to the previous sheet or subsheet.

**< Return**

returns from subsheet

**Description:**

Typing < from a subsheet returns the display of the previous sheet, including the previous cursor position, or subsheet in the same window.

It is possible to return from a List subsheet through its corresponding Variable subsheet to the Variable Sheet, or from a domain or range list subsheet to the User Function Sheet, by using the Return command twice in succession.

**Usage Notes:**

- The Select command (=) can be used, instead of the Return command, to return from a subsheet directly to another specified sheet.

**Related Commands:**

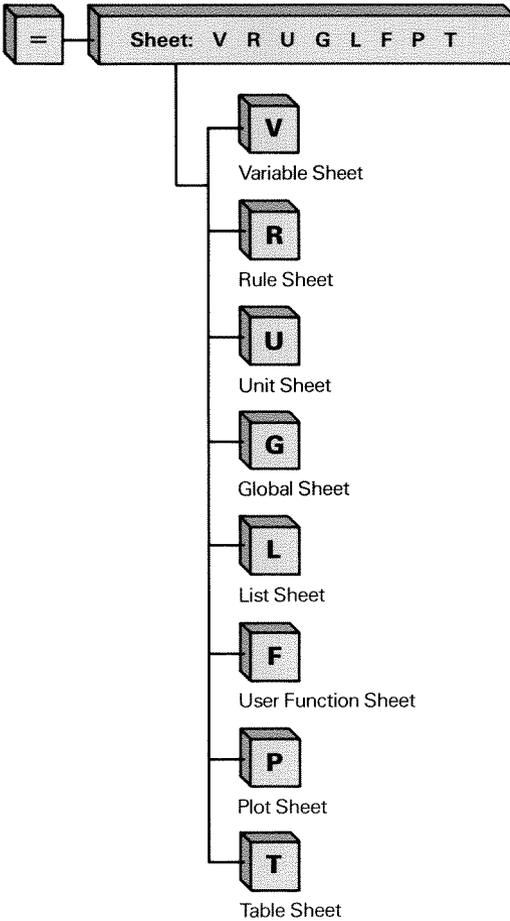
- The Dive command (>) displays a dependent subsheet for the current sheet or subsheet in the same window.
- The Select command (=) displays the specified sheet in the current window.

## Select (sheet)

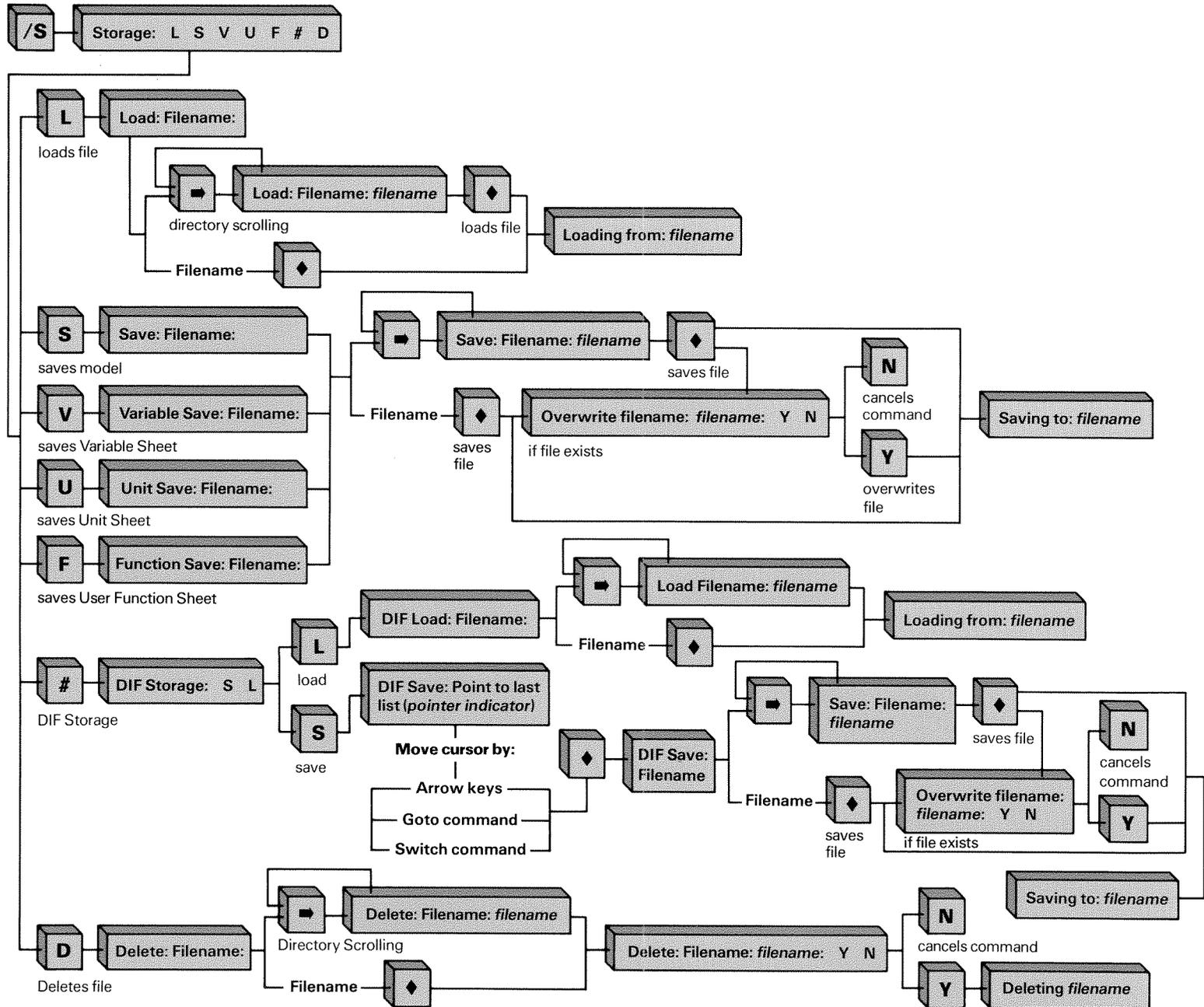
### = Select Command

The Select command displays the specified sheet in the current window.

### = Select sheet



**/S Storage**





**Description:**

Typing = *sheet letter* displays the specified sheet in the current window, causing the previous sheet to disappear:

<b>V</b>	Selects the Variable Sheet
<b>R</b>	Selects the Rule Sheet
<b>U</b>	Selects the Unit Sheet
<b>G</b>	Selects the Global Sheet
<b>L</b>	Selects the List Sheet
<b>F</b>	Selects the User Function Sheet
<b>P</b>	Selects the Plot Sheet
<b>T</b>	Selects the Table Sheet

If it is the first time the specified sheet has been selected during the current session, the cursor moves to the initial cursor position for the selected sheet. If the specified sheet is being re-accessed, the cursor moves to the most recent display and cursor position for that sheet.

**Usage Notes:**

- The Select command may be used to access any sheet from a subsheet without using the Return command (<).

**Errors**

- Typing = followed by ⊗ or any character which is not a sheet designator cancels the Select command. The computer produces an error signal, the Select prompt disappears from the Prompt/Error line, and the command is cancelled.

**Related Commands:**

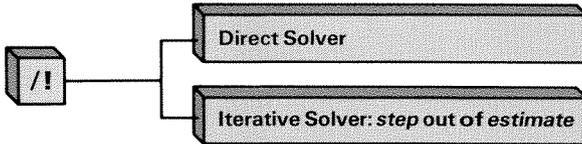
- The Two-window option of the Window command (/W*sheet letter*) splits the screen into two windows from a one-window display, with the current sheet in the top window and the specified sheet in the bottom window. The One-window option (/W1) displays a single window in the Information Area, retaining only the current sheet from a two-window display.
- The Switch command (;) moves the cursor from the current window to the other window.
- The Dive command (>) displays a dependent subsheet for the current sheet or subsheet in the same window.
- The Return command (<) returns the display of the previous sheet or subsheet to the same window after a Dive command (>).

## Solve

### /! Solve Command

The Solve command solves the currently defined model from any sheet.

#### /! Solve model



#### Description:

Typing `!/` with the cursor positioned in any field on any sheet solves the currently defined model by invoking the Direct Solver or the Iterative Solver. A Status message appears on the Prompt/Error line indicating the solver being used:

`Direct Solver`

or

`Iterative Solver: step out of estimate`

For further discussion of the solvers, see the Solve action of the Action command (!) and the Mathematics section.

#### Related Commands:

- The Solve action of the Action command (!) solves the currently defined model from the Variable or Rule Sheet only.
- The Solve option (/L!) and Block option (/LB) of the List command solve a model for lists of values associated with variables.

**/S Storage Command**

The Storage command saves, loads, or deletes files containing models or portions of models.

**Description:**

Typing /S calls the storage facility and offers a choice of options for loading, saving, or deleting files:

**Storage:** L S V U F # D

Each option of the Storage command prompts for a filename. (See the Hardware Reference section for information on valid file specifications for your computer.)

A new filename may be specified, with a disk drive specifier if one is necessary, and completed with the Enter key, or Directory Scrolling may be used to select an existing filename from the filenames already on an inserted diskette.

Directory Scrolling selects appropriate files from the default or specified diskette according to the option of the Storage command:

- the Delete file option of the Storage command (/SD) selects all files on the diskette;
- the DIF Save (/S#S) and DIF Load (/S#L) options select only DIF files;
- all other options of the Storage command select only TK!Solver storage files.

If a filename is omitted or partially specified for any option of the Storage command, Directory Scrolling displays each matching filename so that one may be selected and used. Directory scrolling is initiated by pressing the Right Arrow key (➡) alone without any file specification, or after a partial file specification.

A partial file specification is determined by the computer system being used and might include a disk specifier by itself, a file specification using system wildcards, a filename without an extension (for the Delete file option), or some combination with a disk specifier. (See the Hardware Reference section for more information on valid file and disk specifications and the use of wildcards and file extensions, if any, for your system.)

## Storage

Directory Scrolling displays the appropriate prompt and the first selected filename on the Prompt/Error line:

Load filename: *filename*

Save filename: *filename*

Delete filename: *filename*

Filenames can be scrolled through in only one direction by pressing the Right Arrow key. The current filename may be selected by pressing the Enter key (◆). If all the files are scrolled through before the Enter key is pressed, the Storage command is cancelled.

Typing **/S#** calls the DIF storage facility and offers two options for saving and loading DIF format files:

DIF Storage: S L

DIF files contain lists of data in a format interchangeable with other programs. All labels and string values in a DIF file must conform to the TK!Solver conventions for valid name items and symbolic values before the file can be loaded into the TK!Solver program correctly. (For more information about DIF files, see the appendix on TK!Solver Files.)

### **Saving:**

If a filename already exists, the program prompts for overwriting:

Save: Overwrite filename: *filename*: Y N

Typing **Y** copies the new data to the specified filename and erases any previous data; typing **N** cancels the Storage command.

### **Save Model Option (/SS)**

Typing **/SS** displays a prompt for a filename:

Save: Filename:

Typing **/SSfilename◆** saves current data from all sheets to the specified file. Directory Scrolling can also be used. After the final Enter key is pressed, a Status message appears on the Prompt/Error line: "Saving to *filename*".

**Variable Save Option (/SV)**

Typing /SV displays a prompt for a filename:

Variable Save: Filename :

Typing /SV*filename* ♦ saves the current data from the Variable Sheet and Variable subsheets to the specified file. Directory Scrolling can also be used. After the final Enter key is pressed, a Status message appears on the Prompt/Error line: “Saving to *filename*”.

**Unit Save Option (/SU)**

Typing /SU displays a prompt for a filename:

Unit Save: Filename :

Typing /SU*filename* ♦ saves current data from the Unit Sheet to the specified file. Directory Scrolling can also be used. After the final Enter key is pressed, a Status message appears on the Prompt/Error line: “Saving to *filename*”.

**User Function Save Option (/SF)**

Typing /SF displays a prompt for a filename:

Function Save: Filename :

Typing /SF*filename* ♦ saves current data from the User Function Sheet, List Sheet, and related subsheets to the specified file. Directory Scrolling can also be used. After the final Enter key is pressed, a Status message appears on the Prompt/Error line: “Saving to *filename*”.

**DIF Save Option (/S#S)**

The DIF Save option can only be used from the List Sheet. The cursor must be positioned on the first list to be saved. Typing /S#S displays a prompt for the last list to save:

DIF Save: Point to last list (*Pointer Indicator*)

The Pointer Indicator is displayed at the end of the Prompt/Error line to indicate the current cursor position as it changes during pointing; the Position Indicator in the upper left-hand corner of the screen continues to display the starting cursor position. The Arrow keys may be used in any combination in any direction with the Goto and Switch commands to reach the final cursor position on the same sheet.

## Storage

When the last list is indicated by pressing the Enter key, the prompt on the Prompt/Error line is replaced by a prompt for a filename:

D I F   S a v e :   F i l e n a m e :

Typing `/S#Spointing ♦ filename ♦` saves the specified lists on the List Sheet in the specified DIF-format file. Directory Scrolling can also be used. After the final Enter key is pressed, a Status message appears on the Prompt/Error line: “Saving to *filename*”.

DIF files save information from the List Sheet and List sub-sheets, including Storage and Display units.

### Loading:

#### Load Option (/SL)

Typing `/SL` displays a prompt for a filename:

L o a d :   F i l e n a m e :

Typing `/SLfilename ♦` loads previously saved data from the specified file into the program. Directory Scrolling can also be used. After the final Enter key is pressed, a Status message appears on the Prompt/Error line: “Loading from *filename*”.

When a file is loaded over a model already on the screen, the information from the file is added to what is already displayed. Information from the specified file from the Rule, Unit, Table, and Plot sheets is inserted at the bottom of the appropriate sheet, even if the same information then appears twice. Information from the Variable, List, and Function sheets is added to the bottom of the appropriate sheet only if there are no duplicate names. If names are duplicated, the values from the loaded file are assigned to the variable or user function names on the screen, overwriting any displayed values. If list names are duplicated, a new list is created without a name. Header Areas of sheets and sub-sheets are always reset by the loaded information.

**DIF Load Option (/S#L)**

Typing **/S#L** displays a prompt for a filename:

**D I F   L o a d :   F i l e n a m e :**

Typing **/S#Lfilename** ♦ from the List Sheet loads the specified DIF-format file. Directory Scrolling can also be used. After the final Enter key is pressed, a Status message appears on the Prompt/Error line: “Loading from *filename*”.

When a DIF file is loaded over a model already on the screen, the information from the file is added to what is already displayed on the List Sheet and subsheets. If names are duplicated, the list information from the DIF file is loaded without names below the values displayed on the screen.

**Deleting:****Delete File Option (/SD)**

Typing **/SD** displays a prompt for a filename:

**D e l e t e :   F i l e n a m e :**

Typing **/SDfilename** ♦ deletes the specified file from the specified or current diskette. Directory Scrolling can also be used. If you specify a filename without an extension, the program finds the first instance of the filename with any extension and prompts for confirmation of the complete filename. ♦ selects that filename. ➡ scrolls to the next instance of the filename.

After the final Enter key is pressed, the program prompts for confirmation of the delete.

**D e l e t e   f i l e n a m e :   *filename* :   Y   N**

Typing **Y** deletes the current file; typing **N** cancels the Storage command. After the **Y** confirmation is typed, a Status message appears on the Prompt/Error line: “Deleting *filename*”.

## Storage

### Usage Notes:

- The Edit command (E) can be used to edit the filename on the Prompt/Error line any time before the Enter key is pressed to complete the filename or the Right Arrow key is pressed to begin Directory Scrolling.

### Related Commands:

- The Reset command (/R) clears the program or data from sheets.
- The Quit command (/Q) exits the program. The Storage command must be used to save data since the Quit command clears the program.

**; Switch Command**

The Switch command moves the cursor between windows.

**; Switch window**

moves cursor between windows

**Description:**

Typing the Switch command (;) with two windows displayed on the screen moves the cursor from its present position to the other window. The cursor is moved to the initial cursor position or the most recent cursor position for the sheet or subsheet in that window. Typed a second time, the Switch command returns the cursor to its position on the original window.

**Usage Notes:**

- The Switch command can be used with the Copy (/C) command to move towards a final destination on any sheet displayed in the other window.
- The Switch command can be used to move towards the final row or destination for the Blank (/B), Delete (/D), Move (/M), Print (/P), and DIF Save (/S#S) commands if both windows contain views of the same sheet.

**Errors:**

- If the Switch command is typed and one window is displayed, the computer produces an error signal. The cursor does not move.

**Related Commands:**

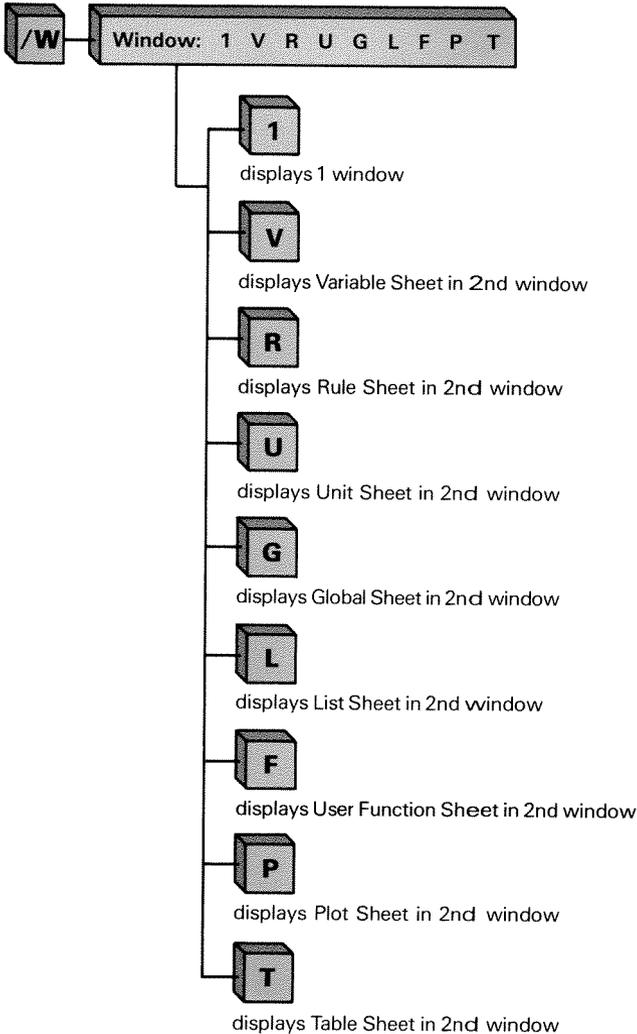
- The Two-window option of the Window command (/Wsheet letter) splits the screen into two windows when only one window is displayed on the screen.
- The Select command (=) displays the specified sheet in the current window.

## Window

### /W Window Command

The Window command displays a single window or splits the screen into two windows.

#### /W Display window



**Description:****One-window Option (/W1)**

Typing the One-window option (/W1) when two windows are displayed creates one window in the Information Area. To retain a sheet, type /W1 with the cursor on any field on that sheet. The sheet expands to fill the Information Area, and the other sheet disappears. The cursor remains on the same field.

**Two-window Option (/Wsheet letter)**

Typing the Two-window option (/Wsheet letter) when one window is displayed splits the Information Area into two windows.

To split a single window into two equal-sized windows, type /Wsheet letter:

<b>V</b>	Displays the Variable Sheet
<b>R</b>	Displays the Rule Sheet
<b>U</b>	Displays the Unit Sheet
<b>G</b>	Displays the Global Sheet
<b>F</b>	Displays the User Function Sheet
<b>L</b>	Displays the List Sheet
<b>P</b>	Displays the Plot Sheet
<b>T</b>	Displays the Table Sheet

The upper portion of the current sheet remains in the top window, the specified sheet displays in the newly created bottom window, causing the lower portion of the current sheet to disappear. The cursor moves to the initial cursor position for the specified sheet in the bottom window.

