

# MC100



ALL ABOUT THE  
AMSTRAD NOTEPAD

*Vic Gerbardi  
& Dave Hampson  
Edited by  
Gill Gerbardi*



**MC100 MAGIC**

ALL ABOUT THE AMSTRAD NOTEPAD  
Vic Gerbardi & Dave Hampson, Edited by Gill Gerbardi

**KUMA**





# Amstrad NC100 Magic

by Vic Gerhardi & Dave Hampson

Edited by Gill Gerhardi

ISBN 07457 0236 8



# **Amstrad NC100 Magic**

©1993 Vic Gerhardi & Andy Berry

This book and the programs within are supplied in the belief that the contents are correct and that they operate as specified, but the authors and Kuma Computers Ltd shall not be liable in any circumstances whatsoever for any direct or indirect loss or damage to property incurred or suffered by the customer or any other person as a result of any fault or defect in the information contained herein.

**ALL RIGHTS RESERVED**

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, scanning, recording or otherwise without the prior written permission of the author and the publisher.

Published by:

Kuma Computers Ltd  
12 Horseshoe Park  
Pangbourne  
Berks  
RG8 7JW

Tel 0734 844335  
Fax 0734 844339



# About the Authors, the Book and Acknowledgements

The Authors have been working with small portable computers since 1979. Vic and Gill run Rakewell and Dave helps to run Ranger.

Vic and Gill started writing Z88 Magic in 1989. At the end of writing that book they vowed never to write another one. Since then, we have broken these vows often. They have written several books with Andy Berry, ranging from the Psion Organiser to Windows 3.

This time, however, Vic agreed to write this book without checking first with the other authors, resulting in a colourful start! The outcome was that Gill agreed to EDIT the book. A new member, Dave (who was working on a disk drive for the Notepad) joined the team to write the technical and programming bits.

The authors would like to thank themselves jointly (especially Vic for getting him out of a scrape) and all the people who have helped to put this book together. Jon and Tim at Kurna for being most patient (again!), Cliff Lawson at Amstrad, Douglas Thompson at Arnor and everybody else whose lives have been strained because of NC100 Magic!

Above all we would like to thank you for buying the book. We hope that you will enjoy reading it and will get much pleasure out of using your Notepad.

December 1992

## Other Computing Titles From Kuma:

### IBM PC & Compatible Micros

DTP Sourcebook - Fonts & Clip Art for the PC by J. L. Borman	07457 0030 6
PageMaker 4.0 for Windows by William B. Sanders	07457 0031 4
ZBasic Quick Reference Guide for PC & Mac by John Sumner	07457 0140 X
The Windows Guide Book by Gill Gerhardi, Vic Gerhardi & Andy Berry	07457 0041 1
A Practical Guide to Timeworks Publisher 2 on the PC by Terry Freedman	07457 0147 7
The DR DOS 6 Quick Start Guide by John Sumner	07457 0038 1
Illustrated DR DOS 6 - The First 20 Hours by John Sumner	07457 0044 6
The User's Guide to Money Manager PC by John Sumner	07457 0047 0
Breaking Into Windows 3.1 by Bill Stott & Mark Brearley	07457 0056 X
The Utter Novice Guide to GW Basic by Bill Aitken	07457 0045 4
The Utter Novice Guide to Q Basic by Bill Aitken	07457 0046 2
PagePlus Illustrated by Richard Hunt	07457 0062 4
DOS 5 Quick Start Guide by John Sumner	07457 0054 3

### Commodore Amiga

Program Design Techniques for the Amiga by Paul Overaa	07457 0032 2
Intuition A Practical Programmers Guide by Mike Nelson	07457 0143 4
The Little Red Workbench 1.3 Book by Mark Smiddy	07457 0048 9
The Little Blue Workbench 2 Book by Mark Smiddy	07457 0055 1

### Apple Macintosh

DTP Sourcebook - Fonts & Clip Art for the Mac by J. L. Borman	07457 0050 0
ZBasic Quick Reference Guide for PC & Mac by John Sumner	07457 0140 X
The Quark Book by Rod Lawton & Isaac Davis	07457 0052 7

### Psion Organiser

Psion Organiser Deciphered by Gill Gerhardi, Vic Gerhardi & Andy Berry	07457 0139 6
Using & Programming the Psion Organiser by Mike Shaw	07457 0134 5
File Handling on the Psion Organiser by Mike Shaw	07457 0135 3
Machine Code Programming on the Psion Organiser 2nd Ed. by Bill Aitken	07457 0138 8
Psion Organiser Comms Handbook by Gill & Vic Gerhardi & Andy Berry	07457 0154 X

### Psion Series 3

First Steps in Programming the Psion Series 3 by Mike Shaw	07457 0145 0
Serious Programming on the Psion Series 3 by Bill Aitken	07457 0035 7

### Atari ST

Atari ST Explored 2nd Ed. by John Braga & Malcolm McMahon	07457 0141 8
Program Design Techniques for the Atari ST by Paul Overaa	07457 0029 2
Programming by Example - ST Basic by Dr. G. McMaster	07457 0142 6
A Practical Guide to Calamus Desktop Publishing by Terry Freedman	07457 0159 0
A Practical Guide to Timeworks on the Atari ST by Terry Freedman	07457 0158 2

### Cambridge Z88

Z88 Magic by Gill Gerhardi, Vic Gerhardi & Andy Berry	07457 0137 X
---	--------------

### Games

Sega Megadrive Secrets by Rusel deMaria	07457 0037 3
Sega Megadrive Secrets Volume 2 by Rusel deMaria	07457 0043 8
Corish's Computer Games Guide	07457 0150 7
Awesome Sega Megadrive Secrets	07457 0226 0

### Sharp IQ 7000 & 8000

Using Basic on the Sharp IQ by John Sumner	07457 0034 9
--	--------------

**A selection from our fast-expanding range - latest full details on request**



# Table of Contents

<b>Introduction</b>	<b>1</b>
Welcome	1
Opening the Box	1
What Did We Get it For?	2
New to Computers?	3
Definitions and Tips	3
What to do if You Don't Know What to do?	3
How to Avoid Major Disasters	4
Power to the Notepad	4
Batteries	4
The Main Adapter	5
<b>Chapter 1 The Major Controls</b>	<b>7</b>
The On/Off Switch	7
The Contrast Control Knob	7
The STOP Key	7
The ENTER Key	7
The Secret/Menu Key and Menus	9
Leaving a Menu	10
The Secret/Menu Key and Secrets	10
The Cursor keys	10
The Two Delete keys	10
The TAB key	10
<b>The Driving Controls</b>	<b>10</b>
The Function Key	11
Function Key - Short Cuts	11
The Control Key	11
The Shift Keys	11
The Caps Lock Key	11
The Symbol Key	12
<b>The Main Menu</b>	<b>12</b>
The Information Bar	12
System Settings Menu	13
Power off Delay (mins, 0=Never)	14
Preserve Context During Power Off	14
Sticky Shift Keys?	14
Time Display Format	14
Set Time and Date - The Clock	15
The Options We have Left Out Here	15
The Template	15
<b>Looking Around Your Notepad</b>	<b>16</b>
On the Back Edge	16
On the Sides	16
Getting to the Bottom of it	16
<b>Reset</b>	<b>17</b>
A Forced Reset	17
Resetting Your Notepad by Choice	17
Doing a Reset - Step by Step	17
Soft Reset	17
Why Do a Soft Reset?	18
After a Reset	18
To Find the Version Number	18
<b>Password</b>	<b>18</b>
Setting a Password	19
Secret Document	19
Changing Your Password	19
Resetting the Password (Function <- Del)	20
Locking the Notepad 20	
<b>Chapter 2 The Programs and Their Commands</b>	<b>21</b>
Time Manager (Function T)	21
To Set an Alarm Call (Function A)	22
To Edit Existing Alarm Calls	23
What Happens When You Get an Alarm Call?	23
Editing Time Zones - Function Z	24
BST or GMT	24
Changing the Place Name	24

# Contents

Changing the Time Zone	25
Changing the Time difference	25
The CALENDAR/DIARY (Function C)	26
Deleting a Diary Date	27
What You Can't do in Diary	27
<b>Chapter 3 Address Book (Function D)</b>	<b>29</b>
Add New Address	29
Edit Address	30
Find Address	30
Delete Address	30
Browse	30
Transfer	30
<b>Chapter 4 The Calculator</b>	<b>31</b>
What the keys do	31
Using the Calculator	32
Using a Constant Value	33
Using the Calculator's Memory	33
Percentages	34
Square roots	34
<b>Chapter 5 Word Processing</b>	<b>35</b>
Introduction	35
Why Use a Word Processor?	35
Much More Than a Typewriter	36
The Enter Key	37
Use Tabs to Line Up Your Text	37
Tab and Enter keys	38
Shift Enter and Shift Tab	38
Your First Letter	38
Printing Your First Letter	39
Your First Printout	39
Telling the Word Processor What to do	40
Using Menus	40
The Word Processor Menu	41
The Configure Menu	41
The Layout Menu	42
The Nowhere Commands	42
Command Lines	43
What is a Ruler Line?	44
Turning the Ruler line ON (Control V R)	44
The Default Ruler Line (Control D)	44
Creating a Ruler Line	44
Indenting Text With the Ruler line	46
Copying the Previous Ruler (Control R)	46
The Status Bar (Function 8 or Control V I)	47
Where You Are in The Document	47
Insert on/off (Control 4 or Control Tab)	48
Right Justify (Control J)	48
Word Wrap (Control W)	48
The Time	49
Cap Lock On	49
<b>Chapter 6 Your Second Letter</b>	<b>50</b>
Opening a New Document (Function N)	50
Choosing a Name	50
PC Compatible Document Names	50
Other Things You Can Do to a Document Name	51
Your Letterhead	51
Transferring an Address into a Document	51
Moving The Address to the Right	52
Making Your Master Copy	53
Using Your Master	53
Insert Document (Function 2, select document, I)	53
Continuing the Letter	54
Insert Line (Control 2 or Control I)	54
Right Align	54
The 'R' Stop on the Ruler Line	54
Adding Today's Date (Symbol D)	54
The Rest of the Letter	55
Mopping Up the Format Commands	55
Word Count (Function =)	55



Format text (Control F)	55
Format Complete Document (Word processor menu, Text formatting, F)	56
Centre Text (Control = or Control C)	56
<b>Macros</b>	56
Displaying the existing Macros (Word processor menu, D)	57
Recording Your Own Macro (Shift Control M)	57
Using Your New Macro	58
Macros and Reset	58
<b>Special Characters and Accents</b>	58
Picking Your Special Character (Symbol Menu)	59
<b>Drawing Boxes with Lines and Characters</b>	60
Box Drawing Modes	60
Box Drawing with Single lines	60
Box Drawing with Double Lines (Shift Control D)	61
Box Drawing with Characters (Shift Control C)	61
To Return to Line Drawing Mode (Shift Control L)	61
Lines or Boxes Not Printing?	61
<b>Chapter 7 Page Formatting Commands</b>	<b>62</b>
<b>Headers and Footers</b>	62
Define Header Text and Turn Headers On (>HB'text')	63
Define Footer Text and Turn Footers On (>FO'text')	63
Turn Headers On/Off (>HE ON/OFF)	63
Turn Footers On/OFF (>FO ON/OFF)	63
<b>Page Numbers in Headers and Footers</b>	63
<b>Odd and Even Pages</b>	63
Define Odd Page Header Text and Turn Headers On (>OH'text')	63
Define Odd page Footer Text and Turn Footers On (>OF'text')	63
Define Even Page Header Text and Turn Headers On (>EH'text')	64
Define Even Page Footer Text and Turn Footers On (>EF'text')	64
<b>Designing a Header or Footer Line</b>	64
<b>Print Control Commands</b>	64
Page Number of Next Page (>PN n)	64
Continuous/Single Sheet Printing (>CP ON/OFF)	64
Number of Copies (>NC n)	65
Line Spacing (>LS n)	65
<b>Which Pages to Print</b>	65
Start at Page Number (>SA n)	65
End Printing at Page Number (>EA n)	65
Print Even Pages Only (>PE ON/OFF)	65
Print Odd Pages Only (>PO ON/OFF)	65
Enable/Disable New Page at The End of Printing (>NP ON/OFF)	66
<b>Page Throw (&gt;PA or Control P)</b>	66
Even Page Throw (>EP)	66
Odd Page Throw (>OP)	66
<b>Conditional Page Throws</b>	66
Form feeds Enable/Disabled (>FP ON/OFF)	67
<b>Formatting Commands used with</b>	68
<b>Mail Merge</b>	68
Formatting Whilst Printing On/Off (>FP ON/OFF)	69
Right Justification On/Off (>RJ ON/OFF)	69
Centre Line (>CE'text')	69
<b>Special Hidden Characters</b>	69
View Tabs and Returns (Control V T)	69
Insert Space (Control Space)	69
<b>Special Formatting Characters</b>	70
Soft and Hard Returns	70
Non-break space (Control N space)	70
View Hard Spaces (Control V S)	70
Using Hyphens	71
Non-break Hyphens (Control N -)	71
Soft Hyphens (Control H)	71
<b>Chapter 8 Using the Style Attributes</b>	<b>72</b>
<b>Selecting Bold, Underlining or Italic Print (Control 7, 8, -)</b>	72
Short Cut Style Attributes (Control X)	72
View Codes On/Off (Control V V)	73
<b>Scrolling</b>	73
Vertical Scrolling (Shift Cursor Up or Cursor Down)	73
Horizontal Scrolling	74
<b>Turbocharging the Cursor Keys</b>	74
Page Mode (Control Shift P)	75

# Contents

Goto Line / Page / Column (Control G Ln / Pn / Ca)	75
Markers	76
Absolute Markers (Control @ 0 to 9)	76
Multiple Markers (Control @ 7)	76
Goto Previous Marker (Control Shift 5)	77
Goto Next Marker (Control Shift 6)	77
Goto Margin Markers (Control @ L or R)	77
Goto Block Markers (Control @ I or D)	77
Goto Last position (Control L)	77
The Delete Commands	78
Undelete (Control U)	78
Special Insertion and Deletion Commands	78
Transpose Characters (Control A)	78
Changing Cases	78
Lower to Upper Case (Control /)	79
Upper to lower case (Control \)	79
<b>Chapter 9 Blocks</b>	<b>80</b>
Mark Block (Function 9 or Control Z)	80
Clear Block (Control K)	80
What You Can do With a Marked Block	80
Search, Find & Replace	81
Find (Function 5)	82
A-All	82
B-Backwards	82
C-match Case	82
G-Global	83
W-whole word	83
n-Nth Occurrence	83
Special Characters	83
Using 7 in Find	83
Replace (Function 6)	84
Other Find Commands	84
Find Next (Control 6)	84
Find Previous (Control 5)	84
<b>Chapter 10 Spell Checker</b>	<b>85</b>
What the Spell Checker Does	85
Spell Checker Commands	85
Spell Checker Rules	86
What Happens When the Spell Checker Finds a Mistake?	86
I - Ignore Word	86
L - Lookup Word	86
E - Edit Word	87
S - Store Word	87
User Dictionary	87
View User Dictionary (Word processor menu, Editing, View user dictionary)	87
Remove Word From User Dictionary (Word processor menu, Editing, Remove word)	87
<b>Chapter 11 The List (Function L)</b>	<b>88</b>
Select a Document	89
Document Operations Menu	89
E - Edit	89
I - Insert	89
D - Deleting Documents	90
P - Print	90
R - Rename	90
F - Format Memory Card	90
T - Transfer	90
Showing Other Files	91
Memory	92
Running Out of Memory	92
How Big Can Your Document Be	93
How to Print Multiple Documents	93
<b>Chapter 12 Printing</b>	<b>94</b>
Print Text Page	94
How Wide Can Your Printer Print?	94
Typing a Wider Line	94
Page Layout (Function 7)	95
Getting Rid of All the Margins	95
Fill the Page With Numbers	96
Designing Your Page Layout	96



Test Whether Your Printer Can Print Lines	96
Advanced Printer Commands	98
Printer Codes	98
Different Numbers That Mean the Same Thing	98
Set Double Height Example	99
Escape	99
Output Code to Printer (>OC n)	103
Multiple Commands	103
Printer Code Not Working	104
Hex Dump	104
Lines and Foreign Characters Don't Print	105
Microspacing and Proportional Printing	105
Define Microspace Code Sequence (>MC n n)	105
Define Character Width (> CW)	106
MicroSpacing On/Off (>MS ON/OFF)	106
Proportional Printing On/Off (>PP ON/OFF)	106
<b>Chapter 13 Layout</b>	<b>107</b>
Configure Menu (Function 3)	107
Decimal Character	107
Key Repeat Startup Delay (100th secs)	107
Key Repeat Period (100th secs)	108
Cursor Flash Period (100th secs)	108
View Options Menu	108
Page Layout	108
Page Layout Menu	109
Why Change These Settings?	110
Change the Menu or Use the Commands	110
Template Files	110
Examples of Template files	110
Letter template	110
Fifofax Template	110
Page Layout Commands	111
Top Margin (>TM n)	111
Header Margin (>HM n)	111
Side Margin (>SM n)	112
Odd side Margin (>OM n)	112
Even side Margin (>EM n)	112
Footer Margin (>FM n)	112
Bottom Margin (>BM n)	112
Page Length (>PL n)	112
Zero Margins (>ZM)	112
Miscellaneous Commands	113
Comment Line (>CO text or >>> text)	113
Messages	113
Clear Screen and Display Message (>CS text)	113
Display Message (>DM text)	113
Wait and Display Text (>WT text)	114
Stop Printing and Display Text (>ST text)	114
Insert Document (>IN document name)	114
<b>Chapter 14 Mail Merge</b>	<b>115</b>
Precautions When Using Mail Merge	116
Designing a control document	116
Using Comment Lines and Insert in Mail Merge	116
Testing	116
Check If It Is Worth Doing Mail Merge	116
Keep Your Control Documents Friendly	117
Display Message (>DM text)	117
Where Should You Start?	117
Designing Your Data File	117
Definition - Fields and Records	117
Input for Mail merge	118
Data File	118
Making up a Data File	118
Typing The Address Directly	118
Using Addresses from the Notepad's	119
Building Up The Control Document	120
Defining the Data File	120
Define Data File (>DF filename)	120
Variables	120
What's in a name?	121

# Contents

Variable Name Rules	121
Variables not found - message	122
Reading Variables in the Control Document	122
Read Variables - (>RV name)	122
Read Variables Unconditionally - (>RV name)	123
Read Variables and pad with nulls - (>RV name)	123
Another Way of Using The >RV Command	123
Close Data File - (>CF)	124
Adding Your Letterhead	124
Print the Address to Each Person	124
Print the Rest of the Letter	125
Testing & Putting It All Together	125
First Mail Merge File	126
Testing, Testing, Testing -	126
<b>Chapter 15 Address Labels</b>	<b>127</b>
Practical Printer Problems	127
Positioning the Print on the Label	127
Page Layout for Labels	127
Altering the Page Length to the Length of a Label	129
Line Feed Option	129
Form Feed Option	129
Chopping Long Address Lines	130
More About Variables	130
Ask for Variable - (>AV message var)	131
Restricting the Number of Characters in a Variable	131
Setting Multiple Variables on the Same Line.	131
Set Variables to a value - (>SV name value)	132
Adding and Splitting Variables	133
Adding to Variables	133
Adding Text Variables	133
Adding Number Variables	133
Splitting Variables	134
Chopping Long Address Lines - continued	135
Printing the Label	136
Printing the Single Address Label - Finale	136
Printing Two Labels Side by Side	136
Print Blank Lines	137
Shuffling Address Lines	138
Conditionals	138
The Test	138
The IF Commands	139
Special Condition Commands	141
IF variable is Defined - (ID v)	141
IF variable is Undefined - (IU v)	141
Updating ScrPrintTEST	141
IF Data File is Exhausted (IE)	142
SKip IF Condition True (SK cond)	142
REPEAT .. UNTIL Condition (RP)	142
Nesting	143
Shuffling Address Lines - continued	144
Testing the Shuffle	145
Printing two Addresses continued	145
Additional Files for the Second Addresses	146
Printing the Two Labels	147
Printing the Double Address Label - Finale	148
<b>Chapter 16 Working with Numbers</b>	<b>149</b>
Centigrade to Fahrenheit and vice versa	149
Brackets	149
Minimum and Maximum Numbers	150
<b>Chapter 17 Going Mathematical with Forms</b>	<b>151</b>
<b>Chapter 18 Calculating VAT</b>	<b>156</b>
<b>Chapter 19 Printing to Files</b>	<b>157</b>
Open File for Writing (appending) (>WF file)	157
Writing to File on/off (WF ON/OFF)	157
Write Message to File - used with WF (WM text)	157
Write File Close (WC)	157
Writing a Log	159
Control Document to Add New Addresses to a Data File	159
Converting Other Data Files	160



Conclusion	160
<b>Chapter 20 Transferring Documents</b>	<b>161</b>
Backups	161
No space left	161
Somebody Else Needs It	161
Document Format	162
Transfer Method	162
Setting Up for Transfer	163
What Goes at The Other End?	164
What About a Cable?	164
Making Them Talk	165
Transfer Protocol	165
Plain ASCII Transfer	166
Plain ASCII Send	166
Plain ASCII Receive	166
XModem Transfer	167
XModem Send	168
XModem Receive	168
Address Book Transfer	169
Using a Modem	169
The Memory Card	170
Installing a Memory Card	170
Formatting a Memory Card	171
Using Your Card	171
Moving Documents Using a Card	171
The Address Book on a Memory Card	172
Programming a Memory Card	172
<b>Chapter 21 BASIC</b>	<b>173</b>
Why Bother?	173
What is BASIC?	174
Our Aims	175
The Layout of the Examples	175
A Few Simple Rules	175
A Word of Reassurance	176
Your First Program	176
LIST	177
Line Numbers	177
Commands Used in Your First Program	178
REM	178
PRINT	178
END	178
The Next Step	179
Saving Your Work	179
Variables	179
What To Call A Variable	180
Variables Never Forget	181
Variables on Both Sides	181
Real Variables	182
Integer Variables	182
String Variables	182
The INPUT Command	183
Keep it Friendly - Please	183
The GOTO Trap	184
Halting Gracefully	185
What is Truth?	186
The Question Mark	189
Arrays - Back to Variables Again	190
Numeric Operators	192
The Hidden Array Element	195
Subroutines and Procedures	196
LOCAL Variables	199
Bugs	202
Why Bother With Procedures?	203
Get Rid of the GOTO's	203
FOR..NEXT	204
Getting Out Too Soon	206
REPEAT..UNTIL	207
Loops Within Loops	208
Logical Operators	209
AND	209

# Contents

OR	210
BOR	211
NOT	213
Logical Operator Summary	213
To Continue The Program	213
The Three Modes of BASIC	218
Why Three Modes?	218
Immediate Mode	219
Program Mode	219
Edit Mode	219
Editing Your Program	219
Switching modes	220
Moving On	221
The Hotel Reception Desk	222
The Program Specification	223
Two Dimensional Arrays	223
How do you Write a Large Program?	225
PROC_welcome_screen	227
TAB	227
PROC_initialise	228
PROC_get_decision	228
The ASCII Code	230
PROC_do_input	230
PROC_do_output	231
What is a String	232
PROC_get_digit	233
INKEY	233
PROC_get_room, PROC_get_floor	234
PROC_no_message	234
PROC_message	235
PROC_get_enter	235
Summing Up the Hotel Message Example	235
Playing With Strings	236
LEN	237
Adding Strings Together	237
Subtracting Strings	238
LEFT\$, RIGHT\$, MID\$	238
Comparing Strings	240
Searching Within a String	243
INSTR	243
Changing the Variable Type	246
CHR\$ and ASC	246
Getting Strings From the Keyboard	250
INKEY\$	250
GET\$	250
INKEY\$ Versus GET\$	250
A Good use of GET\$	250
A Good use of INKEY\$	250
INPUT LINE	251
STRINGS and SPC	252
More Maths Functions	253
Trigonometric Functions	253
Pi	253
RAD	254
DEG	254
The six 'Trig' Functions	254
Logarithms	255
Changing Numbers - ABS	255
SGN	255
RND	255
DIV, MOD	256
INT	257
SQR	257
VAL	258
STR\$	258
EVAL	258
Including Information in the Program	259
DATA	259
READ	259
RESTORE	260
More Control Over Printing	261

POS and VPOS	261
Counting Printed Characters	262
WIDTH and COUNT	262
Changing How PRINT Prints - the @% Variable	263
The Effect of the Punctuation - in summary	265
The Comma	265
The Semi-colon	265
The Tilde	266
The Space	266
The Apostrophe	267
Combining Punctuation Marks	267
Print Format Control - the @% Variable	268
STR\$ Format Control - SS	269
Format Selection - NN	269
Number of Digits - PP	269
General Format - PP values	270
Exponential Format - PP values	270
Fixed Format - PP values	271
Zone Width - ZZ	272
A Graphical View of the Screen	272
CLG	273
MOVE	274
DRAW	275
POINT	275
PLOT	275
A First Look at Files	279
Different Types of File Handling	280
OPENOUT	280
CLOSE	282
OPENIN	282
EOF	282
OPENUP	283
BPUT and BGET	283
EXT	285
PTR	286
FN	287
Reentrant Functions	288
Pretend Files	295
Handling Errors	296
ON ERROR	296
REPORT	298
Avoiding the Errors in the First Place	298
Wrapping up the Commands	298
TIME	299
TIME\$	299
SOUND	300
CLEAR	301
OSCLI	301
Key definition	303
The Machine Code Interface	303
PUT	304
CALL, USR	304
HIMEM, LOMEM	304
PAGE	305
The VDU Emulator	305
The Practicalities of Writing a Program	307
Summing Up	309
BBC BASIC Keywords	311
Stored commands	315
Key	315
Paper layout commands	315
Page formatting commands	315
Miscellaneous commands	316
Printer control commands	316
Variable and data input - mail merging	317
Conditional printing and mail merging	317
Key commands	318
Block commands	318
Cursor movement	318
Extra characters	319
Formatting and rulers	320

# Contents

Find and Replace, text	320
Find and Replace, marker	320
Insertion and deletion	320
Miscellany	321
Other commands	321
Printing	322
Spell checking	322
<b>The Printer</b>	<b>323</b>
Why are There so Many Types of Printers?	323
<b>How printers work</b>	<b>324</b>
Fonts and Printer Attributes	324
<b>Different Types of Printers</b>	<b>325</b>
The Dot Matrix printer	325
Daisy Wheel Printer	325
Ink or Bubble Jet Printers	326
Laser Printers	326
<b>Serial or Parallel Connection</b>	<b>326</b>
How to Tell What Type of Connection Your Printer Uses	327
Getting the Correct Cable	327
The Trouble with Serial Printers	327
Mix and Match	327
Parallel Printer	328
Printer Type Setting	328
Emulations	328
<b>It's Not That Bad</b>	<b>329</b>
<b>ASCII Table</b>	<b>330</b>
<b>Diagnostic Test</b>	<b>331</b>
<b>Useful Names and Addresses</b>	<b>334</b>
<b>Index</b>	<b>336</b>

# Introduction

## Welcome

Welcome to the Notepad Magic book. We are ready and waiting to guide you through the magic maze. Although the Notepad is billed as easy to work with there are many dark alleyways for you to get lost in when you start exploring deeper. With our help you will find the way through and get much more out of this exciting computer.

Don't be deceived by the small size of the Notepad. Targislike, its power and abilities bear no relation to its size. It allows you to take your computer with you (without requiring the skills of a weight lifter) and use it at meetings, libraries or just out in the garden.

What do we hope to achieve with this book? We will show you round this magical little computer and tell you what you can use it for in your everyday life. We also hope we can save you time by showing you the quickest and easiest ways of doing things. Above all we hope we can convince you that using your Notepad is fun. We must stress, however, that the Notepad can only do all this for you if you USE it. The number of people who buy computers and then leave them sitting on the shelf is depressing when you know what they could have achieved with them.

The publicity about the Notepad is right, it is easy to turn it on and use it for the first five minutes. However, when you want to do more complex tasks with your Notepad the steps you need to take are not always so obvious. This book will help you over those hurdles helping you get the most out of your Notepad.

## Opening the Box

We were excited when we got our Notepads. We took them home and opened the box straight away. The first thing we saw was the large notice stuck on the front.

**PLEASE DO NOT ATTEMPT TO OPERATE THIS MACHINE WITHOUT REFERENCE TO THE FIRST SECTION OF THE INSTRUCTION BOOK.**

We headed back to the box and unearthed the book. After reading the first section we fitted the batteries. Turning it On, to our delight it burst into life. Then we had to ask ourselves a very serious question.



# What Did We Get it For?

We had had all sorts of ideas in the shop; now what were they? Oh yes, we were going to take it into the garden and write to all those people that we've been meaning to write to for ages. The fact that we have already got big computers sitting on our desks doesn't make any difference. When the weather is good, who wants to be stuck indoors anyway? Being able to work outside, however, might just be the inducement we need to get down to those letters.

Another thing we thought we could use it for was collecting the addresses of our friends. They all keep moving and changing their names, not to mention telephone numbers. My address book is beginning to look like a medieval paper patchwork that would probably get prizes as an art exhibit. Keeping them all on the Notepad would mean that we could change them as much as we like (or at least they could) and we would still have a tidy address file.

We also wanted to use the Notepad's mailmerge facility. That would put the lucky friend's address into the top of the letter without us having to retype it. That sounds like a very good idea to us. Mailmerge would also print all the address labels for our Christmas card list. Our arms feel as if they are going to drop off after writing a hundred or so envelopes. Using the Notepad we will only have a sticky tongue, from putting the stamps on, to contend with.

What about Time Management we hear you ask? Oh, very well then, the Notepad's Clock, Diary and Alarm will also be useful. Our last New Year's resolution was that we were going to be more organised and more punctual for appointments. The Notepad wasn't there then, it is now. Watch this space.

One of us almost fell out of bed the first morning we used the Notepad's Alarm. It was lucky that they couldn't find anything to throw at it; it may not have survived. The alarm is LOUD, unlike some alarm systems that we have been able to sleep through.

There are other things in the Notepad that support the Word Processor which will also be useful. These are: -

- The Spell Checker for removing most of your typing mistakes
- The List for displaying the names of the documents you have written
- The Terminal for sending documents to another computer
- The Printer Server for putting your documents onto paper

There is also a totally separate BBC BASIC program on board your Notepad. This enables you to program your Notepad. You may not be a computer programmer now but

if you want to make your Notepad do even more useful things for you, this book will enable YOU to write programs. It might not be able to make the tea in the morning but the Notepad's magic will rub off on you. More of that much later, however you have to learn to tell one end of the broomstick from the other before you can go soaring over the rooftops.

Above all, we hope you will enjoy getting to know what the Notepad can do for you.

## **New to Computers?**

If you are new to computers don't worry. We were new to computers once and can remember how daunting a prospect it seemed. We will do our best to explain everything as simply as we can.

There are a few points that we would like to tell you about before we start delving into the depths of your Notepad.

## **Definitions and Tips**

As with all subjects, people in the computer field invent new names for the things that they are working with that ordinary people cannot understand. Although the boffins would probably argue with us, we reckon they do this so that they can make everything look more complicated than it is.

An everyday example of the sort of language boffins use would be if you have to go to see a doctor with a bad toe. The doctor would call it a fancy Latin name while to you it is still a bad toe.

Working with computers is very similar to that. To get over the problem, when we come across a computer related word, we will explain what it means in plain English. These will be called DEFINITIONS.

We will put additional or important information into TIPS. These are buried in the book and are listed separately in the Index. This means that if you want to reread them again later you can look them up instead of thinking "I know it's in here somewhere, but I can't find it."

## **What to do if You Don't Know What to do?**

If you don't know what to do with a computer DON'T DO NOTHING. Many people worry that they will do some damage to their new computer if they do something wrong. The good news is that you CANNOT damage any computer physically by pressing the

## Introduction

wrong key or selecting the wrong function. The worst thing that can happen is that you lose everything that you have typed into it so far. You can always return your Notepad to the state it was when you first took it out of the box.

## How to Avoid Major Disasters

The best way to avoid losing everything in your Notepad's memory (your life's work) is to keep a copy of it. The copy is called a backup.

---

### Definition - Backups

**A backup is a copy of some or all the information that you have typed into your Notepad.**

---

You can copy the information in several different ways. You could print it out on paper but you would then have to type it all back into your Notepad. It is better to copy it to a memory card or disk, or transfer it to another computer. You can then copy it straight back into your Notepad's memory if the need arises. You could call Backup your anti Amnesia Spell!

Although losing everything in memory is rare, backups are important. You should regard them like insurance policies. We will cover the subject in more detail later.

## Power to the Notepad

You can run your Notepad on batteries when you are out and about but when you are near a power socket it makes sense to use the mains adaptor.

### Batteries

The Notepad uses two totally different sorts of batteries simultaneously. The first sort is the 'AA' size Alkalines that you have just put in. You can use rechargeable Nicads in place of these if you want to cut the cost of batteries. The second sort is a Lithium battery that you may not have come across before. The reason the Notepad has two sets of batteries is to minimise the chance of it losing your information. To keep your information safe, the memory of the Notepad always requires power. It's a bit like our short term memory. If you are trying to remember a number or something but get distracted, you are likely to forget what you were trying to remember. If the Notepad runs out of power its memory forgets everything.

The lithium battery provides power to the memory when your main batteries have run down. You can't do anything else with your Notepad until you have changed the 'AA'

batteries. The lithium battery is only there to power the memory. It is supposed to last up to five years. We haven't checked that, however, as we have not had our Notepads that long yet.

The worst scenario that could happen would be both types of battery going flat simultaneously. You would then lose everything in memory. This would be extreme bad luck but there is actually no way you can ensure that it won't happen. One reason backups are important.

You must replace either type of battery when you get a battery low warning sign on the screen. You cannot do anything with your Notepad until you have changed them. We recommend that you get at least one spare set of 'AA' batteries and a lithium cell to keep with your Notepad. Our local camera shop has the lithium ones in stock, so there is a good chance that you probably could get one locally as well. With spare batteries to hand you can still carry on working with your Notepad even if the batteries run down while you are in the middle of nowhere.

We are impressed by how long a set of batteries lasts.

---

### TIP - Changing Batteries

When changing the 'AA' size batteries you should make sure the ends are clean so that they make good contact with one another. A lint free clean cloth - a shirt sleeve - will do. Sometimes dirt on the ends of the batteries can stop the current from flowing.

**You should avoid touching the ends of any batteries with your fingers since any sweat can lead to contamination and corrosion later.**

---

## The Mains Adapter

If you are using a mains adaptor always use the one that comes with the Notepad. Don't be tempted to use a different one. There is no standard for power supplies and using the wrong one with your Notepad is a sure way of generating expensive smoke! If you have several mains adapters, it might be a good idea to get some coloured tape and colour code the plugs. You can then tell which one to use with what piece of equipment.

---

### TIP - The Hidden Fuse

The Notepad has an inbuilt fuse fitted on the main electronics board. This is not easy to change. If you manage to blow it using the wrong mains adapter you will lose all your files. It will appear that you have done your Notepad a lot of damage

## **Introduction**

**because nothing will work. Once the fuse is changed, however, it should be all right.**

---

If you are going to another country, leave the mains adapter at home and take a supply of spare batteries with you, including a spare lithium one.

Let's get to know some of the keys and the Contrast Control knob.



# Chapter 1

## The Major Controls

Unlike the Notepad manual, we are not going to call the multifunction keys by all their names at once. For example, we will call the red key labelled 'Left Arrow' and 'Word' simply the Cursor Left key. We are assuming that you have already put in the batteries. This section will introduce you to the important keys so that you can start using your Notepad.

### The On/Off Switch

The On/Off switch is like the bathroom light switch. You press it to switch the Notepad on. You press it again to switch the Notepad off. If you leave your Notepad switched on without pressing a key it will switch itself off automatically, after a set period, to save battery power.

### The Contrast Control Knob

After turning Notepad on, the screen maybe completely black, completely white or somewhere between the two.

At this point you may need to use the Contrast Control knob that is on the right-hand edge of the Notepad. Its function is similar to the brightness control knob on the television. Adjust this so that you can read the screen comfortably.

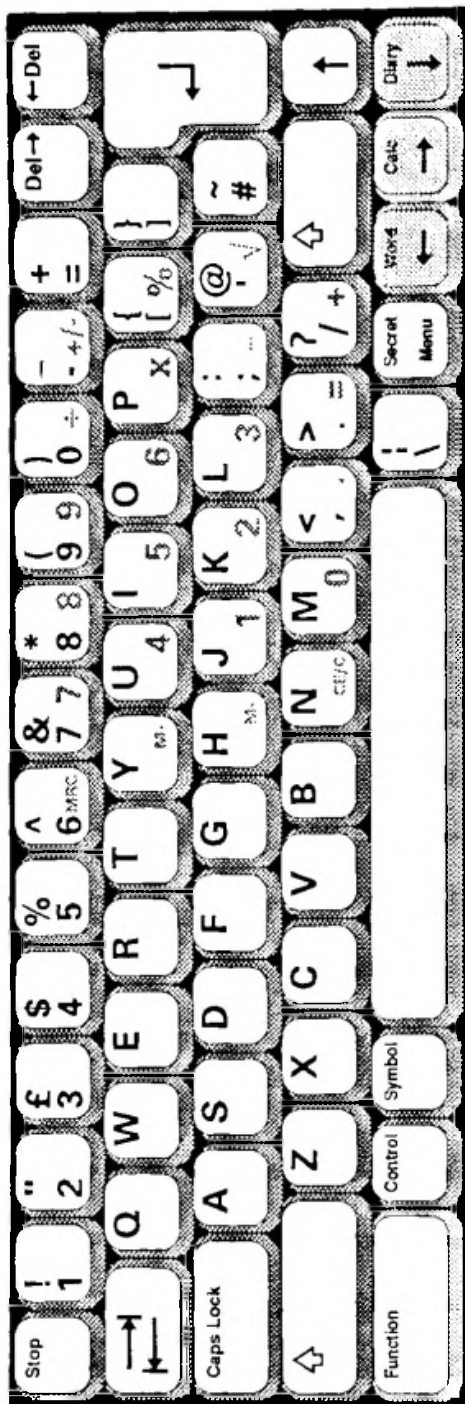
### The STOP Key

If you have learnt how to drive a car, you may remember being taught how to stop early in your first lesson. We thought we would do the same thing here. The key in the top left-hand corner of the keyboard is the STOP key. You use it to stop whatever you are doing at that particular moment.

You can also use it when you are somewhere in the Notepad that you don't want to be. Remember what we told you earlier about doing something when things are not going quite the way you want them to go? Well pressing the STOP key (several times) is something you can do because that will return you back to the beginning.

### The ENTER Key

The instruction book calls this 'the key with no name'. You will remember it quickly



because you are going to use it a lot. It is normally called the ENTER key - it's the large one on the far right of the keyboard.

You should regard this as the opposite to the STOP key. We wish we could call it a GO key, but sometimes when you use it you may not GO anywhere.

## The Secret/Menu Key and Menus

This key displays different menus or allows you to enter your secret document. We know what some readers may be thinking here. What is the lunch menu doing in a Notepad?

---

### Definition - Menus

**A menu is a list of choices that are available for you to choose from.**

---

A menu on a computer is like a menu in a Cafe. Normally you can't ask for anything that isn't on it. However, the Notepad is different because some of its commands are not on menus at all. We will cover these later.

There are several different ways to select an item on a menu. You can either move around the menu with the cursor keys and then select the command you are on by pressing ENTER, or type the first letter of the line containing the command.

---

### TIP - Short Cut Commands

There are short cut ways of selecting some commands that you can use without going through a menu. These are normally a combination of the Control key and one other key. Any command with a short cut is listed in the menu with its key combination written in **BOLD** text.

---

### TIP - Altering Settings in a Menu

Many menu entries have a choice. Some allow you to type in the number you want while others display all the various options that are available. If, when you select a menu option, a grey rectangle appears on the end of the line, it means that you can type something into it. If there is something already there, you will need to delete that first with the Del key before typing in what you want. When typing in numbers, you will need to use the white number keys. If on the other hand several different possible settings appear you can select the one you want with the cursor and ENTER keys.

If you use the wrong key nothing will change. So don't be frightened, try different

keys until something works.

---

### Leaving a Menu

As always with the Notepad if you want to stop looking at the menu press the STOP key.

### The Secret/Menu Key and Secrets

The Secret key gets you into a secret document that is protected by a password. You can also use the Secret key to set a password to protect everything in your Notepad. There will be more about this later.

### The Cursor keys

These are the arrow keys. They move a flashing square (called the cursor, details coming up soon) around the screen. This is really magic because it looks as though it works by remote control. Among other things the cursor keys are used to select items or change numbers and when used with other keys they can even start programs running.

### The Two Delete keys

Just to keep those mortals happy who make typing mistakes, there are two delete keys. Gobble left and gobble right. These remove characters either to the right or to the left of where the cursor is.

### The TAB key

For users who have not come across a TAB key before, this causes the cursor to 'jump' to a defined vertical stop on the screen. We will cover this in greater detail in the word processing section.

## The Driving Controls

We have already introduced you to some lesser known keys like the STOP, ENTER, and cursor keys. Now it is time to have a look at the rest of them in turn.

The next group of keys are special keys. Like the Shift key on a typewriter, you need to press them first and then press another key as well. Unless you have turned on the 'Sticky shift keys' from the Systems Settings screen, pressing these keys on their own has no effect. Their purpose is to alter the result of pressing other keys.

## The Function Key

The Function key starts different programs as well as selecting some commands in the Word Processor. These commands are shown on the template.

You may have noticed from the Main Menu that you can start three different programs by pressing the Function key and one other key together.

Pressing Function and Cursor Left will start the Word Processor.

Pressing Function and Cursor Right will start the Calculator.

Pressing Function and Cursor Down will start the Diary.

## Function Key - Short Cuts

Pressing the Function key with other letter keys is a short cut way of selecting the different programs available in the Notepad.

The disadvantage of these short cuts is that you need to remember the key combinations. That is not easy to start with, but you will quickly start remembering the ones you use most often. The other combinations will come as you venture further into the labyrinth. So don't panic, you will pick it up as you go along.

## The Control Key

Although this is a boring black key it behaves like the Function key to provide you with commands inside programs. We will cover these in the sections about the individual programs.

## The Shift Keys

This key allows you to select Capital letters for all the alphabetic keys and the upper white characters on keys that have two different white characters.

## The Caps Lock Key

If you wish to write in CAPITAL LETTERS ALL THE TIME, you will find it easier to press this key once rather than always holding down the Shift key. If you press Caps Lock once it will turn Caps Lock ON. To turn Caps Lock OFF again you just press the key again.

If you have Caps Lock turned ON, the Shift key will do the opposite to normal, that is, it



will give you lower case characters.

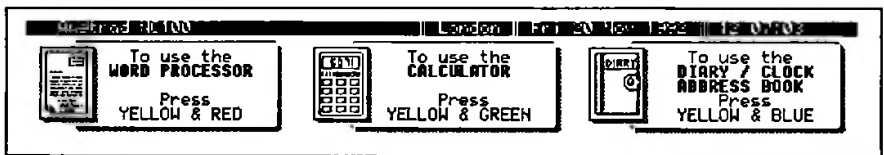
You can see when you have got Caps Lock turned ON because there will be a capital 'C' on the right of the Information Bar.

## The Symbol Key

Although the Symbol key is another black one, we reckon it is in disguise because this one can control some really powerful magic. It can start Macros or select special symbols. Again we will be covering this one in detail later but at least you know that they all do something.

Just in case you were wondering, you use the green numbers and letters on the keyboard with the Calculator.

## The Main Menu



When you first start your Notepad you normally see the Main Menu. To get there from anywhere else in the Notepad all you need to do is to press the STOP key several times. There are two things that you should notice here.

## The Information Bar

Most of the different programs that make up the Notepad display a black bar along the top of the screen. As well as telling you which program you are running, this bar gives you a range of other information. When you are in the Main Menu, this bar shows you:

Amstrad NC100	-	(you can't change this)
London	-	where your Notepad thinks you are
Today's date	-	the date that the clock has been set to
Current Time	-	a clock displaying the Time
A Capital 'C'	-	if the Caps Lock is set to ON

The remainder of the screen is taken up by three large pictures. These illustrate three things that you can choose to do from the main menu. They do not show you everything

that you can do - just what the designers thought you were going to use most frequently.

We will cover all the programs shortly.

## System Settings Menu

Let's change a few items in our System Settings menu. This will do two things. First, it will show you what you can change straight away and secondly it will give you immediate results. If you don't like the settings we've suggested, you know what you can do.

That's right, change them back again. What did you think we were going to say?

```

Power off delay (mins, 0=Never)      (5)
Preserve context during power off    (No)
Document sizes and date display     (dd/mm/yy)
Document transfer port and Format    (Serial/Protect)
Sticky shift keys?                  (No)
Time display format                   (24 hour)
Set time and date
  
```

To get to the SYSTEM SETTINGS Menu, turn your Notepad on and press the STOP key a few times to take you back to the Main Menu screen. Then you just press the Secret/Menu key.

The settings that you see on the screen are the Default settings.

---

### Definition - Default

When you start working with a new computer, there are many things that you can change. The programmer will have put in some settings to enable you to see the whole thing working. These preset settings are the 'Default'. Often the default choices will be the best ones, but this should not stop you from changing them.

---

### TIP - Numbers and Letters are Different

Unlike a typewriter, numbers and letters mean different things to a computer like the Notepad. A few typists have got into the habit of using a lower case 'L' for the number one, and an upper case 'O' for the number zero. If you are one of these, break the habit now. It causes misunderstandings between you and your Notepad. They may look the same but they have different meanings.

---

### Power off Delay (mins, 0=Never)

The power off delay is the time that the Notepad will allow you to leave it doing nothing before turning itself off automatically. The default is five minutes. Let's change it to one minute to save even more battery power.

When you first enter the System Settings menu, the cursor (the black square) will have a white number five in it. It should be sitting at the end of the Power Off Delay line; that is just where you want it to be. Pressing the <Del key will delete the number five. If you then press the white number '1' key, you should see the number you have typed in place of the original number five. There is one good thing about using the Notepad; you can always change your mind. Nothing is cast in concrete.

Does everything look all right? Then press the STOP key to leave this menu or use the Cursor Down key to select the next item in the menu.

### Preserve Context During Power Off

This function isn't the name of a type of jam. It sets whether, when you turn the Notepad on, you wish to return to the Main Menu or to where you were when you last turned the Notepad off. We prefer returning to where we were before we switched the Notepad off so we have changed this with the Cursor Left key from No to Yes.

### Sticky Shift Keys?

When we first saw this function we thought we knew what the jam did, but we were wrong. Some keys on Notepad like Function, Shift and Control require you to press them while pressing other keys as well. If you prefer typing with one finger you need to set the Sticky Shift Keys option to 'Yes'. That will allow you to select one of these keys first followed by another key without having to hold them both down together.

Although this is called Sticky Shift Keys it should be called 'Sticky Function, Control, Symbol and Shift keys' to cover all the keys it affects.

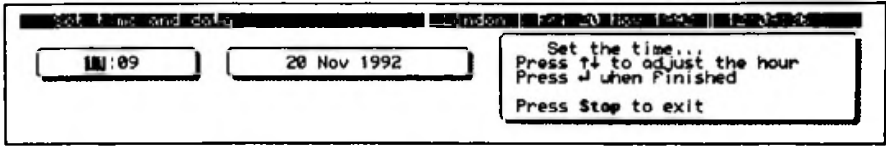
### Time Display Format

If you wish to change the displayed time from a 12 to a 24-hour clock you can select it with the left or right cursor keys.

We wish we could change this to a 72-hour day. That way we could write a lot more in a day! But not even our magic is strong enough to slow the sun down.

## Set Time and Date - The Clock

This is the first menu that you see when you first run your Notepad after fitting the batteries and switching it on.



It's easy to set. Use the up and down cursor keys to change the number where the cursor is and then move the cursor left or right and alter the other numbers. When you've correctly set up the time and date, use the STOP key to leave the screen.

You may be wondering why you should have a clock in the Notepad. In fact the clock is used frequently. It is permanently on display when you are in the Word Processor but it is used in other places as well. It is used by:

- The List (stored documents) - to display when you saved your files
- The Diary - to check if you have any appointments for today
- The Alarm - to blast you at the appointed time
- Time Zones - to change your time when you are on the move

You need to make sure that the clock shows the right time and date. If you don't, it will make your Notepad much more difficult to use.

## The Options We have Left Out Here

There are two functions on the System Settings Menu that are a bit too advanced to describe here. For details of Document Date Display option which effects the document sizes and date display in the List, see the List. For details of Document Transfer Port and Format see file transfer.

## The Template

Beneath the screen you will find the template. While you are finding your way around your Notepad this will remind you of some of the keys to press, particularly when you are using the Word Processor.

# Looking Around Your Notepad

We have looked at the top of the Notepad and the keys. Now let's go round the other edges and discover what is there too.

## On the Back Edge

Looking at the back edge there are three sockets on the left hand side. The first one is for the Mains Adapter. The second socket is a nine pin plug labelled Serial. You can use this to connect the Notepad to another computer, a modem, an external disk drive or a serial printer.

The last socket on the back is a 25-way Parallel socket for connecting to a parallel printer.

---

### TIP - Get a Printer

If you haven't already got a printer we recommend that you get one as soon as possible. No computer is much good without a printer because everything put into it can only stay in it. If you have a printer you can get your writing onto paper so that other people can read it. The only way without a printer would be to post your Notepad to your friends, which might get a bit expensive.

---

## On the Sides

On the Left side there is a slot for a memory card. Memory cards give you extra space to store documents or allow you to run additional programs written for your Notepad.

You will already be familiar with the screen contrast knob on the right side.

## Getting to the Bottom of it

Underneath the Notepad there are the battery compartments and two fold down feet. The feet will save you from using the manual to prop up your Notepad so that you can see the screen easily. There is also a 'hidden door' on the bottom with a screw in it but don't bother opening it and having a look. All that is behind it is the program chip that controls your Notepad. If Amstrad make any amendments to the programs that run your Notepad this is where you change the chip.

# Reset

This is a good time to introduce the Reset functions. There are two different sorts of reset, a hard one and a soft one. A hard reset allows you to return your Notepad to the state it was in when you got it out of its box. It will clear the Notepad's memory and return all settings back to the default values, except the time. A soft reset, on the other hand, will destroy the last document that you were working with but otherwise will leave your memory alone. It returns all the settings to their default. You could do a reset by choice or you could have to do one.

## A Forced Reset

Very occasionally computers can get themselves tied up in knots so that they don't know where they are. This happens less with the Notepad than with many other laptops. But if your Notepad does start doing strange things, when it had been perfectly well behaved before, a reset might put things back to normal.

## Resetting Your Notepad by Choice

You can do a reset before lending your Notepad to a friend; to stop them being nosy and looking at your documents. But be careful because it will stop you looking at them too as they will no longer be there. Don't do a hard reset unless you have a backup.

If you haven't got any documents yet, why not try this out now. Press the Function, STOP and <-Del keys together while turning the Notepad ON - resetting Notepad is a bit like a playing chord on a piano.

## Doing a Reset - Step by Step

OK. We'll talk you through it in sequence. The Notepad should be OFF before you start. With your left hand, press the Function and STOP keys; with your right hand, press the <-Del key and the ON/OFF switch. You must press them together. If you run out of fingers, you'll have to press the ON/OFF switch with your nose! Your reward, should you get this right, will be a loud LOW BEEP to show that you have successfully reset your Notepad.

## Soft Reset

This is a less dramatic reset function, which only resets the System and Macro settings. To do a soft reset press the Function and STOP keys together, while turning your



Notepad ON. A soft reset is very likely to destroy the last document that you were working with, although the rest of your memory should be left intact. Your Notepad will give you a loud HIGH BEEP this time, so be prepared.

### Why Do a Soft Reset?

There are two scenarios where a soft reset might be useful.

First, if you had changed some settings in your Notepad but didn't like the effect the changes had made, a soft reset would restore the default settings for you.

Secondly, if you have to do a reset because of your Notepad's erratic behaviour, it might be better to try a soft reset first before you do a hard one.

### After a Reset

If you have used any of these Reset functions, it may be wise to change the 'System Settings' back to those that we changed earlier.

### To Find the Version Number

On the subject of pressing keys as you turn your Notepad on, you can find out which software version number you have by another combination of keys. Pressing the Function, Symbol and ON keys together starts an inbuilt diagnostic test that tests various parts of your Notepad. What these tests do is covered in the Appendix.

The first screen you should see is the 'Press a key to start diagnostic test (Stop to skip) . . . ' screen. Our Notepad has 'ROM v1.00 (c) Amstrad 0 Apr 1992, written by Gavin, Mark and Ray at Armor' on the bottom of the screen. You can leave this screen by pressing the STOP key to return to the Main Menu.

## Password

You can stop anybody (and yourself if you forget your password) from prying into the information stored in your Notepad. The same password offers two levels of protection that you can use.

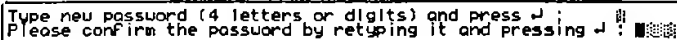
You can use a password to protect the whole Notepad. If you turn the Password Lock on before you turn your Notepad off you would need to use the password when you switch your Notepad on again. If you don't need to protect everything in your Notepad however, you can just set a password to protect one secret document.

It is very important that you set a password, even if you do not intend using either the lock or the secret document. There are two reasons for this. The first one is that if you don't set it someone else can. If you don't know what the password is you can't use your Notepad.

The second reason why you need to set a password is that you can select the Password Set Screen without meaning to. Once you have selected it, you cannot return to anything without typing in a password. So you might as well do it before you start.

## Setting a Password

You enter the password setting screen by pressing the Control and Menu keys together. You can type in up to four characters and they are not case sensitive. That means that you can either type your password in capital or lower case characters, it does not matter. You need to type in the password followed by the ENTER key twice to confirm that you know what it is.

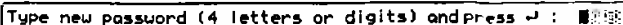


## Secret Document

You can enter your secret document by pressing the Function and Menu keys together. You will then need to type in your password. The secret document has no name and you can only have one of them. It is a cut down version of the Word Processor that we will be covering later. The information always stays in memory. It is never saved as a document.

Use the STOP key to leave the document.

## Changing Your Password



You can change the password that you have set by pressing the Menu key from inside

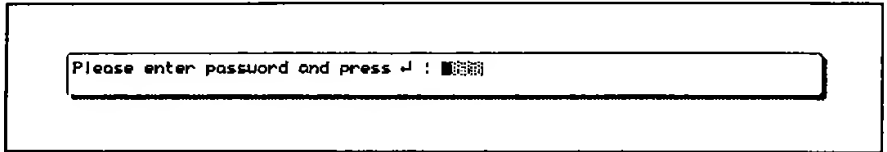
your secret document. You will need to type in the old one first, followed by the new one twice.

### Resetting the Password (Function <- Del)

If you want to reset your password, select either the Password or Secret Document screens and use Function <- Del. This will give you a warning message and if you carry on your secret document and password will be deleted.

### Locking the Notepad

You can stop your Notepad from being used by anyone else by selecting your password before switching the Notepad off. When it is switched on again, the password needs to be typed in before you can use your Notepad.



# Chapter 2

## The Programs and Their Commands

Now that you have been introduced to the Notepad's driving Controls, it is about time you took it out for a run. We will start with the easy programs and leave the more difficult ones until later.

We know what you would really like to do is to speed down the motorway and use the Word Processor straight away. However, if you follow the procedure covered here, everything that needs to be set will be. And in addition you will understand some of the terminology used.

To help you, we've put the quick key strokes with the heading. You can find whatever you are looking for more quickly.

### Time Manager (Function T)

The Time Manager Menu has six preset time zone areas, with their time differences, which can be changed. It also allows you to set up to six Alarm calls. The Alarms can occur at any time on any day. They can display a message and be set to happen once only or repeat every day.

To select the Time Manager Menu from the Main Menu you press the Function and Cursor Down keys together to select the Diary Menu and then press the Cursor Down key again to select the Time Manager.

Alternatively you can use the short cut key sequence Function T from any other menu.



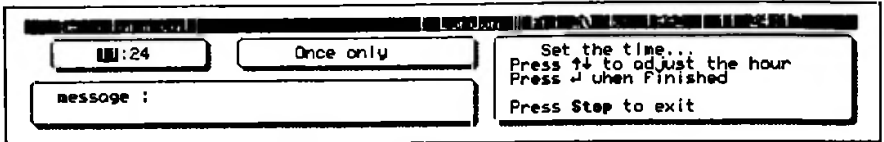
You have three choices from this menu. These are:

- To set an alarm call - press the Cursor Left key
- To edit an existing alarm - press the Cursor Right key
- To set the time zones - press the Cursor Down key

## To Set an Alarm Call (Function A)

To Set an alarm call from the Main Menu you use Function and Cursor Down, Cursor Down to select the Time Manager Menu and then Cursor Left.

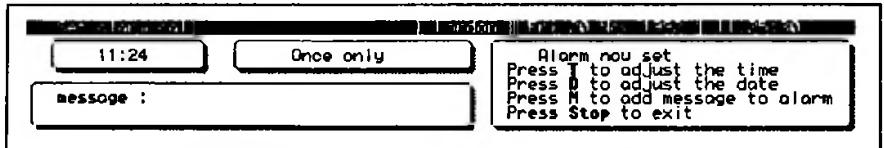
Alternatively you can use the short cut key sequence of Function A from any other menu.



Note that if you have all your six alarms set, when you select this menu you will automatically be given the next menu ('To edit existing alarm calls') instead. This allows you to delete one alarm first before setting a new one.

It is quite easy to set an alarm, since many commands are the same as we have already covered.

First, set the time of the alarm with the cursor keys. Up/down keys again change the numbers while the left/right keys select which number to change. When you have set the correct time press the ENTER key. The alarm is now set and you have more choices.



These are T to adjust the time, D to adjust the date, M to add a message to the alarm and STOP to exit.

T changes the time again. Perhaps you made a mistake the first time round.

D changes the 'Once only' to 'Repeats every day' or by using the up/down keys change the date that the alarm will occur on.

---

### TIP - Alarm - Once You Press D You Lose 'Once Only'

**Once you press D you will lose the 'Once only' option for good on that alarm call. The only way to get it back is to delete the call and set another alarm.**

---

You can use M to display a message with your alarm call. A message length of up to forty-two characters can be stored here to remind you what you thought you ought to be

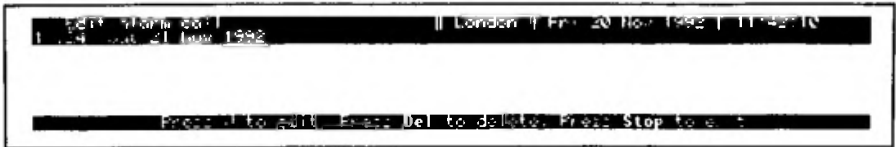
doing when the alarm goes off.

Finally STOP leaves the Menu, leaving the alarm set.

### To Edit Existing Alarm Calls

You can edit an existing alarm call straight from the Main Menu. Just press the Function and Cursor Down keys together, then the Cursor Down again (to select the Time Manager Menu ) and then the Cursor Right key.

There is NO short cut key sequence to get here.



Note that if you have no alarms set, you can't enter this menu.

Edit alarm call displays a list of your set alarms. There are three things you can do here and these options are also displayed along the bottom of the screen.

To use these commands, you should first use the cursor keys to highlight the alarm call you want to alter. You can then edit the alarm by pressing the ENTER key. This will give you the familiar Set Alarm call Menu that you set it with originally.

To delete an alarm, select the one you don't want any more with the cursor keys and then press either Del key. You will need to confirm that you really do want to delete it by pressing 'Y'.

Again, STOP allows you to leave the program.

### What Happens When You Get an Alarm Call?

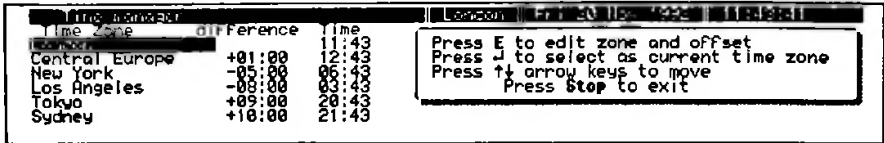
Whether your Notepad is on or not, the Alarm will turn it on. It will display the alarm time complete with the message (if you set one) and beep for a minute. If you ignore the alarm, the Notepad will turn off. When you switch your Notepad on again, the alarm will be displayed, showing you that you have missed an alarm. If the alarm call was not a 'Repeats every day' one, it will be automatically deleted from your 'Edit alarm call' list.

## Editing Time Zones - Function Z

The Time Zone option allows you to change the time quickly on Notepad when you are jetsetting around the world. Or, if you just want to telephone your friends in another country, the time manager will show you what time it is over there. There should be no excuses for getting them out of bed in the middle of your day.

To list the existing time zone information from the Main Menu you press the Function and Cursor Down keys together, and then Cursor Down again twice.

Alternatively you can use the short cut key sequence of Function Z from any other menu.



### BST or GMT

You ought to be aware that all the time differences set on Notepad are based on London being at GMT (Greenwich Mean Time). This is fine for half the year but what should you do when we are on BST (British Summer Time) that is one hour ahead of GMT?

There is no easy answer except to be aware of the differences. Other countries may have summer and winter times as well that may change on different dates.

### Changing the Place Name

The default setting for place name is 'London'. We're not in London. We are at the publisher's address in Pangbourne. Let's change this setting; why should London have all the glory?

If the time where you are is the same as London, then just change the place name from London to where you are. To change this, do the following.

The cursor should be already over London. Press the E key (to edit zone and offset). The rest of that area should turn grey.

Do you remember about grey areas? That is right, you can type in them.

Pressing the STOP key deletes London and leaves you to type where you are. In our case we'll type "Pangbourne - Vic." You can use the Del keys and the cursor keys to correct

any mistakes you make. When you are happy with your entry, press the ENTER key to return you back to the Time Manager Menu.

Now you can press the STOP key again several times until you are back at the Main Menu and there, on the Information Bar, is your new entry. No longer London, it is the place where YOU live.

### **Changing the Time Zone**

If you are in New York's time zone for example, then you should first let Notepad know which time zone you are in before changing the place name. Let's pretend that you are in Boston to illustrate how this is done.

Select the Time Manager Menu, then move the cursor down to New York with the Cursor Down key. Then press the ENTER key (to select it as the current time zone).

You should now see that all the time differences will have been recalculated to what they should be from New York. Both the place and time shown on the Information Bar will have changed to the new setting.

To change the place setting from 'New York' to 'Boston', follow the instructions above 'Changing the Place Name'.

---

### **TIP - Are You Still at the Right Time?**

**When you initially set the Clock your Notepad assumes that you are in London. If you are in a different Time Zone, New York for example, you should select that time zone area and then set the Clock to the correct time.**

---

### **Changing the Time difference**

All that remains now is how to change the time difference. You would only need to change this if you wanted to be able to select somewhere that wasn't already listed.

Let's change Tokyo at +09:00 difference to Afghanistan for example.

Getting out the telephone directory we discover that the time in Afghanistan is four and a half hours ahead of GMT.

Assuming that you are in a GMT area, check the Place name on the Information Bar, move the cursor over Tokyo.

Change the Place Name (as we did earlier) only this time since the time difference is



## Chapter 2 The Programs and Their Commands

displayed on the same line, pressing the ENTER key moves the cursor over to the Time Difference number. No grey areas are shown this time. You use the cursor keys to change the time difference in increments of half an hour.

In our example, this should be set to +04:30.

Well that is the Time Manager out of the way. To leave this (or any other program come to that), press the STOP key to Exit.

## The CALENDAR/DIARY (Function C)

The calendar and diary program run from 1st January 1920 to the 31st of December 2099. You can make diary entries within that period.

To use the CALENDAR/DIARY Menu from the Main Menu you press the Function and Cursor Down keys together which will select the Diary Menu and then the Cursor Right key.

Alternatively you can use the short cut key sequence of Function C from any other menu.

Time Zone : London	Mon Tue Wed Thu Fri Sat Sun	To make a diary entry select date using the ←↑↓→ keys and press ↵
Friday 20 Nov 1992	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1991 ↑ October ← + December ↓ 1993

The Calendar does not need much explaining because you will find all the help you need on the screen. Every day has a separate document where you can put in your diary entries.

Your Notepad will always start you in the Diary on today's date. To select a page in your diary, move the cursor over the date you want and press the ENTER key.

Making an entry in the diary does two things

- 1 It marks the calendar on that date with a '\*', to show you that you have something on.
- 2 When you have a diary entry for today, the Main Menu displays a message on the information bar and the Notepad makes a beeping noise. Pressing the ENTER key will show the correct diary page with your entry. You can choose whether to keep the entry or delete it when you leave the diary with the STOP key.

---

### **TIP - The Calendar Moves in Mysterious Ways**

When moving the cursor around the calendar, be careful when you move it towards the edge. Moving the cursor off the right of the list of dates changes the calendar to the next month. Moving the cursor off the left of the list of dates changes the calendar to the previous month. You will soon find yourself (like we did) on a day that you didn't want to be on. Take care, watch the cursor and double check the date before going into your diary.

---

### **TIP - You Can Miss Your Diary Dates**

Although your Notepad will remind you when you have a diary entry, this only works if you turn Notepad on during the same day AND you use the Main Menu. It will not remind you if you missed an appointment for yesterday. This can happen if you don't turn your Notepad on during the day.

If you want to make sure that you know about your diary entries, you need to do two things.

- 1 Set an Alarm call every day in the morning. That would make sure that the Notepad gets turned on.
  - 2 In the System Settings Menu check that you have the 'Preserve context during power off' set to 'No'. That will ensure that the Main Menu will be displayed when you first turn the Notepad on which should enable you to see your diary message.
- 

## **Deleting a Diary Date**

Normally, the Notepad deletes your diary dates after you have viewed them. This releases more memory for you to use elsewhere.

To delete other dates in your diary, move the cursor over the date on your calendar and press either of the Del keys. You then need to press 'Y' to confirm that you wish to remove that diary entry or STOP to exit.

## **What You Can't do in Diary**

There are three things that you can't do with your diary.

- 1 Print it.
- 2 Move appointments from one day to another.

## **Chapter 2 The Programs and Their Commands**

3 Make multiple entries, like birthdays, monthly subscriptions or even daily events, other than by retyping the entry into every day on which you want it.

One topic we will be covering later is making Macro's. No, this is not a strong Mexican lunch, but a way of storing text that can be recalled using two keys together. You can use Macro's to make multiple entries in your diary. Be patient; we will get there eventually.

After you found your way around the Word Processor, you can use some commands available there (displayed on the template) in the diary as well.

