

POCKET COMPUTER PROGRAMS

Nat Wadsworth



**50 PRACTICAL PROGRAMS FOR
PEOPLE ON THE MOVE —**

for business, science, the home, or just for fun.
Expertly written for the Sharp PC-1500 and
Radio Shack PC-2

HAYDEN

4.00

POCKET COMPUTER PROGRAMS

Nat Wadsworth



HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

Acquisitions Editor: DOUGLAS McCORMICK
Production Editor: MARSHALL E. OSTROW
Art Director: JIM BERNARD
Text design: SUSAN BROREIN
Cover photo: RON SCALERA
Composition: McFARLAND GRAPHICS & DESIGN, INC.
Printed by: THE BOOK PRESS
Bound by: SLOVES ORGANIZATION

Library of Congress Cataloging in Publication Data

Wadsworth, Nat.

Pocket computer programs.

1. PC-1500 (Computer) — Programming. 2. TRS-80 pocket computer PC-2 — Programming. I. Title. QA76.8.P147W3 1983 001.64'25 83-12766
ISBN 0-8104-6283-4

Sharp® is a registered trademark of Sharp Corporation, and TRS-80™ is a trademark of Radio Shack, a division of Tandy Corporation. Neither company is affiliated with Hayden Book Company, Inc.

Copyright © 1983 by Nat Wadsworth. All rights reserved. No part of this book may be reprinted, or reproduced, or utilized in any form or by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying and recording, or in any information storage and retrieval system, without permission in writing from the Publisher.

Printed in the United States of America

1	2	3	4	5	6	7	8	9	PRINTING
83	84	85	86	87	88	89	90	91	YEAR

Preface

This is a book of practical, easy-to-use, ready-to-load-and-run programs for Sharp Electronics Corporation's PC-1500 Pocket Computer and the equivalent Radio Shack customized version TRS-80 PC-2 Pocket Computer.

It is not a tutorial book: It does not contain information on how to program or operate a pocket computer.

It does, however, contain a wealth of material. There are 50 programs covering a wide range of subjects. Each of these programs can immediately be applied to the everyday types of problems one is likely to encounter at work and around the home. There is also a good supply of programs you may simply use to enjoy during your leisure time.

Each program listing is accompanied by an explanation of its use and purpose. In many cases an actual example of the program's operation is provided for illustrative and checking purposes.

While this book is not designed to be a tutorial publication, much may be learned by examining the program listings. A variety of authors with diversified backgrounds have revealed within these pages their programming talents and techniques. You are free to select, try, modify, and adapt, as you see fit. That is the very essence of personal computing.

The advent of the pocket computer (PC) and the availability of pocket computer program libraries such as this, make computers about as personal as they can ever be—custom tailored by the individual, for the individual!

Nat Wadsworth

EQUIPMENT NEEDED

To run the programs in this book, you will need either a Sharp Model PC-1500 Personal Computer or a Radio Shack TRS-80 PC-2 Pocket Computer. For a few of the programs a printer/plotter and/or a memory expansion module will be needed.

Contents

1. INTRODUCTION	1
2. PROGRAMS FOR BUSINESS	6
2.1 Calendar	6
2.2 Amortization	8
2.3 Table/Chart Formatter	10
2.4 Printer's Helper	14
2.5 Trip Expenses	17
2.6 Data File	20
2.7 Payroll Taxes	27
2.8 Banners	30
3. MATHEMATICS PROGRAMS	32
3.1 Any Base Conversion	32
3.2 Polynomials	34
3.3 Roots of Polynomials	37
3.4 Fractions and Decimals	39
3.5 Prime Factors	41
3.6 Simultaneous Equations	43
3.7 Factorial!	45
3.8 Regression Analysis	46
4. SCIENCE AND ENGINEERING PROGRAMS	51
4.1 Fast Fourier Transform	51
4.2 Low-Pass Filter Design	55
4.3 Interpolation	57
4.4 Triangles	60
4.5 RPN Calculator	63
4.6 Curve Fitting	68
4.7 Z Plot	72
4.8 Histogram	75

5. PERSONAL PROGRAMS 81

- 5.1 Aviation 81
- 5.2 Supermarket 83
- 5.3 Moon Phases 85
- 5.4 Memo Pad 87
- 5.5 Weather Forecaster 92
- 5.6 Wind Chill 95
- 5.7 Temperature/Humidity Index 97
- 5.8 Morse Code 99
- 5.9 Simple Interest 102

6. PROGRAMS FOR ENTERTAINMENT 106

- 6.1 Blackjack 106
- 6.2 Hangman 109
- 6.3 Super-Wumpus 112
- 6.4 Rock, Paper, Shears 117
- 6.5 Word Squares 119
- 6.6 Vegas Craps 121
- 6.7 Jackpot Slots 124
- 6.8 Tic-Tac-Toe 127
- 6.9 Music 129
- 6.10 LCD Graphics 132
- 6.11 Notes 136
- 6.12 American Roulette 137

7. PROGRAMMER'S AIDS 140

- 7.1 Converting Bases 2 to 16 140
- 7.2 Memory Dump 143
- 7.3 Renumber 144
- 7.4 Sidelister 150
- 7.5 Sort 153

Introduction

A few years ago I started a newsletter devoted to the use and application of pocket computers. That publication was warmly received. Before long I was receiving from avid enthusiasts all over the world practical programs designed specifically to run on pocket computers. The best of these are published regularly in the pages of the *Pocket Computer Newsletter*. (Current subscription information may be obtained by writing to: PCN, P.O. Box 232, Seymour, CT 06483.) The publication of that newsletter has led to this book. As usual, there is a little story behind it all . . .

The pocket computer industry was started by Sharp Electronics Corporation, headquartered in Japan. In 1980 they introduced a revolutionary model called the PC-1211. It was a complete computer, capable of executing the BASIC language—and, it could be fit into a pocket. Soon, a functionally equivalent model was being marketed in the United States by Radio Shack (a division of Tandy Corporation). The Radio Shack version was originally dubbed the Radio Shack TRS-80 Pocket Computer. It was touted as an extension of Radio Shack's trademarked TRS-80 line of desktop personal computers. Later, to distinguish it from a more advanced second-generation pocket computer, the original PC was renamed model TRS-80 PC-1. The later and more advanced unit was then designated the TRS-80 PC-2.

This later model, the Radio Shack PC-2, was really just a customized version of Sharp Electronics' second-generation pocket unit called the PC-1500. From a programming and operational viewpoint there is virtually no difference between the PCs carrying the Sharp or Radio Shack brand names.

The Sharp PC-1500 and Radio Shack PC-2 pocket computers are, however, vastly enhanced versions of their predecessors. They sport a much more sophisticated and powerful implementation of the BASIC language. When the new models were introduced, they were billed as being "upwards compatible"

with the earlier units; that is, you could theoretically load programs written for the earlier units into the newer models and have them execute without modification.

Well, as is often the case involving high technology products, there were a few oversights. The most devastating, from a program compatibility viewpoint, turned out to be due to differences in the way array elements could be assigned in a program. The net result was that many sophisticated programs written for the Sharp PC-1211 and the Radio Shack PC-1 could not be directly transferred to the new PC-1500 and PC-2.

By the time the PC-1500 and PC-2 became available, many good, well-tested programs for the PC-1211 and PC-1 had been published in early editions of the *Pocket Computer Newsletter*. Many of these programs would be of great value running on the new PCs. The greatly enhanced operating speed made the prospects of such adaptations especially appealing. However, I began to find myself faced with a dilemma.

Many subscribers to *Pocket Computer Newsletter* had been with the newsletter since its beginning and were now old hands at programming pocket computers. Those that had obtained second-generation units were most interested in receiving current information and new programs for these more advanced models. They had indicated that they would not appreciate seeing the newsletter filled with slightly modified “re-runs” of programs previously published for the earlier models. Furthermore, this group of readers was well-qualified to adapt earlier programs to the new models.

On the other hand, newcomers to the field of pocket computing were being deprived of a wealth of programs that had originally been created for the early models. Most of these programs, with some judicious modifications, would be of considerable value running on the PC-2 and PC-1500 models.

So now you know where this book comes in! The obvious answer to the dilemma was to convert the programs of value to new machine owners from the early issues of the newsletter and publish them as a ready-to-use compendium. Newcomers would thus be able to instantly acquire a library of practical programs, already proven to be of value by extensive use in the field. Even old-timers who felt a little shaky about doing such

conversions would find such a compilation of value. There is nothing easier than “loading and running” when all the tough conversion work has already been done! Thus I went to work converting a collection of programs so that they would be ready to load and use on the new Sharp PC-1500 and Radio Shack PC-2 pocket computers. Also, in order to round the book out, I tacked on a few “super” programs written exclusively for the new models. These are not translations of programs written for the earlier PCs. Most notable among these bonuses are the Data File, Renumber, Histogram, and Morse Code programs included in this book.

For the most part, I served primarily as the editor and “translator” of the programs provided in this book, which is mostly a compilation of programs from a variety of authors. In fact, I think one of the appeals of this library of programs is the diversity of the programmers involved. You will observe in these pages many programming techniques and different approaches to having a computer perform tasks. The credit for the original development of these programs goes to the individual program authors who are identified in the text associated with each program.

Any errors or omissions in the translation process, of course, fall on my shoulders. While great care and actual testing of each translated program was performed, I am all too aware that in a project of this magnitude it is possible that some slip-ups could have occurred. If you come across such a case, we (the publisher and I) would appreciate your letting us know so that we can make the necessary corrections in future editions.

I know you are eager to try out some of these programs. Before you start, however, there are a few points that may be of assistance to you.

All of these programs are designed to be keyed into a Sharp Electronics PC-1500 or Radio Shack TRS-80 PC-2 pocket computer without further modification. The majority are designed to run in a “minimum configuration” system; that is, just a standardly equipped “stand-alone” pocket computer. A few of these programs, however, are designed to be used with the optional printer/plotter unit that is available for these pocket computers, and several of the packages require that an

additional “expansion” memory module be installed in the PC. The special requirements of these programs are indicated in the text. When you observe a long listing you should, of course, become suspicious that the program may require extra memory.

The actual number of bytes required for the storage of each program is indicated at the end of each program listing by the *STATUS 1* value. This value does not necessarily tell you whether or not you have enough memory to execute a particular program. This is because the *STATUS 1* value measures only the amount of memory consumed by storage of the program. It does not account for additional memory that may be required for the storage of certain variables and for any variable array elements. When in doubt, or when all else fails, read the text accompanying each program. (Really, I recommend that all the time, but experience has taught me that in their anxiety to use the programs some purchasers of this kind of material tend to skip over the related text matter.)

It is assumed in this book that you are at least somewhat familiar with the operation of your PC. I do not provide a tutorial on how to operate your unit. There is, however, little more to applying these programs than loading them (with your PC in the *PROgram* mode) and then executing them (with your PC in the *RUN* mode). Some programs make use of *DEFinable* keys to access different portions and operations. These options are explained, when applicable, in the text accompanying each program.

Occasionally, you may find a program line that is very long (more than 80 characters) in the listing. In order to load such a line you must remember to abbreviate statement keywords and perhaps eliminate spaces (except within quoted strings) as you key in the line. Remember, keying *P.* or *PR.* in a program line is the same as keying the entire statement keyword *PRINT*, but it takes less space and time. When you press the *ENTER* key at the end of a program line, all such abbreviated keywords will be expanded to their full names. The fully expanded line is what is presented in the program listings.

When you have loaded a program, check the *STATUS 1* value you obtain against that shown at the end of the listing. If it does not match, recheck your work.

Remember, if you are keying in a program that contains keywords used by the optional printer/cassette unit, you should have the PC plugged into the unit. Otherwise, the PC will not be able to properly interpret the keywords.

I strongly recommend that once you have a program loaded and checked out, you save it on a cassette tape. Yes, you have permission to make such copies for your own personal, archival purposes. Of course, we do hope you will respect our copyrights. If you like this book, tell your friends with similar interests to buy a copy. That way we (the publisher, all the programmers whose programs appear in this book, and I) will be encouraged to produce more such material. It has taken many hundreds of work-hours by many people to create, program, and publish this material. Much of this is done with the fervent hope that it will be appreciated by you.

Programs for Business

2.1 CALENDAR

Do you need a monthly calendar for any period between March 1700 and February 2200? This program will provide it!

Emerich Auersbacher came up with this little gem. Operation is simplicity illustrated. Just load the program, place the PC in the RUN mode and execute the *RUN* command. When prompted, enter the number of the month desired (counting January as month number 1, February as number 2, . . . , December as month 12). The next prompt asks for the year in question. Enter that using four digits, such as 1983.

In just a few seconds your little machine will “beep” and then show the days of the week to let you know it is ready to present the calendar for the month of interest. Press the ENTER key to see each week in sequence.

Here is a sample run-through:

```
MONTH: 6
YEAR: 1983
S   M   T   W   T   F   S
-   -   -   1   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
```

If you would like to make hardcopies of the output from this program, all it takes is a few minor program alterations. First add a line such as:

115 CSIZE 1:COLOR 2

Of course, you can put in whatever pen color you choose in the event you do not like *COLOR 2*.

Now, change the *PRINT* statements in lines 120 through 145 to *LPRINT* statements. Bingo! That is all you need to do to drive the standard PC printer interface. The initial prompts for the month and year of interest will still show up on the liquid-crystal display (LCD), but the actual calendar of the month will be briskly printed out on the printer for your permanent records.

A few reminders—always turn off both the PC and the printer/cassette unit when plugging or unplugging the PC and the accessory together. Failure to do so can result in loss of the program from the PC or cause other types of malfunctions!

One final note—you might want to keep two copies of this program in your files (on cassette, I hope!); one for use with the LCD on the pocket computer for quick reference and the other for driving the printer unit.

Program listing: CALENDAR

```

10: CLEAR : DIM A$(
    73)*3: A$(6)="
    1": A$(7)=" 2
    ": A$(8)=" 3":
    A$(9)=" 4": A$
    (10)=" 5"
15: A$(11)=" 6": A
    $(12)=" 7": A$
    (13)=" 8": A$(
    14)=" 9": A$(1
    5)=" 10": A$(16
    )=" 11"
20: A$(17)=" 12": A
    $(18)=" 13": A$
    (19)=" 14": A$(
    20)=" 15": A$(2
    1)=" 16"
25: A$(22)=" 17": A
    $(23)=" 18": A$
    (24)=" 19": A$(
    25)=" 20": A$(2
    6)=" 21": A$(27
    )=" 22"
30: A$(28)=" 23": A
    $(29)=" 24": A$
    (30)=" 25": A$(
    31)=" 26": A$(3
    2)=" 27": A$(33
    )=" 28"
35: A$(34)=" 29": A
    $(35)=" 30": A$
    (36)=" 31"
40: WAIT : INPUT "M
    ONTH: "; A, "YEA
    R: "; B: E=62104
    8: C=365.25
45: IF A<=2LET D=
    INT (30.6*(A+1
    3))+INT (C*(B-
    1))-E: GOTO 55
50: D=INT (30.6*(A
    +1))+INT (B*C)
    -E
55: IF D>146097LET
    D=D-1
60: IF D<=73047LET

```

```

D=D+1:IF D<=36
522LET D=D+1
70:D=D/7:D=INT (7
*(D-INT D)+1.4
9)
75:IF (A=4)+(A=6)
+(A=9)+(A=11)
LET C=30
80:IF (A=1)+(A=3)
+(A=5)+(A=7)+(
A=8)+(A=10)+(A
=12)LET C=31
90:IF A=2LET C=28
:IF (B/4=INT (
B/4))+(B/100<
INT (B/100))=2
LET C=29
95:IF A=2IF B=200
0LET C=29
105:FOR E=1TO D-1:
A$(E+36)=" - "
:NEXT E
110:FOR E=1TO C:A$
(D+E+35)=A$(E+
5):NEXT E
120:BEEP 3:PRINT "

S M T W T
F S"
125:PRINT A$(37);A
$(38);A$(39);A
$(40);A$(41);A
$(42);A$(43)
130:PRINT A$(44);A
$(45);A$(46);A
$(47);A$(48);A
$(49);A$(50)
135:PRINT A$(51);A
$(52);A$(53);A
$(54);A$(55);A
$(56);A$(57)
140:PRINT A$(58);A
$(59);A$(60);A
$(61);A$(62);A
$(63);A$(64)
145:PRINT A$(65);A
$(66);A$(67);A
$(68);A$(69);A
$(70);A$(71):
PRINT A$(72);A
$(73)
150:GOTO 10
STATUS 1 1215

```

**Sample output for: MONTH 6
YEAR 1983**

```

S M T W T F S
- - - 1 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

```

2.2 AMORTIZATION

So you decided to buy a pocket computer instead of a desktop machine, and you are saving your money to purchase a house? Let me tell you, fluctuating interest rates can raise havoc with the payment schedules on any such investment. Just how much a few points change in the interest rates can impact upon plans can be clearly presented by this compact but powerful program.

Suppose you needed a \$40,000.00 mortgage over a period of 30 years (360 months), and the bank said they would loan the money at a rate of 12 percent interest. Just load the program and

execute it in the RUN mode. Answer the prompts, and quick as a flash you will know the costs right down to the penny!

```

AMORTIZATION SCHEDULE
LOAD AMOUNT $: 40000
PERIODS: 360
INTEREST RATE (%): 12
PERIODIC PMT= $ 411.44
PERIOD 1
I= 399.99 P= 11.44
BAL=$ 39988.55
PERIOD 2
I= 399.88 P= 11.55
BAL=$ 39976.99
.
.

```

Just remember, this program assumes that periods will be specified as the number of months, but the interest rate will be entered as an annual rate. This annual rate is automatically converted to its monthly equivalent within the program.

Oh sure, you don't have to use it just for buying a house. It can give you the low-down (or high-up!) on purchasing a car, furniture, a vacation, or anything paid for on an installment basis.

Program listing: AMORTIZATION

```

5: BEEP 1: PAUSE "
  AMORTIZATION
  SCHEDULE"
10: CLEAR : INPUT "
  LOAN AMOUNT$:
  "; U, "PERIODS:
  "; N, "INTEREST
  RATE (%): "; Y
15: Y=Y/100: Y=Y/12
20: P=U*(Y/(1-(1+Y
  )^(-N)))
25: BEEP 1
30: PRINT "PERIODI
  C PMT= $";
  USING "#####.#
  #"; P
35: L=U
40: FOR Z=1 TO N
  STEP 1: USING
45: K=(1+Y)^(-Z): B
  =1/K*(P*(K-1)/
  Y+U): IF B<.001
  LET B=0
48: BEEP 1: PAUSE "
  PERIOD "; Z
50: I=B-L+P: L=B: R=
  P-I
60: USING "#####.#
  #"
65: PRINT "I="; I; "
  P="; R: USING
  "#####.#"
68: PRINT "BAL=$";
  B
70: NEXT Z
80: END
STATUS 1 386

```


If you want a real eye opener, try the same example given above, only change the interest rate to 16 percent. Whew! Quite a difference in those monthly payments, isn't there?

This is another example of efficient programming for the pocket computer by that venerable expert Emerich Auersbacher.

2.3 TABLE/CHART FORMATTER

If you have ever had to type charts or tables of data, you know how frustrating it can be to arrange columns neatly on a page. Where should you set the tabs to obtain even spacing between columns? How wide should the margins be to facilitate a nice presentation? Usually it takes several attempts by the old standby method, trial and error. Your pocket computer and this program, however, can put an end to the guesswork formerly associated with laying out such charts and tables.

This program enables you to plan layouts containing columns that are of the same or of varying widths. It also compensates for the use of Elite type (102 characters across a standard 8½-inch-wide piece of paper) or Pica elements (having just 85 characters across a standard page). If you wish, you can also vary the width of the paper. You can even consider simple modifications that would allow other font sizes to be accommodated.

The Setup

To use this program effectively you need to do just a little bit of preliminary planning with regard to the following information:

1. The number of columns you want to have across the sheet of paper.
2. The number of characters within each column (maximum), so that you can obtain the total number of characters for all of the columns.
3. The width of the paper (the program defaults to an assumed width of 8½ inches).

Now, that isn't so bad, is it? Here is an example of the program's operation. For the sake of illustration, there will be six

columns, each containing six character positions, and the paper will have the standard 8½-inch width.