**Usage Notes:**

- When the TK!Solver program is loaded, two windows are displayed. The top window contains the Variable Sheet and the bottom one the Rule Sheet. The initial cursor position is in the Rule field on the first row of the Rule Sheet.
- Two different views of the same sheet can be displayed simultaneously in two windows. Changes made to one display of a sheet appear on the other.
- The Window command can be used from a subsheet as well as from a sheet.
- The Window command does not function during actions that pre-empt the entire screen, such as use of the HELP Facility (?), the Plot and Table actions (!), or the Print command (/P).

## Window

### Errors:

Certain errors cancel the Window command. The computer produces an error signal, the Window prompt disappears from the Prompt/Error line, and the command is cancelled. Errors occur when:

- /W1 is typed and there is one window displayed on the screen;
- /W*sheet letter* is typed and there are two windows displayed on the screen;
- /W is followed by ⊗ or any character which is not an option of the Window command.

### Related Commands:

- The Switch command (;) moves the cursor from the current window to the other window.
- The Select command (=) displays the specified sheet in the current window.
- The Dive command (>) displays a dependent subsheet for the current sheet or subsheet in the current window.
- The Return command (<) returns from the display of a subsheet to the previous sheet or subsheet in the same window after a Dive command (>).





## **Mathematics**

### **Equation Components**

This section details the components of rules or equations in the TK!Solver program. In the TK!Solver program, equations are written in standard algebraic notation. Standard computer conventions are used to represent arithmetic operators. These conventions are detailed in the Arithmetic Operators section below.

Values in the TK!Solver program can be either numeric values or symbolic values.

#### **Numeric Values**

##### **Use**

Specify quantity.

##### **Format**

Decimal Form and Scientific Notation

Decimal Form

Real numbers written in standard form.

Examples:

12.5	-6.7893
321313243125	.54783

The TK!Solver program does not accept commas in numeric input. For numbers containing a decimal point, a leading “0” entered to the left of the decimal point will be deleted by the program.

Scientific Notation

Real numbers written as a product of a number whose absolute value is between 1 and 10 and a power of ten. The TK!Solver syntax for scientific notation is the number between 1 and 10 separated from the exponent of 10 by an “E”.

Examples:

$3.4 \times 10^4 = 3.4E4$
$5.676 \times 10^{-3} = 5.676E-3$

## Equation Components

The “E” can be entered in either lowercase or uppercase, but is always displayed as uppercase. Entries may be typed with numbers less than 1 or greater than 10 before the “E”, but all output is in standard form.

### Range of Numbers

System dependent. See the Hardware Reference section for the range for your system.

### Precision

All values are rounded to fit display fields, without affecting the internal representation of the number. The maximum number of significant digits is 11–12. The full precision of any displayed number can be seen by moving the cursor over the number and reading the display on the Status line. Digits beyond the significant digits are displayed as zeros.

### Symbolic Values

#### Use

Represent a non-numeric value.

#### Format

Any alphanumeric set of characters including the special characters:

@ # \$ % \_

Symbolic values cannot contain spaces or begin with a number; they must be preceded by an apostrophe to be entered in the program. These values are case sensitive on computers that support both cases.

Examples:

'a                    'B2                    'b2                    '%0tj3  
'wk\_4@%            'Q756                    '\$h6H5

Maximum Length: 200 characters

### **Variables**

#### **Use**

A symbol used to represent any element of a set of values.

#### **Format**

Must begin with an alphabetic character or one of the special characters shown below and contain only alphanumeric characters and the special characters:

@ # \$ % \_

They cannot contain spaces. Variables in the TK!Solver program are case sensitive on computers that support both cases.

Maximum Length: 200 characters

### **Arithmetic Operators**

#### **Use**

Designate appropriate computation.

#### **Operators**

The following symbols are used to specify the standard arithmetic operators:

+ Addition  
- Subtraction  
\* Multiplication  
/ Division  
^ Exponentiation

All multiplication must be indicated by an asterisk (\*); there is no implicit multiplication in the program.

Example:

$4(x-y)(3x+y)$   
must be written as  
 $4*(x-y)*(3*x+y)$

For precedence, see section on Evaluation of Expressions below.

### Functions

There are two types of functions in the TK!Solver program, built-in functions and user functions.

### Syntax

$$f(x) = y$$

function name    argument    function value

### Function Name

#### Built-in Functions

The names of the built-in functions appear in the next part of the manual. Names of built-in functions are case insensitive.

#### User Functions

Must begin with a letter and contain only alphanumeric characters and the special characters:

@ # \$ % \_

### Argument

Any valid TK!Solver expression, including symbolic values. Some built-in functions take multiple arguments.

### Function value

Any valid TK!Solver value, including symbolic values.

### Expressions

TK!Solver expressions have the same form as algebraic expressions.

Examples:

$$2 + x \qquad 3 * \sin(3 * \pi()) / 2$$
$$4 * x^3 - 2 * \text{int}(a/b)^{-c+d}$$

Maximum Length: 200 characters

**Rules**

In the TK!Solver program all rules are equations. Equations are made up of two expressions joined by an equal sign.

Examples:

$$a/b^3-4*b = \text{sqrt}(a*b)-3E12$$

$$\text{time} = \text{period}/(1 + .25*\sin(\text{theta}/4)^2)$$

$$s = v*t + 1/2*a*t^2$$

Maximum Length: 200 characters

### Expression Evaluation

This section describes the rules of precedence and how the calculator feature of the program works.

#### Precedence

Within an expression, the order of operations performed is:

+	-	Unary plus or minus
^		Exponentiation
/	*	Division or multiplication
+	-	Binary addition or subtraction

Operations at the same level are performed left to right. Precedence can be altered by the use of parentheses. Operations inside parentheses are performed before any others.

Example:

$$-5 + 33/(5*3-4) - 2^3/1.6$$

Step-by-step solution:

$$-5 + 33/(5*3-4) - 2^3/1.6$$

$$-5 + 33/(15-4) - 2^3/1.6$$

$$-5 + 33/11 - 2^3/1.6$$

$$-5 + 33/11 - 8/1.6$$

$$-5 + 3 - 8/1.6$$

$$-5 + 3 - 5$$

$$-2 - 5$$

$$-7$$

#### Evaluation

Expressions are evaluated according to the rules of precedence. Variables are replaced with their given or computed value and numeric and symbolic values evaluate to themselves.

#### The Calculator

Any field that accepts numeric entries can be used as a calculator to evaluate expressions. Fields that permit only integer values accept only expressions that evaluate to integers. If a variable name is used as part of an expression to be evaluated, the program uses the value of the Calculation unit, not the Display unit. When values are displayed in mixed units, make cautious use of the Calculator.

## Model Solution

This section begins with the solution process for a single equation. The remainder of the section describes in detail the Direct and Iterative Solvers of the program.

### Direct Solver

#### Use

Solves models. It is activated by typing `/!` on any sheet or `!` on the Variable or Rule Sheet.

#### Prompt/Error Line Message

```
D i r e c t   S o l v e r
```

### Single-Rule Evaluation

Evaluation of a model containing a single equation is very similar to the evaluation of a model containing many equations. The expression on each side of the equal sign is evaluated separately. If each can be evaluated completely, the comparison tolerance is used to check the equality of the two expressions. If they are not equal, an appropriate error message is generated.

If the two expressions cannot be evaluated, the program attempts to algebraically place the unknown value on one side of the equation. Once this is accomplished, the remaining expression is evaluated to provide the value of the unknown. If this algebraic manipulation fails to solve the problem, an Iterative solution may still be possible.

In certain cases the value of a variable is indeterminable. In six of these cases, the TK!Solver program marks the rule either satisfied or unsatisfied without further evaluation of the rule. These cases are:

$ATAN2(0,0) = x$	Since ATAN is undefined at this point, the program returns no value and leaves the rule unsatisfied.
$1 = 1^x$	Since $x$ can be any value, the program returns no value, yet marks the rule satisfied.
$1 = x^0$	Since $x$ can be any non-zero value, the program returns no value, yet marks the rule satisfied.

## Model Solution

- $0 = 0^x$       Since 0 to any non-zero power must equal 0, the program returns no value, yet marks the rule satisfied.
- $x = 0^0$       Since  $0^0$  is undefined, the program returns no value and leaves the rule unsatisfied.
- $x = 0/0$       Since  $0/0$  is undefined, the program returns no value and leaves the rule unsatisfied.

In order for the Direct Solver to solve a single equation, it must be possible for it to isolate the variable being solved for. Three conditions must be met to make this possible:

- All variables in the equation, except the one being solved for, must have input values.
- The variable being solved for must appear only once in the equation.  
For example, the Direct Solver CANNOT solve for  $x$  in these equations:

$$x + x = y \text{ (can solve written as } 2*x = y\text{)}$$
$$x*x = y \text{ (can solve written as } x^2 = y\text{)}$$

- The variable being solved for cannot be the argument of a function without a unique inverse.  
For example, the Direct Solver CANNOT solve for  $x$  in these equations:

$$\text{SGN}(x) = y \qquad \text{INT}(x) = y \qquad \text{ABS}(x) = y$$

### Model Solution Process

The Direct Solver attempts to evaluate every equation in a model. As it satisfies the equality conditions for one equation, it uses the values it determined in that solution process to satisfy other equations.

As the Direct Solver determines unknown values it not only checks that the values are consistent with the equation involved, but with all other known values in the model.

### Comparison Tolerance

The comparison tolerance is used by the solvers of the program to determine equality. Two values,  $x$  and  $y$ , are considered equal if the following inequality is true:

$$\frac{|x - y|}{\max(|x|, |y|)} \leq \text{comparison tolerance}$$

For values near zero, determined by the typical value, the comparison tolerance formula for equality changes to:

$$\frac{|x - y|}{\text{typical value}} \leq \text{comparison tolerance}$$

As the comparison tolerance gets smaller, the solution process becomes accurate to more decimal places.

Default value: .000001 (set on Global Sheet)

**Typical Value**

The typical value is used in conjunction with the comparison tolerance. The product of the typical value and the comparison tolerance defines at what point the program will treat values as near zero. For example, if:

$$\begin{aligned} \text{Comparison tolerance} &= .000001 \\ \text{Typical value} &= .01 \end{aligned}$$

Then, any number with an absolute value less than  $(.000001) * (.01) = .0000001$ , or  $1E-8$ , is considered near zero.

When the absolute value of a quantity approaches zero as defined by the above product, the ratio associated with the comparison tolerance changes. The change replaces “ $\max(x,y)$ ” with “typical value” for the denominator of the ratio.

Since typical value alters at what decimal place zero equivalence is obtained, it can be a useful tool. Increases in the typical value will have a concurrent increase in the decimal place at which values are considered equal to zero and vice versa.

Default value: 1 (set on Global Sheet)

## Model Solution

### Iterative Solver

#### Use

Solves models that the Direct Solver is unable to solve. To activate the Iterative Solver follow these steps:

- Determine which variables cannot be solved by the Direct Solver: Either check the conditions for Direct solution or perform a Direct solution.
- Type **G** in the Status field of each variable that requires an Iterative solution.
- Enter a guess value in the Input field of each variable requiring an Iterative solution.
- Type **/!** on any sheet or **!** on the Variable or Rule Sheets.

#### Prompt/Error Line Message

*Iterative Solver: step out of estimate*

The “step” in this message relates what iteration step the solver is currently performing. The “estimate” is an estimate of the total number of steps the Iterative Solver will need to solve the problem.

This number changes frequently during solving. If it stays higher than 20, this may indicate either a poor initial guess or possibly a problem without a solution.

#### Solution Process

The Iterative Solver uses the initial guess value(s) to solve the model. The equation(s) containing the iterated variable(s) and the entire model are checked for equality by the comparison tolerance. If there are any inconsistencies, another guess is generated by the solver. The comparison checks are performed on each step of the iteration. The Iterative Solver stops when two successive guesses are evaluated as equal and all equations possible in the model are satisfied.

#### Intermediate Redisplay

Each step of iteration generates a trial solution of the model. These values can either be displayed or suppressed. The mechanism for control of display is the Intermediate Redisplay ON field on the Global Sheet. If this field is set to “Yes”, the values for each step of iteration are displayed. Set to “No”, the values

are suppressed. Suppression of the intermediate values will shorten processing time for the Iterative Solver. The default value for this field is “Yes”.

**Maximum Iteration Count**

The number of iteration steps that are performed by the Iterative Solver is set on Global Sheet in the Maximum Iteration Count field. The default value is “10”. If the model does not solve in the number of steps set, typing the Action command again provides for another full set of iterations.

**Guess Values**

A list solving can help find an appropriate guess value for an Iterative solution of a polynomial. Set up a wide range of values for the variable which requires Iterative solution. The output list of polynomial values can point to possible guess values. A root of the equation will be between any two values of the polynomial with different signs.

Example:

$$x^8 - 5x^6 + 3x^2 + 10x + 1 = y$$

The roots of this equation are not obvious. Applying the method described above, make a list of 100 elements for  $x$  ranging from 20 to  $-20$ . A List solving for  $y$  shows four changes of sign. Any one of these can be used as a guess value for  $x$  with  $y$  set to “0”.

To determine multiple roots of a polynomial, division of the polynomial by terms of the form  $(x-r)$  where  $r$  is a root of the polynomial will result in a different root being found on the next attempted solution.

**Comparison Tolerance and Typical Value**

See under Direct Solver.

**Automatic Iteration****Use**

If a model requires iteration for particular sets of input values, Automatic Iteration alleviates the need to explicitly enter a guess value for variables for each solution of the model. It is particularly useful in conjunction with the Solve option of the List Command.

## Model Solution

To use Automatic Iteration:

- Set the Automatic Iteration ON field on the Global Sheet to “Yes”, the default value.
- On the Variable subsheet of each variable that may require an Iterative solution provide an initial guess value in the First Guess field.
- Type ! or /! to solve the model.

### **Solution Process**

The Direct Solver calls the Iterative Solver if it is unable to solve the model and the appropriate Automatic Iteration values are set.

For some models it is difficult to set a guess value general enough to be useful for all input values of interest. Automatic Iteration is best suited for models that will be solved repeatedly within a moderate range of values.

**Model Development Overview**

TK!Solver models define relationships among variables within a system of equations. A model may be as simple as a single equation or so complex that it involves all the sheets and has over 100 equations.

**Simple Models**

The simplest model possible in the TK!Solver program involves only the Variable and Rule sheets and contains a single equation such as either of the following:

$$a - c = b * d$$

$$\textit{Volume} = \textit{length} * \textit{height} * \textit{depth}$$

The Rule Sheet contains the equation and the Variable Sheet, the variables.

**Model Expansion Overview**

There are several ways to expand a simple model. The options include the following enhancements:

- Entering new rules that refine or expand the model's utility
- Using TK!Solver functions
- Associating units with variables
- Associating lists of values with variables
- Creating functions to relate values
- Documenting the model with comments

### **Model Development Using Sheets**

In this section each sheet is listed with a number of ways to use the sheet when building complex models. The lists of ideas are not exhaustive.

#### **Variable Sheet**

##### **Units**

Write equations in the units most convenient for calculation, with whatever units are appropriate chosen for the display of the variable on the Variable Sheet.

##### **Lists**

Associate lists with variables to solve models for several values of selected variables.

##### **Iterative Solver**

Solve models for variables not found by the Direct Solver.

##### **Comments**

Document variables.

#### **Rule Sheet**

##### **List Functions**

Several TK!Solver built-in functions manipulate lists. They are COUNT, DOT, ELEMENT, MAX, MIN, and SUM. Rules may be created to use these functions.

##### **Editing Rules**

Rules may be edited as long as the expressions on both sides of the equal sign remain valid and the overall character length does not exceed 200.

##### **Comments**

Document rules for further reference.

#### **Unit Sheet**

##### **Variables**

Provide unit conversions from Calculation to Display units. Also permits unit conversions among variables.

##### **Lists**

Provide unit conversions between the Storage units of lists and the Calculation units of variables.

**Global Sheet****Solving**

Speed processing time for Iterative and List solutions by switching Intermediate Redisplay ON to “No”.

**Automatic Iteration**

Set models to switch to Iterative Solver when the Direct Solver is unable to fully solve a model. Automatic Iteration ON must be set to “Yes” and appropriate first guesses set on selected Variable subsheets.

**Precision**

See Comparison Tolerance and Typical Value for details affecting equality and zero determination using these features.

**Printing**

Set parameters affecting printing.

**List Sheet and List Subsheets****Lists**

Create lists to be used with the Solve option of the List command.

**Functions**

Create lists to be used with functions.

**User Function Sheet****Functions**

Create functions for use in equations.

**Plot Sheet****Iterative Solver**

If looking for simultaneous solutions to a set of equations, solve the model for a list of wide-ranging values and then plot the function values to determine the appropriate neighborhood for guess values.

**Both Solvers**

After list solving, plot to determine behavior of data.

**Table Sheet****Lists**

Display lists either for purposes of analysis or to see a table of results.

### Model Development Using TK!Solver Features

In this section, three features of the TK!Solver program (units, lists, and functions) are described in terms of how they can be used in developing models.

#### Units

Although units are not necessary in models, when they are used, they affect every sheet, except the Global Sheet. This section contains three parts:

- **On the Unit Sheet:** Describes how to set up unit conversions on the Unit Sheet.
- **On other Sheets:** Describes each sheet on which units can be used.
- **Between Sheets:** Describes unit interactions with the sheets of the program.

#### On the Unit Sheet

There are four columns of fields on the Unit Sheet. The Multiply By fields relate the units in the From and To fields. The Add offset field adjusts the zeros of two related scales, such as Celsius and Fahrenheit temperature scales.

Examples:

yd and ft; Celsius and Fahrenheit

From	To	Multiply By	Add Offset
yd	ft	3	
C	F	1.8	32

Unit conversions can be made indirectly. An indirect conversion makes use of a third unit. In an indirect conversion, there must be a unit conversion between the third unit and each of the units to be converted.

Example:

yd and mi through ft

From	To	Multiply By	Add Offset
ft	yd	.3333333333	
mi	ft	5280	

Conversion from miles to yards is possible because each has a conversion defined in feet. It is not possible to extend this feature. Even if yd had a conversion defined in meters, the program could not convert mi to meters.

**On Other Sheets** Variable Sheet

**Calculation Unit:** The unit assumed for a variable when it is used in a rule.

**Display Unit:** The unit for display of variable value.

**Unit Entry:** The first entry in a unit field on this sheet sets the Calculation and Display units. Subsequent entries in the unit field change the Display unit, but do not affect the Calculation unit. Change in the Display unit is reflected on the Variable sub-sheet with the concurrent conversion of the value for the variable.

**Unit Conversion:** If the Display and Calculation units for any variable are different, their unit conversion must be defined on the Unit Sheet.

 Variable Subsheets

**Calculation Unit:** The unit assumed for a variable value when it is used in a rule.

**Display Unit:** The unit for display of a variable value.

**Unit Entry:** Calculation and Display values are entered on separate fields on this sheet. Change in the Display unit is reflected on the Variable Sheet with the concurrent conversion of the value for the variable.

**Unit Conversion:** If the Display and Calculation units for any variable are different, their unit conversion must be defined on the Unit Sheet.

 List Sheet

**Storage Unit:** The unit of values in a list (DIF file) created in another program and/or the unit for conversion to the Calculation unit of an associated variable.

