

# *Microsoft<sup>®</sup> Excel Functions and Macros*

---

*Complete Spreadsheet with Business Graphics and Database*

*Version 2.0*

*For IBM<sup>®</sup> Personal System/2<sup>™</sup>,  
IBM PC AT<sup>®</sup>, and Compatibles*

Microsoft Corporation

Information in this document is subject to change without notice and does not represent a commitment on the part of Microsoft Corporation. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. The purchaser may make one copy of the software for backup purposes. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose other than the purchaser's personal use without the written permission of Microsoft Corporation.

© Copyright Microsoft Corporation, 1987. All rights reserved.  
Simultaneously published in the U.S. and Canada.

Microsoft®, MS-DOS®, Multiplan®, and the Microsoft logo are registered trademarks of Microsoft Corporation.

Apple® is a registered trademark, and Macintosh™ is a trademark of Apple Computer, Inc.

Hewlett-Packard® and Laserjet +® are registered trademarks, and HP™ is a trademark of Hewlett-Packard Corporation.

IBM® and PC AT® are registered trademarks, and Personal System/2™ is a trademark, of International Business Machines Corporation.

Lotus® and 1-2-3® are registered trademarks of Lotus Development Corporation.



# C CONTENTS

<b>Chapter 1 Worksheet Function Basics</b>	<b>1</b>
What Is a Worksheet Function?	2
How to Use Functions	4
Error in Formula	7
Finding the Function You Need	9
Data Types	9
Types of Arguments	10
Translating Data Types	12
<b>Chapter 2 Worksheet Function Directory</b>	<b>15</b>
Conventions	16
Syntax	16
Commas	17
Argument Data Types	18
Functions by Subject Category	18
Database Functions	19
Date and Time Functions	20
Financial Functions	20
Information Functions	21
Logical Functions	22
Lookup Functions	23
Mathematical Functions	23
Matrix Functions	24
Statistical Functions	24
Text Functions	25
Trigonometric Functions	26
Directory	27
ABS(number)	27
ACOS(number)	27
AND(logical1,logical2,...)	28
AREAS(reference)	28
ASIN(number)	29
ATAN(number)	29
ATAN2(x_number,y_number)	30
AVERAGE(number1,number2,...)	30

CELL( <i>type_of_info,reference</i> ) . . . . .	31
CHAR( <i>number</i> ) . . . . .	34
CHOOSE( <i>index_number,value1,value2,...</i> ) . . . . .	34
CLEAN( <i>text</i> ) . . . . .	35
CODE( <i>text</i> ) . . . . .	36
COLUMN( <i>reference</i> ) . . . . .	36
COLUMNS( <i>array</i> ) . . . . .	37
COS( <i>radians</i> ) . . . . .	37
COUNT( <i>value1,value2,...</i> ) . . . . .	38
COUNTA( <i>value1,value2,...</i> ) . . . . .	39
DATE( <i>year,month,day</i> ) . . . . .	39
DATEVALUE( <i>date_text</i> ) . . . . .	41
DAVERAGE( <i>database,field,criteria</i> ) . . . . .	42
DAY( <i>serial_number</i> ) . . . . .	42
DCOUNT( <i>database,field,criteria</i> ) . . . . .	43
DCOUNTA( <i>database,field,criteria</i> ) . . . . .	43
DDB( <i>cost,salvage,life,period</i> ) . . . . .	43
Database Functions . . . . .	44
Dfunction( <i>database,field,criteria</i> ) . . . . .	44
DMAX( <i>database,field,criteria</i> ) . . . . .	48
DMIN( <i>database,field,criteria</i> ) . . . . .	48
DOLLAR( <i>number,decimals</i> ) . . . . .	48
DPRODUCT( <i>database,field,criteria</i> ) . . . . .	49
DSTDEV( <i>database,field,criteria</i> ) . . . . .	49
DSTDEVP( <i>database,field,criteria</i> ) . . . . .	49
DSUM( <i>database,field,criteria</i> ) . . . . .	49
DVAR( <i>database,field,criteria</i> ) . . . . .	50
DVARP( <i>database,field,criteria</i> ) . . . . .	50
EXACT( <i>text1,text2</i> ) . . . . .	50
EXP( <i>number</i> ) . . . . .	50
FACT( <i>number</i> ) . . . . .	51
FALSE() . . . . .	51
FIND( <i>find_text,within_text,start_at_num</i> ) . . . . .	52
FIXED( <i>number,decimals</i> ) . . . . .	52
FV( <i>rate,nper,pmt,pv,type</i> ) . . . . .	53
GROWTH( <i>known_y's,known_x's,new_x's</i> ) . . . . .	53
HLOOKUP( <i>lookup_value,table_array,row_index_num</i> ) . . . . .	55
HOURL( <i>serial_number</i> ) . . . . .	56
IF( <i>logical_test,value_if_true,value_if_false</i> ) . . . . .	57
INDEX( <i>ref,row_num,column_num,area_num</i> ) . . . . .	59
INDEX( <i>array,row_num,column_num</i> ) . . . . .	60
INDIRECT( <i>ref_text,type_of_ref</i> ) . . . . .	61
INT( <i>number</i> ) . . . . .	62
IPMT( <i>rate,per,nper,pv,fv,type</i> ) . . . . .	62

IRR(values,guess)	63
IS Functions	65
ISfunction(value)	65
LEFT(text,number_of_characters)	67
LEN(text)	67
LINEST(known_y's,known_x's)	67
LN(number)	72
LOG(number,base)	72
LOG10(number)	73
LOGEST(known_y's,known_x's)	73
LOOKUP(lookup_value,lookup_vector,result_vector)	75
LOOKUP(lookup_value,array)	76
LOWER(text)	77
MATCH(lookup_value,lookup_array,type_of_match)	78
MAX(number1,number2,...)	80
MDETERM(array)	80
MID(text,start_number,number_of_characters)	81
MIN(number1,number2,...)	82
MINUTE(serial_number)	82
MINVERSE(array)	83
MIRR(values,finance_rate,reinvest_rate)	84
MMULT(array1,array2)	86
MOD(number,divisor_number)	87
MONTH(serial_number)	88
N(value)	89
NA()	90
NOT(logical)	90
NOW()	91
NPER(rate,pmt,pv,fv,type)	92
NPV(rate,value1,value2,...)	92
OR(logical1,logical2,...)	94
PI()	95
PMT(rate,nper,pv,fv,type)	95
PPMT(rate,per,nper,pv,fv,type)	96
PRODUCT(number1,number2,...)	96
PROPER(text)	97
PV(rate,nper,pmt,fv,type)	97
type	98
rate	98
nper	99
pmt	99
pv	99
fv	99
RAND()	100

RATE(nper,pmt,pv,fv,type,guess)	102
REPLACE(old_text,start_num,num_chars,new_text)	102
REPT(text,number_times)	103
RIGHT(text,number_of_chars)	103
ROUND(number,number_of_digits)	104
ROW(reference)	104
ROWS(array)	105
SEARCH(find_text,within_text,start_at_num)	105
SECOND(serial_number)	106
SIGN(number)	107
SIN(radians)	108
SLN(cost,salvage,life)	108
SQRT(number)	109
STDEV(number1,number2,...)	109
STDEVP(number1,number2,...)	110
SUBSTITUTE(text,old_text,new_text,instance_number)	111
SUM(number1,number2,...)	112
SYD(cost,salvage,life,per)	112
T(value)	113
TAN(radians)	114
TEXT(value,format_text)	115
TIME(hour,minute,second)	115
TIMEVALUE(time_text)	116
TRANSPOSE(array)	116
TREND(known_y's,known_x's,new_x's)	117
TRIM(text)	120
TRUE()	120
TRUNC(number)	121
TYPE(value)	121
UPPER(text)	122
VALUE(text)	122
VAR(number1,number2,...)	123
VARP(number1,number2,...)	123
VLOOKUP(lookup_value,table_array,col_index)	124
WEEKDAY(serial_number)	125
YEAR(serial_number)	126

<b>Chapter 3 Macro Basics</b>	<b>129</b>
What Is a Macro?	130
When Are Macros Useful?	131
How Macros Work	132
Running a Command Macro	133
Running a Function Macro	135

Recording a Command Macro . . . . .	136
Absolute and Relative Recording . . . . .	137
Using Macro Sheets . . . . .	139
Editing Macro Sheets . . . . .	140
Organizing Macro Sheets . . . . .	140
Documenting Macro Sheets . . . . .	141
Formatting Macro Sheets . . . . .	142
Sample Macro Sheets . . . . .	142
Modifying a Recorded Command Macro . . . . .	144
Dialog Box Functions . . . . .	145
INPUT Function . . . . .	145
ALERT and MESSAGE Functions . . . . .	147
WAIT Function . . . . .	147
IF Function . . . . .	148
Value-returning Macro Functions . . . . .	149
Using Values from Documents . . . . .	150
Changing Macro Structure . . . . .	150
<b>Chapter 4 Writing Macros . . . . .</b>	<b>155</b>
Decide What Your Macros Will Do . . . . .	157
Plan How Your Macros Will Work . . . . .	158
Differences Between Command and Function Macros . . . . .	161
Function Macros . . . . .	161
Command Macros . . . . .	161
Creating Macros . . . . .	162
Opening Macro Sheets . . . . .	163
Writing or Recording Macros . . . . .	163
Naming Macros . . . . .	164
Debugging Macros . . . . .	165
Documenting Macros . . . . .	165
Macro Structure . . . . .	165
Macro Functions . . . . .	167
Using References . . . . .	168
Error Handling . . . . .	170
Creating Command Macros . . . . .	171
Recording a Command Macro . . . . .	171
About the Recorder Range . . . . .	172
Running Out of Room in the Recorder Range . . . . .	173
Absolute and Relative Recording . . . . .	174
What Actions Are Recorded? . . . . .	175
Stopping the Recorder . . . . .	175
Restarting the Recorder . . . . .	176
Cleaning Up a Recorded Macro . . . . .	176
Writing a Command Macro Without the Recorder . . . . .	176

Function Macros . . . . .	176
Order of Functions . . . . .	177
Using Arguments . . . . .	177
Using the Second Form of ARGUMENT . . . . .	178
Returning Results . . . . .	179
Types of Arguments and Results . . . . .	180
Function Macro Example . . . . .	181
Using the Function Macro . . . . .	182

## Chapter 5 Debugging and Testing

Macros . . . . .	183
Debugging Macros . . . . .	184
Stepping Through Macros . . . . .	185
Interrupting a Macro . . . . .	186
Stepping Through Sections . . . . .	186
Adding Return Functions . . . . .	187
Viewing Values . . . . .	187
Other Methods . . . . .	188
Testing Macros . . . . .	189
Use Test Data . . . . .	189
Check the Limits . . . . .	190
Anticipate Mistakes . . . . .	190

## Chapter 6 Advanced Macros . . . . . 191

Running Macros Automatically . . . . .	193
Using Autoexec Macros . . . . .	193
Running a Macro When Opening a Document . . . . .	194
Running a Macro When Closing a Document . . . . .	195
Making Demos . . . . .	195
Adding Onscreen Explanations to Your Demos . . . . .	195
Slowing Down Your Demos . . . . .	195
Speeding Up Your Demos . . . . .	196
Drawing the Viewer's Attention . . . . .	196
Creating Customized Menus and Dialog Boxes . . . . .	196
Custom Menus . . . . .	197
Creating New Menu Bars . . . . .	199
Deleting Menu Bars . . . . .	199
Adding Menus or Commands . . . . .	200
Deleting Menus or Commands . . . . .	202
Renaming Commands . . . . .	202
Adding or Removing Grey from Custom Commands . . . . .	203
Adding or Removing Checkmarks from Commands . . . . .	203
Switching Menu Bars . . . . .	204
Finding What Menu Bar Is Displayed . . . . .	204

Custom Dialog Boxes . . . . .	204
Item Column . . . . .	211
X and Y Columns . . . . .	211
Width and Height Columns . . . . .	211
Text Column . . . . .	212
Init/Result Column . . . . .	212
Additional Information on Items . . . . .	212
Limits . . . . .	217
Custom Dialog Box Example . . . . .	218
Using Custom Help . . . . .	219
Protecting Macros . . . . .	220
Protecting and Hiding Cells and Documents . . . . .	220
Using Customized Menus and Dialog Boxes . . . . .	220
Preventing a Macro from Interruption . . . . .	220
Text File Input and Output . . . . .	221
Using Macros to Start Other Applications . . . . .	221
Communicating with Other Windows Applications . . . . .	222
Applications That Don't Support DDE . . . . .	223
Speeding Up Your Macros . . . . .	223
<b>Chapter 7 Macro Function Directory . . . . .</b>	<b>225</b>
Types of Macro Functions . . . . .	226
Functions That Perform Actions . . . . .	227
Command Equivalent Functions . . . . .	227
Dialog Box Functions . . . . .	227
Other Action Equivalent Functions . . . . .	228
Customizing Functions . . . . .	228
Functions That Don't Perform Actions . . . . .	228
Control Functions . . . . .	229
Value-returning Functions . . . . .	229
Macro Function Arguments . . . . .	229
Macro Functions by Category . . . . .	230
Command Equivalent Functions . . . . .	230
Dialog Box Functions . . . . .	236
Other Action Equivalent Functions . . . . .	236
Customizing Functions . . . . .	238
Control Functions . . . . .	240
Value-returning Functions . . . . .	241
<b>Directory . . . . .</b>	<b>243</b>
A1.R1C1(r1c1) . . . . .	243
ABS(number) . . . . .	243
ABSREF(ref_text,ref) . . . . .	243
ACOS(number) . . . . .	243

ACTIVATE( <i>window_text,pane_num</i> )	243
ACTIVATE.NEXT()	244
ACTIVATE.PREV()	244
ACTIVE.CELL()	244
ADD.ARROW()	245
ADD.BAR()	245
ADD.COMMAND( <i>bar_num,menu_pos,menu_ref</i> )	245
ADD.MENU( <i>bar_num,menu_ref</i> )	246
ADD.OVERLAY()	246
ALERT( <i>message_text,type_num</i> )	247
ALIGNMENT( <i>type_number</i> )	248
ALIGNMENT?( <i>type_number</i> )	248
AND( <i>logical1,logical2,...</i> )	248
APP.ACTIVATE( <i>title_text,wait_log</i> )	249
APP.MAXIMIZE()	249
APP.MINIMIZE()	249
APP.MOVE( <i>x_num,y_num</i> )	249
APP.MOVE?( <i>x_num,y_num</i> )	249
APP.RESTORE()	249
APP.SIZE( <i>x_num,y_num</i> )	250
APP.SIZE?( <i>x_num,y_num</i> )	250
APPLY.NAMES( <i>name_array,ignore,use_rowcol,omit_col,omit_row, name_order,append</i> )	250
APPLY.NAMES?( <i>name_array,ignore,use_rowcol,omit_col,omit_row, name_order,append</i> )	250
AREAS( <i>reference</i> )	251
ARGUMENT( <i>name_text,data_type_num</i> )	251
ARGUMENT( <i>name_text,data_type_num,ref</i> )	252
ARRANGE.ALL()	252
ASIN( <i>number</i> )	252
ATAN( <i>number</i> )	252
ATAN2( <i>x_number,y_number</i> )	252
ATTACH.TEXT( <i>attach_to_num,series_num,point_num</i> )	252
ATTACH.TEXT?( <i>attach_to_num,series_num,point_num</i> )	253
AVERAGE( <i>number1,number2,...</i> )	253
AXES( <i>main_cat,main_value,over_cat,over_value</i> )	253
AXES?( <i>main_cat,main_value,over_cat,over_value</i> )	253
BEEP( <i>number</i> )	254
BORDER( <i>outline,left,right,top,bottom,shade</i> )	254
BORDER?( <i>outline,left,right,top,bottom,shade</i> )	254
BREAK()	254
CALCULATE.DOCUMENT()	254
CALCULATE.NOW()	254
CALCULATION( <i>type_num,iter,max_num,max_change,update, precision,date_1904</i> )	255



CALCULATION?( <i>type_num,iter,max_num,max_change,update,precision,date_1904</i> )	255
CALL( <i>call_text,argument1,...</i> )	255
CALLER()	256
CANCEL.COPY()	257
CANCEL.KEY( <i>enable,macro_ref</i> )	257
CELL( <i>type_of_info,reference</i> )	257
CELL.PROTECTION( <i>locked,hidden</i> )	257
CELL.PROTECTION?( <i>locked,hidden</i> )	257
CHANGE.LINK( <i>old_link,new_link</i> )	257
CHANGE.LINK?( <i>old_link,new_link</i> )	257
CHAR( <i>number</i> )	258
CHECK.COMMAND( <i>bar_num,menu_pos,command_pos,check</i> )	258
CHOOSE( <i>index_number,value1,value2,...</i> )	258
CLEAN( <i>text</i> )	258
CLEAR( <i>number</i> )	258
CLEAR?( <i>number</i> )	258
CLOSE( <i>save_logical</i> )	259
CLOSE.ALL	259
CODE( <i>text</i> )	259
COLUMN( <i>reference</i> )	259
COLUMN.WIDTH( <i>width_num,ref</i> )	260
COLUMN.WIDTH?( <i>width_num,ref</i> )	260
COLUMNS( <i>array</i> )	260
COMBINATION( <i>number</i> )	261
COMBINATION?( <i>number</i> )	261
COPY()	261
COPY.CHART( <i>number</i> )	261
COPY.CHART?( <i>number</i> )	261
COPY.PICTURE( <i>appearance,size</i> )	261
COS( <i>radians</i> )	262
COUNT( <i>value1,value2,...</i> )	262
COUNTA( <i>value1,value2,...</i> )	262
CREATE.NAMES( <i>top,left,bottom,right</i> )	263
CREATE.NAMES?( <i>top,left,bottom,right</i> )	263
CUT()	263
DATA.DELETE()	263
DATA.DELETE?()	263
DATA.FIND( <i>logical</i> )	263
DATA.FIND.NEXT()	264
DATA.FIND.PREV()	264
DATA.FORM()	264
DATA.SERIES( <i>row_col,type,date,step,stop</i> )	264
DATA.SERIES?( <i>row_col,type,date,step,stop</i> )	264

DATE(year,month,day)	265
DATEVALUE(date_text)	265
DAVERAGE(database,field,criteria)	265
DAY(serial_number)	265
DCOUNT(database,field,criteria)	265
DCOUNTA(database,field,criteria)	266
DDB(cost,salvage,life,period)	266
DEFINE.NAME(name_text,refers_to,macro_type,shortcut_text)	266
DEFINE.NAME?(name_text,refers_to,macro_type,shortcut_text)	266
DELETE.ARROW()	268
DELETE.BAR(bar_num)	268
DELETE.COMMAND(bar_num,menu_pos,command_pos)	268
DELETE.FORMAT(format_text)	268
DELETE.MENU(bar_num,menu_pos)	269
DELETE.NAME(name_text)	269
DELETE.OVERLAY()	269
DEREF(reference)	269
DIALOG.BOX(dialog_ref)	270
DIRECTORY(path_text)	270
DISABLE.INPUT(logical)	271
DISPLAY(formula,gridline,heading,zero,color)	271
DISPLAY(cell,formula,value,format,protect,names,precedents, dependents,note)	271
DMAX(database,field,criteria)	272
DMIN(database,field,criteria)	272
DOCUMENTS()	272
DOLLAR(number,decimals)	272
DPRODUCT(database,field,criteria)	273
DSTDEV(database,field,criteria)	273
DSTDEVP(database,field,criteria)	273
DSUM(database,field,criteria)	273
DVAR(database,field,criteria)	273
DVARP(database,field,criteria)	273
ECHO(logical)	273
EDIT.DELETE(num)	274
EDIT.DELETE?(num)	274
ENABLE.COMMAND(bar_num,menu_pos,command_pos,enable)	274
ERROR(enable,macro_ref)	274
EXACT(text1,text2)	275
EXEC(program_text>window_number)	275
EXECUTE(channel_num,execute_text)	276
EXP(number)	277
EXTRACT(unique_log)	277
EXTRACT?(unique_log)	277

FACT(number)	277
FALSE()	277
FCLOSE(file_number)	277
FILE.CLOSE()	277
FILE.DELETE(name_text)	278
FILE.DELETE?(name_text)	278
FILES(directory_text)	278
FILL.DOWN()	278
FILL.LEFT()	278
FILL.RIGHT()	279
FILL.UP()	279
FIND(find_text,within_text,start_at_num)	279
FIXED(number,decimals)	279
FOPEN(file_text,access_number)	279
FOR(counter_name,start_num,end_num,step_num)	280
FORMAT.FONT(name_text,size_num,bold,italic,underline,strike)	280
FORMAT.FONT?(name_text,size_num,bold,italic,underline,strike)	280
FORMAT.FONT(color,backgd,apply,name_text,size,bold,italic,underline,strike)	280
FORMAT.FONT?(color,backgd,apply,name_text,size,bold,italic,underline,strike)	281
FORMAT.LEGEND(position_num)	282
FORMAT.MOVE(x_pos,y_pos)	282
FORMAT.MOVE?(x_pos,y_pos)	282
FORMAT.NUMBER(format_text)	283
FORMAT.NUMBER?(format_text)	283
FORMAT.SIZE(width,height)	283
FORMAT.SIZE?(width,height)	283
FORMAT.TEXT(x_align,y_align,vert_text,auto_text,auto_size, show_key,show_value)	283
FORMULA(formula_text,ref)	284
FORMULA.ARRAY(formula_text,ref)	286
FORMULA.FILL(formula_text,ref)	286
FORMULA.FIND(text,in_num,at_num,by_num)	286
FORMULA.FIND?(text,in_num,at_num,by_num)	286
FORMULA.FIND.NEXT()	287
FORMULA.FIND.PREV()	287
FORMULA.GOTO(reference)	287
FORMULA.GOTO?(reference)	287
FORMULA.REPLACE(find_text,replace_text,look_at,look_by, current_cell)	288
FORMULA.REPLACE?(find_text,replace_text,look_at,look_by, current_cell)	288
FPOS(file_number,position_number)	289
FREAD(file_number,num_chars)	289
FREADLN(file_number)	289
FREEZE.PANES(logical)	290
FSIZE(file_number)	290

FULL(logical)	290
FV(rate,nper,pmt,pv,type)	290
FWRITE(file_number,text)	290
FWRITELN(file_number,text)	291
GALLERY.AREA(number,delete_overlay)	291
GALLERY.AREA?(number,delete_overlay)	291
GALLERY.BAR(number,delete_overlay)	291
GALLERY.BAR?(number,delete_overlay)	291
GALLERY.COLUMN(number,delete_overlay)	292
GALLERY.COLUMN?(number,delete_overlay)	292
GALLERY.LINE(number,delete_overlay)	292
GALLERY.LINE?(number,delete_overlay)	292
GALLERY.PIE(number,delete_overlay)	292
GALLERY.PIE?(number,delete_overlay)	292
GALLERY.SCATTER(number,delete_overlay)	292
GALLERY.SCATTER?(number,delete_overlay)	292
GET.BAR()	293
GET.CELL(type_of_info,reference)	293
GET.CHART.ITEM(x_y_index,point_index,item_text)	294
GET.DEF(def_text,document)	296
GET.DOCUMENT(type_of_info,name_text)	296
GET.FORMULA(reference)	299
GET.NAME(name_text)	299
GET.NOTE(cell_ref,start_char,count_char)	300
GET.WINDOW(type_of_info,name_text)	300
GET.WORKSPACE(type_of_info)	302
GOTO(reference)	303
GRIDLINES(cat_major,cat_minor,value_major,value_minor)	304
GRIDLINES?(cat_major,cat_minor,value_major,value_minor)	304
GROWTH(known_y's,known_x's,new_x's)	304
HALT()	304
HELP(topic_number)	304
HIDE()	305
HLINE(number_cols)	305
HLOOKUP(lookup_value,table_array,row_index_num)	305
HOURL(serial_number)	305
HPAGE(number_windows)	305
HSCROLL(scroll,col_log)	306
IF(logical_test,value_if_true,value_if_false)	306
INDEX(ref,row_num,column_num,area_num)	306
INDEX(array,row_num,column_num)	306
INDIRECT(ref,type_of_ref)	307
INITIATE(app_text,topic_text)	307
INPUT(prompt,type,title,default,x_pos,y_pos)	307

INSERT(shift_num)	310
INSERT?(shift_num)	310
INT(number)	310
IPMT(rate,per,nper,pv,fv,type)	310
IRR(values,guess)	310
ISBLANK(value)	310
ISERR(value)	311
ISERROR(value)	311
ISLOGICAL(value)	311
ISNA(value)	311
ISNONTEXT(value)	311
ISNUMBER(value)	311
ISREF(value)	311
ISTEXT(value)	311
JUSTIFY()	312
LEFT(text,number_of_characters)	312
LEGEND(logical)	312
LEN(text)	312
LINEST(known_y's,known_x's)	312
LINKS(doc_text)	312
LIST.NAMES	313
LN(number)	313
LOG(number,base)	313
LOG10(number)	313
LOGEST(known_y's,known_x's)	313
LOOKUP(lookup_value,lookup_vector,result_vector)	313
LOOKUP(lookup_value,array)	313
LOWER(text)	313
MAIN.CHART(type,stack,100,vary,overlap,drop, hilo,overlap%,cluster,angle)	313
MAIN.CHART.TYPE(type)	315
MATCH(lookup_value,lookup_array,type_of_match)	315
MAX(number1,number2,...)	315
MDETERM(array)	315
MESSAGE(logical,text)	315
MID(text,start_number,number_of_characters)	316
MIN(number1,number2,...)	317
MINUTE(serial_number)	317
MINVERSE(array)	317
MIRR(values,finance_rate,reinvest_rate)	317
MMULT(array1,array2)	317
MOD(number,divisor_number)	317
MONTH(serial_number)	317
MOVE(x_pos,y_pos>window_text)	317
N(value)	318

NA()	318
NAMES( <i>doc_text</i> )	318
NEW( <i>type_number</i> )	318
NEW?( <i>type_number</i> )	318
NEW.WINDOW()	319
NEXT()	319
NOT(logical)	319
NOTE( <i>add_text,cell_ref,start_char,count_char</i> )	319
NOW()	319
NPER( <i>rate,pmt,pv,fv,type</i> )	319
NPV( <i>rate,value1,value2,...</i> )	320
OFFSET( <i>ref,rows,cols,height,width</i> )	320
ON.DATA( <i>document_text,macro_text</i> )	320
ON.KEY( <i>key_text,macro_text</i> )	321
ON.TIME( <i>time,macro_text,tolerance,insert_log</i> )	321
ON.WINDOW( <i>window_text,macro_text</i> )	322
OPEN( <i>file_text,update_ext,read_only_rem</i> )	322
OPEN?( <i>file_text,update_links,read_only</i> )	322
OPEN.LINKS( <i>doc_text1,doc_text2,...,read_only_log</i> )	323
OPEN.LINKS?( <i>doc_text1,doc_text2,...,read_only_log</i> )	323
OR(logical1,logical2,...)	324
OVERLAY( <i>type,stack,100,vary,overlap,drop,</i> <i>hilo,overlap%,cluster,angle,series,auto</i> )	324
OVERLAY.CHART.TYPE( <i>type</i> )	325
PAGE.SETUP( <i>head,foot,left,right,top,bot,heading,grid</i> )	325
PAGE.SETUP?( <i>head,foot,left,right,top,bot,heading,grid</i> )	326
PAGE.SETUP( <i>head,foot,left,right,top,bot,size</i> )	326
PAGE.SETUP?( <i>head,foot,left,right,top,bot,size</i> )	326
PARSE( <i>parse_text</i> )	327
PASTE()	327
PASTE.LINK()	327
PASTE.SPECIAL( <i>paste_what,operation,skip_blanks,transpose</i> )	327
PASTE.SPECIAL?( <i>paste_what,operation,skip_blanks,transpose</i> )	327
PASTE.SPECIAL( <i>row_col,series,categories,apply</i> )	328
PASTE.SPECIAL?( <i>row_col,series,categories,apply</i> )	328
PASTE.SPECIAL( <i>paste_what</i> )	329
PASTE.SPECIAL?( <i>paste_what</i> )	329
PATTERNS( <i>b_auto,b_style,b_color,b_wt,shadow,</i> <i>a_auto,a_pattern,a_fore,a_back,APPLY</i> )	330
PATTERNS( <i>LINE,t_major,t_minor,t_label</i> )	330
PATTERNS( <i>LINE</i> )	330
PATTERNS( <i>LINE,m_auto,m_style,m_fore,m_back,APPLY</i> )	330
PATTERNS( <i>LINE,h_width,h_length,h_type</i> )	330
PI()	334

PMT(rate,nper,pv,fv,type)	334
POKE(channel_num,item_text,data_ref)	334
PPMT(rate,per,nper,pv,fv,type)	334
PRECISION(logical)	335
PREFERRED()	335
PRINT(range,from,to,copies,draft,preview,parts)	335
PRINT?(range,from,to,copies,draft,preview,parts)	335
PRINTER.SETUP(printer_text)	336
PRINTER.SETUP?(printer_text)	336
PRODUCT(number1,number2,...)	336
PROPER(text)	336
PROTECT.DOCUMENT(contents,windows)	336
PROTECT.DOCUMENT?(contents,windows)	336
PV(rate,nper,pmt,fv,type)	337
QUIT()	337
RAND()	337
RATE(nper,pmt,pv,fv,type,guess)	337
REFTEXT(ref,a1)	337
REGISTER(module_text,procedure_text,argument_text)	337
RELREF(ref,rel_to_ref)	340
REMOVE.PAGE.BREAK()	340
RENAME.COMMAND(bar_num,menu_pos,command_pos,name_text)	340
REPLACE(old_text,start_num,num_chars,new_text)	341
REPLACE.FONT(font,name_text,size_num,bold,italic,underline,strike)	341
REPT(text,number_times)	341
REQUEST(channel_num,item_text)	341
RESTART(reference)	342
RESULT(type_number)	343
RETURN(value)	343
RIGHT(text,number_of_chars)	344
ROUND(number,number_of_digits)	344
ROW(reference)	344
ROW.HEIGHT(height_num,ref,standard_height)	344
ROW.HEIGHT?(height_num,ref,standard_height)	344
ROWS(array)	345
RUN(reference)	345
RUN?(reference)	345
SAVE()	345
SAVE.AS(name_text,type_num,passwd_text,backup)	345
SAVE.AS?(name_text,type_num,passwd_text,backup)	345
SAVE.WORKSPACE(name_text)	346
SAVE.WORKSPACE?(name_text)	346
SCALE(cross,cat_labels,cat_marks,between,max,reverse)	347
SCALE(min,max,major,minor,cross,logarithmic,reverse,max)	347

SEARCH(find_text,within_text,start_at_num)	348
SECOND(serial_number)	348
SELECT(selection,active_cell)	348
SELECT(item_text)	349
SELECT.CHART()	351
SELECT.END(direction_num)	351
SELECT.LAST.CELL()	351
SELECT.PLOT.AREA()	352
SELECT.SPECIAL(type_number,value_types,levels)	352
SELECTION()	353
SEND.KEYS(key_text,wait_log)	353
SET.CRITERIA()	354
SET.DATABASE()	354
SET.NAME(name_text,value)	354
SET.PAGE.BREAK()	355
SET.PREFERRED()	355
SET.PRINT.AREA()	355
SET.PRINT.TITLES()	356
SET.VALUE(ref,values)	356
SHORT.MENUS(logical)	356
SHOW.ACTIVE.CELL()	357
SHOW.BAR(bar_num)	357
SHOW.CLIPBOARD()	357
SHOW.INFO(enable_log)	358
SIGN(number)	358
SIN(radians)	358
SIZE(width,height>window_text)	358
SLN(cost,salvage,life)	358
SORT(sort_by,key1,order1,key2,order2,key3,order3)	358
SORT?(sort_by,key1,order1,key2,order2,key3,order3)	358
SPLIT(col_split,row_split)	360
SQRT(number)	360
STDEV(number1,number2,...)	360
STDEVP(number1,number2,...)	361
STEP()	361
STYLE(bold,italic)	362
STYLE?(bold,italic)	362
Subroutines: ref(arg1,arg2,...)	363
SUBSTITUTE(text,old_text,new_text,instance_number)	363
SUM(number1,number2,...)	363
SYD(cost,salvage,life,per)	363
T(value)	364
TABLE(row_ref,column_ref)	364
TABLE?(row_ref,column_ref)	364



TAN(radians)	365
TERMINATE(channel_num)	365
TEXT(value,format_text)	365
TEXTREF(text,a1)	365
TIME(hour,minute,second)	365
TIMEVALUE(time_text)	366
TRANSPOSE(array)	366
TREND(known_y's,known_x's,new_x's)	366
TRIM(text)	366
TRUE()	366
TRUNC(number)	366
TYPE(value)	366
UNDO()	366
UNHIDE(window_text)	367
UNLOCKED.NEXT()	367
UNLOCKED.PREV()	367
UPPER(text)	367
VALUE(text)	367
VAR(number1,number2,...)	367
VARP(number1,number2,...)	367
VLINE(number_rows)	367
VLOOKUP(lookup_value,table_array,col_index)	368
VPAGE(number_windows)	368
VSCROLL(scroll,row_log)	368
WAIT(serial_number)	369
WEEKDAY(serial_number)	369
WHILE(logical_test)	369
WINDOWS()	369
WORKSPACE(fixed,decimals,r1c1,scroll,formula,status,menu,remote)	370
WORKSPACE?(fixed,decimals,r1c1,scroll,formula,status,menu,remote)	370
YEAR(serial_number)	371

## Appendix . . . . . 373

Specifying Special Characters	373
Combining Keys with SHIFT, CONTROL, ALT	375
Repeating Key Sequences	375



# Functions



# Functions





# Chapter 1

## Worksheet Function Basics

What Is a Worksheet Function?	2
How to Use Functions	4
Error in Formula	7
Finding the Function You Need	9
Data Types	9
Types of Arguments	10
Translating Data Types	12

## What Is a Worksheet Function?

A worksheet function takes a value or values, performs some operation on them, and returns a value or values. The values that you give to a function are called **arguments** to the function. The values that the function returns are called the **results** of the function.

For example, SUM is a worksheet function that performs addition. If you type `SUM(5,15)` in a formula, the SUM function takes the values 5 and 15, adds them together, and returns a value of 20. The numbers 5 and 15 are arguments to the SUM function, and 20 is the result of the SUM function.

There are also many worksheet functions that perform complicated calculations. The MIRR function, for example, calculates the modified internal rate of return for a series of periodic cash flows, using the following formula:

$$\left[ \frac{-\text{NPV}(\text{rrate}, \text{values}[\text{positive}]) * (1 + \text{rrate})^n}{\text{NPV}(\text{frate}, \text{values}[\text{negative}]) * (1 + \text{frate})^{\frac{1}{n-1}}} \right] - 1$$

It's easier to use the MIRR function than the formula above.

Even a function as simple as the SUM function can be very useful because you can specify arguments in many ways. The SUM function, for example, requires numbers as arguments. If you use something other than a number as an argument, Microsoft Excel tries to interpret that argument as a number. Suppose your worksheet looks like this:

	A	B	C	D	E	F	G	H
1	5		2					
2	15		2					
3			2					
4			2					
5			2					
6			2					
7								

If you type the formula `=SUM(A1:A2)` in a cell, Microsoft Excel looks in cells A1 and A2 and uses the numbers in those cells as the arguments to the SUM function.

If you want to add the first 6 values in column C, you can use either of the following formulas:

$$= 2 + 2 + 2 + 2 + 2 + 2$$

$$= \text{SUM}(C1:C6)$$



The second formula is easier to enter and to update. If you change a number in the range C1:C6, you have to edit the first formula, while the second formula is still correct.

The different kinds of values that you can use as arguments are called **data types**. Numbers, like 3 or 150, are one data type. Text, like “Here’s a sentence,” is another data type (text values used in formulas are always enclosed within double quotation marks). For information on data types, and how Microsoft Excel can translate from one data type to another, see “Data Types,” later in this chapter.

### Tip

For an introduction to worksheet functions, as well as practice using them, try the lesson “Using Worksheet Functions” in the Microsoft Excel Tutorial.

## Updating References

If you use cell references as arguments to functions, Microsoft Excel can often update those references for you when you edit a worksheet. References are updated if you insert or delete a row or column that is within a reference.

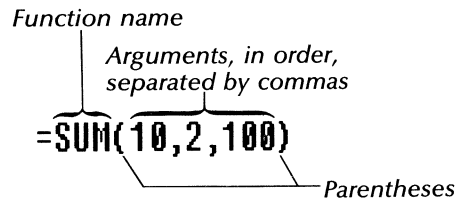
If you add a row just above or below the reference, the reference is not expanded.

You can add a blank row, or a row containing values the function will ignore, above and below the actual figures. Then have the references in your formulas include the blank or ignored values. If you add a row above or below your actual data, the new row will still be within the reference, so it will be updated.

## How to Use Functions

You use functions by entering them in formulas on a worksheet or on a macro sheet. For information on entering formulas, see *Formula in Microsoft Excel Reference*.

The format in which you must enter a function is called the **syntax** of a function. All functions have the same basic syntax:



There are two ways to enter a function in a formula:

- Type the function name.

Type the function name using uppercase or lowercase letters, or a combination of both. When you enter a formula containing a function, Microsoft Excel automatically changes the name of the function to capital letters.

**Tip** | Typing a function name in lowercase letters automatically checks your spelling: if you've misspelled the name of the function, Microsoft Excel won't change the name to capital letters.

Make sure you don't put spaces in function names — `S UM(5,15)`, for example — or before the first parenthesis — `SUM (5,15)`, for example — or Microsoft Excel won't understand the function.

- Use the Formula Paste Function command.

- 1 Select the cell or cells in which you want to enter the formula, or position the insertion point in the formula bar where you want to paste the function.

- 2 Choose Formula Paste Function.

Microsoft Excel displays a dialog box that lists all built-in worksheet functions alphabetically, as well as any functions you have written yourself. (User-defined functions, called **function macros**, appear at the bottom of the list, and are discussed in Chapter 3, “Macro Basics.”)

- 3 Select the function you want.

- 4 If you want Microsoft Excel to enter placeholders for the arguments to the function, turn on the Paste Arguments check box.

Placeholders for arguments are the same as the names used for the arguments in this manual. For example, the syntax for the REPT function is REPT(text,number\_times). Both *text* and *number\_times* are placeholders. If you choose the Formula Paste Function command, select REPT, and turn on Paste Arguments, you will see “REPT(text,number\_times)” in the formula bar. An ellipsis ( . . . ) following an argument means you can enter more than one argument of the previous data type. For more information, see “Syntax” in Chapter 2, “Worksheet Function Directory.”

- 5 Choose the OK button.

Microsoft Excel enters the function name and parentheses in the formula bar. If you’re starting a new formula with this function, Microsoft Excel precedes the function name with an equal sign.

Some functions, such as INDEX, have more than one form. If you turned on the Paste Arguments check box for one of those functions, another dialog box appears. Select the set of arguments that you want, then choose the OK button.

- 6 Type in or edit the arguments in the formula bar.

If there’s more than one argument, separate the arguments by commas.

If you turned on the Paste Arguments check box, be sure to replace the argument placeholders with actual arguments. For example, if you turned on Paste Arguments with the RIGHT function, you see “RIGHT(text,number\_of\_chars)” in the formula bar. Replace *text* with a text argument and *number\_of\_chars* with a number argument. If you forget to replace a placeholder, Microsoft Excel assumes that the placeholder is a name. In most cases, you would see the #NAME? error value displayed in the cell after you entered the formula. If you have a name defined that is the same as one of the placeholders, Microsoft Excel uses that name as the argument.

**Tip** | If, after choosing the Formula Paste Function command, you select the list box in the dialog box and then type the first letter of the function you want, Microsoft Excel scrolls to the first function beginning with that letter. This works for any list in a dialog box.

For information on the order operations are performed in, see Operator in *Microsoft Excel Reference*.

**Note** | Equal signs aren't part of functions; they tell Microsoft Excel that cells contain formulas. If you omit the equal sign before a formula, Microsoft Excel regards the formula as text. Even if you use more than one function in a formula, you only need one equal sign, as in the following formula:

=SUM(E12:E15,AVERAGE(F12:F15))\*LOG(A3)

The following formula would not be allowed:

=SUM(E12:E15,=AVERAGE(F12:F15))\*=LOG(A3)

For information on formulas, see Formula in *Microsoft Excel Reference*.

## Getting Quick Information

To get quick information about worksheet functions:

- **Use Help.**

Help has concise, convenient information on all the worksheet functions. For information on using Help, see Help in *Microsoft Excel Reference*.

- **Use the Formula Paste Function command.**

If you forget how to spell a particular worksheet function (is it DSTDEV or DSTDDEV?), it's easy to find the correct name in the Formula Paste Function dialog box.

- **Use the Paste Arguments option of the Formula Paste Function command.**

If you don't remember what arguments a function takes, the Paste Arguments option is a handy way to get a quick reminder.

For complete information on any worksheet function, see Chapter 2, "Worksheet Function Directory," where functions are listed in alphabetical order.

For information on how Microsoft Excel worksheet functions relate to functions in other worksheet products, see Functions in the Help index. For information on using Help, see Help in *Microsoft Excel Reference*.

## Error in Formula

If you see a message that says “Error in formula,” it means that Microsoft Excel doesn’t understand the formula you’re trying to enter. To correct some common errors, make sure:

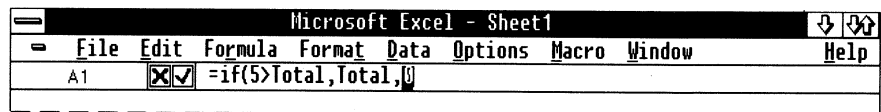
- The name of the function is spelled correctly.
- There are no spaces in the name of the function, or between the name of the function and the first parenthesis.
- There are the same number of left parentheses as right parentheses.
- Multiple arguments are separated by commas.
- You’ve entered the correct number of arguments in the correct order.

### Array Note

If you used an array as an argument, make sure all the rows in the array are the same width. For example, if you try to enter the formula `=SUM({1,2,3;4,5})` you’ll see the message “Error in formula.”

For information on arrays, see *Array* in *Microsoft Excel Reference*.

Microsoft Excel helps you find errors in formulas by highlighting the approximate location of the error. In most cases, either the error is before the highlight, or the highlight is at the end of the formula, which means the formula is incomplete. For example, look at the formula in the formula bar below:



In this case, the highlight is at the end of the formula. The formula is incomplete because the final parenthesis was omitted.

### Tips

- Another common error is to omit the multiplication operator (\*). If you've written the formula  $= (10 + 12)(56000)$  on paper, it's easy to forget that Microsoft Excel doesn't know you mean multiply 10 + 12 by 56000. The correct Microsoft Excel formula would be:

$= (10 + 12) * 56000$

- Put quotes around a text value, and omit quotes around a name. If you are using the word "Price" as an argument, it should be enclosed in double quotation marks. If Price is a name that you have defined as a particular value, reference, or formula, it should not be enclosed in double quotation marks.

If you omit putting double quotation marks around a text argument, Microsoft Excel assumes it's a name; if there's no name defined for that word, the function returns the error value #NAME?. If you use double quotation marks around a name, Microsoft Excel assumes it's a text value; if a text value isn't a valid argument, you'll see the "Error in formula" message.

If you aren't sure if a word is text or a name, you can check this way:

- 1 Choose Formula Define Name.
- 2 Look for the word in the dialog box.
- 3 If it's listed, it's a name. Select the name in the list box to see what it's defined as.

If it's not listed, it's text and should be enclosed in double quotation marks.

- If you can't find the problem right away, you may want to enter the formula as text so you don't lose what you've already typed.

To store the information as text:

- 1 Delete the equal sign.
- 2 Enter the text.
- 3 When you find the problem, edit the text to correct the problem and insert the equal sign.

## Finding the Function You Need

If you want to perform a particular operation, and think there may be a function that can help you, the first place to look is in the section “Functions by Subject Category” in Chapter 2, “Worksheet Function Directory.” That section contains a list of all the worksheet functions, grouped into categories like text functions and trigonometric functions. The directory contains a detailed description for each worksheet function, along with examples and a list of related functions.

If you don’t find a built-in worksheet function that does what you need, you may want to consider writing a function macro. For information on macros, see Chapter 3, “Macro Basics.”

**Tip** | Help also contains lists of functions grouped by subject. For information on using Help, see Help in *Microsoft Excel Reference*.

## Data Types

There are six types of data in Microsoft Excel.

### Numbers

For example, 5.003, 0, 150.286, or –30. Numbers without decimals are called **integers**, for example 5, 0, 150, or –30. Numbers are accurate to 15 digits.

### Text

For example, “a”, “Word”, “w/punc.”, “”, or “ ”. Text values used in formulas must be enclosed in double quotation marks. If the text itself contains quotation marks, use two double quotation marks for each double quotation mark in the text. For example, to find the length in characters for this text—in the “good” old days—enter the formula:

LEN(“in the ““good”” old days”)

Text values can be from 0 to 253 characters long if they are enclosed in quotation marks and from 0 to 255 characters long if they are not. A text constant that contains no characters is written as “”, and is sometimes called “empty text.”

### Logical Values

There are only two: TRUE and FALSE.

### Arrays

For example, {1,2,3;4,5,6}. For information on arrays, see Arrays in *Microsoft Excel Reference*.

### Error Values

For example, #NUM!, #N/A, or #DIV/0!. For information on the different error values, see Error value in *Microsoft Excel Reference*.

### References

For example, \$A\$10, A10, \$A10, A\$10, R1C1, or R[10]C[-10]. References can refer to single cells, ranges, or multiple selections, and can be relative, absolute, or mixed. For information on references, see References in *Microsoft Excel Reference*.

### Types of Arguments

A function can have from 0 to 14 arguments. These arguments can be numbers, text, logical values, arrays, error values, or references.

Arguments can be anything that produces a desired data type. For example, the SUM function, which adds its arguments, can take 1 to 14 arguments. You can give the SUM function any of the following arguments that produce a number or numbers.

- A value that is a number, such as:  
=SUM(1,10,100)
- A formula that results in a number, such as:  
=SUM(.5 + .5,AVERAGE(5,5),10^2)

Using a function as an argument to a function, as in the example =SUM(.5 + .5,AVERAGE(5,5),10^2) above, is called **nesting** functions. In that example, the AVERAGE function is an argument to the SUM function. You can nest up to eight levels of functions in a formula. For example, the following formula is allowed:

=SUM(SUM(SUM(SUM(SUM(SUM(SUM(SUM(SUM(5))))))))))



**Tip** If you are entering a complicated formula, using the Formula Paste Function command to enter the functions can help you get the parentheses in the right place. You should, however, avoid nesting formulas that deeply. They're hard to read and to edit.

If you work with a lot of complicated formulas, you might want to learn about function macros. Function macros let you create your own customized functions. For information on function macros, see Chapter 3, "Macro Basics."

- A reference to a cell that contains a number or a formula that results in a number, such as:

=SUM(A1,A2)

- A reference to more than one cell that contains a number or a formula that results in a number, such as:

=SUM(A1:A5)

The example =SUM(A1:A5) above is equivalent to the formula =SUM(A1,A2,A3,A4,A5). An advantage of using the A1:A5 form is that the argument A1:A5 counts as only 1 argument, while the second form counts as 5 arguments. If you wanted to add more than 14 numbers, you would have to use the first form, because you can only use 14 arguments with any function.

- The name of any of the above, such as:

=SUM(CurrentRate,Inflation)

**Note** When you use the contents of a cell or cells as an argument, the formatting of that cell or cells does not affect the value that is used.

The LEN function, for example, returns the length in characters of a text argument. Suppose cell A1 contains the text value "river", and is formatted with the custom format "@\*.". Cell A1 will look like this:

	A	B	C	D	E	F	G	H
1	river.....							
2								

But LEN(A1) is equal to 5 (5 characters), no matter how cell A1 is formatted. The periods displayed in the cell are not part of the contents of the cell, so they are not counted as part of the argument.

## Translating Data Types

If you use an argument that does not produce the correct data type, Microsoft Excel tries to translate the argument into the desired data type. For example, the LEN function takes one text argument and returns the number of characters in that argument. LEN("abc") is equal to 3. If you enter the formula =LEN(1202) in a cell, Microsoft Excel internally translates the number 1202 into the text value "1202". The formula =LEN(1202) is therefore equal to 4.

**Names and references** are translated as described in the previous section, "Types of Arguments."

**Error values** are not translated. If you use an error value as an argument to a function that does not accept error values as arguments, the function returns another error value.

**Numbers, text, and logical values** are translated as follows:

		If argument should be:		
But is given as:		Number	Text	Logical
	Number	No translation necessary. If a number is supposed to be an integer, the fractional part is deleted, just as in the INT function.	Number is translated to text, as if it were enclosed in double quotation marks.	If the number is zero, it is translated to FALSE. If the number is not zero, it is translated to TRUE.
	Text	If the text value is in any Microsoft Excel standard number, date, time, or currency format, it is interpreted as if the quotation marks weren't there, just as in the VALUE function.  If the text is not in a standard format, it is not translated.	No translation necessary.	If the text is "true" (in upper- and/or lowercase), it is translated to TRUE. If the text is "false," it is translated to FALSE. Otherwise the text is not translated.
	Logical	TRUE is translated to 1; FALSE is translated to 0.	TRUE is translated to "TRUE," FALSE is translated to "FALSE."	No translation necessary.

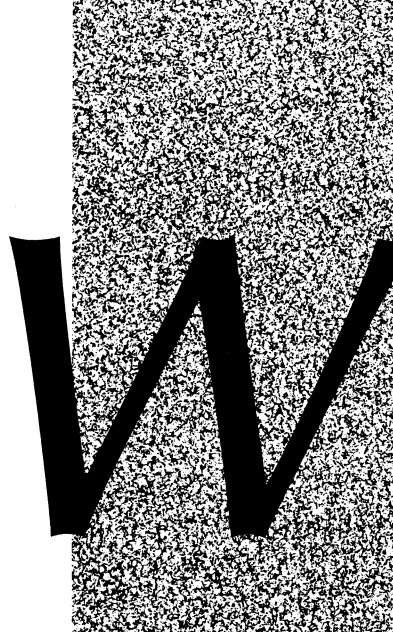
**Arrays** are translated three ways:

- Some worksheet functions (AND, AVERAGE, COUNT, COUNTA, MAX, MIN, OR, STDEV, STDEVP, SUM, VAR, and VARP) can take from 1 to 14 arguments, all of the same data type. If an array is used as an argument to one of those functions, each element of the array is used to produce a single result. For example, =SUM({1,2,3;4,5,6}) equals 21. This operates just as if you had entered the formula =SUM(A1:C2) on the following worksheet:

	A	B	C	D	E	F	G	H
1	1	2	3					
2	4	5	6					
3								

- If the formula is an array formula, the function acts on each element of the array and produces an array result. The ABS function uses a single number as an argument, and returns a positive number. The array formula {=ABS({1,-2,3;-4,5,-6})} equals {1,2,3;4,5,6}.
- If the formula is not an array formula and the function is not one of those listed above, the function takes the first element of the array and ignores the rest. For example, the formula =ABS({1,-2,3;-4,5,-6}) equals 1.





# *Chapter 2*

.....

# *Worksheet Function Directory*

Conventions . . . . .	16
Syntax . . . . .	16
Commas . . . . .	17
Argument Data Types . . . . .	18
Functions by Subject Category . . . . .	18
Database Functions . . . . .	19
Date and Time Functions . . . . .	20
Financial Functions . . . . .	20
Information Functions . . . . .	21
Logical Functions . . . . .	22
Lookup Functions . . . . .	23
Mathematical Functions . . . . .	23
Matrix Functions . . . . .	24
Statistical Functions . . . . .	24
Text Functions . . . . .	25
Trigonometric Functions . . . . .	26
Directory . . . . .	27

The worksheet function directory contains descriptions of all the Microsoft Excel worksheet functions, listed alphabetically. If you don't know the name of the function you want, see the section "Functions by Subject Category" in this chapter.

## Conventions

The following conventions are used in this manual:

Convention	Shows
<i>Italics</i>	Optional arguments
. . . (ellipsis)	The preceding argument can be repeated

### Note

When the names of arguments appear within a paragraph instead of in a line showing syntax, the names of both required and optional arguments appear in italics, just so it's easier to see them. For example, the CELL function has one required argument and one optional argument. Its syntax is CELL(*type\_of\_info*,*reference*). However, if we talk about the *type\_of\_info* and *reference* arguments within text, they both appear in italics.

## Syntax

The syntax for each function is given in the heading of its description. When an argument is printed in italics, it means that argument is optional. For example, the syntax for the LEFT worksheet function looks like this:

LEFT(*text*,*number\_of\_characters*)

Either of the following formulas is allowed:

=LEFT("Hiram",2)

=LEFT("Corley")

The formula =LEFT() is not allowed; the argument "text" is not shown in italics, so it is required.

When an argument is followed by an ellipsis (. . .), it means that you can have more than one argument of the previous data type. For example, the syntax for the MAX worksheet function looks like this:

MAX(*number1*,*number2*,. . .)

Any of the following formulas would be allowed:

=MAX(26)

=MAX(26,31)

=MAX(26,31,29)

For information on the arguments of each function, see the description of the specific function in the directory.

### Commas

Separate arguments in functions with commas, and be careful not to type extra commas. If you use commas to hold a place for an argument, but don't enter the argument, Microsoft Excel substitutes an appropriate value for that argument. For example, for a function that takes three arguments, if you enter (,arg2,arg3) as the argument, Microsoft Excel substitutes an appropriate value for *arg1*. If you enter (arg1,,), it substitutes appropriate values for *arg2* and *arg3*.

For most arguments, the value substituted for an omitted argument is 0, FALSE, or "", the empty text, depending on what the data type of the argument should be. If an omitted argument is assumed to be some other value, the description of the function in the directory will tell you so.

For example, the HLOOKUP worksheet function requires three arguments. The description of the HLOOKUP function does not specify that any of the arguments are an assumed value if omitted. If you enter =HLOOKUP(3,A2:A5,) in a cell, Microsoft Excel uses 0 for the third argument. If you enter =HLOOKUP(3,A2:A5) in a cell, Microsoft Excel displays the "Error in formula" message.

#### Note

If you are using a reference as an argument and that reference uses a comma as a union operator, enclose the reference in parentheses. The AREAS function, for example, takes one argument: a reference. If you try to enter the formula =AREAS(A1,C1) Microsoft Excel interprets A1 and C1 as 2 separate arguments, and displays the "Error in formula" message. The correct form would be =AREAS((A1,C1)). For information on reference operators, see Operator in *Microsoft Excel Reference*.

### Argument Data Types

Many of the names used for arguments in the worksheet function directory tell you what data type that argument should be. For example, in the function `LEFT(text,number_of_characters)`, the first argument must be text and the second argument must be a number.

If the abbreviations *num*, *ref*, or *log* appear in the name of an argument, that argument must be a number, a reference, or a logical value, respectively. Similarly, the words *number*, *reference*, *logical*, *text*, and *array* in the name of an argument specify that the argument must be of that data type. *Value* or its abbreviation, *val*, means that the argument can be anything that results in a single value. That value can be a number, text, logical, or error value.

**Note** | Not every argument name specifies the argument's data type. For functions that have many arguments, adding the data type abbreviations makes the syntax listing too long to easily read. For these functions, the data types for each argument are specified in the description of the function.

### Functions by Subject Category

This section lists functions by the following subject categories:

- Database functions
- Date and time functions
- Financial functions
- Information functions
- Logical functions
- Lookup functions
- Mathematical functions
- Matrix functions
- Statistical functions
- Text functions
- Trigonometric functions

All the functions are listed alphabetically.



## Database Functions

All the database functions are listed in the section “*Dfunction*” in this chapter.

DAVERAGE(database,field,criteria)

Average of numbers in specified *field* of records in *database* matching *criteria*

DCOUNT(database,field,criteria)

Count of numbers in specified *field* of records in *database* matching *criteria*

DCOUNTA(database,field,criteria)

Count of non-empty cells in specified *field* of records in *database* matching *criteria*

DMAX(database,field,criteria)

Maximum of numbers in specified *field* of records in *database* matching *criteria*

DMIN(database,field,criteria)

Minimum of numbers in specified *field* of records in *database* matching *criteria*

DPRODUCT(database,field,criteria)

Product of numbers in specified *field* of records in *database* matching *criteria*

DSTDEV(database,field,criteria)

Estimate of standard deviation of a population, based on a sample, using numbers in specified *field* of records in *database* matching *criteria*

DSTDEVP(database,field,criteria)

Standard deviation of a population, based on the entire population, using numbers in specified *field* of records in *database* matching *criteria*

DSUM(database,field,criteria)

Sum of numbers in specified *field* of records in *database* matching *criteria*

DVAR(database,field,criteria)

Estimate of variance of a population, based on a sample, using numbers in specified *field* of records in *database* matching *criteria*

DVARP(database,field,criteria)

Variance of a population, based on the entire population, using numbers in specified *field* of records in *database* matching *criteria*

### Date and Time Functions

DATE(year,month,day)  
Serial number of specified date

DATEVALUE(date\_text)  
Serial number of *date\_text*

DAY(serial\_number)  
Converts *serial\_number* to a day of the month

HOUR(serial\_number)  
Converts *serial\_number* to an hour of the day

MINUTE(serial\_number)  
Converts *serial\_number* to a minute

MONTH(serial\_number)  
Converts *serial\_number* to a month of the year

NOW( )  
*Serial\_number* of current date and time

SECOND(serial\_number)  
Converts *serial\_number* to a second

TIME(hour,minute,second)  
Serial number of specified time

TIMEVALUE(time\_text)  
Serial number of *time\_text*

WEEKDAY(serial\_number)  
Converts *serial\_number* to a day of the week

YEAR(serial\_number)  
Converts *serial\_number* to a year

### Financial Functions

DDB(cost,salvage,life,period)  
Depreciation of an asset using the double-declining balance method

FV(rate,nper,pmt,pv,type)  
Future value of investment

IPMT(rate,per,nper,pv,fv,type)  
Interest payment for an investment

IRR(values,guess)  
Internal rate of return of *values*

MIRR(values,finance\_rate,reinvest\_rate)  
Modified internal rate of return of *values*

NPER(rate,pmt,pv,fv,type)  
Number of payments of investment

NPV(rate,value1,value2,...)  
Net present value of *values*

PMT(rate,nper,pv,fv,type)  
Periodic payment of investment

PPMT(rate,nper,pv,fv,type)  
Payment on the principal for an investment

PV(rate,nper,pmt,fv,type)  
Present value of investment

RATE(nper,pmt,pv,fv,type,guess)  
Rate returned on investment

SLN(cost,salvage,life)  
Straight-line depreciation for an asset

SYD(cost,salvage,life,per)  
Sum-of-years' digits depreciation for an asset

### Information Functions

AREAS(reference)  
Number of areas in *reference*

CELL(type\_of\_info,reference)  
Information about formatting, location, or contents of *reference*

COLUMN(reference)  
Column numbers in *reference*

COLUMNS(array)  
Number of columns in *array*

INDIRECT(ref\_text,type\_of\_ref)  
Contents of the cell from its *ref*

**Note** | The first nine functions listed below are listed in the section “ISfunction” in this directory.

ISBLANK(value)  
True if *value* is blank

ISERR(value)  
True if *value* is any error value except #N/A

ISERROR(value)  
True if *value* is any error value

ISLOGICAL(*value*)  
True if *value* is a logical value

ISNA(*value*)  
True if *value* is the error value #N/A

ISNONTEXT(*value*)  
True if *value* is not text

ISNUMBER(*value*)  
True if *value* is a number

ISREF(*value*)  
True if *value* is a reference

ISTEXT(*value*)  
True if *value* is text

N(*value*)  
*Value* translated into a number

NA( )  
Error value #N/A

ROW(*reference*)  
Row numbers in *reference*

ROWS(*array*)  
Number of rows in *array*

T(*value*)  
*Value* translated into text

TYPE(*value*)  
Type of *value*

## Logical Functions

AND(*logical1,logical2,...*)  
True if every argument is TRUE; otherwise, FALSE

FALSE( )  
Logical value FALSE

IF(*logical\_test,value\_if\_true,value\_if\_false*)  
*Value\_if\_true* if *logical\_test* is TRUE; *value\_if\_false* if *logical\_test* is FALSE

NOT(*logical*)  
True if *logical* is FALSE; false if *logical* is TRUE

OR(*logical1,logical2,...*)  
True if any argument is TRUE; otherwise, FALSE

TRUE( )  
Logical value TRUE

## Lookup Functions

CHOOSE(index\_number,value1,value2,...)

Uses *index\_number* to select a value from *values*

HLOOKUP(lookup\_value,table\_array,row\_index\_num)

Value in a table selected by *lookup\_value*

INDEX(ref,row\_num,column\_num,area\_num)

INDEX(array,row\_num,column\_num)

Reference in *ref* or value in *array* selected by index values

LOOKUP(lookup\_value,lookup\_vector,result\_vector)

LOOKUP(lookup\_value,array)

Value in a table selected by *lookup\_value*

MATCH(lookup\_value,lookup\_array,type\_of\_match)

Index of a value selected by *lookup\_value*

VLOOKUP(lookup\_value,table\_array,col\_index)

Value in a table selected by *lookup\_value*

## Mathematical Functions

ABS(number)

Absolute value of *number*

EXP(number)

$e(2.718\dots)$  to the power *number*

FACT(number)

Factorial of *number*

INT(number)

*Number* rounded down to the nearest integer

LN(number)

Natural logarithm of *number*

LOG(number,base)

Logarithm of *number* in *base*

LOG10(number)

Base 10 logarithm of *number*

MOD(number,divisor\_number)

Remainder of *number* divided by *divisor\_number*

PI()

Value of  $\pi$

## Worksheet Functions

PRODUCT(number1,number2,...)

Product of *numbers*

RAND( )

Random number between 0 and 1

ROUND(number,number\_of\_digits)

Rounds *number* to *number\_of\_digits*

SIGN(number)

Sign of *number*

SQRT(number)

Square root of *number*

TRUNC(number)

Integer part of *number*

## Matrix Functions

MDETERM(array)

Determinant of *array*

MINVERSE(array)

Inverse of *array*

MMULT(array1,array2)

Product of two arrays

TRANSPOSE(array)

Transpose of *array*

## Statistical Functions

AVERAGE(number1,number2,...)

Average of numbers

COUNT(value1,value2,...)

Count of numbers in *values*

COUNTA(value1,value2,...)

Count of values in *values*

GROWTH(known\_y's,known\_x's,new\_x's)

Values on exponential trend

LINEST(known\_y's,known\_x's)

Parameters of linear trend

LOGEST(known\_y's,known\_x's)

Parameters of exponential trend

MAX(number1,number2,...)  
Maximum number in *numbers*

MIN(number1,number2,...)  
Minimum number in *numbers*

STDEV(number1,number2,...)  
Estimate of standard deviation of a population based on a sample

STDEVP(number1,number2,...)  
Standard deviation of a population based on the entire population

SUM(number1,number2,...)  
Sum of *numbers*

TREND(known\_y's,known\_x's,new\_x's)  
Values on linear trend

VAR(number1,number2,...)  
Estimate of variance of a population based on a sample

VARP(number1,number2,...)  
Variance of a population based on the entire population

### Text Functions

CHAR(number)  
ASCII character corresponding to *number*

CLEAN(text)  
Removes control characters from *text*

CODE(text)  
ASCII code of the first character in *text*

DOLLAR(number,decimals)  
Rounds *number* and gives as text in currency format

EXACT(text1,text2)  
Tests to see if *text1* and *text2* are exactly the same

FIND(find\_text,within\_text,start\_at\_num)  
Finds *find\_text* within *within\_text*

FIXED(number,decimals)  
Rounds *number* and gives as text

LEFT(text,number\_of\_characters)  
Extracts first *number\_of\_characters* from *text*

LEN(text)  
Length of *text*

LOWER(text)  
Converts *text* to lowercase

## Worksheet Functions

MID(text,start\_number,number\_of\_characters)

Extracts *number\_of\_characters* from *text*

PROPER(text)

Converts *text* to initial capitals

REPLACE(old\_text,start\_num,num\_chars,new\_text)

Replaces *num\_chars* characters in *old\_text* with *new\_text*

REPT(text,number\_times)

Repeats *text* *number\_times* times

RIGHT(text,number\_of\_chars)

Last *number\_of\_chars* characters in *text*

SEARCH(find\_text,within\_text,start\_at\_num)

Searches for *find\_text* within *within\_text*

SUBSTITUTE(text,old\_text,new\_text,instance\_number)

Substitutes *new\_text* for *old\_text* in *text*

TEXT(value,format\_text)

Converts *value* to text using format *format\_text*

TRIM(text)

Removes spaces from *text*

UPPER(text)

Converts *text* to uppercase

VALUE(text)

Converts *text* to a number

## Trigonometric Functions

ACOS(number)

Arccosine of *number*

ASIN(number)

Arcsine of *number*

ATAN(number)

Arctangent of *number*

ATAN2(x\_number,y\_number)

Arctangent of point (*x\_number*,*y\_number*)

COS(radians)

Cosine of *radians*

SIN(radians)

Sine of *radians*

TAN(radians)

Tangent of *radians*



# Directory

## ABS(number)

Returns the absolute value of *number*. The absolute value of a number is the number without its sign.

### Examples

$\text{ABS}(-2)$  equals 2

$\text{ABS}(2)$  equals 2

### Related Functions

SIGN returns the sign of a number as a value: 1 (positive), -1 (negative), or 0 (zero).

## ACOS(number)

Returns the arccosine of *number*. The arccosine is the angle whose cosine is *number*. *Number* must be in the range -1 to 1. The angle is given in radians in the range 0 to  $\pi$ .

If you want to convert the result from radians to degrees, multiply the result by  $180/\text{PI}()$ .

### Examples

$\text{ACOS}(-0.5)$  equals 2.094 ( $2\pi/3$  radians)

$\text{ACOS}(-0.5)*180/\text{PI}()$  equals 120 (degrees)

### Related Functions

COS returns the cosine of a number. PI returns the value  $\pi$ .

### AND(logical1,logical2,...)

Returns the logical value TRUE if all the arguments are TRUE. If any arguments are FALSE, AND returns the logical value FALSE.

AND can have from 1 to 14 arguments. The arguments should be logical values, or arrays or references that contain logical values. If an array or reference argument contains text or empty cells, those values are ignored.

If there are no logical values in the range specified, AND returns the error value #VALUE!.

#### Examples

AND(TRUE,TRUE) *equals* TRUE

AND(FALSE,TRUE) *equals* FALSE

AND(2 + 2 = 4, 2 + 3 = 5) *equals* TRUE

If B1:B3 contains the values TRUE, FALSE, and TRUE, then:

AND(B1:B3) *equals* FALSE

If B4 contains a number between 1 and 100, then:

AND(1 < B4, B4 < 100) *equals* TRUE

If B4 contains a number that is less than or equal to 1 or greater than or equal to 100, then:

AND(1 < B4, B4 < 100) *equals* FALSE

#### Related Functions

NOT reverses the logic of its argument. OR is TRUE if one or more arguments are TRUE.

### AREAS(reference)

Returns the number of areas in *reference*. An **area** is a range of cells or a single cell. *Reference* can be a reference to multiple areas. For information on multiple selections, see *Multiple selection in Microsoft Excel Reference*.

**Macro Note**

This function is especially useful in a macro for testing if a reference is a multiple selection.

## Examples

AREAS(B2:D4) *equals* 1

If the name “Prices” refers to the areas B1:D4, B2, and E1:E10, then:

AREAS(Prices) *equals* 3

## Related Functions

CELL, COLUMN, COLUMNS, ROW, and ROWS all return information about a reference. INDEX can be used to return a particular area from a reference.

## ASIN(number)

Returns the arcsine of *number*. The arcsine is the angle whose sine is *number*. *Number* must be in the range  $-1$  to  $1$ . The angle is given in radians in the range  $-\pi/2$  to  $\pi/2$ .

If you want to convert the result from radians to degrees, multiply the result by  $180/\text{PI}()$ .

## Examples

ASIN( $-0.5$ ) *equals*  $-0.524$  ( $-\pi/6$  radians)

ASIN( $-0.5$ )\* $180/\text{PI}()$  *equals*  $-30$  (degrees)

## Related Functions

SIN returns the sine of a number. PI returns the value  $\pi$ .

## ATAN(number)

Returns the arctangent of *number*. The arctangent is the angle whose tangent is *number*. The angle is given in radians in the range  $-\pi/2$  to  $\pi/2$ .

If you want to convert the result from radians to degrees, multiply the result by  $180/\text{PI}()$ .

### Examples

$\text{ATAN}(1)$  equals 0.785 ( $\pi/4$  radians)

$\text{ATAN}(1)*180/\text{PI}()$  equals 45 (degrees)

### Related Functions

$\text{ATAN2}$  returns the arctangent from  $x$  and  $y$  coordinates.  $\text{TAN}$  returns the tangent of a number.  $\text{PI}$  returns the value  $\pi$ .

## $\text{ATAN2}(x\_number, y\_number)$

Returns the arctangent of the  $x$  and  $y$  coordinates represented by  $x\_number$  and  $y\_number$ . The arctangent is the angle from the  $x$ -axis to the coordinates  $x\_number, y\_number$ . The angle is given in radians in the range  $-\pi$  to  $\pi$ , excluding  $-\pi$ . A positive result represents a counterclockwise angle from the  $x$ -axis; a negative result represents a clockwise angle.

If you want to convert the result from radians to degrees, multiply the result by  $180/\text{PI}()$ .

If both  $x\_number$  and  $y\_number$  are 0,  $\text{ATAN2}$  returns the error value  $\#DIV/0!$ .

**Note** |  $\text{ATAN2}(a,b)$  equals  $\text{ATAN}(b/a)$ , except that  $a$  can equal 0 in  $\text{ATAN2}$ .

### Examples

$\text{ATAN2}(1,1)$  equals 0.785 ( $\pi/4$  radians)

$\text{ATAN2}(-1, -1)$  equals  $-2.356$  ( $-3\pi/4$  radians)

$\text{ATAN2}(-1, -1)*180/\text{PI}()$  equals  $-135$  (degrees)

### Related Functions

$\text{ATAN}$  returns the arctangent of an angle.  $\text{TAN}$  returns the tangent of a number.  $\text{PI}$  returns the value  $\pi$ .

## $\text{AVERAGE}(\text{number1}, \text{number2}, \dots)$

Returns the average of the arguments.

$\text{AVERAGE}$  can have from 1 to 14 arguments. The arguments should be numbers, or arrays or references that contain numbers. If an array or reference argument contains text, logical values, or empty cells, those values are ignored.

## Examples

	A	B	C	D	E	F	G	H
1	10							
2	20							
3	6							
4								

In the worksheet above:

AVERAGE(A1:A3) *equals* 12

AVERAGE(A1:A3,4) *equals* 10

AVERAGE(A1:A3) *equals* SUM(A1:A3)/COUNT(A1:A3)

This example uses an array constant:

AVERAGE({8,9,"text",10}) *equals* 9

## Related Functions

COUNT and COUNTA count numbers or values. SUM adds its arguments. DAVERAGE returns the average of selected database entries.

## CELL(type\_of\_info,reference)

Returns information about the formatting, location, or contents of the upper-left cell in *reference*. If *reference* is omitted, it is assumed to be the current selection. If *reference* is a multiple reference, CELL returns the error value #VALUE!.

*Type\_of\_info* is a text value that specifies what type of cell information you want. The following list shows the possible values of *type\_of\_info* and the corresponding result:

Value	Result
"width"	Column width of cell, rounded off to an integer. Units for column width are the width of 1 character in font number 1.
"row"	Equals the row number in <i>reference</i>
"col"	Equals the column number in <i>reference</i>

Value	Result
“protect”	If cell is not locked, returns 0. If cell is locked, returns 1.
“address”	The reference of the first cell in <i>reference</i> , as text
“contents”	The value contained in <i>reference</i>
“format”	A text value corresponding to the format of the cell. See the list below to find what text values correspond to the Microsoft Excel formats.
“prefix”	A text value corresponding to the “label prefix” of the cell. In Microsoft Excel, this means that CELL returns: “” if the cell contains left-aligned text; “” if the cell contains right-aligned text; “^” if the cell contains centered text; “”, the empty text, if the cell contains anything else.
“type”	A text value corresponding to the type of data in the cell: “b” for blank if the cell is empty; “l” for label if the cell contains a text constant; “v” for value if the cell contains anything else

The following list describes what text values correspond to the Microsoft Excel built-in formats:

Microsoft Excel format	Text value returned
General	“G”
0 or #,##0	“F0”
0.00 or #,##0.00	“F2”
\$#,##0;(\$#,##0) or \$#,##0;[RED](\$#,##0)	“C0”
\$#,##0.00;(\$#,##0.00) or \$#,##0.00;[RED](\$#,##0.00)	“C2”
0%	“P0”
0.00%	“P2”

Microsoft Excel format	Text value returned
0.00E + 00	“S2”
m/d/yy <i>or</i> m/d/yy h:mm	“D4”
d-mmm-yy	“D1”
d-mmm	“D2”
mmm-yy	“D3”
h:mm AM/PM	“D7”
h:mm:ss AM/PM	“D6”
h:mm	“D9”
h:mm:ss	“D8”

If *type\_of\_info* is “format” and *reference* is formatted with a customized number format, Microsoft Excel returns a code representing the cell format. For information on formats, see *Formatting a document in Microsoft Excel Reference*.

**Note** | The CELL function is provided for compatibility with other worksheet programs. If you need to use cell information in a macro, GET.CELL provides a broader set of attributes.

## Examples

If B12 has the format “d-mmm”, then:

CELL(“format”,B12) *equals* “D2”

CELL(“row”,A20) *equals* 20

If the upper-left cell in the current selection contains TOTAL, then:

CELL(“contents”) *equals* “TOTAL”

## Related Functions

AREAS, COLUMN, COLUMNS, ROW, ROWS, and the IS functions all return information about a reference.

## CHAR(number)

Returns the ASCII character corresponding to the code *number*. *Number* can be any number between 1 and 255.

### Examples

CHAR(65) equals "A"

CHAR(33) equals "!"

### Related Functions

CODE returns the ASCII code for a character.

## CHOOSE(index\_number,value1,value2,...)

Uses *index\_number* to choose from the list of values: *value1,value2,...*. If *index\_number* is 1, CHOOSE returns *value1*; if *index\_number* is 2, CHOOSE returns *value2*; and so on.

If *index\_number* is less than 1 or greater than the number of the last value in the list, CHOOSE returns the error value #VALUE!.

### Macro Note

If you are using CHOOSE in a macro, the *values* can also be GOTO functions or action-taking functions. For example, the following functions are allowed in a macro:

=CHOOSE(level,GOTO(begin),GOTO(intermed),GOTO(adv))

=CHOOSE(file,ACTIVATE.NEXT(),ACTIVATE.PREV())

If *index\_num* is an array, every *value* is evaluated when CHOOSE is executed. If some of those *values* are action-taking functions, all of the actions are taken. For example, the following formula opens both a new worksheet and a new chart:

=CHOOSE({1,2},NEW(1),NEW(2))



## Note

The *value* arguments to CHOOSE can be range references as well as single values. For example, the formula:

=SUM(CHOOSE(2,A1:A10,B1:B10,C1:C10))

returns the sum of the values in cells B1:B10. This is powerful, but can get a little confusing. Suppose you entered the following formula in cell A1:

=CHOOSE(2,A1:A10,B1:B10,C1:C10)

The formula returns the reference B1:B10. Now Microsoft Excel interprets that result just as if you had entered in cell A1:

=B1:B10

Microsoft Excel uses the implicit intersection of cell A1 and cells B1:B10, and evaluates the final result to be equal to the contents of cell B1. For information on implicit intersection, see Names in *Microsoft Excel Reference*.