Load the program into the PC and place it in the RUN mode. The DEF key is used in conjunction with specific alphabetical keys to select desired operations. Start the process by pressing the DEF key and then the character C key. (This type of combination will be shown as DEF/C throughout this discussion.) Now respond to the prompts as illustrated:

Computer Displays	Your Action
RETURN TO INITIALIZE PGM	Press the ENTER key.
PICA OR ELITE (P/E)?	Enter E and press ENTER.
TABULATION DESIGNER	Program is now in the regular operating mode. You do not need to use the initialize mode again unless you wish to change type sizes. Press the ENTER key.
USING ELITE TYPE.	Press the ENTER key.
HORIZONTAL INCHES?	Press ENTER to have the program default to the standard 8-½ inch paper assumed for this example.
USING 8.5 INCHES	Press ENTER.
PREF. LEFT MARGIN.	The program wants to know how wide (number of characters) you want the margins? The program warns you if you make them too wide. Enter the value 10 for this example, followed by pressing ENTER.
HOW MANY COLUMNS?	Input a value of 6 for this example, then press ENTER.
ALL THE SAME SIZE?	Press Y and ENTER for this case.
COLUMN WIDTH (SPACES)?	There are six character positions in each column, so enter 6 and press ENTER.

Computer Displays	Your Action
MAX. SPCS./COL.=9.2	There is an answer! There will be about 9 spaces between each column.

“Aha!” you say. The spacing given in the example is not an even integer value, and your typewriter doesn’t offer fractional spacing! Well, friends, you could just round off to the nearest integer and still end up with a fairly well-spaced page. But if that doesn’t suit your fancy, then you can continue to use the program to refine the between-column spacing. You could, for instance, redefine the margin spacing as illustrated next:

Computer Displays	Your Action
RE-DEFINE MARGIN (Y/N)?	Input Y and press ENTER.
OLD MARGIN=10.	Isn’t that nice? It reminds you of your previous margin value. Press the ENTER key.
NEW MARGIN?	Let’s try a value of 8 this time, followed by ENTER.
MAX. SPCS./COL.=10	How convenient. Now we have a nice integer value between columns. But, there is more. Press ENTER.
RE-DEFINE MARGIN (Y/N)?	Say N since the 10 spaces between columns seems acceptable. ENTER.
LEFT MARGIN=8	The program reminds you where the first column starts.
COLUMN # 2 = 24	Press ENTER.
COLUMN # 3 = 40	Press ENTER.
COLUMN # 4 = 56	Press ENTER.
COLUMN # 5 = 72	Press ENTER.
COLUMN # 6 = 88	There you have it—the exact positions at which to start each column. Press ENTER.

Computer Displays	Your Action
---END RUN	If you press ENTER here, you will return to the starting point (assuming the same size type) so you can plan another chart/graph! Alternatively, you could press the DEF/SPACE combination at any time to restart calculations.

If you do statistical typing, accounting work, or design report formats for computerized jobs, this program can save you a lot of trial-and-error effort. Many thanks to Ken Slaughter for this fine labor-saving application of the PC.

Program listing: TABLE/CHART FORMATTER

```

1: "C" WAIT : PRINT
  "RETURN TO INITIALIZE PGM":
  CLEAR : DIM A(20)
2: Q$="X": INPUT "
  PICA OR ELITE (P/E)? "; Q$
3: IF Q$="E" LET S=12: Q$="ELITE"
4: IF Q$="P" LET S=10: Q$="PICA"
5: IF Q$="X" PRINT "ERROR!": GOTO 2
6: " PRINT "TABULATION COMPUTE
  R"
7: PRINT "USING " ; Q$ ; " TYPE."
8: H=8.5: INPUT "HORIZONTAL INCHES? "; H
9: PRINT "USING " ; H ; " INCHES."
10: H=S*H
11: INPUT "PREF. LEFT MARGIN? "; M
12: INPUT "HOW MANY COLUMNS? "; T
    IF (T<1)+(T>20) THEN 12
13: INPUT "ALL SAME SIZE (Y/N)? "; R$
14: IF R$="N" GOTO 18
15: INPUT "COLUMN WIDTH (SPACES)? "; W
16: FOR I=1 TO T: A(I)=W: NEXT I
17: GOTO 23
18: FOR I=1 TO T
19: PAUSE "COLUMN #"; I
20: INPUT A(I): IF A(I)>=16 GOTO 22
21: GOTO 19
22: NEXT I
23: X=0
24: FOR I=1 TO T
25: X=X+A(I)
26: NEXT I
27: W=X+(T-1)
28: IF W<H GOTO 30
29: PRINT "MIN. SPACES REQD: "; W
    GOTO 6
30: D=H-(2*M)
31: IF D>X+(T-1) GOTO 38
32: PRINT "MARGIN

```

```

    TOO LARGE..."
33: U=D-(X+(T-1))/
    2: PRINT U; "=MA
    X. AVAIL. MARG
    IN"
34: N=0: INPUT "PLE
    ASE RE-ENTER M
    ARGIN: "; N
35: IF N=0 GOTO 34
36: M=N
37: GOTO 30
38: U=(D-X)/(T-1)
39: PRINT "MAX SPC
    S/COL="; U
40: R$="X": INPUT "
    RE-DEFINE MARG
    IN (Y/N)? "; R$
41: IF R$="X" THEN
    40
    42: IF R$="N" THEN
    46
    43: PRINT "OLD MAR
    GIN="; M
    44: N=0: INPUT "NEW
    MARGIN? "; N
    45: M=N: GOTO 27
    46: PRINT "LEFT MA
    RGIN="; M
    47: L=M
    48: FOR I=2 TO T
    49: L=L+U+A(I-1)
    50: USING "####":
    PRINT "COLUMN
    #"; I; " ="; L
    51: NEXT I
    52: PRINT "---END
    RUN": GOTO 6
STATUS 1      1115

```

2.4 PRINTER'S HELPER

Here is a program that will be considered a blessing by anyone involved in the business of printing, publishing, typesetting, or running an art studio.

The program has two sections. One is selected using DEF/A (when the PC is in the RUN mode with the program previously loaded into memory). This section is used to size galleys after selecting an appropriate type font, size, and line spacing (leading). A second section is called up using DEF/B. It is used to scale artwork and photographs.

Sizing Galleys

When the typesetting aid is started, using DEF/A, the program prompts the user to input the type point-size, leading, and the lengths of the lines proposed for the typeset copy. Typical inputs for these parameters might be: 10 point type on 11 point leading with a line length of 22 picas.

Next, the program asks if the user wishes to *COMPENSATE* the copy. If the response is Y (for yes), then the program deducts five characters from the number computed to fit on each typeset line. This value, five, is the average number of characters that will typically be lost when setting justified or ragged-right copy. If

compensation is not requested, then the program emulates most manual typefitting procedures.

The program then asks the operator to indicate the *TYPE STYLE* to be used in its calculations. Select one of the following abbreviations to indicate the desired type style font:

ENG English Times
HEL Helios
HELC Helios Condensed
HELL Helios Light
CLE Clearface
KOR Korina
ORA Oracle
SOU Souvenir

If the typeface you want to use is not among those in the above list, then input *MAN* (for manual) and respond to the prompt:

CHRS. PER PICA?

with an appropriate value to two decimal places (such as 2.43).

By the way, remember that characters-per-pica (CPP) values can vary among phototypesetter manufacturers and typesetters. To customize this program for different CPP values, use the reciprocal of the 10 point value in variable C. (Refer to lines 130–200 of the program for illustration.) Thus, if your version of 10 point Helios had a CPP of 2.43, you would set C equal to the value $(1*0.1)/2.43$ or 0.04115. The values given in this program are for typefaces made by Compugraphic Corporation.

When the appropriate information has been given, the program proceeds to churn out the calculated number of typeset lines and the copy depth in points and inches. If desired, it will then automatically loop back to begin another set of calculations.

Scaling Artwork

Pressing DEF/B calls up a handy, compact routine for sizing artwork. The program prompts for the dimensions of the original

photograph or drawing in inches and tenths. You can then have the PC calculate the percent reduction or enlargement needed to obtain a desired height or width. Alternatively, given a percentage change desired, the program will output new height and width figures.

Try it—you'll like it!

Program listing: PRINTER'S HELPER

```

20: "A" CLEAR
30: INPUT "CHRS PE
    R MMS LINE? ";
    B
40: INPUT "# OF MM
    S LINES? "; A
50: INPUT "POINT S
    IZE? "; S
60: INPUT "LEADING
    ? "; L
70: INPUT "LINE LN
    GTH/PICAS? "; D
80: INPUT "COMPENS
    ATE (Y/N)? "; Z
    $
90: INPUT "TYPE ST
    YLE? "; Q$
100: IF Z$="Y" LET H
    =5
110: IF Z$="N" LET H
    =0
120: IF Q$="MAN"
    GOTO 290
130: IF Q$="ENG" LET
    C=1/(S*.03731)
140: IF Q$="HEL" LET
    C=1/(S*.04115)
150: IF Q$="HELC"
    LET C=1/(S*.03
    344)
160: IF Q$="HELL"
    LET C=1/(S*.03
    802)
170: IF Q$="CLE" LET
    C=1/(S*.03322)
180: IF Q$="KOR" LET
    C=1/(S*.04)
190: IF Q$="ORA" LET
    C=1/(S*.03861)
200: IF Q$="SOU" LET
    C=1/(S*.03571)
210: IF C=0 PRINT "B
    AD ENTRY": GOTO
    90
220: E=(A*B)/(C*D+H
    )+.9: E=INT E
230: F=L*E: G=F/72
240: PRINT "# TYPES
    ET LINES "; E
250: PRINT "COPY DE
    PTH/PNTS "; F
260: PRINT "COPY DE
    PTH/INCHES ";
    USING "####.##
    "; G
270: USING
280: GOTO 30
290: INPUT "CHRS. P
    ER PICA? "; C:
    GOTO 220
305: "B" CLEAR
310: INPUT "ORIGINA
    L WIDTH? "; J
320: INPUT "ORIGINA
    L HEIGHT? "; K
325: L=0: M=0: N=0
330: INPUT "WIDTH W
    ANTED? "; L
335: IF L LET Q=L/J:
    P=Q*K: O=L: Q=Q*
    100: GOTO 360
340: INPUT "HEIGHT
    WANTED? "; M
345: IF M LET Q=M/K:
    O=Q*J: P=M: Q=Q*
    100: GOTO 360
350: INPUT "% WANTE
    D? "; N

```

```

355:N=N*.01:O=J*N:          380:PRINT "PERCENT
    P=K*N:Q=N*100            = ";Q
360:USING "####.##          390:GOTO 325
    "                        400:END
370:PRINT "W=";O;"          STATUS 1      1010
    H=";P

```

2.5 TRIP EXPENSES

This program, engineered by Jack Forbes, is designed to keep track of the various expenses one is likely to incur during an automobile trip. It will tally such parameters as your average miles per gallon for an entire trip, the miles per gallon for each fillup, the highest and lowest number of miles per gallon (mpg) during a trip, total trip mileage, gasoline and oil expenses, and expenses in several other categories. Here is a list of the various categories available:

- GAS** Gasoline. A subcategory keeps track of oil usage/expense.
- MEAL** Food expenses at regular meal stops.
- SNACK** Records those extra "between meals" expenses.
- ROOM** Lodging costs.
- FUN** Another popular name for this is "entertainment."
- MISC** What isn't accounted for in the other categories.

Once this program has been loaded into memory and the PC placed in the RUN mode, appropriate functions are called up using the DEFine key and an appropriate label key. The four labels assigned to this program are as follows:

- DEF/N** This entry point is used at the start of a trip. It is used to clear variables.
- DEF/C** Is used to continue (add) entries during a trip.
- DEF/=** Calls up current totals for the various categories.
- DEF/L** Is used to recall high and low mpg figures.

Are you ready to make a trip? Okay, fill up your gas tank. Now fire up your pocket computer, and select DEF/N. Enter the starting mileage from your odometer. If there are no other expenses at this point, you may shut your PC off. Remember, you

do not enter the cost of the initial tank of gas. Ideally, however, you will have topped off your tank at the conclusion of your previous trip. The cost of that last fillup would be considered a part of previous trip expenses.

When you incur an expense during the trip, just whip out your PC, turn the power on, and select DEF/C. When the message:

ENTER EXPENSE NAME

appears, enter the appropriate category such as *GAS*, *MEAL*, *ROOM*, or whatever (it must exactly match one of the categories previously listed).

Then simply respond to the appropriate prompts presented by the computer. For example, if you indicated the category *GAS*, the program would ask for the present mileage, the number of gallons purchased, the price, whether or not you bought oil, and if so, the expenditure for that. When you have finished making entries, you can just power-down the PC or let it time-out and turn itself off.

When you want to examine totals for a trip, use DEF/= . It is a good idea to do this between gas purchases. Then you will not miss the reporting of a new high or low mpg rating.

If you start using this program on a regular basis, you may be quite surprised to find out how fast various kinds of extra and miscellaneous expenses add up. (Which is worse, not knowing where it is all going, or knowing?)

Program listing: TRIP EXPENSES

```
5: "N"PAUSE "
   TRIP EXPENSES
   "
10: BEEP 1:PAUSE "
   START WITH F
   ULL TANK":
   CLEAR :INPUT "
   ENTER STARTING
   MILEAGE ";K:G
   =K
15: "C"INPUT "ENTE
   R EXPENSE NAME
   ";A$:GOTO A$
20: "GAS"INPUT "EN
   TER PRESENT MI
   LEAGE ";F,"HOW
   MANY GALLONS
   ";D:C=C+D:IF D
   =0THEN 60
25: INPUT "WHAT WA
   S THE COST $";
   B:E=E+B:INPUT
   "DID YOU BUY O
   IL Y/N? ";A$:
```



```

        IF A$="N" THEN
        60
30: INPUT "    HOW
    MANY QUARTS? "
    ;B:L=B+L: INPUT
    "    COST OF OI
    L $";B:Q=B+Q:
    GOTO 60
35: "MEAL" INPUT "E
    NTER COST OF M
    EAL $";B:Q=B+Q
    ;GOTO 60
40: "FUN" INPUT "EN
    TER FUN COST $
    ";B:M=B+M: GOTO
    60
45: "MISC" INPUT "
    ENTER MISC COS
    T $";B:N=B+N:
    GOTO 60
50: "ROOM": INPUT "
    ENTER ROOM COS
    T $";B:R=B+R:
    GOTO 60
55: "SNACK" INPUT "
    ENTER SNACK CO
    ST $";B:P=B+P
60: INPUT "ARE TOT
    ALS WANTED Y/N
    ? ";A$: IF A$="
    N" THEN 15
65: "=" I=F-K: IF I<
    =0 THEN 75
70: PRINT " TOTAL
    TRIP MILES";I
75: IF (D=0)+(C=0)
    THEN 125
80: B=I/C: PRINT "
    GRAND TOTAL MP
    G=";B:H=F-G:B=
    H/D: IF B<=0
    THEN 125
85: G=F: PRINT "TRI
    P MILES PER GA
    L";B: IF S<>0
    GOTO 95
90: S=B:T=B
95: IF B<S GOTO 110
100: IF B>T GOTO 115
105: GOTO 120
110: BEEP 2:S=B:
        PRINT " NEW LO
        W MPG":GOTO 12
        0
115: BEEP 2:T=B:
        PRINT "NEW HIG
        H MPG":GOTO 12
        0
120: "L"U=INT (S*10
        0)/100:PRINT "
        LO MPG=";U;"
        HI=";T
125: IF C=0 THEN 135
130: PRINT " GAS TO
        TAL IS";C;" GA
        LS"
135: IF E=0 THEN 145
140: PRINT " TOTAL
        GAS COST $";E
145: IF L=0 THEN 155
150: PRINT " TOTAL
        OIL USED";L;"
        QTS"
155: IF Q=0 THEN 165
160: PRINT " TOTAL
        OIL COST $";Q
165: IF O=0 THEN 175
170: PRINT " TOTAL
        MEALS COST $";
        O
175: IF P=0 THEN 185
180: PRINT " TOTAL
        SNACK COST $";
        P
185: IF M=0 THEN 195
190: PRINT " TOTAL
        FUN COST $";M
195: IF N=0 THEN 205
200: PRINT " TOTAL
        MISC COST $";N
205: IF R=0 THEN 215
210: PRINT " TOTAL
        ROOM COST $";R
215: B=E+Q+O+R+N+M+
        P: IF B=0 THEN 2
        25
220: PRINT " TOTAL
        TRIP COST $";B
225: IF I<=0 THEN 24
        0
230: J=B/I: IF J=0
        THEN 240

```

```
235: J=INT (J*1000)
      /1000: PRINT "
      *COST PER MILE
      * $"; J
```

```
240: PRINT "      TOT
      ALS COMPLETED"
      : GOTO 15
STATUS 1      1520
```

2.6 DATA FILE

There is more power in a pocket computer than some people realize. This program, produced by Stephen Tomback, illustrates some of the remarkable capabilities of the PC for practical applications. The program presented is a complete data-file maintenance program.

This program enables you to set up a computerized filing system in which you define the specific needs for your application as they relate to record size, number of fields, and field length. The file you create resides in memory when in use. The number of records that can be present in a file depends on the record size as dictated by the number and length of the fields that you define. Once a file has been formatted, you may add records, change their contents, and retrieve records of interest. An entire file may be saved on a cassette for safekeeping and later recall.

This is a relatively large (and powerful) program. In order to use it you will need to equip your PC with an 8K memory module and a printer/cassette unit.

Here is a summary of the capabilities of this program, showing the maximum number and sizes of records and fields allowed:

Record Length 80 bytes maximum.

Number of Records 255 maximum (80 maximum if using 80 byte records with 7 fields per record).

Fields per Record 20 maximum

File Name 16 bytes (fixed) maximum.

Field Name 16 bytes (fixed) maximum.

Field Length 80 bytes maximum.

Field Search Maximum of 5 fields at one time.

To start using the program, load it into memory and place the PC in the RUN mode. You can start the program when there

is no data in memory (as will be explained shortly) using a normal *RUN* command. Once you have data in memory, you should use the directive: *GOTO 20*. Entering the program at this point will keep the current data file intact.

Once the program has been placed in operation a menu will begin to continuously cycle on the LCD. The menu presents the following selections:

- S** Set up a new data file (erases current file).
- O** Output the current file to tape.
- I** Input a file from tape.
- 1** Print current file parameters.
- A** Add a record to the current file.
- P** Print a record.
- F** Find record(s) in the file.
- C** Change a record in the file.

Pressing any key representing a selection causes the program to immediately branch to perform the desired operation. The program will prompt you for each input as necessary. An invalid input will cause the prompt to be repeated. Exceeding programmed limits causes an appropriate error message to be displayed. When an operation has been completed, the program resumes display of the menu.

If you want to stop the program, press the **BREAK** key. However, restarting the program with a *RUN* directive will result in the current data file being cleared. Should you want to avoid this, you can re-enter the program using the previously mentioned *GOTO 20* statement. This will cause the program to resume displaying its menu without disturbing its data arrays.

Here is an example of how you can establish a data file for storing names and addresses after pressing the **S** key as the main menu is flashing its selections:

```
NEW FILE NAME: SAMPLE
HOW MANY RECORDS: 80
FIELDS PER RECORD: 5
FIELD 1 NAME: NAME
HOW MANY BYTES: 20
FIELD 2 NAME: ADDRESS
HOW MANY BYTES: 24
```

```
FIELD 3 NAME: CITY/ST
HOW MANY BYTES: 12
FIELD 4 NAME: ZIP
HOW MANY BYTES: 5
FIELD 5 NAME: DATE
HOW MANY BYTES: 8
```

At this point, having finished prompting for the record-defining criteria, the program resumes the display of its options menu. You can check the file parameters at this point by pressing the digit 1 on the keyboard. The printer responds by outputting:

```
FILENAME : SAMPLE
NO OF RECORDS : 80
RECORDS USED : 0
BYTES PER RECORD : 69
MEMORY LEFT : 698
```

FIELD 1 NAME	BYTES 20
FIELD 2 ADDRESS	BYTES 24
FIELD 3 CITY/ST	BYTES 12
FIELD 4 ZIP	BYTES 5
FIELD 5 ZIP	BYTES 8

When the printout has been completed, the program once again reverts back to displaying its menu. At this point you can press the letter A to select the option to add records to the file. After records have been added you can print them (using P), change their contents (using C), save the file on tape (using O), or search the file for a particular record or groups of records (using F).

The latter option is quite intriguing, as you may establish a different search string in each of up to five fields. Just respond to the prompts for the number of fields to be searched, the number of each field to be examined, and the search string to be used in that field. It is important to remember that the search technique is a “logical AND” process. In other words, if you specify three search fields, a record must contain a match with the search string associated with each field in order for the record to be outputted. A match in only one or two fields will not result in a record match. Furthermore, it is important to realize that the search string specified will be considered a match if it occurs

anywhere within a specified field. Experimentation with the search option (F) will quickly yield a better understanding of its operation and reveal its full potential for extracting information from a data file.