# Chapter 3

## Address Book (Function D)

You can use the address book to store your names, addresses, telephone numbers and even fax numbers.

There are several useful features. These include:

- Finding a name or place, when you know who you are looking for
- Browsing, when you've forgotten
- Changing the entry when they've moved or got married
- Transferring the address to the Word Processor

To get to the Address Book Menu from the Main Menu press the Function and Cursor Down keys together (to select the Diary Menu) followed by the Cursor Left key.

Alternatively you can use the short cut key sequence Function D from any other menu.



The Address Book is easy to work with since a menu of its commands is displayed with the addresses. There are two ways of selecting the commands. The first way is to move the cursor over the command you want and then to select it with the ENTER key. The other way is just to press the letter shown in brackets alongside the command.

Let's just go through all the commands

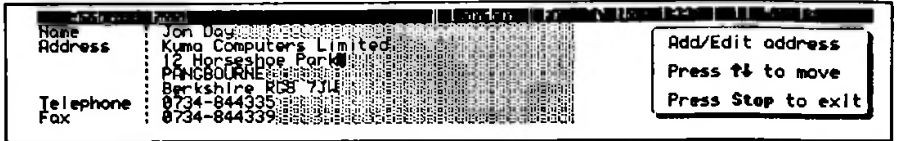
### Add New Address

Use this command to add new addresses to the list. Keep the address format constant. That is, don't allow the address to go into the Telephone or Fax sections. This allows you to use the contents of the address book to mailmerge later.

A good address to add in first is your own. You can use this later to make up your 'Headed Notepaper'.

### Edit Address

This command is just like the Add Address command except it uses the address you've already selected.



### Find Address

Find will find anything that you ask it for within the name, address and phone number. The best way to use Find is to ask for part of the word you are looking for. For example if you wanted to find 'Fred Smith' and you thought he was either at Southend or Southampton, rather than finding 'Smith' you would probably find him more quickly by searching for 'South'. That would find all the names and addresses with 'South' in them.

### Delete Address

Fallen out with anybody yet? This gets rid of them and releases memory.

### Browse

Use the left and right cursor keys to browse through all the addresses (in the order that you entered them).

### Transfer

There is an additional function called Transfer that we are just going to touch on here. It will only appear on the menu if you have a word processing document open simultaneously. We will cover this feature when we get to word processing.

# Chapter 4

## The Calculator

Your Notepad has a simple calculator that is immediately available from anywhere within the Notepad system. You can get to the calculator from the main menu by following the instructions on the screen, but you can go directly to it by pressing the Function and Tight Cursor keys together.

Whenever you are in the calculator there are few keys that will 'do' anything. The most important is the one labelled STOP that will always take you back to the main menu. The only other keys that are active are the ones with the green characters printed on them, the ENTER key and the <-DEL key. The ENTER does the same job as the key labelled '=' in green.

The keys behave slightly differently then you may be used to in the rest of the Notepad system. Everywhere else the keys 'auto-repeat'. This means that if you hold down a key for more than half a second the key will repeat for you automatically. However this is not so in the Calculator.

The calculator has a memory feature that is able to store one a number at a time. You can add or subtract from this number and leave the result in memory.

The calculator can also maintain a constant for use in a sequence of different calculations. Whenever there is a constant being maintained the letter 'K' appears on the far left of the screen. You can clear the constant by pressing the CE/C key twice.

### What the keys do

- 0 to 9 Enters the appropriate digit into the calculator
- . Enters a decimal point into the number being typed in
- = Causes the calculator to perform a calculation and put the result on the screen. The 'ENTER' key performs the same function.
- <-DEL Removes the last digit from the number currently displayed.
- CE/C When you press this key once it 'C'ancels the last 'E'ntry you typed in. If you are part way through a calculation pressing the key once only removes the last number entered. The previous parts of the calculation are left intact. When a constant is being maintained the display is cleared ready for the start of the next

calculation using the constant.

When you press this key twice it 'C'ancels the whole calculation and cancels a constant if one was being maintained.

- + x + -** The normal arithmetic operators divide, multiply, add and subtract.
- +/-** Changes the sign of the number currently displayed.
- %** Allow the calculation of a percentage.
- {** Calculates the square root of the number currently displayed. This operation can sometimes take a few seconds to complete.
- MRC** 'M'emory 'R'e'C'all displays whatever is currently in the calculator memory.
- M-** Takes the currently displayed number away from whatever is in the memory and stores the result back in the memory.
- M+** Adds the currently displayed number to whatever is in the memory and stores the result back in the memory.

The calculator is able to show up to twelve digits at a time. If the number is negative, a 'minus' sign appears at the far left of the screen. If you have tried to do a calculation that exceeds the maximum range of the calculator a flashing 'E' symbol will appear at the top left of the screen. The error symbol will only be removed when you press the CE/C key.

Before you use the memory you need to clear it. This can be done by either pressing the MRC key twice followed by the CE/C key or by pressing the MRC key followed by the M- key. Either route will result in the number '0' being stored in memory.

## Using the Calculator

The calculator works in a similar manner to most simple pocket calculators. In the following examples we have listed a variety of calculations using standard mathematical notation. Against each example we have shown the keys that you need to press and the sequence in which to press them to evaluate the calculation.

Example	Key sequence	Answer
27x4+15	27 x 4 + 15 =	123

$12+(-3)$

$12 + 3 +/- =$

$-4$

## Using a Constant Value

The calculator can save the value of one constant. This is a number that you wish to use in several different calculations. So for example if you wanted to multiply 154 by 73, 427 by 73 and 991 by 73 as three separate calculations you could save the number 73 as a constant and only have to enter it once. As you will see in the examples below the number to be used as a constant always has to be entered first, even when it logically comes second in the calculation.

Example	Key sequence	Answer
154x73	73 x x 154 = (pressing x twice saves 73 as a constant)	11242
427x73	CE/C 427 =	31171
991x73	CE/C 991 =	72343
12.36v1.8	1.8 v v 12.36 = (note that the constant was entered first)	6.866
8.94+1.8	CE/C 8.94 =	4.966
6.28+1.8	CE/C 6.28 =	3.488

## Using the Calculators Memory

The memory is a more powerful feature than the constant. The calculator allows you to add or subtract values to or from the number held in memory. The result of the calculation is always re-stored in memory as well as being displayed on the screen.

Example	Key sequence	Answer
1.8v12.36 (save 1.8 in memory to use in next sum)	MRC MRC 1.8 M+ v 12.36 =	0.146
1.8v8.94	MRC v 8.94 =	0.201
1.8v6.28	MRC v 6.28 =	0.287

## Percentages

23% of 523	523 x 23 %	120.29
23% increase on 523	523 + 23 %	643.29



## Chapter 4

23% decrease from 523	$523 - 23\%$	402.71
45 as a %age of 274	$45 \div 274 \%$	16.42

### Square roots

square root of 47.8  $47.8 \sqrt{\quad}$  6.91

square root of  
47.8-16.3  $47.8 - 16.3 = \sqrt{\quad}$  5.61

$(14 \times 8) - (72 \div 4.3)$  MRC MRC  $14 \times 8$  M+  
 $72 \div 4.3$  M- 95.256

The Notepad calculator does not include a facility for converting measurements between the Metric and Imperial systems. We have included the following table of constants to provide a quick way of doing the conversions if you need to.

In each case, if you want to convert from a metric measurement to an imperial one you multiply your metric measurement by the constant in the first column of numbers. Multiplying by the constant in the second column will convert an imperial measure to its metric equivalent.

Conversion req'd	Metric to Imp	Imp to Metric
kilometres, miles	0.6214	1.609
metres, yards	1.0936	0.9144
centimetres, inches	0.3937	2.54
hectares, acres	2.471	0.4047
kilograms, pounds	2.2046	0.4536
grams, ounces	0.03215	28.35
litres, gallons	0.22	4.545

# Chapter 5

## Word Processing

### Introduction

#### Why Use a Word Processor?

Unlike when you use a typewriter or a pen, what you write on a word processor need not be the final version on paper. This has several advantages because it means that you can alter anything, from one word to the order of the paragraphs, without having to write everything again.

For example, you could use it to list the things you wanted to cover in a letter. If you then changed your mind, you could move things around in the list to wherever you wanted them to be. This would save writing all over the list, drawing arrows to show where you wanted everything moved to and then rewriting the whole thing again.

If you wanted to add more things into the list or expand on things that were already there, that would be easy too. You could just type in the additional things creating extra lines in the list as necessary. You could even type your letter around the list that you have made.

As your document gets larger, you can leave markers in it. These allow you to return to a particular point in your document whenever you want to. Although, if you use all ten of the markers available you may have a problem remembering what you used them all for.

It is easy to make changes to your document, since nothing is fixed until you print it. Even after you have printed it, you can still change it in your Notepad and then print it again, always providing that you have not deleted the document from your Notepad in the meantime.

When you have finished writing, leaving the word processor automatically saves what you have typed into a document. You can then print it out or return to it later and make some more changes. Word Processors are particularly good at working with documents that have similarities in common. Letters, for example, have lots of features in common. Any letter you send out is likely to have your address, 'Yours sincerely' and your name in a format that remains the same every time. These similarities can be stored in a 'master' document. Whenever you want to start a new letter you can copy your 'master' into your new document. You don't have to retype them every time. You can take it further than that if you send out the same type of letter to different people. You can store the whole letter in a document and just add the appropriate name and address as you need them.

There is even a way of asking your Notepad to add the names and addresses automatically as you will see later.

The icing on the cake for word processor users is a Spell Checker. No, this is not something that will check what you have put in your magic potions. A spell checker will check the spelling of all the words in your document against those stored in its dictionary. It won't find all your mistakes because if it finds the word in the dictionary it will pass it whether it is the right word or not. If you meant to say 'it' but you had actually typed in 'to', the spell checker would not consider that a mistake.

If you have not used a word processor before, then the best way to get to know how to use it, is to use it. Just type anything in you like and try out the commands that we will be describing. After all, you don't even have to try to find a piece of paper to write on do you? How about the weekly shopping list or the note to the newsagent or milkman? You will soon discover how you can save time by re-using documents that you only need to change slightly.

To use your shopping list as an example, you probably need to buy some items every week. You might find that every week you need at least two bags of sugar. Great, you won't need to write that out every week. You could list the regular items and just change the quantities as appropriate.

You may be one of those clever people who like to organise their list so that everything appears in the order that you pick them off the shelves. If so, do supermarket managers seem to like to make your life difficult by changing everything around? They won't be able to sustain the confusion for anywhere near so long with you from now on. You can change your list to the new layout very easily. You could even take your Notepad shopping and change the list as you go round the shop. That way you won't even need to print out your shopping list.

## **Much More Than a Typewriter**

In most respects a word processor behaves in the same way as a typewriter. When you press a key on the keyboard the character for that key is printed on the screen. The character is always printed where the cursor is. The cursor is the black flashing rectangle that starts at the top left of the screen when you first enter a new document.

At the simplest level the only difference between a word processor and a typewriter is that a word processor allows you to change what you have typed as often as you wish, quickly and easily, without the need to resort to the Tippex.

In practice there are many more differences, mainly to do with the extra things that a

word processor can do for you to make entering text quicker and easier.

Here are a few of the differences between a typewriter and the Notepad's word processor.

## **The ENTER Key**

When typing text you only use the Enter key at the end of paragraphs, in which case you have to press it twice to leave a blank line. While you are working on a paragraph the word processor will sort out when to move the cursor down to the next line for you.

---

### **Definition - Word Wrap**

Word processors can decide when to start a new line in a paragraph. This function is called 'Word Wrap'. This means that you don't need to worry about the length of the lines at all. Just type a paragraph as if it is one huge long line, leaving two spaces between sentences. The word processor will sort it all out for you.

Unless you are typing something that you want to control where a new line starts, leave word wrap turned on.

---

## **Use Tabs to Line Up Your Text**

If you want to include columns of information or to indent a section of text within a document, you could line up the words by pressing the space bar until the cursor was in the correct position. This scheme would work fine while the document is on the screen but may not look so good once it was printed. However, by setting a tab stop to where you wanted the text to start and then pressing the Tab key to move the cursor there, you could guarantee that the printed copy will be aligned correctly.

---

### **Definition - Insert or Overtyp**

There are two modes in which you can use the word processor. Normally it starts in Insert mode. Insert mode means that when you add more text into the middle of a line or paragraph the text in front of the cursor moves along letting the additional text that you are typing into the gap that has opened up.

For example, if you type 'A black cat' but then you decide it should be 'A fat lazy black cat', all you need to do is to move the cursor to the space between 'A' and 'black' and type 'fat lazy' and a space. You will then have the desired effect.

In Overtyp mode, the character you type replaces the one where the cursor is. So in Overtyp mode if you started with 'A black cat' and tried to change it to 'A fat

black cat', you would end up with 'A fatck cat'.

---

### Tab and ENTER keys

You will find that you frequently use these two keys. There are two ways that these can work.

If you are in insert mode pressing the tab key will insert a tab character and move any text on the right of the cursor to the next tab stop. The ENTER key will insert a carriage return and move any text on the right of the cursor onto the next line.

When you are in overtype mode pressing the tab key will move the cursor to the next tab stop but leave your text untouched. The ENTER key will move the cursor to the start of the next line, again leaving your text untouched.

### Shift Enter and Shift Tab

Pressing the Shift key with either the Tab or ENTER keys reverses the way they work.

If you are in insert mode, both behave as though you were in overtype mode.

If you are in overtype mode, both behave as though you were in insert mode.

### Your First Letter

Turn your Notepad on and press the Function and N keys together to start a new document. Type in a name for your first letter - 'thankyou' - and press the ENTER key. Now you are in the word processor ready to start typing into your new document.

Dear Aunty Tawny

Thank you for my Christmas present. I really needed the bat's wing potion. I hope you are right because I really do want a hairy chest. I'll rub it in every night but I do wish you could do something about the smell.

Hope to see you before Easter.

Love

Vic

Now press the STOP key. This will leave the word processor and save your document for you. You can go back to it at any time by pressing Function L, choosing the document from the list displayed and then pressing the ENTER key.

### **Printing Your First Letter**

Now that you have typed a document into your Notepad, you will probably want to print it out. This section is designed to get you going. If you run into trouble you will need to read The Printing section for more details on how to get your printer working.

- 1 Connect a cable between your Notepad and printer. You should have either a serial or parallel printer cable, depending on which your printer needs.
- 2 Select the Print document Menu with Function and P.
- 3 Select the Printer menu with the Menu key.
- 4 Press the Up Cursor key to select the Printer Configuration Menu.
- 5 Change the Printer setting to the type of printer you have, if you are not sure, leave it set to 'simple' for now.
- 6 Change the Printer Port to Serial if the connection to your printer is with a serial cable.
- 7 Leave the Menu with the STOP key.
- 8 Switch on the printer, make sure that the paper is ready and the on-line light is ON.
- 9 Move the cursor over 'Thank you' and press the ENTER key.

Your printer should now be printing out the letter.

If you get the message -'Plug in or switch on the printer and press ENTER or press STOP to cancel...' the printer is not working. Do as it says and if nothing appears to be wrong, go to the Printing section later.

### **Your First Printout**

Once the document has been printed, providing you have got your printer to work, you will see how it has been laid out by the word processor. The word processor has a set of

default settings for controlling the layout of documents. You do not have to change these settings. However, as you get to know the word processor in more depth, you will find that you may want to alter some settings and 'customise' what your documents look like.

# Telling the Word Processor What to do

The name of the word processor in the Notepad is Protex. You need to tell Protex what to do by entering commands. You can enter commands in several different ways.

The menu screen lists some commands; the template lists some more. There are a range of commands not listed anywhere and there are some commands that can be included within your text. In fact some commands can be used in two or three different ways.

Commands do a multitude of different things. Some commands move the cursor around the document much quicker than the Cursor keys. These allow you to type text into different places. There are commands to copy or move sections of text from one place to another and there are commands to control how your document will look when it is printed.

We will be covering the commands in groups that do similar jobs. So for example we will include all the commands concerned with controlling the look of your printed document in one section.

This approach has one main advantage and one possible disadvantage. The advantage is that it will make finding a command to do a specific job straightforward. You will always know where to look when you need help. The possible disadvantage is that of necessity we will be describing some complicated commands early on. However, if you remember that you don't have to learn everything immediately, we are sure that this approach will help you in the long term.

First we need to cover how to get to the different commands.

## Using Menus

A menu is just a list of possible choices. The choice may offer a direct action, for example 'turn word wrap on/off', or it may lead to another sub-menu with a further set of possible choices.

A menu choice can frequently be made directly by pressing a special key combination. So instead of entering a menu, and then possibly a sub-menu, and choosing what to do, you can press two or three keys together to make your choice from within your document. If you wanted to know how many words there were in your document, you would press

Function and the '=' key both together.

If you wanted to clear any markers you have placed in your text you have a choice how you do it. You can either:

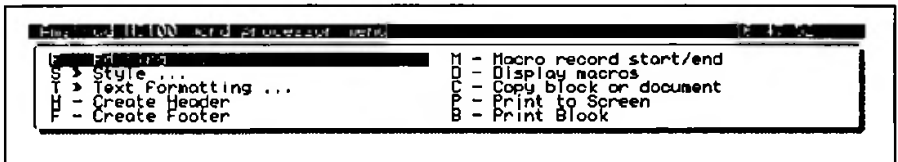
go into the word processor menu, select 'editing commands' and then select 'clear markers' or press Control and K.

If you find yourself in a menu by mistake, pressing the STOP key will return you to wherever you were in your document.

There are three menus that you will use frequently. We will quickly cover these and explain the types of choices that you can make using them. You will find the detailed description in the relevant sections later in this book. We will explain any unfamiliar terms as we come across them.

## The Word Processor Menu

To get to the word processor menu you press the Menu key.



The editing commands cover selecting a character, moving around your document, some block operations, the user dictionary, inserting the date and inserting the time.

The style commands allow you to change the way the characters in your document will be printed.

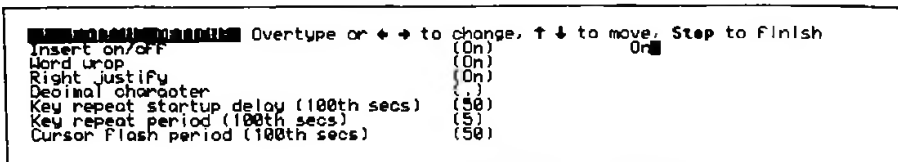
The text formatting commands insert ruler lines into your text, insert special characters and markers and format paragraphs.

The Word Processor menu also allows you to create headers and footers, handle Macros, copy or print a block of text and print your document to the screen.

## The Configure Menu

To get to the Configure menu press Function and 3. This is shown on the template underneath the screen.



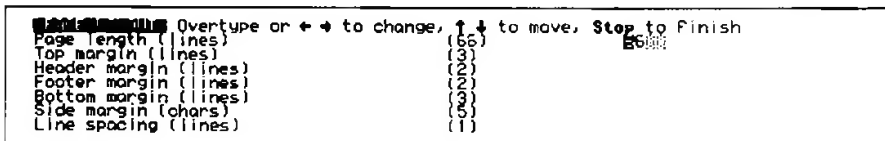


The Configure menu tells Protex how you want it to behave. You can change things like insert or overtype mode, how fast the keys should repeat if you hold them down and whether you want special characters displayed on the screen. There are more choices within the menu than will fit on the screen in one go. As you move the cursor down from the last choice on page one, page two will be displayed.

These settings have default values that Protex will use if you don't tell it otherwise. You don't have to change any of them. These values are used for all the documents that you work on. So if you change a value in this screen when typing in one document, all documents will have the new settings.

### The Layout Menu

To get to the Layout menu you press Function and 7. This is shown on the template underneath the screen.



This menu allows you to tell Protex how you want your document to be printed. It controls things like the number of lines on a page or how many blank lines to leave at the top of a new page.

As with the configure menu, these settings have default values that Protex will use unless you change them. However, the settings in the layout menu are saved with your document. This means that changing these settings will affect your current document and any new documents that you start, but will not affect any existing documents already on your Notepad.

### The Nowhere Commands

There are several commands that don't appear on the Template or any of the menus. Don't panic. Just keep this book by your side. You will quickly pick up the commands

that you use frequently. The ones you only use occasionally you will probably have to look up when you need them.

## **Command Lines**

All commands to alter the settings of the word processor have '>' as the first character on the line. This tells Protex that it should not print the line but action the command instead.

The characters in the command can usually be in either UPPER or lower case. It is generally a good idea to stay in UPPER case since this makes the commands obvious when you are reading your document on the screen.

You will not always see the effect of the commands immediately. Some commands are only actioned on the next new page and some only come to life during printing. As we describe each command we will tell you when it comes into effect.

---

### **TIP - Lines Missing in your Printed Document?**

**If you have lines of text missing from your printout, check on your Notepad to see if you have any '>' characters at the beginning of a line. If so delete them, the line should then be printed.**

---

### **TIP - Don't Use '>' Characters in Word Processing Documents**

**The beauty of word processing is that you can make as many changes as you like to a document. To accommodate these changes Protex is reformatting your document (repositioning all the words) as you go. This means that you cannot guarantee where any character will be in the final version of the document. If you use the character '>' you could find it working its way to the beginning of the line. There would then be a line missing in the printed copy of your document. So it is probably better not to use the '>' character.**

---

Now that we have told you how to make a command work, let's start working through the groups of commands and describing them in detail.

There is usually more than one way to access each command. When this is the case we will give you the quickest route. Generally one route will be via a menu command and thus you won't have to remember how to get to it. The second route will be faster, a special key combination that you will need to learn.

## What is a Ruler Line?

A Ruler line allows you to change the settings for your margins and tab stops at any point in your document.

### Turning the Ruler line ON (Control V R)

Normally, the current ruler line is hidden from view to prevent using one of those seven valuable lines on the screen. If you are changing rulers often, it can be helpful to display the current ruler line to remind yourself which one you are using.

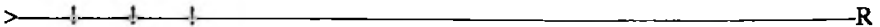
There is a short cut key sequence of Control V and R, to 'Show ruler'. This is a toggle function. If the ruler line is hidden, Control V R will show it. If it is already on view, Control V R will hide it.

### The Default Ruler Line (Control D)

This command will copy the default ruler line into your text. You can then change the ruler line to suit the layout you want for this document.

When you enter the Word Processor to start a new document, Protex uses the default ruler line. It determines where the Left Margin and Right Margins are and includes three preset tab stops.

The default ruler line is:



If the first line in your document is your own ruler line, Protex will use this as the default ruler line.

### Creating a Ruler Line

The Ruler line is formed from a set of different characters. The position and number of these characters can be changed to give a different layout.

These characters are;

'>' which tells Protex that this line is a command line; it is not part of the text.

'-' which fills all the spaces between any other characters in the line.

- '!' which shows where a Tab stop will be
- 'L' which shows the left margin mark. It is only used when your text does not start from the first column on the left-hand side.
- 'R' which is a right align mark for the line.
- '.' which aligns the decimal points in a column of numbers.

You can create a ruler line by typing in the different characters, or you can modify a ruler line that you have already.

Suppose we wanted to enter a document that was only forty characters wide instead of the normal seventy characters.

By pressing Control D the default ruler line would be copied into our document.

> !-----!-----!-----R

Then we just have to modify it by moving the cursor onto the line and deleting characters until it looks like this:

> !-----!-----!-----R

Any text typed in after our new ruler line would be limited to forty characters on each line.

---

### TIP On Ruler Lines

**Although it does not matter whether you use UPPER or lower case, don't use the lower case 'l'. It looks very similar to the '!' and you will get confused.**

---

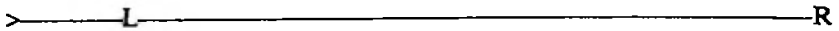
The '!' shows where a tab stop is placed. Any text written below the ruler line will use these tab stops. Each time you press the Tab key Protext will insert a tab character into your text and move the cursor to the next tab stop. Tab characters are not normally visible but you can see their effect and you can delete them just as you would a visible character.

## Indenting Text With the Ruler line

Suppose you wanted to print a set of terms and conditions using your Notepad. The general layout would need to look something like this:

- 1.1 You will pay the bearer when he asks for payment, blah blah blah and even more text in the hope that the reader doesn't bother to look at it any more.
- 1.2 And notwithstanding even more long words, hereby, and we wouldn't mind going to Scotland one day.

Paragraph numbers start at the far left of the page but all the text is aligned to seven characters in from the left edge. The following ruler line would indent the text for you.



After you have entered the new ruler line, the cursor will be positioned at the left of the screen. This is where you would type the paragraph numbers in our example. When you press the Tab key the cursor will jump to line up with the 'L' in the ruler line. This is where you type in the paragraph text. As you type, Protex will automatically line up the left edge of the paragraph for you underneath the 'L'.

When you have finished one paragraph and you want to enter the next paragraph number, use the cursor keys to move the cursor back over to the left of the screen and the process starts again. A quick way of doing this is with the Control and Left Cursor keys together.

You can change the ruler line at any point in your document simply by creating a new one.

Only text below a ruler line is affected by it. Any text above a ruler line will be affected by the previous one.

## Copying the Previous Ruler (Control R)

We don't like typing, so any quick way of getting something to save time is going to get our wholehearted approval. Rather than typing in a ruler line we'll copy the previous ruler line into our document and then modify it. As we have seen above, we can copy the default ruler line into our text by pressing Control D. The key combination Control R will copy the previous ruler line into our text. The first time it is used this will be the default ruler line, since we haven't yet changed it. But, once you start adding your own ruler lines in, Control R will give you your previous line.

## The Status Bar (Function 8 or Control V I)

The Status Bar in Protex normally shows you the document name followed by the page that you are on. After that there is a long message 'Press Stop to finish or print' and the time. By now you would probably know that the STOP key is the way out. We could replace that message with more useful information to help us with the page layout.

This command will add the following information to the status bar -

- The cursor position: page number, line and column numbers
- Whether insert is turned on or off
- Whether text is right justified or not (RJ shown if on)
- Whether word wrap is turned on or not (WW shown if on)
- The time
- Whether the caps lock is on or not (C shown if caps lock on)

The command operates as a toggle. If the status bar is off and the 'Press Stop...' message is shown, Function 8 will turn the status bar on. Pressing Function 8 again will turn it off.

## Where You Are in The Document

At any one time you can only see seven lines of your document. This is a little inconvenient but you can keep track of where you are by reference to the page, line and column numbers shown on the Status bar.

By default a standard page in Protex has 66 lines that are 70 characters wide. Each character on a line occupies one column. If the Status bar tells you that the cursor is on page 1, line 17 and column 34, the next character that you type will appear as the thirty-fourth character on the seventeenth line of the first page. All pretty sensible really.

Generally it is of little interest where you are on any particular page but it can become important in some circumstances.

Suppose you wanted to include a table of values in your document. You may not want the table split between two pages. To check if you have enough space to include the table on the current page you would need to know the current line number.

Having typed in your table and several pages of text after it, you may want to go back to the table. You could just keep moving the cursor up your document until you got there, but if you knew what page the table was on, you could tell Protex to go straight there.

## Insert on/off (Control 4 or Control Tab)

The Status Bar will tell you whether insert is on or off. You can change the setting of insert by pressing Control 4. It acts as a toggle switch - press once for on, press again for off.

Whether Insert is on or not determines whether the next character you type is inserted into the current line or whether it just overtypes what was there before.

## Right Justify (Control J)

If Right Justify is turned on the Status Bar will include the letters 'RJ' about three-quarters of the way along. Right Justify is another toggle switch available from the configure menu.

---

### Definition - Right Justify and soft spaces

Right Justify means that all the lines on the page are made the same length. Normally, when you type a page with a typewriter, the lines will be different lengths and the right-hand edge will be uneven.

Protext adds 'soft spaces' between words in the line so that the right-hand edge of the text is straightened up, like the left-hand edge normally is.

Soft spaces are temporary spaces that Protext uses to pad out the line. Because they are temporary Protext can add them in or take them out as you edit the line.

---

## Word Wrap (Control W)

If Word Wrap is turned on 'WW' will appear on the Status Bar three-quarters of the way along.

Word Wrap is also a toggle switch.

---

### TIP Typing a Long Line Unintentionally

Turning Word Wrap off will allow you to type your document in one very long line. If you find you are doing this unintentionally, turn Word Wrap on.

---

## **The Time**

This should be showing the right time. If it is wrong, you'd better set the clock again.

## **Cap Lock On**

A capital C will be shown at the end of the Status bar if the Caps Lock is on.



# Chapter 6

## Your Second Letter

Let's write another letter and get to know some more of the commands on the way. The first thing that would be useful would be a letterhead with your name and address on it. Once we have done that you can use it on future letters as well.

### Opening a New Document (Function N)

To select a new document from the Main Menu you first press the Function and Left Cursor keys to select the word processor menu and then the Left Cursor key again.

Alternatively you can use the short cut key sequence of Function N from anywhere in the Notepad.

### Choosing a Name

What's in a name? You must give all your documents a name so that you can find them again later. It needs to be a name that will mean something to you not only now but when you want to return to the document in six months time. You can have a name of up to twelve letters, including spaces, on the Notepad.

### PC Compatible Document Names

Other computers have taken the PC standard for document names. They, and the equipment that works with them, can only cope with names that are eight characters long. No spaces can be included but you can separate words with the underline character. PC compatible names can have an 'extension' on the end. This is an additional three characters separated from the name by a full stop. 'MANE.123' would be a perfectly acceptable filename.

A name that meets the standard requirements would in fact be twelve characters long (eight for the name, one for the full stop and three for the extension) which fits in nicely with your Notepad. If you are likely to ever want to transfer your documents to another computer, or to a disk drive, it may be a good idea to restrict yourself to document names that are eight characters long. You can always use an extension on the end to tell you what sort of document it is. You could use '.ltr' for letters or '.rpt' for reports for example.

If you don't worry about the PC standard now, you may well live to regret it. If in the future you do get a desktop computer, the only way you could put your Notepad

documents onto it would be to labouriously change all the names by hand.

To give you some examples of Notepad and PC compatible names; a document called 'Labyrinth' would be valid on the Notepad but not on another computer, while a document called 'Maze' would be valid on any PC compatible computer as well as the Notepad.

## Other Things You Can Do to a Document Name

You can use numbers in document names on PC's and your Notepad. You could add numbers to the document name to show different versions of your text.

If you want to use a name that is too long, removing the vowels (aeiou) can help you shorten it. The name will still be understandable even like that. Always remember, however, that the important thing is that you understand the name.

## Your Letterhead

Since we are going to create a letterhead that we can use for all our future letters, we're going to call this document 'letterhd.doc.' Type that in after the 'Please type a name for the new document' and press the ENTER key.

The first thing we need in a letterhead is our address. We typed that information into the Address Book so we don't need to enter it again - the Notepad can transfer it into our document for us.

---

### TIP Put Something in a New Document, or it Will Disappear

When you first create a new document it is empty. If you leave it and go somewhere else in the Notepad there is nothing for Protext to save for you. Although you have given it a name, your new document doesn't exist. If you want to be able to come back to the document again and don't just want to abandon it, you must put something into it.

Pressing the Space bar once is enough to tell Protext that we really do want this document.

---

## Transferring an Address into a Document

From your new document you can select the Address Book by pressing the Function and D keys. Select the address that you want using the Browse or Find commands. If you



You should repeat this key sequence until the whole address is on the right-hand edge of the page. If you want it further over to the right or left, change the Ruler Line above the address and then reformat it again.

At the bottom of the address, you ought to put back a default ruler line, so that anything you type below the address will start back in column 1. You can get this by putting the cursor at the bottom and using the Control D command. Hey presto, there is a new ruler line.

### **Making Your Master Copy**

You don't want to do this all again when you type another letter do you? Let's save this document, which is done automatically when you leave Protex.

### **Using Your Master**

Now let's load our letterhead into another document. Start another new document (Function N) and call it 'test1.ltr'. We can now insert the other document into this one.

### **Insert Document (Function 2, select document, I)**

This command allows you to insert one document into another. We are going to use it to insert our letterhead into the new document 'test1.ltr'.

Press the Function and 2 keys together and you will be presented with a list of the names of other documents stored in your Notepad. Move the cursor to the name of the document with your letterhead in, 'letterhd.doc', and press the Menu key.

You will be presented with a menu of commands titled 'Document operations' with the document name you have selected. We will cover all the individual commands in detail later, but for the moment just press the 'I' key for Insert.

You will be returned to your 'test1.ltr' document with the 'letterhd.doc' document inserted where the cursor was.

It is now easy for you to insert 'letterhd.doc' into every letter you write on your Notepad from now on.

## Continuing the Letter

How about putting in today's date? We need a couple of blank lines and then the date over on the far right of the screen.

### Insert Line (Control 2 or Control I)

There are two ways of placing blank lines in your document. One way is to use the **Insert** line command (Control 2) shown on the template. If, however, you have 'Insert On' then you just press the ENTER key.

### Right Align

If you Tab across to the right-hand side of your ruler line (where the 'R' is) anything that is typed there will be right aligned. Alternatively, if you have already typed something and you want to move it over to the right, place the cursor at the beginning of the line and Tab across. If you are moving text that you have already typed, don't go right to the end because you need to leave enough space to fit it on the line.

### The 'R' Stop on the Ruler Line

Although we are using the 'R' as the end of the line, there is nothing to stop you having more than one 'R' stop in your ruler line. Text typed at any of those stops will be right aligned.

### Adding Today's Date (Symbol D)

There is a quick way of inserting today's date. Here is what you do. Move the cursor just below your address. Insert two blank lines. Move the cursor over to the right by pressing the Tab key several times until it is just past the 'R' in the ruler line. Then press Symbol and 'D'. Easy isn't it?

---

#### TIP - Adding The Time (Symbol T)

There is also a quick way of adding the time into your text as well. You could use this for sending a fax, or even for a list of what you've done in a day.

---

## The Rest of the Letter

Now you can write the rest of the letter. Make sure that you are below the 'Default Ruler line' you have in your letter before typing the rest of the letter otherwise you will have a very narrow strip of writing.

Dear Somebody I know,

Well, I went out and got myself an Amstrad Notepad computer and quite an interesting book on it. It is taking a long time to get through because the authors keep introducing new commands.

Oh dear, here is the start of another paragraph, where I use what he calls the ENTER key a couple of times. It is surprising that when the text reaches the end of the line, it jumps down to the next line taking any whole words with it.

Well, I had better go, here comes another one of those commands 'Centring' text. How am I going to remember all this lot? I must be lucky, this time, there is a 'Centre' command on the template, so I'll type the text first, then use the command Control =.

Kind regards,

My name

## Mopping Up the Format Commands

### Word Count (Function =)

This command simply counts all the words there are in your document. See if the number of words here agrees with the actual number of words in your letter.

### Format text (Control F)

While typing in text, Protext is constantly formatting the text for you. But if you go back later and delete characters or words you may find that the text needs to be reformatted. If

you move the cursor to the start of a paragraph pressing Control and F will reformat the whole paragraph for you.

If you have added or changed a ruler line above a piece of text you will need to implement the changes by reformatting the text. This is how we moved the address across to the right when creating our letterhead.

### **Format Complete Document (Word processor menu, Text formatting, F)**

Unlike Control F, which only formats a line or a paragraph at a time, this command formats the whole document in one go. Take care when using it because you have no control over what it does to your document while reformatting.

### **Centre Text (Control = or Control C)**

This centres the text between the left and right margin settings. Make sure that the cursor is on the line you wish to centre before issuing the command.

The command does not take into account what sort of printing you may be using. If you are using double width printing for example, then this command will not centre the text correctly. It may need human intervention, putting in extra spaces or changing the margin settings, to get the effect you require.

## **Macros**

Wouldn't it be good if there was a way of pressing a couple of keys and the Notepad would type out some well used phrases? For example:

Kind Regards,  
(a couple of carriage returns),  
Yours sincerely, (Centre the text),  
(a few more carriage returns),  
followed by your name, underlined and centred again.

Well there is a way and it's called a Macro.

Think of a Macro as a tape/cassette recorder that can record your keystrokes and play them back when you want to. You are not limited to one Macro either. You can store two different Macros on most of the keys on the keyboard. One is saved against each key on its own and another is saved against the same key together with the Shift key.

## Displaying the existing Macros (Word processor menu, D)

This command will show you all your stored Macros, a screenfull at a time. There are some pre-recorded Macros already in the Notepad. These cover the date and time and many standard foreign characters. If you do not need any of these pre-recorded macros you can store your own instead.

Here is a list of the preprogrammed macros -

Sym-a		ä
Sym-c		ç
Sym-d	^788^	the date
Sym-e		æ
Sym-h		½
Sym-m		u
Sym-n		ñ
Sym-o		ö
Sym-p		(decimal 20)
Sym-q		¼
Sym-s		ß
Sym-t	^789^	the time
Sym-u		ü
Sym-A		Ä
Sym-C		Ç
Sym-E		Æ
Sym-N		Ñ
Sym-O		Ö
Sym-S		ß
Sym-U		Ü
Sym-<		«
Sym->		»
Sym-!		¡
Sym-?		¿

## Recording Your Own Macros (Shift Control M)

You can only record a Macro in the Word Processor. You will be presented with the message 'Press Symbol and letter, then key sequence:'. Now is the time to show the Notepad which keystroke you wish to use to call your new macro. Perhaps you could use



Symbol Y for 'Yours sincerely' for example.

Once you press the Symbol and another key. Protext will start recording your macro for you. You will see an 'R' on the right-hand end of the information bar. This is to remind you that you are in Record mode. Now whatever you type, as well as being typed into your document, will also be recorded into the Macro.

Try typing the following -

Yours sincerely  
(Enter key)  
(Enter key)

When you have finished, press the keys that you used to start the Macro again to get out of record mode.

## Using Your New Macro

After recording a macro, you can play it back simply by pressing the symbol key followed by the other key you used to record the Macro. For the example above, Symbol Y would playback 'Yours sincerely' followed by two carriage returns.

You should place the cursor where you wish the Macro to run before you run it. You can play back your macro in a document, and in any other Notepad program like the Address Book, Alarm, the List, BBC BASIC or the Diary.

If you made a mistake when recording your macro, don't worry, simply overwrite it with the correct version.

## Macros and Reset

Unfortunately, if you have to reset your Notepad you will lose all the Macros that you have created. There is no facility to save your macros to a document so make a note of your macros in a word processor document. Then if the worst happened, you could create them all again.

# Special Characters and Accents

Many additional characters including those with European accents are available. You can select these in different ways.

Some of the more common characters are available as macros using the Symbol key followed by one other key. We have shown these in the list of existing macros above.

If the accented character is not in the list above but you know exactly what you want, you can press a key sequence below to get directly to it.

If you don't know which character you want you can see the whole available list by pressing the Symbol and Menu keys together.

To access a special character directly you have to press the Symbol key and the minus key together. Then, still holding the Symbol key, press another key followed by the character you want the accent with. Got that?

Suppose you want an 'a umlaut' -

press the Symbol key and '-'  
keep pressing the Symbol key but now press '2'  
let go of both keys and press 'a'

and your special character turns up on the screen. It isn't quite as difficult to do as it sounds and if you need the characters regularly you soon remember which keys to press.

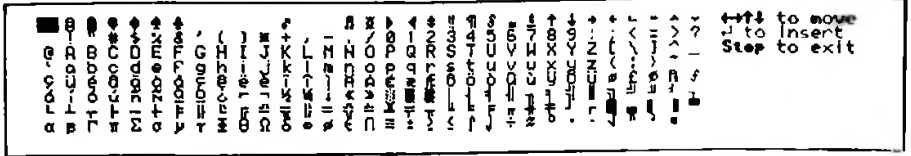
The possible accents and their valid characters are as follows -

Accent	Keys	Valid letters
Umlaut	Sym '-' then Sym '2'	aeiouAOU
Ring	Sym '-' then Sym '5'	aA
Circumflex	Sym '-' then Sym '6'	aeiou
Acute	Sym '-' then Sym '>'	aeiouE
Grave	Sym '-' then Sym '\'	aeiou

Select which accent you want first, then select the letter you want to use it with.

## Picking Your Special Character (Symbol Menu)

You can select these and other characters in the Symbol Menu. This menu will show you all the different characters including all those covered above that can be displayed on the screen. To select any of them, move the cursor over the desired one and press the ENTER key.



If, when you print these characters they don't look the same, see the Printer section.

## Drawing Boxes with Lines and Characters

There are several characters displayed on the Symbol Menu that allow you to draw either single or double horizontal or vertical lines on the screen. These, with corners and other joining line characters, can even be used to draw boxes.

Using the Symbol Menu to select them individually would be both labourious and impractical, unless you were desperate. So there is an easier way to access the box drawing characters using the cursor keys.

### Box Drawing Modes

You can either draw boxes with lines or characters. If you select Line mode you can choose to have your boxes drawn with single or double lines. If you select Character Mode, you can choose characters or Arrows to make up your boxes. Character mode allows you to select any character on the keyboard, while Arrows just draws the box with a series of small arrows.

If you are in line mode, Protex will correctly insert corners and 'T' shapes where appropriate. In Character mode, however, no special action is taken when lines cross.

After selecting the mode you want, you can draw the box by moving the cursor with the Cursor keys while holding down the Symbol key. Wherever the cursor moves to it will leave a trail of the chosen character behind it.

### Box Drawing with Single lines

Let's do the easy one first. Move the cursor to where you wish to start drawing, hold down the Symbol key and draw with the cursor keys. You will leave a trail of line characters behind you. The correct characters will be inserted automatically as you change the direction of the cursor and as you cross previous lines.

## **Box Drawing with Double Lines (Shift Control D)**

By pressing the Shift, Control and 'D' keys together you can switch to the double line mode. This behaves in the same way as the single line mode apart from the characters used to form the lines.

The command is a toggle and so using it again returns you to single line mode.

## **Box Drawing with Characters (Shift Control C)**

If you press this command once you will switch to displaying arrow symbols when the Symbol and cursor keys are pressed. This command has to be used twice to make use of characters for drawing lines.

After pressing the Shift, Control and 'C' keys twice you will be asked to press the character you want to use to draw with (the question is in the information bar at the top of the screen). Press any character you wish to use to draw with and then start drawing in the same way as before.

You cannot select a character from the symbol menu but you can select an accent followed by a letter.

## **To Return to Line Drawing Mode (Shift Control L)**

Use this command twice to get back to drawing lines with lines rather than characters.

## **Lines or Boxes Not Printing?**

If, when you print these boxes they don't look the same, see the Printer section.

# Chapter 7

## Page Formatting Commands

Most of the page formatting commands are listed as command lines. You should place them at the beginning of the document if you want them to affect the whole document. If you only want them to affect part of your document then you put the command line at the beginning of the relevant section and another command to reverse the action at the end of the section.

Some page formatting commands are in the Print Options menu. Any formatting commands in the document will override the setting in the print options menu.

First some general points regarding command lines.

Some command lines can switch things ON or OFF and we will show these followed by 'ON/OFF'. You need to include the command followed by either 'ON' or 'OFF' as appropriate. Other commands require a number to be included with them. We will show these followed by 'n'. You need to include the command followed by an actual number, for example 50.

The last type of command operates directly on the text that follows it. We will show these followed by 'text'. You would include the actual text that you wanted the command to operate on. For example:

>CE	would centre the line
>PL 50	would set the page length to 50 lines.
>FP ON	would turn on the 'Format While Printing' command.

Some command lines appear to do the same thing as commands we have already described above. A good example is centring text. However there is a subtle difference between them even when they look the same at first glance.

Commands used directly on the text that you type in are implemented immediately and then forgotten. If you have centred a line of text with Control '=' and then you go back and change the text, it will no longer be centred. However, if you add the command line >CE'text' to your document, the text will always be centred no matter how often you alter it. However the centring only takes place when you print it out.

## Headers and Footers

Headers and Footers are extra lines of text that, if defined, can be added at the top and

bottom of every page automatically. For example if you were writing a report you may want its title in the header line and the page number in the footer.

### **Define Header Text and Turn Headers On (>HE'text')** **Define Footer Text and Turn Footers On (>FO'text')**

You can define one header line and/or one footer line to be active at any one time. The default is for no headers or footers.

### **Turn Headers On/Off (>HE ON/OFF)** **Turn Footers On/Off (>FO ON/OFF)**

You can turn Headers and Footers off and then back on anywhere within your text without altering their contents. This command is only actioned on the next page which means that you must issue the command one page before you require it.

## **Page Numbers in Headers and Footers**

You can print the current page number in the Header and/or Footer by positioning the character % in either/both of those lines. If you need an actual % character in the line you need to put in two, one will then be printed.

If you wanted to print the page number at the bottom of every page in the document, you could include the following command line -

```
>FO                               Page number %
```

## **Odd and Even Pages**

Protect has an additional feature that allows you to print different headers and footers on odd and even pages. This enables you to lay out a small booklet. You could print the page number on the outside edge of both left-hand and right-hand pages.

### **Define Odd Page Header Text and Turn Headers On (>OH'text')** **Define Odd page Footer Text and Turn Footers On (>OF'text')**

Page one is normally a right-hand page, so if we wanted to use our footer from above we could modify it such that the page number was printed on the right thus -

>OF

Page number %

**Define Even Page Header Text and Turn Headers On (>EH'text')**

**Define Even Page Footer Text and Turn Footers On (>EF'text')**

A left-hand page would need the page number printed on the left -

>EF Page number %

## **Designing a Header or Footer Line**

To design and type either a header or footer line, begin by just typing it in a blank line at the top of your document. Arrange the text as you wish, including any centring or justification, and then add the header or footer command at the start of the line.

# **Print Control Commands**

## **Page Number of Next Page (>PN n)**

Normally the first page printed is numbered one and each time a new page is printed in the same document the page number is incremented by one. However, if you want to change this scheme you can use >PN to change the page number to anything you wish up to '999'.

## **Continuous/Single Sheet Printing (>CP ON/OFF)**

When a printer uses single sheets, you may want to stop Notepad sending text to the printer after each page. This would allow you time to get the next sheet into your printer. Setting >CP to OFF will cause Protex to display 'Press SPACE to print' on the screen at the start of the print run and at the start of each page. Pressing any other character (apart from Space) will skip that page. You could use this facility to print a random selection of pages.

If you use an automatic sheet feeder on your printer, Continuous Printing should be set to ON.

The default setting is ON.

## Number of Copies (>NC n)

Do you want to print several copies at the same time? Just put the number of copies you want here.

This command is ignored if there is a data document defined. See Mail Merge for details of a data document.

Setting this value to zero prints an infinite number of copies. Pressing the STOP key stops the print (fortunately).

The default value is one.

## Line Spacing (>LS n)

This defines the number of Linefeed characters there are for every printed line. You can use half line feeds (e.g. have one and a half line spacing) if your printer supports it. There is only one way to find out and that is to try it. An example is >LS 1.5.

The default value is one.

## Which Pages to Print

### Start at Page Number (>SA n)

This sets the number of the first page in the document. The default value is one.

### End Printing at Page Number (>EA n)

This sets the number of the last page to be printed in the document. The default value is the last page.

### Print Even Pages Only (>PE ON/OFF)

### Print Odd Pages Only (>PO ON/OFF)

If you place either of these commands at the start of your document you can print on both sides of the paper. Print all the odd pages first, then turn all your sheets over, (making sure that they are still in the correct order), and print the even pages.

Setting either of these commands to OFF will print all the pages, which is the default.



## **Enable/Disable New Page at The End of Printing (>NP ON/OFF)**

This command ejects the last page of your document out of the printer. It is normally turned ON. Set this command to OFF if you want the last page to remain in your printer after printing the document.

## **Page Throw (>PA or Control P)**

### **Even Page Throw (>EP)**

### **Odd Page Throw (>OP)**

The Page Throw command breaks your document up into pages. It allows you to say where you want to end a page in your document. Page throws are automatically put in by the word processor when it has worked out that you have no more lines left on that page. It is shown by a row of three lines of dashes across the screen.

Paragraphs or tables that you want to stay together can be split by page throws. The word processor doesn't know the significance of the text that you type. It will just chop your document into the number of lines on a page as specified on the Layout menu.

With the page throw commands, you can decide where the end of the page is. To keep a paragraph together that has been broken up by a page throw, put a page throw command at the top of it. This will mean that the page will be broken above the paragraph. The whole of the paragraph will be put on the next page. You could use Control P to type the command in quickly.

If you want a particular page to start on a right-hand or left-hand page you would use the Odd or Even Page Throw commands instead. If the page you have selected to be on the right-hand side falls on an even number (which would put it on the left normally), Protex will move the text onto the next page leaving that page blank.

## **Conditional Page Throws**

Each of the above three page throw commands can optionally include a number. This number tells Protex to check how many lines there are left on the page. If there are fewer lines than the number you have put in, the page will be broken where the command is. This is useful when you do not want a paragraph split in the middle.

When you are working on a document, you do not know where anything is going to be by the time you have finished editing. If you had a 25 line paragraph that you wanted kept

together, putting the command >PA 25 just above it would ensure that it would all stay together on one page. The Conditional Page Throw will only have an effect if there is not enough room left for the whole paragraph. It will start a new page at the beginning of the paragraph.

## Form feeds Enable/Disabled (>FF ON/OFF)

The form feed character tells your printer that it has reached the end of the page and it must move to the top of the next one. This is a useful command if you are using either a laser printer or a printer with a sheet feeder. The default setting is OFF.

---

### TIP - Help With Page Lengths and Formfeeds

There are some tell tale signs that point to an incorrect setting for page length and/or formfeed commands. Let's try to understand what the printer and your word processor are trying to do.

The important point to remember is that the word processor only knows about the page length. This is the number of lines of text that will fit on a page. The printer, however, knows about the page length and the paper length. The paper length is the physical length of each sheet of paper. The page length and the paper length are not necessarily the same.

There are two ways that the word processor can tell the printer to move to the next page.

One way is to give the printer multiple Line Feed characters. These add the correct number of blank lines to finish one page and move onto the next. For this to work correctly, the page length that Protex is using must agree with the actual length of the paper. If the two values don't agree the first line of text on a new page won't start at the top of a new sheet of paper.

The other way for Protex to move to the next page is for it to send the printer a Form Feed character. Form Feed characters tell the printer to move onto the next page. In this case the printer has control of the page length. For this to work correctly the printer needs to know the length of the paper you are using.

You may need to consult your printer's manual to find out how to change the paper length. Normally you can either change the printer's default paper length setting with switches/buttons, or you can send the printer a special command from the word processor (see Outputting codes to the printer).

It may sound crazy to have two different ways of moving to the next page but there are circumstances where this is necessary. Suppose you wanted to print address labels. Typically there are at least eight labels on each sheet of paper. Rather than printing just one address on each sheet, you can tell the word processor that the page is very short. It will think that it is moving to a new sheet of paper after each page but in fact it is only moving down your single sheet. The printer will be able to move to the next sheet after your eight labels have been printed.

To test the printer's paper length setting put a sheet of paper into the printer then, with the printer 'off-line', press the formfeed button. If the paper length setting is correct the printer will fully eject one sheet of paper and feed the next sheet through ready to start printing on the first line.

If the next page is not at the top, or the previous page has not been ejected, then you have the page length set incorrectly. Remember, for A4 paper the page length is 11 - 2/3 inches.

---

## Formatting Commands used with Mail Merge

There are several commands that are normally only used when mail merging. Mail merging is a topic dealt with in a separate chapter but we have included the formatting commands here for completeness.

When using mail merge words and extra lines are inserted into the text during printing. The final layout will not therefore always be the same as that displayed on the screen. Using the Formatting, Centre Line and conditional Page Throw commands that work while printing is in progress, you can make sure that the printout stays the way you want it.

Most of the formatting commands that we have already covered act immediately on the text as you type it in. If you type a line of text and then issue the command to centre it (Control =) the text is immediately centred. If you go back to that line and edit the text, the line will no longer be centred. You would have to issue the command again.

The commands to format while print behave differently. There are only actioned at the time of printing. Using the same example of centring text, if you included the command >CE above the line to be centred, Protex would perform the centring at the time the line was printed. Therefore you could edit the text to your heart's content, knowing that it would always be centred on the printout.

## **Formatting Whilst Printing On/Off (>FP ON/OFF)**

You use this with the Right Justification command if you want your text to be formatted during a Mail Merge.

You must remember to turn it OFF at the start of any tables, however, and ON again at the end of the table. If you forget you could find that reformatting has destroyed your table's layout.

## **Right Justification On/Off (>RJ ON/OFF)**

This command works in the same way as Control J apart from the fact that it will re-justify during printing. It will only take effect if you have included the >FP command to turn on formatting while printing. The default value is the same setting as in the Configure menu within Protext.

## **Centre Line (>CE'text')**

Use this command instead of Control C if you wish to centre your text when printing.

# **Special Hidden Characters**

There are other characters that you can use that tell Protext to do different things when formatting the text.

## **View Tabs and Returns (Control V T)**

You can display these characters by pressing the Control V and T keys. Pressing them again will turn it off.

## **Insert Space (Control Space)**

When you are in 'Overtyping' mode, you can use this command to insert spaces into your document.

## Special Formatting Characters

There are some special formatting characters that are used with the word processor.

### Soft and Hard Returns

There are two types of carriage return characters that Protex uses to mark where the end of line is. They are soft and hard return characters.

If you turn Word Wrap ON, the word processor is constantly formatting your text. When you reach the end of one line, it inserts a Soft Return character. It needs this character so that if you decide to change the length of the line and reformat the text, it knows which character it should change.

This is different to the character that is used when you press the ENTER key. This is called a 'Hard Return' and is used to mark the end of a paragraph or a line in a list.

These characters are not normally displayed but they can be shown on the screen by selecting View Tabs and Returns (Control V T).

### Non-break space (Control N space)

You can use this character to 'glue' two words together with a space. This stops the two words from being split across two lines. It also prevents any 'soft spaces' being added between the words if you have selected the Right Justification command. You can use non-break spaces, for example, when typing a sentence or word that you want 's p a c e d o u t'.

To select this character, use Control N followed by a space.

You can also select this command from the Text Formatting menu.

### View Hard Spaces (Control V S)

Hard spaces are put into your text when you press the space bar. You normally see them just as spaces between words but they are an actual character.

To display the hard spaces as a character, use this command. It is a toggle, so you can change it back with the same command.

## Using Hyphens

When a hyphen '-' is used between two words in the text, if the hyphen falls near the end of the line it will normally be used to split the line with a 'Soft Return'. You may not want this to happen, so there are some special ways how you can change this.

### Non-break Hyphens (Control N -)

You can use a hyphen to glue two words together in a visible form. Do this by using a 'Non-break Hyphen'. To select this character, use Control N followed by a hyphen '-'.

### Soft Hyphens (Control H)

Sometimes, it is quite acceptable to split a hyphenated word across two lines. Unfortunately, the word processor doesn't normally do that because it can only 'wrap' whole words between different lines. You can use Soft Hyphens to tell the word processor where it can split the word in two.

The good thing about a soft hyphen is that you can't see it if the word is anywhere on the line except the very end. It only becomes visible when the word needs to be split between two lines.

To select this character, use Control H. The hyphen will be shown inverted on the screen but will only be printed if it is at the end of the line.

# Chapter 8

## Using the Style Attributes

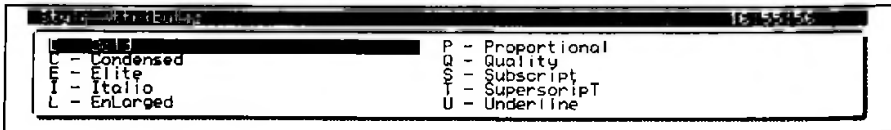
Using the what? On the Notepad, what we would have called print commands are called 'Style Attributes'. These are things like underlining, bold, italics and so on that can be used to make your document look more professional.

Normally when you start printing text, it is in ordinary text. This is technically known as 'pica' and is ten characters per inch. Whether you can use other types of print depends on what type of printer you have. We are going to assume that you have a printer that can print all these for now.

### Selecting Bold, Underlining or Italic Print (Control 7, 8, -)

You select a Style Attribute by pressing the relevant key sequence at the beginning and end of the text you want the attribute to apply to. Some of these marks can be selected two different ways. These attributes, like Bold, Underline and Italics are shown on the emplate. Use the shown key sequences (Control 7, 8 and -) to select them quickly.

These and all the other attributes are in the Style Attributes menu. You can see a list of all the possible attributes by pressing the Menu key followed by S.



These are, Bold (emphasised), Condensed print, Elite (twelve characters per inch), Italic, EnLarged, Proportional, Quality (near letter quality NLQ), Subscript, Superscript and Underline.

### Short Cut Style Attributes (Control X)

Alternatively there is a short cut key sequence for selecting these attribute codes. It is designed for those who have good memories. Use Control X followed by one of the following letters:

b	Bold	q	Quality
c	Condensed	s	Subscript
e	Elite	t	superscripT
i	Italic	u	Underline
l	enLarged		
p	Proportional		

## View Codes On/Off (Control V V)

When you use these 'Style attributes' you have a choice about how they are displayed on the screen. With the View codes turned on, if you select underline for example, you will see an inverted 'u' at the beginning and end of the text you have marked for underlining. If you turn the View codes off, you will see the text underlined on the screen. The Codes on/off command is on the template. Press Function 4 to toggle this command.

## Scrolling

As the display is normally smaller than the document you are typing, Protex needs to scroll your text so that you can see what you are working on.

There are two types of scrolling that Protex uses. These are Vertical Scrolling and Horizontal Scrolling.

### Vertical Scrolling (Shift Cursor Up or Cursor Down)

As you type more and more lines of text into your document any text above the line you are on will be moved up the screen until it disappears off the top. This effect is called 'Vertical' scrolling.

You can use the Up/Down Cursor keys to scroll through the entire document. The cursor can move to lines that have already been scrolled of the top or bottom of the screen. You can move the text on the screen so that you can see any point in your document.

Sometimes, it is useful to leave the cursor where it is but scroll the text up or down so that you can read the text surrounding it. You can do this by pressing the shift key with the up or down cursor keys.



## Horizontal Scrolling

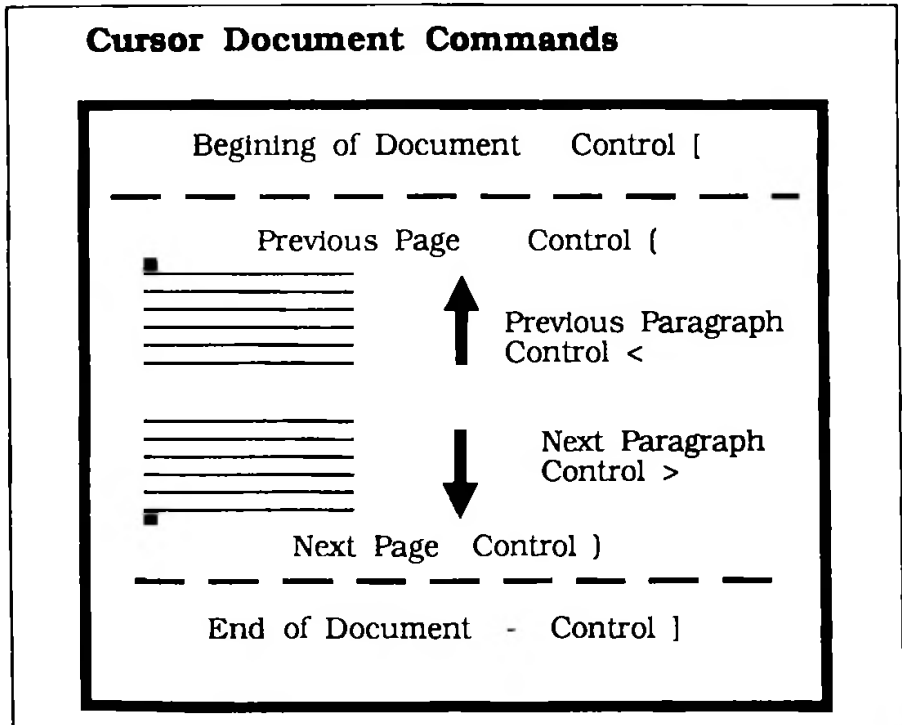
This type of scrolling is not used as often as vertical scrolling because the default ruler is set to the width of the screen. With Word Wrap ON, the text stays within the screen size.

When the document is wider than the screen, or Word Wrap is turned OFF, then horizontal scrolling will take place. If you find that you lose the left-hand side of your text, this is due to horizontal scrolling. To get back to the left-hand side again use the Shift and Enter keys.

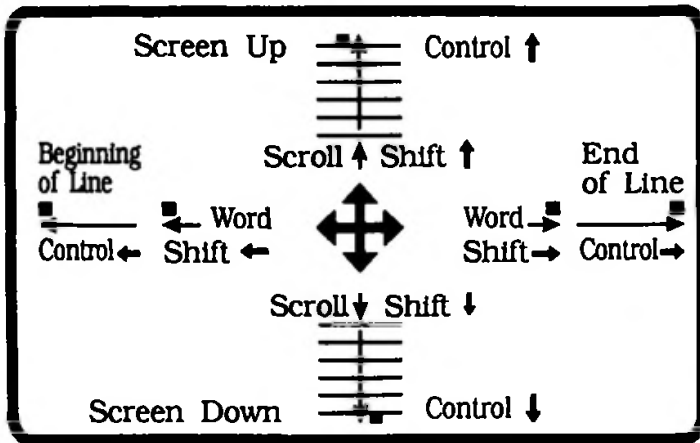
## Turbocharging the Cursor Keys

When a document becomes larger, having to move the cursor a single line up or down and a single character left or right becomes tedious. Is there a faster way? Fortunately the answer is yes.

The diagrams below show both the cursor movements on the screen as well as moving around the document.



## Cursor Screen Commands



If you know where you want to move to there are commands that will help you get there a great deal faster than by just using the cursor keys.

## Page Mode (Control Shift P)

Page mode will change the information displayed on the Status Bar. Normally the Status Bar shows where the cursor is by Page No, Line No and Column. Page mode changes it to Character Number, Global Line number and Column. In this mode you can't move forward and backwards a page with Control 9 or Control 0 command.

You can use it to show the global line number for use with the following command.

## Goto Line / Page / Column (Control G Ln / Pn / Cn)

If you know where you wish to move to in your document, 'Goto' will help you move around your document quickly. It will either move your cursor to the top of the page number you have asked for, or to the line number or column on the page you are now on.

The line number is not normally displayed on the Status Bar but it can be displayed by turning off the Page mode.

If, for example, you wish to go to the top of Page 4, you would type Control GP4.

### Markers

There are several marker commands that allow you to leave bookmarks in your document so that you can quickly return to them at any time. Think of it like a number of differently coloured balls of wool that you can use to find your way back to the correct marker.

There are four types of markers that you can use. They are absolute, multiple, margin and block markers.

When you set a marker it is shown as an inverted number 0 - 9, ?, [ or ] depending on which character you have selected. To remove any markers, use the delete key to delete the inverted character.

---

#### TIP Markers Are Only Shown When Codes Are On

All markers you put in are saved in your document. If you have your 'Codes on/off' set to OFF (Function 4), you will not see your markers even though they are still there. So if you lose them, remember this tip.

---

### Absolute Markers (Control @ 0 to 9)

You have ten absolute markers (numbered 0 to 9), which allow you to define absolute positions in your text to which you can return. The first time that this command is used with a number, it sets the marker. Subsequent uses of the same command then return you to that marker.

### Multiple Markers (Control @ ?)

Multiple markers are selected by pressing the '?' after Control @. You can have any number of these markers within your document.

You can use multiple markers to allow you to move quickly to common points in your document. For example, if you added a multiple marker at the start of every main paragraph, the commands detailed below would allow you to skip through the document a paragraph at a time.

## **Goto Previous Marker (Control Shift 5)**

## **Goto Next Marker (Control Shift 6)**

These two commands allow you to jump to all the markers you have set in order. It is the only way that you can get to your 'Multiple markers'.

A 'bug' in early versions of the Notepad prevented the Right Shift key working with Control 5. This is not a serious problem since it is more convenient to use the Left Shift anyway.

## **Goto Margin Markers (Control @ L or R)**

This moves the cursor to the left or right margin.

---

**TIP Use Goto Left Marker to Move To a New Left Margin**

Control and Left Cursor always moves the cursor hard over to the left of the screen. Control @ L moves the cursor to your left-hand margin. This is very useful when you have redefined your left-hand margin with a new Ruler Line and need to move there.

---

## **Goto Block Markers (Control @ [ or ])**

We haven't covered 'Blocks' yet, but you can use this command in a marked block to goto the beginning (Control @ [) or the end (Control @ ]) of a marked block.

Although there are separate commands to mark a block, the first time that this command is used, it also marks the block. So you have plenty of choice on which key strokes you use.

## **Goto Last position (Control L)**

Protex remembers the last two places that your cursor was within the document. Repeated use of the command Control L will move you backwards and forwards between these two points.

Now you know how to move, let's dance!

Well, we've covered many different ways of moving around the document. Let's look at a different set of commands that actually do things to your document.

## The Delete Commands

Just like moving the cursor, there are quicker ways of deleting the text in the document than just using the two delete keys.

- Delete line (Control 3)
- Delete to Start of Line (Control <- Del)
- Delete to End of Line (Control Del ->)
- Delete character before cursor (<- Del)
- Delete character at cursor (Del ->)
- Delete word left (Shift <- Del)
- Delete word right (Shift Del ->)

### Undelete (Control U)

There is also an Undelete command that remembers what you have just deleted and puts it back where the cursor is. This helps if you have deleted something by mistake.

When you delete anything (including a 'Block' of text, see later) other than a single character, the text is stored in a temporary memory until you use another delete command. If you leave the cursor alone after you have deleted something, undelete will put it back where it was.

If you move the cursor and then use the Undelete command, what you deleted will be inserted at the new cursor position. You can use this to move or even make several copies of the text that you deleted.

### Special Insertion and Deletion Commands

We have already covered commands that insert and delete lines, (look at your template if you've forgotten them). Here are a few that are slightly different.

#### Transpose Characters (Control A)

Do you ever type letters the wrong way round? Here is a good command to swap these letters around. Try mistyping would as wuold, then move the cursor over the 'u' and do the command above and by magic the uo changes to ou.

#### Changing Cases

There are occasions when you may have wished to either change something that you have

already typed into CAPITALS or visa-versa. Here are the two commands to use.

### **Lower to Upper Case (Control /)**

Move the cursor to the start of the text you wish to change. Hold down the Control key and press the / key until you reach the end of what you wish to change.

### **Upper to lower case (Control \)**

Here it is the other way round as well.

A good way of remembering these commands is with the direction of the line. Lower to upper case starts at the bottom left to the top right. You could imagine that it can make letters larger. The same is true the other way round.