## Examples

CHOOSE(2,"1st","2nd","3rd","Finished") *equals* "2nd"

SUM(A1:CHOOSE(3,A10,A20,A30)) *equals* SUM(A1:A30)

## Related Functions

INDEX uses an index to choose a value from a reference or array.

## CLEAN(text)

Removes all non-printable characters from *text*.

## Example

Since CHAR(7) returns a non-printable character:

=CLEAN(CHAR(7)&"text"&CHAR(7)) *equals* "text"

## Related Functions

TRIM removes spaces from text. CODE returns the ASCII code for a character.

## CODE(text)

Returns the numeric ASCII code of the first character in *text*.

### Example

CODE("Alphabet") *equals* 65

### Related Functions

CHAR returns the character corresponding to an ASCII code.

## COLUMN(reference)

Returns the column number of *reference*. If *reference* is a range of cells, COLUMN returns the column numbers of *reference* as a horizontal array. *Reference* cannot be a reference to multiple areas.

If *reference* is omitted, COLUMN refers to its own cell.

### Examples

COLUMN(A3) *equals* 1

COLUMN(A3:C5) *equals* {1,2,3}

If COLUMN is entered in C5, then:

COLUMN() *equals* COLUMN(C5) *equals* 3

### Related Functions

COLUMNS returns the number of columns in an array. ROW returns the row number(s) in a reference. ROWS returns the number of rows in an array.

## COLUMNS(array)

Returns the number of columns in *array*.

### Examples

COLUMNS(A1:C4) *equals* 3

COLUMNS({1,2,3;4,5,6}) *equals* 3

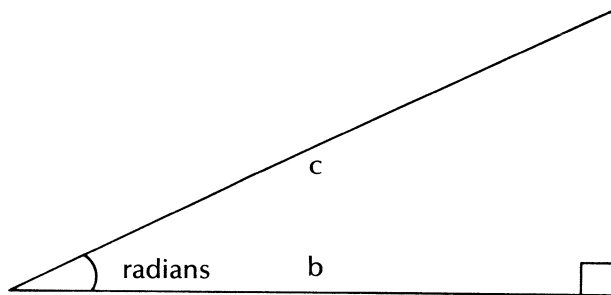
### Related Functions

COLUMN returns the column number(s) in a reference. ROW returns the row number(s) in a reference. ROWS returns the number of rows in an array.

## COS(radians)

Returns the cosine of *radians*, where *radians* is the number of radians in an angle.

If your argument is in degrees, multiply the argument by  $\text{PI}()/180$  to convert the argument to radians.



$$\text{COS}(\text{radians}) = b/c$$

### Examples

COS(1.047) *equals* 0.5

COS(60\*PI()/180) *equals* 0.5

### Related Functions

ACOS returns the arccosine of a number. PI returns the value  $\pi$ .

# COUNT(value1,value2,...)

Counts how many numbers are in the list of arguments.

COUNT can have from 1 to 14 arguments. Arguments that are numbers, empty cells, logical values, or text representations of numbers are counted; arguments that are error values or text that cannot be translated into numbers are ignored. If an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, logical values, text, or error values in the array or reference are ignored.

## Examples

	A	B	C	D	E	F	G	H
1	0.1							
2	TRUE							
3	three							
4	4							
5								
6	6.6666							
7	700							
8								
9	9							
10	#DIV/0!							
11								

In the worksheet above:

COUNT(A6:A7) equals 2

COUNT(A4:A7) equals 3

COUNT(A2,A6:A9) equals 3

COUNT(A1:C10) equals 5

COUNT(0.1,TRUE,“three”,4,,6.6666,700,,9,#DIV/0!) equals 8

## Related Functions

COUNTA counts values. AVERAGE averages its arguments. SUM adds its arguments. DCOUNT counts the cells containing numbers from selected data-base entries.

## COUNTA(value1,value2,...)

Counts how many values are in the list of arguments. If an argument is an array or reference, empty cells within the array or reference are ignored.

COUNTA can have from 1 to 14 arguments.

### Examples

	A	B	C	D	E	F	G	H
1	0.1							
2	TRUE							
3	three							
4	4							
5								
6	6.6666							
7	700							
8								
9	9							
10	#DIV/0!							
11								

In the worksheet above:

COUNTA(A6:A7) *equals* 2

COUNTA(A4:A7) *equals* 3

COUNTA(A2,A6:A9) *equals* 4

COUNTA(A1:C10) *equals* 8

COUNTA(1,,1) *equals* 3

COUNTA(A4:A7,10) *equals* 4

### Related Functions

COUNT counts numbers. AVERAGE averages its arguments. SUM adds its arguments. PRODUCT multiplies its arguments. DCOUNTA counts the cells containing values from selected database entries. DCOUNT counts the cells containing numbers from selected database entries.

## DATE(year,month,day)

Returns the serial number of the date corresponding to *year*, *month*, and *day*.

The serial number is an integer in the range 1 to 65380, representing dates from January 1, 1900 through December 31, 2078. Each valid date in this range is numbered sequentially: June 30, 1914 corresponds to 5295, July 1, 1914 corresponds to 5296, and so on.

To represent a valid date, *year* should be between 1900 and 2078; *month* should be between 1 and 12; and *day* should be between 1 and 31. *Year* values between 0 and 178 are interpreted as the years 1900 to 2078. You can also use negative numbers as arguments, as long as the resulting serial number is positive. For example, to find the serial number for the date one week before 9/1/1988, you could use DATE(88,9,1 - 7).

### Tip

The DATE function is most useful in formulas where *year*, *month*, or *day* are formulas, not constants.

It's unlikely that you'll enter a formula like =DATE(1988,4,15) in a cell because it's easier to enter the date as a constant, such as 4/15/1988 or 15-April-1988. Microsoft Excel will calculate the serial number from the date constant. Likewise, it is not necessary to use the DATE function within a formula that uses dates as constants. Use the text form of the date, "4/15/1988" or "15-April-1988"; the text is automatically converted to a serial number.

### Note

This description assumes that the 1904 Date System check box in the Options Calculation dialog box is turned off. If this check box is turned on, then serial number 1 in the active document is the start of day January 2, 1904, instead of the start of day January 1, 1900. The 1904 Date System box is turned on automatically if you open a document from Microsoft Excel for the Macintosh, which has a different date system. You can also turn on the check box yourself if you are creating a document to be used in Microsoft Excel for Macintosh. For more information, see Date in the *Microsoft Excel Reference*.

## Examples

DATE(87,1,1) equals 31778, the serial number corresponding to January 1, 1987.

	A	B	C	D	E	F	G	H
1	Date of Sales							
2	Month	Day	Year	Term (in days)				
3	1	24	87	90				
4								

In the worksheet above, to find the due date for a bill due 90 days from 1/24/87, use: =DATE(C3,A3,B3 + D3), which equals 31891. The date corresponding to serial number 31891 is April 24, 1987.

## Related Functions

DATEVALUE is like DATE, but requires a text argument. YEAR, MONTH, DAY, and WEEKDAY convert serial numbers into years, months, days, or weekdays. NOW returns the serial number of the current date and time.

## DATEVALUE(date\_text)

Returns the date serial number of *date\_text*. *Date\_text* must represent a date within the range January 1, 1900 to December 31, 2078. Dates can be entered in any of the Microsoft Excel built-in date formats, for example “6/30/87” or “30-Jun-87”.

If the year portion of *date\_text* is omitted, DATEVALUE uses the current year from your computer’s built-in clock. If *date\_text* contains any time information, it is ignored.

### Note

This description assumes that the 1904 Date System check box in the Options Calculation dialog box is turned off. If this check box is turned on, then serial number 1 in the active document is the start of day January 2, 1904, instead of the start of day January 1, 1900. The 1904 Date System box is turned on automatically if you open a document from Microsoft Excel for the Macintosh, which has a different date system. You can also turn on the check box yourself if you are creating a document to be used in Microsoft Excel for the Macintosh. For more information, see Date in *Microsoft Excel Reference*.

## Examples

DATEVALUE(“8/22/55”) equals 20323.

DATEVALUE(“22-Aug-55”) equals 20323.

Assuming your computer’s built-in clock is set to the year 1987:

DATEVALUE(“5-Jul”) equals 31963

DATEVALUE(“22-Aug-55 2:24 AM”) equals DATEVALUE (“22-Aug-55”) equals 20323

## Related Functions

DATE is like DATEVALUE, but requires numeric arguments. YEAR, MONTH, DAY, and WEEKDAY convert serial numbers into years, months, days, or weekdays. NOW returns the serial number of the current date and time.

## DAVERAGE(database,field,criteria)

For information on Microsoft Excel database functions, see the section “Dfunction” later in this directory.

## DAY(serial\_number)

Returns the day of the month corresponding to *serial\_number*. The day is given as an integer, ranging from 1 to 31.

*Serial\_number* is the date-time code used by Microsoft Excel for date and time calculations. Numbers range from 1 to 65380, corresponding to the dates January 1, 1900 through December 31, 2078. Numbers to the right of the decimal point in *serial\_number* represent the time; numbers to the left represent the date. For example, the date-time combination 12:00 P.M., January 1, 1901 is represented as 367.5.

*Serial\_number* may be given as text, such as “4-15-1985” or “15-Apr-1985”, instead of a number. The text is automatically converted to a serial number.

### Note

This description assumes that the 1904 Date System check box in the Options Calculation dialog box is turned off. If this check box is turned on, then serial number 1 in the active document is the start of day January 2, 1904, instead of the start of day January 1, 1900. The 1904 Date System box is turned on automatically if you open a document from Microsoft Excel for the Macintosh, which has a different date system. You can also turn on the check box yourself if you are creating a document to be used in Microsoft Excel for the Macintosh. For more information, see Date in *Microsoft Excel Reference*.

## Examples

DAY(6) equals 6

DAY(6.5) equals 6

DAY(29690) equals 14

DAY(“4-Jan”) equals 4



## Related Functions

YEAR, MONTH, WEEKDAY, HOUR, MINUTE, and SECOND convert serial numbers into years, months, weekdays, hours, minutes, or seconds. NOW returns the serial number of the current date and time.

### DCOUNT(database, *field*, criteria)

For information on Microsoft Excel database functions, see the section “Dfunction” later in this directory.

### DCOUNTA(database, *field*, criteria)

For information on Microsoft Excel database functions, see the section “Dfunction” later in this directory.

### DDB(cost, salvage, life, period)

Returns the depreciation of an asset for a specific *period* using the double-declining balance method based on the asset's initial *cost*, *salvage* value (the value of the asset at the end of its life), and useful *life*. All four arguments must be positive numbers. *Period* and *life* must be given in the same units. For example, if *period* is given in months, *life* must also be given in months.

The double-declining balance method computes depreciation at an accelerated rate. Depreciation is highest in the first period and decreases in successive periods. DDB uses the formula:

$$((cost - \text{total depreciation from prior periods}) * 2) / life$$

to calculate depreciation for a period.

## Examples

Suppose you've purchased a new sewing machine for your shoe factory. The sewing machine cost \$2400 and has a lifetime of 10 years. The salvage value of the machine is \$300.

DDB(2400,300,3650,1) equals \$1.32, the first day's depreciation

DDB(2400,300,120,1) equals \$40.00, the first month's depreciation

DDB(2400,300,10,1) equals \$480.00, the first year's depreciation

DDB(2400,300,10,2) equals \$384.00, the second year's depreciation

DDB(2400,300,10,10) equals \$22.12, the tenth year's depreciation

## Related Functions

SLN returns the straight-line depreciation of an asset for one period. SYD returns the sum-of-year's digits depreciation of an asset for a specified period.

## Database Functions

### *Dfunction*(database,field,criteria)

This section describes the eleven worksheet functions used for Microsoft Excel database calculations. For information on database structure and criteria, see Database in *Microsoft Excel Reference*. Each of these functions, referred to collectively as *Dfunction*, uses three arguments: *database*, *field*, and *criteria*. These arguments determine which worksheet cells are used in the database function.

A sample database and criteria range.

	A	B	C	D	E	F	G	H
1	Name	Species	Age	Value				
2				>\$500				
3	Name	Species	Age	Value				
4	Paul	Mallard	2.6	\$5				
5	Wally	Wombat	2	\$650				
6	Jo	Sun Bear	2	\$700				
7	John	Crow	2.2	\$150				
8	Steve	Carp	2.5	\$100				
9	Lesley	White Tiger	2	\$1,000				
10	Dayle	Puffin	1.5	\$50				
11								

To average the ages of all animals that are worth more than \$500, you could use the DAVERAGE function as follows:

=DAVERAGE(A3:D10,"Age",A1:D2)

*Database* is the range of cells that make up the database. A Microsoft Excel database is a contiguous range of cells organized into records (rows) and fields (columns). The *database* reference can be entered as a cell range, such as A3:D10 in the example above, or as a name assigned to a range. If you use the Data Set Database command on a selected range of cells, Microsoft Excel automatically names the range Database. If, for example, you had selected the range A3:D10 in the example above and chosen the Data Set Database command, you could use this formula:

=DAVERAGE(Database,"Age",A1:D2)

Remember that when you use a name as an argument, the name should not be enclosed within double quotation marks.

*Field* indicates which field is used in the function. Database fields are columns of data with an identifying field name in the first row. The *field* argument can be given as text, such as “Age” or “Value” in the example above, or as a field number: 1 for the first field (Name, in the example above), 2 for the second (Species), and so on.

*Criteria* is the range of cells that contains the database criteria. The *criteria* reference can be entered as a cell range, such as A1:D2 in the example above, or as a name assigned to a range. If you use the Data Set Criteria command on a selected range of cells, Microsoft Excel automatically names the range Criteria. If, for example, you had selected the range A1:D2 in the example above and chosen the Data Set Criteria command, you could use this formula:

=DAVERAGE(A3:D10,“Age”,Criteria)

**Tip**

To perform an operation on an entire column in a database, enter a blank line or lines below the field names in the criteria range.

Every database function operates on the values in the *field* column of records in the *database* that satisfy the *criteria*. The following list describes the different database functions:

Function	Description
DAVERAGE	Averages the values in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> .
DCOUNT	Counts the cells that contain numbers in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> .  In DCOUNT, the <i>field</i> argument is optional. If <i>field</i> is omitted, DCOUNT counts all records in the <i>database</i> that match the <i>criteria</i> .

Function	Description
DCOUNTA	Counts the cells that are not blank in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> . In DCOUNTA, the <i>field</i> argument is optional. If <i>field</i> is omitted, DCOUNTA counts all non-blank records in the <i>database</i> that match the <i>criteria</i> .
DMAX	Returns the largest number in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> .
DMIN	Returns the smallest number in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> .
DPRODUCT	Multiplies the values in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> .
DSTDEV	Estimates the standard deviation of a population based on a sample, using the numbers in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> .
DSTDEVP	Calculates the standard deviation of a population based on the entire population, using the numbers in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> .
DSUM	Adds the numbers in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> .
DVAR	Estimates the variance of a population based on a sample, using the numbers in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> .
DVARP	Calculates the variance of a population based on the entire population, using the numbers in the <i>field</i> column of records in the <i>database</i> that satisfy the <i>criteria</i> .

### Examples

A sample database of employee records. Each record contains information about one employee.

Database

Criteria

	A	B	C	D	E	F	H
1	<b>Name</b>	<b>Hire Date</b>	<b>Age</b>	<b>Sex</b>	<b>Income</b>		
2	Andrews, Jane	5/6/79	42 F		\$17,000		
3	Brown, Jessica	6/7/85	21 F		\$39,000		
4	Miller, Jim	12/1/70	64 M		\$38,000		
5	Sebring, Dave	9/23/84	34 M		\$18,000		
6	Smith, Cindy	2/1/83	34 F		\$21,000		
7							
8							
9	<b>Name</b>	<b>Hire Date</b>	<b>Age</b>	<b>Sex</b>	<b>Income</b>		
10		>1/1/84					
11							

DAVERAGE(Database,"Income",Criteria) equals \$28,500, the average Income for employees hired after 1/1/84 (Jessica Brown and Dave Sebring).

DAVERAGE(Database,3,Criteria) equals 27.5, the average Age for employees hired after 1/1/84.

DCOUNT(Database,"Income",Criteria) equals 2. This function looks at the records of employees hired after 1/1/84 and counts how many of the Income fields in those records contain numbers.

DCOUNT(Database,"Name",Criteria) equals 0. This function looks at the records of employees hired after 1/1/84 and counts how many of the Name fields in those records contain numbers.

DCOUNTA(Database,"Income",Criteria) equals 2. This function looks at the records of employees hired after 1/1/84 and counts how many of the Income fields in those records are not blank.

DMAX(Database,"Income",Database) equals \$39,000, the maximum Income value.

DMIN(Database,5,Criteria) equals \$18,000, the minimum Income for employees hired after 1/1/84.

A database for a small orchard. Each record contains information about one tree.

Database

Criteria

	A	B	C	D	E	F	G	H
1	Tree	Height	Age	Yield	Profit			
2	Apple	18	20	14	\$105.00			
3	Pear	12	12	10	\$96.00			
4	Cherry	13	14	9	\$105.30			
5	Apple	14	15	10	\$75.00			
6	Pear	9	8	8	\$76.80			
7	Apple	8	9	6	\$45.00			
8								
9	Tree	Height	Age	Yield	Profit	Height		
10	Apple	>10				<16		
11	Pear							
12								

DSUM(Database,"Profit",A9:A10) equals \$225.00, the total Profit from Apple trees.

DSUM(Database,"Profit",A9:F10) equals \$75.00, the total Profit from Apple trees with a Height between 10 and 16.

DPRODUCT(Database,"Yield",A9:F10) equals 10, the product of the Yields from Apple trees with a Height between 10 and 16.

DAVERAGE(Database,3,Database) equals 13, the average Age of all trees in the database.

DAVERAGE(Database,"Yield",A9:A11) equals 9.60, the average Yield for Apple and Pear trees.

DSTDEV(Database,“Yield”,A9:A11) *equals* 2.97, the estimated standard deviation in the Yield of the Apple and Pear trees if the data in the database is only a sample of the total orchard population.

DSTDEVP(Database,“Yield”,A9:A11) *equals* 2.65, the true standard deviation in the Yield of Apple and Pear trees if the data in the database is the entire population.

DVAR(Database,“Yield”,A9:A11) *equals* 8.80, the estimated variance in the Yield of the Apple and Pear trees if the data in the database is only a sample of the total orchard population.

DVARP(Database,“Yield”,A9:A11) *equals* 7.04, the true variance in the Yield of Apple and Pear trees if the data in the database is the entire orchard population.

### Related Functions

AVERAGE, COUNT, COUNTA, MAX, MIN, PRODUCT, STDEV, STDEVP, SUM, VAR, and VARP perform the same operations as the corresponding database functions. However, they operate on their lists of arguments, instead of on selected database entries.

### DMAX(database,field,criteria)

For information on Microsoft Excel database functions, see the section “Dfunction” earlier in this directory.

### DMIN(database,field,criteria)

For information on Microsoft Excel database functions, see the section “Dfunction” earlier in this directory.

### DOLLAR(number,decimals)

Rounds *number* to *decimals*, formats it in currency format, and returns the result as text. The format used is \$#,##0.00;(\$#,##0.00). *Decimals* is the number of digits to the right of the decimal point.

If *decimals* is negative, *number* is rounded to the left of the decimal point. If you omit *decimals*, it is assumed to be 2.

**Note** | The major difference between formatting a cell containing a number with the Format Number command and formatting a number directly with the DOLLAR function is that the DOLLAR function converts its result to text. A number formatted with the Format Number command is still a number.

## Examples

DOLLAR(1234.567,2) *equals* "\$1,234.57"

DOLLAR(1234.567, - 2) *equals* "\$1,200"

DOLLAR(- 1234.567, - 2) *equals* "(\$1,200)"

DOLLAR(- 0.123,4) *equals* "(\$0.1230)"

DOLLAR(99.888) *equals* "\$99.89"

## Related Functions

FIXED and TEXT format a number and convert it to text. VALUE converts a text argument into a number.

## DPRODUCT(database,field,criteria)

For information on Microsoft Excel database functions, see the section "*Dfunction*" earlier in this directory.

## DSTDEV(database,field,criteria)

For information on Microsoft Excel database functions, see the section "*Dfunction*" earlier in this directory.

## DSTDEVP(database,field,criteria)

For information on Microsoft Excel database functions, see the section "*Dfunction*" earlier in this directory.

## DSUM(database,field,criteria)

For information on Microsoft Excel database functions, see the section "*Dfunction*" earlier in this directory.

### DVAR(database,field,criteria)

For information on Microsoft Excel database functions, see the section “Dfunction” earlier in this directory.

### DVARP(database,field,criteria)

For information on Microsoft Excel database functions, see the section “Dfunction” earlier in this directory.

### EXACT(text1,text2)

Returns the logical value TRUE if *text1* and *text2* are exactly the same. If *text1* and *text2* are not identical (that is, exactly the same characters), EXACT returns the logical value FALSE.

#### Examples

EXACT(“word”,“word”) *equals* TRUE.

EXACT(“Word”,“word”) *equals* FALSE.

EXACT(“w ord”,“word”) *equals* FALSE.

#### Related Functions

LEN returns the length of text. SEARCH finds one text value within another.

### EXP(number)

Returns e (2.71828182845904) raised to the power of *number*. The constant e equals 2.71828182845904, the base of the natural logarithm.

To calculate powers of other bases, use the exponentiation operator (^).

EXP is the inverse of LN, the natural logarithm of *number*.

#### Examples

EXP(1) *equals* 2.71828182845904 (the value of e)

EXP(LN(3)) *equals* 3



## Related Functions

LN returns the natural logarithm (inverse of the EXP function). LOG returns the logarithm of a number to a specified base.

## FACT(number)

Returns the factorial of *number*. The factorial of *number* is equal to:

$$(number) * (number - 1) * (number - 2) * \dots * (number - n)$$

where  $number - n = 1$ . If *number* is not an integer, it is truncated.

## Examples

FACT(1) *equals* 1

FACT(1.9) *equals* FACT(1) *equals* 1

FACT(0) *equals* 1

FACT(-1) *equals* #NUM!

FACT(5) *equals* 5\*4\*3\*2\*1 *equals* 120

## Related Functions

PRODUCT multiplies all of its arguments together.

## FALSE()

Returns the logical value FALSE.

## Example

FALSE() *equals* FALSE

## FIND(find\_text,within\_text,start\_at\_num)

Finds *find\_text* within *within\_text*, starting the search at the character specified by *start\_at\_num*. The first character in *within\_text* is character number 1. FIND returns the number of the character at which *find\_text* first occurs. If *find\_text* does not appear within *within\_text*, if *start\_at\_num* is not greater than zero, or if *start\_at\_num* is greater than the length of *within\_text*, FIND returns the #VALUE! error value.

If you omit *start\_at\_num*, it is assumed to be 1. If *find\_text* is "", it will match the first character that is searched (that is, the character numbered *start\_at\_num* or 1). FIND is case-sensitive. *Find\_text* may not contain any wildcard characters.

### Examples

FIND("A","Average VanGun") equals 1

FIND("a","Average VanGun") equals 5

FIND("V","Average VanGun") equals 9

FIND("v","Average VanGun") equals 2

### Related Functions

SEARCH is just like FIND, except SEARCH is not case-sensitive and allows wildcard characters. EXACT checks to see if two text values are identical. LEN returns the length of text.

## FIXED(number,decimals)

Rounds *number* to *decimals*, formats it in decimal format using a period and commas, and returns the result as text. *Decimals* is the number of digits to the right of the decimal point.

If *decimals* is negative, *number* is rounded to the left of the decimal point. If you omit *decimals* it is assumed to be 2. In any case, numbers never have more than 15 significant digits, but *decimals* can be as large as 127.

### Note

The major difference between formatting a cell containing a number with the Format Number command and formatting a number directly with the FIXED function is that the FIXED function converts its result to text. A number formatted with the Format Number command is still a number.

## Examples

FIXED(1234.567,1) *equals* "1,234.6"

FIXED(1234.567, -1) *equals* "1,230"

FIXED( -1234.567, -1) *equals* " - 1,230"

FIXED(44.332) *equals* "44.33"

## Related Functions

DOLLAR and TEXT format a number and convert it to text. VALUE converts a text argument into a number.

## FV(rate,nper,pmt,pv,type)

Returns the future value of an investment based on periodic and constant payments and a constant interest rate. The optional *pv* and *type* arguments are assumed to be 0 if omitted.

For a complete description of the arguments in the FV function, see PV.

## Examples

FV(0.5%,10, - 200, - 500,1) *equals* 2,581.40

FV(1%,12, - 1000) *equals* 12,682.50

FV(11%/12,35, - 2000,,1) *equals* 82,846.25

## Related Functions

IPMT returns the interest payment for an investment for a given period. NPER returns the number of periods (payments) for an investment. PPMT returns the principal payment for an investment for a given period. PMT returns the periodic total payment for an investment. PV returns the present value of an investment. RATE returns the interest rate per period of an investment.

## GROWTH(known\_y's,known\_x's,new\_x's)

GROWTH fits an exponential curve to the data *known\_y's* and *known\_x's*. Then it returns the y-values along that curve for the array of *new\_x's* values that you specify.

The array *known\_x's* can include one or more sets of variables. If only one variable is used, *known\_y's* and *known\_x's* can be ranges of any shape as long as they have the same dimension. If more than one variable is used, *known\_y's* must be a vector (that is, a range with a height or width of 1). If the array *known\_y's* is in a single column, then each column of *known\_x's* is interpreted as a separate variable. If the array *known\_y's* is in a single row, then each row of *known\_x's* is interpreted as a separate variable.

If *new\_x's* is included, one dimension of that array must be the same as *known\_x's*. If *known\_y's* is in a single column, *known\_x's* and *new\_x's* should have the same number of columns. If *known\_y's* is in a single row, *known\_x's* and *new\_x's* should have the same number of rows. If you omit *new\_x's*, it is assumed to be the same as *known\_x's*. If you omit both *known\_x's* and *new\_x's*, they are assumed to be the array {1,2,3,...}, of the same size as *known\_y's*. If any of the numbers in *known\_y's* are negative, GROWTH returns the error value #NUM!.

For a discussion of the worksheet functions that fit lines and curves to data, see LINEST.

### Note

Formulas that return arrays must be entered as array formulas. To enter an array formula, press CONTROL+SHIFT+ENTER, or, with a mouse, press CONTROL+SHIFT while you click the check box in the formula bar. For information on arrays, see Array in *Microsoft Excel Reference*.

## Examples

Monthly sales values are entered in cells B2:B7 for months 1–6.

	A	B	C	D	E	F	G	H
1	Month	Sales	Growth					
2	1	\$11,232	\$11,255					
3	2	\$11,661	\$11,856					
4	3	\$12,090	\$12,070					
5	4	\$12,519	\$12,499					
6	5	\$12,948	\$12,943					
7	6	\$13,377	\$13,404					
8								
9	7		\$13,880					
10	8		\$14,371					
11	9		\$14,885					
12	10		\$15,414					
13	11		\$15,962					
14	12		\$16,530					
15								

To calculate the sales values for months 7–12, based on the preceding six months' data:

=GROWTH(B2:B7,A2:A7,A9:A14) equals the values in cells C9:C14.

To calculate the values along the calculated exponential curve for months 1–6, use:

=GROWTH(B2:B7) equals the values in cells C2:C7.

## Related Functions

LOGEST also calculates a regression curve, but it returns the parameters of that curve instead of an array of y-values along the curve. TREND and LINEST are analogous to GROWTH and LOGEST, but they fit your data to a straight line.

## HLOOKUP(lookup\_value,table\_array,row\_index\_num)

Looks in *table\_array* for a column whose first row contains *lookup\_value*, moves down the column according to *row\_index\_num*, and returns the value of the cell. A *row\_index\_num* of 1 returns the first row value in *table\_array*, a *row\_index\_num* of 2 returns the second row value in *table\_array*, and so on.

The values in the first row of *table\_array* can be text, numbers, or logical values. They must be placed in ascending order: . . . -2, -1, 0, 1, 2, . . . , A–Z, FALSE, TRUE; otherwise HLOOKUP may not give the correct value. Upper- and lowercase text are equivalent.

If HLOOKUP can't find the *lookup\_value*, it uses the largest value which is less than or equal to *lookup\_value*. If *lookup\_value* is smaller than the smallest value in the first row of *table\_array*, HLOOKUP returns the error value #N/A.

HLOOKUP returns the error value #VALUE! if *row\_index\_num* is less than 1, and #REF! if *row\_index\_num* is greater than the number of rows in *table\_array*.

## Examples

	A	B	C	D	E	F	G	H
1								
2		0	50	60	70	80	90	Absent
3	F	F	D	C	B	A	F	
4	F	E	D	C	B	A	Unassigned	
5								

In the worksheet above:

HLOOKUP(82,A2:G4,2) equals "B"

HLOOKUP(55,A2:G4,3) equals "E"

HLOOKUP(55,A2:G4,2) equals "F"

HLOOKUP("absent",A2:G4,3) equals "Unassigned"

*Table\_array* can also be an array constant:

HLOOKUP(3,{1,2,3;"a","b","c";"d","e","f"},2) equals "c"

### Related Functions

VLOOKUP looks in the first column of an array and moves across the row to return the value of a cell. LOOKUP and MATCH also look up values in an array or reference. INDEX uses an index to return a value from a reference or array.

## HOUR(serial\_number)

Returns the hour corresponding to *serial\_number*. The hour is given as an integer, ranging from 0 (12:00 A.M.) to 23 (11:00 P.M.).

*Serial\_number* is the date-time code used by Microsoft Excel for date and time calculations. It ranges from 0 to 65380. The numbers to the right of the decimal point in *serial\_number* represent the time; the numbers to the left represent the date. For example, the date-time combination 12:00 P.M., January 1, 1901 is represented as 367.5.

*Serial\_number* may be given as text, such as "16:48:00" or "4:48:00 PM," instead of a number. The text is automatically converted to a serial number.

### Note

This description assumes that the 1904 Date System check box in the Options Calculation dialog box is turned off. If this check box is turned on, then serial number 1 in the active document is the start of day January 2, 1904, instead of the start of day January 1, 1900. The 1904 Date System box is turned on automatically if you open a document from Microsoft Excel for the Macintosh, which has a different date system. You can also turn on the check box yourself if you are creating a document to be used in Microsoft Excel for the Macintosh. For more information, see Date in *Microsoft Excel Reference*.

## Examples

HOUR(0.7) *equals* 16

HOUR(29747.7) *equals* 16

HOUR("3:30:30 PM") *equals* 15

## Related Functions

YEAR, MONTH, DAY, WEEKDAY, MINUTE, and SECOND convert serial numbers into years, months, days, weekdays, minutes, or seconds. NOW returns the serial number of the current date and time.

## IF(logical\_test,value\_if\_true,value\_if\_false)

Returns *value\_if\_true* if *logical\_test* is TRUE, and *value\_if\_false* if *logical\_test* is FALSE. If omitted, *value\_if\_false* is assumed to be FALSE.

Use IF to make conditional tests of cell values and formulas. The outcome of the *logical\_test* determines the value returned by the IF function. The *value\_if\_true* and *value\_if\_false* arguments can be any value.

Up to seven IF functions can be nested as *value\_if\_true* and *value\_if\_false* arguments to construct more elaborate tests. See the last example for a demonstration.

### Macro Note

If you are using IF in a macro, *value\_if\_true* and *value\_if\_false* can also be GOTO functions or action-taking functions. For example, the following functions are allowed in a macro:

= IF(number>10,GOTO(large),GOTO(small))

= IF(file="chart",NEW(2),NEW(1))

If any of the arguments to IF are arrays, every argument is evaluated when the IF statement is executed. If some of the arguments are action-taking functions, all of the actions are taken. For example, the following formula displays two messages:

= IF({TRUE,FALSE},ALERT("One",2),ALERT("Two",2))

## Examples

	A	B	C	D	E	F
1		<b>Actual Expenses</b>	<b>Predicted Expenses</b>			
2	January	1500	900			
3	February	500	900			
4	March	500	925			
5	April	600	925			
6						

IF(B2>C2,“Over Budget”,“OK”) equals “Over Budget”

IF(B3>C3,“Over Budget”,“OK”) equals “OK”

Suppose you want to assign letter grades to numbers referenced by the name Avg (average):

If Avg is	Then return
Greater than 90	“A”
Between 80 and 90	“B”
Between 70 and 80	“C”
Between 60 and 70	“D”
Less than 60	“F”

You could use the following nested IF function:

IF(Avg>90,“A”,IF(Avg>80,“B”,IF(Avg>70,“C”,IF(Avg>60,“D”,“F”))))

## Related Functions

AND is TRUE if all arguments are TRUE. OR is TRUE if one or more arguments are TRUE. NOT reverses the logic of its argument.



## INDEX(ref,row\_num,column\_num,area\_num)

The INDEX function has two forms, reference and array. The reference form always returns a reference; the array form always returns a value or array of values. This section discusses the reference form; the next section discusses the array form.

The reference form of the INDEX function returns the reference of the cell or cells determined by *ref*, *row\_num*, *column\_num*, and *area\_num*.

*Ref* is a reference to one or more cell ranges; *area\_num* selects the range. The first area selected or entered is numbered 1, the second is 2, and so on. For example, if *ref* describes the cells (A1:B4,D1:E4,G1:H4), then *area\_num* 1 is the range A1:B4, *area\_num* 2 is the range D1:E4, and *area\_num* 3 is the range G1:H4. If you are entering a multiple selection for *ref*, enclose the *ref* in parentheses. If *area\_num* is omitted, INDEX uses area 1. For an example of using INDEX with a multiple selection, see the fifth example.

If each area in *ref* contains only one row or column, the *row\_num* or *column\_num* argument, respectively, is optional. For a single row *ref*, use INDEX(*ref*,*column\_num*). For a single column *ref*, use INDEX(*ref*,*row\_num*).

After *ref* and *area\_num* have selected a particular range, *row\_num* and *column\_num* select a particular cell: *row\_num* 1 is the first row in the range, *column\_num* 1 is the first column, and so on. Or, you can set *column\_num* and/or *row\_num* to 0; in this case, INDEX returns the reference for the entire row and/or column, respectively.

*Row\_num*, *column\_num*, and *area\_num* must point to a cell within *ref*; otherwise, INDEX gives the error value #REF!.

The result of the INDEX function is a reference and is interpreted by other formulas as such. Depending on the formula, the return value of INDEX may be used as a reference or as a value.

For example, the formula:

=CELL("width",INDEX(A1:B2,1,2))

is equivalent to

=CELL("width",B1).

The CELL function uses the return value of INDEX as a cell reference. On the other hand, the formula =2\*INDEX(A1:B2,1,2) translates the return value of INDEX into the number in cell B1.

## Examples

	A	B	C	D	E	F	G	H
1		Price	Count (lbs)					
2	Apples	\$0.69	40					
3	Bananas	\$0.34	38					
4	Lemons	\$0.55	15					
5	Oranges	\$0.25	25					
6	Pears	\$0.59	40					
7								
8	Almonds	\$2.80	10					
9	Cashews	\$3.55	16					
10	Peanuts	\$1.25	20					
11	Walnuts	\$1.75	12					
12								

Stock

Nuts

Fruit

INDEX(Fruit,2,3) equals the reference C3, containing 38

INDEX(Nuts,1,1) equals the reference A8, containing "Almonds"

INDEX(Fruit,2,4) equals #REF!, because the column argument (4) is out of range

INDEX(Stock,1,2) equals B2, containing "Price"

INDEX(Stock,2,2,2) equals INDEX((A1:C6,A8:C11),2,2,2) equals the reference B9, containing \$3.55

SUM(INDEX(Stock,0,3,1)) equals SUM(C1:C6) equals 158

SUM(B2:INDEX(Fruit,5,2)) equals SUM(B2:B6) equals 2.42

## INDEX(array,row\_num,column\_num)

The INDEX function has two forms, reference and array. The reference form always returns a reference; the array form is used when the first argument to INDEX is an array constant, it always returns a value or array of values. This section discusses the array form; the previous section discusses the reference form.

The array form of the INDEX function returns the value of an element in array, selected by the row\_num and column\_num indexes.

Column\_num selects the array column: 1 selects the first column, 2 selects the second, and so on. Row\_num selects the array row.

If array contains only one row or column, the row\_num or column\_num argument is optional. For example, use INDEX(array,column\_num) for a single row array. For a single column array, use INDEX(array,row\_num).

Generally, *row\_num* and *column\_num* will be positive integers; for the special case where either value is 0, INDEX returns the array of values for the entire row or column:

INDEX(*array*,0,*column\_num*) returns the array containing values from *column\_num*.

INDEX(*array*,*row\_num*,0) returns the array containing values from *row\_num*.

*Row\_num* and *column\_num* must point to a cell within *array*; otherwise, INDEX returns the error value #REF!.

## Examples

INDEX({1,2;3,4},2,2) equals 4

If entered as an array formula, then:

INDEX({1,2;3,4},0,2) equals {2;4}

## Related Functions

CHOOSE uses an index to select a value from its other arguments.

HLOOKUP, VLOOKUP, LOOKUP, and MATCH look up values in an array or reference.

## INDIRECT(*ref\_text*,*type\_of\_ref*)

Returns the reference specified by *ref\_text*.

The cell indicated by *ref\_text* must contain a name assigned to a cell, an A1 reference, or an R1C1 reference. *Type\_of\_ref* is a logical value that specifies what type of reference is contained in the cell *ref\_text*. If *type\_of\_ref* is FALSE, the reference contained in *ref\_text* is interpreted as the R1C1 style. If *type\_of\_ref* is TRUE or omitted, the reference contained in *ref\_text* is interpreted as the A1 style.

If *ref\_text* is not a valid cell reference, INDIRECT gives the error value #REF!.

## Examples

If cell A1 contains the text "B2", and cell B2 contains the value 1.333, then:

INT (INDIRECT(A1)) equals 1

If A1 contains “R2C2”, then:

INT(INDIRECT(R1C1,FALSE)) *equals* 1

### Related Functions

CELL, the IS functions, and TYPE all return information about a cell.

## INT(number)

Rounds *number* down to the nearest integer.

### Examples

INT(8.9) *equals* 8

INT(−8.9) *equals* −9

### Related Functions

TRUNC truncates a number to an integer. ROUND rounds a number to a specified number of digits. MOD gives the remainder from division.

## IPMT(rate,per,nper,pv,fv,type)

Returns the interest payment for a given period for an investment based on periodic, constant payments and a constant interest rate. The optional *f<sub>v</sub>* and *type* arguments are assumed to be zero if omitted.

*Per* specifies the period, and must be in the range 1 to *nper*. For a complete description of the other arguments in IPMT, see PV.

### Note

In the annuity functions, cash you pay out, such as deposits to savings, is represented by negative numbers; cash you receive, such as dividend checks, is represented by positive numbers.

For example, a \$1000 deposit to the bank would be represented by the argument −1000 if you were the depositor, and by the argument 1000 if you were the bank.

## Examples

The following IPMT function finds the interest due in the first month of a 3-year \$8000 loan at 10% annual interest:

`IPMT(0.1/12,1,36,8000)` equals –\$66.67

The following function finds the interest due in the last year of a 3-year \$8000 loan at 10% annual interest, where payments are made yearly:

`IPMT(0.1,3,3,8000)` equals –\$292.45

## Related Functions

FV returns the future value of an investment. NPER returns the number of periods (payments) for an investment. PPMT returns the principal payment for an investment for a given period. PMT returns the periodic total payment for an investment. PV returns the present value of an investment. RATE returns the interest rate per period of an investment.

## IRR(values,guess)

Returns the internal rate of return for a series of periodic cash flows represented by the numbers in *values*. These cash flows do not have to be even, as they would be for an annuity. The internal rate of return is the interest rate received for an investment consisting of payments (negative values) and income (positive values) that occur at regular periods.

*Values* must be an array or a reference to cells that contain numbers. There must be at least one positive value and one negative value in order to calculate the internal rate of return.

IRR uses the order of *values* to interpret the order of cash flows. Be sure to enter your payment and income values in the correct sequence. Text, logical, and empty cell *values* are ignored.

Microsoft Excel uses an iterative technique for calculating IRR. *Guess* is a number that you guess is close to the result of IRR. Starting with *guess*, IRR cycles through the calculation until the result is accurate within .00001%. If after 20 tries it can't find a result that works, IRR returns the error value #NUM!.

In most cases you will not need to provide *guess* for the IRR calculation. If omitted, *guess* is assumed to be 0.1 (10%). However, if IRR gives the error value #NUM!, or if the result is not close to what you expected, try again with different values of *guess*.

IRR is closely related to the net present value function, NPV. The rate of return calculated by IRR is the interest rate corresponding to a “zero” net present value.

### Examples

Suppose you’re interested in starting a restaurant business. You estimate it will cost \$70,000 to start the business and expect to net the following income in the first five years: \$12,000, \$15,000, \$18,000, \$21,000, and \$26,000.

	A	B	C	D	E	F	G	H
1	Initial Cost	(\$70,000)						
2	Year 1 Income	\$12,000						
3	Year 2 Income	\$15,000						
4	Year 3 Income	\$18,000						
5	Year 4 Income	\$21,000						
6	Year 5 Income	\$26,000						

To calculate the investment’s internal rate of return after four years, use:

`IRR(B1:B5)` equals  $-2.12\%$

To calculate the internal rate of return after five years, use:

`IRR(B1:B6)` equals  $8.66\%$

To calculate the internal rate of return after two years, you’ll need to include a *guess*:

`IRR(B1:B3, -10%)` equals  $-44.35\%$

To demonstrate how NPV and IRR are related, use:

`NPV(IRR(B1:B5),B1:B5)` equals  $3.60E-08$

Within the accuracy of the IRR calculation, the value  $3.60E-08$  is effectively 0.

### Related Functions

MIRR gives the internal rate of return where positive and negative cash flows are financed at different rates. NPV gives the net present value of an investment based on cash flows that do not have to be constant. RATE gives the interest rate for an investment based on constant cash flows.

## IS Functions

### ISfunction(value)

This section describes the nine worksheet functions used for testing the type of a value or reference. Each of these functions, referred to collectively as ISfunction, checks the type of *value* and returns TRUE or FALSE depending upon the outcome. For example, the ISBLANK function returns the logical value TRUE if *value* is a reference to an empty cell; otherwise it returns FALSE.

The *value* arguments to the ISfunctions are *not* translated. For example, in most cases where a number is required, the text value “19” is translated into the number 19. However, in the formula =ISNUMBER(“19”), “19” is not translated from a text value and the ISNUMBER function returns FALSE.

The value test functions are most useful in formulas and macros for testing the outcome of a calculation. When combined with the IF function, they provide a method for locating errors in formulas.

For example, suppose you want to calculate the average of the range A1:A4, but you can’t be sure that the cells contain numbers. The formula =AVERAGE(A1:A4) gives the error value #DIV/0! if A1:A4 does not contain numbers. To allow for this case, you could use the following formula to locate potential errors:

=IF(ISERROR(AVERAGE(A1:A4)),“No Numbers”, AVERAGE(A1:A4))

Function	Description
ISBLANK	Returns TRUE if <i>value</i> refers to an empty cell; gives FALSE otherwise
ISERR	Returns TRUE if <i>value</i> is any Microsoft Excel error value except #N/A; gives FALSE otherwise
ISERROR	Returns TRUE if <i>value</i> is any Microsoft Excel error value: #N/A, #VALUE!, #REF!, #DIV/0!, #NUM!, #NAME?, or #NULL!; gives FALSE otherwise
ISLOGICAL	Returns TRUE if <i>value</i> is a logical value; gives FALSE otherwise
ISNA	Returns TRUE if <i>value</i> is the error value #N/A (no value available); gives FALSE otherwise

Function	Description
ISNONTEXT	Returns TRUE if <i>value</i> is not text; gives FALSE otherwise
ISNUMBER	Returns TRUE if <i>value</i> is a number; gives FALSE otherwise
ISREF	Returns TRUE if <i>value</i> is a reference; gives FALSE otherwise
ISTEXT	Returns TRUE if <i>value</i> is text; gives FALSE otherwise

### Examples

	A	B	C	D	E	F	G	H
1		#N/A	0					
2	Hello		#REF!					
3	Price	\$1.99						
4								

*Name of cell is Price*

ISBLANK(C1) *equals* FALSE

ISERROR(C2) *equals* TRUE

ISLOGICAL(TRUE) *equals* TRUE

ISLOGICAL("TRUE") *equals* FALSE

ISNA(C2) *equals* FALSE

ISNA(B1) *equals* TRUE

ISNUMBER(B3) *equals* TRUE (as long as the \$1.99 was entered as a number, and not as text)

ISREF(Price) *equals* TRUE (if Price is defined as a range name)

ISTEXT(A2) *equals* TRUE

### Related Functions

CELL gives information about the formatting, location, or contents of a cell. TYPE returns a number indicating the data type of a value.



## LEFT(text,number\_of\_characters)

Returns the first (or leftmost) *number\_of\_characters* in *text*. *Number\_of\_characters* must be greater than zero. If *number\_of\_characters* is greater than the length of *text*, LEFT returns all of *text*. If *number\_of\_characters* is omitted, it is assumed to be 1.

### Examples

LEFT("Paul Irving",4) *equals* "Paul"

If A1 contains "abalone", then:

LEFT(A1) *equals* "a"

### Related Functions

RIGHT extracts the rightmost characters from a text value. MID extracts characters based on a starting position and number of characters.

## LEN(text)

Gives the number of characters in *text*.

### Examples

LEN("Phoenix, AZ") *equals* 11

LEN("") *equals* 0

### Related Functions

EXACT checks to see if two text values are identical. SEARCH finds one text value within another.

## LINEST(known\_y's,known\_x's)

Calculates a straight line ( $m_1x_1 + m_2x_2 + \dots + b$ ) that fits your data, and returns an array  $\{m_1, m_2, \dots, b\}$  that describes that line.

The arguments to LINEST are a set of known y-values and, optionally, a set of known x-values. The arguments are both arrays.

The array *known\_x's* can include one or more sets of variables. If only one variable is used, *known\_y's* and *known\_x's* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, *known\_y's* must be a vector (that is, a range with a height or width of 1). If the array *known\_y's* is in a single column, then each column of *known\_x's* is interpreted as a separate variable. If the array *known\_y's* is in a single row, then each row of *known\_x's* is interpreted as a separate variable.

If *known\_x's* is omitted, it is assumed to be the array {1,2,3,...}, of the same size as *known\_y's*.

You can describe any straight line with these two numbers:

■ **Slope (m)**

To find the slope of a line, often written as *m*, take two points on the line, (*x*<sub>1</sub>,*y*<sub>1</sub>) and (*x*<sub>2</sub>,*y*<sub>2</sub>). The slope is equal to (*y*<sub>2</sub> - *y*<sub>1</sub>)/(*x*<sub>2</sub> - *x*<sub>1</sub>). In this case, the slope of the line is \$1000/month.

■ **Y-intercept**

The *y*-intercept of a line, often written as *b*, is the value of *y* at the point where the line crosses the *y*-axis. In this case, the *y*-intercept is equal to \$2000.

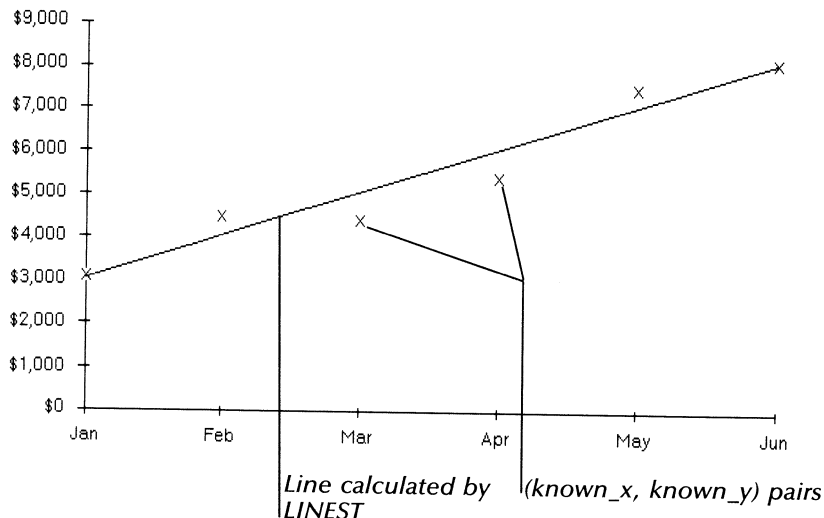
The equation of a straight line is  $y = mx + b$ . Once you know the values of *m* and *b*, you can calculate any point on the line by plugging the *y*- or *x*-value into that equation.

Let's take a look at an example using only one variable.

This worksheet shows your monthly sales for the first six months of the year.

	A	B	C	D	E	F	G	H
1	<b>Month</b>	<b>Sales</b>						
2	January	\$3,100						
3	February	\$4,500						
4	March	\$4,400						
5	April	\$5,400						
6	May	\$7,500						
7	June	\$8,100						
8								

LINEST calculates this line from the sales figures.



To use the LINEST function to find m and b for this example, use:

=LINEST(B2:B7), which equals {1000,2000}

You don't have to specify *known\_x's* because the x-values are the same as the default *known\_x's* ({1,2}).

The slope value is given by:

INDEX(LINEST(A1:F1),1) equals 1000

The y-intercept value is given by:

INDEX(LINEST(A1:F1),2) equals 2000

The accuracy of the line calculated by LINEST depends on the degree of scattering in the data that you give. The more linear the data, the more accurate the LINEST model. LINEST uses the method of least squares for determining the best fit for the data. The calculations for m and b are based on the following formulas:

$$m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

$$b = \frac{(\sum y)(\sum x^2) - \sum x(\sum xy)}{n(\sum x^2) - (\sum x)^2}$$

$$= \text{AVERAGE}(\text{known\_y's}) - m * \text{AVERAGE}(\text{known\_x's})$$

**Note** | The line- and curve-fitting functions can calculate the best straight line or exponential curve to fit your data, but you have to decide which of the two best fits your data. To see if the calculated line or curve fits your data:

- 1 Calculate TREND(known\_y's,known\_x's) — for a straight line — or GROWTH(known\_y's,known\_x's) — for an exponential curve. These functions, without any new\_x's argument, return an array of y-values predicted along that line, or curve, at your actual data points.
- 2 Compare the predicted values with the actual values. You may want to chart them both for a visual comparison.

**Note** | Formulas that return arrays must be entered as array formulas. To enter an array formula, press CONTROL+SHIFT+ENTER, or, with a mouse, press CONTROL+SHIFT while you click the check box in the formula bar. For information on arrays, see Array in *Microsoft Excel Reference*.

### Examples

LINEST({1,9,5,7},{0,4,2,3}) equals {2,1}, the slope = 2, y-intercept = 1

You can use the slope m and y-intercept b in the formula  $y = m * x + b$  for estimating new y-values for a given x-value:

SUM({m,b}\*{x,1}) equals  $m * x + b$ , the estimated y-value, for a given x-value.

	A	B	C	D	E	F	G	H
1	Month	Sales						
2	January	\$3,100						
3	February	\$4,500						
4	March	\$4,400						
5	April	\$5,400						
6	May	\$7,500						
7	June	\$8,100						
8								

For example, to estimate sales for the ninth month based on the sales in months 1–6:

$\text{SUM}(\text{LINEST}(\text{B2:B7})*\{9,1\})$  equals  $\text{SUM}(\{1000,2000\}*\{9,1\})$  equals \$11,000

	A	B	C	D	E	F	G	H
1	Age	Height	Weight					
2	26	65	133					
3	19	72	140					
4	38	69	130					
5	62	70	160					
6								

In the worksheet above:

$=\text{LINEST}(\text{C2:C5},\text{A2:B5})$  equals  $\{1.55,0.51,15.18\}$

## Related Functions

TREND also calculates a straight line, but it returns an array of y-values instead of the parameters of the line. GROWTH and LOGEST are analogous to TREND and LINEST, but they fit an exponential curve to your data instead of a straight line.

### LN(number)

Returns the natural logarithm of *number*. Natural logarithms are the logarithms based on the constant *e* (2.71828182845904).

*Number* must be positive.

LN is the inverse of the EXP function, *e* raised to the power *number*.

### Examples

LN(86) *equals* 4.45

LN(2.7182818) *equals* 1

LN(EXP(3)) *equals* 3

EXP(LN(4)) *equals* 4

### Related Functions

EXP returns *e* raised to a given power (inverse of the LN function). LOG returns the logarithm of a number to a specified base. LOG10 returns the base 10 logarithm of a number.

### LOG(number,base)

Returns the logarithm of *number* to *base*.

*Number* must be positive. If *base* is omitted, it is assumed to be 10.

### Examples

LOG(10) *equals* 1

LOG(8,2) *equals* 3

LOG(86,2.7182818) *equals* 4.45

### Related Functions

LOG10 returns the base 10 logarithm of a number. LN returns the natural logarithm of a number. EXP returns *e* raised to a given power.

## LOG10(number)

Returns the base 10 logarithm of *number*.

*Number* must be positive.

### Examples

LOG10(86) equals 1.93

LOG10(10) equals 1

LOG10(1E5) equals 5

LOG10(10^5) equals 5

### Related Functions

LOG returns the logarithm of a number to a specified base. LN returns the natural logarithm of a number. EXP returns e raised to a given power.

## LOGEST(known\_y's,known\_x's)

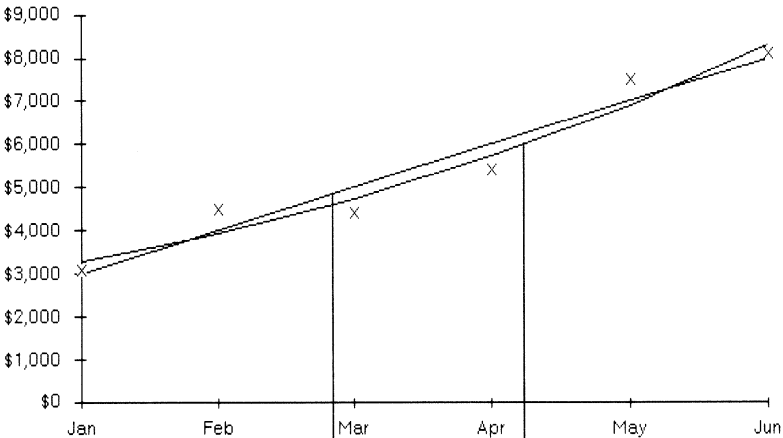
Fits an exponential curve ( $b \cdot (m_1^{x_1}) \cdot (m_2^{x_2})$ ) to the array's *known\_y's* and *known\_x's*, and returns the values *m* and *b* as elements in a horizontal array {*m*<sub>1</sub>,*m*<sub>2</sub>,...*b*}.

The array *known\_x's* can include one or more sets of variables. If only one variable is used, *known\_y's* and *known\_x's* can be ranges of any shape as long as they have equal dimensions. If more than one variable is used, *known\_y's* must be a vector (that is, a range with a height or width of 1). If the array *known\_y's* is in a single column, then each column of *known\_x's* is interpreted as a separate variable. If the array *known\_y's* is in a single row, then each row of *known\_x's* is interpreted as a separate variable.

If you omit *known\_x's*, LOGEST uses the values {1,2,3,...}, in an array the same size as *known\_y's*.

The closer your data resembles an exponential curve, the better the calculated line will fit your data.

Like LINEST, LOGEST returns an array of values that describe a curve, but LOGEST fits a different curve to your data. LINEST fits a straight line to your data; LOGEST fits an exponential curve.



For the same data,

LINEST fits  
this line

LOGEST fits  
this curve

**Note** Formulas that return arrays must be entered as array formulas. To enter an array formula, press CONTROL+SHIFT+ENTER, or, with a mouse, press CONTROL+SHIFT while you click the check box in the formula bar. For information on arrays, see Array in *Microsoft Excel Reference*.

### Examples

LOGEST({1,3.2,10,100},{0,0.5,1,2}) equals {9.98,1.00} (m = 9.98, b = 1.00)

	A	B	C	D	E	F	G	H
1	Time	Population						
2	0	3						
3	1	10						
4	2	38						
5	2.5	62						
6	3	110						
7	3.5	353						
8								

In the worksheet above:

LOGEST(B2:B7,A2:A7) equals {3.68,2.77}

If you want just the value m or b, you can use the INDEX function. The value m is given by:

INDEX(LOGEST(B2:B7,A2:A7),1) equals 3.68

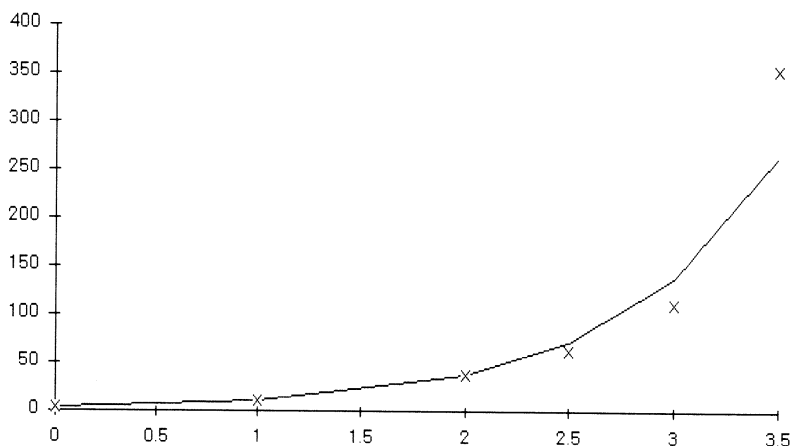


The value b is given by:

`INDEX(LOGEST(B2:B7,A2:A7),2)` equals 2.77

Suppose you wanted to estimate the population at time 4. First, check whether the exponential curve estimated by LOGEST fits your data. Plot your data with the values returned by this array formula:

`GROWTH(B2:B7,A2:A7)`



To find the value at time = 4 on the exponential curve, use the y and m values returned by LOGEST to calculate the formula  $y = b * m^x$ :

`INDEX(LOGEST(B2:B7,A2:A7),2)*INDEX(LOGEST(B2:B7,A2:A7),1)^4`  
equals 505

## Related Functions

GROWTH also calculates an exponential curve, but it returns an array of y-values instead of the parameters of the curve. TREND and LINEST are analogous to GROWTH and LOGEST, but they fit a straight line instead of an exponential curve to your data.

## LOOKUP(lookup\_value,lookup\_vector,result\_vector)

The LOOKUP function has two forms, vector and array. This section discusses the vector form; the next section discusses the array form.

A **vector** is an array that contains only one row or one column. The vector form of LOOKUP looks in *lookup\_vector* for *lookup\_value*, moves to the corresponding position in *result\_vector* and gives this value.

If LOOKUP can't find the *lookup\_value*, it uses the largest value which is less than or equal to *lookup\_value*. If *lookup\_value* is smaller than the smallest value in *lookup\_vector*, LOOKUP gives the error value #N/A.

The values in *lookup\_vector* can be text, numbers, or logical values. They must be placed in ascending order: . . . -2, -1, 0, 1, 2, . . . , A-Z, FALSE, TRUE; otherwise LOOKUP may not give the correct value. Upper- and lowercase text are equivalent.

### Examples

	A	B	C	D	E	F	G	H
1	Frequency	Color						
2	4.54E+14	red						
3	4.19E+14	orange						
4	5.17E+14	yellow						
5	5.77E+14	green						
6	6.38E+14	blue						
7	7.31E+14	violet						
8								

In the worksheet above:

LOOKUP(4.91E14,A2:A7,B2:B7) equals "orange"

LOOKUP(5.00E14,A2:A7,B2:B7) equals "orange"

LOOKUP(7.66E14,A2:A7,B2:B7) equals "violet"

LOOKUP(7.66E - 14,A2:A7,B2:B7) equals #N/A, because 7.66E - 14 is less than the smallest value in the *lookup\_vector* A2:A7

### Related Functions

HLOOKUP, VLOOKUP, and MATCH also look up values in a reference or array.

## LOOKUP(lookup\_value,array)

The LOOKUP function has two forms, vector and array. This section discusses the array form; the previous section discusses the vector form.

The array form of LOOKUP looks in the first row or column of *array* for *lookup\_value*, moves down or across to the last cell and gives the value of the cell.

The array form of LOOKUP is very similar to the HLOOKUP and VLOOKUP functions. The difference is that HLOOKUP searches for *lookup\_value* in the first row, VLOOKUP searches in the first column, and LOOKUP searches according to the dimensions of *array*. If *array* is square, or covers an area that is wider than it is tall (more columns than rows), LOOKUP searches for *lookup\_value* in the first row; if *array* is taller than wide (more rows than columns), LOOKUP searches in the first column. Another difference is that HLOOKUP and VLOOKUP allow you to index down or across, but LOOKUP always selects the last value in the row or column.

The values in the LOOKUP search range (first row or column) can be text, numbers, or logical values. They must be placed in ascending order:  $\dots - 2, -1, 0, 1, 2, \dots$ , otherwise, LOOKUP may not give the correct value. Upper- and lowercase are equivalent.

If LOOKUP can't find the *lookup\_value*, it uses the largest value which is less than or equal to *lookup\_value*. If *lookup\_value* is smaller than the smallest value in the first row or column (depending on the *array* dimensions), LOOKUP gives the error value #N/A.

In general, it's best to use the HLOOKUP or VLOOKUP function in place of the array form of LOOKUP. This form of LOOKUP is provided for compatibility with other worksheet programs.

## Related Functions

HLOOKUP and VLOOKUP look in an array and move down or across, respectively, to give cell contents. INDEX uses an index to choose a value from a reference or array.

## Examples

LOOKUP("C",{ "a","b","c","d";1,2,3,4}) equals 3

LOOKUP("bump",{ "a",1;"b",2;"c",3}) equals 2

## LOWER(text)

Converts all uppercase letters in *text* to lowercase. Characters in *text* that are not letters are unchanged.

## Related Functions

PROPER capitalizes the first letter in each word of a text value. UPPER capitalizes every letter in a text value.

## Examples

LOWER("E. E. Cummings") *equals* "e. e. cummings"

LOWER("Apt. 2B") *equals* "apt. 2b"

## MATCH(lookup\_value,lookup\_array,type\_of\_match)

Returns the relative position of the element in *lookup\_array* that matches the *lookup\_value*, according to the *type\_of\_match*. The three different types of matching that you can choose are:

Type_of_match	MATCH finds	Lookup_array
1, or omitted	The largest value that is less than or equal to <i>lookup_value</i>	Must be placed in ascending order: ... -2, -1, 0, 1, 2, ... A-Z, FALSE, TRUE
-1	The smallest value that is greater than or equal to <i>lookup_value</i>	Must be placed in descending order: TRUE, FALSE, Z-A, ... 2, 1, 0, -1, -2, ...
0	The first value that is exactly equal to <i>lookup_value</i>	Can be in any order

If *type\_of\_match* is omitted, it is assumed to be 1. MATCH gives the position of the matched value within *lookup\_array*, not the value itself. For example, MATCH("b",{ "a","b","c"},0) gives the value 2, the relative position of "b" within the array {"a","b","c"}.

*Lookup\_value* can be a value (number, text, logical value) or a cell reference. *Lookup\_array* can be an array or an array reference. MATCH does not distinguish between upper- and lowercase letters when matching text values. If MATCH is unsuccessful in finding a match, it gives the error value #N/A.

Note

If *type\_of\_match* is 0, and *lookup\_value* is text, *lookup\_value* can contain the wildcard characters, asterisk (\*) and question mark (?). MATCH will look for text with any single character in the question mark (?) position and any string of characters in the asterisk (\*) position.

## Examples

	A	B	C	D	E	F	G	H
1	100	Hot			1	Line		
2	80	Warm			2	Angle		
3	60	Mild			3	Triangle		
4	40	Chilly			4	Square		
5	20	Cold			5	Pentagon		
6	0	Very Cold			6	Hexagon		
7	-20	Frigid			7	Heptagon		
8								

In the worksheet above:

MATCH(70,A1:A7,-1) equals 2

MATCH(-25,A1:A7,-1) equals 7

MATCH(60,A1:A7,0) equals 3

MATCH(60,A1:A7,1) equals #N/A, because the range A1:A7 is ordered incorrectly for type 1 matching

MATCH("Balmy",B1:B7,0) equals #N/A, because the value "Balmy" cannot be found in the range B1:B7

MATCH("Cold",B1:B7,0) equals 5

MATCH("cOLD",B1:B7,0) equals 5

MATCH("c\*",B1:B7,0) equals 4, because MATCH looks for the first text beginning with "c"

MATCH(3.5,D1:D7) equals 3

MATCH("he?agon",E1:E7,0) equals 6

MATCH("he??agon",E1:E7,0) equals 7

## Related Functions

HLOOKUP, VLOOKUP, and LOOKUP all look up values in a reference or array. INDEX uses an index to choose a value from a reference or array.

## MAX(number1,number2,...)

Returns the largest number in the list of arguments.

MAX can have from 1 to 14 arguments. Arguments that are numbers, empty cells, logical values, or text representations of numbers are used; arguments that are error values or text that cannot be translated into numbers cause errors. If an argument is an array or reference, only numbers in that array or reference are used. Empty cells, logical values, text, or error values in the array or reference are ignored.

### Examples

If A1:A5 contains the numbers 10, 7, 9, 27, and 2, then:

MAX(A1:A5) *equals* 27

MAX(A1:A5,30) *equals* 30

### Related Functions

DMAX returns the maximum value from selected database entries. MIN returns the minimum value from its list of arguments.

## MDETERM(array)

Returns the matrix determinant of *array*, where *array* must be a numeric array with an equal number of rows and columns.

*Array* can be given as a cell range, for example A1:C3, or as an array constant, such as {1,2,3;4,5,6;7,8,9}. If cells are empty or contain text, MDETERM returns the error value #VALUE!. MDETERM also returns the error value #VALUE! if *array* is not a square-shaped array.

The **matrix determinant** is a number derived from the values in *array*. For a 3-row, 3-column array, A1:C3, the determinant is defined as:

MDETERM(A1:C3) *equals*

$$\begin{aligned} &A1*(B2*C3 - B3*C2) \\ &+ A2*(B3*C1 - B1*C3) \\ &+ A3*(B1*C2 - B2*C1) \end{aligned}$$

Matrix determinants are generally used for solving systems of mathematical equations that involve several variables.

Examples

	A	B	C	D	E	F	G	H
1	0	0	0	0				
2	0	1	0	2				
3	0	1	1	0				
4	0	4	0	1				
5								

In the worksheet above:

MDETERM(B2:D4) *equals* -7

MDETERM(A1:D4) *equals* 0

MDETERM(B2:C3) *equals* 1

MDETERM(A1:D2) *equals* #VALUE!, because the range A1:D2 is not a square-shaped array.

MDETERM(A1:E5) *equals* #VALUE!, because A1:E5 contains empty cells.

Related Functions

MINVERSE returns the matrix inverse of an array. MMULT returns the matrix product of two arrays. TRANSPOSE returns the transpose of an array.

MID(text,start\_number,number\_of\_characters)

Returns *number\_of\_characters* from *text*, starting at *start\_number*. The first character in *text* has *start\_number* 1, and so on.

If	MID returns
<i>Start_number</i> is greater than the length of <i>text</i>	"" (the empty text)
<i>Start_number</i> is less than the length of <i>text</i> , but <i>start_number</i> plus <i>number_of_characters</i> exceeds the length of <i>text</i>	The characters up to the end of <i>text</i>
<i>Start_number</i> is less than 1	The error value #VALUE!
<i>Number_of_characters</i> is negative	The error value #VALUE!

### Examples

MID("Hello world",1,5) *equals* "Hello"

MID("Hello world",7,20) *equals* "world"

MID("1234",5,5) *equals* "", empty text

### Related Functions

LEFT and RIGHT extract the leftmost and rightmost characters from a text value.

## MIN(number1,number2,...)

Returns the smallest number in the list of arguments.

MIN can have from 1 to 14 arguments. The arguments should be numbers, or arrays or references that contain numbers. If an array or reference argument contains text or logical values, those values are ignored. If the arguments contain no numbers, MIN gives 0.

### Examples

If A1:A5 contains the numbers 10, 7, 9, 27, and 2, then:

MIN(A1:A5) *equals* 2

MIN(A1:A5,0) *equals* 0

### Related Functions

DMIN returns the minimum value from selected database entries. MAX returns the maximum value from its list of arguments.

## MINUTE(serial\_number)

Returns the minute corresponding to *serial\_number*. The minute is given as an integer, ranging from 0 to 59.

*Serial\_number* is the date-time code used by Microsoft Excel for date and time calculations. It ranges from 0 to 65380. The numbers to the right of the decimal point in *serial\_number* represent the time; the numbers to the left represent the date. For example, the date-time combination 12:00 P.M., January 1, 1901 is represented as 367.5.



*Serial\_number* may be given as text, such as “16:48:00” or “4:48:00 P.M.”, instead of a number. The text is automatically converted to a serial number.

Note

This description assumes that the 1904 Date System check box in the Options Calculation dialog box is turned off. If this check box is turned on, then serial number 1 in the active document is the start of day January 2, 1904, instead of the start of day January 1, 1900. The 1904 Date System box is turned on automatically if you open a document from Microsoft Excel for the Macintosh, which has a different date system. You can also turn on the check box yourself if you are creating a document to be used in Microsoft Excel for the Macintosh. For more information, see Date in *Microsoft Excel Reference*.

Examples

MINUTE(“4:48:00 PM”) equals 48  
MINUTE(0.01) equals 14  
MINUTE(4.02) equals 28

Related Functions

YEAR, MONTH, DAY, WEEKDAY, HOUR, and SECOND convert serial numbers into years, months, days, weekdays, hours, or seconds. NOW returns the serial number of the current date and time.

MINVERSE(array)

Gives the inverse matrix for the matrix stored in *array*.  
*Array* must be a square array containing numbers. It can be given as a cell range, for example A1:C3, an array constant, such as {1,2,3;4,5,6;7,8,9}, or as a reference. If cells are empty or contain text, MINVERSE gives the error value #VALUE!. MINVERSE also gives #VALUE! if *array* is not a square-shaped array.  
Inverse matrices, like determinants, are generally used for solving systems of mathematical equations involving several variables. The product of a matrix and its inverse is the identity matrix, the square array in which the diagonal values equal 1 and all other values equal 0. For example, the 2-row, 2-column identity matrix is shown in cells A1:B2, and the 4-row, 4-column identity matrix is shown in cells D1:G4 below.

	A	B	C	D	E	F	G	H
1	1	0		1	0	0	0	
2	0	1		0	1	0	0	
3				0	0	1	0	
4				0	0	0	1	
5								

2x2 identity array

4x4 identity array

## Array

## Note

*Values* must be an array or a reference to cells that contain numbers. These numbers represent a series of payments (negative values) and income (positive values) occurring at regular periods. There must be at least one positive value and one negative value in order to calculate the modified internal rate of return. Otherwise, MIRR gives the error value #DIV/0!.

MIRR uses the order of *values* to interpret the order of cash flows. Be sure to enter your payment and income values in the correct sequence and with the correct signs (positive values for cash received, negative values for cash paid). Text, logical, and empty cell *values* are ignored.

If *n* is the number of cash flows in *values*, *frate*, the *finance\_rate*, and *rrate*, the *reinvest\_rate*, the formula for MIRR is:

$$\left[ \frac{-\text{NPV}(\text{rrate}, \text{values}[\text{positive}]) * (1 + \text{rrate})^n}{\text{NPV}(\text{frate}, \text{values}[\text{negative}]) * (1 + \text{frate})^{\frac{1}{n-1}}} \right] - 1$$

## Example

Suppose you're a commercial fisherman and you've just completed your fifth year of operation. When you started the business, you borrowed \$120,000 at 10% annual interest to purchase a boat. Your catch yielded \$39,000, \$30,000, \$21,000, \$37,000, and \$46,000 in the first five years. During these years you reinvested your profits, earning 12% annually.

	A	B	C	D	E	F	G	
1	Boat Cost	(\$120,000)						
2	Year 1 income	\$39,000						
3	Year 2 income	\$30,000						
4	Year 3 income	\$21,000						
5	Year 4 income	\$37,000						
6	Year 5 income	\$46,000						
7								

To calculate the investment's modified rate of return after five years, use:

MIRR(B1:B6,10%,12%) equals 12.61%

To calculate the modified rate of return after three years, use:

MIRR(B1:B4,10%,12%) equals -4.80%

To calculate the five-year modified rate of return based on a *reinvest\_rate* of 14%, use:

`MIRR(B1:B6,10%,14%) equals 13.48%`

### Related Functions

IRR returns the internal rate of return for an investment, without financing costs or reinvestment gains. RATE returns the interest rate for an investment based on constant cash flows.

## MMULT(array1,array2)

Returns the matrix product of *array1* and *array2*.

The number of columns in *array1* must be the same as the number of rows in *array2*, and both arrays must contain only numbers. They can be given as cell ranges, array constants, or as references. If any cells are empty or contain text, or if the number of columns in *array1* is different than the number of rows in *array2*, MMULT gives the error value #VALUE!.

The result of MMULT is an array with the same number of rows as *array1* and the same number of columns as *array2*.

The matrix product array a of two arrays b and c is given by:

$$a_{ij} = \sum_{k=1}^n b_{ik} c_{kj}$$

where i is the row number and j is the column number.

**Note** | Formulas that return arrays must be entered as array formulas. To enter an array formula, press CONTROL+SHIFT+ENTER, or, with a mouse, press CONTROL+SHIFT while you click the check box in the formula bar. For information on arrays, see *Array* in *Microsoft Excel Reference*.

## Examples

	A	B	C	D	E	F	G	H
1	1	3	0	2	0			
2	7	2	0	0	2			
3	1	0	0					
4								

In the worksheet above:

MMULT(A1:B2,D1:E2) equals {2,6;14,4}

MMULT(B1:C2,D1:E2) equals {6,0;4,0}

MMULT(A1:C3,D1:E2) equals #VALUE!, because A1:C3 has 3 columns and D1:E2 has only 2 rows.

## Related Functions

MDETERM returns the determinant of an array. MINVERSE returns the inverse of an array. TRANSPOSE returns the transpose of an array.

## MOD(number,divisor\_number)

Returns the remainder (modulus) after *number* is divided by *divisor\_number*. The result has the same sign as *divisor\_number*.

If *divisor\_number* is 0, MOD returns the error value #DIV/0!.

The MOD function can be expressed in terms of the INT function:

$$\text{MOD}(n,d) = n - d * \text{INT}(n/d)$$

## Examples

MOD(3,2) equals 1

MOD(-3,2) equals 1

MOD(3,-2) equals -1

MOD(-3,-2) equals -1

## Related Functions

ROUND rounds a number to a specified number of digits. INT rounds a number down to the nearest integer. TRUNC truncates a number to an integer.

## MONTH(serial\_number)

Returns the month corresponding to *serial\_number*. The month is given as an integer, ranging from 1 to 12.