It must be pointed out that while a separate delete option was not provided in order to conserve program space, it is easy to reuse records. When a record is no longer needed, change one of its fields so that it contains the word *DELETED* or a code to that effect. Then, when space is needed for a new record, use the *FIND* function to locate a “deleted” record. Once located, the *CHANGE* operation may be used to edit its contents so that it represents a new record.

Remember that when in the *ADD* or *CHANGE* mode, responding to a prompt for input with just the *ENTER* key alone will result in that particular field being filled with spaces. You can locate records that contain space-filled fields using the *FIND* option.

Finally, for you experimenters who may be interested in modifying or enhancing the basic program, here is a list of the critical variables used in the package along with their key uses:

- F\$** Filename.
- N\$()** Field names array.
- R\$()** Records array.
- L()** Field lengths array.
- F** Number of fields per record.
- R** Number of records per file.
- X** Length of records (bytes).
- U** Number of records currently used.

You may want to make note of the fact that this program locks the PC into the *RUN* mode once the program has been started. If you decide that you want to go into the *PRO* mode in order to make alterations, you will first need to execute an *UNLOCK* directive. Refer to your PC instruction manual if you are not familiar with this command.

How many new uses can you find for your pocket computer now that you have a program with this kind of versatility?

Program listing: DATA FILE

```

1:REM File maint
  enance
2:REM FM 10/11/8
  2 12/04/82
10:REM MENU
15:CLEAR :LOCK :
  DIM U$(0)*80
20:CLS :WAIT 0:E=
  8:K=0:RESTORE
  :FOR I=1TO E:
  READ U$(0):
  PRINT U$(0)
25:FOR J=1TO 60:U
  $=INKEY$ :IF U
  $<>" "LET J=60
30:NEXT J
35:IF U$=""THEN 5
  5
40:FOR J=1TO E:IF
  MID$ ("SOI1APF
  C",J,1)=U$LET
  K=J:J=E
45:NEXT J
50:IF KCLS :ON K
  GOSUB 100,400,
  500,600,700,85
  0,900,1000:
  GOTO 20
55:NEXT I:GOTO 20
60:DATA "S - SETU
  P NEW FILE","O
  - OUTPUT FILE
  ","I - INPUT F
  ILE"
65:DATA "1 - PRIN
  T FILE PARAMET
  ERS","A - ADD
  A RECORD","P -
  PRINT A RECOR
  D"
70:DATA "F - FIND
  RECORD(S)","C
  - CHANGE A RE
  CORD"
100:REM NEW FILE
105:CLEAR :INPUT "
  NEW FILE NAME:
  ";F$
110:IF LEN F$<1
  THEN 105
115:INPUT "HOW MAN
  Y RECORDS: ";R
120:IF R<1OR R>255
  THEN 115
125:INPUT "FIELDS
  PER RECORD: ";
  F
130:IF F<1OR F>20
  THEN 125
135:DIM N$(F-1):
  DIM L(F-1)
140:FOR I=1TO F
145:CLS :PRINT "FI
  ELD";I;" NAME:
  ":CURSOR 14:
  INPUT " ";N$(I-
  1)
150:IF LEN N$(I-1)
  <1GOTO 145
155:CLS :INPUT "HO
  W MANY BYTES:
  ";L(I-1)
160:IF L(I-1)=0
  THEN 155
165:IF X+L(I-1)<81
  LET X=X+L(I-1)
  :GOTO 175
170:PRINT 80-X;" B
  YTES LEFT":
  BEEP 15:GOTO 1
  55
175:NEXT I
180:IF STATUS 3-
  STATUS 2-R*X-1
  81<1THEN 190
185:DIM R$(R-1)*X:
  R$(0)="" :DIM I
  $(0)*80:DIM U$
  (0)*80:RETURN
190:PRINT "NOT ENO
  UGH MEMORY":
  BEEP 15:GOTO 1
  05
400:REM OUTPUT
405:IF LEN F$=0
  GOSUB 2000:

```



```

RETURN
410: INPUT "TAPE RE
      ADY? Y to outp
      ut "; I$
415: IF I$ <> "Y" THEN
      410
420: PRINT #F$; F, R,
      X, U
425: I$ = F$ + "*"
430: PRINT #I$; N$(X
      ), R$(X), L(X):
      RETURN
500: REM INPUT
505: CLEAR : INPUT "
      INPUT FILE NAM
      E: "; F$
510: INPUT #F$; F, R,
      X, U
515: DIM N$(F-1):
      DIM R$(R-1)*X:
      DIM L(F-1)
520: I$ = F$ + "*"
525: INPUT #I$; N$(X
      ), R$(X), L(X)
530: DIM I$(0)*X:
      DIM U$(0)*X:
      RETURN
600: REM PARAMETERS
605: IF LEN F$=0
      GOSUB 2000:
      RETURN
610: TEXT : CSIZE 1:
      LF 5
615: LPRINT "FILENA
      ME : "; F$
620: LPRINT "NO OF
      RECORDS : "; R
625: LPRINT "RECORD
      S USED : "; U
630: LPRINT "BYTES
      PER RECORD : ";
      X
635: LPRINT "MEMORY
      LEFT : ";
      STATUS 3-
      STATUS 2: LF 1
640: FOR I=1 TO F:
      LPRINT "FIELD"
      ; I; " "; N$(I-1)
      ; TAB 28; USING
      "###"; "BYTES";

```

```

      L(I-1): USING
645: NEXT I: LF 6:
      RETURN
700: REM ADD
705: IF LEN F$=0
      GOSUB 2000:
      RETURN
710: FOR I=0 TO R-1:
      REM FIND 1ST A
      VAILABLE RECOR
      D
715: IF LEN R$(I)=0
      THEN 725
720: NEXT I: PRINT "
      NO MORE RECORD
      S!": BEEP 15:
      RETURN
725: FOR J=0 TO F-1
730: CLS : I$(0) = "";
      PRINT N$(J); ":
      ": CURSOR LEN
      N$(J)+2: INPUT
      "; I$(0)
735: IF LEN I$(0) > L
      (J) THEN 760
740: IF LEN I$(0) = L
      (J) THEN 750
745: I$(0) = I$(0) + "
      ": GOTO 740
750: R$(I) = R$(I) + I$
      (0): I$(0) = ""
755: NEXT J: U = U + 1:
      CLS : PRINT "AD
      D RECORD"; U:
      BEEP 15: RETURN
760: GOSUB 3000:
      GOTO 730
800: REM OUT TO PRI
      NTER
805: TEXT : CSIZE 1:
      LF 1: J=0
810: LPRINT "RECORD
      : "; N: LF 1
815: FOR I=0 TO F-1
820: IF I=0 LET J=1
825: LPRINT N$(I); "
      : "; MID$(R$(N
      -1), J, L(I)): J=
      J+L(I)
830: NEXT I: LF 1:
      RETURN

```

```

850:REM PRINT
855:IF U=0GOSUB 20
    00:RETURN
860:N=0:INPUT "REC
    ORD NUMBER: ";
    N
865:IF N<1OR N>U
    THEN 860
870:GOSUB 800:LF 5
    :RETURN
900:REM FIND
905:IF U=0GOSUB 20
    00:RETURN
910:CLS :Q=0:INPUT
    "HOW MANY FIEL
    D SEARCH? ";Q
915:IF Q<1OR Q>5OR
    Q>FTHEN 910
920:RESTORE 990:
    FOR I=1TO Q:
    READ W$:Z$=
    STR$ I
925:CLS :Q(I)=0:
    CURSOR :WAIT 0
    :PRINT Z$:W$:
    CURSOR 4:INPUT
    "FIELD NUMBER:
    ";Q(I)
930:IF Q(I)<1OR Q(I)
    >FTHEN 925
935:CLS :Q(I)="" :
    CURSOR 0:PRINT
    "SEARCH";Q(I);
    " FOR: ";
    CURSOR 14:
    INPUT "":Q$(I)
940:IF LEN Q$(I)>L
    (Q(I)-1)GOSUB
    3000:GOTO 935
945:IF LEN Q$(I)<1
    THEN 935
950:NEXT I
955:FOR P=0TO U-1:
    FOR I=1TO Q
960:CLS :PRINT "RE
    CORD";P+1:BEEP
    1
965:S=Q(I):GOSUB 4
    000
970:T=LEN Q$(I)
975:IF MID$ (R$(P)
    ,Z,T)<>Q$(I)
    NEXT P:RETURN
980:NEXT I
985:N=P+1:LF 2:
    GOSUB 800:NEXT
    P:RETURN
990:DATA "st","nd"
    ,"rd","th","th
    "
1000:REM CHANGE
1005:IF U=0GOSUB
    2000:RETURN
1010:N=0:INPUT "R
    ECORD NUMBER
    : ";N
1015:IF N<1OR N>U
    THEN 1010
1020:P=0:INPUT "F
    IELD NUMBER:
    ";P
1025:IF P<1OR P>F
    THEN 1020
1030:I$(0)="" :
    INPUT "CHANG
    E TO: ";I$(0
    )
1035:IF LEN I$(0)
    >L(P-1)THEN
    1050
1040:IF LEN I$(0)
    =L(P-1)THEN
    1055
1045:I$(0)=I$(0)+
    " ":GOTO 104
    0
1050:GOSUB 3000:
    GOTO 1030
1055:IF P=FTHEN 1
    080
1060:S=P+1:GOSUB
    4000:T=X-Z+1
1065:U$(0)=MID$ (
    R$(N-1),Z,T)
1070:I$(0)=I$(0)+
    U$(0)
1075:IF P=1LET R$
    (N-1)=I$(0):
    RETURN
1080:S=P:GOSUB 40
    00:R$(N-1)=
    MID$ (R$(N-1

```

), 1, Z-1)+I\$(S OF FIELD I
0):RETURN	N R\$
2000:PRINT "MEMOR	4001:REM ENTER/F
Y EMPTY!";	IELD # IN R\$
BEEP 15;	4002:REM EXIT/PO
RETURN	S IN Z
3000:CLS :PRINT "	4005:Z=1:IF S=1
FIELD LENGTH	RETURN
EXCEEDED!";	4010:FOR K=1TO S-
BEEP 15;	1:Z=Z+L(K-1)
RETURN	:NEXT K:
3336:+6701	RETURN
4000:REM START PO	STATUS 1 3256

2.7 PAYROLL TAXES

Most people running small businesses are plagued with the weekly task of having to calculate payroll deductions for federal withholding of income and social security taxes. The calculations must be done for each employee. It takes a lot of time, especially if you have to rely on FICA and FWT withholding lookup tables. There are many possibilities for error not only in the lookup process but in the manipulations involving calculations. What is more, knowing where all that money is going can sometimes be downright depressing!

The program presented here can greatly ease the task of determining weekly payroll deductions. A few years ago the thought of performing such a task on a machine the size of a PC would have been simply incredible.

The program uses a piece-wise linear model of the sliding federal tax scale to permit FWT amounts to be calculated. This model accounts for a person's marital status and number of exemptions. The model is based on 1981 FWT figures. Line 80 reduces the FWT values by 15 percent to account for the five percent reduction of 1981 and the ten percent reduction of 1982. By the time this book is printed it may be possible to increase this discount by another ten percent. Of course, tax rates are subject to change at any time, and Uncle Sam says you are responsible for seeing that the proper rate is withheld. So it is up to you to keep this program up to date. Remember, though, that you have some leeway. If you are within a few percent, you can

make up the difference (better yet, perhaps rate a refund) at the end of the year. The FICA value used in the program (see line 70) is based on the 1982 rate. You most likely will have to alter this value by the time you read this book.

The program is a snap to use. Load it into memory, place the PC in the RUN mode, and start the program with a *RUN* command. Answer the prompts for the weekly gross pay, the number of claimed exemptions and marital status. The program will respond with its calculated FWT and FICA withholding amounts. Next, the program will ask the amount of any other withholdings. You can use this opportunity to insert deductions for insurance, local taxes, or other items. (Pressing RETURN alone is the same as entering 0. Do this if there are no other payroll deductions.) The program will then display the net pay figure. Pressing ENTER when the net pay figure is on the screen causes the program to loop back to do another set of calculations.

Here is a sample run of the program for checking purposes:

```

ENTER GROSS PAY: 35*4.8+8*1.5*4.8
# OF EXEMPTIONS? 3
SINGLE(0)/MARRIED(1)? 1
FWT =          16.59
FICA =          15.12
OTHER WITHHOLDINGS: 10.52+3.23
NET PAY =       180.14
  
```

You may use mathematical expressions to enter amounts for gross pay and other withholdings. In the example an employee's pay for 35 hours of regular time (at \$4.80 per hour) and 8 hours at time-and-a-half was entered as a mathematical expression. Similarly, several miscellaneous deductions were summed by using a mathematical expression when inputting *OTHER WITHHOLDINGS*.

If you have to routinely specify other withholdings, calculate overtime pay, or make other regular entries, you might want to consider enhancing the program so that it handles such inputs separately.

As with any tax-related program, you should periodically review your withholding procedures with your accountant to

ensure that you are complying with current government regulations and withholding rates.

This powerful little payroll tax accounting package can really knock some time off of that onerous task.

Program listing: PAYROLL TAXES

```

10:WAIT :USING "#
    ###.##":H=100
20:G=0:INPUT "ENTER GROSS PAY:
    ";G
30:E=0:INPUT "# OF EXEMPTIONS?
    ";E
40:M=0:INPUT "SINGLE(0)/MARRIED
    (1)? ";M:M=INT
    (ABS (M))
50:IF (M<0)+(M>1)
    GOTO 40
60:X=0:T=G-19.23*
    E:GOSUB 600+20
    0*M
70:F=.067*G
80:BEEP 2:X=.85*X
    :X=INT (H*X+.5
    )/H:PRINT "FWT
    = ";X
90:F=INT (H*F+.5)
    /H:PRINT "FICA
    = ";F
100:Z=0:INPUT "OTHER WITHHOLDINGS
    : ";Z
110:Z=INT (H*Z+.5)
    /H
120:G=G-X-F-Z:
    PRINT "NET PAY
    = ";G
130:GOTO 20
600:IF T>=433LET X
    =103.39+.39*(T
    -433):RETURN
610:IF T>=331LET X
    =68.71+.34*(T-
    331):RETURN
620:IF T>=273LET X
    =51.31+.3*(T-2
    73):RETURN
630:IF T>=196LET X
    =31.29+.26*(T-
    196):RETURN
640:IF T>=131LET X
    =17.64+.21*(T-
    131):RETURN
650:IF T>=63LET X=
    5.4+.18*(T-63)
    :RETURN
660:IF T>=27LET X=
    .15*(T-27)
670:RETURN
800:IF T>=556LET X
    =119.35+.37*(T
    -556):RETURN
810:IF T>=454LET X
    =86.71+.32*(T-
    454):RETURN
820:IF T>=369LET X
    =62.91+.28*(T-
    369):RETURN
830:IF T>=288LET X
    =43.47+.24*(T-
    288):RETURN
840:IF T>=210LET X
    =27.09+.21*(T-
    210):RETURN
850:IF T>=127LET X
    =12.15+.18*(T-
    127):RETURN
860:IF T>=46LET X=
    .15*(T-46)
870:RETURN
STATUS 1      855

```

2.8 BANNERS

You can use your PC coupled with its printer/plotter unit to make message banners. You can paste-up banners to form bulletins, use them in advertising artwork, or send large messages to friends who can't take a subtle hint!

This program enables you to choose the size of lettering from about ½-inch characters to giant 1¾-inch monoliths. You may also select whatever pen color you desire. You may input messages having up to 54 characters. If, however, you want to generate even larger banners, the program operates in a manner that permits continuation from one input string to the next. Remember, though, at close to two inches per character a 54-letter message will be about eight feet long—it is easy to get carried away when using this little program!

Once the program is in memory and the PC is connected to the printer unit, all you need to do is place the computer in the RUN mode and issue a *RUN* command. The program prompts for the message you want on the banner, the size of the characters, and the pen color. Once this information has been inputted, the program takes over and generates the banner.

After a banner has been drawn you are given the option to repeat the same message or input a new message. If you elect to repeat the same message, you will still be given an opportunity to change the character size and pen color.

For familiarization, here is a quick run-through of the program's operation:

```
***  BANNERS  ***  
MESSAGE (MAX 54 CHARS.)? HELLO!  
BANNER SIZE (1-25)? 10  
PEN COLOR (0-3)? 2
```

. . . . (printer draws banner here)

```
NEW MSG OR RPT (N/R)? R  
BANNER SIZE (1-25)?
```

By the way, if you respond to the query concerning inputting a new message with any letter other than N or R, the program will end.

Program listing: BANNERS

```

10: PAUSE "      **
    * BANNERS **
    *"
20: CLEAR : DIM A$(
    0)*54: A$(0)=""
    : INPUT "MESSAG
    E (MAX 54 CHAR
    S.)? "; A$(0)
30: INPUT "BANNER
    SIZE (1-25)? "
    : S=S+INT (ABS
    (S)): IF (S<1)+
    (S>25)BEEP 1:
    GOTO 30
40: GRAPH : POKE &7
    9F4, S+10
50: INPUT "PEN COL
    OR (0-3)? "; P:
    P=INT (ABS (P)
    ): IF (P<0)+(P>
    3)BEEP 1: GOTO
    50
60: COLOR P: ROTATE
    1: CLS : GOSUB 1
    00
70: B$="": INPUT "N
    EW MSG OR RPT
    (N/R)? "; B$: B$
    =LEFT$ (B$, 1):
    IF B$="N" THEN
    20
80: IF B$="R" THEN
    30
90: PAUSE "BYE-BYE
    ": END
100: FOR I=1 TO LEN
    A$(0)
110: SORGN : LPRINT
    MID$ (A$(0), I,
    1)
120: NEXT I
130: RETURN
STATUS 1      409

```

SAMPLE OUTPUTS

HELLO!

HELLO!

Mathematics Programs

3.1 ANY BASE CONVERSION

Engineers, scientists, computer programmers, and mathematicians frequently need to convert values from one numbering system to another. This program is particularly valuable to those who may need to dabble from time to time in esoteric bases. For instance, do you have the need to translate from a base of 19 to a base of 78? This program can do it. Yep, it will handle any base from 2 to 99! Even if you don't need such capability for professional chores, it can be a lot of fun to put numbers into strange bases (just to confuse some of your acquaintances). You might even gain some insight into the higher realms of mathematics.

There is one important point to keep in mind concerning this program. All numbers above base 10 must be entered as a two-digit symbol or series of such symbols. For example, computer programmers conventionally refer to the digits above 9, when using a base of 16, as A, B, C, D, E and F. These symbols actually represent the two-digit numbers 10, 11, 12, 13, 14, and 15, respectively. This program is used with two-digit representations. Thus the number FF (base 16) would be represented as 1515 (where 15=F). If this system wasn't adopted, this clever program would run out of alphabetical symbols long before it got to a base of 99.

As is the case with all programs prepared by Emerich Auersbacher, this program is easy to operate. Just load it into your PC, place the unit into the RUN mode and execute a *RUN* directive. Respond to the prompts that appear on the display:

BASE CONVERSION PROGRAM

KNOWN NUMBER: 255
 BASE OF NUMBER: 10
 BASE DESIRED: 16
 A= 1515 BASE 16

KNOWN NUMBER: 1515
 BASE OF NUMBER: 16
 BASE DESIRED: 10
 A= 255 BASE 10

You will notice that the time required to perform conversions varies considerably, depending on the bases involved. For instance, in the above example converting from base 10 to base 16 takes about 10 seconds, but the reverse conversion requires only a few seconds. The size of the number being translated also dictates the conversion delay.

As indicated earlier, this program is most suitable for applications involving rarely-used bases. Another program in this book uses the conventional symbols A through F to represent digits for bases 11 through 16. This conversion program may be more suitable to computer programmers who frequently need to convert between bases such as 2 (binary), 8 (octal), 10 (decimal), and 16 (hexadecimal).