# Chapter 9

## Blocks

One of the most powerful functions of any word processor is the ability to work with blocks. This allows you to mark a bit of text that you have already typed and then do something with it, like deleting, moving or copying it.

As this is a two-stage operation, we will first start by marking the block, then when it is marked do something with it.

### **Mark Block (Function 9 or Control Z)**

There are two different key strokes that do the same thing. You need to mark the beginning and the end of the block of text that you want to do something with. Place the cursor at the beginning or the end of the block, use the command, and then do the same thing again for the other end.

Once you have used the mark block command twice, you will see that the text within the block is now shown as inverted text. At the beginning there is the character [ and the end of the block there is the character ]. You don't need to worry about these characters as they are not actually in your document. They are just used to highlight the beginning and end of the block.

### **Clear Block (Control K)**

If you make a mistake marking a block, the simplest way of putting it right is to clear it by unmarking it and then marking it up again. You can also use this command if you have finished using a block.

### **What You Can do With a Marked Block**

Once you have marked a block, there are six things you can do with it (apart from clearing it as above).

Block Delete (Function Del)	Yes, this command deletes the block. If you didn't mean to delete it after all, remember the Undelete command Control U
Block Word Count (Word processor menu Editing, Count words)	This allows you to count the number of words you have in the marked block.
Copy Block (Function O)	You can copy a block of text from one place in the document to another. The block will be copied to wherever the cursor is when you enter this command.
Copy Block/Document (Word processor menu Copy Block)	This is different to the copy command available by pressing Function O. This command will copy either your marked block or your whole document to another document.  It is particularly useful for moving sections of text between documents and in fact is the only way to make a copy of a document in the Notepad
Block Print (Word processor menu Print Block)	Use this command if you only want to Print part of your document.
Block Move (Control M)	This command will move a block to wherever the cursor is when you issue the command.

## Search, Find & Replace

These commands are very useful when writing large documents. You know that you may have covered a particular subject but you can't remember where you did so. Using 'Find' will help you, as you can move forward or backwards within your document looking for every occurrence of a word or phrase.

Replace is really a Find and Replace. You look for a word and replace it with another word. This can be used to change the spelling of somebody's name throughout the whole document. It is useful if you need to change a word several times.

With both these commands the searching starts from where your cursor is when you use the command.

---

### TIP Get the Spelling Right When Using Find and Replace

When looking for a word, part of a word, or a phrase, make sure you spell it the same way you used in the text otherwise Protext will not find it.

The thing to remember with these two commands is that they will only do what you tell them to do. So if you have typed 'Jane' and the correct spelling is 'Jayne', you could find Jane and replace with Jayne. It will only replace the ones that you have



**spelt like that. It will not recognise anything that you have spelt differently even if you didn't mean to. It is therefore a good idea, unless you are an immaculate typist, to use this command globally only after you have proofread your document.**

---

### **Find (Function 5)**

When you select this command Protex will ask you to enter the string to search for with the message 'FIND string:'. No, this is not the time to look for the ball of string that the cat was playing with a week ago. You can type in a word, part of a word or even a phrase for Protex to search for.

If you asked it for an 'a' for example, Find would find every single 'a', even those within a word. If you asked it for ' a ', that is an 'a' with a space on either side of it, your choice would be narrower and Find would only find an 'a' on its own. Alternatively, you could choose 'Whole word' which would look for your match in a single word.

Pressing the ENTER key shows you the following options - You can use any combination of these to help narrow down your search.

#### **A-All**

Normally when Find finds what you have asked for it stops on the first match.

Selecting 'All' will put Find into express mode so that it will not stop until it has found all the occurrences of your search string. Thus Protex counts the number of times your search string appears in the document starting from wherever the cursor was to the end.

#### **B-Backwards**

Backwards puts this operation into reverse. Instead of looking forward in the document for your search string, Find goes backwards from where you are to the beginning of the document.

Using this option with Find or Find Previous will cause them to do the opposite to what you expect. Find Previous will Find forward and Find will Find backwards. Keeps you on your toes though.

#### **C-match Case**

Normally Find doesn't care whether what you are looking for is in upper or lower case. If you are particular, use this option and make sure that you have typed the correct case in

your 'Find string' section. This is very useful when used with Replace as you can ask for a word without any capital letters and replace it with one that has.

For example, Find 'brighton' and replace with 'Brighton'.

## G-Global

Use Global if you wish to check the complete document rather than just from where the cursor is to either end of the document.

## W-whole word

To restrict your choice during a search even further, choosing this option will only select your match if it is a whole word. For example if you are searching for 'and' the Find command would normally find words like 'sand' as well. Using 'whole word' would only find 'and'.

## n-Nth Occurrence

This command counts the number of matches (between one and 255) it finds before stopping. This could be used for example to search for every second quotation mark.

## Special Characters

When using Find to look for certain characters and printer codes, you need to put an exclamation mark in front of them. This is because these characters are used differently in 'Find' or 'Find and Replace'. Here is a list of these differences.

Printer control codes ! followed by the letter of the printer code.

Question mark	!?
exclamation mark	!!
hard return	!.
soft hyphen	!-
non-break hyphen	!_
non-break space	! space
search for a specific code	! number

## Using ? in Find

If you are not sure of a letter or a group of letters, you can use the '?' character to indicate the unknown letter. For example, if you wanted to find 'could' but you couldn't

remember how you spelt it, you could use 'c???d' which will find any five letter word beginning with a 'c' and ending with a 'd'.

### **Replace (Function 6)**

Replace works just like Find except that you get an additional line 'REPLACE with' asking you what you wish to replace the word or string with that you are searching for. After using this, you will get the same options as described above.

The only point to be wary of is that if you use the 'All' option you need to be sure that the command will do exactly what you think it will.

### **Other Find Commands**

#### **Find Next (Control 6)**

#### **Find Previous (Control 5)**

Once you have used aFind or aReplace, you can use either of these commands to repeat the command without having to enter the search string again.

# Chapter 10

## Spell Checker

After typing a document, it is not a bad idea to run the Spell Checker as it can often find words that are either misspelt or mistyped. It WILL NOT find ALL your mistakes as a spell checker only looks at each word you have typed and compares it to two dictionaries. It has a main dictionary that has a large number of words already included and a user dictionary that is initially empty.

### What the Spell Checker Does

If the spell checker cannot find a match between a word in your document and one of its dictionaries it will let you know. It lists your options for that word and waits for you to select an option. Then it continues checking the rest of the text.

Spell checking can be stopped at any time by pressing the STOP key.

It will not tell you if you had put 'bet' instead of 'but' as those are both valid words. It will find a spelling mistake if you had typed 'bwt' (which may look like but) and will give you a choice of options.

The spell checker will look out for words where capital letters have been put inside the word. For example the spell checker will want to change 'tHEy' to 'they'.

### Spell Checker Commands

There are three commands that start the spell checker.

<b>Spell check the complete document (Function F)</b>	Use the 'Spell Text' command after you have typed in your complete document and you want to spell check everything.
<b>Spell check word to the right of cursor (Control I or Control Q)</b>	Use the 'Spell Word' command while you are typing and you are not sure of the spelling of a particular word. If the spelling is correct, you will get a message on the menu bar, 'Word is in dictionary'.
<b>Spell check the document from the cursor (Control S)</b>	You can check part of a document by using Control S to start checking from where the cursor is and then pressing the Stop key when you have reached the end of the section to check.

## Spell Checker Rules

Here are a few rules that the spell checker follows.

Foreign letters used in words are recognised and the words can be added to the user's dictionary.

All possessive apostrophes ('s) are ignored but contractions of two words (can't, won't, haven't) are stored as separate words in their own right. The main dictionary contains most of the common contractions.

Words beginning with numbers (1st, 2nd, 3rd) are ignored.

## What Happens When the Spell Checker Finds a Mistake?

You get a choice of four options. These are:

- I - Ignore word
- L - Lookup word
- E - Edit Word
- S - Store Word

### I - Ignore Word

This does as it says; it ignores that word and makes no changes at all. If you select this option it means that you are happy with the spelling even if the spell checker isn't.

### L - Lookup Word

This allows the spelling checker to have a look in both dictionaries and make some intelligent guesses at what the word might be. Initially you should see the 'Looking . . .' message followed by a list of close matches or nothing at all.

You use the Cursor up or down keys to select the correct word (if it is there) followed by the ENTER key. The word that was wrong in your text will then be replaced by the word that you have selected.

If the word you wanted is not on the list, then press the Stop key to exit the selection. You will then get the following options 'E - Edit word' and 'Enter - Continue'. If you choose 'Continue' then the spell checker will leave the wrong word alone.

## **E - Edit Word**

Edit Word will show you the wrong word in a small window so that you can correct it. When you press ENTER, the spell checker will check the word you have edited before moving onto the next mistake.

## **S - Store Word**

This command adds the selected word to your own user dictionary. This word could be your name or a special word that is not used often. It is a good idea to build up your user dictionary as it will save you time in the future.

## **User Dictionary**

After storing some new words with the above command, you can use the following commands to view the words you have entered or to delete them.

### **View User Dictionary (Word processor menu, Editing, View user dictionary)**

This command will show you all the words you have put into your user dictionary.

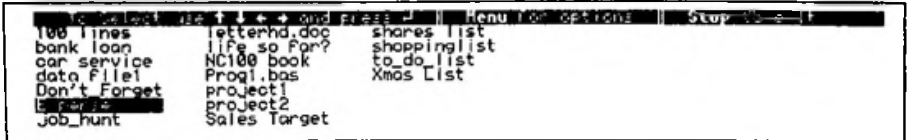
### **Remove Word From User Dictionary (Word processor menu, Editing, Remove word)**

You need to type the word to delete from the user dictionary and press the ENTER key. It is 'Case sensitive' so make sure that if the word in the dictionary starts with a capital letter, you do likewise when entering the word to delete.

# Chapter 11

## The List (Function L)

The list is where all your documents or files are stored. You can select any of your documents/files that you have created in your Notepad. If you want to re-read, print or do some more work on a document, you need to get the Select menu on the screen.



From the Main menu of the Notepad you use Function and Left Cursor keys to select the Word Processor menu and then the Right Cursor key.

Alternatively you can use the short cut key sequence (Function L) from any other program in the Notepad.

When you leave the Word Processor, Protex automatically saves your document for you under the name you gave the new document before you started typing. The new name gets added to the List.

---

### Definition - Documents and Files

There is no difference between a document and a file. Generally we have used the word document because it is more descriptive of the 'thing' that holds the text. Other computer systems tend to use the word file.

There are times when the word file makes more sense. The Address Book for example holds a list of names and addresses in a manner analogous to a paper filing system. BASIC programs are stored in a file and the programs can store information to a file.

The distinction we use is that documents can be loaded into the word processor; files are used to save other types of information.

You just need to remember that in reality both words mean the same thing.

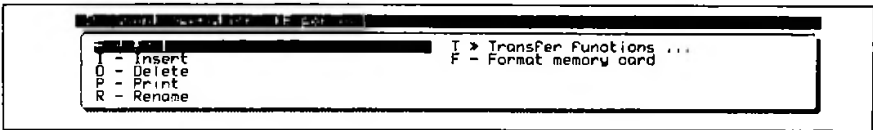
---

## Select a Document

To select a document to use with the Word Processor from the list, just move the black bar cursor over the name you want and press the ENTER key.

As well as selecting an existing document, there are several other things that you can do from the List. The important thing is to select the document first and then press the Menu key to select what you want to do with it.

## Document Operations Menu



When you have entered this menu make sure that the name of the document you wish to operate on appears at the top of the screen.

Here is what you can do and what you should not do with your documents.

### E - Edit

This command performs the same function as hitting the ENTER key from within the List.

### I - Insert - DO NOT USE THIS COMMAND

DO NOT USE THIS COMMAND. The Insert command should only be used to insert one document into another. If you have reached the List from the Main menu there will not be a document open to insert the one you have just selected into. The result will be at worst a computer crash or at best a warning. YOU HAVE BEEN WARNED.

---

### Definition - Bugs and Crashes

Complex programs written for computers like the Notepad owe their existence to a team of programmers who design and write the programs. No matter how much testing is done on a program they can never be completely certain they have found all the bugs. This is true for all large computer programs.

Sometimes bugs manifest themselves in trivial ways that cause no serious problems. A good example within Protext is the fact that you cannot use the Right Shift key to access the 'Find Previous' command.



Unfortunately there are times when a bug is serious and this is the case with the insert command in the List. If you use the command as it was intended to be used it works correctly, but if you select it without having another document to insert into you may lose some or all your documents in the Notepad. Then you would start wishing you'd done a backup.

While writing this book on the Notepad, we've come across a few bugs and have highlighted the problems they can cause. This does not mean that we've found them all; you will probably find more if you are inquisitive.

Crashes are the disasters of the computer world. They come in different guises but the results are normally the same. You will not be able to do anything with your Notepad until you have done a Reset. You will have definitely lost the document you were working on and if you are unlucky you will have lost all your other documents as well.

---

### **D - Deleting Documents**

As you type in more documents in your Notepad you are filling its available memory. You can release this memory for new documents by deleting any old documents that you no longer require.

### **P - Print**

Here is another place that you can print documents from as well as the Print File menu.

### **R - Rename**

Rename allows you to change the name of a document.

### **F - Format Memory Card**

When you get a memory card to store even more files on to you need to format it first. More about that later.

### **T - Transfer**

Transfer brings up another menu with more commands on it. These are used when you transfer documents to another computer. We will cover these in the Serial Terminal section later.

---

**TIP - Only Copy a Block or Document From the Word Processor**

You can't copy a document from the List. You must select the document you want first and then go into the Word Processor. Selecting 'Copy block or Document' will allow you to make the copy.

---

## Showing Other Files

The most useful way of showing the list of documents in your Notepad is to show only the name of the document. This allows you to see thirty-five names at a time when you enter the List. In this case only documents that have been created by the word processor are displayed. Files containing BASIC programs or your Address Book are hidden from view, as is the size of each document.

You can change the way documents are displayed by setting the 'Document sizes and date display' option in the System Settings menu to anything other than 'Not shown'. The List will then include more information about each document and show all the documents in the Notepad. The disadvantage is that you will only be able to see the names of fourteen documents at a time.

ADDRESS BOOK	104	U	20	-1	-92	17:00	Job hunt	NNNNNN	U	20	-1	-92	17:00
bank loan	2	U	20	-1	-92	16:30	Letterhd.doc	NNNNNN	U	20	-1	-92	16:30
car service	2	U	20	-1	-92	16:50	life so far?	NNNNNN	U	20	-1	-92	16:50
data File1	1	U	20	-1	-92	16:50	NC100 book	NNNNNN	U	20	-1	-92	16:50
Don't Forget	2	U	20	-1	-92	17:04	Progl.bas	NNNNNN	U	20	-1	-92	17:04
							project1	NNNNNN	U	20	-1	-92	17:04
							project2	NNNNNN	U	20	-1	-92	17:04

---

**WARNING - Don't Try to Select a File That Isn't a Document**

After changing 'Document sizes and date display' to show all the files, be careful NOT to select any of these other types of files with your word processor because it could have disastrous results.

If, for example, you selected a BBC BASIC file from inside Protext the file would be corrupted and you couldn't use it again.

To avoid this from happening you could change the setting back to 'Not shown' which will prevent you from selecting any other files as they would not be shown on the list.

### **Only Select documents created by the word processor.**

---

The extra information shown by the List is the document size expressed as a number of bytes, whether the document is stored on a memory card, whether it is in Upper memory or Lower memory and the time and date that the document was created.

## **Memory**

Your Notepad has just under 49K bytes (1K is 1024 characters) of memory that has been split into two areas. These areas are called 'Upper' and 'Lower' memory. If you plug a memory card into your Notepad you are adding extra memory in which to store documents.

Each character that you type (including spaces) into your document takes up one byte of memory. The memory is used in blocks of 256 bytes. You will notice the number of bytes of 'Free memory' jumping down in steps of 256 as a document gets larger.

Lower memory is used by the programs within the Notepad to carry out your instructions. Documents are stored in upper memory but if this area becomes full then lower memory starts to get used for documents as well.

When you enter the word processor to edit a document Protex needs an area of lower memory to operate in. This area must be large enough to hold your whole document.

The number of bytes used for each document is followed by a 'U' for upper memory, a 'L' for lower memory or a 'C' for the card. This information can help you to decide which files to get rid of when you are low on memory.

## **Running Out of Memory**

When you get the 'memory full' message it is time to delete some old documents. It does not matter if you delete the documents from upper or lower memory. If you have freed up some space in upper memory by deleting a document and you then edit a document previously stored in lower memory, Protex will move the document to upper memory for you.

Your Diary entries are also stored as files. Deleting them in the diary will also clear more space.

If you no longer require a particular document, you can just delete it altogether. If you

want to save the document there are several ways to copy it from your Notepad.

Memory cards can remember what has been saved onto them even when they are removed from the Notepad. You could keep one card just to hold all your important documents.

If you have access to another computer, in particular a PC, transferring a document onto that computer's storage system is a cheaper approach than keeping an extra memory card.

You can even connect the Notepad to a stand alone disk drive to give you as much storage as you wish.

We will be covering transferring your documents to another computer later.

## **How Big Can Your Document Be**

No matter how much memory a computer has, there is always a limit on how big a document can be. Protext uses the largest continuous block in the lower memory. The largest document that you can have is 38144 bytes, which is roughly eighteen A4 sheets.

## **How to Print Multiple Documents**

If the restriction in document size is 'cramping your style' there is a way of getting round it. You could get more memory by buying an extra memory card. That wouldn't mean that any one document could be larger than 38144 bytes but it would mean that you could have more of them.

There is a way of joining documents together and printing a really long document from several smaller ones. You can use the Insert command. Look out for details in the 'Commands' section later.

# Chapter 12

## Printing

### Print Test Page

Using all the commands we have covered so far, a good tool to make is a 'Print Test Page'. This does several things.

It checks that your printer and Notepad are set up correctly.

It finds out where your text is positioned on the page. This can be used for lining up where address labels are on the paper, where addresses need to be for use in window envelopes or lining up pre-printed forms.

The ideal way of doing this is by printing the test page on a transparent sheet (the sort they use for overhead projectors). Then you can see what you want the text printed on as well as where it will be printed.

The position of your text on the page depends on two things. First the word processor you are using and secondly the printer you have. Writing a test page with the word processor and then printing it with your printer will show you where characters will be printed on the paper.

### How Wide Can Your Printer Print?

Most printers can print up to eighty characters on a line in normal mode or one hundred and thirty-two characters on a line in condensed mode. What is the best way to find out? Send the printer a line that is far too long and see how many characters fit on the line. When your printer can't print any more characters on the line it will either put the remainder on the following line or throw it away.

### Typing a Wider Line

The default right margin is normally set at 70. We need to widen this to see when the printer runs out of characters. You need to decide which mode you would like to test your printer in before making the long line that follows.

Start a new document called 'longline' and change the ruler line so that you have either eighty or one hundred and thirty-two characters in the line.

Can you remember all the necessary commands yet? Here's a reminder.

Add a ruler line into the text (Control R), move the cursor onto the ruler line and Goto (Control G) column 132 (c132) or Column 80 (c80). Put an 'R' there, turn Insert on/off to OFF (check the Status Bar) and Goto (Control G) column 70 (c70). Overtyping the whole line with '-' until you reach the new 'R' at column 132 or 80. Remember to change back Insert on/off back to ON when you have finished.

All you need to do now is to type in some text that uses the new width of the document and print it. A simple sequence of numbers will allow you to count the printed width easily.

When you know the number of characters that you can type across the page you need to find out how long the page is. Again, use the trick of typing more lines than the printer will print on the sheet. Seeing which line remains at the bottom of the sheet will tell you the number of lines your printer can print on a page.

## Page Layout (Function 7)

Overtyping or ← → to change, ↑ ↓ to move, Stop to Finish	
Page length (lines)	(66)
Top margin (lines)	(3)
Header margin (lines)	(2)
Footer margin (lines)	(2)
Bottom margin (lines)	(3)
Side margin (chars)	(5)
Line spacing (lines)	(1)

Let's see how Protex lays out the text on the page.

Protex uses a default set of values for all the margin settings. This means that the first character on each line of text is not printed hard over to the left of the sheet of paper as you may have expected. The default value for the side margin is '5'. This causes five spaces to be printed at the start of every line.

The same is true at the top and bottom of the page because there are margins there as well. At the top of the page there is the Top Margin that is set to three lines. This is followed by two lines for the Header Margin that is used for any header text. At the bottom there is a similar set called the Footer Margin and the Bottom Margin of two and three lines respectively.

---

### TIP - Page Length Warning

Do not set the Page length to zero or greater than 126 lines. Protex will ignore your setting of zero page length and will use your previous setting. If you use a page

length greater than 127, you will discover that the page length will be set to that number less 128 lines. This effect is similar to a mileometer in a car, when it reads 99999999. One mile later and you have a new car because the mileometer has run out of numbers.

---

## Getting Rid of All the Margins

The next thing you need to do is to get rid the margins for the test page. That will tell you the number of columns and lines that your printer will print on the page without having to worry what the margins are set to.

There are two ways of doing this. Either set all the margins in the Layout menu to zero or use the >ZM - Zero Margins command at the top of the document.

## Fill the Page With Numbers

Copy your line of numbers from the page width test above about eighty times so that you have more than a pagefull and then print the page.

```
>ZM  
1234567890123456.....678901234567890123456789012
```

You can use the test page to print on different sizes of paper like filofax or other non-standard sizes of paper. Printing numbers across the sheet gives you the coordinates (line and column) to position different parts of your letter. Using the line drawing mode you can even make up a ruler looking line as shown in the screen shot opposite.

## Designing Your Page Layout

After you have printed your test page, you can design your own page layout by changing the page length and margins to suit your own preferences.

## Test Whether Your Printer Can Print Lines

By adding a few boxes on your test page, you can also test whether your printer can print the lines as well. In our example, we have positioned the lines round the space for the address that you want to use with window envelopes.

If these lines print, there is a good chance that the other characters in the symbol menu will print also.





## Advanced Printer Commands

You can add the following commands to your documents. They are all printer related. Whether you can use them or not will depend on the type of printer you are using and whether it supports any of the functions listed here. The other point we would like to make is that you would probably not use these straight away. If this is your first time through the book you may like to miss this bit out!

Most of the functions to do with the printer are already set in the printer driver and are selected in the Printer Configuration menu. However, if you wish to send different codes to your printer, you can include them in the document.

---

### Definition - Printer Driver

The Printer Configuration menu allows you to tell your Notepad what type of printer you will be using. The choices are Simple, Canon BJ10e, Epson 24 pin, Epson 9 pin, IBM 24 pin and LaserJet.

The Notepad contains a printer driver for each of these types of printer. The printer driver holds information such as what codes are needed to feed the next sheet of paper or to turn underline on and off.

Because the Notepad has a range of built-in drivers you don't normally have to be concerned with the nuances of controlling your printer.

If your printer is not one of those catered for by a built-in driver, or you want to use a feature on your printer that is not supported by the driver, you may need to use some of the following commands to get the most from your printout.

---

## Printer Codes

In some printer manuals there are helpful sections on how to use printer codes. If you have a printer manual that just lists the codes, you will have to put up with our helpful suggestions instead.

### Different Numbers That Mean the Same Thing

Before finding some codes to work with, we will explain about the numbering systems that the codes are written in.

You may remember that the Romans had a different numbering system to the one that we

have now. Number 12 was written as xii for example. The numbers look different, but they are the same value.

The codes in the printer manual are usually described in one of three different ways; decimal, hexadecimal or ASCII. Fortunately we don't need to understand them to be able to use them.

Protect can understand all three different types of numbers, but you do need to show which sort you are using when typing in the codes. Let's look at a few codes in a printer manual. It is time to get a strong cup of coffee (beer if you must).

## Set Double Height Example

Here is a command that our printer can use in three different emulations. Let's look at these one by one, that way we can also test that it works as well. This is what the manual has in it.

*Quote from Manual  
Epson FX-850 Mode*

*set double height <ESC>w<n>*

*By means of this code sequence the characters of the subsequent text are specified with double height.*

*<ESC>w followed by 1, sets double height printing. To reset the double height to the normal height, specify <n> by 0.*

*Quote Ends*

That looks very straight forward doesn't it? So how do we use the information?

## Escape

The <ESC> is an ASCII Escape character. To type this into your command line use either the decimal number 27 or hexadecimal equivalent &1B. The lower case w is also an ASCII character, just put "w" in the line. Finally the number one or zero is a number so type that in too. These codes may be separated with a space or a comma. So here are the results.

```
27,"w",1 will turn double height on
27,"w",0 will turn double height off
```

Let's do that again with a different printer.

---

**TIP - What are ASCII Tables?**

An ASCII table is the list of codes that the printer uses to print out the characters. For example, a capital A is decimal 65. The standard ASCII table finishes at decimal 127, but there are a possible 256 codes available. There is an ASCII table in the Appendices.

---

*Quote from manual  
IBM Proprinter XL24e Mode*

*set double height      <ESC>[@<n1><n2><m1>...<m4>*

*By means of this code sequence the characters of the subsequent text are specified with double width, double height or both combined. Additionally line spacing can be controlled by this code sequence.*

*n1 and n2 specify the number of mode bytes contained in the sequence. Usually n1 is defined by 4, n2 by 0.*

*The mode bytes m1 and m2 are without function, i.e. they are assigned code <NULL>(hex.00).*

*The mode bytes m3 and m4 control the print parameters:*

*m3 controls line spacing (high-order half-byte) and character height (low-order half-byte).*

*m4 controls the character width. Only the low-order half-byte is functional in the mode byte.*

*For detailed information on modes bytes m3 and m4, see next paragraph.*

*Definition-Mode*

*Mode-byte m3:*

*The high-order half-byte contains the multiplier 1 or 2 for line spacing, for example*

*multiplier = 1 m3 = bin.0001 0001*

*multiplier = 2 m3 = bin.0010 0001*

The low-order half-byte contains the multiplier 1 or 2 for line spacing, for example

multiplier = 1 m3 = bin.0001 0001  
 multiplier = 2 m3 = bin.0001 0010

If one of the half-height assigned 0 (Zero), i.e..bin.0000, the current values of line spacing or character height remain unchanged.

m3 (dec)	m3 (hex.)	m3 (bin.)	Line feed	Character height
0	00	0000 0000	unchanged	unchanged
1	01	0000 0001	unchanged	standard
2	02	0000 0010	unchanged	double
16	10	0001 0000	single	unchanged
17	11	0001 0001	single	standard
18	12	0001 0010	single	double
32	20	0010 0000	double	unchanged
33	21	0010 0001	double	standard
34	22	0010 0010	double	double

Mode-byte<m4>:

m4 (dec)	m4 (hex.)	m4 (bin.)	Character width
0	00	0000 0000	unchanged
1	01	0000 0001	standard
2	02	0000 0010	double

The high-order half-byte is ignored, the low-order half-byte contains the multiplier 1 or 2 for character width

Quote End

Gosh, remind me not to get an IBM Proprinter XL24e printer! Does this description look bad? Well, yes and no. There certainly is a lot of it. That is because the same command changes the width of the characters as well as their height.

Let's just go for the double height, like the other commands we have used before. To start with there is the <ESC> code again. This is followed by the [ and @ that we put in quotes. Now for the <n> and <m> bits.

The <n> bits are there to tell the printer how many mode bytes we are going to send. In the text it is stated that they are normally 4 and 0 which makes a bit of sense as we are going to use all four mode bytes. So four and zero are the next numbers to go onto the line.

The text describes the mode bytes m1 and m2 as 'without function'. In plain English that means they don't do anything so we will use a zero for each.

To use mode bytes m3 and m4 we need to look at the tables. The option we want to use is double height line feed, which would move the paper up two lines, and double height characters. This is the last entry in the table for mode byte m3 and it tells us to use the number 34 decimal (22 in hexadecimal). Finally, there is the mode byte m4 that alters the character width, which we are going to leave unchanged. The number here is a zero.

So by substituting the numbers we have just worked out into the Escape code quoted in the Proprinter manual we arrive at -

```
<ESC>[@<n1><n2><m1><m2><m3><m4>
```

```
27,[,@,4,0,0,0,34,0
```

Easy, isn't it?

Finally, let's look at a laser printer's code. There isn't a double height command here, but we can change the point size of the font that is selected.

*Quote from manual*

*Point Size (primary font) ESC(s#V*

*Point Size (secondary font) ESC)s#V*

*Function Selects the primary or secondary character font height. Value # equals the height in points, and may include decimal fractions.*

*The following are typical values for font height:*

*Value    Font Height*

*7 point    (Not available using the resident fonts)*

*8 point    (Not available using the resident fonts)*

*8.5 point*

*10 point*

*12 point*

*14.4 point    (Not available using the resident fonts)*

*Description*

*Character height is based on points with larger fonts having a higher point size than smaller ones (12 point Courier is larger than 8.5 point Courier). If you select a point size that is not available, the printer uses a font with the closest point size.*

*For scaleable fonts the value field is from 0.25 to 999.75 points in increments of 0.25 point.*

*One point equals 1/72 inch.*

Quote end

Well here is an easy one. To change the point size to 24 point just use

27,(,s,"2","4",V

**Output Code to Printer (>OC n n)**

Here is the command that you put in front of the code to send to your printer.

The command line to change a laser printer to double height becomes -

>OC 27","(", "S", "2", "4", V

**Multiple Commands**

There is nothing to stop you issuing multiple commands to the printer. Normally, you just keep adding the next command straight after the previous one.

Laser printers follow a different rule. The last character in each command is usually a capital letter. To join two commands together you change the previous command's last letter to lower case and continue the next command without the escape character sequence.

For example, if you wanted to combine the two commands ESC&[10E and ESC&[6D, you would end up with the sequence ESC&[10e6D.

---

**TIP Gotcha Printer Codes**

There are several tricks that printer manufactures get up to when designing codes. Their favourite ones include using the following:

using a lower case L that looks like a number one

using a capital O that look like a number zero  
a number (0-9), which is used as an ASCII code instead of a number, which  
needs to be put into quotes.

---

### Printer Code Not Working

There are more wrong printer codes than there are right ones. You need not be afraid of getting it wrong. There are no self destruct codes. All that happens when the code is wrong is that the printer doesn't do what you expected. It might stop working when it reaches the code, or it might print different characters (including part of the code) onto the paper.

Before throwing your printer out of the window, just take one last look at the printer manual. Perhaps there was a small phrase in the description that you didn't understand the first time through.

If you find some of the code being printed, then the letters or codes before that are probably wrong. If all else fails, there is another trick that you can use.

### Hex Dump

A good way to find what Notepad is really sending to your printer is to put the printer into Hex Dump mode. How to put your printer into this mode is explained in your printer's manual. It should then print out both the ASCII characters and the hexadecimal number.

To test a code put the printer in Hex Dump mode and write a three line document. For example:

The start before the code

>OC - put the code string here

Here is the changed text

At the start of the printed document you will see the initialisation codes that Notepad sends to the printer. Your text should follow that.

The escape character in hex is 1B and you will also see some 0D and 0A numbers that are linefeed and return codes.

By printing out a small amount of text with your code(s) you could find out what is

wrong with your code. If you find that the correct code is being sent to the printer from your Notepad, then it is time to get in touch with your printer supplier for advice.

## Lines and Foreign Characters Don't Print

If your printer does not print lines, there is a good chance that it is not using the correct table. You should compare the printer's ASCII table to the Notepad's ASCII table. They should be identical for the first half, with possible changes in the second half of the table.

Often, the printer has several tables that it can use. Make sure that the correct one is selected and that the Printer character set in the Printer Configuration menu is set to IBM on the Notepad.

## Microspacing and Proportional Printing

Microspacing and Proportional Printing have to be turned on from within your document. By default they are turned off. Furthermore, you will only see any effect from them if you have right justified your document.

With ordinary character spacing the spare space is collected on the left and right of the line. When microspacing is turned on, the amount of space that soft spaces take up is redistributed evenly between every word in that line. This makes the text look better, and can be used on printers that do not support proportional printing.

Proportional printing is even better than microspacing. Instead of allocating the same space for each character, proportional printing changes the amount of space each character uses. It might be obvious, but an 'm' takes up more space than an 'i' when you write it. When you print them normally there is a lot of space on either side of an 'i' but the 'm' looks a bit squashed.

## Define Microspace Code Sequence (>MC n n)

The commands for microspacing and proportional printing should already be defined in your printer driver. If microspacing doesn't work, then add the commands into your document.

The Microspace Code defines the sequence of codes that moves the print head by the smallest amount. Other names that it can be known as include Horizontal Mode Increment and Graphics 120 dpi.

A typical example would be >MC 27,L,1,0,0



### **Define Character Width (> CW)**

This defines the width of a character when microspacing. The default number is twelve. Other examples are ten for elite, seven for condensed and fourteen for condensed enlarged.

### **MicroSpacing On/Off (>MS ON/OFF)**

This turns Microspacing on and off. The default is off.

### **Proportional Printing On/Off (>PP ON/OFF)**

This is selected by turning it ON with this command and using the Style attribute 'p' command at the start and end of text you wish to be printed.

If your printer supports proportional printing the Notepad has to alter the number of characters that it can print on each line while printing. If you had a line of 'i's you would get more of those on a line than would if you had a line of 'm's.

This command has been designed for use with proportional daisywheel or laser printers. Dot matrix printers can work very slowly in this mode. Fixed pitch daisywheel and non-proportional dot matrix printers do not use this command.

# Chapter 13

## Layout

### Configure Menu (Function 3)

One of the commands listed on the template is Configure. Select this by holding the function key down first and then the number 3. There are two menus that can be viewed here. The first one is shown straight away and is called Editing Options menu. To select the View Options menu instead move the cursor up one line. To change any of the default settings move the cursor to the required line and change 'No' to 'Yes' with the left or right cursor keys. Press Stop to return to your document.

```

XXXXXXXXXXXXXXXXXXXX Overtyping or ← → to change, ↑ ↓ to move, Stop to Finish
Insert on/off (On)
Word wrap (On)
Right justify (On)
Decimal character (.)
Key repeat startup delay (100th secs) (50)
Key repeat period (100th secs) (5)
Cursor flash period (100th secs) (50)

```

Insert on/off, Word wrap and Right we have already covered earlier.

### Decimal Character

The decimal character (.) is the character that is used for the decimal point. In the UK this is usually a full stop, but in other countries a comma is often used. If you use a different character, change the full stop to the character you use.

### Key Repeat Startup Delay (100th secs)

When you press and hold a key down the Word Processor waits for a certain length of time before it starts repeating the character again.

If you do not want a repeating character you can set this to a maximum of 1000 (10 seconds) before the character is repeated.

By changing the time delay you can customise Notepad to your typing speed.

## Key Repeat Period (100th secs)

Once the word processor has started repeating a key the repeat rate is controlled by this setting. The lower the number - the faster a key will repeat.

## Cursor Flash Period (100th secs)

This number just changes the speed the cursor flashes.

```
Press ← → to change, ↑ ↓ to move, Stop to Finish
Show printer codes? (No)
Show spaces? (No)
Show tabs and returns? (No)
Show status information? (No)
Show ruler (No)
```

## View Options Menu

All these options are simple YES/NO settings which determine whether the particular information is shown in your document.

Often in Protex, you will find there are subtle differences in using what may be at first sight the same command. This is particularly true when dealing with these menus. Once you change any of the settings here, all your future new documents will use this set up.

## Page Layout

When a document is typed into your Notepad, what you see typed on the screen is what will be printed in the active area of the page. The margins around the sheet of paper are not normally printed on, or shown on the screen.

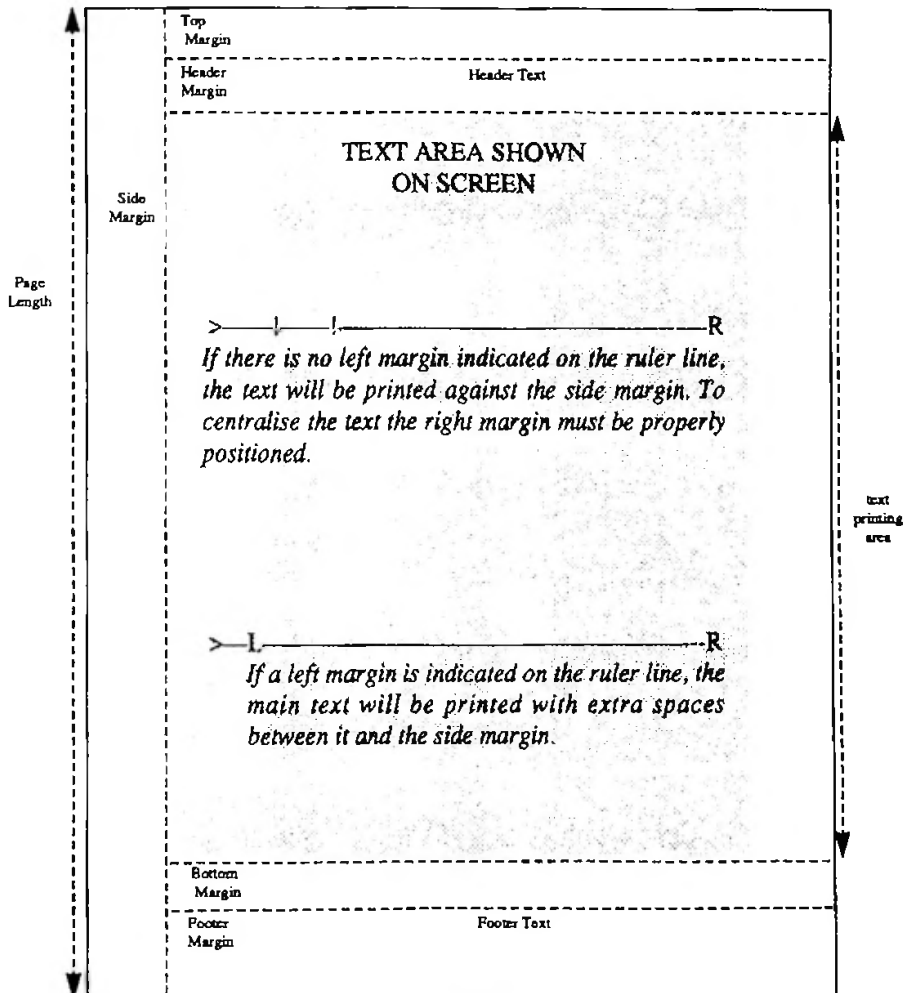
This means that although you type from the left-hand edge of the screen, the printed text will be placed five characters to the right. This is due to the side margin that is set to five unless you have changed it.

There are margins all around the sheet of paper that you can alter from the Page Layout menu.

## Page Layout Menu

Page length (lines)	(66)	66
Top margin (lines)	(3)	
Header margin (lines)	(2)	
Footer margin (lines)	(2)	
Bottom margin (lines)	(2)	
Side margin (chars)	(0)	
Line spacing (lines)	(1)	

To select this menu press the Function and the 7 keys.



## Why Change These Settings?

The default settings in this menu have been designed for use with A4 paper. Should you want to use a different size paper, or just change the look of your printout, you will need to change these settings.

A good way to plan a new layout is to use the 'Print Test Page' to see where all your margins should be placed. This can be easier than trying to calculate how much space each line takes up.

## Change the Menu or Use the Commands

When you change the settings in the Page Layout menu, they will be used on your current document and any new ones you create in the future. So you should have these set to your standard layout preferences.

If you are printing just one document that needs a different layout, it is a good idea to use the page layout commands instead. These commands will be saved with the document and will be executed when printing takes place. They will leave your standard layout settings alone.

## Template Files

You may decide to take this even further by creating separate template files, which contain the various page layout and margin settings in the document. These files can either be loaded into any document before you start typing or used with the Insert command.

## Examples of Template files

### Letter template

```
>CO letter template
>SM 10      ; - Side margin 10
>PL 72      ; - Page length 72 (for 12 inch paper)
>CP OFF     ; - Continuous printing off (single sheet)
>-----R
```

### Filofax Template

If you are using continuous paper with holes down the edge, you can use this layout. It will also enable you to use the holes down the side to fix the page into the book.

There are two layouts. One uses single line spacing and the other uses half line spacing, to try to get more information on the sheet. When using this version, change your Style Attributes in the document to condensed and superscript (if possible).

```
>CO filofax template - ordinary version
>LS 1      ; - single line spacing
>SM 0      ; - Side margin 0
>PL 19     ; - Page length 19 (for filofax paper)
>CP ON     ; - Continuous printing on (continuous sheet)
>FP ON     ; - Format text whilst printing
>-----R
```

```
>CO filofax template - tight version
>LS 0.5    ; - half a line spacing
>SM 0      ; - Side margin 0
>PL 38     ; - Page length 19 * 2 (for filofax paper)
>CP OFF    ; - Continuous printing on (continuous sheet)
>FP ON     ; - Format text whilst printing
>-----R
>CO set condensed (Control X c) and superscript (Control X t)
```

## Page Layout Commands

The following commands are used with a number after them that is indicated by the 'n'. They do the same thing whether they are changed in the Layout menu or as a command.

### Top Margin (>TM n)

Starting at the top of the page the top margin allows you to define how many blank lines you want at the top. Some printers are not able to print the first few lines, especially if they use a sheet feeder.

The other good use for this command is if you are using preprinted paper, say with your name and address already shown. You must experiment with the number of blank lines you need to set together with the next command if you use headers in your document.

The default value is three.

### Header Margin (>HM n)

The header margin comes after the top margin. If you have defined a header it is two lines long, otherwise it is ignored.

## **Side Margin (>SM n)**

The side margin refers to the number of blank spaces that the Word Processor prints on the left-hand edge of the sheet. It is called the side margin to stop you thinking of it as the left margin. The left and right margins are defined separately by the ruler line.

The default value is five.

## **Odd side Margin (>OM n)**

## **Even side Margin (>EM n)**

These commands are used instead of the side margin command if you want a different side margin on odd and even pages. This is usually used when printing pages for a book to keep the margins further apart in the centre of the spine. You must be content with the same right-hand margin in the ruler line though.

The default value for both these margins is five.

## **Footer Margin (>FM n)**

This is like the header margin only for the footer. If the footer is not used then neither is this margin.

The default value is two.

## **Bottom Margin (>BM n)**

Last but not least is the bottom margin. Again, some printers lose their grip on the paper at the bottom.

The default value is three.

## **Page Length (>PL n)**

The page length is the total number of lines that could be printed on the paper you are using. This should include the top and header margins, the text that you type and the footer and bottom margins.

## **Zero Margins (>ZM)**

A quick command to get rid of all the margins in one go. Good for creating draft copies

of documents and the 'Printer Test Page'. By removing all the margins to start with, you can then decide where to place them.

## Miscellaneous Commands

This section contains several easy-to-use commands that display messages on the screen while you are printing. We have also included some other commands here that don't belong to any other section.

### Comment Line (>CO text or >>> text)

Comment lines are used to remind you what you were doing within your document. They don't get printed, but remain in the text when you save your document.

You could use ">CO I could never get this code to work, leave it alone" in front of another command. You could use '>>>' in front of another command as a temporary patch, perhaps until you could get the code to work.

You can also add a comment after using a command by using a ';' mark between the command and your comment. You could use this to remind you why you were using that command.

Whether designing a mail merge document, or simply leaving a note, comment lines are worth their weight in gold, should you remember to use them in the first place.

### Messages

Rather than throwing paper aeroplanes round the room while printing a document, you can send messages to the Notepad's screen to tell you how it is getting on. We will be covering other aspects of displaying messages to the screen when we discuss Mail Merging.

### Clear Screen and Display Message (>CS text)

This is a good command to use at the start of any messages, as it clears the screen so that messages start at the top.

### Display Message (>DM text)

This is the message command. You could use it to display a message like

'>DM half way through', or '>DM not much longer ....'



### **Wait and Display Text (>WT text)**

Using a Wait command with an optional message will give you a choice of actions. If you press any key except Stop, the printing will continue. The Stop key will return you to your document where the Wait command was placed.

### **Stop Printing and Display Text (>ST text)**

This command immediately stops the printing and returns you to the Stop statement in the document.

### **Insert Document (>IN document name)**

The Insert document command is very important for those users who wish to print documents greater than the 38K restriction that the Notepad gives. But you don't need to write a large document to benefit from using this command. We talked about creating a template document to hold your page formatting instructions. The Insert Document command allows you to include a standard set of layout instructions whenever you need them. You don't have to type them in again every time. There is nothing to stop you even printing a one page letter from several smaller files.

Here is a small example to show how to use Insert on your Notepad. The following lines are typed into a document and become a print list. The documents that are listed will be printed in sequence. You just tell the Notepad to print the list of documents. Page numbers in headers or footers will be counted as if the print document were one large document.

```
>CO Print out document for Main Newsletter  
>IN header.doc ; this is my address header  
>IN 1stqtr.ltr ; the news in January to March  
>IN 2ndqtr.ltr ; the news in April to June  
>IN 3rdqtr.ltr ; the news in July to September  
>IN 4thqtr.ltr ; the news in October to December  
>IN all4now.ltr ; All for now - close  
>IN lovefrom.ltr ; Love from ending
```

If you didn't want to print any of the quarters, you could just leave them out of the main list.

# Chapter 14

## Mail Merge

Mail Merge gives you the ability to combine elements from several different files and/or the keyboard and send them to several different places. The finished product from the Mail Merge process can be sent to either the screen, another file or the printer. Why would anybody want to use such a program?

The most common example is writing a mail shot letter to several different addresses, where the addresses are stored in one file while the main letter is stored in another file. Many companies use this method for customer enquiries. The names and addresses are typed into a data file during the day, then shortly before the postman is due to pick up the mail, the letters are printed with each person's name and address on it. That should give the impression to the customers that an individual letter has been written to them.

Mail Merge can be used for other applications as well. It can generate address labels, issue invoices and calculate totals or even generate a data file.

---

### Definition - Data File

**A file like the Address Book file (but you can't use the Address Book directly) that stores information in a predetermined format.**

---

As the Notepad has BBC BASIC, there are applications where you could use that to do the same thing as you could do with Mail Merge. Many key ideas are the same for both Mail Merge and BASIC, things like variables, operators and conditionals like If Then Else. We have covered these here as well as in BASIC so that you don't have to keep jumping into the BASIC section while you are trying to read about Mail Merge.

## Precautions When Using Mail Merge

Here is yet another reminder about making backups. With Mail Merge you are writing programs. These programs use commands that, if used incorrectly, will crash your Notepad. The result is that you will lose all your files (just like we did). The same may also be true if your Notepad runs 'Out of Memory'. Once your Notepad has no more space in memory to work in, it will not respond to any key presses. If this happens, all you can do is to turn it off and then on again. If you are lucky you will only have lost the document you were working on last. It is possible, however, that all your documents will have vanished from the list. Please reduce the risk of losing your documents and files by keeping copies. See the Introduction for details of how to do a Backup.

## Designing a control document

There are several steps in writing a mail merge file, which we are now going to call the control document.

### Using Comment Lines and Insert in Mail Merge

It is a good idea with Mail Merge to write some brief notes to describe what you want to do before diving into the commands. If you make the notes into a list, you could use them as comment lines in your control document. Comment lines are notes and reminders that you write into a Control Document to remind yourself what the program should be doing. They are useful when you return to the Control Document later to modify it.

There are groups of commands that are repeated in each Control Document that form naturally into modules. One module will contain the instructions to call up a data file. While the instructions that tell your Control Document what the data file holds in what order can be written as a totally separate module. Each module will be stored in a separate file. You can then use the Insert command to join all the modules together. This keeps mail merge files smaller and easier to use.

As you design more of these modules you can build quite complicated Control Documents without having to start from scratch every time.

### Testing

Most Control Documents start life out-of-control. To bring them back under control you need to test each module. There are a few commands that can help you.

To test your mail merge file, there is a useful facility called Print to Screen. This saves printing on lots of paper before you have finally got your document correct. You can put Wait statements (with or without a message) into your Control Document so that while you are printing to the screen you have time to read each screenfull before it scrolls off the top at the speed of sound.

The Display Messages command can also help when you are testing your Control Document because it can tell you where the program is instead of where you think it might be.

### Check If It is Worth Doing Mail Merge

When we first started writing control documents, it quickly becomes apparent how long it

took to make everything work. At the beginning if we had three letters to write we would use Mail Merge. Now if we only have a few letters to write we don't bother because it is quicker to copy the file and change the names and addresses manually.

If you are sending a particular letter often, Mail Merge comes into its own.

## Keep Your Control Documents Friendly

When designing a Control Document, the Display Message command will show you on the screen what is being printed. This makes the difference between seeing what the document is doing and wondering if anything is happening. We will use this command often in the examples - so here it is.

### Display Message (>DM text)

Just type in the text you want to display at this point.

There are other commands already covered that can indirectly display messages on the screen. These are 'Clear Screen', 'Wait' and 'Stop'. Some examples include:

```
>CS This message will be displayed at the top of the screen
>WT Change the paper, press any key to continue or 'Stop' to Stop
>ST This is the end of the mail merge control document
```

## Where Should You Start?

Let's start with a simple example - sending three letters using mail merge. Yes, you are right, normally you wouldn't use this to send just three letters, but you can always add more addresses later. The first thing we need is a data file. This should contain the name and address of the people we want to send the letters to.

## Designing Your Data File

Before making up a data file, you need to plan its structure before you start typing in anything. You need to decide how many fields there are in a record. We have decided to use the same number of fields that the address book uses.

### Definition - Fields and Records

Let's define a couple of words that are database terminology.

A record is one entry made into the database. In this case the name, address, the

telephone and fax numbers.

Each segment of a record is called a field. The fields are the name, four address lines, the telephone number and finally the fax number. So we have seven fields in one record, and as we are going to create three different addresses, there will be three records.

### **Input for Mail merge**

There are three ways of getting the necessary data into your Control Document. First there is the command file itself, followed by the data file. You can also input data from the keyboard while the document is being printed.

#### **Data File**

The data file is a separate document that contains the data that you wish to use with the Control Document. The data that we are using here is names and addresses, but it could also be other information like invoice numbers and amounts.

Your data file becomes your database; that is the base where all your data is held.

#### **Making up a Data File**

There are three ways of making your data file.

The simplest is just to type the addresses into the file, which is what we are going to do now. Of course this would not normally justify the use of Mail Merge.

Secondly you could use the same names and addresses that are in the Address book.

and

Thirdly we could make up a control document that allows you to type in the different fields and add them to your data file. We will cover this after we have described all the mail merge commands.

#### **Typing The Address Directly**

To type the names addresses in directly, start a new document and call it `typeaddr.dta`. Then just type in some names and addresses in seven lines.

Jon Day  
 Kuma Computers Limited  
 Unit 12 Horseshoe Park  
 Horseshoe Road  
 PANGBOURNE Berkshire RG8 7JW  
 0734 844335  
 0734 844339

Witch of The North  
 Deadly Nightshade Cottage  
 Web Lane  
 Greater Cockoospit  
 Scarpa Flow Shetland SD234 76qd  
 08642 45398786  
 08642 45390765

John Smith  
 1000 High Street  
 Braintree  
 Essex CM7 8QN

0222 215555 Car Phone 0850 555123  
 0222 215556

You could replace the above names and addresses with some of your own instead. If you have a short address, remember to leave a blank line so that the field numbers still line up. Let's look at the last address again to illustrate what we mean.

John Smith	Field 1
1000 High Street	Field 2
Braintree	Field 3
Essex CM7 8QN	Field 4
	Field 5
0222 215555 Car Phone 0850 555123	Field 6
0222 215556	Field 7

You will notice that field 5 is blank in this case. If you hadn't left it blank but put the phone number there instead, you would find that the phone number would be printed as part of the address. Not a feature that you would be proud of.

### Using Addresses from the Notepad's

Unfortunately, you are not able to use the ADDRESS BOOK file with your names and addresses directly with Mail Merge. If you want to use them with Mail Merge, you need to transfer each address separately to the data file. We covered transferring addresses into documents earlier.

## Building Up The Control Document

We talked earlier about saving different modules of the complete Control Document into small files. You can then join them together using the Insert command.

Here is the list of modules;

- Define the data file
- Print our letterhead
- Print the address of each person
- Print the rest of the letter

All we have to do now is to work through the list.

## Defining the Data File

Now that we have a small data file, it is time to start writing our Command Document. This is the file that should be in charge of the whole operation. Let's start by giving the information about our data file.

Open another new document and call it `addrdf.def`, which is short for address data file definition.

### Define Data File (>DF filename)

This tells the control document the name of the data file. Remember what we called it? That's it `typeaddr.dta`, so the first line of the control document should read:

```
>DF typeaddr.dta
```

Note When you are in advanced mode is that you can use more than one data file in the control document, but you need to close the data file first before defining another one.

## Variables

When describing something to an unintelligent computer, you need to tell it **everything**. To start with, it is necessary to define what items you want to work with. In a **cooking** book, this may be the ingredients like four tomatoes and a loaf of bread. To repair a **car**, for four spark plugs and so on. In a computer the data is stored as variables.

The three variables we mentioned above, need to be defined first before the value they have can be printed.

Tomatoes = 4  
 loaf = 1  
 Spark Plug = 4

The items or variables that need defining for names and addresses are the field names that we have just covered.

Once a variable has been created, the value that it contains can be printed by using the variable name between two '&' characters.

## What's in a name?

When defining anything, we need to think of a name for what we are defining. It usually helps if the name that we come up with has some meaning of what it is describing so that you will remember what it means or represents.

Since you can call a variable anything, choosing a name is quite easy. Look at one of our records in our data file for example.

John Smith	Field 1
1000 High Street	Field 2
Braintree	Field 3
Essex CM7 8QN	Field 4
	Field 5
0222 215555 Car Phone 0850 555123	Field 6
0222 215556	Field 7
blank line	

We have seven variables and a blank line to define here. We could define them as:

Field1, Field2, Field3, Field4, Field5, Field6, Field7 and dummy for the blank line.

Is this a good choice of names? They could be improved, by changing the names to indicate what each field represents. Say, Name, Addr1, Addr2, Addr3, Addr4, Tel\_No, Fax\_no and dummy.

## Variable Name Rules

You cannot have a space in a variable name, nor start one with a number. As with document names you can use the underscore character to join one or more words together in a name so it looks like one word. Although you can make a variable name up to 255 characters long, it is a good idea to keep them short. This does two things. First you do not have to type many characters each time you call them and secondly shorter names use



less of the Notepad's memory.

### Variables not found - message

If you use a variable that you haven't created yet, you will get the Unknown variable message when you print the file.

---

#### TIP - Do Not Leave Variable Values Between &s in the Comment Lines

If you want a variable's value printed you have to put the variable name between &s. Putting &addr1& in your Control Document would tell Mail Merge to find the variable and print it's value.

Although Mail Merge is supposed to ignore everything in a comment line, it will recognise and try to print the variable's value. If it is a variable that you have already created there isn't a problem. Mail Merge will try to print the variable value but then it will realise that it is a comment line and ignore it. If, however, that variable has not been created beforehand, it will stop your Control Document in its tracks and give you an 'Unknown variable' message.

---

### Reading Variables in the Control Document

Back to our data file. There are two different commands to read data from a file and assign it to variables. Both commands clear the values that the variables are holding to make way for new values. But they stop reading new values for different reasons.

#### Read Variables - (>RV name)

>RV just reads data and assigns it to a list of named variables until it reads a blank line. Any unread variables in the list are set to be blank. This is the simplest approach but it has a serious drawback when dealing with addresses. Address data frequently includes a blank line within a valid address. For example:

```
John Major  
10 Downing Street  
London
```

```
W1 1AB
```

If we use a command line something like:

```
>RV name, addr1, addr2, addr3, postcode
```

the first three variables would have the correct values assigned to them. But the variables `addr3` and `postcode` would be left blank because of the blank line in the address data.

## Read Variables Unconditionally - (>RU name)

>RU assigns data from a file to variables in the same way as >RV. The difference is in the way that blank lines are treated. >RU always reads as many items from the file as there are variable names in the list. It is quite happy to read a blank line.

You would use >RV if your data file has varying numbers of fields for each record with a blank line at the end of the record. You would use >RU if your data file had the same number of fields in each record and/or included blank lines in the data.

In our example we have the same number of lines (one for each field) for each record. This means that we should use the >RU command.

## Read Variables and pad with nulls - (>RV name)

If you need to use >RV because of differing record lengths but need to have the flexibility of including blank lines in the data, you can start each blank line with a dollar character. The '\$' character tells Mail Merge that this is a blank field within a record and not the start of a new record.

```
John Smith
1000 High Street
Braintree
Essex CM7 8QN
$
0222 215555 Car Phone 0850 555123
```

## Another Way of Using The >RV Command

There is another way of using the >RV command without putting the '\$' into blank lines in the records. If you split the command line into several shorter RV lines and end each line with the variable that might be blank, Protect will read the record correctly.

For example, if you thought that `Addr3`, `Addr4`, `Tel_No` and `Fax_no` would be blank, you could re-write the RV line as follows:

```
>RV Name Addr1 Addr2 Addr3
>RV Addr4
>RV Tel_No
>RV Fax_no
>RV dummy
```

## Close Data File - (>CF)

This command closes the data file when you need to use several different data files in the same command document. You would use this for example if you were pulling information out of more than one data file. The data files don't need to be in the same format provided you change your >RU or >RV to reflect the change of variables in the new data file.

Our first Control Document should look like this:

Name of Document addrdf.def

```
>CO Address Definition file
>DM Reading Address Data File           ; this displays a message
>DF typeaddr.dta                        ; define the data file
>CO ** Read the variables **
>RU Name Addr1 Addr2 Addr3 Addr4 Tel_No Fax_no dummy
```

You can add as many comments as you like to remind yourself what these commands are doing.

## Adding Your Letterhead

Adding your letterhead to the Control Document so that it will be printed is easy. You just need to add 'letterhd.doc' to the end of the list of files to be printed in the Control Document. Just to remind you, this document should have been created previously when you wrote 'the first letter'.

## Print the Address to Each Person

Now for the interesting bit - printing the variables that we have read from the data file. When you want a variable to be printed, you should put an '&' character at the start and the end of that variable's name.

Time to get another new document and call it praddr.mmg - short for Print the address of each person. If you type in:

```
&Name&
&Addr1&
&Addr2&
&Addr3&
&Addr4&
```

it will print the name and address.

## Print the Rest of the Letter

You can repeat these variables as many times as you like within the letter, so you can now write a letter that says:

Dear Mr Smith,

Yes, Mr Smith, you are one of the chosen few at 45 Orchard Drive that have won the Golf GTI should you attend a meeting near the north pole on Tuesday.

You would type the letter into the control document, replacing the words with the variable names. That would make it look like this.

Get another new document, call it restltr short for Print the rest of the letter and type in the following.

Dear &Name&,

Yes, &Name&, you are one of the chosen few at &Addr1& that have won the Golf GTI should you attend a meeting near the north pole on Tuesday.

## Testing & Putting It All Together

That's it. All that remains now is to make a file to call up these files that we have just created. You are not expecting it to work first time are you? It is always a good idea to test your Control Documents by printing it to the screen.

As the Notepad's screen can only display eight lines, printing the complete page to the screen is difficult to read. Unless you have a set of eyes that can roll vertically we suggest making an additional screen print layout file for testing purposes.

Here is the file, complete with comments, using commands already covered.

Name of document ScrPrintTEST

```
>WT Press any key to continue . . .;Wait to display previous
messages
>CS *****
>DM * This file changes the page length of any document so *
>DM * that it can be displayed on the screen in viewable chunks *
>DM * ScrPrintTEST document selected. *
>DM * REMOVE DOCUMENT FROM LIST WHEN FINISHED TESTING *
>DM *****
```

## Chapter 14 Mail Merge

```
>CO There is a full-stop in the next line
>DM
>WT Press a key to TEST or 'Stop' twice to STOP ;Wait to display
previous messages
>CS Testing mail merge document ;Clear the screen
>CO Clear both the top and bottom margins
>TM 0 ; Top Margin
>FM 0 ; Bottom Margin
>PL 7 ; Set the page length to seven lines
>CP off ; Continuous printing off, pauses between pages
```

### First Mail Merge file

Remember our list earlier of what we wanted to do? Well here is the final list of files that you can use to print your first Control Document. The ScrPrintTEST line at the beginning of the document, is used to test the Control Document first. This should be removed when testing is complete.

Name of new document Mailm1

```
>IN ScrPrintTEST ; FOR TESTING ONLY
>IN addrdf.def ;Define the data file
>IN letterhd.doc ;Print our letterhead
>IN praddr.mmj ;Print the address to each person
>IN restltr ;Print the rest of the letter
```

### Testing, Testing, Testing ...

To test this mail merge document, use Print to Screen - Menu P first. You should see each letter being printed on the screen in small chunks, pausing every eight lines. Pressing the space bar moves you to the next eight lines until all the addresses in the data file have been printed. If any part of the letter goes wrong, try to identify which of the files is causing the problem.

If different parts of the address are being printed incorrectly, check that the data file fields are in the right order.

When you are happy that everything is all right, remove the test line, connect the printer and print out those letters.

# Chapter 15

## Address Labels

When you need to print lots of letters, it is often cheaper to get the letters photocopied and print self adhesive labels to stick onto the envelopes. At first sight this seems easy as printing address labels is just like printing multiple letters (as above) without the letter.

There are, however, a few differences that need to be discussed before looking at some more commands.

### Practical Printer Problems

Not all printers can print labels. When using dot matrix printers you should set the paper thickness near to the maximum setting. This is usually set by a lever that moves the print head away from the roller. If the setting is too close, print smudging or even label jamming can occur.

Printer labels come on a variety of sizes and types of sheets. It is important with these labels that your printer can print consistently at the same place on each sheet every time. This is not always possible with printers that rely on the manual insertion of sheets .

If you have a tractor feed on your printer, then you should get the labels with the perforations down the edge.

If your printer has a sheet feeder, then use single sheets of labels that are designed for laser printers.

Depending on the size of the label, you can get sheets that have one, two or even three labels across the width of the sheet. To start with we will just consider how to print one label at a time on a sheet with two rows of labels.

### Positioning the Print on the Label

If you print a 'Test page' its will give you the information you need to correctly position the addresses.

### Page Layout for Labels

Unlike a letter, where we can use the default settings for page layouts, labels are different. They are smaller and there are several of them on the sheet.



## Altering the Page Length to the Length of a Label

When printing the Mail Merge letter, we got one address on every page. You don't want that to happen when printing labels because you would only print one and miss out the other labels on the sheet. To make the Word Processor print each label, change the page length to the pitch of the label. The pitch of the label is the sum of the label height plus the space between each label.

Depending on the labels and printer that you are using, you may have to change the page length setting. Some printers are happy with the Word Processor just sending blank lines to move to the next page (or label in this case). Other printers require the correct page length to be set and a form-feed character to be sent to tell it when to move to the next page. In that case you need to let the Word Processor and the printer know the page length.

On the Word Processor it's easy, just use the >PL Page Length command.

To find out which way your printer prefers to change the page length try the Line Feed option first since it is easier. If that fails, you'll need to change the page length on the printer.

In the following examples the pitch of the labels is nine lines deep.

### Line Feed Option

Name of document labbl1.set

```
>ZM      ; (zero margin) - no margins required
>RJ OFF; right justify OFF
>FP ON ; page formatting ON
>PL 9   ; set the page length to 9 lines
>FF OFF; Don't send the form feed character after each label
>SM 1   ; Side margin
```

### Form Feed Option

If the Line Feed option doesn't work properly you may need to use this option instead. You need to send the printer the codes to change the page length. You will need to consult your printer manual to find the correct code. After the code has been sent (using the >OC command), you can check that you have set it correctly by



Putting the printer off-line,

and

Pressing the form feed button.

The printer should just move the paper up to the start of the next label. If it doesn't, then you have sent the wrong code. See details on using the >OC command and sending codes to printers for more information.

Here is our file to set up the page length using Form Feed.

Name of document lab1.set

```
>ZM ; (zero margin) - no margins required
>RJ OFF: right justify OFF
>FP ON ; page formatting ON
>PL 9 ; set the page length to 9 lines
>>> CHANGE PAGE LENGTH TO 9 LINES ON PRINTER
>OC #XQ #1B #4F #1B #43 #09 ; these are Epson printer codes
>SM 1 ; Side margin
```

## Chopping Long Address Lines

When printing an address in a letter the length of each address line does not matter. When printing an address on a label, we need to cut off any long lines that may not fit. You do this by splitting variables. Before you can split your variables, however, you need to know about some more about variables.

### More About Variables

So far we have only covered using variables in a data file, where we can put an address line (Addr1) into a variable and then print it. It is also possible to create variables in the control document and manipulate them. You can add other characters and combine the value of two variables into one. Equally you can split a variable's values. You can even separate one of your address lines into variables that hold individual words or characters.

There are two more commands (in addition to the RU and RV commands that you use with a data file) that allow you to work with variables and their values. One is AV that enables you to create a variable and/or give it a value with the keyboard while the Control Document is actually working. You can also create variables as constants.

When inputting values for variables from the keyboard it is essential that messages appear

on the screen. Then you know what the control document is waiting for when it stops in the middle of printing. These messages must be as clear as possible to whoever is going to use the Control Document.

### Ask for Variable - (>AV message var)

This is the command that you should use to tell the Control Document to get the value of a variable from the keyboard. You can type in an optional message and the name that you want this variable known by. The message is then displayed on the screen as a prompt when the Control Document reaches the AV command. What is typed while the Control Document is being printed is stored as the value for that variable.

Use the following example to show that this works by printing it to the screen. There is nothing to stop you adding this line to the previous Mail Merge letter. You do this by placing the >AV line at the top of the letter and the &date& where you want the date to be printed in the letter.

```
>AV "Today's Date? (Sym'D') " date
&date&
>ST
```

If you leave out the message part of the command, the prompt will become the name of the variable followed by a question mark. In the example above you will get the prompt, date?

### Restricting the Number of Characters in a Variable

If you place a number between 0 and 255 after the variable name you restrict the number of characters that can be typed by that number. Here is another example:

```
>AV "Try typing more than 5 characters here!" test 5
>DM You typed &test& didn't you
>ST
```

Notice that you can also display variables on the screen by using the >DM display message command with the variable.

### Setting Multiple Variables on the Same Line.

You can create multiple variables by separating them with a space, comma or a full stop and adding them to the same line, just like the Read Variable line. The following commands:

## Chapter 15 Address Labels

```
>DM Type in the Name, Address (4 lines), Telephone and Fax Number  
>AV Name,Addr1,Addr2,Addr3,Addr4,Tel_No,Fax_no
```

```
Type in the Name, Address (4 lines), Telephone and Fax Number  
Name?  
Addr1?  
Addr2?  
Addr3?  
Addr4?  
Tel_No?  
Fax_no?
```

If you have a blank line in an address press the Enter key after the prompt for that variable's value. That will leave the particular variable empty.