**Display Unit:** The unit for display of list values.

**Unit Entry:** The first entry in a Unit field on this sheet sets the Storage and Display units. Subsequent entries in the Unit field change the Display unit, but do not affect the Storage unit. Change in the Display unit is reflected on the List Subsheet with the concurrent conversion of the value for the variable.

**Unit Conversion:** If the Display and Storage units for any list are different, their unit conversion must be defined on the Unit Sheet.

List Subsheets

**Storage Unit:** The unit of values in a list (DIF file) created in another program and/or the unit for conversion to the Calculation unit of an associated variable.

**Display Unit:** The unit for display of list values.

**Unit Entry:** Storage and Display values are entered on separate fields on this sheet. Change in the Display unit is reflected on the List subsheet with the concurrent conversion of the value for the variable.

**Unit Conversion:** If the Display and Storage units for any list are different, their unit conversion must be defined on the Unit Sheet.

### **Between Sheets**

Variable–Rule Sheets

Assignment of Display and Calculation units for variables can affect rules. All rule evaluation is performed in Calculation units. There must therefore be a coordination between equations containing variables associated with units and the assignment of units to the appropriate variables.

Variable–List Sheets

For a list associated with a variable, the conversion from Calculation unit of the variable to Storage unit of the list must be defined on the Unit Sheet.

User Function–List Sheets

If the lists involved in a user-defined function contain units, the function relationship is determined by the relationship between the Display units of the two lists.

If either list of the function is associated with a variable on the Variable Sheet, the Display unit of the list must be the same as the Calculation unit of the variable.

**Lists**

This section describes how lists are created and used.

**List Creation**

Lists are created on a List subsheet or a User Function subsheet. There are two methods for creating lists of values:

- Dive to a List subsheet or a User Function subsheet and type in each element of the desired list.
- For lists that are uniformly sequential and numerical, type the first and last values in the appropriate element position. Typing ! prompts:

Fill List: Y N

Typing Y fills in the intervening values using linear interpolation. Typing N cancels the command.

**Unit Compatibility**

Lists associated with units and used in the following ways need special attention when they are created:

- Associated with variables for list solution
- Used as domain and range lists for functions

Lists associated with variables must have a conversion provided on the Unit Sheet between the Storage unit of the list and the Calculation unit of the associated variable.

A list used as a domain or range for a function must be compatible with several units. If either list of the function is associated with a variable on the Variable Sheet, the Display unit of the list must be the same as the Calculation unit of the variable.

**List Command**

Solve Option

The Solve option of the List command uses either the Direct or Iterative Solver to solve a model for a list or lists of input values creating the desired output list(s) of values.

The required steps for using the Solve option are:

- Determine which variables are to be associated with input and output lists.
- Type **L** in the Status field of each variable to be associated with a list.
- Assign a value to each variable associated with an input list. If no value is assigned, the variable is assumed to be output.
- Associate all input lists with the desired variables. Typing **L** in the Status field of a variable fills in the Associated List field on the Variable subsheet with the same name as the variable. If a list by a different name is to be associated with the variable, dive to the Variable subsheet and enter the different list name.
- Type **/L!** to solve the model.

### Block Option

The Block option of the List command solves for an interval of elements as specified by the user. The steps for using the Block option are:

- Set up the model for the Solve option of the List command.
- Type **/LB**.
- Enter the element number of the first element in the interval in response to the prompt:

List Block: First element:

- Enter the element number of the last element in the interval in response to the prompt:

List Block: Last element:

The Solve option is invoked for the specified interval.

### Put List

The Put option of the List Command places the current value of any variable associated with a list in the specified element position of the associated list. The steps for its use are:

- Set up the model for Solve option of the List Command.
- Type **/LP**.
- In response to the prompt:

`Put to element:`

enter the element number the current values should be assigned to.

**Get List**

The Get option of the List Command gets the specified element value of any variable associated with a list and enters it on the Variable Sheet. The steps for its use are:

- Set up the model for Solve option of the List Command.
- Type **/LG**.
- In response to the prompt:

`Get from element:`

enter the element number of the values desired to be the current values on the Variable Sheet.

### Functions

This section describes both user functions and built-in functions in the TK!Solver program.

#### User Functions

##### Syntax

See functions under Equation components.

##### Function Name

See functions under Equation components.

##### Defining

Lists can be associated with one another through the User Function Sheet. The lists can either be created on the User Function Subsheet or be ones that already exist on the List Sheet. The associations are defined through the Mapping field on the User Function Sheet (see section on mappings which follows). Enter the list that supplies the values for the function argument as the domain and the list that supplies the values for the function value as the range.

##### Mappings/Interpolation

There are three options for the Mapping field: Table, Linear, and Step. The mapping defines the type of interpolation desired between explicit values of related lists. Upon an entry in the Name field, the default value, Table, is displayed in the Mapping field.

The three options are:

Table

No interpolation between explicit values.

Linear

Linear interpolation between explicit values. For example, consider two lists,  $A$  and  $B$ :

List $A$	List $B$
1	10
2	20
3	30

To find the value, say  $x$ , in list  $B$  associated with 2.5 in the list  $A$ , linear interpolation, a proportional method, is used to find that 25 is the correct list  $B$  value.

$$\frac{\text{List } A}{2.5 - 2} = \frac{\text{List } B}{30 - 20}$$

## □ Step

Interpolation such that for a given set of domain values, there is a constant value in the range. Between two domain values, the function evaluates to the range value associated with the lower numbered domain element.

Example:

List A	List B
1	10
2	20
3	30

To find the value in list *B* associated with 2.5 in the list *A*, step interpolation checks which elements of list *A* are involved and returns the range value associated with element 2, 20.

To define standard linear or step functions, all list values must be entered in either ascending or descending order because all interpolation is done point to point. No internal ordering takes place.

**Irregular Interpolations**

If lists used for User Functions are not in either ascending or descending order, interpolation may not act as expected. TK!Solver interpolation operates point to point. Given a value that requires interpolation, it looks only for the first interval of two values that can contain the desired point.

Example:

List A	List B
3	30
5	70
2	20
4	40

Using linear interpolation, a value of 3.8 in the domain returns a value of 46 because the interpolation interval was 30 to 70, not 30 to 40 as might be expected.

### Built-in Functions

There are two distinct classes of built-in functions, standard mathematical functions and special TK!Solver functions. The mathematical functions are written in their standard form and return their normal function values, whereas the syntax and behavior of the TK!Solver functions should be checked before use.

The elementary transcendental functions, such as  $\sin(x)$ ,  $\tan(x)$ ,  $e^x$ , and  $y^x$  are typically evaluated by the TK!Solver program to an accuracy of 10 significant digits.

However, as on many computer systems, evaluation of some functions for extreme argument values can lead to erroneous results. Thus, if the argument of the  $\sin(x)$  is  $10^n$ , the accuracy of the function value diminishes as  $n$  increases in size. At about  $10^{10}$  the accuracy associated with the function value is about two significant digits. The best values for evaluation of the trigonometric functions are in the principal value region.

Although specified throughout the documentation in uppercase, built-in functions names are case insensitive.





**Built-in Functions**

The following built-in functions are available in the TK!Solver program. They are divided into standard mathematical functions and special TK!Solver functions.

<b>Mathematical Functions</b>		<b>TK!Solver Functions</b>	
ABS	EXP	APPLY	MIN
ACOS	LN	ATAN2	NPV
ACOSH	LOG	COUNT	PI
ASIN	MOD	DOT	POLY
ASINH	SIN	E	SGN
ATAN	SINH	ELEMENT	SQRT
ATANH	TAN	GIVEN	STEP
COS	TANH	INT	SUM
COSH		MAX	

The following definitions are used for function arguments in the descriptions below. If more than one argument of the same type appears in the same function, they are numbered sequentially.

- x* Any numeric-valued expression.
- series* A set of two or more expressions separated by commas.
- listname* Any expression that evaluates to a symbolic value that is a list name. **Note:** A symbolic value **must** be preceded by an apostrophe (').
- ABS** Absolute Value  
ABS(*x*)  
Function value: the magnitude of the specified argument.
- ACOS** Arc Cosine  
ACOS(*x*)  $-1 \leq x \leq +1$   
Function value: the inverse cosine between 0 and  $\pi$ , in radians, of the specified argument.
- ACOSH** Arc Hyperbolic Cosine  
ACOSH(*x*)  $x \geq 1$   
Function value: the inverse hyperbolic cosine of the specified argument.

## APPLY

APPLY

Apply

APPLY(*function, domain*)

function: any value that evaluates to a user function name written as a symbolic value.

domain: any expression that evaluates to a value in the domain of the specified function.

Function value: The first argument evaluates to a user function name and this function is then evaluated for the value of the second argument.

Example: Use APPLY to generate the square or cube of the numbers 1,2,3,4 as desired.

Apply makes use of two functions for this example:

square(a) = b	cube(a) = c
square(2) = 4	cube(2) = 8

apply(*function, x*) yields:

apply('square,2) = square(2) = 4  
apply('cube, 2) = cube(2) = 8

ASIN

Arc Sine

ASIN( $x$ )  $-1 \leq x \leq +1$

Function value: the inverse sine between  $-\pi/2$  and  $\pi/2$ , in radians, of the specified argument.

ASINH

Arc Hyperbolic Sine

ASINH( $x$ )

Function Value: the inverse hyperbolic sine of the specified argument.

ATAN

Arc Tangent

ATAN( $x$ )

Function value: the inverse tangent between  $-\pi/2$  and  $\pi/2$ , in radians, of the specified argument.

ATANH

Arc Hyperbolic Tangent

ATANH( $x$ )  $-1 \leq x \leq +1$

Function value: the inverse hyperbolic tangent of the specified argument.

- ATAN2**      Arc Tangent  
                   $ATAN2(x1, x2)$   
                  Function value:  $ATAN(x1/x2)$ . Note this function returns the correct values by quadrant. This function returns values from 0 to  $2\pi$ , as opposed to  $atan$  which only returns values in the interval  $-\pi/2$  to  $\pi/2$ .
- Example: Use  $ATAN2$  to find:  
                   $atan(-.7071068/-.7071068)$ .
- $ATAN2(-.7071068, -.7071068) = -2.356194$  radians (or  $-135$  degrees)
- COS**            Cosine  
                   $COS(x)$   $x$  in radians  
                  Function value: the cosine of the specified argument.
- COSH**         Hyperbolic Cosine  
                   $COSH(x)$   
                  Function value: the hyperbolic cosine of the specified argument.
- COUNT**        Count  
                   $COUNT(series)$   
                   $COUNT(listname)$   
                  Function value: the number of elements in the specified list or series, or 0 if the specified listname contains no elements.
- DOT**            Dot Product  
                   $DOT(listname1, listname2)$   
                   $DOT(listname1, series)$   
                  Lists used with this function may contain blank entries which are treated as zeros.
- Function value: the dot product of the two specified lists. Each element of the first list is multiplied by its corresponding element in the second list and the products are summed.

## E

**E** Natural Log Base  
 $E()$  no argument  
Function value: an approximation of the value of  $e$ .

**ELEMENT** Element  
 $ELEMENT()$  no argument  
Function value: the number of the element currently being processed by the List Solver, or 0 if the List Solver is not processing.  
 $ELEMENT(listname, element\ number)$   
Function value: the value of the specified element in the specified list.  
 $ELEMENT(listname, element\ number, x)$   
Function value:  $x$  is a default function value. Returns the value of the specified element in the specified list or the specified default value, if that element does not exist.  
Example: Use **ELEMENT** to determine the value of the 5th and 6th elements of list  $Q$ .

Element #	List Q	$ELEMENT('Q, 5) = 11$
1	3	$ELEMENT('Q, 6, 1) = 1$
2	5	
3	7	
4	9	
5	11	

**EXP** Exponentiation  
 $EXP(x)$   
Function value:  $e$  raised to the power of the specified argument.

**GIVEN** Given Input  
 $GIVEN(variables, x1, x2)$   
variables: A series of expressions that evaluate to variable names written as symbolic values.  
 $x1$  is the value to return if all the variables have input values.  
 $x2$  is the value to return if any of the variables do not have input values.

Function value:  $x1$  if all of the variables have input values,  $x2$  if any of the variables do not have input values.

Examples: Use GIVEN with the variables  $a$  and  $b$ .

Example 1:

Input	Name	
	$a$	GIVEN('a','b', 0,1) = 1
5	$b$	

Example 2:

Input	Name	
3	$a$	GIVEN('a','b', 0,1) = 0
2	$b$	

INT

Integer Truncation

INT( $x$ )

Function value: the integer portion of the specified argument; effectively rounds down for positive numbers and up for negative numbers.

LN

Natural Logarithm

LN( $x$ )  $x > 0$

Function value: the base  $e$  logarithm of the specified argument.

LOG

Logarithm

LOG( $x$ )  $x > 0$

Function value: the base 10 logarithm of the specified argument.

MAX

Maximum

MAX(*series*)

MAX(*listname*)

Function value: the greatest value within the specified series or list. Symbolic values are ignored.

**MIN**

**MIN** Minimum  
MIN(*series*)  
MIN(*listname*)  
Function value: the least value within the specified series or list. Symbolic values are ignored.

**MOD** (MODULUS) Modulo  
MOD(*x1*, *x2*)  
Function value: For *x1* and *x2* greater than 0, the remainder of division of *x1* by *x2*, which in the general case is defined as the unique value of *r* that satisfies:  $x1 = x2 * n + r$ , where *n* is an integer and *r* is between 0 and *x2* (including 0 and not including *x2*).

**NPV** Net Present Value  
NPV(*x*, *series*)  
NPV(*x*, *listname*)  
*x* = the rate.  
*series* = cashflow values.  
*listname* = list of cashflow values.  
Function value: the net present value at the specified rate using the specified series or list of cash flow values. The result is equal to:

$$\sum_{i=0}^n \text{list}[i + 1] \left( \frac{1}{1 + x} \right)^i$$

where *i* takes values from 0 to *n* and *n* + 1 is the number of elements in the specified list. The rate is any expression that evaluates to a percent written as a decimal. The first cash flow is taken to be the cash flow at time zero.

Example: Use NPV to evaluate the cash flows of -1000, 600, 400, 300, 150, 10, and 10 with a rate of 12%.

$$\text{NPV}(.12, -1000, 600, 400, 300, 150, 10, 10) = 174.19$$

**PI**  $\pi$   
PI() no argument  
Function value: an approximation for the value of  $\pi$ .

POLY

Polynomial

POLY(*x, series*)

POLY(*x, listname*)

*x* = the variable value for the polynomial.

*series* = coefficients of the polynomial beginning with the coefficient of the variable with the greatest exponent. Zeros should be entered for any nonexistent standard terms.

*listname* = a list of coefficients.

Function value: the value of the polynomial for the specified variable using the specified series or list of coefficients. The result is equal to the sum of *n + 1* terms of a polynomial of the form:

$$\sum_{i=0}^n \text{list}[i + 1]x^{n-i}$$

where *i* takes values from 0 to *n* and there are *n + 1* elements in the specified list.

Example:

Use POLY to evaluate

$$6x^5 - 3x^3 + 7x^2 - 2x + 8 \text{ for } x = 4$$

$$\text{POLY}(4, 6, 0, -3, 7, -2, 8) = 6064$$

SGN

(SIGNUM) Sign

SGN(*x*)

Function value:

$x < 0$	$x = 0$	$x > 0$
$\text{sgn}(x) = -1$	$\text{sgn}(x) = 0$	$\text{sgn}(x) = 1$

SIN

Sine

SIN(*x*) *x* in radians

Function value: the sine of the specified argument.

SINH

Hyperbolic Sine

SINH(*x*)

Function value: the hyperbolic sine for the specified argument.







## Hardware Reference

This section describes how to use the TK!Solver program with your Radio Shack® TRS-80® Model 4 personal computer. The hardware information given here is intended only as a general guide. For specific information about your computer and printer, see your user manuals.

### Requirements

To use the TK!Solver program, you need:

#### Hardware

- The TRS-80 Model 4 computer with at least 128K bytes of memory and one or more disk drives. A hard disk is optional. A hard disk must be formatted before using. Refer to the *TRS-80 Model 4 Hard Disk Start-Up Manual* for more information.
- Blank, formatted storage diskettes (optional). See the *TRS-80 Model 4 Disk System Owner's Manual* or Chapter 2 of the TK!Solver Instruction Manual for information on formatting diskettes.
- A Radio Shack parallel printer (optional). Refer to “Global Sheet Settings” later in this section for information on printer defaults in the TK!Solver program.

#### Software

- The TK!Solver program diskette. The program diskette contains all the TRSDOS® files needed to boot the system.

#### Files

Your TK!Solver program diskette contains program files and Instruction Files. You may make six backup copies of the TK!Solver program and one backup copy on a hard disk for use on a single computer.

Any other copying of the TK!Solver program diskette, the TK!Solver program files, or the TRSDOS System files is unauthorized and is a violation of copyright law and the TK!Solver license agreement.

# TK!Solver Reference Manual

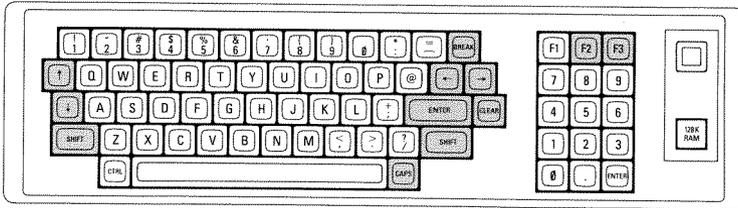
## TK!Solver Keys

The Instruction Files are sample files that are used throughout the Instruction Manual. The names of these files are:

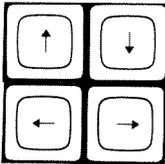
- COST/TK
- GRAVITY/TK
- TUNITS/TK
- LINEAR/TK
- MORTGAGE/TK
- NPOWER/TK
- FUNC/TK
- TFUNC/TK

### TK!Solver Keys

The TK!Solver program uses certain keys to perform actions and commands. Listed below are descriptions of the TK!Solver keys and the names and symbols for the keys as used in this text.



#### Arrow keys



The keys marked with the up, down, left, and right arrows are called the Arrow keys. These keys are represented in the text by the symbols  $\uparrow$ ,  $\downarrow$ ,  $\leftarrow$ , and  $\rightarrow$ .  $\leftarrow$  and  $\rightarrow$  are on the right side of the alphabetic keys.  $\uparrow$  and  $\downarrow$  are located on the left side of the alphabetic keys. Pressing  $\leftarrow$ ,  $\rightarrow$ ,  $\uparrow$ , or  $\downarrow$  moves the cursor in the direction of the arrow.

#### Backspace key



The Backspace key is marked **F2** and is located at the top of the numeric keypad on the right side of the keyboard. It is represented in the text by the symbol  $\leftarrow$ . Pressing  $\leftarrow$  erases the character to the left of the cue.

#### Break key



The Break key is located at the top right of the keyboard and is marked **BREAK**. Pressing this key breaks out of or cancels the current entry. It is represented in the text by the symbol  $\otimes$ .

#### Clear key



The Clear key is located on the right side of the keyboard and is marked **CLEAR**. Hold down the Clear key to type a command or to display special symbols on the screen.

In the TK!Solver program, use the Clear key to type the following special characters:

To type:                      Hold down:      and Press:

$\wedge$  (exponentiation symbol)      Clear      ;

$\backslash$  (backslash)      Clear      /

$\_$  (underscore)      Clear      Enter

#### Edit key



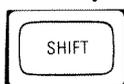
The Edit key calls the TK!Solver Editor. The Editor allows you to change an entry. The Edit key is labeled **F3** on your keyboard and is located above the numeric keypad on the right side of your keyboard. The F3 key is represented in this text by the symbol  $\boxplus$ .