*Serial\_number* is the date-time code used by Microsoft Excel for date and time calculations. It ranges from 1 to 65380, corresponding to the dates January 1, 1900 through December 31, 2078. The numbers to the right of the decimal point in *serial\_number* represent the time; the numbers to the left represent the date. For example, the date-time combination 12:00 P.M., January 1, 1901 is represented as 367.5.

*Serial\_number* may be given as text, such as “4-15-1985” or “15-Apr-1985”, instead of a number. The text is automatically converted to a serial number.

### Note

This description assumes that the 1904 Date System check box in the Options Calculation dialog box is turned off. If this check box is turned on, then serial number 1 in the active document is the start of day January 2, 1904, instead of the start of day January 1, 1900. The 1904 Date System box is turned on automatically if you open a document from Microsoft Excel for the Macintosh, which has a different date system. You can also turn on the check box yourself if you are creating a document to be used in Microsoft Excel for the Macintosh. For more information, see Date in *Microsoft Excel Reference*.

## Examples

MONTH(“6-May”) equals 5

MONTH(366) equals 12

MONTH(367) equals 1

## Related Functions

YEAR, DAY, WEEKDAY, HOUR, MINUTE, and SECOND convert serial numbers into years, days, weekdays, hours, minutes, or seconds. NOW returns the serial number of the current date and time.

## N(value)

Returns *value* translated into a number. N translates values as follows:

If value is or refers to	N returns
A number	That number
TRUE	1
Anything else	0

It is not generally necessary to use the N function in a formula, since Microsoft Excel automatically translates values as necessary. This function is provided for compatibility with other worksheet programs. For information on how Microsoft Excel translates values, see “Translating Data Types” in Chapter 1, “Worksheet Function Basics.”

## Examples

	A	B	C	D	E	F	G	H
1	7	odd	TRUE					
2	8	even	TRUE					
3	13	even	FALSE					
4	14	odd	FALSE					
5								

In the worksheet above:

N(A1) *equals* 7

N(B2) *equals* 0, because B2 contains text

N(C2) *equals* 1, because C2 contains TRUE

N(“7”) *equals* 0, because “7” is text

N(“4/17/87”) *equals* 0, because “4/17/87” is text

## Related Functions

CELL returns information about the formatting, location, or contents of a cell. T translates its argument into text.

### NA()

Returns the error value #N/A. #N/A is an error value which indicates “No value is Available.”

You can also type the value #N/A directly into a cell. The NA function is provided for compatibility with other worksheet programs.

#N/A is used primarily for marking empty cells. By entering #N/A in cells where you are missing information, you can avoid the problem of unintentionally including empty cells in your calculations. When a formula tries to use a cell containing #N/A, it returns the error value #N/A.

Even though NA does not have an argument, you must include the empty parentheses with the function name. Otherwise, Microsoft Excel will not recognize it as a function.

### Examples

Cell B4 contains the formula =NA()

	A	B	C	D	E	F	G	H
1	10	10						
2	20	20						
3	30	30						
4		#N/A						
5	40	40						
6								

In the worksheet above:

AVERAGE(A1:A5) *equals* 25

AVERAGE(B1:B5) *equals* #N/A

IF(ISNA(B4), “No Value”, B4) *equals* “No Value”

IF(ISBLANK(A4), NA(), A4) *equals* #N/A

### NOT(logical)

Reverses the value of *logical*. If *logical* is FALSE, NOT returns TRUE; if *logical* is TRUE, NOT returns FALSE.

### Examples

NOT(FALSE) *equals* TRUE

NOT(1 + 1 = 2) *equals* FALSE



## Related Functions

AND is TRUE if all its arguments are TRUE. OR is TRUE if one or more arguments are TRUE.

## NOW()

Returns the serial number of the current date and time, according to your computer's built-in clock.

The serial number is the date-time code used by Microsoft Excel for date and time calculations. It ranges from 1 to 65380, corresponding to the dates January 1, 1900 through December 31, 2078. The numbers to the right of the decimal point in the serial number represent the time; the numbers to the left represent the date. For example, the date-time combination 12:00 P.M., January 1, 1901 is represented as 367.5.

Even though NOW does not have an argument, you must include the empty parentheses with the function name. Otherwise, Microsoft Excel will not recognize it as a function.

**Note** | The NOW function changes only when the worksheet is calculated. It does not update continuously.

**Note** | This description assumes that the 1904 Date System check box in the Options Calculation dialog box is turned off. If this check box is turned on, then serial number 1 in the active document is the start of day January 2, 1904, instead of the start of day January 1, 1900. The 1904 Date System box is turned on automatically if you open a document from Microsoft Excel for the Macintosh, which has a different date system. You can also turn on the check box yourself if you are creating a document to be used in Microsoft Excel for the Macintosh. For more information, see Date in *Microsoft Excel Reference*.

## Examples

If your computer's built-in clock is set to 12:30:00 P.M., 1-Jan-1987, then:

NOW() equals 31778.52083

Ten minutes later:

NOW() equals 31778.52778

## Related Functions

YEAR, MONTH, DAY, WEEKDAY, HOUR, MINUTE, and SECOND convert serial numbers into years, months, days, weekdays, hours, minutes, or seconds.

## NPER(rate,pmt,pv,fv,type)

Returns the number of periods for an investment based on periodic, constant payments and a constant interest rate. The optional *f<sub>v</sub>* and *type* arguments are assumed to be 0 if omitted.

For a complete description of the arguments in NPER, see PV.

## Examples

NPER(12%/12, -100, -1000, 10000, 1) equals 60

NPER(1%, -100, -1000, 10000) equals 60

NPER(1%, -100, 1000) equals 11

## Related Functions

FV returns the future value of an investment. IPMT returns the interest payment for an investment for a given period. PPMT returns the principal payment for an investment for a given period. PMT returns the periodic total payment for an investment. PV returns the present value of an investment. RATE returns the interest rate per period of an investment.

## NPV(rate,value1,value2,...)

Returns the net present value of an investment based on a series of periodic cash flows, *value1*, *value2*, . . . , and a discount rate equal to *rate*.

The net present value of an investment is today's value of a future series of payments (negative values) and income (positive values). *Rate* is the rate of discount over the length of one period. *Value1*, *value2*, . . . must be equally spaced in time and occur at the end of each period. NPV uses the order of *value1*, *value2*, . . . to interpret the order of cash flows. Be sure to enter your payment and income values in the correct sequence. The list of values can contain up to 13 arguments. Arguments that are numbers, empty cells, logical values, or text representations of numbers are counted; arguments that are error values or text that cannot be translated into numbers are ignored. If an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, logical values, text, or error values in the array or reference are ignored.

The NPV investment begins one period before the date of the *value1* cash flow, and ends with the last cash flow in the list. It is important to realize that the NPV calculation is based on future cash flows. If your first cash flow occurs at the beginning of the first period, the first value must be added to the NPV result, not included in the *values* arguments. For more information, see the examples below.

If *n* is the number of cash flows in the list of *values*, the formula for NPV is:

$$\text{NPV} = \sum_{i=1}^n \frac{\text{values}_i}{(1 + \text{rate})^i}$$

#### Note

NPV is similar to the PV function (Present Value). The primary difference between PV and NPV is that PV allows cash flows to begin either at the end or at the beginning of the period. Unlike the variable NPV cash flow values, PV cash flows must be constant throughout the investment. For information on Present Value and other related financial functions, see PV.

NPV is also related to the IRR function (Internal Rate of Return). IRR is the *rate* for which NPV equals zero:  $\text{NPV}(\text{IRR}(. . .), . . .) = 0$ .

## Examples

Suppose you're considering an investment in which you pay \$10,000 one year from today, and receive an annual income of \$3,000, \$4,200, and \$6,800 in the three years that follow. Assuming an annual discount rate of 10%, the net present value of this investment is:

$\text{NPV}(10\%, -10000, 3000, 4200, 6800)$  equals \$1,188

In this example, you include the initial \$10,000 cost as one of the *values*, because the payment does not occur at the beginning of the first period.

Let's consider an investment that starts at the beginning of the first period. Suppose you're interested in buying a shoe store. The cost of the business is \$40,000, and you expect to receive the following income for the first five years of operation: \$8,000, \$9,200, \$10,000, \$12,000, \$14,500. The annual discount rate is 8%; this might represent the rate of inflation, or the interest rate of a competing investment.

	A	B	C	D	E	F	G	H
1	Investment	(\$40,000)						
2	Year 1 income	\$8,000						
3	Year 2 income	\$9,200						
4	Year 3 income	\$10,000						
5	Year 4 income	\$12,000						
6	Year 5 income	\$14,500						
7								

The net present value of the shoe store investment is given by:

$\text{NPV}(8\%, \text{B2:B6}) + \text{B1}$  equals \$1,922

In this example, you don't include the initial \$40,000 cost as one of the *values*, because the payment occurs at the beginning of the first period.

Suppose your shoe store's roof collapses during the sixth year and you assume a loss of \$9,000 for that year. The net present value of the shoe store investment after six years is given by:

$\text{NPV}(8\%, \text{B2:B6}, -9000) + \text{B1}$  equals -\$3,749

## Related Functions

FV returns the future value of an investment. IRR returns the internal rate of return for an investment. PV returns the present value of an investment.

## OR(logical1, logical2, ...)

Returns the logical value TRUE if any of the values in the list of arguments are TRUE. If all of the values in the list of arguments are FALSE, OR returns the logical value FALSE.

OR can have from 1 to 14 arguments. The arguments should be logical values, numbers, or text representations of logical values. Arguments that are empty cells, error values, or text that cannot be translated into logical values cause errors. If arguments are arrays or references, any values other than logical values within an array or reference are ignored.

## Examples

$\text{OR}(\text{TRUE})$  equals TRUE

$\text{OR}(1 + 1 = 1, 2 + 2 = 5)$  equals FALSE

If A1:A3 contains the values TRUE, FALSE, and TRUE, then:

$\text{OR}(\text{A1:A3})$  equals TRUE

## Related Functions

AND is TRUE if all its arguments are TRUE. NOT reverses the logic of its argument.

## PI()

Returns the number 3.14159265358979, the mathematical constant  $\pi$ , accurate to 15 digits.

Even though PI does not have an argument, you must include the empty parentheses with the function name. Otherwise, Microsoft Excel will not recognize it as a function.

## Examples

PI()/2 *equals* 1.57079. . .

SIN(PI()/2) *equals* 1

## PMT(rate,nper,pv,fv,type)

Returns the payment for an investment based on periodic, constant payments and a constant interest rate. The optional *f**v* and *t**y**p**e* arguments are assumed to be 0 if omitted.

For a complete description of the arguments in PMT, see PV.

## Examples

PMT(8%/12,10,0,10000,1) *equals* -\$963.94

PMT(8%/12,10,0,10000) *equals* -\$970.37

PMT(12%/12,5,-5000) *equals* \$1,030.20

## Related Functions

FV returns the future value of an investment. IPMT returns the interest payment for an investment for a given period. NPER returns the number of periods (payments) for an investment. PPMT returns the principal payment for an investment for a given period. PV returns the present value of an investment. RATE returns the interest rate per period of an investment.

## PPMT(rate,per,nper,pv,fv,type)

Returns the payment on the principal for a given period for an investment based on periodic, constant payments and a constant interest rate. The optional *fv* and *type* arguments are assumed to be 0 if omitted.

*Per* specifies the period and must be in the range 1 to *nper*. For a complete description of the other arguments in PPMT, see PV.

### Note

In annuity functions, cash paid out, such as deposits to savings, is represented by negative numbers; cash received, such as dividend checks, is represented by positive numbers.

For example, a \$1000 deposit to the bank would be represented by the argument  $-1000$  if you were the depositor, and by the argument  $1000$  if you were the bank.

## Examples

The following function returns the principal payment for the first month of a 2-year \$2000 loan at 10% annual interest:

PPMT(0.1/12,1,24,2000) *equals*  $-75.62$

The following function returns the principal payment for the last year of a 10-year \$200,000 loan at 8% annual interest:

PPMT(0.08,10,10,200000) *equals*  $-27598.10$

## Related Functions

FV returns the future value of an investment. IPMT returns the interest payment for an investment for a given period. NPER returns the number of periods (payments) for an investment. PMT returns the periodic total payment for an investment. PV returns the present value of an investment. RATE returns the interest rate per period of an investment.

## PRODUCT(number1,number2,...)

Multiplies all the numbers given as arguments and returns the product. PRODUCT can have from 1 to 14 arguments. Arguments that are numbers, empty cells, logical values, or text representations of numbers are counted; arguments that are error values or text that cannot be translated into numbers cause errors. If an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, logical values, text, or error values in the array or reference are ignored.

## Examples

	A	B	C	D	E	F	G	H
1	February	March	April	May	June	July		
2	5	15	30	40	50			
3								

PRODUCT(A2:C2) *equals* 2250

PRODUCT(A2:C2,2) *equals* 4500

## Related Functions

SUM adds its arguments. FACT returns the factorial of a number.

## PROPER(text)

Capitalizes the first letter in *text* and any other letters in *text* that follow any character other than a letter. Converts all other letters to lowercase.

## Examples

PROPER("this is a TITLE") *equals* "This Is A Title"

PROPER("2-cent's worth") *equals* "2-Cent'S Worth"

PROPER("76tromBones") *equals* "76Trombones"

## Related Functions

LOWER and UPPER change every letter in a text value to lowercase or uppercase respectively.

## PV(rate,nper,pmt,fv,type)

These functions:

FV(rate,nper,pmt,pv,type)

NPER(rate,pmt,pv,fv,type)

PMT(rate,nper,pv,fv,type)

RATE(nper,pmt,pv,fv,type,guess)

apply to annuities. An annuity is a series of constant cash payments made over a continuous period of time. For example, a car loan or a mortgage is an annuity.

Microsoft Excel solves for one financial argument in terms of the others. If the *rate* is not 0, then:

$$pv*(1+rate)^{nper}+pmt(1+rate*type)*\left(\frac{(1+rate)^{nper}-1}{rate}\right)+fv=0$$

If the *rate* is 0, then:

$$pv+pmt*nper+fv=0$$

The following sections describe the arguments *rate*, *nper*, *pmt*, *pv*, and *fv*. For a description of the *guess* argument, see RATE.

### Note

For all the arguments, cash you pay out, such as deposits to savings, is represented by negative numbers; cash you receive, such as dividend checks, is represented by positive numbers.

For example, a \$1000 deposit to the bank would be represented by the argument *-1000* if you were the depositor, and by the argument *1000* if you were the bank.

### type

The *type* argument indicates when payments are due.

If payments are due	Then type equals
At the end of the period	0
At the beginning of the period	1

If *type* is omitted it is assumed to be 0.

### rate

*Rate* is the **interest rate** per period.

For example, if you get a car loan at a 10% annual interest rate and make monthly payments, your interest rate per month is 10%/12, or .83%. You would enter 10%/12, or .83%, or .0083, into the formula as the *rate*.

Make sure that you are consistent about the units you use for specifying *rate* and *nper*. If you make monthly payments on a 4-year loan at 12% annual



interest, use  $12\%/12$  for *rate* and  $4*12$  for *nper*. If you make annual payments on the same loan, use  $12\%$  for *rate* and  $4$  for *nper*.

## nper

*Nper* is the total **number of payment periods** in an annuity.

For example, if you get a 4-year car loan and make monthly payments, your loan has  $4 \times 12$ , or 48 periods. You would enter 48 into the formula as the number of *periods*.

## pmt

*Pmt* is the **payment** made each period, and cannot change over the life of the annuity. Typically, *pmt* contains principal and interest.

For example, the monthly payments on a \$10,000, 4-year car loan at 12% are \$263.33. You would enter  $-263.33$  into the formula as the *pmt*.

## pV

*Pv* is the **present value**, or the lump sum amount that a series of payments to be paid in the future is worth right now.

For example, when you borrow money to buy a car, the loan amount is the present value to the lender of the monthly car payments you will make.

## fV

*Fv* is the **future value**, or a cash balance you want to be attained sometime in the future after the last payment is made. If *fV* is omitted, it is assumed to be 0. The future value of a loan, for example, is 0.

For example, if you think you will need \$50,000 in 18 years to pay for your child's education, then \$50,000 is the future value. You could then make a conservative guess at an interest rate and determine how much you must save each month.

## Examples

An insurance company wants to sell you an annuity that pays \$500 at the end of every month for the next 20 years. The cost of the annuity is \$60,000 and

the money paid out will earn 8%. Should you buy it? Using the PV function we find that the present value of the annuity is:

$PV(0.08/12, 12*20, 500, , 0)$  equals  $-59777.15$

(The result is negative because it represents money that you would pay — an outgoing cash flow.) This tells you that they want you to pay more than the present value of the annuity; therefore, you should not buy it.

Suppose you want to save money to take a special vacation a year from now. You deposit \$1,000 into a savings account that earns 6% annual interest compounded monthly (monthly interest of 6%/12, or .5%). You plan to deposit \$100 at the beginning of every month for the next 12 months. How much money will be in the account at the end of 12 months?

$FV(0.5\%, 12, -100, -1000, 1)$  equals 2301.40

You want to save \$50,000 in 18 years for your child's education. If you assume you'll be able to get 6% interest on your savings, you can use PMT to determine how much to save each month:

$=PMT(6\%/12, 18*12, 0, 50000)$  equals  $-129.08$

If you pay \$129.08 into a 6% savings account every month for 18 years, you will have \$50,000.

### Related Functions

IPMT returns the interest payment for an investment for a given period.

PPMT returns the principal payment for an investment for a given period.

## RAND()

Returns a random number greater than or equal to 0 and less than 1. A new random number is generated every time the worksheet is calculated.

**Tip**

If you want to use RAND to generate a random number, but don't want the numbers to change every time the cell is calculated:

- 1 Type `=RAND()` in the formula bar.
- 2 Choose Options Calculate Now.
- 3 Enter the formula.

Or, you could:

- 1 Enter the formula normally.
- 2 Copy the cell.
- 3 Choose Edit Paste Special.
- 4 Select Values.

Microsoft Excel replaces the formula with the equivalent constant value.

**Examples**

	A	B	C	D	E	F	G	H
1	0	0.33333	0.66667					
2	Schilke	Spotty	Droopy					
3	Temis	Max	Buddy					
4								

If you wanted to choose one of the names in row 2 at random, you could use either of the following formulas:

```
=HLOOKUP(RAND(),A1:C3,2)
=INDEX(A2:C3,1,1+3*RAND())
```

(The INDEX formula is convenient because it doesn't require the numbers in row 1.) One of the names "Schilke", "Spotty", or "Droopy" is chosen at random whenever the worksheet is calculated.

To choose either "Temis", "Max", or "Buddy" at random whenever the worksheet is calculated, use either of the following formulas:

```
=HLOOKUP(RAND(),A1:C3,3)
=INDEX(A2:C3,2,1+3*RAND())
```

## RATE(*nper*,*pmt*,*pv*,*fv*,*type*,*guess*)

Returns the interest rate per period for an annuity.

See PV for a complete description of the arguments *nper*, *pmt*, *pv*, *fv*, and *type*.

RATE is calculated by iteration, and can have zero or more solutions. If the successive results of RATE do not converge to within 0.0000001 after 20 iterations, RATE returns #NUM!. *Guess* is your guess for what the rate will be. If you omit *guess*, it is assumed to be 10%.

**Note** | If RATE does not converge, try different values for *guess*. RATE usually converges if given a *guess* between 0 and 1.

### Example

The rate of a 4-year \$8000 loan with monthly payments of \$200 is given by the following formula:

=RATE(48, -200,8000) equals 0.0077

This is the monthly rate, because the period is monthly. The annual rate is given by .0077\*12, which equals 9.2%.

### Related Functions

FV returns the future value of an investment. IPMT returns the interest payment for an investment for a given period. NPER returns the number of periods (payments) for an investment. PPMT returns the principal payment for an investment for a given period. PMT returns the periodic total payment for an investment. PV returns the present value of an investment.

## REPLACE(*old\_text*,*start\_num*,*num\_chars*,*new\_text*)

Removes *num\_chars* characters from *old\_text* starting at *start\_num*, then replaces them with *new\_text*. The first character in the text is numbered 1.

### Examples

REPLACE("abcde",1,1,"\*") equals "\*bcde"

REPLACE("abcde",3,2,"\*") equals "ab\*e"

REPLACE("any text",1,LEN("any text"),"") equals ""

If NOW() *equals* 12/8/86, and cell \$A\$1 contains the text “And that’s the way it is,” then:

REPLACE(\$A\$1,LEN(\$A\$1)+1,1,TEXT(NOW(),"mmm d, yyyy"))  
*equals*

REPLACE(\$A\$1,27,1,"December 8, 1986") *equals*  
“And that’s the way it is, December 8, 1986”

## Related Functions

SUBSTITUTE also replaces characters within text, but instead of specifying a start number and number of characters to replace, you specify the exact text to replace. TRIM removes spaces from text.

## REPT(text,number\_times)

Takes *text* and repeats it *number\_times* times to make a new text value. *Number\_times* must be greater than or equal to 0. If it is 0, REPT returns “”, the empty text. If *number\_times* is not an integer, it is truncated. The result of the REPT function cannot be longer than 255 characters.

### Examples

REPT("\*\_",3) *equals* "\*\_\*\_\*\_"

REPT("Louie",2.9) *equals* "Louie Louie"

## RIGHT(text,number\_of\_chars)

Returns the last *number\_of\_chars* characters in *text*. *Number\_of\_chars* must be greater than zero. If *number\_of\_chars* is greater than the length of *text*, the entire text is returned. If *number\_of\_chars* is omitted it is assumed to be 1.

### Examples

RIGHT("Paul Irving",6) *equals* "Irving"

RIGHT("Hedgehog?") *equals* "?"

## Related Functions

LEFT extracts the leftmost characters from a text value. MID extracts characters based on a starting position and number of characters.

### ROUND(*number*,*number\_of\_digits*)

Rounds *number* to *number\_of\_digits* digits.

If *number\_of\_digits* is greater than 0, then *number* is rounded to the specified number of decimal places. If *number\_of\_digits* is 0, then *number* is rounded to the nearest integer. If *number\_of\_digits* is less than 0, then *number* is rounded to the left of the decimal point.

### Examples

ROUND(2.15,1) equals 2.2

ROUND(2.149,1) equals 2.1

ROUND(−1.475,2) equals −1.48

ROUND(21.5, −1) equals 20

## Related Functions

INT rounds a number down to the nearest integer. TRUNC truncates a number to an integer. MOD returns the remainder from division.

### ROW(*reference*)

Returns the row number(s) of *reference*. If *reference* includes more than one row, ROW returns the row numbers as a vertical array. If *reference* is omitted, it is assumed to be the reference of the cell(s) in which the ROW function appears.

*Reference* cannot be a reference to multiple areas.

### Examples

ROW(A3) equals 3

ROW(A3:B5) equals {3;4;5}

If ROW is entered in C5, then:

ROW() equals ROW(C5) equals 5

## Related Functions

COLUMN returns the column number(s) in a reference. COLUMNS returns the number of columns in an array. ROWS returns the number of rows in an array.

## ROWS(array)

Returns the number of rows in *array*.

## Examples

ROWS(A1:C4) equals 4

ROWS({1,2,3;4,5,6}) equals 2

## Related Functions

COLUMN returns the column number(s) in a reference. COLUMNS returns the number of columns in an array. ROWS returns the row number(s) in a reference.

## SEARCH(find\_text,within\_text,start\_at\_num)

Searches for *find\_text* within *within\_text*, starting the search at the character specified by *start\_at\_num*. The first character in *within\_text* is character number 1. SEARCH returns the number of the character at which *find\_text* first occurs. If *find\_text* does not appear within *within\_text*, if *start\_at\_num* is not greater than zero, or if *start\_at\_num* is greater than the length of *within\_text*, SEARCH returns the error value #VALUE!.

If you omit *start\_at\_num*, it is assumed to be 1. If *find\_text* is "", it will match the first character that is searched (that is, the character numbered *start\_at\_num*).

While searching, SEARCH ignores case distinctions. *Find\_text* may contain the following wildcard characters:

Wildcard	Meaning
? (question mark)	Any single character can occupy position
* (asterisk)	Any sequence of characters can occupy position

### Examples

SEARCH("e","Beautiful Noise") *equals 2*

SEARCH("e","Beautiful Noise",2) *equals 2*

SEARCH("", "Beautiful Noise",7) *equals 7*

SEARCH("u\*i","Beautiful Noise",1) *equals 4*

SEARCH("u\*i","Beautiful Noise",5) *equals 8*

### Related Functions

FIND is just like SEARCH, except FIND is case-sensitive and does not allow wildcard characters. EXACT checks to see if two text values are identical. LEN returns the length of text.

## SECOND(serial\_number)

Returns the second corresponding to *serial\_number*. The second is given as an integer, ranging from 0 to 59.

*Serial\_number* is the date-time code used by Microsoft Excel for date and time calculations. It ranges from 0 to 65380. The numbers to the right of the decimal point in *serial\_number* represent the time; the numbers to the left represent the date. For example, the date-time combination 12:00 P.M., January 1, 1901 is represented as 367.5.

*Serial\_number* may be given as text, such as "16:48:00" or "4:48:00 P.M.", instead of a number. The text is automatically converted to a serial number.



**Note** | This description assumes that the 1904 Date System check box in the Options Calculation dialog box is turned off. If this check box is turned on, then serial number 1 in the active document is the start of day January 2, 1904, instead of the start of day January 1, 1900. The 1904 Date System box is turned on automatically if you open a document from Microsoft Excel for the Macintosh, which has a different date system. You can also turn on the check box yourself if you are creating a document to be used in Microsoft Excel for the Macintosh. For more information, see *Date* in *Microsoft Excel Reference*.

## Examples

SECOND(0.007) *equals* 5

SECOND(29747.007) *equals* 5

SECOND("3:30:30 PM") *equals* 30

## Related Functions

YEAR, MONTH, DAY, WEEKDAY, HOUR, and MINUTE convert serial numbers into years, months, days, weekdays, hours, or minutes. NOW returns the serial number of the current date and time.

## SIGN(number)

Returns 1 if *number* is positive, 0 if *number* is 0, and -1 if *number* is negative.

## Examples

SIGN(10) *equals* 1

SIGN(4 - 4) *equals* 0

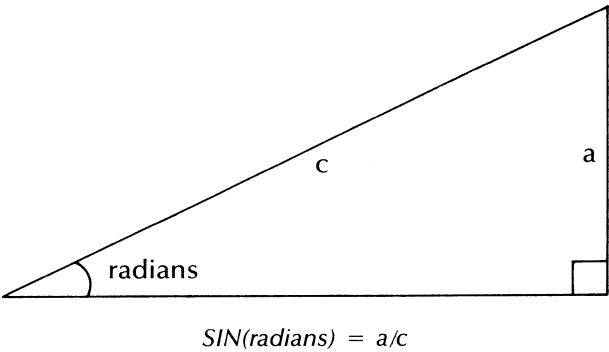
SIGN(-0.00001) *equals* -1

## Related Functions

ABS returns the absolute value of a number.

# SIN(radians)

Returns the sine of the given number of *radians*.  
If your argument is in degrees, multiply the argument by PI()/180 to convert the argument to radians.



## Examples

- SIN(−PI()) equals 0
- SIN(PI()/2) equals 1
- SIN(30\*PI()/180) returns 0.5, the sine of 30 degrees

## Related Functions

ASIN returns the arcsine of a number. PI returns the value  $\pi$ .

# SLN(cost,salvage,life)

Returns the straight-line depreciation for an asset for a single period. The following list summarizes the arguments:

Argument	Description
<i>cost</i>	Initial cost of the asset
<i>salvage</i>	Value at the end of the depreciation (sometimes called the <b>salvage value</b> of the asset)
<i>life</i>	Number of periods over which the asset is being depreciated (sometimes called the <b>useful life</b> of the asset)

## Example

If you've bought a truck for \$30,000 that has a useful life of 10 years and a salvage value of \$7500, the yearly depreciation allowance for the first year is given by:

`SLN(30000,7500,10)` equals \$2250

## Related Functions

**SYD** returns the depreciation for an asset for a specified period, using the sum-of-years' digits method. **DDB** returns the depreciation for an asset for a specified period, using the double-declining balance method.

## SQRT(number)

Calculates the positive square root of *number*. If *number* is negative, **SQRT** returns the error value **#NUM!**.

## Examples

`SQRT(16)` equals 4

`SQRT(-16)` equals **#NUM!**

`SQRT(ABS(-16))` equals 4

## Related Functions

**PRODUCT** multiplies its arguments together. **EXP** raises a number to a power.

## STDEV(number1,number2,...)

Calculates the standard deviation of a population based on a sample given as arguments. The standard deviation is calculated using the "non-biased" or "n - 1" method.

Note	STDEV assumes that its arguments are a sample of the population. If your data represents the entire population, you should compute the standard deviation using <b>STDEVP</b> .
------	---

STDEV can have from 1 to 14 arguments. All text, logical, or empty cell values cause errors.

STDEV uses the following formula:

$$\sqrt{\frac{n \sum(x^2) - (\sum x)^2}{n(n-1)}}$$

## Example

	A	B	C	D	E	F	G	H
1	January	February	March	April	May	June	July	
2	5000	15000	30000	40000	50000			
3								

STDEV(A2:E2) equals 18,234.58

## Related Functions

STDEVP calculates the true standard deviation when you have all the data for a population. VAR and VARP calculate variance. DSTDEV estimates the standard deviation from selected database entries.

## STDEVP(number1,number2,...)

Calculates the standard deviation of a population given the population as arguments. The standard deviation is calculated using the “biased” or “n” method.

**Note** | STDEVP assumes that its arguments are the entire population. If your data represents a sample of the population, you should compute the standard deviation using STDEV.

STDEVP can have from 1 to 14 arguments. All text, logical, or empty cell values cause errors.

STDEVP uses the following formula:

$$\sqrt{\frac{n \sum x^2 - (\sum x)^2}{n^2}}$$

## Example

	A	B	C	D	E	F	G	H
1	January	February	March	April	May	June	July	
2	5000	15000	30000	40000	50000			
3								

STDEVP(A2:E2) *equals* 16,309.51

## Related Functions

STDEV estimates the standard deviation from a sample. VARP and VAR calculate variance. DSTDEVP calculates the standard deviation from selected database entries.

## SUBSTITUTE(text,old\_text,new\_text,instance\_number)

Substitutes *new\_text* for *old\_text* in *text*. If you specify *instance\_number* only that instance of *old\_text* is replaced. Otherwise, every *old\_text* in *text* is changed to *new\_text*.

## Examples

SUBSTITUTE("waffles","f","d") *equals* "waddles"

SUBSTITUTE("Ann And Nick","An","Da",1) *equals* "Dan And Nick"

SUBSTITUTE("Ann And Nick","An","Da",2) *equals* "Ann Dad Nick"

SUBSTITUTE("Ann And Nick","An","Da") *equals* "Dan Dad Nick"

SUBSTITUTE("hi de hi","hi","ho",2) *equals* "hi de ho"

## Related Functions

REPLACE also substitutes characters within text, but instead of specifying the exact text to replace you specify a starting point and number of characters to replace. TRIM removes spaces from text.

## SUM(number1,number2,...)

Adds all the numbers given as arguments and returns the sum. SUM can have from 1 to 14 arguments. Arguments that are numbers, empty cells, logical values, or text representations of numbers are counted; arguments that are error values or text that cannot be translated into numbers cause errors. If an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, logical values, text, or error values in the array or reference are ignored.

### Examples

	A	B	C	D	E	F	G	H
1	January	February	March	April	May	June	July	
2	5	15	30	40	50			
3								

SUM(A2:C2) equals 50

SUM(B2:E2,15) equals 150

### Related Functions

AVERAGE averages its arguments. PRODUCT multiplies its arguments together. COUNT and COUNTA count numbers and values.

## SYD(cost,salvage,life,per)

Returns the sum-of-years' digits depreciation for an asset for a specified period. The following list summarizes the arguments:

Argument	Description
<i>cost</i>	Initial cost of the asset
<i>salvage</i>	Value at the end of the depreciation (sometimes called the <b>salvage value</b> of the asset)
<i>life</i>	Number of periods over which the asset is being depreciated (sometimes called the <b>useful life</b> of the asset)
<i>per</i>	Period (must use same units as <i>life</i> )

## Examples

If you've bought a truck for \$30,000 that has a useful life of 10 years and a salvage value of \$7500, the yearly depreciation allowance for the first year is given by:

`SYD(30000,7500,10,1)` equals \$4090.91

The yearly depreciation allowance for the tenth year is given by:

`SYD(30000,7500,10,10)` equals \$409.09

## Related Functions

`SLN` returns the straight-line depreciation of an asset for one period. `DDB` returns the depreciation for an asset for a specified period, using the double-declining balance method.

## T(value)

Returns *value* translated into text. If *value* is or refers to text, `T` returns *value*. Otherwise, `T` returns "", the empty text.

It is not generally necessary to use the `T` function in a formula, since Microsoft Excel automatically translates values as necessary. This function is provided for compatibility with other worksheet programs. For information on how Microsoft Excel translates values, see "Translating Data Types" in Chapter 1, "Worksheet Function Basics."

## Examples

	A	B	C	D	E	F	G	H
1	Year	Rainfall						
2	1983	19						
3	1984	13						
4	1985	9						
5	1986	7						
6	1987	6						
7								

`T(B1)` equals "Rainfall"

`T(B2)` equals ""

`T("TRUE")` equals "TRUE"

`T(TRUE)` equals ""

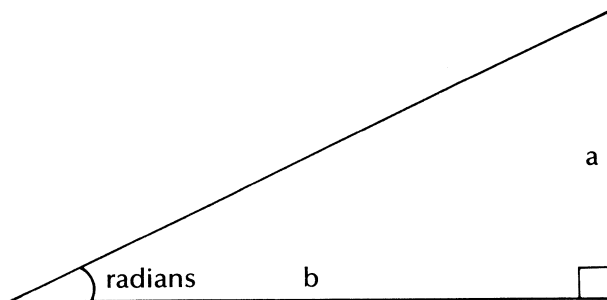
## Related Functions

CELL returns information about the formatting, location, or contents of a cell. N translates its argument into a number.

## TAN(radians)

Returns the tangent of the angle specified by given number of *radians*.

If your argument is in degrees, multiply the argument by  $\text{PI}()/180$  to convert the argument to radians.



$$\begin{aligned}\text{TAN}(\text{radians}) &= a/b \\ &= \frac{\text{SIN}(\text{radians})}{\text{COS}(\text{radians})}\end{aligned}$$

## Examples

$\text{TAN}(0.785)$  equals 1

$\text{TAN}(45 * \text{PI}()/180)$  equals 1

## Related Functions

ATAN returns the arctangent of a number. ATAN2 returns the arctangent from x- and y-coordinates. PI returns the value  $\pi$ .



## TEXT(value,format\_text)

Formats *value* as specified by *format\_text*, and returns the result as text.

*Format\_text* must be text formatted with the Format Number command and cannot contain an asterisk (\*) and cannot be "General." For information on the Format Number command, see Format Number command in *Microsoft Excel Reference*.

**Note** | The major difference between formatting a cell containing a value with the Format Number command and formatting a value directly with TEXT is that TEXT converts its result to text. A number formatted with the Format Number command is still a number.

### Examples

TEXT(2.715,"\$0.00") equals "\$2.72"

TEXT("4/15/1987","mmm d, yyyy") equals "April 15, 1987"

### Related Functions

DOLLAR and FIXED format a number and convert it to text. VALUE converts a text argument to a number.

## TIME(hour,minute,second)

Returns the serial number of the time specified by the numbers *hour*, *minute*, and *second*. You can use positive or negative numbers as arguments, as long as the resulting serial number is positive. For example, to find the serial number for the time 50 seconds before 2:00:02, you could use TIME(2,0,2-50).

The serial number is a decimal fraction ranging from 0 through 0.9999, representing the times from 0:00:00 or 12:00:00 A.M. to 23:59:59 or 11:59:59 P.M.

### Examples

TIME(12,0,0) equals 0.5 equals 12:00:00

TIME(16,48,0) - TIME(12,0,0) equals 0.2 equals 4:48:00

"16:48:00" - "12:00:00" equals 0.2 equals 4:48:00

## Related Functions

TIMEVALUE is like TIME, but takes a text argument. HOUR, MINUTE, and SECOND convert serial numbers into hours, minutes, or seconds. NOW returns the serial number of the current date and time.

## TIMEVALUE(time\_text)

Returns the serial number of the time specified by *time\_text*. *Time\_text* can be in any of the Microsoft Excel time formats.

The serial number is a decimal fraction ranging from 0 through 0.9999, representing the times from 0:00:00 or 12:00:00 A.M. to 23:59:59 or 11:59:59 P.M. If *time\_text* contains any date information, it is ignored.

## Examples

TIMEVALUE("2:24 AM") equals 0.1

TIMEVALUE ("22-Aug-55 2:24 AM") equals TIMEVALUE ("2:24 AM") equals 0.1

## Related Functions

TIME is like TIMEVALUE, but takes numeric arguments. HOUR, MINUTE, and SECOND convert serial numbers into hours, minutes, or seconds. NOW returns the serial number of the current date and time.

## TRANSPOSE(array)

Returns the transpose of *array*. The transpose of an array is created by using the first row of the array as the first column of the new array, the second row of the array as the second column of the new array, and so on.

### Note

Formulas that return arrays must be entered as array formulas. To enter an array formula, press CONTROL+SHIFT+ENTER, or, with a mouse, press CONTROL+SHIFT while you click the check box in the formula bar. For information on arrays, see Array in *Microsoft Excel Reference*.

## Examples

	A	B	C	D	E	F	G	H
1	1	2	3					
2	4	5	6					
3								
4	1	4						
5	2	5						
6	3	6						
7								

TRANSPOSE(A1:C2) *equals* the values in A4:B6

If A1:A5 contains the numbers 10, 7, 9, 27, and 2; that is, the array {10;7;9;27;2}, then:

TRANSPOSE(A1:A5) *equals* {10,7,9,27,2}

## Related Functions

MDETERM returns the determinant of an array. MINVERSE returns the matrix inverse of an array. MMULT returns the matrix product of two arrays.

## TREND(*known\_y's*,*known\_x's*,*new\_x's*)

Fits a straight line (using the method of least squares) to the arrays *known\_y's* and *known\_x's*. Then it returns the y-values along that line for the array of *new\_x's* that you specify.

The array *known\_x's* can include one or more sets of variables. If only one variable is used, *known\_y's* and *known\_x's* can be a range of any shape as long as they have the same dimensions. If more than one variable is used, *known\_y's* must be a vector (a range with a height or width of 1). If the array *known\_y's* is in a single column, then each column of *known\_x's* is interpreted as a separate variable. If the array *known\_y's* is in a single row, then each row of *known\_x's* is interpreted as a separate variable.

If *new\_x's* is included, one dimension of that array must be the same as *known\_x's*. If *known\_y's* are in a single column, *known\_x's* and *new\_x's* should have the same number of columns. If *known\_y's* are in a single row, *known\_x's* and *new\_x's* should have the same number of rows. If you omit *new\_x's*, it is assumed to be the same as *known\_x's*. If you omit both *known\_x's* and *new\_x's*, they are assumed to be the array {1,2,3,...}, of the same size as *known\_y's*.

For information on how Microsoft Excel fits a line to data, see LINEST.

You can use TREND for polynomial curve fitting by regressing against the same variable raised to different powers. For example, suppose column A contained y-values and column B contained x-values. You could enter  $x^2$  in column C,  $x^3$  in column D, and so on, then regress columns B through D against column A.

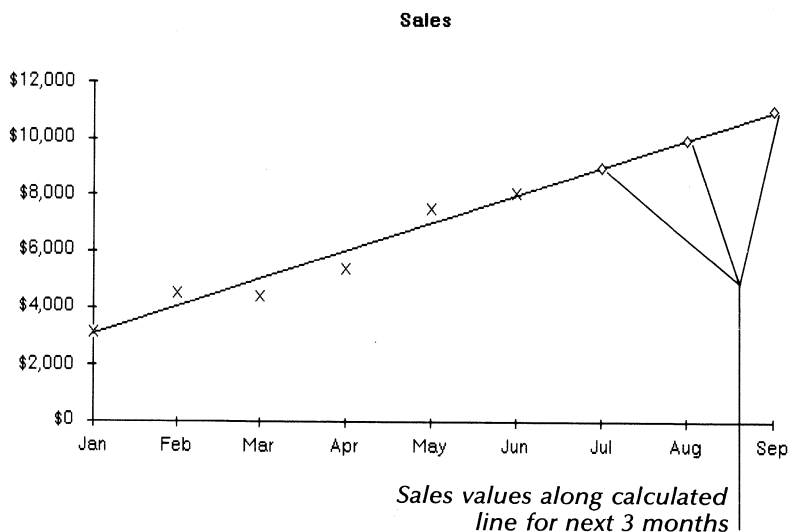
**Note**    Formulas that return arrays must be entered as array formulas. To enter an array formula, press CONTROL+SHIFT+ENTER, or, with a mouse, press CONTROL+SHIFT while you click the check box in the formula bar. For information on arrays, see Array in *Microsoft Excel Reference*.

### Examples

	A	B	C	D	E	F	G	H
1	Month	Sales						
2	January	\$3,100						
3	February	\$4,500						
4	March	\$4,400						
5	April	\$5,400						
6	May	\$7,500						
7	June	\$8,100						
8								

Suppose you had the data in the worksheet above, and wanted to project what sales would be for the months July, August, and September if sales continued to grow at the same rate. The dollar values are the *known\_y's*, the months are the *known\_x's*. The months are equally spaced, so TREND uses {1,2,3,4,5,6} if you don't specify *known\_x's*. The next 3 months in the series, 7, 8, and 9, are the *new\_x's*. Use the following TREND formula:

TREND(B2:B7,,{7,8,9}) equals {9000,10000,11000}



You can also enter an array constant as an argument, instead of a reference. If your monthly sales figures for the first 6 months of the year are \$3100, \$4500, \$4400, \$5400, \$7500, and \$8100, the trend of your sales is calculated by:

**TREND({3100,4500,4400,5400,7500,8100}) equals**  
**{3000,4000,5000,6000,7000,8000}**

If you want to project your sales ahead to the next 3 months, then:

**TREND({3100,4500,4400,5400,7500,8100},,{7,8,9}) equals**  
**{9000,10000,11000}**

Data obtained for a study of the volume at which musicians play.

	A	B	C	D	E	F
1	Volume	Expected	Musician's	Resulting		
2	Marking	Volume	Lung Capacity	Volume of Note		
3	mf	8.0	2.4	6.4		
4	mf	8.0	3.5	7.0		
5	mf	8.0	4.0	8.1		
6	mf	8.0	2.8	8.0		
7	f	10.0	3.2	11.2		
8	f	10.0	3.8	9.9		
9	f	10.0	3.0	9.1		
10	ff	12.0	4.2	13.2		
11	ff	12.0	6.0	14.1		
12	ff	12.0	5.0	14.0		
13						

In this study, the resulting y-value is the resulting volume of the note. This variable depends on two other variables: the expected volume of the note and each musician's lung capacity. You can use this data with the TREND function to predict how loudly musicians with a certain lung capacity will play a note marked at a certain volume.

For example, to use this data to predict how loudly a musician with a 3.5 lung capacity will play a note with an expected volume of 10, use:

`TREND(D3:D12,B3:C12,{10,3.5})` *equals* 10.2

To use this data to predict how loudly a musician with a lung capacity of 3.5 will play notes marked at volumes of 4, 6, 8, and 10, use:

`TREND(D3:D12,B3:C12,{4,3.5;6,3.5;8,3.5;10,3.5})` *equals* {2.2;4.9;7.5;10.2}

### Related Functions

`LINEST` also calculates a line, but it returns the parameters of the line instead of an array of y-values. `GROWTH` and `LOGEST` are analogous to `TREND` and `LINEST`, but they fit your data to an exponential curve.

## TRIM(text)

Removes all spaces from *text*, except for one space between words.

### Examples

`TRIM(" the final frontier. . .")` *equals* "the final frontier. . ."

`TRIM(" 4 score and 7 years ago,")` *equals* "4 score and 7 years ago,"

### Related Functions

`REPLACE` and `SUBSTITUTE` replace characters within text. `CLEAN` removes control characters from text.

## TRUE()

Returns the logical value `TRUE`.

### Example

`TRUE()` *equals* `TRUE`

## TRUNC(number)

Truncates *number* to an integer by removing the fractional part.

### Examples

TRUNC(8.9) *equals* 8

TRUNC(−8.9) *equals* −8

### Related Functions

INT rounds a number down to the nearest integer. ROUND rounds a number to a specified number of digits. MOD returns the remainder from division.

## TYPE(value)

Returns a number indicating the data type of *value*.

If value is	TYPE returns
A number	1
Text	2
A logical value	4
An error value	16
An array	64

### Examples

If A1 contains the text “Smith”, then:

TYPE(A1) *equals* TYPE(“Smith”) *equals* 2

TYPE(“Mr. ”&A1) *equals* 2

TYPE(2 + A1) *equals* TYPE(#VALUE!) *equals* 16

TYPE({1,2;3,4}) *equals* 64

## Related Functions

CELL and the ISfunction return information about a specified cell.

## UPPER(text)

Converts all lowercase letters in *text* to uppercase.

### Example

UPPER("What did you say?") *equals* "WHAT DID YOU SAY?"

## Related Functions

PROPER capitalizes the first letter in each word of a text value. LOWER makes every letter in a text value lowercase.

## VALUE(text)

Converts *text* to a number.

*Text* can be in any of the constant number, date, or time formats recognized by Microsoft Excel. If it is not in one of those formats, VALUE returns the error value #VALUE!.

It is not generally necessary to use the VALUE function in a formula, since Microsoft Excel automatically converts the text to numbers, as necessary. This function is provided for compatibility with other worksheet programs. For information on how Microsoft Excel translates values, see "Translating Data Types" in Chapter 1, "Worksheet Function Basics."

### Examples

VALUE("\$1,000") *equals* 1000

VALUE("16:48:00") – VALUE("12:00:00") *equals* "16:48:00" – "12:00:00"  
*equals* 0.2

This is the serial number equivalent to 4 hours and 48 minutes.



## Related Functions

DOLLAR, FIXED, and TEXT all format a number and convert it to text.

### VAR(number1,number2,...)

Calculates the variance of a population based on a sample given as arguments.

#### Note

VAR assumes that its arguments are a sample of the population. If your data represents the entire population, you should compute the variance using VARP.

VAR can have from 1 to 14 arguments. All text, logical, or empty cell values cause errors.

VAR uses the following formula:

$$\frac{n\sum(x^2) - (\sum x)^2}{n(n-1)}$$

### Example

	A	B	C	D	E	F	G	H
1	February	March	April	May	June	July		
2	5	15	30	40	50			
3								

In the worksheet above:

VAR(A2:F2) equals 332.5

## Related Functions

VARP calculates the variance when you have all the data for a population. STDEV and STDEVP calculate standard deviation. DVAR calculates variance from selected database entries.

### VARP(number1,number2,...)

Calculates the variance of a population based on the population given as arguments.

**Note** | VARP assumes that its arguments are the entire population. If your data represents a sample of the population, you should compute the variance using VAR.

VARP can have from 1 to 14 arguments. All text, logical, or empty cell values cause errors.

## Example

	A	B	C	D	E	F	G	H
1	February	March	April	May	June	July		
2	5	15	30	40	50			
3								

In the worksheet above:

VARP(A2:F2) equals 266

## Related Functions

VAR estimates the variance from a sample. STDEVP and STDEV calculate standard deviation. DVARP calculates variance from selected database entries.

## VLOOKUP(lookup\_value,table\_array,col\_index)

Looks in *table\_array* for a row whose first column contains *lookup\_value*, moves across the row according to *col\_index*, and returns the value of the cell. A *col\_index* of 1 returns the first column value in *table\_array*, a *col\_index* of 2 returns the second column value in *table\_array*, and so on.

The values in the first column of *table\_array* can be text, numbers, or logical values. They must be placed in ascending order: . . . -2, -1, 0, 1, 2, . . . , A-Z, FALSE, TRUE; otherwise VLOOKUP may not give the correct value. Upper- and lowercase text are equivalent.

If VLOOKUP can't find the *lookup\_value*, it uses the largest value which is less than or equal to *lookup\_value*. If *lookup\_value* is smaller than the smallest value in the first column of *table\_array*, VLOOKUP returns the error value #N/A.

VLOOKUP returns the error value #VALUE! if *col\_index* is less than 1 and the error value #REF! if *col\_index* is greater than the number of columns in *table\_array*.

Examples

	A	B	C	D	E	F
1	Air at 1 atm pressure					
2	Density	Viscosity	Temp			
3	(kg/cubic m)	(kg/m*s)*1E+05	(degrees C)			
4	0.457	3.55	500			
5	0.525	3.25	400			
6	0.616	2.93	300			
7	0.675	2.75	250			
8	0.746	2.57	200			
9	0.835	2.38	150			
10	0.946	2.17	100			
11	1.09	1.95	50			
12	1.29	1.71	0			
13						

In the worksheet above:

VLOOKUP(1,A4:C12,1) equals 0.946

VLOOKUP(1,A4:C12,2) equals 2.17

VLOOKUP(1,A4:C12,3) equals 100

VLOOKUP(0.1,A4:C12,2) equals #N/A, because 0.1 is less than the smallest value in column A

VLOOKUP(2,A4:C12,2) equals 1.71

Table\_array can also be an array constant:

VLOOKUP("d",{1,2,3;"a","b","c";"d","e","f"},2) equals "e"

Related Functions

HLOOKUP looks in the first row of an array and moves down the column to give cell contents. LOOKUP and MATCH also look up values in an array or reference. INDEX uses an index to choose a value from a reference or array.

WEEKDAY(serial\_number)

Returns the number of the day of the week corresponding to serial\_number. The day is given as an integer, ranging from 1 (Sunday) to 7 (Saturday).

Serial\_number is the date-time code used by Microsoft Excel for date and time calculations. It ranges from 1 to 65380, corresponding to the dates January 1, 1900 through December 31, 2078. The numbers to the right of the decimal point in serial\_number represent the time; the numbers to the left represent the date. For example, the date-time combination 12:00 P.M., January 1, 1901 is represented as 367.5.

*Serial\_number* may be given as text, such as “4-15-1985” or “15-Apr-1985”, instead of a number. The text is automatically converted to a serial number.

### Note

This description assumes that the 1904 Date System check box in the Options Calculation dialog box is turned off. If this check box is turned on, then serial number 1 in the active document is the start of day January 2, 1904, instead of the start of day January 1, 1900. The 1904 Date System box is turned on automatically if you open a document from Microsoft Excel for the Macintosh, which has a different date system. You can also turn on the check box yourself if you are creating a document to be used in Microsoft Excel for the Macintosh. For more information, see *Date* in *Microsoft Excel Reference*.

## Examples

WEEKDAY(“2/14/87”) equals 7

WEEKDAY(31975) equals 6

WEEKDAY(“23-December-1987”) equals 4

If your computer’s built-in clock is set to 1988, then:

WEEKDAY(“5-Jul”) equals 3

## Related Functions

YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND convert serial numbers into years, months, days, hours, minutes, or seconds. NOW returns the serial number of the current date and time.

## YEAR(serial\_number)

Returns the year corresponding to *serial\_number*. The year is given as an integer, ranging from 1900 to 2078.

*Serial\_number* is the date-time code used by Microsoft Excel for date and time calculations. It ranges from 1 to 65380, corresponding to the dates January 1, 1900 through December 31, 2078. The numbers to the right of the decimal point in *serial\_number* represent the time; the numbers to the left represent the date. For example, the date-time combination 12:00 P.M., January 1, 1901 is represented as 367.5.

*Serial\_number* may be given as text, such as “4-15-1985” or “15-Apr-1985”, instead of a number. The text is automatically converted to a serial number.

**Note** | This description assumes that the 1904 Date System check box in the Options Calculation dialog box is turned off. If this check box is turned on, then serial number 1 in the active document is the start of day January 2, 1904, instead of the start of day January 1, 1900. The 1904 Date System box is turned on automatically if you open a document from Microsoft Excel for the Macintosh, which has a different date system. You can also turn in the check box yourself if you are creating a document to be used on Microsoft Excel for the Macintosh. For more information, see Date in *Microsoft Excel Reference*.

## Examples

YEAR(.007) *equals* 1900

YEAR(29747.007) *equals* 1981

YEAR("7/5/90") *equals* 1990

If your computer's built-in clock is set to 1989, then:

YEAR(NOW()) *equals* 1989

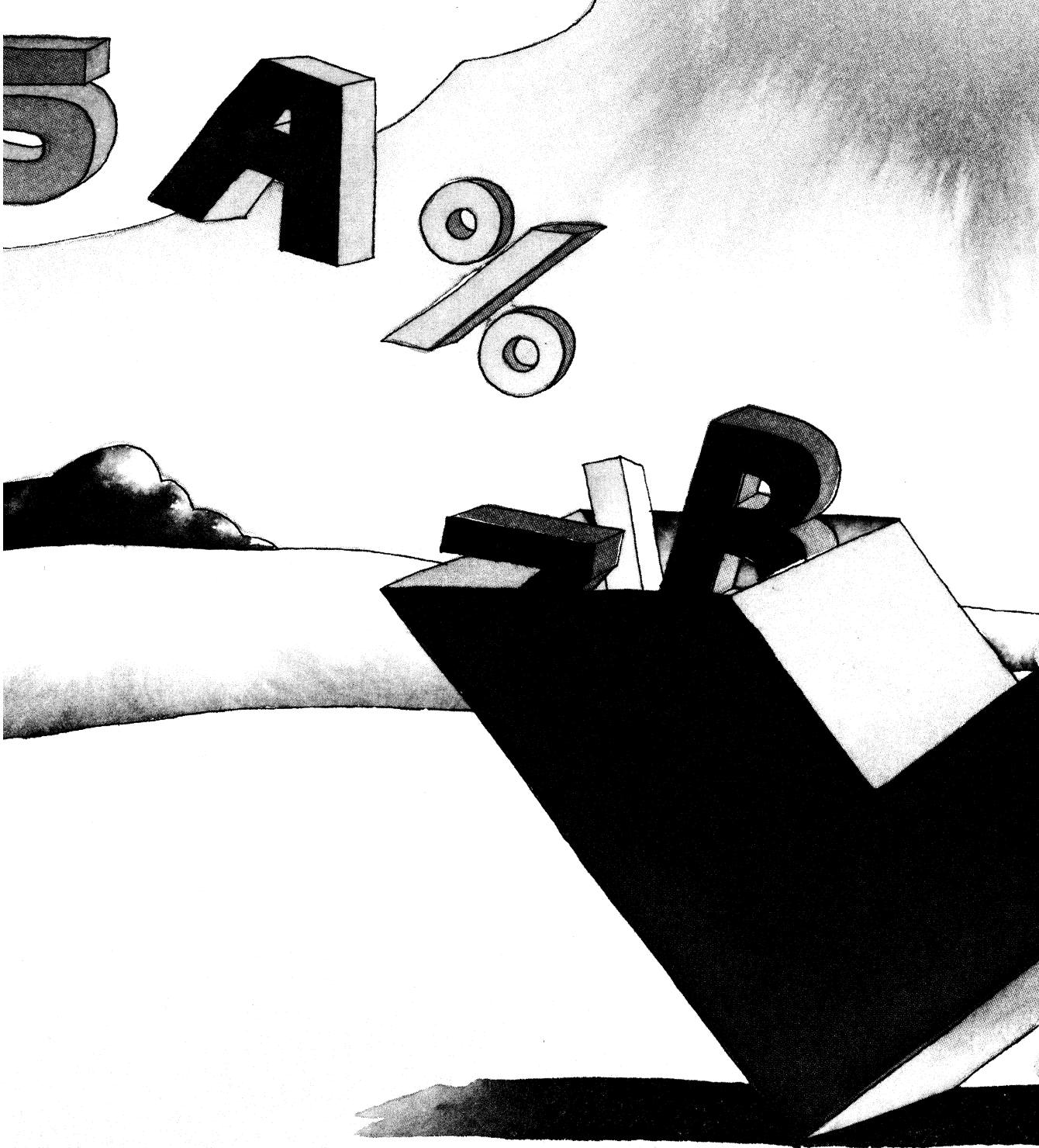
## Related Functions

MONTH, DAY, WEEKDAY, HOUR, MINUTE, and SECOND convert serial numbers into months, days, weekdays, hours, minutes, or seconds. NOW returns the serial number of the current date and time.



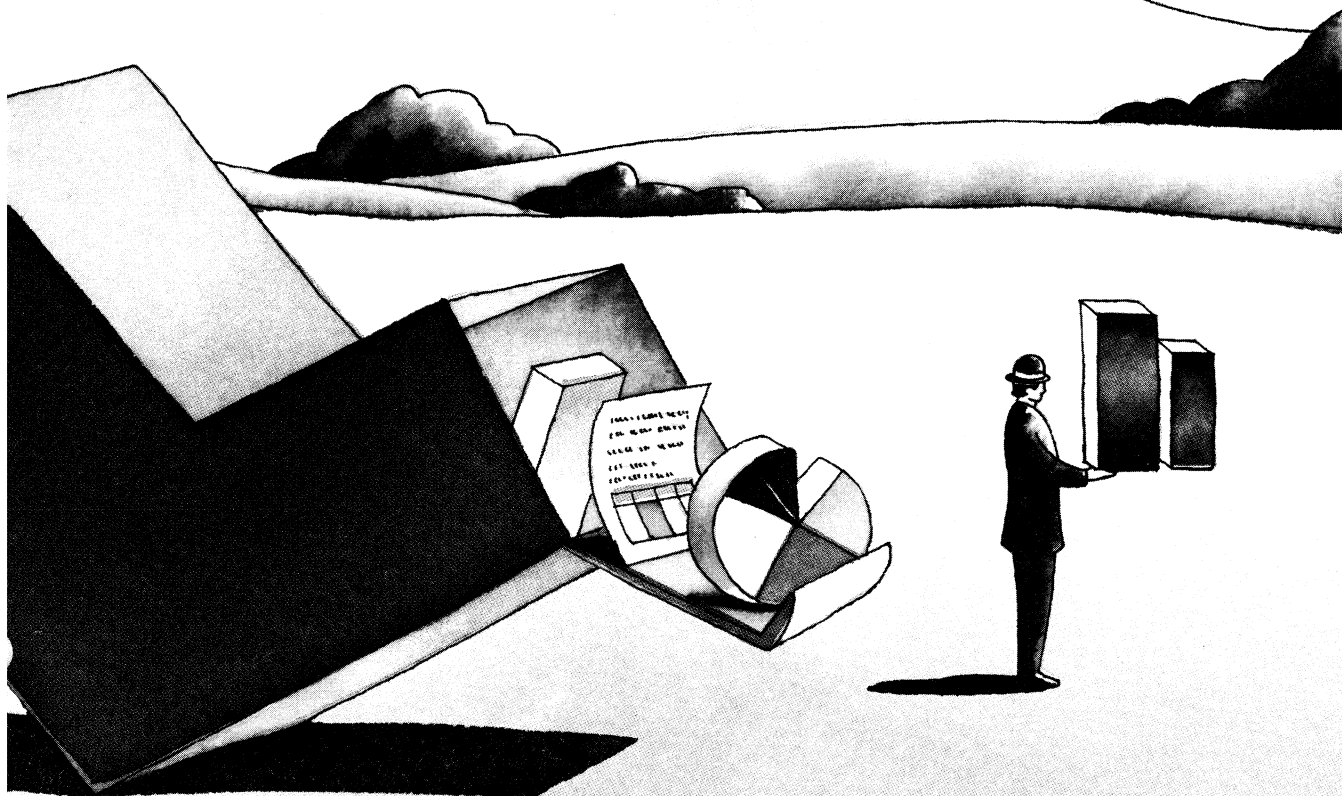
**M**acros

---

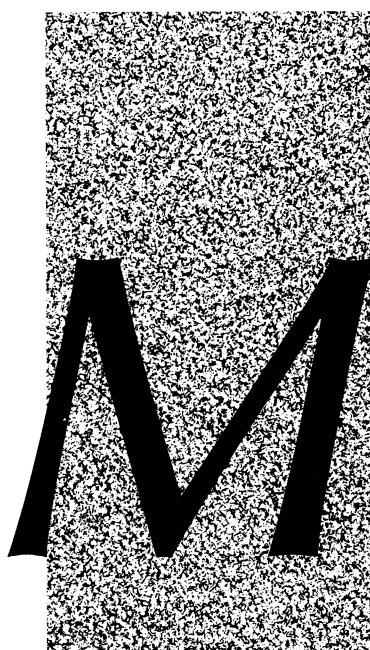




# Macros







## Chapter 3

.....

# Macro Basics

What Is a Macro? . . . . .	130
When Are Macros Useful? . . . . .	131
How Macros Work . . . . .	132
Running a Command Macro . . . . .	133
Running a Function Macro . . . . .	135
Recording a Command Macro . . . . .	136
Absolute and Relative Recording . . . . .	137
Using Macro Sheets . . . . .	139
Editing Macro Sheets . . . . .	140
Organizing Macro Sheets . . . . .	140
Documenting Macro Sheets . . . . .	141
Formatting Macro Sheets . . . . .	142
Sample Macro Sheets . . . . .	142
Modifying a Recorded Command Macro . . . . .	144
Dialog Box Functions . . . . .	145
INPUT Function . . . . .	145
ALERT and MESSAGE Functions . . . . .	147
WAIT Function . . . . .	147
IF Function . . . . .	148
Value-returning Macro Functions . . . . .	149
Using Values from Documents . . . . .	150
Changing Macro Structure . . . . .	150

This chapter explains:

- What macros are, what they look like, and how they work
- How to use an existing macro
- How to create a macro by recording
- How to make simple modifications to a recorded macro

### Tip

For an online introduction to macros, take a look at the Macros Module in the Microsoft Excel Feature Guide, or go through the following lessons in the Microsoft Excel Tutorial:

- What Is a Macro?
- Using a Command Macro
- Recording a Command Macro

### Lotus 1-2-3 User's Note

There's a utility program included in your Microsoft Excel package that can help you translate Lotus 1-2-3 macros into Microsoft Excel macros. The program is called the Macro Translation Assistant. To get information about the Macro Translation Assistant:

- 1 Choose the Help Index command. If Help is already running, choose the Index button in the Help window.
- 2 Choose Macro Translation Assistant from the index.
- 3 Choose Overview.

## What Is a Macro?

A **macro** is a set of instructions that you can create for Microsoft Excel to follow. Macros give you a short way of telling Microsoft Excel to do something.

Think of it this way: imagine you've hired a new assistant. The first time you want your assistant to prepare the month-end report, you have to explain a long series of instructions. You may have to say, "Gather the following data from these six people, calculate trends, make a chart, print the chart," and so on. After your assistant has had some experience, you can just say "Prepare the month-end report." One instruction is enough, because the list of steps is now stored in your assistant's mind.

Macros work a lot like that, except the list of steps is stored on a **macro sheet**. Macro sheets look a lot like worksheets.

Part of a macro sheet that contains a macro named Month.End.

	A	B	C	D	E	
1	Month.End	<i>Prepares month-end report</i>				
2	=OPEN("PRICE.XLS")	<i>Opens worksheets</i>				
3	=OPEN("EAST.XLS")					
4	=OPEN("SOUTH.XLS")					
5	=OPEN("WEST.XLS")					

Once you have written a macro on a macro sheet, it's easy to tell Microsoft Excel to carry out the steps contained in the macro. You can create macros yourself, or use macros written by other people. There are also some sample macros included on the Library disk.

Lotus 1-2-3  
User's Tip

In Microsoft Excel, macros are stored on macro sheets instead of on your worksheet. This makes it easy to use the same macros with many different worksheets and charts, and also saves space on your worksheets.

There are two kinds of macros. Our Month.End macro is an example of a **command macro**. A command macro carries out a sequence of actions. They're called command macros because of their similarity to the commands built into Microsoft Excel. If you can think of a command that you wish Microsoft Excel had, you may be able to write a command macro to create that command yourself. For example, you could write a command macro that centers text in selected cells and makes that text bold, or a command macro that writes today's date in the corner of your worksheet and then prints the worksheet.

The other kind of macro is called a **function macro**. A function macro computes a value. Function macros are like Microsoft Excel worksheet functions. For example, do you wish Microsoft Excel had a worksheet function that would take the number of items a customer has ordered, add sales tax and shipping and handling charges, and then tell you the total price? Or would you like to have a worksheet function that converts measurements in furlongs per fortnight to miles per hour? You can write a function macro to do it for you.

# When Are Macros Useful?

In the example of the Month.End macro that prepares a month-end report, a macro is a convenient way to tell Microsoft Excel how to do a task that you perform often.

**Tip** | Anytime you find yourself frequently repeating an action in Microsoft Excel, it's likely that a macro can save you some time and effort.

Since you can run command macros with only two keystrokes, even macros that perform one action, such as making selected cells bold, can be convenient.

Automating a frequent task is a very good use of macros, but not the only one. You can also use macros in the following ways:

- **To simplify a complex worksheet.** If you work with a very large worksheet, or a worksheet that takes too long to calculate, you can probably use macros to simplify your system. See the example in the section “Changing Macro Structure” later in this chapter.
- **To create customized menus and dialog boxes.** You can set up applications that run along with Microsoft Excel to do specialized tasks. Do you work with someone who doesn't know much about worksheets, for example, but who could save you time if they could enter or extract their own data? You may want to consider writing a special application for that person, using Microsoft Excel macros. For more information, see “Creating Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”
- **To run other applications under Microsoft Windows.** If you have Microsoft Windows (version 2.0), you can use Microsoft Excel macros to run other applications and to move data between Microsoft Excel and other applications. Software developers can also use macros to run procedures in a Microsoft Windows dynamic library. For more information, see “Using Macros to Call Other Applications” in Chapter 6, “Advanced Macros.”

## How Macros Work

Let's take a look at how you would use a particular macro, and what happens when you use it. Suppose you need to make several scatter charts, and a friend gives you a copy of a macro sheet containing a command macro that helps make scatter charts.

The first step in using any macro is to make sure that the macro sheet containing the macro is open. To open an existing macro sheet:

- 1 Choose File Open.
- 2 Select the macro sheet you want to open.
- 3 Choose the OK button.

**Note** You can run a macro from any open macro sheet, even if that macro sheet is hidden.

**Tip** Most macro files have the “.XLM” extension. To list all the files with this extension, type \*.XLM in the text box of the File Open dialog box.

You open the macro sheet and find the command macro, which is named Scatter.

	A	B	C
1	Scatter	Makes scatter chart from selection	
2		Select data before running the macro	
3		Shortcut key=s	
4	=COPY()	Copies the current selection	
5	=NEW(2)	Opens chart	
6	=PASTE.SPECIAL(2,FALSE,TRUE)	Pastes chart	
7	=GALLERY.SCATTER(2)	Chooses type of scatter chart	
8	=RETURN()		
9			

Macro instructions

Macro name

These comments describe how the macro works.

## Running a Command Macro

There are two basic ways to run a command macro: first, make sure the macro sheet containing the macro is open. Then, either:

- 1 Choose Macro Run.
- 2 Select the name of the command macro you want to run (in this case, “CHARTS.XLM!Scatter”).
- 3 Choose the OK button.

or:

- Use a **shortcut key**.

Shortcut keys give you a quick way to run command macros — just press CONTROL and the shortcut key. If you want to use a shortcut key for running the macro, type a letter in the box labeled “Key: Ctrl +” when you are defining a name for the macro.

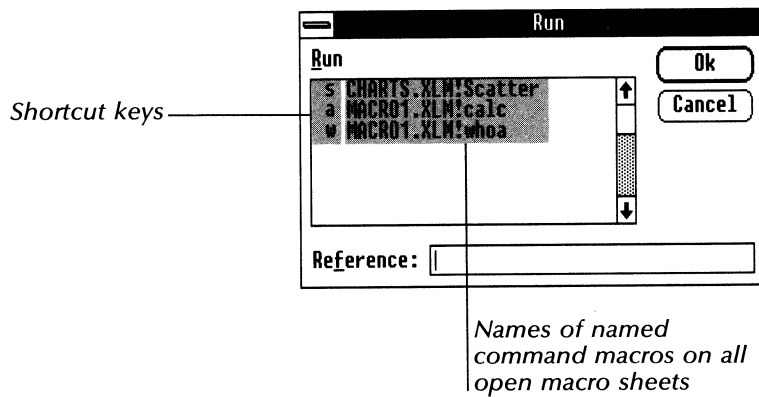
**Note** To make a reference to a macro sheet other than the active sheet, specify the name of the sheet, followed by an exclamation point (!), followed by the reference on that sheet. For example, CHARTS.XLM!Scatter refers to the name Scatter on the macro sheet CHARTS.XLM. PRICES.XLM!A31 refers to cell A31 on the macro sheet PRICES.XLM.

For more information, see “Using References” in Chapter 4, “Writing Macros,” or References in *Microsoft Excel Reference*.

**Note** | Microsoft Excel distinguishes between upper- and lowercase shortcut keys.

The person who wrote the Scatter macro did a good job of documenting their macro—notice that in cell B3 they wrote down what the shortcut key is. In this case, you can run the Scatter macro from a worksheet by pressing CONTROL+S.

**Tip** | Sometimes you may need to figure out for yourself if there's a shortcut key for a particular macro. You may have forgotten to write down the shortcut, or you may suspect that somebody wrote down the wrong shortcut. To check, just choose the Macro Run command and look in the dialog box.



There are some other ways to run command macros. For more information, see “Running Macros Automatically” and “Creating Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

What happens when you run the Scatter macro? First, Microsoft Excel looks on the CHARTS.XLM macro sheet and finds the Scatter macro.

**Note** | If more than one macro has the same shortcut key, the first macro listed in the Macro Run dialog box is used.

Then Microsoft Excel starts reading the cells in the macro, one at a time, and doing whatever the formula says to do.



Starts here. →  
Performs each step in order. ↓

	A	B	C	
1	Scatter	Makes scatter chart from selection		
2		Select data before running the macro		
3		Shortcut key=s		
4	=COPY()	Copies the current selection		
5	=NEW(2)	Opens chart		
6	=PASTE SPECIAL(2,FALSE,TRUE)	Pastes chart		
7	=GALLERY.SCATTER(2)	Chooses type of scatter chart		
8	=RETURN()			
9				

Many macros work just like this—they start at the top and run down a column until they reach the end of the macro. If you write a macro, or modify a recorded macro, you can structure macros in other ways as well. For more information, see the section “Modifying a Recorded Command Macro” later in this chapter.

## Running a Function Macro

Function macros run very much like command macros, but you start them differently. You don’t choose the Macro Run command, or press a shortcut key. You use a function macro the same way you use the worksheet functions that are built into Microsoft Excel, with one exception: the macro sheet that the function macro is written on must be open before you can use the macro.

To run a function macro:

- 1 Open the macro sheet.
- 2 Either:
  - Choose the Formula Paste Function command and select the function name (user-defined functions are at the bottom of the list), then enter any arguments.
  - or:
  - Type an equal sign, followed by the name of the function (the external reference of the first cell in the macro), followed by any arguments enclosed in parentheses.

If there’s more than one argument, separate the arguments by commas. If you leave out an argument, Microsoft Excel uses the #N/A error value as the value of the argument.

For example, suppose you wanted to use this simple function macro on the GEO.XLM macro sheet, which calculates the surface area of a sphere.

	A	B	C	D	E
1	<b>Sphere.Surface</b>	<i>Calculates surface area of sphere</i>			
2	=ARGUMENT("radius")	Given radius of sphere			
3	=4*PI()*radius^2	Area=4*pi*(r squared)			
4	=RETURN(A3)	Return area			
5					

To calculate the surface area of a sphere with a radius of 2, enter the following formula on your worksheet: =GEO.XLM!Sphere.Surface(2)

### Tip

If you can't find the name of a macro you want in the appropriate dialog box listing:

- 1 Make sure that the macro sheet is open.
- 2 Make sure the macro name is defined: choose Formula Define Name and look for the macro name in the dialog box.

If the name isn't defined, you can use the Formula Define Name command to name the macro yourself. If you are trying to run a command macro, you can run the macro even if it has no name. Just enter the reference of the first cell of that macro in the Macro Run dialog box.

## Recording a Command Macro

There's an easy way to create command macros, called **recording**. When you record a macro, you turn on the recorder and Microsoft Excel records all the actions you take until you tell Microsoft Excel to stop.

## Opening Macro Sheets Automatically

If you want to open macro sheets automatically (or "attach" a macro sheet to another document), you can do this one of two ways:

### ■ Create a workspace file.

If you use several files together, try saving your workspace. Then when you want to use the files again, you open just one file, the workspace file, and Microsoft Excel opens all the files for you. For more information, see *Workspace file* in *Microsoft Excel Reference*.

### ■ Create an autoexec file.

An autoexec file specifies that a specific macro be run every time you open or close a given document. Suppose you always want to open a certain macro sheet when you open a certain worksheet. You can specify a macro on the macro sheet as the autoexec macro for a worksheet. For more information on autoexec files, see "Running Macros Automatically" in Chapter 6, "Advanced Macros."

**Note** | You can't record function macros, because they don't perform actions, they make calculations.

To record a command macro:

- 1 Choose Macro Record.
- 2 Microsoft Excel displays a dialog box asking for a name and shortcut key. Make any changes you want in the proposed names, then choose the OK button.  
If your status bar is on, the word "Recording" appears there. For information on the messages displayed in the status bar, see *Status bar in Microsoft Excel Reference*.

**Note** | Make sure that the name you specify is a legal Microsoft Excel name, and that the shortcut key is a letter.

Microsoft Excel now opens a new macro sheet, behind any other open documents, and gives your macro the name that you've specified. (If you have already recorded a macro, the recorder doesn't open a new macro sheet—it records the macro in the first empty column on the already opened macro sheet.)

- 3 Carry out whatever actions you want recorded.  
Microsoft Excel records these actions on the macro sheet.
- 4 Choose Macro Stop Recorder.

Now you can take a look at the macro. It's a good idea to format and document it as discussed in the next section, "Using Macro Sheets." This makes your macro easier to read.

For information on recording macros, see "Recording a Command Macro" in Chapter 4, "Writing Macros."

## Absolute and Relative Recording

When you record a command macro, you can choose between two settings for the recorder: Relative Record and Absolute Record. When Relative Record is on, a selected cell is recorded using a relative reference. When Absolute Record is on, Microsoft Excel uses absolute references. By default, Microsoft Excel uses Absolute Record. You can change between the two at any time, even in the middle of recording a macro.

To change between absolute and relative recording:

- Choose Macro Relative/Absolute Record.

For example, suppose you are recording a macro with Absolute Record on (the default setting), and the active cell on the worksheet is A1. When you select cells A1:L1, that action is recorded using absolute references. Later, when you run the macro, the macro selects cells A1:L1 no matter what the current active cell is. If, instead, you recorded the macro with Relative Record on, selecting cells A1:L1 is recorded using relative references. If C3 is the active cell when you run the macro, the macro selects cells C3:N3.

## Try Recording a Macro

Suppose you use the custom number format "0.000" on all of your worksheets. To use that format:

- 1 Select the area to format.
- 2 Choose Format Number.
- 3 Choose 0.000 (or enter 0.000, if you haven't used the format before).
- 4 Choose the OK button.

That's only four steps, but you can do it even faster if you record a simple macro that applies the format to a selected area. With such a macro, to use a custom number format:

- 1 Select the area to format.
- 2 Run the macro.