### Set Variables to a value - (>SV name value)

This command creates a variable that is a constant. A constant variable can contain

```
    a string of text  
    another variable's value  
or  
    a number or a mathematical expression
```

An expression can involve other variable values that we will cover later in detail.

How do you use this? Suppose you want to make a variable called 'name' to have 'Betty Norma StJohnStephens' in it, you would type the command:

```
>SV name="Betty Norma StJohnStephens"
```

This has the same effect as using variables in the data file. After this variable has been defined, you can use &name& to print out Betty Norma StJohnStephens wherever you want in the document.

Why should you use this method rather than just typing out the value in the first place? Well you might have a variable that is repeated several times like;

```
>SV year_end="30th September 1992"  
>SV policy_no="Q45/8965/2491"  
>SV count=1
```

You can change the values inside these variables very easily. By putting ALL your variables at the beginning of the text or altogether they will be easy to find. You don't

have to search through the document for each occurrence of the word or phrase.

## Adding and Splitting Variables

Now you know how to define the values of variables, you can use the same command (>SV) to add and split them up.

### Adding to Variables

Adding variables does different things depending on what the variables are holding.

### Adding Text Variables

If you add two variables containing text together, the second value will be joined onto the end of the first. Try out the following examples.

```
>>> Defining some variables to play with
>SV first_name="Vic"
>SV middle_name="Ian"
>SV surname="Gerhardi"
>SV space=" "
>SV full_name=first_name+space+middle_name+space+surname
>DM The final result is &full_name&
```

The answer you should get when printing these commands to the screen is

The final result is Vic Ian Gerhardi.

Why not replace the variables with your own? You may notice that one of the variables called space has been defined and called up twice to give the space between the three words. Without the space, all the variables would be joined together into one long word.

### Adding Number Variables

If your variables are numbers, they are added.

```
>>> Use the Print to Screen facility when printing this document
>>> Defining some variables to play with
>SV first_num=12
>SV second_num=34
>SV third_num=56
>SV add_num=first_num+second_num+third_num
>DM The final result is &add_num&
```

## Chapter 15 Address Labels

The answer you should get when printing these commands to the screen this time is:

The final result is 102.00

You will notice that when working with numbers, they get displayed to two decimal places. You can stop these extra digits from being shown by using the following command before the final result line.

```
>SV add_num=add_num[w1]
```

which turns add\_num into one word i.e. 102. More details coming up.

If one of the variables that you add is text, Protect will regard the numbers as text and join them one after the other.

```
>>> Use the Print to Screen facility when printing this document
>>> Defining some variables to play with
>SV first_num=12
>SV second_num=34
>SV third_num=56
>SV string="Hello "
>SV add_num=string+first_num+second_num+third_num
>DM The final result is &add_num&
```

The answer you should get when printing these commands to the screen this time is:

The final result is Hello 123456

### Splitting Variables

Remember, we wanted to chop off long address lines? Well, we're here at last.

This is done by specifying which word or character in the variable you want to split or even chop in the following way.

var[a:b]	from character 'a' to character 'b' inclusive
var[a:]	from character 'a' to the end
var[a]	character 'a' only
var[W1]	first word only - useful with numbers
var[Wa:b]	from word 'a' to word 'b' inclusive
var[Wa:]	from word 'a' to the end
var[Wa]	word 'a' only
var[w-1]	the last word only

Words are defined as any group of letters or numbers separated by a space, a full-stop, or a comma. This means that you can use [w1] to stop numbers from being printed with two decimal places as shown in the 102 example above.

Let's look at a few examples.

Using a variable called 'name' which contains 'Betty Norma StJohnStephens', you would get the following results.

```
name[4:9]      is'ty Nor'
name[11:]     is'a StJohnStephens'
name[3]       is't'
name[W1:2]    is'Betty Norma'
name[W2:]     is'Norma StJohnStephens'
name[W-1]     is'StJohnStephens'
```

We can put the result of the split into another variable:

```
>SV newname=name[W1]+" "+name[W-1]      gives'Betty StJohnStephens'
>SV newname=name[W2:]                   gives'Norma StJohnStephens'
>SV newname=name[1:4]+" "+name[W-1]     gives'Norm StJohnStephens'
>SV forenames=name[W1:2]                 gives'Betty Norma'
>SV newname="Miss"+name[W-1]            gives'Miss StJohnStephens'
```

## Chopping Long Address Lines - continued

The following file puts into practise what we have just covered it will cut the address lines to just thirty characters.

Note, you may need to change the value of the Labelw variable to a different number that corresponds to the width of the labels that you are using. Get the width from your Label Test Page.

Name of document choline

```
>DM Chop off long address lines
>>> Put the label width in the following variable
>SV Labelw=30 ; Label Width
>>> Chop all the variables in the file to be the width of the label
>SV Name=Name[1:&Labelw&}
>SV Addr1=Addr1[1:&Labelw&}
>SV Addr2=Addr2[1:&Labelw&}
>SV Addr3=Addr3[1:&Labelw&]}
```

```
>SV Addr4=Addr4[1:&Labelw&]
```

### Printing the Label

Well we are nearly there. All we have to do now is to write an address label. Not too difficult as it is just like writing the address on the letter. The >PA command is used at the end of the address to move the printer to the next page, which is of course the next label.

You will notice that a ruler line has been added. This is used to line up the left-hand margin for the label. Again, you may need to change where the left-hand margin or the Side Margin command in labl1.set or labl.set is for your labels. The TAB marks will be used in the next example - printing two labels side by side.

Name of document addlabel.

```
>>>PRINTING STARTS HERE*****
>DM Going to print Label to &Name& at &Addr1&
>>>
>-----!-----!-----R
&Name&
&Addr1&
&Addr2&
&Addr3&
&Addr4&
>PA
```

### Printing the Single Address Label - Finale

Just like the letter, we have created the modules as we went along. Now we need to make a list file to call up the modules. When testing on the screen, put the 'ScrPrintTEST' after the labl1.set line, so that it will override the label page layout.

Name of document praddlabl

```
>IN addrdf.def ;Define the data file
>IN labl1.set ;Set up label page layout (or labl.set)
>IN ScrPrintTEST ; FOR TESTING ONLY - REMOVE WHEN FINISHED
>IN chopline ;Chop off long address lines
>IN printbl ;Print the address label with blank lines
```

### Printing Two Labels Side by Side

There is only one problem with the above example. There are normally two labels side

by side on the sheet as shown on the sample address label page. If you have lots of labels to print and you don't want to worry about the proper way, there is a cheeky way of doing it. Print all the labels down the left-hand side, turn the sheets round the other way and print the remaining labels up side down on the other side.

A better way would be to print both the left and right-hand labels in one go. There is a way but to do it, we need to consider some more problems that need to be resolved.

When printing the address, you may notice that if there is nothing defined in a variable, like a blank address line, it is not printed. This is desirable when printing addresses on letters or single labels. However, when printing labels side by side, a blank line in one address will cause a problem.

There are two things that are needed. First, to stop blank lines from NOT being printed (Protext default is not to print blank variables). Secondly, a method that looks for any blank address lines when the label is being printed and replaces any found with a space of a specified length.

## Print Blank Lines

When printing variables, we normally use the '&' character on either side of the variable name to show Protext that the contents of the variable should be printed. If you use the '!' character instead, then even if the variable is blank, it will still be printed.

To show this you could copy and modify the previous addlabel1 file to the following file.

Name of document printbl

```
>>>PRINTING STARTS HERE*****
>DM Going to print Label WITH BLANK LINES to &Name& at &Addr1&
>>>
>-----!-----!-----R
!Name!
!Addr1!
!Addr2!
!Addr3!
!Addr4!
>PA
```

Running this file, you will notice that any blank variables would now be printed as blank lines.



## Chapter 15 Address Labels

Name of document praddlabl1

```
>IN addrdf.def ;Define the data file
>IN labbl1.set ;Set up label page layout (or labl.set)
>IN ScrPrintTEST ; FOR TESTING ONLY
>IN chopline ;Chop off long address lines
>IN printbl ;Print the address label
```

### Shuffling Address Lines

Now it is time to use a method that looks for any blank address lines when the label is being printed. It will replace the blank line with the line below.

It is given the grand name of Conditionals!

### Conditionals

Whenever you want to do something (on your Notepad) that needs a decision to be made, you need to use conditionals. First there is the test, which can be regarded as the question, followed by what to do if that test is true or false.

Fortunately there are only these two answers true or false. There are no 'maybes' or 'I'll think about it'.

#### The Test

Before looking at the commands, let's look at what we are testing to make a decision. This test is usually described in 'computer speak' as a condition. There are two things that need to be compared to do a test. The answer to the test will be true or false. Let's look at a few examples.

```
mother = mother TRUE
mother = father FALSE
2+2 = 4 TRUE
```

There are EIGHT possible comparisons that can be made. These are:

```
= equal to
<> not equal to
< less than
<= less than or equal to
> greater than
>= greater than or equal to
```

IN is contained in (e.g. “br” IN “Brett” is true)  
 (doesn’t care which case text is in)  
 NOTIN is not contained in

## The IF Commands

When Protex has processed our previous examples of control documents, every command line in the document has been executed. The command lines are executed in the order in which we have included them in the control document. The IF command line allows us to change the order that the control document is executed depending on the result of a test.

There are three parts to an IF command. The IF itself starts the process and includes the test that needs to be performed. All the command lines following the IF will be executed if the test is TRUE. The ELSE command tells Protex what to do if the test was FALSE. All the command lines following the ELSE will be executed if the test was FALSE. The END IF command tells Protex where the end of the IF command is. For example, the following few command lines would cause a different message to be printed depending on what the variable ‘colour’ holds:

```
>IF colour = “red”
>>> this is our TRUE block of command lines
>>> show a message for a red jumper
>DM Your jumper is red
>EL
>>> this is our FALSE block of command lines
>>> do this bit if jumper is not red
>DM Your jumper is not red
>EI
>>> finished the IF command
```

You don’t have to include any command lines in either the TRUE or FALSE blocks. If we wanted to change the above example so that nothing was printed when ‘colour’ was any value other than ‘red’, we could just leave out the ELSE part of the IF command.

```
>IF colour = “red”
>>> this is our TRUE block of command lines
>>> show a message for a red jumper
>DM Your jumper is red
>EI
>>> finished the IF command without doing a FALSE block
```

We could equally well reverse the situation so that a message was only printed when the

## Chapter 15 Address Labels

variable was not 'red'.

```
>IF colour = "red"
>EL
>>> this is our FALSE block of command lines
>>> do this bit if jumper is not red
>DM Your jumper is not red
>EI
>>> finished the IF command without doing a TRUE block
```

Let's create a Control Document to play with the different comparisons using the IF commands. To use any of the IF commands you simply pick which command you want to use, followed by the condition. We will type the different variables and conditions from the keyboard and display messages from within the true and false blocks using the ELSE command.

Name of document playcond.

```
>>> Use the Print to Screen facility when printing this control
document
>>> to play with conditionals (and get to know them)
>CS Test two variables - Document playcond
>DM Type in two things to compare with one another
>DM Remember to put text in "quotes"
>>> There is a full stop in the next line to print a blank line.
>DM
>AV word1 word2 "Comparison?" "comp
>IF &word1& &comp& &word2&           ; Start of the comparison
>>> this is our TRUE block of command lines
>DM the test was TRUE
>DM &word1& &comp& &word2&           ; If it's true, print this
>EL                                   ; Else
>>> this is our FALSE block of command lines
>DM the test was FALSE
>DM &word1& &comp& &word2&           ; print this
>EI                                   ; end of the IF comparison
```

Use the Print Screen facility to run this Control Document. You can use it to test out any comparisons you wish. Here are some to get you going.

word1	word2	Comparison	Answer
2	2	=	TRUE
"mum"	"dad"	=	FALSE
"mother"	"mother"	=	TRUE
2+2	4	=	TRUE
"TIP"	"MULTIPLE"	IN	TRUE
""	""	=	TRUE

The last one on the list is similar to the question that we originally wanted to ask. We needed a way of looking for blank lines in an address when a label was being printed. Before we return to our address labels let's just look at the remaining IF commands and nesting.

### Special Condition Commands

**IF variable is Defined - (ID v)**

**IF variable is Undefined - (IU v)**

These two commands are a special form of the IF command since they can test whether a variable has been created by any of the AV, RU, RV or SV commands.

They are useful when used with Ask Variable commands since if a variable has already been defined you need not ask for it again. For example at the beginning of a letter that you are going to use with Mail Merge, you could test to see if the variable 'date' is defined. If it isn't, then you could use the Ask Variable command to ask for the date.

### Updating ScrPrintTEST

Are you fed up watching the message that is displayed when inserting the ScrPrintTEST document? Let's modify the document and use the one of those special condition commands to only show the message the first time through.

```
>ID flag      ; has the flag been defined yet? Yes jump to end if
>EL          ; Else
>SV flag=1   ; No .. better set it and display messages
>WT Press any key to continue . . . ;Wait to display previous
messages
>CS *****
>DM * This file changes the page length of any document so      *
>DM * that it can be displayed on the screen in viewable chunks *
>DM *                ScrPrintTEST document selected.           *
>DM * REMOVE DOCUMENT FROM MAIN LIST WHEN FINISHED TESTING    *
>DM *****
>>> There is a full-stop in the next line
```

```
>DM
>>> wait to display the previous message
>WT Press a key to TEST or 'Stop' twice to STOP
>CS Testing mail merge document ;Clear the screen
>EI ; End of messages block
>CO Clear both the top and bottom margins
>TM 0
>FM 0
>PL 7 ; Set the page length to seven lines
>CP off ; Continuous printing off, pauses between pages
```

You need only add four extra lines (we've marked them by putting them in bold) to the document. The first time the Control Document is printed the variable 'flag' has not been defined. The test is false so the block after the ELSE is printed, which sets the flag and prints the messages. Next time round the 'flag' variable has been defined and there is nothing in the true block, so printing continues from the END IF command line.

### **IF Data File is Exhausted (IE)**

This command checks to see if the Control Document has reached the end of the data file. You can use this at the end of the file to print a summary and tell you that it has finished.

### **SKip IF Condition True (SK cond)**

This block of text would be ignored if the condition was true.

**REMEMBER**, all these IF commands need an END IF command at the end of the block.

### **REPEAT .. UNTIL Condition (RP)**

The two commands REPEAT and UNTIL allow you to perform a block of command lines until a condition becomes TRUE.

Use REPEAT to mark the beginning of the text block and the UNTIL command to mark the end. The REPEAT command cannot be nested. What's nesting? Don't worry about it yet. We'll cover it later.

Unlike the IF command where you put the condition at the beginning, with REPEAT the condition is placed in the UNTIL command. If the condition is FALSE, UNTIL will return to the previous REPEAT command. In the following example, which is an extension of the conditional example above, we have given UNTIL a condition that cannot be satisfied so that it will always return to the REPEAT command. This means that the control document carries on repeating until the STOP key is pressed twice.

```

>>> Use the Print to Screen facility when printing this control
document
>>> to play with conditionals (and get to know them)
>SV repeat = 1 ; Set a repeat flag
>RP ; Make this part repeating
>CS Test two variables - Document playcond
>DM Type in two things to compare with one another
>DM Remember to put text in "quotes"
>>> There is a full stop in the next line to print a blank line.
>DM
>AV word1 word2 "Comparison? "comp
>IF &word1& &comp& &word2& ; Start of the comparison
>DM TRUE &word1& &comp& &word2& ; If it's true, print this
>EL ; Else
>DM FALSE &word1& &comp& &word2& ; print this
>EI ; end of the IF comparison
>WT Press STOP TWICE to END or any other key to continue ...
>UN repeat = 0 ; end of this repeating part

```

## Nesting

Nesting is when you have an IF block within another IF block. Using nesting you can ask several IF statements and then do what you want with the answers. Here is a document that shows how to use nesting.

Suppose we want to find a man who doesn't have a car but has thought of buying one. We could just ask three questions and select the result as appropriate. But, if the answer to the first question is that the user is female, there is no point asking the other two questions. Similarly, if the user already has a car there is no point asking the last question.

By nesting three IF commands we need only ask those questions that are relevant.

```

>>> Use the Print to Screen facility when printing this document
>>> The control document will stop only if you are male, do not have
a car
>>> and have thought of buying one. You can add your own bits in
other
>>> sections if you like.
>SV count = 0 ; put into loop
>RP ; repeat
>CS Please answer questions ; clear screen
>AV "Are you Male or Female? (M/F)" MF
>IF MF[1] = "M" ; (IF level 1) You are male
>>> only ask this question if the user is male
>AV "Have you got a car? (Y/N)" CAR

```

## Chapter 15 Address Labels

```
>IF CAR[1] = "N" ; (IF level 2) You are male and
don't have a car
>>> only ask this question if the user is a carless male
>AV "Have you thought of buying one ? (Y/N)" CARYN
>IF CARYN[1] = "Y" ; (IF level 3) Thought of buying one
>DM I have an E-Type for sale
>ST ; STOP HERE
>EI ; (EI level 3) end of buying one
>EI ; (EI level 2) end of male and no
car
>EI ; (EI level 1) end of male
>DM Sorry to have bothered you
>WT Press any key to have another go or Stop twice to leave
>UN count = 1 ; back in the loop
```

You will notice that every time an IF statement is added, you go down a level. The maximum number of levels that you can use is seven. Every level of IF needs its own END IF.

### Shuffling Address Lines - continued

After that slight diversion, back to the problem at hand. We want to print address labels and we need a way of making sure that there are no blank lines printed in the middle of an address.

We need to check each line of the address for being blank. If it is blank, change the variable to the next one down, then make that one a blank line. This is repeated for each line in the address. To make quite sure that no blank lines are left within the address, this is repeated three times.

Name of document shaddln1

```
>>>Sort out blank lines in the address THREE times
>DM Shuffling First Address lines
>SV count = 3 ; set count to three
>RP ; start of Repeat Block
>If Addr1="" ; if there is a blank line (undefined)
>SV Addr1=Addr2 ; make the next line down this variable
>SV Addr2="" ; and make the next line down - blank
>EI ; end this IF statement
>If Addr2="" ; Repeat for all the other address lines
>SV Addr2=Addr3
>SV Addr3=""
>EI
>If Addr3="" ; Another address line
```

```

>SV Addr3=Addr4
>SV Addr4=""
>EI
>SV count = count-1 ; done a shuffle, take one away from count
>UN count = 0 ; end of repeat block

```

## Testing the Shuffle

All that we need to do now is to test the shuffle by adding the line >IN shaddln1 into the previous list file.

Name of document testshuf

```

>CS Testing Shuffle address labels with NO blank lines
>IN addrdf.def ; Define the Data File (for the first address)
>IN labbl1.set ; Set up label page layout (or lab1.set)
>IN ScrPrintTEST ; FOR TESTING ONLY ...
>IN chopline ; Chop address lines (for first address)
>IN shaddln1
>IN printbl ; Print label with blank lines
>WT Press any key to continue . . .

```

## Printing two Addresses continued

How can we print two addresses side by side from our data file?

By adding some new variables using the >RU command we can effectively fool Protect into thinking that we have double the number of variables in a record to print on a page. We both know that what we really have is a pair of records that need to be printed on a page.

There is no difference, look at the following file.

Name of document 2addrdf.def

```

>CO 2nd Address Definition file
>DM Reading 2nd Address Definition file ...
>IE ; If end of data file
>DM End of data file detected ; Display message
>SV Name2="" Addr21="" ; Set the 2nd set of variables
>SV Addr22="" Addr23="" ; to null to prevent previous
>SV Addr24="" ; address being printed again
>SV Tel_No2="" Fax_no2=""
>EL ; Else if there is some data in the
file

```



```
>CO * Add some more variables for the second address *
>RU Name2 Addr21 Addr22 Addr23 Addr24 Tel_No2 Fax_no2 dummy
>EI                               ; End of the If Exhausted command
```

This document appears to work backwards, as the command that we are using is called **>IE** 'If Exhausted' and not 'If there is any Data left in the data file.' This means that it can make more sense when reading the first time, to start from the **ELSE** command and then the rest of it from the beginning.

The first thing this document does is to check if you've reached the end of the data file (**If Exhausted**). If you have then all the second address variables (that you haven't yet defined the first time round) are set to a 'null' value to prevent the previous second address from being printed again.

After the **ELSE** command (which is where you should start) a similar **>RU** line to the previous **>RU** line is added but with different variable names for the second address. These names are used to print the second address label.

### Additional Files for the Second Addresses

Now we have got the variables for the second address, we need get rid of the blank lines and chop the lines, just like we did in the first address.

The quickest way of doing this is to Open a new document and copy the document used with the first address and alter the variable names. Here are the additional files.

Name of document shaddln2

```
>>>Sort out blank lines in the address THREE times
>DM Shuffling Second Address lines
>SV count=3           ; set count to three
>RP                   ; start of Repeat Block
>If Addr21=""         ; if there is a blank line (undefined)
>SV Addr21=Addr23     ; make the next line down this variable
>SV Addr22=""         ; and make the next line down - blank
>EI                   ; end this IF statement
>If Addr22=""         ; Repeat for all the other address lines
>SV Addr22=Addr23
>SV Addr23=""
>EI
>If Addr23=""         ; Another address line
>SV Addr23=Addr24
>SV Addr24=""
>EI
```

```
>SV count = count-1      ; done a shuffle, take one away from count
>UN count = 0            ; end of repeat block
```

Name of document chopline2

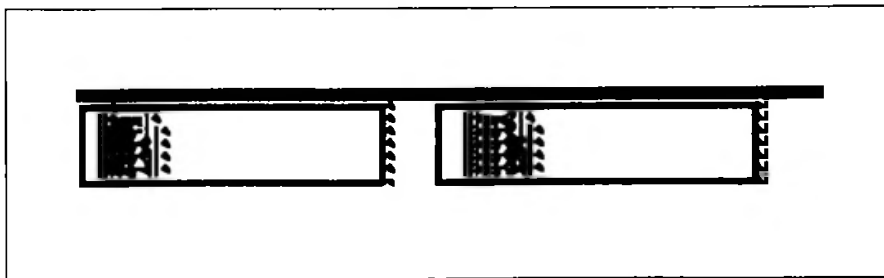
```
>DM Chop off long address lines (second address)
>>> Put the label width in the following variable
>SV Labelw=30      ; Label Width
>>> Chop all the variables in the file to be the width of the label
>SV Name2=Name2[1:&Labelw&]
>SV Addr21=Addr21[1:&Labelw&]
>SV Addr22=Addr22[1:&Labelw&]
>SV Addr23=Addr23[1:&Labelw&]
>SV Addr24=Addr24[1:&Labelw&]
```

## Printing the Two Labels

When printing two labels side by side it is very important to use the Ruler Line with the TAB marks set for the end of the first label and the start of the second label. Then USE the TAB character, not multiple spaces to separate the text.

It is also possible to draw boxes around the address as shown in the following screen shot.

Without the boxes the printing document should look like this.



Name of document addlbbx2

```
>>>PRINTING STARTS HERE*****
>IF Addr21=""           ; If 2nd address is blank
>DM ONLY Printing Label to &Name& at &Addr1& and a blank
>EL                      ; Else display following message
>DM Printing Labels to &Name& at &Addr1& and &Name2& at &Addr21&
>EI                      ; End If
```

```
>-----!-----!R
!Name!                !Name2!
!Addr1!               !Addr21!
!Addr2!               !Addr22!
!Addr3!               !Addr23!
!Addr4!               !Addr24!
>pa
```

### Printing the Double Address Label - Finale

All you do now is to add these additional documents that we have written so far into yet another list file. The beauty of using a list file is that you can build up quite a complicated Control Document from several simple documents. Any improvements made to the simple documents will be automatically included when the list file is printed.

Name of document 2lab

```
>CS Printing TWO address labels - side by side
>IN addrdf.def ; Define the Data File (for the first address)
>IN 2addrdf.def ; Define the Data File (for the second address)
>IN labbl1.set ; Set up label page layout (or lab1.set)
>IN shaddln1 ; Shuffle address lines (for the first address)
>IN shaddln2 ; Shuffle address lines (for the second address)
>IN chopline ; Chop address lines (for first address)
>IN chopline2 ; Chop address lines (for second address)
>IN addlbbx2 ; Print TWO labels in boxes
```

# Chapter 16

## Working with Numbers

As well as using text, it is also possible to do mathematical calculations with Mail Merge. We have already seen in the 'Adding to Variables' section that we can add numbers with the '+' character. You can also subtract, multiply and divide using the '-', '\*' and '/' operators. For example, to convert centigrade to fahrenheit:

### Centigrade to Fahrenheit and visa versa

```
>>> Use the Print to Screen facility when printing this document
>>> This demonstrates how you can convert Centigrade to Fahrenheit
>>> Expression is C = F-32 * 5/9
>SV count = -1
>RP
>CS Fahrenheit to Centigrade
>AV "Number to convert? " fah
>SV cent = fah-32 * 5/9
>DM the answer is &cent&
>>>
>>> This demonstrates how you can convert Fahrenheit to Centigrade
>>> Expression is F = (C * 1.8) + 32
>DM Centigrade to Fahrenheit
>AV "Number to convert? " cent
>SV fah = cent * 1.8 + 32
>DM the answer is &fah&
>WT
>UN count =1
```

### Brackets

You cannot use brackets to force Mail Merge to do a calculation in a certain order. You can get round this by splitting the calculation into smaller parts and using temporary variables.

For example taking a small part out of the document above you could force Protext to do the bracket first by doing this:

```
>>> This demonstrates how you can convert Fahrenheit to Centigrade
>>> Expression is F = (C * 1.8) + 32
>DM Centigrade to Fahrenheit
>AV "Number to convert? " cent
>SV bracket = cent * 1.8
```

```
>SV fah = bracket + 32
>DM the answer is &fah&
```

In this example however, it makes no difference, just my luck!

### Minimum and Maximum Numbers

As with most computers there is a maximum number that you can count up to before the mileometer effect takes place. In Mail Merge the upper and lower limits are:

```
Maximum      2,147,483.64
Minimum      -4,292,819,813.5
```

You can prove it to yourself (just like we did) with the following Control Document.

```
>>> Use the Print to Screen facility when printing this document
>SV count = 2147470      ; Set a large number to start with
>RP                      ; Start of Repeat Loop
>DM &count&              ; display short message (number only)
>IF count => 0           ; Test where numbers go though zero
>SV count = count + 0.01 ; add a bit to count
>EL
>ST FAILED . . .        ; Stop when failed
>EI
>UN count = 1
>ST Failed
```

# Chapter 17

## Going Mathematical with Forms

Mail Merge can be very useful for dealing with pre-printed forms that need to be filled out at regular intervals using calculations. The following document shows how an expense form (which had to be sent monthly) could be prepared while it is being printed. You do this by typing in the various information from the keyboard.

Use the Test Page again to locate the exact position of where the printer will print on the form. When using pre-printed forms you may find that the spacing of the lines is different to those normally used by your printer. If this is the case, you need to change the line spacing or character height on the printer to the same spacing as your form. You should find the printer code in the printer manual to change this. Use the >OC command to send this code to your printer when printing another copy of the test page and the document used with the form.

The document is fully commented and uses commands that we have already covered. If you find some funny characters printed on the screen, this is due to the document not being properly formatted. Delete the funny characters (although you won't be able to see them on the screen) around the area that they are printed and then reformat the paragraph with Control F.

**REMEMBER** to use ruler lines and the TAB key in-between the printed variable names so that they get correctly lined up on the form when printed. **NEVER** use multiple spaces to line the text up because it doesn't work when formatting.

Name of document mathform

```
>IN SCRPRINTTEST ; TESTING remove line when finished testing
>>>OC ; Set the printer code for the correct line height/
>>> ; spacing (if needed)
>zm 0 ; set to zero margins
>DM *****
>DM * THIS FORM IS FOR EXPENSES FORM USD 16 *
>DM * AFTER THE FORM HAS BEEN PRINTED - TURN IT OVER TO *
>DM * PRINT THE ADDRESS FOR WINDOW ENVELOPE 16-11-92 VG *
>DM *****
>WT Press any key to continue . . .
>>>
>>> *** Default Messages for variables ***
>SV CLAIM1, "Various Business Meetings" ; To save time typing
>SV CLAIM2, CLAIM1 ; the same thing
>SV DATE, "See Date on enclosed lists" ; every time, you
```

## Chapter 17 Going Mathematical with Forms

```
>SV RAIL, "See enc.list and tickets"           ; can set global
>SV ACCOM, "See enclosed Receipts"           ; variables.
>SV MEALS, "See enclosed Receipts"
>>>
>>> *** Screen MESSAGES START HERE ***
>>> * Message shows what is printed on *
>>> * the form for each line *
>>> ***                                     ***
>>>
>CS "Claim for expenses incurred by me attending a meeting of"
>AV "Default message &CLAIM1& - Change it? Y/N " YN ; Change Message?
>IF YN="Y"                                           ; YES - Display
>DM "23 Chars [                                     ]" ; the length of
line
>AV "LINE 1      :",CLAIM1 23                       ; and no. of chars.
>DM "33 Chars [                                     ]" ; Notice the number
>AV "LINE 2      :",CLAIM2 33                       ; at the end of >AV
>EI                                                 ; line.
>CS                                                 ; Clear Screen
>AV "on (date) &DATE& - Change it? Y/N " YN        ; Another change?
>IF YN="Y"
>DM Use 'Sym' 'D' for TODAY'S date
>DM "23 Chars [                                     199   ]"
>AV "on (date) :",DATE  23
>EI
>CS
>AV "First/Second class rail fare to London from ...&RAIL& - Change it?
/N
" YN
>IF YN="Y"
>DM First/Second class rail fare to London from ...
>DM "23 Chars [                                     ]"
>AV "LINE 1      :",RAIL 23
>EI
>AV "Total Amount for RAIL Travel? ",RAILAM        ; First Number to be
>CS                                                 ; entered.
>AV "Accommodation &ACCOM& - Change it? Y/N " YN
>IF YN="Y"
>DM Accommodation
>DM "45 Chars [                                     ]"
>AV "LINE 1      :",ACCOM 45
>EI
>AV "Total Amount for Accommodation? ",ACCOMAM     ; Another amount!
>CS
>AV "Meals &MEALS& - Change it? Y/N " YN
>IF YN="Y"
>DM Meals
```

```

>DM "45 Chars [                               ]"
>AV "LINE 1      :",MEALS 45
>EI
>AV "Total Amount for Meals? ",MEALSAM
>CS
>DM Other Expenses (Line 1)
>DM "45 Chars [                               ]"
>AV "LINE 1      :",OTHEX1 45
>AV "Total Amount for Expenses? ",OTHEXAM1
>CS
>DM Other Expenses (Line 2)
>DM "45 Chars [                               ]"
>AV "LINE 1      :",OTHEX2 45
>AV "Total Amount for Expenses? ",OTHEXAM2
>>>
>>>          *** SET all UNUSED figures used to 0
>>>
>IF RAILAM=""                               ; If you pressed
the
>SV RAILAM,0                               ; Return Key for
a'0'
>EI                                         ; amount, this
would
>IF ACCOMAM=""                             ; have put a 'null'
in
>SV ACCOMAM,0                             ; the variable. W
>EI                                         ; check them all
here
>IF MEALSAM=""                             ; and if there is a
null
>SV MEALSAM,0                             ; replace them with
a
>EI                                         ; zero.
>IF OTHEXAM1=""
>SV OTHEXAM1,0
>EI
>IF OTHEXAM2=""
>SV OTHEXAM2,0
>EI
>>> *** ADD UP THE FIGURES FOR TOTAL ***
>>> TOTAL=&RAILAM&+&ACCOMAM&+&MEALSAM&+&OTHEXAM1&+&OTHEXAM2&
>>>
>>> *** PRINTING STARTS HERE ***
>FP OFF ; as we want to print variables at the position shown.
>>>
>>> The following page has been mapped with the test page, and
>>> the pre-printed form so that the variables are printed at

```



>>> the right place.

>>>

>>> Ruler lines are used to show where the text is to be printed

> L-----R

Your name  
and Address to be printed  
in a pre-printed box on the form

Post code TW56 7TR

>SV count = 9 ; There are 9 lines until we print again, to save paper

>RP ; we'll repeat printing a blank line

>SV count = count-1

>UN count = 0 ; until we reach the end

> L-----R

&CLAIM1&

> L-----!-----R

&CLAIM2&

&DATE&

>>> \*\* Notice the full-stop within the ruler line that is used to \*\*  
>>> \* line up the decimal points for the numbers below the line. \*  
>>> \* Use the TAB key to separate the words and numbers and use \*  
>>> \*\* the Format command to line up left margins. \*\*

> L-----.-R

&RAIL&

&RAILAM&

> L-----.-R

&ACCOM&

&ACCOMAM&

&MEALS&

&MEALSAM&

&OTHEX1&

&OTHEXAM1&

&OTHEX2&

&OTHEXAM2&

&TOTAL&

>pa

>-----!-----!-----R

>>> \*\* The other side of the form was blank, so this could be used \*\*

>>> \* to print the address of where it had to sent each month. \*

>>> \*\* Fold the form - use a window envelope. \*\*

>>>

>WT Turn Sheet Over for the address

>SV count = 43 ; There are 43 lines until we print the address, to save

space in this control document

```
>RP          ; we'll repeat printing a blank line
```

```
>SV count = count-1
```

```
>UN count = 0          ; until we reach the end
```

```
>  L-----R
```

```
    To the Expenses Dept
```

```
    Another address
```

```
    to a different place
```

```
    saves time writing the envelope
```

```
>pa
```

```
>>>
```

```
*** END ***
```

# Chapter 18

## Calculating VAT

There may be times when you want to calculate the Value Added Tax when adding up a list of items in a Control Document. What is often overlooked is the rounding error that can occur when multiplying two numbers together. The following document shows how to resolve that error.

```
>>> Use the Print to Screen facility when printing this document
>>> Mail merge document to calculate the Value Added Tax to a net
amount.
>>>
>SV count = 1
>RP                               ; let's put this in a loop
>AV amount                        ; Input an amount
>SV vat_rate = 1.75                ; 10th of %
>SV vat_amount = vat_rate * amount
>SV vat_amount = vat_amount + 0.005 ; Add a half penny to VAT amount
>SV amount = amount * 10           ; Multiply net amount by 10
>SV total = amount + vat_amount
>>>
>>> Convert Total and VAT to the correct figure to display
>>>
>SV vat_amount = vat_amount / 10   ; Divide both the VAT amount
>SV total = total/10               ; and total by 10
>>>
>>> Display message
>>>
>DM vat amount = &vat_amount& Amount = &amount& Total = &total&
>UN count = 0                       ; End of repeat block
```

That is the end of the mathematical functions, now for something completely different.

# Chapter 19

## Printing to Files

We have only been looking at Mail Merge with a view to printing the Control Document to the printer or the Notepad's screen. You can also print a Control Document to a file.

A file is just like a Word Processor document. You have to be careful when choosing a name for the file that your Mail Merge Control Document is going to print to. If you choose a name that you have used before for a document, the Control Document will overwrite whatever was in the document, no questions asked.

The following commands will Print to a File as well as to the screen or the printer. If you only want to print the Control Document to the File, you should use the Print to Screen option or unplug the printer.

### **Open File for Writing (appending) (>WF file)**

To select the name of the file you wish to print to, use this command before using any of the commands that write to a file. Opening a file does not mean that the file will be written to where this command is placed. See the >WF ON/OFF commands.

If you put an 'A' at the end of the WF command line, then the printed data will be added to the end of an existing file.

### **Writing to File on/off (WF ON/OFF)**

The >WF ON/OFF command marks the start and finishing points of the section of the control document that you want sent to a file. Everything in the Control Document (commands, printer control codes, rulers and text) are sent to the file.

### **Write Message to File - used with WF (WM text)**

This is similar to the >DM command that sends messages to the screen, except this one sends it to the file without using the >WF ON/OFF commands.

### **Write File Close (WC)**

This command is used to close the file at the end of the Control Document. You can only have one file open at a time so you must close one before opening another one.

## Chapter 19 Printing to Files

Here are a few more examples:

### Name of Document WF

```
>>> Test for writing a file
>>> Make sure you do not have a document called test and test1
before
>>> printing this document
>WF test
>WM Message to file test
>WF ON
This should be sent to the file test
>WF OFF
>DM Display message . . . ; screen message
>WT Press any key to continue . . ; pause
>WF ON
The second line to the file test
>WF OFF
>WC ; Close the file
>WF test1
>WM Display message test1
>WF ON
This should be sent to the file test1
>WF OFF
>DM Test document running . . . T1 ; screen message
>WT Press any key to continue . . ; pause
>WF ON
The second line to the file Test1
>WF OFF
>WC ; Close the file
```

Here is a time saving file for those Notepad users that get caught in detention at the end of the day.

### Name of Document 100lines

```
>SV count = 100 ; one hundred lines to write
>WF 100.out
>RP
>WF ON
I must lern how to spel and work hard at skool
>WF OFF
>SV count = count -1
>UN count = 0
```

## Writing a Log

You could use the Write Message to File command to maintain a log of whom you have sent letters to. You must create a one line log file first before printing the first time otherwise you will get the 'file not found' message.

```
>WF log a          ; append to log file
>WM Thank you letter sent to &name& on &date& &time&
```

## Control Document to Add New Addresses to a Data File

Remember we said that you could write a Control Document to create an address data file? No? Well here it is anyway. We are using the same variable names as we used three years ago. Well it seems like it anyway.

Name of Document adddata

```
>ZM      ;set all margins to zero in the data file
>CS      ;clear screen
>DM "This will create an address data file - Answer all the lines"
>AV "Name of data file to append " data_file
>AV "Is this a NEW data file (Y/N)?" new
>IF new[1] ="y"
>DM Yes selected
>WF &data_file&
>EI
>RP
>AV "Enter name (or press enter key to finish) " Name
>IF name>""
>AV Addr1 Addr2 Addr3 Addr4 Tel_No Fax_no
>SV dummy=""
>WF &data_file& a
>WF ON    ;turn the write file ON
!Name!
!Addr1!
!Addr2!
!Addr3!
!Addr4!
!Tel_No!
!Fax_no!
!dummy!
>WF OFF   ;turn the write file OFF
>EI
>UN name=""
>ST
```

## Converting Other Data Files

You could also use this technique to convert a data file that has been written in another format into the format that we have been using. To do this you would replace the Ask Variable line that asked you for the name and address on the screen with both the Define File and Read Variable lines so that the Control Document reads the other data file that needs converting. For example:

```
>DF Different ; Data File Name to be converted
>RU First_name Name Name2 Name3 Addr1 Addr2 Addr3 Addr4 ; the
variables in
>RU Bingo_No Tel_No Fax_no dummy ; the
different Data File
```

This file has more fields than our data file has. Printing this document now would create a new data file leaving out the First\_name, Name2 Name3 and Bingo\_No variables.

## Conclusion

Well that's it on Mail Merge and the Word Processor. You can go even further than we have done, but time has caught up with us at last. Using Mail Merge you could create other databases by modifying the file above to store invoices, cricket scores, your CD disks and more.

If you have enjoyed Mail Merge (and if not, why not!) you should find the section on BBC BASIC to be even more enthralling, allowing you to really start programming, manipulating numbers as well as the way you think!

# Chapter 20

## Transferring Documents

Your Notepad can act as your only computer. You can type documents into it, edit them, print them or just save them. You don't have to connect to any other computer and you don't have to transfer any documents. But if you want to - it's easy.

Why would you want to transfer a document to another computer? There are several possible reasons.

### Backups

If you want to keep a document safe it makes sense to have more than one copy of it. You could just print it out and keep the paper copy but this would mean having to type it in again if anything went wrong with the original in your Notepad. This makes it sound as though we expect something to go wrong. In fact that is exactly what we do expect. Not very often and not to everybody. But some people will always forget to change the batteries, some people will always drop the Notepad and break it, some Notepad's will go missing, some Notepad's will go wrong. Backups are an insurance policy.

### No space left

You can add a memory card into your Notepad to give you extra storage space but eventually, if you keep lots of documents, you are bound to run out of space. Depending on the type of information you are keeping you could well delete some of it. However it would be much more convenient if you could save old documents onto a PC for example.

### Somebody Else Needs It

When you have written your masterpiece on the life and habits of the Madagascon Snake Rat you may well want to pass it on to colleagues or friends. You may be working from home on documents that need to go onto an office system at work. Copying any document straight into another Notepad or PC allows the easy transfer of information.

So there are lots of reasons to want to transfer a document. Before you take the plunge you need to make two decisions. First you need to consider the format of the document and secondly you need to consider the method of transfer.



## Document Format

No matter how they were created, all documents on all computers are likely to contain more information than is immediately evident from just reading them. For a document created in a word processor there may be information about how the titles are laid out, about special paragraph settings or where bold text has been used. For a document created in an address book there may be information about the order in which to search for a name, hidden fields for secret data or special commands to control a modem. None of this type of information is normally apparent but it all has to be stored in the document and it is frequently stored differently on different computers. The way that this extra information is stored is called the document format.

The documents that you are most likely to want to transfer to another computer are those created with the word processor. This has been developed from the Protex word processor produced by Arnor and uses their own document format to save your documents in. Protex is available from Arnor for a range of other computers (see the appendix for Arnor's address). If you have Protex available, you can transfer your document without needing to consider its format at all.

If you don't have Protex available and you want to work further on your documents on a PC, you will need to change the format of your documents. The format will need to be one that is readable by the word processor that is available to you. You can change the format by altering an option in the System Settings screen. These settings are detailed below.

## Transfer Method

Here the choice is straightforward. You can purchase a Lapcat Link from Arnor and thus use the parallel connector on your Notepad or you can use the serial connector with a simple cable and standard communications software.

Using the parallel transfer method has three advantages. The first is speed, although you would need a fairly large document before it became an issue. The second is ease of use. There are few settings to consider when using a parallel connection to another computer - there is quite a lot to consider if you are using a serial link. The third advantage is the cable. If you buy the Lapcat Link from Arnor there is a cable supplied. If you wish to use the serial transfer method you need to find the right a cable yourself.

However, as long as you have access to some form of communications software on your PC the serial transfer is entirely adequate. Once you have used it a few times it doesn't seem so complicated and most computer outlets are likely to be able to sell you the right cable cheaply.

## Setting Up for Transfer

You have to check that the settings in two different menu screens are correct before you start transferring documents.

The System Settings menu screen is accessed by pressing the Menu key when you are at the Notepad's main title screen. The only setting that you need to check here is called Document transfer port and format. These are the available settings:

Serial/ASCII	will transfer your document via the serial connector and convert it to a plain ASCII file ready to be loaded into almost any available PC word processor.
Serial/Protex	will transfer your document via the serial connector but will not change its format. You will need to have access to Protex on your PC to be able to edit the document.
Lapcat/ASCII	will transfer your document via the parallel connector and convert it to a plain ASCII file ready to be loaded into almost any available PC word processor.
Lapcat/Protex	will transfer your document via the parallel connector but will not change its format. You will need to have access to Protex on your PC to be able to edit the document.

The Printer Configuration menu is accessed by pressing the Menu key from within the Serial Terminal program. The serial terminal can be entered from anywhere within the Notepad's system by pressing the Function and S keys together.

If you have chosen to use the parallel connector to transfer your documents you do not need to check any of the settings in the Printer Configuration menu. Full instructions for using the parallel link are included with the Lapcat Link.

For serial transfer you need to check the following items. In each case it is simply a matter of setting the Notepad value and the PC value to be the same.

Baud rate	This sets the speed of transfer - the higher the number - the faster the transfer. Possible values are 300, 600, 1200, 2400, 4800 or 9600.
Data/stop bits	This determines how many data bits form each character and how many stop bits are sent after each character. It is very rare that this needs setting to other than 8/1, although if you are using some types of modem you may need to change this to 8/2. Possible values are 8/1,

8/1.5, 8/2, 7/1, 7/1.5+ or 7/2.

- Parity** This determines the parity check bit sent with each character. Again it is very rare that this needs setting to other than None. Possible values are None, Odd or Even.
- Handshake** You can set this to either on or off. When this option is turned on the Notepad will use XON/XOFF handshaking to ensure that no characters are lost during transfer. Most standard communications packages available for the PC support this type of handshaking. It is often called 'Software handshaking'. Many PC packages also support 'Hardware handshaking'. Alternative names used may be 'CTS/RTS' or 'DTR/DSR'. Hardware handshaking has not been implemented on the Notepad.

### What Goes at The Other End?

Again, the choice is straightforward. If you are using the Lapcat Link then your PC needs to run the Lapcat software supplied by Amnor.

If you are using a serial link then you need to provide the software. This is not as difficult as it sounds. Most 'Desktop' packages such as 'Smart', 'Lotus Works' or 'Microsoft Works' include communications programs and 'Windows' has an inbuilt Terminal Program. If you don't have access to one of these packages there are a wide range of 'Shareware' packages available at minimal cost.

Any software that you use on your PC will have some form of settings menu similar to the Notepad's. All you are trying to achieve is to have the same settings on the PC and the Notepad. Precisely what the settings are is normally unimportant.

### What About a Cable?

The cable provides the physical connection between your Notepad and the PC. If you are using the Lapcat Link you will already have a cable. It is supplied with the transfer software.

If you are using serial transfer you need the correct cable. The cable you need is known as a 'Null modem cable' and is readily available from computer outlets. If you are in any doubt about the cable show the following diagrams to your local supplier. You only need to specify whether the serial connector on the back of your PC has a 9 or 25 pin plug.

Notepad end	PC end
9 pin 'D' type	9 pin 'D' type

RX 2 _____	3 TX
TX 3 _____	2 RX
DTR 4 _____	6 DSR
GND 5 _____	5 GND
DSR 6 _____	4 DTR
RTS 7 _____	8 CTS
CTS 8 _____	7 RTS

Notepad end	PC end
9 pin 'D' type	25 pin 'D' type

RX 2 _____	2 TX
TX 3 _____	3 RX
DTR 4 _____	6 DSR
GND 5 _____	7 GND
DSR 6 _____	20 DTR
RTS 7 _____	5 CTS
CTS 8 _____	4 RTS

## Making Them Talk

Once you have connected the two computers with your cable you are ready to make them talk to each other. The Notepad has a Terminal Program available by pressing the Function and S keys together. When you are in the terminal on your Notepad and your PC is running its communications package, pressing a key on the Notepad will cause a character to be printed on the screen of the PC. Similarly, pressing a key on your PC will cause a character to be printed on the screen of the Notepad. If this exchange of characters doesn't happen you need to check the option settings to make sure that both computers are set up to the same values.

Once you have the two computers talking to each other you can transfer your documents.

## Transfer Protocol

This is another one of those computer buzzwords. The protocol is the name given to any extra commands or characters that two computers use to make sure each is in step with the other. There are only two choices for the Notepad; either plain ASCII, which effectively means none, or XModem.

The XModem protocol is by far the preferable option if your PC communications package supports it.

You will now have your two computers talking to each other and you will have sorted out how you will transfer a document. The only thing left to do is the actual transfer.

### **Plain ASCII Transfer**

You can use this method as a last resort. When you ask the Notepad to send a document using plain ASCII, it just sends each character from the document out through the serial port. When the Notepad receives a plain ASCII document, it simply saves any characters appearing at the serial port into a document.

#### **Plain ASCII Send**

- 1) Make sure that the Notepad and your PC are talking to each other from the Terminal program in each computer.
- 2) Prepare the PC to receive a document. How you do this will depend on the software package you are using. Alternatives to the word 'Transfer' sometimes used are 'Download' or 'Capture'.
- 3) In the Notepad go to the list of stored documents using Function L and position the cursor over the document name to send.
- 4) Press the Menu key and select T to Transfer.
- 5) Select S to send and the Notepad will start sending the document. You can press the STOP key at any time to halt the transfer.
- 6) When the whole document has been sent the Notepad will display the number of characters transferred and ask you to press the STOP key. This will return you to the list of stored documents.
- 7) Stop the PC communications package. How you do this will depend on the package you are using. It may just be a case of hitting the ESC key on the PC or you may need to select a menu option.

#### **Plain ASCII Receive**

- 1) Make sure that the Notepad and your PC are talking to each other from the Terminal program in each computer.

- 2) Prepare the Notepad to receive a document. Go to the list of stored documents using Function L.
- 3) Press the Menu key and select T to Transfer.
- 4) Select R to receive and type in the name of the document you want to receive. If the document already exists on your Notepad you will be asked to confirm that you want to overwrite it.
- 5) Start the PC sending the document. Again, the way that this is done depends on the package you are using. Common words used to describe the procedure are 'Send Text File' or 'Upload'.
- 6) The transfer can be stopped at any time by pressing the STOP key. If characters stop appearing at the serial port for more than 2.5 seconds the Notepad assumes that the end of the document has been reached.
- 7) You will be asked to press the STOP key and this will take you back to the list of stored documents.

### **XModem Transfer**

XModem is a more secure method of transferring documents between two computers. The details of the protocol are not important other than the fact that it allows the receiving computer to check that all the characters it has received are correct. The XModem protocol is well established and it is likely that any software package for the PC will support it. You should always choose to use XModem if it is available to you.

The Notepad supports the XModem/Checksum variant of the protocol because it is the older, and therefore more widely used, method of transfer. A slight disadvantage of this method is that if your PC software automatically uses the XModem/CRC variant, there will be a few seconds pause before transfer begins while the PC package switches to using XModem/Checksum.

The XModem protocol always sends documents in 'packets' of 128 characters at a time. The only effect of this is that it may appear that more characters have been sent than are actually in the document. These extra characters are used to fill-up the last packet to 128 characters. They are removed automatically from your document the first time you load it into the word processor.

### XModem Send

- 1) Make sure that the Notepad and your PC are talking to each other from the Terminal program in each computer.
- 2) In the Notepad go to the list of stored documents using Function L and position the cursor over the document name to send.
- 3) Press the Menu key and select T to Transfer but do not start the Notepad sending just yet.
- 4) Prepare the PC to receive a document. How you do this will depend on the software package you are using. You may find word 'Download' used instead of 'transfer'.
- 5) On your Notepad Select X to begin the XModem transfer. You can press the STOP key at any time to halt the transfer. While the document is being sent to the PC a count of the numbers of characters transferred is displayed.
- 6) When the whole document has been sent, the Notepad will ask you to press any key. This will return you to the list of stored documents.
- 7) The PC communications package will stop automatically.

### XModem Receive

- 1) Make sure that the Notepad and your PC are talking to each other from the Terminal program in each computer.
- 2) Prepare the Notepad to receive a document. Go to the list of stored documents using Function L.
- 3) Press the Menu key and select T to Transfer.
- 4) Select R to receive and type in the name of the document you want to receive but do not press the ENTER key just yet.
- 5) Start the PC sending the document. Again, the way that this is done depends on the package you are using. Common words used to describe the procedure are 'Protocol Send' or 'Upload'.
- 6) Once the PC has started sending the document press the ENTER key on your Notepad.

7) When the transfer is complete you will be asked to press the STOP key and this will take you back to the list of stored documents.

## **Address Book Transfer**

The Address Book uses a document to store your names and addresses that is a little different to a normal word processing document. However you can still transfer it to and from a PC by using special commands within the Transfer screen.

To be able to see the Address Book document you need to alter the setting of one of the options in the System Settings Menu. Set the option called 'Document date/size display' to anything other than 'Not shown' and the Address Book will be listed with all your other documents. You can then transfer it using the same method as above except that you need to use the 'Receive Address Book' or 'XModem Receive Address Book' options instead of 'Receive Document'.

## **Using a Modem**

In principle it is possible to transfer documents between a notepad and another computer via a telephone connection using a modem at each end. There are two main problems that need to be overcome.

Whenever you leave the Terminal on your Notepad to go to any other part of the system, the Notepad turns off the serial port. This is done to conserve battery power. However, many modems will disconnect from the telephone line if they detect the serial port being turned off. The result of this is that you will lose your connection with the computer at the other end. Some types of modems have special commands to specifically prevent losing the connection in this instance. Alternatively, it may be possible to 'fool' the modem into believing the serial port is still turned on by extra wire links on the modem plug. In either case, the exact method of solving the problem will depend on the particular modem.

The other problem when using modems concerns the software at the 'non-Notepad' end. Since you are calling up the computer via telephone you will not be able to press any keys on the computer. It must be ready to automatically answer the telephone and to be able to be controlled by you from your Notepad. There certainly are such software packages available to run on most computers, but they are not straightforward to use and will require more 'setting up' than the simple transfer software mentioned above.



## The Memory Card

The Notepad has a limited amount of internal memory to store your documents. This memory may be more than sufficient for many users but if you require more space the main option open to you is to add a memory card.

Memory cards are available in a variety of sizes starting at 128K and currently going up to 1Meg, although the upper limit is being continuously extended. What do these sizes mean in English?

The size refers to the amount of memory inside the card - not its physical size. That is standard for all the cards and is the same for a whole range of computers, not just the Notepad.

There are smaller cards advertised but the smallest card generally available is 128K. This means that it can store one hundred and twenty-eight thousand characters. That may sound a great deal of information but an average page of text in the word processor takes up about two thousand characters, so a 128K card can store roughly sixty-four pages of text.

The 1Meg memory card holds one million characters or about five hundred pages of text. That is quite a lot of typing but don't think that it is way beyond your requirements. You may think that a 128K card will cover all you ever need but nobody will buy it back from you if you find you need a larger one in the future. And the price difference between the small cards and the large ones is not that great. You would be wise to go for the largest card you can afford, and then just hope that you don't catch the typing 'bug' and fill it. end up filling it anyway.

### Installing a Memory Card

When you buy a memory card, you should find that it is supplied with its own battery. This allows the card to 'remember' the information it holds even when it is taken out of the Notepad. You need to fit the battery before you insert the card into your Notepad. This is just a case of removing a small draw in the end of the card, dropping the battery in and replacing the draw.

Most cards have a tiny lock on the draw to prevent its accidental removal. All cards will have a 'write protect' tab next to the battery draw. If the write protect tab is slid over to the outside edge of the card, your Notepad will be unable to write to the card. This feature can sometimes be useful for transferring documents between Notepads as we will explain in a moment.

The only thing to be wary of when you fit the card is to make sure that your Notepad is turned off. The same applies if you are going to remove the card. It is possible to destroy information on the card if the Notepad is turned on during insertion or removal.

Once you have plugged your new card into the Notepad the only requirement is to 'Format' it. This process clears the memory inside the card and prepares it for saving documents. It can also be used to wipe clear any documents stored there so you need to make sure that you don't choose this option by mistake once you have started saving information to it.

### **Formatting a Memory Card**

The command to do this is available by pressing the Menu key once you are in the list of stored documents on your Notepad. Select F to format and you will be asked to confirm that this is really what you want to do. The process will only take a few seconds and it can't be stopped once started. There is no way of recovering any information from a memory card that has been accidentally formatted.

### **Using Your Card**

In practice you don't need to do anything to use your card once it is inserted and it has been formatted. The word processor and the address book will both use a memory card by preference if it is available. They will automatically store any documents you look at onto the card and delete them from the internal memory. You can use this fact to copy documents onto a card to transfer to another Notepad.

### **Moving Documents Using a Card**

Suppose that you have a document on one Notepad held in its internal memory that you wish to transfer to another Notepad.

If you insert a memory card into the first Notepad, select the document into the word processor then just press the STOP key, the document will be moved to the card for you. The document is no longer in your Notepad - it is on the card. If you want to keep a copy of the document in the original Notepad, slide the write protect tab over on the card and select the document again.

When you press STOP this time, you will find you have two copies of the document - one on the card and one in your Notepad.

If you insert the card into another Notepad, still with the write protect tab on, and access the document you want to copy, pressing STOP will take a copy of the document into the second Notepad.

### **The Address Book on a Memory Card**

As with the word processor the Address Book will automatically use a memory card if it is available. If you have been using the address book with a card installed and then you remove the card you will find that your address book is empty - all the information was stored on the card.

If you add details into the address book without your card inserted, when you next insert the card you have effectively created two address books - one saved on the card and one saved in the internal memory of the Notepad. You can merge these two books into one by simply accessing the address book and pressing the STOP key. Your Notepad will then ask you if you want to merge the two books or leave them separate. There is no harm in keeping the two books separate apart from the fact that you will only be able to access the information stored in the internal memory of the Notepad.

### **Programming a Memory Card**

This is very much a specialised function of the Notepad. It can store additional programs directly onto a memory card and then run them as though they were part of the Notepad's inbuilt system.

This feature is aimed at software developers and is far beyond the scope of this book to cover.

# Chapter 21

## BASIC

### Why Bother?

"I've just bought this computer that I can use in five minutes. It does everything that I want to do. Why on earth do I need BASIC, and what is it anyway?"

These are fairly sensible questions. Let us take each question in turn and see if we can persuade you that it is worth playing with BASIC.

First, does the Notepad really do everything you would like it to? It is a very powerful little box of magic with a range of ready-to-use functions but it doesn't do everything. No designer can include everything that everybody might want in a new design.

Suppose you wanted to know how much money to put aside each month to cover your quarterly electricity bill. You could search out last year's bills, add the four of them up (if you found them that is), add a bit for inflation then divide by twelve. The Notepad's Calculator would do the maths for you so you would get a perfectly good answer. But what happens when the next bill comes in and the figures have changed? You would then need to do the calculation all over again.

A small BASIC program could allow you to store the old figures and any new ones as they came in, as well as the calculation. It could then instantly produce a summary telling you the best sum to save each month. Once written BASIC would make the whole process very easy.

That is a trivial example but the point is that BASIC is very good at financial calculations, especially if they need doing regularly. You could of course do them from scratch each time but we're sure that you can see the advantages of letting BASIC do the hard work for you. All you have to do is tell Basic how to do the calculation once and your Notepad will 'remember' how to do it again using different figures if necessary.

BASIC is also very useful when you want to do the same operation on a lot of information. For example, suppose you use your Notepad to work on legal documents at home. These documents typically have a blank line between each line of text. That is not too difficult to cope with on a large PC screen, but on the Notepad you would only see four lines of text at a time. Half the screen would be wasted. It would be more convenient to write the document normally and put the blank lines in later. A BASIC program could add the blank lines automatically for you when you wanted to print the document out or transfer it to the office PC.

Again, this is a trivial example, but hopefully you can see that there are no real limits to the uses to which you can put a BASIC program. It may not be as fast at processing information as other programming languages but it will get there and it is easy to pick up.

The last reason for learning about BASIC is perhaps the most compelling. It is great fun. There are few things more rewarding than spending some time writing a program and then seeing your computer do exactly what you planned.

## What is BASIC?

Let's start with the correct definition. 'Beginners All-purpose Symbolic Instruction Code.' It's much easier to say BASIC.

Now let us see a more useful definition. 'BASIC is a way of telling a computer how to do something'.

The biggest problem with computers is that they are unbelievably stupid. They don't know how to do anything. They can't even talk with the real world. Left alone they would sit silently in their little box, using batteries but not doing anything. A program is a way of getting a computer to do something. Just like magic it brings the computer to life.

Programs come in lots of different flavours. The programs you have been using on your Notepad so far have told the computer what to do when you press the 'ON' button, what to write on the display for you to read and what to do when you press a key. They are the inbuilt programs that make up the operating system. Together they literally Operate the System. The programs have been written especially for the Notepad in a language that the computer can understand directly. The language is difficult to understand for any normal human being.

BASIC is different because it is easy for a normal human being to understand. Many individual instructions that make up the BASIC language are literally 'English'. For example 'LOAD' is the command to load a program, 'SAVE' to save a program, 'LIST' to list a program or 'INKEY' to read the keyboard.

However, don't get the idea that BASIC is limited in what it can do because of its 'understandability'. There are many full-blown professional computer programs written in BASIC. Indeed, many years ago Bill Gates, the founder and chairman of the largest software company in the world, made a simple challenge. He maintains that he can solve a problem with a BASIC program faster than anyone can solve it with a different language. He has never yet been proved wrong.

## Our Aims

The best way to learn how to program is to write programs. The best way to learn BASIC is to write programs in BASIC. Our aim is not to teach you the language or programming techniques to any great depth. Don't worry - it isn't going to get complicated. There are many books on the subject and we don't have the space here to even cover all the commands in detail. We want to whet your appetite and allow you to start learning if you want to.

In the following pages you will find lots of examples of programs. We recommend that you type them into your Notepad and see what they do. Don't leave it at that; try altering them a little. See what you can make happen.

## The Layout of the Examples

For each major topic or idea of BASIC that we will be covering we have adopted a similar framework for the explanation.

First we will explain what the new concept or BASIC command does in Plain English, usually with an analogy.

Then we will show you a short example program that makes use of the BASIC command.

Finally we explain the example program in detail highlighting any points to be wary of.

## A Few Simple Rules

To start with, here are a few simple rules.

- 1 You can switch to BASIC from anywhere in the Notepad by pressing the Function key followed by 'B'.
- 2 You need to type the commands into BASIC in capital letters.
- 3 Each line of a BASIC program starts with a line number. At the end of each command or line of commands you need to press the ENTER key.
- 4 BASIC won't normally do anything until you tell it to 'RUN' a program.
- 5 THIS IS IMPORTANT, PLEASE NOTE you must remember to 'SAVE' your program whenever you leave BASIC because it does not automatically save your work when you leave it. If you forget to Save your work you will lose everything

you did in that visit to Basic. There is a way around this however. If you set 'Preserve context during power off' to 'yes' in the System Settings menu your Notepad will not lose your work even if you switch it off without saving your BASIC file.

## **A Word of Reassurance**

You are not going to destroy your computer by getting things wrong. The most likely result of a mistake is that BASIC will say 'ERROR' and you can probably live with that. You can't damage your Notepad with any of the commands we are going to use. There are some commands however that could cause you to lose the contents of documents if you make a mistake. We will highlight these where appropriate. If you have followed our instructions earlier in this book you will have everything backed up anyway so there shouldn't be a problem even if you do lose something.

## **Your First Program**

One day computers will understand the English language, in which case you could enter a program like this:

'This is the first version of my first program. Add the two numbers 3 and 4 together and put the result on the screen. Then stop'.

Unfortunately computers cannot yet understand 'English' so we have to tell them what to do in language they can understand.

```
10 REM The First One    version 1
20 PRINT 3+4
30 END
```

Ok, so what does that all mean then? First, try typing the program into your Notepad.

Step 1: Go into BASIC. Function 'B' will take you there from anywhere else in the Notepad.

BASIC will respond with the following screen:

```
BBC BASIC (NC100) Version 3.10
(C) Copyright R.T.Russell 1992
>
```

Step 2: Type in the program exactly as above. You need to press the ENTER key at the end of each line.

The screen should now look like this:

```
BBC BASIC (NC100) Version 3.10
(c) Copyright R.T.Russell 1992
>10 REM The First One    version 1
>20 PRINT 3+4
>30 END
>
```

Step 3: Run the program. To do this you just need to type 'RUN', on its own this time without a line number, followed by the ENTER key.

Now you should see the number '7' turn up on the screen. If you do, you have just written your first successful program. If all BASIC does is to say something like 'Error in line 20' check that you have typed in exactly what appears above. BASIC is very unforgiving about typing mistakes.

## LIST

You can see exactly what you typed in by using the 'LIST' command. This simply lists your program, starting from the lowest line number and stopping at the highest line number. That is fine for short programs, but what happens when a program 'grows' to more than eight lines. If you type 'LIST' followed by a line number, BASIC will list only that line number. If you type 'LIST' followed by two numbers, BASIC lists all the lines that fall between the two numbers. So, using the above example program:

```
LIST           would list the program
LIST 20       would list only line 20
LIST 20,30    would list lines 20 and 30
```

## Line Numbers

The program contains three commands. BASIC knows the order in which to run the commands because you have put a line number against each one. Without you telling it otherwise, BASIC always starts at the lowest line number and works up. Had you entered your first program like this:

```
30 END
20 PRINT 3+4
10 REM The First One    version 1
```

the result would have been the same. So line numbers are important. They don't have to increase in steps of ten; the steps can be as large or as small as you like. Leaving gaps in



the numbers allows you to alter the program without having to retype it all. For example, you could add:

```
25 PRINT 45+61
```

and 'RUN' the program again. Now you have the results of two sums and you have only had to enter one extra line.

## Commands Used in Your First Program

The commands are close to English.

### REM

'REM' is short for remark. You can type anything you like after 'REM' because BASIC will ignore it. You use 'REM' lines to leave notes to yourself about the program, or that particular bit of it. It is a good idea to get into the habit of adding lots of remarks to your program. You may understand your thinking behind the program now, but you will have forgotten in six months time. If you want to alter the program but you can't quite remember what it does you will have to work it out from scratch again. Plenty of remarks make life easier.

### PRINT

'PRINT' is one of the most powerful of BASIC's commands. It is one of the main ways that your BASIC program can produce some output. It would be pointless if you wrote a program to work out a complicated calculation and then kept the answer secret. Your program must output the answer to somewhere where you can use it. 'PRINT' lets you put things on the screen, change what is in a document, send characters to the printer or send characters to the serial port. Here we have just used it to add two numbers together and print the answer onto the screen. We will cover the ways that 'PRINT' can output information to places other than the screen later.

### END

In the above example, we didn't need the command 'END'. When BASIC runs off the end of a program, it stops anyway. However, it is good practice for you to decide where the program should stop, rather than letting BASIC decide by accident. In later program examples, you will see that the 'END' command becomes essential.

## The Next Step

If you have typed in the example, try changing the program and see what the results are. You have seen how we added line 25; try adding some new lines, or changing the sum that you want printed. What would you expect to be printed by the following line?

```
26 PRINT 8-4, 8+4, 73-46
```

The command is asking BASIC to print the answers to three sums one after the other.

The print command can be extended by adding messages. We could add line 15:

```
15 PRINT "The sum of 3 and 4 is ";
```

The semicolon is important. It tells BASIC not to move the cursor when it has finished printing. This allows the answer, "7" here, to be printed on the same line as the message. You could combine the message with the sum, as in:

```
27 PRINT "The sum of 5 and 4 is ";5+4
```

## Saving Your Work

BASIC won't save your work unless you tell it to. The command is quite straight forward:

```
SAVE "PROG_ONE"
```

The rules for naming a BASIC document are the same as for the rest of the Notepad. As always it is wise to give your document a name that shows what it is so that you can find it again.

The next time you want to use this program, just enter BASIC and type:

```
LOAD "PROG_ONE"
```

## Variables

An important idea in programming is the use of variables. You can think of a variable as simply 'a place to store some information'. The easiest way to explain the use of variables is to draw an analogy. Suppose you had three boxes, each with a blank sheet of paper in it. Then suppose that you wrote 'a' on the front of the first box, 'b' on the front of the second box and 'sum' on the front of the third box.