#### Enter key



The Enter key is marked **ENTER** and is a large key located on the right side of the alphabetic keys. It is represented in the text by the symbol  $\blacklozenge$ . Pressing  $\blacklozenge$  enters information at the current cursor position or completes a command.

#### Shift keys



There are two Shift keys, one on either side of the bottom row of alphabetic keys. The Shift keys are marked **SHIFT**. Hold down a Shift key to type uppercase characters and the special symbols shown on the top half of the number keys.

## Global Sheet Settings



To lock in uppercase letters, press the **CAPS** key to the right of the space bar. To return the keyboard to both uppercase and lowercase characters, press the CAPS key again.

## Global Sheet Settings

The TK!Solver Global Sheet contains a list of default settings for your printer. The following two default printer settings are set by TRSDOS:

- Printer Device or Filename: \*PR
- Line End: System-defined

The default for the **Printer Device or Filename** field is the TRSDOS device specification **\*PR**. If you leave \*PR in this field, you can send information contained in a TK!Solver model to a printer. See “Devices” later in this section for more information about device specifications.

If you want to alter the way a file is printed, use the TRSDOS command FORMS. See the *TRS-80 Model 4 Disk System Owner's Manual* for details.

The default for the **Line End** field is **System-defined (S)**. This means that TRSDOS already knows when to end a line. You cannot change the value of this field.

### Printer Setup Strings

A setup string assigns printing options such as underlining and boldfacing. Enter a setup string into the **Printer Setup String** field, located on the TK!Solver Global Sheet.

Some examples of printer strings for the TRS-80 Model 4 and the Radio Shack DMP 200 printer are listed below.

- Underline ON: ^H0F
- Underline OFF: ^H0E
- Boldface ON: ^H1B^H1F
- Boldface OFF: ^H1B^H20
- Form Feed: ^H0C
- Correspondence Quality Printing: ^H1B^H12

- Standard Character Printing: ^H1B^H13
- Condensed Character Printing: ^H1B^H14
- Compressed Character Printing: ^H1B^H17

For further information on setup strings, see the section “Global Sheet” in Part II of the TK!Solver Reference Manual and the user’s guide for your printer.

## **Filenames**

Filenames have certain restrictions placed on them by TRSDOS. These restrictions are:

- Filenames must be one to eight characters long.
- Filenames may contain any uppercase or lowercase letter and any number from 0 through 9.
- Filenames must begin with a letter.
- Filenames cannot contain any other characters.

## **File Specifications**

File specifications are added to a filename to identify or locate a file. In the TK!Solver program, you use file specifications when naming a file for the Storage command.

You do not always have to use these specifications when you name a file. If you leave out a specification, the TK!Solver program uses a default. These defaults are described below. The file specifications used by TRSDOS include the drive name, the file extension, the password, and the device. The general form for the TRSDOS file specifications is:

*filename/extension.password:device*

Each of these specifications is explained below.

### **Extension**

An extension identifies different types of files. Extensions identify BASIC files, text files, TK!Solver files, and other application files. An extension is one to three alphanumeric characters preceded by a slash (/). The first character of an extension must be a letter.

## Passwords

The TK!Solver program uses three different types of extensions. You do not have to specify the extension when you load or save a file in the TK!Solver program.

Files saved using the Storage command with the S, V, U, and F options are automatically given the extension /TK. Files loaded with the L option must have filenames with the extension /TK, although you do not have to specify this extension.

Files saved using the DIF™ option (#) of the Storage command are automatically given the extension /DIF. Files loaded using the DIF option must have filenames with the extension /DIF.

Files saved using the Print Command (/P) are automatically given the extension /PRE.

### Passwords

A password, when added to a filename, restricts other users from looking at or changing the file, unless they know the password. Passwords are one to eight alphanumeric characters preceded by a period (.). The first character of a password must be a letter. When you name a file, add the password after the filename or after its extension.

You do not have to include a password when creating a file in the TK!Solver program. However, if an existing file already has a password, you must include the password after the filename when you load or save the file.

### Devices

TRSDOS has two different kinds of devices: physical and logical. A physical device is a piece of hardware, such as a disk drive, a keyboard, or a printer. The disk drives are identified by the numbers 0 through 9. You can have up to two disk drives installed in the computer case next to the screen. The bottom drive is called drive 0 and the drive above it is called drive 1. A hard disk is also considered a drive and is called drive 0 after it is installed.

A logical device is the connection between TRSDOS and the drives, a printer, or other physical devices. TRSDOS uses a device name, also called **devspec**, to identify a connection between phys-

ical devices. The devspec is a two-character abbreviation preceded by an asterisk (\*). The TRSDOS devspecs are listed below:

- \*DO Display Output (video)
- \*JL Job Log
- \*KI Keyboard Input
- \*PR Printer
- \*SO Standard Output
- \*SI Standard Input

By separating the connection of a physical device from the device itself, you can sometimes substitute one physical device for another. You can also substitute a file for a physical device.

In the TK!Solver program, the devspec **\*PR** is the default setting for the Printer Device or Filename field on the Global Sheet. If you replace **\*PR** in this field with a filename, the information that would have been printed will be saved to that filename instead.

## **System Notes**

- To load the TK!Solver program, type **TK** and press **◆**.
- To quit the TK!Solver program, type **/Q**. Confirm this command by typing **Y**.
- To use the TK!Solver program, you must have a copy of the file **TK/OVL** available on a diskette in a drive or on a hard disk. If you want to use the Help Facility, you must also have a copy of the file **TK/HLP**.
- Any diskette you have in drive **0** must contain the following TRSDOS System files: **SYS0/SYS** through **SYS4/SYS**, **SYS6/SYS**, **SYS7/SYS**, and **SYS10/SYS** through **SYS13/SYS**.
- If you use the Save option of the Storage command to save a file in the TK!Solver program and do not specify a drive number, the TK!Solver program may cancel the Storage command and display the following message:

```
Disk Space or Disk Directory Full
```

## System Notes

This message refers to one of the following problems:

- there is not enough space on any of the diskettes
- all the diskettes in the drives are write-protected

Although you were not able to save the file to any of the diskettes currently in the drives, the TK!Solver program still contains the information.

Before saving the file again, check to see if the diskette that is to contain the file is write-protected. A write-protected diskette is one that has either no square notch on the left or a metallic tab covering the notch.

If the diskette has a write-protect tab, remove the tab and reinsert the diskette. If the diskette is notchless, replace the diskette with a notched formatted diskette. Then use the Save option of the Storage command again to save the file on the unprotected diskette.

If the diskette is not write-protected, then it does not have enough space left to contain the file to be saved. Replace this diskette with another notched, unprotected diskette and save the file.

- In the TK!Solver program, you can use uppercase or lowercase letters for commands and built-in functions. However, the program does distinguish between uppercase and lowercase letters in all other names, including variables, units, user functions, and lists.
- The error signal is a beep. It is sometimes accompanied by a message on the Prompt/Error line.
- The numeric range for your computer is any number with an absolute value between 1E-120 and 1E120.
- The TK!Solver program supports the TRSDOS wildcard \$. When used with the Load or Save command, \$ replaces one character in a filename or its specifications. Refer to the *TRS-80 Model 4 Disk System Owner's Manual* for more information about this wildcard.





## **TK!Solver Files**

This appendix describes the types of files that can be created using the TK!Solver program. These files are all created from the program onto the operating system level. (See the Hardware Reference section for the filename conventions for your computer.)

### **Print Files**

A print file is created with the Print command (/P) when a filename is specified in the Printer Device or Filename field on the Global Sheet. Each print command overwrites the previous contents of the file, so the filename should be changed on the Global Sheet if the results of more than one Print command are to be saved.

A print file contains data from a single TK!Solver sheet or subsheet and looks similar to a sheet or subsheet displayed on the screen, except that it can display as much as an entire sheet or as little as a single row. A print file for a subsheet begins with a heading that identifies the variable, list, or user function name on the first line of the file. The options for printing defined on the Global Sheet are used in the creation of the print file. The lines printed into the file are displayed simultaneously on the screen with the defined output format as the file is created.

### **Plot and Table Files**

A plot file is created from the Plot Sheet, and a table file from the Table Sheet, with the Action command (!) given two conditions:

- The Screen or Printer field on the Plot or Table Sheet is set to the printer option (P).
- A filename is specified in the Printer Device or Filename field on the Global Sheet.

Each Plot or Table action overwrites the previous contents of a file if a new name is not specified.

A plot file contains a single-page plot and a table file contains a single- or multiple-page table. The plot or table looks similar to a plot or table displayed only on the screen except that the options for printing defined on the Global Sheet are used in the plot or table file. The plot or table printed to the file is displayed simultaneously on the screen with the defined output format as it is created.

### Storage Files

A storage file is created with the Save Model, Variable Save, Unit Save, and Function Save options of the Storage command (/S); a TK!SolverPack™ file has the same format. Either may be loaded into the program with the Load option of the Storage command. If the user tries to save data using an existing filename, the program prompts to see whether the file should be overwritten.

A storage file can be recognized from the operating system level by “TK!1” on the first line of the file.

A storage file saved with the Save Model option (/SS) contains information used to recreate a particular TK!Solver session, including all data, and the current screen display and cursor position. The other kinds of storage files save data related to variables, units, or functions, but do not save a cursor position or current display.

### DIF Format Files

The DIF™ file format is a data interchange format developed by Software Arts, Inc., designed to provide a standard convention for the exchange of data between programs. A DIF file can be created with the DIF Save option of the Storage command (/S#S) within the TK!Solver program, or by another program that uses the DIF format.

DIF files are loaded into the TK!Solver program with the DIF Load option of the Storage command. If created by another program, they must be organized in a format consistent with the List Sheet and List subsheets. In order to load properly, all labels and string values in a DIF file must conform to the name item and symbolic value conventions for the TK!Solver program. All labels and string values cannot begin with a digit and can contain only letters, digits, and the special characters allowed in name items:

@ # \$ % \_

The TK!Solver program inserts the apostrophe at the beginning of string values to make them into symbolic values. If any invalid character exists in a label or string value, the name or symbolic value loaded into a field is truncated to include only the valid characters.

A TK!Solver DIF Save file contains list information from the List Sheet and List subsheets only and must be saved from the List Sheet.

A DIF file can be recognized from the operating system level by the Header Item label “TABLE” on the first line of the file and others such as “VECTOR”, “TUPLE”, and “DATA” in the top portion.

The DIF Clearinghouse coordinates and distributes information about DIF and commercially available programs that use DIF. For information, write to:

DIF Clearinghouse  
PO Box 638  
Newton Lower Falls, MA 02162

The clearinghouse would like to add to its list of programs that support DIF. No royalties or fees are charged for using DIF, but the clearinghouse makes three requests of authors of commercially available programs using DIF:

- Please send a one-page description of your program to the DIF Clearinghouse. Describe what the program does, what computers it runs on, how it uses DIF, and how it can be purchased.
- Because DIF is a trademark of Software Arts Products Corp., the “tm” designation must be used if DIF is mentioned in any material you publish about your program. The material should also include the statement “DIF is a trademark of Software Arts Products Corp.”
- If you are unable to test your program with the TK!Solver and VisiCalc® programs, please write the DIF Clearinghouse before advertising your product as supporting DIF. Any program employing DIF should be capable of transferring data to and from both programs.

### Unit Name Conventions

The following abbreviations for the names of standard units of measurement are used in TK!SolverPack™ models and are recommended conventions for use with the TK!Solver program (although the program accepts any unit names the user chooses). The abbreviations are given here in uppercase and lowercase to show their use on computers that use both cases. No periods are used.

#### Unit Abbreviations

##### Length

Inch	in
Foot	ft
Yard	yd
Mile	mi
Angstrom	Ang
Micron	um
Millimeter	mm
Centimeter	cm
Meter	m
Kilometer	km

##### Area

Square meter	m <sup>2</sup>
Hectare	ha
Acre	acre

##### Volume

Ounce	fl_oz
Pint	pt
Quart	qt
Gallon	gal
Cubic centimeter	cm <sup>3</sup>
Milliliter	ml
Liter	l

**Weight, Mass**

Ounce	oz
Ounce, troy	oz_t
Pound	lb
Pound, troy	lb_t
Ton, long	ton_l
Ton, short	ton

Milligram	mg
Gram	g
Kilogram	kg
Metric ton	t
Slug	slug

**Time**

Millisecond	ms
Second	s
Minute	min
Hour	h
Day	day
Week	wk
Month	mo
Year	yr

**Temperature**

Kelvin	K
Degree Centigrade	oC
Degree Fahrenheit	oF
Degree Rankine	oR

**Planar Angle**

Radian	rad
Degree	deg
Minutes of arc	'
Seconds of arc	"

**Pressure**

Atmosphere	atm
Centimeters of mercury	cm_Hg
Feet of water	ft_H2O
Pounds per square inch	lb/in^2
Pascal	Pa

## Unit Name Conventions

### Energy, Work, Heat, Force, Power, Frequency

British thermal unit	Btu
Calorie	cal
Kilocalorie	kcal
Erg	erg
Horsepower	hp
Watt	W
Kilowatt	kW
Joule	J
Dyne	dyn
Newton	N
Hertz	Hz
KiloHertz	kHz
MegaHertz	MHz

### Electricity

Ampere	A
Milliampere	mA
Volt	V
Ohm	Ohm

### Miscellaneous

Percent	%
United States dollars	\$
Deutsche mark	DM

Certain conventions have been devised and are recommended to maintain consistency in the development of new abbreviations:

- All units raised to a power are expressed as exponents of the standard abbreviations. For example:  
A square inch is expressed as in<sup>2</sup>  
A cubic yard is expressed as yd<sup>3</sup>
- The product of two units is symbolized by an asterisk between the two unit names. For example:  
An inch-pound is expressed as in\*lb  
A kilowatt-hour is expressed as kW\*h
- All units derived from the division of two units are expressed with a slash between them. For example:  
Miles per hour is expressed as mi/h (NOT mph)
- Underscores are used for clarity in certain combined or qualified unit names. For example:  
A fluid ounce is expressed as fl\_oz
- Compound unit names are formed by combining the above principles. For example:  
Pounds per square inch is expressed as lb/in<sup>2</sup> (NOT psi)

## Prompt Messages

### Startup Screen:

The following prompt message appears on the bottom of the initial startup screen of the TK!Solver program on most computers:

**\*\*\*\*\* Press ENTER to Start \*\*\*\*\***

- Cause:            Typing the command to begin the TK!Solver program, with a disk specifier if necessary, from the operating system level of a personal computer.
- Actions:          Pressing the Enter key to begin the program (to stop the program after pressing the Enter key, wait until the TK!Solver program finishes loading and type /QY); pressing the Break key to return immediately to the operating system level.

### Prompt/Error Line:

The following prompt messages appear on the Prompt/Error line. Any command can be cancelled and the prompt erased by pressing the Break key.

**/ Command: B C D E I L M P Q R S W !**

- Cause:            Typing /.
- Actions:          Choosing one of the Slash commands:
- |   |                    |
|---|--------------------|
| B | Blank command      |
| C | Copy command       |
| D | Delete command     |
| E | Edit Field command |
| I | Insert command     |
| L | List command       |
| M | Move command       |
| P | Print command      |
| Q | Quit command       |
| R | Reset command      |
| S | Storage command    |
| W | Window command     |
| ! | Solve command      |

Cancelling the command by pressing the Break key or typing any character that is not a valid Slash command.

## Prompts

### **Blank: Point to last field** (*Pointer Indicator*)

**Cause:** Typing the Blank command (/B) with the cursor on the first item to blank.

**Actions:** Pointing to the last item to blank in the same column, if necessary, by using the Arrow keys and the Goto command (:) in any combination, and completing the command with the Enter key. (The Pointer Indicator changes to indicate the cursor position during pointing.)

### **Copy: Point to destination** (*Pointer Indicator*)

**Cause:** Typing the Copy command (/C) with the cursor on the item to be copied.

**Actions:** Pointing to the destination field by using the Arrow keys, the Goto command (:), and the Switch command (;) in any combination, and completing the command with the Enter key. (The Pointer Indicator changes to indicate the cursor position during pointing.) If the copied item is invalid in the new location, an error message appears on the Prompt/Error line describing the type of error, and the Editor is called automatically to edit the item.

### **DIF Load: Filename:**

**Cause:** Typing the DIF Load option of the Storage command (/S#L).

**Actions:** Typing a file specification followed by the Enter key, or calling the Directory Scrolling feature with the Right Arrow key to select a filename.

### **DIF Save: Filename:**

**Cause:** Typing the DIF Save option of the Storage command (/S#S), pointing to the last list in the block of lists to save, and completing the command with the Enter key.

**Actions:** Typing a file specification followed by the Enter key, or calling the Directory Scrolling feature with the Right Arrow key to select a filename.

**DIF Save: Point to last list** (*Pointer Indicator*)

**Cause:** Typing the DIF Save option of the Storage command (/S#S) with the cursor on the first list to save on the List Sheet.

**Actions:** Pointing to the last list to save by using the Arrow keys and Goto command (:) in any combination, and completing the command with the Enter key. (The Pointer Indicator changes to indicate the cursor position during pointing.) A prompt appears for a filename when the pointing is complete.

**DIF Storage: S L**

**Cause:** Typing the DIF option of the Storage command (/S#).

**Actions:** Choosing one of the List command options:

S	Save
L	Load

    Cancelling the command by typing any character that is not a valid option.

**Delete: Filename:**

**Cause:** Typing the Delete option of the Storage command (/SD).

**Actions:** Typing a file specification followed by the Enter key, or calling the Directory Scrolling feature with the Right Arrow key to select a filename.

**Delete filename:** *filename*

**Cause:** Typing the Delete option of the Storage command (/SD) and calling the Directory Scrolling feature by pressing the Right Arrow key or entering a partial file specification with the Right Arrow key.

**Actions:** Pressing the Right Arrow key to display the next filename until the appropriate filename appears and then pressing the Enter key to select it; cancelling the command by using the Right Arrow key past the last filename or pressing the Break key.

## Prompts

### Delete filename: *filename* : Y N

- Cause: Typing the Delete option of the Storage command (/SD) and providing a filename by entering it or selecting it with the Directory Scrolling feature.
- Actions: Typing Y to delete the file; typing N to cancel the command.

### Delete: Point to last row (*Pointer Indicator*)

- Cause: Typing the Delete command (/D) with the cursor on the first row to delete.
- Actions: Pointing to the last row to delete, if necessary, by using the Arrow keys and the Goto command (:) in any combination, and completing the command with the Enter key. (The Pointer Indicator changes to indicate the cursor position during pointing.)

### Fill List: Y N

- Cause: Typing the Action command (!) with the cursor on a List subsheet or on the domain or range list of a User Function subsheet with a numeric value in the first and last elements to be filled in.
- Actions: Typing Y to fill the list; typing N to cancel the command.

### Function Save: Filename:

- Cause: Typing the Function Save option of the Storage command (/SF).
- Actions: Typing a file specification followed by the Enter key, or calling the Directory Scrolling feature with the Right Arrow key to select a filename.

### Get from element:

- Cause: Typing the Get option of the List command (/LG).
- Actions: Typing an integer or integer expression to indicate the element number from which to retrieve values for all associated lists, and pressing the Enter key to complete the command.

**Goto: Destination or search:**

- Cause: Typing the Goto command (:).
- Actions: Using the Destination option by providing a row number and/or label letter to move the cursor to the specified destination, and pressing the Enter key to complete the command; using the Search option by typing a quotation mark (") followed by an exact match of the portion of an item displayed in a field in the same column, and pressing the Enter key to complete the command.

**Help: ? or topic:**

- Cause: Typing the Help command (?) or completing a previous Help request.
- Actions: Typing ? to display Help Facility instructions; typing a topic name and the Enter key to search for a topic; pressing the Enter or Break key to exit the Help Facility.