To record the macro:

- 1 Select the area on your worksheet that you want to format.
- 2 Choose Macro Record.
- 3 The name and shortcut key are proposed in the dialog box, but let's change them to something easier to remember. Change the suggested name Record1 to digi3, and change the suggested CONTROL key from a to d.

- 4 Choose the OK button.
- 5 Choose Format Number.
- 6 Type 0.000 in the text box.
- 7 Choose the OK button.
- 8 Choose Macro Stop Recorder.

To take a look at the macro, choose the Window Macro1 command.

	A	B	C	D	E	F
1	digi3					
2	=FORMATNUMBER("0.000")					
3	=RETURN()					
4						

*The finished macro.*

## Using Macro Sheets

All macros are made up of formulas in cells on a macro sheet. Let's take another look at the macro sheet we saw in the last section.

	A	B	C	D	E	F	
1	digi3						
2	=FORMAT.NUMBER("0.000")						
3	=RETURN()						
4							

How does the macro sheet look different than a worksheet? The most obvious difference is that the macro sheet is displaying formulas, not values. By default, macro sheets display formulas and worksheets display values. You can change this:

- 1 Choose Options Display.
- 2 Choose what you want displayed.

Because formulas are being displayed, the columns are also wider. You can change the column width if you want, just like on a worksheet. By default, it's set wider to make it easier to see your formulas.

You can change formatting, such as fonts and borders, on a macro sheet just like you can on a worksheet. Note, however, that cells are always displayed left-aligned when a worksheet or macro sheet is displaying formulas instead of values.

You may also notice that formulas on macro sheets use some functions other than worksheet functions. You can use all the worksheet functions in macros, as well as another group of functions called **macro functions**. You can learn more about macro functions in "Macro Functions" in Chapter 4, "Writing Macros."

The important difference between macro sheets and worksheets is that Microsoft Excel doesn't calculate an entire macro sheet, the way it calculates a worksheet. The formulas in a macro are executed or run one formula at a time. Command macros aren't run until you choose them, and function macros aren't run until a cell containing a reference to the function macro must be calculated.

**Note** | Macro functions and function macros are not the same.

**Macro functions** are a type of built-in function used only in macros, such as the FILE.DELETE macro function and the FORMAT.NUMBER macro function.

**Function macros** are a type of macro you write. They're called function macros because they behave like the functions that are built into Microsoft Excel. We'll talk more about function macros in the next section.

## Editing Macro Sheets

Editing and file handling are done the same way with macro sheets as with worksheets. For information on documents, see *Editing a document*, *Opening a document*, *Printing a document*, and *Saving and closing a document* in *Microsoft Excel Reference*.

You can make your macros easier to read and understand by:

- Organizing macro sheets
- Documenting macro sheets
- Formatting macro sheets

## Organizing Macro Sheets

You can have many macros on the same macro sheet. If you have several macros that you use frequently, it can be convenient to have them all on one sheet. For example, all the macros you use to generate reports or analyze data can go on one macro sheet. There are no restrictions on how many macros can go in one row or one column — arrange them in whatever way you find convenient.

You can also have more than one macro sheet open at a time. If you use many macro sheets at the same time, you may want to hide one or more of them. You can still run macros on hidden macro sheets, but the macro sheet must be visible to edit it. For information on hiding documents, see *Window Hide command* in *Microsoft Excel Reference*.

**Tip** | You can have Microsoft Excel run macros every time you open or close a particular document, just by defining a special name (Auto\_Open or Auto\_Close). You can use autoexec macros to automatically open the appropriate macro sheets for any given document. For more information, see “Using Autoexecs” in Chapter 6, “Advanced Macros.”

## Documenting Macro Sheets

Using notes, comments, macro names, and cell labels makes it easier to read and understand your macros.

A well-documented macro.

Macro name

Cell label

	A	B	C	D
1	Month.End	<i>Prepares month end report</i>		
2		Shortcut key=m		
3	=OPEN("PRICE.XLS")	Opens worksheets		
4	=OPEN("EAST.XLS")			
5	<i>begin</i>	Initializes values		
6	=SET.NAME("current",EAST.XLS!\$A\$2)			
7				

Comments

Comments, macro names, and cell labels are text that is entered in a cell on your macro sheet. When a macro is running, and encounters a cell that contains a constant value, like text or a number, it skips the constant and goes on to the next cell.

You can use the same name or label on more than one macro sheet, but names and labels on a single macro sheet must be unique.

### Note

Don't use equal signs (=) in labels within a macro. Microsoft Excel interprets those statements in a special way. For more information, see "SET.NAME" in Chapter 7, "Macro Function Directory."

**Notes** Use notes with a macro sheet the same way you would with a worksheet. Notes are useful when lengthy explanations are needed that might be too long to write conveniently on the macro sheet. Notes are also a good way to explain how to use your macro, since anyone can look at a list of all the notes attached to a document. For more information, see Notes in *Microsoft Excel Reference*.

**Comments** Brief explanations, usually written in a column next to your macro. Comments are a good way to document how your macro works.

**Macro Names** If you use a descriptive name in the first cell of your macros, it's easy to identify macros. For more information, see "Naming a Macro" in Chapter 4, "Writing Macros."

When you run a macro by specifying its name, Microsoft Excel finds the named cell, then executes the formula in the cell directly below the named cell, then continues down the column.

**Cell Labels** Text that is entered in a cell and then defined as a name. Once you have defined a label, you can use that label anywhere in your macro to refer to that cell. Using labels to identify key cells in a macro makes your macros easier to understand.

To label a cell:

- 1 Enter text in the cell.

That text should be a legal Microsoft Excel name, starting with a letter and consisting only of letters, numbers, periods (.), and underlines(\_).

- 2 With the cell selected, choose Formula Define Name.

Microsoft Excel suggests using the text you entered in the cell as the cell's name unless the name already exists.

- 3 Choose the OK button.

Now you can refer to that cell by name throughout your macro.

For example, in the macro above, "begin" is the label for cell A5. If you wanted to refer to cell A5 in a macro function, such as GOTO, you could enter either *GOTO(A5)* or *GOTO(begin)*.

## Formatting Macro Sheets

You can use any formatting that you like on a macro sheet. Formatting does not affect the execution of a macro. Removing gridlines and using borders and shading to emphasize certain parts of your macros may make them easier to read. It's also a good idea to use different fonts to make names, comments, and labels stand out. Using the same formatting scheme for all of your macros makes them easier to understand. For more information, see *Formatting a document in Microsoft Excel Reference*.

**Note** | Alignment options don't show up when you're displaying formulas instead of values.

## Sample Macro Sheets

An example of how to set up a macro sheet.

	A	B	C	D
1	<b>Month.End</b>	<i>Prepares month end report</i>		
2		Shortcut key=m		
3	=OPEN("PRICE.XLS")	Opens worksheets		
4	=OPEN("EAST.XLS")			
5	<i>begin</i>	Initializes values		
6	=SET.NAME("current",EAST.XLS!\$A\$2)			
7				

All the macros are written in column A; later macros can be added below the first macro in column A. Notice that the macro starts with a name — formatted in bold. Comments are written in column B, including a comment that tells what the shortcut key is. To make it easier to identify labels in column A, they're formatted in italic.



Another possible layout.

	A	B	C	D
1		<b>Month.End</b>	<i>Prepares month end report</i>	
2			Shortcut key=m	
3		=OPEN("PRICE.XLS")	Opens worksheets	
4		=OPEN("EAST.XLS")		
5	<b>begin</b>		Initializes values	
6		=SET.NAME("current",EAST.XLS!\$A\$2)		
7				

This is a lot like the first system, except all labels are written in column A, the rest of the information is moved one column to the right, and the formatting is slightly different. The advantage of this is that it's very easy to label your cells:

- 1 Select both the label and program columns.
- 2 Choose Formula Create Names.
- 3 Select Left Column.

You can also use borders and shading to make your macros easy to read. In the next example, gridlines are turned off and the entire macro is bordered. Comments are shaded. The shortcut key is in parentheses after the name of the macro. If you put the shortcut key after the macro name and then choose the Formula Define Name command, you may want to edit the name that Microsoft Excel suggests, so that the notation for the shortcut key — in this case, (*m*) — is not included.

	A	B	C	D
1	<b>Month.End(m)</b>	<i>Prepares month end report</i>		
2		Shortcut key=m		
3	=OPEN("PRICE.XLS")	Opens worksheets		
4	=OPEN("EAST.XLS")			
5	<b>begin</b>	Initializes values		
6	=SET.NAME("current",EAST.XLS!\$A\$2)			

## Behind the Scenes

We already know that macro sheets, by default, display formulas instead of values.

You can display values instead of formulas:

- 1 Choose Options Display.
- 2 Turn off the Formulas check box.

Macro names, labels, and comments appear the same whether you're displaying values or formulas. That's because names, labels, and comments are entered as constants, not as formulas.

Macro functions return values, just like worksheet functions. When a macro sheet displays values instead of formulas, you see the values the macro functions returned.

In most cases, you won't be interested in the return value of a macro function, but return values are useful in these instances:

- Return values give you a lot of useful information when you're debugging a macro.
- Return values let you share values easily in your macros.

## Modifying a Recorded Command Macro

The previous section of this chapter explained how to add documentation and formatting to your macros to make them easier to read. In this section we'll discuss some of the other modifications you can make to a command macro.

To	See
Choose something from a command's dialog box	Dialog Box Functions
Type in a value	INPUT Function
Prompt the user	ALERT and MESSAGE Functions
Suspend macros	WAIT Function
Make choices after starting macros	IF Function
Use information about the active sheet, open windows, references, names, the current selection, and so on	Value-returning Macro Functions
Run macros within macros	Changing Macro Structure
Loop, or repeat part of your macro	Changing Macro Structure

For detailed information on any of the macro functions, see Chapter 7, "Macro Function Directory," where functions are listed in alphabetical order.

## Dialog Box Functions

All the command-equivalent functions for commands that bring up dialog boxes have two forms: one that's followed by a question mark, such as OPEN?, and one that's not, such as OPEN. The functions that are followed by question marks are called **dialog box functions**. When a macro reaches a dialog box function, the macro displays the dialog box. You can use arguments to specify suggested responses. If you don't specify any suggested responses, Microsoft Excel provides standard responses in the dialog box.

When you record a macro, the recorder always uses the form of the function that's not followed by a question mark, and records the options that you used while you were recording the function. If you need to make different choices from a dialog box while your macro is running, just add a question mark after the appropriate macro function.

For example, if while you were recording you opened a macro sheet called "CWB.XLM" in your root directory on drive C:, when you were done you'd see the following macro function in your command macro:

```
= OPEN("C:CWB.XLM")
```

If you usually want to open CWB.XLM, but you'd like to keep the option of choosing a different document when you run the macro, just edit the OPEN macro function to:

```
= OPEN?("C:CWB.XLM")
```

CWB.XLM will be the proposed response in the dialog box, but you'll be able to choose any document.

If you don't want to use CWB.XLM as the proposed response, edit the OPEN macro function to:

```
= OPEN?()
```

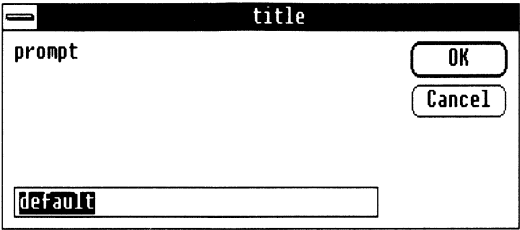
The dialog box will be displayed with its default proposed response.

## INPUT Function

If you want your macro to get a single value, array, or formula, use the INPUT function.

**Note** | To get more detailed or lengthy input, you may want to write a customized dialog box. For more information, see “Creating Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

The dialog box displayed by the INPUT function.



You can enter cell references in the text box, type text or numbers, or choose names or functions with the Formula Paste Name and Formula Paste Function commands. The second argument to INPUT, *type*, specifies what type of values INPUT will accept. You can also move the dialog box.

If you choose the OK button or press ENTER, the INPUT function returns whatever was entered in the edit box. If you choose the OK button without entering anything in the edit box, the INPUT function returns *default*. If you choose the Cancel button, the INPUT function returns FALSE.

Part of a recorded macro that writes a series of dates in a row on a worksheet.

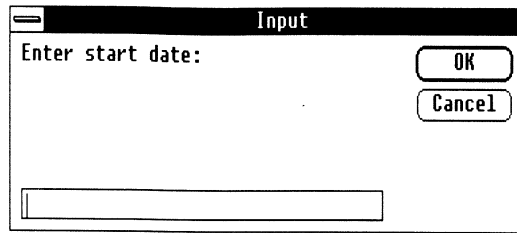
	A	B	C	D	E	F
4	=FORMULA("=NOW()")					
5	=DATA.SERIES(1,3,2,1)					
6	=FORMAT.NUMBER("d-mmm-yy")					
7						

The FORMULA function enters today’s date in the active cell. Then the DATA.SERIES function writes successive weekday dates in the rest of the row, and the FORMAT.NUMBER function formats those dates. Suppose you want to specify start dates other than the current date when you run this macro. What if, for example, you ran the macro on November 2, 1986 but wanted to specify a start date of October 28, 1986. You can modify the FORMULA function to:

=FORMULA(INPUT(“Enter start date:”,1))

The argument 1 specifies that the value entered must be a number.

The dialog box displayed after running the macro.



## ALERT and MESSAGE Functions

The ALERT function is used when you want a particular message to appear each time a macro is run. ALERT displays a dialog box. You must choose the OK or Cancel button before the macro will continue. It can be annoying, however, to have to read too many of these. You may want to use the MESSAGE function instead, which displays text in the status bar. For information on the types of information displayed in the status bar, see Status bar in *Microsoft Excel Reference*.

One practical combination is to start with an ALERT function that tells people to watch the status bar for any messages, and use the MESSAGE function for the rest of your explanations. Or, you may want to use the BEEP function, which beeps, before displaying each new message.

To display a message in the message area of the status bar, enter the following form of the MESSAGE function:

```
=MESSAGE(TRUE,"Message")
```

To stop displaying a message, enter either of the following:

```
=MESSAGE(TRUE,"")
```

```
=MESSAGE(FALSE)
```

The first formula leaves the status bar blank; the second formula returns the status bar to its normal display.

## WAIT Function

If you want your macro to stop for a certain length of time, or until a certain time, use the WAIT function. The WAIT function has the following form:

```
WAIT(serial_number)
```

The WAIT function waits until the time specified by *serial\_number*, then continues. One second is equivalent to about .00001 of a serial number, one minute is equivalent to about .0007 of a serial number, and one hour is equivalent to about .042 of a serial number. You could use the following formula, for example, to make your macro wait for three seconds:

= WAIT(NOW() + .00003)

Or you could use the following formula to make your macro wait until 10:30 P.M.:

= WAIT("10:30 PM")

This can be useful for tasks you want to start late at night, for example, printing a document when the printer is not in use.

The ON.TIME function is also useful for performing tasks at certain times. WAIT keeps Microsoft Excel busy until a specified time. ON.TIME allows you to do other work, then starts a macro at a specified time. For more information, see Chapter 7, "Macro Function Directory," where functions are listed in alphabetical order.

## IF Function

Use the IF function when you want to make a decision within a macro while it's running. The IF function is a worksheet function (you can use all the worksheet functions in macros as well as on worksheets), and takes the following form:

IF(condition,value\_if\_true,value\_if\_false)

Suppose, for example, you want to check two totals on your worksheet and make sure they're identical. The following IF function compares the contents of cells C40 and J40; if they're identical it displays "OK" in the cell and if they're not it displays a dialog box that says "Totals Don't Match."

= IF(C40 = J40,"OK",ALERT("Totals Don't Match"),3))

**Note**

Be careful when you test numbers to see if they are equal. Suppose that cells C40 and J40, in the example above, contain the following formulas respectively:

`= 1.5 + 2.28`

`= 3 + .78`

Cells C40 and J40 would both display the value 3.78, but the test `C40 = J40` might return FALSE instead of TRUE because of the way Microsoft Excel stores numbers. The two numbers can differ by a very small amount, such as 0.000000000000000001, so they are not exactly equal. If you test for equality of numbers, you should first round them to an equal number of decimal places, or you can use the Options Calculation command and select Precision as Displayed. Using this option, however, permanently alters the precision of numbers on your worksheet.

## Value-returning Macro Functions

Value-returning macro functions return certain values that can be very useful when you are modifying or writing a macro. For example, suppose you want to make a decision based on the contents of the active cell. You could use the formula:

`=IF(ACTIVE.CELL() = A1, RETURN())`

Or suppose you want to know the name of the active document, so that you can use the name in a message. The following formula returns the name of the active document as it appears in the title bar, without a path:

`=GET.DOCUMENT(1)`

If this formula was in cell A5, for example, you could then use that name in a formula like this:

`=MESSAGE(TRUE, "Recalculating worksheet "&A5)`

If you find you want access to some information while you're writing a macro, look in the listing of "Macro Functions by Category," in Chapter 7, "Macro Function Directory," and see if there's a value-returning macro function that gives you what you need.

## Using Values from Documents

Even though you can't give values directly to a command macro, you can still use values from documents in a command macro. For example, you can:

- Use value returning macro functions, such as `GET.FORMULA`, to get values from documents. You may want to use functions such as `SET.VALUE` to write values directly onto your macro sheet.
- Cut or copy values and paste them onto a convenient part of your worksheet for your macro to operate on.
- Make external references to worksheets, or use a reference preceded by an exclamation point (!) to refer to the active sheet. For more information, see “Using References” in Chapter 4, “Writing Macros.”

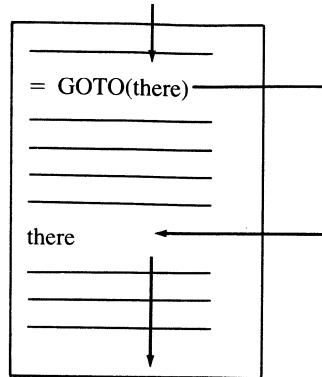
In the special case where a command macro is used only as a subroutine macro, you can send values to and return values from the subroutine macro the same way you pass values to and from function macros.

## Changing Macro Structure

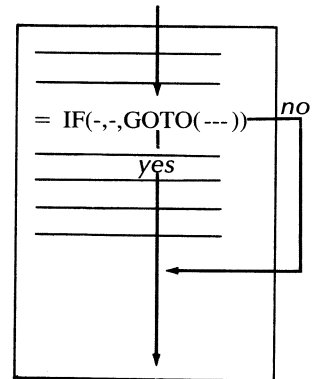
Macros can have structures other than just running straight down a column.



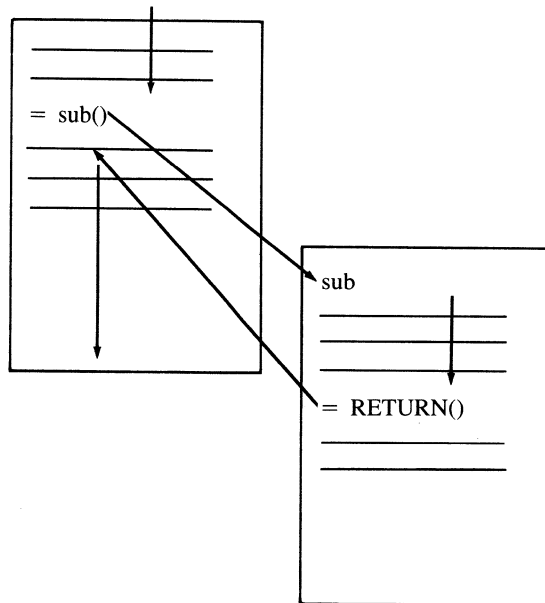
### Jumping



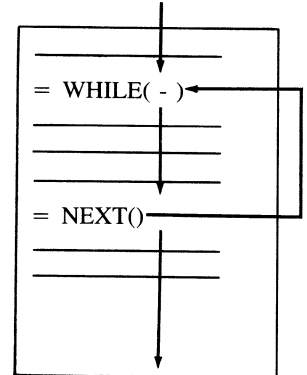
### Branching



### Calling



### Looping



**Jumping** The GOTO macro function redirects execution of a macro to a particular cell. The formula `=GOTO(B1)`, for example, redirects execution to cell B1 on the current macro sheet.

**Branching** The IF function arguments *value\_if\_true* and *value\_if\_false* can be GOTO functions instead of values, which lets you **branch**, or redirect

execution of the macro to a choice of cells. This tests a condition and then goes to a macro or part of a macro (if you go to a label) depending on the results of the test.

For example, suppose you're writing a macro to update your database, and you want to delete the names of anyone who hasn't made a purchase in the last year. You might write a section in your macro that deletes records. Label the first cell in the section "Delete." The following IF function branches to the cell named Delete if the last purchase was made over one year ago:

```
=IF(lastpurchase<NOW()-365,GOTO(Delete))
```

**Starting Subroutine Macros** A subroutine is a macro that is run by another macro. To start a macro, use the following formula:

```
=name_of_macro()
```

If the subroutine macro is on a different macro sheet, *name\_of\_macro* will be an external reference. For example, to start a command macro named Print on a macro sheet named UTIL.XLM, you would use the following formula:

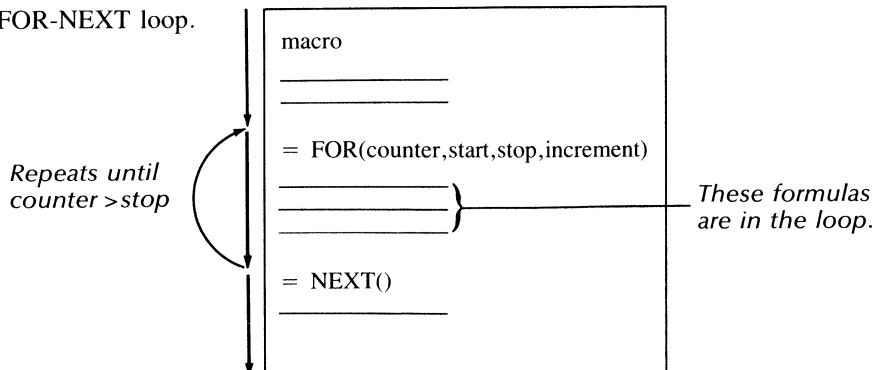
```
=UTIL.XLM!Print()
```

Command macros that are used *only* as subroutine macros can accept and return values just like function macros.

If you're starting a macro that has arguments, enter the arguments between the empty parentheses and separate multiple arguments by commas. If you're calling a macro that does not have arguments, be sure to include the empty parentheses after the name of the subroutine macro.

**Looping** If you want your macro to repeat an action or calculation, use the looping functions FOR, WHILE, NEXT, and BREAK.

A FOR-NEXT loop.



In a FOR-NEXT loop, you set the following values:

- **Counter** — The counter is a name that keeps counting the number of times the loop has run.
- **Increment** — The increment is a value that is added to the counter. You set the counter to a start value, and each time the loop is executed, an increment is added to the counter.
- **Stop value** — The stop value is a number that stops the loop.

For example, look at the following command macro:

Counter is named  
"count."

	A	B	C	D	E	F
4	=FOR("count",1,5,1)					
5	=Select("R[1]C")					
6	=Formula("=AVERAGE9RC[-5].RC[1])")					
7	=NEXT()					
8	=RETURN()					
9						

Formula counts from 1 to 5 by ones.

It will enter AVERAGE in five successive cells. For more information, see the description of FOR in Chapter 7, "Macro Function Directory."

**Note** | References used in macros are often in R1C1 style. For more information, see "Using References" in Chapter 4, "Writing Macros."

In a WHILE-NEXT loop, you set a condition in WHILE that tells the loop to run until the condition becomes FALSE. If the condition is FALSE the first time the macro reaches WHILE, the loop is never started.

The following example is a fragment of a command macro that calculates a series of averages from data on a worksheet, and enters just the results of those averages in cells on the worksheet. It continues looping until it reaches a cell containing a hyphen (–), because a hyphen is used on the example worksheet to mark the end of a column of data.

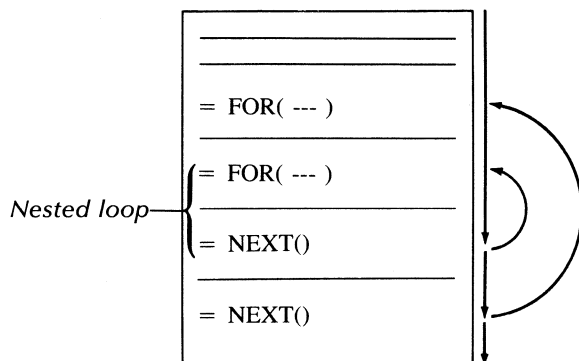
	A	B	C
20	=WHILE(GET.FORMULA(ACTIVE.CELL())<>"-")	Starts loop	
21	=SELECT("R[1]C")	Selects next cell	
22	=AVERAGE("RC[-10].RC[-1]")	Calculates	
23	=FORMULA(A22)	Enters value on worksheet	
24	=NEXT()	Ends loop	
25	=RETURN()	Returns	
26			

## Tip

If your worksheets contain many complicated formulas, and it's not important that formulas be calculated frequently, it can save time to have macros perform the calculations.

For example, the formula in cell A23 (on the worksheet above) enters the result of the formula in cell A22 in the active cell.

When you are writing loops in a macro, you can include one loop within another. This is called **nesting**.



If you want to exit either a FOR-NEXT or WHILE-NEXT loop before it's done, use BREAK. This can be convenient for error handling. If you are in a nested loop when BREAK is started, BREAK terminates the innermost loop that includes BREAK.

## Try Modifying a Macro

In the section "Try Recording a Macro," we recorded a macro that formats a selected area with the custom format "0.000." Suppose you wanted to modify that macro so that it lets you choose a format from, or enter a new format in, a dialog box. Then you could use the macro to apply any type of format.

To make the modification, just add a question mark after FORMAT.NUMBER in cell A2:

To rename the macro "Format":

- 1 Enter *Format* in cell A1.  
Leave A1 selected.
- 2 Choose Formula Define Name.  
In the dialog box, "Format" is the proposed name for the macro.
- 3 Select the Command button.
- 4 Type *f* in the Key: Ctrl+ text box.
- 5 Choose the OK button.

	A	B	C	D	E	F
1	digi3					
2	=FORMAT.NUMBER?("0.000")					
3	=RETURN()					
4						

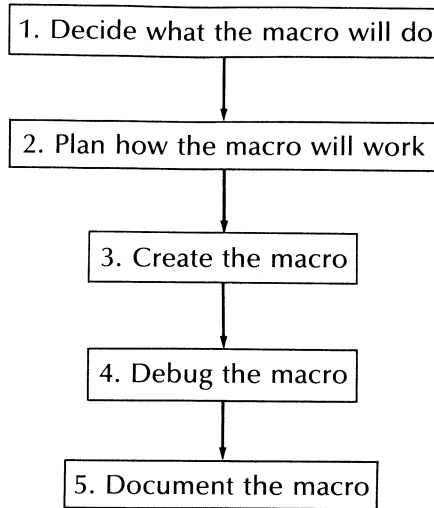
## Chapter 4

# Writing Macros

Decide What Your Macros Will Do . . . . .	157
Plan How Your Macros Will Work . . . . .	158
Differences Between Command and Function Macros . . . . .	161
Function Macros . . . . .	161
Command Macros . . . . .	161
Creating Macros . . . . .	162
Opening Macro Sheets . . . . .	163
Writing or Recording Macros . . . . .	163
Naming Macros . . . . .	164
Debugging Macros . . . . .	165
Documenting Macros . . . . .	165
Macro Structure . . . . .	165
Macro Functions . . . . .	167
Using References . . . . .	168
Error Handling . . . . .	170
Creating Command Macros . . . . .	171
Recording a Command Macro . . . . .	171
About the Recorder Range . . . . .	172
Running Out of Room in the Recorder Range . . . . .	173
Absolute and Relative Recording . . . . .	174
What Actions Are Recorded? . . . . .	175

Stopping the Recorder . . . . .	175
Restarting the Recorder . . . . .	176
Cleaning Up a Recorded Macro . . . . .	176
Writing a Command Macro Without the Recorder . . . . .	176
<b>Function Macros . . . . .</b>	<b>176</b>
Order of Functions . . . . .	177
Using Arguments . . . . .	177
Using the Second Form of ARGUMENT . . . . .	178
Returning Results . . . . .	179
Types of Arguments and Results . . . . .	180
Function Macro Example . . . . .	181
Using the Function Macro . . . . .	182

Writing a macro involves a series of steps:



### Decide What Your Macros Will Do

Taking a minute to think about what you want the macro to do can save you wasted effort later in the process. Suppose that you have several large worksheets, and you often want to print just part of them. You decide to create a macro that will print part of each worksheet. There are many ways to do this:

- First, how do you want to specify the print area?  
Do you want to select the area each time, then run the macro? Or do you want the macro to ask you to select an area? Do you want to print named areas only, and have the macro ask you to specify the name?
- Second, how do you want the area to be printed?  
With gridlines? Without? Do you want to specify a title?
- Third, how many areas should be printed?  
Do you want to print just one area, or after the first area is printed, should the macro go back to the beginning and ask you if you want another area printed?

You can create a macro to do any of these things. This is also the time to start thinking about **error handling**, the way your macro will react if something goes wrong. For example, suppose you've decided to have the macro ask you for the name of the area. What should the macro do if you type in the name of an area it can't find? You may want to have the macro ask you to try again.

## Plan How Your Macros Will Work

To plan the logic of your macro, write down the steps you want the macro to follow. It doesn't have to be elaborate — a simple list of steps that describe how the macro works is enough. Or you may want to diagram how the macro should work. Having your plan down on paper will make it much easier to write or record your macro. If you're recording, a diagram helps keep track of each step. If you're writing, you'll be able to concentrate on how to write each specific step, one at a time, rather than planning the actions of your entire macro all at once.

For the Sphere.Surface function macro we discussed in the last chapter, for example, you might jot down a list like this:

- Reads radius
- Calculates surface area by multiplying by  $4\pi r^2$
- Returns area

You can also draw a simple **flow chart**, a picture that uses standard symbols to show how your macro works.



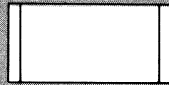
You can make most flow charts with just 5 symbols:



*Start and end*



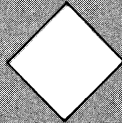
*Microsoft Excel action (for example, opening a file or adding numbers)*



*Another macro*



*Input or output*



*A decision (IF)*

Just write each step in the appropriate symbol. Connect symbols with arrows to show the direction of flow.

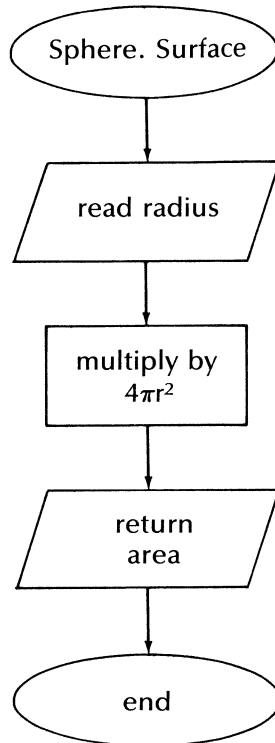
**Flow Charting Layout Rules:**

1. Don't let arrows cross.
2. Space symbols consistently.
3. Show how every step connects to the next and previous step.
4. Don't show the same step more than once.

Flow charts can help you:

- Think logically. If you draw a flow chart, you're more likely to remember to include every step you need.
- Identify parts of your macro that are too complicated. If you try to make a flow chart and find it too complicated, there's a good chance the structure of your macro will also be too complicated. In the section "Macro Structure," later in this chapter, we'll talk about a good way to organize your macros.

A flow chart for the  
Sphere.Surface macro.



While you're figuring out the logic of your macro, you'll need to decide whether you need a **command macro**, or a **function macro**, or more than one macro. If your macro needs to perform actions, rather than just make calculations, you need a command macro. Examples of actions are:

- Opening, saving, closing, or printing a document
- Opening, closing, moving, scrolling, or sizing a window
- Changing the formatting, width, or height of a cell
- Changing the active cell or the current selection
- Naming a cell or cells

Function macros, though very powerful, apply to a much narrower range of situations. If you want to take one or more values and perform some calculations on them to find another value, use a function macro.

Here's another way to think of it: if you can describe what you want the macro to do by writing a formula or series of formulas on a worksheet, you probably want a function macro. Otherwise, use a command macro.

## Differences Between Command and Function Macros

Understanding the difference between command macros and function macros can be a little confusing at first. They do look similar when you read them on a macro sheet. Let's look at a simple example of each type.

### Function Macros

A simple function macro.

	A	B	C	D	E
1	<b>Feet Per Sec</b>	<i>Reads speed in miles/hr</i>			
2	=ARGUMENT("miles.per.hr")				
3	=miles.per.hr*5280/3600	Converts			
4	=RETURN(A3)	Returns speed in ft/sec			
5					

This function macro takes a speed, measured in miles per hour, and converts it to an equivalent number of feet per second.

### Command Macros

Now let's take a look at a simple command macro and see how it's different.

This macro formats the selected area so that all numbers are displayed to three decimal places.

	A	B	C	D	E	F
1	digi3					
2	=FORMAT.NUMBER("0.000")					
3	=RETURN()					
4						

This command macro looks a lot like our function macro example. It starts with a name and ends with the RETURN function. But there are also some differences. Notice there's no ARGUMENT function. You can't give arguments to command macros. There's also no return value specified by the RETURN function. Command macros don't return values, they perform actions. (There is one exception to this rule. For more information, see Chapter 3, "Macro Basics.")

We can also compare the formulas that do the main work of each function.

- In the function macro Feet.Per.Sec, the formula

=miles.per.hr\*5280/3600

does the main work of the macro. It takes a value that was given as an argument and performs a calculation on that value.

- In the command macro digi3, the formula

=FORMAT.NUMBER("0.000")

does the main work of the macro. It uses the "0.000" format on the contents of the current cell. It performs an action, rather than making a calculation.

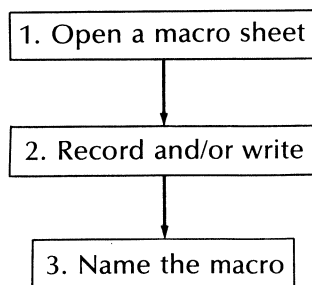
You also run a command macro differently than you run a function macro. Instead of typing the name of a command macro in a cell, you either choose the Macro Run command and choose the command macro from a list, or press a shortcut key. For information on shortcut keys, see "How Macros Work" in Chapter 3, "Macro Basics."

Don't bother memorizing all of these differences — just remember that a function macro is like a worksheet function and a command macro is like a command. If you remember that, you'll know all of the following points:

Function macros	Command macros
Make calculations	Perform actions
Take values and return values	Don't take values or return values
Are entered in cells as part of formulas	Are run like commands

## Creating Macros

There are three basic steps to creating macros:



## Opening Macro Sheets

The first step in creating a macro is opening a macro sheet. If you record a command macro using the Macro Record command, Microsoft Excel will open a new macro sheet for you. For more information, see “Recording a Command Macro” later in this chapter.

To create a new macro sheet:

- 1 Choose File New.
- 2 Select Macro Sheet.
- 3 Choose the OK button.

To open an existing macro sheet:

- 1 Choose File Open.
- 2 Select the macro sheet you want to open.
- 3 Choose the OK button.

## Writing or Recording Macros

Once your macro sheet is open, you can write a macro on it. If you choose to write your macro, you enter formulas in cells on a macro sheet to perform each step. If you are creating a command macro, you can **record** the macro instead. You just turn on the recorder, do what you want the macro to do, and turn off the recorder.

### Note

The recorder can be very useful for command macros even if you’re not going to record the entire macro. Recording part or all of a command macro and then editing it is often faster than writing the command macro from scratch. Recording saves you typing time, and you don’t need to look up the name and syntax of every macro function you want to use.

You can't record function macros, because they don't perform actions, they make calculations.

For more information, see "Recording a Command Macro" later in this chapter.

## Naming Macros

Before you use a macro, define a name for it. If you use the Macro Record command to record a command macro, Microsoft Excel defines the name for you. Otherwise you must use the Formula Define Name command to name your macro.

To name a macro:

- 1** Select the first cell of the macro on the macro sheet.
- 2** Choose Formula Define Name.
- 3** Type a name for the macro in the Name box.  
The name must be a legal Microsoft Excel name, starting with a letter and consisting only of letters, numbers, periods (.), and underlines(\_).
- 4** If it is a function macro, select Function.
- 5** If it is a command macro, first, select Command.  
Then, if you want to choose a shortcut key for running the macro, type a letter in the Key: Ctrl + text box.
- 6** Choose the OK button.

It's important to name all your macros, even though you don't have to name them to use them (to use a macro that is not named, just substitute the reference of the first cell of the macro for the name of the macro). Naming your macros, and specifying whether they're function or command macros, has several advantages:

- Function macros are listed by name in the Formula Paste Function dialog box.
- Command macros are listed by name in the Macro Run dialog box.
- Shortcut keys are assigned to named command macros.
- Named macros are easier to identify.

**Note** | Uppercase and lowercase shortcut keys are both allowed, and are recognized by Microsoft Excel as different keys. You can, for example, assign the shortcut key “a” to one macro and “A” to another.

## Debugging Macros

Once you’ve written or recorded your macro, there’s a good chance there will be a mistake or two in it. These mistakes are called **bugs**. Some bugs will be obvious the first time you try to run your new macro. The kind of bug you need to watch for is the bug that you don’t see at once; that’s why you should get in the habit of testing all of your macros. Finding and solving mistakes is called **debugging**. For information on debugging, see Chapter 5, “Debugging and Testing Macros.”

## Documenting Macros

When you’ve written your macro, take a minute to **document**, or add comments and notes to explain how the macro works, how to use it, restrictions on arguments and results, and so on. This makes it much easier for someone else to understand your macro.

## Macro Structure

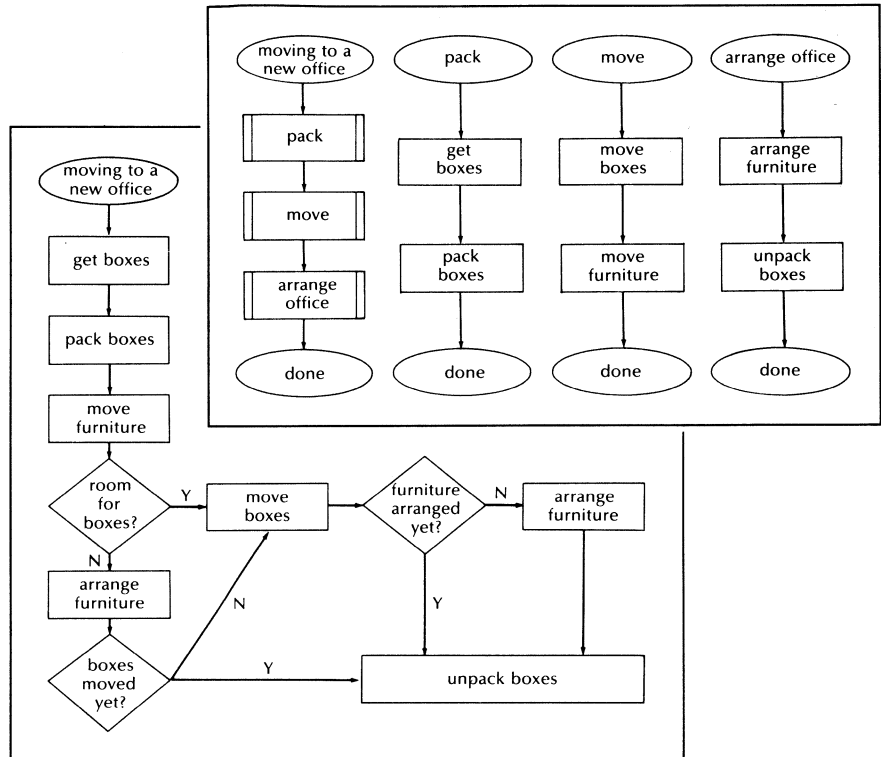
The most important fact about macro structure is that smaller macros are easier to work with than larger macros. Smaller macros are easier to write, to debug, to modify, and to reuse. You’re usually better off with an organized group of small macros than with one large macro that does too much.

If you have a complicated task that you want to perform, break the task down into parts. One good way to structure a macro is to follow these guidelines:

- Create a **control macro** that starts a series of **subroutine** macros. A control macro is a macro that tells other macros to perform tasks. Subroutine macros are run by control macros.
- Create one **entrance point** and one **exit point** in each of the subroutine macros.

The first flow chart follows the rules above. Notice that a well-organized process has such simple structure that writing a flow chart seems almost unnecessary.

However, the second flow chart for the same process breaks several rules for good structure. Imagine how difficult it would be to add one more step to this process, such as “install new carpeting.”



### Tip

Here’s a quick test to see if your macros are too complicated: how easy is it to explain what each of your subroutine macros does? A macro that “takes speed in miles per hour and returns speed in feet per second” is probably fairly simple. But a subroutine macro that creates your customer database, updates it, generates reports from it, and produces mailing labels is generally trying to do too much work. You’re better off breaking complicated macros into pieces.

Once you’ve determined the overall structure of the task, you can plan the structure of the control macro and each subroutine macro. The simplest kind of macro starts in one cell and runs straight down a column until it reaches a RETURN function.



**Lotus 1-2-3  
User's Tip**

In Lotus 1-2-3, macros stop when they reach a blank line. In Microsoft Excel, macros ignore blank lines. To stop a Microsoft Excel macro, use special functions like RETURN and HALT.

There are many functions that you can use to make different structures in your macros. For information on the GOTO and IF functions, subroutines, and looping functions, see "Changing Structure" in Chapter 3, "Macro Basics." For information on macro control functions, see "Control Functions" in Chapter 7, "Macro Function Directory." The following list summarizes some of these control functions:

If you want to	Then
Make a decision (for example, "if income is greater than \$50,000, then send letter offering gold card, otherwise don't")	Use the IF function
Repeat a step or steps (for example, "keep printing labels until out of names")	Use the FOR, NEXT, WHILE, and BREAK functions
Jump ahead or back in the macro. This is often used with the IF function (for example, "if income is greater than \$50,000, then go to the section that prints the letter, otherwise continue")	Use the GOTO function
Run a different macro from within your macro (for example, within a macro that prints mailing labels, you may want to run another macro that counts the number of names in a column)	Start the macro from within your macro. To do this, enter in a cell the macro's name followed by parentheses in your macro. For more information, see "Subroutines:" in Chapter 7, "Macro Function Directory"

## Macro Functions

You can use all of the worksheet functions in macros, as well as a special set of functions called **macro functions**. Macro functions can only be used on macro sheets.

When you choose the Formula Paste Function command from an active worksheet it lists only worksheet functions. When you choose the Formula Paste Function command from an active macro sheet, it lists both the worksheet functions and the macro functions.

There are two basic kinds of macro functions:

- Macro functions that perform an action
- Macro functions that don't perform an action

You can use any type of function in a command macro.

In a function macro, you can use only worksheet functions and macro functions that don't perform actions. Any macro functions that perform an action are ignored if they appear in a function macro. For example, the macro function OPEN opens a file. If you write the macro function OPEN in a function macro, when the function macro runs it doesn't open a file. It ignores the OPEN function.

For information on macro functions, see "Types of Macro Functions" in Chapter 7, "Macro Function Directory."

## Using References

For information on references, see References in *Microsoft Excel Reference*.

Macro function arguments that are supposed to be references are described in detail in Chapter 7, "Macro Function Directory." In general, reference arguments can be either:

- References to the active document

These references can be given either as text or as references. If they are given as text, they must be in R1C1 form. If they are given as references, they must be preceded with an exclamation point (!), and must be given in the style currently used in your workspace. By default, this is A1 style. The formula =SELECT("R[8]C[3]"), for example, selects the cell that is 8 rows down and 3 columns to the left of the active cell on the active document. If your workspace is currently set to A1 style, the formula =SELECT(!A1) selects cell A1 on the active document.

### ■ References to a macro sheet

These references must be given in the form that the macro sheet is set to: either A1 or R1C1. They can be external references, except when noted under the description for specific functions.

Relative references to the active document, unless specified otherwise in the description of a specific function, are translated into absolute references when the macro is executed. For example, the function SET.NAME("total",L1) is equivalent to the function SET.NAME("total", \$L\$1).

Some macro functions, such as FORMULA, take an explicit cell reference as an argument. Other functions, such as ALIGNMENT, operate on the current selection. If you want to operate on a specific cell on the active sheet with a function of the latter type, first use the SELECT function to select that cell, and then perform the operation.

As usual, other arguments can also be given as references to a cell or cells that contain appropriate values. These arguments can be either references to the active document or references to a macro sheet, and they must be given in the form that the document being referred to is set to: either A1 or R1C1.

The macro recorder records all references in R1C1 form, as either absolute or relative references depending on whether the Macro Absolute Record command (the default) or the Macro Relative Record command was used.

### Lotus 1-2-3 User's Tip

In Microsoft Excel, relative references are used instead of the "directional macro keys" ({down} and {up}, for example) in Lotus 1-2-3. To move one cell to the right, you would use the following formula:

```
=SELECT("RC[1]")
```

To move ten cells to the right, use the following formula:

```
=SELECT("RC[10]")
```

If you display values on a macro sheet, you may notice that references aren't displayed in the cells of macro functions that return references. That's because the reference is then translated into the contents of that cell. For example, look at the following macro sheet:

	A	B	C	D	E	F	
1	values						
2	=TEXTREF("R1C1",FALSE)						
3							

A macro sheet with formulas displayed.

The same macro sheet with values displayed:

	A	B	C	D	E	F	G	H
1	values							
2	values							
3								

The TEXTREF function returns the reference R1C1, so cell A2 displays the value contained in cell R1C1.

## Error Handling

It's important to plan your macros in advance, so that the following common problems won't cause errors while your macros are running:

- Invalid data encountered on a worksheet.
- Invalid data entered in a dialog box.
- Invalid data given to a function macro.
- Choosing Cancel in a dialog box.
- Choosing the Data Find or Formula Find command and failing to find a value.

If you want your macros to run smoothly, make sure you check for unexpected error values or FALSE return values from functions on your macro sheet.

An easy way to check for incorrect data types in function macros is to specify the expected data types in the ARGUMENT and RESULT functions. For more information, see "Using Arguments" and "Returning Results" later in this chapter.

In macros that require data to be entered, customized dialog boxes can help you check for the correct data type. For more information, see "INPUT Function" in Chapter 3, "Macro Basics," and "Custom Dialog Boxes" in Chapter 6, "Advanced Macros."

If your macro encounters an error, an easy way to handle it is to start an **error-handling subroutine macro**. This macro may be used to halt your macro or to try to recover from anticipated errors.

If you plan to write a detailed error-handling subroutine macro yourself, you can use the ERROR macro function to customize the process. For more information, see "ERROR" in Chapter 7, "Macro Function Directory."

## Creating Command Macros

If you want to create a command macro, it can be advantageous to use the macro recorder and then make modifications to get the command macro you want. Using the macro recorder is fast and easy, and never introduces typos. If you make an error while recording, you can either start over again or edit the macro to correct the error.

### Recording a Command Macro

There are two methods for recording a command macro:

- Use the Macro Record command.

This is the basic method for recording macros and is described in “Recording a Command Macro” in Chapter 3, “Macro Basics.”

- Use the Macro Start Recorder (Full menus) command.

This method is described in this section, and is useful if you want to record a macro in specific cells on an existing macro sheet.

To record a command macro using the Macro Start Recorder command:

- 1 Open a new or existing macro sheet.
- 2 Select the recorder range and choose Macro Set Recorder.  
For more information, see “About the Recorder Range” later in this section.
- 3 Select the document on which you will carry out the actions to be recorded.
- 4 If you want to start recording with relative references instead of absolute references, choose Macro Relative Record. For more information, see “Absolute and Relative Recording” later in this section.
- 5 Choose Macro Start Recorder.  
If your status bar is on, the word “Recording” appears.
- 6 Carry out the actions you want to record.
- 7 Choose Macro Stop Recorder.  
When you stop recording, Microsoft Excel records a RETURN function at the end of the new command macro.

## About the Recorder Range

The **recorder range** is the area on the macro sheet where Microsoft Excel will enter the formulas that make up the command macro.

### Note

You can select your own recorder range even if you use the Macro Record command. Before opening a new macro sheet, the Macro Record command checks to see if there is a recorder range defined. If there is a range defined and the first cell in that range is blank, it records the macro in that range. If there is a range defined but the first cell in that range is not blank, Microsoft Excel redefines the recorder range to be the first completely empty column on the macro sheet.

Microsoft Excel automatically sets a new recorder range after recording a macro with the Macro Record command. When you stop recording the macro, Microsoft Excel redefines the recorder range to be the next completely empty column on the macro sheet.

There are two ways to select the recorder range when using the Macro Set Recorder command (Full menus). Either:

- 1 Select one cell.
- 2 Choose Macro Set Recorder.

Microsoft Excel uses the entire column, from the cell you selected down, as the recorder range.

or:

- 1 Select a range of cells.
- 2 Choose Macro Set Recorder.

Microsoft Excel uses the range you selected as the recorder range.

When you start recording, Microsoft Excel checks the first cell in the recorder range. If it is either blank or contains a RETURN function, the recorder starts recording there. Otherwise the recorder searches upward from the bottom of the recorder range until it finds the first cell that contains something other than RETURN, and starts recording immediately below that cell. The recorder does not record over any cell contents other than the RETURN function.

Each action you take, such as entering a formula, making a new selection, or carrying out a command, takes up one cell in the recorder range.

If your recorder range is more than one column wide, recording proceeds down the first column of the range, then down the second column, and so on. Microsoft Excel enters a GOTO function in the last cell of each column in the range, so when the macro is running it will jump from the bottom of one column to the top of the next.

### Running Out of Room in the Recorder Range

If the recorder reaches the end of the recorder range, Microsoft Excel displays the message: “Recorder range full” and stops recording. If you want to use the macro as it exists at this point, make sure you add a RETURN function. When the range fills up, the recorder doesn’t have any place to put the RETURN function, so it is left out. The macro recorder only records in cells that are empty or contain RETURN functions. If you run the macro without adding a RETURN function, the macro will run with unpredictable results.

To extend the recorder range (if there is room on your macro sheet):

- Set a new recorder range that includes the old recorder range but extends beyond it.

To extend the recorder range (if there isn’t room on your macro sheet):

- 1 Set a new recorder range that is completely separate from the old recorder range.
- 2 Add a GOTO function at the end of the old recorder range that goes to the start of the new recorder range.

## Watching the Recorder

To see how the macro recorder works:

- 1 Open a macro sheet and your worksheet.
- 2 Put both windows next to each other and size them so you can see both.
- 3 Set the recorder range on the macro sheet.
- 4 Choose Macro Start Recorder.
- 5 Carry out whatever actions you want.  
You can see how each step is recorded on the macro sheet.

## Absolute and Relative Recording

When you record a command macro, you can choose between two settings for the recorder: **Relative Record** and **Absolute Record**. By default, Microsoft Excel uses Absolute Record.

To change between the two at any time (even in the middle of recording a macro):

- Choose Macro Relative Record, or Macro Absolute Record.

When Relative Record is selected, cells are recorded with relative references. When Absolute Record is selected, cells are recorded with absolute references.

For example, suppose you are recording a macro with Relative Record selected, and the active cell on your worksheet is A1. If you select cells A1:L1, the action is recorded using relative references. When you run the macro, if C3 is the active cell, the macro will select cells C3:N3. If you instead record the macro with Absolute Record selected, the macro uses cells A1:L1, regardless of what the active cell is.

## Recording in the Middle of a Macro

You can also use the macro recorder to record just a line or two in the middle of a macro. Suppose you wanted to run the File Page Setup command in this command macro before the macro prints out your worksheet:

	A
13	
14	
15	= PRINT( --- )
16	
17	
18	= RETURN()

To record the File Page Setup command:

- 1 Insert a blank line before line 15.
- 2 Select cell A15.
- 3 Choose Macro Set Recorder.

- 4 Choose Macro Start Recorder.
- 5 Choose File Page Setup.
- 6 Make the choices you want in the dialog box, then choose the OK button.\*  
The "Recorder range full" message appears.
- 7 Choose the OK button.

Now the macro looks like this:

	A
13	
14	
15	= PAGE.SETUP( --- )
16	= PRINT( --- )
17	
18	
19	= RETURN()



## What Actions Are Recorded?

Most of the actions you perform with Microsoft Excel can be recorded, including:

- Selecting cells and entering formulas
- Choosing commands
- Opening and closing documents

For convenience, Microsoft Excel does not record actions you cancel or actions that fail. For example, if you choose a command and then cancel it in a dialog box, Microsoft Excel does not record any function corresponding to that action. (If, however, you use the Edit Undo command, the recorder does record an UNDO function.)

Macro recorder commands are also not recorded. A macro can't turn on the macro recorder.

Some actions are recorded in a slightly modified form:

- Moving, sizing, scrolling, and changing the current selection are consolidated.  
If, for example, you scroll up and down or side to side in a window, Microsoft Excel consolidates all those actions into one function that scrolls from your start location to your end location. This makes your macro smaller and faster.
- Choosing the Edit Repeat command records another copy of the statement you repeated into the command macro. There is no command-equivalent macro function for the Edit Repeat command.
- Changing the current drive or directory in the dialog box of the OPEN command is recorded as a DIRECTORY macro function, as well as an OPEN macro function.
- Recording the Formula Paste Name and Formula Paste Function commands records a FORMULA macro function.

## Stopping the Recorder

When you choose the Macro Stop Recorder command (Full menus), Microsoft Excel automatically enters `=RETURN()` as the last formula of the macro. When the macro is run, the RETURN function ends the macro.

### Restarting the Recorder

If you choose the Macro Start Recorder command (Full menus) after recording a macro, without resetting the recorder range, Microsoft Excel writes over the RETURN function with your next action and continues recording. Thus, as long as there are empty cells in the recorder range, you can stop in the middle of recording a macro, make changes to your documents, carry out commands, or do nothing at all, and then start the recorder again without resetting it. If you are starting a new macro, however, you should start with a new recorder range.

### Cleaning Up a Recorded Macro

You may want to delete functions from your macro sheet after recording. For example, if you scrolled to go to a cell and then selected the cell, you could delete the macro functions that were recorded while you were scrolling. Or you may need to add a function or change an argument, if you omitted a step or selected the wrong option.

For information on modifying macros that you've already recorded, see "Modifying a Recorded Command Macro" in Chapter 3, "Macro Basics."

#### Tip

If you record a macro that involves opening new documents and moving between them, the names of the new documents are recorded in the macro. You may want to edit the macro so that specific document names are not used. For example, a chart named CHART1.XLC when you recorded the macro might be named CHART2.XLC the next time you run the macro. Functions such as ACTIVATE.PREV and ACTIVATE.NEXT can be useful in removing specific document names from macros.

### Writing a Command Macro Without the Recorder

If you'd rather write your command macro from scratch, just enter the formulas in order in cells on a macro sheet and add comments, labels, and so on. Don't forget to enter a RETURN function at the end of the macro.

## Function Macros

This section discusses using arguments and returning results, explains the order in which to enter functions in your function macro, and gives a detailed example of a function macro.

Remember that function macros:

- Are a lot like the functions built into Microsoft Excel (such as the worksheet function SUM or the macro function OFFSET).
- Are made up of functions written in cells on a macro sheet (just like command macros).
- Are easier to work with when they're small and modular.
- Can't be recorded, because they only make calculations and can't carry out actions.

Function macros take arguments and return results, just like built-in worksheet functions. To specify arguments to your function macro, use the ARGUMENT function, as described in "Using Arguments" later in this section. To specify the results your function macro should return, use the RETURN function and the RESULT function, described in "Returning Results" later in this section.

### Order of Functions

In a function macro, functions should appear in the following order:

Order	Function macro
First	The RESULT function. An optional function that specifies the data type of the result of your function macro.
Second	The ARGUMENT function(s). One ARGUMENT function is needed for each argument that can be given to your function macro.
Third	Formulas that carry out the calculations of your function macro.
Fourth	The RETURN function.

### Using Arguments

There must be one ARGUMENT function for each argument the function macro should accept. The first ARGUMENT function corresponds to the first argument given, the second to the second argument, and so on. Function macros can take as many as 14 arguments.

There are two forms for the ARGUMENT function:

### ■ ARGUMENT(name\_text,data\_type\_num)

This form defines the name *name\_text* for the value of the argument. For example, the formula =ARGUMENT("Width") defines the name Width for the argument corresponding to that ARGUMENT function.

### ■ ARGUMENT(name\_text,data\_type\_num,ref)

This form enters the value of the argument in the cell specified by *ref* on the macro sheet. If *name\_text* is specified, it defines the name *name\_text* for the cell *ref*. For example, the formula =ARGUMENT("Width",,C1) enters the argument corresponding to that ARGUMENT function in cell C1 on the macro sheet, and defines the name Width for cell C1 on the macro sheet.

## Note

Remember that a name can only have one definition at a time on any given document. For more information, see Names in *Microsoft Excel Reference*.

In both forms, the optional argument *data\_type\_num* specifies the data type of the argument, following the rules described in "Types of Arguments and Results" later in this chapter. If you omit the *data\_type\_num* argument, Microsoft Excel assumes that the argument is a number, text, or logical value. If the argument received by your function macro is not of the specified type, Microsoft Excel first attempts to convert it to the specified type. If it cannot be converted, the function macro returns the #VALUE! error value.

If a macro contains an ARGUMENT function, and you omit the corresponding argument when starting the function macro, the macro uses the #N/A error value as the value of the argument.

## Using the Second Form of ARGUMENT

The second form of ARGUMENT:

ARGUMENT(name\_text,data\_type\_num,ref)

can be used in any case except when the argument is a reference. When you use this form, make sure:

- The cells specified in *ref* don't contain other formulas or constants that you want to save.
- The reference is large enough to hold the argument.

If you've specified a name, that name is defined only for the part of *ref* that is actually filled by the argument. You can use the ROWS and COLUMNS functions to find out the size of the actual argument that was used.

Array Note

If the argument is an array and you don't need to change values within the array, it is a good idea to define a name for the array, rather than entering it on the macro sheet. This way, you do not need to reserve space on the macro sheet for the array. To obtain the values of individual elements in the array, use the INDEX function and the name of the array. If you do need to change values in an array, you can use the second form of ARGUMENT and use the SET.VALUE function to change individual values.

Tip

If you want a function macro to behave differently depending on what cell starts the macro, you can use the CALLER macro function. CALLER returns the reference of the cell containing the formula that starts the currently running function macro. For example, you may want the macro to behave differently depending on whether the function starts from a worksheet or from a macro sheet.

Returning Results

To specify the result that your function macro should return, give that result as the argument to the RESULT function.

	A	B	C	D	E	F	G
1	Len.Or						
2	=ARGUMENT("length")						
3	=calculate("length")						
4	=RETURN(A3)						
5							

A3 means the function macro returns the value in cell A3 of the macro sheet.

You can use the RESULT function to specify a result of a certain data type. This is very useful for error handling. If you don't use a RESULT function, Microsoft Excel assumes that your function macro gives a number, text, or logical value as its result. The form of this RESULT function is:  
RESULT(type\_number).

The argument *type\_number* specifies the data type of the return value, following the rules described in the next section, "Types of Arguments and Results." If the return value is of a different type than *type\_number* specifies, Microsoft Excel first attempts to convert it to the specified type. If the value cannot be converted, the function macro returns the #VALUE! error value.

## Types of Arguments and Results

Type	Result
1	Number
2	Text
4	Logical
8	Reference
16	Error
64	Array

For all types except Reference and Array, you can specify a value by adding numbers together. For example, the number 6 indicates that a value can be either a Text (2) or Logical value (4). The number 17 indicates that a value can be either a Number (1) or an Error value (16). The default value for type in ARGUMENT and RESULT functions is 7, which indicates that a value can be either a Number (1), Text (2), or Logical value (4).

**Tip** | If an ARGUMENT function accepts arguments of different data types, you can get the type of the actual argument with the worksheet function TYPE.

Data types of arguments to function macros are translated just like arguments to built-in worksheet functions. For example, an argument that is supposed to be a number could be given as a formula that returns a number or as a reference to a cell that contains a number.

When you give a reference as an argument, it is given as an external reference. If you give something other than a reference to a function macro that expects a reference as an argument, Microsoft Excel returns the #VALUE! error value. That is because Microsoft Excel does not translate arguments into references. If you specify a reference as an argument, make sure you use that argument as the reference of the cell and not its contents. If you use a function such as DEREf to get the contents of a reference argument, Microsoft Excel does not guarantee that the cell's value has been correctly calculated yet.

**Array Note** | If an argument to a function macro is not specified as an array or a reference, and you give an array for the argument, the function macro uses the first element in the array as the argument. Microsoft Excel handles arrays that are given to built-in functions differently. For information on how array arguments are translated in built-in worksheet functions, see "Translating Data Types," in Chapter 1, "Worksheet Function Basics."

## Function Macro Example

A simple function macro.

	A	B	C	D	E
1	<b>Feet.Per.Sec</b>	<i>Reads speed in miles/hr</i>			
2	=ARGUMENT("miles.per.hr")				
3	=miles.per.hr*5280/3600	Converts			
4	=RETURN(A3)	Returns speed in ft/sec			
5					

This function macro takes a speed, measured in miles per hour, and converts it to the equivalent number of feet per second.

Cell contents	Description
<b>Feet.Per.Sec</b>	This is a constant text value; when the macro runs, it will skip this cell.
=ARGUMENT("miles.per.hr")	Any information that you want to give to the function macro is called an <b>argument</b> to the macro. In this case, you want to give the macro a number of miles per hour. The ARGUMENT function says there will be one argument to the Feet.Per.Sec macro, and we're going to call that argument <i>miles.per.hr</i> . The name of an argument is only important within the macro itself: the name <i>miles.per.hr</i> won't have any meaning on your worksheet, for example.
=miles.per.hr*5280/3600	We multiply the <i>miles.per.hr</i> argument by 5280, because there are 5280 feet in a mile, and divide by 3600, because there are 3600 seconds in an hour (60 minutes/hour × 60 seconds/minute = 3600 seconds/hour).
=RETURN(A3)	Every macro, whether it's a command macro or a function macro, must contain a RETURN function to tell Microsoft Excel that the macro is through running, and it's time to return control to the worksheet or macro that started the macro. The "A3" in the RETURN function means the function should return the value that's in cell A3 of the macro sheet. In this case, cell A3 contains the feet per second value calculated in the formula =miles.per.hr*5280/3600.

### Using the Function Macro

The macro sheet that the function macro is written on has to be open in order to use it. Suppose, for example, the macro sheet `EXAMPLE.XLM` is open, and you're working on a worksheet. To convert 55 miles per hour to feet per second, you could type the following in a cell:

```
=EXAMPLE.XLM!feet.per.sec(55)
```

When you enter this formula in a cell formatted for two decimal places, you see the value 80.67 (55 miles per hour is equivalent to 80.67 feet per second).

You could also use the Formula Paste Function command to enter the function macro:

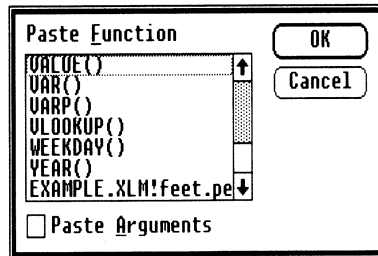
- 1 Open the macro sheet `EXAMPLE.XLM`.
- 2 Choose Formula Paste Function.

The names of function macros on all open macro sheets are listed at the bottom of the Formula Paste Function list box.

- 3 Scroll to the bottom of the list.

The list box containing the function macro you want.

Function macro



- 4 Select the `EXAMPLE.XLM!feet.per.sec` function macro.
- 5 Type 55
- 6 Enter the function macro.

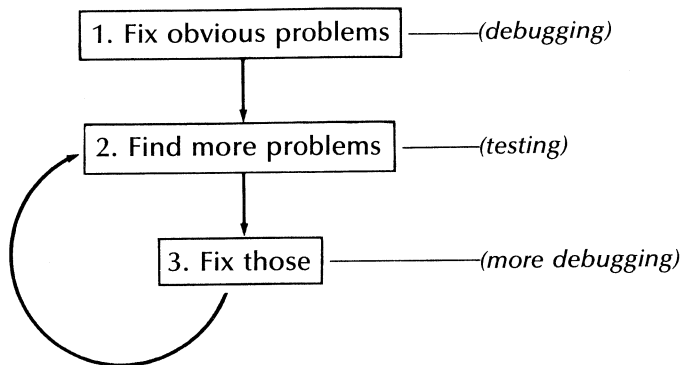


## Chapter 5

# *Debugging and Testing Macros*

Debugging Macros . . . . .	184
Stepping Through Macros . . . . .	185
Interrupting a Macro . . . . .	186
Stepping Through Sections . . . . .	186
Adding Return Functions . . . . .	187
Viewing Values . . . . .	187
Other Methods . . . . .	188
Testing Macros . . . . .	189
Use Test Data . . . . .	189
Check the Limits . . . . .	190
Anticipate Mistakes . . . . .	190

In general, making your macro work just like you want it to is a three-step process:



The reason that the process looks so simple and can be so complicated is that it is cyclical. Often, once you've fixed one bug, another one becomes obvious. Even worse, when you fix one bug you may unknowingly create another.

This can be frustrating. But when you find that your sales commission macro is incorrectly dividing all commissions by 10, remember that it's better for you to find the problem when you test your macro than for your salespeople to find the problem when they receive their paychecks.

## Debugging Macros

Once you see that something's not working correctly, start narrowing down the possible locations of the problem. The best way to do this will depend on your situation. The following sections describe a variety of methods for finding and fixing bugs. The list below indicates when the different methods are most likely to be useful:

For	See
An overview of what your macro is doing	Stepping Through Macros Viewing Values
Functions in your macro that aren't running in the correct order	Stepping Through Macros
A macro that is performing some actions incorrectly	Stepping Through Macros Adding Return Functions
A macro that is making calculations incorrectly	Viewing Values

For	See
A macro whose beginning seems to work, but has a problem later on	Interrupting
A macro whose end seems to work, but has a problem in the beginning	Stepping Through Macros Adding Return Functions

Some debugging methods involve editing your macro sheet and adding functions that help the debugging process. After you've found and corrected the bug, remember to remove these functions.

**Tip** | To get Help on any messages you may run across, press F1 while the message is on your screen.

### Stepping Through Macros

**Stepping through** a macro means that you have Microsoft Excel follow directions given in one cell of your macro sheet, then stop and display a dialog box before going on to the next cell. This gives you time to check your work, and is also an excellent way to make sure that functions are executing in the correct order.

To step through, use the STEP macro function:

- ❶ Decide where in your macro you'd like to start stepping.
- ❷ If there's no empty cell above where you'd like to start, insert a row or cell.
- ❸ Enter `=STEP()` in the cell.
- ❹ Start the macro.

Microsoft Excel stops when it reaches the `=STEP()` formula and displays a dialog box. You can move the dialog box if it's in your way. The dialog box tells you which cell in the macro Microsoft Excel is about to calculate and asks if you want to halt, continue normal execution, or keep stepping through the macro.

**Note** | The dialog box displays the reference of the cell that will be executed next, not the reference of the cell that was just executed.

For a description of the single-step dialog box, see “STEP” in Chapter 7, “Macro Function Directory.”

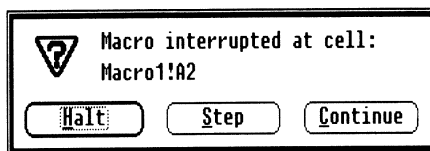
**Note** | Microsoft Excel executes all the functions in a single cell in one step. If, for example, you perform two operations in a formula, Microsoft Excel will carry out both operations before pausing.

## Interrupting a Macro

You can also stop a macro while it’s running.

- 1 Press ESCAPE.

Microsoft Excel displays the following dialog box:



- 2 Choose the Halt button to stop the macro, the Step button to continue to the next formula in the macro, or the Continue button to resume normal execution of the macro.

## Stepping Through Sections

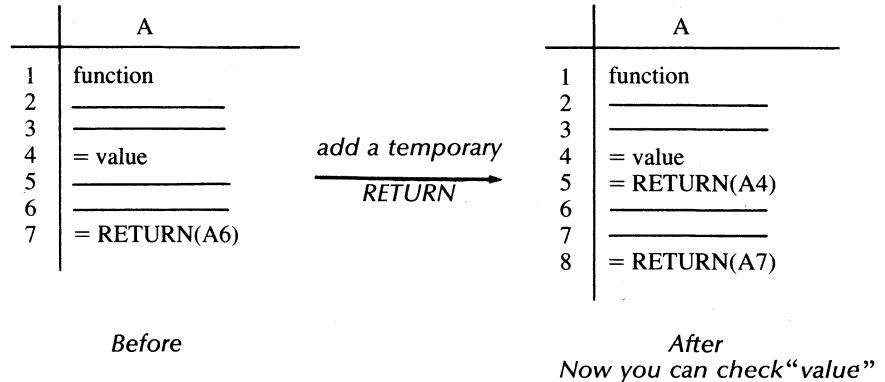
**Note** | This technique only works with command macros, because it uses an action-taking function, ALERT.

You can have your command macro ask if you want to step through certain sections. This can be convenient if you’re debugging a lengthy or complex macro. You need to add two lines in front of any area you’d like to step through, as shown below. First use the ALERT function to bring up a dialog box that will ask you if you’d like to step through. Then use the IF function to start single-stepping (if you selected the CANCEL button in the ALERT box).

	A	B	C
11	=ALERT("Choose OK to start stepping through.",1)		
12	=IF(A11,STEP())		
13			

## Adding Return Functions

In some cases, you may want to temporarily add a HALT or RETURN function before the end of your macro. This can be useful if you just want to work with the first part of your macro, or if you're debugging a function macro and want to check a particular value.



## Viewing Values

It's easy to forget that macro sheets usually display formulas instead of values, because you're usually concentrating on the formulas on your macro sheet. Looking at the values can be very useful when you're debugging a macro.

For an overview of your macro's formulas and values while your macro is running:

- 1 Open up two windows on your macro.
- 2 Set one window to display values by choosing Options Display and turning off the Formulas check box. Leave the other window displaying formulas.
- 3 Size both windows so that you can see both of them.
- 4 Run or step through your macro.

Suppose you interrupt your macro at cell A35, and see this:

Error value shows that SELECT didn't work.

FALSE shows that DATA.FIND.NEXT didn't find a record.

	A
29	TRUE
30	#REF!
31	TRUE
32	TRUE
33	TRUE
34	FALSE

This window shows values.

	A
29	= SET.VALUE( --- )
30	= SELECT( --- )
31	= SET.VALUE( --- )
32	= SELECT( --- )
33	= SET.VALUE( --- )
34	= DATA.FIND.NEXT()

This window shows formulas.

Note

Most action-taking macro functions return FALSE before they run, TRUE after they run successfully, and FALSE or an error value if they don't run successfully. For information on what a specific function returns, look up the macro function in Chapter 7, "Macro Function Directory."

Other Methods

To quickly check values on your macro sheet, you can use these two macros:

	A	B	C	D	E
1	Values(v)	Displays values			
2	=DISPLAY(FALSE)				
3	=RETURN()				
4					
5	Formulas(f)	Displays formulas			
6	=DISPLAY(TRUE)				
7	=RETURN()				
8					

Or, you can use the SET.VALUE macro function to write a particular value onto a cell on your macro sheet.

## Testing Macros

Testing a macro means trying to find all the bugs in the macro. Here are a few ways that you can test your macros.

### Use Test Data

Come up with situations where you know what the result of your macro should be. Then run the macro and see if it does what you think it should. You may want to calculate the result of a function macro by hand, for example, and then see if the function macro returns the same value you calculated. You may want to see what your macro does when given a set of test data such as a worksheet filled with zeros, or a database filled with  $-1$ 's.

### Important

Whenever you're testing a macro, make sure you're using a copy of your document and not the original. An untested macro may accidentally destroy part or all of your document. You should also save your macro sheet before testing. Be especially careful about testing macros that you use as autoexec macros. After making a change to an autoexec macro, turn off the autoexec feature by deleting the Auto\_Close or Auto\_Open name. Test the macro sheet as thoroughly as possible before using it as an autoexec macro.

## For Beginners: If You're Really Stuck

- One of the most valuable debugging techniques when you're really stuck is to take a break and do something else. This gives you time to regain your perspective on the problem.
- Ask someone else to look at your macro. This is most useful for finding blatant errors, like a misspelled function name or a GOTO function that goes to the middle of an unrelated macro. Blatant errors can be hard to see yourself — after you've been working with a macro for a while it's natural to assume that you've already found all the obvious mistakes.
- Try to explain the problem to someone. The goal of this technique isn't to have the other person discover what's wrong. The idea is that explaining the problem helps you to think it through logically and describe any assumptions you're making. This can be very effective.
- Adapt the goal of your macro. If you find that your macro doesn't work unless a database is already defined, you don't necessarily have to re-write the macro so that it will define the database for you. Just make it clear in the documentation of your macro that it requires a defined database, and, better yet, have the macro check for a defined database before it runs.
- Try breaking your macro up into short, simple macros. If your macro is really complicated, you may be better off starting over with a modular design, as described in "Macro Structure" in Chapter 4, "Writing Macros." This solution isn't usually as severe as it sounds — you may well be able to reuse some sections from your old macro.

### Check the Limits

Try to think of the limits built into your macro, and test using those values. For example, if you've written a macro that should print up to 100 labels, try running the macro to print 100 labels, to print 101 labels, to print 0 labels, and to print 1 label.

### Anticipate Mistakes

Try to anticipate possible mistakes people may make while using your macro and see what happens if they make that error. If your macro asks people to type in the number of labels to print, for example, what happens if they type in "twenty," or -10, or .5? What happens if someone chooses the Cancel button in one of your INPUT or custom dialog boxes? For information on how to handle errors, see "Error Handling" in Chapter 4, "Writing Macros."