## Chapter 21 Basic

```
10 REM Using variables
20 LET a = 27
30 LET b = 13
40 LET sum = a + b
50 PRINT sum
60 END
```

The variables in this program are 'a', 'b' and 'sum'. The numbers after the equals signs are the figures stored in each box. The command 'LET' assigns a value to a variable. So:

20 LET a = 27 tells BASIC to store the value 27 in the variable 'a'.

30 LET b = 13 stores 13 in the variable 'b'.

These two lines are the same as you writing 27 on the sheet of paper in box 'a' and 13 on the sheet of paper in box 'b'.

40 LET sum = a + b This is the line that does the work.

It tells BASIC to take the value stored in variable 'a', add it to the value stored in variable 'b' and store the result in the variable 'sum'. To go back to our analogy, you would look at the number written on the sheet of paper in box 'a', add that to the number written on the sheet of paper in box 'b' and write the answer on the sheet of paper in the box labelled 'sum'.

50 PRINT sum

This will print the contents of variable 'sum', just like you reading the sheet of paper in the box 'sum'.

## What To Call A Variable

The important concept to grasp is that BASIC is only concerned with what a variable contains. It just uses the name as a way to find the contents so you can call the variable anything you like. You can use this fact to your advantage because you can choose a name that is convenient for you. So for example, if you are programming a calculation using the rate of VAT, call the variable holding that rate 'VAT\_rate', then you will always know what it means. By carefully choosing your variable names, you can make your programs much easier to write in the first place and make them far easier to change in the future.

There are very few restrictions on variable names but you need to remember that they are 'case sensitive'. This means for example that the variable names 'TOTAL\_SUM' and

'Total\_sum' are different as far as BASIC is concerned. The underscore character can be used to good effect to separate words; space is one of the few characters that cannot be used in a name.

You need to make sure that you don't confuse both yourself and BASIC by giving variables names that are commands.

It is easier to read a BASIC program if you can immediately tell which words are BASIC commands and which words that are variable names. A quick way of doing this is to use 'lower case' letters for your variable names. This is the policy we have adopted in all the example programs contained in this book.

## Variables Never Forget

Variables don't forget what they hold. They can only be overwritten by your program. In the above program, just by using the contents of the variables 'a' and 'b', you did not destroy them. BASIC took copies of what the variables held and left the original contents alone. So we could add:

```
55 LET sum = a - b
56 PRINT sum
```

When BASIC reaches line 55, variable 'a' still holds the value 27 and variable 'b' still holds the value 13. This line will put a new value into variable 'sum'. This is the same as you writing a new number on the sheet of paper in the box labelled 'sum'. The new value is printed in line 56.

## Variables on Both Sides

The same variables can appear on both sides of the equal's sign. This doesn't sound sensible until you follow through what BASIC is actually doing. Suppose we added:

```
57 LET sum = sum + 5
```

Thinking of our boxes again, we want to look at the number written on the sheet of paper in the box 'sum', add 5 to it and write the answer back on the sheet of paper in box 'sum'. We could do it, and so can BASIC. In fact, the most used BASIC line in history is probably something like  $\text{count} = \text{count} + 1$

So far we have assumed that a variable can hold anything we want, but that isn't quite true. There are three types of variables available in BASIC.

## Real Variables

These can hold normal everyday numbers and BASIC doesn't need to be told anything about them. You just use a variable name and BASIC creates somewhere to store the contents of that variable. All the following numbers could be stored in real variables:

27, 456.8, 752467, -37.9, -100.001

## Integer Variables

These variables also store numbers, but they can only hold whole numbers. There are two very good reasons for using integer variables wherever you can; speed and accuracy.

Speed comes about for two reasons. First, the space that BASIC needs to store an integer variable is less than that required for storing a real variable. Because it takes less space, it is quicker to fetch and store the contents. The second reason for increased speed is simply that the maths is easier. It wouldn't take you as long to add 54 and 78 as it would to add 54.768 and 78.923. The same applies to BASIC.

Accuracy is available from the use of integer variables purely because they can only be whole numbers. If you have defined a variable as an integer and assigned it the value of 2538, then its value is exactly 2538. It can never be 2537.99999999 or 2538.00000001. If you were keeping a count of the number of pints of milk you bought in a month, an answer of 98.999997 would be totally unexpected. Since you know the answer must be a whole number, it is much more sensible to use an integer variable to keep the count.

You don't have to use an integer variable just because the contents will always be whole numbers. You will improve the speed and accuracy of your programs if the contents of the variable can only be a whole number.

How do you create an integer variable? You give it a name the same as you would if it was a real variable but you add '%' after the name.

count%, increment%, Total\_num% — are all integer variables  
sum, Total, rate, maximum — are all real variables

## String Variables

String variables can contain almost anything, but are usually used for text. So you could store a name and address, the title of a book or a telephone number in a string variable. The only limit on strings is that they cannot be more than 255 characters long. You tell BASIC that a variable is a string variable by adding '\$' after its name.

Name\$, Tel\$, ROOM\_NOS — are all string variables

## The INPUT Command

So far our demonstration program has been of little practical value. It can add two numbers and print the answer for us but we have to change the program to change the numbers we want to add. It would be much more useful if the program could stay the same but could operate on different information.

```
10 REM First program version 2
20 INPUT value_1
30 INPUT value_2
40 LET sum = value_1 + value_2
50 PRINT sum
60 END
```

In our original program, we put the numbers into our variables using the BASIC 'LET' command. The INPUT command performs the same task as LET. Instead of assigning the variable a value from the program, it takes the value from the Notepad's keyboard. You type the number in. You can type in any number you like.

When you RUN the program, a '?' appears on the screen. This is BASIC's way of asking for you to type something. In this example we have asked BASIC to 'INPUT' a real variable (there is no '%' or '\$' after the variable name) so you need to type a number. BASIC will then print another '?' and you type in another number. The program adds the two numbers together and prints the answer.

## Keep it Friendly - Please

We have a useful program but it is very unfriendly. If somebody else tried to use our program, they would not know what to do to get an answer. You may not even remember what to do if you ran this program again in six months time. We need to add some instructions for BASIC to print on the screen.

One way to add some instructions would be with the 'PRINT' command that we have already used:

```
15 PRINT "Type in the first number"
25 PRINT "Type in the second number"
50 PRINT "The sum of the two numbers is",sum
```

Remember that BASIC will put these lines into the correct order based on the line numbers. Our program would then read:

```
10 REM First program version 2
15 PRINT "Type in the first number"
20 INPUT value_1
25 PRINT "Type in the second number"
30 INPUT value_2
40 LET sum = value_1 + value_2
50 PRINT "The sum of the two numbers is",sum
60 END
```

There is a shorthand route to achieving the same results if we change the program to look like this:

```
10 REM First program version 2
20 INPUT "Type in the first number",value_1
30 INPUT "Type in the second number",value_2
40 LET sum = value_1 + value_2
50 PRINT "The sum of the two numbers is",sum
60 END
```

Here we are telling BASIC to print the instructions contained within the quotes and then to input a value to put into a variable all in one go.

## The GOTO Trap

When BASIC runs a program it starts at the lowest line number it can find, actions that line and then looks for the next highest line number it can find - unless you tell it otherwise.

Traditionally, the most common way of telling BASIC to action a line other than the next highest is the GOTO command. This literally tells BASIC to 'GOTO' another line. Once BASIC gets there, it carries on running through the line numbers as before.

```
10 REM First program version 3
20 INPUT "Type in the first number",value_1
30 INPUT "Type in the second number",value_2
40 LET sum = value_1 + value_2
50 PRINT "The sum of the two numbers is",sum
55 GOTO 20
60 END
```

Line 55 puts the program into an endless loop. BASIC asks for each number, prints the sum and then goes back to ask for another two numbers. It will continue to go round this loop until you press the 'STOP' key.

In this example, using 'GOTO' is not a problem. But you need to be wary of its use when you start to write complicated programs. As your program becomes more complex it is good practice, and ultimately becomes essential, to structure the program into manageable sections. You can write and test each section independently before moving onto the next section. If you tried to write a large program in one big lump both you and BASIC would probably get lost.

The GOTO command allows you to jump from anywhere in your program to anywhere else in your program. This potentially ruins any structure that you have been following. Does it really matter about structure? Try following this program:

```

10 GOTO 200
20 GOTO 100
30 PRINT "This is printed first"
40 GOTO 70
50 PRINT "This is printed last"
60 GOTO 150
70 GOTO 250
80 PRINT "This is not printed"
100 GOTO 50
150 END
200 GOTO 30
250 PRINT "This is printed next"
260 GOTO 20

```

The program works and does exactly what it is supposed to do. But it is a pain to follow. Imagine what it would be like if this was a five hundred line program with lots of processing between the GOTO commands. How would you ever find a mistake in the program?

GOTO has its uses, but there are better ways of doing the same thing that don't have the pitfalls of GOTO. Indeed Richard Russell, the author of the BASIC in your Notepad, claims never to use the GOTO command at all. We can only agree with his policy.

## Halting Gracefully

By introducing 'GOTO' into version 3 of our program we have put the program into an endless loop. The only way of stopping it is to press the STOP key on the Notepad. This is not a very elegant way to halt proceedings. What we need is a 'graceful' exit route.

```

10 REM First program version 4
20 INPUT "Type in the first number",value_1
25 IF value_1 = 999 THEN END
30 INPUT "Type in the second number",value_2

```



## Chapter 21 Basic

```
40 LET sum = value_1 + value_2
50 PRINT "The sum of the two numbers is", sum
55 GOTO 20
```

Line 25 is a test that BASIC can perform. It means 'Do a test. If the answer to the test is true, do something. If the answer to the test is not true, do something else'.

In this example we have said:

If the variable 'value\_1' is equal to nine hundred and ninety-nine, stop the program.

We haven't told it what to do if the variable is not nine hundred and ninety-nine, so BASIC will just carry on with the next line. We have effectively said:

If the variable 'value\_1' is equal to nine hundred and ninety-nine, stop the program. Otherwise, carry on running the program.

What will the program actually do? It will ask for the first number and assign it to the variable 'value\_1'. Then it will test to see if 'value\_1' is equal to nine hundred and ninety-nine. If it is equal, the program will stop. If it isn't equal, the program will ask for the second number, assign it to the variable 'value\_2', print the sum of 'value\_1' and 'value\_2' and then go back and ask for the first number again. We can just keep on typing in pairs of numbers to be added. When we have had enough, we can enter nine hundred and ninety-nine as the first number and the program will stop.

The test that BASIC does in line 25 relies on BASIC's ability to decide if the answer to a question is true or not. Our question was 'is the number held in value\_1 equal to nine hundred and ninety-nine'? BASIC needs to decide if the answer is TRUE, that is 'value\_1' holds the number 999, or FALSE, that is 'value\_1' holds a number other than 999.

In this example we have used the number nine hundred and ninety-nine to end the program but we could have used any number we liked. We just need to choose a number that is convenient to us. Line 25 is testing whether the contents of the variable 'value\_1' is equal to the number we have chosen to stop the program. If it is equal, the answer to the test will be TRUE and the program will stop. If it isn't equal the answer to the test will be FALSE and the program will continue.

## What is Truth?

In English the words true and false are not numbers; they don't have a value. But computers can't understand abstract ideas like truth and falsehood. They can only work

with numbers so we need to give a value to TRUE and a different value to FALSE. What value does BASIC use for TRUE?

The simple answer is 'not zero'. It will probably come as no surprise therefore to find out that BASIC considers FALSE to be 'zero'. If only life was this simple. Why?

The answer to that goes back to the very roots of computers and the way they store numbers which is beyond the scope of this book. The correct answer is that TRUE is actually represented by the number '-1'. Interested readers may like to refer to books concerned with Binary Arithmetic' and the logical 'NOT' function.

We don't have to understand why it is so to use TRUE and FALSE in our programs.

The two values are so important that BASIC has commands just to set them for you.

```
10 REM Using TRUE and FALSE
20 LET flag = FALSE
30 IF flag = TRUE THEN PRINT "Flag is TRUE"
40 LET flag = TRUE
50 IF flag = TRUE THEN PRINT "Flag is TRUE"
60 END
```

Line 20 assigns the value of FALSE to the variable 'flag'. In line 30 we perform a test. Remember how the test is worked out:

'Do a test. If the answer to the test is true, do something. If the answer to the test is not true, do something else'.

At line 30 we are asking BASIC this question:

Is the value in the variable 'flag' equal to TRUE?

The test is going to fail because we have just set 'flag' to be FALSE, so BASIC won't print our text. It will carry on to line 40. Here we have assigned 'flag' the value of TRUE, so when 'flag' is tested again in line 50, the test will pass and the text will be printed.

The job of the BASIC command 'IF' is to do a test. If the result of the test is TRUE, BASIC will do something. But we could save BASIC the trouble of doing the test if we know the answer. In both of the tests at line 30 and line 50 we ask:

Is 'flag' equal to TRUE?

But in our program we have set the variable 'flag' ourselves. Since BASIC is only concerned about what a variable contains the question that we actually ask in line 30 is:

Is FALSE equal to TRUE?

The answer will always be FALSE so BASIC would get the same answer if we changed the question to:

Is FALSE?

The question that we actually ask in line 50 is:

Is TRUE equal to TRUE?

The answer will always be TRUE so BASIC would get the same answer if we changed the question to:

Is TRUE?

What is the point of a question like that? No point at all if we leave it there, but remember that our variable 'flag' contains either TRUE or FALSE. So the question in line 30 could be:

Is 'flag'?

If the value in 'flag' is FALSE, we are asking:

Is FALSE?

So the two questions are the same.

Using this principle we can abbreviate the test in lines 30 and 50:

```
10 REM Using TRUE and FALSE
20 LET flag = FALSE
30 IF flag THEN PRINT "Flag is TRUE"
40 LET flag = TRUE
50 IF flag THEN PRINT "Flag is TRUE"
60 END
```

By bringing in the 'ELSE' part of the 'IF' command, we can make the output of the program more helpful:

```

10 REM Using TRUE and FALSE
20 LET flag = FALSE
30 IF flag THEN PRINT "Flag is TRUE" ELSE PRINT "Flag is FALSE"
40 LET flag = TRUE
50 IF flag THEN PRINT "Flag is TRUE" ELSE PRINT "Flag is FALSE"
60 END

```

At line 30 'flag' is FALSE so BASIC will follow the 'ELSE' part of the 'IF' command and print 'Flag is FALSE'. At line 50 'flag' is TRUE so BASIC will follow the 'THEN' part of the 'IF' command and print 'Flag is TRUE'.

## The Question Mark

BASIC is very particular about what you type in. You have to get the commands exactly correct in order for them to work. This 'correctness' is called the 'Syntax' of the command. You may already have seen the message:

```
'Mistake at line xx'
```

This is BASIC's way of telling you that the syntax of the command is wrong. There are two reasons why BASIC is so pedantic; speed and subtlety.

The speed advantage comes because BASIC only has to be able to recognise a tightly defined range of commands. If it isn't a command BASIC can immediately recognise, it must be wrong. If we left out the vowels in a phrase, you could recognise what was said, but it would take you longer to read it. It would be the same for BASIC.

The subtlety comes from the fact that very often a slight change in the command changes the way it operates. The question mark with 'INPUT' is a good example of this.

```
20 INPUT "Type in the first number",value_1
```

This line prints the following text to the screen:

```
Type in the first number?
```

```
20 INPUT "Type in the first number" value_1
```

This line leaves off the question mark:

```
Type in the first number
```

The text between the quotation marks is called the 'prompt string'. If you include a comma between the prompt string and the variable name, BASIC prints a question mark;

no comma - no question mark.

You can take this principle further. Suppose you just wanted to input a value without either a prompt string or a question mark.

```
20 INPUT ""value_1
```

There are no characters between the two sets of quotation marks and no comma so BASIC prints no prompt string and no question mark.

## Arrays - Back to Variables Again

Suppose you wanted to test whether a dice was accurate or not. If you threw the dice one hundred times you would expect a good dice to land on one face roughly the same number of times as any other face, sixteen times in this case. Performing the test is easy. All you do is to throw the dice and record which face is uppermost. After a hundred throws you count the 'hits' for each face and decide about the dice accordingly.

How could we write a program to do the scoring for us? To start with let's try to write a set of instructions that we could give somebody else to follow.

I am going to give you a number between one and six. If the number is one, put a tick on a sheet of paper. If the number is two, put a tick on a second sheet of paper. If the number is three, four, five or six, put a tick on the third, fourth, fifth or sixth sheets of the paper. When I have given you one hundred numbers tell me how many ticks you have written on each sheet of paper.

To turn this into instructions for BASIC we first need to decide what we are going to call our variables. We need to count the number of throws of the dice so let's have a variable called 'throws'. We need to go round a loop to collect the one hundred results so we are going to need a test to decide when to stop. We need to tell the program which face of the dice to count this time round the loop - 'this\_face'. We need to count the number of times the dice lands on each of the six faces - 'count\_face1', 'count\_face2'.. 'count\_face6'.

```
10 REM Counting dice throws version 1
20 LET throws = 1
25 REM say which throw we are on
30 PRINT "Throw number ",throws

35 REM ask which number the dice landed on
40 INPUT "Which face is on top ",this_face
```

```

45 REM keep a running count for each face

49 REM If the number is one, put a tick on the first sheet
50 IF this_face = 1 THEN count_face1 = count_face1 + 1

59 REM If the number is two, put a tick on the second sheet
60 IF this_face = 2 THEN count_face2 = count_face2 + 1

69 REM If the number is three, put a tick on the third sheet
70 IF this_face = 3 THEN count_face3 = count_face3 + 1

79 REM If the number is four, put a tick on the fourth sheet
80 IF this_face = 4 THEN count_face4 = count_face4 + 1

89 REM If the number is five, put a tick on the fifth sheet
90 IF this_face = 5 THEN count_face5 = count_face5 + 1

99 REM If the number is six, put a tick on the sixth sheet
100 IF this_face = 6 THEN count_face6 = count_face6 + 1

105 REM add one to the number of throws we have taken
110 throws = throws + 1

115 REM loop back for 100 throws
120 IF throws < 101 THEN GOTO 30

125 REM now tell me how many ticks are on each sheet
130 PRINT "Counts for number 1 ",count_face1
140 PRINT "Counts for number 2 ",count_face2
150 PRINT "Counts for number 3 ",count_face3
160 PRINT "Counts for number 4 ",count_face4
170 PRINT "Counts for number 5 ",count_face5
180 PRINT "Counts for number 6 ",count_face6
190 END

```

In line 40 the program asks for the number on the dice which it assigns to the variable 'this\_face'. In lines 50 to 100 it tests the value of 'this\_face' against each of the possible outcomes for the dice. The test will only be true for one line of code; all the rest will test false. A count is incremented in the line that tests true.

In line 110 we keep a count of the number of times we have thrown the dice in the variable 'throws'. This variable is tested in line 120 and when it reaches one hundred and one, we can print out our results.

The test uses one of BASIC's 'operators'. An 'operator' is simply a name for what you do in a sum or a test. So for example, '+' and '-' are both operators. Below is the full list

of operators for numbers. We will deal with operators for strings later to avoid confusion.

### Numeric Operators

- + adds two numbers together  
e.g.  $4 + 3 = 7$
- takes the second number from the first  
e.g.  $4 - 3 = 1$
- / divides the first number by the second  
e.g.  $10 / 2 = 5$
- \* multiplies two numbers together  
e.g.  $5 * 2 = 10$
- ^ raises the first number to the power of the second number  
e.g.  $4 ^ 2 = 16$  (four squared to you and me)  
 $4 ^ 3 = 64$  (similarly four cubed)
- < tests for the first number being less than the second  
e.g.  $5 < 6$  is TRUE  
 $6 < 5$  is FALSE
- > tests for the first number being greater than the second  
e.g.  $5 > 6$  is FALSE  
 $6 > 5$  is TRUE
- <> tests for two numbers being different  
e.g.  $5 <> 6$  is TRUE  
 $6 <> 6$  is FALSE
- = tests for two numbers being equal  
e.g.  $5 = 6$  is FALSE  
 $6 = 6$  is TRUE
- <= tests for the first number being less than or equal to the second number  
e.g.  $5 <= 6$  is TRUE  
 $6 <= 6$  is TRUE  
 $7 <= 6$  is FALSE
- => tests for the first number being equal to or greater than the second number  
e.g.  $5 => 6$  is FALSE

6 => 6 is TRUE

7 => 6 is TRUE

Now, let's go back to our program. It would work. It would do exactly what we asked it to do. But just think of writing a similar program to test a dice with twenty faces. What if we wanted to record separate counts for two hundred different things. We would need two hundred different variable names. There must be an easier way to record lots of information.

Suppose you had six boxes. On the label on the first box you write the number one. On the label on the second box you write two and so on for all six. You then lay the boxes in order in a straight line and label the whole group of boxes 'count\_face'. If you want to store something in one of the boxes, you now need two pieces of information. First you need to know which row of boxes to use and secondly how far along the row your box is.

This is the principle of BASIC's arrays. Instead of having six variables with different names, BASIC uses one name to refer to a row of variables. Since you tell it which array to use and how far along the row to go, BASIC can identify the particular box you want.

BASIC has to be told about an array before you start to use it. This is so that BASIC can reserve enough space to line up all the boxes in one row. Each box in an array is called an 'array element'. Once you have told BASIC about the array, you can use any element in the array by referring to the array name and the element number.

Let's change our instructions for the dice counting program to make use of the principle of position in a row.

I am going to give you a number between one and six. If the number is one, put a tick in the first column on a sheet of paper. If the number is two, put a tick in the second column on the sheet of paper. If the number is three, four, five or six, put a tick in the third, fourth, fifth or the sixth column on the paper. When I have given you one hundred numbers tell me how many ticks you have written in each column.

```

10 REM Counting dice throws version 2
15 REM tell BASIC to save space for six columns of ticks
20 DIM count_face(6)
30 LET throws = 1
40 PRINT "Throw number ",throws
50 INPUT "Which face is on top ",this_face
55 REM keep a running count for each face
59 REM add a tick to column 1,2,3,4,5 or 6
60 LET count_face(this_face) = count_face(this_face) + 1
65 REM keep a count of the number of throws
70 LET throws = throws + 1

```



## Chapter 21 Basic

```
75 REM loop back for 100 throws
80 IF throws < 101 THEN GOTO 30
85 REM now print our results
90 LET print_this = 1
99 REM print the results for one column at a time
100 PRINT "Counts for number ",print_this,count_face(print_this)
110 LET print_this = print_this + 1
120 IF print_this < 7 THEN GOTO 100
130 END
```

Line 20 tells BASIC about our array. We have reserved enough space for six elements that BASIC will now know by the variable names `count_face(1)`, `count_face(2)` ... `count_face(6)`. These are our six columns of ticks.

Line 60 stores our count for us. It replaces the six lines of code we needed in our first version of the program. To understand how BASIC knows where to store the count we need to think of our row of boxes again.

Assume that the first time you throw the dice it lands on a six (you cheated didn't you). The variable 'this\_face' will hold the number '6'. When BASIC reaches line 60, it interprets the code as follows (remember 'this\_face' holds the number '6'):

```
count_face(6) = count_face(6) + 1
```

We have pointed at the sixth box in the line of boxes called 'count\_face'. BASIC looks at the number that the box contains, adds '1' to it and puts it back in the box. BASIC has added a tick to the sixth column.

In line 80 we have looped back to get the next reading until the value stored in 'throws' is equal to one hundred and one.

We have printed the results by pointing at each box in turn. We have set up a loop from line 100 to 120. In line 100 we are asking BASIC to print the box number ('print\_this') and the contents of the box ('count\_face(print\_this)').

```
100 PRINT "Counts for number ",print_this,count_face(print_this)
```

The first time round the loop the variable 'print\_this' is '1' so BASIC interprets the code as follows:

```
100 PRINT "Counts for number ",1,count_face(1)
```

The second time round the loop we have added '1' to the value of 'print\_this' so BASIC interprets line 100 differently:

```
100 PRINT "Counts for number ",2,count_face(2)
```

When we have printed the contents of all six boxes the program will stop.

## The Hidden Array Element

We always think of counting from one but BASIC starts counting from zero. When we allocate space for an array using:

```
DIM count_face(6)
```

BASIC has actually reserved enough space for the elements `count_face(0)`, `count_face(1)` ... `count_face(6)`. The result is that we always get one more element than the number we have asked for. So we could have changed line 20 above to read:

```
20 DIM count_face(5)
```

and we would still have had enough space to store the six numbers for the dice. However, it would have been a little inconvenient. When the dice landed with number '1' face up, the program would have had to store the count in array element '`count_face(0)`'. The number '2' would have had to be stored in array element '`count_face(1)`'. This scheme would work just as well as the scheme we chose, but the program would not have been as straight forward to understand. As a rule it is much easier to ignore the extra array element provided by BASIC and make the relationship between the information to be stored and the array element number as simple as possible. (We stored the count for number '1' in '`count_face(1)`', the count for number '2' in '`count_face(2)`' etc.)

The example programs that we have so far shown have all been relatively short and have been written as one piece of code. They have involved some looping back to run sections of the code more than once but we have always been able to show the whole program in one go. As programs become more complex they soon get to the point where they are too complicated to take in as one block of code. We need a way of splitting the programs into sections that we can write and test on their own and then combine with other sections to form the whole program.

This requirement is the essence of subroutines and procedures. They are a way of separating a program into manageable chunks.

## Subroutines and Procedures

Suppose you have been asked to clean and service a car. A friend has offered to help you. Your friend knows about how to service cars and you don't, so you agree that you will do the cleaning and tell him about anything that needs servicing. He will then service what you have told him to.

You wash the car and then start to polish it. When you get to the wheel on the driver's side at the front, you notice that it is flat. You call your friend over to fix the tyre and you stop for a cup of tea. When he has finished, you carry on polishing. When you get to the rear wheel on the driver's side, you notice that it is flat as well. More tea for you, and another job for your friend. By the time you have polished the whole car, your friend has had to fix all four tyres.

Your prime task has been to clean the car. You weren't concerned about tyres. You don't know whether the tyres just needed extra air in them or whether they had to be replaced. In fact you don't even care since your friend could sort it out and just let you know when he had finished the job. You were free to concentrate on your primary task, cleaning the car.

Your friend was acting like a subroutine in BASIC. A subroutine is a piece of program code that just knows how to do a particular task. It is usually a task that is likely to be necessary more than once. The main program doesn't have to know how the subroutine works; it just has to know that it works and how to find it.

When the main program needs that task performing it will call the subroutine. The subroutine will take over complete control and perform the task. When it has finished it will 'RETURN' control to the main program again. The subroutine can be called lots of times or just once.

```
10 REM Using subroutines
20 INPUT "Input the first number ",value_1
30 IF value_1 = 0 THEN END
40 INPUT "Input the second number ",value_2
50 IF value_1 = value_2 THEN GOSUB 100 ELSE GOSUB 200
60 PRINT "Back in main program"
70 GOTO 20

90 REM Print if numbers are equal
100 PRINT "The numbers are equal"
110 RETURN

190 REM Print if number are different
200 PRINT "The numbers are different"
```

## 210 RETURN

Lines 20 to 70 are the main program. It asks for a number, checks to see if it is zero and stops if it is. If the first number is not zero, the program asks for a second number and tests to see if the two numbers are the same. The program loops back to asking for another number after the subroutines have finished. The main program doesn't know what the subroutines do and it doesn't care.

Line 50 determines which subroutine to call. If the numbers are equal, 'GOSUB 100' transfers control to the subroutine starting at line 100. If the numbers are different, 'GOSUB 200' transfers control to the subroutine starting at line 200.

When either subroutine has finished, in our case just printing a message, it transfers control back to the main program with the 'RETURN' command.

To call the subroutine at line 100 we just had to use the BASIC command 'GOSUB' followed by the line number to run next. It is a very similar principle to the 'GOTO' command. The only difference is that when BASIC meets a 'GOSUB' command, it remembers where it is in the program so that it can 'RETURN' again when we tell it to.

We can trace the path that BASIC would follow through this program:

Line 10 Ignore because of the 'REM' command.

Line 20 Print the message and wait for a number to be typed in which will be stored in the variable 'value\_1'. Assume the number '5' is typed in.

Line 30 Test the contents of 'value\_1' to see if they are zero. Since 'value\_1' holds the number '5', continue to line 40.

Line 40 Print the message and wait for a number to be typed in which will be stored in the variable 'value\_2'. Assume the number '5' is typed in.

Line 50 Test the contents of 'value\_1' to see if they are the same as the contents of 'value\_2'. Since both variables contain the same number you need to go to the subroutine starting at line 100. Before you go to the subroutine, remember the line number you would have gone to, had it not been for the subroutine.

Line 100 Print a message.

Line 110 Return from the subroutine to where you would have gone had it not been for the subroutine.

Line 60 Print a message.

Line 70 Go to line 20

Subroutines have been around since the early days of computers and programming. They are a very useful feature but they have several drawbacks. The first problem is to do with re-usability of code.

When you write your first program everything is new to you. If you want your program to perform a particular task you have to write the code to do the job. However, as you continue to write other programs you will soon find yourself writing very similar 'bits' of code. Although the program you happen to be writing at the time appears to have nothing to do with any program you have previously written, you will find that some functions required will be the same. For example, if your program has to ask for the date, you will need a routine to ask for and check that date. The routine you write could be used in other programs so it would be useful to make it as 'general purpose' as possible. Although a subroutine would do the job it would make your life more difficult later.

Suppose you had written a date checking routine as a subroutine numbered from line 1000 to 1230. In the program for which it was originally written this would have fitted perfectly. But suppose you wanted to use the subroutine in another program. The second program already has lines numbered 1000 to 1230, so you have to change the line numbers in your subroutine to start at line 5000. Even if the subroutine was straightforward, changing all the line numbers to fit in a new program would be inconvenient and may well lead to errors, but think of the problems of its name. In the last program, when you wanted to check a date, you called subroutine 1000. In this program, when you want to perform the same task, you have to call subroutine 5000. How long will it be before you type in the wrong number? On the examples we have seen so far this may not be a very serious problem, but remember that a complex program can easily grow to five hundred or a thousand lines of code. You won't then be able to remember where everything is.

The subroutine name is also meaningless. It doesn't help when you are trying to trace through a program if it keeps leaping off to lots of subroutines. Unless you have written lots of remarks in the code you would have to follow through each subroutine to find out what it did. Life would be much easier if you could give meaningful names to subroutines. In fact you can, and they are called Procedures.

Procedures have two great advantages over subroutines: you can refer to them by name and the variables used within them can be made private.

Why is a name so important? Consider these two lines of code:

```
1000 IF total = 0 THEN GOSUB 5000 ELSE GOSUB 6000
1000 IF total = 0 THEN PROC_no_entries ELSE PROC_print
```

They could both perform the same task. However, the second line of code is much easier to follow. Procedures don't care about line numbers. Provided that the procedure appears somewhere in your program, BASIC will find it. So our date routine mentioned above could be copied into a new program anywhere and we would not have to remember the line number.

## LOCAL Variables

Variables can be made private to a procedure by telling BASIC that they are 'LOCAL'. For example:

```
1200 DEF PROC_no_entries
1210 LOCAL month$,day$
... all our procedure code
1290 ENDPROC
```

In this case the two variables 'month\$' and 'day\$' are private to this particular procedure each time it is run. They don't interfere with any other variables in the program.

Why bother with private variables? Using our date checking routine as an example, it is likely that in such a routine we would need variables to store the year, month and day that the user had entered. Suppose we had written the code as a subroutine and called the variables simply 'year', 'month' and 'day'. The subroutine works as planned in our original program and since we knew we had already used those variable names, we didn't confuse them with anything else.

Now we are writing our second program. We need the user to input a variable for the month and not surprisingly we called it 'month'. That isn't a problem because we don't have our date routine yet so there is no confusion. After writing some more of our second program we find we need to check a date so we add the routine we wrote a while back that we know works perfectly. Suddenly our program stops working the way we expect. Eventually we find that the problem has been caused by two different variables having the same name. Then we kick ourselves for being so stupid. This would never happen in real life, would it? Oh yes it would, and it does.

As far as BASIC is concerned any reference to the same variable name means the same variable. Pretty sensible really. The trouble is we don't always want that. If a variable is

only ever used within a routine 'en route' to producing an answer, we don't want anything else except this routine to interfere with it's value. Procedures allow us to make a variable private. This means that only the procedure itself knows about the variable. The rest of the program cannot affect the value of the variable and doesn't even know it exists.

This ability to make variables private may not seem important but it makes for clean easy to read programs and in the case of reentrant procedures it is essential. We will cover reentrant procedures later. For the moment just think about writing code that is easy to read. That means using obvious names for your variables and that inevitably means variable names that will clash with others elsewhere in the program. If you make the variables private whenever possible, you will avoid any possibility of clashing.

Within reason you can call a procedure any name you like, provided that it doesn't contain any spaces. If you use the underscore character, you can produce meaningful names that you will understand in future.

We can rewrite our subroutine example using procedures to see the difference in the 'readability':

```
10 REM Using procedures
20 INPUT "Input the first number ",value_1
30 IF value_1 = 0 THEN END
40 INPUT "Input the second number ",value_2
50 IF value_1 = value_2 THEN PROC_is_equal
                               ELSE PROC_is_different
60 PRINT "Back in main program"
70 GOTO 20

100 DEF PROC_is_equal
110 PRINT "The numbers are equal"
120 ENDPROC

200 DEF PROC_is_different
210 PRINT "The numbers are different"
220 ENDPROC
```

In this new version the 'REM'arks become superfluous by a sensible choice of procedure names.

The mechanics of using procedures are quite straightforward. You have to define the procedure somewhere in your program and then you just call it with the BASIC command 'PROC' followed by the procedure name. In the above example line 50 calls the procedure.

Line 100 starts the definition of the first procedure. Line 120 ends the definition. You can put any code you like between the start and end of the definition.

We can trace the path that BASIC would follow through this program to see how it differs from the program using subroutines:

Line 10 Ignore because of the 'REM' command.

Line 20 Print the message and wait for a number to be typed in which will be stored in the variable 'value\_1'. Assume the number '5' is typed in.

Line 30 Test the contents of 'value\_1' to see if they are zero. Since 'value\_1' holds the number '5', continue to line 40.

Line 40 Print the message and wait for a number to be typed in which will be stored in the variable 'value\_2'. Assume the number '5' is typed in.

Line 50 Test the contents of 'value\_1' to see if they are the same as the contents of 'value\_2'. Since both variables contain the same number, you need to go to the procedure called `_is_equal`. Before you go to the procedure, remember the line number you would have gone to, had it not been for the procedure. You now need to find the procedure. It can be anywhere in the program. All we know about it is its name. In this example the procedure we want starts at line 100.

Line 100 This marks the start of a procedure. The line of code doesn't actually 'do' anything other than allow BASIC to find the procedure we have called.

Line 110 Print a message.

Line 120 End the procedure. You need to return to where you would have gone had it not been for the procedure.

Line 60 Print a message.

Line 70 Go to line 20

There are few practical differences between using subroutines and using procedures. Essentially they perform the same task but we tend to use procedures by preference because of the advantages to be gained in having private variables and in having code that is easier to follow.

There are no restrictions on the code that a procedure or a subroutine can contain but you must always be careful to leave either type of routine at the end. This comment relates



back to our dislike of the 'GOTO' command. You can get some very interesting 'bugs' in the program by jumping between procedures and subroutines. What's a 'bug' then? A short history lesson will explain.

### Bugs

When computers were still a novelty kept by mad professors in ivory towers, the only way to get information into or out of them was by paper tape. This was literally a long strip of paper with lots of holes punched in it. The holes were placed along the tape so that it meant something to the computers. All the programs were entered into the computers by paper tape and programmers had to have their programs converted onto the tape before the computers could accept them. If an extra hole were punched onto the tape by mistake the program would go wrong. The story goes that if a program suddenly stopped working the way it was meant to work, a bug, of the animal variety, was blamed for having chewed an extra hole in the tape. It must have been a wonderful excuse for the programmers. The name stuck and so nowadays if a program doesn't work as expected, the fault is called a bug.

Let us go back to creating bugs with procedures. Consider this example:

```
1000 DEF PROC_first_routine
  ..
  ..
lots of program
  ..
1460 IF total = 0 THEN GOTO 2570
  ..
  .. some more program
  ..
2470 ENDPROC
2500 DEF PROC_another_routine
  ..
  .. program again
  ..
2570 REM a valid line in this procedure
  ..
2620 ENDPROC
```

We have only shown a small part of the code and clearly the programmer intended that line 1460 should have jumped to line 2470. But instead the program will jump into another procedure. This is an easy mistake to make, especially if all the intervening lines of code had been present. Remember that on the Notepad you can only see eight lines of code at a time. The result of this 'bug' is that BASIC would get very confused about where it was supposed to be. The program would not work as expected. It is essential

that you leave a procedure exactly where you are supposed to.

## Why Bother With Procedures?

There are lots of reasons. Here are the ones we feel are of most benefit:

- 1) The program is easier to read. You can follow the main flow of the program without getting 'bogged down' in the detail.
- 2) You can keep related actions in one place. For example, if you have one procedure that deals with all output to the screen you will know precisely where to look if you want to make screen changes. You also know that you only have to look in one place rather than scattered sections throughout your program.
- 3) The procedures can be written and tested one at a time thus breaking a large programming task into lots of smaller and therefore easier tasks.
- 4) A procedure written for one program can easily be used in another program if it is appropriate.
- 5) The program tends to be shorter. You only write the code to do a task once. If you need to do the same task again, you just call the procedure again.

Have we persuaded you to use procedures yet? We hope so.

## Get Rid of the GOTO's

We have said several times that you should try to avoid the use of the BASIC command 'GOTO' but so far we haven't given you any alternatives. So here they are. They can broadly be described as 'Loop control' commands and have several distinct advantages over the good old 'GOTO'.

- 1) They make your program easier to read (doesn't this reason turn up a lot - you will soon find that it is the most important reason for most of the programming techniques used).
- 2) They are line number independent. This means that you can move them without causing any problems elsewhere in the program since they stay in the same logical place in the program.
- 3) They make changing the program easier because the 'loop control' statements are obvious and self-contained.

There are two commands; FOR..NEXT and REPEAT..UNTIL. Usually either command could be used to perform a given task but generally they are used to do different jobs. FOR..NEXT is best used if you want to go round a loop a specific number of times that you know at the start of the loop. REPEAT..UNTIL is best used if you want to keep going round a loop until something happens but you don't necessarily know when it will happen.

### FOR..NEXT

The 'FOR' command is a way of telling BASIC to start counting something for you automatically. It marks the start of a loop of program code that you are telling BASIC is going to repeat several times. You also have to tell BASIC the name of a variable to use to keep the count, when to stop counting and how much to add or subtract to the count each time round the loop.

So BASIC interprets this line:

```
FOR loop_count = 1 TO 100 STEP 1
```

as:

Remember where you are now. Set the variable 'loop\_count' to 1 and run the program from here on. Next time you are back at this point increase the contents of 'loop\_count' by '1' and see if it is greater than '100'. If it isn't, run the program from here again. If 'loop\_count' is greater than '100' you have finished this loop.

We can choose any variable name we like; we can choose any number to start at, any number to finish at and any number to step up by.

The 'NEXT' command tells BASIC where the loop of program ends. BASIC interprets the command as:

Go back to the line number we told you to remember and action the command you find there.

```
10 REM Using a for next loop
20 FOR silly_name = 1 TO 100 STEP 1
30 PRINT "The current value of silly_name is ";
40 PRINT silly_name
50 NEXT silly_name
60 END
```

Line 20 starts off the FOR loop. It tells BASIC to use a variable 'silly\_name', to give the

variable the value of '1' and to start going round a loop until 'silly\_name' reaches the upper limit of '100'. Each time round, the loop BASIC will add '1' to the contents of 'silly\_name'.

Lines 30 and 40 are the 'body' of the loop. They are what do the work. In our case they just print the value of 'silly\_name' but it could be any number of lines of program doing any task that you want performing one hundred times.

Line 50 is where the loop ends. This tells BASIC to go back to the start of the program loop. BASIC returns to line 20, adds '1' to the value of 'silly\_name' and tests to see if 'silly\_name' is greater than '100'. If 'silly\_name' is less than or equal to '100' BASIC runs the body of the loop again. When 'silly\_name' reaches '101' BASIC finishes the loop - it goes to the first line of code after the NEXT command. In our case this will be line 60.

BASIC emphasises the nature of the loop by adding extra spaces to the body of the loop. If you LIST the example on your Notepad the screen will look like this:

```
10 REM Using a for next loop
20 FOR silly_name = 1 TO 100 STEP 1
30   PRINT "The current value of silly_name is ";
40   PRINT silly_name
50 NEXT silly_name
60 END
```

The FOR..NEXT command is not limited to counting in ones. In fact it isn't even limited to counting upwards.

```
10 REM Using a for next loop
20 FOR count = 100 TO 1 STEP -2.5
30 PRINT "The current value of count is ";
40 PRINT count
50 NEXT count
60 END
```

In this case we have started the count at '100' and told BASIC that each STEP is '-2.5'. That means that BASIC has to take '2.5' away from count on each step. The first time through the body of the loop BASIC will print '100'. The next time through it will print '97.5' and so on down to '2.5'. At this point BASIC takes '2.5' away from 'count' and gets '0'. Since it is counting down and since the new value of 'count' is below the value we have told it to stop at, BASIC jumps to the first line after NEXT.

If the loop is counting up, it is stopped when the controlling variable, 'count' in this case, is higher than the upper limit contained in the FOR command.

If the loop is counting down, it is stopped when the controlling variable is lower than the lower limit contained in the FOR command.

You need to make sure that the loop will finish when you expect it to. The following program would only run the body of the loop once:

```
10 REM Using a for next loop
20 FOR count = 100 TO 1 STEP 1
30 PRINT "The current value of count is ";
40 PRINT count
50 NEXT count
60 END
```

We have set up a FOR..NEXT loop but it is going to count in the wrong direction. We have started the loop with the variable 'count' holding the number '100'. We have told BASIC to continue going round the loop until 'count' reaches the number '1' but we have told it to 'add 1' each time round the loop. Left alone, the loop would never finish. However, BASIC would notice this error and end the loop immediately on reaching line 50. Therefore, the body of the loop would only be run once.

BASIC will make one assumption about a FOR..NEXT loop if we let it. The assumption is that STEP is '1'. So we don't need to add this to the line unless we want STEP to be something other than '1'. It is good practice to always put a value for STEP because it is then clear what we meant. The following two lines of code would do the same thing because of the assumption that BASIC makes:

```
20 FOR count = 18 TO 27 STEP 1

20 FOR count = 18 TO 27
```

### Getting Out Too Soon

We might not always want a FOR..NEXT loop to run its full course, but we can't just jump out of the loop. That would confuse both BASIC and anyone reading the program. We have to finish off the loop correctly. The surest way of doing this is simply to set the control variable to a number higher than the upper limit.

```
10 REM Leaving a for next loop
20 FOR count = 1 TO 100 STEP 1
30 INPUT "Enter a number ", number
40 PRINT count, number
50 IF number < count THEN count = 101
60 NEXT count
70 END
```

This example will ask us to type in a number, print the current value of 'count' and the number we typed in and then loop back and ask for another number. In fact as long as we type in a number that is higher than or equal to the current value of 'count', BASIC will keep asking for numbers until it has looped for one hundred times.

Line 50 provides us with an early escape route. If the number we type in is greater than the current value of 'count', BASIC will store the value of '101' in 'count'. Then when the next STEP is taken, in this case adding '1', the answer will be greater than the upper limit contained in the FOR command. BASIC will finish the loop and run line 70.

The above scheme would work and is a perfectly valid way of achieving a loop. Its only real disadvantage is that it may not be obvious what is causing the loop to finish. In a real program the line setting the loop control variable to the upper limit may be some distance from the FOR or NEXT commands, so at first sight it is not clear what conditions stop the loop.

In the circumstances where you want a loop to continue until a particular condition is met it is much clearer to use the REPEAT..UNTIL command.

## REPEAT..UNTIL

The 'REPEAT' command tells BASIC that we are going to repeat a loop of program code several times. Unlike the 'FOR' command it does not tell BASIC anything about any variables to use, when to stop the loop or what to do each time round the loop. We have to do all that ourselves.

```
10 REM Using repeat until
20 count = 0
30 REPEAT
40 count = count + 1
50 INPUT "Enter a number ",number
60 PRINT count,number
70 UNTIL count = 100 OR number < count
80 END
```

Line 20 sets a variable 'count' to the value '0'. We have previously written this as 'LET count = 0', but assigning a value to a variable is such a common command that BASIC assumes the presence of the word 'LET'. Thus, the following two lines of code would do the same thing; that is they assign the value of zero to the variable 'count':

```
20 LET count = 0

20 count = 0
```

Most programmers don't bother adding the command 'LET' since there is never any confusion possible about what the line of code is doing.

Line 30 is telling BASIC that it has to REPEAT something but it doesn't say how often or what to do each time round the loop.

The body of the loop is pretty much as before except that we now have to add the STEP ourselves; remember REPEAT..UNTIL doesn't do it for us.

Line 70 tells BASIC when to stop REPEATING. It has to repeat the loop UNTIL either the value in 'count' is 100 'OR' the value in 'number' is less than the value in 'count'. We will discuss the 'OR' operator in a moment. The main advantage that REPEAT..UNTIL has in this example is that the conditions for finishing the loop are both in the same place. It is easy to see why the loop has finished. It must have stopped because either the value in 'count' was '100' or because the value in 'number' was less than the value in 'count'. We don't have to search through the program to find any other conditions.

If you list this example on your Notepad you will find that BASIC indents the body of the loop again in just the same way as it did for the FOR..NEXT loop.

```
10 REM Using repeat until
20 count = 0
30 REPEAT
40   count = count + 1
50   INPUT "Enter a number ",number
60   PRINT count,number
70 UNTIL count = 100 OR number < count
80 END
```

### Loops Within Loops

Loops can have other loops within them. This is called 'nesting' since one loop is nested inside another. The only point to be aware of is that you must not mix up the different levels of nesting. Consider this example:

```
10 REM Nested loops gone wrong
20 FOR outer_loop = 1 TO 100 STEP 10
30 FOR inner_loop = 1 TO 10 STEP 1
40 PRINT outer_loop,inner_loop
50 NEXT outer_loop
60 NEXT inner_loop
70 END
```

In line 50 we have ended the body of the outer loop before we have ended the body of the inner loop. If you LIST the example in BASIC the program structure will become more obvious:

```
10 REM Nested loops gone wrong
20 FOR outer_loop = 1 TO 100 STEP 10
30   FOR inner_loop = 1 TO 10 STEP 1
40     PRINT outer_loop,inner_loop
50   NEXT outer_loop
60 NEXT inner_loop
70 END
```

The outer loop should consist of lines 20 to 60; the inner loop should be lines 30 to 50. But we have instructed BASIC to take the next step in the outer loop while BASIC still thinks it is in the inner loop.

```
10 REM Nested loops corrected
20 FOR outer_loop = 1 TO 100 STEP 10
30   FOR inner_loop = 1 TO 10 STEP 1
40     PRINT outer_loop,inner_loop
50   NEXT inner_loop
60 NEXT outer_loop
70 END
```

The REPEAT..UNTIL example above used an operator we haven't covered yet. It is one of a group of operators known as the 'Logical Operators'.

## Logical Operators

BASIC's set of logical operators can be used in two very different ways. We are only interested in one of these, the use of logical operators to combine several tests to produce a single answer.

All of the logical operators have 'English' like names and behave very much as one would expect from a strict interpretation of the English word.

### AND

Suppose we were conducting a survey of examination results from a group of children. We may want to work out the percentage of children that achieved a 'grade 1' in both the Maths and the English examinations. We are combining the results of two tests to produce a single result. We are only interested in those children that passed in Maths AND passed in English. The following BASIC line would combine the results:



```
1000 IF maths_mark = 1 AND english_mark = 1 THEN PRINT "Well done"
```

The test that BASIC performs in line 1000 is effectively in two parts. First it checks to see if the variable 'maths\_mark' contains the number '1', then it checks to see if the variable 'english\_mark' contains the number '1'. Only if both tests are TRUE is 'Well done' printed. We can extend the use of AND by adding more tests.

Suppose we now want to know if any children did well in Maths, English and French.

```
1000 IF maths_mark = 1 AND english_mark = 1 AND french_mark = 1 THEN  
PRINT "Very well done"
```

This time, just getting a pass mark in Maths and English is not good enough; the test requires a pass mark in Maths AND English AND French.

### OR

A sequence of tests joined by the OR logical operator is much easier to pass. Now we are just looking for one of the results to be TRUE. If we wanted to see the children who were good at either Maths OR English, the following line would show them up:

```
1100 IF maths_mark = 1 OR english_mark = 1 THEN PRINT "Pretty good"
```

This time BASIC will print our message if either of the examination results were 'grade 1'. Again we can extend the test by adding more conditions:

```
1150 IF maths_mark = 1 OR english_mark = 1 OR french_mark = 1 THEN  
PRINT "Could do better"
```

Here the child only needs to get a pass grade in one subject to receive a comment from the program.

Note that extra passes are not excluded so a child getting any of the following results would receive the comment:

- grade 1 in English, some other grades in French and Maths
- grade 1 in Maths, some other grades in English and French
- grade 1 in French, some other grades in Maths and English
- grade 1 in English AND Maths, some other grade in French
- grade 1 in English AND French, some other grade in Maths
- grade 1 in Maths AND French, some other grade in English
- grade 1 in Maths AND English AND French

The OR operator looks for one or more TRUE tests to be satisfied.

## EOR

This is short for 'Exclusive OR'. It is very similar to the logical OR operator except that only one test is allowed to be TRUE.

Consider the following two lines:

```
1100 IF maths_mark = 1 OR english_mark = 1 THEN PRINT "Pretty good"
1200 IF maths_mark = 1 EOR english_mark = 1 THEN PRINT "Pretty good"
```

As we have seen above, in line 1100 we would get a message printed if either of the two tests proved to be true or if both of the tests proved to be true. In line 1200 however, we only get a message printed in the case where one of the two tests is true. We won't get a message if they are both true. The 'E'xclusive 'OR' operator has excluded the case when both tests are true.

When we extend the principle it becomes rather more difficult to analyse the results. This is because the EOR operator can only work on two tests at a time. If there are more than two tests linked by EOR, the tests are broken down into pairs working from the left. The easiest way to illustrate this is to work through some examples. We can change line 1200 to read:

```
1200 IF maths_mark = 1 EOR english_mark = 1 EOR french_mark = 1 THEN
PRINT "Could do better"
```

Suppose a student obtained grade '1' in Maths and grade '2' in English and French. The sequence of decisions that BASIC would take when processing line 1200 would be:

Do the first test: Is maths\_mark = 1?

Result is TRUE

Do the next test: Is english\_mark = 1?

Result is FALSE

Since we now have two results combine them to produce a single result using EOR:

TRUE EOR FALSE

Result is TRUE

Do the next test: Is french\_mark = 1? Result is FALSE

Since we now have two results combine them to produce a single result using EOR:

TRUE EOR FALSE

Result is TRUE

Here the result, TRUE, is as expected and would have been the same had we used OR instead of EOR.

What if the student obtained grade '1' in Maths and English and grade '2' in French?

Do the first test: Is <code>maths_mark = 1</code> ?	Result is TRUE
Do the next test: Is <code>english_mark = 1</code> ?	Result is TRUE
Since we now have two results combine them to produce a single result using EOR: TRUE EOR TRUE	Result is FALSE
Do the next test: Is <code>french_mark = 1</code> ?	Result is FALSE
Since we now have two results combine them to produce a single result using EOR: FALSE EOR FALSE	Result is FALSE

This is exactly what we want. We are trying to catch those students that only obtained one grade '1'. So the line of BASIC has correctly analysed the results for us when this student obtained two grade '1' marks.

But there is a drawback. Look at what happens if the student obtains three grade '1' marks.

Do the first test: Is <code>maths_mark = 1</code> ?	Result is TRUE
Do the next test: Is <code>english_mark = 1</code> ?	Result is TRUE
Since we now have two results combine them to produce a single result using EOR: TRUE EOR TRUE	Result is FALSE
Do the next test: Is <code>french_mark = 1</code> ?	Result is TRUE
Since we now have two results combine them to produce a single result using EOR: FALSE EOR TRUE	Result is TRUE

This is not the answer we want. If a student obtained grade '1' for Maths, English and French, line 1200 would print "Could do better".

Unfortunately, if there are an odd number of tests joined by EOR and all the tests are TRUE, the answer will always be wrong. However this is not a serious problem since we already know how to check for all the tests being TRUE by using the AND operator. Hence as long as we remember the problem we can always get round it.

If the child obtained grade 1 in both Maths and English, the use of EOR would exclude BASIC from printing the comment.

The use of AND and OR can be extended to combine many tests in the same line. This is not true for EOR. The results are straightforward if you combine two tests with EOR. We have just seen how to combine three tests by making sure that the case of all the tests being TRUE was trapped first. But you cannot combine more than three tests with EOR. Unfortunately the results from combining four or more tests are inconsistent - if there are an odd number of TRUE answers EOR will always give the wrong result. However, as long as we stay within the known limitations of the command, the results will always be as expected.

The EOR operator looks for one, and only one, TRUE test to be satisfied (but remember to check for all TRUE first and don't combine more than three tests).

## NOT

The logical operator NOT simply reverses the action of any of the previous operators. Consider the following two lines of code:

```
1150 IF maths_mark = 1 OR english_mark = 1 OR french_mark = 1
      THEN PRINT "Could do better"
1160 IF NOT (maths_mark = 1 OR english_mark = 1 OR
      french_mark = 1) THEN PRINT "No grade 1's!"
```

As we saw above, line 1150 would print "Could do better" if a child received a grade 1 in Maths OR English OR French. Line 1160 reverses the test and so would print the unkind remark if the child didn't get any grade 1's.

The brackets are important. They tell BASIC to work out the result of everything inside the bracket before actioning the NOT operator. If we took the brackets away, the NOT operator would only affect the result of the first test (Is maths\_mark = 1?).

In this example it would have been more efficient to have added an ELSE to the first IF command in line 1100.

## Logical Operator Summary

If two or more tests in an IF command are joined by the logical operator AND, all the tests must be true before the THEN path is followed.

If two or more tests in an IF command are joined by the logical operator OR, one or more of the tests must be true for the THEN path to be followed.

If two or more tests in an IF command are joined by the logical operator EOR, one and only one of the tests must be true for the THEN path to be followed unless there are an odd number of tests and they are all true.

The NOT logical operator reverses the effect of any of the above.

## To Continue The Program

To illustrate the use of the logical commands we will complete the program to grade the children's results.

## Chapter 21 Basic

We want the program to print one of four possible messages depending on the results of the grades. The messages are as follow:

No grade 1's should print "No grade 1's!"  
One grade 1 should print "Could do better"  
Two grade 1's should print "Pretty good"  
Three grade 1's should print "Very well done"

We can split the program up into two sections: input and output. The input section has to ask for the three grades and store them somewhere. The output section has to decide which message to print and then print it. Clearly only one message should be printed for each child. Let's start with the main flow of the program.

```
10 REM Grading program
15 REM set a flag to go round the loop until stopped
20 the_end = FALSE
30 REPEAT
40   PROC_get_grades
50   PROC_print_message
55   REM have we been stopped yet?
60 UNTIL the_end
70 END
```

We have set up a straightforward loop to act as our main program. We have used the variable 'the\_end' to control a REPEAT..UNTIL command. As long as this variable is FALSE BASIC will continue to go round our loop. Within the REPEAT..UNTIL loop we have our two sections - one to deal with input (PROC\_get\_grades) and one to deal with output (PROC\_print\_message). All we have effectively done is to split our complicated program into two smaller and less complicated sections. We can deal with each section in turn, making life much easier.

```
100 DEF PROC_get_grades
105 REM ask for the grades - give a reminder how to stop
110 INPUT "English mark (0 to end) ",english_mark
115 REM test for us to stop program
120 IF english_mark = 0 THEN the_end = TRUE:ENDPROC
130 INPUT "Maths mark           ",maths_mark
140 INPUT "French mark          ",french_mark
150 ENDPROC
```

This procedure has two jobs to perform. The obvious job is to ask the user to enter the grades ready for processing. The less obvious job is to allow the user to end the program if he so wishes. We have taken the arbitrary decision that a grade of '0' for English will cause the program to stop. Line 110 tells the user of this fact and line 120 checks to see if

the user does want to stop. If the user entered '0' then this procedure must finish, making sure that the main program knows the user wants to stop. Remember that the main program is using the variable 'the\_end' to control the REPEAT..UNTIL loop so if we set this variable to TRUE the main program will stop.

If the user entered a valid grade for English, we carry on to ask for the grades for Maths and French then end the procedure.

```

200 DEF PROC_print_message
205 REM make sure we don't print if we have been stopped
210 IF the_end THEN ENDPROC
215 REM check for all three grade 1's
220 IF english_mark = 1 AND maths_mark = 1 AND french_mark = 1
      THEN PROC_great:ENDPROC
225 REM check for a single grade 1
230 IF english_mark = 1 EOR maths_mark = 1 EOR french_mark = 1
      THEN PROC_do_better:ENDPROC
235 REM check for a pair of grade 1's
240 IF english_mark = 1 OR maths_mark = 1 OR french_mark = 1
      THEN PROC_good:ENDPROC
250 PROC_poor
260 ENDPROC

```

The job of this procedure is to print different messages depending on how good the grades were. As usual we have split the job into two parts - that of deciding which message to print and that of printing the message.

The decision making process uses the logical operators to control which of the printing procedures are called. The logical flow of the decision can be summarised as follows:

Check to see if all the grades were '1' and if so call the procedure `_great` and finish (line 220).

Check to see if there was a single grade '1' and if so call the procedure `_do_better` and finish (line 230). Note that we can safely use the EOR operator here since the previous line trapped the only condition that would cause EOR to go wrong.

When we get to here, we have removed students with three grade 1's and we have removed students with a single grade 1. Thus if any mark grade is a 1 there must be two grade 1's. If so call procedure `_good` and finish (line 240).

If the program reaches this point it must mean that there were no grade '1' marks so call the procedure `_poor` and finish.

## Chapter 21 Basic

The variable 'the\_end' was set to TRUE by the procedure `_get_grades` if the user entered '0' as the grade for English. Thus line 210 stops our procedure `_print_message` from printing a message if the user has signalled the program to stop.

```
300 DEF PROC_poor
310 PRINT "No grade 1's!"
320 ENDPROC

400 DEF PROC_do_better
410 PRINT "Could do better"
420 ENDPROC

500 DEF PROC_good
510 PRINT "Pretty good"
520 ENDPROC

600 DEF PROC_great
610 PRINT "Very well done"
620 ENDPROC
```

The four procedures to actually print out the right message are very simple. So why bother with separate procedures anyway? In this example there isn't really much point but it does give us added flexibility.

Suppose later we wanted to collect statistics on the number of students obtaining each message. We would have to add variables to count the number of times each message was printed and how many students' results were entered. That would be easy to do in our example because it is very structured and thus expandable. We haven't limited the job the program can do just because at the moment the job is easy.

There are two new points to note in the above example. The first one is the use of the colon ":" character on lines 120, 220, 230 and 240.

So far in all our examples we have only ever had one command on a line, but this need not be the case. BASIC only needs the line numbers in a program to figure out the order in which to execute the commands. If two commands will always be executed one after the other we can place them on the same line separated by a colon. We can extend this principle even further. Indeed the only limit on the length of a line is 255 characters, so the following two pieces of program code are identical:

```
10 A = 1
20 B = 2
30 C = A + B
40 PRINT C
50 C = C + B
```

```
60 PRINT C
10 A = 1:B = 2:C = A + B:PRINT C:C = C + B:PRINT C
```

In the above case it makes very little difference which style we choose to write the code in. It is often convenient to keep several related commands on a single line to emphasise that they are related but there is little real benefit to code clarity. However the real power of the colon comes when used with an IF command.

```
120 IF english_mark = 0 THEN the_end = TRUE:ENDPROC
```

BASIC interprets this line as follows:

Look at the contents of the variable 'english\_mark' and see if it equals '0'. If it doesn't, go on to the next line. If it does then set the variable 'the\_end' to TRUE and then end the procedure.

In other words, we have linked two commands to be performed in the case where the IF command is true. Both the setting of the variable and ENDPROC depend on the IF command.

As always we can extend the concept:

```
1000 IF this_count = 100 THEN PRINT "all done":end_flag = TRUE
      :sum = sum + 1:PROC_clear_up:ENDPROC
```

In this case, if the value of 'this\_count' is equal to '100' we will print a message, set a flag, add one to another variable, do a procedure and then end this procedure. All these commands rely on the IF command. None of these commands will be processed if the IF test is FALSE.

The second new idea that we have introduced in the grading program above is the use of flags. Flags are just variables. There is nothing special about them but they do make for clear programs. In our case we have a variable called 'the\_end' and we start by giving it the value of FALSE. At the end of each loop of the main program, this variable is tested. If its value is still FALSE, the program loops back to ask for more grades. If its value is TRUE, the program stops.

We set the variable to TRUE in line 120 if the user has entered a grade of '0'.

There are two main advantages of using variables as flags. First, as you no doubt expect by now, is program clarity. If your main program loop keeps repeating until a variable called something like 'the\_end' is set to TRUE, it is rather obvious what stops the



program. Similarly, if several procedures can only be run after some initialisation has been done, testing a flag called 'init\_done' would highlight that point precisely.

The second advantage of flags is that you can control different parts of the program very methodically. Suppose you had a program that wrote information into the memory of the Notepad. You would need to ensure that there was still enough room in the memory to carry on. A flag called something like 'mem\_full' could signal a problem to lots of different procedures throughout the program if you ran out of memory. These other procedures may not have anything directly to do with memory use, but they may need to know of a problem. The flag would be controlled by the procedure that was writing to memory and would be read by any other procedure that needed to know.

## The Three Modes of BASIC

Officially BASIC has three very different operating modes. We haven't said which mode BASIC has been in at any given time because this is only important to BASIC - we don't need to know as long as we realise what BASIC will do next.

The three modes are called 'Immediate', 'Program' and 'Edit'.

### Why Three Modes?

Essentially the three modes are just a way of telling BASIC what to do next. In Immediate mode the next command always comes to BASIC from the keyboard. In Program mode the next command always comes from the next line of the program that BASIC is currently running. In Edit mode the next command comes from the keyboard but BASIC has to save what you have edited first.

You never need to think in terms of switching modes. You will just type 'RUN' or 'EDIT' or press the STOP key; you will know exactly what BASIC is going to do next and BASIC will do as it's told (well most of the time anyway).

### Immediate Mode

When you first enter BASIC by pressing Function B it will be in the Immediate mode. You can tell this because BASIC puts a '>' on the screen waiting for your instructions. This is the mode where you do things like SAVE a file, LOAD a file or type in the lines of your program.

BASIC will always respond 'Immediately' to your commands.

## Program Mode

Once you have either typed in or 'LOAD'ed a program you can tell BASIC to 'RUN' the program. You are actually telling BASIC to switch to its Program Mode.

This is where BASIC runs your program. Your program has complete control over what is printed on the screen and what keys to expect from the keyboard.

## Edit Mode

If you type in a line of your program and make a typing mistake you can correct the mistake by just re-typing the line. For a short line this is probably the quickest way of correcting it. If, however the line of code is very long you can tell BASIC to switch to its 'Edit Mode' to allow you to correct the error.

## Editing Your Program

To edit a line you just type 'EDIT' followed by the number of the line you want to edit. If you want to change something in line 40, you just type EDIT 40 and BASIC will present the line for you to change. Although EDIT is not a 'full-blown' text editor like your Notepad's Word Processor it will allow you to quickly correct the odd typing mistake.

Once you have told BASIC which line to edit, it displays the line at the top of the screen and places the cursor before the start of the line. As you type characters they are inserted where the cursor is. The following keys move the cursor and/or delete characters:

Del > - deletes the character on which the cursor rests closing up the rest of the line

< Del - deletes the character immediately to the left of the cursor, closing up the rest of the line

Right Cursor - moves the cursor one character to the right, stopping after the last character on the line

Left Cursor - moves the cursor one character to the left, stopping at the start of the line

ENTER - finishes editing this line and returns BASIC to its immediate mode

STOP - this leaves the editor without putting any of the changes you have typed

into your program.

As well as changing your program a line at a time EDIT is also a useful way of copying lines of code. For example, if you EDIT 200 and just change its line number to 210 you will find that you now have two identical lines of program, one numbered 200 and the other numbered 210.

If you want to remove a line from your program you just type its line number followed by the ENTER key.

### Switching modes

You don't really need to think about it. The rules for switching modes are straightforward and will soon become automatic. However, for the record, here they are:

When you enter BASIC you will be in Immediate Mode.

To switch to Edit Mode you type 'EDIT' followed by the line number you want to edit.

To switch back to Immediate mode without changing anything on the line you're editing, press the STOP key. This ignores any changes you may have made to the line.

To switch back to Immediate mode keeping any changes you've made to the line you're editing, press the ENTER key.

To switch to Program Mode you type 'RUN'.

To switch back to Immediate Mode you press the STOP key twice. This just stops the program running wherever it happens to be.

BASIC will also stop running a program if it finds a line of program code that it cannot understand. It will then display a message telling you which line number had the error.

---

#### **WARNING - The Panic Button**

You need to be pretty careful about pressing the Function key by mistake. This could well be described as the 'Panic escape route'. It will stop any program running and the Notepad will go off to the application that you directed it to, e.g. pressing Function C will take you to the Calendar. But in going to another application the Notepad will throw away any changes you made to the current BASIC program. It is a quick way out but beware! If you have made any changes to your program and not saved them, the changes will be lost.

---

## TIP - A Sneak Preview of BASIC Commands

Apart from actually typing in your program, the most useful aspect of the Immediate mode is that you can see what BASIC will do. Many commands that are available to a BASIC program can be entered in the immediate mode and will get an immediate response.

If you type `PRINT 3+4` without a line number BASIC will immediately print the answer, hence the name immediate mode. You can print out the current values of variables. So for example if you ran the following program:

```
10 REM What does a variable hold
20 first = 34
30 second = 65
40 total = first + second
50 END
```

and then type `'PRINT first'`, BASIC would respond by printing 34 on the screen. Although the program has stopped and BASIC is in Immediate Mode, it hasn't forgotten what was in each variable. So the variable `'second'` still holds the number `'65'` and the variable `'total'` stills holds the number `'99'`.

The variables will only lose their contents when either you `'RUN'` the program again or you `'LOAD'` in a new program.

This is particularly useful if you are trying to remove bugs from a program. If you run the program, press the STOP key twice when the program goes wrong and then print the values of variables, it can give you a clue as to the cause of the problem.