Program listing: ANY BASE CONVERSION

2: PAUSE " BASE C	GOSUB 180: Y=N
ONVERSION PROG	12: BEEP 3: PAUSE "
RAM": USING	ANSWER>>-->"
4: CLEAR : WAIT :	14: PRINT "A="; Y; "
INPUT "KNOWN N	BASE "; B
UMBER: "; X, "BA	16: GOTO 4
SE OF NUMBER:	20: N=0: F=10: IF B>
"; B, "BASE DESI	10 LET F=100
RED: "; P	25: IF X<1 GOTO 90
6: IF P=10 LET W=X	30: T=X/(10^N): IF
: GOSUB 20: B=P:	T<.1 GOTO 50
GOTO 12	40: N=N+1: GOTO 30
8: IF B=10 LET S=X	50: IF N=0 GOTO 90
, U=S, B=P: GOSUB	55: S=INT (X/(F^(N
180: Y=N: GOTO 1	-2)))
2	58: IF N<=2 GOTO 90
10: W=X: GOSUB 20: S	60: FOR Z=1 TO N-2
=Y: U=S: B=P:	65: D=INT (X/(F^(N

```

-2-Z))) - F*INT
(X/(F^(N-1-Z))
)
70: S=(S*B)+D
80: NEXT Z
90: Y=0: IF X-INT X
    =0 GOTO 150
95: X=X-INT X
100: N=1
105: U=X*F^N: IF U-
    INT U=0 GOTO 12
    5
110: N=N+1: GOTO 105
125: FOR Z=1 TO N
126: IF Z=1 LET D=
    INT (U/(F^(N-1
    ))) : GOTO 135
130: D=INT (U/(F^(N
    -Z))) - F*INT (U
    /(F^(N-Z+1)))
135: Y=Y+D*B^(-Z)
140: NEXT Z
150: Y=Y+S: RETURN
180: IF 10>=B LET R=
    10
185: IF B>10 LET R=1
    00
190: N=0: T=0
195: IF S<=0 GOTO 22
    0
200: W=LN S/LN B
205: W=INT (W)
210: N=N+R^W: IF T=N
    GOTO 220
215: S=S-B^W: T=N:
    GOTO 195
220: RETURN
STATUS 1 799

```

3.2 POLYNOMIALS

Are you a mathematician, engineer, scientist or professional who does a lot of work involving polynomial equations? If so, you will love this program, because it is capable of doing a lot of the “grind-it-out” type of calculations. It is capable of multiplying, dividing, and raising polynomials to integer powers.

Just load the program into memory and place the PC into its normal RUN mode. You access each type of operation using the so-called DEFined keys; that is, you press the key marked DEF and then a designated alphabet key. Here are your options:

DEF/M puts the PC in position to multiply polynomials.

DEF/D sets it up to divide polynomials.

DEF/N is used to raise polynomials to integer powers.

Once you have selected the desired operation, enter the appropriate data for each polynomial degree as prompted by the program (coefficients are entered in orders of decreasing powers). If you lose your place while working, press the ENTER key without any data. The program will then prompt you with the degree of the term you should be entering.

Suppose, for example, that you wish to divide the polynomial $6X^6+7X^5+11X^4+30X^3+10X^2+3X-7$ by the polynomial $3X^2+5X+2$. Start off by pressing DEF/D (the DEFine key and then

the letter D). Next, enter data in response to the prompts as illustrated here:

```
NUM DEG? 6
? 6
? 7
? 11
? 30
? 10
? 3
? -7
DEN DEG? 2
? 3
? 5
? 2
```

The PC will then display the results for this problem as:

```
QUO:
DEG 4, 2
DEG 3, -1
DEG 2, 4
DEG 1, 4
DEG 0, -6
REM:
DEG 1, 25
DEG 0, 5
```

Thus the quotient is $2X^4 - 1X^3 + 4X^2 + 4X - 6$ with a remainder of $25X + 5$.

As a second illustration, consider raising a polynomial to a power, such as finding $(2X^2 - 3X + 5)^5$. Start this process by selecting DEF/N, and then inputting values according to the prompts:

```
DEGREE? 2
? 2
? -3
? 5
POWER? 5
```

Handwritten notes:
 deg 2
 deg 5
 2 5 1
 2 4 6
 1 1 1
 1 1 1

A problem such as this can take half-a-minute or so for the PC to obtain the results. You can check to see that the coefficients provided for the above problem yield the expanded

form of the polynomial as: $32X^{10}-240X^9+1120X^8-3480X^7+8210X^6-14643X^5+20525X^4-21750X^3+17500X^2-9375X+3125$.

Program listing: POLYNOMIALS

```

10: "N" CLEAR : DIM                GOSUB 30: INPUT 150
    A(124): INPUT "                "DEG OF 2ND? "
    DEGREE? "; D: A=                ; D
    5: GOSUB 30: 150                41: A=E+7: GOSUB 30 150
    INPUT "POWER? 163                : FOR A=ATO E+7
    "; C                            STEP -1: FOR B=
11: FOR A=5TO B: A(                A+1TO B+E: A(B)
    A+D+1)=A(A):                    =A(A)*A(B-A+5)
    NEXT A: FOR A=2                116 42: NEXT B: A(A)=0:
    TO C                            NEXT A: A=E+8: B
12: FOR B=A*D+A+4                =B+D: BEEP 1:
    TO D+6STEP -1:                GOTO 20 170
    FOR C=B+1TO C+                50: "D" CLEAR : DIM
    D: A(C)=A(C)+A(                A(124): INPUT "
    B)*A(C-B+4):                NUM DEG? "; D: A
    NEXT C: A(B)=0:                =6: E=D+7: GOSUB
    NEXT B 150
13: FOR B=BTO A*D+                30: INPUT "DEN
    D+7: A(B)=A(B+1                DEG? "; D
    ): NEXT B: NEXT                51: A=E: GOSUB 30: 150
    A: A=D+6: B=B-1:                PRINT "QUO: ";
    BEEP 1                        IF D+7>EPRINT
14: 20: FOR D=ATO B: C=                0: A=6: GOTO 54 180
    B-D: PRINT "DEG                52: FOR A=6TO E-D-
    "; USING "###";                1: C=A(A)/A(E):
    C; ", "; USING ;                IF DFOR B=A+1
    A(D): NEXT D:                TO A+D: A(B)=A(
    END                            B)-C*A(B+E-A):
15: 30: B=A+D: FOR A=A                NEXT B
    TO B                            53: B=E-D-A-1:
16: 31: INPUT A(A):                PRINT "DEG";
    NEXT A: RETURN                USING "###"; B;
17: 32: C=B-A: PAUSE "D                ", "; USING ; C:
    EGREE "; C: GOTO                NEXT A: A=A+SGN 17
    31 156                            D
18: 40: "M" CLEAR : DIM                54: IF DPRINT "REM
    A(124): INPUT "                : "; B=E-1: GOTO
    DEG OF 1ST? ";                20 180
    E: A=6: D=E:                STATUS 1 140 770

```

Norlin Rober created this polynomial-handling program. He adds this note of caution: please keep degrees of polynomials to a reasonable level, say less than 100. Degrees beyond this range can cause the program to malfunction. That seems fair

enough to me. Who wants to deal with polynomials approaching infinity, anyhow?

3.3 ROOTS OF POLYNOMIALS

Norlin Rober came up with this program that determines the complex roots of polynomials having degrees of two through five and real coefficients.

To use the program, load it into memory, place the PC into its RUN mode and execute a *RUN* directive. Enter coefficients as prompted, using the following formats:

- Degree 2** $AX^2+BX+C=0$
- Degree 3** $AX^3+BX^2+CX+D=0$
- Degree 4** $AX^4+BX^3+CX^2+DX+E=0$
- Degree 5** $AX^5+BX^4+CX^3+DX^2+EX+F=0$

Note: coefficient A in the above expressions must not be equal to zero.

Once the coefficients have been given, the program will calculate and display all complex roots. If there is a conjugate pair of complex roots, then the real part will be indicated as *RE(X)* and the imaginary parts as *IM(X)*.

There may be some loss of accuracy, particularly in the case of complex roots. This is unavoidable with floating-point arithmetic as used in the pocket computer.

Here is a sample run using the program to solve the equation, $8X^5-75X^4+275X^3-485X^2+387X-90=0$:

```

DEGREE? 5
A=8
B=-75
C=275
D=-485
E=387
F=-90
    
```

Once the inputs have been provided, the PC may take a few moments to perform the calculation before it begins to crank out the results.

X= 0.375
RE(X)= 2
IM(X)= 1
X= 3
X= 2

* Statement IF K
GOSUB 39 probably
means if $K > 0$...

You may have to give the PC a little time to extract the first root of a 5th degree equation; perhaps as much as 15 or 20 seconds will be necessary.

Program listing: ROOTS OF POLYNOMIALS

```

10: INPUT "DEGREE?"; N: GOTO 10*N
20: INPUT "A="; A, "B="; B, "C="; C
21: IF 4*A*C-B*B
    LET X=-B/(2*A)
    :Y=J(4*A*C-B*B)/(2*A): PRINT
    "RE(X)="; X: PRINT "IM(X)="
    ;Y: END
22: X=(-B+J(B*B-4*A*C))/(2*A): Y=
    (-B-J(B*B-4*A*C))/(2*A):
    PRINT "X="; X: PRINT "X="; Y:
    END
30: INPUT "A="; A, "B="; B, "C="; C, "D="; D
31: G=B*B-3*A*C: H=
    4.5*A*(B*C-3*A*D)-B*B*B: K=H*
    H-G*G*G
32: IF K LET M=SGN
    H*(G/(JK+ABS H)^(1/3)+(JK+
    ABS H)^(1/3)): Y
    =(-M-2*B)/(6*A)
33: Z=J(3*M*M-12*G)/ABS(6*A): IF
    K=0 LET M=J(4*G)*SGN H: Y=(-JG
    *SGN H-B)/(3*A

```

```

): Z=Y
34: IF -KGOSUB 39:
    Z=(-J(12*G-3*M*M)-M-2*B)/(6*
    A)
35: X=(M-B)/(3*A):
    PRINT "X="; X:
    IF K PRINT "RE(
    X)="; Y: PRINT "
    IM(X)="; Z: END
36: PRINT "X="; Y:
    PRINT "X="; Z:
    END
39: M=J(4*G)*COS(
    ACS(H/J(G*G*G)))/3: Y=(J(12*
    G-3*M*M)-M-2*B)/(6*A): RETURN
40: INPUT "A="; A, "B="; B, "C="; C, "D="; D, "E="; E:
    IF -A LET A=-A:
    B=-B: C=-C: D=-D
    : E=-E
41: GOSUB 49: K=H*H
    -G*G*G: IF K=0
    LET M=JG*(1+(H
    )^3)
42: IF K LET M=SGN
    H*(G/(JK+ABS H)^(1/3)+(JK+
    ABS H)^(1/3))
43: IF -K LET M=J(4
    *G)*COS(ACS(
    H/J(G*G*G))/3)
44: Q=9*B*B+12*A*(

```

```

M-2*C): IF -Q
LET Q=0
245: R=0: IF (C+M)*(
C+M)/9-4*A*E
LET R=J((C+M)*
(C+M)/9-4*A*E)
: IF 6*A*D-B*C-
B*MLET R=-R
246: FOR S=-1 TO 1
STEP 2: T=6*B*(
3*B-S*JQ)+12*A
*(6*R*S-M-4*C)
247: IF -TLET X=(S*
JQ-3*B)/(12*A)
: Y=J-T/(12*A):
PRINT "RE(X)="
: X: PRINT "IM(X)
)="; Y: NEXT S:
END
248: X=(S*JQ-3*B+JT
)/(12*A): Y=(S*
JQ-3*B-JT)/(12
*A): PRINT "X="
: X: PRINT "X=";
Y: NEXT S: END
249: G=C*C+12*A*E-3
*B*D: H=C*C*C+4
.5*(3*A*D*D-8*
A*C*E+3*B*B*E-
B*C*D): RETURN
250: GOSUB 59: IF -A
LET A=-A: B=-B:
C=-C: D=-D: E=-E
: F=-F
251: X=-SGN F
252: P=(((((A*X+B)*
X+C)*X+D)*X+E)
*X+F): IF F*P
LET X=2*X: GOTO
252
253: IF ABS PLET W=
SGN X*INT (ABS
(.5*X)): Y=X: X=
.5*(W+Y)
254: IF ABS PGOSUB
258: IF X<>YGOTO
254
255: PRINT "X="; X: E
=(((((A*X+B)*X+
C)*X+D)*X+E): D
=((((A*X+B)*X+C
)*X+D): C=((A*X
+B)*X+C)
256: B=A*X+B: GOTO 4
1
258: P=(((((A*X+B)*
X+C)*X+D)*X+E)
*X+F): @ (24-SGN
(F*P))=X: X=.5*
(W+Y): RETURN
259: INPUT "A="; A, "
B="; B, "C="; C, "
D="; D, "E="; E, "
F="; F: RETURN
STATUS 1 1586

```

3.4 FRACTIONS & DECIMALS

This program, created by Norlin Rober, will perform two types of mathematical operations. You may select the option desired using the DEFine key and the F (for fractions) or D (for decimal) key. This is done, naturally, after having loaded the program into the PC and placing it into its standard program RUN mode.

DEF/F is used to obtain a fraction that approximates a given decimal number. The number given must be positive with a nonzero fractional part. The program provides a sequence of approximations that increase in accuracy, with each such approximation given in its reduced or lowest terms form. For

example, selecting DEF/F and inputting Pi (3.141592654) as a decimal value, would yield the following action on the part of the pocket computer:

```

DECIMAL? 3.141592654
3.142857143
=          22/ 7
3.141509434
=          333/ 106
3.14159292
=          355/ 113
3.141592654
=          104348/ 33215

```

The last result shown agrees to 10 significant figures.

DEF/D is used to obtain a decimal value for a given fraction. Both the numerator and denominator of the given fraction must be positive. The program will reduce the fraction to its lowest terms, display those terms, and then proceed to determine the infinite decimal expansion.

For illustrative purposes, consider the inputting of the fraction 735/456:

```

NUM? 735
DEN? 456
INTEGER PART,  1
FRAC PART (REDUCED),
                93/ 152
NR. OF DECIMAL PLACES,
    3 (NON-REPEATING);
    18 (REPEATING BLOCK).
1ST 4 PLACES,  6118      (The repeating block begins with digit 8)
NEXT 4,      4210
NEXT 4,      5263
NEXT 4,      1578
NEXT 4,      9473
NEXT 4,      6842      (The block of 18 begins again with digit 8)
NEXT 4,      1052
NEXT 4,      6315
.
.
.

```

You get the picture.

Program listing: FRACTIONS & DECIMALS

```

10: "F"N=0:D=1:A=1
   :B=1:C=0: INPUT
   "DECIMAL? ";X:
   F=X
11: E=INT (A/F):G=
   F:F=A-E*G:A=G:
   G=N:N=B+E*G:B=
   G:G=D:D=C+E*G:
   C=G: IF D<2GOTO
   11
12: G=N/D:PRINT G:
   PRINT " =";
   USING "#####
   #####";N;" / ";
   USING ;D: IF G<
   >XGOTO 11
13: PRINT " AGREES
   TO 10 DIGITS"
   :GOTO 12
20: "D"INPUT "NUM?
   ";N, "DEN? ";D
   :C=INT (N/D):N
   =N-C*D:A=N:B=D
21: E=B:B=A-B*INT
   (A/B):A=E: IF B
   GOTO 21
22: N=N/E:D=D/E:A=
   D:F=0
23: PRINT "INTEGER
   PART, ";C:
   PAUSE "FRAC PA
   RT (REDUCED), "
   :PRINT USING "
   #####";N
   ;"/";USING ;D
24: IF .5*A=INT (.
   5*A)LET A=.5*A
   :B=B+1:GOTO 24
25: IF .2*A=INT (.
   2*A)LET A=.2*A
   :F=F+1:GOTO 25
26: C=1:E=0: IF F>B
   LET B=F
27: C=10*C-A*INT (
   10*C/A):E=E+1:
   IF C>1GOTO 27
28: C=10000*N:BEEP
   1:PAUSE "NR. 0
   F DECIMAL PLAC
   ES, "
29: PRINT USING "#
   ####";B;" (NON
   -REPEATING);":
   PRINT E;" (REP
   EATING BLOCK).
   ":USING
30: A=INT (C/D):
   PRINT "1ST 4 P
   LACES, ";A
31: A=.01*A:C=1000
   0*C-1000000*D*
   INT A-1000000*
   D*(A-INT A):A=
   INT (C/D):
   PRINT "NEXT 4,
   ";
32: PRINT A:GOTO 3
   1
STATUS 1 767

```

3.5 PRIME FACTORS

Norlin Rober whipped up this short program that will yield the prime factors of any positive integer that is less than 1E10.

To put it to work, simply load the program and execute a *RUN* command with the PC in the RUN mode. Respond to the prompt for the integer number that is to be evaluated. The computer will then output the prime factors and their multiplicity.

Here is a sample of its operation so that you can verify that you loaded the program correctly:

```

INTEGER: 13510750
      2 IS A FACTOR
OF MULTIPLICITY 1
      5 IS A FACTOR
OF MULTIPLICITY 3
      11 IS A FACTOR
OF MULTIPLICITY 1
      17 IS A FACTOR
OF MULTIPLICITY 3
END OF LIST

```

The program is designed to hold down execution time. However, as the test number becomes very large, execution time increases considerably. A number such as 999999999967 (which is a prime) can tie up the PC for as long as 40 minutes. Numbers on the order of that used in the above example, however, are typically handled in a few seconds. Well, what do you want out of something that fits in your pocket?

Program listing: PRIME FACTORS

```

10: INPUT "INTEGER
: "; I: J=I: M=0:
T=2: GOSUB 50: T
=3: GOSUB 50: T=
5: GOSUB 50: T=1
20: T=T+6: GOSUB 50
: T=T+4: GOSUB 5
0: T=T+2: GOSUB
50: T=T+4: GOSUB
50: T=T+2: GOSUB
50
30: T=T+4: GOSUB 50
: T=T+6: GOSUB 5
0: T=T+2: GOSUB
50: IF T*T<=J
GOTO 20
40: GOTO 100
50: IF J/T=INT (J/
T) RETURN
60: IF T=1 GOTO 130
70: M=M+1: J=J/T: IF
J/T=INT (J/T)
GOTO 70
80: BEEP 1: PRINT
USING "#####
####"; T; " IS A
FACTOR": PRINT
USING "###"; "
OF MULTIPLICIT
Y"; M
90: M=0: IF T*T<=J
RETURN
100: IF I=J GOTO 130
110: IF J=1 PRINT "E
ND OF LIST":
END
120: T=J: J=1: M=1:
GOTO 80
130: BEEP 1: PRINT
USING "#####
####"; I; " IS A
PRIME": END
STATUS 1 437

```

3.6 SIMULTANEOUS EQUATIONS

This PC program is able to determine the solution of a system of up to 12 linear equations. It was developed by Norlin Rober, and it utilizes the process of Gaussian elimination.

As you may notice, the program itself is quite short. To use it, load it into memory, then place the PC in the RUN mode. Execute a *RUN* directive. Respond to the prompts to set up a problem for solution.

Suppose, for instance, that you wanted to solve the following system of equations:

$$3W + 2X + 6Y + 2Z = 2$$

$$W + 4X + 3Y + 2Z = 6$$

$$2W + X + 9Y + 3Z = 2$$

$$4W + 3X + 6Y + Z = 8$$

Using manual methods it is easy to verify that the solution to this system is: $W = -2$, $X = 3$, $Y = 2$, and $Z = -5$. Thus this system may be used to verify that the program has been loaded properly.

To use the PC to solve this problem, start the program and respond to the first prompt:

NUMBER OF EQUATIONS? 4

As soon as the computer is ready for the first data input, it will beep and then display a question mark. Enter the coefficients for the first equation (3, 2, 6, 2, 2) using ENTER after each value. The computer will beep again when it is ready to accept the next equation. Enter the data for the next three equations in the same manner:

Equation #2 1, 4, 3, 2, 6

Equation #3 2, 1, 9, 3, 2

Equation #4 4, 3, 6, 1, 8

After the data for the last equation has been entered, the PC will mull things over for a few moments, then issue a beep to indicate that the results are ready. Results for the sample problem will be outputted as:

1	-2
2	3
3	2
4	-5

in the format: parameter (column) number and its solution for the system.

By the way, should you lose your place while inputting data, you can start the current row over by pressing DEF/C (the DEFine key and then the letter C).

Also, you should know that occasionally the program may find that the coefficients entered are not acceptable for the computer algorithm being used. In this case the message:

REJECTED

will be displayed. You should continue entering all of the other rows in the equation matrix, and then try once more to enter the offending equation. If you obtain the *REJECTED* message again, then there is no unique solution for the data given (the system is inconsistent or dependent).