**List: ! B P G**

- Cause: Typing the List command (/L).
- Actions: Choosing one of the List command options:
- |   |       |
|---|-------|
| ! | Solve |
| B | Block |
| P | Put   |
| G | Get   |

Cancelling the command by typing any character that is not a valid option.

**List Block: First element:**

- Cause: Typing the Block option of the List command (/LB).
- Actions: Typing an integer or integer expression to indicate the beginning of the block of elements to be processed from all associated lists, and pressing the Enter key. A prompt appears for the final element number.

## Prompts

### List Block: Last element:

- Cause: Typing the Block option of the List command and providing a first list element number (/LBelement ♦).
- Actions: Typing an integer or integer expression to indicate the end of the block of elements to be processed from all associated lists, and pressing the Enter key to complete the command.

### Load filename: *filename*

- Cause: Typing a Load option of the Storage command (/SL or /S#L) and calling the Directory Scrolling feature by pressing the Right Arrow key or entering a partial file specification with the Right Arrow key.
- Actions: Pressing the Right Arrow key to display the next filename until the appropriate filename appears and pressing the Enter key to select it; cancelling the command by using the Right Arrow key past the last filename or pressing the Break key.

### Load: Filename:

- Cause: Typing the Load option of the Storage command (/SL).
- Actions: Typing a file specification followed by the Enter key, or calling the Directory Scrolling feature with the Right Arrow key to select a filename.

### Move: Point to destination (*Pointer Indicator*)

- Cause: Typing the Move command (/M) with the cursor on the row to be moved.
- Actions: Pointing to the destination row by using the Arrow keys and Goto command (:) in any combination, and completing the command with the Enter key. (The Pointer Indicator changes to indicate the cursor position during pointing.)

### **Next page: Y N**

Cause: Typing a topic name in the Help Facility that takes more than one page to display.

Actions: Typing **Y** to display the next page; typing **N** to redisplay the Help topic prompt.

### **No topic topic; More help: Y N**

Cause: Typing a topic that does not exist in the Help Facility.

Actions: Typing **Y** to redisplay the Help topic prompt and entering another topic; typing **N** to exit the Help Facility.

### **Overwrite filename: filename : Y N**

Cause: Typing a Save option of the Storage command (/SS, /SV, /SU, /SF, or /S#S) and providing a filename that already exists by entering it or selecting it with the Directory Scrolling feature.

Actions: Typing **Y** to overwrite the file; typing **N** to cancel the command.

### **Print: Point to last line (Pointer Indicator)**

Cause: Typing the Print command (/P) with the cursor on the first row to be printed.

Actions: Pointing to the last row to print by using the Arrow keys and Goto command (:) in any combination, and completing the command with the Enter key. (The Pointer Indicator changes to indicate the cursor position during pointing.)

### **Put to element:**

Cause: Typing the Put option of the List command (/LP).

Actions: Typing an integer or integer expression to indicate the element number in which to store values from the Variable Sheet for all associated lists, and pressing the Enter key to complete the command.

## Prompts

### Quit: Y N

- Cause: Typing the Quit command (/Q).
- Actions: Typing **Y** to exit the program and delete all data; typing **N** or pressing the Break key to cancel the command and remain in the program.

### Reset all: Y N

- Cause: Typing the All option of the Reset command (/RA).
- Actions: Typing **Y** to clear the program; typing **N** to cancel the command.

### Reset current sheet: Y N

- Cause: Typing the Sheet option of the Reset command (/RS).
- Actions: Typing **Y** to clear the current sheet; typing **N** to cancel the command.

### Reset variable values: Y N

- Cause: Typing the Variable option of the Reset command (/RV).
- Actions: Typing **Y** to reset all variable values; typing **N** to cancel the command.

### Reset: V S A

- Cause: Typing the Reset command (/R).
- Actions: Choosing one of the Reset command options:
- |   |          |
|---|----------|
| V | Variable |
| S | Sheet    |
| A | All      |
- Cancelling the command by typing any character that is not a valid option.

### Save: Filename:

Cause: Typing the Save Model option of the Storage command (/SS).

Actions: Typing a file specification followed by the Enter key, or calling the Directory Scrolling feature with the Right Arrow key to select a filename.

### Save filename: *filename*

Cause: Typing a Save option of the Storage command (/SS, /SV, /SU, /SF, or /S#S) and calling the Directory Scrolling feature by pressing the Right Arrow key or entering a partial file specification with a Right Arrow key.

Actions: Pressing the Right Arrow key to display the next filename until the appropriate filename appears and pressing the Enter key to select it; cancelling the command by using the Right Arrow key past the last filename or pressing the Break key.

### Sheet: V R U G L F P T

Cause: Typing the Select command (=).

Actions: Choosing one of the Select command options:

V	Variable Sheet
R	Rule Sheet
U	Unit Sheet
G	Global Sheet
L	List Sheet
F	User Function Sheet
P	Plot Sheet
T	Table Sheet

Cancelling the command by typing any character that is not a valid option.

### Storage: L S V U F # D

Cause: Typing the Storage command (/S).

Actions: Choosing one of the Storage command options:

## Prompts

L	Load File
S	Save Model
V	Variable Save
U	Unit Save
F	User Function Save
#	DIF Storage
D	Delete File

Cancelling the command by typing any character that is not a valid option.

### Unit Save: Filename:

Cause: Typing the Unit Save option of the Storage command (/SU).

Actions: Typing a file specification followed by the Enter key, or calling the Directory Scrolling feature with the Right Arrow key to select a filename.

### Variable Save: Filename:

Cause: Typing the Variable Save option of the Storage command (/SV).

Actions: Typing a file specification followed by the Enter key, or calling the Directory Scrolling feature with the Right Arrow key to select a filename.

### Window: 1 V R U G L F P T

Cause: Typing the Window command (/W).

Actions: Choosing one of the Window command options:

1	One Window
V	Variable Sheet
R	Rule Sheet
U	Unit Sheet
G	Global Sheet
L	List Sheet
F	User Function Sheet
P	Plot Sheet
T	Table Sheet

Cancelling the command by typing any character that is not a valid option.

## **Status Messages**

### **Cleared Screen:**

The following status message appears at the top of a blank screen:

```
Loading the TK!Solver program
  Copyright 1983 Software Arts, Inc.
```

**Cause:** Typing the Enter key from the TK!Solver startup screen and beginning the program.

**Actions:** Waiting until the message disappears from the screen before beginning to use the program.

### **Status Line:**

The following status messages appear on the Status Line:

**!**

**Cause:** Entering information into any field that might require re-resolution of the model.

**Actions:** Calling one of the solvers with the Action command (!) from the Variable Sheet or the Rule Sheet, or the Solve command (/!) from any sheet.

### **LOW**

**Cause:** Any action that depletes remaining memory below 1K bytes.

**Actions:** Saving information with the Storage command (/S) and clearing some or all of a model with the Reset command (/R), or deleting list values with the Delete command (/D); performing actions which require minimal memory.

### **OUT**

**Cause:** Any action that depletes remaining memory.

**Actions:** Saving information with the Storage command (/S) and clearing some or all of a model with the Reset command (/R), or deleting list values with the Delete command (/D); exiting the program with the Quit command (/Q).

## Status Messages

### Prompt/Error Line:

The following status messages appear on the Prompt/Error Line. Pressing the Break key cancels all actions unless noted otherwise.

### Blanking fields

Cause: Typing the Blank command (/B), pointing to a block of rows, and pressing the Enter key to complete the command.

### Deleting filename

Cause: Typing the Delete option of the Storage command (/SD), providing a filename by entering it or selecting it with Directory Scrolling, and confirming the deletion by typing Y.

### Deleting rows

Cause: Typing the Delete command (/D), pointing to a block of rows, and pressing the Enter key to complete the command. The Break key does not interrupt this action.

### Direct Solver

Cause: Typing the Action command (!) from the Variable Sheet or the Rule Sheet, or the Solve command (/!) from any sheet.

### (Edit)

Cause: Typing the Edit Field command (/E) or the Edit command (☑) to call the Editor. (This message also appears at the beginning of error messages that call the Editor automatically. See the appendix on Error Messages for details.)

### Filling in list

Cause: Typing the Action command (!) from a List subsheet or User Function subsheet and typing Y to confirm.

**For Help, type ?**

Cause: Pressing the Enter key from the TK!Solver startup screen and waiting until the program loads. (This message is only a reminder on how to call the Help Facility and requires no action.)

**Iterative Solver: *step out of estimate***

Cause: Typing the Action command (!) from the Variable Sheet or the Rule Sheet, or the Solve command (/!) from any sheet if a guess value is set for any variable on the Variable Sheet, or automatic iteration is turned on and first guess values are provided on Variable subsheets. The step of the iteration is displayed out of the number of iterations the program expects the solution to take. The estimate is revised during solution.

**List Solver: *element*, Direct Solver**

Cause: Typing the Solve option (/L!) or the Block option (/LB) of the List command to apply the Direct Solver for all associated lists in a model. A count is displayed for the list element currently being processed.

**List Solver: *element*, Iterative Solver: *step out of estimate***

Cause: Typing the Solve option (/L!) or the Block option (/LB) of the List command to apply the Iterative Solver to all associated lists in a model if a guess value is set for any variable on the Variable Sheet, or automatic iteration is turned on and first guess values are provided on Variable subsheets. A count is displayed for the element currently being processed. The step of the iteration is displayed out of the number of iterations the program expects the solution to take. The estimate is revised during solution.

# TK!Solver Reference Manual

---

## Status Messages

### List Solver: *element*, Iterative Solver: *step*

Cause: The same message as above if too long to display on a screen width less than 80 characters.

### Loading from *filename*

Cause: Typing a Load option of the Storage command (/SL or /S#L), and providing a filename by entering it or selecting it with Directory Scrolling.

### Resetting all variables

Cause: Typing the Variable option of the Reset command (/RV) and confirming it by typing Y. The Break key does not interrupt this action.

### Resetting current sheet

Cause: Typing the Sheet option of the Reset command (/RS) and confirming it by typing Y. The Break key does not interrupt this action.

### Resetting everything

Cause: Typing the All option of the Reset command (/RA) and confirming it by typing Y. The Break key does not interrupt this action.

### Saving to *filename*

Cause: Typing a Save option of the Storage command (/SS, /SV, /SU, /SF, or /S#S), pointing to the last list in the case of the DIF Save option, providing a filename by entering it or selecting it with Directory Scrolling, and confirming for overwriting if necessary by typing Y.

### Searching the Help file

Cause: Typing a topic followed by the Enter key in the Help Facility.

**Rule Sheet: Status Column:**

The explanation of a status message in a Status field on the Rule Sheet may be viewed on the Status line at the top of the screen by moving the cursor to the Status field in question. (See the appendix on Error Messages for the error messages that appear in Status fields.)

The following status messages are available:

**Comment**

Cause: The Rule field on the same row contains only a comment, beginning with a quotation mark ("). No character displays in the Status field.

**Satisfied**

Cause: The Rule field on the same row contains an equation evaluated by either of the solvers during the previous solution and determined to have a supplied or derived value for each variable. No character displays in the Status field.

**\* Unsatisfied**

Cause: The Rule field on the same row contains an equation that was not evaluated by either of the solvers during the previous solution. An asterisk displays in the Status field.

## Error Messages

### Error Messages

Error messages are organized according to where they appear on the TK!Solver screen: on the startup screen, on the Prompt/Error line, in the Status fields on the Variable and Rule sheets, at the bottom of the screen, in value locations, and on a cleared screen. The explanations below are approximate and intended to serve as a guide only.

Error messages are documented generally for all machines that use diskettes. An error message may be displayed on your computer that is not listed below. See the Hardware Reference section for further information on machine-specific messages.

If a message is displayed that does not appear anywhere in the Reference Manual, or a condition other than the one described produced the error, write down any pertinent information and contact your dealer.

### Startup Screen:

The following error messages appear at the beginning of the program on the initial screen on most implementations:

**\*\*\*\*\* Disk Read Error \*\*\*\*\***  
**... Press ENTER to Exit ...**

Cause: Attempting to use a worn or damaged TK!Solver program diskette, or encountering a hardware error.

Actions: Pressing the Enter key and using your backup TK!Solver diskette or obtaining a new TK!Solver program diskette. (See your license agreement for details.) Consulting your hardware operations manual for hardware testing procedures; noting the message above and contacting your dealer.

**\*\*\*\*\* Invalid Program Disk \*\*\*\*\***

... Press ENTER to Exit ...

- Cause:** Attempting to use a TK!Solver program diskette that has been overwritten or a diskette that contains the results of an attempt to copy the entire TK!Solver program diskette.
- Actions:** Pressing the Enter key and obtaining a proper TK!Solver program diskette; reformatting the diskette before using it for other purposes. (See your license agreement for details.)

**\*\*\*\*\* Insufficient Memory \*\*\*\*\***

... Press ENTER to Exit ...

- Cause:** Attempting to use a TK!Solver program diskette with insufficient memory available.
- Actions:** Contacting your dealer and obtaining more memory for your computer system. The TK!Solver program requires a recommended minimum memory configuration. (See the Hardware Reference section for details.)

**Prompt/Error Line:**

The following error messages appear on the Prompt/Error line. Expressions or entries that contain multiple errors are diagnosed for the first recognized error. Error messages that begin with "(Edit)" automatically call the Editor for error correction: use the Enter key to enter the corrections and exit the Editor; use the Break key to exit the Editor and return the previous correct item.

**Deleted entity**

- Cause:** Attempting to perform an action on a Variable, List, or User Function subsheet that is still displayed on the screen for a variable, list, or user function that has been deleted using /D or /RSY.
- Actions:** Returning to the corresponding sheet, entering a variable, list or user function name, and diving to a new subsheet.

## Error Messages

### Disk Data or I/O Error

- Cause:** Attempting an action that causes a hardware read/write error or attempting to use a worn or damaged diskette. (This condition should not occur.)
- Actions:** Trying another diskette; referring to your hardware operations manual for hardware diagnostic tests; noting this message and contacting your computer dealer.

### Disk Hardware Error

- Cause:** Attempting an action that causes a hardware read/write error. (This condition should not occur.)
- Actions:** Trying another diskette; referring to your hardware operations manual for hardware diagnostic tests; noting this message and contacting your computer dealer.

### Disk is Protected

- Cause:** Attempting a print or storage action to a protected diskette.
- Actions:** Specifying another filename or drive specifier, or inserting another diskette.

### Disk Related Software Error

- Cause:** Attempting an action that causes an operating system or program error.
- Actions:** Checking that there is no write-protect tab on the diskette to be written to, specifying another disk drive or filename, or trying another storage diskette; saving a copy of the model and a list of the actions that produced the error, noting the error message above, and contacting your dealer.

**Disk Space or Disk Directory Full**

- Cause:** Attempting a print or storage action to a diskette that has reached the limit for the number of files or amount of data permitted.
- Actions:** Inserting another diskette or specifying a different disk drive; deleting filenames with the Delete option of the Storage command (/SD) or at the operating system level. (A partial file may have been created on the original diskette and should be deleted with the Delete option (/SD) of the Storage command or on the operating system level.)

**(Edit) Built-in function already exists**

- Cause:** Attempting to enter a built-in function name in the Name field on the User Function Sheet. Built-in functions are case insensitive, so both uppercase and lowercase names are restricted.
- Actions:** Entering a non-reserved function name. (See the Built-in Functions section for the names of the built-in functions available in the TK!Solver program that cannot be used.)

**(Edit) Equal sign within parentheses**

- Cause:** Attempting to enter a rule that contains an equal sign within a set of parentheses or a left parenthesis not closed before the equal sign.
- Actions:** Editing the rule so that all sets of parentheses are closed on either side of the equal sign.

**(Edit) Expression too complex**

- Cause:** Attempting to enter a rule that contains too many repeated positive or negative signs or more than ten levels of parentheses.
- Actions:** Rewriting the rule more simply or dividing a rule into less complex rules, with extra variables if necessary.

## Error Messages

### **(Edit) Function already exists**

**Cause:** Attempting to enter a user function name in the Name field on the User Function Sheet that is the same as a previously entered user function name.

**Actions:** Entering a unique user function name.

### **(Edit) Invalid column width**

**Cause:** Attempting to enter an integer value that is invalid for the Width field on the Table Sheet.

**Actions:** Entering an integer entry or expression between 3 and 128 (or the number of character positions to be printed in each column, less than the printed page or screen width).

### **(Edit) Invalid comparison tolerance**

**Cause:** Attempting to enter a numeric value in the Comparison Tolerance field on the Global Sheet that multiplied by the typical value is equal to or greater than 1E120.

**Actions:** Entering a numeric value or expression that when multiplied by the typical value is less than 1E120. (See the Overview section for more information on valid numeric value entries.)

### **(Edit) Invalid first element**

**Cause:** Attempting to enter an integer value that is invalid for the First field on the Table Sheet.

**Actions:** Entering an integer entry or expression between 1 and the integer limit for your machine (or any number up to the number of elements in the longest list).

### **(Edit) Invalid form length**

**Cause:** Attempting to enter an integer value that is invalid for the Form Length field on the Global Sheet.

**Actions:** Entering an integer entry or expression between 6 and 1000 (or the number of lines per page on a printer page).

**(Edit) Invalid function name**

**Cause:** Attempting to enter a rule that contains a number before a left parenthesis where a function name is expected.

**Actions:** Editing the rule so that it contains an operator or a built-in or defined user function name before the parenthesis.

**(Edit) Invalid integer**

**Cause:** Attempting to enter a value or expression that does not evaluate to an integer in an integer entry field, or is outside machine limits.

**Actions:** Entering a valid integer entry or expression. (See the Overview and Hardware Reference sections for more information on valid integer entries.)

**(Edit) Invalid iteration count**

**Cause:** Attempting to enter an integer value that is invalid for the Iteration Count field on the Global Sheet.

**Actions:** Entering an integer entry or expression between 1 and 10000 (or the maximum number of iterations before the Iterative Solver stops).

**(Edit) Invalid left margin**

**Cause:** Attempting to enter an integer value that is invalid for the Left Margin field on the Global Sheet.

**Actions:** Entering an integer entry or expression that is between 0 and 200 (or the number of character positions to indent before printing the first character).

**(Edit) Invalid name**

**Cause:** Attempting to enter an invalid name in a Name field on the Variable Sheet, List Sheet, or User Function Sheet, or any place a list name can be assigned and created.

**Actions:** Entering a valid name entry. (See the Overview section for more information on valid name entries.)

## Error Messages

### **(Edit) Invalid numeric value**

**Cause:** Attempting to enter a symbolic value in an integer or numeric value field.

**Actions:** Entering a valid integer or numeric entry or expression. (See the Overview section for more information about valid integer or numeric entries.)

### **(Edit) Invalid operator**

**Cause:** Attempting to enter a rule that contains an invalid character where an operator is expected. Valid characters are numbers, letters, the operators, an equal sign, sets of parentheses, apostrophes, commas within function argument lists, and special characters allowed in name entries:  
@ # \$ % \_

**Actions:** Editing the rule so that it contains only valid characters. (See the Overview and Mathematics sections for more information on valid rules.)

### **(Edit) Invalid page length**

**Cause:** Attempting to enter an integer value that is invalid for the Printed Page Length field on the Global Sheet.

**Actions:** Entering an integer entry or expression that is between 6 and 1000 (or the number of lines available for printing on each output page).

### **(Edit) Invalid page width**

**Cause:** Attempting to enter an integer value that is invalid for the Printed Page Width field on the Global Sheet.

**Actions:** Entering an integer entry or expression that is between 6 and 200 (or the number of character positions available to the program for printing across an output page).