---

## Moving On

We have covered enough of the commands available in BASIC for you to be able to start writing useful programs. So that is precisely what we are going to do next. We will continue to add new commands as we go through the example but in the next section we will be concentrating on the mechanics of putting together a program to do a specific job.

## The Hotel Reception Desk

Suppose you ran a hotel. The hotel has four floors and on each floor there are twenty rooms. Behind the reception desk are a set of shelves where you keep messages for the guests. There are four shelves and each shelf is divided into boxes. Each box is labelled with a room number, from one to twenty, reading from left to right. The bottom shelf is

labelled 'First floor' and the top shelf is labelled 'Fourth floor'.

When somebody wants to leave a message for one of your guests, you write the message on a sheet of paper and place it in the box corresponding to the room the guest is staying in.

If the guest from room 15 on the second floor asks for any messages, you can look along the second shelf to box 15 and take out anything you find for him.

We want to replace the shelves in our hotel with a computer system that will store messages for guests. The program must allow us to input a message for a particular guest, store that message until we ask for it and print out the message to the screen when we tell it to.

Before we start to write the program we need to plan what it will do and broadly how it will work. As programs become more complex, the planning stages become more important.

### **The Program Specification**

For a commercial program the Program Specification is a detailed document that describes what the program will do. It details what input information it will collect, how the information is stored and what output the program will generate. It will frequently include detailed descriptions of how the various parts of the program will operate and how they will interact with the other parts of the program. It will always include any assumptions that have been made about the information it must handle or the limits on the functionality required.

We don't need all that in a formal document, but it is important to think about what you want the program to do before you start writing it. You don't even have to write the specification down, but it would help to get things clear in your own mind if you wrote a few lines describing the purpose of the program.

If you don't plan, you will probably end up with a program that will

- 1) be difficult to get working properly because you have had to 'tack on' little bits here and there as you've thought of them
- 2) probably not do what you wanted it to because you have run out of time and/or patience to finish it
- 3) be difficult to change or adapt to a new use

So, to our specification.

We have already stated our principal objective - to replace an existing manual hotel message system. Before we consider how the program will operate we need to consider the assumptions we will make and what information will be available to us.

First, we will always know the room number for the guest; secondly the information can be thrown away once it has been shown once; thirdly that there will only ever be a maximum of one message per guest at once and that the message will be short; and lastly that this is the only use for the computer. These assumptions are pretty restrictive in the real world, particularly the last one, but they will help us to get started quickly. The assumptions can be modified later to extend the usefulness of the program.

## Two Dimensional Arrays

We haven't yet said exactly how we are going to store the messages. We know that there are four floors and on each floor there are twenty rooms so we know that there will be up to eighty messages to store. There are several possible ways of storing the information.

We could have eighty different variables, each with a different name, and then decide where to save the information by having a long list of 'IF..THEN' commands. This would work but it isn't really practical for the amount of data we have to deal with.

We have already covered the concept of arrays and they would seem a good candidate for use here. Remember that arrays are a way of pointing to one in a line of boxes by simply telling BASIC how far along the line the box is. We have eighty messages to store so we could have an array with eighty boxes. This would allow us to store all the information we need but it wouldn't be very convenient. Where would we put a message for the guest in room 18 on the 4th floor? The answer is in box 78 but it isn't immediately obvious. We want to design our program so that everything is obvious wherever possible.

If we think back to our existing manual system for saving the messages we already have a very efficient and obvious way of deciding where to store the information. We have four rows of twenty boxes corresponding to the four floors of twenty rooms. To find a message for the guest in room 18 on the fourth floor we are using two pieces of information. The floor number points at a row of boxes; the room number points at a particular box in the row. Our information has been stored in two dimensions - the vertical dimension determines which floor we need and the horizontal dimension determines which room we need.

BASIC can copy this format exactly with a 'two-dimensional' array. The command:

## Chapter 21 Basic

`DIM message (4, 20)`

reserves enough space for four rows of twenty boxes. If we want to look at the message for our guest in room 18 on the fourth floor we just have to point BASIC to the right place:

`PRINT message (4, 18)`

We are effectively saying:

Print the contents of the eighteenth box in row four.

For our example a two-dimensional array containing 4 rows of 20 boxes is the ideal way to store the information. We do need to make one slight change to cope with the fact that the messages are plain text. The statement above:

`DIM message (4, 20)`

would reserve enough space for our 80 boxes (in 4 rows of 20) but we have told BASIC that each box will contain a real variable (there is no % or \$ after the name). A real variable can only store numbers; it can't store text. Text is stored in strings so we need to tell BASIC that our array will hold strings:

`DIM message$(4, 20)`

Now we can think about how the program will operate.

The program is really in two halves. We need a way of collecting the messages and storing them for later retrieval and we need a way of retrieving the messages and printing them on the screen. Apart from the fact that both halves of the program are working with the same information, the messages, they are completely separate. We can write and test them as individual programs and then combine them at the end when they are both working as we wish. Sections of a large program that can be written on their own like this are called 'program modules'. A program module may be a single procedure or a subroutine. However, it normally consists of a group of several procedures that together perform a clearly defined task.

We are going to have an 'information input module' and an 'information output module'. The two modules will together form our program.

We can now write down the essentials of how our program will operate:

- 1) Decide what to do, either input messages or output messages.

2) If we are inputting messages, get the floor and room numbers followed by the message and store the message.

3) If we are outputting, get the floor and room numbers, print the message and then clear the message.

Now we can start writing our program; we know what it has to do and roughly how it is going to do it.

## **How do you Write a Large Program?**

There are as many answers to this as there are programmers but we can give you a few guidelines.

1) Break the task up into lots of smaller tasks. If you try to do everything at once you will soon get hopelessly lost in the complexity of the program.

2) Keep sections of the program dealing with the same thing in the same place. It makes it easier to find a bug and to alter the program in future.

3) Don't worry about the detail until you have got the main outline sorted out.

4) Test each subroutine or procedure thoroughly before you use it in the main program.

5) Assume the user of your program is stupid and that he wants to 'break' your program by typing the wrong things in. In practice neither of these assumptions are often valid but if you think that they are while you are programming your program will work far more reliably when finished.

6) Avoid writing long stretches of program. Keep each section short, if possible short enough to read the whole procedure in one screen.

7) Wherever possible make sure that if a procedure is designed to collect some information, the same procedure is responsible for checking that the information is correct. By trapping invalid input as soon as possible, with the procedure that collects it, the general reliability of the program is greatly improved.

First we need to decide what our main program has to do. The obvious task is to decide whether to input a message or to output a message but we will also need to set up some variables before running the program. This 'setting up' is called initialisation and some form of initialisation procedure will be found in almost all programs.



## Chapter 21 Basic

We have enough information to write our program.

```
10 REM Hotel Reception Desk Message Saver Version 1
20 PROC_welcome_screen
30 PROC_initialise
40 the_end = FALSE
50 REPEAT
60   PROC_get_decision
70   IF decision = 1 THEN PROC_do_input ELSE PROC_do_output
80 UNTIL the_end
90 END
```

There it is. Wasn't that easy. That is our main program written. It doesn't do anything yet but it is the complete overall structure.

All we have done is to break to complex task down into five easier tasks, these being:

PROC\_welcome\_screen - say something helpful on the screen

PROC\_initialise - set up any variables that we need in the program

PROC\_get\_decision - find out what the user wants to do

PROC\_do\_input - input and store a message

PROC\_do\_output - print and clear a message

We have created a program loop between lines 50 and 80 with a 'REPEAT..UNTIL' command. In fact the loop will never end because in line 40 we set the controlling variable 'the\_end' to FALSE and we never set it to TRUE. The 'UNTIL' command in line 80 tests the value of 'the\_end' against TRUE so the program will always loop back to line 40.

This is an unusual way to control a program but it has the advantage of making our first attempt very straightforward because we don't have to worry about re-starting the program. In a normal application we would want to be able to stop this program, go off and do something else with the computer and then come back and carry on with the messaging system. This requirement causes several complications that we will cover later. So for the moment we will assume that the Notepad is dedicated to the task of storing and retrieving messages.

The loop of program code between lines 40 and 80 only has to do two things. It needs to collect a decision from the user, which we will store in the variable 'decision', and it needs to either call a procedure to input a message or to call a procedure to output a message.

Effectively we now have to write five small programs to act as the procedures for our main program and we have already stated what each program must do.

## PROC\_welcome\_screen

When somebody starts running a program they expect to see something happen immediately. If the program has to search for documents and then prepare them for use, or if there are lots of variables to initialise before use, it can take several seconds before the program is ready to do useful work. Therefore it is always a good idea to put something on the screen straight-away so that the user knows the program is running and what it is doing. In our example there is not very much to do before the real work starts so we don't really need to bother about any delays before the program is ready to use. However, good habits are best formed early so we will write a short procedure to say hello to the user.

```
1000 DEF PROC_welcome_screen
1010 PROC_banner
1040 PRINT TAB(28,2) "VERSION 1"
1060 PRINT TAB(24,4) "please wait....."
1070 ENDPROC
```

```
1100 DEF PROC_banner
1110 CLS
1120 PRINT TAB(15,0) "HOTEL RECEPTION DESK MESSAGING SYSTEM"
1130 ENDPROC
```

We have split this procedure into two parts because we will want to use our banner in several places. It is therefore sensible to have a single routine to print the banner.

We have also introduced two new commands. The first one, 'CLS', simply clears the Notepad's screen. The 'TAB' command however is much more powerful and needs a short explanation.

## TAB

The 'TAB' command allows the program to place the cursor anywhere on the screen. The cursor marks the place where anything printed on the screen will be positioned. It is the same as the cursor in the Notepad Word Processor. Thus 'TAB' provides a means of organising the screen layout to suit the particular application. To tell BASIC where to put the cursor, you need to supply a column number and a line number.

Column numbers run from 0 to 79; line numbers run from 0 to 7. Thus to place the cursor at the top left of the screen you would write:

## Chapter 21 Basic

```
TAB (0, 0)
```

and to place the cursor at the bottom right of the screen you would write:

```
TAB (79, 7)
```

The 'TAB' command can be used to position the cursor prior to either a 'PRINT' command or an 'INPUT' command. For example, line 1120 above is telling BASIC to move the cursor to column 15 on the top line of the screen and then print the banner message.

Once the procedure `_welcome_screen` has finished, the screen will show the following:

```
HOTEL RECEPTION DESK MESSAGING SYSTEM  
  
VERSION 1  
  
please wait.....
```

### PROC\_initialise

In this program we don't have any variables that need setting to special values before the program can run. Hence we don't need to do much initialisation. However we do need to tell BASIC about the array we will be using to store the messages in. This is the purpose of line 2010. We don't need to worry about what is in the array to start with: BASIC takes care of that for us. Whenever an array is 'DIM'ensioned BASIC always sets all the array elements to either zero, for a real or integer array, or an empty string, for a string array.

This routine then just positions the cursor and calls a procedure to ask the user to press the ENTER key.

```
2000 DEF PROC_initialise  
2010 DIM message$(4, 20)  
2020 PRINT TAB(19, 4);  
2030 PROC_get_enter  
2040 ENDPROC
```

We have called another procedure to wait for the user to press the ENTER key. We will cover the definition of the procedure `_get_enter` later.

## PROC\_get\_decision

This procedure has to ask the user what he wants to do. We have used the number 1 to mean input (see the IF statement in the main program). We haven't specified a number to use for output so we will use the number 2. So this procedure has to set the variable 'decision' to the value of 1 or 2 depending on whether the user wants to input or output a message.

Since any other numbers apart from 1 and 2 are invalid, the procedure must reject them.

```

3000 DEF PROC_get_decision
3005 REM use a variable to signal when we have a good decision
3010 got_decision = FALSE
3015 REM sit in a loop until we have a decision
3020 REPEAT
3025   REM clear the screen and put up our heading
3030   PROC_banner
3040   PRINT TAB(12,2)
           "Please enter requirement   Input message - 1"
3050   PRINT TAB(39,3) "Print message - 2 ";
3055   REM get a digit
3060   PROC_get_digit
3065   REM has the user pressed 1 or 2
3070   IF this_digit = 49
           OR this_digit = 50 THEN got_decision = TRUE
3080 UNTIL got_decision
3085 REM the user pressed either 1 or 2 so return the answer
3090 decision = this_digit - 48
3100 ENDPROC

```

Lines 3020 to 3080 form a loop which is only left when the value in the variable 'got\_decision' is TRUE. To begin with we set this variable to be equal to FALSE. It is only set to TRUE when the value of the variable 'this\_digit' is either 49 or 50. This means that if the user presses a key other than '1' or '2' in response to our question, the question will be asked again.

We have called a procedure to get the digit. For the moment we don't care how this is done as long as the procedure returns a number.

There are some magic numbers that need some explanation. Line 3070 does not look as though it's testing for the keys '1' and '2' but it is.

### The ASCII Code

Computers can only deal with numbers. In the real world we have lots of different symbols in addition to just numbers that we use to communicate with. Apart from the letters of the alphabet we have all the punctuation characters, maths symbols etc. So that computers can deal with these different symbols they have to be represented by numbers.

Over the years there have been many different ways in which the symbols have been translated into numbers but by far the most common, and more importantly for us the method used in the Notepad is the code known as ASCII.

ASCII stands for the American Standard Code for Information Interchange. All we need to realise is that it is a standard way of saying what number means the letter 'A', what number means the letter 'B', etc. In fact all of the keys on the Notepad's keyboard are translated into a number before the computer can do anything with them.

When you press the key 'A' with the SHIFT key held down as well, the computer sees the number '65'. When you press 'SHIFT' and 'B' the computer sees the number '66'. For reference we have included a list of the complete ASCII CODE as an appendix but for now all we need to know is that when you press the key '1' on it's own, the computer sees the number 49. So we can translate line 3070 of our program into English as follows:

If the user pressed the Notepad key labelled '1' OR if he pressed the Notepad key labelled '2' then put the value 'TRUE' into the variable 'got\_decision'.

```
3070 IF this_digit = 49
      OR this_digit = 50 THEN got_decision = TRUE
```

Line 3090 sets our variable 'decision' to the value of 'this\_digit' less '48'. Remember that we had decided that we wanted this procedure to set the value of 'this\_digit' to either 1 or 2 depending on which key the user pressed. Since the ASCII code for the key '1' is 49 and the ASCII code for the key '2' is 50, by taking 48 from the ASCII code we have arrived at the required answer.

### PROC\_do\_input

The requirement for this procedure is to ask the user for a room number and a floor number and then to store the message for the appropriate guest.

```
4000 DEF PROC_do_input
4005 REM put up our header again
4010 PROC_banner
```

```

4015 REM make sure the user knows what the program is doing
4020 PRINT TAB(22,2) "READY TO STORE YOUR MESSAGE"
4025 REM ask for the room number and put into the variable 'room'
4030 PROC_get_room
4035 REM ask for the floor and put into the variable 'floor'
4040 PROC_get_floor
4050 PRINT TAB(16,6) "What is the message";
4060 INPUT message$(floor,room)
4065 REM add some reassurance
4070 PRINT TAB(7,7) "The message has been stored - ";
4080 PROC_get_enter
4090 ENDPROC

```

We have defined two new procedures; `_get_room` and `_get_floor`. These must each ask the user for the necessary information and check to make sure that it is in range.

Line 4060 does the job of both collecting the message for a particular guest and of storing the message in the correct array element. Since we don't need to do anything to the message for the guest, all we have to tell BASIC is where to put it.

## PROC\_do\_output

The requirement for this procedure is very similar to that for `PROC_do_input`. We need to ask the user for a room number and a floor number and then print any message on the screen.

```

5000 DEF PROC_do_output
5005 REM put up our header again
5010 PROC_banner
5015 REM make sure the user knows what the program is doing
5020 PRINT TAB(22,2) "CHECKING FOR STORED MESSAGES"
5025 REM ask for the room number and put into the variable 'room'
5030 PROC_get_room
5035 REM ask for the floor and put into the variable 'floor'
5040 PROC_get_floor
5045 REM test for no message for this guest
5050 IF message$(floor,room) = "" THEN PROC_no_message
      ELSE PROC_message
5060 ENDPROC

```

Line 5050 is the only one that does any real work in this procedure. We need to decide whether or not there is a message for this guest. If there is then we need to print it. If there is no message we want to say something helpful on the screen. Line 5050 takes this decision and calls either `PROC_no_message` or `PROC_message` as appropriate.

To understand how the test works we need to clarify precisely what a string is.

### What is a String

A string is a sequence of up to 255 characters. The characters can be anything we wish. There are no 'special' characters that signify the start or end of a string. When we store a sequence of characters in a string variable, BASIC saves the string for us together with the number of characters in the string. So for example:

```
10 my_string$ = "This is a string"
```

causes BASIC to put the character sequence "This is a string" in the variable 'my\_string' and to store the length of the string, 16 in this case, with the variable name.

Having assigned a line of text to our string variable there are lots of operations that we can perform on the text. We will be covering these in detail later but for the moment we will just consider printing the string.

```
20 PRINT my_string$
```

This tells BASIC to look for the variable 'my\_string' and print whatever happens to be in the variable. BASIC knows how many characters it needs to print because it stored the length of the string when it stored the actual text. So in this example BASIC knows it has to print 16 characters.

```
30 my_string$ = "This"  
40 PRINT my_string$
```

In line 30 we have given the variable a different sequence of characters. Line 40 prints it for us and again BASIC knows how long the string is - only 4 character now.

```
50 my_string$ = "T"  
60 PRINT my_string$
```

Line 60 only has to print 1 character.

```
70 my_string$ = ""  
80 PRINT my_string$
```

Line 80 has nothing to print. But the string still exists. As far as BASIC is concerned it is just an empty string variable. In fact the concept of an empty string is so important that it even has a special name - a NULL string. Line 70 set the variable 'my\_string\$' to be a NULL string.

Returning to our program, line 5050 is checking to see if `message$(floor,room)` is a NULL string. If it is then there is no message for this particular guest so `PROC_no_message` is called. If `message$(floor,room)` is not a NULL string, there must be a message so `PROC_message` is called.

Having called the correct procedure to print to the screen, `PROC_do_output` has finished its job.

## PROC\_get\_digit

The definition for this procedure is quite straightforward. It simply has to collect one of the keys '0' to '9' from the Notepad's keyboard.

Since we want the user to enter a number, we could use BASIC's `INPUT` command in a routine something like this:

```
6000 DEF PROC_get_digit
6010 INPUT this_digit
6020 ENDPROC
```

This would work fine but would be a little irritating for the user. Having pressed a number key he would then have to press the `ENTER` key. We can produce a much better result by effectively reading the keyboard directly.

```
6000 DEF PROC_get_digit
6010 got_digit = FALSE
6020 REPEAT
6030   key = INKEY(0)
6040   IF key > 47 AND key < 58 THEN got_digit = TRUE
6050 UNTIL got_digit
6060 this_digit = key
6070 ENDPROC
```

First notice that we are back with those ASCII codes again. They do tend to crop up regularly. Here line 6040 is checking to make sure that the ASCII code of the key that the user pressed is higher than 47 and lower than 58. 47 is the ASCII code immediately lower than that for the key '0'; 58 is the ASCII code immediately higher than that for the key '9'. So line 6040 just checks to make sure the user really did press a digit.

## INKEY

The command in line 6030 looks directly at the keyboard. If the user has not pressed a key, this command returns the number 'minus 1'. In our example the test in line 6040 would fail and BASIC would repeat the loop for us.



When the user does press a key, the command `INKEY` returns the ASCII code for that key immediately - it doesn't wait around for the `ENTER` key to be pressed.

The number in the brackets after `INKEY` tells BASIC how long to look at the keyboard before giving up if no key is pressed. This number is in hundredths of a second.

```
INKEY(0)      means don't wait at all
INKEY(100)    means wait for 1 second before giving up
INKEY(6000)   means wait for 1 minute before giving up
```

There is a very similar command, `GET`, that does the same job as `INKEY` but will wait indefinitely for a key to be pressed. For a discussion of the different uses of these two commands see the section concerning `INKEY$` later.

### **PROC\_get\_room, PROC\_get\_floor**

These two procedures are very similar. They each have to ask the user for a number and then check that it is valid. As with all the input routines, they should only return to the section of program that called them when they have received good data. The sooner bad data is trapped out of a program the better.

```
300 DEF PROC_get_room
310 valid_room = FALSE
020 REPEAT
7030 PRINT TAB(4,4) "What is the guests room number      "
7040 INPUT TAB(35,4),room
7050 IF room > 0 AND room < 21 THEN valid_room = TRUE
7060 UNTIL valid_room
7070 ENDPROC
```

```
8000 DEF PROC_get_floor
8010 valid_floor = FALSE
8020 REPEAT
8030 PRINT TAB(20,5) "On which floor          "
8040 INPUT TAB(35,5),floor
8050 IF floor > 0 AND floor < 5 THEN valid_floor = TRUE
8060 UNTIL valid_floor
8070 ENDPROC
```

### **PROC\_no\_message**

The procedure that is called when there is no message for the guest simply has to point this out to the user. By calling `PROC_get_enter` we are assured that the user has time to read our message before the screen is cleared ready for the next operation.

```

9000 DEF PROC_no_message
9010 PRINT TAB(4,6) "There is no message for this guest - ";
9015 REM make sure the user reads this by waiting for a key
9020 PROC_get_enter
9030 ENDPROC

```

## PROC\_message

Now we get to the whole point of the program - displaying a previously stored message for our guest.

```

10000 DEF PROC_message
10010 PRINT TAB(5,6) "The message for this guest is: ";
        message$(floor,room)
10020 PRINT TAB(19,7) "";
10025 REM make sure the user reads this by waiting for a key
10030 PROC_get_enter
10040 message$(floor,room) = ""
10050 ENDPROC

```

Apart from actually displaying the guest's message the only other job this procedure has to do is to clear the variable containing the message. This is achieved by setting the variable to a NULL string - remember that this is just telling BASIC that the string no longer has anything in it but that it does still exist.

## PROC\_get\_enter

On to the last procedure in this example. Its definition is easy - wait for the user to press the ENTER key. It uses the INKEY command just like PROC\_get\_digit and again has to test against an ASCII code. Here the magic number produced by the ENTER key is 13.

```

10100 DEF PROC_get_enter
10110 PRINT "Press the ENTER key when ready";
10120 got_enter = FALSE
10130 REPEAT
10140     key = INKEY(0)
10150     IF key = 13 THEN got_enter = TRUE
10160 UNTIL got_enter
10170 ENDPROC

```

## Summing Up the Hotel Message Example

We now have a complete application program. We defined a requirement and wrote a structured program that met its specification. If you are new to BASIC and have

followed through each procedure you deserve congratulations. But before you get too carried away we should consider what the problems are with our program.

- 1) It isn't really very helpful to the user. If the user types in an invalid room number for example, our program doesn't tell him what the problem is - it just asks again.
- 2) It can only store one message at a time for each guest.
- 3) There is no way to look at a message without clearing it from memory.
- 4) There is no way to go back and change a message without first removing it and then re-entering it completely.
- 5) If you stop the program and use your Notepad to write a letter or check the diary, when you run the program again all the old messages will have been lost.

Looking on the bright side however, because the program is structured, we can address these problems without having to re-think our whole strategy. Indeed most of the above problems could well be corrected by using commands that we have already discussed and you may like to consider how this could be done.

The only difficult nut to crack is the problem of how you stop the program, go off and do something else with your Notepad and then come back to the program with all the messages still intact. To solve this we need to save our information to a file - a subject we will cover later.

## Playing With Strings

Working with numbers and generally 'doing sums' is a common experience for most people, but the idea that you can do analogous operations with lines of text may be new.

Any operation on a string is normally referred to as a function and the BASIC within your Notepad has a wide range of string functions.

Before we cover the functions we will review what we mean by a string.

A string is literally a sequence of characters that BASIC stores in memory and knows by the name of the string variable to which we assigned the characters. In order for BASIC to be able to work with the string, it needs to know how long the string is. Thus whenever we assign a sequence of characters to a string variable, BASIC always stores the string length as well. Under normal circumstances we are never aware of this but if we need to know BASIC has a way for us to find out.

## LEN

When we give the string function LEN a variable name it will return the length of the string for us.

```
10 first_string$ = "This is a string"
20 second_string$ = "Here is another string"
30 your_string$ = ""
40 PRINT LEN(first_string$);
50 PRINT LEN(second_string$);
60 PRINT LEN(your_string$)
```

When you RUN this program you should see the following on the screen:

```
16          22          0
```

The variable 'first\_string\$' holds 16 characters, the variable 'second\_string\$' holds 22 characters and the variable 'your\_string\$' doesn't hold any characters.

The number that LEN returns is just that - a number, so we can use it just like any other number.

```
70 total_length = LEN(first_string$) + LEN(second_string$) +
  LEN(your_string$)
80 PRINT total_length
```

This should print the number 38.

## Adding Strings Together

Strings are not added in the same way as numbers. It would make little sense trying to add "This is a string" to "Here is another string". When two strings are added there are just joined. So the lines

```
90 your_string$ = first_string$ + second_string$
100 PRINT your_string$
```

would show

```
This is a stringHere is another string
```

on the screen. BASIC has joined the second string onto the end of the first string and put the resultant string into the variable 'your\_string\$'.

As usual we can extend the principal. If we changed line 90 thus

```
90 your_string$ = first_string$ + " " + second_string$
```

the screen would show

```
This is a string Here is another string
```

BASIC has added a space to the end of the first string and then added the second string. Remember that the two variables 'first\_string\$' and 'second\_string\$' will not have been changed by this operation. Only the contents of the variable 'your\_string\$' has been altered.

## Subtracting Strings

BASIC can't subtract strings. It can split them, but there is nothing analogous to the sum 'A = B - 2' as far as strings are concerned.

There are three functions that can split up a string - one takes the left-hand end, one takes the right-hand end and one takes the middle.

### LEFT\$, RIGHT\$, MID\$

```
10 a_string$ = "Here is where we start"  
20 b_string$ = LEFT$(a_string$,4)  
30 c_string$ = RIGHT$(a_string$,5)  
40 d_string$ = MID$(a_string$,9,5)  
50 PRINT b_string$: PRINT c_string$: PRINT d_string$
```

After running this program our screen would show

```
Here  
start  
where
```

Each of these functions needs to know what string it has to work on and how many characters to take. MID\$ also needs to know where to start from.

Line 20 tells BASIC to use the variable 'a\_string\$', take the 4 leftmost characters of the string and put them into the variable 'b\_string\$'. Similarly, line 30 tells BASIC to use the variable 'a\_string\$', take the 5 rightmost characters of the string and put them into the variable 'c\_string\$'.

Line 40 needs an extra piece of information. The MID\$ function needs to know where to start. In our case we started at the 9th character and assigned 5 characters to the variable 'd\_string\$'.

If the string that we have told these functions to work on is not long enough, BASIC will just try it's best. So for example if we changed line 10 to:

```
10 a_string$ = "AA"
```

and tried the program again, we would get:

```
AA
AA
```

BASIC will print as many characters as it can so we get two lots of AA's followed by a blank line. LEFT\$ and RIGHT\$ were able to print something but MID\$ had nothing to print. We have told it to start at the 9th character and there isn't a 9th character.

A variation when using MID\$ is to only supply the start position. The function will then return all the characters from the start position to the end of the string. For example:

```
10 a_string$ = "Here is where we start"
20 d_string$ = MID$(a_string$,9)
50 PRINT d_string$
```

would print the following to the screen:

```
where we start
```

The numbers that we give to the string functions don't have to be fixed; they can be variables themselves. Indeed we don't even need to give the functions a string variable; the string can be included in the command.

```
10 FOR test_loop = 1 TO 19 STEP 3
20 PRINT LEFT$("abcdefghijklmnopqrstuvw",test_loop)
30 NEXT test_loop
```

This program would print

```
a
abcd
abcdefg
abcdefghij
abcdefghijk
lm
abcdefghijklm
nop
abcdefghijklmnop
qrs
```

The first time round the loop the variable 'test\_loop' has the value of '1' so line 20 effectively reads:

```
20 PRINT LEFT$("abcdefghijklmnopqrstuvw",1)
```

This tells BASIC to print 1 character starting from the left of the string. The next time round the loop the variable 'test\_loop' has the value of '4' (the STEP part of the FOR .. NEXT added '3') so line 20 now reads:

```
20 PRINT LEFT$("abcdefghijklmnopqrstuvw",4)
```

We have asked BASIC to print 4 characters starting from the left of the string. The loop will continue until the value of 'test\_loop' is greater than '19'.

## Comparing Strings

Checking that two strings are the same is exactly analogous to checking that two numbers are the same.

```
1000 IF num1 = num2 THEN PRINT "The numbers are the same"  
1010 IF a$ = b$ THEN PRINT "The strings are the same"
```

When BASIC processes line 1010 it is actually checking two different things about the strings. The first test is to make sure that the two strings are the same length. If they are different lengths then the test will fail before BASIC has even looked at any of the characters.

Assuming that the two strings are the same length, then BASIC looks at each character in turn to make sure it is the same in both strings. The first character in one string is checked against the first character in the other string. If the first characters are the same, BASIC checks the second characters and so on to the end of the string. If BASIC gets to the end without finding a difference, the test will pass and in our case a message will be printed.

The characters have to be identical in each position. The test is 'case sensitive' so the following two strings are different:

```
"this is a string"    and    "THIS IS A STRING"
```

Checking that two strings are different is not the same as checking that two numbers are different.

When BASIC compares two numbers to test if one is greater than the other it only has to consider one thing - which number is the greater. When BASIC compares two strings it has to consider the length of each string as well as what is in each string.

We will consider the contents of the strings to begin with. Suppose we had the following test program to show us how strings are compared:

```

10 REM Test program for string compares
20 INPUT "First string",a$
30 INPUT "Second string",b$
40 IF a$ < b$ THEN PRINT "Second string is greater"
50 IF a$ > b$ THEN PRINT "First string is greater"
60 IF a$ = b$ THEN PRINT "Strings are equal"
70 GOTO 20

```

This program will just ask us to type in two strings, compare them and then loop back to the start again. We can stop the program by pressing the STOP key.

Using the test program we can show how BASIC performs the comparison.

Entering "abc" and "abc" will produce "Strings are equal". Both strings are the same length and they have the same characters.

Entering "abc" and "xyz" will produce "Second string is greater". Both strings are the same length so the comparison has been based purely on the contents of the strings. Why has BASIC decided that "xyz" is greater than "abc"? The reason goes back to the ASCII CODE we described earlier.

Remember that a computer cannot understand letters - it can only understand numbers. The ASCII CODE defines a number for each letter on the Notepad keyboard as well as numbers for lots of characters not present on the Notepad keyboard. When you press the key 'a' BASIC sees the number 97. When you press the letter 'b' BASIC sees the number 98. It is these numbers that BASIC is comparing when checking our strings.

If we write our two strings not as characters but as their ASCII CODEs we can see why BASIC printed "Second string is greater".

The ASCII CODEs for "abc" are 97,98,99  
 The ASCII CODEs for "xyz" are 120,121,122

BASIC compared the first ASCII CODE from our first string with the first ASCII CODE from our second string. In this case the test ended because '120' is greater than '97' so BASIC gave us our message.

Entering "abc" and "abd" will produce "Second string is greater". This time the ASCII CODEs for each of the first two characters in the strings are equal. It is only on the third character that the ASCII CODE from the second string is higher than that from the first string.



## Chapter 21 Basic

Entering "abc" and "ABC" will produce "First string is greater". Looking at the ASCII CODEs again:

The ASCII CODEs for "abc" are 97,98,99

The ASCII CODEs for "ABC" are 65,66,67

Lower case letters have higher ASCII CODEs than capital letters.

Now lets add the complication of length to our comparisons. The rules for comparing the individual characters are the same but now the test will be stopped for one of two reasons. First, if the code for a character in one string is greater than that for the corresponding character in the second string, BASIC has found the difference. Secondly, if the end of one string is reached before the end of the other string, the longer string is greater.

Using our test program again, entering "abc" and "ab" will produce "First is greater". The first two characters in each string are the same so the test rests on what to do about the extra character in the first string. Effectively BASIC has said that since there is no third character in the second string, it's ASCII CODE is zero. BASIC then compares the ASCII CODE for "c" against '0' and thus prints "First is greater".

The general rule is that if two strings have the same characters up to the point where one of the strings ends, then the longer string is the greater.

Entering "ABC" and "ab" will produce "Second string is greater". Here we have the situation that although the first string is longer than the second string, the length never gets considered. The test has already been determined before the end of the second string is reached because the ASCII CODEs are different.

We can summarise these results in a table as follows:

First string	Second string	Result
abc	abc	strings are equal
abc	xyz	second is greater
abc	ABC	first is greater
abc	XYZ	first is greater
ab	ABC	first is greater
ab	abc	second is greater
ab	ABCDEFGHIJ	first is greater

## Searching Within a String

It is often useful to be able to look for one string inside another string. BASIC provides a powerful function able to do this for us.

### INSTR

```
10 first_string$ = "This is a string"
20 second_string$ = "a s"
30 got_it = INSTR(first_string$,second_string$)
```

The INSTR function searches one string variable to see if it contains a second string. If the second string is found the function returns the position that the second string starts at. If the second string isn't found the function returns the value zero. Unless we tell it otherwise, INSTR will always start searching from the first character in the first string.

In our example we have set the first string to "This is a string". Our second string is "a s". INSTR will find this second string starting at position 9 in the variable 'first\_string' and will therefore set the value of 'got\_it' to '9'.

By giving INSTR some extra information (another parameter) we can tell it to start searching at some point other than the first character in the first string.

```
30 got_it = INSTR(first_string$,second_string$,6)
```

This time INSTR will not start looking for the second string until it has reached position six in the first string. A typical use of this extra parameter is to check for more than one occurrence of the second string in the first string. Suppose we had the following program example:

```
10 first_string$ = "This is a string"
20 second_string$ = "is"
30 got_it_1 = INSTR(first_string$,second_string$)
40 got_it_2 = INSTR(first_string$,second_string$,got_it_1 + 1)
```

This time our second string occurs twice in the first string. As above line 30 will find the occurrence, now at position 3. But the second string is also at position 6. This will be found by line 40 which doesn't start searching until one character later than line 30. If we had left off the extra parameter in line 40 INSTR would have found the first occurrence again because it always starts at the first character in the first string.

This is very much 'special case' programming - the example took account of the fact that there were two occurrences of the second string in the first string. What we need is a more general approach to checking for one string inside another.

Suppose we were writing a program that had to be able to accept the current month, typed in using letters, as input from the user. We would need a way to check that the user had indeed entered a valid month. If we had a list of the twelve months as a single string, we could use the INSTR function to make sure that we had received valid input from the user.

```
100 REM Get a valid month from the keyboard
105 REM make a string with all the months in
110
month$=":January:February:March:April:May:June:July:August:September:October:November:December"
120 INPUT "Type in the month ";user$
125 REM we need to count the number of times user$ is found
130 found = 0:start = 1
140 REPEAT
145   REM see if user$ is good
150   position = INSTR(month$,user$,start)
160   IF position > 0 THEN PROC_found ELSE PROC_not_found
170 UNTIL found > 79
180 IF found = 0 THEN PRINT "Not found"
190 IF found = 1 THEN PRINT "Found at ";save_pos
200 IF found > 1 THEN PRINT "Not enough characters"
210 GOTO 120
```

Line 110 sets up our string for us. We have separated each month by a colon to help us check later on that the data is good. Line 120 prompts the user to enter the month and puts the user's input into the string variable 'user\$'.

From the programmer's point of view it would be easiest if we made the user type in the whole name for the month he wanted. However this would not be terribly 'user friendly' so we are going to allow the user to just type in enough characters to uniquely identify the correct month. For some months this is easy - 'O' would give us October; 'S' would give us September. But 'J' on it's own is not good enough - does the user mean January, June or July. If we can be sure that the variable 'user\$' only occurs once in the list of months then we know what the user means. If 'user\$' occurs more than once we need to ask for more information. Hence the need for the loop between lines 140 and 170.

We begin the test by setting the variable 'found' to zero in line 130. We also set a position to start searching - the variable 'start'. Then we start a REPEAT..UNTIL loop which controls where INSTR will begin looking for 'user\$' in 'month\$'. If INSTR is successful the value of 'position' will be greater than '0' so PROC\_found will be called. If INSTR failed to find 'user\$' the value of 'position' will be '0' and hence PROC\_not\_found will be called. The loop will end when the value of the variable 'start' is greater than '79'. Where did '79' come from then? It is the position in 'month\$' where the last month begins. If we don't find a match when we start from here, we won't find a match any later.

Lines 180 to 200 print out the result of the test. If the value of 'found' is still zero at the end of the loop, no occurrences of 'user\$' were found in 'month\$' so the user must have typed in invalid data. If the value of 'found' is '1', only one occurrence of 'user\$' was found so we can be sure we know what the user meant. If 'found' is greater than '1' then we can't be sure - 'user\$' was found more than once.

The two procedures change the value of the variable 'start'.

```

300 DEF PROC_found
305 REM record the fact that we found a match
310 found = found + 1
315 REM save the place where the match was found
320 save_pos = position
325 REM we can start the next search loop from where the match was found
330 start = position + 1
340 ENDPROC

```

When we find an occurrence of 'user\$' in 'month\$' there are several variables to change. The value of 'found' has to be incremented to record our match; we need to remember where this match occurred so that we can tell the user if there is only a single match; and we can move the start of the next search to after the point where the match was found. Note that line 330 sets the new value of 'start' to the current match position plus one so that we don't find the same match twice.

```

400 DEF PROC_not_found
410 start = 80
420 ENDPROC

```

If we don't find a match for 'user\$' this procedure only has one job to do. Since a match wasn't found from the current start there can't be a match anywhere further along 'month\$' so we can stop the loop from going round again. All we need to do is to set the variable controlling the loop, 'start', to a number greater than we have given in our UNTIL statement - '80'.

If you type this example in to your Notepad and 'RUN' it you will find that it works correctly if you give it good data. If you type in 'S' it will print "Found at 52". If you type in 'J' it will print "Not enough characters". However the program does have a fundamental flaw - try typing in 'eb' and the program will print "Found at 11". Not quite what we wanted.

The problem lies in the fact that we haven't trapped parts of months. As it stands the program will tell us we have good data if it can find a single match against any group of characters typed on the keyboard. Perhaps the reader would like to consider how to correct this flaw - HINT: the colons we added to 'month\$' can tell you when you are at the start of a month but beware of the user typing in something like "t:S".

## Changing the Variable Type

We have already talked about real and integer variables and lists of characters that can be stored in string variables. We have also covered the ASCII CODE and the way that each character is stored as a number. Since each character in a string is in fact just a number, it should come as no surprise that we can switch between strings and numbers. BASIC has two commands to allow us to do this.

### CHR\$ and ASC

These two commands perform opposite functions: CHR\$ will convert a number to a string and ASC will convert a string to a number. There are however very tight constraints on when this will work.

The ASCII CODE only has two hundred and fifty-six possible values running from '0' to '255'. Therefore when we use CHR\$ to convert a number to a string we can only give it a number between '0' and '255'. In fact BASIC will make an attempt at whatever number we give it, but the results are only meaningful if we stay within the valid range.

```
10 FOR my_char = 65 TO 90
20 PRINT CHR$(my_char)
30 NEXT my_char
```

This would print all the capital letters of the alphabet. The ASCII CODE for the letter 'A' is '65' and the ASCII CODE for the letter 'Z' is '90'. (see the appendix for a full list of the ASCII CODEs).

Similarly the ASC function can only accept a string one character long to convert to a number. If we give it a longer string it will just convert the first character in the string for us.

```
10 my_string$ = "A very long string that ASC cannot cope with"
20 PRINT ASC(my_string$)
```

This will only print the number '65', being the value of the first character in the string. If we wanted to find out the ASCII CODE for a character other than the first character we would need to isolate the correct character first.

The following example steps through each character in the string printing out its ASCII CODE.

```
5 REM find the ASCII CODE
10 my_string$ = "A very long string that ASC cannot cope with"
15 REM find out how long the string is
20 length = LEN(my_string$)
```

### **Define Character Width (> CW)**

This defines the width of a character when microspacing. The default number is twelve. Other examples are ten for elite, seven for condensed and fourteen for condensed enlarged.

### **MicroSpacing On/Off (>MS ON/OFF)**

This turns Microspacing on and off. The default is off.

### **Proportional Printing On/Off (>PP ON/OFF)**

This is selected by turning it ON with this command and using the Style attribute 'p' command at the start and end of text you wish to be printed.

If your printer supports proportional printing the Notepad has to alter the number of characters that it can print on each line while printing. If you had a line of 'i's you would get more of those on a line than would if you had a line of 'm's.

This command has been designed for use with proportional daisywheel or laser printers. Dot matrix printers can work very slowly in this mode. Fixed pitch daisywheel and non-proportional dot matrix printers do not use this command.

wrong with your code. If you find that the correct code is being sent to the printer from your Notepad, then it is time to get in touch with your printer supplier for advice.

## Lines and Foreign Characters Don't Print

If your printer does not print lines, there is a good chance that it is not using the correct table. You should compare the printer's ASCII table to the Notepad's ASCII table. They should be identical for the first half, with possible changes in the second half of the table.

Often, the printer has several tables that it can use. Make sure that the correct one is selected and that the Printer character set in the Printer Configuration menu is set to IBM on the Notepad.

## Microspacing and Proportional Printing

Microspacing and Proportional Printing have to be turned on from within your document. By default they are turned off. Furthermore, you will only see any effect from them if you have right justified your document.

With ordinary character spacing the spare space is collected on the left and right of the line. When microspacing is turned on, the amount of space that soft spaces take up is redistributed evenly between every word in that line. This makes the text look better, and can be used on printers that do not support proportional printing.

Proportional printing is even better than microspacing. Instead of allocating the same space for each character, proportional printing changes the amount of space each character uses. It might be obvious, but an 'm' takes up more space than an 'i' when you write it. When you print them normally there is a lot of space on either side of an 'i' but the 'm' looks a bit squashed.

### Define Microspace Code Sequence (>MC n n)

The commands for microspacing and proportional printing should already be defined in your printer driver. If microspacing doesn't work, then add the commands into your document.

The Microspace Code defines the sequence of codes that moves the print head by the smallest amount. Other names that it can be known as include Horizontal Mode Increment and Graphics 120 dpi.

A typical example would be >MC 27,L,1,0,0

using a capital O that look like a number zero  
a number (0-9), which is used as an ASCII code instead of a number, which  
needs to be put into quotes.

---

### Printer Code Not Working

There are more wrong printer codes than there are right ones. You need not be afraid of getting it wrong. There are no self destruct codes. All that happens when the code is wrong is that the printer doesn't do what you expected. It might stop working when it reaches the code, or it might print different characters (including part of the code) onto the paper.

Before throwing your printer out of the window, just take one last look at the printer manual. Perhaps there was a small phrase in the description that you didn't understand the first time through.

If you find some of the code being printed, then the letters or codes before that are probably wrong. If all else fails, there is another trick that you can use.

### Hex Dump

A good way to find what Notepad is really sending to your printer is to put the printer into Hex Dump mode. How to put your printer into this mode is explained in your printer's manual. It should then print out both the ASCII characters and the hexadecimal number.

To test a code put the printer in Hex Dump mode and write a three line document. For example:

The start before the code

>OC - put the code string here

Here is the changed text

At the start of the printed document you will see the initialisation codes that Notepad sends to the printer. Your text should follow that.

The escape character in hex is 1B and you will also see some 0D and 0A numbers that are linefeed and return codes.

By printing out a small amount of text with your code(s) you could find out what is



*Description*

*Character height is based on points with larger fonts having a higher point size than smaller ones (12 point Courier is larger than 8.5 point Courier). If you select a point size that is not available, the printer uses a font with the closest point size.*

*For scaleable fonts the value field is from 0.25 to 999.75 points in increments of 0.25 point.*

*One point equals 1/72 inch.*

Quote end

Well here is an easy one. To change the point size to 24 point just use

27,(,s,"2","4",V

**Output Code to Printer (>OC n n)**

Here is the command that you put in front of the code to send to your printer.

The command line to change a laser printer to double height becomes -

>OC 27","(", "S", "2", "4", V

**Multiple Commands**

There is nothing to stop you issuing multiple commands to the printer. Normally, you just keep adding the next command straight after the previous one.

Laser printers follow a different rule. The last character in each command is usually a capital letter. To join two commands together you change the previous command's last letter to lower case and continue the next command without the escape character sequence.

For example, if you wanted to combine the two commands ESC&[10E and ESC&[6D, you would end up with the sequence ESC&[10e6D.

---

**TIP Gotcha Printer Codes**

**There are several tricks that printer manufactures get up to when designing codes. Their favourite ones include using the following:**

**using a lower case L that looks like a number one**

## Chapter 12 Printing

The <n> bits are there to tell the printer how many mode bytes we are going to send. In the text it is stated that they are normally 4 and 0 which makes a bit of sense as we are going to use all four mode bytes. So four and zero are the next numbers to go onto the line.

The text describes the mode bytes m1 and m2 as 'without function'. In plain English that means they don't do anything so we will use a zero for each.

To use mode bytes m3 and m4 we need to look at the tables. The option we want to use is double height line feed, which would move the paper up two lines, and double height characters. This is the last entry in the table for mode byte m3 and it tells us to use the number 34 decimal (22 in hexadecimal). Finally, there is the mode byte m4 that alters the character width, which we are going to leave unchanged. The number here is a zero.

So by substituting the numbers we have just worked out into the Escape code quoted in the Proprinter manual we arrive at -

```
<ESC>[@<n1><n2><m1><m2><m3><m4>
```

```
27,[,@,4,0,0,0,34,0
```

Easy, isn't it?

Finally, let's look at a laser printer's code. There isn't a double height command here, but we can change the point size of the font that is selected.

*Quote from manual*

*Point Size (primary font) ESC(s#V*

*Point Size (secondary font) ESC)s#V*

*Function Selects the primary or secondary character font height. Value # equals the height in points, and may include decimal fractions.*

*The following are typical values for font height:*

*Value    Font Height*

*7 point    (Not available using the resident fonts)*

*8 point    (Not available using the resident fonts)*

*8.5 point*

*10 point*

*12 point*

*14.4 point    (Not available using the resident fonts)*

The low-order half-byte contains the multiplier 1 or 2 for line spacing, for example

multiplier = 1 m3 = bin.0001 0001  
 multiplier = 2 m3 = bin.0001 0010

If one of the half-height assigned 0 (Zero), i.e..bin.0000, the current values of line spacing or character height remain unchanged.

m3 (dec)	m3 (hex.)	m3 (bin.)	Line feed	Character height
0	00	0000 0000	unchanged	unchanged
1	01	0000 0001	unchanged	standard
2	02	0000 0010	unchanged	double
16	10	0001 0000	single	unchanged
17	11	0001 0001	single	standard
18	12	0001 0010	single	double
32	20	0010 0000	double	unchanged
33	21	0010 0001	double	standard
34	22	0010 0010	double	double

Mode-byte<m4>:

m4 (dec)	m4 (hex.)	m4 (bin.)	Character width
0	00	0000 0000	unchanged
1	01	0000 0001	standard
2	02	0000 0010	double

The high-order half-byte is ignored, the low-order half-byte contains the multiplier 1 or 2 for character width

Quote End

Gosh, remind me not to get an IBM Proprinter XL24e printer! Does this description look bad? Well, yes and no. There certainly is a lot of it. That is because the same command changes the width of the characters as well as their height.

Let's just go for the double height, like the other commands we have used before. To start with there is the <ESC> code again. This is followed by the [ and @ that we put in quotes. Now for the <n> and <m> bits.

---

**TIP - What are ASCII Tables?**

---

An ASCII table is the list of codes that the printer uses to print out the characters. For example, a capital A is decimal 65. The standard ASCII table finishes at decimal 127, but there are a possible 256 codes available. There is an ASCII table in the Appendices.

---

*Quote from manual  
IBM Proprinter XL24e Mode*

*set double height      <ESC>[@<n1><n2><m1>...<m4>*

*By means of this code sequence the characters of the subsequent text are specified with double width, double height or both combined. Additionally line spacing can be controlled by this code sequence.*

*n1 and n2 specify the number of mode bytes contained in the sequence. Usually n1 is defined by 4, n2 by 0.*

*The mode bytes m1 and m2 are without function, i.e. they are assigned code <NULL>(hex.00).*

*The mode bytes m3 and m4 control the print parameters:*

*m3 controls line spacing (high-order half-byte) and character height (low-order half-byte).*

*m4 controls the character width. Only the low-order half-byte is functional in the mode byte.*

*For detailed information on modes bytes m3 and m4, see next paragraph.*

*Definition-Mode*

*Mode-byte m3:*

*The high-order half-byte contains the multiplier 1 or 2 for line spacing, for example*

*multiplier = 1 m3 = bin.0001 0001*

*multiplier = 2 m3 = bin.0010 0001*

have now. Number 12 was written as xii for example. The numbers look different, but they are the same value.

The codes in the printer manual are usually described in one of three different ways; decimal, hexadecimal or ASCII. Fortunately we don't need to understand them to be able to use them.

Protext can understand all three different types of numbers, but you do need to show which sort you are using when typing in the codes. Let's look at a few codes in a printer manual. It is time to get a strong cup of coffee (beer if you must).

## Set Double Height Example

Here is a command that our printer can use in three different emulations. Let's look at these one by one, that way we can also test that it works as well. This is what the manual has in it.

*Quote from Manual  
Epson FX-850 Mode*

*set double height <ESC>w<n>*

*By means of this code sequence the characters of the subsequent text are specified with double height.*

*<ESC>w followed by 1, sets double height printing. To reset the double height to the normal height, specify <n> by 0.*

*Quote Ends*

That looks very straight forward doesn't it? So how do we use the information?

## Escape

The <ESC> is an ASCII Escape character. To type this into your command line use either the decimal number 27 or hexadecimal equivalent &1B. The lower case w is also an ASCII character, just put "w" in the line. Finally the number one or zero is a number so type that in too. These codes may be separated with a space or a comma. So here are the results.

```
27,"w",1 will turn double height on
27,"w",0 will turn double height off
```

Let's do that again with a different printer.

## Advanced Printer Commands

You can add the following commands to your documents. They are all printer related. Whether you can use them or not will depend on the type of printer you are using and whether it supports any of the functions listed here. The other point we would like to make is that you would probably not use these straight away. If this is your first time through the book you may like to miss this bit out!

Most of the functions to do with the printer are already set in the printer driver and are selected in the Printer Configuration menu. However, if you wish to send different codes to your printer, you can include them in the document.

---

### Definition - Printer Driver

The Printer Configuration menu allows you to tell your Notepad what type of printer you will be using. The choices are Simple, Canon BJ10e, Epson 24 pin, Epson 9 pin, IBM 24 pin and LaserJet.

The Notepad contains a printer driver for each of these types of printer. The printer driver holds information such as what codes are needed to feed the next sheet of paper or to turn underline on and off.

Because the Notepad has a range of built-in drivers you don't normally have to be concerned with the nuances of controlling your printer.

If your printer is not one of those catered for by a built-in driver, or you want to use a feature on your printer that is not supported by the driver, you may need to use some of the following commands to get the most from your printout.

---

## Printer Codes

In some printer manuals there are helpful sections on how to use printer codes. If you have a printer manual that just lists the codes, you will have to put up with our helpful suggestions instead.

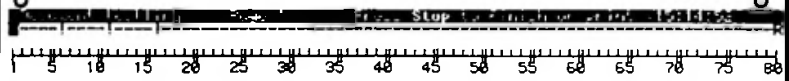
### Different Numbers That Mean the Same Thing

Before finding some codes to work with, we will explain about the numbering systems that the codes are written in.

You may remember that the Romans had a different numbering system to the one that we

1234567890123456789012345678901234567890123456789012345678901234567890  
1234567890123456789012345678901234567890123456789012345678901234567890  
1234567890123456789012345678901234567890123456789012345678901234567890  
1234567890123456789012345678901234567890123456789012345678901234567890  
1234567890123456789012345678901234567890123456789012345678901234567890  
1234567890123456789012345678901234567890123456789012345678901234567890  
1234567890123456789012345678901234567890123456789012345678901234567890  
1234567890123456789012345678901234567890123456789012345678901234567890  
1234567890123456789012345678901234567890123456789012345678901234567890  
1234567890123456789012345678901234567890123456789012345678901234567890

Numbers repeated down the complete page.



You can even make up a ruler with the Box Drawing Commands if you want to.

length greater than 127, you will discover that the page length will be set to that number less 128 lines. This effect is similar to a mileometer in a car, when it reads 99999999. One mile later and you have a new car because the mileometer has run out of numbers.

---

## Getting Rid of All the Margins

The next thing you need to do is to get rid the margins for the test page. That will tell you the number of columns and lines that your printer will print on the page without having to worry what the margins are set to.

There are two ways of doing this. Either set all the margins in the Layout menu to zero or use the >ZM - Zero Margins command at the top of the document.

## Fill the Page With Numbers

Copy your line of numbers from the page width test above about eighty times so that you have more than a pagefull and then print the page.

>ZM

1234567890123456.....6789012345678901234567890123456789012

You can use the test page to print on different sizes of paper like filofax or other non-standard sizes of paper. Printing numbers across the sheet gives you the coordinates (line and column) to position different parts of your letter. Using the line drawing mode you can even make up a ruler looking line as shown in the screen shot opposite.

## Designing Your Page Layout

After you have printed your test page, you can design your own page layout by changing the page length and margins to suit your own preferences.

## Test Whether Your Printer Can Print Lines

By adding a few boxes on your test page, you can also test whether your printer can print the lines as well. In our example, we have positioned the lines round the space for the address that you want to use with window envelopes.

If these lines print, there is a good chance that the other characters in the symbol menu will print also.



Can you remember all the necessary commands yet? Here's a reminder.

Add a ruler line into the text (Control R), move the cursor onto the ruler line and Goto (Control G) column 132 (c132) or Column 80 (c80). Put an 'R' there, turn Insert on/off to OFF (check the Status Bar) and Goto (Control G) column 70 (c70). Overtyping the whole line with '-' until you reach the new 'R' at column 132 or 80. Remember to change back Insert on/off back to ON when you have finished.

All you need to do now is to type in some text that uses the new width of the document and print it. A simple sequence of numbers will allow you to count the printed width easily.

When you know the number of characters that you can type across the page you need to find out how long the page is. Again, use the trick of typing more lines than the printer will print on the sheet. Seeing which line remains at the bottom of the sheet will tell you the number of lines your printer can print on a page.

## Page Layout (Function 7)

Overtyping or ← → to change, ↑ ↓ to move, Stop to Finish		
Page length (lines)	(65)	65:
Top margin (lines)	(3)	
Header margin (lines)	(2)	
Footer margin (lines)	(2)	
Bottom margin (lines)	(3)	
Side margin (chars)	(5)	
Line spacing (lines)	(1)	

Let's see how Protext lays out the text on the page.

Protext uses a default set of values for all the margin settings. This means that the first character on each line of text is not printed hard over to the left of the sheet of paper as you may have expected. The default value for the side margin is '5'. This causes five spaces to be printed at the start of every line.

The same is true at the top and bottom of the page because there are margins there as well. At the top of the page there is the Top Margin that is set to three lines. This is followed by two lines for the Header Margin that is used for any header text. At the bottom there is a similar set called the Footer Margin and the Bottom Margin of two and three lines respectively.

### TIP - Page Length Warning

**Do not set the Page length to zero or greater than 126 lines. Protext will ignore your setting of zero page length and will use your previous setting. If you use a page**

# Chapter 12

## Printing

### Print Test Page

Using all the commands we have covered so far, a good tool to make is a 'Print Test Page'. This does several things.

It checks that your printer and Notepad are set up correctly.

It finds out where your text is positioned on the page. This can be used for lining up where address labels are on the paper, where addresses need to be for use in window envelopes or lining up pre-printed forms.

The ideal way of doing this is by printing the test page on a transparent sheet (the sort they use for overhead projectors). Then you can see what you want the text printed on as well as where it will be printed.

The position of your text on the page depends on two things. First the word processor you are using and secondly the printer you have. Writing a test page with the word processor and then printing it with your printer will show you where characters will be printed on the paper.

### How Wide Can Your Printer Print?

Most printers can print up to eighty characters on a line in normal mode or one hundred and thirty-two characters on a line in condensed mode. What is the best way to find out? Send the printer a line that is far too long and see how many characters fit on the line. When your printer can't print any more characters on the line it will either put the remainder on the following line or throw it away.

### Typing a Wider Line

The default right margin is normally set at 70. We need to widen this to see when the printer runs out of characters. You need to decide which mode you would like to test your printer in before making the long line that follows.

Start a new document called 'longline' and change the ruler line so that you have either eighty or one hundred and thirty-two characters in the line.

want to save the document there are several ways to copy it from your Notepad.

Memory cards can remember what has been saved onto them even when they are removed from the Notepad. You could keep one card just to hold all your important documents.

If you have access to another computer, in particular a PC, transferring a document onto that computer's storage system is a cheaper approach than keeping an extra memory card.

You can even connect the Notepad to a stand alone disk drive to give you as much storage as you wish.

We will be covering transferring your documents to another computer later.

## **How Big Can Your Document Be**

No matter how much memory a computer has, there is always a limit on how big a document can be. Protext uses the largest continuous block in the lower memory. The largest document that you can have is 38144 bytes, which is roughly eighteen A4 sheets.

## **How to Print Multiple Documents**

If the restriction in document size is 'cramping your style' there is a way of getting round it. You could get more memory by buying an extra memory card. That wouldn't mean that any one document could be larger than 38144 bytes but it would mean that you could have more of them.

There is a way of joining documents together and printing a really long document from several smaller ones. You can use the Insert command. Look out for details in the 'Commands' section later.

### **Only Select documents created by the word processor.**

---

The extra information shown by the List is the document size expressed as a number of bytes, whether the document is stored on a memory card, whether it is in Upper memory or Lower memory and the time and date that the document was created.

## **Memory**

Your Notepad has just under 49K bytes (1K is 1024 characters) of memory that has been split into two areas. These areas are called 'Upper' and 'Lower' memory. If you plug a memory card into your Notepad you are adding extra memory in which to store documents.

Each character that you type (including spaces) into your document takes up one byte of memory. The memory is used in blocks of 256 bytes. You will notice the number of bytes of 'Free memory' jumping down in steps of 256 as a document gets larger.

Lower memory is used by the programs within the Notepad to carry out your instructions. Documents are stored in upper memory but if this area becomes full then lower memory starts to get used for documents as well.

When you enter the word processor to edit a document Protex needs an area of lower memory to operate in. This area must be large enough to hold your whole document.

The number of bytes used for each document is followed by a 'U' for upper memory, a 'L' for lower memory or a 'C' for the card. This information can help you to decide which files to get rid of when you are low on memory.

## **Running Out of Memory**

When you get the 'memory full' message it is time to delete some old documents. It does not matter if you delete the documents from upper or lower memory. If you have freed up some space in upper memory by deleting a document and you then edit a document previously stored in lower memory, Protex will move the document to upper memory for you.

Your Diary entries are also stored as files. Deleting them in the diary will also clear more space.

If you no longer require a particular document, you can just delete it altogether. If you

**TIP - Only Copy a Block or Document From the Word Processor**

**You can't copy a document from the List. You must select the document you want first and then go into the Word Processor. Selecting 'Copy block or Document' will allow you to make the copy.**

## Showing Other Files

The most useful way of showing the list of documents in your Notepad is to show only the name of the document. This allows you to see thirty-five names at a time when you enter the List. In this case only documents that have been created by the word processor are displayed. Files containing BASIC programs or your Address Book are hidden from view, as is the size of each document.

You can change the way documents are displayed by setting the 'Document sizes and date display' option in the System Settings menu to anything other than 'Not shown'. The List will then include more information about each document and show all the documents in the Notepad. The disadvantage is that you will only be able to see the names of fourteen documents at a time.

Document Name	Type	Date	Size
100 Lines	2 U	20-11-92	17:01
ADDRESS BOOK	104 U	20-11-92	9:30
bank loan	2 U	20-11-92	16:58
car service	2 U	20-11-92	7:00
data file	1 U	20-11-92	16:58
Don't Forget	2 U	20-11-92	17:04
job hunt	2 U	20-11-92	17:54
letterhd.doc	U	20-11-92	16:58
life so far?	U	20-11-92	16:59
NC100 book	U	20-11-92	16:59
Progl.bas	U	20-11-92	17:02
project1	U	20-11-92	17:05
project2	U	20-11-92	17:05

**WARNING - Don't Try to Select a File That Isn't a Document**

After changing 'Document sizes and date display' to show all the files, be careful NOT to select any of these other types of files with your word processor because it could have disastrous results.

If, for example, you selected a BBC BASIC file from inside Protext the file would be corrupted and you couldn't use it again.

To avoid this from happening you could change the setting back to 'Not shown' which will prevent you from selecting any other files as they would not be shown on the list.

Unfortunately there are times when a bug is serious and this is the case with the insert command in the List. If you use the command as it was intended to be used it works correctly, but if you select it without having another document to insert into you may lose some or all your documents in the Notepad. Then you would start wishing you'd done a backup.

While writing this book on the Notepad, we've come across a few bugs and have highlighted the problems they can cause. This does not mean that we've found them all; you will probably find more if you are inquisitive.

Crashes are the disasters of the computer world. They come in different guises but the results are normally the same. You will not be able to do anything with your Notepad until you have done a Reset. You will have definitely lost the document you were working on and if you are unlucky you will have lost all your other documents as well.

---

### **D - Deleting Documents**

As you type in more documents in your Notepad you are filling its available memory. You can release this memory for new documents by deleting any old documents that you no longer require.

### **P - Print**

Here is another place that you can print documents from as well as the Print File menu.

### **R - Rename**

Rename allows you to change the name of a document.

### **F - Format Memory Card**

When you get a memory card to store even more files on to you need to format it first. More about that later.

### **T - Transfer**

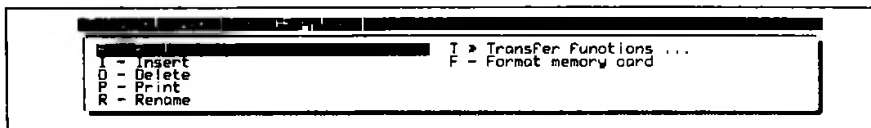
Transfer brings up another menu with more commands on it. These are used when you transfer documents to another computer. We will cover these in the Serial Terminal section later.

## Select a Document

To select a document to use with the Word Processor from the list, just move the black bar cursor over the name you want and press the ENTER key.

As well as selecting an existing document, there are several other things that you can do from the List. The important thing is to select the document first and then press the Menu key to select what you want to do with it.

## Document Operations Menu



When you have entered this menu make sure that the name of the document you wish to operate on appears at the top of the screen.

Here is what you can do and what you should not do with your documents.

### E - Edit

This command performs the same function as hitting the ENTER key from within the List.

### I - Insert - DO NOT USE THIS COMMAND

**DO NOT USE THIS COMMAND.** The Insert command should only be used to insert one document into another. If you have reached the List from the Main menu there will not be a document open to insert the one you have just selected into. The result will be at worst a computer crash or at best a warning. **YOU HAVE BEEN WARNED.**

---

### Definition - Bugs and Crashes

Complex programs written for computers like the Notepad owe their existence to a team of programmers who design and write the programs. No matter how much testing is done on a program they can never be completely certain they have found all the bugs. This is true for all large computer programs.

Sometimes bugs manifest themselves in trivial ways that cause no serious problems. A good example within Protex is the fact that you cannot use the Right Shift key to access the 'Find Previous' command.

# Chapter 11

## The List (Function L)

The list is where all your documents or files are stored. You can select any of your documents/files that you have created in your Notepad. If you want to re-read, print or do some more work on a document, you need to get the Select menu on the screen.



From the Main menu of the Notepad you use Function and Left Cursor keys to select the Word Processor menu and then the Right Cursor key.

Alternatively you can use the short cut key sequence (Function L) from any other program in the Notepad.

When you leave the Word Processor, Protex automatically saves your document for you under the name you gave the new document before you started typing. The new name gets added to the List.

---

### Definition - Documents and Files

There is no difference between a document and a file. Generally we have used the word document because it is more descriptive of the 'thing' that holds the text. Other computer systems tend to use the word file.

There are times when the word file makes more sense. The Address Book for example holds a list of names and addresses in a manner analogous to a paper filing system. BASIC programs are stored in a file and the programs can store information to a file.

The distinction we use is that documents can be loaded into the word processor; files are used to save other types of information.

You just need to remember that in reality both words mean the same thing.

---



## **E - Edit Word**

Edit Word will show you the wrong word in a small window so that you can correct it. When you press ENTER, the spell checker will check the word you have edited before moving onto the next mistake.

## **S - Store Word**

This command adds the selected word to your own user dictionary. This word could be your name or a special word that is not used often. It is a good idea to build up your user dictionary as it will save you time in the future.

## **User Dictionary**

After storing some new words with the above command, you can use the following commands to view the words you have entered or to delete them.

### **View User Dictionary (Word processor menu, Editing, View user dictionary)**

This command will show you all the words you have put into your user dictionary.

### **Remove Word From User Dictionary (Word processor menu, Editing, Remove word)**

You need to type the word to delete from the user dictionary and press the ENTER key. It is 'Case sensitive' so make sure that if the word in the dictionary starts with a capital letter, you do likewise when entering the word to delete.

## Spell Checker Rules

Here are a few rules that the spell checker follows.

Foreign letters used in words are recognised and the words can be added to the user's dictionary.

All possessive apostrophes ('s) are ignored but contractions of two words (can't, won't, haven't) are stored as separate words in their own right. The main dictionary contains most of the common contractions.

Words beginning with numbers (1st, 2nd, 3rd) are ignored.

## What Happens When the Spell Checker Finds a Mistake?

You get a choice of four options. These are:

- I - Ignore word
- L - Lookup word
- E - Edit Word
- S - Store Word

### I - Ignore Word

This does as it says; it ignores that word and makes no changes at all. If you select this option it means that you are happy with the spelling even if the spell checker isn't.

### L - Lookup Word

This allows the spelling checker to have a look in both dictionaries and make some intelligent guesses at what the word might be. Initially you should see the 'Looking . . .' message followed by a list of close matches or nothing at all.

You use the Cursor up or down keys to select the correct word (if it is there) followed by the ENTER key. The word that was wrong in your text will then be replaced by the word that you have selected.

If the word you wanted is not on the list, then press the Stop key to exit the selection. You will then get the following options 'E - Edit word' and 'Enter - Continue'. If you choose 'Continue' then the spell checker will leave the wrong word alone.