## For Beginners: Making It Easier Next Time

When you create macros, remember that you'll eventually be debugging and testing them. These guidelines can help:

- **Keep macros small and simple.**  
Have each macro perform a single operation and write a control macro to start them. Smaller macros are easier to work with and to reuse.
- **Try to reuse macros that already work.**  
You may want to keep a master list of the macros you have, so that you don't waste time rewriting similar macros.
- **Document everything.**  
It may be obvious to you when you write a macro that you have to set up a worksheet in a certain way before running the macro, but months or even days later you can forget these assumptions. Write them down. For more information, see "Documenting Macro Sheets" in Chapter 3, "Macro Basics."
- **Take the time to write error handling into your macro.**



## Chapter 6

# Advanced Macros

Running Macros Automatically	193
Using Autoexec Macros	193
Running a Macro When Opening a Document	194
Running a Macro When Closing a Document	195
Making Demos	195
Adding Onscreen Explanations to Your Demos	195
Slowing Down Your Demos	195
Speeding Up Your Demos	196
Drawing the Viewer's Attention	196
Creating Customized Menus and Dialog Boxes	196
Custom Menus	197
Creating New Menu Bars	199
Deleting Menu Bars	199
Adding Menus or Commands	200
Deleting Menus or Commands	202
Renaming Commands	202
Adding or Removing Grey from Custom Commands	203
Adding or Removing Checkmarks from Commands	203
Switching Menu Bars	204
Finding What Menu Bar Is Displayed	204
Custom Dialog Boxes	204
Item Column	211
X and Y Columns	211

Width and Height Columns . . . . .	211
Text Column . . . . .	212
Init/Result Column . . . . .	212
Additional Information on Items . . . . .	212
Limits . . . . .	217
Custom Dialog Box Example . . . . .	218
Using Custom Help . . . . .	219
<b>Protecting Macros . . . . .</b>	<b>220</b>
Protecting and Hiding Cells and Documents . . . . .	220
Using Customized Menus and Dialog Boxes . . . . .	220
Preventing a Macro from Interruption . . . . .	220
<b>Text File Input and Output . . . . .</b>	<b>221</b>
<b>Using Macros to Start Other Applications . . . . .</b>	<b>221</b>
Communicating with Other Windows	
Applications . . . . .	222
Applications That Don't Support DDE . . . . .	223
<b>Speeding Up Your Macros . . . . .</b>	<b>223</b>

## Running Macros Automatically

The following list summarizes some methods for running macros automatically and where this information can be found in this chapter:

To	See
Run a macro whenever you open and/or close a given document	Using Autoexec Macros
Run a macro whenever you choose a particular command	Creating Customized Menus and Dialog Boxes
Run a macro at a particular time, or after a certain amount of time has elapsed	ON.TIME, in Chapter 7, “Macro Function Directory”
Run a macro whenever a specific key is pressed	ON.KEY, in Chapter 7, “Macro Function Directory”
Run a macro whenever an error occurs in a macro	ERROR, in Chapter 7, “Macro Function Directory”
Run a macro whenever a particular macro is interrupted	CANCEL.KEY, in Chapter 7, “Macro Function Directory”
Run a macro whenever data linked to another application changes in a given document	ON.DATA, in Chapter 7, “Macro Function Directory”

### Using Autoexec Macros

An **autoexec** macro runs every time you open or close a particular document. Before you define a macro as an autoexec macro, make sure it is thoroughly debugged and tested. For information on debugging and testing macros, see Chapter 5, “Debugging and Testing Macros.”

## Running a Macro When Opening a Document

- 1 Select the document.
- 2 Choose Formula Define Name.
- 3 In the Name text box, type *Auto\_open*
- 4 In the Refers to text box, enter the reference for the macro that you want to run.
- 5 Choose the OK button.

Now, when you open your document, the macro that you referred to will run. If the reference you entered is an external reference, Microsoft Excel opens the macro sheet (if it is not already open) and then runs the macro.

### Note

If a macro opens a particular document, and the document opened has an autoexec macro, the autoexec macro will not run. If you want to run the autoexec macro after opening the file, use the RUN macro function.

To open your document without running an autoexec macro:

- 1 Choose File Open.
- 2 Select your document.
- 3 Hold down SHIFT while either choosing Open, or pressing ENTER.

### Tip

Combining autoexec macros with automatically loading documents can be very convenient. If you want to load a file every time you start Microsoft Excel, you can create a **batch file** that starts Microsoft Excel with a particular document. For information on batch files, see your DOS documentation.

If you have Microsoft Windows (version 2.0, or higher), you can also add *OPEN=document\_name* to the Microsoft Excel section of your WIN.INI file. For information on your WIN.INI file, see Default settings in *Microsoft Excel Reference*.

## Running a Macro When Closing a Document

- 1 Select your document.
- 2 Choose Formula Define Name.
- 3 In the Name text box, type *Auto\_Close*
- 4 In the Refers To text box, enter the reference for the macro that you want to run.
- 5 Choose the OK button.

Now, whenever you close the document, the macro that you referred to will run, as long as its macro sheet is open.

To close the document without running the macro:

- 1 Select the document.
- 2 Select the File menu.
- 3 Hold down SHIFT while choosing Close.

## Making Demos

- 1 Turn on the macro recorder.
- 2 Carry out the actions you want to demonstrate.
- 3 Turn off the macro recorder.
- 4 Play back the macro you just recorded.
- 5 Edit the macro, if necessary.

You might want to add an explanation, change speed, or draw the viewer's attention to something in particular.

## Adding Onscreen Explanations to Your Demos

Use the ALERT and/or MESSAGE functions to add onscreen explanations to your demos. For more information, see "The ALERT and MESSAGE Functions" in Chapter 3, "Macro Basics."

## Slowing Down Your Demos

Use the WAIT function to slow down your onscreen demos. For more information, see "The WAIT Function" in Chapter 3, "Macro Basics."

## Speeding Up Your Demos

You can speed up parts of your demo if you don't need to see all of what's happening on screen. Use the ECHO macro function to turn off screen updating.

If there are intermediate steps in your demo that aren't important, you can try to remove them. For example, if your macro does a lot of calculations, you can save time by having the results saved on another worksheet, copying them for your demo instead of calculating them each time.

## Drawing the Viewer's Attention

To point out an area, try:

- Shading or adding borders to the area, using the BORDER macro function
- Displaying the marquee around the area, using the COPY macro function
- Changing fonts in the area, using the FONT macro function

You can add a message with the ALERT or MESSAGE function to point out and explain important areas. Use the WAIT function to make the macro pause.

## Creating Customized Menus and Dialog Boxes

Custom menus help you:

- Run command macros
- Set up the Microsoft Excel commands and menus in whatever way is most convenient for you
- Create special applications with Microsoft Excel

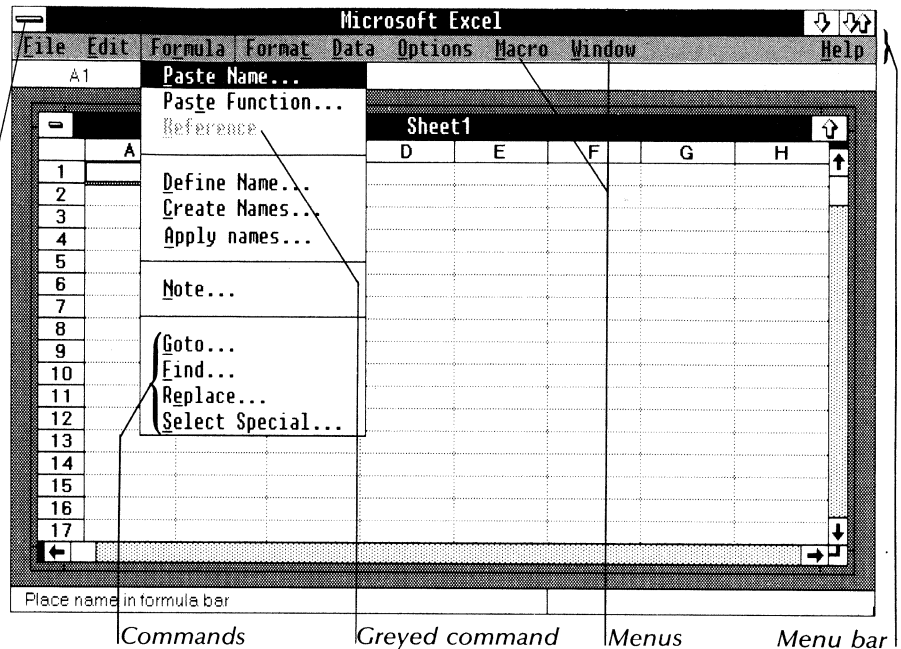
Custom dialog boxes help you:

- Enter data easily
- Check data to make sure it's the correct type
- Access data conveniently from within macros

## Custom Menus

The Microsoft Excel command structure is made up of **menu bars**, **menus**, and **commands**.

*The Control menu is the only menu that cannot be altered. It is always displayed.*



There are six built-in menu bars in Microsoft Excel. Whenever Microsoft Excel is running, one of those six menu bars is displayed (or “active”). Only one menu bar at a time is displayed. The built-in menu bars are numbered 1–6:

Full menu id = 1  
Short menu id = 5



Macro sheet and worksheet menu bars

Full menu id = 2  
Short menu id = 6

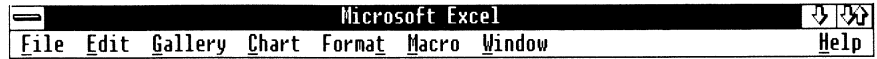


Chart menu bars

Menu id = 3



Nil menu bar

Menu id = 4



Info menu bar

Each menu bar contains one or more menus. The Info menu bar, for example, contains the File, Info, Macro, and Window menus. Whenever the Info menu bar is displayed, these four menus are displayed.

Each menu contains commands. In the Info menu bar, for example, the Macro menu contains the Record, Run, and Start Recorder commands. Commands are sometimes **greyed**, which means they are not currently available. Commands can also be **checked**, which means that a checkmark is drawn next to the command. A checkmark is used in the Window menu, for example, to show you which window is currently active.

You can use macros to:

- Create new menus, or delete menu bars you have created
- Add or delete menus from menu bars (either built-in menus or menus you have created)
- Add or delete commands from menus (either built-in commands or commands you have created)
- Change the name of commands (either built-in commands or commands you have created)
- Grey or remove grey from commands (only commands you have created)
- Check or remove checkmarks from commands (either built-in commands or commands you have created)



For every command you create, you specify a command macro. Whenever the command is chosen, its command macro runs.

### Creating New Menu Bars

Use the the macro function:

**ADD.BAR.**

The **ADD.BAR** function creates a new, empty menu bar, but does not display it.

To	Use
Display a particular menu bar	<b>SHOW.BAR</b>
Add one or more menus to a menu bar	<b>ADD.MENU</b>
Add one or more commands to a menu	<b>ADD.COMMAND</b>

The **ADD.BAR** function returns an ID number that identifies the new menu bar. When you display the new menu bar, or add menus or commands to it, you identify it by using the ID number. When you use this ID number, refer directly to the **ADD.BAR** function instead of the number. That way, your macro will work correctly regardless of how many bars have been previously added.

**Note** | You can define as many as 15 new menu bars.

### Deleting Menu Bars

Use the **DELETE.BAR** macro function to delete menu bars that you have created. If, for example, you have 15 menu bars (the limit) defined and need to add another, first delete any you are not using with **DELETE.BAR**. Add the new menu bar with **ADD.BAR**. Make sure you don't use any deleted menu bar ID numbers as arguments in other customizing functions.

## Adding Menus or Commands

When you want to add a menu to a new or built-in menu bar, use the macro function:

```
ADD.MENU(bar_id,menu_ref)
```

When you want to add a command to a new or built-in menu, use the macro function:

```
ADD.COMMAND(bar_id,menu_pos,menu_ref)
```

If you're adding a menu, it's added immediately to the right of the right-most menu on the menu bar (except the Help menu). If you're adding a command, it's added immediately below the last command on the menu.

The *Bar\_id* argument specifies the ID number of a menu bar.

The *Menu\_pos* argument specifies the menu a command should be added to. You can either give the name of the menu as text (for example, "Format"), the number of a built-in menu, or the number returned by the ADD.MENU function. For built-in menus, the left-most menu on the menu bar is numbered 1, the next menu to the right is numbered 2, and so on.

The *Menu\_ref* argument refers to an area on the macro sheet that is used to define menus and commands.

A sample *menu\_ref* area.

Menu name

Command names

	C	D	E	F	G	H
9						
10						
11	Report					
12	Aging	FINANCE.XLM!agerpt		Produces aging report	F.HLP!2	
13	Average Balance	FINANCE.XLM!avgrpt		Produces report of average balances	F.HLP!3	
14	YTD Totals	FINANCE.XLM!ytdrpt		Produces report of year-to-date totals	F.HLP!4	
15	-					
16	Include Inactive Accounts	FINANCE.XLM!inact		Checks to include inactive accounts	F.HLP!5	
17						

Command macros

Status bar command help messages

Custom help topics attached to commands

### Command Names

The first column in the *menu\_ref* argument tells Microsoft Excel what to display on the menu. For the ADD.MENU function, the first row specifies the name of the menu. If a cell in the first column contains a single hyphen (–), that position in the menu will contain a **separator line**, like the line between the Edit Repeat and Edit Cut commands.

If a character in the menu or command name is preceded by an ampersand (&), the character is underlined on your screen and can be used to choose the menu or command (with the keyboard). To include an ampersand in any menu or command name, use two ampersands in the name.

### Command Macros

The second column tells what command macros correspond to each command name. Give macro names as external references in the form of text.

### Optional Column

The third column is optional and is ignored.

### Status Bar Command Help Messages

The fourth column (optional) gives text to display in the status bar when the command is selected.

### Help Topics

The fifth column (optional) specifies a Help topic to use with the command. For more information, see “Using Custom Help” later in this chapter. If you specify a Help topic, when Help is requested on a particular menu or command, a custom Help topic is displayed.

### Example

The following macro fragment creates a new menu bar, puts one menu on the menu bar using the *menu\_ref* area shown above, and displays the menu bar:

	A	B	C	D	E	F	G
9							
10							
11	=ADD.BAR()		Report				
12	=ADD.MENU(A11,C11:D16)		Aging	FINANCE.XLM!agerpt		Produces aging report	F.HLP12
13	=SHOW.BAR(A11)		Average Balance	FINANCE.XLM!avgrpt		Produces report of average balances	F.HLP13
14	=RETURN()		YTD Totals	FINANCE.XLM!ytdrpt		Produces report of year-to-date totals	F.HLP14
15			-				
16			Include Inactive Accounts	FINANCE.XLM!inact		Checks to include inactive accounts	F.HLP15
17							

If the *menu\_ref* area shown above is still cells C11:F16, the following formula adds the four commands in *menu\_ref* to the Data menu in the built-in full worksheet or macro sheet menu bar:

```
=ADD.COMMAND(1,"Data",C12:F16)
```

## Deleting Menus or Commands

When you want to delete a built-in or custom menu or command, use the following functions:

```
DELETE.MENU(bar_id,menu_pos)
```

```
DELETE.COMMAND(bar_id,menu_pos,cmd_pos)
```

When a menu is deleted, 1 is subtracted from the position of all menus to the right of that menu. Similarly, when a command is deleted, 1 is subtracted from the position of all commands below that command.

### Example

The following function deletes the Macro menu from the Short Worksheet/Macro menu bar:

```
DELETE.MENU(5,7)
```

If the ADD.BAR function is still in cell A11, the following function deletes the command Average Balance from the Report menu created in “Adding Menus and Commands.”

```
DELETE.COMMAND(A11,"Report","Average Balance")
```

## Renaming Commands

Use the macro function:

```
RENAME.COMMAND(bar_id,menu_pos,cmd_pos,name)
```

The *name* argument is the new name for the command.

### Example

If the ADD.BAR function is still in cell A11, the following function changes the name of the command “YTD Totals” to “Cumulative Report”:

```
=RENAME.COMMAND(A11,“Report”,“YTD Totals”,“Cumulative Report”)
```

## Adding or Removing Grey from Custom Commands

Use the macro function:

```
ENABLE.COMMAND(bar_id,menu_pos,cmd_pos,state)
```

If the *state* argument is TRUE, the command is not greyed. If the *state* argument is FALSE, the command is greyed.

### Example

If the ADD.BAR function is still in cell A11, the following function greys the Include Inactive Accounts command:

```
ENABLE.COMMAND(A11,“Report”,“Include Inactive Accounts”,FALSE)
```

## Adding or Removing Checkmarks from Commands

Use the macro function:

```
CHECK.COMMAND(bar_id,menu_pos,cmd_pos,state)
```

If the *state* argument is TRUE, the command is checked. If the *state* argument is FALSE, checkmarks are removed from the command.

### Example

If the ADD.BAR function is still in cell A11, the following function checks the Include Inactive Accounts command:

```
CHECK.COMMAND(A11,“Report”,“Include Inactive Accounts”,TRUE)
```

## Switching Menu Bars

Use the macro function:

SHOW.BAR

You may want to use the ON.WINDOW macro function to display different menu bars depending on what window is active.

## Finding What Menu Bar Is Displayed

Use the macro function:

GET.BAR

The GET.BAR function returns the menu bar ID number of the currently displayed menu bar.

## Example

If the short chart menu bar is displayed, then:

GET.BAR() equals 6

## Custom Dialog Boxes

A macro sheet that creates a custom dialog box.

	A	B	C	D	E	F	G	H	I	J
1	Enter records									
2	=DIALOG BOX(B6:H22)									
3	=RETURN(I)									
4		item	x	y	width	height	text	init/result		
5									alto sax	
6	blank	12	0	0	552	177			bari sax	
7	text	5	24	14	50	18	&Name		baritone	
8	text edit	6	74	12	170	18		Sal Smith	bass clarinet	
9	text	5	24	44	90	18	&Dues Paid		bassoon	
10	number edit	8	114	42	130	18		36 41	clarinet	
11	checkbox box	13	24	72	220	18	&Available during summer	TRUE	conductor	
12	group box	14	24	96	220	72	Student Status		flute	
13	radio group	11	24	114	180	45		3	French horn	
14	radio button	12	32	114	172	15	&High school student		mallets	
15	radio button	12	32	132	164	15	&College student		oboe	
16	radio button	12	32	150	164	15	Non-&student		percussion	
17	text	5	268	108	68	12	&Instrument		tenor sax	
18	text edit	6	268	126	120	18		conductor	trombone	
19	linked list box	16	268	12	120	90	15/120	7	trumpet	
20	default ok button	1	412	12	120	21	Enter record		tuba	
21	cancel button	2	412	39	120	21	Cancel			
22	ok button	3	412	66	120	21	Done entering			
23										

When you run the Enter.Records macro, the DIALOG.BOX macro function displays this dialog box.

Name:

Dues Paid:

☒ Available during summer

Student Status

☐ High school student

☐ College student

☒ Non-student

Instrument:

alto sax  
bari sax  
baritone  
bass clarinet  
bassoon  
clarinet  
conductor

Enter record  
Cancel  
Done entering

Now you can make choices and enter values in the dialog box the same way you do in the Microsoft Excel built-in dialog boxes. Suppose you entered the following information:

Name:

Dues Paid:

☐ Available during summer

Student Status

☐ High school student

☒ College student

☐ Non-student

Instrument:

alto sax  
bari sax  
baritone  
bass clarinet  
bassoon  
clarinet  
conductor

Enter record  
Cancel  
Done entering

If you choose the OK button (which, in this case, is labeled Enter record), the choices you have made are entered on your macro sheet.

The values entered in the dialog box are now in cells H8, H10, H11, H13, H18, and H19.

	A	B	C	D	E	F	G	H	I	J
1	Enter records									
2	=DIALOG BOX(B6:H22)									
3	=RETURN()									
4		item	x	y	width	height	text	init/result		
5									alto sax	
6	blank	12	0	0	552	177			bari sax	
7	text	5	24	14	50	18	&Name:		baritone	
8	text edit	6	74	12	170	18		Chuck Anthony	bass clarinet	
9	text	5	24	44	90	18	&Dues Paid:		bassoon	
10	number edit	8	114	42	130	18		20	clarinet	
11	check box	13	24	72	220	18	&Available during summer	FALSE	conductor	
12	group box	14	24	96	220	72	Student Status		flute	
13	radio group	11	24	114	180	45		2	French horn	
14	radio button	12	32	114	172	15	&High school student		maillets	
15	radio button	12	32	132	164	15	&College student		oboe	
16	radio button	12	32	150	164	15	Non-&student		percussion	
17	text	5	268	108	88	12	&Instrument		tenor sax	
18	text edit	6	268	126	120	18		bassoon	trumpet	
19	linked list box	16	268	12	120	90	I5:I20	5	tuba	
20	default ok button	1	412	12	120	21	Enter record			
21	cancel button	2	412	39	120	21	Cancel			
22	ok button	3	412	66	120	21	Done entering			
23										

To create and use a custom dialog box:


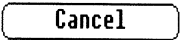
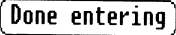

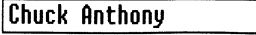
- 1 Describe the dialog box in an area on your macro sheet or worksheet.
- 2 Write a command macro to use the dialog box. Use the DIALOG.BOX(dialog\_ref) macro function to display the dialog box. The dialog\_ref argument refers to the area on the macro sheet or worksheet where the dialog box is described.
- 3 When you run the command macro, choices made in the dialog box are reflected in the appropriate locations in the dialog\_ref area on your macro sheet or worksheet.
- 4 The rest of your macro can refer to the values stored in the dialog\_ref area.

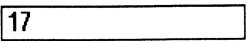
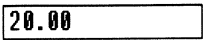

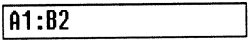
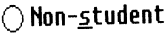
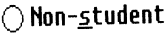
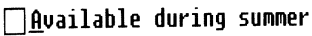
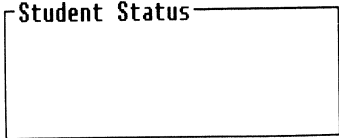
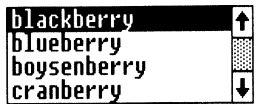
Note

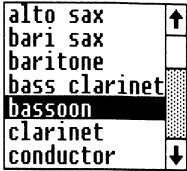

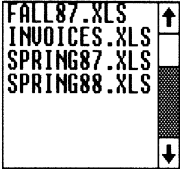

You can define custom dialog boxes and then use them as custom data forms for use with data bases. You can only use text items and text box items on custom data forms. In the init/result field you enter database field names to link items in the form to fields in your database. You set up the dialog\_ref area the same as you would for custom dialog boxes. For an example of a custom data form, see Database in *Microsoft Excel Reference*.

The following list summarizes the elements that you can use in your dialog boxes:



Example	Item	Type number
	The OK button, chosen as the default. That is, it is displayed with a thick black border and is chosen if you press ENTER. It closes the dialog box, enters the data from the dialog box into the Init/Result column of the <i>dialog_ref</i> area, and returns control to the macro.	1
	The Cancel button. It closes the dialog box and returns control to the macro.	2
	The OK button. It closes the dialog box, enters the data from the dialog box into the Init/Result column of the <i>dialog_ref</i> area, and returns control to the macro.	3
	The Cancel button, chosen as the default. That is, it is displayed with a thick black border and is chosen if you press ENTER or ESCAPE. It closes the dialog box and returns control to the macro.	4
<u>Dues Paid:</u>	Text. This is fixed text you can use to label other items in your dialog box.	5
	Text box. This is a box in which you can enter text.	6

Example	Item	Type number
	Integer box. This is like a text box except you can only enter integers from - 32768 to 32767.	7
	Number box. This is like a text box except you can only enter numbers.	8
	Formula box. This is like a text box except you can only enter a formula.	9
	Reference box. This is like a text box except you can only enter a reference.	10
	Option button group. Each group of option buttons must be preceded by an option button group item.	11
	Option button. In a set of option buttons, only one button can be selected.	12
	Check box.	13
	Group box. This is just a box that you can use to visually group items within your dialog box.	14
	List box. Contains a list of items.	15

Example	Item	Type number
	<p>Linked list box. This is like a list box, but must be preceded by a text box. When an item is chosen in the linked list box, the text of that item is entered in the text box.</p>	16
	<p>Icon. Displays a question mark (?), asterisk (*), or exclamation point (!).</p>	17
	<p>Linked file list box. Lists files in a directory. Must precede a drive and directory list box and must follow a text box. The text box is used to screen file names to appear in the linked file list box. For example, if *.XLM is entered in the text box, files with the extension .XLM in the current directory are listed in the linked file list box.</p>	18
	<p>Linked drive and directory list box. Lists available drives and directories. Must follow a linked file list box, which must follow a text box. If a type 5 item (text) immediately follows a type 19 item, the type 5 item displays the name of the current drive and directory, and is updated if either the drive or directory is changed.</p>	19

Example	Item	Type number
C:\ORDERS	Directory text. Displays the name of the current directory. Does not change after the dialog box is displayed. To display the directory name so it will change as the directory is changed, insert a type 5 item after a type 19 item.	20

The *dialog\_ref* area must be seven columns wide and at least two rows high. Let's take a closer look at the *dialog\_ref* area in the earlier example:

*Dialog\_ref* in this example is B6:H22.

	A	B	C	D	E	F	G	H	I	J
1	Enter records									
2	=DIALOG BOX(B6:H22)									
3	=RETURN()									
4		item	x	y	width	height	text	init/result		
5		12	0	0	552	177				alto sax
6	blank	5	24	14	50	18	&Name:			bari sax
7	text	6	74	12	170	18		Sal Smith		baritone
8	text edit	5	24	44	90	18	&Dues Paid:			bass clarinet
9	text	8	114	42	130	18		36 41		bassoon
10	number edit	13	24	72	220	18	&Available during summer	TRUE		clarinet
11	check box	14	24	96	220	72	Student Status			conductor
12	group box	11	24	114	180	45		3		flute
13	radio group	12	32	114	172	15	&High school student			french horn
14	radio button	12	32	132	164	15	&College student			mallets
15	radio button	12	32	150	164	15	Non-&student			oboe
16	radio button	5	268	108	88	12	&Instrument			percussion
17	text	6	268	126	120	18		conductor		tenor sax
18	text edit	16	268	12	120	90	!5!20	7		trombone
19	linked list box	1	412	12	120	21	Enter record			trumpet
20	default ok button	2	412	39	120	21	Cancel			tuba
21	cancel button	3	412	66	120	21	Done entering			
22	ok button									
23										

## Item Column

This column contains the number that describes the type of item. The possible numbers for the *type* argument are those in the previous list.

The upper-left cell in the *dialog\_ref* area must either be blank or contain a reference to a Help topic. The first line of *dialog\_ref* is used to describe the position and size of the entire dialog box. You can specify that a custom Help topic be displayed when users ask for Help on your dialog box by entering a Help reference in the top-left cell of the *dialog\_ref* area. For more information, see “Using Custom Help” later in this chapter.

## X and Y Columns

You can, optionally, enter a number in these columns to specify the horizontal and vertical position of each item in the dialog box, as well as the position of the dialog box on your screen. If you omit these values, when Microsoft Excel executes the `DIALOG.BOX` statement, it positions the items and dialog box for you.

The x and y values entered for the first row of the *dialog\_ref* area specify the position of the dialog box. If they are omitted or are zero, the box is centered. You can also specify either an x or y value, and have the box automatically centered in the other dimension.

The horizontal position of the box is measured in points from the left edge of the screen to the left edge of the dialog box.

The vertical position of the box is measured in points from the top of your screen to the top of the dialog box.

The horizontal position of an item within the dialog box is measured in horizontal screen units from the left edge of the dialog box to the left edge of the item. One horizontal screen unit is 1/8 of the width of one character in the system font. The vertical position of an item is measured in vertical screen units from the top of the dialog box to the top of the item. One vertical screen unit is 1/12 of the height of one character in the system font. If you omit an x and/or y position for an item, Microsoft Excel positions the item for you.

## Width and Height Columns

You can specify the width and/or height of the dialog box and of items within the dialog box by entering values in these columns. The width and height values entered in the first row specify the width and height of the dialog box. If you omit one or both, Microsoft Excel sizes the box for you.

Width is measured in horizontal screen units; height is measured in vertical screen units.

## Text Column

For items that display text, that text value is entered in this column.

If a text character is preceded by an ampersand (&), the character is underlined on your screen and can be used to quickly move to that item by pressing ALT and the underlined character. To quickly move to an item that does not display text, such as a list box, immediately precede the item with a type 5 item (a text item). When you press ALT and the underlined character in a text item, Microsoft Excel selects the item following the text item.

For type 15 and 16 items, this column should contain an R1C1-style or named reference, in the form of text, to a range of cells that contain entries for the list box.

This column is ignored for type 6, 7, 8, 19, and 20 items.

## Init/Result Column

In this column you can enter an initial value for an item. After the dialog box is closed, the settings and values chosen are entered in this column.

## Additional Information on Items

The following list gives more information about the different types of items:

Item	Column	Description
The dialog box	Item	Must be blank or a Help reference.
	Position	The distance from the edge of your screen, measured in points. If 0, or omitted, the dialog box is centered.
	Size	Size of the dialog box, in screen units.
	Text	Ignored.
	Init/Result	Optionally, the number of the item that should initially be selected. Items are numbered starting with 1 for the second row in the <i>dialog_ref</i> area.

Item	Column	Description
1: Default OK button	Text	Text to appear on the button, which does not have to be OK.
2: Cancel button	Text	Text to appear on the button, which does not have to be Cancel.
3: OK button	Text	Text to appear on the button, which does not have to be OK.
4: Default cancel button	Text	Text to appear on the button, which does not have to be Cancel.
5: Text	Text	Text to display.
6: Text box	Text	Ignored.
	Init/Result	Initial value of the item is the formula bar representation of the Init/Result column.
7: Integer box	Text	Ignored.
	Init/Result	Must be an integer. Initial value of the item is the formula bar representation of the Init/Result column.
8: Number box	Text	Ignored.
	Init/Result	Must be a number. Initial value of item is the formula bar representation of the Init/Result column.

<b>Item</b>	<b>Column</b>	<b>Description</b>
9: Formula box	Init/Result	<p>Must be a formula. Initial value of the item is the formula bar representation of the Init/Result column.</p> <p>References in the Init/Result column are always R1C1-style references in the form of text. When these references are displayed in the dialog box, the references are converted to the style the Microsoft Excel workspace is currently set to (by default, A1-style is set).</p>
10: Reference box	Init/Result	<p>Must be a reference. Initial value of the item is the formula bar representation of the Init/Result column.</p> <p>References in the Init/Result column are always R1C1-style references in the form of text. When these references are displayed in the dialog box, the references are converted to the style the Microsoft Excel workspace is currently set to (by default, A1-style is set).</p>

---



Item	Column	Description
11: Option button group	Init/Result	The number of the option button to select or that was selected in the following group. The button in the row below the option button group item is button number 1, the next is number 2, and so on. If the Init/Result column is blank, it is assumed to contain 1. If the Init/Result column contains #N/A, no buttons are selected.
12: Option button	Text	The name of the button.
13: Check box	Init/Result	TRUE to turn on the check box, FALSE to turn off the check box, or #N/A to grey the check box.
14: Group box	Text	Text to appear at the top of the group box. Text column does not have to contain a value.

Item	Column	Description
15: List box	Text	An R1C1-style or named reference, in the form of text, to a range of cells that contains entries for the list box. If the reference is empty or invalid, no entries are displayed in the list box.
	Init/Result	The number of the list box item to select in the list box. The first item in the list box is item number 1, and so on. If blank, the first item is selected. If #N/A, no items are selected.
16: Linked list box	Text	An R1C1-style or named external reference, in the form of text, to a range of cells that contains entries for the list box. If the reference is empty or invalid, no entries are displayed in the list box.
	Init/Result	The number of the list box item to select in the list box. The first item in the list box is item number 1, and so on. If blank, the first item is selected. If #N/A, no items are selected.

Item	Column	Description
17: Icon	Text	1 to display question mark (?), 2 to display an asterisk (*), and 3 to display an exclamation point (!). See ALERT for pictures of these icons.
18: Linked file list box	Text	Ignored.
	Init/Result	Ignored.
19: Linked drive and directory list box		You can change the current directory by entering the new directory path in the text box preceding the linked file list box.
	Text	Ignored.
	Init/Result	Ignored.
20: Directory text	Text	Ignored.

## Limits

In a single custom dialog box, you can have up to:

- 64 items
- 32 items that can take or return arguments
- 4 list boxes
- 1024 text characters

## Custom Dialog Box Example

	A	B	C	D	E	F	G	H	I	J
1										
2					516	186		1		Shirt
3	1		436	6	72	21	OK			Blouse
4	2		436	30	72	21	Cancel			Dress
5	5	8	6	120	12		Select Clothing			Skirt
6	5	8	30	48	12		Mod&el:			Socks
7	6	80	30	96	18			Golf Pro		Tie
8	5	8	54	48	12		&Type:			Jacket
9	6	80	54	96	18			Pants		Sweater
10	5	8	78	72	12		&All Types			Raincoat
11	16	8	96	112	90		DIALOG.XLMIR2C10:R10C10	1		
12	14	140	90	88	72		Sizes:			
13	11							1		
14	12	148	105	72	18		&Small			
15	12	148	123	72	18		&Medium			
16	12	148	141	72	18		&Large			
17	13	140	168	88	18		I&n Stock	TRUE		
18	14	242	1	2	180					
19	5	260	6	96	12		Select File			
20	5	260	30	40	12		&File:			
21	6	260	48	160	18			*.XL*		
22	5	260	78	72	12		F&iles in			
23	18	260	96	116	84					
24	5	392	102	88	12		&Directories			
25	19	392	120	116	60					
26	5	332	78	88	12					
27										

Dialog box produced  
by the macro above.

<b>Select Clothing</b> Model: <input type="text" value="Golf Pro"/> Type: <input type="text" value="Pants"/> All Types <input type="checkbox"/> Pants <input type="checkbox"/> Shirt <input type="checkbox"/> Blouse <input type="checkbox"/> Dress <input type="checkbox"/> Skirt <input type="checkbox"/> Socks <input type="checkbox"/> Tie		Sizes: <input checked="" type="radio"/> Small <input type="radio"/> Medium <input type="radio"/> Large <input checked="" type="checkbox"/> In Stock	<b>Select File</b> File: <input type="text" value="*.XL*"/> Files in C:\ORDERS FALL87.XLS INVOICES.XLS SPRING87.XLS SPRING88.XLS Directories <input type="text" value="[-A-]"/> <input type="text" value="[-C-]"/> <input type="text" value="[FYEAR87]"/> <input type="text" value="[FYEAR88]"/> <input type="text" value="[FYEAR89]"/>	<input type="button" value="OK"/> <input type="button" value="Cancel"/>
--	--	---	---	--

## Using Custom Help

You can write your own help information to be displayed:

- When requesting context-sensitive help for a custom command or menu
- When requesting context-sensitive help for a custom dialog box
- When HELP is executed

You enter help information in one or more Help files. Each Help file can contain one or more Help topics. The format of a Help file is:

```
*topic_number comment_text
One or more lines of text
*topic_number comment_text
One or more lines of text
:
:
```

The *topic\_number* can be any integer. *Topic\_numbers* do not have to be in consecutive order within a given Help file. *Comment\_text* is an optional comment that you can use to identify topics within the file. When the Help topic is displayed, the comment is not displayed. Only the text below the topic number and before the next topic number is displayed.

To specify that a particular Help topic should be displayed, you use a reference similar to an external reference. The form of a Help reference is:

*file\_name!topic\_number*

For example, to specify the topic numbered 58 in USER.HLP, you would use the Help reference USER.HLP!58.

To specify help topics	Enter a Help reference
For a custom command	In the last column of the <i>menu_ref</i> area, in the row specifying the command or menu
For a custom dialog box	In the top-left corner of the <i>dialog_ref</i> describing the dialog box
With the HELP macro function	As the argument to the HELP function  HELP starts Help, if Help is not already running, and displays the specified topic

## Protecting Macros

If you write macro applications, there are many ways you can prevent your macros from being incorrectly used.

### Protecting and Hiding Cells and Documents

The Format Cell Protection (Full menus), Format Column Width, Format Row Height, Options Protect Document (Full menus), and Window Hide (Full menus), and Window Unhide (Full menus) commands are all useful, and all have command equivalent macro functions. For more information, see Protecting or hiding data in *Microsoft Excel Reference*.

When you protect a worksheet with the Options Protect Document command (Full menus), you can use the UNLOCKED.NEXT and UNLOCKED.PREV macro functions to move to the next or previous unlocked cell. For more information on these functions, see Chapter 7, “Macro Function Directory,” where macro functions are listed alphabetically.

### Using Customized Menus and Dialog Boxes

Customized menus and dialog boxes generally reduce the amount of typing, which reduces the risk of typos and makes it easier for you to check entered values for the correct data type.

### Preventing a Macro from Interruption

By using the CANCEL.KEY macro function, you can prevent users from stopping a macro while the macro is running. For information on this macro function, see Chapter 7, “Macro Function Directory,” where macro functions are listed alphabetically. You can also stop all user input except responses to dialog boxes by using the DISABLE.INPUT macro function.

## Text File Input and Output

You can use macros to work with text (ASCII) files. The following list summarizes the macro functions to use. For more information, see the functions in Chapter 7, “Macro Function Directory,” where macro functions are listed alphabetically.

To	Use
Open a text file	FOPEN
Read a line	FREADLN
Read a certain number of characters	FREAD
Write a line	FWRITELN
Write a certain number of characters	FWRITE
Reposition the file	FPOS
Find the size of the file	FSIZE
Parse a file	PARSE
Close the file	FCLOSE

**Note** | Whenever you work with a text file, be sure to close the file when you’re done, using the FCLOSE macro function.

## Using Macros to Start Other Applications

If you have Microsoft Windows (version 2.0, or higher), you can:

- **Start other Microsoft Windows applications**  
To start another application under Windows, use the EXEC macro function. For information on the EXEC macro function, see “EXEC” in Chapter 7, “Macro Function Directory.”
- **Start procedures in a Microsoft Windows library**  
To start a procedure, use the CALL and REGISTER macro functions. For more information, see Chapter 7, “Macro Function Directory,” where macro functions are listed alphabetically.
- **Send information to or receive information from another Windows application**
- **Execute commands in another Windows application**

## Communicating with Other Windows Applications

You can send or receive information or execute commands in other Windows applications that support the built-in Dynamic Data Exchange (DDE) facility in Microsoft Windows.

To use DDE:

- ❶ Open a **channel**, using the INITIATE macro function.

A channel is the specific conversation between Microsoft Excel and the other application. Once a channel is open, you can:

- Send information, by using the POKE macro function
- Receive information, by using the REQUEST macro function
- Execute commands in the other application by using the EXECUTE macro function

- ❷ Close the channel by using the TERMINATE macro function.

It's very important that you close any channels you've opened. Some applications, for example, have only a limited number of channels. If you leave a channel open to an application that has only one channel, for example, you would not be able to access that application again through DDE.

If you interrupt a macro before it's closed all open channels, you should run another macro to close all channels.

Many of the arguments used by DDE macro functions depend on what application you are trying to open a channel to. For example, the first argument of the INITIATE macro function, *app\_text*, tells Microsoft Excel what application you want to access. Every application that supports DDE specifies the value of the *app\_text* argument necessary in DDE.

If you want to access a particular application from Microsoft Excel, you need to look in the documentation for that application and find out:

- If that application supports DDE
- If it does, what values are recognized for apps, topics, and items



Most applications that support DDE recognize some form of their name in the *app\_text* argument. If you were using another application to access Microsoft Excel, for example, the value of *app\_text* that Microsoft Excel responds to is “Excel”. For more information, see Exchanging data between applications in *Microsoft Excel Reference*.

You can even have more than one instance of a particular program running at the same time and communicating through DDE.

**Note** | Without using macros, you can use DDE to get information from other applications. For more information, see Linking documents in *Microsoft Excel Reference*.

## Applications That Don't Support DDE

You can communicate with other Windows applications, even if those applications do not support DDE, by:

- Exchanging data using the Clipboard  
For more information, see Exchanging data between applications in *Microsoft Excel Reference*.
- Remote controlling the application  
To remote control an application:
  - 1 Make sure the application has been started and is running under Windows.
  - 2 Use the APP.ACTIVATE macro function to activate the application.
  - 3 Use the SEND.KEYS macro function to send specific key sequences to the application.

SEND.KEYS sends a key sequence to the active application, just as if the keys had been typed at the keyboard.

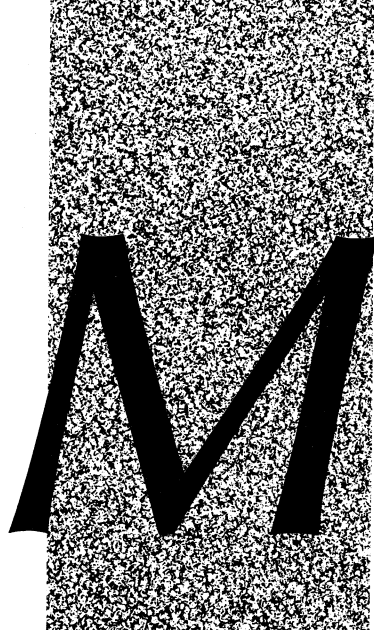
The format you use to specify a key sequence is described in the Appendix.

## Speeding Up Your Macros

If speed is of the essence, you can try the following methods for speeding up macros:

- Use the ECHO macro function to turn off screen updating.
- In some situations if you have turned off automatic calculation, you can use CALCULATE.DOCUMENT instead of CALCULATE.NOW to calculate your documents. CALCULATE.DOCUMENT only calculates the active document — other open documents are not calculated. Make sure that you don't need to calculate any other documents.





# Chapter 7

## Macro Function Directory

Types of Macro Functions . . . . .	226
Functions That Perform Actions . . . . .	227
Command Equivalent Functions . . . . .	227
Dialog Box Functions . . . . .	227
Other Action Equivalent Functions . . . . .	228
Customizing Functions . . . . .	228
Functions That Don't Perform Actions . . . . .	228
Control Functions . . . . .	229
Value-returning Functions . . . . .	229
Macro Function Arguments . . . . .	229
Macro Functions by Category . . . . .	230
Command Equivalent Functions . . . . .	230
Dialog Box Functions . . . . .	236
Other Action Equivalent Functions . . . . .	236
Customizing Functions . . . . .	238
Control Functions . . . . .	240
Value-returning Functions . . . . .	241
Directory . . . . .	243

This chapter has three sections:

- An overview of the different types of macro functions
- A listing of the macro functions organized by subject
- An alphabetical directory of all macro functions

For your convenience, the worksheet functions and their syntax are also listed. Macro functions use the same rules for syntax and argument definition as the worksheet functions. For information on worksheet functions, see Chapter 1, “Worksheet Function Basics,” and Chapter 2, “Worksheet Function Directory.”

## Types of Macro Functions

There are two basic kinds of macro functions:

- Macro functions that perform an action
- Macro functions that don't perform an action

You can use any type of function in a command macro. (The only exceptions to this rule are the two functions **ARGUMENT** and **RESULT** — they can only be used in command macros that are used only as subroutines.) In a function macro, you can only use worksheet functions and macro functions that don't perform an action.

## Functions That Perform Actions

There are four kinds of macro functions that perform actions:

Description	Name
Functions that correspond to commands	Command equivalent functions
Functions that correspond to commands bringing up dialog boxes	Dialog box functions
Functions that correspond to keyboard or mouse actions (other than choosing commands)	Other action equivalent functions
Functions that perform actions that can only be performed by macros	Customizing functions

## Command Equivalent Functions

Executing a command-equivalent function is the same as choosing a particular command from one of the Microsoft Excel menus. Executing the FILE.DELETE function, for example, is like choosing the File Delete command. You give arguments to a command equivalent function to specify options associated with the command.

## Dialog Box Functions

If there is a dialog box associated with a command, you can use a **command equivalent function**, like FILE.DELETE, or you can use a macro that brings up a dialog box that lets you make choices.

Functions that are followed by question marks, like FILE.DELETE?, are called **dialog box functions**—they bring up the dialog box. You can also give arguments to a dialog box function. They are used as default choices in the dialog boxes.

There are also some dialog box functions that do not bring up dialog boxes. The APP.MOVE? dialog box function, for example, lets you move the Microsoft Excel window using the keyboard.

## Other Action Equivalent Functions

**Other action equivalent functions** correspond to actions you take without choosing commands, such as selecting a cell (the SELECT function) or scrolling a window (the HSCROLL and VSCROLL functions).

## Customizing Functions

**Customizing functions** let you perform actions that can only be performed by macros, such as creating customized menus and dialog boxes. The following list summarizes where you can get information on customizing functions:

For	See
Displaying INPUT dialog boxes	“INPUT Function” in Chapter 3, “Macro Basics”
Displaying messages	“ALERT and MESSAGE Functions” in Chapter 3, “Macro Basics” <sup>+</sup>
Creating customized dialog boxes and menus	“Creating Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros”
Using macros to access text files	“Text File Input and Output” in Chapter 6, “Advanced Macros”
Using macros to start other applications	“Using Macros to Start Other Applications,” Chapter 6, “Advanced Macros”

## Functions That Don’t Perform Actions

There are two kinds of functions that don’t perform actions:

Description	Name
Functions that direct the flow of macros	Control functions
Functions whose primary purpose is to return specific values	Value returning functions

## Control Functions

**Control functions** tell a running macro to go someplace other than the next cell in a column. For example, GOTO lets you redirect execution anywhere; FOR and NEXT let you make loops in your macros. For information on the control functions, see “Modifying a Recorded Command Macro” in Chapter 3, “Macro Basics.” You can use control functions in both command and function macros.

## Value-returning Functions

These functions return values that you can use in macros. The ACTIVE.CELL function, for example, gives the reference of the active cell. The WINDOWS function gives the names of all the open windows. Value-returning functions are very useful in function macros or in command macros that you are modifying.

### Note

Just as with worksheet functions, Microsoft Excel usually converts a reference argument to the value contained within that reference. For example, the function IF(ACTIVE.CELL()=“R1C1”,,) does not check to see if R1C1 is the active cell; instead it checks to see if the active cell contains the text “R1C1”.

If you want to use the reference, instead of the contents of that reference, use the REFTXT macro function to convert the reference to text. You can then manipulate that reference in the form of text and use TEXTREF to convert it back to a reference.

The specific values that value-returning functions return are described in the alphabetical directory. If the description of a function doesn’t tell you what value that function returns, it is not a value-returning function.

## Macro Function Arguments

Macro function arguments follow the same rules as worksheet function arguments. For information on function arguments, see “Data Types” in Chapter 1, “Worksheet Function Basics,” and “Conventions” in Chapter 2, “Worksheet Function Directory.”

Most macro functions that describe movement or position on your screen take arguments measured in **points**. One point is equal to about 1/72 inch (72 points are equal to about 1 inch). Points are commonly used in typesetting—the height of fonts in Microsoft Excel, for example, is measured in points.

The macro recorder can be very convenient for recording specific movement or window sizing.

# Macro Functions by Category

## Command Equivalent Functions

Command	Equivalent macro function
Chart Add Arrow	ADD.ARROW
Chart Add Legend	LEGEND
Chart Add Overlay (Full menus)	ADD.OVERLAY
Chart Attach Text	ATTACH.TEXT
Chart Axes	AXES
Chart Calculate Document	CALCULATE.DOCUMENT
Chart Calculate Now	CALCULATE.NOW
Chart Delete Arrow	DELETE.ARROW
Chart Delete Legend	LEGEND
Chart Delete Overlay (Full menus)	DELETE.OVERLAY
Chart Full Menus	SHORT.MENUS
Chart Gridlines	GRIDLINES
Chart Protect Document (Full menus)	PROTECT.DOCUMENT
Chart Select Chart	SELECT
Chart Select Plot Area	SELECT
Chart Short Menus (Full menus)	SHORT.MENUS
Chart Unprotect Document (Full menus)	PROTECT.DOCUMENT
Control Close (application)	QUIT
Control Close (document)	CLOSE
Control Maximize (application)	APP.MAXIMIZE
Control Maximize (document)	FULL



Command	Equivalent macro function
Control Minimize	APP.MINIMIZE
Control Move (application)	APP.MOVE
Control Move (document)	MOVE
Control Restore (application)	APP.RESTORE
Control Restore (document)	FULL
Control Run	none
Control Size (application)	APP.SIZE
Control Size (document)	SIZE
Control Split	SPLIT
Data Delete (Full menus)	DATA.DELETE
Data Exit Find	DATA.FIND
Data Extract (Full menus)	EXTRACT
Data Find	DATA.FIND
Data Form	DATA.FORM
Data Parse (Full menus)	PARSE
Data Series	DATA.SERIES
Data Set Criteria	SET.CRITERIA
Data Set Database	SET.DATABASE
Data Sort	SORT
Data Table (Full menus)	TABLE
Edit Clear	CLEAR
Edit Copy	COPY
Edit Copy Picture	COPY.PICTURE
Edit Cut	CUT
Edit Delete	EDIT.DELETE

Command	Equivalent macro function
Edit Fill Down	FILL.DOWN
Edit Fill Left	FILL.LEFT
Edit Fill Right	FILL.RIGHT
Edit Fill Up	FILL.UP
Edit Insert	INSERT
Edit Paste	PASTE
Edit Paste Link (Full menus)	PASTE.LINK
Edit Paste Special (Full menus)	PASTE.SPECIAL
Edit Repeat (Full menus)	none
Edit Undo	UNDO
File Close	FILE.CLOSE
File Close All	none
File Delete (Full menus)	FILE.DELETE
File Exit	QUIT
File Links (Full menus)	OPEN.LINKS or CHANGE.LINKS
File New	NEW
File Open	OPEN
File Page Setup	PAGE.SETUP
File Print	PRINT
File Printer Setup	PRINTER.SETUP
File Record Macro	none
File Save	SAVE
File Save As	SAVE.AS
File Save Workspace (Full menus)	SAVE.WORKSPACE
File Unhide Window	UNHIDE

Command	Equivalent macro function
Format Alignment	ALIGNMENT
Format Border	BORDER
Format Cell Protection (Full menus)	CELL.PROTECTION
Format Column Width	COLUMN.WIDTH
Format Font	FORMAT.FONT and REPLACE.FONT
Format Justify (Full menus)	JUSTIFY
Format Legend	FORMAT.LEGEND
Format Main Chart	MAIN.CHART
Format Move	FORMAT.MOVE
Format Number	FORMAT.NUMBER
Format Overlay	OVERLAY
Format Patterns	PATTERNS
Format Row Height	ROW.HEIGHT
Format Scale	SCALE
Format Size	FORMAT.SIZE
Format Text	FORMAT.TEXT
Formula Apply Names (Full menus)	APPLY.NAMES
Formula Create Names	CREATE.NAMES
Formula Define Name	DEFINE.NAME and DELETE.NAME
Formula Find	FORMULA.FIND
Formula Goto	FORMULA.GOTO
Formula Note	NOTE
Formula Paste Function	none

Command	Equivalent macro function
Formula Paste Name	LIST.NAMES
Formula Reference	none
Formula Replace (Full menus)	FORMULA.REPLACE
Formula Select Special (Full menus)	SELECT.SPECIAL
Gallery Area	GALLERY.AREA
Gallery Bar	GALLERY.BAR
Gallery Column	GALLERY.COLUMN
Gallery Combination (Full menus)	COMBINATION
Gallery Line	GALLERY.LINE
Gallery Pie	GALLERY.PIE
Gallery Preferred (Full menus)	PREFERRED
Gallery Scatter	GALLERY.SCATTER
Gallery Set Preferred (Full menus)	SET.PREFERRED
Info Cell	DISPLAY
Info Dependents	DISPLAY
Info Format	DISPLAY
Info Formula	DISPLAY
Info Names	DISPLAY
Info Note	DISPLAY
Info Precedents	DISPLAY
Info Protection	DISPLAY
Info Value	DISPLAY
Macro Absolute Record	none
Macro Record	none

Command	Equivalent macro function
Macro Relative Record	none
Macro Run	RUN
Macro Set Recorder	none
Macro Start Recorder	none
Macro Stop Recorder	none
Options Calculate Document	CALCULATE.DOCUMENT
Options Calculate Now	CALCULATE.NOW
Options Calculation	CALCULATION or PRECISION
Options Display	DISPLAY
Options Freeze Panes (Full menus)	FREEZE.PANES
Options Full Menus	SHORT.MENUS
Options Protect Document (Full menus)	PROTECT.DOCUMENT
Options Remove Page Break (Full menus)	REMOVE.PAGE.BREAK
Options Set Page Break (Full menus)	SET.PAGE.BREAK
Options Set Print Area	SET.PRINT.AREA
Options Set Print Titles (Full menus)	SET.PRINT.TITLES
Options Short Menus (Full menus)	SHORT.MENUS
Options Unfreeze Panes (Full menus)	FREEZE.PANES
Options Unprotect Document (Full menus)	PROTECT.DOCUMENT
Options Workspace (Full menus)	WORKSPACE
Window <i>document</i>	ACTIVATE
Window Arrange All	ARRANGE.ALL

Command	Equivalent macro function
Window Hide (Full menus)	HIDE
Window New Window	NEW.WINDOW
Window Show Document (Full menus)	SHOW.INFO
Window Show Info (Full menus)	SHOW.INFO
Window Unhide (Full menus)	UNHIDE

## Dialog Box Functions

For every command that brings up a dialog box, there is a dialog box function. It has the same name as the command equivalent function, but is followed by a question mark. For example, the File Open command brings up a dialog box. To find the name of the dialog box function for the File Open command, look up File Open in the table of command equivalent functions (the command equivalent function is OPEN) and add a question mark to form the dialog box function OPEN?.

There are also some dialog box functions that do not display dialog boxes. Their actions are described in the entry for the specific function in the Macro Function Directory.

## Other Action Equivalent Functions

Function	Action
A1.R1C1	Displays A1 or R1C1 references
ACTIVATE	Selects a window
ACTIVATE.NEXT	Selects the next window
ACTIVATE.PREV	Selects the previous window
CANCEL.COPY	Cancels the marquee
COPY.CHART	Copies a picture of a chart

Function	Action
DATA.FIND.NEXT	Finds next matching record in a database
DATA.FIND.PREV	Finds previous matching record in a database
DELETE.FORMAT	Deletes a Format Number command format
DIRECTORY	Changes directories and returns a new pathname
FORMULA	Enters a formula in a cell, or text on a chart
FORMULA.ARRAY	Enters an array formula on a document
FORMULA.FILL	Fills a range with a formula
FORMULA.FIND.NEXT	Finds the next cell, as described in the Formula Find dialog box
FORMULA.FIND.PREV	Finds the previous cell, as described in the Formula Find dialog box
HLINE	Horizontally scrolls the active window by columns
HPAGE	Horizontally scrolls the active window one full window at a time
HSCROLL	Horizontally scrolls a document by percentage or by column number
SELECT	Selects an item on a chart
SELECT	Selects a reference
SELECT.END	Changes the active cell
SELECT.LAST.CELL	Selects cell at end of document
SHOW.ACTIVE.CELL	Displays the active cell
SHOW.CLIPBOARD	Displays the Clipboard
STYLE	Changes font
UNLOCKED.NEXT	Moves to the next unlocked cell

Function	Action
UNLOCKED.PREV	Moves to the previous unlocked cell
VLINE	Vertically scrolls the active window by rows
VPAGE	Vertically scrolls the active window one full window at a time
VSCROLL	Vertically scrolls document by percentage or by row number

## Customizing Functions

Function	Action
ADD.BAR	Adds a custom menu bar
ADD.COMMAND	Adds a custom command
ADD.MENU	Adds a custom menu
ALERT	Displays a dialog box
APP.ACTIVATE	Starts another application
BEEP	Makes a sound
CALL	Calls Microsoft Windows library
CANCEL.KEY	Alters ESCAPE key
CHECK.COMMAND	Marks a command
DELETE.BAR	Deletes a custom menu bar
DELETE.COMMAND	Deletes a command
DELETE.MENU	Deletes a menu
DIALOG.BOX	Displays a custom dialog box
DISABLE.INPUT	Stops all input to Microsoft Excel
ECHO	Turns screen update on and off



Function	Action
ENABLE.COMMAND	Greys and removes grey from a custom command
ERROR	Specifies an action to take if an error occurs while a macro is running
EXEC	Starts another application
EXECUTE	Carries out a command in another application
FCLOSE	Closes a text file
FOPEN	Opens a text file
FPOS	Returns position in a text file
FREAD	Reads characters from a text file
FREADLN	Reads a line from a text file
FSIZE	Returns size of a text file
FWRITE	Writes characters to a text file
FWRITELN	Writes a line to a text file
HELP	Displays a customized Help topic
INITIATE	Opens a channel to another application
INPUT	Displays a simple dialog box
MESSAGE	Displays a message in status bar
ON.DATA	Runs a macro when data is sent to Microsoft Excel by another application
ON.KEY	Runs a macro when a particular key is pressed
ON.TIME	Runs a macro at a certain time
ON.WINDOW	Runs a macro when a window is changed
POKE	Sends data to another application

Function	Action
REGISTER	Used for accessing Microsoft Windows library
RENAME.COMMAND	Renames command
REQUEST	Returns data from another application
SEND.KEYS	Sends a key sequence to an application
SET.NAME	Defines a name as a certain value
SET.VALUE	Enters values in a cell
SHOW.BAR	Displays a menu bar
STEP	Single-steps through a macro
TERMINATE	Closes a channel to another application
WAIT	Stops a macro from running

## Control Functions

Function	Action
ARGUMENT	Describes arguments to a macro
BREAK	Breaks out of a FOR-NEXT or WHILE-NEXT loop
FOR	Starts a FOR-NEXT or WHILE-NEXT loop
GOTO	Jumps to another cell
HALT	Stops a macro(s) from running
NEXT	Ends a FOR-NEXT loop
RESTART	Removes return addresses from the stack
RESULT	Specifies the data type of a function macro's return value

Function	Action
RETURN	Returns control to whatever started the macro
WHILE	Starts a WHILE-NEXT loop

Subroutine macros (listed in the Macro Function Directory under “Subroutines:”) are also control functions.

### Value-returning Functions

Function	Action
ABSREF	Returns the absolute reference of a cell
ACTIVE.CELL	Returns the reference of the active cell
CALLER	Returns the reference of the cell that started the function macro
DEREF	Returns the value of a cell in a reference
DOCUMENTS	Returns the name(s) of an open document
FILES	Returns the name(s) of a file in a specific directory
GET.BAR	Returns the number of the active menu bar
GET.CELL	Returns information about a cell
GET.CHART.ITEM	Returns the location of a chart element in a chart window
GET.DEF	Returns a name matching a definition
GET.DOCUMENT	Returns information about a document
GET.FORMULA	Returns the contents of a cell
GET.NAME	Returns the definition of a name
GET.NOTE	Returns characters from a note

Function	Action
GET.WINDOW	Returns information about a window
GET.WORKSPACE	Returns information about the workspace
LINKS	Returns the names of all linked documents
NAMES	Returns an array of defined names on a document
OFFSET	Returns one reference offset from a given reference
REFTEXT	Converts a reference into text
RELREF	Returns a relative reference
SELECTION	Returns the reference of a selection
TEXTREF	Converts text to a reference
WINDOWS	Returns the name(s) of the window on your screen

# Directory

## A1.R1C1(*r1c1*)

Equivalent to choosing the Options Workspace command (Full menus) and turning on the R1C1 check box if the *r1c1* argument is TRUE, or turning off the R1C1 check box if the *r1c1* argument is FALSE.

## ABS(*number*)

Returns the absolute value of the *number* argument. For more information, see ABS in Chapter 2, “Worksheet Function Directory.”

## ABSREF(*ref\_text*,*ref*)

Returns the reference of the cells that have the relative relationship to *ref* that is specified by *ref\_text*. *Ref\_text* must be an R1C1-style relative reference in the form of text, such as “R[1]C[1]”. *Ref\_text* is considered relative to the upper-left corner cell of *ref*. *Ref* can be an external reference.

### Examples

ABSREF(“R[–2]C[–2]”,C3) *equals* A1

ABSREF(“R[–2]C[–2]”,FINANCE.XLM!C3) *equals* FINANCE.XLM!A1

ABSREF(“R[–2]C[–2]:R[2]C[2]”,C3:G7)

*equals* ABSREF(“R[–2]C[–2]:R[2]C[2]”,C3) *equals* A1:E5

ABSREF(RELREF(A1,C3),D4) *equals* B2

## ACOS(*number*)

Returns the arccosine of *number*. For more information, see ACOS in Chapter 2, “Worksheet Function Directory.”

## A1C1(*window\_text*,*pane\_num*)

Equivalent to the action of activating a pane in a window.

*Window\_text* is the name of a window in the form of text, for example, “Sales” or “Sales:1”. *pane\_num* is the number of the pane to activate.

Pane_num	Pane
1	Top left. If the window is not split, this is the only pane. If the window is only split horizontally, this is the top pane. If the window is only split vertically, this is the left pane.
2	Top right. If the window is only split vertically, this is the right pane.
3	Bottom left. If the window is only split horizontally, this is the bottom pane.
4	Bottom right.

If a document is displayed in more than one window and *window\_text* does not specify which window to select, the first window containing the document is selected. If *window\_text* is omitted, the specified pane on the current document is selected.

If *pane\_num* is omitted, the active pane is not changed when *window\_text* is selected.

### ACTIVATE.NEXT()

### ACTIVATE.PREV()

Equivalent to pressing CONTROL + F6 or CONTROL + SHIFT + F6, respectively. They select the next and previous windows.

### ACTIVE.CELL()

Returns the reference of the active cell as an external reference.

If you use the value returned by ACTIVE.CELL in a function or operation, you will usually get the value contained in the active cell instead of its reference. That's because the reference is automatically translated into the contents of the reference. If you want to work with the actual reference, use the REFTEXT function to convert the active cell reference to text, which you can then store or manipulate (or convert back to a reference with TEXTREF).

## Examples

If the document in the active window is named “SALES.XLS,” and if cell A1 is the active cell, then:

ACTIVE.CELL() *equals* SALES.XLS!A1

SET.NAME(“act”,ACTIVE.CELL()) assigns the name act to the active cell.

## ADD.ARROW()

Equivalent to the Chart Add Arrow command.

## ADD.BAR()

Creates an empty menu bar in the menu system. If ADD.BAR is successful, it returns the bar ID number. If there are too many menu bars defined, ADD.BAR returns the #VALUE! error value. Only 15 custom menu bars can be defined at one time.

ADD.BAR does not display the new bar. Use SHOW.BAR to display a bar.

For information on custom menus, see “Making Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

## ADD.COMMAND(bar\_num,menu\_pos,menu\_ref)

Adds one or more commands described in menu area *menu\_ref* to the menu at *menu\_pos* in bar number *bar\_num*. *Menu\_ref* is a reference to a macro sheet area that describes the new command. *Menu\_pos* can either be the number of a menu or the name of a menu as text.

*Bar\_num* can either be the number of one of the Microsoft Excel built-in menu bars or the number returned by a previously executed ADD.BAR function. The numbers of the built-in menu bars are:

Built-in menu bar	Number
Worksheet and Macro menu, Full menus	1
Chart menu, Full menus	2
Nil menu (that is, the menu displayed when no documents are open)	3
Info window menu	4
Worksheet and Macro menu, Short menus	5
Chart menu, Short menus	6

The `ADD.COMMAND` function returns the command position of the first command added.

For information on custom menus, see “Making Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

### ADD.MENU(*bar\_num*,*menu\_ref*)

Adds a menu described in menu area *menu\_ref* to the bar with the bar ID number *bar\_num*. If the `ADD.MENU` function is successful, the menu is added immediately to the right of the existing menus on bar *bar\_num*, and `ADD.MENU` returns the position number for the new menu.

*Bar\_num* can either be the number of one of the Microsoft Excel built-in menu bars or the number returned by a previously executed `ADD.BAR` function.

For information on custom menus, see “Making Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

### ADD.OVERLAY()




Equivalent to the Chart Add Overlay command (Full menus). Adds an overlay to a chart.

If the active chart already has an overlay, `ADD.OVERLAY` takes no action and returns `TRUE`.



## ALERT(message\_text,type\_num)

Displays a dialog box and waits for you to choose a button. The dialog box contains the message *message\_text*, a symbol, and one or two buttons, determined by *type\_num* as follows:

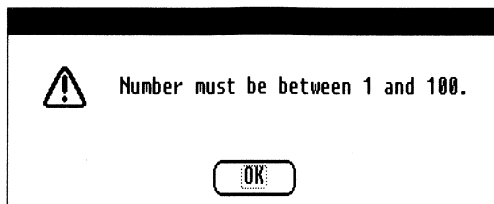
Type_num	Use when	Dialog box displayed
1	Need to make a choice	
2	Need to present information	
3	Error occurred but no choice is available	

ALERT returns the logical value TRUE if you choose the OK button, and returns FALSE if you choose the Cancel button.

### Examples

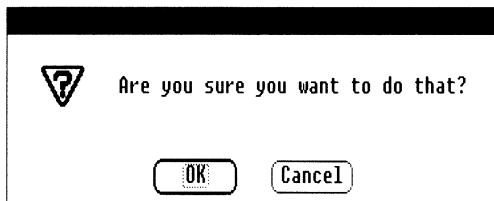
The following formula displays the dialog box shown below:

ALERT("Number must be between 1 and 100.",3)



The following formula displays the dialog box shown below:

`ALERT("Are you sure you want to do that?",1)`



### ALIGNMENT(*type\_number*)

### ALIGNMENT?(*type\_number*)

Equivalent to the Format Alignment command.

*Type\_number* specifies the type of alignment:

Type_number	Alignment
1	General
2	Left
3	Center
4	Right
5	Fill

### AND(*logical1,logical2,...*)

Returns true if every argument is TRUE; otherwise, returns FALSE. For more information, see AND in Chapter 2, "Worksheet Function Directory."

## APP.ACTIVATE(*title\_text*,*wait\_log*)

Activates the application with *title\_text* title bar. If *title\_text* is omitted, APP.ACTIVATE activates Microsoft Excel.

If *wait\_log* is TRUE or omitted, Microsoft Excel waits to be activated before activating the specified application *title\_text*. Microsoft Excel flashes to indicate that it is waiting. If *wait\_log* is FALSE, Microsoft Excel activates the application immediately.

### Example

This formula starts the “Calculator” application:

```
=APP.ACTIVATE(“Calculator”)
```

## APP.MAXIMIZE()

Equivalent to the Control Maximize command for the application window. Maximizes the Microsoft Excel application window.

## APP.MINIMIZE()

Equivalent to the Minimize command on the Control menu for the application window. Minimizes the Microsoft Excel application window.

## APP.MOVE(*x\_num*,*y\_num*)

## APP.MOVE?(*x\_num*,*y\_num*)

Equivalent to the Move command for the application Control menu. *X\_num* specifies the horizontal position of the Microsoft Excel window, measured in points from the left edge of your screen to the left side of the Microsoft Excel window.

*Y\_num* specifies the vertical position of the Microsoft Excel window, measured in points from the top edge of your screen to the top of the Microsoft Excel window.

APP.MOVE?, the dialog box form of APP.MOVE, doesn’t actually display a dialog box. Instead, it is equivalent to pressing ALT,SPACEBAR,M or to clicking on the title bar with the mouse. This allows you to move the window with the keyboard or mouse. For information on moving a window, see Windows in *Microsoft Excel Reference*.

## APP.RESTORE()

Equivalent to the Restore command on the Control menu for the application window.

## APP.SIZE(*x\_num*,*y\_num*)

## APP.SIZE?(*x\_num*,*y\_num*)

Equivalent to the Size command for the application Control menu. *X\_num* specifies the width of the Microsoft Excel window in points. *Y\_num* specifies the height in points.

APP.SIZE?, the dialog box form of APP.SIZE, doesn't actually display a dialog box. Instead, it is equivalent to pressing ALT,SPACEBAR,S or to dragging a window border with the mouse. This allows you to size the window with the keyboard or mouse. For information on sizing a window, see Windows in *Microsoft Excel Reference*.

## APPLY.NAMES(*name\_array*,*ignore*,*use\_rowcol*,*omit\_col*,*omit\_row*, *name\_order*,*append*)

## APPLY.NAMES?(*name\_array*,*ignore*,*use\_rowcol*,*omit\_col*,*omit\_row*, *name\_order*,*append*)

Equivalent to choosing the Formula Apply Names command (Full menus). Searches for the definition of the name or names in *name\_array* in formulas and replaces definitions with names.

The following list summarizes the arguments.

Argument	Description
<i>name_array</i>	The name or names to apply, as text elements in an array
<i>ignore</i>	A logical value corresponding to the Ignore Relative/Absolute check box. TRUE to turn on the check box, FALSE to turn off the check box.
<i>use_rowcol</i>	A logical value corresponding to the Use Row and Column Names check box. TRUE to turn on the check box, FALSE to turn off the check box. If <i>use_rowcol</i> is not TRUE, the next two arguments are ignored.
<i>omit_col</i>	A logical value corresponding to the Omit Column Name if Same Column check box. TRUE to turn on the check box, FALSE to turn off the check box.
<i>omit_row</i>	A logical value corresponding to the Omit Row Name if Same Row check box. TRUE to turn on the check box, FALSE to turn off the check box.
<i>name_order</i>	1 = Row Column 2 = Column Row
<i>append</i>	A logical value. If <i>append</i> is TRUE, APPLY.NAME replaces the definitions of the names in <i>name_array</i> and also replaces the definition or definitions of the name or names most recently defined by the Formula Define Name command or the Formula Create Names command (Full menus). If <i>append</i> is FALSE, APPLY.NAME replaces the definitions of the names in <i>name_array</i> only.

## AREAS(reference)

Returns the number of areas in *reference*. For more information, see AREAS in Chapter 2, "Worksheet Function Directory."

## ARGUMENT(name\_text,data\_type\_num)

## ARGUMENT(*name\_text,data\_type\_num,ref*)

Describes arguments to be given to a function macro. A function macro must contain one ARGUMENT function for each argument to be given. Function macros can accept as many as 14 arguments. If a macro contains an ARGUMENT function and you omit the corresponding argument in the function starting the macro, the macro uses the error value #N/A as the value of the argument.

The following table lists the arguments to the ARGUMENT function:

Argument	Description
<i>name_text</i>	The name of the argument, or of the cells containing the argument if <i>ref</i> is specified
<i>data_type_num</i>	The type of the argument
<i>ref</i>	The macro sheet cell(s) in which to store the argument

For information on the ARGUMENT function, see Chapter 4, “Writing Macros.”

## ARRANGE.ALL()

Equivalent to the Window Arrange All command.

## ASIN(number)

Returns the arcsine of *number*. For more information, see ASIN in Chapter 2, “Worksheet Function Directory.”

## ATAN(number)

Returns the arctangent of *number*. For more information, see ATAN in Chapter 2, “Worksheet Function Directory.”

## ATAN2(x\_number,y\_number)

Returns the arctangent of point (*x\_number,y\_number*). For more information, see ATAN2 in Chapter 2, “Worksheet Function Directory.”

## ATTACH.TEXT(attach\_to\_num,series\_num,point\_num)

## ATTACH.TEXT?(*attach\_to\_num,series\_num,point\_num*)

Equivalent to the Chart Attach Text command. *Attach\_to\_num* specifies what to attach text to, as follows:

Attach_to_num	Item
1	Chart title
2	Value axis
3	Category axis
4	Series or data point

If *attach\_to\_num* is 4, use *series\_num* to specify the number of the series and, optionally, *point\_num* to specify the number of the point.

## AVERAGE(*number1,number2,...*)

Returns the average of numbers. For more information, see AVERAGE in Chapter 2, “Worksheet Function Directory.”

## AXES(*main\_cat,main\_value,over\_cat,over\_value*)

## AXES?(*main\_cat,main\_value,over\_cat,over\_value*)

Equivalent to the Chart Axes command. The arguments correspond to the four check boxes in the Chart Axes dialog box, as shown in the list below. (If a chart has no overlay, there are only two check boxes, corresponding to the first two arguments.) If an argument is TRUE, its check box is turned on; if an argument is FALSE, its check box is turned off.

Argument	Item
<i>main_cat</i>	Main Chart: Show Category Axis
<i>main_value</i>	Main Chart: Show Value Axis
<i>over_cat</i>	Overlay Chart: Show Category Axis
<i>over_value</i>	Overlay Chart: Show Value Axis

## BEEP(*number*)

Sounds the tone specified by *number*, which must be between 1 and 4. If *number* is omitted, it is assumed to be 1. Typical uses of BEEP are to signal the end of a macro and a new dialog box.

The exact tone made for each number depends on your hardware configuration and system software. For some machines, such as the IBM PC, all four tones are the same.

## BORDER(*outline, left, right, top, bottom, shade*)

## BORDER?(*outline, left, right, top, bottom, shade*)

Equivalent to the Format Border command.

Each argument is a logical value corresponding to the check box in the Format Border dialog box. If the argument is TRUE, the check box is turned on; if the argument is FALSE, the check box is turned off.

## BREAK()

Interrupts a FOR-NEXT or a WHILE-NEXT loop. If BREAK is encountered within a loop, that loop is terminated and execution proceeds to the statement following the NEXT statement at the end of the current loop. For information on looping, see “FOR,” “WHILE,” or “Looping” in Chapter 3, “Macro Basics.”

## CALCULATE.DOCUMENT()

Equivalent to the Options Calculate Document and Chart Calculate Document commands. Calculates the active document.

## CALCULATE.NOW()

Equivalent to the Options Calculate Now and Chart Calculate Now commands. Calculates all open documents.



CALCULATION(*type\_num,iter,max\_num,max\_change,update,precision,date\_1904*)

CALCULATION?(*type\_num,iter,max\_num,max\_change,update,precision,date\_1904*)

Equivalent to the Options Calculation command.

The following list summarizes the arguments. If logical values are TRUE, their corresponding check box is turned on; if FALSE, the check box is turned off.

Argument	Description
<i>type_num</i>	Indicates the type of calculation, as follows: 1 = Automatic 2 = Automatic Except Tables 3 = Manual
<i>iter</i>	A logical value corresponding to the Iteration check box
<i>max_num</i>	The maximum number of iterations
<i>max_change</i>	The maximum change
<i>update</i>	A logical value corresponding to the Update Remote References check box
<i>precision</i>	A logical value corresponding to the Precision as Displayed check box
<i>date_1904</i>	A logical value corresponding to the 1904 Date System check box

CALL(*call\_text,argument1,...*)

Important

This is a powerful but dangerous function provided for expert programmers only. If you use the CALL function incorrectly you could accidentally cause errors in your system's operation.

Calls a procedure in a Microsoft Windows dynamic library.

*Call\_text* is the value returned by a previously executed REGISTER function. You give REGISTER the name of the module and procedure you want to access, as well as a text value that describes the number and data types of arguments you want to give and the data type of the return value. REGISTER then returns the value of *call\_text* to use in the CALL function.

The *arguments* are the arguments to be given to the procedure.

See the REGISTER function for information on the data types and how they are given and returned.

### Examples

The example for the REGISTER function shows how to use REGISTER to get the *call\_text* value for calling the GetSysColor procedure. If the result of that REGISTER function was in cell A5, you could use the following CALL function to call the GetSysColor procedure, giving the argument 2:

```
=CALL(A5,2)
```

You can also combine the two functions in one formula, as follows:

```
=CALL(REGISTER("USER","GetSysColor","JH"),2)
```

If, however, you plan to call the GetSysColor procedure more than once, it's more efficient to use the REGISTER function only once and have all your CALL functions refer to that result.

## CALLER()

Returns the reference of the cell containing the function that started the currently running function macro. If the function was part of an array formula entered into a range of cells, CALLER returns the reference of the range. If the currently running macro is a command macro started by the user, CALLER returns the #REF! error value. CALLER is useful in a macro whose behavior depends on the location or size of the reference that started the macro.

### Examples

In a function macro that is started from a function in an array formula, the following functions return the number of rows and columns, respectively, in the reference that started the macro:

```
ROWS(CALLER())
```

```
COLUMNS(CALLER())
```

## CANCEL.COPY()

Equivalent to the action of canceling the marquee by pressing ESCAPE after you copy or cut a selection.

## CANCEL.KEY(enable,macro\_ref)

Disables macro interruption, or specifies a macro to run when a macro is interrupted. If *enable* is FALSE or omitted, pressing ESCAPE will not interrupt a macro.

If *enable* is TRUE and *macro\_ref* is omitted, ESCAPE is reactivated. If *enable* is TRUE and *macro\_ref* is specified, macro execution automatically goes to *macro\_ref* if you press ESCAPE.

Note	The effects of CANCEL.KEY only last until the end of the currently running macro. Once the macro stops, ESCAPE is reactivated as normal.
------	--

## CELL(type\_of\_info,reference)

Returns information about formatting, location, or contents of *reference*. For more information, see CELL in Chapter 2, “Worksheet Function Directory.”