**(Edit) Invalid symbolic value**

**Cause:** Attempting to enter a rule that contains a number preceded by an apostrophe.

**Actions:** Editing the rule so that the apostrophe is omitted or the symbolic value begins with an apostrophe and a letter or @ # \$ % \_ (see the Mathematics section for more information about valid symbolic values).

**(Edit) Invalid typical value**

**Cause:** Attempting to enter a numeric value in the Typical Value field on the Global Sheet that multiplied by the comparison tolerance is equal to or greater than 1E120.

**Actions:** Entering a numeric value or expression in the Typical Value field on the Global Sheet that when multiplied by the comparison tolerance is less than 1E120. (See the Overview section for more information on valid numeric value entries.)

**(Edit) List already exists**

**Cause:** Attempting to enter a list name in a Name field on the List Sheet that is the same as a previously entered list name.

**Actions:** Entering a unique list name.

**(Edit) Misplaced comma**

**Cause:** Attempting to enter a rule that contains a comma between values or names that are not function arguments.

**Actions:** Editing the rule so that the only use of commas is to separate arguments of a function.

**(Edit) Missing equal sign**

**Cause:** Attempting to enter a rule that does not contain an equal sign.

**Actions:** Editing the rule so that it contains an equal sign.

## Error Messages

### (Edit) Missing function arguments

Cause: Attempting to enter a rule that contains a function with too few arguments.

Actions: Editing the rule so that it contains functions with the proper number of arguments: user functions take one argument; built-in functions take pre-determined numbers of arguments. (See the Mathematics and Built-in Functions sections for more information on valid function argument formats.)

### (Edit) Missing name

Cause: Attempting to enter a rule that contains an apostrophe followed by a character other than a letter or @ # \$ % \_

Actions: Editing the rule so that an apostrophe is followed by a valid name beginning with a letter or valid character or removing the apostrophe. (See the Overview section for more information on valid name entries.)

### (Edit) Missing operand

Cause: Attempting to enter a rule that contains a dangling operator or equal sign; or a constant value that is outside the numeric limits for the program.

Actions: Editing the rule so that plus and minus signs precede names, values, or parenthetical expressions; and each equal sign or other operator has a name, value, or valid expression on either side of it; assigning a constant value with an absolute value less than 1E120.

### (Edit) Missing operator

Cause: Attempting to enter a rule that contains two parenthetical expressions unrelated by an operator.

Actions: Editing the rule so that it contains an operator between the expressions within the two sets of parentheses.

**(Edit) Too many equal signs**

Cause: Attempting to enter a rule that contains more than one equal sign.

Actions: Editing the rule so that it contains only one equal sign or splitting the rule into two or more rules, each with one equal sign.

**(Edit) Too many function arguments**

Cause: Attempting to enter a rule that contains a function with too many arguments.

Actions: Editing the rule so that it contains functions with the proper number of arguments; user functions take one argument; built-in functions take predetermined numbers of arguments. (See the Mathematics and Built-in Functions sections for more information on valid function argument formats.)

**(Edit) Undefined function**

Cause: Attempting to enter a rule that contains a variable name or an undefined user function name before a left parenthesis.

Actions: Editing the rule so that it contains an operator before the parenthesis or defining a user function before entering it in a rule; checking the spelling of the intended built-in or user function name.

**(Edit) Unevaluated expression**

Cause: Attempting to enter an expression that contains a syntax error or a value outside the machine number limits in any value field.

Actions: Entering a value with an absolute value less than 1E120; entering a corrected expression; checking the expression by making it a rule on the Rule Sheet and looking at the Status field message for a more explicit error message.

## Error Messages

### (Edit) Unmatched left parenthesis

Cause: Attempting to enter a rule that contains an unmatched opening parenthesis.

Actions: Editing the rule so that it contains an equal number of right and left parentheses.

### (Edit) Unmatched right parenthesis

Cause: Attempting to enter a rule that contains an unmatched closing parenthesis.

Actions: Editing the rule so that it contains an equal number of right and left parentheses.

### (Edit) Variable already exists

Cause: Attempting to enter a variable name in the Name field on the Variable Sheet that is the same as a previously entered variable name.

Actions: Entering a unique variable name.

### End of File

Cause: Attempting to load an incomplete or improperly formed file with the Load file (/SL) or DIF Load (/S#L) option of the Storage command. (This condition should not occur.)

Actions: Creating a new copy of the file and repeating the load action.

### File Does Not Exist

Cause: Attempting to load a file that does not exist with the Load file (/SL) or DIF Load (/S#L) options of the Storage command.

Actions: Beginning the command again and entering a properly spelled filename; using the Directory Scrolling feature of the Storage command to locate the files that exist on the current diskette or inserting the diskette that contains the specified filename. (See the Storage command in the Command section for more information on Directory Scrolling.)

**File is protected**

- Cause:** Attempting a print or storage action to a filename that is protected by the operating system.
- Actions:** Specifying another filename or drive specifier, or inserting another diskette.

**Header too long**

- Cause:** Attempting to produce a table with the Action command (!) on the Table Sheet, with the Vertical or Horizontal field set to the Horizontal option, when the header specified in the Header field or the default list name header is too long to display any values on the same line. The total number of characters in the list name or Header field plus the value of the Width field must be less than the screen width or the value in the Printed Page Width field on the Global Sheet.
- Actions:** Entering a shorter header; setting the Vertical option; decreasing the value in the Width field; or increasing the integer in the Printed Page Width field on the Global Sheet (if the Screen or Printer field is set to the Printer option).

**Incomplete load**

- Cause:** Attempting to load a damaged file with the Load options (/SL and /S#L) of the Storage command; or pressing the Break key during loading.
- Actions:** Beginning the Storage command again and allowing the file to load completely; recreating and resaving the data in the damaged file. If the file is damaged, the load is completed up to the damaged section.

## Error Messages

### Incomplete save

- Cause:** Attempting a save action that causes a disk error; pressing the Break key during the use of the Save options of the Storage command (/S).
- Actions:** Beginning the Storage command again; specifying a different disk drive or filename; checking if the diskette is full or has a write-protect tab; or inserting a different diskette. A partial file may be created on the original diskette and should be deleted with the Delete option of the Storage command (/SD) or on the system level.

### Invalid DIF file

- Cause:** Attempting to load a non-DIF file with the DIF Load option of the Storage command (/S#L); attempting to load a file that does not conform to the DIF™ file format.
- Actions:** Beginning the Storage command again and using the Load option of the Storage command (/SL) instead of the DIF Load option (/S#L) if the file is a regular TK!Solver save file; resaving the file with the DIF Save option (/S#S) within the TK!Solver program, or with the program which produced the file, if the file is damaged; or correcting the program that produced the file to conform with the DIF format. (See the Appendix on the files used with the TK!Solver program for further information concerning DIF files.)

### Invalid element

- Cause:** Specifying no element number; attempting to enter a value or expression that evaluates to a negative integer or zero, in response to the prompt of the Put (/LP) or Get (/LG) options of the List command.
- Actions:** Beginning the List command again and entering a valid positive integer value or expression. (See the Overview section for more information on valid integer entries.)

### **Invalid File Specification**

- Cause:** Attempting to specify an invalid filename for one of the Storage command (/S) options.
- Actions:** Supplying a file specification valid for your operating system or computer. (See the Hardware Reference section or your computer manual for further information about valid file specifications.)

### **Invalid first element**

- Cause:** Attempting to enter a value or expression that evaluates to a negative integer or zero in response to the prompt for the first element number by the Block option of the List command (/LB).
- Actions:** Beginning the List command again and entering a valid positive integer value or expression. (See the Overview section for more information on valid integer entries.)

### **Invalid integer**

- Cause:** Attempting to enter a value or expression that does not evaluate to an integer, or an integer outside the valid integer limits on the computer being used, on the Prompt/Error line in response to the Block (/LB), Put (/LP), and Get (/LG) options of the List command.
- Actions:** Beginning the List command again and entering a valid integer entry or expression. (See the Overview and Hardware Reference sections for more information on valid integer entries.)

### Invalid last element

**Cause:** Designating no input list; attempting to enter a value or expression that evaluates to an integer smaller than the value specified for the first element number in response to the prompt for the last element number by the Block option of the List command (/LB).

**Actions:** Beginning the List command again and entering a valid positive integer value or expression greater than the value specified for the first element number. (See the Overview section for more information on valid integer entries.)

### Invalid storage file

**Cause:** Attempting to load a file with the Load option of the Storage command (/SL) that does not conform to the TK!Solver storage file format.

**Actions:** Beginning the Storage command again and selecting a valid storage file name; or recreating the data and saving it properly with one of the Save options of the Storage command (/S).

### Line Too Long

**Cause:** Attempting to load a file that contains a line that is too long (this condition could be caused by removing a line end character with an editor on the operating system level); attempting to load an improperly formatted file.

**Actions:** Pressing the Break key to cancel the load action; trying to recreate the file and reload it using the same machine.

### List Solver: No associated lists

**Cause:** Attempting to use the Solve option of the List command (/L!) when there are no lists associated with variables on the Variable Sheet.

**Actions:** Creating an associated list by typing the List option (L) in the Status field for at least one variable on the Variable Sheet and beginning the List command again.

**List Solver: No input list values**

**Cause:** Attempting to use the Solve option of the List command (/L!) when there is no list associated with an input variable or no associated list containing values.

**Actions:** Assigning at least one variable associated with a list as an input variable with the Input option (I) on the Status field on the Variable Sheet, or entering values on the associated List subsheet, and beginning the List command again; using the Block option of the List command (/LB) to generate output values.

**Missing domain list**

**Cause:** Attempting to enter a value in a domain element field on a User Function subsheet before a domain list name has been entered.

**Actions:** Entering a name in the Domain List field of the User Function subsheet before entering values.

**Missing range list**

**Cause:** Attempting to enter a value in a range element field on a User Function subsheet before a range list name has been entered.

**Actions:** Entering a name in the Range List field of the User Function subsheet before entering values.

**No ! action for this sheet**

**Cause:** Attempting to use the Action command (!) on the Unit, Global, List, or User Function Sheet, or a Variable subsheet.

**Actions:** Using the Solve command (/!) to solve the current model from the current sheet or the Action command (!) by moving the cursor to the appropriate sheet or subsheet: the Variable or Rule sheets to solve the current model; the List or User Function subsheets to fill in list values; the Table Sheet to produce a table; or the Plot Sheet to produce a plot.

## Error Messages

### No associated lists

- Cause:** Attempting to use the Put (/LP) or Get (/LG) option of the List command when there are no lists associated with variables on the Variable Sheet.
- Actions:** Creating an associated list by typing the List option (L) in the Status field for at least one variable on the Variable Sheet and beginning the List command again.

### No matching filenames

- Cause:** Attempting to use a complete file specification or a partial file specification containing a computer system wildcard in response to one of the options of the Storage command (/S) that does not match any filename on the default or specified diskette.
- Actions:** Using a wildcard correctly to indicate known filenames; including the appropriate drive specifier; using the Directory Scrolling feature of the Storage command without a filename to locate existing files. (See the Storage command in the Command section for more information on Directory Scrolling.)
- On file systems with file extensions, it may be necessary to specify an extension or provide a wildcard for the extension if it is different from the default file extension.

### No points to plot

- Cause:** Attempting to produce a plot with the Action command (!) on the Plot Sheet with no X or Y list specified, or no values in the X or Y axis lists.
- Actions:** Entering an X axis list name and at least one Y axis list name; entering at least one value in the X axis list and at least one value in at least one Y axis list before plotting a graph.

**No Such Device or Device Not Ready**

- Cause:** Attempting a print or storage action to a drive specifier that does not exist or a device name that is not connected.
- Actions:** Specifying a valid drive specifier or device name for a connected device and repeating the action. (See your operations manual for further information about device names and drive specifiers.)

**No topic <topic>; More help: Y N**

- Cause:** Attempting to locate a topic in the Help Facility that does not exist.
- Actions:** Typing **Y** to redisplay the Help topic prompt and entering another topic (such as “Topics” to provide a list of valid topic names); typing **N** or pressing the Break key to exit the Help Facility and check the TK!Solver manuals for detailed information about the program.

**No values to print**

- Cause:** Attempting to produce a table with the Action command (!) on the Table Sheet with no lists specified or no values in the specified lists.
- Actions:** Entering at least one list name in the List field and at least one value in at least one list and repeating the Action command.

**No variable name**

- Cause:** Attempting to enter a List (L) or Guess (G) option in the Status field on the Variable Sheet for an unnamed variable.
- Actions:** Entering a variable name in the Name field on the Variable Sheet before associating a list or setting a guess value.

## Error Messages

### Not on List Sheet

Cause: Attempting to use the DIF Save option of the Storage command (/S#S) without the cursor on a list row on the List Sheet.

Actions: Moving the cursor to any field on the row for the first list to be saved on the List Sheet and repeating the command.

### Options: I O L G B

Cause: Attempting to type an invalid option letter on the Status field on the Variable Sheet.

Actions: Entering one of the valid options:

- I Input variable assignment
- O Output variable assignment
- L List association
- G Guess variable assignment
- B Blank value

### Options: L C

Cause: Attempting to type an invalid option letter on the Line End (CR/LF or CR) field on the Global Sheet.

Actions: Entering one of the valid options:

- L Carriage Return/Line Feed line ending on output
- C Carriage Return line ending on output

### Options: S P

Cause: Attempting to type an invalid option letter on the Screen or Printer field on the Plot or Table Sheet.

Actions: Entering one of the valid options:

- S Screen display
- P Printer or file output

**Options: T S L**

Cause: Attempting to type an invalid option letter on the Mapping field on the User Function Sheet or a User Function subsheet.

Actions: Entering one of the valid options:

T Table Lookup  
S Step Interpolation  
L Linear Interpolation

**Options: V H**

Cause: Attempting to type an invalid option letter on the Vertical or Horizontal field on the Table Sheet.

Actions: Entering one of the valid options:

V Vertically displayed table  
H Horizontally displayed table

**Options: Y N**

Cause: Attempting to type an invalid option letter on the Display Scale ON field on the Plot Sheet; or the Variable Insert ON, Intermediate Redisplay ON, Automatic Iteration ON, Page Breaks ON, and Page Numbers ON fields on the Global Sheet.

Actions: Entering one of the valid options:

Y Yes  
N No

**Out of memory**

Cause: Attempting to perform an action that requires more than the remaining available memory or produces a program error.

Actions: Saving information with the Storage command (/S) and clearing some or all of a model with the Reset command (/R), or deleting list values with the Delete command (/D); exiting the program with the Quit command (/Q); noting the actions that produced the error and contacting your dealer.

## Error Messages

### Output field only

**Cause:** Attempting to type, edit, copy into, or blank fields used only by the program for output: Output fields on the Variable Sheet, Status fields on the Rule Sheet, Elements fields on the List Sheet, or the blank lines on the Global Sheet.

**Actions:** Moving the cursor to a field that accepts user input; blanking an Output field on the Variable Sheet with the Blank option (B) in the Status field on the same row, or moving the output value to the corresponding Input field with the Input (I) or Guess (G) option.

### System error: parser error

**Cause:** Attempting an action that causes an unknown program error. (This condition should not occur.)

**Actions:** Saving a copy of the model and a list of the actions that produced the error, noting the error message above, and contacting your dealer.

### System error: unknown message

**Cause:** Attempting an action that causes an unknown program error. (This condition should not occur.)

**Actions:** Saving a copy of the model and a list of the actions that produced the error, noting the error message above, and contacting your dealer.

### Unable to open printer

**Cause:** Attempting a print action to a device name or filename that cannot be opened.

**Actions:** Specifying a different device name or filename in the Printer Device or Filename field on the Global Sheet.

**Unblankable block**

- Cause:** Attempting to blank a block of rows in more than one column, or lines and rows in both the Header and Field areas of a sheet with the same Blank command (/B).
- Actions:** Using the Blank command (/B) for one column or one area at a time.

**Unblankable field**

- Cause:** Attempting to blank a required entry field by copying a blank field from another location with the Copy command (/C) or by backspacing over all characters with the Edit () or Edit Field (/E) commands.
- Actions:** Replacing the current value or option with a different value or option.

**Unblankable field in block**

- Cause:** Blanking a block of fields that contains one or more required entry or output only fields; attempting to use the Blank command on a Variable, List, or User Function subsheet still displayed on the screen for a variable, list, or user function that has been deleted.
- Actions:** This message is a warning only: blankable fields are blanked, and all others remain unchanged.

**Undefined list block**

- Cause:** Attempting to perform the list fill-in action of the Action command (!) on a List or User Function subsheet when the first or last value is not specified or contains a symbolic value.
- Actions:** Entering a numeric value or expression for the first element in the list and defining the end of the list to be filled in with another numeric value or expression, and repeating the Action command.

## Error Messages

### Undeletable block

- Cause: Attempting to use the Delete command (/D) to delete a block of lines and rows from the Field Area into the Header Area of a sheet.
- Actions: Using the Delete command to delete rows in a Field Area only.

### Unevaluated expression

- Cause: Attempting to enter an expression that contains a syntax error or a value outside the machine number limits on the Prompt/Error line in response to a prompt for an integer value by the List command (/L).
- Actions: Beginning the command again and entering a valid integer value or expression with an absolute value less than 1E120. (See the Overview and Hardware Reference sections for more information about valid integer entries.)

### Unknown Error

- Cause: Attempting an action that causes an unknown program error. (This condition should not occur.)
- Actions: Saving a copy of the model and a list of the actions that produced the error, noting the error message above, and contacting your dealer.

### UNKNOWN ERROR CLASS

- Cause: Attempting an action that causes an unknown program error. (This condition should not occur.)
- Actions: Saving a copy of the model and a list of the actions that produced the error, noting the error message above, and contacting your dealer.

### Variable Sheet: Status Column:

An error symbol (E) displays in the Status field on the Variable Sheet for each variable identified by the solvers as being involved in the termination of a solution. The explanation of an error symbol may be viewed on the Status line at the top of the screen by moving the cursor to the Status field in question.

The following explanatory error messages are provided:

› **Error**

**Cause:** The variable on the same row is probably involved in the error conditions that terminated the solution.

**Actions:** Moving the cursor to the error symbols in the Status column on the Rule Sheet for more explicit error diagnostics.

› **Overdefined**

**Cause:** The variable on the same row is probably inconsistent with at least one other variable value in at least one equation.

**Actions:** Blanking one or more values on the Variable Sheet for the overdefined variable, or editing any rules marked “› Inconsistent” in the Status field on the Rule Sheet that assign the variable with contradictory values.

**Rule Sheet: Status Column:**

An error symbol (›) displays in the Status field on the Rule Sheet for each rule identified by the solvers as being involved in the termination of a solution. The explanation of an error symbol may be viewed on the Status line at the top of the screen by moving the cursor to the Status field in question. (See the appendix on Status Messages for the status messages that appear in Rule Status fields.)

Errors marked in the Status column on the Rule Sheet can usually be corrected by editing the rule or correcting input values on the Variable Sheet.

Occasionally an error message is provided for a function that is called indirectly during solution.

For further information about valid arguments for TK!Solver built-in functions, see the Built-in Functions section.

The following explanatory error messages are provided:

› **ACOS: Argument error**

**Cause:** Supplying an argument value that is not in the domain of the function.

## Error Messages

### › ACOSH: Argument error

Cause: Supplying an argument value that is not in the domain of the function.

### › APPLY: Argument error

Cause: Supplying a first argument for the APPLY function that does not evaluate to a symbolic value; supplying a symbolic value or expression for a symbolic value that is not the name of a previously defined user function.

### › ASIN: Argument error

Cause: Supplying an argument value that is not in the domain of the function.

### › ASINH: Argument error

Cause: Supplying an argument value that is not in the domain of the function.