# Chapter 10

## Spell Checker

After typing a document, it is not a bad idea to run the Spell Checker as it can often find words that are either misspelt or mistyped. It WILL NOT find ALL your mistakes as a spell checker only looks at each word you have typed and compares it to two dictionaries. It has a main dictionary that has a large number of words already included and a user dictionary that is initially empty.

### What the Spell Checker Does

If the spell checker cannot find a match between a word in your document and one of its dictionaries it will let you know. It lists your options for that word and waits for you to select an option. Then it continues checking the rest of the text.

Spell checking can be stopped at any time by pressing the STOP key.

It will not tell you if you had put 'bet' instead of 'but' as those are both valid words. It will find a spelling mistake if you had typed 'bwt' (which may look like but) and will give you a choice of options.

The spell checker will look out for words where capital letters have been put inside the word. For example the spell checker will want to change 'tHEy' to 'they'.

### Spell Checker Commands

There are three commands that start the spell checker.

<b>Spell check the complete document (Function F)</b>	Use the 'Spell Text' command after you have typed in your complete document and you want to spell check everything.
<b>Spell check word to the right of cursor (Control I or Control Q)</b>	Use the 'Spell Word' command while you are typing and you are not sure of the spelling of a particular word. If the spelling is correct, you will get a message on the menu bar, 'Word is in dictionary'.
<b>Spell check the document from the cursor (Control S)</b>	You can check part of a document by using Control S to start checking from where the cursor is and then pressing the Stop key when you have reached the end of the section to check.

remember how you spelt it, you could use 'c???d' which will find any five letter word beginning with a 'c' and ending with a 'd'.

### **Replace (Function 6)**

Replace works just like Find except that you get an additional line 'REPLACE with' asking you what you wish to replace the word or string with that you are searching for. After using this, you will get the same options as described above.

The only point to be wary of is that if you use the 'All' option you need to be sure that the command will do exactly what you think it will.

### **Other Find Commands**

#### **Find Next (Control 6)**

#### **Find Previous (Control 5)**

Once you have used aFind or aReplace, you can use either of these commands to repeat the command without having to enter the search string again.

your 'Find string' section. This is very useful when used with Replace as you can ask for a word without any capital letters and replace it with one that has.

For example, Find 'brighton' and replace with 'Brighton'.

## G-Global

Use Global if you wish to check the complete document rather than just from where the cursor is to either end of the document.

## W-whole word

To restrict your choice during a search even further, choosing this option will only select your match if it is a whole word. For example if you are searching for 'and' the Find command would normally find words like 'sand' as well. Using 'whole word' would only find 'and'.

## n-Nth Occurrence

This command counts the number of matches (between one and 255) it finds before stopping. This could be used for example to search for every second quotation mark.

## Special Characters

When using Find to look for certain characters and printer codes, you need to put an exclamation mark in front of them. This is because these characters are used differently in 'Find' or 'Find and Replace'. Here is a list of these differences.

Printer control codes ! followed by the letter of the printer code.

Question mark	!?
exclamation mark	!!
hard return	!\
soft hyphen	!-
non-break hyphen	!\_
non-break space	! space
search for a specific code	! number

## Using ? in Find

If you are not sure of a letter or a group of letters, you can use the '?' character to indicate the unknown letter. For example, if you wanted to find 'could' but you couldn't

spelt like that. It will not recognise anything that you have spelt differently even if you didn't mean to. It is therefore a good idea, unless you are an immaculate typist, to use this command globally only after you have proofread your document.

---

### Find (Function 5)

When you select this command Protex will ask you to enter the string to search for with the message 'FIND string:'. No, this is not the time to look for the ball of string that the cat was playing with a week ago. You can type in a word, part of a word or even a phrase for Protex to search for.

If you asked it for an 'a' for example, Find would find every single 'a', even those within a word. If you asked it for ' a ', that is an 'a' with a space on either side of it, your choice would be narrower and Find would only find an 'a' on its own. Alternatively, you could choose 'Whole word' which would look for your match in a single word.

Pressing the ENTER key shows you the following options - You can use any combination of these to help narrow down your search.

#### A-All

Normally when Find finds what you have asked for it stops on the first match.

Selecting 'All' will put Find into express mode so that it will not stop until it has found all the occurrences of your search string. Thus Protex counts the number of times your search string appears in the document starting from wherever the cursor was to the end.

#### B-Backwards

Backwards puts this operation into reverse. Instead of looking forward in the document for your search string, Find goes backwards from where you are to the beginning of the document.

Using this option with Find or Find Previous will cause them to do the opposite to what you expect. Find Previous will Find forward and Find will Find backwards. Keeps you on your toes though.

#### C-match Case

Normally Find doesn't care whether what you are looking for is in upper or lower case. If you are particular, use this option and make sure that you have typed the correct case in

<b>Block Delete</b> (Function Del)	Yes, this command deletes the block. If you didn't mean to delete it after all remember the Undelete command Control U.
<b>Block Word Count</b> (Word processor menu, Editing, Count words)	This allows you to count the number of words you have in the marked block.
<b>Copy Block</b> (Function O)	You can copy a block of text from one place in the document to another. The block will be copied to wherever the cursor is when you enter this command.
<b>Copy Block/Document</b> (Word processor menu, Copy Block)	This is different to the copy command available by pressing Function O. This command will copy either your marked block or your whole document to another document.  It is particularly useful for moving sections of text between documents and in fact is the only way to make a copy of a document in the Notepad
<b>Block Print</b> (Word processor menu, Print Block)	Use this command if you only want to Print part of your document.
<b>Block Move</b> (Control M)	This command will move a block to wherever the cursor is when you issue the command.

## Search, Find & Replace

These commands are very useful when writing large documents. You know that you may have covered a particular subject but you can't remember where you did so. Using 'Find' will help you, as you can move forward or backwards within your document looking for every occurrence of a word or phrase.

Replace is really a Find and Replace. You look for a word and replace it with another word. This can be used to change the spelling of somebody's name throughout the whole document. It is useful if you need to change a word several times.

With both these commands the searching starts from where your cursor is when you use the command.

---

### TIP Get the Spelling Right When Using Find and Replace

When looking for a word, part of a word, or a phrase, make sure you spell it the same way you used in the text otherwise Protext will not find it.

The thing to remember with these two commands is that they will only do what you tell them to do. So if you have typed 'Jane' and the correct spelling is 'Jayne', you could find Jane and replace with Jayne. It will only replace the ones that you have

# Chapter 9

## Blocks

One of the most powerful functions of any word processor is the ability to work with blocks. This allows you to mark a bit of text that you have already typed and then do something with it, like deleting, moving or copying it.

As this is a two-stage operation, we will first start by marking the block, then when it is marked do something with it.

### **Mark Block (Function 9 or Control Z)**

There are two different key strokes that do the same thing. You need to mark the beginning and the end of the block of text that you want to do something with. Place the cursor at the beginning or the end of the block, use the command, and then do the same thing again for the other end.

Once you have used the mark block command twice, you will see that the text within the block is now shown as inverted text. At the beginning there is the character [ and the end of the block there is the character ]. You don't need to worry about these characters as they are not actually in your document. They are just used to highlight the beginning and end of the block.

### **Clear Block (Control K)**

If you make a mistake marking a block, the simplest way of putting it right is to clear it by unmarking it and then marking it up again. You can also use this command if you have finished using a block.

### **What You Can do With a Marked Block**

Once you have marked a block, there are six things you can do with it (apart from clearing it as above).



already typed into CAPITALS or visa-versa. Here are the two commands to use.

### **Lower to Upper Case (Control /)**

Move the cursor to the start of the text you wish to change. Hold down the Control key and press the / key until you reach the end of what you wish to change.

### **Upper to lower case (Control \)**

Here it is the other way round as well.

A good way of remembering these commands is with the direction of the line. Lower to upper case starts at the bottom left to the top right. You could imagine that it can make letters larger. The same is true the other way round.

## The Delete Commands

Just like moving the cursor, there are quicker ways of deleting the text in the document than just using the two delete keys.

- Delete line (Control 3)
- Delete to Start of Line (Control <- Del)
- Delete to End of Line (Control Del ->)
- Delete character before cursor (<- Del)
- Delete character at cursor (Del ->)
- Delete word left (Shift <- Del)
- Delete word right (Shift Del ->)

### Undelete (Control U)

There is also an Undelete command that remembers what you have just deleted and puts it back where the cursor is. This helps if you have deleted something by mistake.

When you delete anything (including a 'Block' of text, see later) other than a single character, the text is stored in a temporary memory until you use another delete command. If you leave the cursor alone after you have deleted something, undelete will put it back where it was.

If you move the cursor and then use the Undelete command, what you deleted will be inserted at the new cursor position. You can use this to move or even make several copies of the text that you deleted.

### Special Insertion and Deletion Commands

We have already covered commands that insert and delete lines, (look at your template if you've forgotten them). Here are a few that are slightly different.

#### Transpose Characters (Control A)

Do you ever type letters the wrong way round? Here is a good command to swap these letters around. Try mistyping would as wuold, then move the cursor over the 'u' and do the command above and by magic the uo changes to ou.

#### Changing Cases

There are occasions when you may have wished to either change something that you have

## **Goto Previous Marker (Control Shift 5)**

## **Goto Next Marker (Control Shift 6)**

These two commands allow you to jump to all the markers you have set in order. It is the only way that you can get to your 'Multiple markers'.

A 'bug' in early versions of the Notepad prevented the Right Shift key working with Control 5. This is not a serious problem since it is more convenient to use the Left Shift anyway.

## **Goto Margin Markers (Control @ L or R)**

This moves the cursor to the left or right margin.

---

**TIP Use Goto Left Marker to Move To a New Left Margin**

Control and Left Cursor always moves the cursor hard over to the left of the screen. Control @ L moves the cursor to your left-hand margin. This is very useful when you have redefined your left-hand margin with a new Ruler Line and need to move there.

---

## **Goto Block Markers (Control @ [ or ])**

We haven't covered 'Blocks' yet, but you can use this command in a marked block to goto the beginning (Control @ [) or the end (Control @ ]) of a marked block.

Although there are separate commands to mark a block, the first time that this command is used, it also marks the block. So you have plenty of choice on which key strokes you use.

## **Goto Last position (Control L)**

Protext remembers the last two places that your cursor was within the document. Repeated use of the command Control L will move you backwards and forwards between these two points.

Now you know how to move, let's dance!

Well, we've covered many different ways of moving around the document. Let's look at a different set of commands that actually do things to your document.

If, for example, you wish to go to the top of Page 4, you would type Control GP4.

### Markers

There are several marker commands that allow you to leave bookmarks in your document so that you can quickly return to them at any time. Think of it like a number of differently coloured balls of wool that you can use to find your way back to the correct marker.

There are four types of markers that you can use. They are absolute, multiple, margin and block markers.

When you set a marker it is shown as an inverted number 0 - 9, ?, [ or ] depending on which character you have selected. To remove any markers, use the delete key to delete the inverted character.

---

#### TIP Markers Are Only Shown When Codes Are On

All markers you put in are saved in your document. If you have your 'Codes on/off' set to OFF (Function 4), you will not see your markers even though they are still there. So if you lose them, remember this tip.

---

### Absolute Markers (Control @ 0 to 9)

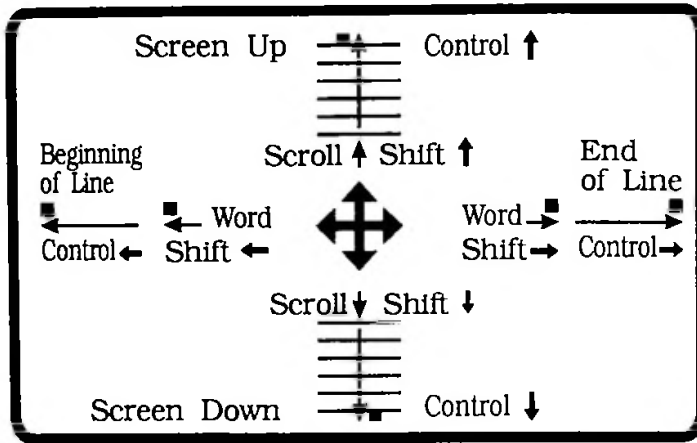
You have ten absolute markers (numbered 0 to 9), which allow you to define absolute positions in your text to which you can return. The first time that this command is used with a number, it sets the marker. Subsequent uses of the same command then return you to that marker.

### Multiple Markers (Control @ ?)

Multiple markers are selected by pressing the '?' after Control @. You can have any number of these markers within your document.

You can use multiple markers to allow you to move quickly to common points in your document. For example, if you added a multiple marker at the start of every main paragraph, the commands detailed below would allow you to skip through the document a paragraph at a time.

## Cursor Screen Commands



If you know where you want to move to there are commands that will help you get there a great deal faster than by just using the cursor keys.

## Page Mode (Control Shift P)

Page mode will change the information displayed on the Status Bar. Normally the Status Bar shows where the cursor is by Page No, Line No and Column. Page mode changes it to Character Number, Global Line number and Column. In this mode you can't move forward and backwards a page with Control 9 or Control 0 command.

You can use it to show the global line number for use with the following command.

## Goto Line / Page / Column (Control G Ln / Pn / Cn)

If you know where you wish to move to in your document, 'Goto' will help you move around your document quickly. It will either move your cursor to the top of the page number you have asked for, or to the line number or column on the page you are now on.

The line number is not normally displayed on the Status Bar but it can be displayed by turning off the Page mode.

## Horizontal Scrolling

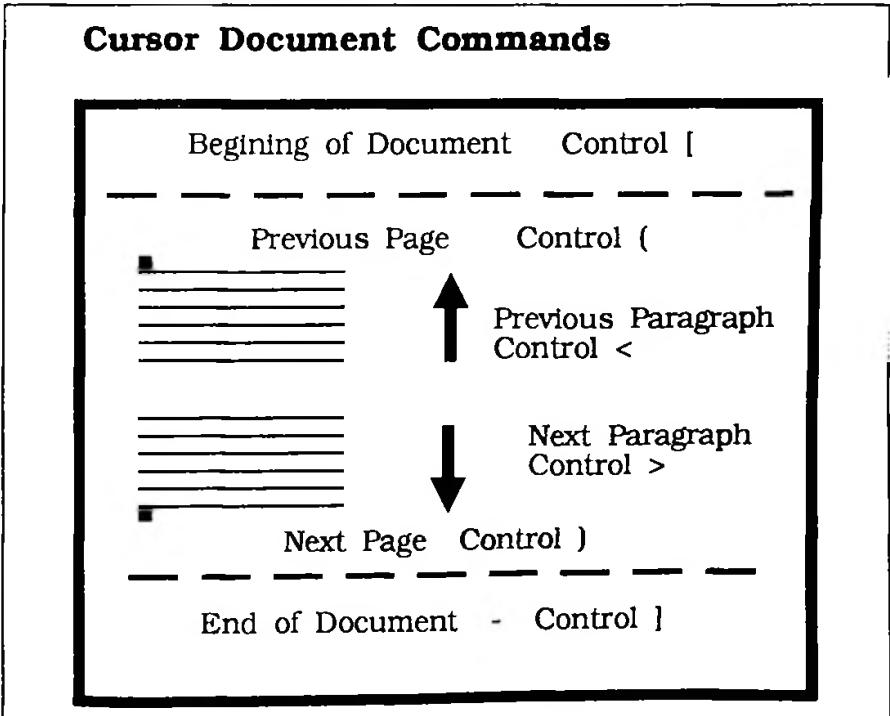
This type of scrolling is not used as often as vertical scrolling because the default ruler is set to the width of the screen. With Word Wrap ON, the text stays within the screen size.

When the document is wider than the screen, or Word Wrap is turned OFF, then horizontal scrolling will take place. If you find that you lose the left-hand side of your text, this is due to horizontal scrolling. To get back to the left-hand side again use the Shift and Enter keys.

## Turbocharging the Cursor Keys

When a document becomes larger, having to move the cursor a single line up or down and a single character left or right becomes tedious. Is there a faster way? Fortunately the answer is yes.

The diagrams below show both the cursor movements on the screen as well as moving around the document.



b	Bold	q	Quality
c	Condensed	s	Subscript
e	Elite	t	superscriptT
i	Italic	u	Underline
l	enLarged		
p	Proportional		

## View Codes On/Off (Control V V)

When you use these 'Style attributes' you have a choice about how they are displayed on the screen. With the View codes turned on, if you select underline for example, you will see an inverted 'u' at the beginning and end of the text you have marked for underlining. If you turn the View codes off, you will see the text underlined on the screen. The Codes on/off command is on the template. Press Function 4 to toggle this command.

## Scrolling

As the display is normally smaller than the document you are typing, Protect needs to scroll your text so that you can see what you are working on.

There are two types of scrolling that Protect uses. These are Vertical Scrolling and Horizontal Scrolling.

### Vertical Scrolling (Shift Cursor Up or Cursor Down)

As you type more and more lines of text into your document any text above the line you are on will be moved up the screen until it disappears off the top. This effect is called 'Vertical' scrolling.

You can use the Up/Down Cursor keys to scroll through the entire document. The cursor can move to lines that have already been scrolled of the top or bottom of the screen. You can move the text on the screen so that you can see any point in your document.

Sometimes, it is useful to leave the cursor where it is but scroll the text up or down so that you can read the text surrounding it. You can do this by pressing the shift key with the up or down cursor keys.

# Chapter 8

## Using the Style Attributes

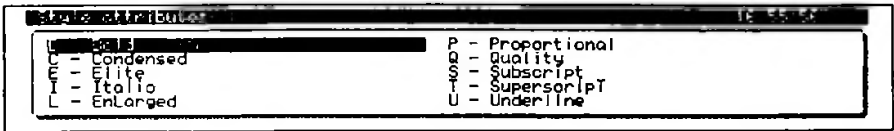
Using the what? On the Notepad, what we would have called print commands are called 'Style Attributes'. These are things like underlining, bold, italics and so on that can be used to make your document look more professional.

Normally when you start printing text, it is in ordinary text. This is technically known as 'pica' and is ten characters per inch. Whether you can use other types of print depends on what type of printer you have. We are going to assume that you have a printer that can print all these for now.

### Selecting Bold, Underlining or Italic Print (Control 7, 8, -)

You select a Style Attribute by pressing the relevant key sequence at the beginning and end of the text you want the attribute to apply to. Some of these marks can be selected two different ways. These attributes, like Bold, Underline and Italics are shown on the template. Use the shown key sequences (Control 7, 8 and -) to select them quickly.

These and all the other attributes are in the Style Attributes menu. You can see a list of all the possible attributes by pressing the Menu key followed by S.



These are, Bold (emphasised), Condensed print, Elite (twelve characters per inch), Italic, EnLarged, Proportional, Quality (near letter quality NLQ), Subscript, Superscript and Underline.

### Short Cut Style Attributes (Control X)

Alternatively there is a short cut key sequence for selecting these attribute codes. It is designed for those who have good memories. Use Control X followed by one of the following letters:



## Using Hyphens

When a hyphen '-' is used between two words in the text, if the hyphen falls near the end of the line it will normally be used to split the line with a 'Soft Return'. You may not want this to happen, so there are some special ways how you can change this.

### Non-break Hyphens (Control N -)

You can use a hyphen to glue two words together in a visible form. Do this by using a 'Non-break Hyphen'. To select this character, use Control N followed by a hyphen '-'.

### Soft Hyphens (Control H)

Sometimes, it is quite acceptable to split a hyphenated word across two lines. Unfortunately, the word processor doesn't normally do that because it can only 'wrap' whole words between different lines. You can use Soft Hyphens to tell the word processor where it can split the word in two.

The good thing about a soft hyphen is that you can't see it if the word is anywhere on the line except the very end. It only becomes visible when the word needs to be split between two lines.

To select this character, use Control H. The hyphen will be shown inverted on the screen but will only be printed if it is at the end of the line.

# Special Formatting Characters

There are some special formatting characters that are used with the word processor.

## Soft and Hard Returns

There are two types of carriage return characters that Prototext uses to mark where the end of line is. They are soft and hard return characters.

If you turn Word Wrap ON, the word processor is constantly formatting your text. When you reach the end of one line, it inserts a Soft Return character. It needs this character so that if you decide to change the length of the line and reformat the text, it knows which character it should change.

This is different to the character that is used when you press the ENTER key. This is called a 'Hard Return' and is used to mark the end of a paragraph or a line in a list.

These characters are not normally displayed but they can be shown on the screen by selecting View Tabs and Returns (Control V T).

## Non-break space (Control N space)

You can use this character to 'glue' two words together with a space. This stops the two words from being split across two lines. It also prevents any 'soft spaces' being added between the words if you have selected the Right Justification command. You can use non-break spaces, for example, when typing a sentence or word that you want's p a c e d o u t'.

To select this character, use Control N followed by a space.

You can also select this command from the Text Formatting menu.

## View Hard Spaces (Control V S)

Hard spaces are put into your text when you press the space bar. You normally see them just as spaces between words but they are an actual character.

To display the hard spaces as a character, use this command. It is a toggle, so you can change it back with the same command.

## **Formatting Whilst Printing On/Off (>FP ON/OFF)**

You use this with the Right Justification command if you want your text to be formatted during a Mail Merge.

You must remember to turn it OFF at the start of any tables, however, and ON again at the end of the table. If you forget you could find that reformatting has destroyed your table's layout.

## **Right Justification On/Off (>RJ ON/OFF)**

This command works in the same way as Control J apart from the fact that it will re-justify during printing. It will only take effect if you have included the >FP command to turn on formatting while printing. The default value is the same setting as in the Configure menu within Protex.

## **Centre Line (>CE'text')**

Use this command instead of Control C if you wish to centre your text when printing.

# **Special Hidden Characters**

There are other characters that you can use that tell Protex to do different things when formatting the text.

## **View Tabs and Returns (Control V T)**

You can display these characters by pressing the Control V and T keys. Pressing them again will turn it off.

## **Insert Space (Control Space)**

When you are in 'Overtyping' mode, you can use this command to insert spaces into your document.

It may sound crazy to have two different ways of moving to the next page but there are circumstances where this is necessary. Suppose you wanted to print address labels. Typically there are at least eight labels on each sheet of paper. Rather than printing just one address on each sheet, you can tell the word processor that the page is very short. It will think that it is moving to a new sheet of paper after each page but in fact it is only moving down your single sheet. The printer will be able to move to the next sheet after your eight labels have been printed.

To test the printer's paper length setting put a sheet of paper into the printer then, with the printer 'off-line', press the formfeed button. If the paper length setting is correct the printer will fully eject one sheet of paper and feed the next sheet through ready to start printing on the first line.

If the next page is not at the top, or the previous page has not been ejected, then you have the page length set incorrectly. Remember, for A4 paper the page length is 11 - 2/3 inches.

---

## Formatting Commands used with Mail Merge

There are several commands that are normally only used when mail merging. Mail merging is a topic dealt with in a separate chapter but we have included the formatting commands here for completeness.

When using mail merge words and extra lines are inserted into the text during printing. The final layout will not therefore always be the same as that displayed on the screen. Using the Formatting, Centre Line and conditional Page Throw commands that work while printing is in progress, you can make sure that the printout stays the way you want it.

Most of the formatting commands that we have already covered act immediately on the text as you type it in. If you type a line of text and then issue the command to centre it (Control =) the text is immediately centred. If you go back to that line and edit the text, the line will no longer be centred. You would have to issue the command again.

The commands to format while print behave differently. There are only actioned at the time of printing. Using the same example of centring text, if you included the command >CE above the line to be centred, Protext would perform the centring at the time the line was printed. Therefore you could edit the text to your heart's content, knowing that it would always be centred on the printout.

together, putting the command >PA 25 just above it would ensure that it would all stay together on one page. The Conditional Page Throw will only have an effect if there is not enough room left for the whole paragraph. It will start a new page at the beginning of the paragraph.

## Form feeds Enable/Disabled (>FF ON/OFF)

The form feed character tells your printer that it has reached the end of the page and it must move to the top of the next one. This is a useful command if you are using either a laser printer or a printer with a sheet feeder. The default setting is OFF.

---

### TIP - Help With Page Lengths and Formfeeds

There are some tell tale signs that point to an incorrect setting for page length and/or formfeed commands. Let's try to understand what the printer and your word processor are trying to do.

The important point to remember is that the word processor only knows about the page length. This is the number of lines of text that will fit on a page. The printer, however, knows about the page length and the paper length. The paper length is the physical length of each sheet of paper. The page length and the paper length are not necessarily the same.

There are two ways that the word processor can tell the printer to move to the next page.

One way is to give the printer multiple Line Feed characters. These add the correct number of blank lines to finish one page and move onto the next. For this to work correctly, the page length that Protex is using must agree with the actual length of the paper. If the two values don't agree the first line of text on a new page won't start at the top of a new sheet of paper.

The other way for Protex to move to the next page is for it to send the printer a Form Feed character. Form Feed characters tell the printer to move onto the next page. In this case the printer has control of the page length. For this to work correctly the printer needs to know the length of the paper you are using.

You may need to consult your printer's manual to find out how to change the paper length. Normally you can either change the printer's default paper length setting with switches/buttons, or you can send the printer a special command from the word processor (see Outputting codes to the printer).

In either case the user is prompted for the room number and the floor number. If a message is to be stored it is then typed in. If the message is being retrieved it is printed to the screen. The program then returns to the main menu screen.

What we wish to do now is to save the messages to a file rather than to an array. This will allow the user to stop and start the program whenever he wishes while maintaining all the saved messages. In fact we need to make very few changes to the original program to achieve this new feature - one of the main benefits of a structured approach to program writing.

We have listed the full example below for clarity, although many of the original procedures did not change. However to make the use of the 'PTR' command more general we have used a command not yet covered.

## FN

'FN' defines a new function for BASIC to use. You include the program to perform the function within your program. Functions are closely related to procedures in the way they operate. They both do very similar jobs. Indeed in many cases you could use either to carry out a particular task. The differences are mainly to do with how each command is used. A function is able to return a value to the BASIC line that called it; a procedure cannot. Typically a function is a short piece of program code, often only one line long, that does a very specific and frequently required task. A procedure is typically longer and may control a variety of tasks.

BASIC already has a wide range of built-in functions: 'LEN', 'ASC' and 'VAL' are examples. With the 'FN' command you can add as many new functions as you wish.

For example, suppose your program had to frequently determine the average of two numbers. It is a simple calculation that you could include in your code whenever you needed it, but if you defined a function to do the job you would just call the function each time.

```

10 REM using a new function
20 REPEAT
30   INPUT "Enter two numbers ", num1, num2
40   PRINT "The average is ", FN_average(num1,num2)
50 UNTIL FALSE
60 END

100 DEF FN_average(n1,n2) = (n1+n2)/2

```

Line 40 calls the function within a print statement. When BASIC processes this line it passes the value of the variables 'num1' and 'num2' to the function and prints whatever

the function returns.

Line 100 defines the function. This definition can appear anywhere within the program and can be called any convenient name. The definition tells BASIC what the function expects to receive, in this case two numbers, and what it will return. The function returns whatever is to the right of the equals sign. In this case that is the average that we want.

### Reentrant Functions

BASIC has to be told about any parameters that are passed to a function. These parameters are called formal parameters and are local to the function. Because the parameters are only known to the function, the function can call itself.

The easiest way to think of this feature is to imagine that each time the function is called a new copy of the function and all the local variables it uses is created. The new copy of the function does not interact with the old copy in any way. You can have as many copies of a function active at the same time as you like but of course you do have to finish running all the copies otherwise you can produce some very interesting bugs.

We covered some of the advantages of using local variables earlier but here is an example where local variables and the ability of a function to be called recursively are essential to allow a simple function to perform a complex task. Consider the task of converting an integer into a string.

Why do we need a recursive program to convert an integer to a string? Suppose we want to change the number '123' to a string. If we do the following calculation:

$$123 \text{ MOD } 10$$

we would get the answer of '3' (the remainder after 10 has been divided into 123 as many times as possible). This is one of the digits we want so we can add it to our string. If we then divide '123' by '10' and do the same trick:

$$12 \text{ MOD } 10$$

we would get the answer '2' which can again be added to our string. Repeat the procedure again to arrive at:

$$1 \text{ MOD } 10$$

and add '1' to our string. That all sounds great and has produced the three digits '3', '2' and '1' for us to add to our string. But they are in the wrong order. We need to generate the digits backwards.

```

10 REM using a recursive function
20 INPUT "Enter an integer ",anything
30 digit = FN_convert(anything)
40 our_string$ = our_string$ + CHR$(digit)
50 PRINT "Converted to a string ";our_string$
60 END

100 DEF FN_convert(a)
110 IF INT(a/10) THEN digit = FN_convert(INT(a/10)):our_string$
    = our_string$ + CHR$(digit)
130 =(a MOD 10) + 48

```

The formal parameter passed to a function is local so each time the function is called it has its own copy of the variable. This allows a function to call itself as we have done here. The easiest way to see how `_convert` works is to trace through an example.

Suppose you entered the number '123' in line 20. The first time `_convert` is called it is given '123' as its parameter. When this is divided by '10' in line 110 the result is not zero. As far as BASIC is concerned this means that it is TRUE so the THEN part of the IF statement is followed. This calls `_convert` again but now with a parameter of '12'. The second copy of `_convert` does the test in line 110 and again follows the THEN part of the IF statement and calls `_convert` with the parameter of '1'. The test in line 110 of the third copy of `_convert` will fail since the integer of '1' divided by '10' is zero. Hence the third copy of `_convert` will return to its caller in line 130 and pass back the number '49'. This is the ASCII CODE for the character '1'.

We are now back in the second copy of `_convert` which can add the ASCII CODE for the character '1' to the string variable `'our_string$'` and return to its caller passing the number '50'. This is the ASCII CODE for the character '2'.

Back in the first copy of `_convert` it can add the ASCII CODE for the character '2' to the string variable `'our_string$'` and return to its caller passing the number '51'.

We have wound our way back to the main program which can add the ASCII CODE for the character '3' to the string variable and print it.

Using this function we have added the three characters to our string in the correct order. The concept of reentrant functions can be difficult to grasp at first which is why we have chosen a simple example. Indeed BASIC will do precisely the same job as our function `_convert` with the command `STR$`.

The potential for a function to be able to call itself recursively can be extremely useful when handling complex calculations. The only real pitfall to watch out for is to make sure that all the copies of a function that you start are able to complete.



Back to our reception desk. In the program listing below we have only explained the changes from the original version. Hopefully you are now familiar enough with the commands to be able to follow the rest of the program flow. On the way through we have added a few new commands which are explained as we go.

```
10 REM Hotel Reception Desk Message Saver Version 2
20 PROC_welcome_screen
30 PROC_initialise
40 the_end = FALSE
50 REPEAT
60   PROC_get_decision
70   ON decision PROC_do_input, PROC_do_output
80 UNTIL the_end = TRUE
90 END
```

Line 70 has the new command 'ON'. This simply allows a switch between different subroutines or procedures. If the variable used with 'ON' has the value of '1', BASIC will go to the first procedure listed. If the variable has the value of '2', BASIC will go to the second procedure listed. You can include as many procedures in your list as you wish, up to the maximum linelength of 255 characters.

If the variable has a value which is higher than the number of procedures in the list, BASIC will produce an error.

You can trap this possibility by adding an 'ELSE' part to the 'ON' command which would be executed if BASIC cannot match the variable value to the list.

```
10 ON decision GOSUB 100,2000,350,100 ELSE PRINT "Wrong value"
```

If 'decision' had the value '1' the subroutine at line 100 would be called. A value of '3' would cause the subroutine at line 350 to be called. Notice that the routines don't have to be different for each value of the variable. In this example if the variable is either '1' or '4' then the subroutine at line 100 is called. If 'decision' holds any number other than 1 to 4 our own error message is printed rather than BASIC's error message.

'ON' can be used with 'GOTO', 'GOSUB' or 'PROC'. For the reasons already highlighted we generally stay with just using 'ON' for procedure calls.

```
1000 DEF PROC_initialise
1010 handle = OPENUP("MESSAGE")
1020 IF handle = 0 THEN PROC_make_file
1030 PRINT TAB(19,4) """;
1040 PROC_get_enter
1050 ENDPROC
```

We have had to re-write our initialisation procedure. Previously we just needed to

dimension an array large enough to hold all the messages. Instead we now have to open the file where the messages have been kept. We also need to be aware of the fact that the first time this program is run there won't be a file to open! Line 1010 tries to open our file. If it fails the variable 'handle' will contain the value '0' in which case we call another procedure to create the file for us.

```

1100 DEF PROC_make_file
1110 handle = OPENOUT("MESSAGE")
1120 m$=STRING$(40," ")
1130 FOR floor = 1 TO 4
1140   FOR room = 1 TO 20
1145     REM step through each entry in our file putting
           something into the record
1150     PTR#handle=FN_make_ptr
1155     REM mark this entry as empty
           (ASCII CODE for 'E' is 69)
1160     BPUT#handle,69
1165     REM now write a blank record
1170     PRINT#handle,m$
1180   NEXT room
1190 NEXT floor
1200 CLOSE#handle
1205 REM open the file ready for updating
1210 handle = OPENUP("MESSAGE")
1220 ENDPROC

```

The job of this new procedure is to create an empty file for us the first time we run the program. It is important that the file is in the correct format to store the messages. The format we have chosen is straightforward.

Each room on each floor will have its own record in the file. Each record will consist of two parts; a flag to say whether or not there is a message and then the message itself. Note that even when there is no message, the record must still be there otherwise we wouldn't know where the next record started. We have used the character 'E' to mean that the record is empty. Later we will use 'F' when we write a message to it.

Line 1150 sets the file pointer to the correct place for us. Strictly speaking we don't need to call this function here since we are writing to the file sequentially and thus the pointer would always be in the correct place. However for consistency we have called it anyway.

```

1300 DEF FN_make_ptr
1310 =(42 * (room - 1)) + (20 * 42 * (floor - 1))

```

This function has to return a value to point to the start of the record for any room on any floor. Where do the magic numbers come from? To find out we need to analyse the record a little more.

We have said that we are using a one character flag to show whether the record has a message or not. We have set an arbitrary limit on the length of any message of 40 characters. When BASIC prints a string to a file it always includes an extra character to mark the end of the string. So one record will consist of a one character flag, a forty character message and a one character end-of-string marker. Hence each record will be 42 characters long.

The record for room 1 on floor 1 will start at pointer position '0'. The record for room 2 on floor 1 will start 42 characters later at pointer position '42'. Likewise the record for room 3 on floor 1 will start at pointer position '84'.

So if we only had one floor to think about we could calculate where the record for a room started by taking 1 from the room number and multiplying by the record length.

```
file pointer = 42 * (room - 1)
```

However this would only work for the first floor. If the room we need is on the second floor we need to add one floor's worth of records as well. Since there are 20 rooms on each floor we can move the pointer by one floor's worth of rooms thus:

```
file pointer = 20 * 42 * (floor - 1)
```

Hence the position of any room on any floor will be given by the formula:

```
file pointer = (42 * (room - 1)) + (20 * 42 * (floor - 1))
```

This is the formula that FN\_make\_ptr uses.

```
2000 DEF PROC_welcome__screen
2010 PROC_banner
2040 PRINT TAB(28,2) "VERSION 2"
2060 PRINT TAB(24,4) "please wait....."
2070 ENDPROC

2100 DEF PROC_banner
2110 CLS
2120 PRINT TAB(15,0) "HOTEL RECEPTION DESK MESSAGING SYSTEM"
2130 ENDPROC

3000 DEF PROC_get_decision
3010 got_decision = FALSE
3020 REPEAT
3025 PROC_banner
3030 PRINT TAB(12,2) "Please enter requirement      Input message - 1"
3040 PRINT TAB(39,3) "Print message - 2 ";
3050 PROC_get_digit
3060 IF this_digit = 49 OR this_digit = 50 THEN got_decision = TRUE
3070 UNTIL got_decision
```

```

3075 REM make sure we return either 1 or 2
3080 decision = this_digit - 48
3090 ENDPROC

4000 DEF PROC_do_input
4005 PROC_banner
4010 PRINT TAB(22,2) "INPUTTING A MESSAGE"
4020 PROC_get_room
4030 PROC_get_floor
4040 PRINT TAB(16,6) "What is the message";:INPUT m$
4045 REM the message must be 40 characters long
4050 m$=LEFTS(m$+STRING$(40," "),40)
4055 REM move the file pointer to the start of this record
4060 PTR#handle=FN_make_ptr
4065 REM mark this record as having a message in
4070 BPUT#handle,70
4075 REM write the message to the file
4080 PRINT#handle,m$
4090 PRINT TAB(7,7) "The message has been stored - ";
4100 PROC_get_enter
4110 ENDPROC

```

There are a few changes to this procedure to cope with using a file. Line 4050 makes sure that the message is always exactly forty characters long. This is necessary to ensure that we can always go to the start of a record. The line works by first adding forty spaces to the end of the message and then uses the first forty characters of the new longer message.

Line 4060 puts the file pointer to the start of the correct record for this message. Line 4070 writes the character 'F' (ASCII CODE 70) to the flag and then line 4080 stores the message. Note that we have not checked to see if a message was already there - we just overwrite any existing message.

```

5000 DEF PROC_do_output
5005 PROC_banner
5010 PRINT TAB(22,2) "OUTPUTTING A MESSAGE"
5020 PROC_get_room
5030 PROC_get_floor
5035 REM point at the record for this room
5040 PTR#handle=FN_make_ptr
5045 REM get the flag
5050 empty=BGET#handle
5055 REM get the message
5060 INPUT#handle,m$
5070 IF empty = 69 THEN PROC_no_message ELSE PROC_message
5080 ENDPROC

```

We have made similar changes to the procedure for retrieving a message. Line 5070 checks the value of our flag, 'empty', to see if there is a message for this guest and calls the relevant procedure accordingly.

## Chapter 21 Basic

```
6000 DEF PROC_get_digit
6010 got_digit = FALSE
6020 REPEAT
6030   key = INKEY(0)
6040   IF key > 47 AND key < 58 THEN got_digit = TRUE
6050 UNTIL got_digit
6060 this_digit = key
6070 ENDPROC

7000 DEF PROC_get_room
7010 valid_room = FALSE
7020 REPEAT
7030   PRINT TAB(4,4) "What is the guests room number      "
7035   INPUT TAB(35,4),room
7040   IF room > 0 AND room < 21 THEN valid_room = TRUE
7050 UNTIL valid_room
7060 ENDPROC

8000 DEF PROC_get_floor
8010 valid_floor = FALSE
8020 REPEAT
8030   PRINT TAB(20,5) "On which floor          "
8035   INPUT TAB(35,5),floor
8040   IF floor > 0 AND floor < 5 THEN valid_floor = TRUE
8050 UNTIL valid_floor
8060 ENDPROC

9000 DEF PROC_no_message
9010 PRINT TAB(4,6) "There is no message for this guest - ";
9020 PROC_get_enter
9030 ENDPROC

10000 DEF PROC_message
10010 PRINT TAB(5,6) "The message for this guest is: ";m$
10020 PRINT TAB(5,7) "Do you want to delete it (Y/N) ";
10030 IF FN_get_y_n = 78 THEN ENDPROC
10035 REM put the pointer back to the start of this record
10040 PTR#handle=FN_make_ptr
10045 REM mark the record as empty
10050 BPUT#handle,69
10060 ENDPROC
```

The only necessary change to this procedure was to line 10010 - instead of printing the contents of an array element it now has to print the string read from our file. However we have also taken the opportunity of allowing the user just to look at a message without deleting it. If the function 'get\_y\_n' returns 'N' (ASCII CODE 78) then we end the procedure without deleting the message.

If the user pressed 'Y' in answer to our question then we only have to mark the record as empty - we do not need to overwrite the message.

```

10100 DEF PROC_get_enter
10110 PRINT "Press the ENTER key when ready";
10120 REPEAT UNTIL INKEY(0) = 13
10130 ENDPROC

```

We have not changed the functionality of this procedure but we have shortened it by doing all the work in line 10120. BASIC will just keep repeating this one line until the value returned by 'INKEY' is equal to '13' (the ENTER key).

```

10200 DEF FN_get_y_n
10210 LOCAL answer
10220 REPEAT
10230   answer = INKEY(0)
10240   IF answer > 96 AND answer < 123 THEN answer = answer - 32
10250 UNTIL answer = 78 OR answer = 89
10260 =answer

```

This is the last new function. It is designed to return either 'Y' or 'N' depending on which key the user presses. The program flow will be familiar to you: line 10230 gets a key; line 10240 changes it to upper case if necessary and line 10250 checks that it is one of the two keys we want.

We have defined the variable 'answer' as local because it will never be used outside this function. Line 10260 returns the answer directly to the caller.

We have now removed most of the criticisms of our Hotel Reception Desk program. It is not yet ideal. There is still the possibility of the user getting unhelpful error messages by pressing the wrong keys; we still only allow one message per guest; there is no way to scan through all the stored messages. But the program would be usable and as you have seen it could always be extended in the future.

## Pretend Files

As far as BASIC is concerned files don't have to be documents held within the Notepad. They can equally well be either the serial communications port or the printer port.

If you entered the following program:

```

10 REM send characters to the printer
20 any_name = OPENOUT("LPT:")
30 REPEAT
40   INPUT a_string$
50   PRINT#any_name,a_string$
60 UNTIL FALSE

```

any data that you typed on the keyboard would be printed to your printer. Likewise

opening a file called "COM:" would send the data to the serial port.

It would make little sense to try to open the printer port for input but it does make sense to do so for the serial port. Indeed this is by far the easiest way of allowing BASIC to handle the port if your application requires it.

## Handling Errors

BASIC will produce many different types of errors depending on exactly what has gone wrong. The types of errors you will be most familiar with are those that BASIC produces when you use the wrong command or when you miss-type a command. The occurrence of these errors will reduce as you learn more about BASIC and as long as you test your program thoroughly they will not occur at all once the program is finished. However there is always a possibility of BASIC producing some types of errors if the user enters the wrong information. You have no way of stopping these - they depend on the user - but you can trap and deal with them from within your program.

### ON ERROR

The command 'ON ERROR' stops BASIC putting its own error message on the screen when something goes wrong. It allows your program to take control and print a message that is relevant to the particular situation.

The command doesn't 'do' anything in itself. But BASIC remembers that the command was present in the program and then, if an error occurs, BASIC goes to wherever the 'ON ERROR' command directed.

```
10 REM using ON ERROR
20 ON ERROR GOTO 100
30 INPUT "Enter a number ";a_number
40 ON a_number GOTO 50,60,70
50 PRINT "Seen 1":GOTO 30
60 PRINT "Seen 2":GOTO 30
70 PRINT "Seen 3":GOTO 30

100 IF ERR = 17 THEN END
110 PRINT "Error line ";ERL
120 PRINT "Error number ";ERR
130 GOTO 30
```

The command in line 20 tells BASIC that if an error should occur it must jump to line 100. As long as you type in the numbers '1', '2' or '3' when requested the program will just print them to the screen and ask for the next number. But when you type a number outside this range or you type a letter the program will jump to line 100 and start running code from there. In our case we have just printed the line number that caused the error

using the function 'ERL' and then printed an error number using the function 'ERR'. The test in line 100 makes sure that we can stop the program when we want to. (ERR is equal to 17 when you press the 'STOP' key - see the list below)

The two new functions, 'ERL' and 'ERR' allow your program to find out what caused the error. You could then decide on what action was necessary. This may just be a case of printing a relevant message for the user and allowing him to retry an operation or maybe you need to perform a calculation differently. If the error is such that the program cannot run you can at least stop with a sensible message on the screen.

The possible error numbers that the function 'ERR' may return are as follow:

1 Out of range	23 Accuracy lost
4 Mistake	24 Exp range
5 Missing ,	6 No such variable
6 Type mismatch	27 Missing )
7 No FN	28 Bad HEX
9 Missing "	29 No such FN/PROC
10 Bad DIM	30 Bad call
11 DIM space	31 Arguments
12 Not LOCAL	32 No FOR
13 No PROC	33 Can't match FOR
14 Array	34 FOR variable
15 Subscript	36 No TO
16 Syntax error	38 No GOSUB
17 Escape	39 ON syntax
18 Division by zero	40 ON range
19 String too long	41 No such line
20 Too big	42 Out of DATA
21 -ve root	43 No REPEAT
22 Log range	45 Missing #

You can have more than one 'ON ERROR' within a program. BASIC will action the most recent command if an error occurs. This allows you to trap different types of errors in different places in your program. For example, any error caused by insufficient space while you are initialising your program effectively prevents the program from running. But once your initialisation is finished you are only interested in those errors that could happen due to bad input from the user. So you may well have the first 'ON ERROR' at the start of your initialisation and then a second 'ON ERROR' when you've got going.

If you want to stop the action of BASIC's error trap you just include the command 'ON ERROR OFF' and BASIC will return to handling any errors itself.



## REPORT

It is unlikely that your program will be able to cope with most types of errors - those caused by faults in the program for example. In this case you just need to tell the user that something has gone seriously wrong and to stop. 'REPORT' will print the message that BASIC would have produced had you not had the 'ON ERROR' command.

```
1000 REPORT
```

## Avoiding the Errors in the First Place

Error trapping can be useful under some circumstances but in practice it is far better to write your program in such a way that errors cannot occur. The main problem with 'ON ERROR' is that it is line number dependent. We discussed the problems that this brings when we covered the 'GOTO' command. In particular with this command you can produce some wonderful bugs by leaping off to process an error and then returning to the wrong place in your program.

As long as you take care when designing your program you can effectively eliminate the possibility of errors while the program is running.

As an example, you can always make sure that you never get a 'Division by zero' error:

```
10 REM this would error if the second number was zero
20 INPUT "First number ";first_number
30 INPUT "Second number ";second_number
40 PRINT first_number/second_number
50 END

10 REM this doesn't
20 INPUT "First number ";first_number
30 INPUT "Second number ";second_number
40 IF second_number = 0 THEN PRINT "no zeros here" ELSE
    PRINT first_number/second_number
50 END
```

## Wrapping up the Commands

There are a few commands left that don't fall conveniently into any of the subjects so far covered.

## TIME

The value of 'TIME' is incremented 100 times each second. You can therefore use it either to wait for a specific period or to time an event. For example if you wanted a program to pause for one second between displaying two messages on the screen:

```
10 PRINT "This message is printed immediately"
20 TIME = 0
30 REPEAT UNTIL TIME > 99
40 PRINT "This message is print 1 second later"
50 END
```

You have to set TIME equal to a known value to begin with although it doesn't have to be zero. We could have started by setting it to '5000' and waited until it reached '5100'.

Line 30 will keep repeating until the value of 'TIME' is greater than '99'. The use of the operator 'greater than' is important. If you checked for 'TIME' being equal to a number, the test might always fail because the loop may take more than one hundredth of a second to run. In this example the loop is very short so there would be no problem testing for equality, but consider the following:

```
10 REM a long timed loop
20 TIME = 0
30 REPEAT

.. lots of program code

200 UNTIL TIME = 100
210 END
```

Here we have set up a similar loop to the first example except that we have assumed that the program has more code to run inside the loop. How could the timer fail to be equal to 100 in line 200? Suppose that on the last but one go round the loop the value of 'TIME' was '99'. The loop would be run again. But what if the loop takes longer than one hundredth of a second? The next time that line 200 is executed the value of 'TIME' may be '101', causing the loop to be run again. In fact the program may never finish. If we change line 200 thus:

```
200 UNTIL TIME > 99
```

the loop will always finish correctly.

## TIMES

This is a similar command to 'TIME' except that it deals with real time. It returns the time and date that the Notepad is set to or it allows you to change the setting of the Notepad time and date.

The format of the string is as follows:

```
Sun.25 Oct 1992,11:55:36
```

You can use the string functions to split up TIMES if you only want part of it.

```
10 now$ = TIMES
15 REM print the hours and minutes
20 PRINT "The time is ";MID$(now$,17,5);
30 month$ = "JanFebMarAprMayJunJulAugSepOctNovDec"
35 REM split out the text of the month
40 text$ = MID$(now$,8,3)
45 REM find the text month in all months
50 place = INSTR(month$,text$)
55 REM convert the position in the string to numeric month
60 the_month = INT(place/3) + 1
65 REM create the date as a string
70 date$ = MID$(now$,5,2) + "/" + STR$(the_month) + "/" +
      MID$(now$,14,2)
80 PRINT " on ";date$
90 END
```

Line 50 uses INSTR to find where the text month returned by 'TIMES' is in our list of all the months. Line 60 then converts this to a number in the range '1' to '12'. Hence the result of running the program is to print the time and date in the more usual format of:

```
The time is 11:55 on 25/10/92
```

You can set 'TIMES' equal to a string that you have formed in order to change the time and date within the Notepad but you need to be careful to ensure that your string is in the correct format.

```
10 new_time$ = "Mon.26 Oct 1992,12:03:27"
20 TIMES = new_time$
30 END
```

## SOUND

This command makes a sound on the Notepad's speaker. The Notepad has two sound channels that can be played simultaneously. You need to give 'SOUND' four parameters

which have the following possible ranges:

SOUND channel, volume, pitch, duration

channel	must be '1' or '2'
volume	ignored so just use '0'
pitch	from 4 to 256 in steps of 4 (100 is middle c)
duration	-1 to 32768

The duration is measured in twentieths of a second so a value of 10 would last for half a second. If you give a channel a duration of '-1' it will continue playing the note indefinitely. Thus you can play two notes together by starting one channel off with a duration of '-1', starting the other channel off for the required duration and then stopping the first channel by giving it a duration of '0'.

```
10 SOUND 1,0,100,10
```

would play middle 'C' on sound channel one for half a second.

```
10 SOUND 1,0,100,-1
20 SOUND 2,0,148,40
30 SOUND 1,0,0,0
```

would play two 'C's one octave apart for two seconds.

## CLEAR

Whenever you start to 'RUN' a BASIC program all the variables are cleared to either zero, in the case of reals and integers, or a NULL string in the case of strings. It is occasionally useful to be able to do the same task from within your program and this is precisely what the command 'CLEAR' does. It effectively resets BASIC to the state it was in when you started the program.

## OSCLI

This command, which stands for Operating System Command Line Interpreter (bet you're glad we told you that) allows you to execute some Notepad commands directly from BASIC. The command:

```
10 OSCLI("BYE")
```

would leave BASIC and return you to the main menu of your Notepad. The possible commands are:

## Chapter 21 Basic

BYE QUIT	both return you to the Operating System
CAT DIR	both list all the stored files
DELETE filename ERASE filename	both delete the specified file
ESC ESC ON	both turn on the action of the STOP key (this is the normal state of affairs)
ESC OFF	causes the STOP key to return the ASCII CODE 27 to a program instead of stopping the program
EXEC filename	reads a file as though it were being typed on the keyboard
KEY n string	redefines a key - see explanation below
LOAD filename aaaa	loads the specified filename into memory at the hexadecimal address aaaa
PRINTER n	selects the printer as parallel if n = 0 or serial if n = 1
RENAME old, new	renames oldfile to newfile
SAVE filename aaaa bbbb	saves an image of memory to a file starting at address aaaa and ending at address bbbb
SAVE filename aaaa +llll	saves an image of memory to a file starting at address aaaa for llll bytes
SPOOL (filename)	copies everything that will be printed to the screen to filename
SPOOL filename	stops copying screen output to filename
I	adds a comment - anything after I is ignored

All of these commands can be used directly from BASIC in its immediate mode by prefixing them with a 'star'. For example if you wanted a copy of your BASIC program

to load into the word processor you could enter the following commands:

```
*SPOOL "test"
LIST
*SPOOL
```

This would start copying everything that was printed on the screen to the file 'test', list the file and hence write it to the file and then close the file.

## Key definition

In the word processor on your Notepad you can define **Macros** for commonly used phrases that are actioned whenever you press the Symbol key together with one of the other keys on the keyboard. BASIC allows you to set these macros directly from within a program. You could for example have alternate sets of macros for doing different jobs and then write a BASIC program to load each set in as you needed them.

When you redefine a key in this way you need to tell BASIC which key to use and what you want the key to do.

```
10 OSCLI "KEY 41 Hello everyone"
```

This line would cause 'Hello everyone' to be inserted into whatever you were typing whenever you pressed the Symbol key with the letter 'A'.

You can include the ENTER key in the string by adding 'IM'.

```
10 OSCLI "KEY 41 Yours sincerely|MIM|Dave Hampson|IM"
```

The value for key can be in the range 0 to 127 but the useful ranges are:

41 to 66 corresponds to 'A' to 'Z'

67 to 92 corresponds to 'a' to 'z'

## The Machine Code Interface

There are a range of commands that allow you to add machine code to your BASIC program. However we do not intend to cover these in any detail for two reasons. First, to do justice to them would require at least as much coverage as we have already given to BASIC itself. The second reason for effectively ignoring them is that they are potentially disastrous in an environment such as the Notepad.

The BASIC in your Notepad was originally written for the BBC Micro and as such

included the ability to add machine code. On the BBC machine it was not too much of a problem if users of the BASIC made a mistake with their machine code and caused the computer to crash. You just had to turn it off and start again. It was unlikely that you would cause damage to documents on the disk no matter what you did.

On the Notepad however the situation is very different. This computer holds all its information permanently in memory. Any minor mistake in a piece of machine code could well wipe out the entire contents of your machine. There would be no way of recovering anything. So be warned. The facility is there if you wish to use it be we strongly recommend against any experimentation.

### PUT

This command allows you to directly output data to one of the hardware ports within the Notepad. These hardware ports typically control functions such as which memory is currently available to the Z80 microprocessor and what interrupts are currently active. If you output incorrect data to any of these ports you WILL CRASH YOUR Notepad.

```
10 PUT 128,32
```

would output '32' to port address '128'.

### CALL, USR

These allow you to directly call a memory address to begin running a machine code program. You would have to have already loaded the machine code into the area of memory before calling it.

```
10 CALL 4864:REM call to machine code program
20 answer = USR(4864):REM call machine code but get a result
```

### HIMEM, LOMEM

You can change the area of memory that BASIC will use for its variables by moving HIMEM down or by moving LOMEM up. This gives you somewhere to store your machine code routines.

```
10 top = HIMEM:REM get the current top of BASIC's memory
20 HIMEM = top - 200:REM move it down a bit
30 bottom = LOMEM:REM get the current base of BASIC's variables
40 LOMEM = bottom + 200:REM move it up a bit
```

## PAGE

This does for a BASIC program what LOMEM does for variables.

```
10 base = PAGE:REM get the start of BASIC
20 PAGE = base + 200:REM move it up a bit
```

## The VDU Emulator

Whenever you print anything from within a BASIC program BASIC does not put the information directly onto the screen. Instead it sends your output to a software emulator of the BBC Micro's VDU drivers. You never need to be aware of this feature except to the extent that the emulator provides a number of useful additional services. This allows your program to have a great deal of control over what appears on the screen.

The BASIC command 'VDU' can be used to send screen control codes to the emulator to cause a variety of effects.

In the following list any parameters to 'VDU' not given are either ignored or just print characters from the ASCII character set.

**VDU 1,n** The byte 'n' is sent to the printer if it is enabled, otherwise 'n' is ignored. 'n' can be in the range '0' to '255' and will work even when the screen is disabled.

**VDU 2** Enables the printer such that all subsequent printable characters that are sent to the screen are echoed to the printer as well. A few non-printable characters are also passed to the printer:- BEL (7), BS (8), HT (9), LF (10), VT (11), FF (12) and CR (13). Any characters that form part of another VDU command are not passed to the printer.

**VDU 3** Disables the printer.

**VDU 4** Enables the text cursor. This is the default condition whenever you enter BASIC.

**VDU 5** Disables the text cursor such that it is hidden.

**VDU 6** Enables the screen display.

**VDU 7** Sounds the Notepad buzzer.

**VDU 8** Moves the text cursor one character to the left. If the cursor was at the



left margin of the text window, it is moved to the right margin of the previous line. If it was also on the top line of the window it is moved to the bottom line.

**VDU 9** Moves the text cursor one character to the right. If the cursor was at the right margin of the text window, it is moved to the left margin of the next line. If it was also on the bottom line of the window it is moved to the top line.

**VDU 10** Moves the text cursor down one line. If the cursor was on the bottom line of the text window, the window is scrolled up by one line.

**VDU 11** Moves the text cursor up one line. If it was on the top line of the text window it is moved to the bottom line.

**VDU 12** Clears the text window to all spaces and moves the text cursor to the top left corner of the window.

**VDU 13** Moves the text cursor to the left margin of the window on the current line.

**VDU 14** Enables inverse text. All subsequent characters will be displayed in reverse.

**VDU 15** Disables inverse text.

**VDU 16** Clears the graphics window to all pixels off.

**VDU 17** Enables bold text. All subsequent characters will be displayed in bold.

**VDU 18** Disables bold text.

**VDU 19** Enables underline. All subsequent characters will be underlined.

**VDU 20** Disables underline.

**VDU 21** Disables screen output. Use this with care! All subsequent VDU commands except numbers 1 to 6 are ignored and nothing is printed on the screen. If the printer is enabled the following list of VDU commands will be sent to the printer: 7, 8, 9, 10, 11, 12 and 13.

**VDU 24, x-left, y-bottom, x-right, y-top**

Sets new co-ordinates for the graphics window. The window does not have to occupy the whole physical screen area. The valid ranges are '0' on the left to '479'

on the right for the x co-ordinates and '0' at the bottom to '63' at the top for the y co-ordinates.

VDU 25,n,x,y

This is identical to the BASIC command PLOT n,x,y.

VDU 26 Resets the text and graphics window to their default conditions. These are that both windows occupy the full screen area, the text cursor is at the top left, the graphics cursor is at 0,0 and the graphics origin is at 0,0.

VDU 27,n Sends the character 'n' directly to the screen without interpreting it as a control code. This allows you to print characters that would otherwise cause an action by the emulator.

VDU 28, x-left, y-bottom, x-right, y-top

Sets new co-ordinates for the text window. The window does not have to occupy the whole physical screen area. The valid ranges are '0' on the left to '79' on the right for the x co-ordinates and '0' at the top to '7' at the bottom for the y co-ordinates.

VDU 29,x,y

Moves the graphics origin to the co-ordinates given by x and y. The valid ranges are '0' on the left to '479' on the right for the x co-ordinates and '0' at the bottom to '63' at the top for the y co-ordinates.

VDU 30 Moves the text cursor to the top left of the text window.

VDU 31,x,y

This is identical to the BASIC command TAB(x,y).

VDU 127 Moves the text cursor one character to the left and deletes the character there.

## **The Practicalities of Writing a Program**

You can't write a large BASIC program from within BASIC. The builtin editor isn't up to it. Indeed that was not the purpose of the editor since the Notepad has a perfectly good word processor.

Having said that, most of the examples in this chapter were written within BASIC but for a large program it would be impractical.

So how do you get a file from your word processor into BASIC? It turns out to be very simple as long as you follow two easy rules.

The first rule is to write your program without any line numbers. That way you can add and delete lines of code without having to bother whether or not the line numbers match up. BASIC can add the line numbers for you automatically when you load the program. This is yet another good reason not to use 'GOTO's.

The second rule is not to use any of the word processor commands to highlight text or centre lines. Just type in your program without any embellishments. That way BASIC will be able to understand everything that you load into it.

To demonstrate the principal type in the following into a document called 'test' in your word processor:

```
NEW
AUTO
FOR test_loop = 1 TO 10
PRINT "Hello"
NEXT test_loop
END
```

Now press Function and 'B' to take you to BASIC and type '\*EXEC test'. BASIC will load your file as though you were typing it in at the keyboard, only very much faster. When the file has been loaded you need to press the 'STOP' key to drop out of the 'AUTO' command and your program is ready to 'RUN'.

You can use this technique on even very large programs although these may take a few seconds to load.

'AUTO' is one of the editing commands we have yet to cover. It tells BASIC to automatically generate the next line number for you. By default BASIC will start numbering at line 10 and go up in steps of 10 but you can change this if you wish.

```
AUTO 100,20
```

would start at line number 100 and increment in steps of 20.

BASIC can also renumber your program for you.

```
RENUMBER 1000,100
```

would renumber all the lines of your program to start at line 1000 and go up in steps of 100.

You can delete a single line by just typing in its number followed by the 'ENTER' key. You can delete more than one line thus:

```
DELETE 100,145
```

would delete all the lines from 100 up to and including line 145. You can delete lines from the start of a program by using '0' as the first parameter.

```
DELETE 0,30
```

would delete all lines up to and including line 30.

You can load and run another program from the current program by using the 'CHAIN' command, but in this case the second program must have been saved from BASIC. You cannot \*EXEC a second program into BASIC.

```
2000 CHAIN "test2"
```

would load and then run the program 'test2'.

As a final point on loading files into BASIC it is possible to automatically load and run a program whenever you enter BASIC if that program is called 'AUTO' and if it has been previously saved from BASIC. This would be a useful feature if you had written an application such as the Hotel Reception Desk. Your user would never be faced with having to load the file himself.

He could use the Notepad functions such as the Diary or Address Book and then whenever he pressed the keysFunction and 'B' your program would automatically run.

## **Summing Up**

If you have read all the way through this chapter - congratulations. If you have just skipped to this last section then you don't know what you are missing.

BASIC is the most widely known and used programming language in the world. Microsoft have just launched a new version called Visual BASIC that is designed to run on the latest, most powerful PC's. The BASIC language will be around for many years to come. It is still being extended and there is no doubt that this will continue.

We hope that we have given you a taste of BASIC's capabilities. We haven't been able

to cover all the commands in great detail but we have given you a good start.

We can assure you that it really is worth the effort of trying to write your own programs.

Good luck!