Program listing: SIMULTANEOUS EQUATIONS

1: "A" DIM A(172):	REJECTED": GOTO
INPUT "NUMBER	2
OF EQUATIONS?	6: B=B+1: IF A-B-1
"; A: A=A+1: B=0	GOTO 2
2: "C" BEEP 1: FOR	7: FOR B=1-ATO -1
C=7TO A+6:	: F=6-A*B: E=A(F
INPUT A(C+A*B)) / A(F+B): A(F)=
: NEXT C: IF B=0	E: IF B+2GOTO 9
THEN 5	8: FOR C=1TO -B-1
3: FOR D=1TO B: E=	: F=A*C+6: A(F)=
A(A+A*B-D+6): F	A(F)-A(F+B)*E:
=A(A*D-D+6)	NEXT C: NEXT B
4: FOR C=7+A*BTO	9: BEEP 1: FOR B=1
A+C-1: A(C)=A(C	TO A-1: E=A(A*A
)-A(C+A*D-A-A*	-A*B+6): PRINT
B)*E/F: NEXT C:	B; " "; E: NEXT
NEXT D	B: END
5: IF A(C-B-1)=0	STATUS 1
BEEP 1: PRINT "	406

Naturally the execution time required to solve a system is dependent on the number of equations involved. A system of 12 proper equations may take several minutes to solve. The four-parameter sample demonstrated here takes but a few seconds to complete. Not bad for a little machine!

3.7 FACTORIAL!

The factorial (!) function is not included in the normal repertoire of mathematical functions of your PC. It is possible, however, to program your PC to calculate the function using BASIC. One way this can be accomplished is to use a program line such as:

```
10 F=1:FOR J=N TO 2 STEP -1:F=F*J:NEXT J
```

Norlin Rober has, however, devised a method that is four times faster. How does he do it, with magic? Nope, but almost. He uses a machine language routine!

The code to do this is tucked away in the part of the memory normally used to hold the string variables A\$, B\$, and C\$. If you use this program just the way it is presented in the accompanying listing, it will set up the necessary machine codes each time you use it. However, if you want to use the program as part of a larger package, you must take care to avoid a conflict with the memory area used to hold string variables named A\$, B\$, or C\$. The best advice is simply don't use them after this routine has been loaded.

To use the program after it has been placed in memory, turn your PC to the RUN mode and issue a *RUN* command. Respond to the prompt for a number in the range of 1 to 69. The program will immediately perform the function and return the answer in the display. You can repeat the operation by pressing the ENTER key when the answer is being displayed. Here is an example so that you may verify proper loading of the program:

```
FACTORIAL (1-69)? 9
9! = 362880
```

If you want to use the factorial-finding machine language routine as part of a larger program, take note of the following:

1. Lines 10 to 30 are used to stuff the machine code into the area normally used by the string variables named A\$ through C\$. You could place these lines anywhere in a program as long as they were executed before attempting to use the factorial function. These lines would only have to be executed once to set up the machine code, provided they were not disturbed thereafter by attempting to use string variables A\$, B\$, or C\$.

2. To obtain the factorial value of any integer in the range 1 through 69, place the integer into variable N, then execute the BASIC statement: *CALL &78C0,N*. The value of factorial N will be returned in the variable named F. You may then use that value in further calculations, display it, or use it in whatever manner suits your fancy.

The time required to compute 69! using this method is approximately one-half of a second. Can you beat that?

Program listing: FACTORIAL!

10: POKE &78C0, &FD	, &11, &9A
, &6A, &FD, &A8, &	40: INPUT "FACTORI
CD, &10, &80, &E6	AL (1-69)? ";N
, &FD, &2A, &62, &	:N=ABS (INT (N
6E, 2, &81, 9, &FD): IF (N<1)+(N
20: POKE &78D0, &A8	>69) THEN BEEP
, &CD, &10, &80, &	1: GOTO 40
CD, &7E, &9E, &11	50: CALL &78C0, N:
, &BE, &F9, &32, &	WAIT :PRINT N;
58, &79, &5A, &28	"! =";F: GOTO 4
, &BE	0
30: POKE &78E0, &F7	STATUS 1 268

3.8 REGRESSION ANALYSIS

Mathematicians, statisticians, scientists, and engineers frequently have the need to perform regression analysis on experimentally derived data. Now it is possible to perform this type of powerful data-fitting on a completely portable pocket computer. If you had wanted to go out in the field ten years ago with a computer capable of performing such data massaging, you would have had to pull it on a trailer behind your jeep!

The use of this program is straightforward. After loading the program into memory, place the PC into its RUN mode and start program operation by giving the *RUN* command. You may then enter N pairs of X,Y coordinates. After the data points have been entered, the program will determine the slope and Y-intercept of the regression line (also known as the least-squares or trend line). The program also provides additional statistical information concerning the plotted data.

The example that follows illustrates the operation of the program and details its special capabilities.

Assume a set of data points at the X,Y coordinates: (15,37), (18,40), (17,40), (22,46), and (20,48). Place the program into operation and begin by entering the data points in response to prompts:

NR OF PAIRS? 5

X? 15

Y? 37

X? 18

Y? 40

X? 17

Y? 40

X? 22

Y? 46

X? 20

Y? 48

REGRESSION LINE:

SLOPE= 1.52739726

Y-INT = 14.095889042

Thus, you now know that the equation of the regression line (with coefficients rounded) would be: $Y=1.527X + 14.096$. You can continue using the program to obtain more useful information regarding the regression line. Just press the ENTER key again:

VAR.= 5.559360731

STD DEV= 2.357829665

The values just presented represent the variance and standard deviation of the errors of the sample values. The standard

deviation is also known as the standard error of estimate. The program calculates these as unbiased values.

The program can provide still more. Continue using the ENTER key to bring up more information:

CORR.= 8.962837176E-01

This gives the coefficient of correlation for the sample data.

Keep going to obtain:

SIG LEV FOR LINEARITY:

1.97333026E-02

This may be interpreted to mean that if there is no linear relationship between X and Y, the probability of a sample such as that used in the illustration occurring would be roughly 0.0197. Customarily, a probability factor of less than 0.05 is considered sufficient evidence that a linear relationship does exist.

If the variable T used in the program were displayed at this time, it would show 3.500507443. This is the value of Student's T. With N-2 degrees of freedom, which is 3 in this case, the probability of a value of T being this large is the 0.0197 value shown above.

Press ENTER again for still more information:

95% C.I. FOR SLOPE:

MIN, 0.13878151

MAX, 2.91601301

This reveals the 95-percent confidence interval for the slope of the population regression line.

Continue using ENTER:

95% CI FOR CORR COEFF:

MIN, 6.69917843E-02

MAX, 9.931811736E-01

The population coefficient of correlation is thus most likely to lie between approximately 0.067 and 0.993 for the sample data used.

Finally, predictions of Y for given values of X may be obtained, along with amplifying information. Press ENTER again:

```
GIVEN X?16
Y= 38.53424658
```

Thus with X equal to 16, the regression line value of Y is predicted to be approximately 38.534. To obtain more information about this prediction, continue pressing ENTER to obtain:

```
95% CI FOR MEAN Y:
MIN,  33.80479176
MAX,  43.2637014
95% CI FOR PREDICTED Y:
MIN,  29.66448046
MAX,  47.4040127
```

This information may be interpreted as follows: When X is 16, the mean value of Y should be between 33.80 and 43.26. In an individual case, Y may be expected to fall between 29.66 and 47.40.

Pressing ENTER at this point will cause the program to loop back for another given value of X, if desired.

If you want to enter a different set of data points to obtain a new regression line, then use the BREAK key to stop the program. Start the program over again by executing a *RUN* directive.

This sophisticated data analyzer was provided through the courtesy and skill of Norlin Rober.

Program listing: LINEAR REGRESSION

```
10: INPUT "NR OF P          N=A*B: H=E*N-B*
    AIRS? "; N: A=0:        B: K=N-2: R=G/J(
    B=0: C=0: D=0: E=       F*H): U=(F*H-G*
    0: FOR Z=1 TO N         G)/(F*K*N): S=J
    11: INPUT "X? "; X,      U: M=G/F
        "Y? "; Y: A=A+X:    13: L=(B-A*M)/N:
        B=B+Y: C=C+X*X:     PRINT "REGRESS
        D=D+X*Y: E=E+Y*X   ION LINE: ":
        Y: NEXT Z          PRINT "SLOPE=
    12: F=C*N-A*A: G=D*     "; M: PRINT "Y-I
```

```

NT = ";L:PRINT
"VAR.= ";U
14:PRINT "STD DEV
= ";S:PRINT "C
ORR.= ";R
20:T=ABS M*J(F/(N
*U)):X=K/(K+T*
T):P=K>1:Y=P
21:IF K>2FOR Y=K-
3TO 2STEP -2:P
=1+P*X*Y/(Y+1)
:NEXT Y
22:IF Y=1LET P=.5
*P*T/(J(K+T*T)
):GOTO 24
23:P=P*T*X/JK/PI+
ATN (T/JK)/180
24:P=.5-P:BEEP 1:
PRINT "SIG LEV
FOR LINEARITY
":PRINT P
30:IF K<8GOTO 32+
K
31:T=(((.17/K+.57
8)/K+.7359)/K+
1.58953)/K+2.5
5585)
32:T=((T/K+2.8224
986)/K+2.37227
123)/K+1.95996
3985):GOTO 40
33:T=TAN 85.5:
GOTO 40
34:T=38/J78:GOTO
40
35:T=3.182446305:
GOTO 40
36:T=2.776445105:
GOTO 40
37:T=2.570581836:
GOTO 40
38:T=2.446911851:
GOTO 40

```

```

39:T=2.364624252
40:Y=T*J(N*U/F):X
=M-Y:Y=M+Y:
PRINT "95% C.I
. FOR SLOPE:":
PRINT "MIN, ";
X:PRINT "MAX,
";Y
41:IF N=3GOTO 50
42:Y=LN ((J(F*H)+
G)/(J(F*H)-G))
:Z=3.919927969
/J(N-3):X=(EXP
(Y-Z)-1)/(1+
EXP (Y-Z))
43:Y=(EXP (Y+Z)-1
)/(1+EXP (Y+Z)
):PRINT "95% C
I FOR CORR COE
FF:":PRINT "MI
N, ";X
44:PRINT "MAX, ";
Y
50:INPUT "GIVEN X
?":X:Y=M*X+L:
PRINT "Y=";Y:U
=X-A/N:U=N*U*U
/F+1/N
51:Z=T*J(U*U):I=Y
-Z:J=Y+Z:PRINT
"95% CI FOR ME
AN Y:":PRINT "
MIN, ";I:PRINT
"MAX, ";J
52:Z=T*J((U+1)*U)
:I=Y-Z:J=Y+Z:
PRINT "95% CI
FOR PREDICTED
Y:":PRINT "MIN
, ";I
53:PRINT "MAX, ";
J:GOTO 50
STATUS 1 1231

```

Science and Engineering Programs

4.1 FAST FOURIER TRANSFORM

The fast Fourier transform is a mathematical tool used by engineers and scientists in certain analytical disciplines. Since the subject matter is usually treated at the advanced college level, I will not attempt to explain its use or application here. Suffice it to say, for those who can use such a tool, that as powerful programmable calculators and, more recently, fully functional pocket computers have become available the Fourier transform has become increasingly popular. The Fourier transform becomes more useful as the number of data points used in an analysis increases. However, since the amount of computation increases in proportion to the square of the data, the “fast” implementation greatly speeds the problem-solving process.

The compact fast Fourier transform program shown here can handle up to 64 complex points as data. It uses only 11 memory locations for uses such as loop counters and the storage of intermediate calculations. The data points are stored in up to 128 elements of an array. Execution speed is quite rapid, taking just a few minutes to carry out a transform of 32 complex points. This typically beats a programmable calculator by a factor of two to one.

To use the program, load it and execute *RUN* with the PC in the normal RUN mode. Respond to the prompt for “N” with the

number of data points. Remember, the maximum is 64 and the number of points must be equal to a power of 2 (thus: 2, 4, 8, 16, 32 or 64). Next, respond to the prompt of "D" by inputting the value 1 if you want to perform the "forward" transform (from the time domain to the frequency domain). Respond with -1 if you desire to perform the "reverse" (frequency domain to time domain) transform.

The PC will reflect the data classification selected (frequency or time) and ask for the data points. These are inputted individually by real and imaginary parts. All points, including those that are zero, must be entered.

When the number of data points originally specified have been entered, the program proceeds to work on the problem. First, a bit-reversal routine is executed. Then, the transform itself is performed. (The *RADIAN* statement in line 4 of the program separates the bit-reversal procedure from the transform.) At the conclusion of the process, the display shows the domain to which the data has been transformed. The transformed data values may then be viewed.

This version of the fast Fourier transform was developed by A. D. Chamier. A sample of its operation assuming the inputting of information relating to a square-wave signal over 16 data points (representing one cycle) is illustrated here for checking purposes:

```

N=16
D=1
TIME
N    REAL    IMAG
REAL = 0
IMAG = 0
  1. 0.000 0.000
REAL = 0
IMAG = 0
  2. 0.000 0.000
REAL = 0
IMAG = 0
  3. 0.000 0.000
REAL = 0
IMAG = 0
  4. 0.000 0.000
REAL = .5

```



```

IMAG = 0
  5. 0.5000 0.0000
REAL = 1
IMAG = 0
  6. 1.0000 0.0000
REAL = 1
IMAG = 0
  7. 1.0000 0.0000
REAL = 1
IMAG = 0
  8. 1.0000 0.0000
REAL = 1
IMAG = 0
  9. 1.0000 0.0000
REAL = 1
IMAG = 0
 10. 1.0000 0.0000
REAL = 1
IMAG = 0
 11. 1.0000 0.0000
REAL = 1
IMAG = 0
 12. 1.0000 0.0000
REAL = .5
IMAG = 0
 13. 0.5000 0.0000
REAL = 0
IMAG = 0
 14. 0.0000 0.0000
REAL = 0
IMAG = 0
 15. 0.0000 0.0000
REAL = 0
IMAG = 0
 16. 0.0000 0.0000

```

Pressing ENTER after the 16th set of data points has been displayed results in the transform being performed. With only 16 points, the PC will take 15 to 20 seconds to perform the transform calculations. When it is through, you can observe the results by repeated pressings of the ENTER key:

```

FREQ
N    REAL    IMAG
  1. 0.5000  0.0000
  2. -0.314  0.0000

```

```

3. 0.000 0.000
4. 0.093-0.000
5. 0.000 0.000
6. -0.041 0.000
7. 0.000 0.000
8. 0.012 0.000
9. 0.000 0.000
10. 0.012-0.000
11. 0.000 0.000
12. -0.041 0.000
13. 0.000 0.000
14. 0.093-0.000
15. 0.000 0.000
16. -0.314-0.000

```

Note that the program normalizes results by dividing by N, the number of data points.

Program listing: FAST-FOURIER TRANSFORM

```

1: DIM A(128):
   INPUT "N="; A, "
   D="; D: B=1: C=1:
   GOSUB 8: I=1:
   FOR K=1 TO A-2:
     J=A
2: IF I>K LET E=K:
   F=I: G=A(E): A(E
   )=A(F): A(F)=G:
   H=A(E+A): A(E+A
   )=A(F+A): A(F+A
   )=H
3: J=J/2: IF J<I
   LET I=I-J: GOTO
   3
4: I=I+J: NEXT K:
   CLS : RADIAN :
   FOR K=1 TO A/2:
     E=A-2*K: FOR J=
     1 TO K: B=(J-1)*
     D*PI/K: C=COS B:
     B=SIN B
5: FOR I=J TO J+E
   STEP 2*K: F=I+K
   : G=C*A(F)-B*A(
   F+A): H=C*A(F+A
   )+B*A(F): A(F)=
   A(I)-G
6: A(F+A)=A(I+A)-
   H: A(I)=A(I)+G:
   A(I+A)=A(I+A)+
   H: NEXT I: NEXT
   J: K=2*K-1: NEXT
   K: D=-D: B=1
7: IF -D LET B=A
8: IF D PRINT "TIM
   E"
9: IF -D PRINT "FR
   EQ"
10: PRINT "N REAL
   IMAG": FOR K=
   1 TO A: J=K+A: IF
   C INPUT "REAL="
   : A(K), "IMAG=":
   A(J)
11: G=A(K)/B: H=A(J
   )/B: PRINT
   USING "###."; K
   : USING "##.###
   " : G; H: NEXT K:
   IF C RETURN
STATUS 1          552

```

4.2 LOW-PASS FILTER DESIGN

Electronics engineers and technicians often find it desirable to be able to quickly design various types of electronic signal filters. A pocket computer can make the design of something such as the commonly used low-pass filter almost a trivial exercise. The program shown here, created by Hank Librach, selects the components needed for a second-order low-pass filter using an integrated circuit (IC) operational amplifier. It also calculates the theoretical frequency response of the filter, using the recommended components. This permits you to gauge the expected performance characteristics of the circuit.

The placement of relevant components within the circuit is indicated in the accompanying schematic diagram (Fig. 4.2). Load the program into your PC and put the machine in the RUN mode. Start the program and respond to the prompts for the desired filter passband gain, cutoff frequency in hertz, and the desired damping factor. Select a value for the capacitor identified in the diagram as C2. The program then calculates the design values for the remaining components used in the circuit.

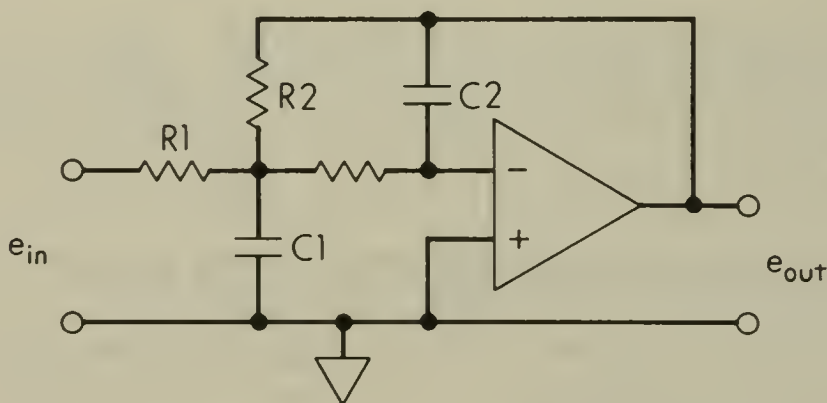


Fig. 4.2 Low-pass filter circuit.

Here is an example of a low-pass filter designed by the program:

```
LOW PASS FILTER
PASSBAND GAIN=? 1
CUTOFF FREQ. HZ.=? 100
DESIRED DAMPING=? .707
CHOOSE VALUE FOR C2=? .01E-6
```

COMPONENT VALUES ARE:

C1 = 4.001E-08
 C2 = 1.000E-08
 R1 = 1.125E 05
 R2 = 1.125E 05
 R3 = 5.626E 04

Once the circuit components have been outputted, you may elect to input frequencies at which you wish to have the response calculated. To do so, just continue operating the program as illustrated:

FREQ. RESPONSE
 INPUT F= 1
 K=-1.000E 00
 K(DB)= 2.171E-07
 INPUT F= 100
 K=-7.072E-01
 K(DB)=-3.008E 00
 INPUT F= 200
 K=-2.425E-01
 K(DB)=-1.230E 01
 .
 .
 .

You might want to draw the frequency response curve on chart paper using the data outputted by this program. Alternatively, perhaps you can enhance this program so that the printer/plotter interface does that for you?

Program listing: LOW-PASS FILTER DESIGN

10: PAUSE "LOW PAS	70: D=Z/(2*PI*F*B):
S FILTER"	C=D/H: E=D/(H+1
20: INPUT "PASSBAN)
D GAIN=? "; H	80: G=1/(C*E*A*B):
30: INPUT "CUTOFF	I=(1/C+1/D+1/E
FREQ. HZ.=? ";)/A: N=1/(D*E*A
F	*B)
40: INPUT "DESIRED	90: PAUSE "COMPONE
DAMPING=?"; Z	NT VALUES ARE:
50: INPUT "CHOOSE	"
VALUE FOR C2=?	100: WAIT :PRINT
"; B	USING "#.###^"
60: A=(1+H)*B/(Z*Z	;"C1="; A
)	105: PRINT "C2="; B

```

110:PRINT "R1=";C
115:PRINT "R2=";D
117:PRINT "R3=";E
120:PAUSE "FREQ. R
      ESPONSE:"
130:INPUT "INPUT F
      =" ;F
135:J=4*PI*PI*F*F:M=
      ABS (N-J)
140:K=-G/(J(M*M+J*
      I*I))
145:L=20*LOG (ABS
      K)
150:PRINT USING "#
      .###^";"K=";K
155:PRINT "K(DB)="
      ;L
160:GOTO 130
STATUS 1          500

```

4.3 INTERPOLATION

This program, by Paul T. Meredith, provides three methods of interpolation, all in one package. The methods available are linear, Lagrangian, and first- or second-order least squares. The latter routine also provides a curve fit through the given data points.