### › ATAN2: Argument error

Cause: Supplying a second argument value that is not possible with the specified function value in the first argument.

### › ATANH: Argument error

Cause: Supplying an argument value that is not in the domain of the function.

### › Argument error

Cause: Supplying an argument value to a function that causes an unknown error.

### › COSH: Argument error

Cause: Supplying an argument value that is not in the domain of the function.

**> COUNT: Argument error**

Cause: Supplying a single argument for the COUNT function that is not a symbolic value or an expression that evaluates to a symbolic value for a list name; supplying a value or expression for a list name that contains no values.

**> DOT: Argument error**

Cause: Supplying either argument to the DOT function that is not a symbolic value or an expression that evaluates to a symbolic value for a list name; one of the named lists contains no values.

**> Division by zero**

Cause: Evaluating a rule or function necessitating division by zero, which is undefined.

**> ELEMENT: Argument error**

Cause: Supplying a first argument for the ELEMENT function that is not a symbolic value or an expression that evaluates to a symbolic value for a list name; supplying a value or expression for a list name that contains no values; supplying a value or expression for the second argument that does not evaluate to an integer, or is not a valid element number and no third argument is specified.

**> EXP: Argument error**

Cause: Supplying a value or expression that is equal to or greater than 1E120 as an argument to the EXP function.

**> Exponent overflow**

Cause: Evaluating a rule or function that results in a value equal to or greater than 1E120.

## Error Messages

### › Inconsistent

Cause: Attempting a solution for a rule that involves inconsistent variable values.

### › Invalid symbolic computation

Cause: Attempting a solution for a rule or function in which a numeric calculation is attempted with a symbolic value.

### › Iteration: Dependency error

Cause: Attempting a solution for a rule that is dependent on the solution of another rule. Solution of one rule implies solution of the other.

### › Iteration: Out of memory

Cause: Attempting an Iterative solution that exhausts the remaining available memory.

### › Iteration: Too many guesses

Cause: Attempting an Iterative solution that involves too many guess variables.

### › LN: Argument error

Cause: Supplying a value or expression for a negative number or zero as an argument to the LN function.

### › LOG: Argument error

Cause: Supplying a value or expression for a negative number or zero as an argument to the LOG function.

### › MAX: Argument error

Cause: Supplying a single argument for the MAX function that is not a symbolic value or an expression that evaluates to a symbolic value for a list name; supplying a value or expression for a list name that contains no values.

**> MIN: Argument error**

Cause: Supplying a single argument for the MIN function that is not a symbolic value or an expression that evaluates to a symbolic value for a list name; supplying a value or expression for a list name that contains no values.

**> MOD: Argument error**

Cause: Supplying a value or expression as a second argument that evaluates to zero.

**> NPV: Argument error**

Cause: Supplying a first argument for the NPV function that does not evaluate numerically; supplying a single second argument that is not a symbolic value or an expression that evaluates to a symbolic value for a list name; supplying a value or expression for a list name that contains no values.

**> POLY: Argument error**

Cause: Supplying a first argument for the POLY function that does not evaluate numerically; supplying a single second argument that is not a symbolic value or an expression that evaluates to a symbolic value for a list name; supplying a value or expression for a list name that contains no values.

**> SINH: Argument error**

Cause: Supplying an argument value that is not in the domain of the function.

**> SQRT: Argument error**

Cause: Supplying a value or expression for a negative number as an argument.

## Error Messages

### › **SUM: Argument error**

Cause: Supplying a single argument that is not a symbolic value or an expression that evaluates to a symbolic value for a list name; supplying a value or expression for a list name that contains no values.

### › **TANH: Argument error**

Cause: Supplying an argument value that is not in the domain of the function.

### › **User function: Argument error**

Cause: Supplying a value or expression as an argument to a user function that is not in the domain of the function.

The following system errors may also appear in the Status column on the Rule Sheet by attempting an action that causes an unknown program error. (These conditions should not occur.) In the case of such an error, save a copy of the model and a list of the actions that produced the error, note which error message was displayed, and contact your dealer.

› **System error: evaluation error**

› **System error: stack overflow**

› **System error: stack underflow**

### **Bottom of Screen:**

The following messages appear on the bottom line of the screen when a hardware interface problem occurs for most implementations of the TK!Solver program:

#### **DISK ERROR - Press ENTER to try again.**

Cause: Attempting a print or storage action to: a disk drive that does not exist; an empty disk drive; or an unformatted, full, worn, write-protected, or damaged diskette.

Actions: Specifying a valid drive specifier, or inserting a usable diskette and pressing the Enter key; or pressing the Break key to return to the program and interrupt the disk action. Sometimes a more diagnostic message appears on the Prompt/Error line after pressing the Break key.

**Insert program disk, then press ENTER.**

Cause: Attempting an action that requires dynamic loading from a program diskette that has been removed from the disk drive.

Actions: Inserting the TK!Solver program diskette or a working diskette that contains the TK!Solver Overlay File and pressing the Enter key; or pressing the Break key to produce a confirmation message to cancel the program.

**Press BREAK again to cancel the program.**

Cause: Pressing the Break key in response to the message to insert a program diskette.

Actions: Pressing the Break key a second time to cancel the program, or any other key to repeat the message to insert a program diskette.

**Press ENTER when printer is ready.**

Cause: Attempting to output to a printer that is unavailable for some reason.

Actions: Verifying that a printing device is attached, on-line, not currently printing, supplied with paper, etc., and pressing the Enter key; or pressing the Break key to return to the program and interrupt the TK!Solver print action.

**Value Locations:**

The following message can appear anywhere program output values are displayed:

= **ERROR** =

Cause: Attempting to enter numbers in lists for a plot that generate display scales that are too large; attempting a calculation or solution with the program that results in a value that is outside the numeric limits for the program in an output or list value field.

## Error Messages

**Actions:** Verifying that the difference between the smallest and largest values in lists used in plots has an absolute value less than  $1E120$ . Generating a calculation or solution that has an absolute value less than  $1E120$ ; saving a copy of the model and a list of the actions that produced the error in a field, noting the error message above, and contacting your dealer.

### **Cleared Screen:**

The following error message appears at the top of a cleared screen:

**A NON-RECOVERABLE PROGRAM ERROR  
has occurred. Please write down  
the information listed below and  
refer to the Reference manual  
for further instructions.**

**Cause:** Attempting an action that causes an unknown program error. (This condition should not occur.)

**Actions:** Writing down the numbers listed below the message, noting the error message above and the actions that may have produced the error, saving the diskette, and contacting your dealer.





**Glossary**

The following terminology is used in the documentation for the TK!Solver program:

- Action** An activity performed by the program in response to a user command.
- Address** The specification representing a field position on a sheet. An address may be represented by a row number, a character representing the last row, a label letter, or a combination of these. An address is used with the Destination option of the Goto command to move the cursor to a new position.
- Argument** A variable or expression representing a value in the domain for a built-in or user function.
- Arithmetic Operator** The special symbol used to represent an arithmetic operation in a rule or expression. The operators in the TK!Solver program include:  
+ - / \* ^
- Arrow Keys** Keys used to move the cursor around a sheet in Command mode or to move the cue in the Editor.
- Associated List** A list associated with a particular variable.
- Automatic Iteration** The process of setting up guess values so that if the Iterative Solver is required for a solution it can be called automatically.
- Block** A number of rows or column of fields in the program. A block is designated by the user by pointing to or specifying its beginning and ending points.
- Bootling** Loading the diskette containing the computer's operating system.
- Built-in Function** A function provided by the TK!Solver program.
- Byte** A unit of measurement for computer memory. On most microcomputers, one byte equals one character. Memory is often approximated in K bytes (1 K byte = 1024 bytes).

## Calculation Unit

<b>Calculation Unit</b>	The unit of measurement in which a variable or list is first defined and calculations are performed.
<b>Calculator</b>	The TK!Solver feature that instantly evaluates expressions entered into value entry fields, if all variables and arguments to functions have defined values.
<b>Case Insensitive</b>	Interpreting uppercase and lowercase characters interchangeably. In the TK!Solver program, the names of built-in functions are case insensitive.
<b>Case Sensitive</b>	Distinguishing between uppercase and lowercase characters. In the TK!Solver program, all names except built-in functions are case sensitive.
<b>Character</b>	Any printable letter, digit, or special symbol produced by a single keystroke.
<b>Column</b>	A vertical list of fields under a label in the Field Area of a sheet.
<b>Command</b>	A keystroke or set of keystrokes that directs the program to perform some action.
<b>Command Mode</b>	The mode in which the TK!Solver program accepts commands.
<b>Comment</b>	A note users can enter in a field for their own reference or to display text. The TK!Solver program does not evaluate comments.
<b>Comparison Tolerance</b>	The number defined on the Global Sheet used in establishing equality. Generally, two numbers are considered equal if their difference divided by the larger of the two numbers is less than the comparison tolerance.
<b>Cue</b>	The marker that moves within text in an entry field or a user response on the Prompt/Error line before the Enter key is pressed.
<b>Current</b>	The present status of the program, cursor, cue, or screen display.
<b>Cursor</b>	The marker used to move around the Information Area and highlight a field.
<b>Data</b>	Information that can be entered into the program.
<b>Default</b>	The value supplied or action taken when no alternative has been specified.

<b>Destination</b>	The position to which an item is moved or copied.
<b>DIF File</b>	A file using the data interchange format standard developed by Software Arts, Inc. to permit exchange of data between programs.
<b>Direct Solver</b>	The solver that solves by substituting known values to determine unknown values.
<b>Directory Scrolling</b>	Successively displaying a selection of files from a diskette with an option of the Storage command so that one can be chosen for use.
<b>Disk Drive</b>	The hardware component where the diskette is inserted, allowing the computer to read the contents of the diskette.
<b>Diskette</b>	A plastic disk that stores computer programs magnetically.
<b>Display Size</b>	The portion of the current item displayed in a field or on the Status line, or the part of a user response that fits in the display on the Prompt/Error line.
<b>Display Unit</b>	The unit of measure in which the values for a variable or list are currently displayed.
<b>Diving</b>	Accessing a subsheet from a related sheet or subsheet by using the Dive command.
<b>Domain</b>	The permissible values for the argument or arguments of a function.
<b>Drive Specifier</b>	The name used to designate the appropriate disk drive in file specifications.
<b>Edit Indicator</b>	The word "Edit" in parentheses that appears on the Prompt/Error line when the user is using the Editor.
<b>Editing</b>	Changing an item or user response by calling the Editor on the current entry field or on the Prompt/Error line.
<b>Editor</b>	The facility used to edit items or user responses.
<b>Element</b>	The sequential number for a value in a list.
<b>Entering</b>	Putting an item in a field by pressing the Enter key or one of the Arrow keys, or completing a user response on the Prompt/Error line by pressing the Enter key.
<b>Entry</b>	The characters considered valid for a particular field.

## Entry Field

<b>Entry Field</b>	A field that accepts characters entered with the Enter key or an Arrow key. Entry fields accept Name, Unit Name, Comment, Value, Numeric Value, Integer, Rule, Filename, or Setup String entries.
<b>Entry Mode</b>	The mode in which new information is typed and entered into entry fields.
<b>Equation</b>	Two expressions related by an equal sign.
<b>Error Condition</b>	A mistake or inconsistency recognized by the program and indicated by an error signal and an error message on the Prompt/Error line, an error symbol in a Status field on the Variable and Rule Sheets, a question mark in a field for a unit error, or a message on the bottom of the screen for a hardware error.
<b>Error Message</b>	A message displayed by the program whenever an attempt is made to enter an invalid entry or option, or perform an invalid action.
<b>Evaluation</b>	Calculation of the value of an expression by the Calculator, or of variables in an equation by one of the solvers.
<b>Expression</b>	One or more characters that can be evaluated, containing numeric values, symbolic values, variable names, arithmetic operators, or function names in any valid combination.
<b>Extended Command</b>	A command that requires more than one keystroke, beginning with one of the following characters:  / = : ? !
<b>Extension</b>	Characters following a delimiter in a file specification, often indicating the type of file.
<b>Field</b>	A position on a sheet that contains items.
<b>Field Area</b>	The sheet area containing fields in rows under column labels in which specific information for a sheet can be entered.
<b>Field Type</b>	The quality of a field that determines what can be entered into it and the actions that can be performed on it. The TK!Solver program has three kinds of fields: entry, option, and output.

<b>File</b>	A collection of data on a diskette in which information or programs can be stored. A number of files can usually be stored on the same diskette.
<b>File Specification</b>	The full name of a file, dependent on the conventions for the computer system being used, including such parts as a disk specifier, filename, and file extension.
<b>Filename</b>	The name used to identify a particular file.
<b>Formatting</b>	The process performed on the operating system level that prepares diskettes to accept data for the first time.
<b>Function</b>	A relationship between two sets of values, called the domain and range. Each element in the domain is paired with a unique element in the range. The TK!Solver program uses both built-in and user-defined functions.
<b>Global Setting</b>	A pre-defined and changeable value in a field on the Global Sheet that affects automatic insertion of variables, the solvers, or use of a printer or output file.
<b>Guess</b>	An estimated value for a variable used with the Iterative Solver.
<b>Hardware</b>	The physical components of the computer system being used, which might include a video screen, memory unit, keyboard, disk drives, a printer, and diskettes.
<b>Header Area</b>	The sheet area containing general information about the contents or use of a sheet. It consists of a block of rows, each containing one label and one field.
<b>Help Facility</b>	The TK!Solver facility accessed by typing ? that describes specific topics related to use of the program.
<b>Information Area</b>	The part of the screen below the Message Area where sheets, subsheets, and the Help Facility are displayed.
<b>Initial Cursor Position</b>	The position of the cursor the first time a sheet is accessed.
<b>Input</b>	Data entered by the user or loaded from a file.

## Invalid

<b>Invalid</b>	An attempted command, action, user response, or entry that is not recognized by the program as correct for a particular location or at a particular time.
<b>Item</b>	The valid contents of a field.
<b>Item Size</b>	The complete contents of an item, regardless of what is actually displayed on the screen.
<b>Iteration</b>	A method of problem solution that uses successive approximations of the answer to converge to a desired mathematical result.
<b>Iterative Solver</b>	The solver that starts with a guess value for a variable, manipulates the guess to approximate the final value more closely, and continues this process until a solution is reached.
<b>Label</b>	A heading that describes a field or column of fields.
<b>Label Letter</b>	The letter used in field addresses to represent a label.
<b>Label Line</b>	A line of column labels near the top of all sheets (except the Global Sheet and the Variable subsheets) describing the contents of the fields below them.
<b>List</b>	A set of numbered values.
<b>List Solver</b>	The solver that calls the Direct or Iterative Solver for lists of values associated with one or more variables.
<b>Loading</b>	Copying data from a file on a diskette into the internal memory of the computer.
<b>Memory</b>	The computer component that stores and retrieves information.
<b>Memory Indicator</b>	A number in the righthand corner of the Message Area that indicates amount of memory remaining, expressed in thousands of bytes (K). When "LOW" is displayed, less than 1K remains; when "OUT" is displayed, there is only enough memory to save information, unless something is deleted.
<b>Message Area</b>	The top two lines of the screen used by the program to display status information, prompts, and error messages, and to accept user responses.
<b>Mode</b>	A program environment in which particular user actions are valid. The TK!Solver program has three modes: Command, Entry, and Option.

<b>Model</b>	All data entered into the TK!Solver program necessary to express a particular class of problem.
<b>Name</b>	The symbol used to refer to a variable, a list, or a user function in a name entry field, in an expression in a value field, or in a rule.
<b>Operating System</b>	The program or series of programs that use the hardware facilities of the computer to run programs and perform actions.
<b>Operating System Level</b>	The computer level at which the operating system works to perform such activities as copying diskettes and loading programs.
<b>Option</b>	A single character that is valid for the current option field.
<b>Option Field</b>	A field that accepts only one of a specific set of choices and does not require use of the Enter key.
<b>Option Mode</b>	The mode in which only particular options are valid. The user is automatically placed in Option mode on option fields.
<b>Output</b>	Information produced by a program.
<b>Output Field</b>	A field that displays information generated by the program.
<b>Plot</b>	A graph of list values defined on the Plot Sheet and displayed with the Action command.
<b>Pointer Indicator</b>	A number and letter (or letter only) at the end of certain prompts that displays the current cursor position during pointing.
<b>Pointing</b>	Moving the cursor to a final cursor position in response to a command prompt. The Blank, Copy, Delete, Move, Print, and DIF Save commands all use pointing.
<b>Position Indicator</b>	The display of the current cursor position in the upper lefthand corner of the screen.
<b>Precedence</b>	The order in which an expression or rule is evaluated, which can be altered by using parentheses.
<b>Print File</b>	A file that contains the output from a plot or table, or a copy of a sheet or portion of a sheet in the program.

## Program

<b>Program</b>	A set of sequential instructions that direct the computer to accomplish a specific task.
<b>Program Environment</b>	The context in which actions take place. The TK!Solver includes Command, Entry, and Option modes, the Editor, and the Help Facility.
<b>Prompt</b>	A message generated by the program requesting a user response.
<b>Prompt/Error Line</b>	The line that displays prompts and error messages. It is the second line in the Message Area.
<b>Random Access Memory</b>	Computer memory that allows data to be stored and retrieved.
<b>Range</b>	The possible function values.
<b>Row</b>	A horizontal line of fields describing a particular entity in a Field Area, or a label and a field in the Header Area.
<b>Rule</b>	A statement in mathematical notation about the relationships between variables. In the TK!Solver program, all valid rules are equations.
<b>Running</b>	Causing the computer to start following a set of instructions to perform an activity or execute a program.
<b>Satisfied</b>	The condition of a rule when it has been solved.
<b>Screen</b>	The video display portion of the computer.
<b>Scrolling</b>	Changing the view of a sheet or field. The view of a sheet is scrolled by moving the cursor past the top or bottom of the current display. The view of a field is scrolled by typing or moving the cue past the right or left edge of the current display field.
<b>Search Item</b>	A specification of the exact displayed contents of a field in order to move to it. A search item is preceded by a quotation mark (").
<b>Series</b>	Two or more expressions, separated by commas, used as arguments to a built-in function.
<b>Setup String</b>	A set of control codes and characters used to direct a printing device.
<b>Sheet</b>	A location for entering data and defining models. In the TK!Solver program, there are a variety of sheets containing different kinds of information.

<b>Sheet Letter</b>	A letter representing a sheet name that is used to access that sheet.
<b>Sheet Title Line</b>	The line at the top of a window that identifies the sheet being displayed.
<b>Slash Command</b>	An extended command beginning with a slash (/).
<b>Solution</b>	The process of solving a model.
<b>Solution Indicator</b>	An exclamation point at the extreme right of the Message Area that is visible when equations are not satisfied and disappears when solution of any equation is accomplished.
<b>Solver</b>	A TK!Solver feature for solving models. There are two solvers in the TK!Solver program: Direct and Iterative. The List Solver calls the Direct or Iterative Solver for list values associated with variables.
<b>Special Keys</b>	Computer keys that perform actions and do not display characters. They include the Arrow keys, the Backspace key, the Break key, and the Enter key.
<b>Startup</b>	The process of loading the TK!Solver program.
<b>Status</b>	The current state or condition of the program or a model. Program status is displayed in the Message Area at the top of the screen; solution status is displayed in the Status fields of the Rule and Variable sheets.
<b>Status Line</b>	The line at the top of the screen that displays cursor position, field contents, amount of remaining memory, and solution status of the model.
<b>Status Message</b>	A message that displays on the Prompt/Error line during an action that takes time to perform.
<b>Storage File</b>	A file that contains information that can be saved from the TK!Solver program or loaded into it. Storage files can be created by the user or purchased as part of a TK!Solver Pack™.
<b>Storage Unit</b>	The unit of measurement a list is stored in and calculations performed in.
<b>Subsheet</b>	A sheet that provides detailed information about a defined variable, list or user function. Accessed by diving from a sheet.