## CELL.PROTECTION(locked,hidden)

## CELL.PROTECTION?(locked,hidden)

Equivalent to the Format Cell Protection command (Full menus).

*Locked* corresponds to the Locked check box; *hidden* corresponds to the Hidden check box. If an argument is TRUE, the corresponding check box is turned on; if an argument is FALSE, its check box is turned off. If an argument is omitted, the setting is not changed.

## CHANGE.LINK(old\_link,new\_link)

## CHANGE.LINK?(old\_link,new\_link)

Equivalent to choosing the File Links command (Full menus), selecting *old\_link*, selecting the Change button, and entering *new\_link*. *Old\_link* and *new\_link* must be the names of documents in the form of text. If you use CHANGE.LINK? and specify *old\_link* (omitting *new\_link*), only the second dialog box is displayed.

## CHAR(*number*)

Returns the ASCII character corresponding to *number*. For more information, see CHAR in Chapter 2, “Worksheet Function Directory.”

## CHECK.COMMAND(*bar\_num*,*menu\_pos*,*command\_pos*,*check*)

Adds or removes a check mark from beside the command in position *command\_pos* on the menu *menu\_pos* in bar number *bar\_num*. *Menu\_pos* can either be the number of a menu (the first menu on a bar is menu 1) or the name of a menu, as text. *Command\_pos* can be either the number of the command to be checked (the first command on a menu is command 1) or the title of the command, as text.

*Bar\_num* can either be the number of one of the Microsoft Excel built-in menu bars or the number returned by a previously executed ADD.BAR function.

If *check* is TRUE, checks command, if FALSE, removes check. The checkmark doesn't affect execution of the command. When Microsoft Excel checks and unchecks built-in commands, it may add or remove checkmarks you have set on built-in commands.

For information on custom menus, see “Making Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

## CHOOSE(*index\_number*,*value1*,*value2*,...)

Uses *index\_number* to select a value from *values*. *Values* can also be references, in which case CHOOSE returns a reference. For more information, see CHOOSE in Chapter 2, “Worksheet Function Directory.”

## CLEAN(*text*)

Removes control characters from *text*. For more information, see CLEAN in Chapter 2, “Worksheet Function Directory.”

## CLEAR(*number*)

## CLEAR?(*number*)

Equivalent to the Edit Clear command.

Number	Clears
1	All
2	Formats
3 or omitted	Formulas
4	Notes

## CLOSE(*save\_logical*)

Equivalent to the File Close command on the Control menu. Closes the active window. *Save\_logical* tells Microsoft Excel what to do if unsaved changes have been made to the document in the active window, and this is the last window displaying the document.

Save_logical	Action
TRUE	Document is saved
FALSE	Document is not saved
Omitted	A message is displayed asking if you want to save your document

## CLOSE.ALL()

Equivalent to the File Close All command. Closes all unprotected windows.

## CODE(text)

Returns the ASCII code of the first character in *text*. For more information, see CODE in Chapter 2, "Worksheet Function Directory."

## COLUMN(*reference*)

Returns the column numbers in *reference*. For more information, see COLUMN in Chapter 2, "Worksheet Function Directory."

## COLUMN.WIDTH(*width\_num*,*ref*)

## COLUMN.WIDTH?(*width\_num*,*ref*)

Equivalent to the Format Column Width command. The columns contained in *ref* are changed to the width *width\_num*. The units of *width\_num* are the width of one character in the first font in the Format Font dialog box for the current document.

If *ref* is omitted, it is assumed to be the reference of the current selection. If *ref* is specified, it must be either an external reference to the active worksheet, such as !\$A:\$C or !Database, or an R1C1-style reference in the form of text, such as “C1:C3”, “C[−4]:C[−2]”, or “Database”. If *ref* is a relative R1C1-style reference in the form of text, it is assumed to be relative to the active cell.

If you are recording a macro while using a mouse, and you change column widths by dragging the column border, Microsoft Excel records the reference of the columns using R1C1-style references in the form of text. If Relative Record is on, Microsoft Excel uses R1C1-style relative references. If Absolute Record is on, Microsoft Excel uses R1C1-style absolute references.

### Examples

If the selection is the range C1:E50, this function changes the width of columns C:E to 15:

```
COLUMN.WIDTH(15)
```

If the active cell is C1, this function hides columns D:F by making their widths zero:

```
COLUMN.WIDTH(0,“C[1]:C[3]”)
```

Each of the following functions changes the width of columns C and D to 15:

```
COLUMN.WIDTH(15,!$C:$D)
```

```
COLUMN.WIDTH(15,“C3:C4”)
```

## COLUMNS(*array*)

Returns the number of columns in *array*. For more information, see COLUMNS in Chapter 2, “Worksheet Function Directory.”

COMBINATION(number)

COMBINATION?(number)

Equivalent to the Gallery Combination command (Full menus).  
*Number* must correspond to a format in the gallery.

COPY()

Equivalent to the Edit Copy command.

COPY.CHART(number)

COPY.CHART?(number)

Note

This macro function is included for compatibility with macros written with Microsoft Excel for the Apple Macintosh. In Microsoft Excel for Windows, it is the same as the COPY.PICTURE macro function with the *appearance* argument omitted.

Equivalent to the Microsoft Excel for the Apple Macintosh Edit Copy Chart command.

Number	Description
1	As shown on screen
2	As shown when printed

COPY.PICTURE(appearance,size)

Equivalent to the Edit Copy Picture command. This command appears on the Edit menu if you press and hold down SHIFT while choosing the menu. It copies the selection to the Clipboard as a picture.

*Appearance* is a number describing how to copy the picture:

Appearance	Description
1	As shown on screen
2	As shown when printed

Your screen often has very different properties than your printer does. To copy a picture as close as possible to the picture displayed on your screen, use *appearance* equal to 1. To copy the printed version of the picture, use *appearance* equal to 2.

*Size* only applies if the current selection is a chart, and is a number describing how to copy the picture:

Size	Description
1	As shown on screen
2	As shown when printed

### COS(radians)

Returns the cosine of *radians*. For more information, see COS in Chapter 2, “Worksheet Function Directory.”

### COUNT(value1,value2,...)

Returns the count of numbers in *values*. For more information, see COUNT in Chapter 2, “Worksheet Function Directory.”

### COUNTA(value1,value2,...)

Returns the count of values in *values*. For more information, see COUNTA in Chapter 2, “Worksheet Function Directory.”



CREATE.NAMES(*top,left,bottom,right*)

CREATE.NAMES?(*top,left,bottom,right*)

Equivalent to the Formula Create Names command (Full menus).

Each argument corresponds to a check box. If the argument is TRUE, the corresponding check box is turned on; if the argument is FALSE or omitted, the corresponding check box is turned off.

Argument	Check Box
<i>top</i>	Top Row
<i>left</i>	Left Column
<i>bottom</i>	Bottom Row
<i>right</i>	Right Column

CUT()

Equivalent to the Edit Cut command.

DATA.DELETE()

DATA.DELETE?()

Equivalent to the Data Delete command (Full menus).

If you execute the dialog box form, DATA.DELETE?, Microsoft Excel displays a message warning you that records will be permanently deleted, and you can approve or cancel. If you execute the plain form, DATA.DELETE, the records are deleted without any message being displayed.

DATA.FIND(logical)

Equivalent to the Data Find and Data Exit Find commands.

If *logical* is TRUE, Microsoft Excel carries out the Data Find command. If *logical* is FALSE, Microsoft Excel carries out the Data Exit Find command.

## DATA.FIND.NEXT()

## DATA.FIND.PREV()

Equivalent to pressing the DOWN or UP direction key after the Data Find command has been chosen. Finds the next or previous matching record in a database. If the function cannot find a matching record, it returns the logical value FALSE.

## DATA.FORM()

Equivalent to the Data Form command.

## DATA.SERIES(row\_col,type,date,step,stop)

## DATA.SERIES?(row\_col,type,date,step,stop)

Equivalent to the Data Series command (Full menus).

*Row\_col* is a number that specifies whether the series should be entered in:

Row_col	Series
1	Rows
2	Columns

*Type* is a number that specifies the type of series, as follows:

Type	Series
1	Linear
2	Growth
3	Date

*Date* is a number that specifies the type of date series, as follows:

Date	Series
1	Day
2	Weekday
3	Month
4	Year

*Step* and *stop* are the step and stop values, respectively.

## DATE(year,month,day)

Returns the serial number of a specified date. For more information, see DATE in Chapter 2, “Worksheet Function Directory.”

## DATEVALUE(date\_text)

Returns the serial number of *date\_text*. For more information, see DATEVALUE in Chapter 2, “Worksheet Function Directory.”

## DAVERAGE(database,field,criteria)

Returns the average of numbers in specified *field* of *database* records matching *criteria*. For more information, see DAVERAGE in Chapter 2, “Worksheet Function Directory.”

## DAY(serial\_number)

Converts *serial\_number* to a day of the month. For more information, see DAY in Chapter 2, “Worksheet Function Directory.”

## DCOUNT(database,field,criteria)

Returns the count of numbers in specified *field* of *database* records matching *criteria*. For more information, see DCOUNT in Chapter 2, “Worksheet Function Directory.”

## DCOUNTA(database,field,criteria)

Returns the count of non-empty cells in specified *field* of *database* records matching *criteria*. For more information, see DCOUNTA in Chapter 2, “Worksheet Function Directory.”

## DDB(cost,salvage,life,period)

Returns the depreciation of an asset using the double-declining balance method. For more information, see DDB in Chapter 2, “Worksheet Function Directory.”

## DEFINE.NAME(name\_text,refers\_to,macro\_type,shortcut\_text)

## DEFINE.NAME?(name\_text,refers\_to,macro\_type,shortcut\_text)

Equivalent to the Formula Define Name command. It defines the name *name\_text* on the active worksheet.

*Refers\_to* describes what *name\_text* should refer to, and can be any of the following values:

If refers_to is	Then name_text is
A number, text, or logical value	Defined to refer to that value
An external reference, such as !\$A\$1 or Sales!\$A\$1:\$C\$3	Defined to refer to those cells
A formula in the form of text, such as “=2*PI()/360”. (If the formula contains references, those references must be R1C1-style references, such as “=R2C2*(1+RC[-1])”.)	Defined to refer to that formula
Omitted	Defined to refer to the cells in the current selection

If you are recording a macro, and you define a name to refer to a formula, Microsoft Excel converts A1-style references to R1C1-style references. For example, if the active cell is C2, and you define the name Previous to refer to =B2, Microsoft Excel records that command as  
DEFINE.NAME(“Previous”,“=RC[-1]”).

The arguments *macro\_type* and *shortcut\_text* apply only if the document in the active window is a macro sheet.

*Macro\_type* is a number that indicates the following:

Macro_type	Macro
1	Function macro
2	Command macro
3	None (that is, <i>name_text</i> does not refer to a macro)

If *macro\_type* is omitted, it is assumed to be 3.

*Shortcut\_text* is a text value that specifies the shortcut key, and must be a single letter, such as “z” or “Z”.

## Examples

The following function defines the name Inflation on the active worksheet to refer to 0.12:

```
DEFINE.NAME("Inflation",0.12)
```

If the selection is A1:C3, each of the following functions defines the name Sales on the active worksheet to refer to cells A1:C3:

```
DEFINE.NAME("Sales")
```

```
DEFINE.NAME("Sales",SELECTION())
```

The following function defines the name Previous to refer to the relative reference RC[ - 1]:

```
DEFINE.NAME("Previous","=RC[ - 1]")
```

The following function defines the name Sales on the active worksheet to refer to A1:C1, the first row of A1:C3:

```
DEFINE.NAME("Sales",INDEX(SELECTION(),1,0))
```

## DELETE.ARROW()

Equivalent to the Chart Delete Arrow command. Deletes the selected arrow.

If the selection is not an arrow, returns FALSE.

## DELETE.BAR(*bar\_num*)

Deletes the custom menu bar numbered *bar\_num*. *Bar\_num* must be the number returned by a previously executed ADD.BAR function and cannot be the currently displayed menu bar.

For information on custom menus, see “Creating Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

## DELETE.COMMAND(*bar\_num*,*menu\_pos*,*command\_pos*)

Deletes the command in position *command\_pos* on the menu *menu\_pos* in bar number *bar\_num*. *Menu\_pos* can either be the number of a menu or the name of a menu, as text. *Command\_pos* can be either the number of the command to be deleted (the first command on a menu is command 1) or the title of the command, as text.

*Bar\_num* can either be the number of one of the Microsoft Excel built-in menu bars or the number returned by a previously executed ADD.BAR function.

If the specified command does not exist, DELETE.COMMAND returns the #VALUE! error value.

After a command is deleted, the *command\_pos* number for all commands below that command is decreased by 1.

For information on custom menus, see “Creating Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

## DELETE.FORMAT(*format\_text*)

Equivalent to the action of deleting the format specified by *format\_text* with the Format Number command. It deletes the specified format.

*Format\_text* must be a format string, for example, “#,##0.00”, as described in Number in *Microsoft Excel Reference*.

## DELETE.MENU(*bar\_num*,*menu\_pos*)

Deletes the menu *menu\_pos* in bar number *bar\_num*. *Menu\_pos* can either be the number of a menu or the name of a menu, as text.

*Bar\_num* can either be the number of one of the Microsoft Excel built-in menu bars or a reference to an ADD.BAR function that added a new menu bar.

If the specified menu does not exist, DELETE.MENU returns the #VALUE! error value.

After a menu is deleted, the *menu\_pos* number for all menus to the right of that menu is decreased by 1.

For information on custom menus, see “Creating Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

## DELETE.NAME(*name\_text*)

Equivalent to the action of deleting the name *name\_text* with the Formula Define Name command. It deletes the specified name.

## DELETE.OVERLAY()

Equivalent to the Chart Delete Overlay command (Full menus). Deletes an overlay from a chart. If the chart has no overlay, DELETE.OVERLAY takes no action and returns TRUE.

## DEREF(*reference*)

Returns the value of the cells in *reference*. If *reference* is the reference of a single cell, DEREF returns the value of that cell. If *reference* is the reference of a range of cells, DEREF returns the array of values in those cells.

In most formulas, there is no difference between using a value and using the reference of a cell which has that value. The reference is automatically translated to the value, as necessary. For example, if cell A1 contains the value 2, the formula =A1+1, like the formula =2+1, returns the result 3, because the reference A1 is converted to the value 2. However, in a few functions, such as the SET.NAME function, references are not automatically converted to values. Instead, those functions behave differently depending on whether an argument is a reference or a value.

If *reference* refers to the active sheet, it must be an absolute reference. Relative references are translated to absolute references.

For an example of the use of this function, see SET.NAME later in this chapter.

### DIALOG.BOX(dialog\_ref)

Displays the dialog box described in the area *dialog\_ref* on your macro sheet or worksheet. *Dialog\_ref* is described in “Creating Custom Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

If the OK button in the dialog box is chosen, DIALOG.BOX enters values in fields as specified in the *dialog\_ref* area and returns the item number of the button pressed. Items are numbered starting with 1 for the second row in *dialog\_ref*. (In other words, if the item number of the button pressed is *item\_number*, DIALOG.BOX returns  $\text{MATCH}(\text{item\_number}, \text{dialog\_ref}, 0) - 1$ .) If the Cancel button in the dialog box is chosen, DIALOG.BOX returns FALSE. If *dialog\_ref* is invalid, DIALOG.BOX returns the #VALUE! error value and when the macro runs, a message is displayed that tells you what cell in *dialog\_ref* contains the error found.

*Dialog\_ref* must be seven columns wide and at least two rows high.

For more information, see “Creating Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

### DIRECTORY(path\_text)

Sets the current drive and directory to the path specified by *path\_text* and returns the name of the new directory as text. If *path\_text* is not specified, it returns the name of the current directory as text. If *path\_text* does not include a drive specifier, the current drive is assumed.

#### Examples

The following function sets the directory to \EXCEL\MODELS on the current drive, and returns the value “drive:\EXCEL\MODELS”:

```
DIRECTORY (“\EXCEL\MODELS”)
```

The following function sets the current drive to E: and sets the directory to \EXCEL\MODELS on E:. It returns the value “E:\ EXCEL\ MODELS:”

```
DIRECTORY (“E:\EXCEL\MODELS”)
```



## DISABLE.INPUT(logical)

Blocks all input from the keyboard and mouse to Microsoft Excel (except input to displayed dialog boxes) if *logical* is TRUE; reenables input if *logical* is FALSE.

This can be useful if you are using the Dynamic Data Exchange facility (DDE) to communicate with Microsoft Excel from another application. For information on Dynamic Data Exchange, see Exchanging data between applications in *Microsoft Excel Reference*.

## DISPLAY(formula,gridline,heading,zero,color)

## DISPLAY(cell,formula,value,format,protect,names,precedents,dependents,note)

The first form is equivalent to the Options Display command; the second form is equivalent to the commands on the Info menu.

The arguments in the Info menu form correspond to the commands with the same names. For all the arguments except *precedents* and *dependents*: if the argument is TRUE, the corresponding Info item is displayed; if the argument is FALSE, it is not. If the argument is omitted, the status of the item is unchanged.

*Precedents* and *dependents* specify what precedents and dependents to list:

Precedents or Dependents	List
0	None
1	Direct only
2	All levels

The arguments for the Options Display form are:

Argument	Description
<i>formula</i>	Corresponds to the Formulas check box
<i>gridline</i>	Corresponds to the Gridlines check box
<i>heading</i>	Corresponds to the Row & Column Headings check box

Argument	Description
<i>zero</i>	Corresponds to the Zero Values check box
<i>color</i>	A number from 0–8. Numbers 1–8 correspond to the 8 colors in the Options Display dialog box; 0 corresponds to selecting the Automatic button. If <i>color</i> is 0, the color of gridlines and headers is the default text color for your Microsoft Windows configuration.

### DMAX(database,field,criteria)

Returns the maximum of numbers in a specified *field* of *database* records matching *criteria*. For more information, see DMAX in Chapter 2, “Worksheet Function Directory.”

### DMIN(database,field,criteria)

Returns the minimum of numbers in a specified *field* of *database* records matching *criteria*. For more information, see DMIN in Chapter 2, “Worksheet Function Directory.”

## DOCUMENTS()

Returns, as an array of text values, the names of all the open documents, in alphabetical order. With the INDEX function, you can select individual document names from the array to use in other functions that take document names as arguments.

### Example

If your workspace contains windows named BUDGET.XLS, Chart1, ACTUAL.XLS:1, and ACTUAL.XLS:2, each containing documents named BUDGET.XLS, Chart1, and ACTUAL.XLS, then:

DOCUMENTS() equals {"BUDGET.XLS","Chart1","ACTUAL.XLS"}

### DOLLAR(number,decimals)

Rounds *number* and gives as text in currency format. For more information, see DOLLAR in Chapter 2, “Worksheet Function Directory.”

## DPRODUCT(database,field,criteria)

Returns the product of numbers in a specified *field* of *database* records matching *criteria*. For more information, see DPRODUCT in Chapter 2, “Worksheet Function Directory.”

## DSTDEV(database,field,criteria)

Estimates standard deviation of a population based on a sample, using numbers in a specified *field* of *database* records matching *criteria*. For more information, see DSTDEV in Chapter 2, “Worksheet Function Directory.”

## DSTDEVP(database,field,criteria)

Returns the standard deviation of a population, based on the entire population, using numbers in a specified *field* of *database* records matching *criteria*. For more information, see DSTDEVP in Chapter 2, “Worksheet Function Directory.”

## DSUM(database,field,criteria)

Returns the sum of numbers in a specified *field* of *database* records matching *criteria*. For more information, see DSUM in Chapter 2, “Worksheet Function Directory.”

## DVAR(database,field,criteria)

Estimates variance of a population based on a sample, using numbers in a specified *field* of *database* records matching *criteria*. For more information, see DVAR in Chapter 2, “Worksheet Function Directory.”

## DVARP(database,field,criteria)

Returns the variance of a population based on the entire population, using numbers in a specified *field* of *database* records matching *criteria*. For more information, see DVARP in Chapter 2, “Worksheet Function Directory.”

## ECHO(logical)

Controls screen updating while a macro is running. If *logical* is FALSE, Microsoft Excel turns off screen updating. If *logical* is TRUE or omitted, Microsoft Excel turns on screen updating. Screen updating is always turned back on when a macro ends.

Large command macros generally run faster when screen updating is turned off.

## EDIT.DELETE(*num*)

## EDIT.DELETE?(*num*)

Equivalent to the Edit Delete command.

*Num* specifies the direction to shift cells, as follows:

<b>num</b>	<b>Direction</b>
1	Shift cells left
2	Shift cells up

## ENABLE.COMMAND(*bar\_num*,*menu\_pos*,*command\_pos*,*enable*)

Enables or disables the command in position *command\_pos* on the menu *menu\_pos* in menu bar number *bar\_num*. *Menu\_pos* can either be the number of a menu or the name of a menu, as text. *Command\_pos* can be either the number of the command to be checked (the first command on a menu is command 1) or the title of the command, as text. If *command\_pos* is 0, the entire menu is enabled or disabled.

*Bar\_num* must be the number of a built-in menu bar or the number returned by a previously executed ADD.BAR function.

If *enable* is TRUE, ENABLE.COMMAND enables command, if FALSE, it disables it. Disabled commands appear greyed and can't be selected.

If the specified command is a built-in Microsoft Excel command or does not exist, ENABLE.COMMAND returns the #VALUE! error value.

For information on custom menus, see "Creating Customized Menus and Dialog Boxes" in Chapter 6, "Advanced Macros."

## ERROR(*enable*,*macro\_ref*)

Specifies what action to take if an error is encountered while a macro is running.

If *enable* is FALSE, all error checking is turned off. When error checking is turned off, if an error is encountered while a macro is running, Microsoft Excel ignores it and continues.

If *enable* is TRUE, you can either turn on normal error checking (by omitting the other argument) or specify *macro\_ref*: the reference of a macro to run whenever an error is encountered. When normal error checking is active, a dialog box is displayed when an error is encountered. You can choose to halt the macro, start single-stepping through the macro, or continue normal running of the macro.

**Important** | Both `ERROR(TRUE,macro_ref)` and `ERROR(FALSE)` keep Microsoft Excel from displaying any messages at all. `ERROR(FALSE)` also bypasses the message you get when you close an unsaved document. Make sure that you won't need any of the Microsoft Excel messages while your macro is running before using either of these forms of the `ERROR` function.

## EXACT(text1,text2)

Tests to see if *text1* and *text2* are exactly the same. For more information, see `EXACT` in Chapter 2, "Worksheet Function Directory."

## EXEC(program\_text>window\_number)

**Note** | This function is only supported with Microsoft Windows (version 2.0, or higher).

Starts the program named *program\_text* as a separate program running under Microsoft Windows (version 2.0, or higher). *Program\_text* uses exactly the same form as the File Run command in the Windows MS-DOS Executive; it can include any arguments and switches that the program to be started accepts. If *program\_text* is the name of a file with an extension specific to an installed application, `EXEC` starts the application, using the filename as a parameter.

*Window\_number* specifies how the window containing the program should appear, as follows:

Window_number	Window Type
1	Normal window
2	Minimized window
3	Maximized window

If *window\_number* is omitted, it is assumed to be 2.

If the `EXEC` function is successful, it returns the Microsoft Windows task ID number of the started program. The task ID number is a unique number that identifies a program running under Microsoft Windows (version 2.0, or higher). The task ID number can be useful if, for example, you have several instances of one program running, and want to use `INITIATE` to open a channel to one particular program.

If the `EXEC` function is unsuccessful, it returns the `#VALUE!` error value.

### Examples

=EXEC("SALES.XLS") starts Microsoft Excel and loads the document SALES.XLS.

=EXEC("SALES.EXE") starts the program SALES.EXE.

=EXEC("EXCEL.EXE SALES.WKS") starts Microsoft Excel and loads the document SALES.WKS.

## EXECUTE(channel\_num,execute\_text)

**Note** | This function is supported only if you have Microsoft Windows (version 2.0, or higher).

Carries out the command or commands described by *execute\_text* in the application connected to the channel *channel\_num*. The form of *execute\_text* depends on the application you are accessing. To include specific key sequences in *execute\_text*, use the format described in the appendix.

The channel *channel\_num* must have been opened by the INITIATE macro function.

If EXECUTE is not successful, it returns the following values:

Situation	Return value
<i>Channel_num</i> is not a valid channel number	#VALUE!
The application you are accessing is busy doing something else	#N/A
The application you are accessing does not respond after a certain length of time and ESCAPE is pressed to cancel	#DIV/0!
The EXECUTE request is refused	#REF!

For information on accessing other applications, see "Using Macros to Start Other Applications" in Chapter 6, "Advanced Macros."

## EXP(number)

Returns *e* to the power *number*. For more information, see EXP in Chapter 2, “Worksheet Function Directory.”

## EXTRACT(unique\_log)

## EXTRACT?(unique\_log)

Equivalent to the Data Extract command (Full menus).

*Unique\_log* is a logical value corresponding to the Unique check box. If *unique\_log* is TRUE, the check box is turned on; if *unique\_log* is FALSE, the check box is turned off.

## FACT(number)

Returns the factorial of *number*. For more information, see FACT in Chapter 2, “Worksheet Function Directory.”

## FALSE()

Returns the logical value FALSE. For more information, see FALSE in Chapter 2, “Worksheet Function Directory.”

## FCLOSE(file\_number)

Closes the file specified by *file\_number*. The file specified by *file\_number* must have been opened by the FOPEN function. *File\_number* is the number returned by the FOPEN function that opened the file.

If *file\_number* is not a valid file number, FCLOSE returns the #VALUE! error value.

## FILE.CLOSE()

Equivalent to the File Close command. Closes the active document.

## FILE.DELETE(*name\_text*)

## FILE.DELETE?(*name\_text*)

Equivalent to the File Delete command (Full menus).

*Name\_text* is the name of the document to delete. If Microsoft Excel can't find the document *name\_text*, it displays a dialog box requesting that you insert a disk containing *name\_text*.

In the dialog box form, FILE.DELETE?, you can use an asterisk (\*) to represent any series of characters and a question mark (?) to represent any single character. The formula =FILE.DELETE?(\*.XL?) for example, displays the File Delete dialog box listing all documents whose extension begins with the letters XL.

## FILES(*directory\_text*)

Returns a horizontal text array of the names of files in the directory specified by *directory\_text*. You can use the wild cards asterisk (\*) and question mark (?) in the *directory\_text* argument. If *directory\_text* is not specified, it is assumed to be "\*. \*". FILES can return as many as 256 file names in the array.

### Example

If you enter =FILES() as an array formula in five cells in a row on your macro sheet, FILES returns the name of the first five documents in the current directory. If there are less than five documents in the current directory, the #N/A! error value is entered in the remaining cells.

## FILL.DOWN()

Equivalent to the Edit Fill Down command.

## FILL.LEFT()

Equivalent to the Edit Fill Left command.



## FILL.RIGHT()

Equivalent to the Edit Fill Right command.

## FILL.UP()

Equivalent to the Edit Fill Up command.

## FIND(*find\_text*,*within\_text*,*start\_at\_num*)

Finds *find\_text* within *within\_text*. For more information, see FIND in Chapter 2, “Worksheet Function Directory.”

## FIXED(*number*,*decimals*)

Rounds *number* and gives as text. For more information, see FIXED in Chapter 2, “Worksheet Function Directory.”

## FOPEN(*file\_text*,*access\_number*)

Opens the document named *file\_text*. *Access\_number* specifies what type of access to allow to the document, as follows:

<b>access_number</b>	<b>Type of access</b>
1	Can both read and write to the file (Read/Write access)
2	Can read the file, but can't write to the file (Read-only access)
3	Creates a new file, with read/write access

If the document doesn't exist and *access\_number* is 3, FOPEN creates a new document. If the document doesn't exist and *access\_number* is 1 or 2, FOPEN returns the #N/A! error value.

If the document is opened successfully, FOPEN returns a document ID number. If it can't open the document, FOPEN returns the #N/A! error value.

Make sure you use FCLOSE to close a document after you're done using it.

For information on using text documents, see “Text File Input and Output” in Chapter 6, “Advanced Macros.”

## FOR(counter\_name,start\_num,end\_num,step\_num)

Starts a FOR-NEXT loop. *Counter\_name* must be a name in the form of text.

Step Order	Action
First	Sets <i>counter_name</i> to the value <i>start_num</i> .
Second	<p>If the value of <i>counter_name</i> is greater than <i>end_num</i>, execution continues with the function after the NEXT function.</p> <p>If the value of <i>counter_name</i> is less than or equal to <i>end_num</i>, execution continues.</p>
Third	Executes functions up to the following NEXT function. The next NEXT function must be in the same column as the FOR function, and below it.
Fourth	Adds <i>step_num</i> to <i>counter_name</i> . If <i>step_num</i> is omitted, it is assumed to be 1.
Fifth	Execution moves to the FOR function.

You can also interrupt FOR-NEXT loops by using the BREAK function.

For more information, see “Looping” in Chapter 3, “Macro Basics.”

**FORMAT.FONT**(name\_text,size\_num,bold,italic,underline,strike)

**FORMAT.FONT?**(name\_text,size\_num,bold,italic,underline,strike)

**FORMAT.FONT**(color,backgd,apply,name\_text,size,bold,italic,underline,stri

## FORMAT.FONT?(*color,backgd,apply,name\_text,size,bold,italic,underline,strike*)

Equivalent to the Format Font command. FORMAT.FONT applies one of the four current fonts to the current selection.

The first form applies if the active document is a worksheet or macro sheet; the second form applies if the active document is a chart.

The following list summarizes the arguments to the worksheet and macro sheet form:

Argument	Description
<i>name_text</i>	The name of the font, as it appears in the dialog box. For example, "Courier" and "TmsRmn" are names of fonts
<i>size_num</i>	The size of the font, in points
<i>bold</i>	TRUE if the new font is bold, FALSE otherwise
<i>italic</i>	TRUE if the new font is italic, FALSE otherwise
<i>underline</i>	TRUE if the new font is underlined, FALSE otherwise
<i>strike</i>	TRUE to draw a line through (strikeout) the new font, FALSE otherwise

The following list summarizes the arguments to the chart form:

Argument	Description
<i>color</i>	A number from 0–8. Numbers 1–8 correspond to the 8 colors in the Format Font dialog box; 0 corresponds to selecting the Automatic button
<i>backgd</i>	1 = Automatic 2 = Transparent 3 = White Out

Argument	Description
<i>apply</i>	Corresponds to the Apply to All check box. This argument applies to data labels only
<i>name_text</i>	The name of the font, as it appears in the dialog box. For example, “Courier” and “TmsRmn” are names of fonts
<i>size</i>	The size of the font, in points
<i>bold</i>	Corresponds to the Bold check box
<i>italic</i>	Corresponds to the Italic check box
<i>underline</i>	Corresponds to the Underline check box
<i>strike</i>	Corresponds to the Strikeout check box

**Note** | If you specify a font that is not one of the current four fonts, Microsoft Excel tries to find a similar font and reformats using that font.

### FORMAT.LEGEND(position\_num)

Equivalent to the Format Legend command.

*Position\_num* is an integer specifying the position of the legend, as follows:

Position_num	Position
1	Bottom
2	Corner
3	Top
4	Vertical

### FORMAT.MOVE(x\_pos,y\_pos)

### FORMAT.MOVE?(x\_pos,y\_pos)

Equivalent to the Format Move command.

Moves the base of the selected chart object to the horizontal position specified by *x\_pos* and the vertical position specified by *y\_pos*.

**Note** | The base of a text label is the lower-left corner of the text rectangle. The base of an arrow is the end without the arrowhead. The base of a pie slice is the point.

The units for *x\_pos* and *y\_pos* are points and are measured from the lower-left corner of the window.

If the selected object cannot be moved, FORMAT.MOVE returns FALSE.

## FORMAT.NUMBER(format\_text)

## FORMAT.NUMBER?(format\_text)

Equivalent to the Format Number command.

*Format\_text* is a format string, such as “#,###0.00”. For information on number and text formats, see Number in *Microsoft Excel Reference*.

## FORMAT.SIZE(width,height)

## FORMAT.SIZE?(width,height)

Equivalent to the Format Size command.

Sizes the selected chart object to the horizontal size specified by *width* and the vertical size specified by *height*.

**Note** | The base of a text label is the lower-left corner of the text rectangle. The base of an arrow is the end without the arrowhead. Pie charts can't be sized.

The units for *width* and *height* are points.

If the selected object cannot be sized, FORMAT.SIZE returns FALSE.

## FORMAT.TEXT(x\_align,y\_align,vert\_text,auto\_text,auto\_size,show\_key,show\_value)

Equivalent to the Format Text command. Formats the selected text. The following list describes the arguments. Arguments that correspond to check boxes are TRUE if the check box is turned on and FALSE if the check box is turned off.

Argument	Description
<i>x_align</i>	Specifies the horizontal alignment: 1 = Left 2 = Center 3 = Right
<i>y_align</i>	Specifies the vertical alignment: 1 = Top 2 = Center 3 = Bottom
<i>vert_text</i>	Corresponds to the Vertical Text check box
<i>auto_text</i>	Corresponds to the Automatic Text check box
<i>auto_size</i>	Corresponds to the Automatic Size check box
<i>show_key</i>	Corresponds to the Show Key check box. Argument applies only if the selected text is an attached data label
<i>show_value</i>	Corresponds to the Show Value check box. Argument applies only if the selected text is an attached data label

### FORMULA(*formula\_text*,*ref*)

- **If the active document is a worksheet:** Equivalent to the action of entering a formula in a cell. It enters the formula specified by *formula\_text* into the cell specified by *ref*. If *ref* is omitted, the formula is entered in the active cell.

The formula is entered just as if you typed it in the formula bar.

*Formula\_text* can be a formula in the form of text, such as “=2\*PI()/360”, or a number, text, or logical value. If *formula\_text* is a formula, the formula is entered. If *formula\_text* is a number, text, or logical value, the value is entered as a constant.

If *formula\_text* contains references, those references must be R1C1-style references, such as “=RC[-1]\*(1+R1C1)”. If you are recording a macro when you enter a formula, Microsoft Excel converts A1-style

references to R1C1-style references. For example, if you enter the formula `=B2*(1+$A$1)` in cell C2, Microsoft Excel records that action as `=FORMULA("=RC[-1]*(1+R1C1)")`.

- **If the active document is a chart:** Enters text labels or SERIES functions.

If	Then
<i>Formula_text</i> can be treated as a text label, and the current selection is a text label	The selected text label is replaced with <i>formula_text</i>
<i>Formula_text</i> can be treated as a text label, and the current selection is not a text label	<i>Formula_text</i> makes a new text label
<i>Formula_text</i> can be treated as a SERIES formula, and the current selection is a SERIES formula	The selected SERIES formula is replaced with <i>formula_text</i>
<i>Formula_text</i> can be treated as a SERIES formula, and the current selection is not a SERIES formula	<i>Formula_text</i> makes a new SERIES formula

## Examples

- **If the active document is a worksheet:**

The following formula enters the constant number 523 in the active cell:

`=FORMULA(523)`

If the active cell is C2, the following formula enters the formula `=RC[-1]*(1+R1C1)`, or `=B2*(1+$A$1)`, in cell C2:

`=FORMULA("=RC[-1]*(1+R1C1)")`

The following formula takes the result of the INPUT function and enters it in the active cell:

`=FORMULA(INPUT("Enter a formula:"),0)`

### ■ If the active document is a chart:

The following formula enters a SERIES formula on the chart. If the current selection is a SERIES formula, it is replaced:

=FORMULA("=SERIES("Chart Title",,1,2,3,1)")

**Tip** | Don't forget that, within a text value, you must enter two sets of double quotation marks (""") to represent a single quotation mark. So if one part of the formula that you want to enter is a text value, you will need to use two sets of double quotation marks.

## FORMULA.ARRAY(formula\_text,ref)

Equivalent to the action of entering an array formula while pressing CONTROL+SHIFT+ENTER. It enters the specified formula as an array formula in the range specified by *ref* or in the current selection.

For information on *formula\_text*, see FORMULA earlier in this chapter.

## FORMULA.FILL(formula\_text,ref)

Equivalent to the action of entering a formula while pressing SHIFT. It enters the specified formula in *ref*, or, if *ref* is omitted, in the current selection.

For information on *formula\_text*, see FORMULA earlier in this chapter.

## FORMULA.FIND(text,in\_num,at\_num,by\_num)

## FORMULA.FIND?(text,in\_num,at\_num,by\_num)

Equivalent to the Formula Find command. *Text* specifies what to find. The other three arguments specify how to search, as follows:

Argument	Description
<i>in_num</i>	1 = Formulas 2 = Values 3 = Notes
<i>at_num</i>	1 = Whole 2 = Part
<i>by_num</i>	1 = Rows 2 = Columns



If a matching cell is not found, FORMULA.FIND returns FALSE and Microsoft Excel displays a message.

Note | The dialog box form of FORMULA.FIND is also equivalent to pressing SHIFT+F5.

FORMULA.FIND.NEXT()

FORMULA.FIND.PREV()

Equivalent to the actions of pressing F7 and SHIFT+F7, respectively. They find the next and previous cells on the worksheet, as specified in the Formula Find dialog box. If a matching cell is not found, the function returns FALSE.

FORMULA.GOTO(reference)

FORMULA.GOTO?(reference)

Equivalent to choosing the Formula Goto command, or pressing F5.

*Reference* should be either an external reference to a document or an R1C1-style reference in the form of text. For information on references, see “Using References” in Chapter 4, “Writing Macros.”

If you are recording a macro, the reference you enter in the Goto dialog box is recorded as text, with A1-style references converted to R1C1-style references.

If the Formula Goto command has already been carried out, *reference* is optional. If omitted, it is assumed to be the reference of the selection that was current before Formula Goto was carried out.

### Examples

Each of the following functions goes to cell A1 on the active worksheet:

FORMULA.GOTO(!\$A\$1)

FORMULA.GOTO(“R1C1”)

Each of the following functions goes to the cells named Sales on the active worksheet:

FORMULA.GOTO(!Sales)

FORMULA.GOTO("Sales")

Each of the following functions goes to cells B2:C3 on a worksheet named BUDGET.XLS:

FORMULA.GOTO(BUDGET.XLS!\$B\$2:\$C\$3)

FORMULA.GOTO("BUDGET.XLS!R2C2:R3C3")

**FORMULA.REPLACE**(*find\_text*,*replace\_text*,*look\_at*,*look\_by*,  
*current\_cell*)

**FORMULA.REPLACE?**(*find\_text*,*replace\_text*,*look\_at*,*look\_by*,  
*current\_cell*)

Equivalent to the Formula Replace command (Full menus). *Find\_text* is what to find; *replace\_text* is what to replace *find\_text* with.

*Look\_at* and *look\_by* are numbers describing how to search. Use *look\_at* = 1 to look at the whole; *look\_at* = 2 to look at a part. Use *look\_by* = 1 to look by rows; *look\_by* = 2 to look by columns.

*Current\_cell* is a logical value specifying what cells to replace *find\_text* in. If *current\_cell* is TRUE, *find\_text* is replaced in the current cell only. If *current\_cell* is FALSE or omitted, *find\_text* is replaced in the entire selection, or in the entire document, if the selection is a single cell.

You can use the wildcard characters question mark (?) and asterisk (\*) in *find\_text*. A question mark (?) matches any single character; an asterisk (\*) matches any sequence of characters. To match a question mark (?) or asterisk (\*), precede the character with a tilde (~).

In the dialog box form of FORMULA.REPLACE, omitted arguments are assumed to be the same arguments used in the previous replace operation. If there was no previous replace operation, omitted text arguments are assumed to be "" (the empty text).

### Examples

The following function replaces 7 with 8 in the entire selection, or in the entire document if the selection is a single cell:

FORMULA.REPLACE("7","8")

The following function replaces any instances in the current cell of y followed by any two characters with y:

FORMULA.REPLACE("y??","y",TRUE)

The following function replaces any instances in the current cell of y followed by two question marks with y:

FORMULA.REPLACE("y~??","y",TRUE)

## FPOS(file\_number,position\_number)

Positions the document *file\_number* to the position *position\_number*, where the first position in the document is 1. The document *file\_number* must have been opened with the FOPEN function. *File\_number* is the number returned by the FOPEN function. If *position\_number* is not specified, FPOS returns the current position of the document.

The document must have been opened with the FOPEN statement.

If *file\_number* is not a valid document number, FPOS returns the #VALUE! error value.

For information on using text files, see "Text File Input and Output" in Chapter 6, "Advanced Macros."

## FREAD(file\_number,num\_chars)

Reads *num\_chars* characters from the document *file\_number*, starting at the current position in the document. The document *file\_number* must have been opened with the FOPEN function. *File\_number* is the number returned by the FOPEN function.

If FREAD is successful, it returns the text read. If *file\_number* is not a valid document number, FREAD returns the #VALUE! error value. If the end of the document is reached or if FREAD can't read the document, it returns the #N/A! error value.

For information on using text files, see "Text File Input and Output" in Chapter 6, "Advanced Macros."

## FREADLN(file\_number)

Reads from the current document position in the document *file\_number* to the end of the line. The document *file\_number* must have been opened with the FOPEN function. *File\_number* is the number returned by the FOPEN function.

If FREADLN is successful, it returns the text read, up to but not including the carriage-return and linefeed characters at the end of the line. If *file\_number* is not a valid document number, FREADLN returns the #VALUE! error value. If the current document position is the end of the document or if FREADLN can't read the document, it returns the #N/A! error value.

For more information, see “Text File Input and Output” in Chapter 6, “Advanced Macros.”

### FREEZE.PANES(logical)

Equivalent to the Options Freeze Panes command (Full menus) if *logical* is TRUE; equivalent to the Options Unfreeze Panes command (Full menus) if *logical* is FALSE.

### FSIZE(file\_number)

Returns the number of characters in the document *file\_number*. The document *file\_number* must have been opened with the FOPEN function. *File\_number* is the number returned by the FOPEN function.

If *file\_number* is not a valid document number, FSIZE returns the #VALUE! error value.

### FULL(logical)

If *logical* is TRUE, equivalent to the action of maximizing the active document window by pressing CONTROL+F10. If *logical* is FALSE, equivalent to the action of restoring the size of the active window by pressing CONTROL+F5.

### FV(rate,nper,pmt,pv,type)

Returns the future value of investment. For information on FV, see Chapter 2, “Worksheet Function Directory.”

### FWRITE(file\_number,text)

Writes *text* to the document *file\_number*, starting at the current position in that document. The document *file\_number* must have been opened with the FOPEN function. *File\_number* is the number returned by the FOPEN function.

If *file\_number* is not a valid document number, FWRITE returns the #VALUE! error value. If FWRITE can't write to the document, it returns the #N/A! error value.

For more information, see “Text File Input and Output” in Chapter 6, “Advanced Macros.”

## **FWRITELN(*file\_number*,*text*)**

Writes *text*, followed by a carriage return and linefeed, to the document *file\_number*, starting at the current position in that document. The document *file\_number* must have been opened with the FOPEN function. *File\_number* is the number returned by the FOPEN function.

If *file\_number* is not a valid document number, FWRITELN returns the #VALUE! error value. If FWRITELN can't write to the document, it returns the #N/A! error value.

For more information, see “Text File Input and Output” in Chapter 6, “Advanced Macros.”

## **GALLERY.AREA(*number*,*delete\_overlay*)**

## **GALLERY.AREA?(*number*,*delete\_overlay*)**

Equivalent to the Gallery Area command. *Number* is the number of the format in the gallery. *Delete\_overlay* is a logical value. If *delete\_overlay* is TRUE, the function deletes an overlay chart if one is present, and applies the new format to the main chart. If *delete\_overlay* is FALSE or omitted, the new format is applied to the chart containing the currently selected chart item.

## **GALLERY.BAR(*number*,*delete\_overlay*)**

## **GALLERY.BAR?(*number*,*delete\_overlay*)**

Equivalent to the Gallery Bar command. *Number* is the number of the format in the gallery. *Delete\_overlay* is a logical value. If *delete\_overlay* is TRUE, the function deletes any overlay charts present and applies the new format to the main chart. If *delete\_overlay* is FALSE or omitted, the new format is applied to the chart that contains the currently selected chart item.

GALLERY.COLUMN(*number,delete\_overlay*)

GALLERY.COLUMN?(*number,delete\_overlay*)

Equivalent to Gallery Column command. *Number* is the number of the format in the gallery. *Delete\_overlay* is a logical value. If *delete\_overlay* is TRUE, the function deletes any overlay charts present and applies the new format to the main chart. If *delete\_overlay* is FALSE or omitted, the new format is applied to the chart that contains the currently selected chart item.

GALLERY.LINE(*number,delete\_overlay*)

GALLERY.LINE?(*number,delete\_overlay*)

Equivalent to the Gallery Line command. *Number* is the number of the format in the gallery. *Delete\_overlay* is a logical value. If *delete\_overlay* is TRUE, the function deletes any overlay charts present and applies the new format to the main chart. If *delete\_overlay* is FALSE or omitted, the new format is applied to the chart that contains the currently selected chart item.

GALLERY.PIE(*number,delete\_overlay*)

GALLERY.PIE?(*number,delete\_overlay*)

Equivalent to the Gallery Pie command. *Number* is the number of the format in the gallery. *Delete\_overlay* is a logical value. If *delete\_overlay* is TRUE, the function deletes any overlay charts present and applies the new format to the main chart. If *delete\_overlay* is FALSE or omitted, the new format is applied to the chart that contains the currently selected chart item.

GALLERY.SCATTER(*number,delete\_overlay*)

GALLERY.SCATTER?(*number,delete\_overlay*)

Equivalent to the Gallery Scatter command. *Number* is the number of the format in the gallery. *Delete\_overlay* is a logical value. If *delete\_overlay* is TRUE, the function deletes any overlay charts present and applies the new format to the main chart. If *delete\_overlay* is FALSE or omitted, the new format is applied to the chart that contains the currently selected chart item.

## GET.BAR()

Returns the number of the active menu bar.

For information on menu bars, see “Creating Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

## GET.CELL(*type\_of\_info,reference*)

Returns information about the formatting, location, or contents of the cell in the upper-left corner of the first area in *reference*. If *reference* is omitted, it is assumed to be the current selection.

Use the *type\_of\_info* argument to specify what type of cell information you want. The following list shows the possible values of *type\_of\_info* and the corresponding results:

Type_of_info	Result
1	Reference of the top-left cell in <i>reference</i> , as text
2	Equals the row of the top cell in <i>reference</i>
3	Equals the column of the left-most cell in <i>reference</i>
4	Equals TYPE(reference)
5	The contents of <i>reference</i>
6	The formula in <i>reference</i> , as text
7	Format of cell, as text (for example “m/d/yy” or “General”)
8	A number indicating the cell’s alignment: 1 = General 2 = Left 3 = Center 4 = Right 5 = Fill
9	If cell has a left border, returns TRUE, otherwise FALSE
10	If cell has a right border, returns TRUE, otherwise FALSE

Type_of_info	Result
11	If cell has a top border, returns TRUE, otherwise FALSE
12	If cell has a bottom border, returns TRUE, otherwise FALSE
13	If cell is shaded, returns TRUE, otherwise FALSE
14	If cell is locked, returns TRUE, otherwise FALSE
15	If cell is hidden, returns TRUE, otherwise FALSE
16	Column width of cell, measured in characters of the font 1 for the document
17	Row height of cell, in points
18	Name of font, as text
19	Size of font, in points
20	If cell is bold, returns TRUE, otherwise FALSE
21	If cell is italic, returns TRUE, otherwise FALSE
22	If cell is underlined, returns TRUE, otherwise FALSE
23	If cell is struck over, returns TRUE, otherwise FALSE

## GET.CHART.ITEM(*x\_y\_index*,*point\_index*,*item\_text*)

Returns the vertical or horizontal position of a point on a chart item. The following list summarizes the arguments:

Argument	Description
<i>x_y_index</i>	1 if you want the horizontal coordinate of the position; 2 if you want the vertical coordinate.



Argument	Description
<i>point_index</i>	An index specifying the point on the chart object. These indexes are described below. If <i>point_index</i> is omitted it is assumed to be 1.
<i>item_text</i>	A selection code. For information on how to specify <i>item_text</i> , see the chart form of SELECT. If <i>text_item</i> is omitted it is assumed to be the currently selected item. If no item is selected, GET.CHART.ITEM returns #VALUE!.

If the selected object is a point, *point\_index* must be 1.

If the selected object is any line other than a data line, use these values for *point\_index*:

Point_index	Chart object position
1	Lower left
2	Upper right

**Note** | To get the position of a data line, specify a specific point on the line.

If the selected object is a rectangle or an area in an area chart, use these values for *point\_index*:

Point_index	Chart object position
1	Upper left
2	Upper middle
3	Upper right
4	Right middle
5	Lower right
6	Lower middle
7	Lower left
8	Left middle

If the selected object is an arrow, use these values for *point\_index*:

Point_index	Chart object position
1	The base
2	The head

If the selected object is a pie slice, use these values for *point\_index*:

Point_index	Chart object position
1	Outermost counter-clockwise point
2	Outer center point
3	Outermost clockwise point
4	Midpoint of the most clockwise radius
5	Center point
6	Midpoint of the most counter-clockwise radius

### GET.DEF(*def\_text*,*document*)

Returns as text the name, for the definition *def\_text* in *document*. If the definition is a reference, give the reference in R1C1 style, such as “R[2]C”. If there is more than one name for *def\_text*, GET.DEF returns the first name.

### GET.DOCUMENT(*type\_of\_info*,*name\_text*)

Returns information about the document named *name\_text*. *Name\_text* must be the name of a document that is currently open. If *name\_text* is omitted, it is assumed to be the active document.

Use the *type\_of\_info* argument to specify what type of document information you want. The following list shows the possible values of *type\_of\_info* and the corresponding results:

Type_of_info	Result
1	Name of the document <i>name_text</i> , as text. The document name does not include the drive, directory, or window number.

Type_of_info	Result
2	Pathname of directory containing <i>name_text</i> , as text. If the document <i>name_text</i> hasn't been saved yet, returns #N/A
3	1 if the document is a worksheet, 2 if the document is a chart, 3 if the document is a macro sheet, 4 if the active window is the Info window
4	TRUE if changes have been made to the document since it was last saved, FALSE if they haven't
5	TRUE if read-only, FALSE otherwise
6	TRUE if file is protected, FALSE otherwise
7	TRUE if document contents are protected, FALSE otherwise
8	TRUE if document windows are protected, FALSE otherwise

These three values of *type\_of\_info* apply only to charts:

Type_of_info	Result
9	A number from 1–6 indicating the type of the main chart: 1 = Area 2 = Bar 3 = Column 4 = Line 5 = Pie 6 = Scatter
10	A number from 1–6 indicating the type of the overlay chart. Same numbers as for main chart, above. If there is no overlay chart, returns #N/A!
11	Number of series in main chart
12	Number of series in overlay chart

The remaining values of *type\_of\_info* apply only to worksheets and macro sheets:

Type_of_info	Result
9	Number of first used row. If the document is empty, returns 0
10	Number of the last used row. If the document is empty, returns 0
11	Number of the first used column. If the document is empty, returns 0
12	Number of the last used column. If the document is empty, returns 0
13	Number of windows
14	Number indicating calculation mode: 1 = Automatic 2 = Automatic Except Tables 3 = Manual
15	TRUE if iteration is enabled, FALSE otherwise
16	Maximum number of iterations
17	Maximum change between iterations
18	TRUE if updating remote references is enabled, FALSE otherwise
19	TRUE if set to Precision As Displayed, FALSE otherwise
20	TRUE if document is set to 1904 numbering system, FALSE otherwise
21	4-item horizontal text array of the names of the four fonts
22	4-item horizontal number array of the sizes of the four fonts
23	4-item horizontal logical array indicating which of the four fonts is bold. (If, for example, fonts 1 and 4 are bold, would return the array {TRUE,FALSE,FALSE,TRUE}.)

Type_of_info	Result
24	4-item horizontal logical array indicating which of the four fonts are italic
25	4-item horizontal logical array indicating which of the four fonts are underlined
26	4-item horizontal logical array indicating which of the four fonts are struck over

## GET.FORMULA(reference)

Returns the contents of the upper-left corner cell in *reference* as they would appear in the formula bar. The contents are given in the form of text, for example, “=2\*PI()/360.” *Reference* can be an external reference.

If the formula contains references, they are given as R1C1-style references, such as “=RC[-1]\*(1+R1C1).”

### Examples

If cell A3 on the active worksheet contains the number 523, then:

GET.FORMULA(!A3) *equals* “523”

If cell C2 on a worksheet named BUDGET.XLS contains the formula  
=B2\*(1+\$A\$1), then:

GET.FORMULA(BUDGET.XLS!C2) *equals* “=RC[-1]\*(1+R1C1)”

## GET.NAME(name\_text)

Returns the definition of the name *name\_text*, as it would appear in the Refers to text box of the Formula Define Name command.

*Name\_text* can be a name defined on the macro sheet, an external reference to a name defined on the active document, or an external reference to a name defined on a particular worksheet.

If the definition of *name\_text* contains references, they are given as R1C1-style references.

### Examples

If the name Sales on a macro sheet is defined to be the number 523, then:

GET.NAME(“Sales”) *equals* “523”

If the name Profit on the active worksheet is defined to be the formula  
= Sales – Costs, then:

GET.NAME(“!Profit”) equals “= Sales – Costs”

If the name Database on a worksheet named STOCK.XLS is defined to be  
the range A1:F500, then:

GET.NAME(“STOCK.XLS!Database”) equals “= R1C1:R500C6”

### GET.NOTE(*cell\_ref*,*start\_char*,*count\_char*)

Returns *count\_char* characters from the note attached to cell *cell\_ref*, starting  
at cell *start\_char*. *Count\_char* must be less than or equal to 255.

If *start\_char* is omitted, it is assumed to be 1. If *count\_char* is omitted, it is  
assumed to be the length of the note attached to *cell\_ref*.

### GET.WINDOW(*type\_of\_info*,*name\_text*)

Returns information about the window named *name\_text*. If *name\_text* is  
omitted, it is assumed to be the active window.

Use the *type\_of\_info* argument to specify what type of window information  
you want. The following list shows the possible values of *type\_of\_info* and  
the corresponding results:

Type_of_info	Result
1	The name of the document in the window <i>name_text</i> , as text
2	Number of window <i>name_text</i>
3	X position, measured in points from the left edge of your screen to the left edge of the window
4	Y position, measured in points from the top edge of your screen to the top edge of the window
5	Width, measured in points
6	Height, measured in points
7	TRUE if window is hidden, FALSE otherwise

The rest of the values for *type\_of\_info* only apply to worksheets and macro sheets:

Type_of_info	Result
8	TRUE if formulas are displayed, FALSE otherwise
9	TRUE if gridlines are displayed, FALSE otherwise
10	TRUE if row and column headings are displayed, FALSE otherwise
11	TRUE if zeros are displayed, FALSE otherwise
12	A number from 0–8 specifying the color of gridlines and headings. Numbers 1–8 correspond to the colors shown in the Options Display dialog box. Number 0 corresponds to selecting the Automatic button

The next four values of *type\_of\_info* return horizontal numeric arrays that specify what rows or columns are at the edges of the panes in the window *name\_text*. The first number in the resulting array specifies the edge of the first pane, the second number specifies the edge of the second pane, and so on. If the edge of the pane occurs at the boundary between rows or columns, the number returned is an integer. If the edge of the pane occurs within a row or column, the number returned has a fractional part, which represents the fraction of the row or column that is visible within the pane. The numbers returned can be used as arguments to the SPLIT macro function to split a window at specific column or row boundaries.

Type_of_info	Result
13	The left-most column of each pane, in a numeric array
14	The top row of each pane, in a numeric array
15	The right-most column of each pane, in a numeric array
16	The bottom row for each pane, in a numeric array

### Examples

If the active window contains the document MACRO1, then:

=GET.WINDOW(1) *equals* "MACRO1"

If the title of the active window is Macro1:3, then:

=GET.WINDOW(2) *equals* 3

## GET.WORKSPACE(type\_of\_info)

Returns information about the workspace.

Use the *type\_of\_info* argument to specify what type of workspace information you want. The following list shows the possible values of *type\_of\_info* and the corresponding results:

Type_of_info	Result
1	The name of the environment in which Microsoft Excel is running, as text, followed by the environment's version number
2	The version number of Microsoft Excel, as text (for example, "2.0")
3	If auto-decimal is set, returns the number of decimals. Otherwise returns 0
4	TRUE if in R1C1 mode, FALSE if in A1 mode
5	TRUE if scroll bars are displayed, FALSE otherwise
6	TRUE if the status bar is displayed, FALSE otherwise
7	TRUE if the formula bar is displayed, FALSE otherwise
8	TRUE if remote requests are enabled, FALSE otherwise



Type_of_info	Result
9	Returns the alternate menu key, as text, or #N/A if no alternate menu key is set
10	A number indicating special modes: 1 = Data Find 2 = Copy 3 = Cut 0 if no special modes
11	X position of the Microsoft Excel window, measured in points from the left edge of your screen to the left edge of the window
12	Y position of the Microsoft Excel window, measured in points from the top edge of your screen to the top edge of the window
13	Usable workspace width, in points
14	Usable workspace height, in points
15	1 if Microsoft Excel is neither maximized nor minimized, 2 if minimized, 3 if maximized
16	Amount of memory free (in K bytes)
17	Total memory available to Microsoft Excel (in K bytes)
18	TRUE if a math coprocessor is present, FALSE otherwise
19	TRUE if a mouse is present, FALSE otherwise

## GOTO(reference)

Causes running of a macro to continue at the upper-left cell of *reference*.

*Reference* can be an external reference to another macro sheet. If that macro sheet is not open, GOTO displays a message.

### Example

On a macro sheet, if A1 contains the #N/A! error value, and if C3 contains the formula =ALERT("Not available.", 2), then when the following formula is calculated, calculation branches to C3 and Microsoft Excel displays a message:

=IF(ISERROR(A1),GOTO(C3),GOTO(D4))

## GRIDLINES(*cat\_major,cat\_minor,value\_major,value\_minor*)

## GRIDLINES?(*cat\_major,cat\_minor,value\_major,value\_minor*)

Equivalent to the Chart Gridlines command. The arguments correspond to the four check boxes in the Chart Gridlines dialog box, as shown in the list below. If an argument is TRUE, its check box is turned on; if an argument is FALSE, its check box is turned off.

Argument	Check box
<i>cat_major</i>	Category Axis: Show Major Gridlines
<i>cat_minor</i>	Category Axis: Show Minor Gridlines
<i>value_major</i>	Value Axis: Show Major Gridlines
<i>value_minor</i>	Value Axis: Show Minor Gridlines

## GROWTH(*known\_y's,known\_x's,new\_x's*)

Returns the values on an exponential trend. For more information, see GROWTH in Chapter 2, “Worksheet Function Directory.”

## HALT()

Stops all macros from running.

### Example

If A1 contains the #N/A! error value, then when the following formula is calculated, the macro halts:

=IF(ISERROR(A1),HALT(),GOTO(D4))

## HELP(*help\_ref*)

Starts Help, if it's not running already, and displays the Help topic specified by *help\_ref*. *Help\_ref* is a reference to a topic in a custom help file, in the form *filename!topic\_number*. If *help\_ref* is omitted, HELP displays the Microsoft Excel Help index.

For information on custom Help topics, see “Using Custom Help” in Chapter 6, “Advanced Macros.”

## HIDE()

Equivalent to the Window Hide command (Full menus). Hides the active window.

**Tip** | Hiding windows you are currently working with can speed up your macros. You activate hidden windows with the ACTIVATE function macro.

## HLINE(number\_cols)

Equivalent to the action of scrolling the active window by columns. Scrolls the active window horizontally by *number\_cols* columns.

If *number\_cols* is positive, HLINE scrolls to the right. If *number\_cols* is negative, HLINE scrolls to the left.

## HLOOKUP(lookup\_value,table\_array,row\_index\_num)

Returns the value in a table selected by *lookup\_value*. For more information, see HLOOKUP in Chapter 2, “Worksheet Function Directory.”

## HOURL(serial\_number)

Converts *serial\_number* to an hour of the day. For more information, see HOUR in Chapter 2, “Worksheet Function Directory.”

## HPAGE(number\_windows)

Equivalent to the action of scrolling the active window one window at a time. Scrolls the active window horizontally by *number\_windows* windows.

If *number\_windows* is positive, HPAGE scrolls to the right. If *number\_windows* is negative, HPAGE scrolls to the left.

## HSCROLL(scroll,col\_log)

Equivalent to the action of scrolling the active window horizontally.

If *col\_log* is TRUE, then HSCROLL scrolls to column *scroll*.

If *col\_log* is FALSE or omitted, then HSCROLL scrolls to the horizontal position represented by the fraction *scroll*. If *scroll* is 0, HSCROLL scrolls to the left-most edge of your document. If *scroll* is 1, HSCROLL scrolls to the right-most edge of your document.

To scroll to a specific column *n*, either use HSCROLL(*n*,TRUE) or use HSCROLL(*n*/256). To scroll to column 38, for example, use HSCROLL(38,TRUE) or HSCROLL(38/256).

If you are recording a macro and move the scroll box several times in a row, the recorder only records the final location of the scroll box, omitting any intermediate steps. Remember that scrolling does not change the active cell or the selection.

### Examples

All of the following functions scroll to column 128, 50% of the way across the worksheet:

HSCROLL(128,TRUE)

HSCROLL(50%)

HSCROLL(.5,FALSE)

HSCROLL(128/256)

## IF(logical\_test,value\_if\_true,value\_if\_false)

Returns *value\_if\_true* if *logical\_test* is TRUE; returns *value\_if\_false* if *logical\_test* is FALSE. For more information, see IF in Chapter 2, “Worksheet Function Directory.”

## INDEX(ref,row\_num,column\_num,area\_num)

## INDEX(array,row\_num,column\_num)

Returns a reference in *ref* or value in *array* selected by index values. For more information, see INDEX in Chapter 2, “Worksheet Function Directory.”

## INDIRECT(*ref,type\_of\_ref*)

Returns the contents of the cell from its *ref*. For more information, see INDIRECT in Chapter 2, “Worksheet Function Directory.”

## INITIATE(*app\_text,topic\_text*)

**Note** | This function is supported only if you have Microsoft Windows (version 2.0).

Opens a DDE channel to an application. For more information, see “Communicating with Other Windows Applications” in Chapter 6, “Advanced Macros.”

*App\_text* is the DDE name of the application you are accessing; the form of *app\_text* depends on the application you are accessing. The DDE name of Microsoft Excel, for example, is “Excel.”

*Topic\_text* describes something in the application that you are accessing; the form of *topic\_text* depends on the application you are accessing. Microsoft Excel accepts the names of the current documents as *topic\_text*, as well as the name “System.”

If INITIATE is successful, it returns the number of the open channel. All the subsequent DDE macro functions use this number to specify the channel.

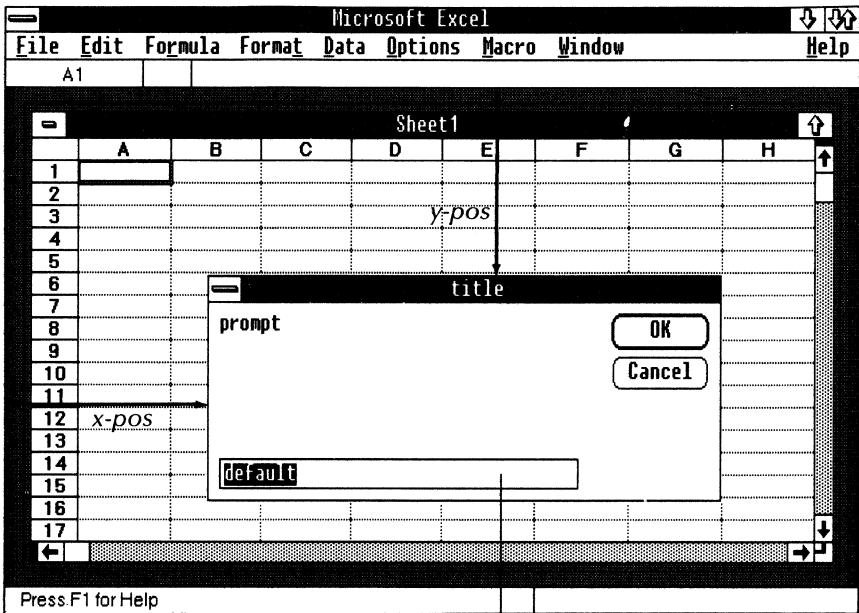
You can specify an instance of an application by appending the application’s task ID number to the *app\_text* argument. If you start an application by using the EXEC macro function, EXEC returns the task ID number for that instance of the application. If more than one instance of an application is running, and you do not specify which instance you would like to open a channel to, INITIATE displays a dialog box from which you can choose the instance you want. You can prevent this dialog box from appearing by disabling or redirecting errors with the ERROR function.

## INPUT(*prompt,type,title,default,x\_pos,y\_pos*)

Displays a dialog box. Returns the information entered in the dialog box. *Prompt*, *title*, and *default* must be text. The other three arguments must be numbers.

The dialog box produced by INPUT looks like this:

```
INPUT("prompt",type,"title",x-pos,y-pos)
```



Only accepts specified values

If *title* is omitted, it is assumed to be "Input". If *default* is omitted, the edit box is displayed empty. If *x-pos* or *y-pos* is omitted or 0, the dialog box is centered in that direction. The *x-pos* and *y-pos* arguments are given in points.

You can enter cell references in the edit box by selecting cells, and you can use editing commands such as Formula Paste Name and Formula Paste Function.

If you choose the OK button, or press ENTER, INPUT returns the value of what is in the edit box. If you choose the Cancel button, INPUT returns FALSE.

*Type* specifies the data type to be entered:

Type	Data type
0	Formula
1	Number
2	Text

Type	Data type
4	Logical
8	Reference
16	Error
64	Array

You can also use a sum of the allowable data types for *type*. For example, for an input box that accepts text or numbers, use *type* equal to 3.

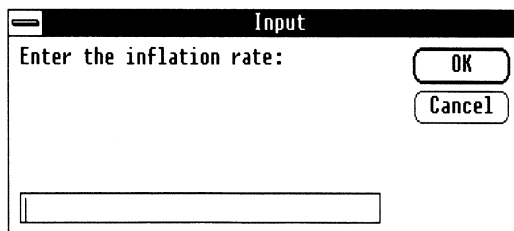
If *type* is 8, INPUT returns an absolute reference to the specified cells. If *type* is 0, INPUT returns the formula in the edit box in the form of text, for example, “=2\*PI()/360”. If the formula contains references, they are given as R1C1-style references, for example, “=RC[-1]\*(1+R1C1)”.

If the information entered in the edit box is not of the correct data type, Microsoft Excel attempts to translate it to the specified type. If it can't, it displays an error message.

## Examples

The following function displays the dialog box shown below:

INPUT(“Enter the inflation rate”,1)



If you then enter 12%, INPUT returns the value 0.12.

If in response to the function INPUT(“Select your database”,8) you select cells A1:C3 on the worksheet named BUDGET.XLS, INPUT returns the reference BUDGET.XLS!\$A\$1:\$C\$3.

If the active cell is C2, and in response to the function INPUT(“Enter your monthly increase formula”,0) you enter the formula =B2\*(1+\$A\$1), INPUT returns “=RC[-1]\*(1+R1C1)”.

If the value entered in the INPUT dialog box is a reference, and you use that value in a function or operation, you will usually get the value contained in the reference instead of the reference itself. That's because the reference is automatically translated into the contents of that reference. If you want to work with the actual reference, use the REFTEXT function to convert the reference to text, which you can then store or manipulate.

### INSERT(shift\_num)

### INSERT?(shift\_num)

Equivalent to the Edit Insert command.

*Shift\_num* specifies which way to shift the cells, as follows:

Shift_num	Direction
1	Shift Cells Right
2	Shift Cells Down

### INT(number)

Returns *number* rounded down to the nearest integer. For more information, see INT in Chapter 2, "Worksheet Function Directory."

### IPMT(rate,per,nper,pv,fv,type)

Returns the interest payment for an investment. For more information, see IPMT in Chapter 2, "Worksheet Function Directory."

### IRR(values,guess)

Returns the internal rate of return of *values*. For more information, see IRR in Chapter 2, "Worksheet Function Directory."

### ISBLANK(value)

Returns TRUE if *value* is blank. For more information, see ISBLANK in Chapter 2, "Worksheet Function Directory."



## ISERR(value)

Returns TRUE if *value* is any error value except #N/A. For more information, see ISERR in Chapter 2, “Worksheet Function Directory.”

## ISERROR(value)

Returns TRUE if *value* is any error value. For more information, see ISERROR in Chapter 2, “Worksheet Function Directory.”

## ISLOGICAL(value)

Returns TRUE if *value* is a logical value. For more information, see ISLOGICAL in Chapter 2, “Worksheet Function Directory.”

## ISNA(value)

Returns TRUE if *value* is the #N/A error value. For more information, see ISNA in Chapter 2, “Worksheet Function Directory.”

## ISNONTEXT(value)

Returns TRUE if *value* is not text. For more information, see ISNONTEXT in Chapter 2, “Worksheet Function Directory.”

## ISNUMBER(value)

Returns TRUE if *value* is a number. For more information, see ISNUMBER in Chapter 2, “Worksheet Function Directory.”

## ISREF(value)

Returns TRUE if *value* is a reference. For more information, see ISREF in Chapter 2, “Worksheet Function Directory.”

## ISTEXT(value)

Returns TRUE if *value* is text. For more information, see ISTEXT in Chapter 2, “Worksheet Function Directory.”

## JUSTIFY()

Equivalent to the Format Justify command (Full menus).

## LEFT(text,number\_of\_characters)

Extracts first *number\_of\_characters* from *text*. For more information, see LEFT in Chapter 2, “Worksheet Function Directory.”

## LEGEND(logical)

Equivalent to the Chart Add Legend command if *logical* is TRUE or omitted equivalent to the Chart Delete Legend command if *logical* is FALSE. If *logical* is FALSE and the active chart has no legend, LEGEND takes no action.

## LEN(text)

Returns the length of *text*. For more information, see LEN in Chapter 2, “Worksheet Function Directory.”

## LINEST(known\_y's,known\_x's)

Returns the parameters of a linear trend. For more information, see LINEST in Chapter 2, “Worksheet Function Directory.”

## LINKS(doc\_text)

Returns, as a horizontal array of text values, the names of all worksheets referred to by external references in the document specified by *doc\_text*. With the INDEX function, you can select individual worksheet names from the array for use in other functions that take document names as arguments. If *doc\_text* is omitted, it is assumed to be the name of the active document. If the active document contains no external references, LINKS returns the #N/A error value.

### Example

If the chart named VARIANCE.XLS is open and contains data series that refer to worksheets named BUDGET.XLS and ACTUAL.XLS, then when this function is calculated, Microsoft Excel opens the worksheets named BUDGET.XLS and ACTUAL.XLS:

OPEN.LINKS (LINKS(“VARIANCE.XLS”))

## LIST.NAMES()

Equivalent to choosing the Formula Paste Name command and pressing the Paste List button.

## LN(number)

Returns the natural logarithm of *number*. For more information, see LN in Chapter 2, “Worksheet Function Directory.”

## LOG(number,base)

Returns the logarithm of *number* in base *base*.

## LOG10(number)

Returns the base 10 logarithm of *number*. For more information, see LOG10 in Chapter 2, “Worksheet Function Directory.”

## LOGEST(known\_y's,known\_x's)

Returns the parameters of an exponential trend. For more information, see LOGEST in Chapter 2, “Worksheet Function Directory.”

## LOOKUP(lookup\_value,lookup\_vector,result\_vector)

## LOOKUP(lookup\_value,array)

Returns a value in a table selected by *lookup\_value*. For more information, see LOOKUP in Chapter 2, “Worksheet Function Directory.”

## LOWER(text)

Converts *text* to lowercase. For more information, see LOWER in Chapter 2, “Worksheet Function Directory.”

## MAIN.CHART(type,stack,100,vary,overlap,drop, hilo,overlap%,cluster,angle)

Equivalent to the Format Main Chart command.

Type	Chart
1	Area
2	Bar
3	Column
4	Line
5	Pie
6	Scatter

The next six arguments correspond to check boxes. If an argument is TRUE, its corresponding check box is turned on; if FALSE, its check box is turned off.

Argument	Check box
<i>stack</i>	Stacked
<i>100</i>	100%
<i>vary</i>	Vary by Categories
<i>overlap</i>	Overlapped
<i>drop</i>	Drop Lines
<i>hilo</i>	Hi-Lo Lines

The last three arguments are numbers:

Argument	Description
<i>overlap%</i>	% Overlap
<i>cluster</i>	% Cluster spacing
<i>angle</i>	Angle of first pie slice (degrees)

Not all of the arguments apply to every type of chart. Arguments that don't apply to the type of chart specified by *type* are ignored. The following table summarizes which arguments apply to which type of chart:

	Area	Bar	Column	Line	Pie	Scatter
Stack	■	■	■	■		
100	■	■	■	■		
Vary		■	■	■	■	■
Overlap		■	■			
Drop	■			■		
Hi-lo				■		
Overlap%		■	■			
Cluster		■	■			
Angle					■	

## MAIN.CHART.TYPE(type)

**Note** | This macro function is included for compatibility with macros written with Microsoft Excel for the Apple Macintosh.

Equivalent to MAIN.CHART(*type*).

## MATCH(lookup\_value,lookup\_array,type\_of\_match)

Returns the number of a value selected by *lookup\_value*. For more information, see MATCH in Chapter 2, “Worksheet Function Directory.”

## MAX(number1,number2,...)

Returns the maximum number in *numbers*. For more information, see MAX in Chapter 2, “Worksheet Function Directory.”

## MDETERM(array)

Returns the determinant of *array*. For more information, see MDETERM in Chapter 2, “Worksheet Function Directory.”

## MESSAGE(logical,text)

Displays and removes messages in the message area of the status bar. (For information on the status bar, see Status Bar in *Microsoft Excel Reference*.) MESSAGE is useful for displaying text that doesn’t need a response.

If *logical* is TRUE, Microsoft Excel displays *text* in the message area of the status bar. If *text* is “” (the empty text), any messages currently displayed in

the status bar are removed. If *logical* is FALSE, any messages are removed and the status bar is returned to normal (that is, command help messages are displayed).