# BBC BASIC Keywords

ABS	var = ABS(n)	ABSolute value of n
ACS	var = ACS(n)	ARC-cosine of n
ADVAL		- Not supported -
AND	var = n AND n	AND two numeric arguments
ASC	var = ASC(s)	ASCii value of 1st char in string
ASN	var = ASN(n)	Arc SiNe of var in radians
ATN	var = ATN(n)	Arc TaNgent of var in radians
AUTO	AUTO start step	AUTOMATIC line numbers from start in step
BGET	BGET#n	Gets next char from opened file (n)
BPUT	BPUT#n,value	Puts value to an opened file (n)
CALL	CALL add, param	Calls a machine code subroutine
CHAIN	CHAIN s	Loads and runs program in s
CHR\$	ar = CHR\$(n)	String var set to ASCII code n
CLEAR		CLEARs all dynamic variables
CLG		CLears Graphics screen to `white`
CLOSE	CLOSE#n	CLOSEs the opened file (n)
CLS		CLears Screen to spaces
COLOUR	COLOR	- Not supported -
COS	var = COS(n)	COSine of the angle n in radians
COUNT	var = COUNT	COUNTs char sent to display on line
DATA	DATA constant . .	Used for constant(s) DATA with READ
DEF	DEF PROCname	DEFinition of a named PROCEDURE
	DEF FNname	
DEG	var = DEG(n)	Converts n from radians to DEGREEs
DELETE	DELETE start fin	DELETES range of lines from start to fin
DIM	DIM var, size	Reserves space for an array of items
DIV	var = n DIV n	DIVides n by n. Use MOD for remainder
DRAW	DRAW x,y	DRAWs line from current position to x,y
ELSE	IF cond THEN ELSE	Used to provide alternative if cond false
END		Marks the END of a BASIC program
ENDPROC		Marks the end of a procedure definition
ENVELOPE		- Not supported -
EOF	var = EOF#n	End Of File (n) if var is true (-1)
EOR	var = n EOR n	Exclusive OR of n to n
ERL	var = ERL	ERRor of Line number that caused error
ERR	var = ERR	ERRor detected and given number
1	Out of range	
4	Mistake	
5	Missing ,	
6	Type mismatch	
7	No FN	
9	Missing "	
10	Bad DIM	
11	DIM space	

## BBC Basic Keywords

12	Not LOCAL
13	No PROC
14	Array
15	Subscript
16	Syntax error
17	Escape
18	Division by zero
19	String too long
20	Too big
21	-ve root
22	Log range
23	Accuracy lost
24	Exp range
26	No such variable
27	Missing )
28	Bad HEX
29	No such FN/PROC
30	Bad call
31	Arguments
32	No FOR
33	Can't match FOR
34	FOR variable
36	No TO
38	No GOSUB
39	ON Syntax
40	ON range
41	No such line
42	Out of DATA
43	No REPEAT
45	Missing #

ERROR	ON ERROR GOTO	Used to trap ERRORS
ERROR	ON ERROR OFF	
EVAL	var = EVAL	Passes string to BASIC expression handler
EXP	var = EXP(n)	n raised to the power of e
EXT	var = EXT(n)	Length of file (n)
FALSE	var = FALSE	Used in conditionals IF and UNTIL
FN	var = FNname	Used to define and name a function
FOR	var = sn TO en step	Start Loop from Start n TO End n in Steps
GCOL		- Not supported -
GET	var GET	GETs ASCII value of next key pressed
GET\$	var\$ = GET\$	GETs string variable of next key pressed
GOSUB	GOSUB line	Jumps to line until next RETURN command
GOTO	GOTO line	Jumps to line
HIMEM	HIMEM = n	Set/Find out what the new high address
HIMEM	var = HIMEM	is in memory. Best avoided.
IF	IF cond THEN	Used to conditionally execute statements
INKEY	var = INKEY(time)	Like GET - waits time in 1/100th seconds

INKEY\$	var\$ = INKEY\$(time)	Like GET\$ - waits time in 1/100th seconds
INPUT	INPUT "prompt", var	INPUT from keyboard prompt on screen
INPUT LINE	INPUT LINE var\$	INPUT string and assign to var\$
INPUT#	INPUT#n, var	INPUT variable from File(n)
INSTR	var = INSTR(s,find,n)	String is search in find, points to n
INT	var = INT(n)	Converts real number to lower integer
LEFT\$	var\$ = LEFT\$(s\$,n)	Takes the LEFTmost no chars in string
LEN	var = LEN(s\$)	No of characters in string
LET	LET var = value	Assigns value to a variable
LINE		See INPUT LINE
LIST	LIST n,n	LISTs program from line n to n
LISTO	LISTO n	Eight formats of LIST. n = 0-7
LN	var = LN(n)	Natural Log of number
LOAD	LOAD prog_name	LOADs prog_name (in quotes or string var)
LOCAL	LOCAL var	Defines var used only within PROC or FUNC
LOG	var = LOG(n)	Log (to base 10) of number
LOMEM	LOMEM = var	Set/Find out where dynamic structures
LOMEM	var = LOMEM	are placed in memory. Best avoided.
MID\$	var\$ = MID\$(s\$,st,n)	Sets var\$ = s\$ starting at st for n
MOD	var = n MOD n	Divides n by n and gives remainder
MODE		- Not supported -
MOVE	MOVE x,y	MOVEs graphic cursor to (x,y)
NEW	NEW	Clears current program from memory
NEXT	NEXT var	Marks the end of a FOR loop (var)
NOT	var = NOT n	Sets var = bit by bit inversion of n
OLD	OLD	Recovers program if used after NEW
ON	ON var GOTO line	Used to GOTO or GOSUB depending on var
OPENIN	var = OPENIN(s)	Opens Doc/File (named s) to read
OPENOUT	var = OPENOUT(s)	Opens Doc/File (named s) to write
OPENUP	var = OPENUP(s)	Combined effect of OPENIN and OPENOUT
OR	var = n OR n	Sets var = logical bitwise of two n
OSCLI	OSCLI(s)	String passed to the operating system
PAGE	var = PAGE	Set/Find out starting address of
PAGE	PAGE = var	current program area. Best avoided.
PI	var = PI	Sets var to Pi
PLOT	PLOT x,y	Various graphical commands
POINT	POINT x,y	Sets var = current state of pixel (x,y)
POS	var = POS	Sets var = current horz cursor position
PRINT	PRINT var	Prints var (various options)
PROC	PROCname	Used to invoke defined procedure
PTR	PTR#n = var	Set/Read random access point to File (n)
PTR	var = PTR#n	
PUT	PUT port,var	Outputs to I/O device. Best avoided
RAD	var = RAD(n)	Converts degrees to radians
READ	READ var	READs data from a DATA statement
REM	REM comment	REMark or comment line in program
RENUMBER	RENUMBER st,step	RENUMBERS program lines from st in step
REPEAT	REPEAT	Begins loop that ends with UNTIL



## BBC Basic Keywords

REPORT	REPORT	REPORT's message used ON ERROR GOTO
trap		
RESTORE	RESTORE line	Moves pointer to a line when READING DATA
RETURN	RETURN	End of GOSUB routine
RIGHT\$	var\$ = RIGHT\$(s\$,n)	Takes the RIGHTmost no chars in string
RND	var = RND(n)	RaNDom number generated from 1 to n
RUN	RUN	Starts running program in memory
SAVE	SAVE prog_name	Saves current program to a file
SGN	var = SGN(n)	Sets var = -1 if n is -ve, 0 if +ve
SIN	var = SIN(n)	SINe of the angle n in radians
SOUND	SOUND ch,vol,p,d	Ch 1/2, vol not used, pitch, duration
SPC	PRINT SPC(n)	Prints or Inputs n number of spaces
SPC	INPUT SPC(n)	
SQR	var = SQR(n)	SQaRe root of a number
STEP	FOR var st TO f STEP	Allows a FOR NEXT var to be increased or decreased in steps greater than 1
STEP		
STOP	STOP	Like END but prints message where STOP
STR\$	var\$ = STR\$(n)	ets a string = n
STRING\$	var\$ = STRING\$(n,s)	Sets var\$ = n repetitions of s
TAB	PRINT TAB(x,y)	Prints or displays message on screen at location (x,y)
TAB	INPUT TAB (x,y)	
TAN	var = TAN(n)	TANgent of the angle n in radians
THEN	IF cond THEN	Optional statement after IF command
TIME	TIME = var	Sets/Reads elapsed time in 1/100 second
TIME	var = TIME	
TIMES	TIMES = var	Sets/Reads current date and time in fixed format.
TIMES	var = TIMES	
TO	FOR var = st TO f	Used in FOR statement
TRACE	TRACE ON/OFF (n)	Prints line numbers program is executing
TRUE	var = TRUE	Sets var = -1 used in IF and UNTIL
UNTIL	UNTIL cond	Ends loop returns to REPEAT cond TRUE
USR	var = USR(n)	Calls a machine code routine at add no
VAL	var = VAL(s)	Converts string into a numeric value
VDU	VDU(n)	Sends n to BASIC's terminal emulator
VPOS	var = VPOS	Sets var = current vert cursor position
WIDTH	WIDTH(n)	Sets the width of print zones

### Key

n	number
s	string
cond	condition
var	variable
st	start
f	finish

# Stored commands

## Key

f	a filename
n	an integer between 0 and 255
v	a variable identifier
(x)	an optional parameter
{x}	a parameter that may occur 0 or more times
expr	a string expression
text	a string of characters, optionally in quotes
cond	a conditional string expression
*	the command takes effect immediately

## Paper layout commands

f 7	Page Layout	Menu Defaults
>BM n	Bottom margin	3
* >EM n	Even side margin	
>FM n	Footer margin	2
>HM n	Header margin	2
* >OM n	Odd side margin	
>PL n	Page length	66
* >SM n	Side margin	5
>TM n	Top margin	3
>ZM	Zero margins	

## Page formatting commands

>CE text	Centre line
>CP on/off	Continuous/Single sheet printing
>EA nn	End at page number
>EF text	Define even footer text and turn footer on
>EH text	Define even header text and turn headers on
* >EP (n)	Even page throw (can be conditional)
>FF on/off	Form feeds enabled/disabled
>FO text	Define footer text and turn footer on
>FO on/off	Turn footer on/off

## Stored Commands

>FP on/off	Formatting whilst printing on/off
>HE text	Define header text and turn headers on
>HE on/off	Turn headers on/off
* >LS n	Line spacing
>NC n	Number of copies
>NP on/off	Enable/disable new page at the end of printing
>OF text	Define odd footer text and turn footer on
>OH text	Define odd header text and turn headers on
>OP (n)	Odd page throw (can be conditional)
>PA (n)	Page throw (can be conditional)
>PE on/off	Print even pages only
>PN nn	Page number of next page
>PO on/off	Print odd pages only
* >RJ on/off	Right justifying on/off
>sa nn	Start at page number

## Miscellaneous commands

* >CO test	Comment line
* >>>	Comment line
* >CS text	Clear screen and display text
* >DM text	Display message
* >IN f	Insert file
* >ST text	Stop printing and display text
* >WC	Write file close
* >WF f (A)	Open file for writing (appending)
* >WF on/off	Writing to file on/off
* >WM text	Write text to file - used with WF
* >WT text	Wait and display text

## Printer control commands

* >CW n	Define microspace character width
* >MC n {n}	Define microspace code sequence
* >MS on/off	Microspacing on/off
* >OC n {n}	Output codes to printer
* >PP on/off	Proportional spacing on/off

## Variable and data input - mail merging

- \* >AV v {v}      Ask for variables
- \* >CF            Close data file
- \* >DF f {f}      Define data file
- \* >RU v {v}      Read variables unconditionally
- \* >RV v {v}      Read variables and pad with nulls
- \* >SV v=expr     Set variables

## Conditional printing and mail merging

- \* >EI            End of ID, IE, IF or IU block
- \* >EL            Else - print block if previous IF condition false
- \* >ID v          Print block if variable defined
- \* >IE            Print block if data file exhausted
- \* >IF cond       Print block if condition true
- \* >IU v          Print block if variable undefined
- \* >RP            Repeat - until the following UN condition is true
- \* >SK cond       Skip printing if condition true
- \* >UN cond       Until condition is true - repeat from RP command

# Key commands

This list shows all Protex key commands and token numbers. Some commands can be selected two different ways. We have shown both keystrokes or Menu combinations so you can make your choice which to use.

Where we show a keystroke e.g. ^sL, the Control, Shift and L keys should be pressed together. A menu command e.g. Menu E K, the keys should be pressed one after another.

Token numbers are Protex key command numbers and are not normally used by users. They are listed for reference as you do see them when displaying Macros e.g. Sym-d ^788^.

Token	Key	Token	Key	Description
-------	-----	-------	-----	-------------

## Block commands

523	^K	Menu C	Copy Block or Document
525	^M	Menu E K	Clear Block Markers
538	^Z	f -	Move Block
744	^Del	F 9	Set Block marker
746	F 0	F Del ->	Delete Block
933	Menu E B		Copy Block
			Count words in block

## Cursor movement

519	^G	Menu E G	Goto Line, Page or Column
524	^L		goto Last position
539	^[		Move to start of Document
541	^]		Move to end of Document
730	^<		Back paragraph
731	^>		Forward paragraph
732	^(		Back page
733	^)		Forward page
740	s Tab		move to next Tab
747	^ space		Insert a Space
748	s Enter		Move to left margin on next line
752	Up Arrow		Cursor up
753	Down Arrow		Cursor down

754	Left Arrow	Cursor left
755	Right Arrow	Cursor right
756	s Up Arrow	scroll up one line
757	s Down Arrow	scroll down one line
758	s Left Arrow	move word left
759	s Right Arrow	move word right
760	^ Up Arrow	scroll up one screenful
761	^ Down Arrow	scroll down one screenful
762	^ Left Arrow	move to start of line
763	^ Right Arrow	move to end of line

## Extra characters

Sym H	half
Sym Q	quarter
Sym !	Inverted exclamation mark
Sym <	Open quotes (French)
Sym >	Close quotes (French)
Sym ?	Inverted question mark
Sym c or C	cedilla
Sym e or E	ae diphthong
Sym S	double s (German)
Sym n or N	n tilde
Sym o or O	o slash

accents are followed by a letter

705	Sym ‘	accent grave
707	Sym “	accent diaeresis/umlaut
709	Sym %	accent ring
710	Sym ^	accent circumflex
711	Sym \	accent acute
	^sD	Line drawing single/double
	Sym Up Arrow	line/character drawing
	Sym Down Arrow	line/character drawing
	Sym Right Arrow	line/character drawing
	Sym Left Arrow	line/character drawing
776	^sL	Line drawing mode on/off
777	^sC	Line drawing with character
	Sym-Menu 915    Menu E C	Choose character

## Formatting and rulers

	^ P	990 ^ P	Insert Page Break >PA
515	^ C	^ +	Centre Line
516	^ D	Menu T D	Default Ruler
518	^ F		Format to End of Paragraph
522	^ J		Justify On/Off - Configure (On)
530	^ R	Menu T R	Copy previous Ruler but one
535	^ W		Word wrap on/off - Con (On)
720	^ F		Format whole paragraph
984	Menu H		Create Header
985	Menu F		Create Footer
991	Menu T F		Format text

## Find and Replace, text

	f 5	Find
	f 6	Replace
	^ 6	Find next
	^ 5	Find previous

## Find and Replace, marker

512	^ @		Set or Goto marker
	^ @ n		Set/Go to marker (0 - 9)
	^ @ ?	940 Menu T M	Insert multiple marker
	^ @ ]		Go to ] block marker
	^ @ {		Go to [ block marker
	^ @ L		Go to Left margin
	^ @ R		Go to Right margin
727	^ s6		next Marker
728	^ s5		previous Marker
729	^ 5		previous Find
736	^ 6		Next Find
	f 4		Codes on/off (for markers)

## Insertion and deletion

737	^ 4	^ Tab	Insert mode on/off - Con (On)
	f 2		Insert Document
521	^ I	^ 2	Insert Line

513	^ A		Swap adjacent characters
517	^ E	^ DEL	Delete to End of Line
545	Del ->		Delete forwards
639	<-Del		Delete backwards
722	f <-Del		Clear Text
723	s <-Del		Delete word backwards
724	^ <-Del		Delete to start of line
741	s Del		Delete word right
742	^ 3		Delete Line
533	^ U	Menu E U	Undo last delete operation

## Miscellany

	^sStop		Enter command mode (not implemented)
	^sH		File Hide (do not use)
788	Sym - D	Menu E D	Current Date
789	Sym - T	Menu E T	Current Time
902	Menu		Menu key
988	^sS	Screen Dump	

## Other commands

739	^ H	Menu T S	Soft hyphen
	938	Menu T H	Non-break hyphen
	939	Menu T N	Non-break space
774	^sM	Menu M	Macro Record
	Sym - letter		Macro Invoke
983	Menu D		Display Macro
540	^ \		Convert to lower case
743	^ /		Convert to Upper case
526	^ N		Non-break space/hyphen
528	^sP		Page Mode On/Off
534	^ V		View - various options
	^ V I	f 8	Status line on/off - Con (No)
	^ V R		Ruler line hidden/visible - Con (No)
	^ V S		Hard spaces hidden/visible - Con (No)
	^ V T		Tabs and Returns hidden/visible - Con
	^ V V		Control codes hidden/visible
	f =		Word count in Document
764	Stop		Escape



## **Printing**

	^7	834	Menu S B	Insert BOLD code
	^-	841	Menu S I	Insert ITALIC code
	^8	853	Menu S U	Insert UNDERLINE code
536	^X			Enter printer control code
835	Menu S C			Insert CONDENSED code
837	Menu S E			Insert ELITE code
843	Menu S L			Insert ENLARGED code
848	Menu S P			Insert PROPORTIONAL code
849	Menu S Q			Insert QUALITY code
851	Menu S S			Insert SUBSCRIPT code
852	Menu S T			Insert SUPERSCRIPT code
919	Menu B			Print Block
987	Menu P			Print to Screen

## **Spell checking**

	f 1			Spell text in Document
529	^Q	^ 1		Spell Check single word
531	^S			Spell Check from cursor to end
922	Menu E R			Remove word from User Dictionary
931	Menu E V			Display User Dictionary

Codes are

f	Function
^	Control
s	Shift

‘Con’ at the end of the description refers to the Configure menus f 3.

The value in brackets are the default value from either the editing or view options.

# The Printer

## Why are There so Many Types of Printers?

In the beginning when computers began to appear the world was unprepared when it came to printing. Whose job was it anyway? Computer manufacturers felt it wasn't theirs, and the nearest thing to a printer seemed an electric typewriter. Of course then typewriter manufacturers thought that computers were always going to be large and cumbersome and small printers were never thought necessary.

The result is that there are three different things that we need to be aware of.

First there is the method of printing that the printer uses. This affects the look of the printout as well as the different sorts of printout that can be produced from that printer.

Secondly there is the physical means of connecting the printer to the computer.

And thirdly there is the 'standard' that the printer may be able to use (or emulate in computer terms). These are non-printable codes (defined by the printer manufacturer) that the computer uses to tell the printer to change its print. For example when you want to print something in BOLD characters your Notepad has to tell the printer to switch to printing bold. A code is sent when the change is required, then the text to be printed in bold, then a different code to tell the printer to stop printing in bold.

When choosing a printer you should ask yourself what is more important: the final quality of the printout or the price?

What your readers see is the final printout. They do not know that you had used the Notepad to type the document do they? For all they know you could have been using the most expensive computer in the world.

The trap that many users fall into is that they think they should start with a cheap printer with poor quality printout, and get a better printer 'later on'. The trouble is that printers never give you the opportunity to change them later as they keep going on and on. Some of us had to wait five years before our first nine pin printer gave up working. So choose with care.

The other thing to consider is whether you get a printer with a sheet feeder or not? Most printers employ what is called a 'tractor' feed, which uses continuous sheets of paper with holes down the edges. This is fine for doing address labels but do you want to thread this paper through the printer every time you need to change the type of paper you use?

## Printers

A sheet feeder that uses ordinary A4 size paper is a lot easier to use than continuous paper but it is only usually available as 'an optional extra'. The temptation here is not to bother to get it to start with and get it later. Unfortunately, printer manufacturers have a habit of changing models every year or so. Unless you are quick, there is a good chance that when you decide to get a sheet feeder, your supplier will not be able to get it for you. The answer is if you think you might want a sheet feeder, get it with the printer.

Before we get too involved in what we should do with these different sorts of printers, let's first look at how printers work with your Notepad.

## How printers work

After you have written a document in your Notepad, it is stored in the 'List' as a series of numbers. Each character that you typed in to Protext has been stored as a number. These numbers are the same as shown in the ASCII table and are the same numbers that the printer uses to print your text. This is why everything works so well (most of the time).

When you tell your Notepad to print a document it sends these numbers to the printer down a cable. When the printer receives a number, depending what sort of printer it is, it either has to convert that number into a series of dots or get a hammer to print out a letter at the appropriate time.

## Fonts and Printer Attributes

All printers have at least one font, which refers to the look of each character, and a number of printer attributes, which refers to things like bold or underline. If the printer is able to print in a higher resolution, that is print more dots per inch, you will find the printout looking better and you are likely to be able to choose from more fonts.

All these different fonts and printer commands can be sent from the Notepad to your printer but whether you can use any of these functions will depend on your printer. Some printers for example are unable to underline text, so even if you used the underline text attribute in your document, the printer would not be able to print it.

A very important document for every printer is the printer manual. This should have details in it that explain which codes and character fonts the printer uses. Please do not be tempted into getting a printer without its manual.

Let's now look at some of the different types of printers available.

# Different Types of Printers

## The Dot Matrix printer

Starting at the bottom, the dot matrix printer is generally the cheapest. Printing is achieved by a print head that has several pins mounted vertically, the height of a normal printed line. Characters are made up from many small dots that get printed while the print head travels along each line. The result is characters that look like those displayed on the screen of your Notepad (which are also made up from dots).

There are several variations on dot matrix printers. Nine, eighteen or twenty-four pin printers refer to the number of vertical dots there are. The twenty-four pin printer will give a better quality printout than the nine pin because of the higher definition, but they also use a lot more printer ribbon.

As this printer uses dots to produce characters it also lends itself to printing other characters besides normal text, such as boxes on the Notepad and even graphics with other computers. Some twenty-four pin printers can give a printout similar to the definition of laser printers.

The dot matrix printer is a percussion instrument and is also known as 'the rat in the box' because of the noise it generates.

Thermal printers work in a similar fashion to dot matrix except that they use special thermal paper. This gives a printout like a fax machine as well as needing more expensive paper.

## Daisy Wheel Printer

In 1984 daisy wheel printers were all the rave because they did not use characters made out of dots. They worked by using a wheel that contained all the characters.

When a character was printed, the wheel spun round to the right place and a hammer then banged on the wheel to print the selected character. This resulted in a typewritten printed output. To change the typeface, you just change the daisy wheel.

These printers are not so popular now because of their cost but they are still in use.

## Ink or Bubble Jet Printers

Ink Jet and Bubble Jet printers are quiet and work by squirting little droplets of ink onto

## **Printers**

the paper. The print quality is a lot better than the dot matrix printer, which at first sight is surprising since the characters are still made up from dots.

The paper quality used with these printers is quite important. It should neither be too absorbent, like blotting paper, or have a shiny finish on it. The technology is still changing to improve the range of different papers that it can use.

When these printers run out of ink, the complete print head and ink reservoir is changed. This ensures that good print quality is maintained and that any improvements made to the ink or bubble jet technology can be used by both new and existing users.

Unfortunately, changing the ink cartridge is more expensive than just changing printer ribbons.

This technology is very popular for portable computer users as there are several battery powered ink jet printers on the market.

## **Laser Printers**

Not so long ago, having a laser printer meant having to part with several thousand pounds. The price has fallen and these printers are currently regarded as the best. They work like a photocopier machine, but instead of having optics to scan a sheet of paper that you wish to copy, a laser scans the complete page from your computer onto a drum which then prints it onto the paper.

These printers are quiet and fast. The print definition on normal laser printers is about three hundred dots per inch which means that you can no longer see the dots. The latest top quality laser printers have a resolution of six hundred dots to the inch. This gives a printout that is comparable with typeset printing in terms of quality. If print quality is important and you can afford one, forget the rest, go for a laser printer.

## **Serial or Parallel Connection**

We need to connect our printer to the Notepad, but there are two different systems that printers can use. Just like video recorders, you need to know which system your printer is using. Sometimes printers can use either system, in which case choose the parallel option; it is simpler to set up and to use.

## How to Tell What Type of Connection Your Printer Uses

There should be an indication in your printer manual of the interface your printer uses. Most printers are now parallel or Centronics interface printers. The printer connection to the computer should have a socket on it like a tongue and groove with strips of gold contacts.

If your printer is a serial printer, names to look out for are RS-232 or even RS 232-C and the connector on the printer will look like the parallel printer connection on your Notepad.

## Getting the Correct Cable

As you can see, these standards do not even use the same connector on each end of the cable you need to use. If you have a parallel printer, the cable you require is a 25 way 'D' connector to a 'Centronics' connector cable. If you have a serial printer, then the cable you require is a 9 way 'D' connector to a 25 way 'D' connector. Both these cables are used with Personal Computers and most computer outlets should be able to advise you on which cable to use.

## The Trouble with Serial Printers

If you have a serial printer you will find that there are many options that you can change in an attempt to get the Notepad and the printer talking to each other. It really doesn't matter what you choose providing that both your Notepad and the printer are set up to the same values.

The Notepad's default values are 9600 baud, which refers to the speed, 8 data bits, 1 stop bit, No parity and Handshaking On. All this information may look a bit gobbledygook, but if you dive into your printer manual you should find the same sort of information in there under the serial port or RS-232 section.

## Mix and Match

The way to get your printer working correctly is by changing the printer's settings to be the same as the computers, or the other way round.

If you are planning to use a modem with your Notepad, change the printer to the same settings as the modem. Then you won't have to remember to keep swapping the values on your Notepad each time you switch between printer and modem.

## Printers

On your Notepad the way to look at or change the settings is to select the printer (Function P) and then the Menu key. Using the Cursor Up key (to select the second screen) should display the printer settings. As you are using a serial printer you should change the 'Printer Port' from 'parallel' to 'serial' then check with your printer manual and the settings on the printer to establish what else if anything needs changing in this menu. Again, manufactures use different words that mean the same thing.

The speed, (in Baud's), data bits (7 or 8), the number of stop bits (None, 1, 1-1/2 or 2) or the parity bits (None, odd or even) should be easy to find. Handshaking can also be known as Xon/Xoff and should be left turned on, otherwise you may find that part of your document goes missing when you print it out.

There are different ways of changing the settings on your printer. You should check with the printer manual to find out how to do this.

## Parallel Printer

If you have a parallel printer, you only need to make sure that the 'Printer Interface' is set to 'Parallel'; nothing else needs setting. All other options are for the serial interface only.

## Printer Type Setting

Yes, we are nearly there. All that remains is to tell the Notepad what sort of printer you are using so that it can send the correct codes to your printer.

The default setting for the printer is 'Simple', which is the last setting that you should use because 'Simple' will not send any printer codes, only the characters. Better safe than sorry I suppose, this setting should only be used if all else fails. The choices of printers you have are the BJ10e, which is the popular bubble jet printer, Epson 24 pin, Epson 9 pin and IBM 24 pin, which are the dot matrix printers and finally the Laser Jet.

You should also be aware that although the manufacturers name appears with these printer drivers that does not mean that you can only use Epson or IBM printers with your Notepad.

## Emulations

When we had a Star printer it used an 'Epson FX-80' printer emulation that meant that the computer could think that it was printing to an 'Epson 9 pin' printer. On our laser printer we can also change the printer emulator to an 'Epson FX-80' printer.

So here is yet another mix and match situation where you can change the type of printer emulation you are using at the printer's end and the printer driver at the Notepad's end to get a combination that works.

Use the printer's 'test page' to check that you have the correct settings for your printer. Once you have found the settings that work, take the opportunity to write them down. Then if you need to reset the Notepad you will not have to do these tests again just because you hadn't made a note of them.

You will have to dive into that printer manual again to find out what printer emulations you have (if any) but on the Notepad use the printer's menu again to change the printer setting.

## **It's Not That Bad**

Although we have highlighted many different things to do with printers, most users will have a parallel printer and find a printer setting that matches what they already have. This has been written for those users who are possibly having a hard time trying to get their printer working. It is only when things go wrong that we wish we knew the answers.



# ASCII Table

DEC		0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
DEC	HEX	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
0	0	█	▶	SP	0	@	P	˘	p	ç	É	á	⋮	⊥	⊥	α	≡
1	1	☉	◀	!	1	A	Q	a	q	ü	æ	í	⌘	⊥	⊥	β	±
2	2	☉	↕	"	2	B	R	b	r	é	Æ	ó	⌘	⊥	⊥	Γ	≥
3	3	♥	!!	#	3	C	S	c	s	â	ô	ú		⊥	⊥	π	≤
4	4	♦	¶	\$	4	D	T	d	t	ä	ö	ñ	⊥	-	⊥	Σ	∫
5	5	♣	§	%	5	E	U	e	u	à	ò	Ñ	⊥	+	⊥	o	∫
6	6	♠	-	&	6	F	V	f	v	â	û	æ	⊥	⊥	π	μ	÷
7	7		↕	'	7	G	W	g	w	ç	ù	ø	⊥	⊥	⊥	τ	≈
8	8		↑	(	8	H	X	h	x	ê	ý	¿	⊥	⊥	⊥	Φ	°
9	9	TAB	↓	)	9	I	Y	i	y	ë	ö	⊥	⊥	⊥	⊥	θ	•
10	A	LF	→	*	:	J	Z	j	z	è	ü	⊥	⊥	⊥	⊥	Ω	•
11	B	δ	←	+	;	K	[	k	{	ï	ø	½	⊥	⊥	█	δ	√
12	C		-	,	<	L	\	l		î	é	¾	⊥	⊥	█	∞	n
13	D	CR	↔	-	=	M	]	m	}	ï	ø	ı	⊥	=	█	φ	²
14	E	⌘	▲	.	>	N	^	n	~	Ä	ß	«	⊥	⊥	█	€	■
15	F	⊛	▼	/	?	O	_	o		Å	f	»	⊥	⊥	█	∩	⌋

# Diagnostic Test - 'Function' 'Symbol' (whilst turning ON)

There is a full set of diagnostic software on your Notepad. Use this if you suspect that something is wrong or just if you want to make sure that your Notepad passes the test.

You should bear in mind that if your Notepad isn't working at all, you cannot run this test. To start the test, hold down the Function and Symbol keys while turning the Notepad on.

The first screen you should see is the 'Press a key to start diagnostic test (Stop to skip) . . . ' screen. My version has 'ROM v1.00 (c) Amstrad 0 Apr 1992, written by Gavin, Mark and Ray at Amor' written on the bottom of the screen.

If you want to bottle out at this stage, press the 'STOP' key now to return to the Main menu.

## The screen

The first test should make the screen black all over. If you find some white spots on it, clean the screen with a moist cloth and if that doesn't clear them, you may have a faulty screen. Press the space bar for the next test.

The next screen test shows all the possible characters that can be displayed. This is similar to the Symbol menu. There should be a character missing to the left of the '!'. That is the space character. There is a Table of Characters available in the Notepad in the Appendix and in the manual on page 201 if you suspect some characters have gone missing. The characters are displayed in the numerical order of the table.

With all these tests, pressing the 'Space' key repeats the test. Pressing the 'ENTER' key moves to the next test.

## Memory/battery/parallel status byte

This test checks a status byte inside the Notepad. If all is well it will return the result 'F1'.

## Real time clock test

The line under 'set' should have the current date and time.

## Diagnostic Test

```
Real time clock test
Set
1992 08 29 19:02:03
1990 01 01 00:00:00
```

## ROM Test

This test reads all the internal program memory of the Notepad and checks to make sure that it is present and correct.

```
ROM test
00 PASS 01 PASS 02 PASS 03 PASS 04 PASS 05 PASS 06 PASS 07 PASS
08
PASS 0A PASS 0C PASS 0E PASS
PASS
```

## Internal Ram Test

This checks the area of the Notepad's memory that is used for saving your work.

```
RAM Test
PASS
```

## Keyboard test

The normal programs inside the Notepad read the keyboard and convert what they see to the character that you pressed. This test displays the actual number that represents each key rather than the character you see displayed. The values returned for each key are shown below.

Stop	0001	TAB	0010	CapL	001E	LSh	142B	Func	1138
1	0002	q	0011	a	001F	z	102C	Cont	1239
2	0003	w	0012	s	1020	x	102D	Sym	183A
3	0004	e	0013	d	1021	c	102E	Sp	103B
4	0005	r	0014	f	1022	v	102F	\	103C
5	0006	t	0015	g	1023	b	1030	Mnu	103D
6	0007	y	0016	h	1024	n	1031	<-	103E
7	0008	u	0017	j	1025	m	1032	->	103F
8	0009	i	0018	k	1026	,	1033	Darr	1040
9	000A	o	0019	l	1027	.	1034		
0	000B	p	001A	;	1028	/	1035		
-	000C	[	001B	`	1029	RSh	1436		
=	000D	]	001C	#	102A	Uarr	1037		

```
Del> 000E    Ret  001D
```

```
Del< 000F
```

```
Shift key  x4xx
```

```
Function  x1xx
```

```
Control   x2xx
```

```
Sym       x8xx
```

The way the Notepad deals with multiple key combinations like Shift, Function, Control or Symbol when used with another key is quite interesting. The values of these numbers are added to give a new number for the combination.

For example the Stop key gives 0001. If it used with the Shift key, the number is 0401 which is 0001 and 0400 added together.

## Parallel port test

Connect a printer to the parallel port and run the test. The printer will print a row of different characters starting with the '!'. The order of these characters is the same as in the 'Table of characters' except that the first few 'unprintable characters' are missed out.

## Sound test - channel A and B

The sound test gives two different notes from the speaker.

Complete

When the test is completed, you can either select all the tests to be run again by pressing the ENTER key or return to the Main menu with any other key.

# Useful Names and Addresses

Whenever a new product is launched, it is often difficult to get in touch with anybody for advice or accessories. To help you on your way, here is a list of useful names and addresses.

This list was compiled at the end of 1992. Companies may be doing different things or the list may be out of date by the time you contact them.

Amstrad PLC  
Brentwood House  
169 Kings Road  
BRENTWOOD  
Essex  
CM14 4EF

Tel. 0277 228888  
Fax 0277 211350 / 0277 208065

Manufacturers of the Notepad

Arnor Limited  
611 Lincoln Road  
PETERBOROUGH  
PE1 3HA

Tel. 0733 68909  
Fax 0733 67299

Authors of the software (except BBC BASIC) on the Notepad.  
Protext.

CompuServe Information Services (UK) Limited  
15/16 Lower Park Row  
PO Box 676  
BRISTOL  
BS99 1YN

Tel. 0800 289 378  
0272 255111  
Fax 0272 252210

If you have a modem and want to use Electronic Mail, ring CompuServe for further details.

Data Protection Registrar (Office Of The)  
Wycliffe House  
Water Lane  
WILMSLOW  
Cheshire  
SK9 5AF

Tel. 0625 535711  
Fax 0625 524510  
Telex 265871

If you are a Limited Company and store names on a computer (even like the Notepad), it is a legal requirement that you should register. Certain groups are exempt, if in doubt give them a call.

Rakewell Limited  
24 Putnams Drive,  
Aston Clinton,  
Aylesbury,  
Bucks  
HP22 5HH

**Notepad Training Days. Interested? Call for dates. Memory Cards**

Richard Russell MA CEng MIEE  
59 Campbell Road  
GRAVESEND  
Kent  
DA11 0JZ

**Author of BBC BASIC on the Notepad.**

# Index

## Notes -

- 1) All BASIC commands are shown in capital letters
- 2) A list of BASIC keywords is included on pages 311 - 314
- 3) A list of stored commands is included on pages 315 - 317
- 4) A list of key commands is included on pages 318 - 322
- 5) A list of Tips is included at the end of this index

## Abandon, 7

ABS, 255

Absolute marker, 76

Absolute value, 255

Accounts, 59

ACS, 254

Adding a new address, 29

- an alarm message, 22
- instructions to the Word Processor, 43
- lines - BASIC, 177
- memory, 170
- new functions - BASIC, 287 - 289
- strings, 237
- the date/time - Word Processor, 54
- today's date - Mail Merge, 131
- variables - Mail Merge, 133
- your dictionary, 87

Address - right justified, 52

Address book, 29

- adding to, 29
- as a data file, 119
- browse, 30
- delete, 30
- edit, 30
- find, 30
- in List, 91
- transfer, 30
- on a memory card, 171
- k transfer, 169

Address labels, 115, 127 - 134

Address lines - truncating, 130

Addresses - blank lines, 137

Alarm - altering, 23

- changing, 23
- deleting, 23
- editing, 23
- message, 22
- once only, 22
- removing, 23
- repeating, 22
- setting, 22

Alarm going off, 23

Alarm list, 23

Alarms, 21 - 23

Aligning text on the right, 54

Alkaline batteries, 4

Alter a document, 89

Altering menu settings, 9

- the time and date, 15
- where you are, 24
- your password, 19

Arc cosine, 254

- size, 254
- tangent, 254

Array element, 192

Arrays, 190, 191, 223, 224

- one dimensional, 190, 191
- two dimensional, 225

ASC, 246 - 248, 258

ASCII code, 230, 242, 243, 246 - 248

- conversion, 163

Ask for variable - Mail Merge, 131

ASN, 254

Assembler programming, 304

Assigning TRUE or FALSE, 187, 188

ATN, 254

Attributes - style, 72

AUTO, 308

Auto-repeat, 31

Automatic line numbering, 308

- running, 309
- turn on, 23

Automatically saving BASIC, 175

Avoiding errors, 298

Backup, 4, 5, 161, 176

Backward page, 75

BASIC, 173 - 309

BASIC program - in List, 91

Batteries, 4, 5

- alkaline, 4
- lithium, 4
- nicad, 4

Battery compartment - memory card, 170

- compartments, 16
- for memory card, 170

Baud rate, 163

BGET, 283 - 285

Blank lines in address, 137

- lines in data, 123
- memory card, 171

Block copy, 81

- delete, 81
- mark - clearing, 80
- marker, 76
- move, 81
- print, 81
- transfer, 81
- word count, 81

Blocks of text, 80

Bold text, 72

Bottom margin, 95, 112

Boxes - drawing them, 60, 61

BPUT, 283 - 285

Branch instruction, 185, 290, 291

British summer time, 24

Browse through addresses, 30

Bubble jet printer, 325

Bugs, 202

Cable, 164

Calculations during Mail Merge, 149

Calculator, 31, 32

- constant, 31
- memory, 31

Calendar, 26

CALL, 304

Calling a function, 288, 289

- machine code programs, 304

Cancelling alarms, 23

- an entry - calculator, 32

Capitals - changing to, 78

Caps lock key, 11

- lock on, 47, 49
- on, 12

Capture, 165

Card memory, 92

Case - changing, 78

Centre text, 55, 69

CHAIN, 309

Change the place name, 25

- Changing a number to a string, 258
  - a program, 219
  - an address in address book, 30
  - batteries, 5
  - case, 78, 248, 249
  - characters, 248, 249
  - degrees to radians - RAD, 254
  - document format, 162
  - number format - BASIC, 162
  - how STR\$ works, 269
  - line numbers, 308
  - modes - BASIC, 220
  - page numbers, 64
  - radians to degrees - DEG, 254
  - the place name, 24
  - the prompt, 189
  - the time and date, 15
  - the time difference, 25
  - time zones, 24, 25
  - to an integer, 257
  - number type - BASIC, 246 - 248
  - what is printed, 268 - 271
  - your password, 19
- Character transposition, 78
  - width, 106
- Characters - special, 59
- Check spelling, 85, 86
  - word, 85, 86
- Checking a file exists, 282
- Checksum - XModem, 167
- Choosing a document, 89
  - a name, 50, 51
  - to reset, 17
- CHRS, 246 - 248
- CLEAR, 301
- Clear block mark, 80
  - graphics screen, 273
  - memory card, 171
  - screen, 117, 227
- Clearing memory, 17
- Clearing memory - calculator, 32
- CLG, 273
- CLOSB, 281
- Close data file, 124
- Close file - Mail Merge, 157
- Close files, 281
- CLS, 227
- Colon, 217
- Column, 75
- Column - going to, 75
  - number, 47
- Columns - using Tab, 37
- Columns and lines, 227
- COM, 295
- Combining statements on a line, 217
- Comma's in input, 251, 252
- Command lines, 43, 62
- Comment line, 113, 115
- Communications program, 164
- Comparing strings, 240 - 243
- Comparing variables - Mail Merge, 138 - 141
- Comparisons, 192
- Compatible document names, 50
- Compound statements, 248
- Concatenating strings, 237
- Condensed characters, 72
- Conditional new page, 66
  - printing, 187, 188
  - printing - Mail Merge, 138
- Configuration menu - printer, 163
- Configure Menu, 42, 107
- Constant value - calculator, 31, 33
- Context saving, 14
- Continuous printing, 64
- Contrast control, 7
- Control codes - screen, 305 - 307
  - document, 115
  - document - testing, 116
  - key, 11
- Controlling loops, 204 - 206
- Controlling print width, 262
- Conversions - metric and imperial, 34
- Copy address into document, 51
  - block, 81
  - previous ruler line, 46
- Copying a document, 53
  - documents to a memory card, 171
  - the screen to a file, 302, 303
- COS, 254
- Cosine - COS, 254
- COUNT, 262
- Counting words, 55
- CRC - XModem, 167
- Creating a data file, 118
  - a letterhead, 51
  - a ruler line, 44
  - a variable, 182
- CTS and RTS, 164
- Cursor, 36
  - flash - rate, 108
  - home, 273
  - keys, 10
  - left, 7
  - movements, 74
  - position, 47, 261, 262
- Daisy wheel printer, 325
- DATA, 259 - 261
- Data bits, 163
- Data field, 117
- Data file, 115
  - blank lines, 123
  - closing, 124
  - creating, 118
  - defining, 120
  - designing, 117
  - end of, 142
- Data record, 117
- Database, 117
- Date, 15
  - getting and setting - BASIC, 300
  - in a document, 54
  - of document, 92
- Decimal character, 107
  - places in variables, 134
- Default, 13
  - ruler, 52
  - ruler line, 44
  - settings, 18
- Defining a data file, 120
  - an array, 193 - 195
  - functions, 287 - 289
  - keys, 302, 303
- Definitions, 3
- DEG, 254
- Degrees, 254
- DELETE, 308
- Delete block, 81
  - character/word/line, 78
  - keys, 10
- Deleting a diary entry, 27
  - addresses, 30



- alarms, 23
- documents, 90
- documents on a memory card, 171
- files, 302
- from dictionary, 87
- markers, 76
- Designing a data file, 117
- Diagnostic test, 18
- Diary, 26
- Diary entries, 26
- Difference in time, 25
- Different address books, 172
- DIM, 193 - 195, 223 - 225
- Dimensioning an array - DIM, 193 - 195
- Diak, 4
- Display message, 113, 117
- Displaying macros, 57
- DIV, 257
- Document date and size display, 169
  - disappears, 51
  - format, 162
  - formatting, 56
  - insert, 53
  - receive, 166
  - send, 165
  - size, 92
  - transfer, 161
  - transfer port and format, 163
- Documents and files, 88, 89
- Doing a reset, 17
- Doing maths on characters, 248, 249
- Dot matrix printer, 325
- Double height, 99
- Double line drawing, 61
- Download, 165
- DRAW, 274, 275
- Drawing boxes, 60, 61
- Drawing lines, 60, 61, 274, 275
- DSR and DTR, 164
- Edit a document, 89
  - an address, 30
  - an alarm, 23
  - mode - BASIC, 219
  - word, 86
- Editing a program, 219
  - commands, 219
  - place name, 24
  - time zones, 24
  - your dictionary, 87
- Elite characters, 72
- ELSE - IF - THEN, 185
- ELSE - Mail Merge, 139 - 143
- Empty document, 51
- Empty string, 233
- Emulations, 328
- END, 178
- END IF, 139 - 143
- End of file, 282
- End of line - deleting to, 78
- Ending a FOR - NEXT loop, 206
- Enlarged characters, 72
- Enter key, 7, 37, 38
- EOF, 282
- EOR, 211, 212
- Equal operator, 192
- ERI, 296, 297
- ERR, 296, 297
- Error line, 296, 297
- Error message, 298
- Errors - handling them, 296, 297
- Errors - mistake at line, 189
- Errors - syntax, 189
- ESC - turn on/off, 302
- BVAL, 258
- Even - parity, 164
  - and odd pages, 63, 64
  - margin, 112
  - page throw, 66
  - pages - printing, 65
- Exclusive OR, 211, 212
- Existing macros, 57
- EXP, 254, 255
- Exponential format for print, 269 - 271
- EXT, 285
- Extending the IF statement, 217
- Extra array element, 195
- Extra memory, 16, 170
- FALSE, 187, 188
- Fetch a key, 234, 235
- Field - data, 117, 118
- File access, 280
  - end, 282
  - handle, 280
  - pointer, 286, 287, 292
  - size, 92, 285
  - transfer, 162
- Files, 278 - 285
  - byte at a time, 283 - 285
  - closing, 281
  - controlling where to read from, 286, 287, 292
  - deleting, 302
  - opening, 280
  - opening for input, 282
  - opening for update, 283
  - renaming, 302
  - and documents, 88, 89
- Find, 82 - 84
  - next, 84
  - previous, 84
- Finding a marker, 77
  - a name, 29
  - a telephone number, 29
  - an address, 30
  - the percent, 34
  - where you are in a document, 47
- Fixed format for print - BASIC, 269 - 271
- Flags, 217, 218
- Flash rate - cursor, 108
- Flat batteries, 5
- Flow control, 290, 291
- FN, 287 - 289
- Fonts, 72, 324
- Footer margin, 112
- Footers and headers, 62 - 64
- FOR - NEXT, 205, 206
- Forced reset, 17
- Forcing a new line, 267
- Foreign characters, 59
- Form feed option - Mail Merge, 129
- Form feeds, 67, 68
- Form filling with Mail Merge, 151 - 155
- Format, 268
- Format document, 56
  - of a document, 162
  - of time 12/24, 14
- Formatting a memory card, 171
  - a page, 62
  - characters, 70
  - on, 69
  - attaching text, 44, 55

- with Mail Merge, 68
- Forward page, 75
- Function key, 11
- Functions - adding your own, 287 - 289
- Fuse, 5
- General format for print, 269 - 271
- General purpose number format, 199
- GET, 235
- GETS, 250
- Getting the time, 300
- Getting to the main menu, 12
- GOSUB, 196 - 202
- GOTO, 184
- Goto a marker, 77
- Goto last position, 77
- Goto page, 75
- GOTO's - removing them, 203
- Graphical output, 272, 273
- Graphics effects, 275 - 278
- Greater than operator, 192
- Greater than or equal operator, 192
- Green keys, 31
- Greenwich mean time, 24
- Halting gracefully, 185
- Handshake, 164
- Hard carriage returns, 70
  - hyphens, 71
  - reset, 17
  - spaces, 70
- Hardware handshaking, 164
- Header margin, 111
- Headers and footers, 62 - 64
- Hex dump, 104
- Hexa decimal numbers, 266
- Hidden characters, 69
  - documents, 91
  - information, 162
- High beep, 17
- HIMEM, 304
- Home cursor, 273
- Horizontal scrolling, 74
- Hyphens - hard, 71
- IF - Mail Merge, 139 - 143
- IF - THEN, 217
- IF - THEN - ELSE, 185
- Ignore word, 86
- Immediate mode, 218, 220
- Imperial conversions, 34
- Import block, 81
- Including data in your program, 259 - 261
- Increasing by a percent, 34
- Indenting text, 45
- Initial settings, 13
- Ink jet printer, 325
- INKEY, 234, 235
- INKEYS, 249, 250
- INPUT, 183
- INPUT LINE, 251, 252
- Inputting from a file, 282
  - strings, 249
- Insert, 37
- Insert a document, 89
  - document, 114
- Insert on, 47, 48
- Inserting a document, 53
  - a space, 69
  - addresses in address book, 29
  - lines, 177
  - the date, 54
  - the time, 54
- Inspecting variable contents, 221
- Installing a memory card, 170
- INSTR, 243 - 248
- INT, 257
- Integer variables, 182
- Integers, 257
- Internal fuse, 5
- Internal memory, 170
- Italics, 72
- Joining strings, 237
- Joining variables - Mail Merge, 134
- Justifying output, 264 - 266
- Key definition, 302, 303
  - repeat, 107
- Keyboard input, 183
- Keys - short cuts, 9
  - special, 7, 9 - 11
- Label length, 129
- Label position, 127, 128
- Lapcat, 162
- Large documents, 93
  - programs, 225, 307
- Laser printer, 326
- Last position - goto, 77
- Layout, 108
- Layout Menu, 42
- Layout of BASIC examples, 175
- Leading spaces, 251, 252
- Learn word, 86
- Leaving a menu, 10
  - BASIC, 301
  - loops early, 206
  - procedures correctly, 202
- Left arrow, 7
- LEFTS, 238 - 240
- Left-hand page, 63, 64
- LEN, 237
- Length of a file, 285
  - of a string, 237
  - of page, 67, 68, 112
- Less than operator, 192
- Less than or equal operator, 192
- LET, 180, 207
- Line, 75
  - going to, 75
  - delete, 78
  - feed option - Mail Merge, 129
  - number, 47
  - numbers, 177
  - spacing, 65
- Lines - drawing, 274, 275
  - drawing them, 60, 61
  - not printing, 105
  - and columns, 227
- Linking commands, 217
- LIST, 177
- List of alarms, 23
  - of documents, 88, 89
  - of files, 302
  - of items to print, 263
- Listing a program, 177
  - single lines, 177
  - time zones, 24
- Lithium batteries, 4
- LN, 254, 255
- Loading a program, 179

- Loading programs - chain in BASIC, 307
  - LOCAL, 199 - 201, 261, 262
  - Locking your notepad, 18
  - Logarithms, 254, 255
  - Logical operators, 209 - 216
  - Logical operators - summary, 213
  - LOMEM, 304
  - Long address lines, 130
  - Long lines, 48, 94
  - Lookup word, 86
  - Loop control, 204 - 206
  - Loops within loops, 208, 209
  - Losing characters, 164
  - Low beep, 17
    - memory, 161
  - Lower case to upper case, 79
  - Lower memory, 92
  - LPT, 295
  - Machine code, 303, 304
  - Macros, 56, 57
  - Macros and react, 58
  - Mail Merge, 29, 115
    - testing - pause screen, 126
    - text formatting, 68
  - Main menu, 12
  - Mains adaptor, 4, 5, 16
  - Major keys, 7, 9 - 11
  - Making a noise, 300, 301
  - Manipulating strings, 236
  - Margin marker, 76
  - Margins, 95, 109 - 112
  - Mark block, 80
  - Marker - goto, 77
  - Marker - previous, 77
  - Markers, 76
  - Maths functions, 253
  - Maximum document size, 93
  - Maximum range in Mail Merge, 150
  - Memory - calculator, 31
    - out of, 92
    - card, 4, 16, 170 - 172
    - card and the address book, 171
    - shortage, 161
    - types, 92
    - used, 92
  - Menu - Configure, 42, 107
    - Layout, 42
    - Word Processor, 41
  - Menu key, 9
  - Menus - how to use, 40, 41
    - leaving them, 10
    - selecting from, 9
  - Merging address books, 172
  - Message in alarm, 22
  - Messages, 113
  - Metric conversions, 34
  - Microspacing, 105
  - MIDS, 238 - 240, 247, 248
  - Minimum range in Mail Merge, 150
  - Miscellaneous appointment, 27
  - Missing lines, 43
  - Missing your diary dates, 27
  - MOD, 257
  - Modem transfer, 169
  - Modes - changing, 220
  - Modes of BASIC, 218
  - Modulus, 257
  - MOVE, 274, 275
  - Move block, 81
  - Moving appointments, 27
    - around the calendar, 26
  - BASIC, 305
  - documents with a memory card, 171
  - memory, 304
  - text to the centre, 55
  - the cursor, 74, 227, 261, 262
  - the graphics cursor, 274, 275
- Multiple copies - printing, 65
    - diary entries, 28
    - marker, 76
    - statements, 217
    - strings, 252
  - Name a document, 50, 51
  - Naming variables, 180
    - variables - Mail Merge, 121
  - Narrow screen, 262
  - Natural logs, 254, 255
  - Near letter quality characters, 72
  - Nesting, 208, 209
  - Nesting - Mail Merge, 143
  - New document, 50
    - memory card, 171
    - page after print, 66
    - page during print, 66
  - Next, 77, 204 - 206
  - Next find, 84
  - Next marker - move to, 77
  - Nicad batteries, 4
  - NLQ characters, 72
  - No margins, 96, 112
  - No memory left, 170
  - No space left, 161
  - Non exclusive OR, 210
  - Non-break hyphens, 71
  - Non-break space, 70
  - Non-printable characters, 248
  - None - parity, 164
  - None - protocol, 165
  - Not equal operator, 192
  - Not shown - system setting menu, 169
  - Null modem cable, 164
  - Null string, 233
  - Numbering lines, 177
  - Numbers from strings, 258
  - Numeric operators, 192
  - Odd - parity, 164
  - Odd and even pages, 63, 64
  - Odd margin, 112
  - Odd page throw, 66
  - Odd pages - printing, 65
  - ON - GOSUB, 290, 291
  - ON - GOTO, 290, 291
  - ON - PROC, 290, 291
  - ON ERROR, 296, 297
  - On switch, 7
  - One dimensional arrays, 190, 191
  - One finger operation, 14
  - Open file - Mail Merge, 157
  - OPENIN, 282
  - Opening a file for output, 280
  - Opening a file for update, 283
  - Opening a new document, 50
  - OPENOUT, 280
  - OPENUP, 283
  - Operating system commands, 301 - 303
  - Operators, 192
  - Operators - logical, 209 - 216
  - OR, 210
  - Order of running, 184

- Original state, 17
- OSCLI, 301 - 303
- Out of memory, 92, 161
- Out of room, 161
- Output printer codes, 98 - 104
- Output to a file, 157
- Outputting to a port, 304
- Outputting variables - Mail Merge, 125
- Outstanding alarms, 23
- Overtyping, 37
  
- Padding with spaces, 252
- PAGE, 305
- Page - going to, 75
  - formatting, 62
  - forward, 75
  - layout, 95, 108
  - layout commands, 108, 111
  - length, 67, 68, 112, 129
  - margins, 95, 109, 111
  - mode, 75
  - number, 75
  - number - changing, 64
  - numbers, 62 - 64
  - numbers - which to print, 65
  - setup, 95
  - throw after print, 66
  - throw during print, 66
  - titles, 62 - 64
- Parallel cable, 164
  - connection - printer, 326
  - link, 162
  - port - access from BASIC, 295
  - socket, 16
  - to ASCII, 163
  - to protect, 163
- Parity - Odd, 164
- Part of a string, 238 - 240
- Partial list, 177
- Password, 10, 18, 19
  - changing it, 19
  - deleting it, 19
  - setting one, 19
- Paste block, 81
- Pausing Mail Merge screen, 126
- PC names, 50
- Percentages, 34
- Physical connection, 164
- PI, 253
- Pixel - on, 275
- Pixels, 273
- Place name - changing, 24
- Placing the cursor - in BASIC, 227
- Plain ASCII, 165
- Plain ASCII document, 163
- Playing notes, 300, 301
- PLOT, 275 - 278
- POINT, 275
- POS, 261, 262
- Position of label, 127, 128
- Positioning within a file, 286, 287, 292
- Power off delay, 14
- Power saving, 14
- Power switch, 7
- Pre-printed forms, 151 - 155
- Preprogrammed macros, 57
- Preserve context, 14, 27
- Pracet settings, 13
- Previous find, 84
- Previous marker - goto, 77
- PRINT, 178
  
- Print - add new line, 267
  - block, 81
  - control, 263 - 271
  - formatting, 263 - 267
  - left-justified, 264 - 266
  - list, 263
  - right-justified, 264 - 266
  - test page, 94
  - time messages, 113
  - two address labels, 144 - 148
  - zones, 264 - 267
- Printable characters, 248
- Printer, 16
  - selection, 302
  - attributes, 324
  - codes, 98 - 104
  - codes - showing, 108
  - commands, 96
  - configuration menu, 163
  - driver, 98
  - type setting, 328
  - width, 94
- Printers - types of, 323
- Printing, 94, 95, 113
  - line spacing, 65
  - single sheets, 64
  - a document, 90
  - address labels, 127 - 136
  - blank lines, 137
  - continuously, 64
  - graphics, 272 - 278
  - in hexadecimal, 266
  - lines, 96
  - multiple copies, 65
  - some pages, 65
  - to a file - in BASIC, 281
  - to files - in Mail Merge, 157
  - two address labels, 136
  - variables - Mail Merge, 125
  - your diary, 27
  - your letter, 39
- Private variables, 199 - 201
- PROC, 196 - 202
- Procedures, 196 - 202, 287, 288, 289
  - leaving them correctly, 202
  - why use them, 203
- Program changes, 219
  - flow, 185
- Program loading, 179
- Program loops - Mail Merge, 142
- Program mode - in BASIC, 219
  - saving, 179
  - specification, 222, 223
  - speed, 182
- Programming a memory card, 172
- Prompt for input, 189
- Prompt string, 190
- Proportional characters, 72
  - printing, 105
- Protecting your information, 18
- Protecting your memory card, 171
- Protect, 40
- Protocol, 165
  - checksum, 167
  - CRC, 167
  - send, 168
- PTR, 286, 287, 292
- PUT, 304
  
- Quality characters, 72
- Question mark, 189

- Quiting BASIC, 301
- Quotient, 257
- RAD, 254
- Radians, 254
- Random access, 280, 286, 287, 292
  - numbers, 256
- Re-starting a program, 226
- Re-usability, 199
- READ, 259 - 261
- Reading the keyboard, 234, 235
  - variables - Mail Merge, 122
- Real variables, 181
- Receive document, 166
  - XModem, 168
- Receiving the address book, 169
- Record - data, 117
- Recording a new macro, 57
- Recording keystrokes, 56, 57
- Redefining keys, 302, 303
- Reducing by a percent, 34
- Reentrant functions, 288, 289
- Releasing memory, 27
- REM, 178
- Remainder, 257
- Remarks, 178
- Remote connection, 169
- Remove margins, 112
- Removing a diary entry, 27
  - a document, 90
  - addresses, 30
  - alarms, 23
  - from dictionary, 87
  - GOTO commands, 203
  - lots of lines, 308
  - margins, 96
  - your password, 19
- Renaming a document, 90
  - files, 302
- RENUMBER, 308
- REPEAT - UNTIL, 207, 208
- Repeat rate - keys, 107
- REPEAT UNTIL - Mail Merge, 142
- Repeating alarms, 22
  - loops, 204 - 206
- Replace, 82 - 84
- REPORT, 298
- Reset, 17, 18
  - effect on macros, 58
- Resetting variables, 301
  - your password, 20
- Rest of a string - MID\$, 238 - 240
- RESTORE, 260, 261
- Returning to the same point, 14
- Right justify on, 47, 48
- RIGHTS, 238 - 240
- Right-align, 54
- Right-hand page, 63, 64
- Right-justification on, 69
- Right-justify an address, 52
- RND, 256
- Rounding error in Mail Merge, 156
- RTS and CTS, 164
- Ruler - showing, 108
  - copy previous, 46
  - creating, 44
  - default, 44
  - extra width, 94
  - indenting text, 45
- Ruler lines, 44, 52
- Rules of BASIC, 175
- RUN, 176, 219
- Running another program, 309
- Running from mains, 5
- Running order, 184
- Running out of memory, 92
- Running system commands from BASIC, 301 - 303
- Safe transfer, 167
- Saved alarms, 23
- Saving data to files, 278 - 285
  - documents on a memory card, 171
  - names and addresses, 29
  - power, 7
  - your work, 179
- Scanning the address book, 30
- Screen brightness, 7
  - control, 7
  - control codes, 305 - 307
  - position, 261, 262
- Scrolling, 73, 74
- Search, 82 - 84
- Searching for a name, 30
  - strings, 243 - 245
  - the address book, 30
- Secret document, 10, 18
  - key, 9
- Secure link, 167
- Seeing newlines and TABs, 69
- Selecting a diary date, 26
  - a document, 89
  - items, 9
  - special characters, 59
  - the printer, 302
  - where you are, 24
- Send document, 165
- Send XModem, 167
- Sending the address book, 169
- Sequential access, 280
- Serial cable, 164
  - connection - printer, 326
  - link, 162
  - port - access from BASIC, 295
  - socket, 16
  - terminal, 163
  - to ASCII, 163
  - to pretext, 163
- Set double height, 99
- Setting a marker, 76
  - a password, 19
  - a variable, 132
  - an alarm, 22
  - the time, 300
  - the time and date, 15
  - up for transfer, 163
- Settings - changing them, 9
- SGN, 255
- Shareware, 164
- Shift Enter, 38
- Shift keys, 11
- Shift Tab, 38
- Short cut keys, 9
- Show address book, 91
  - printer codes, 108
  - ruler, 108
  - ruler line, 44
  - spaces, 108
  - status information, 108
  - tabs and returns, 108
- Side margin, 112
- Sign of a number, 255
- SIN, 254

- Sine - SIN, 254
- Single line drawing, 60, 61
- Single sheet printing, 64
- Size of document, 92
- Soft carriage returns, 70
  - hyphens, 71
  - reset, 17
  - spaces, 48, 70
- Software handshaking, 164
- SOUND, 300, 301
- Space - inserting, 69
  - non-break, 70
  - available, 92
- Spaces - hard, 70
  - showing, 108
- Spacing columns, 37
  - of lines, 65
- Spare batteries, 5
- SPC, 252
- Special characters, 59, 248
  - characters - not printing, 105
  - keys, 7, 9 - 11
- Specification of a program, 222, 223
- Speed, 163
  - of execution, 189
- Speeding up your program, 182
- Spell checker, 36, 85, 86
- Splitting paragraphs, 66
  - variables - Mail Merge, 134, 135
- Spool file, 302, 303
- SQR, 257
- Square root, 34, 257
- Standard headings, 51
- Start a new document, 50
- Start of line - deleting from, 78
- Starting BASIC, 176
- Status bar, 47
- Status information - showing, 108
- STEP, 204 - 206
- Sticky shift keys, 14
- Stop bits, 163
- Stop control document, 117
  - key, 7
  - printing, 114
- Stopping a program, 185
- Store word, 86
- Stored documents, 88, 89
- STR\$, 258
- Strange behaviour, 17
- String - converting from a number, 258
  - converting to a number, 258
  - how long, 237
  - null, 233
  - part of, 238 - 240
  - arrays, 223, 225
  - variables, 182
- Strings, 232, 233, 236, 252
  - adding, 237
  - comparing, 240 - 243
  - from the keyboard, 249
  - searching in, 243 - 245
  - subtracting, 238 - 240
- Style attributes, 72
- Subroutines, 196 - 202
- Subscript characters, 72
- Subtracting strings, 238 - 240
- Summer time, 24
- Superscript characters, 72
- Swap characters, 78
- Switching modes, 220
- Symbol key, 12
- Syntax, 189
- System settings menu, 13, 14, 163, 169
- System time, 300
- TAB, 227
- Tab key, 10, 38
  - stop, 37
- Tab and returns - showing, 108
- Take one string from another, 238 - 240
- TAN, 254
- Tangent - TAN, 254
- Telephone number search, 29
- Template files, 110
- Terminal program, 163
- Test page - printing, 94
  - software, 18
- Testing a value, 185
  - control documents, 116
  - for TRUE, 187, 188
  - the notepad, 18
  - variables, 192
- Text fonts, 72
  - formatting, 55
- The clock, 15
- The colon, 217
- The hidden array element, 195
- The information bar, 12
- The template, 15
- The time, 47, 49, 300
- THEN - IF - ELSE, 185
- Time, 15, 299
  - in a document, 54
  - delay, 299
  - difference, 21, 25
  - display format, 14
  - manager, 21
  - of alarm, 22
  - zones, 21, 24, 25
- TIMES, 300
- Tips, 3
- Today's date, 12
- Top margin, 95, 111
- Transfer address, 30
  - address into document, 51
  - block, 81
  - method, 162
  - port, 163
  - protocol, 165
- Transferring between notepads with a memory card, 171
  - documents, 161, 162
  - the address book, 169
- Transpose characters, 78
- Trigonometric functions, 253
- TRUE, 187, 188
- Truncating address lines, 130
- Truncating variables, 134, 135
- Truth, 186
- Turn a pixel on, 275
- Turn capitals on or off, 11
- Turn on every day, 27
- Turning it on, 7
- Turning on automatically, 23
- Turning the ESC key on, 302
- Two address books, 172
- Two dimensional arrays, 223 - 225
- Type changes - in BASIC, 246 - 248
- Types of memory, 92
  - of printers, 323
- Unauthorised access, 18
- Undelete, 78

# Index

- Underline, 72
- Updating your address book, 29
- Upload, 167
- Upper case to lower case, 79
  - memory, 92
- User dictionary, 87
- Using a constant - calculator, 33
  - a master document, 53
  - a memory card, 171
  - a modem, 169
  - a new macro, 58
  - a ruler line, 52
  - memory - calculator, 33
  - menus, 40, 41
  - special characters, 248
  - Tab, 37
  - the mains, 5
- USR, 304
- VAL, 258
- Variable, 263, 268 - 271
  - giving a value, 132
  - accuracy, 182
  - arrays, 190, 191
  - not found, 122
- Variables, 179
  - at print time, 131
  - decimal places, 134
  - integer, 182
  - Mail Merge, 120
  - real, 181
  - string, 182
  - on both sides, 181
  - remember, 181
- VDU commands, 305 - 307
- Version number, 18
- Vertical scrolling, 73, 74
- Viewing character attributes, 73
  - hidden characters, 69
  - your dictionary, 87
- VPOS, 261, 262
- Wait - when merging, 117
  - for a delay, 299
  - message, 114
- Waiting for a key, 234
- What does a variable hold, 221
- What is a string?, 232
- What is BASIC?, 174
- What to call a variable, 180
- Where are you, 24
- Which error, 296, 297
- Why use a word processor?, 35
- Why use procedures?, 203
- Wide characters, 106
- WIDTH, 262
- Width of printer, 94
- Wildcard searches, 84
- Wipe memory card, 171
- Wiping your documents, 17
- Word check, 85, 86
  - count, 55
  - count block, 81
  - key, 7
  - Processor commands, 40
  - Processor mem, 41
  - wrap, 37, 70
  - wrap on, 47, 48
- Write protect tab, 170, 171
  - to file - Mail Merge, 157 - 160
- Writing a large program, 225, 307
- single bytes to a file, 283 - 285
- to the serial port, 295
- XModem, 165
- XModem receive, 168
- XModem send, 167
- XModem transfer, 167
- Xon - Xoff, 164
- Zero margin command, 112
- TIPS
  - Changing Batteries, 5
  - The Hidden Fuse, 5
  - Short Cut Commands, 9
  - Altering Settings in a Menu, 9
  - Numbers and Letters are Different, 13
  - Get a Printer, 16
  - Alarm - Once You Press D, 22
  - Are You Still at the Right Time?, 25
  - The Calendar Moves in Mysterious Ways, 27
  - You Can Miss Your Diary Dates, 27
  - Lines Missing in Your Printed Document?, 43
  - Don't Use '>', 43
  - On Ruler Lines, 45
  - Typing a Long Line Unintentionally, 48
  - Put Something in a New Document, 51
  - Adding The Time, 54
  - Help With Page Lengths and Formfeeds, 67
  - Markers Are Only Shown When Codes Are On, 76
  - Use Goto Left Marker to Move to a New Left Margin, 77
  - Get The Spelling Right When Using Find and Replace, 81
  - Only Copy a Block or Document From the Word Processor, 91
  - Page Length Warning, 95
  - Gotcha Printer Codes, 103
  - Do Not Leave Variable Values Between &s, 122
  - A Sneak Preview of BASIC Commands, 221

# S&S Computer Advice

Education Consultants

Software Designers

Specialists in data transfer software for the Amstrad NC100.

Transfer software is available to link the NC100 to:

Archimedes

BBC B / Master

Macintosh

IBM compatibles

PC Microsoft Windows

Memory card based software is also available.

Titles to date include

Sketchpad

NC Turtle - a LOGO trainer

Foliage - A database

Double height - a Large text processor

T-Touch - The Typing Tutor

Under development

NC Term - Terminal software

NC -base - A relational database

For further details contact us at:

8 Anchor Close, Hathern, Loughborough, Leics LE12 5HP



## **NC100 ACCESSORIES FROM RANGER**

Ranger specialise in providing accessories and support for sub-notebook computers, including Psion series 3, Cambridge Z88, Tandy WP-2 and, of course, NC100. Ranger has an active development programme for NC100 related products, which include the following:

### **RangerDisk3**

The ultimate NC100 accessory, RangerDisk gives unlimited storage of NC100 files on IBM PC-compatible 720K or 1.44M byte diskettes. RangerDisk3 is extremely easy to use, requires no computer experience and comes complete with NC100 software, cable and AC adapter.

Facilities include formatting of new disks and the creation and use of sub-directories, if required. The disk drive is completely compatible with all IBM-PC's that have 3.5" drives and allows files to be copied from PC to NC100 and vice versa.

### **RangerTerm**

RangerTerm allows NC100 to communicate with dial-up mailboxes via a modem connection. The terminal is compatible with the VT52 standard and provides the popular XMODEM CRC/Checksum protocol for file transfer.

RangerTerm is also ideal for transferring files to other computer systems using a direct RS-232 connection at speeds up to 19,200 baud. The terminal is compatible with any computer capable of running XMODEM, including IBM-PC under DOS or Windows and Apple Macintosh.

### **Corporate Applications**

Ranger has a comprehensive software development environment for NC100 and can provide custom application development for specific end user applications.

### **Other Products**

Ranger supports NC100 with a full range of memory cards, cables, printers, spare parts and other accessories intended to complement the basic machine. Ranger also intends to offer an upgrade service to users as future versions of the NC100 operating system become available, subject to applicability. Please enquire for details.

---

### ***Ranger Computers Ltd.***

Ranger House • 2 Meeting Lane • Duston • Northampton • NN5 6JG  
England

Telephone (0604) 589200 • FAX (0604) 589505





# NC100 MAGIC

## *All About the Amstrad Notepad*

This book will help you get to know and love your Amstrad NC100 Notepad *after* the first five minutes. It is written in plain, understandable English. Even those who are familiar with computers will be able to use it and get more from their NC100, especially since it includes many hidden commands.

There is also something in this book for those already using computers on a day-to-day basis. Anyone coming to the Notepad thinking that they know it all will get a surprise. In an attempt to make the Notepad as user friendly as possible, Amstrad has dropped many of the normal computing conventions.

Everything about the NC100 makes it look fun to have, which masks the power lurking inside. 'NC100 Magic' will help its readers tap into that additional power and make the Notepad really work for them.

The book covers *all* the things that the NC100 can do, including:

- *Word processing in depth, including extra hidden commands.*
    - *Storing names and addresses.*
    - *Keeping your diary.*
  - *Using the calculator.*
  - *Mailmerging (in detail).*
  - *Basic.*
- and an extensive section on programming*

*All sections have lots of easy to follow examples.*  
*NC100 Magic is essential reading for all NC100 users.*

Published by

# Kuma

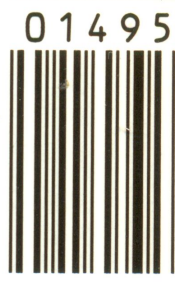
Kuma Computers Ltd,  
Pangbourne, Berkshire, England  
Tel: 0734 - 844335  
Fax: 0734 - 844339

£14.95 net

ISBN 0-7457-0236-8



0 1 4 9 5 >



9 780745 702360