Load the program into memory and place the PC in the RUN mode. Select the desired operation using the DEFine key followed by either the letter A, B, or C key, as explained below.

To perform linear interpolation, select DEF/A. Data is then inputted as two sets of X,Y points. The program then prompts for a value of X. When this is inputted, the program responds with the calculated value of Y. This is a good routine to use for general linear extrapolations.

Using DEF/B brings up the Lagrangian routine. Input the number of data points you intend to enter (maximum of 70) and then the X,Y coordinates for each point. After known points have been entered you may obtain predictions of Y(X) by inputting values of X at the prompt. The time required to obtain an answer is dependent on the number of points originally inputted. This type of interpolation fits a polynomial of order N-1 through the N data points specified. Remember, some higher order polynomials may behave strangely, so it is better to use as few data points as possible. Extrapolation using this type of routine can be risky.

Least squares interpolation is invoked using DEF/C. Data is inputted as X,Y pairs with an unlimited number of points. After the desired points have been specified, you can select first-order

(linear) or second-order (quadratic) curve fitting. The coefficients of the polynomials are calculated by the program using the formula $Y(X)=(A0)+(A1)X+(A2)X^2$. Coefficient (A2) in this equation is equal to zero for a first order fit. Once the coefficients have been shown, entering a value for X yields a calculated Y(X).

As a rule, a least squares fit does not pass exactly through all of the data points. The method is intended to provide a smooth curve such that the sum of the squares of the distances from the curve to each data point is minimized. It is a good routine to use when experimentally-derived data is likely to contain random errors.

The really nice feature about this package is having all three interpolation methods available at one time. Pick the one that best serves your needs, or try several to see which yields the most likely or consistent results.

You can verify that you loaded the program properly using the following test data.

Press DEF/A to bring up:

LINEAR INTERPOLATION

X1= 2

Y1= 2

X2= 4

Y2= 4

X= 3

Y= 3

Using DEF/B brings up:

LAGRANGIAN INTERPOLATION

#X,Y PAIRS= 3

X(I)= 1

Y(I)= 1

X(I)= 3

Y(I)= 3

X(I)= 5

Y(I)= 5

X= 7

Y= 7

And DEF/C will yield:

```

LEAST SQUARES
#X,Y PAIRS= 3
X(I)= 1
Y(I)= 1
X(I)= 5
Y(I)= 5
X(I)= 9
K(I)= 9
1ST OR 2ND ORDER?(1,2) 2
A0= 0
A1= 1
A2= 0
X= 17
Y= 17
.
.
.

```

Program listing: INTERPOLATION

```

10:"A"CLEAR :
  PAUSE "LINEAR
  INTERPOLATION"
  :INPUT "X1=";A
  , "Y1=";B
20: INPUT "X2=";C,
  "Y2=";D:G=(D-B
  )/(C-A)
30: INPUT "X=";X:Y
  =(X-A)*G+B:
  WAIT :PRINT "Y
  =" ;Y:GOTO 30
40:"B":CLEAR :DIM
  A(70):PAUSE "L
  AGRANGIAN INTE
  RPOLATION":
  INPUT "#X,Y PA
  IRS=";E
50:FOR I=1TO E:B=
  (I-1)*2:D=B+1
60: INPUT "X(I)=";
  A(B),"Y(I)=";A
  (D):NEXT I:
  INPUT "X=";A
70:G=0:FOR I=1TO
  E:F=1:B=(I-1)*
  2:D=B+1:FOR J=
  1TO E
80:C=(J-1)*2:IF B
  =CGOTO 100
90:F=F*(A-A(C))/(
  A(B)-A(C))
100:NEXT J:G=G+F*A
  (D):NEXT I
110:BEEP 1:WAIT :
  PRINT "Y=";G:
  INPUT "X=";A:
  GOTO 70
120:"C":CLEAR :
  PAUSE "LEAST S
  QUARES":INPUT
  "#X,Y PAIRS=";
  A
130:FOR I=1TO A:
  INPUT "X(I)=";
  X,"Y(I)=";Y:B=
  B+X:C=C+X*X
140:D=D+X*X*X:E=E+
  X*X*X*X:F=F+Y:
  G=G+X*Y
150:H=H+X*X*Y:NEXT
  I:INPUT "1ST O
  R 2ND ORDER?(1
  ,2)";J
160:IF J=2GOTO 180
170:K=A*C-B*B:L=F*

```

```

      C-G*B:M=A*G-B*
      F:N=0:GOTO 220
180:K=A*(C*E-D*D)+
      B*(D*C-B*E)+C*
      (B*D-C*C)
190:L=F*(C*E-D*D)+
      G*(D*C-B*E)+H*
      (B*D-C*C)
200:M=A*(G*E-D*H)+
      B*(C*H-F*E)+C*
      (F*D-G*C)
210:N=A*(C*H-D*G)+
      B*(F*D-B*H)+C*
      (B*G-C*F)
220:X=L/K:Y=M/K:Z=
      N/K:BEEP 2:
      WAIT:PRINT "A
      0=";X
230:PRINT "A1=";Y:
      PRINT "A2=";Z
240:INPUT "X=";A:B
      =X+Y*A+Z*A*A:
      PRINT "Y=";B:
      GOTO 240
STATUS !          920

```

4.4 TRIANGLES

Given three parts of a triangle, including at least one side, this program proceeds to determine the missing side(s) and angle(s) and then calculates the area of the triangle. Such a program is particularly valuable to surveyors, civil engineers, mathematicians, and students. It can also be of interest to that breed of individuals who consider themselves just plain curious!

Please note that in order to use the program properly your PC should be set to operate in degrees. This is done by placing the unit in the RUN mode and typing the command DEGREE

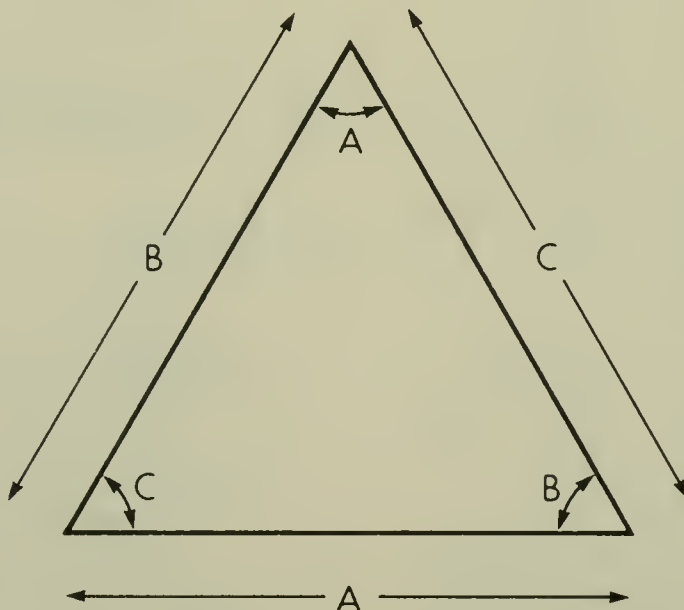


Fig. 4.4 The parts of a triangle.

(followed by ENTER). You should then see the *DEG* annunciator appear at the top part of the display, near the center.

The program is started with the pocket computer in the normal RUN mode by entering the *RUN* directive. Elements of a triangle are designated as angles A, B, and C, with their corresponding sides lying opposite the referenced angles, as shown in Fig. 4.4. To use the program to solve for unknown parts, enter at least one side and two other elements (sides or angles) of a triangle. Unknown parts are designated by pressing the ENTER key without data. Here is an example of the program's operation:

```
SIDE A? 5
SIDE B?      (ENTER key pressed without other input.)
SIDE C? 8
ANGLE A? 35
ANGLE B?      (ENTER key pressed without other input.)
ANGLE C?      (ENTER key pressed without other input.)
```

Now the results are displayed:

FIRST OF TWO SOLUTIONS:

```
SIDE A = 5.
SIDE B = 8.539329288
SIDE C = 8.
ANGLE A = 35.
ANGLE B = 78.40466339
ANGLE C = 66.59533661
AREA = 19.59183225
SECOND SOLUTION:
SIDE A = 5.
SIDE B = 4.567103421
SIDE C = 8.
ANGLE A = 35.
ANGLE B = 31.59533661
ANGLE C = 113.4046634
AREA = 10.47833162
```

Norlin Rober, the designer of this program, provides a bit more information for those who are really curious. Variables I and J in the program are used to keep track of which sides and/or angles are given. If one side and two angles are provided,

then the law of sines is used to solve the problem via lines 20 to 22. If two sides and the angle opposite one of them are given, the law of sines in the form expressed in lines 30 to 36 is used. If two sides and the included angle are given, then lines 40 to 44 are used to solve the problem. Although the law of cosines could be used in this situation, the method used here provides greater accuracy in certain circumstances. Finally, when all three sides are given, the angles are found using the law of cosines in lines 50 to 53.

Program listing: TRIANGLES

```

10: CLEAR : DIM A(6)
    ): INPUT "SIDE
    A? "; A(1): I=1
11: INPUT "SIDE B?
    "; A(2): I=I+2
12: INPUT "SIDE C?
    "; A(3): I=I+3
13: IF I=0 GOTO 50
14: INPUT "ANGLE A
    ? "; A(4): J=1
15: INPUT "ANGLE B
    ? "; A(5): J=J+2
16: INPUT "ANGLE C
    ? "; A(6): J=J+3
17: IF A(1)=0 GOTO
    30
20: L=90-A(4)-A(5)
    -A(6)+90: IF L<
    =0 GOTO 70
21: A(J+3)=L: FOR K
    =1 TO 3: IF K<>I
    LET A(K)=SIN A
    (K+3)*A(I)/SIN
    A(I+3)
22: NEXT K: GOTO 60
30: IF J=1 GOTO 40
31: G=A(J): H=A(6-I
    -J): L=A(J+3):
    IF H*SIN L>G
    GOTO 70
32: M=ASN (H*SIN L
    /G): A(9-I-J)=M
    : IF L+M>=180
    GOTO 70
33: A(I+3)=180-L-M
    : A(I)=G*SIN (L
    +M)/SIN L
34: IF G>H*SIN L IF
    H>G WAIT : PRINT
    "FIRST OF TWO
    SOLUTIONS: ": N=
    1
35: GOTO 60
36: A(9-I-J)=180-M
    : A(I+3)=M-L: A(
    I)=G*SIN (M-L)
    /SIN L: N=N+1:
    PRINT "SECOND
    SOLUTION: ":
    GOTO 60
40: L=A(I+3): FOR K
    =4 TO 6
41: IF K<>I+3 LET G
    =A(K+3): H=A(9-
    I-K): A(K)=90
42: IF H<>G*COS L
    LET A(K)=ATN (
    G*SIN L/(H-G*
    COS L))
43: IF -A(K) LET A(
    K)=180+A(K)
44: NEXT K: A(I)=J(
    G*G*SIN L*SIN
    L+(G*COS L-H)*
    (G*COS L-H)):
    GOTO 60
50: IF (A(1)+A(2)-
    A(3))*(A(1)+A(
    3)-A(2))*(A(2)
    +A(3)-A(1))<=0

```

```

GOTO 70
51: A(4)=ACS ((A(2)
    )*A(2)+A(3)*A(
    3)-A(1)*A(1))/
    (2*A(2)*A(3)))
52: A(5)=ACS ((A(1)
    )*A(1)+A(3)*A(
    3)-A(2)*A(2))/
    (2*A(1)*A(3)))
53: A(6)=ACS ((A(1)
    )*A(1)+A(2)*A(
    2)-A(3)*A(3))/
    (2*A(1)*A(2)))
60: K=.5*A(1)*A(2)
    *SIN (A(6)):
    WAIT :PRINT "S
    IDE A= ";A(1)

61:PRINT "SIDE B=
    ";A(2)
62:PRINT "SIDE C=
    ";A(3)
63:PRINT "ANGLE A
    = ";A(4)
64:PRINT "ANGLE B
    = ";A(5)
65:PRINT "ANGLE C
    = ";A(6)
66:PRINT "AREA= "
    ;K
67:IF NGOTO 36
68:END
70:WAIT :PRINT "N
    O SOLUTION"
STATUS 1      1210

```

4.5 RPN CALCULATOR

Have you ever wished that you could use your PC as a Reverse Polish Notation calculator? Many engineers and scientists like to use this notation when solving mathematical problems. Now, thanks to a program provided by Norlin Rober, you can use this method on your PC. The program provides double-precision RPN capability. It permits addition, subtraction, multiplication, and division. All inputs and outputs use scientific notation. Outputs are to 20 significant digits (with four additional “guard digits” calculated, but not displayed).

The RPN stack created by this program consists of four registers, referred to as X, Y, Z, and T. Register X uses variables A, B, C, and D. Register Y uses E, F, G, and H. Register Z uses variables I, J, K, and L. Register T is stored in M, N, O, and P. The first variable associated with each register contains the first eight digits of the mantissa, and the second variable holds the next eight digits. The third variable holds the last eight digits, and the fourth variable stores the decimal exponent.

For those unfamiliar with RPN, its operation is described briefly as follows: when an input occurs, it is stored in register X, with the previous contents of X shifted into Y, the previous contents of Y into Z, and those of Z into T. This shifting of the register contents is called a *stack lift*. An arithmetic operation is

performed using the contents of X and Y as operands. The calculated result is placed into register X. Such an operation is accompanied by a stack *drop*; that is, register Z is shifted into Y, and T drops into Z (as well as remaining in T).

The four registers may be viewed as “stacked” on top of each other in the form:

REGISTER T
REGISTER Z
REGISTER Y
REGISTER X

The RPN system is efficient, as it is very convenient to use previously calculated results in further operations. The primary principle to keep in mind is that an operation (add, subtract, multiply, or divide) is given *after* the numerical entries for that operation have been made and not between operands as is done with an algebraic calculator.

Program Operation

Load the program into memory, place the PC in its regular RUN mode, then select the desired operation using the DEF key coupled with a key shown in the following list of options:

DEF/L Lift stack, with input to replace X. Note: if ENTER is pressed without any input being keyed, the previous contents of X remain there. This is useful, for example, in squaring a number.

DEF/N Non-lifting input. The input simply replaces the contents of X. No stack lift occurs.

DEF/X Exchange the contents of X and Y. Useful when it is desired to exchange the operands prior to division or subtraction.

DEF/A Add X and Y. Result in X, with stack drop.

DEF/S Subtract X from Y. Result in X, with stack drop.

DEF/M Multiply X by Y. Result in X, with stack drop.

DEF/D Divide Y by X. Result in X, with stack drop.

Remember that inputs must be in scientific notation, including a decimal point after the first digit of the mantissa.

Negative numbers should have the minus sign entered only in the first 8-digit block.

The prompt *READY* is displayed after numerical or operand inputs. Keying ENTER (alone) shows the current contents of register X as a 20-digit mantissa. Keying ENTER a second time displays the exponent.

An Example

To calculate $(2.176513 + 408.99716238) / (79.312294758 - .4851339176288)$, proceed as follows:

1. **DEF/L** Provide these inputs in response to prompts:

1ST 8 DIGITS?	2.1765130
NEXT 8?	0
NEXT 8?	0
EXP?	0 (Note inputs in scientific notation.)

2. **DEF/L** (again) and enter the second operand as prompted:

1ST 8 DIGITS?	4.0899716
NEXT 8?	238000000
NEXT 8?	0
EXP?	2

3. **DEF/A** The contents of X and Y are added. The result is in X. You can view register X by keying ENTER (after *READY* appears). It should contain *4.1117367538000000000 EXP 2* at this point.

4. **DEF/L** Enter the denominator of the sample expression:

1ST 8 DIGITS?	7.9312294
NEXT 8?	758000000
NEXT 8?	0
EXP?	1

5. **DEF/L** (once more!) and provide:

1st 8 digits?	4.8513391
NEXT 8?	76288000

```
NEXT 8?      Ø
EXP?         -1
```

Right at this point, the internal stack contents are:

```
REGISTER Z    411.17367538
REGISTER Y    79.312294758
REGISTER X    .4851339176288
```

6. **DEF/S** to calculate $Y - X$ and drop the stack one notch. You can examine the result of the subtraction by pressing ENTER after the *READY* prompt appears. Following this operation the stack will contain:

```
REGISTER Y    411.17367538      (previously in Z)
REGISTER X    78.827160843712   (after subtracting)
```

7. **DEF/D** This performs the final operation, division, to yield the completely evaluated expression. To display the answer, key ENTER when the *READY* prompt shows up:

```
5.216142139 2893562452 EXP Ø
```

Now you are ready to go off and experiment on your own!

Program listing: RPN CALCULATOR

```
1: "L" M=I: N=J: O=K          ): X=1E-9*(ABS
   : P=L: I=E: J=F: K        (1E2*A)+INT X+
   =G: L=H: E=A: F=B         INT Y)
   : G=C: H=D                5: Y=1E10*(Y-INT
2: "N" Q=1E-4: R=1E          Y): Z=D: IF X=10
   8: S=1E-8                 LET X=1: Z=Z+1
3: INPUT "1ST 8 D           6: PRINT USING "#
   IGITS? "; A: A=1          #.#####"; X
   E7*A: B=0: C=0: D        *SGN A; USING "
   =0: INPUT "NEXT         #####"; Y
   8? "; B, "NEXT          : PRINT USING ;
   8? "; C, "EXP? "        "EXP "; Z: GOTO
   ; D                      6
4: WAIT : PRINT "R          7: ""E=J: F=J: G=K:
   EADY": X=1E-6*B           H=L: I=M: J=N: K=
   : Y=X-INT X+1E-          O: L=P: GOTO 4
   10*INT (C*Q+.5          8: "X" W=A: X=B: Y=C
```

```

:Z=D:A=E:B=F:C
=G:D=H:E=W:F=X
:G=Y:H=Z:GOTO
4
20:"S"A=-A
21:"A"U=SGN (A*E)
:IF U=0LET A=A
+E:B=B+F:C=C+G
:D=D+H:GOTO ""
22:Z=D-H:IF Z*1E1
6+(ABS A-ABS E
)*R+B-F+(C-G)*
SLET W=ABS E:X
=F:Y=G:GOTO 24
23:W=ABS A:X=B:Y=
C:A=E:B=F:C=G:
D=H:Z=-Z
24:IF Z>25GOTO ""
25:IF Z>7LET Z=Z-
8:Y=X+Y*S:X=W:
W=0:GOTO 25
26:IF ZLET Z=10^-
Z:Y=Y*Z+(X*Z-
INT (X*Z))*R:X
=INT (X*Z)+(W*
Z-INT (W*Z))*R
:W=INT (W*Z)
27:U=SGN A:A=ABS
(A)+W*U:B=B+X*
U:C=C+Y*U:GOTO
50
30:"M"U=SGN (A*E)
:A=ABS 1E-3*A:
B=1E-3*B:C=10*
C:D=D+H:E=ABS
E*Q:F=F*Q:W=A-
INT A
31:X=B-INT B:Y=E-
INT E:Z=F-INT
F:T=W*INT E+Y*
INT A:U=W*INT
F+X*INT E+Y*
INT B+Z*INT A
32:C=(B*G+C*F)*Q*
S+(U-INT U+X*Y
+W*Z)*R+A*G*Q+
B*F+C*E*Q
33:B=INT A*INT F+
INT B*INT E+
INT U+(T-INT T
+W*Y)*R:A=INT
A*INT E+INT T:
GOTO 50
40:"D"U=SGN E/SGN
A:D=H-D-I:H=
ABS A+B*S:A=
ABS A*Q:B=B*Q:
C=C*Q:T=A-INT
A:U=B-INT B
41:W=INT ((ABS E*
R+F)/H)*Q:X=W-
INT W:Y=U*INT
W+X*INT B
42:E=(ABS E-INT A
*INT W-T*INT W
-X*INT A-T*X)*
R+F-INT B*INT
W-INT Y
43:F=G-(Y-INT Y)*
R-U*X*R-C*INT
W-C*X:X=INT ((
E*R+F)/H)*Q:Y=
X-INT X
44:C=((((E-INT A*
INT X-T*INT X-
Y*INT A-T*Y)*R
+F-INT B*INT X
-U*INT X-Y*INT
B-U*Y)*R-C*X)/
H
45:A=1E4*W:B=1E4*
X
50:C=C*S+1:B=(B-1
+INT C)*S+1:A=
A-1+INT B:B=(B
-INT B)*R:C=(C
-INT C)*R
51:IF A=0IF B=0IF
C=0LET D=0:
GOTO ""
52:IF A=0LET A=B.
B=INT C:C=(C-
INT C)*R:D=D-8
:GOTO 52
53:IF A>=1E7THEN
56
54:U=1+INT (LOG A
):D=D+U-8:U=10
^-U:C=C*U:B=B*
U:A=A*U*R+INT
B
55:B=(B-INT B)*R+

```

```

INT C:C=(C-INT
C)*R
56: IF A>=RLIST A=
.1*A:B=,1*B+(A
-INT A)*R:C=.1
*C+(B-INT B)*R
:B=INT B:A=INT
A:D=D+1
57:A=A*V:GOTO ""
STATUS 1 1658

```

4.6 CURVE-FITTING

This program, artfully packaged by Thomas S. Cox, will fit a series of X,Y data points to the following types of curves: linear, exponential, logarithmic, and power. It is another good example of how a lot of practical computing power can be packed into a pocket computer.