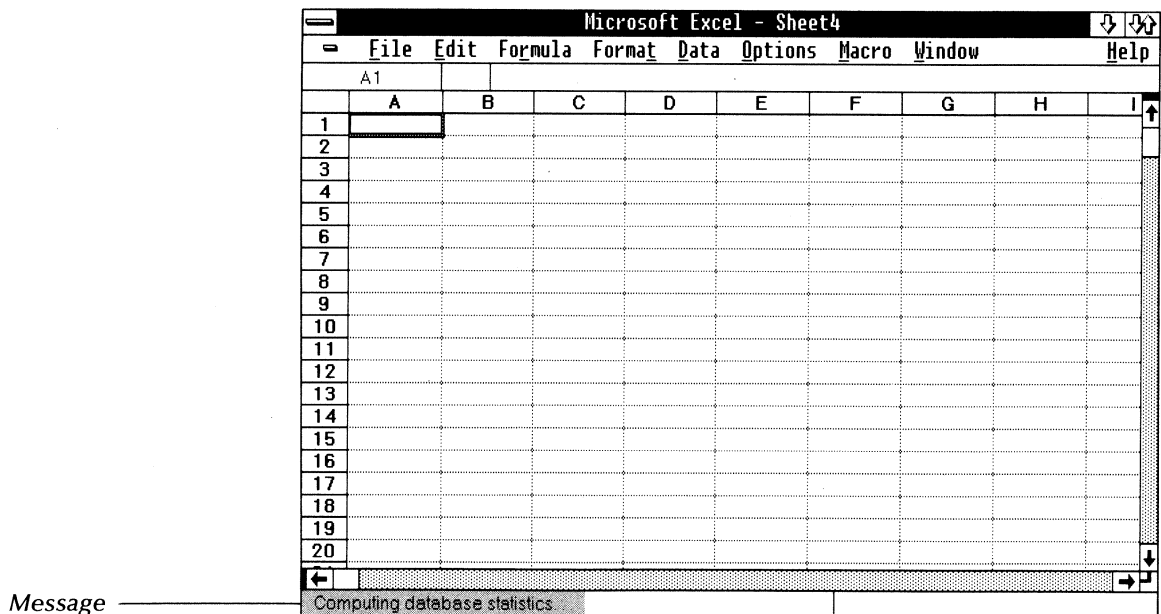
Only one message can be displayed in the status bar at a time. Messages are always displayed in the same place. MESSAGE works the same way whether the status bar is displayed or not. You can, for example, use MESSAGE while the status bar isn't displayed. As soon as you display the status bar, you see your message.

If you display any message (even the empty text) and don't remove it, that message is displayed until you quit Microsoft Excel.

### Example

The following function displays the message shown below:

MESSAGE(TRUE,"Computing database statistics. . .")



### MID(text,start\_number,number\_of\_characters)

Extracts *number\_of\_characters* from *text*. For more information, see MID in Chapter 2, "Worksheet Function Directory."

## MIN(number1,number2,...)

Returns the minimum number in *numbers*. For more information, see MIN in Chapter 2, “Worksheet Function Directory.”

## MINUTE(serial\_number)

Converts *serial\_number* to a minute of the hour. For more information, see MINUTE in Chapter 2, “Worksheet Function Directory.”

## MINVERSE(array)

Returns the matrix inverse of *array*. For more information, see MINVERSE in Chapter 2, “Worksheet Function Directory.”

## MIRR(values,finance\_rate,reinvest\_rate)

Returns the modified internal rate of return of *values*. For more information, see MIRR in Chapter 2, “Worksheet Function Directory.”

## MMULT(array1,array2)

Returns the matrix product of two arrays. For more information, see MMULT in Chapter 2, “Worksheet Function Directory.”

## MOD(number,divisor\_number)

Returns the remainder of *number* divided by *divisor\_number*. For more information, see MOD in Chapter 2, “Worksheet Function Directory.”

## MONTH(serial\_number)

Converts *serial\_number* to a month of the year. For more information, see MONTH in Chapter 2, “Worksheet Function Directory.”

## MOVE(x\_pos,y\_pos>window\_text)

Equivalent to the Move command on the Control menu. Moves the window named *window\_text* so that its upper-left corner is at the horizontal position *x\_pos* and the vertical position *y\_pos*. If *window\_text* is not specified, it is assumed to be the active window.

The *x\_pos* argument is measured in points from the left edge of your workspace to the left edge of the window.

The *y\_pos* argument is measured in points from the top edge of your workspace to the top edge of the window.

MOVE does not change the size of the window or affect whether the specified window is active or inactive.

### Example

MOVE(1,1) places the active window against the top-left edge of the workspace.

## N(value)

Returns *value* translated into a number. For more information, see N in Chapter 2, “Worksheet Function Directory.”

## NA()

Returns the #N/A error value. For more information, see NA in Chapter 2, “Worksheet Function Directory.”

## NAMES(*doc\_text*)

Returns a horizontal text array of all the names that are defined on the document *doc\_text*. If *doc\_text* is omitted, it is assumed to be the active document.

## NEW(type\_number)

## NEW?(type\_number)

Equivalent to the File New command.

*Type\_number* specifies the type of file to open, as follows:

Type_number	Document
1	Worksheet
2	Chart
3	Macro sheet



## NEW.WINDOW()

Equivalent to the Window New Window command.

## NEXT()

Ends a FOR-NEXT or WHILE-NEXT loop. For information on loops, see FOR, WHILE, or “Looping” in Chapter 3, “Macro Basics.”

## NOT(logical)

Returns TRUE if *logical* is FALSE; returns FALSE if *logical* is TRUE. For more information, see NOT in Chapter 2, “Worksheet Function Directory.”

## NOTE(add\_text,cell\_ref,start\_char,count\_char)

Replaces *count\_char* characters, starting at *start\_char*, in the note attached to cell *cell\_ref*, with *add\_text*.

If *add\_text* is omitted, it is assumed to be “”, the empty text. If *cell\_ref* is omitted, it is assumed to be the active cell. If *start\_char* is omitted, it is assumed to be 1. If *count\_char* is omitted, it is assumed to be equal to the length of the note attached to cell *cell\_ref*.

*Add\_text* must be less than or equal to 255 characters long.

### Examples

NOTE() deletes the note attached to the active cell.

NOTE(“Prelude and Happy Dance, by Kazdin”,!A1) attaches a new note, “Prelude and Happy Dance, by Kazdin”, to cell A1 on the active document.

NOTE(“and away we go”,!A1,1000) inserts the text “and away we go” in the note attached to cell A1 at position 1000, or at the end of the note if the note has less than 1000 characters.

## NOW()

Returns the *serial\_number* of current date and time. For more information, see NOW in Chapter 2, “Worksheet Function Directory.”

## NPER(rate,pmt,pv,fv,type)

Returns the number of payments of investment. For more information, see NPER in Chapter 2, “Worksheet Function Directory.”

## NPV(rate,value1,value2,...)

Returns the net present value of *values*. For more information, see NPV in Chapter 2, “Worksheet Function Directory.”

## OFFSET(ref,rows,cols,height,width)

Returns a reference *height* rows high and *width* columns wide, offset from *ref*. *Ref*’s upper-left corner is shifted down by *rows* rows and right by *cols* columns. If *rows* or *cols* is negative, OFFSET returns a reference above or to the left of *ref*, respectively.

If *height* or *width* is omitted, it is assumed to be the same height or width as *ref*.

If *rows* and *cols* offset *ref* over the edge of the worksheet, OFFSET returns the #REF! error value. If *ref* is a multiple reference, it returns the #VALUE! error value.

Note | OFFSET doesn’t actually move any cells, it just returns a reference.

### Examples

OFFSET(C3,2,3,1,1) equals F5

OFFSET(C3:E5, -1,0,3,3) equals C2:E4

OFFSET(C3:E5,0, -3,3,3) equals #REF!

## ON.DATA(document\_text,macro\_text)

Starts the macro specified by *macro\_text* whenever another application sends new data to the document specified by *document\_text*. *Document\_text* must contain one or more remote references. *Macro\_text* must be an R1C1-style reference in the form of text. If the incoming data causes recalculation, Microsoft Excel first performs the recalculation and then executes the macro *macro\_text*.

To turn off ON.DATA, omit the *macro\_text* argument.

Note | ON.DATA remains in effect until either you turn it off or you quit Microsoft Excel. Make sure you don’t close the macro sheet containing *macro\_text*, or you’ll get an error if data is sent to *document\_text*.

## ON.KEY(key\_text,macro\_text)

Tells Microsoft Excel to run the macro specified by *macro\_text* whenever the key specified by *key\_text* is pressed. If *macro\_text* is a reference in the form of text, the macro is executed whenever *key\_text* is pressed. If *macro\_text* is "" (the empty text) nothing happens when *key\_text* is pressed. If *macro\_text* is omitted, *key\_text* reverts to its normal meaning in Microsoft Excel.

*Key\_text* can specify any single key, or any key combined with SHIFT and/or CONTROL and/or ALT. The Appendix explains how to specify *key\_text*.

### Note

ON.KEY remains in effect until either you turn it off or you quit Microsoft Excel. Make sure you don't close the macro sheet containing *macro\_text*, or you'll get an error if you press *key\_text*.

### Example

Suppose you wanted the key combination SHIFT+CONTROL+RIGHT to run the macro Print on the macro sheet PAYROLL.XLM. You would use the following formula:

```
=ON.KEY("+^{RIGHT}","PAYROLL.XLM!Print")
```

To return SHIFT+CONTROL+RIGHT to its normal meaning, you would use the following formula:

```
=ON.KEY("+^{RIGHT}")
```

To disable SHIFT+CONTROL+RIGHT altogether, you would use the following formula:

```
=ON.KEY("+^{RIGHT}","")
```

## ON.TIME(time,macro\_text,tolerance,insert\_log)

Tells Microsoft Excel to run a macro at a specified time.

If *insert\_log* is TRUE or omitted, at time *time* the macro specified by *macro\_text* is executed. *Macro\_text* is an R1C1-style reference to a macro in the form of text. If *insert\_log* is FALSE, any prior requests to execute *macro\_text* at the time *time* are ignored.

If *time* only specifies a time (that is, it is a serial number that is less than 1), the macro *macro\_text* runs every day at that time.

If	Then
At time <i>time</i> the sheet containing <i>macro_text</i> is not in memory	The request is ignored
At time <i>time</i> Microsoft Excel is not in READY, COPY, CUT, or FIND mode	Microsoft Excel waits for the length of time specified by <i>tolerance</i> (a serial number). If <i>tolerance</i> is omitted, it is assumed to be infinite. If Microsoft Excel does not return to one of the listed modes before <i>tolerance</i> length of time, the request is canceled
Two identical ON.TIME statements are issued	The first is executed; the others are ignored and return #N/A!

If *tolerance* is omitted, it is assumed to be the maximum possible serial time.

### ON.WINDOW(*window\_text*,*macro\_text*)

Starts the macro specified by *macro\_text* whenever the *window\_text* window is activated. *Macro\_text* must be a reference to a macro in the form of text and *window\_text* must be the name of a window in the form of text.

If *window\_text* is omitted, ON.WINDOW starts the macro whenever any window is activated, except for windows that are named in other ON.WINDOW statements. If *macro\_text* is omitted, activating *window\_text* no longer starts a macro.

### OPEN(*file\_text*,*update\_ext*,*read\_only\_rem*)

### OPEN?(*file\_text*,*update\_links*,*read\_only*)

Equivalent to the File Open command. Opens the document specified by *file\_text*. *File\_text* can include a drive and pathname.

The following list summarizes the other arguments. If an argument corresponding to a check box is TRUE, the check box is turned on; if the argument is FALSE, the check box is turned off.

Argument	Description
<i>update_links</i>	0 = Update neither external nor remote references 1 = Update external references only 2 = Update remote references only 3 = Update both
<i>read_only</i>	Corresponds to the Read Only check box

In the dialog box form, OPEN?, *file\_text* can include an asterisk (\*) to represent any sequence of characters and a question mark (?) to represent any single character.

**OPEN.LINKS**(*doc\_text1,doc\_text2,...,read\_only\_log*)

**OPEN.LINKS?**(*doc\_text1,doc\_text2,...,read\_only\_log*)

Equivalent to the File Links command (Full menus). OPEN.LINKS can have from 1 to 14 arguments. The *doc\_text* arguments should be the names of documents in the form of text, or arrays or references that contain text.

If the last argument to OPEN.LINKS or OPEN.LINKS? is a logical value, that logical value corresponds to the Read Only check box in the File Links dialog box. If *read\_only\_log* is TRUE, the check box is turned on; if FALSE, the check box is turned off.

If one of the specified documents contains external references to a supporting worksheet, OPEN.LINKS assumes that you want to update the external and remote references.

You can generate an array of the names of linked documents with the LINKS function.

### Example

If a chart named PERFORM.XLC is open and contains data series that refer to worksheets named BUDGET.XLS and ACTUAL.XLS, Microsoft Excel opens BUDGET.XLS and ACTUAL.XLS when this formula is calculated:

=OPEN.LINKS(LINKS())

## OR(logical1,logical2,...)

Returns TRUE if any argument is TRUE; otherwise, FALSE. For more information, see OR in Chapter 2, “Worksheet Function Directory.”

## OVERLAY(type,stack,100,vary,overlap,drop,hilo,overlap%,cluster,angle,series,auto)

Equivalent to the Format Overlay command (Full menus).

*Type* specifies the type of chart, as follows:

Type	Chart
1	Area
2	Bar
3	Column
4	Line
5	Pie
6	Scatter

The next six arguments correspond to check boxes. If an argument is TRUE, its corresponding check box is turned on; if FALSE, its check box is turned off.

Argument	Check box
<i>stack</i>	Stacked
<i>100</i>	100%
<i>vary</i>	Vary by Categories
<i>overlap</i>	Overlapped
<i>drop</i>	Drop Lines
<i>hilo</i>	Hi-Lo Lines

The next three arguments are numbers:

Argument	Description
<i>overlap%</i>	% Overlap
<i>cluster</i>	% Cluster spacing
<i>angle</i>	Angle of first pie slice (degrees)

*Series* is the number of the first series in the overlay chart.

*Auto* corresponds to the Automatic Series Distribution check box.

Not all of the arguments apply to every type of chart. Arguments that don't apply to the type of chart specified by *type* are ignored. The following table summarizes which arguments apply to which type of chart:

	Area	Bar	Column	Line	Pie	Scatter
Stack	■	■	■	■		
100	■	■	■	■		
Vary		■	■	■	■	■
Overlap		■	■			
Drop	■			■		
Hi-lo				■		
Overlap%		■	■			
Cluster		■	■			
Angle					■	
Series	■	■	■	■	■	■
Auto	■	■	■	■	■	■

OVERLAY.CHART.TYPE(*type*)

Note | This macro function is included for compatibility with macros written with Microsoft Excel for the Apple Macintosh.

Equivalent to OVERLAY(*type* – 1). If *type* is 0, equivalent to DELETE.OVERLAY.

PAGE.SETUP(*head,foot,left,right,top,bot,heading,grid*)

PAGE.SETUP?(*head,foot,left,right,top,bot,heading,grid*)

PAGE.SETUP(*head,foot,left,right,top,bot,size*)

PAGE.SETUP?(*head,foot,left,right,top,bot,size*)

Equivalent to the File Page Setup command. The first form applies if the active document is a worksheet or macro sheet. The second form applies if the active document is a chart.

The following list summarizes the arguments to the worksheet or macro sheet form. The logical arguments correspond to check boxes. If an argument is TRUE, its corresponding check box is turned on; if FALSE, its check box is turned off.

Argument	Description
<i>head</i>	Header, as text
<i>foot</i>	Footer, as text
<i>left</i>	Left margin
<i>right</i>	Right margin
<i>top</i>	Top margin
<i>bot</i>	Bottom margin
<i>heading</i>	Corresponds to Row & Column Headings check box
<i>grid</i>	Corresponds to Gridlines check box

The following list summarizes the arguments to the chart form. The logical arguments correspond to check boxes. If an argument is TRUE, its check box is turned on; if FALSE, its check box is turned off.

Argument	Description
<i>head</i>	Header, as text
<i>foot</i>	Footer, as text
<i>left</i>	Left margin



Argument	Description
<i>right</i>	Right margin
<i>top</i>	Top margin
<i>bot</i>	Bottom margin
<i>size</i>	1 = Screen Size 2 = Fit to Page 3 = Full Page

---

## PARSE(parse\_text)

Equivalent to the Data Parse command (Full menus). *Parse\_text* is the parse line, in the form of text. In most cases, it's easier to record the Data Parse command. For more information, see Data Parse command in *Microsoft Excel Reference*.

## PASTE()

Equivalent to the Edit Paste command.

## PASTE.LINK()

Equivalent to the Edit Paste Link command (Full menus).

## PASTE.SPECIAL(paste\_what,operation,skip\_blanks,transpose)

## PASTE.SPECIAL?(paste\_what,operation,skip\_blanks,transpose)

Equivalent to the Edit Paste Special command (Full menus). The PASTE.SPECIAL macro function has three forms. This form applies if you are pasting into a worksheet or macro sheet. The forms for pasting into charts are described in the following sections.

*Paste\_what* specifies what to paste, as follows:

Paste_what	Paste
1	All
2	Formulas
3	Values
4	Formats
5	Notes

*Operation* specifies what operation to perform when pasting:

Operation	Action
1	None
2	Add
3	Subtract
4	Multiply
5	Divide

*Skip\_blanks* corresponds to the Skip Blanks check box and *transpose* corresponds to the Transpose check box. If the argument corresponding to a check box is TRUE, the check box is turned on. If it's FALSE, the check box is turned off.

### PASTE.SPECIAL(*row\_col,series,categories,apply*)

### PASTE.SPECIAL?(*row\_col,series,categories,apply*)

Equivalent to the Edit Paste Special command (Full menus). The PASTE.SPECIAL macro function has three forms. This form applies if you have copied from a worksheet and are pasting into a chart. The form for pasting into worksheets and macro sheets is described in the previous section. The form for copying from a chart and pasting into a chart is described in the following section.

*Row\_col* specifies whether the values are in rows or columns:

Row_col	Values
1	Rows
2	Columns

The other three arguments correspond to check boxes. If the argument corresponding to a check box is TRUE, the check box is turned on. If it's FALSE, the check box is turned off.

Argument	Check box
<i>series</i>	Series Names in First Row
<i>categories</i>	Categories in First Column
<i>apply</i>	Apply Categories to All Series

## PASTE.SPECIAL(paste\_what)

## PASTE.SPECIAL?(paste\_what)

Equivalent to the Edit Paste Special command (Full menus). The PASTE.SPECIAL macro function has three forms. This form applies if you have copied from a worksheet or macro sheet and are pasting into a chart. The forms for pasting into worksheets and macro sheets and for copying from a worksheet or macro sheet and pasting into a chart are described in the previous sections.

*Paste\_what* specifies what to paste, as follows:

Paste_what	Values
1	All
2	Formats
3	Formulas

**PATTERNS**(b\_auto,b\_style,b\_color,b\_wt,shadow,  
a\_auto,a\_pattern,a\_fore,a\_back,APPLY)

**PATTERNS**(LINE,t\_major,t\_minor,t\_label)

**PATTERNS**(LINE)

**PATTERNS**(LINE,m\_auto,m\_style,m\_fore,m\_back,APPLY)

**PATTERNS**(LINE,h\_width,h\_length,h\_type)

The **PATTERNS** macro function is equivalent to the Format Patterns command. It has five forms, depending on what is selected. Certain arguments and groups of arguments, shown in all capital letters in the syntaxes below, appear in more than one form of the **PATTERNS** macro function. They are described here.

The *APPLY* Placeholder:

Arguments	Description
<i>apply</i>	Corresponds to the Apply to All check box

The *LINE* Placeholder:

Arguments	Description
<i>L_auto</i>	Automatic line settings: 0 = set by user 1 = automatic 2 = invisible
<i>L_style</i>	A number from 1–5, corresponding to the 5 line styles in the Format Patterns dialog box
<i>L_color</i>	A number from 1–8, corresponding to the 8 line colors in the Format Patterns dialog box
<i>L_wt</i>	A number from 1–3, corresponding to the 3 line weights in the Format Patterns dialog box

If the current selection is a chart, plot area, legend, text label, area, or bar:

PATTERNS(*b\_auto*,*b\_style*,*b\_color*,*b\_wt*,*shadow*,*a\_auto*,*a\_pattern*,  
*a\_fore*,*a\_back*,*APPLY*)

Arguments	Description
<i>b_auto</i>	Automatic border settings: 0 = set by user 1 = automatic 2 = invisible
<i>b_style</i>	A number from 1–5, corresponding to the 5 border styles in the Format Patterns dialog box
<i>b_color</i>	A number from 1–8, corresponding to the 8 border colors in the Format Patterns dialog box
<i>b_wt</i>	A number from 1–3, corresponding to the 3 border weights in the Format Patterns dialog box
<i>shadow</i>	Corresponds to the Shadow check box. TRUE if turned on, FALSE if turned off. Does not apply to areas in area charts or bars in bar charts.
<i>a_auto</i>	Automatic area settings: 0 = set by user 1 = automatic 2 = invisible
<i>a_pattern</i>	A number from 1–16, corresponding to the 16 area patterns in the Format Patterns dialog box
<i>a_fore</i>	A number from 1–8, corresponding to the 8 area foreground colors in the Format Patterns dialog box
<i>a_back</i>	A number from 1–8, corresponding to the 8 area background colors in the Format Patterns dialog box
<i>APPLY</i>	See description above

If the current selection is an axis:

PATTERNS(LINE,t\_major,t\_minor,t\_label)

Arguments	Description
<i>LINE</i>	See description above
<i>t_major</i>	Describes the type of major tick marks: 1 = Invisible 2 = Inside 3 = Outside 4 = Cross
<i>t_minor</i>	Describes the type of minor tick marks: 1 = Invisible 2 = Inside 3 = Outside 4 = Cross
<i>t_label</i>	Describes the position of tick labels: 1 = None 2 = Low 3 = High 4 = Next to axis

If the current selection is a gridline, hi-lo line, drop line:

PATTERNS(LINE)

Arguments	Description
<i>LINE</i>	See description above

If the current selection is a data line:

PATTERNS(LINE,m\_auto,m\_style,m\_fore,m\_back,APPLY)

Arguments	Description
<i>LINE</i>	See description above
<i>m_auto</i>	Automatic marker settings: 0 = set by user 1 = automatic 2 = invisible
<i>m_style</i>	A number from 1–7, corresponding to the 7 marker styles in the Format Patterns dialog box
<i>m_fore</i>	A number from 1–8, corresponding to the 8 foreground colors in the Format Patterns dialog box
<i>m_back</i>	A number from 1–8, corresponding to the 8 background colors in the Format Patterns dialog box
<i>APPLY</i>	See description above

If the current selection is an arrow:

PATTERNS(LINE,h\_width,h\_length,h\_type)

Arguments	Description
<i>LINE</i>	See description above. Arguments describe the shaft of the arrow
<i>h_width</i>	Describes the width of the arrowhead: 1 = narrow 2 = medium 3 = wide
<i>h_length</i>	Describes the length of the arrowhead: 1 = short 2 = medium 3 = long
<i>h_type</i>	Describes the type of the arrowhead: 1 = no head 2 = open head 3 = closed head

## PI()

Returns the value of  $\pi$ . For more information, see PI in Chapter 2, “Worksheet Function Directory.”

## PMT(rate,nper,pv,fv,type)

Returns the periodic payment of investment. For more information, see PMT in Chapter 2, “Worksheet Function Directory.”

## POKE(channel\_num,item\_text,data\_ref)

**Note** | This function is supported only if you have Microsoft Windows (version 2.0).

Sends the data *data\_ref* to the item specified by *item\_text* in the application connected to channel *channel\_num*. Channel *channel\_num* must have been opened by the INITIATE function. *Data\_ref* is a Microsoft Excel reference to the document containing the data to send. The form of *item\_text* depends on the application connected to *channel\_num*.

If POKE is not successful, it returns the following values:

Situation	Return value
<i>channel_num</i> is not a valid channel number	#VALUE!
The application you are accessing does not respond after a certain length of time, and ESCAPE is pressed to cancel	#DIV/0!
POKE is refused	#REF!

For information on accessing other applications, see “Using Macros to Start Other Applications” in Chapter 6, “Advanced Macros.”

## PPMT(rate,per,nper,pv,fv,type)

Returns the payment on the principal for an investment. For more information, see PPMT in Chapter 2, “Worksheet Function Directory.”



## PRECISION(logical)

Equivalent to choosing the Options Calculation command and selecting the Precision as Displayed check box. If *logical* is TRUE, the check box is turned off. If *logical* is FALSE, the check box is turned on.

## PREFERRED()

Equivalent to the Gallery Preferred command (Full menus).

## PRINT(range,from,to,copies,draft,preview,parts)

## PRINT?(range,from,to,copies,draft,preview,parts)

Equivalent to the File Print command. The following list summarizes the arguments. The logical arguments each correspond to a check box. If the argument is TRUE, the check box is turned on. If the argument is FALSE, the check box is turned off.

Argument	Description
<i>range</i>	Specifies page range. Either 1 (print all) or 2 (print specified range)
<i>from</i>	First page of range (this argument is ignored unless <i>range</i> equals 2)
<i>to</i>	Last page of range (this argument is ignored unless <i>range</i> equals 2)
<i>copies</i>	Number of copies
<i>draft</i>	Corresponds to Draft Quality check box
<i>preview</i>	Corresponds to Preview check box
<i>parts</i>	Specifies what to print 1 = Sheet 2 = Notes 3 = Both  Only applies when printing worksheets or macro sheets

## PRINTER.SETUP(*printer\_text*)

## PRINTER.SETUP?(*printer\_text*)

Equivalent to the File Printer Setup command.

*Printer\_text* is the name of the printer to be activated. Enter *printer\_text* exactly as it appears in the File Printer Setup dialog box.

### Example

PRINTER.SETUP("HP Laserjet+ on COM1:") changes the printer to an HP Laserjet+ attached to COM1:.

## PRODUCT(*number1,number2,...*)

Returns the product of *numbers*. For more information, see PRODUCT in Chapter 2, "Worksheet Function Directory."

## PROPER(*text*)

Converts *text* to initial capitals. For more information, see PROPER in Chapter 2, "Worksheet Function Directory."

## PROTECT.DOCUMENT(*contents, windows*)

## PROTECT.DOCUMENT?(*contents, windows*)

Equivalent to the Options Protect Document and Options Unprotect Document commands (Full menus) if a worksheet or macro sheet is the active document. Equivalent to the Chart Protect Document and Chart Unprotect Document commands (Full menus) if a chart is the active document.

If both arguments are FALSE, PROTECT.DOCUMENT carries out the Unprotect Document command.

If one or both of the arguments are TRUE, it carries out the command. The arguments correspond to the check boxes with the same names. If the argument is TRUE, the check box is turned on. If it's FALSE, the check box is turned off. If *contents* is omitted, it is assumed to be TRUE. If *windows* is omitted, it is assumed to be FALSE.

## PV(rate,nper,pmt,fv,type)

Returns the present value of investment. For more information, see PV in Chapter 2, “Worksheet Function Directory.”

## QUIT()

Quits Microsoft Excel. If open documents have unsaved changes, displays a message asking if you want to save them.

## RAND()

Returns a random number between 0 and 1. For more information, see RAND in Chapter 2, “Worksheet Function Directory.”

## RATE(nper,pmt,pv,fv,type,guess)

Returns the rate returned on investment. For more information, see RATE in Chapter 2, “Worksheet Function Directory.”

## REFTEXT(ref,a1)

Converts the reference *ref* to an absolute reference in the form of text. If *a1* is TRUE, returns an A1-style reference. If *a1* is FALSE or omitted, returns an R1C1-style reference.

## REGISTER(module\_text,procedure\_text,argument\_text)

### Important

This is a powerful but dangerous function provided for expert programmers only. If you use the REGISTER function incorrectly, you could accidentally cause errors in your systems operation.

Returns a text value to be used by the CALL function. The CALL function can then be used to access the Microsoft Windows dynamic library.

The arguments are:

Argument	Description
<i>module_text</i>	The name of the Microsoft Windows dynamic library that contains the procedure you want
<i>procedure_text</i>	The name of the procedure you want to start
<i>argument_text</i>	Specifies the number and data types of the arguments to the procedure, and the data type of the return value of the procedure

The letters A–K are defined as codes for the different data types. To form *argument\_text*, concatenate the code for the data type of the return value with the codes for the data type of each argument.

The following table lists the codes and describes how arguments and return values of each data type are sent and returned:

Code	Data Type	Argument Pushed	Value Returned
A	Boolean	If FALSE, integer 0; if TRUE, integer 1	If AX = 0, FALSE, otherwise TRUE
B	IEEE Float	Four words	DX:AX points to IEEE floating point number
C	Zero terminated ANSI string	Far pointer to zero terminated string	DX:AX points to zero terminated string
D	Byte count, ANSI string	Far pointer to count string	DX:AX points at byte count, followed by string
E	IEEE float buffer	Far pointer to a 4-word number buffer	IEEE float contents of the number buffer
F	String buffer	Far pointer to a 256-byte string buffer	Zero terminated contents of the string buffer
G	String buffer	Far pointer to a 256-byte string buffer	Count string contents of the string buffer
H	Unsigned integer	Word	AX contains the integer
I	Signed integer	Word	AX contains the integer
J	Unsigned long	2 words	DX:AX contains the integer
K	Floating-point array	Far pointer to array structure. First word contains number of rows. Next word contains number of columns. Followed by row times columns IEEE floating point numbers.	DX:AX contains pointer to array structure

For example, the *argument\_text* value “ABBI” means the procedure returns a boolean value (A), and takes three arguments. The first two arguments are IEEE floating point numbers (BB), and the third argument is a signed integer (I). Four words are pushed for each of the first two arguments; one word is pushed for the third argument. The return value is FALSE if AX = 0 and TRUE if AX does not equal 0.

### Example

Suppose you wanted to use the Microsoft Windows GetSysColor procedure to retrieve the background color. GetSysColor is in the USER library, takes one unsigned integer argument (an index specifying which system color to retrieve), and returns a long. First you would need to use the REGISTER function.

=REGISTER("USER","GetSysColor","JH")

This would return a text value which you would use in the CALL function to access USER.

### RELREF(ref,rel\_to\_ref)

Returns the reference of *ref*, relative to the upper-left corner cell of *rel\_to\_ref*. The reference is given as an R1C1-style relative reference in the form of text, such as "R[1]C[1]".

#### Examples

RELREF(A1,C3) equals "R[-2]C[-2]"

RELREF(Finance!A1,Finance!C3) equals "R[-2]C[-2]"

RELREF(A1:E5,C3:G7) equals RELREF(A1:E5,C3)  
equals "R[-2]C[-2]:R[2]C[2]"

### REMOVE.PAGE.BREAK()

Equivalent to the Options Remove Page Break command (Full menus). If the active cell is not below or to the right of a manual page break, REMOVE.PAGE.BREAK takes no action.

### RENAME.COMMAND(bar\_num,menu\_pos,command\_pos,name\_text)

Gives the name *name\_text* to the command in position *command\_pos* on the menu *menu\_pos* in the menu bar number *bar\_num*. *Menu\_pos* can either be the number of a menu or the name of a menu as text. *Command\_pos* can be either the number of the command to be renamed (the first command on a menu is command 1) or the title of the command as text. If *command\_pos* is 0, the menu is renamed.

*Bar\_num* can either be the number of one of the Microsoft Excel built-in menu bars or the number returned by a previously executed ADD.BAR function.

If the specified command does not exist, RENAME.COMMAND returns the #VALUE! error value. When Microsoft Excel renames built-in commands, such as shifted commands, it may rename built-in commands you have renamed.

For information on custom menus, see "Creating Customized Menus and Dialog Boxes" in Chapter 6, "Advanced Macros."

**REPLACE**(old\_text,start\_num,num\_chars,new\_text)

Replaces *num\_chars* characters in *old\_text* with *new\_text*. For more information, see REPLACE in Chapter 2, “Worksheet Function Directory.”

**REPLACE.FONT**(font,name\_text,size\_num,bold,italic,underline,strike)

Equivalent to choosing the Format Font command, selecting the font numbered *font*, choosing the Font button, specifying a new font, and choosing the Replace button. *Font* must be a number from 1–4.

The rest of the arguments describe the new font, as follows:

Argument	Description
<i>name_text</i>	The name of the new font, as it appears in the dialog box. For example, “Courier” and “TmsRmn” are names of fonts
<i>size_num</i>	The size of the new font, in points
<i>bold</i>	TRUE if the new font is bold, FALSE otherwise
<i>italic</i>	TRUE if the new font is italic, FALSE otherwise
<i>underline</i>	TRUE if the new font is underlined, FALSE otherwise
<i>strike</i>	TRUE to draw a line through (strikeout) the new font, FALSE otherwise

**REPT**(text,number\_times)

Repeats the *text* argument *number\_times* times. For more information, see REPT in Chapter 2, “Worksheet Function Directory.”

**REQUEST**(channel\_num,item\_text)

**Note** This function is supported only if you have Microsoft Windows (version 2.0).

Requests the information specified by *item\_text* from the application connected to the channel specified by *channel\_num*. Channel *channel\_num* must have been opened by the INITIATE function. The form of *item\_text* depends on the application connected to *channel\_num*.

For information on accessing other applications, see “Using Macros to Call Other Applications” in Chapter 6, “Advanced Macros.”

REQUEST returns the data as an array. For example, suppose the remote data to be returned came from a worksheet that looked like this:

	A	B	C	D	E	F	G	H
1	1	2	3					
2	4	5	6					
3								

REQUEST would return that data as the following array:

{1,2,3;4,5,6}

If REQUEST is not successful, it returns the following error values:

Situation	Return value
<i>channel_num</i> is not a valid channel number	#VALUE!
The application you are accessing is busy doing something else	#N/A
The application you are accessing does not respond after a certain length of time, and ESCAPE is pressed to cancel	#DIV/0!
The request is refused	#REF!

## RESTART(*level\_number*)

Removes *level\_number* return addresses from the stack. If *level\_number* is omitted, removes all return addresses from the stack. This means that as soon as a RETURN function is encountered, the macro will stop running instead of returning control to the macro that started it.



## RESULT(*type\_number*)

Specifies the data type of a function macro's return value, as follows:

Type_number	Data type
1	Number
2	Text
4	Logical
8	Reference
16	Error
64	Array

For numbers, text, logical values, and error values, *number* can be the sum of the different numbers above, in which case the value returned can be of different types.

If you omit *type\_number*, it is assumed to be 7, which means that the value returned can be either a number (1), text (2), or logical value (4).

For more information on using the RESULT function, see “Returning Results” in Chapter 4, “Writing Macros.”

## RETURN(*value*)

Tells Microsoft Excel to stop the currently running macro, and return control to whatever started the macro. This may be the user, if the macro was started with the Macro Run command or a shortcut key; it may be a formula if the macro is a function macro, or it may be another macro.

If the macro is a function macro, the *value* argument specifies the return value of the macro. If the macro is a command macro run by the user, the *value* argument cannot be included.

For more information, see “Returning Values” in Chapter 4, “Writing Macros.”

## RIGHT(*text,number\_of\_chars*)

Returns the last *number\_of\_chars* characters in *text*. For more information, see RIGHT in Chapter 2, “Worksheet Function Directory.”

## ROUND(*number,number\_of\_digits*)

Rounds *number* to *number\_of\_digits* digits. For more information, see ROUND in Chapter 2, “Worksheet Function Directory.”

## ROW(*reference*)

Returns the row numbers in *reference*. For more information, see ROW in Chapter 2, “Worksheet Function Directory.”

## ROW.HEIGHT(*height\_num,ref,standard\_height*)

## ROW.HEIGHT?(*height\_num,ref,standard\_height*)

Equivalent to the Format Row Height command. The rows contained in *ref* are changed to the height *height\_num* points. If *standard\_height* is TRUE, row height is determined by the height of fonts used in each row just as if the Standard Height check box in the Format Row Height dialog box was turned on.

If *ref* is omitted, it is assumed to be the reference of the current selection. If *ref* is specified, it must be either an external reference to the active worksheet, such as !\$1:\$3 or !Database, or an R1C1-style reference in the form of text, such as “R1:R3”, “R[−4]:R[−2]”, or “Database.” If *ref* is a relative R1C1-style reference in the form of text, it is assumed to be relative to the active cell.

If you are recording a macro while using a mouse, and you change row height by dragging the row border, Microsoft Excel records the reference of the rows using R1C1-style references in the form of text. If the Macro Relative Record command (Full menus) is used, Microsoft Excel uses R1C1-style relative references. If the Macro Absolute Record command (Full menus) is used, Microsoft Excel uses R1C1-style absolute references.

## ROWS(array)

Returns the number of rows in *array*. For more information, see ROWS in Chapter 2, “Worksheet Function Directory.”

## RUN(reference)

## RUN?(reference)

Equivalent to the Macro Run command.

*Reference* should be either an external reference to a macro on a macro sheet, such as MACROS.XLM!\$A\$1 or MACROS.XLM!Months, or an R1C1-style external reference to a macro in the form of text, such as “MACROS.XLM!R1C1” or “MACROS.XLM!Months”.

If you are recording a macro, the reference you enter in the Macro Run dialog box is recorded as text, with A1-style references converted to R1C1-style references.

### Examples

Each of the following functions runs the macro beginning at the upper-left corner of the range named Months on the macro sheet named MACROS.XLM:

RUN(MACROS.XLM!Months)

RUN(“MACROS.XLM!Months”)

## SAVE()

Equivalent to the File Save command.

## SAVE.AS(name\_text,type\_num,passwd\_text,backup)

## SAVE.AS?(name\_text,type\_num,passwd\_text,backup)

Equivalent to the File Save As command.

*Name\_text* specifies the name of a document to save, such as “SALES.XLS”. You can also include a drive specifier and pathname in *name\_text*, such as “C:\EXCEL\ANALYZE.XLM”.

The other arguments specify options available in the File Save As dialog box. *Type\_num* specifies the type of document to save. For information on types of documents, see File Save As command in *Microsoft Excel Reference*. If the document is a chart, this argument does not apply; if the document is a macro sheet, only numbers 1, 2, 3, 6, and 9 apply.

Type_num	Type of document
1	Normal
2	SYLK
3	Text
4	WKS
5	WK1
6	CSV
7	DBF2
8	DBF3
9	DIF

*Passwd\_text* is a password.

*Backup* is TRUE to make a backup document, FALSE otherwise.

### SAVE.WORKSPACE(*name\_text*)

### SAVE.WORKSPACE?(*name\_text*)

Equivalent to the File Save Workspace command (Full menus).

*Name\_text* specifies the name of a workspace document to save, such as "SALES.XLW". You can also include a drive specifier and pathname in *name\_text*, such as "C:\EXCEL\TRACKING.XLW".

If *name\_text* is omitted, it is assumed to be "RESUME.XLW", or the name of the last workspace document, if any, opened in the current session.

## SCALE(*cross*,*cat\_labels*,*cat\_marks*,*between*,*max*,*reverse*)

Equivalent to the Format Scale command. There are two forms to this function. This form applies if the selected axis is a category axis and the chart is not a scatter chart. The other form is discussed in the following section.

*Cross* specifies the number of the category at which the value axis should cross.

*Cat\_labels* specifies the number of categories between tickmark labels.

*Cat\_marks* specifies the number of categories between tickmarks.

The last three arguments correspond to the three check boxes in the Format Scale dialog box. If an argument is TRUE, the check box is turned on; if FALSE, the check box is turned off.

Argument	Check box
<i>between</i>	Value Axis Crosses Between Categories
<i>reverse</i>	Categories in Reverse Order
<i>max</i>	Value Axis Crosses at Maximum Category

## SCALE(*min*,*max*,*major*,*minor*,*cross*,*logarithmic*,*reverse*,*max*)

Equivalent to the Format Scale command. There are two forms to this function. This form applies if the selected axis is a value axis or if the chart is a scatter chart. The other form is discussed in the previous section.

The first five arguments correspond to the five range variables in the Format Scale dialog box, as listed below. Each argument can be either the logical value TRUE or a number. If TRUE, the Automatic check box is turned on. If a number, that number is used for the variable.

Argument	Describes
<i>min</i>	Minimum
<i>max</i>	Maximum
<i>major</i>	Major unit

Argument	Describes
<i>minor</i>	Minor unit
<i>cross</i>	Category/value axis crosses

The last three logical arguments correspond to check boxes. If an argument is TRUE, the check box is turned on. If an argument is FALSE, the check box is turned off.

Argument	Check box
<i>logarithmic</i>	Logarithmic Scale
<i>reverse</i>	Values/Categories in reverse order
<i>max</i>	Category/Value Axis Crosses at Maximum Value/Category

### SEARCH(*find\_text,within\_text,start\_at\_num*)

Searches for *find\_text* within *within\_text*. For more information, see SEARCH in Chapter 2, “Worksheet Function Directory.”

### SECOND(*serial\_number*)

Converts *serial\_number* to seconds. For more information, see SECOND in Chapter 2, “Worksheet Function Directory.”

### SELECT(*selection,active\_cell*)

There are two forms of SELECT. This section discusses the form that applies if the selection is on a worksheet or macro sheet; the next section discusses the form that applies if the selection is on a chart.

The worksheet and macro sheet form is equivalent to the action of selecting cells or changing the active cell. It selects the cells specified by *selection* and makes the cell specified by *active\_cell* the active cell.

*Selection* should be either a reference to the active worksheet, such as !A1:A3 or !Sales, or an R1C1-style reference relative to the active cell in the current selection, in the form of text, such as “R[−1]C[−1]:R[1]C[1]”. If you omit *selection*, SELECT does not change the selection.

*Active\_cell* should be either a reference to a single cell on the active worksheet, such as !A1, or an R1C1-style reference relative to the active cell in the current selection, in the form of text, such as “R[−1]C[−1]”. *Active\_cell* must be inside *selection*. If you omit *active\_cell*, SELECT makes the upper-left corner cell in *selection* the active cell.

If you are recording a macro using the Macro Relative Record command (Full menus), and you make a selection, Microsoft Excel records the action using R1C1-style relative references in the form of text. If you are recording using the Macro Absolute Record command (Full menus), Microsoft Excel records the action using absolute references.

## Examples

The following function selects cells C3:E5 on the active worksheet and makes C5 the active cell:

```
SELECT(!C3:E5,!C5)
```

If the active cell is C3, the following function selects cells E5:G7 and makes cell F6 the active cell:

```
SELECT(“R[2]C[2]:R[4]C[4]”,“R[3]C[3]”)
```

The following sequence of functions moves the active cell to the right, to the left, up, and down within the selection, just as TAB, SHIFT+TAB, ENTER, and SHIFT+ENTER do.

```
SELECT(“RC[1]”)
```

```
SELECT(“RC[−1]”)
```

```
SELECT(“R[1]C”)
```

```
SELECT(“R[−1]C”)
```

## SELECT(item\_text)

There are two forms of SELECT. This section discusses the form that applies if the selection is on a chart; the previous section discusses the form that applies if the selection is on a worksheet or macro sheet.

Selects a chart object as specified by the selection code *item\_text*. The following list describes the selection codes.

To select	Use this text
Entire chart	"Chart"
Plot area	"Plot"
Legend	"Legend"
Main chart value axis	"Axis 1"
Main chart category axis	"Axis 2"
Overlay chart value axis	"Axis 3"
Overlay chart category axis	"Axis 4"
Chart title	"Title"
Label for the main chart value axis	"Text Axis 1"
Label for the main chart category axis	"Text Axis 2"
$n$ th floating text item	"Text $n$ "
$n$ th arrow	"Arrow $n$ "
Major gridlines of value axis	"Gridline 1"
Minor gridlines of value axis	"Gridline 2"
Major gridlines of category axis	"Gridline 3"
Minor gridlines of category axis	"Gridline 4"
Main chart droplines	"Dropline 1"
Overlay chart droplines	"Dropline 2"
Main chart hi-lo lines	"Hiloline 1"
Overlay chart hi-lo lines	"Hiloline 2"
Data associated with point $m$ in series $n$	" $SnPm$ "
Text attached to point $m$ of series $n$	"Text $SnPm$ "
Series title text of series $n$ of an area chart	"Text $Sn$ "



## Examples

SELECT("S1P3") selects the third point in series 1

SELECT("Dropline 2") selects the droplines of an overlay chart

SELECT("Chart") selects the entire chart

SELECT("Legend") selects a legend

SELECT("Text S1") selects the series title text of the first series in an area chart

## SELECT.CHART()

**Note** | This macro function is included for compatibility with macros written with Microsoft Excel for the Apple Macintosh.

Equivalent to the Chart Select Chart command. Same as the SELECT("Chart") function.

## SELECT.END(direction\_num)

Moves the active cell to the next block edge in the direction specified by *direction\_num*:

Direction_num	Direction
1	Left (equivalent to CONTROL + LEFT)
2	Right (equivalent to CONTROL + RIGHT)
3	Up (equivalent to CONTROL + UP)
4	Down (equivalent to CONTROL + DOWN)

## SELECT.LAST.CELL()

Selects the cell at the intersection of the last row and column that contain a formula, value, or format, or that are referred to in a formula.

## SELECT.PLOT.AREA()

**Note** | This macro function is included for compatibility with macros written with Microsoft Excel for the Apple Macintosh.

Equivalent to the Chart Select Plot Area command. Same as the SELECT("Plot") function.

## SELECT.SPECIAL(*type\_number,value\_types,levels*)

Equivalent to the Formula Select Special command (Full menus). *Type\_number* describes what to select, as follows:

Type_number	Description
1	Notes
2	Constants
3	Formulas
4	Blanks
5	Current region
6	Current array
7	Row differences
8	Column differences
9	Precedents
10	Dependents

*Value\_types* is only active when *type\_number* is 2 or 3. It is a number specifying values to be selected, as follows:

Value_type	Selects
1	Numbers
2	Text

Value_type	Selects
4	Logicals
16	Error values

These values can be added to select more than one type.

*Levels* is only active when *type\_number* is 9 or 10. It corresponds to option buttons, as follows:

Levels	Option Button
1	Direct Only
2	All Levels

## SELECTION()

Returns the reference of the selection as an external reference.

If you use the value returned by SELECTION in a function or operation, you will usually get the value contained in the selection instead of its reference.

This is because the reference is automatically translated into the contents of the reference. If you want to work with the actual reference, use the REFTXT function to convert the selection reference to text, which you can then store or manipulate.

### Example

If the document in the active window is named Sales, and if A1:A3 is the selection, then:

SELECTION() equals Sales!A1:A3

## SEND.KEYS(key\_text,wait\_log)

Sends the keys specified by *key\_text* to the active application, just as if they were typed at the keyboard.

Give *key\_text* in the format described in the Appendix.

If *wait\_log* is TRUE, Microsoft Excel waits for the keys to be processed before returning control to the macro. If *wait\_log* is FALSE, the default, the macro continues running without waiting for the keys to be processed.

**Note** | If Microsoft Excel is the active application, *wait\_log* is assumed to be FALSE, even if you enter *wait\_log* equal to TRUE. This is because if *wait\_log* is TRUE, Microsoft Excel waits for the keys to be processed before returning control to the macro. Microsoft Excel doesn't process keys while a macro is running.

**Example**

This macro uses the Calculator application, cuts the result, pastes it into Microsoft Excel, and displays the result in a message.

	A	B	C	D
1	Calculate			
2	=APP.ACTIVATE("Calculator")			
3	=SEND.KEYS("C6968*6889/7082736983=",TRUE)			
4	=SEND.KEYS("%EC",TRUE)			
5	=APP.ACTIVATE(FALSE)			
6	=SELECT(B1)			
7	=PASTE()			
8	=ALERT("The answer is "&B1,2)			
9	=RETURN()			
10				

**SET.CRITERIA()**

Equivalent to the Data Set Criteria command.

**SET.DATABASE()**

Equivalent to the Data Set Database command.

**SET.NAME(*name\_text,value*)**

Defines the name *name\_text* on the macro sheet to refer to *value*. If *value* is omitted, the name *name\_text* is deleted.

The SET.NAME function is useful for storing values during the calculation of a macro.

If *value* is a reference, *name\_text* is defined to refer to that reference. If you want to define *name\_text* to refer to the value of a referenced cell rather than to the reference itself, you must use the DEREf function.

**Note** | You can also enter the SET.NAME function using the following syntax:  
*name* = *value*

For example, the following two formulas are equivalent:

= SET.NAME("I",I + 1)

I = I + 1

## Examples

These formulas define the name I to refer to the constant number 1 on the macro sheet:

```
SET.NAME("I",1)
```

```
I = 1
```

These formulas redefine I to refer to the current value of I plus 1:

```
SET.NAME("I",I+1)
```

```
I = I + 1
```

These formulas define the name Results to refer to the cells A1:C3:

```
SET.NAME("Results",A1:C3)
```

```
Results = A1:C3
```

If cell A1 contains the value 2, these formulas define the name Reference to refer to cell A1:

```
SET.NAME("Reference",A1)
```

```
Reference = A1
```

And these formulas define the name Value to refer to the value 2:

```
SET.NAME("Value",DEREF(A1))
```

```
Value = DEREf(A1)
```

## SET.PAGE.BREAK()

Equivalent to the Options Set Page Break command (Full menus).

If the row or column next to the active cell already has a page break, SET.PAGE.BREAK takes no action.

## SET.PREFERRED()

Equivalent to the Gallery Set Preferred command (Full menus).

## SET.PRINT.AREA()

Equivalent to the Options Set Print Area command.

## SET.PRINT.TITLES()

Equivalent to the Options Set Print Titles command (Full menus).

## SET.VALUE(ref,values)

Changes the value of the cells specified by *ref* to *values*. If a cell previously contained a formula, the formula is not changed. SET.VALUE, like SET.NAME, is useful for assigning initial values, storing values, and for looping during the calculation of a macro. SET.VALUE is generally faster than SET.NAME.

*Ref* must be a reference to cells on the macro sheet. If *ref* is a reference to a range of cells, rather than to a single cell, then *values* should be an array of the same size. If not, Microsoft Excel expands it to that size using the normal rules for expanding arrays.

### Examples

The following function changes the value of cell A1 on the macro sheet to 1:

```
SET.VALUE(A1,1)
```

If that cell contains the formula `=A1 + 1`, the next time that formula is calculated, the value of the cell will become 2. (This method of adding to a counter is faster than repeatedly executing SET.VALUE functions such as SET.VALUE(A1,A1 + 1).)

The following function stores the values 1, 2, 3, and 4 in cells A1, B1, A2, and B2:

```
SET.VALUE(A1:B2,{1,2;3,4})
```

## SHORT.MENUS(logical)

Equivalent to the Options Short Menus command (Full menus) and the Options Full Menus command, if a worksheet or macro sheet is the active document. Equivalent to the Chart Short Menus command (Full menus) and the Chart Full Menus command, if a chart is the active document.

If *logical* is TRUE, displays Short menus. If *logical* is FALSE, displays Full menus.

## SHOW.ACTIVE.CELL()

Equivalent to the action of pressing CONTROL+BACKSPACE. Scrolls the active window so the active cell becomes visible.

## SHOW.BAR(*bar\_num*)

Displays the menu bar specified by menu bar ID number *bar\_num*. *Bar\_num* can either be the number of one of the Microsoft Excel built-in menu bars or the number returned by a previously executed ADD.BAR function. If *bar\_num* is omitted, Microsoft Excel displays the standard menu bar, as follows:

If active window contains	Standard bar
A worksheet or macro sheet (Full menus)	1
A worksheet or macro sheet (Short menus)	5
A chart (Full menus)	2
A chart (Short menus)	6
No active window	3
The Info window	4

When displaying a built-in menu bar, you can display only bars 1 or 5 if a worksheet or macro sheet is active, bars 2 or 6 if a chart is active, and so on. Displaying a custom menu bar disables automatic menu bar switching when different types of documents are selected.

For information on custom menus, see “Creating Customized Menus and Dialog Boxes” in Chapter 6, “Advanced Macros.”

## SHOW.CLIPBOARD()

**Note** | This macro function is included for compatibility with macros written with Microsoft Excel for the Apple Macintosh.

Equivalent to choosing the Run command on the Control menu for the Microsoft Excel application window and selecting Clipboard.

## SHOW.INFO(enable\_log)

If *enable\_log* is TRUE, SHOW.INFO activates the Info window. If the current window is the Info window and *enable\_log* is FALSE, SHOW.INFO activates the document linked to the Info window.

## SIGN(number)

Returns the sign of *number*. For more information, see SIGN in Chapter 2, “Worksheet Function Directory.”

## SIN(radians)

Returns the sine of *radians*. For more information, see SIN in Chapter 2, “Worksheet Function Directory.”

## SIZE(width,height,window\_text)

Equivalent to the Size command on the Control menu.

Changes the size of a window. It moves the lower-right corner of the window named *window\_text* so that the window has the width specified by *width* and the height specified by *height*. If *window\_text* is omitted, it is assumed to be the name of the active window.

SIZE does not change the position of the upper-left corner of the window or affect whether the specified window is active or inactive.

*Width* and *height* are given in points.

## SLN(cost,salvage,life)

Returns the straight-line depreciation for an asset. For more information, see SLN in Chapter 2, “Worksheet Function Directory.”

## SORT(sort\_by,key1,order1,key2,order2,key3,order3)

## SORT?(sort\_by,key1,order1,key2,order2,key3,order3)

Equivalent to the Data Sort command.

*Sort\_by* is a number that specifies whether to sort by rows or columns. Enter 1 to sort by rows, or 2 to sort by columns.



The next three pairs of arguments describe up to three keys. *Key1*, *key2*, and *key3* tell which rows or columns of data to use for sorting. There are two ways to specify sort keys:

Type of key	Examples
An external reference to the active worksheet	!B:B or !Price
R1C1-style references in the form of text. These are assumed to be relative to the active cell in the selection	“C2” or “C[1]” or “Price”

The *order1*, *order2*, and *order3* specify whether to sort in ascending or descending order. Enter 1 to sort in ascending order, or 2 to sort in descending order. The *order1* describes how to sort *key1*, and so on.

## Examples

The screen below shows the top of a database. Columns A–D are named GroupName, Drawing#, Size, and Name, respectively.

	A	B	C	D	E	F	G
6	Group Name	Drawing#	Size	Name			
7	Electrical	6220-306-01	A	Electro-Hydraulic Schematic	2		
8	Electrical	6220-306-04	A	Relay box Schematic	2		
9	Electrical	6220-306-05	A	Tester Schematic	2		
10	Frame	6220-203-15	A	Mount, Proximity Switch	2		
11	Frame	6220-203-16	A	Mount, Head Cylinder	2		
12	Frame	6220-303-02	A	Rubber Grommet	2		
13	Frame	6220-203-02	D	Scabbard Top	2		
14	Frame	6220-203-03	D	Scabbard Bottom	2		
15	Frame	6220-203-04	D	Scabbard Hose Enclosure	2		
16	Frame	6220-203-05	D	Rack Fronts (Right & Left)	2		
17							

Each of the following functions sorts the selection, by rows, in ascending order by the size of the drawing:

`SORT(1,!$C:$C,1)`

`SORT(1,“C3”,1)`

`SORT(1,“Size”,1)`

`SORT(1,!Size,1)`

If the active cell is A8, the following function sorts the selection by rows. The first key is column E, which is sorted in descending order, and the second key is the group name, which is sorted in ascending order.

```
SORT(1,"RC[4]",2,"RC",1)
```

Since cell B3 contains the text "name," the first function below sorts the selection using column B as the key, while the second function below sorts the selection using the column named Name as the key:

```
SORT(1,!$B$3,2)
```

```
SORT(1,DEREF(!$B$3),2)
```

### SPLIT(col\_split,row\_split)

Equivalent to the Split command on the Control menu for the document window. *Col\_split* specifies where to split the window in the x-direction, measured in columns from the left of the window. *Row\_split* specifies where to split the window in the y-direction, measured in rows from the top of the window.

If an argument is 0 and there is a split in that direction, the split is removed. If an argument is omitted, a split in that direction is not changed.

Trying to split a window that has frozen panes produces a message indicating the location of the error in your macro.

#### Example

The following function puts a vertical split after the third column and removes a horizontal split, if there is one:

```
SPLIT(0,3)
```

The following function puts a vertical split after the third column and has no effect on horizontal splitting:

```
SPLIT(,3)
```

### SQRT(number)

Returns the square root of *number*. For more information, see SQRT in Chapter 2, "Worksheet Function Directory."

### STDEV(number1,number2,...)

Estimates standard deviation of a population based on a sample. For more information, see STDEV in Chapter 2, "Worksheet Function Directory."

## STDEVP(number1,number2,...)

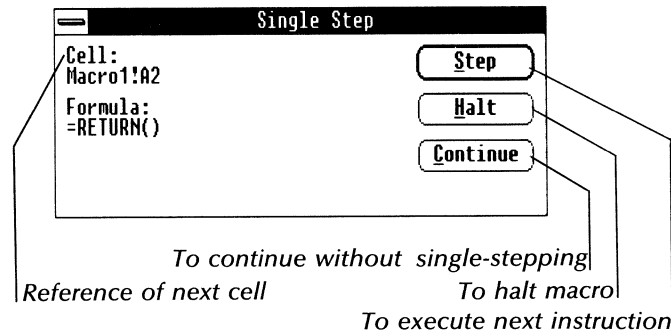
Returns the standard deviation of a population based on the entire population. For more information, see STDEVP in Chapter 2, “Worksheet Function Directory.”

## STEP()

Starts single-stepping, or pausing before calculating each cell in a macro.

Running a macro one cell at a time is called **single-stepping**, and is very useful when you are debugging a macro.

When Microsoft Excel encounters a STEP function, it stops running the macro and displays a dialog box.



The dialog box tells you which cell in the macro Microsoft Excel is about to calculate, and what formula is in that cell.

### Note

You can move the dialog box if it's in your way. The dialog box always appears centered at first, but if you move it, it will remain in that location until you stop single-stepping.

You can start single-stepping while a macro is running by pressing ESCAPE. Microsoft Excel displays a dialog box. Choose the Halt button to stop executing the macro, the Continue button to continue normal execution of the macro, or the Step button to start single-stepping through the macro.

For information on debugging, see Chapter 5, “Debugging and Testing Macros.”

### Examples

This function macro starts single-stepping as soon as it is started.

	A	B	C	D	E	F
25						
26	<b>Payments</b>					
27	=ARGUMENT("annualrate")					
28	=ARGUMENT("value")					
29	=ARGUMENT("years")					
30	=STEP()					
31	=annualrate/12					
32	=years*12					
33	=PMT(A31,A32,0,0)					
34	=RETURN(A33)					
35						

The following command macro lets you single-step through any available command macro. (To be available, the command macro must be on an open macro sheet, and must have a name.) You choose the command macro that you want from a dialog box.

	A	B	C	D	E	F	G	H
1	<b>Single Step</b>							
2	=STEP()							
3	=RUN?()							
4	=RETURN()							
5								

**STYLE**(bold,italic)

**STYLE?**(bold,italic)

#### Note

This macro function is included for compatibility with macros written with Microsoft Excel for the Apple Macintosh.

If you want to change a font to bold or italic, you will typically use the **FORMAT.FONT** or **REPLACE.FONT** functions.

If *bold* is TRUE, Microsoft Excel finds an available bold font and formats the current selection using that font. If *italic* is TRUE, Microsoft Excel finds an available italic font and formats the current selection using that font. If no appropriate font is available, Microsoft Excel finds the most similar font available and formats using that font.

## Subroutines: ref(*arg1,arg2,...*)

When a reference followed by a set of parentheses is encountered in the calculation of a macro, calculation **branches** to the upper-left corner cell of that reference. The macro starting at that cell is said to be a **subroutine** macro that is started by the macro containing the reference.

*Ref* can be a reference or a named reference. One macro can start another macro that is on another macro sheet using an external reference such as Finance!A1 or Finance!Future. Otherwise, *ref* is assumed to be on the same macro sheet. *Ref* can also be a formula that returns a reference, such as INDEX.

The initiating macro can give from 0 to 14 arguments to the subroutine macro.

Subroutines can be either command macros or function macros. Command macros that are used solely as subroutines are the only exception to the rule that command macros cannot accept or return arguments. If a command macro is used only as a subroutine, it can accept arguments and return results the same way function macros can.

For information on subroutine macros, see Chapter 3, “Macro Basics.”

## SUBSTITUTE(*text,old\_text,new\_text,instance\_number*)

Substitutes *new\_text* for *old\_text* in *text*. For more information, see SUBSTITUTE in Chapter 2, “Worksheet Function Directory.”

## SUM(*number1,number2,...*)

Returns the sum of *numbers*. For more information, see SUM in Chapter 2, “Worksheet Function Directory.”

## SYD(*cost,salvage,life,per*)

Returns the sum-of-years’ digits depreciation for an asset. For more information, see SYD in Chapter 2, “Worksheet Function Directory.”

### T(value)

Returns *value* translated into text. For more information, see T in Chapter 2, “Worksheet Function Directory.”

### TABLE(row\_ref,column\_ref)

### TABLE?(row\_ref,column\_ref)

Equivalent to the Data Table command (Full menus).

*Row\_ref* specifies the row input and *column\_ref* specifies the column input. They should be either external references to single cells on the active worksheet, such as !\$A\$1 or !Price, or R1C1-style references to single cells in the form of text, such as “R1C1”, “R[–1]C[–1]”, or “Price”. If *row\_ref* or *column\_ref* are R1C1-style references, they are assumed to be relative to the active cell in the selection.

#### Examples

Each of the following functions creates a two-input table in the selection on the active sheet using A1 and A2 as input cells:

```
TABLE(!$A$1,$A$2)
```

```
TABLE(“R1C1”,“R2C2”)
```

If the selection is C3:E5 and the active cell is C3, this function creates a one-input table in the selection using cell B2 as the column-input cell:

```
TABLE(“R[–1]C[–1]”)
```

If cell B2 on the active worksheet is named Size, each of the following functions creates a one-input table in the selection using cell B2 as the row-input cell:

```
TABLE(!Size,)
```

```
TABLE(“Size”,)
```

If cell B2 on the active worksheet contains the text “Size,” the first function below sets up a one-input table in the selection using cell B2 as the row-input cell, while the second function below sets up a one-input table using the cell named Size as the row-input cell.

```
TABLE(!$B$2,)
```

```
TABLE(DEREF(!$B$2),)
```

## TAN(radians)

Returns the tangent of *radians*. For more information, see TAN in Chapter 2, “Worksheet Function Directory.”

## TERMINATE(channel\_num)

**Note** | This function is supported only if you have Microsoft Windows (version 2.0).

Closes the channel *channel\_num*. The channel must have been opened with the INITIATE function.

If TERMINATE is not successful, it returns the #VALUE! error value.

For information on accessing other applications, see “Using Macros to Start Other Applications” in Chapter 6, “Advanced Macros.”

## TEXT(value,format\_text)

Converts *value* to text using format *format\_text*. For information on TEXT, see Chapter 2, “Worksheet Function Directory.”

## TEXTREF(text,a1)

Converts *text* into a reference. If *a1* is TRUE, *text* is assumed to be an A1-style reference. If *a1* is FALSE or omitted, *text* is assumed to be an R1C1-style reference.

### Examples

TEXTREF(“B7”,TRUE) equals B7

TEXTREF(“R5C5”,FALSE) equals E5

TEXTREF(“B7”,FALSE) equals the #REF! error value, because “B7” can’t be interpreted as an R1C1-style reference

## TIME(hour,minute,second)

Returns the serial number of specified time. For more information, see TIME in Chapter 2, “Worksheet Function Directory.”

## TIMEVALUE(*time\_text*)

Returns the serial number of time specified by *time\_text*. For more information, see TIMEVALUE in Chapter 2, “Worksheet Function Directory.”

## TRANSPOSE(*array*)

Returns the transpose of *array*. For more information, see TRANSPOSE in Chapter 2, “Worksheet Function Directory.”

## TREND(*known\_y's, known\_x's, new\_x's*)

Returns values on a linear trend. For more information, see TREND in Chapter 2, “Worksheet Function Directory.”

## TRIM(*text*)

Removes spaces from *text*. For more information, see TRIM in Chapter 2, “Worksheet Function Directory.”

## TRUE()

Returns the logical value TRUE. For more information, see TRUE in Chapter 2, “Worksheet Function Directory.”

## TRUNC(*number*)

Returns integer part of *number*. For more information, see TRUNC in Chapter 2, “Worksheet Function Directory.”

## TYPE(*value*)

Returns the type of *value*. For more information, see TYPE in Chapter 2, “Worksheet Function Directory.”

## UNDO()

Equivalent to the Edit Undo command.



## UNHIDE(window\_text)

Equivalent to the Window Unhide command (Full menus). *Window\_text* is the name of the window to unhide.

If no window named *window\_text* is available, UNHIDE produces a message indicating the location of the error in your macro.

## UNLOCKED.NEXT()

## UNLOCKED.PREV()

Equivalent to the actions of pressing TAB or SHIFT+TAB to move the active cell to the next or previous unlocked cell in a protected worksheet.

## UPPER(text)

Converts *text* to uppercase. For more information, see UPPER in Chapter 2, “Worksheet Function Directory.”

## VALUE(text)

Converts *text* to a number. For more information, see VALUE in Chapter 2, “Worksheet Function Directory.”

## VAR(number1,number2,...)

Estimates variance of population based on a sample. For more information, see VAR in Chapter 2, “Worksheet Function Directory.”

## VARP(number1,number2,...)

Returns the variance of a population based on the entire population. For more information, see VARP in Chapter 2, “Worksheet Function Directory.”

## VLINE(number\_rows)

Equivalent to the action of scrolling the active window by rows. Scrolls the active window vertically by *number\_rows* rows. If *number\_rows* is positive, Microsoft Excel scrolls down by *number\_rows* rows. If *number\_rows* is negative, Microsoft Excel scrolls up by *number\_rows* rows.

## VLOOKUP(lookup\_value,table\_array,col\_index)

Returns a value in a table selected by *lookup\_value*. For more information, see VLOOKUP in Chapter 2, “Worksheet Function Directory.”

## VPAGE(number\_windows)

Equivalent to the action of scrolling the active window vertically by *number\_windows* windows.

If *number\_windows* is positive, Microsoft Excel scrolls down by *number\_windows*. If *number\_windows* is negative, Microsoft Excel scrolls up by *number\_windows*.

## VSCROLL(scroll,row\_log)

Equivalent to the action of scrolling the active window vertically.

If *row\_log* is TRUE, then VSCROLL scrolls to row *scroll*.

If *row\_log* is FALSE or omitted, then VSCROLL scrolls to the row that is *scroll* percent of the distance to the bottom of the window. If *scroll* is 0, VSCROLL scrolls to the row that is 0% of the distance to the bottom of the window, which is row 1. If *scroll* is 1, VSCROLL scrolls to the row that is 100% of the distance to the bottom of the window, which is row 16384.

*Scroll* is always relative to the top of the document, not to the current location or to how much of the document contains information. For example, if you execute VSCROLL(25%), you always scroll to row 4096, 25% of the way down the worksheet. It doesn't matter where the worksheet was positioned.

To scroll to a specific row *n*, either use VSCROLL(*n*,TRUE) or use VSCROLL(*n*/16384). To scroll to row 138, for example, enter VSCROLL(138,TRUE) or VSCROLL(138/16384).

If you are recording a macro and move the scroll box several times in a row, the recorder only records the final location of the scroll box, omitting any intermediate steps. Remember that scrolling does not change the active cell or the selection.

## Examples

All of the following functions scroll to row 8192, 50% of the way down the worksheet:

VSCROLL(8192,TRUE)

VSCROLL(50%)

VSCROLL(.5,FALSE)

VSCROLL(8192/16384)

## WAIT(serial\_number)

Suspends execution of the macro until the time specified by serial number *serial\_number*. You can also resume execution by pressing ESCAPE (unless ESCAPE is disabled—for more information, see CANCEL.KEY).

For information on using WAIT, see “The WAIT Function” in Chapter 3, “Macro Basics.”

## WEEKDAY(serial\_number)

Converts *serial\_number* to a day of the week. For more information, see WEEKDAY in Chapter 2, “Worksheet Function Directory.”

## WHILE(logical\_test)

Starts a WHILE-NEXT loop. Executes the statements from WHILE to the next NEXT loop until *logical\_test* is FALSE. If *logical\_test* is FALSE the first time the WHILE function is reached, execution skips the loop and resumes at the statement after the next NEXT statement. For information on WHILE-NEXT loops, see “Looping” in Chapter 3, “Macro Basics.”

## WINDOWS()

Returns the names of all windows on your screen.

The names are given as a horizontal array of text values, in order of their level on your screen. The first name is the active window, the next name is the window directly under the active window, and so on. With the INDEX function, you can choose individual window names from the array for use in other functions that take window names as arguments.

### Examples

If the active window, named BUDGET.XLS, is on top of a window named Macro1:2, which is on top of a window named Macro1:1, then:

WINDOWS( ) equals {"BUDGET.XLS","Macro1:2","Macro1:1"}

The following command macro closes the active document, then reopens it. You can use it for reverting to the most recently saved version of a document.

	A	B	C	D	E	F	G
1	<b>Revert</b>						
2	=INDEX(WINDOWS()),1,1)						
3	=CLOSE(FALSE)						
4	=OPEN(A2)						
5	=RETURN()						
6							

**WORKSPACE**(fixed,decimals,r1c1,scroll,formula,status,menu,remote)

**WORKSPACE?**(fixed,decimals,r1c1,scroll,formula,status,menu,remote)

Equivalent to the Options Workspace command (Full menus). If an argument corresponding to a check box is TRUE, the check box is turned on; if the argument is FALSE, the check box is turned off.

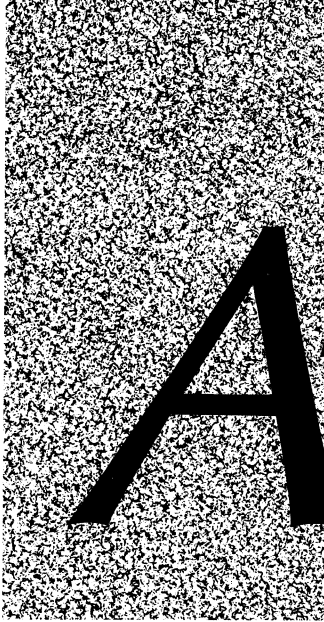
Argument	Description
<i>fixed</i>	Corresponds to the Fixed Decimal check box
<i>decimals</i>	The number of auto-decimal places
<i>r1c1</i>	Corresponds to the R1C1 check box
<i>scroll</i>	Corresponds to the Scroll Bars check box
<i>formula</i>	Corresponds to the Formula Bar check box

Argument	Description
<i>status</i>	Corresponds to the Status Bar check box
<i>menu</i>	A text value, indicating an alternate menu key
<i>remote</i>	Corresponds to the Ignore Remote Requests check box

YEAR(serial\_number)

Converts *serial\_number* to a year. For more information, see YEAR in Chapter 2, “Worksheet Function Directory.”





# Appendix: Key Codes

This appendix explains how to represent different keys on the keyboard, for use with the `ON.KEY` and `SEND.KEYS` macro functions. Each key is represented by one or more characters, such as “a” for the character a, or “{ENTER}” for the ENTER key. To specify more than one key, concatenate the codes for each character.

To specify a character on the keyboard, use that character. For example, to specify the dollar sign (\$) key followed by a (b), use the value “\$b”.

The plus sign (+), caret (^), and percent sign (%) have special meanings, as described in the section “Combining Keys with SHIFT, CONTROL, or ALT,” later in this appendix. To specify a special character, enter the character inside braces. For example, to specify a plus sign, enter {+}.

## Specifying Special Characters

To specify characters that aren’t displayed when you press the key, such as ENTER or TAB, use the codes shown in the following table. Each code in the table represents one key on the keyboard.

Key	Code
BACKSPACE	{BACKSPACE} or {BS}
BREAK	{BREAK}
CAPSLOCK	{CAPSLOCK}

Key	Code
CLEAR	{CLEAR}
DELETE	{DELETE} <i>or</i> {DEL}
DOWN	{DOWN}
END	{END}
ENTER	{ENTER} <i>or</i> ~ (tilde)
ESCAPE	{ESCAPE} <i>or</i> {ESC}
HELP	{HELP}
HOME	{HOME}
INSERT	{INSERT}
LEFT	{LEFT}
NUMLOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRTSC	{PRTSC}
RIGHT	{RIGHT}
TAB	{TAB}
UP	{UP}
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}



Key	Code
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}

## Combining Keys with SHIFT, CONTROL, or ALT

You can also specify keys combined with SHIFT and/or CONTROL and/or ALT. To specify a key combined with another key or keys:

To combine with	Precede the key code by
SHIFT	+ (plus sign)
CONTROL	^ (caret)
ALT	% (percent sign)

To specify that SHIFT, CONTROL, and/or ALT should be held down while another key is pressed, enclose the other key in parentheses. For example, to hold down SHIFT while pressing E followed by C, use “+(EC)”.

## Repeating Key Sequences

Use the form:

{Key number}

For example, {LEFT 42} means press the LEFT key 42 times.