## Symbolic Value

<b>Symbolic Value</b>	A non-numeric value. In the TK!Solver program, a symbolic value is preceded by an apostrophe (?).
<b>Table</b>	A display of list values defined on the Table Sheet and produced with the Action command.
<b>Typical Value</b>	The value which, when multiplied by the comparison tolerance, determines at what point values will be considered near zero and the typical value will replace the comparison tolerance for determining equality.
<b>Typing</b>	Pressing keys to input options, entries, or commands.
<b>Unit</b>	The unit of measure the value of a variable or list is expressed in.
<b>Unit Conversion</b>	The process of converting values from one unit of measurement to another.
<b>Unsatisfied</b>	The condition of a rule when it has not been solved.
<b>User Function</b>	A function defined by the user on the User Function Sheet and subsheets that takes one argument.
<b>User Response</b>	Something typed by the user on the Prompt/Error line in response to a prompt requesting information and ending in a colon.
<b>Valid</b>	A command, user response, or item that is recognized by the program as correct for a particular location or at a particular time.
<b>Value</b>	The assigned or calculated quantity of a variable. In the TK!Solver program, values can be expressed as real or integer numbers, variables, symbolic values, or expressions.
<b>Variable</b>	An entity that can take on different values.
<b>View</b>	The part of a sheet or subsheet currently displayed on the screen.
<b>Wildcard</b>	A character specific to an operating system that can be substituted for one or more characters when specifying an unknown filename or a series of filenames.
<b>Window</b>	The portion of the screen through which sheets, subsheets, and the Help Facility are displayed. The TK!Solver screen can be divided into two windows at one time.





## Index

- ◀ (key) iii; III-3; VI
- ▶ (key) iii; III-3; VI
- ↑ (key) iii; III-3; VI
- ↓ (key) iii; III-3; VI
- ◆ (key) iii; III-4; VI
- ⊗ (key) iii; III-5; VI
- ◀ (key) iii; III-5; VI
  
- : (command) III-23
- = (command) III-42
- > (command) III-16
- < (command) III-41
- ; (command) III-51
- ☑ (command) iii; III-18; VI
- ? (command) I-15; III-26
- / (commands) I-12; III-2; A-7
- !/ (command) III-44; IV-7
- ! (command) III-6; IV-7
- ! (solution indicator) I-4, 23; A-17
  
- ABS function V-1
- absolute value function V-1
- ACOS function V-1
- ACOSH function V-1
- Action command III-6; IV-7
- accuracy I-8; IV-2, 24
- addition IV-3,6
- Add Offset field II-15
- address I-3, 7; III-23
- algebraic expressions IV-4
- All option, Reset command III-40
- Alphabetic Cross-reference v
- alphanumeric characters I-17
- APPLY function V-2
- arc cosine function V-1
- arc hyperbolic cosine function V-1
- arc sine function V-2
- arc hyperbolic sine function V-2
- arc tangent functions V-2,3
- arc hyperbolic tangent function V-2
- argument IV-4, 8; V-1
- argument errors A-45
- arithmetic operators IV-3
- Arrow keys iii; III-3; VI
- ASIN function V-2
- ASINH function V-2
- Associated List field II-9
- associated lists II-4, 9; IV-19
- ATAN function V-2
- ATANH function V-2
- ATAN2 function IV-7; V-3
- automatic item insertion I-16; II-17, 24, 36
- automatic iteration II-9, 17; IV-11, 15
- available memory I-3; VI; A-17, 41
  
- /b (command) III-10
- Backspace key iii; III-5; VI
- backup diskettes VI
- base e logarithm function V-4
- Blank command III-10
- blank diskettes VI
- Blank option, Variable Sheet II-5
- blanking fields III-10
- blanking values II-5
- Block option, List command III-32; IV-20
- boldface type iii
- booting the system VI
- Break key iii; III-5; VI
- built-in functions IV-14, 24; Part V
- bytes I-3; VI; A-17

## /c (command)

/c (command) III-12  
calculation I-8; IV-6  
calculation errors I-23  
Calculation units II-6, 10,  
13; IV-16  
calculator IV-6  
calling a sheet III-42  
cancelling an entry or  
action III-5  
caret character II-21; VI  
Carriage Return/Linefeed  
Option II-22; VI  
changing a sheet III-42, 51  
Character field II-41; III-11  
characters I-17; VI  
clearing sheets III-39  
column labels I-8  
columns I-8  
command errors I-22  
Command mode I-12  
command options I-4  
commands v; I-12; Part III  
comments I-18; II-7; IV-13  
common logarithm V-5  
comparison tolerance II-18,  
IV-8  
computation I-8; IV-6  
computer error signal I-5;  
VI  
control character II-21  
control codes I-20; II-21  
Control key VI  
conversion of units I-17;  
II-13  
Copy command III-12  
copying a field III-12  
COS function V-3  
COSH function V-3  
cosine functions V-1,3  
COUNT function IV-14; V-3  
count list function IV-14; V-3  
Cross-references v, viii  
cue I-10  
current field I-10  
cursor I-10

/d (command) III-14  
default settings II-16; VI  
Delete command III-14  
Delete File option, Storage  
command III-49  
deleted entities A-23  
deleting rows III-14  
destination III-23  
device names I-19; VI  
DIF clearinghouse A-3  
DIF format files IV-18; A-2  
DIF storage options III-47  
Direct Solver III-44; IV-7,19  
directory scrolling III-3, 45;  
VI  
disk drives VI  
disk errors A-22, 24  
disk operating system VI  
disk specifiers VI  
diskettes VI  
Display Scale ON field II-40  
display size I-3, 7, 8, 11;  
IV-2  
Display units I-18; II-6, 10;  
IV-16  
displaying sheets III-39, 52  
Dive command III-16  
division IV-3, 6  
division by zero IV-8  
domain II-30, 31; IV-22  
domain lists II-31; IV-19  
DOT function IV-14; V-3  
Down Arrow key iii; III-3;  
VI  
drive specifiers VI  
  
/e (command) I-14; III-21  
e (base of natural  
logarithm) V-4  
e (scientific notation) IV-1  
E function V-4  
Edit command I-14; III-18  
Edit Field command I-14;  
III-21  
Edit Indicator I-4

- editing a rule III-21; IV-14
- editing a user
  - response III-18
- editing an item I-9, 13;  
III-18, 21
- Editor I-14
- element numbers III-32;  
IV-20
- Element fields II-24, 28, 36
- ELEMENT function IV-14;  
V-4
- Enter key iii; III-4; VI
- entering data I-13
- entry errors I-22
- Entry fields I-16
- Entry mode I-13
- equal sign IV-5
- equality IV-8
- equation components IV-1
- equations I-19; II-11; IV-5
- erasing data III-10, 14, 38,  
39
- error conditions I-22
- error messages vii; I-5, 22;  
A-22
- error signal I-5; VI
- escape character I-20; II-21
- evaluation IV-6, 7, 19
- exclamation point I-4
- exiting the Editor I-14
- exiting the Help
  - Facility I-15; III-26
- exiting the program I-1;  
III-38
- EXP function V-4
- exponentiation IV-1, 3, 6
- expression evaluation IV-6
- expressions IV-4; V-1
- extended commands III-2
- extensions of filenames VI
  
- factors IV-1
- Field Area I-8
- Field Items I-3, 8
- Field Labels I-3, 8
- field types I-16
  
- fields I-7, 16
- file extensions VI
- file manipulation III-45; A-1
- filenames I-19; II-20; VI
- files VI; A-1
- Fill Action, Action
  - command III-7
- filling in a list III-7; IV-19
- final cursor position III-2
- first guesses II-9; IV-11
- Form Length field II-19
- formats for printing II-18
- formatting diskettes VI
- From field II-14
- function argument
  - errors A-45
- function arguments IV-4;  
V-1
- function inverses IV-8
- Function Save option, Storage
  - command III-47
- Functional
  - Cross-reference viii
- functions IV-4, 22
- functions, built-in IV-24;  
Part V
- functions,
  - user-defined II-30, 34;  
IV-22
  
- general program
  - commands ix
- Get option, List
  - command III-33; IV-21
- GIVEN function V-4
- global settings II-16; VI
- Global Sheet II-6; IV-15
- going to a field I-10; III-3,  
23
- going to a sheet III-42, 51,  
52
- going to a subsheet III-16
- Goto command III-23
- graphs II-38; III-8
- guess values II-5, 9; IV-10,  
11

## Hardware Reference

- Hardware Reference Part VI
- hardware requirements VI
- Header Area I-7
- Header Field I-7
- Help command III-26
- Help Facility I-15
- Help File VI
- hexadecimal codes II-21; VI
- Horizontal option, Table Sheet II-44
- hyperbolic cosine function V-3
- hyperbolic sine function V-7
- hyperbolic tangent function V-8
- /i (command) III-29
- inconsistency errors I-23; A-44
- indirect unit conversion II-13; IV-17
- indeterminable values IV-7
- Information Area I-5
- initial cursor position I-10
- initial guess II-9; IV-11
- input lists III-30; IV-19
- Input option, Variable Sheet II-4
- input values II-4, 6, 10, 28, 36
- inputting data I-13, 16
- Insert command III-29
- inserting a row III-29
- inserting variables I-16; II-6, 17
- Instruction Files VI
- insufficient memory I-3; VI; A-17
- INT function V-5
- integer entries I-19
- integer truncation function V-5
- intermediate redisplay II-17; IV-10
- internal representation of numbers I-16; IV-2
- interpolation II-31; IV-22
- invalid entries I-16, 22
- invalid program disk VI
- inverse of functions IV-8
- irregular interpolation IV-23
- italic typeface iii
- item length I-8
- items I-8
- iteration number II-18
- Iterative Solver III-6; IV-10
- k (bytes) I-3; VI
- keyboards iii
- keys, computer specific iii; VI
- /l (command) III-30
- label letters I-3; II-2
- labels I-7
- least value in list V-6
- Left Arrow key iii; III-3; VI
- Left Margin field II-20
- Line End character II-22; VI
- linear interpolation II-31; IV-22
- list association II-4, 9; IV-14
- list block solution III-32; IV-20
- list creation II-9, 23, 26; IV-19
- List command III-30; IV-19
- list count function V-3
- list element function V-4
- list field II-44
- List Fill-in III-7; IV-19
- list functions IV-14; V-3,4,5,6, 8
- list names II-23
- List option, Variable Sheet II-4
- List Sheet II-23
- list storage III-47
- List solver III-30; IV-11, 19
- list subsheets II-24

- list units IV-17, 19
- list value assignment II-28, 36; III-7
- literals I-18; IV-2; A-2
- LN function V-5
- loading a file III-48; VI
- LOG function V-5
- logarithm, base e V-5
- logarithm, natural V-5
- LOW (memory) I-3; A-17
- lowercase iii
- /m (command) III-34
- machine-specific iii; I-1, 8; VI
- making changes I-14
- manual conventions iii
- manual summary i
- mapping II-31; IV-22
- mathematical
  - functions IV-22; Part V
- mathematics Part IV
- max function IV-14; V-5
- maximum integer value I-19; VI
- maximum item length I-16
- maximum iteration count II-18; IV-11
- maximum numeric value IV-2
- maximum value function V-5
- memory I-3; VI; A-17, 38
- Memory Indicator I-3
- Message Area I-2
- MIN function IV-14; V-6
- minimum integer value I-18; VI
- minimum numeric value IV-2
- minimum value function V-6
- MOD function V-6
- model development IV-13, 14, 16
- model reset III-39
- model solution I-23; IV-7
- modes I-12
- modification commands viii
- modulus function V-6
- Move command III-34
- movement I-10
- movement commands viii
- moving a row III-34
- multiplication IV-3,6
- Multiply By field II-14
- name entries I-17
- natural logarithm function V-5
- negative sign IV-6
- net present value function V-6
- NPV function V-6
- numbers IV-1
- numeric value entries I-19
- OCR-B typeface iii
- One-Window option, Window Command III-53
- operating system VI
- operators IV-3
- option errors I-22
- Option field I-20
- option items I-8
- Option mode I-14
- OUT (memory) I-3; VI; A-17
- output device names II-20
- output errors I-23
- output fields I-23
- output items I-8
- output lists III-30; IV-19
- Output option, Variable Sheet II-4
- outputting to a file II-20; III-46; VI; A-1
- overdefined variables A-44
- Overlay File VI
- overview Part I
- overwriting a file III-46
- overwriting during loading III-48

## /p (command)

/p (command) III-36  
Page Breaks ON field II-18  
Page Numbers ON field II-19  
PI function V-6  
Plot Action, Action command III-8  
Plot Files A-1  
Plot Sheet II-38  
plotting II-38; III-8; IV-15  
Pointer Indicator I-4  
pointing to a field I-4; III-2  
POLY function V-7  
polynomial function V-7  
polynomials IV-11  
Position Indicator I-3  
positive sign IV-6  
precedence IV-6  
precision IV-2, 8  
Print command III-36  
Print Files II-18; A-1  
Printed Page Length field II-20  
Printed Page Width field II-20  
printers II-20; VI  
Printer Device Or Filename field II-20  
printer setup strings I-20; II-21; VI  
printing II-39, 43; III-8, 36  
program environments I-12  
program errors I-22; VI; A-22,52  
program overview Part I  
program startup I-1; VI  
prompts I-4; A-7  
Prompt/Error line I-4  
Put option, List command III-32  
putting values into lists III-7, 32; IV-20

/q (command) III-38  
question mark I-23; II-6  
Quit command III-38  
quitting the Help Facility I-15  
quitting the Editor I-14  
quitting the program III-38  
  
/r (command) III-39  
range II-30, 32; IV-22  
range lists II-36  
range of integers I-17, 19  
range of numbers IV-2  
refreshing the screen VI  
remaining memory I-3; VI; A-17  
Reset command III-39  
restoring an item I-13  
Return command III-41  
Right Arrow key iii; III-3; VI  
rounding off numbers IV-2  
rows I-8  
rule errors I-23  
Rule field II-12  
Rule Sheet II-11  
rules IV-5, 7  
running the program VI  
  
/s (command) III-45  
satisfied rules I-23  
save options, Storage command III-46  
saving files III-46  
scientific notation IV-1  
screen I-2; VI  
Screen Or Printer fields II-39, 43  
scrolling through a field I-11  
scrolling through a sheet I-10  
scrolling through filenames III-3, 45; VI  
Search option, Goto command III-24

- Select command III-42
- selecting a filename III-45
- selecting a sheet III-42
- series V-1
- setup string entries I-20; VI
- SGN function V-7
- Sheet option, Reset
  - command III-39
- Sheet Title line I-7
- sheets vii, x; I-5; Part II; IV-14
- sign function V-7
- significant digits IV-2
- signum function V-7
- SIN function V-7
- single-rule evaluation IV-7
- SINH function V-7
- size constraints VI
- slash commands A-7
- solution IV-7
- solution commands ix
- solution errors I-23; IV-7
- Solution Indicator I-4, 23
- Solve action, Action
  - command III-6
- Solve command III-44
- Solve option, List
  - command III-30; IV-19
- solvers III-6, 30; IV-7
- special keys iii; III-1; VI
- splitting windows III-52
- SQRT function V-8
- square root function V-8
- startup I-1
- Startup File VI
- status fields I-20; II-4, 9
- Status line I-1, 3, 5
- status messages I-5; A-17
- STEP function V-8
- step interpolation II-31; IV-23
- stopping an action or
  - command I-22; III-5
- stopping the program III-38
- Storage command III-45
- storage diskettes VI
- Storage Files iv; A-2
- Storage Unit field II-28
- Storage units II-24; IV-17
- strings VI
- subsheets vii, x; I-6; II-2
- subtraction IV-3, 6
- SUM function IV-14; V-8
- Switch command III-51
- symbolic values I-18; IV-2; V-1; A-2
- symbols VI
- system level VI
- system notes VI
- Table action, Action
  - command III-8
- Table Files A-1
- table lookup II-32; IV-22
- Table Sheet II-42; IV-15
- TAN function V-8
- TANH function V-8
- terminology Glossary
- Title fields II-39, 43
- TK!Solver files A-1
- TK!Solver functions IV-24; Part V
- TK!Solver Help File VI
- TK!Solver Overlay File VI
- TK!Solver program
  - diskettes VI
- TK!Solver screen I-2; VI
- TK!Solver Startup File VI
- TK!SolverPack™ VI
- To field II-14
- trigonometric
  - functions IV-24; Part V
- Two-window option, Window
  - command III-53
- typical value II-18; IV-9
- unary plus and minus
  - signs IV-6
- unevaluated expression
  - errors A-31, 44

## unit abbreviations

- unit abbreviations A-4
- unit conversions II-7, 13; IV-16
- unit errors I-22
- unit fields II-6, 22
- unit name conventions A-4
- unit name entries I-17
- Unit Save option, Storage command III-47
- Unit Sheet II-13
- units of measurement I-17; II-7, 13; IV-16
- unsatisfied rules I-23; II-11
- Up Arrow key iii; III-3; VI
- uppercase iii
- user function names I-17
- User Function Sheet II-30
- User Function subsheets II-35
- user functions II-30, 35; IV-22
- user responses I-4
- valid expressions IV-4
- valid filenames I-19; VI
- valid items I-16
- value entries I-18
- Variable Insert ON field II-17
- variable names I-17
- Variable Save option, Storage command III-47
- Variable Sheet II-3
- Variable subsheets II-8
- Variable option, Reset command III-39
- variables I-16; II-3; IV-3
- Vertical option, Table Sheet II-44
- Vertical Or Horizontal field II-44
- video monitors iv, VI
- /w (command) III-52
- wildcards III-45; VI
- Window command III-52
- windows I-5
- x (variable name) V-1
- X-axis field II-40
- Y-axis field II-40
- zero I-17; IV-2, 7, 9





## Software Arts Registration Card

When we receive this card, we will send you a complimentary copy of SATN®, the support journal for Software Arts programs. Please register immediately.

Name \_\_\_\_\_

Company \_\_\_\_\_

Occupation \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_

Zip \_\_\_\_\_ Country \_\_\_\_\_

Date of Purchase \_\_\_\_\_

Program Serial Number(s)  
(on diskette label) \_\_\_\_\_

Program Version Number  
(on program copyright screen) \_\_\_\_\_

How did you learn about the TK!Solver program?

- Advertisement     Another user  
 Computer Dealer     Other

\_\_\_\_\_



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

---

**BUSINESS REPLY MAIL**

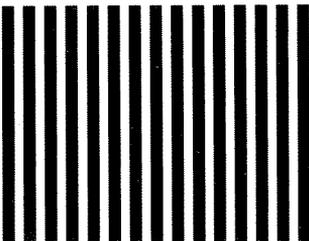
---

FIRST CLASS PERMIT NO. 1774 WELLESLEY, MA 02181

POSTAGE WILL BE PAID BY ADDRESSEE

***Software Arts, Inc.***

27 Mica Lane  
Wellesley, MA 02181





**RADIO SHACK A DIVISION OF TANDY CORPORATION**

**U.S.A.: FORT WORTH, TEXAS 76102  
CANADA: BARRIE, ONTARIO L4M 4W5**

---

**TANDY CORPORATION**

**AUSTRALIA**

**91 KURRAJONG ROAD  
MOUNT DRUITT, N.S.W. 2770**

**BELGIUM**

**PARC INDUSTRIEL DE NANINNE  
5140 NANINNE**

**U.K.**

**BILSTON ROAD WEDNESBURY  
WEST MIDLANDS WS10 7JN**

**PRINTED IN U.S.A.  
4/84**

P/N 200-008A