To apply this package, load the program into your PC, and then place the PC in the normal program execution (RUN) mode. The various capabilities of the package are selected using a combination of the DEFine key and a key assigned to the function or operation that is to be performed, as described below. The combination of the DEFine key and another letter, such as S, will be abbreviated here as: DEF/S.

DEF/SPACE is used to initialize the program (the first time it is used after being loaded into memory or when you want to enter an entirely new curve). It is also used if you want to enter additional data points after a curve has previously been inputted to the PC. Here is an example of the use of DEF/SPACE:

```

CURVE FIT-V4
CLR (Y/N)Y
PT# 1
X=1
Y=1
X 1= 1 Y 1= 1
PT# 2
X=3
Y=3
X 2= 3 Y 2= 3
PT# 3
X=5
Y=5
X 3= 5 Y 3= 5

```


Note that the program will verify each pair of X,Y inputs. When you have entered a sufficient number of points for the curve you are evaluating, select the next desired operation from among those described below.

DEF/F is used to have the program automatically determine the best fit for the data that has been entered. This is selected on the basis of the highest R^2 value obtained. Information concerning the best fit may then be obtained by pressing the ENTER key. Successive use of the ENTER key will result in information being displayed in the following sequence:

1. Curve type and descriptive equation.
2. Constant equation term A.
3. Variable equation term B.
4. Coefficient of correlation R.
5. Coefficient of determination R^2 .

Here is how the information would be outputted for the example data points used earlier in this discussion:

```
BEST FIT WAS # 1
LIN. Y=B*X+A
A=      0.000
B=      1.000
R=      1.000
R^2=    1.000
```

You may view the parameters of any of the four curves provided in the package, for the data that has been inputted, by using the following DEFine key combinations:

- DEF/L linear curve.
- DEF/A exponential curve.
- DEF/S logarithmic curve.
- DEF/D power curve.

You can obtain an estimate of an X or Y value for the currently selected curve (whether selected manually or by the best-fit routine). Use DEF/K to obtain a predicted value for X or DEF/J for a predicted value of Y. Here is an example of program

operation (using the example data points) when DEF/J is used to obtain a predicted value for Y:

ESTIMATE OF Y

X=9

X= 9.000 Y= 9.000

This program is intended for use on curves that are in the quadrant where both X and Y are positive. If either is negative, then the best fit procedure is skipped. The message:

X OR Y OR BOTH NEGATIVE

will then be displayed. You may attempt to use manual curve selection to obtain a best fit. However, if this is not feasible, the PC will yield the same message:

X OR Y OR BOTH NEGATIVE

This program is particularly valuable in applications where you desire to extrapolate data against several different curves, in order to project current data trends. Having all four types of curves available for instant analysis is quite a boon when you are involved with such studies. Remember that the DEF/SPACE allows you to add points to a curve. Thus you can update a curve over time to ascertain whether real-time observations are matching predicted behavior. (Chemical engineers should love having in their pockets a program with this kind of analytical power!)

Program listing: CURVE FITTING

1: " USING :	3: X=L: INPUT "Y="
PRINT "CURVE F	;Y: IF Y<=0LET
IT-U4": INPUT "	T=1
CLR (Y/N)"; Z\$:	4: GOSUB 5: BEEP 2
IF Z\$="Y" CLEAR	: PRINT "X"; N; "
2: C=1: Z=N+1:	="; X; " Y"; N; "=
PAUSE "PT#"; Z:	"; Y: GOTO 2
INPUT "X="; L:	5: D=D+C*X: E=E+C*
IF L<=0LET J=1	X*X: F=F+C*Y: G=

```

G+C*Y*Y:H=H+C*
X*Y:N=N+1*C:IF
J*T=1RETURN
6:IF J<>1LET M=M
+C*LN X:O=O+C*
LN X*LN X:K=K+
C*Y*LN X
7:IF T<>1LET P=P
+C*LN Y:Q=Q+C*
LN Y*LN Y:I=I+
C*X*LN Y
8:IF J+T=0LET S=
S+C*LN X*LN Y
9:RETURN
10:"L"IF W=1THEN
14
11:U=1:PRINT "LIN
, Y=B*X+A"
12:B=(H-D*F/N)/(E
-D*D/N):A=(F/N
-B*D/N)
14:R=(N*H-D*F)/J(
(N*E-D*D)*(N*G
-F*F)):IF W=1
RETURN
16:GOTO 96
20:"A"IF W=1THEN
24
21:U=2:PRINT "EXP
, Y=A*EXP(B*X)
":IF T=1GOTO 6
1
22:B=(I-P*D/N)/(E
-D*D/N):A=EXP
(P/N-B*D/N)
24:R=(N*I-D*P)/J(
(N*E-D*D)*(N*Q
-P*P)):IF W=1
RETURN
26:GOTO 96
30:"S"IF W=1THEN
34
31:U=3:PRINT "LOG
, Y=B*LN X+A":
IF J=1GOTO 61
32:B=(K-M*F/N)/(O
-M*M/N):A=(F-B
*M)/N
34:R=(N*K-F*M)/J(
(N*O-M*M)*(N*G
-F*F)):IF W=1
RETURN
36:GOTO 96
40:"D"IF W=1THEN
44
41:U=4:PRINT "PWR
, Y=A*X^B":IF
T+J>0GOTO 61
42:B=(S-M*P/N)/(O
-M*M/N):A=EXP
(P/N-(B*M/N))
44:R=(N*S-M*P)/J(
(N*O-M*M)*(N*Q
-P*P)):IF W=1
RETURN
46:GOTO 96
48:"F"IF J+T>0
THEN 61
50:W=1:U=0:U=0:
FOR Z=1TO 4:
GOSUB (Z*10):C
=R*R:IF C>U
GOSUB 99
55:NEXT Z:USING :
W=0:BEEP 3:
PRINT "BEST FI
T WAS #":U
59:GOTO (U*10)
60:"K"PRINT "ESTI
MATE OF X":
INPUT "Y=":U:
GOTO (U+61)
61:BEEP 1:PRINT "
X OR Y OR BOTH
NEG.":END
62:L=(U-A)/B:GOTO
97
63:L=(LN U-LN A)/
B:GOTO 97
64:L=EXP ((U-A)/B
):GOTO 97
65:L=EXP ((LN U-
LN A)/B):GOTO
97
67:USING "####.#
##":RETURN
70:"G"BEEP 1:
PRINT "DELETE"
:INPUT "DELETE
LAST PT (Y/N)
":Z$:IF Z$="Y"
THEN 75

```

```

71:PRINT "PT# ";N
   :INPUT "X=";L:
   IF L<=0LET J=1
72:X=L:INPUT "Y="
   :Y:GOSUB 98:
   GOTO 77
75:PRINT "X=";X;"
   Y=";Y:INPUT "
   OK TO DEL(Y/N)
   ";Z$
76:IF Z$="N"THEN
   71
77:C=-1:IF Y<=0
   LET T=1
78:GOSUB 5:END
82:"J"PRINT "ESTI
   MATE OF Y":
   INPUT "X=";L:
   GOTO (U+85)
86:V=B*L+A:GOTO 9

```

```

7
87:V=A*EXP (B*L):
   GOTO 97
88:V=B*LN L+A:
   GOTO 97
89:V=A*L^B:GOTO 9
7
96:GOSUB 67:PRINT
   "A=";A:PRINT "
   B=";B:PRINT "R
   =";R:C=R*R:
   PRINT "R^2=";C
   :END
97:PRINT "X=";L;"
   Y=";V:END
98:PRINT "X=";X;"
   Y=";Y:RETURN
99:U=Z:V=C:RETURN
STATUS 1      1677

```

4.7 Z PLOT

Would you like to be able to represent three-dimensional surfaces using your PC connected to its optional printer/plotter? If so, try this program. It enables the plotter to draw curves showing the intersection of a number of equally spaced cutting planes. The program was designed by Norlin Rober.

To use the program, start off by loading the listing shown into memory. Insert the function(s) you want plotted at lines 100 and 200 (a sample function is included in line 100 of the program listing). Note that you can, if you wish, describe one or two functions. Place the formula for one function at line 100 and the other at line 200. (If necessary you can use several program lines to describe each function. Just be sure the definition ends with a RETURN statement, as provided in lines 199 and 299.)

Once you have defined the function(s) you want to use, place the PC in the RUN mode. Connect the PC to the printer/plotter unit. Place the program in operation by executing a *RUN* command.

Respond to the program prompts given for minimum and maximum values of X, Y, and Z. You will also need to specify the number of cutting planes to be used (number of sections) and

the increment size to be used. Finally, indicate whether you are going to plot one or two functions.

Quite a large number of points will typically be plotted, hence it may take a number of minutes to complete a graph. The use of larger increment sizes and relatively few sections will speed up the plotting process. However, less detail will show in the drawing. Experience indicates that a specification of 7 to 11 sections and an increment size of 2 to 5 usually represents a suitable compromise between detail and the amount of time required to finish a picture.

The X, Y, and Z axes are not included in the drawings, because they tend to clutter up the results. These drawings are intended to show surface shapes rather than provide precise graphs from which to obtain analytical values.

Here is how you might respond to the program prompts to obtain a drawing when the function $Z = \text{EXP}(-(X*X+Y*Y)/2)$ is in line 100:

```
MIN X? -2.5
MAX X? 2.5
MIN Y? -2.5
MAX Y? 2.5
MIN Z? 0
MAX Z? 1.3
NUMBER OF SECTIONS? 9
INCREMENT SIZE? 2
NUMBER OF FUNCTIONS? 1
```

At this point, the program activates the plotter unit, prints the function(s) being plotted as well as the minimum and maximum values of the parameters specified, and then produces the drawing that represents the three-dimensional surface. The output for the function used as an example is included for reference.

By the way, make note of the fact that if you define two functions (at lines 100 and 200), then you should inform the program appropriately when it prompts for the number of functions.

If you are a program “tinkerer,” you may want to modify the program so that it uses different colors to represent the various planes. You could get some fascinating results!

Program listing: Z PLOT

```

10: INPUT "MIN X? " ; A
11: INPUT "MAX X? " ; B
12: INPUT "MIN Y? " ; C
13: INPUT "MAX Y? " ; D
14: INPUT "MIN Z? " ; E
15: INPUT "MAX Z? " ; F
16: INPUT "NUMBER OF SECTIONS? " ; G: G=G-1
17: H=1: INPUT "INCREMENT SIZE? " ; H
18: N=1: INPUT "NUMBER OF FUNCTIONS? " ; N
20: CLS : TEXT : CSIZE 2
21: LPRINT "GRAPHE D: " : FOR I=1 TO N: LLIST 100*I, 100*I+99: LF -3 : NEXT I: LF 1
22: LPRINT "MIN X=" ; A: LPRINT "MAX X=" ; B: LPRINT "MIN Y=" ; C: LPRINT "MAX Y=" ; D: LPRINT "MIN Z=" ; E: LPRINT "MAX Z=" ; F
30: XS=(B-A)/144, YS=(D-C)/144, ZS=(F-E)/144, GS=144/G
40: GRAPH : GLCURSOR (0, -20): SORGN
50: FOR I=1 TO N
51: FOR J=0 TO G: K=1: XC=J*GS: FOR YC=0 TO 144 STEP H: GOSUB 60: NEXT YC: NEXT J
52: FOR J=0 TO G: K=1: YC=J*GS: FOR XC=0 TO 144 STEP H: GOSUB 60: NEXT XC: NEXT J
53: NEXT I
54: GLCURSOR (0, -70): END
60: X=B-XS*XC: Y=C+YS*YC: GOSUB 100*I: L=.5*XC+YC: M=.5*XC+ZS*(Z-E)
61: IF Z<E OR Z>F LET K=1: RETURN
62: IF K GLCURSOR (L, M): K=0: RETURN
63: LINE -(L, M): RETURN
100: Z=EXP (-(X*X+Y*Y)/2)
199: RETURN
200: REM 2ND FUNCTION
299: RETURN
STATUS 1 735

```

Sample output

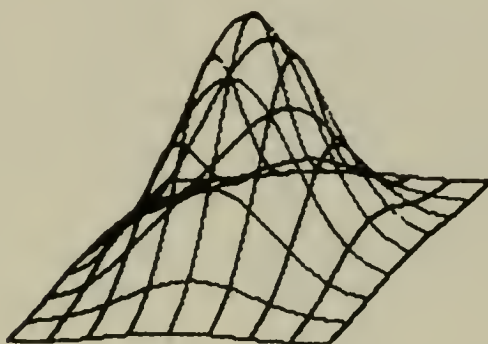
GRAPHED:

```

100: Z=EXP (-(X*X+Y
      *Y)/2)
199: RETURN

```

```
MIN X=-2.5  
MAX X= 2.5  
MIN Y=-2.5  
MAX Y= 2.5  
MIN Z= 0  
MAX Z= 1.3
```



4.8 HISTOGRAM

In order to utilize the full capabilities of this program, you will need to equip your PC with an 8K memory module. You may, however, be able to use the program for limited applications if you have a 4K memory module. If you do want to try it, you must change the DIMension statement in line 100 to about 75 elements for variable X. Either way, you also need to have the companion printer/plotter unit. This program is designed to consolidate data into graphic form by displaying a histogram.

The computer makes a histogram by sorting the data inputted into equally spaced cells or zones. You define the width of these bands, and then the computer counts the number of data points that fall into each zone. Finally, it plots a graph of the results on a cell-by-cell basis. The program also calculates fundamental statistical information about the data sample inputted by the user. This statistical information includes: the mean, standard deviation, minimum, maximum, and the variance.

Load the program into memory, and then place the PC in the RUN mode. Start the program by executing a *RUN* command

or pressing the DEFine key and the letter A. The program will begin by asking if you need instructions. If so, it will print a summary of the keys that control its operation. Next, the program prompts for the data points. Input your data at this time. Here is a sample series you might want to use for test purposes:

X(1)	?3.9
X(2)	?4.2
X(3)	?4.4
X(4)	?5.7
X(5)	?6.3
X(6)	?6.4
X(7)	?6.6
X(8)	?7.1
X(9)	?7.7
X(10)	?7.9
X(11)	?8
X(12)	?8
X(13)	?8.1
X(14)	?8.1
X(15)	?8.1
X(16)	?8.1
X(17)	?8.2
X(18)	?8.5

Once the data has been inputted, the DEFine key is used in combination with another letter to select operations as described below:

DEF/S Statistical data. The computer will calculate and display the data minimum and maximum values, the mean, the standard deviation, and the variance.

DEF/H Histogram plot. The program will ask for the cell width desired when making the plot. After inputting this value (try a width of 1 for the sample data provided above), the PC will pause for awhile as it mulls over the data. After a period of time (depending on the number of data points that must be analyzed) it will draw the histogram. It does this using three different pen colors. The resulting histogram is appropriately labeled. The vertical axis presents the number of data points in the cells, expressed as a percentage of the total points in all cells.

DEF/C Cell data. The program will print a list giving the number of data points within each cell. This command must be preceded by the DEF/S and DEF/H commands in order to be valid.

DEF/L Listing of raw data. This command directs the printer to list the raw data points for review and archival purposes. This command should be preceded by the DEF/S and DEF/H commands.

DEF/A Add more data. You can expand your initial data points by using this directive. The PC will prompt for inputs continuing from the last data point provided by the user. Data inputs are terminated by selecting one of the other DEFine/key commands. Note that you should select the DEF/S and DEF/H directives (after using this option) before utilizing DEF/C or DEF/L.

Please note that the program is written to accept data that has up to two significant digits on each side of the decimal point. If you want a larger range, you will need to alter the *USING* statements in the program. Remember, though, you can frequently scale your data to fall within the original capabilities of this program.

Russ Doughty is the creator of this program. He asks that you be patient when working with a lot of data points. Remember, while your PC is powerful, it is not as fast as an IBM 370!

Program listing: HISTOGRAM

```

100: CLEAR :H=1: DIM          210: PRINT "X(";H; "
      X(255): DIM F(5         )": CURSOR 8:
      1)                      INPUT X(H): CLS
108: WAIT 100: PRINT          220: H=H+1: GOTO 210
      "                      500: "S"TEXT :
      OGRAM"                  LPRINT : COLOR
112: INPUT "NEED IN           0: LPRINT "STAT
      STRUCTIONS(Y/N         ISTICS": LPRINT
      )? "; Z$: IF Z$=       510: FOR J=1 TO H:
      "Y" GOSUB 2000          FOR Q=1 TO H-J
114: PRINT "ENTER D          530: A=X(Q): C=X(Q+1
      ATA"                   )
200: "A" WAIT 5              550: IF A<C THEN 580

```

```

560: X(Q)=C: X(Q+1)=
      A
580: NEXT Q
590: NEXT J
600: Z=0: S=0: FOR I=
      2 TO J: S=S+X(I)
      : NEXT I: M=S/(J
      -1)
605: FOR I=2 TO J: Z=
      Z+X(I)*X(I):
      NEXT I
607: V=(Z-(J-1)*M*M
      )/(J-2): CSIZE
      1
610: LPRINT USING "
      #####.##"; "DA
      TA MIN = "; X(2
      )
620: LPRINT "DATA M
      AX = "; X(J)
630: LPRINT "MEAN
      = "; M:
      LPRINT "STD DE
      V = "; V
640: LPRINT "VARIAN
      CE = "; V
645: LPRINT "ENTRIE
      S = "; H-1
650: WAIT : PRINT
805: "H" INPUT "CELL
      WIDTH "; W
810: Y=INT ((X(J)-X
      (2))/W)+1
815: FOR I=0 TO Y: F(
      I)=0: NEXT I
820: FOR I=0 TO Y
830: FOR Q=2 TO J
835: R=INT (X(2))
840: IF X(Q)>=(R+(I
      *W)) AND X(Q)<
      (R+((I+1)*W))
      THEN 848
844: GOTO 850
848: F(I)=F(I)+1
850: NEXT Q: NEXT I
905: "G" TEXT : LF 2:
      GRAPH :
      GLCURSOR (30, -
      150)
910: SORGN : LINE -(
      180, 170), 0, 1, B
      920: A=0: M=0
      930: FOR I=0 TO Y
      940: IF A<F(I) THEN
          LET A=F(I)
      945: M=M+F(I)
      950: NEXT I
      960: D=A
      970: M=INT (D/M*100
          )
      1000: A=150/D: E=14
          0/(Y+1)
      1010: GLCURSOR (6,
          0): SORGN
      1020: FOR I=0 TO Y
      1030: LINE ((I*E),
          0)-(((I+1)*E
          ), (F(I)*A)),
          0, 3, B
      1040: NEXT I
      1043: GLCURSOR (-6
          , 0): SORGN
      1044: FOR I=0 TO 7
          STEP 2
      1045: LINE (0, (10*
          D*A-I*D*A)/1
          0)- (180, (10*
          D*A-I*D*A)/1
          0), 2, 1: NEXT
          I
      1050: TEXT : CSIZE
          1: COLOR 0: LF
          4: LPRINT
          USING "#####
          .##"; INT (X(
          2)): TAB 26:
          INT (X(2))+W
          *(Y+1)
      1055: LF -18:
          LPRINT TAB (
          15); "HISTOGR
          AM": LF -2
      1056: LPRINT "%":
          LPRINT "TOT"
          : USING "###"
          : LPRINT TAB
          (1); M
      1058: FOR I=2 TO 7
          STEP 2: LF 2:
          LPRINT TAB (
          1); INT ((10*
          M-I*M)/10):