**Note** | There must be a space between the key and the number.



# Index

1-2-3 *See* Lotus 1-2-3  
 100% chart 314, 324  
 1904 date system *See* Date

## A

A1 reference  
   absolute reference, conversion  
     to 337  
   display 236, 243  
   information return 302  
   text conversion to reference 365  
 A1.RIC1 function 236, 243  
 ABS function 23, 27, 243  
 Absolute reference  
   cell selecting 349  
   column width recording 260  
   conversion to 337  
   recording 137-138, 169, 174  
   returning 241, 243  
   row height recording 344  
 Absolute values 23, 27, 243  
 ABSREF function 241, 243  
 ACOS function 26, 27, 243  
 Action-performing functions 34, 57,  
   227-228  
 ACTIVATE function 235, 236,  
   243-244  
 ACTIVATE.NEXT function 236, 244  
 ACTIVATE.PREV function 236, 244  
 Active cell  
   changing 349-350  
   display 357  
   moving 351, 367  
   reference returning 241, 244-245  
 ACTIVE.CELL function 241,  
   244-245  
 Active document 150, 277  
 ADD.ARROW function 230, 245  
 ADD.BAR function 199, 238, 245  
 ADD.COMMAND function 199, 200,  
   238, 245-246  
 ADD.MENU function 199, 200, 238,  
   246  
 ADD.OVERLAY function 230, 246  
 Adding  
   numbers  
     DSUM function procedure 19,  
       46, 47, 273  
     SUM function procedure 2,  
       25, 112, 363  
   while pasting 328

Advanced macro keywords *See* Macro  
   function  
 Alarm sounding 238, 254  
 ALERT function 147, 186, 238,  
   247-248  
 Alignment  
   cell alignment information 293  
   centering  
     dialog box on screen 211  
     text 32, 248, 284  
   formula display 142  
   left-aligned text 32, 139, 248, 284  
   right-aligned text 32, 248, 284  
 ALIGNMENT function 233, 248  
 ALIGNMENT? function 248  
 Alphabetizing 358-359  
 ALT key 212, 375  
 ALT,SPACEBAR,M 249  
 ALT,SPACEBAR,S 250  
 Ampersand (&)  
   command names, use in 200  
   dialog box text column, use  
     in 212  
 AND function 22, 28, 248  
 Annuity  
   defined 97  
   functions  
     *See also* FV function; NPER  
       function; PMT function;  
       RATE function  
     designated, purpose 97-100  
     interest rate *See* Interest  
 APP.ACTIVATE function 238, 249  
 APP.MAXIMIZE function 230, 249  
 APP.MINIMIZE function 231, 249  
 APP.MOVE function 231, 249  
 APP.MOVE? function 249  
 APP.RESTORE function 231, 250  
 APP.SIZE function 231, 250  
 APP.SIZE? function 250  
 Applications *See* Outside applications  
 APPLY.NAMES function 233, 250  
 APPLY.NAMES? function 250-251  
 Arccosine 26, 27, 243  
 Arcsine 26, 29, 252  
 Arctangent 26, 29-30, 252  
 Area chart  
   creating 291, 314  
   overlay chart type 324  
 Area defined 28  
 AREAS function 21, 28-29, 251  
 Argument  
   array use as *See* Array

Argument (*continued*)  
   command macro arguments 161,  
     363  
   dialog box function arguments 227  
   function macro arguments *See*  
     Function macro  
   macro function arguments 229  
   worksheet function arguments *See*  
     Worksheet function  
 ARGUMENT function 177-178, 240,  
   251-252  
 ARRANGE.ALL function 235, 252  
 Array  
   arguments, use as  
     CHOOSE function argu-  
       ment 34  
     function macro arguments 179,  
       180  
     IF function argument 57  
     worksheet function argu-  
       ments 7, 13  
   columns, number of 21, 37, 260  
   data type *See* Data types  
   determinant 24, 80-81, 315  
   formula *See* Array formula  
   INDEX function manipulation  
     60-61  
   inverse matrix 24, 83-84, 317  
   line description 67-71  
   looking up values *See* Looking up  
     values  
   LOOKUP function 77  
   multiplying 24, 86-87, 317  
   naming 179  
   reversing *See herein* inverse  
     matrix  
   rows, number of 22, 105, 345  
   selecting 352  
   transposing 24, 116-117, 366  
   vector *See* Vector  
   window pane edges 301  
 Array formula 54, 70, 74, 84, 87,  
   116, 118, 237, 286  
 Arrow  
   adding to chart 245  
   base 283  
   deleting 268  
   patterns setting 333  
   position determination 296  
   selecting 350  
 Ascending sort order 359  
 ASCII code conversion  
   from character 25, 36

ASCII code conversion (*continued*)  
to character 25, 34

ASIN function 26, 29, 252

Asset

depreciation *See* Depreciation

salvage value 108, 112

useful life 108, 112

Asterisk (\*)

dialog box icon 209, 217

multiplication operator 8

searching for 288

wildcard character 79, 106, 278,  
288, 323

ATAN function 26, 29-30, 252

ATAN2 function 26, 30, 252

ATTACH.TEXT function 230, 252

ATTACH.TEXT? function 253

autoexec file 136, 140

Autoexec macros 140, 189, 193-195

AVERAGE function 24, 30-31, 253

Averaging numbers

AVERAGE function procedure  
24, 30-31, 253

DAVERAGE function procedure  
19, 44-45, 47, 265

AXES function 230, 253

AXES? function 253

Axis

label selecting 350

scale setting 347-348

## B

BACKSPACE key 373

Bar chart 219, 314, 324

Batch file 194

BEEP function 147, 238, 254

Beep sounding 238, 254

Blank cell

information return 21, 65, 310

marking 90

selecting 352

Blank line 166

Bold

cell information return 294

document information return 298

setting 281, 282, 341, 362

Border

cell border 293-294

chart border 331

creating 254

macro sheets border 139, 143

shadow borders 331

worksheet border 196

BORDER function 196, 233, 254

BORDER? function 254

Braces { }, keystroke enclosure 373

Branching *See* Macro

BREAK function 154, 167, 240, 254

BREAK key 373

Bug *See* Debugging

Built-in function *See* Function

## C

CALCULATE.DOCUMENT function  
223, 230, 235, 254

CALCULATE.NOW function 230,  
235, 254

Calculating

documents 223, 254-255

iteration 63, 102, 255, 298

lines 67-71

macro sheet/worksheet differ-  
ences 139

mode information return 298

single-stepping 361-362

speed control 223

worksheets

macro control 154

macro sheet differences 139

CALCULATION function 235, 255

CALCULATION? function 255

CALL function 238, 255-256, 337

CALLER function 179, 241, 256

Cancel button 207, 213

CANCEL.COPY function 236, 257

CANCEL.KEY function 220, 238,  
257

Canceling marquees 236, 257

Capitalization *See* Uppercase charac-  
ters

CAPSLOCK key 373

Caret (^)

exponentiation operator 50

key combinations 375

Case *See* Lowercase characters;  
Uppercase characters

Cash flow *See* Investment

Category axis

scale setting 347-348

selecting 350

Cell

active cell *See* Active cell

alignment information 293

area defined 28

blank cell *See* Blank cell

border *See* Border

column width *See* Column

conditional testing 57-58

data type checking 65-66

filling 278-279, 286

formatting information 21, 31-33,  
241, 293-294

Cell (*continued*)

hiding 220, 294

information return 21, 31-33, 241,  
293-294

jumping to 240, 287-288

label 32, 141-142

last cell selecting 237, 351

locking

determination 32

information return 294

moving to unlocked cell 237,  
238

naming 250-251

non-empty cell counting *See*  
Counting

protecting 32, 220, 257

range *See* Range

recorder range cells 172

reference *See* Reference

selecting 348-349

shifting 274, 310

value *See* Value

CELL function 21, 31-33, 257

Cell label 32, 141-142

CELL.PROTECTION function 233,  
257

CELL.PROTECTION? function 257

Cell reference *See* Reference

Centering

dialog box on screen 211

text 32, 248, 284

CHANGE.LINK function 232, 257

CHANGE.LINK? function 257

Channel *See* Outside applications

CHAR function 25, 34, 258

Character

ASCII code 25, 34, 36

control character 25, 35

counting characters in docu-  
ment 290

non-printable characters 35

reading from file 221, 239, 289

special characters 373-374

underlining 212

writing to file 221, 239, 290

Chart

*See also* Document

area colors/patterns setting 331

area chart *See* Area chart

arrow *See* Arrow

axis

patterns setting 332

scale setting 347-348

background 281

bar chart *See* Bar chart

border setting 331

cluster spacing 314

colors 281, 331

## Chart (continued)

- column chart *See* Column chart
- copying 236, 261
- creating 313-315, 318
- data series 264-265, 297
- default chart setting 355
- drop line *See* Drop line
- flow chart 158-160
- font 281-282
- footer 326
- header 326
- hi-lo line *See* Hi-lo line
- layout 326-327
- legend *See* Legend
- line chart *See* Line chart
- main chart 313-315
- margins 326-327
- object
  - moving 282-283
  - position 241, 294-296
  - selecting 236, 349-351
  - sizing 283
- overlay chart *See* Overlay chart
- pasting text 328-329
- patterns 330-333
- pie chart *See* Pie chart
- scatter chart *See* Scatter chart
- selecting 349-351
- sizing 327
- text
  - attaching 253
  - entry 237
  - label base 283
  - tickmarks 332
  - title selecting 350
  - type return 297
- Chart Add Arrow command 230, 245
- Chart Add Legend command 230, 311
- Chart Add Overlay command 230, 246
- Chart Attach Text command 230, 253
- Chart Axes command 230, 253
- Chart Calculate Document command 230, 254
- Chart Calculate Now command 230, 254
- Chart Delete Arrow command 230, 268
- Chart Delete Legend command 230, 311
- Chart Delete Overlay command 230, 269
- Chart Full Menus command 230, 356
- Chart Gridlines command 230, 304
- Chart Protect Document command 230, 336
- Chart Select Chart command 230, 351
- Chart Select Plot Area command 230, 352
- Chart Short Menus command 230, 356
- Chart Unprotect Document command 230, 336
- CHECK.COMMAND function 203, 238, 258
- Checked commands *See* Command
- CHOOSE function 23, 34-35, 258
- CLEAN function 25, 35-36, 258
- CLEAR function 231, 258
- CLEAR? function 258-259
- CLEAR key 374
- Clearing 258-259
- Clipboard 36, 237
- CLOSE.ALL function 232, 259
- CLOSE function 230, 259
- Closing
  - channels 365
  - documents
    - autoexec macro running 195
    - procedure 221, 239, 277
    - windows 259
- CODE function 25, 36, 259
- Color
  - chart colors 281, 331
  - data line color setting 333
  - display 272
  - font colors 281
  - information return 301
  - line colors 330
- Column
  - array columns, number of 21, 37, 260
  - cell column information return 293
  - chart *See* Column chart
  - custom commands, macro sheet columns 200-201
  - custom dialog box, macro sheet columns 211-217
  - custom menus, macro sheet columns 200-201
  - database column *See* Database
  - document column information return 298
  - heading *See* Column heading
  - numbers 21, 31, 36
  - pasting 329
  - recorder range columns 172
  - scrolling 237, 305, 306
  - sorting 358
  - width
    - changing 260
    - information return 31, 294
    - macro sheets 139

- Column chart
  - creating 292, 314
  - overlay chart type 324
- COLUMN function 21, 36, 259
- Column heading
  - color information return 301
  - creating 326
  - display
    - information return 301
    - options 271
- COLUMN.WIDTH function 233, 260
- COLUMN.WIDTH? function 260
- COLUMNS function 21, 37, 178, 260
- COMBINATION function 234, 261
- COMBINATION? function 261
- Comma (,)
  - function arguments separators 5, 17
  - function macro arguments separator 135
  - union operator 17
- Command
  - See also specific command*
  - adding to menu 199, 200-202
  - checked commands
    - adding checkmarks 203, 238, 258
    - deleting checkmarks 203, 258
    - description 198
  - command macro correspondence 201
  - creating 200-202, 238, 245-246
  - deleting 202, 238, 268
  - disabling 203, 274
  - enabling 203, 274
  - greyed commands
    - adding grey 203, 239, 274
    - deleting grey 203, 239, 274
    - description 198
  - help file specifying 201
  - macro *See* Command macro
  - macro function correspondence *See* Command-equivalent functions
  - message creating 201
  - naming 200-201, 202-203, 240, 340
  - outside application commands executing 239, 276
  - renaming 202-203, 240, 340
  - separator line 200
  - structure 197
- Command-equivalent functions
  - defined 227
  - designated 230-236
  - forms 145

## Command macro

*See also* Macro

- arguments 161, 363
- command correspondence 201
- creating 171-176
- decisions within macros 148-149, 167
- defined 131
- description 161
- directory 243-371
- display 138
- documenting *See* Documenting macros
- function macro differences 161-162
- functions, use in 167-168, 226
- listing 164
- looping *See* Looping macros
- modifying 144-154
- naming *See* Macro
- recording macros *See* Recording macros
- running *See* Macro
- speed control 273
- stepping through *See* Debugging
- stopping *See* Macro
- structure *See* Macro
- subroutine macro, use as 150, 152, 363
- values from documents 150
- writing macros *See* Writing macros

## Commenting *See* Documenting macros

## Conditional testing 57-58

## Constant selecting 352

## Control character deleting 25, 35

## Control Close command 230, 259

## Control function

- defined 228, 229
- designated 240-241
- summary 166

## CONTROL key 133, 375

## Control macro 165

## Control Maximize command 230, 249

## Control Minimize command 231, 249

## Control Move command 231, 249, 317

## Control Restore command 231, 250

## Control Run command 231, 357

## Control Size command 231, 250, 358

## Control Split command 231, 360

## CONTROL+BACKSPACE 357

## CONTROL+DOWN 351

## CONTROL+F5 290

## CONTROL+F6 244

## CONTROL+F10 290

## CONTROL+LEFT 351

## CONTROL+RIGHT 351

## CONTROL+SHIFT+ENTER 54, 70, 74, 84, 87, 116, 118, 286

## CONTROL+SHIFT+F6 244

## CONTROL+UP 351

## COPY.CHART function 236, 261

## COPY.CHART? function 261

## COPY function 231, 261

## COPY.PICTURE function 231, 261

## Copying

- charts 236, 261
- pictures 261-262

## COS function 26, 37-38, 262

## Cosine 26, 37-38, 262

## COUNT function 24, 38, 262

## COUNTA function 24, 39, 262

## Counter *See* Looping macros

## Counting

### non-empty cells

- COUNTA function procedure 24, 39, 262

- DCOUNTA function procedure 19, 46, 47, 266

### numbers

- COUNT function procedure 24, 38, 262

- DCOUNT function procedure 19, 45, 47, 265

## CREATE.NAMES function 233, 263

## CREATE.NAMES? function 263

## Criteria 45, 354

## Currency format 25, 32, 48-49, 272

## Curve fitting

- exponential curves 53-55, 73-75, 304, 313
- least squares procedure 69, 117-120
- polynomial curve fitting 118
- predicting y values 70, 118-119
- straight lines 67-71, 117-120, 312, 366

## Custom command 200-202, 238, 245-246

## Custom data forms 206

## Custom dialog box 204-218, 220, 307-310

## Custom Help 201, 219-220, 239

## Custom menu 196-204, 220, 238, 246

## Custom menu bar 199, 238, 245

## Customizing functions

- defined 227, 228
- designated 238-240

## CUT function 231, 263

## D

## Data Delete command 231, 263

## DATA.DELETE function 231, 263

## DATA.DELETE? function 263

## Data Exit Find command 231, 263

## Data Extract command 231, 277

## Data Find command 231, 263, 264

## DATA.FIND function 231, 263

## DATA.FIND.NEXT function 237, 264

## DATA.FIND.PREV function 237, 264

## Data form 206, 231, 264

## Data Form command 231, 264

## DATA.FORM function 231, 264

## Data line 295, 332-333

## Data Parse command 231, 327

## Data series 264-265, 297

## Data Series command 231, 264

## DATA.SERIES function 231, 264

## DATA.SERIES? function 264-265

## Data Set Criteria command 45, 231, 354

## Data Set Database command 44, 231, 354

## Data Sort command 231, 358

## Data Table command 231, 364

## Data types

- checking 65-66
- defined 3
- designated 9-10
- function macro
  - arguments, specifying 180, 252
  - result, specifying 177, 179-180, 240, 343
- manual conventions 18
- returning 121, 366
- selecting 352-353
- translating 12-13

## Database

### column *See herein* fields

### criteria 45, 354

### data forms 206, 231, 264

### defined 44

### fields 44, 45

### functions 19, 44-48

### range, naming as Database 44

### records

- averaging numbers 19, 44-45, 47, 265
- counting numbers 19, 45, 47, 265
- defined 44
- deleting 263
- finding 237, 263-264
- largest number return 19, 46-47, 272
- matching 237, 264
- multiplying 19, 46, 47, 273

- Database (*continued*)
  - records (*continued*)
    - non-empty cells counting 19, 46, 47, 266
    - smallest number return 19, 46-47, 272
    - summing 19, 46, 47, 273
  - row *See herein* records
  - setting 354
- Database functions 19, 44-48
- Date
  - 1904 date system
    - description 83, 88, 91, 107, 126, 127
    - information return 298
  - entry in cell 146
  - format, text value equivalent 33
  - serial number 20, 39-40, 41, 91-92, 265, 319
- DATE function 20, 39-41, 265
- Date functions 20
- Date series 265
- DATEVALUE function 20, 41-42, 265
- DAVERAGE function 19, 45, 46-47, 265
- Day
  - conversion from serial number
    - to day of month 20, 42, 265
    - to day of week 20, 125-126, 369
  - format, text value equivalent 33
- DAY function 20, 42-43, 265
- DCOUNT function 19, 45, 46-47, 265
- DCOUNTA function 19, 46-47, 266
- DDB function 20, 43-44, 266
- DDE 222-223, 271, 307
- Debugging
  - bug defined 165
  - description 165
  - error handling 158, 170, 274-275
  - macro testing 189-190
  - methods 184-190
  - return functions method 187
  - stepping through macros 185-186, 240, 361-362
  - values display 187-188
- Decimal places 48, 52, 302
- DEFINE.NAME function 233, 266-267
- DEFINE.NAME? function 266-267
- Degrees
  - conversion from radians 27, 29, 30
  - conversion to radians 37, 107, 114
- DELETE.ARROW function 230, 268
- DELETE.BAR function 199, 238, 268
- Delete command *See* Data Delete command; Edit Delete command; File Delete command
- DELETE.COMMAND function 202, 238, 268
- DELETE.FORMAT function 237, 268
- DELETE key 374
- DELETE.MENU function 202, 238, 269
- DELETE.NAME function 233, 269
- DELETE.OVERLAY function 230, 269
- Deleting
  - arrows 268
  - chart overlay 269
  - command
    - checked commands 203, 258
    - greyed commands 203, 239, 274
    - regular commands 202, 238, 268
  - control characters 25, 35
  - documents 278
  - formatting 268
  - legends 312
  - menu bars 199, 238, 268
  - menus 202, 238, 269
  - messages 315-316
  - names 269
  - non-printable characters 35
  - notes 319
  - number formatting 237
  - page breaks 340
  - records 263
  - return addresses 240, 342
  - spaces from text 26, 120, 366
  - window split 360
- Demos 195-196
- Depreciation
  - double-declining balance depreciation 20, 43-44, 266
  - straight-line depreciation 21, 108-109, 358
  - sum-of-years' digits depreciation 21, 112-113, 363
- DEREF function 241, 269-270
- Descending sort order 359
- Determinant 24, 80-81, 315
- Dfunction 44-48
- Dialog box
  - ALERT function dialog box 147
  - centering 211
  - character underlining 212
  - creating 204-218, 220, 307-310
  - custom dialog box 204-218, 220, 307-310
- Dialog box (*continued*)
  - display 238, 239, 247-248, 270, 307-310
  - Help 211, 220
  - message creating 247-248
- DIALOG.BOX function 206, 238, 270
- Dialog box functions 145, 227, 236
- Directional macro keys 169
- Directory
  - changing 237
  - display 210
  - list box 210
  - path setting 270
  - return 297
- DIRECTORY function 237, 270
- Directory of functions 27-127, 243-371
- DISABLE.INPUT function 220, 238, 271
- Disabling commands 203, 274
- Disk drive 210, 270
- DISPLAY function 234, 235, 271
- #DIV/0!
  - AVERAGE function error 65
  - EXECUTE function error 276
  - MIRR function error 85
  - MOD function error 87
  - POKE function error 334
  - REQUEST function error 342
- Dividing
  - remainder 23, 87, 317
  - while pasting 328
- DMAX function 19, 46-47, 272
- DMIN function 19, 46-47, 272
- Document
  - access 279
  - active document
    - closing 277
    - referral notation 150
  - calculating 223, 254-255
  - closing 195, 221, 239, 277
  - creating 318
  - deleting 278
  - information return 296-299
  - linking *See* Linking
  - listing 209, 278
  - loading *See herein* opening
  - macro sheet *See* Macro sheet
  - names
    - definition returning 241, 296, 299-300
    - returning 241, 242, 296, 318
  - open documents listing 272
  - opening
    - autoexec macro running 194
    - FOPEN function procedure 221, 239, 279

Document (*continued*)  
   opening (*continued*)  
     OPEN function procedure 322-323  
     parsing 221, 327  
     positioning 221, 239, 289  
     printing 335  
     protecting 220, 297, 336  
     read-only document 279, 297  
     saving 345-346  
     scrolling *See* Scrolling  
     size 239, 290  
     type return 297  
     worksheet *See* Worksheet  
     writing to 221, 239, 279, 290-291

Documenting macros  
   comments defined 141  
   demos documenting 195  
   notes *See* Note  
   procedures 141-142  
   purpose 165

DOCUMENTS function 241, 272

Dollar format 25, 32, 48-49, 272

DOLLAR function 25, 48-49, 272

Double-declining balance depreciation 20, 43-44, 266

Double quotation marks *See* Quotation marks

DOWN key 374

DPRODUCT function 19, 46, 47, 273

Drive 210, 270

Drop line  
   creating 314  
   overlay chart 324  
   patterns setting 332  
   selecting 350

DSTDEV function 19, 46, 47-48, 273

DSTDEVP function 19, 46, 47-48, 273

DSUM function 19, 46, 47, 273

DVAR function 19, 46, 47-48, 273

DVARP function 19, 46, 47-48, 273

Dynamic Data Exchange 222-223, 271, 307

## E

ECHO function 196, 223, 238, 273

Edit Clear command 231, 258

Edit Copy command 231, 261

Edit Copy Picture command 231, 261

Edit Cut command 231, 263

Edit Delete command 231, 274

EDIT.DELETE function 231, 274

EDIT.DELETE? function 274

Edit Fill Down command 232, 278

Edit Fill Left command 232, 278

Edit Fill Right command 232, 279

Edit Fill Up command 232, 279

Edit Insert command 232, 310

Edit Paste command 232, 327

Edit Paste Link command 232, 327

Edit Paste Special command 232, 327, 328

Edit Repeat command 232

Edit Undo command 232, 366

Editing  
   macro sheets 140  
   notes 319  
   recorded macros 144-154, 176, 195

Ellipsis (. . .), function argument notation 5, 16-17

Empty cell *See* Blank cell

Empty text  
   cell information notation 32  
   character matching 105  
   defined 9  
   function argument substitution 17  
   message removal 315-316

ENABLE.COMMAND function 203, 239, 274

Enabling commands 203, 274

END key 374

ENTER key 349, 374

Equal sign (=)  
   formula indicator 6  
   macro use restrictions 141

ERROR function 239, 274-275

Error handling 158, 170, 274-275

Error value  
   checking for 21, 65, 311  
   data type *See* Data types  
   selecting 353

ESCAPE key  
   altering 238  
   key code 374  
   macro stopping 186

EXACT function 25, 50, 275

Excel *See* Microsoft Excel; Microsoft Excel for the Macintosh

Exchanging data *See* Outside applications

Exclamation point (!)  
   active document, reference to 150, 168  
   dialog box icon 209, 217  
   macro sheet references 133

EXEC function 221, 239, 275-276

EXECUTE function 222, 239, 275

Executing macros *See* Macro

EXP function 23, 50-51, 277

Exponential curve 53-55, 73-75, 304, 313

Exponential trend 24, 53-55

Exponentiation 23, 50, 277

Exponentiation operator (^) 50

Exporting data *See* Outside applications

External reference  
   command macro, use in 150  
   function macro argument 180  
   macro sheet, references to 169  
   returning 353  
   subroutine macro running 152  
   updating 323  
   worksheet names returning 312

EXTRACT function 231, 277

EXTRACT? function 277

## F

F1 key 185, 374

F2 key 374

F3 key 374

F4 key 374

F5 key 287, 374

F6 key 374

F7 key 287, 374

F8 key 374

F9 key 374

F10 key 375

F11 key 375

F12 key 375

F13 key 375

F14 key 375

F15 key 375

F16 key 375

FACT function 23, 51, 277

Factorial 23, 51, 277

FALSE function 22, 51, 277

FALSE value 10, 17

FCLOSE function 221, 239, 277

Field *See* Database

File  
   *See also* Document  
   batch file 194  
   Help file 227  
   linking *See* Linking  
   name *See* Filename  
   Workspace file 136  
   writing to 221, 239, 279, 290-291

File Close All command 232, 259

File Close command 195, 232, 277

FILE.CLOSE function 232, 277

File Delete command 232, 278

FILE.DELETE function 232, 278

FILE.DELETE? function 278

File Exit command 232

File Links command 232, 257, 323

File New command 163, 232, 318



- File Open command
  - document opening 194
  - macro function equivalent 232, 322
  - macro sheet opening 132, 163
- File Page Setup command 232, 326
- File Print command 232, 335
- File Printer Setup command 232, 336
- File Record Macro command 232
- File Save As command 232, 345
- File Save command 232, 345
- File Save Workspace command 232, 346
- File Unhide Window command 232
- Filename
  - definition returning 241, 296, 299-300
  - extension, macro files 133
  - returning 241, 242, 296, 318
- FILES function 241, 278
- FILL.DOWN function 232, 278
- FILL.LEFT function 232, 278
- FILL.RIGHT function 232, 279
- FILL.UP function 232, 279
- Financial functions 20-21
- Find command *See* Data Find command
- FIND function 25, 52, 279
- Finding
  - asterisks 288
  - database records 237, 263-264
  - formulas 286-287
  - question marks 288
  - text
    - FIND function procedure 25, 52, 279
    - SEARCH function procedure 26, 105-106, 348
- FIXED function 25, 52-53, 279
- Flow chart 158-160
- Font
  - cell information return 294
  - changing 196, 237, 280-282, 341
  - chart fonts 281
  - colors 281
  - macro sheet fonts 139, 281, 298-299
  - worksheet fonts 281, 298-299
- Footer 326
- FOPEN function 239, 279
- FOR function 167, 240, 280
- FOR-NEXT loop 280, 319
- Format Alignment command 233, 248
- Format Border command 233, 254
- Format Cell Protection command 220, 233, 257
- Format Column Width command 220, 233, 260
- Format Font command 233, 281, 341
- FORMAT.FONT function 233, 282
- FORMAT.FONT? function 233, 280, 281-282
- Format Justify command 233, 311
- Format Legend command 233, 282
- FORMAT.LEGEND function 233, 282
- Format Main Chart command 233, 313
- Format Move command 233, 282
- FORMAT.MOVE function 233, 282-283
- FORMAT.MOVE? function 282-283
- Format Number command
  - DOLLAR function differences 49
  - FIXED function differences 52
  - macro function equivalent 233, 283
  - number formatting 138
- FORMAT.NUMBER function 146, 233, 283
- FORMAT.NUMBER? function 283
- Format Overlay command 233, 324
- Format Patterns command 233, 330
- Format Row Height command 220, 233, 344
- Format Scale command 233, 347
- Format Size command 233, 283
- FORMAT.SIZE function 233, 283
- FORMAT.SIZE? function 283
- Format Text command 233, 283
- FORMAT.TEXT function 233, 283-284
- Formatting
  - See also specific formatting*
  - cell formatting information 21, 31-33, 241, 293-294
  - checking 32-33
  - clearing 258-259
  - deleting 268
  - document formatting information 241, 296-299
  - effect on cell value 11
  - general format 32
  - legends 282
  - macro sheets 139, 142-143
  - numbers *See* Number
  - pasting 328, 329
  - text 283-284
  - window formatting information 241, 300-302
  - workspace formatting information 242, 302-303
- Formula
  - alignment 142
  - array formula *See* Array formula
  - clearing 258-259
- Formula (*continued*)
  - conditional testing 57-58
  - dialog box formulas 208, 214
  - display
    - information return 301
    - options 271
  - entering into cell 237, 284-286
  - equal sign as indicator 6
  - errors 7-8, 65
  - filling range 237
  - finding 286-287
  - function
    - argument, formula use as 10-11
    - entry into formula 4-6
    - left-aligned in cell 139
    - names, use in 8
    - pasting 328, 329
    - returning 293, 299
    - selecting 352
- Formula Apply Names command 233, 250
- FORMULA.ARRAY function 237, 286
- Formula bar 302
- Formula Create Names command 143, 233, 263
- Formula Define Name command
  - autoexec macro creating 194, 195
  - cell label creating 142
  - function macros listing 136
  - macro function equivalent 233, 266-267, 269
  - macro naming 154, 164
  - names listing 8
- FORMULA.FILL function 237, 286
- Formula Find command 233, 286
- FORMULA.FIND function 233, 286-287
- FORMULA.FIND.NEXT function 237, 287
- FORMULA.FIND.PREV function 237, 287
- FORMULA function 146, 237, 284-286
- Formula Goto command 233, 287
- FORMULA.GOTO function 233, 287-288
- FORMULA.GOTO? function 287-288
- Formula Note command 233
- Formula Paste Function command
  - function
    - entering in formula 4-6
    - listing 5, 6, 168
    - parentheses entry 11
  - function macros
    - listing 164, 182
    - running 135, 182

Formula Paste Function command (*continued*)  
     macro function equivalent 233  
 Formula Paste Name command 234, 313  
 Formula Reference command 234  
 Formula Replace command 234, 288  
 FORMULA.REPLACE function 234, 288-289  
 FORMULA.REPLACE? function 288-289  
 Formula Select Special command 234, 352  
 FPOS function 221, 239, 289  
 Fractions truncating 24, 121  
 FREAD function 221, 239, 289  
 FREADLN function 221, 239, 289-290  
 FREEZE.PANES function 235, 290  
 FSIZE function 221, 239  
 FULL function 230, 231, 290  
 Full menu display 356  
 Function  
     *See also specific function*  
     action-performing functions 34, 57, 227-228  
     command-equivalent functions *See* Command-equivalent functions  
     control functions *See* Control function  
     customizing functions *See* Customizing functions  
     data types *See* Data types  
     database functions 19, 44-48  
     date functions 20  
     dialog box functions 145, 227, 236  
     directory of functions 27-127, 243-371  
     financial functions 20-21  
     information functions 21-22  
     logical functions 22  
     lookup functions 23  
     macro function *See* Macro function  
     macros, use in 167-168, 226  
     mathematical functions 23-24  
     matrix functions 24  
     statistical functions 24-25  
     text functions 25-26  
     time functions 20  
     trigonometric functions 20  
     user-defined functions *See* Function macro  
     value-returning functions *See* Value-returning function  
     worksheet functions *See* Worksheet function

Function macro  
     *See also* Macro arguments  
     arrays 179-180  
     data types *See* Data types  
     defined 181  
     describing 252  
     forms 177-178  
     function order 177  
     number allowed 177  
     omitted, value 135  
     references 180  
     separators 135  
     translation 180  
     array as argument 179-180  
     caller cell 241  
     CALLER function *See* CALLER function  
     command macro differences 161-162  
     defined 131, 140  
     description 177  
     ending 181  
     example 181  
     functions, use in 168, 177, 226  
     listing 5, 136, 164, 182  
     macro function differences 140  
     naming *See* Macro  
     recording inability 137, 164, 177  
     result, data type specifying 177, 179-180, 240, 343  
     running *See* Macro  
     sample 181  
     starting cell 179, 256  
     stepping through *See* Debugging  
     stopping *See* Macro  
     writing macros *See* Writing macros  
 Future value *See* Investment  
 FV function 20, 53, 290  
 FWRITE function 221, 239, 290-291  
 FWRITELN function 221, 291

## G

Gallery Area command 234, 291  
 GALLERY.AREA function 234, 291  
 GALLERY.AREA? function 291  
 Gallery Bar command 234, 291  
 GALLERY.BAR function 234, 291  
 GALLERY.BAR? function 291  
 Gallery Column command 234, 292  
 GALLERY.COLUMN function 234, 292  
 GALLERY.COLUMN? function 292  
 Gallery Combination command 234, 261

Gallery Line command 234, 292  
 GALLERY.LINE function 234, 292  
 GALLERY.LINE? function 292  
 Gallery Pie command 234, 292  
 GALLERY.PIE function 234, 292  
 GALLERY.PIE? function 292  
 Gallery Preferred command 234, 335  
 Gallery Scatter command 234, 292  
 GALLERY.SCATTER function 234, 292  
 GALLERY.SCATTER? function 292  
 Gallery Set Preferred command 234, 355  
 General format 32  
 GET.BAR function 204, 241, 293  
 GET.CELL function 33, 241, 293-294  
 GET.CHART.ITEM function 241, 294-296  
 GET.DEF function 241, 296  
 GET.DOCUMENT function 241, 296-299  
 GET.FORMULA function 241, 299  
 GET.NAME function 241, 299-300  
 GET.NOTE function 241, 300  
 GET.WINDOW function 241, 300-302  
 GET.WORKSPACE function 242, 302-303  
 GOTO function  
     CHOOSE function argument 34  
     IF function argument 57, 167  
     jumping to cell 167, 240, 303  
     macro order control 151, 303  
     recording macros 173  
 Grade assignments 58  
 Greyed commands *See* Command  
 Gridline  
     color information return 301  
     creating 326  
     display  
         information return 301  
         options 271  
         procedure 304  
     patterns setting 332  
     selecting 350  
 GRIDLINES function 230, 304  
 GRIDLINES? function 304  
 GROWTH function 24, 53-55, 70, 304

## H

HALT function 166, 187, 240, 304  
 Halting macros *See* Macro  
 Header 326

## Help

- creating 219-220, 239
- customized Help 201, 219-220, 239
- dialog boxes 211, 220
- functions 6, 9
- index display 304
- message Help 185
- starting 304
- Help file format 227
- HELP function 228, 239, 304-305
- Help Index command 130
- HELP key 374
- Hi-lo line
  - creating 314
  - overlay chart 324
  - patterns setting 332
  - selecting 350
- HIDE function 236, 305
- Hiding
  - cells 220, 294
  - macro sheets 140
  - windows 300, 305, 367
- HLINE function 237, 305
- HLOOKUP function 23, 55-56, 305
- HOME key 374
- Horizontal scrolling *See* Scrolling
- Hour
  - conversion from serial number 20, 56-57, 305
  - format, text value equivalent 33
- HOURLY function 20, 56-57, 305
- HPAGE function 237, 305
- HSCROLL function 237, 306

## I

- Icon 209, 217
- ID number *See* Menu bar; Outside applications
- Identity matrix 83-84
- IF function 22, 57-58, 148-149, 167, 306
- Implicit intersection 35
- Importing data *See* Outside applications
- INDEX function
  - array form 23, 60-61, 84, 179, 306
  - reference form 23, 59-60, 74-75, 306, 312
- INDIRECT function 21, 61-62, 307
- Info Cell command 234
- Info Dependents command 234
- Info Format command 234
- Info Formula command 234
- Info Names command 234

- Info Note command 234
- Info Precedents command 234
- Info Protection command 234
- Info Value command 234
- Info Window 358
- Information functions 21-22
- Initial capitals 26, 97, 336
- INITIATE function 222, 239, 307
- INPUT function 239, 307-310
- INSERT function 232, 310
- INSERT? function 310
- INSERT key 374
- INT function 23, 62, 310
- Integer
  - defined 9
  - dialog box integer 208, 213
  - returning 366
  - rounding down to 23, 62, 310
- Interest
  - internal rate of return 20, 63-64, 84-86, 310, 317
  - payment 20, 62-63, 310
  - rate 21, 98, 102, 337
  - Internal rate of return 20, 63-64, 84-86, 310, 317
- Investment
  - future value 20, 53, 99, 290
  - interest *See* Interest
  - net present value 21, 92-94
  - payment
    - negative number representation 62, 96, 98
    - number of 21, 92, 99, 319
    - periodic payments 21, 95, 99, 334
    - principal payments 21, 96, 334
    - present value 21, 99, 337
- IPMT function 20, 62-63, 310
- IRR function 20, 63-64, 310
- ISBLANK function 21, 65, 66, 310
- ISERR function 21, 65, 311
- ISERROR function 21, 65, 66, 311
- ISfunction 65-66
- ISLOGICAL function 22, 65, 66, 311
- ISNA function 22, 65, 66, 311
- ISNONTEXT function 22, 66, 311
- ISNUMBER function 22, 66, 311
- ISREF function 22, 66, 311
- ISSTRING function *See* ISTEEXT function
- ISTEXT function 22, 66, 311
- Italic
  - cell information return 294
  - chart fonts 282
  - document information return 299
  - function argument notation 16
  - setting 281, 282, 341

- Iteration 63, 102, 255, 298

## J

- Jumping to cells
  - FORMULA.GOTO function procedure 233, 287-288
  - GOTO function procedure 151, 167, 240, 303
- JUSTIFY function 233, 311
- Justifying text 311

## K

- Key
  - alternate menu key 303
  - directional macro keys 169
  - keystrokes, sending to outside application 223, 353-354, 373-375
  - macro running 239, 321
  - shortcut keys *See* Shortcut key

## L

- Label
  - cell labels 32, 141-142
  - dialog box labeling 208
  - tickmark labels 332
- Layout 326-327
- Least squares procedure 69, 117-120
- Left-aligned text 32, 139, 248, 284
- LEFT function 25, 67, 311
- LEFT key 374
- Legend
  - creating 312
  - positioning 282
  - selecting 350
- LEGEND function 230, 311
- LEN function 11, 25, 67, 311
- Line
  - blank lines 166
  - calculating 67-71
  - chart *See* Line chart
  - colors 330
  - equation 67, 68
  - linear trends 24, 25, 67-71, 117-120
  - parse line 327
  - patterns 330
  - predicting y values 70, 118-119
  - reading from file 221, 239, 289-290
  - settings 330
  - slope 68, 69
  - straight line 67-71, 117-120, 312, 366

## Line (continued)

- weights 330
- writing to file 221, 239, 291
- y-intercept 68, 69

## Line chart

- creating 292, 314
- overlay chart type 324

## Linear trend 24, 25, 67-71, 117-120

## LINEST function 24, 67-71, 312

## Linking

- data forms to database 206
- documents
  - linked documents names
    - return 242, 312
  - procedure 257, 323
- list box to text box 209, 216

## LINKS function 242, 312

## LIST.NAMES function 234, 313

## LN function 23, 72, 313

## Loading See Opening

## Locking cells

- determination 32
- information return 294
- moving to unlocked cell 237, 238

## LOG function 23, 72, 313

## LOG10 function 23, 73, 313

## Logarithm

- functions 23, 72-73
- natural logarithm 23, 72, 313
- returning 313

## Logarithmic scale 348

## LOGEST function 24, 73-75, 313

## Logical functions 22

## Logical value

- checking for 22, 65, 311
- data type See Data types
- function argument 12, 17
- function macro, data type specifying 180

## return

- AND function procedure 22, 28, 248
- FALSE function procedure 22, 51, 277
- IF function procedure 22, 57-58, 306
- NOT function procedure 22, 90-91, 319
- OR function procedure 22, 94-95, 324
- TRUE function procedure 22, 120, 336

- reversal 90, 319
- selecting 353

## Looking up values

- HLOOKUP function procedure 23, 55-56, 305
- LOOKUP function procedure 23, 75-77, 313

## Looking up values (continued)

- MATCH function procedure 78-79
- VLOOKUP function procedure 23, 124-125, 368

## LOOKUP function 23, 75-77, 313

## Looping macros

- breaking loop 154, 240, 254
- counters 153
- description 152-153
- ending loop 240, 319
- FOR-NEXT loop 280, 319
- increments 153
- nesting loops 154
- starting loop 240, 241, 280
- stop value 153
- WHILE-NEXT loop 319, 369

## Lotus 1-2-3

- blank lines 166
- directional macro keys 169
- macros translation 130

## LOWER function 25, 77-78, 313

## Lowercase characters

- case-sensitive searching 52
- conversion to 25, 77-78, 313
- shortcut key distinction 134, 165

# M

## Macro

- See also Command macro; Function macro

## 1-2-3 macros translation 130

- autoexec macros 140, 189, 193-195

## branching 151-152, 303, 363

- command macro See Command macro

## commenting 141-142, 165, 195

## control macro 165

## creating 162-165, 190

## debugging See Debugging

- decisions within macros See Command macro

## defined 130

## demos 195-196

## direction 150-153

## directional macro keys 169

## documenting 141-142, 165, 195

## error handling

- action specifying 170, 239, 274-275
- debugging See Debugging
- defined 158

## filename extension 133

## files listing 133

## flow control 229

- function macro See Function macro

## Macro (continued)

## halting See herein stopping

## interrupting 186

- jumping to cell 151, 167, 240, 303

## limits 190

## looping See Looping macros

## message creating 147

## mistakes See Debugging

## moving among cells 169

## naming

- advantages 141, 164
- command procedure 154, 164
- function procedure 266-267
- recording procedure 137, 138

## online lessons 130

## pausing 147-148

## protecting 220

## recording See Recording macros

## references See Reference

## running

- automatically 140, 193-195
- command macros 133-135, 136, 139, 162
- function macros 135-136, 139, 182

## key initiation 239, 321

## order 141, 150-153

## outside data initiation 239, 320

## RUN function procedure 345

- time initiation 239, 321-322, 369

## when closing document 195

- when opening document 193-194

## window change initiation 239, 322, 345

## within macro 167

## sample macros 131

## sheet See Macro sheet

## shortcut keys See Shortcut key

- single-stepping 185-186, 240, 361-362

## size guidelines 165-167

## slowing down 195

## speed control 195-196, 223

## starting See herein running

- stepping through 185-186, 240, 361-362

## stopping

- disabling 257
- error-handling subroutine macro procedure 170
- ESCAPE key procedure 186
- HALT function procedure 166, 240, 304
- prevention 220
- RETURN function procedure 166, 343-344

Macro (*continued*)

- stopping (*continued*)
    - WAIT function procedure 147-148, 240, 369
  - structure
    - changing 150-153
    - guidelines 165-167
  - subroutine macro *See* Subroutine macro
  - testing 189-190
  - text, use with 221
  - translating 130
  - types 131
  - use suggestions 131-132
  - user input prevention 220
  - worksheet calculation control 154
  - writing macros *See* Writing macros
- Macro Absolute Record command 138, 169, 174, 234
- Macro function  
*See also specific function*  
 action-performing functions 34, 57, 227-228  
 arguments 229  
 command-equivalent functions *See* Command-equivalent functions  
 customizing functions *See* Customizing functions  
 defined 140  
 dialog box function 145, 227, 236  
 function macro differences 140  
 listing 168  
 other action-equivalent functions 227, 228, 236-238  
 purpose 167  
 return values 143, 149-150, 188  
 types 168, 226  
 value-returning functions 149, 228, 229, 241-242
- Macro Record command 137, 138, 234
- Macro recording *See* Recording macros
- Macro Relative Record command 138, 169, 174, 235
- Macro Run command  
 command macro  
 listing 164  
 running 133, 136, 162  
 macro function equivalent 235, 345  
 macro shortcut keys listing 134
- Macro Set Recorder command 172, 235
- Macro sheet  
 arrangement 140  
 attaching to document 136

Macro sheet (*continued*)

- borders 139, 143
  - calculating 139
  - column width setting 139
  - command-defining area 200
  - commenting *See* Documenting macros
  - creating 163, 318
  - defined 131
  - documenting *See* Documenting macros
  - editing 140
  - examples 142-143
  - fonts 139, 281, 298-299
  - formatting 139, 142-143
  - formulas display 139, 143, 187-188
  - hiding 140
  - information return 298-299, 301
  - layout 326
  - left-aligned text 139
  - margins 326
  - menu-defining area 200
  - multiple macros 140
  - opening
    - automatically 136, 163
    - macro recording 137
    - manually 132, 163
    - multiple macro sheets 140
  - organization 140
  - pasting text 327-328
  - printing 335
  - recorder range *See* Recording macros
  - references to 133, 169
  - samples 142-143
  - selecting cells 348-349
  - shading 143
  - text alignment 139, 142
  - values
    - checking 188
    - display 143, 169-170, 187-188
    - writing in 150
    - worksheet differences 139
- Macro Start Recorder command 235
- Macro Stop Recorder command 137, 138, 175, 235
- Macro Translation Assistant 130
- Main chart 313-315
- MAIN.CHART function 233, 313-315
- MAIN.CHART.TYPE function 315
- Manual conventions 16
- Margin 326-327
- Marquee 196, 236, 257
- MATCH function 23, 78-79, 315
- Matching  
*See also* Finding; Looking up values; Replacing

Matching (*continued*)

- text 25, 50, 275
  - values 23, 78-79, 148-149, 315
- Math coprocessor 303
- Mathematical functions 23-24
- Matrix  
 determinant 24, 80-81, 315  
 identity matrix 83-84  
 inverse 24, 83-84, 317  
 multiplying 24, 86-87, 317
- Matrix functions 24
- MAX function 25, 80, 315
- Maximize command *See* Control Maximize command
- MDETERM function 24, 80-81, 315
- Memory 303
- Menu  
 adding to menu bar 199, 200-202  
 alternate menu key 303  
 bar *See* Menu bar  
 checked commands *See* Command  
 creating 196-204, 220, 238, 246  
 custom menus 196-204, 220, 238, 246  
 deleting 202, 238, 269  
 description 198  
 greyed commands *See* Command  
 nil menu 246
- Menu bar  
 creating 199, 238, 245  
 deleting 199, 238, 268  
 description 198  
 display 199, 204, 240, 357  
 ID number  
 designated 246  
 purpose 199  
 returning 199, 200, 204, 241, 293  
 maximum number 245  
 types 197
- Message  
 command message creating 201  
 deleting 315-316  
 display 239, 315-316  
 ERROR function, effect 275  
 Help 185  
 macro messages creating 147
- MESSAGE function 147, 239, 315-316
- Microsoft Excel  
 activating 249  
 input blocking 238, 271  
 maximized 303  
 minimized 303  
 quitting 337  
 version number returning 302
- Microsoft Excel for the Macintosh  
 date system 40, 41, 42, 56, 83, 88, 91, 107, 126, 127

Microsoft Excel for the Macintosh  
*(continued)*  
 function compatibility 315, 351,  
 352, 357, 362  
 macro compatibility 261  
 macro function compatibility 325  
 Microsoft Windows 221-223  
 Microsoft Windows library  
 accessing 240, 337-340  
 calling 238, 255-256  
 MID function 26, 81-82, 316  
 MIN function 25, 82, 317  
 Minimize command *See* Control  
 Minimize command  
 Minute  
 conversion from serial number 20,  
 82-83, 317  
 format, text value equivalent 33  
 MINUTE function 20, 82-83, 316  
 MINVERSE function 24, 83-84, 317  
 MIRR function 2, 20, 84-86, 317  
 MMULT function 24, 86-87, 317  
 MOD function 23, 87-88, 317  
 Modes 303  
 Modified internal rate of return 2, 20,  
 84-86, 317  
 Modulus 87  
 Month  
 conversion from serial number 20,  
 88, 317  
 format, text value equivalent 33  
 MONTH function 20, 88, 317  
 Mouse  
 information return 303  
 macro function correspondence *See*  
 Other action-equivalent func-  
 tions  
 Move command *See* Control Move  
 command  
 MOVE function 231, 317-318  
 Moving  
 active cell 349, 351, 367  
 chart objects 282-283  
 windows 249, 317-318  
 Multiplication operator (\*) 8  
 Multiplying  
 arrays 24, 86-87, 317  
 numbers  
 DPRODUCT function pro-  
 cedure 19, 46, 47, 273  
 PRODUCT function procedure  
 24, 96-97, 336  
 while pasting 328

## N

N function 22, 89, 318

#N/A  
 argument value 135, 178  
 checking for 311  
 empty cell marking 90  
 EXECUTE function error 276  
 FILES function error 278  
 FOPEN function error 279  
 FREAD function error 289, 290  
 function *See* NA function  
 function macro argument 252  
 FWRITE function error 291  
 GET.DOCUMENT function  
 error 297  
 GET.WORKSPACE function  
 error 303  
 HLOOKUP function error 55  
 LINKS function error 312  
 LOOKUP function error 76, 77  
 MATCH function error 78  
 ON.TIME function error 322  
 REQUEST function error 342  
 VLOOKUP function error 124  
 NA function 22, 90, 318  
 Name  
*See also* Naming  
 cell label 32, 141-142  
 conventions 142, 164  
 creating  
 CREATE.NAMES function pro-  
 cedure 263  
 DEFINE.NAME function  
 procedure 266-267  
 Formula Define Name command  
 procedure 142  
 RENAME.COMMAND func-  
 tion procedure 240  
 SET.NAME function pro-  
 cedure 240, 354-355  
 definition  
 matching 241  
 return 299-300  
 deleting 269  
 document name  
 definition return 241, 296,  
 299-300  
 return 241, 242, 296, 318  
 duplication restrictions 178  
 formulas, use in 8  
 function argument  
 translating data types 12  
 use as 11, 45  
 listing 8  
 window name return 242, 300  
 #NAME? 5, 8  
 NAMES function 242, 318  
 Naming  
*See also* Name  
 arrays 179

Naming *(continued)*  
 cells 32, 141-142  
 command macros 154  
 commands 200-201, 202-203,  
 240, 340  
 criteria ranges 45  
 database ranges 44  
 macros  
 advantages 141, 164  
 command procedure 154, 164  
 function procedure 266-267  
 recording procedure 137, 138  
 Natural logarithm 23, 72, 313  
 Negative number, payment representa-  
 tion 62, 96, 98  
 Nesting  
 functions 10-11  
 IF functions 57  
 loops 154  
 Net present value 21, 92-94, 320  
 NEW function 232, 318  
 NEW? function 318  
 NEW.WINDOW function 236, 319  
 NEXT function 167, 240, 319  
 Nil menu 246  
 Non-action performing functions  
 228-229  
 Non-empty cell counting *See* Count-  
 ing  
 Non-printable characters 35  
 NOT function 22, 90-91, 319  
 Note  
 characters returning 241  
 clearing 258-259  
 comments defined 141  
 creating 319  
 deleting 319  
 documenting macros  
 demos documenting 195  
 procedures 141-142  
 purpose 165  
 editing 319  
 length restrictions 319  
 pasting 328  
 printing 335  
 returning 300  
 selecting 352  
 NOTE function 233, 319  
 NOW function 20, 91-92, 319  
 NPER function 21, 92, 319  
 NPV function 21, 64, 92-94, 320  
 #NUM!  
 GROWTH function error 54  
 IRR function error 63  
 RATE function error 102  
 SQRT function error 109  
 Number  
 absolute value 23, 27

## Number (continued)

- accuracy limit *See herein* significant digits
- adding
  - DSUM function procedure 19, 46, 47, 273
  - SUM function procedure 2, 25, 112, 363
- arccosine 26, 27, 243
- arcsine 26, 29, 252
- arctangent 26, 29-30, 252
- ASCII character equivalent 25
- averaging
  - AVERAGE function procedure 24, 30-31, 253
  - DAVERAGE function procedure 19, 44-45, 47, 265
- checking for 22, 66, 311
- conversion to text 25, 52-53, 279
- cosine 26, 37-38, 262
- counting
  - COUNT function procedure 24, 38, 262
  - DCOUNT function procedure 19, 45, 47, 265
- data type *See* Data types
- decimal places 48, 52, 302
- description 9
- dialog box number 208, 213
- dividing *See* Dividing
- equality comparison 148-149
- exponentiation 23, 50, 277
- factorial 23, 51, 277
- formatting
  - deleting 237
  - Format Number command procedure 138
  - FORMAT.NUMBER function procedure 283
  - macro procedure 138
  - zero display 301
- function argument, use as 10, 12, 17
- function macro, data type specifying 180
- ID number *See* Menu bar; Outside applications
- matching *See* Matching
- maximum number return
  - DMAX function procedure 19, 46-47, 272
  - MAX function procedure 25, 80, 315
- minimum number return
  - DMIN function procedure 19, 46-47, 272
  - MIN function procedure 25, 82, 317

## Number (continued)

- multiplying
  - DPRODUCT function procedure 19, 46, 47, 273
  - PRODUCT function procedure 24, 96-97, 336
- negative numbers 62, 96, 98
- precision setting 149
- random numbers 24, 100-101, 337
- rounding
  - DOLLAR function procedure 48, 272
  - FIXED function procedure 52, 279
  - INT function procedure 23, 62, 310
  - ROUND function procedure 24, 104, 344
- selecting 352
- serial number *See* Serial number
- sign 24, 107, 358
- significant digits 9, 52
- sine 26, 108, 358
- square roots 24, 109, 360
- tangent 26, 114, 365
- text conversion to number 26, 122, 367
- truncating fractions 24, 121
- values translated to number 22, 89, 318

NUMLOCK key 374

## O

- OFFSET function 242, 320
- OK button *See* Dialog box
- ON.DATA function 239, 320
- ON.KEY function 239, 321, 373
- ON.TIME function 148, 239, 321-322
- ON.WINDOW function 204, 239, 322
- One-input table 364
- Open Freeze Panes command 290
- OPEN function 232, 322-323
- OPEN? function 322-323
- OPEN.LINKS function 232, 323
- OPEN.LINKS? function 323
- Open Unfreeze Panes command 290
- Opening
  - documents
    - autoexec macro running 194
    - FOPEN function procedure 221, 239, 279
    - OPEN function procedure 322-323

## Opening (continued)

- macro sheets
  - automatically 136, 163
  - macro recording 137
  - manually 132, 163
  - multiple macro sheets 140
  - workspace file 136
- Options Calculate Document command 235, 254
- Options Calculate Now command 235, 254
- Options Calculation command 149, 235, 255, 335
- Options Display command
  - macro function equivalent 235, 271
  - macro sheet display changing 139, 143, 187-188
  - worksheet display changing 139
- Options Freeze Panes command 235
- Options Full Menus command 235, 356
- Options Protect Document command 220, 235, 336
- Options Remove Page Break command 235, 340
- Options Set Page Break command 235, 355
- Options Set Print Area command 235, 355
- Options Set Print Titles command 235, 356
- Options Short Menus command 235, 356
- Options Unfreeze Panes command 235
- Options Unprotect Document command 235, 336
- Options Workspace command 235, 243, 370
- OR function 22, 94-95, 324
- Other action-equivalent functions
  - defined 227, 228
  - designated 236-238
- Outside applications
  - activating 249
  - CELL function compatibility 33
  - channel
    - closing 222, 240, 365
    - defined 222
    - opening 222, 239, 307
  - commands executing 222, 239, 276
  - exchanging data with
    - channel *See herein* channel
    - Clipboard procedure 223
    - DDE procedure 222-223
    - ON.DATA function procedure 239, 320

Outside applications (*continued*)  
 exchanging data with (*continued*)  
   POKE function procedure 239, 334  
   remote control procedure 223  
   REQUEST function procedure 240, 341-342  
   SEND.KEYS function procedure 223, 240, 353-354, 373-375  
 ID number *See herein* task ID number  
 keystrokes, sending to 240, 353-354, 373-375  
 LOOKUP function compatibility 77  
 macro running 239, 320  
 N function compatibility 89  
 NA function compatibility 90  
 remote controlling 223  
 starting  
   APP.ACTIVATE function procedure 238, 249  
   EXEC function procedure 221, 239, 275-276  
   T function compatibility 113  
   task ID number 275-276, 307  
   VALUE function compatibility 122  
 Overlapped chart 314, 324, 325  
 Overlay chart  
   adding to chart 246  
   data series 297  
   deleting 269  
   type return 297, 324-325  
 OVERLAY.CHART.TYPE function 325  
 OVERLAY function 233, 324-325

## P

Page break  
   deleting 340  
   setting 355  
 PAGE DOWN key 374  
 Page layout 326-327  
 PAGE.SETUP function 232, 325-327  
 PAGE.SETUP? function 325-327  
 PAGE UP key 374  
 Parentheses (( )), reference enclosure 17  
 PARSE function 221, 231, 327  
 Paste Arguments 5, 6  
 PASTE function 232, 327  
 PASTE.LINK function 232, 327  
 PASTE.SPECIAL function 232, 327-329

PASTE.SPECIAL? function 327-329  
 Pasting text 327-329  
 Path 270, 297  
 Patterns 330-333  
 PATTERNS function 233, 330-333  
 Payment *See* Investment  
 Percent sign (%) 375  
 Period (.), name component 142, 164  
 Pi 23, 95, 334  
 PI function 23, 95, 334  
 Picture copying 261-262  
 Pie chart  
   creating 292, 314  
   overlay chart type 324  
   sizing 283  
   slice *See* Pie slice  
 Pie slice  
   angle 314, 324  
   base 283  
   position determination 296  
 Plot area selecting 350  
 Plus sign (+) 375  
 PMT function 21, 95, 334  
 Point defined 229  
 POKE function 222, 239, 334  
 Polynomial curve fitting 118  
 Powers *See* Exponentiation  
 PPMT function 21, 96, 334  
 Precision As Displayed 298, 335  
 PRECISION function 235, 335  
 Predictions 70, 118-119  
 PREFERRED function 234, 335  
 Present value *See* Investment  
 Print area setting 355  
 PRINT function 232, 335  
 PRINT? function 335  
 Printer, setting up 336  
 PRINTER.SETUP function 232, 336  
 PRINTER.SETUP? function 336  
 Printing 148, 335  
 PRODUCT function 24, 96-97, 336  
 Programs *See* Outside applications  
 PROPER function 26, 97, 336  
 PROTECT.DOCUMENT function 230, 235, 336  
 PROTECT.DOCUMENT? function 336  
 Protecting cells 32, 220, 257  
 Protecting documents 220, 297, 336  
 Protecting macros 220  
 PRTSC key 374  
 PV function 21, 97-100, 337

## Q

Question mark (?)  
   dialog box function notation 145, 236

Question mark (?) (*continued*)  
   dialog box icon 209, 217  
   searching for 288  
   wildcard character 79, 106, 278, 288, 323  
 QUIT function 230, 232, 337  
 Quitting Microsoft Excel 337  
 Quotation marks (" ")  
   cell information notation 32  
   empty text notation 32  
   entering as text 286  
   text enclosure 3, 8, 9

## R

R1C1 reference  
   absolute reference, conversion to 337  
   display 236, 243  
   information return 302  
   macro references 153  
   text conversion to reference 365  
 Radians  
   conversion from degrees 37, 107, 114  
   conversion to degrees 27, 29, 30  
   cosine 26, 37-38, 262  
   sine 26, 108, 358  
   tangent 26, 114, 365  
 RAND function 24, 100-101, 337  
 Random number 24, 100-101, 337  
 Range  
   area defined 28  
   filling in 237  
   naming as Criteria 45  
   naming as Database 44  
   reference *See* Reference  
 RATE function 21, 102, 337  
 Read-only document 279, 297  
 Recalculating *See* Calculating  
 Record *See* Database  
 Recording macros  
   absolute references 137-138, 169, 174  
   command macros 136-138, 171  
   dialog box functions recording 145  
   examples 138, 174  
   function macro, inability 137, 164, 177  
   macro naming 164  
   macro sheet creating 163  
   modifying recorded macros 144-154, 176, 195  
   order 173  
   partial macros 163, 174  
   procedure 136-138, 163, 171, 174



## Recording macros (*continued*)

- recorded actions 175
- recorder range 172-173
- recording over cell contents 172
- relative references 137-138, 169, 171, 174
- row height recording 344
- sample 138, 174
- scrolling recording 306, 368-369
- status bar display 137, 171
- stopping 173, 175, 176
- unrecorded actions 175
- viewing 173

## #REF!

- CALLER function error 256
- EXECUTE function error 276
- HLOOKUP function error 55
- INDEX function error 59, 61
- OFFSET function error 320
- POKE function error 334
- REQUEST function error 342
- VLOOKUP function error 124

## Reference

- absolute *See* Absolute reference
- active cell reference returning 241, 244-245
- changing to values 356
- checking for 22, 66, 311
- column numbers 21, 36
- contents information 21, 31-33
- conversion to text 242, 337
- data type *See* Data types
- dialog box references 208, 214
- external reference *See* External reference
- formatting information 21, 31-33
- function argument, use as 11, 12, 17
- function macro argument restrictions 178, 180
- Help topic display 219
- INDEX function manipulation 59-60
- location information 21, 31-33
- looking up values *See* Looking up values
- macro function argument, reference conversion 229
- number of areas 21, 28-29
- offset reference 242, 320
- recording *See* Recording macros
- relative *See* Relative reference
- remote reference 298, 323
- return 61-62
- selecting 237, 348-349
- selection reference return 242, 353
- text conversion to reference 242, 365

## Reference (*continued*)

- type checking 65-66
- updating 3
- value returning 241, 269-270
- REFTEXT function 242, 310, 337
- REGISTER function 240, 256, 337-340
- Regression curve 55
- Relative reference
  - absolute reference, changing to 169
  - column width recording 260
  - recording 137-138, 169, 171, 174
  - returning 242, 340
  - row height recording 344
- RELREF function 242, 340
- Remote controlling applications 223
- Remote reference 298, 323
- Remote request 302
- REMOVE.PAGE.BREAK function 235, 340
- Removing *See* Deleting
- RENAME.COMMAND function 202-203, 240, 340
- Renaming commands 202-203, 240, 340
- Repeating
  - key sequences 375
  - text 26, 103, 341
- REPLACE.FONT function 233, 341
- REPLACE function 26, 102-103, 341
- Replacing
  - fonts 341
  - formulas 288-289
  - text
    - REPLACE function procedure 26, 102-103, 341
    - SUBSTITUTE function procedure 26, 111, 363
- REPT function 26, 103, 341
- REQUEST function 222, 240, 341-342
- RESTART function 240, 342
- Restore command *See* Control Restore command
- RESULT function 177, 179, 240, 343
- Return address 240, 342
- RETURN function
  - macro
    - debugging 187
    - ending 167, 175, 176, 181, 241, 343-344
    - recording 171, 175
    - writing over 176
- Return value
  - command macros 161
  - function macros 179-180
  - macro functions 143, 149-150, 188

- Right-aligned text 32, 248, 284
- RIGHT function 26, 103-104, 344
- RIGHT key 374
- ROUND function 24, 104, 344
- Rounding numbers
  - DOLLAR function procedure 48, 272
  - FIXED function procedure 52, 279
  - INT function procedure 23, 62, 310
  - ROUND function procedure 24, 104, 344

## Row

- array rows, number of 22, 105, 345
- cell row information 293
- database row *See* Database
- document row information 298
- heading *See* Row heading
- height 294, 344
- numbers 22, 31, 104-105, 344
- pasting 329
- scrolling 238, 367
- sorting 358
- ROW function 22, 104-105, 344
- Row heading
  - color information return 301
  - creating 326
  - display
    - information return 301
    - options 271
- ROW.HEIGHT function 233, 344
- ROW.HEIGHT? function 344
- ROWS function 22, 105, 178, 345
- Run command *See* Control Run command
- RUN function 194, 235, 345
- RUN function? 345
- Running macros *See* Macro

## S

- Salvage value 108, 112
- SAVE.AS function 232, 345-346
- SAVE.AS? function 345-346
- SAVE function 232, 345
- SAVE.WORKSPACE function 232, 346
- SAVE.WORKSPACE? function 346
- Saving
  - documents 345-346
  - workspace 136, 346
- Savings, future value *See* Investment
- SCALE function 233, 347-348
- Scale setting 347-348

- Scatter chart
  - creating 292, 314
  - overlay chart type 324
- Screen
  - scrolling *See* Scrolling
  - updating
    - control 273
    - turning on/off 196, 223, 238
- Scroll bars, display 302
- Scrolling
  - horizontally 237, 305-306
  - recording 306, 368-369
  - rows 238, 367
  - vertically 238, 367, 368-369
  - windows 305, 368-369
- SEARCH function 26, 105-106, 348
- Searching
  - asterisks 288
  - database records 237, 263-264
  - formulas 286-287
  - question marks 288
  - text
    - FIND function procedure 25, 52, 279
    - SEARCH function procedure 26, 105-106, 348
- Second
  - conversion from serial number 20, 106-107, 348
  - format, text value equivalent 33
- SECOND function 20, 106-107, 348
- SELECT.ACTIVE.CELL function 237
- SELECT.CHART function 351
- SELECT.END function 237, 351
- SELECT function 230, 236, 237, 348-351
- SELECT.LAST.CELL function 237, 351
- SELECT.PLOT.AREA function 352
- SELECT.SPECIAL function 234, 352-353
- Selecting
  - arrays 352
  - arrows 350
  - axis label 350
  - category axis 350
  - cells
    - blank cell 352
    - last cell 237, 351
    - procedure 348-349
  - chart objects 236, 349-351
  - chart title 350
  - charts 349-351
  - constants 352
  - data types 352-353
  - dialog box items 6
  - drop lines 350
- Selecting (*continued*)
  - error values 353
  - formulas 352
  - gridlines 350
  - hi-lo lines 350
  - legend 350
  - logical values 353
  - notes 352
  - numbers 352
  - plot area 350
  - recorder range 172
  - references 237, 348-349
  - text 352
  - value axis 350
  - values 23
  - windows 236, 244
  - worksheet names 312
- SELECTION function 242, 353
- Selection reference 242, 353
- SEND.KEYS function 223, 240, 353-254, 373
- Separator line 200
- Serial number
  - conversion
    - to day of month 20, 42, 265
    - to day of week 20, 125-126, 369
    - to hour of day 20, 56-57, 305
    - to minute of hour 20, 82-83, 317
    - to month of year 20, 88, 317
    - to second of minute 20, 106-107, 348
    - to weekday 20, 125-126, 369
    - to year 20, 126-127, 371
  - dates 20, 39-40, 41, 91-92, 265, 319
  - time 20, 91-92, 115, 319, 365, 366
  - tolerance 322
- Series *See* Data series
- SET.CRITERIA function 231, 354
- SET.DATABASE function 231, 354
- SET.NAME function 240, 354-355
- SET.PAGE.BREAK function 235, 355
- SET.PREFERRED function 234, 355
- SET.PRINT.AREA function 235, 355
- SET.PRINT.TITLES function 235, 356
- SET.VALUE function 179, 240, 356
- Shading
  - cell shading information return 294
  - macro sheets border 143
  - worksheet areas 196
- Shadow border 331
- SHIFT key 375
- SHIFT + ENTER 349
- SHIFT + F5 287
- SHIFT + F7 287
- SHIFT + TAB 349, 367
- Short menu display 356
- SHORT.MENU function 230, 235, 356
- Shortcut key
  - command macros running 133
  - duplication, macro order 134
  - key combinations
    - checking 134
    - defining 133, 138, 154, 164, 267
  - listing 134
  - macro sheet, inclusion on 143
  - uppercase/lowercase distinction 134, 165
- SHOW.ACTIVE.CELL function 357
- SHOW.BAR function 199, 204, 240, 357
- SHOW.CLIPBOARD function 237, 357
- SHOW.INFO function 236, 358
- SIGN function 24, 107, 358
- Significant digits *See* Number
- SIN function 26, 108, 358
- Sine 26, 108, 358
- Single-input table *See* One-input table
- Single-stepping 185-186, 240, 361-362
- Size command *See* Control Size command
- SIZE function 231, 358
- Sizing
  - chart objects 283
  - charts 327
  - pie charts 283
  - windows 250, 290, 358
- SLN function 21, 108-109, 358
- Slope of line 68, 69
- SORT function 231, 358-360
- SORT? function 358-360
- Sorting 358-359
- Space
  - deleting from text 26, 120, 366
  - function name, prohibited 4
- Special characters 373-374
- Special functions *See* Information functions; Logical functions; Lookup functions
- Sphere.Surface function macro 136, 158-160
- Split command *See* Control Split command
- SPLIT function 231, 360
- Splitting windows 360
- SQRT function 24, 109, 360

Square root 24, 109, 360  
 Stacked chart 314, 324  
 Standard deviation  
   entire population 19, 25, 46, 48, 110-111, 273, 361  
   sample population 19, 25, 46, 48, 109-110, 273, 360  
 Statistical functions 24-25  
 Status bar  
   command message creating 201  
   display information return 302  
   macro recording display 137, 171  
   message  
     deleting 315-316  
     display 147, 239, 315-316  
 STDEV function 25, 109-110, 360  
 STDEVP function 25, 110-111, 361  
 STEP function 185, 240, 361-362  
 Stepping through macros *See* Single-stepping  
 Stopping macros *See* Macro  
 Straight line *See* Line  
 Straight-line depreciation 21, 108-109, 358  
 Strike through  
   cell information return 294  
   document information return 299  
   setting 281, 282, 341  
 String functions *See* Text functions  
 STYLE function 237, 362  
 Subroutine macro  
   command macro, use as 150, 152, 363  
   control functions 241  
   defined 152, 165, 363  
   entrance point 165  
   error-handling subroutine macro 170  
   exit point 165  
   starting 363  
 SUBSTITUTE function 26, 111, 363  
 Substituting  
   fonts 341  
   formulas 288-289  
   text  
     REPLACE function procedure 26, 102-103, 341  
     SUBSTITUTE function procedure 26, 111, 363  
 Subtracting while pasting 328  
 SUM function 2, 25, 112, 363  
 Sum-of-years' digits depreciation 21, 112-113, 363  
 SYD function 21, 112-113, 363  
 Syntax, function 4-5, 16-17

## T

T function 22, 113-114, 364  
 TAB key 349, 367, 374  
 Table creating 364  
 TABLE function 231, 364  
 TABLE? function 364  
 TAN function 26, 114, 365  
 Tangent 26, 114, 365  
 Task ID number *See* Outside applications  
 TERM function *See* NPER function  
 TERMINATE function 222, 240, 365  
 Testing macros 189-190  
 Text  
   alignment *See* Alignment  
   ASCII code equivalent 25, 36  
   cell formatting, text  
     equivalents 32-33  
   centered 32, 248, 284  
   chart text  
     attaching 253  
     entry 237  
     label base 283  
   checking for 22, 66, 311  
   control characters deleting 25, 35  
   conversion  
     from number 25, 52-53, 279  
     from reference 242, 337  
     to number 26, 122, 367  
     to reference 242, 365  
   data type *See* Data types  
   dialog box text  
     display 212  
     positioning 211  
     text box 208, 213, 215, 216, 217  
   empty text *See* Empty text  
   entering into formula 3, 8, 9  
   extracting characters  
     left 25, 67, 311  
     middle 26, 81-82, 316  
     right 26, 103-104, 344  
   finding *See* Finding  
   formatting 283-284  
   initial capitals 26, 97, 336  
   justifying 311  
   left-aligned 32, 139, 248, 284  
   length 9, 25, 67, 312  
   lowercase, conversion to 25, 77-78, 313  
   macros, use with 221  
   matching 25, 50, 275  
   pasting 327-329  
   quotation mark enclosure 3, 8, 9  
   repeating 26, 103, 341  
   replacing *See* Replacing  
   right-aligned 32, 248, 284  
   searching for *See* Searching

## Text (continued)

  selecting 352  
   spaces deleting 26, 120, 366  
   substituting *See* Substituting  
   uppercase, conversion to 26, 122, 367  
   value translating into text  
     T function procedure 22, 113, 364  
     TEXT function procedure 26, 115, 365  
   vertical text 284  
   writing to file 221, 239, 290-291  
 TEXT function 26, 115, 365  
 Text functions 25-26  
 TEXTREF function 170, 242, 365  
 Tickmark *See* Chart  
 Tilde (~), wildcard character matching 288  
 Time  
   macro running, time specification 239, 321-321, 369  
   serial number 20, 91-92, 115, 319, 365, 366  
 TIME function 20, 115-116, 365  
 Time functions 20  
 TIMEVALUE function 20, 116, 366  
 Tone sounding 238, 254  
 Translating macros 130  
 TRANSPOSE function 24, 116-117, 366  
 TREND function 25, 70, 117-120, 366  
 Trigonometric functions 26  
 TRIM function 26, 120, 366  
 TRUE  
   function *See* TRUE function  
   logical value 10  
 TRUE function 22, 120, 366  
 TRUNC function 24, 121, 366  
 Two-input table 364  
 TYPE function 22, 121-123, 366

## U

Underline ( \_ ), name component 142, 164  
 Underlining  
   cell information return 294  
   dialog box characters 212  
   document information return 299  
   setting 281, 282, 341  
 UNDO function 232, 366  
 Undoing actions 366  
 UNHIDE function 232, 236, 367  
 Union operator 17  
 UNLOCKED.NEXT function 237, 367

UNLOCKED.PREV function 238, 367  
UP key 374  
UPPER function 26, 122, 367  
Uppercase characters  
    case-sensitive searching 52  
    conversion to 26, 122, 367  
    initial capitals 26, 97, 336  
    shortcut key distinction 134, 165  
Useful life 108, 112  
User-defined function *See* Function macro

## V

### Value

absolute value *See* Absolute value  
array *See* Array  
command macro, use in 150  
conversion to text  
    T function procedure 22, 113, 364  
    TEXT function procedure 26, 115, 365  
counting *See* Counting  
data types *See* Data types  
entry in cell 240, 356  
error value *See* Error value  
formatting effect 11  
logical value *See* Logical value  
macro sheet values *See* Macro sheet  
matching 23, 78-79, 148-149, 315  
not available *See* #N/A  
number *See* Number  
pasting 328  
reference *See* Reference  
return values *See* Macro function  
returning 241, 269-270  
selecting 23  
text value *See* Text  
translating  
    into number 22, 89, 318  
    into text *See herein* translating into text  
    no translation 65  
    translating into text  
        T function procedure 22, 113, 364  
        TEXT function procedure 26, 115, 365  
types *See* Data types

### #VALUE!

ADD.BAR function error 245  
AND function error 28  
CELL function error 31  
CHOOSE function error 34

### #VALUE! (continued)

DELETE.COMMAND function error 268  
DELETE.MENU function error 269  
DIALOG.BOX function error 269  
ENABLE.COMMAND function error 274  
EXEC function error 275  
EXECUTE function error 276  
FIND function error 52  
FPOS function error 289  
FREAD function error 289, 290  
FSIZE function error 290  
function macro data type error 178, 179, 180  
FWRITE function error 291  
GET.CHART.ITEM function error 295  
HLOOKUP function error 55  
MDETERM function error 80  
MINVERSE function error 83  
MMULT function error 86  
OFFSET function error 320  
POKE function error 334  
RENAME.COMMAND function error 340  
REQUEST function error 342  
SEARCH function error 105  
TERMINATE function error 365  
VALUE function error 122  
VLOOKUP function error 124

### Value axis

scale setting 347-348  
selecting 350  
VALUE function 26, 122-123, 367  
Value-returning function  
    *See also* Macro function  
    defined 149, 228, 229  
    designated 241-242  
VAR function 25, 123, 367  
Variance 25, 46, 47-48, 123-124, 273, 367  
VARP function 25, 123-124, 367  
Vector 54, 68, 76, 117  
VLINE function 238, 367  
VLOOKUP function 23, 124-125, 368  
VPAGE function 238, 368  
VSCROLL function 238, 368-369

## W

WAIT function 147-148, 240, 369  
Weekday, conversion from serial number 20, 125-126, 369  
WEEKDAY function 20, 125-126, 369

WHILE function 167, 241, 369  
WHILE-NEXT loop 319, 369  
Wildcard character 52, 79, 105-106, 278, 323

WIN.INI file 194

### Window

activating, macro running 322  
changing, macro running 239  
closing 259  
creating 319  
displayed windows listing 369-370  
height information return 300  
hiding 300, 305, 367  
information returning 242, 300-302  
maximizing 249  
menu bar display 204  
minimizing 249  
moving 249, 317-318  
name returning 242, 300  
number 298, 300  
panes  
    activating 243-244  
    edge references 301  
    position information return 300, 303  
    protection information return 297  
    scrolling *See* Scrolling  
    selecting 236, 244  
    sizing 250, 290, 358  
    splitting 360  
    unhiding 367  
    width information return 300  
Window Arrange All command 235, 252  
Window command 235  
Window Hide command 220, 236, 305  
Window Macro command 138  
Window New Window command 236, 319  
Window Show Document command 236  
Window Show Info command 236  
Window Unhide command 220, 236, 367  
Windows applications *See* Microsoft Windows; Outside applications  
WINDOWS function 242, 369-370  
{WINDOWSOFF} *See* ECHO function  
{WINDOWSON} *See* ECHO function  
Worksheet  
    borders 196  
    calculating  
        macro control 154  
        macro sheet differences 139  
    creating 318

## Worksheet (continued)

- display changing 139
- external reference, references to 312
- fonts 281, 298-299
- information 298-299, 301
- layout 326
- left-aligned text 139
- macro sheet differences 139
- margins 326
- marquees 196
- name selecting 312
- pasting text 327-328
- printing 335
- reference *See* Reference
- selecting cells 348-349
- shading 196
- values display 139

## Worksheet function

- argument
  - array, use as 7, 13
  - cell ranges, use as 2-3
  - data types *See* Data types
  - defined 2
  - manual conventions 16
  - nesting functions 10-11
  - number permitted 10
  - omitted argument substitution 17
  - optional arguments 16
  - placeholders 5
  - required arguments 16
  - separators 5, 17
  - translating data types 12-13
  - types 10-11
- categories 18-26
- defined 2
- entering in formulas 4-6
- format *See herein* syntax
- Help request 6, 9
- listing 5, 6, 9, 168
- macros, use in 167-168, 226
- name, spelling check 4
- nesting 10-11
- purpose 2
- result defined 2
- syntax
  - description 4, 16-17
  - multiple forms 5

## Workspace

- creating 370-371
- height 303
- information returning 242, 302-303
- saving 136, 346
- width 303

## Workspace file 136

## WORKSPACE function 235, 370-371

## WORKSPACE? function 370-371

### Writing macros

- description 163, 176
- error handling 158, 170, 274-275
- flow charts 158-160
- planning 157-158
- type of macro 160-161

### Writing over 176

- Writing to file 221, 239, 279, 290-291

## X

- .XLM filename extension 133

## Y

### Year

- conversion from serial number 20, 126-127, 371
- format, text value equivalent 33
- YEAR function 20, 126-127, 371

## Z

- Zero values, display options 272