```

```

        NEXT I
1060:USING "####"
      .##";LF 5;
      LPRINT TAB (
        6);"MAX FREQ
      . = ";D;
      LPRINT TAB (
        6);"CELL WID
      TH= ";W
1065:WAIT :PRINT
1070:"C"LPRINT :
      LPRINT :
      LPRINT TAB 1
      5;"CELL DATA
      "
1080:LPRINT :FOR
      I=0TO Y;
      USING "####"
      :LPRINT TAB
      (5);"CELL ";
      I+1;" = ";;
      USING "####.
      ##";LPRINT F
      (1);
1081:LPRINT INT (
      X(2)+I*W);
      NEXT I
1082:WAIT :PRINT
1085:"L"LPRINT :
      LPRINT :
      LPRINT TAB (
      12);"RAW DAT
      A LISTING";
      LPRINT
1090:FOR I=2TO H:
      USING "####"
      :LPRINT TAB
      (9);"DATA PT
      .";I-1;" = "
      ;;USING "###
      .##";LPRINT
      X(I)
1091:NEXT I
1092:WAIT :PRINT
      :GOTO 100
2000:CSIZE 1;
      LPRINT "DEF/
      S = STATISTI
      CS";LPRINT "
      DEF/H = HIST
      OGRAM"
2010:LPRINT "DEF/
      C = HISTOGRA
      M CELL DATA"
      :LPRINT "DEF
      /L = RAW DAT
      A LISTING"
2015:LPRINT "DEF/
      A = ADD MORE
      DATA"
2020:LPRINT "YOU
      CAN ENTER UP
      TO 255 DATA
      POINTS"
2030:LPRINT "YOU
      MUST USE DEF
      /S & H BEFOR
      E C OR L"
2040:LF 7:RETURN
STATUS 1      1853

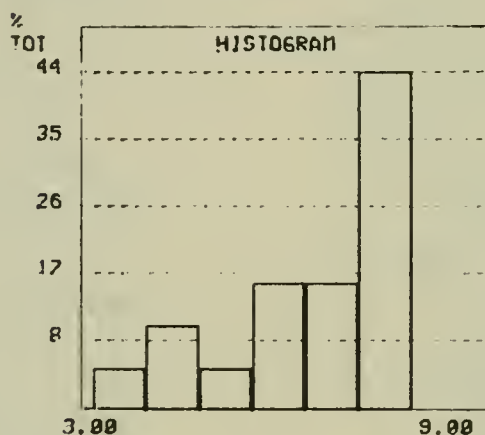
```

Sample output

DEF/S = STATISTICS
 DEF/H = HISTOGRAM
 DEF/C = HISTOGRAM CELL DATA
 DEF/L = RAW DATA LISTING
 DEF/A = ADD MORE DATA
 YOU CAN ENTER UP TO 255 DATA POINTS
 YOU MUST USE DEF/S & H BEFORE C OR L

STATISTICS

DATA MIN = 3.90
 DATA MAX = 8.50
 MEAN = 6.96
 STD DEV = 1.50
 VARIANCE = 2.27
 ENTRIES = 18.00



MAX FREQ. = 8.00
 CELL WIDTH = 1.00

CELL DATA

CELL	1 =	3.00	4.00
CELL	2 =	4.00	5.00
CELL	3 =	5.00	6.00
CELL	4 =	6.00	7.00
CELL	5 =	7.00	8.00
CELL	6 =	8.00	9.00

RAW DATA LISTING

DATA PT.	1 =	3.90
DATA PT.	2 =	4.20
DATA PT.	3 =	4.40
DATA PT.	4 =	5.70
DATA PT.	5 =	6.30
DATA PT.	6 =	6.40
DATA PT.	7 =	6.60
DATA PT.	8 =	7.10
DATA PT.	9 =	7.70
DATA PT.	10 =	7.90
DATA PT.	11 =	8.00
DATA PT.	12 =	8.00
DATA PT.	13 =	8.10
DATA PT.	14 =	8.10
DATA PT.	15 =	8.10
DATA PT.	16 =	8.10
DATA PT.	17 =	8.20
DATA PT.	18 =	8.50

Personal Programs

5.1 AVIATION

Here is a program for aviators to use when plotting a flight plan. It does all those tedious calculations involved in determining the wind correction angle (WCA), finding the true heading needed to fly a particular course, calculating the estimated fuel burn, and so forth.

To put the program to use, load it into your PC, place the PC in the RUN mode, and press DEF/A (the DEFine key and then the letter A). Respond to the prompts given by the program. They go along these lines:

1. The computer will ask for the local variance from magnetic North. Input this value in degrees with an appropriate sign. Use a minus sign for easterly variations. Westerly variations should be entered as positive values. Press ENTER to input all data for this program.

2. Input the wind direction in degrees.

3. Input the anticipated wind velocity in statute miles per hour. Since most flight service stations (FSS) report wind velocity in knots, a separate conversion routine is provided as part of this program package to convert knots to miles per hour. It is invoked using DEF/B. You should perform this conversion, if necessary, prior to using this part of the program.)

4. Enter the true air speed (TAS) at which you plan to fly your craft.

5. Input the fuel burn rate for your craft as gallons per hour.

6. Enter the plotted true course.

7. Enter the distance of the flight (or leg) in statute miles.

8. Read the following calculated information as it is outputted by the PC (press ENTER to obtain each parameter): the wind correction angle, the true heading at which to fly your airplane with the wind taken into account, the magnetic heading, the projected ground speed in miles per hour, the estimated enroute time, and the projected fuel usage.

9. If desired, input additional information for a new flight leg and obtain further flight calculations.

Here is a sample run of the program when it has been placed into operation using DEF/A:

```

FLIGHT DATA
VARIANCE (E-W+)
?5
WIND DIRECTION
?30
WIND VELOCITY
?10
T.A.S.
?110
BURN
?5.5
TRUE COURSE
?10
DISTANCE
?100
W.C.A.                1.781769962
TRUE HEADING          11.78176996
MAG. HEADING          16.78176996
GROUND SPEED          100.5498892
TIME ENROUTE          59.67187082
FUEL USED              5.469921492
NEW LEG? (Y/N)
?N
HAVE A GOOD FLIGHT
    
```

Here is a sample of what you would get if you invoked DEF/B to bring up the knots-to-mpg conversion routine:

```

KNOTS TO MPH
ENTER KNOTS
?15
17.25 M.P.H.
    
```

This program was provided by P. I. C. Gene Robertson.

Program listing: AVIATION

```

10: "A"
20: PAUSE "FLIGHT
    DATA"
30: INPUT "VARIANC
    E(E-W+)", A
40: INPUT "WIND DI
    RECTION", B, "WI
    ND VELOCITY", C
50: INPUT "T.A.S. "
    , D, "BURN", E
60: INPUT "TRUE CO
    URSE", F
70: INPUT "DISTANC
    E", G
80: H=ASN ((C/D)*
    SIN (B-F))
90: WAIT :PRINT "W
    .C.A.", H
100: I=H+F
110: PRINT "TRUE HE
    ADING", I
120: J=I+A
130: PRINT "MAG. HE
    ADING", J
140: K=D*COS (I-F)-
    C*COS (B-F)
150: PRINT "GROUND
    SPEED", K
160: L=G/K
170: M=L*60
180: PRINT "TIME EN
    ROUTE", M
190: N=E*L
200: PRINT "FUEL US
    ED", N
210: INPUT "NEW LEG
    ?(Y/N)", A$
220: IF A$="Y"GOTO
    30
230: PRINT "HAVE A
    GOOD FLIGHT"
240: END
250: "B"
260: PAUSE "KNOTS T
    O MPH"
270: INPUT "ENTER K
    NOTS", O
280: P=O*1.15
290: PRINT P; " M.P.
    H."
300: END
STATUS 1          532

```

5.2 SUPERMARKET

Are you always exceeding your budget at the supermarket? This little program can help put an end to that problem. Take it along with you in your PC. It will keep a tally of your purchases, including sales tax, if desired. Tell it your budget and it will warn you if you exceed the preset amount.

Load the program, and then put the PC into the RUN mode. You can put the package in operation by keying in the normal *RUN* directive. Alternatively, you can key DEF/SPACE to place the program into operation.

The program will ask you for your local (or state) sales tax rate as a percentage. It will also ask for the budget amount

allowed for the current shopping trip. Once these queries have been answered, the program will display: *ENTER AMOUNT*. You now have several options:

1. For nontaxable items, enter the amount of the intended purchase. Thus if a loaf of bread costs \$1.29, enter the value 1.29.

2. If the item is taxable, enter the amount of the item followed by the multiplication sign and the variable name: *T*. Thus an item at \$1.29 would be entered as: $1.29 * T$. The program will then automatically include the sales tax. It does that by storing the sales tax in the variable *T* at the beginning of the program. If, for example, you entered a tax of six percent (6%) when the program was initialized, the variable *T* will contain 1.06. Remember, the PC will evaluate arithmetic expressions as they are inputted. So, entering the statement $1.29 * T$ causes the expression to be evaluated (in this instance) as $1.29 * 1.06$.

3. Press ENTER alone to display totals. This causes the computer to display the current accumulated total and the amount remaining in your current budget.

4. Input the letter E to terminate the input loop. This will cause the program to display final totals. This operation works in the same manner as that used to indicate a taxable item. The variable *E* is initialized to a value that is not likely to be entered as the price of an item. When the program detects this unlikely value as an input, it directs program control to the termination portion.

5. Input a credit. If you want to cancel a previously entered charge, enter the value again as a negative value. If the original item included tax, then append the “times *T*” suffix. An example would be $-1.29 * T$.

If you go over your budget amount, the program issues a beep and a message to warn you. You can, however, continue to enter purchases and have them tallied by the PC.

Ken Slaughter devised this program. You can change the *PRINT* statements to *LPRINT*, if you like, and use the program with a printer to get a hardcopy of your shopping spree.

Here is a brief sample of the program’s operation:


```

SUPERMARKET
INITIALIZE...
SALES TAX (AS%) 7.5
BUDGET LIMIT? 20
ENTER AMOUNT: 15.95
ENTER AMOUNT: 6.50*T
WARNING...
OVER BUDGET!
T= 22.93 R= -2.93
ENTER AMOUNT: E
T= 22.93 R= -2.93

```

Program listing: SUPERMARKET

20: " "CLEAR	0
22: USING "####.#	100: X=0: INPUT "ENT
#"	ER AMOUNT: ";X
25: E=10^-20	110: IF X=0GOSUB 20
30: PAUSE "SUPERMA	0
RKET"	120: IF X=EGOTO 300
40: PAUSE "INITIAL	130: Z=Z+X
IZE..."	140: R=B-Z
50: INPUT "SALES T	150: IF R<=0BEEP 1:
AX (AS%) ";X	PAUSE "WARNING
60: IF X<0PAUSE "W	...":PAUSE "OU
HOLE NUMBERS!"	ER BUDGET!";
:GOTO 50	GOSUB 200
70: IF X=0PAUSE "W	190: GOTO 100
ARNING...":	200: WAIT :PRINT "T
PAUSE "SALES T	=";Z;" R=";R:
AX...":PAUSE "	RETURN
NOT INITIALIZE	300: FOR I=1TO 3
D":GOTO 100	310: PAUSE "END..."
80: T=1+(.01*X)	320: NEXT I
90: INPUT "BUDGET	330: GOSUB 200
LIMIT? ";B	340: END
95: IF B<=0:GOTO 9	STATUS 1 464

5.3 MOON PHASES

Given any date between March 1, 1900 and February 28, 2100, this program will respond with the phase of the moon on that date.

Gardeners, astronomers, astrologists, fishermen, and even plain, ordinary citizens frequently like to muse about the possible influence of the moon. Now you can use your PC to give

you the phase of the moon on any date of interest. Was the moon full the day you were born? Will it be full next Halloween? Plug in the dates and see for yourself!

To use the program, load it into memory, and then place the PC in the RUN mode and execute the *RUN* command. Follow the example here:

```
MOON PHASE CALCULATOR
DATE IN QUESTION>>-->
MONTH: 12
DAY: 25
YEAR: 1983
PHASE: LAST QUARTER
CYCLE IS 6 DAYS INTO 30
```

So there you have it—the moon will be going into its last quarter on Christmas of 1983. By the way, the program considers

Program listing: MOON PHASES

```
10: CLEAR : BEEP 1:
    PAUSE "MOON PH
    ASE CALCULATOR
    "
20: W=694098, X=29.
    53059167, Z=365
    .25: USING
30: BEEP 1: PAUSE "
    DATE IN QUESTI
    ON>>--> "
40: INPUT "MONTH:
    "; M, "DAY: "; D,
    "YEAR: "; Y: IF
    Y<100 LET Y=190
    0+Y
50: IF M>2 GOTO 60
55: S=((INT (30.6*
    (M+13))+INT (Z
    *(Y-1))+D-W)+1
    4.6)/X: S=S-INT
    S: S=INT (S*X+1
    ): GOTO 70
60: S=((INT (30.6*
    (M+1))+INT (Z*
    Y)+D-W)+14.6)/
    X: S=S-INT S: S=
    INT (S*X+1)
65: BEEP 1
70: IF (S=0)+(S=1)
    +(S=29)+(S=30)
    >0 PRINT "PHASE
    : FULL MOON":
    GOTO 110
80: IF (S=14)+(S=1
    5)+(S=16)>0
    PRINT "PHASE:
    NEW MOON": GOTO
    110
90: IF (S=6)+(S=7)
    +(S=8)>0 PRINT
    "PHASE: LAST Q
    UARTER": GOTO 1
    10
100: IF (S=21)+(S=2
    2)+(S=23)>0
    PRINT "PHASE:
    FIRST QUARTER"
110: PRINT "CYCLE I
    S"; S; " DAYS IN
    TO 30"
120: GOTO 30
STATUS 1 581
```

a lunar cycle to begin and end with the moon in its full phase. The days into a cycle for the given date are provided at the conclusion of each use of the routine.

5.4 MEMO PAD

You can use your pocket computer as a portable electronic memo pad. Enter important notes such as dates and times of appointments, names and telephone numbers, shopping lists, data from laboratory experiments, stock prices, or whatever you want. Then use the PC to retrieve the desired information at your command. If you can't remember a contact's phone number, just enter the name, and the computer will fetch it and the corresponding phone number. When you are through with a particular piece of information, electronically erase it from your pocket memo pad.

The program that enables the PC to perform these types of functions is relatively short and quite straightforward. Basically, it enables the computer to perform three basic operations:

1. It provides for the entry of data into the PC in the form of a key word and a corresponding item. For example, you can enter a person's name as a key word. That person's phone number might then be considered the corresponding item.
2. It provides the capability for the operator to edit whatever information is currently stored; that is, you may add to, change, or delete the data whenever desired.
3. Finally, you may use the program to locate an item that is associated with a key word; that is, you can search the computer memory for a particular reference. Type in a person's name, for example, and the program will recover the entry under that name, thereby revealing the associated phone number.

To use the program, load it into memory, place the pocket computer in its standard program execution (RUN) mode, and start the program with a *RUN* command. When the program is started it will ask:

CLEAR ALL (Y/N)?

Enter Y for “yes” only if you wish to erase all of the elements in the data storage array. It is a good idea to do this the very first time you use the program after loading it into memory. It is also a quick way to erase all previous entries when you wish to start filling your electronic memo pad over again from a “clean slate.” Any response other than Y will cause the program to skip the clearing operation and proceed to query:

EDIT OR FIND (E/F)?

If you wish to add to, alter, or otherwise update the current contents of the memo pad, enter an E for “edit.” If you want to search for a particular item using a key word, then input F for “find.”

If you tell the program that you want to edit an item, the PC responds with:

KEY #?

Tell it the number (from 1 to 20) of the entry in the memo pad that you want to edit.

The program will then respond by displaying the current information stored under that entry number. This is how such a display would appear as far as format is concerned:

1 JOHN DOE=123-4567 C/R

The display format is: entry number, key word, data item, and then the prompt C/R to remind the operator to press ENTER when finished reviewing the information on the display. (The prompt C/R will not appear if both the key word and data item contain the maximum 10 characters.) This method of displaying the information currently stored before proceeding to the editing operation allows the user to verify that the correct memo pad entry has been retrieved.

When you have finished observing the entry, press the ENTER key. The PC will ask:

NEW KEY?

If you want to change the key word (the information to the left of the equal sign when a data entry is displayed), simply input the new information. If you want to leave the present key word alone, press ENTER. Now the program will go on to query:

NEW DATA?

Again, if you want to change the data item (the information to the right of the equal sign when a data entry is displayed), input the new data. Otherwise, just press ENTER.

The program will then return back to the prompt:

EDIT OR FIND (E/F)?

That is all there is to entering information in the memo pad. Of course, when you first start using the pad, each entry will be blank, so all you will see is:

1 = C/R

when you, for example, call up the first entry for editing. Once data has been inputted for an entry you will see the new information the next time that entry is referenced.

Once you have data stored in memory you can use the find function to retrieve items of particular importance. Thus when the query:

EDIT OR FIND (E/F)?

comes up you would input F for "find." Now the PC will query:

KEY WORD?

You would now enter the key word of interest. For instance:

KEY WORD? JOHN DOE

The program will respond by searching all of the key word entries in the memo pad for a matching entry. As it searches, it will briefly flash the contents of each entry on the LCD. This provides a means for you to review the contents of the pad. You do not, however, have to bother watching the display if you do

not want to, for when a match is found with the key word specified the PC will alert you by issuing an audible “beep.” It will then display the entry where the match was found. After you have made note of the displayed information, press ENTER. The program will then go back to the prompt:

EDIT OR FIND (E/F)?

If, however, a match is not found, the display will read:

NOT FOUND... C/R

Again, you return to the “edit or find” prompt by pressing the ENTER key.

Applications

Here are several practical uses to which you might want to consider applying this program:

1. An appointment organizer. You can assign days of the week as key words. Then place the time of day and the initials of the person with whom you have an appointment in the data item part. Entries might appear as:

```
1 FRIDAY=11:00 JB
2 SATURDAY=1PM WALT
```

Since entries are limited to 10 characters (both for key words and data items), it is often advisable to abbreviate.

Since the key word can be anything you want, you might want to organize such an appointment scheduler differently. If, for example, you have a lot of appointments on a single day, you might change things so that the key word represented hours/minutes. Thus, typical entries could appear as:

```
1 10:00=JOHN DOE
2 11:15=BRUCE
3 12:30=LUNCH/PAM
4 14:00=AIRPORT
```

When you want to recall what you had scheduled for a particular time, you would use the “find” mode!

2. Keeping track of sales prospects. Suppose you are a salesman, and you decide that you are going to contact 20 prospects during your morning calling campaign. To do this you would load up your electronic memo pad with the names and telephone numbers of your potential clients. Typical entries might appear as:

```
1 DECKERT=877-1501
2 PECK=455-1950
```

Now you would be set to work your way down through the list in your memo pad. Each time you made a sales contact you would erase the entry from the PC. When the memo pad was back to all blanks, you would have met your calling objectives! The little PC has actually become your personal taskmaster!

There are a lot of other similar applications. You can use this program for making up shopping lists and even take it with you to do comparative shopping. You might want to use it on the job

Program listing: MEMO PAD

```
600: INPUT "CLEAR A
      LL (Y/N)? ";A$
      : IF A$="Y"
        GOSUB 750
610: A$=" "; INPUT "
      EDIT OR FIND (
      E/F)? ";A$; IF
      A$="E" THEN 640
620: IF A$="F" THEN
      690
630: GOTO 610
640: INPUT "KEY #?
      ";A: IF (A)=1)*
      (A<=20)LET C=2
      0+A: GOTO 660
650: GOTO 640
660: PRINT A;" ";A$
      (A);"=";A$(C);
      " C/R": INPUT
      "NEW KEY? ";A$
      :A$(A)=A$
670: INPUT "NEW DAT
      A? ";A$:A$(C)=
      A$
      680: GOTO 610
690: INPUT "KEY WOR
      D? ";A$
700: FOR D=1 TO 20: C
      =20+D: PAUSE D;
      " ";A$(D);"=";
      A$(C): IF A$(D)
      =A$ THEN 730
710: NEXT D: PRINT "
      NOT FOUND...
      C/R"
720: GOTO 610
730: BEEP 1: PRINT D
      ;" ";A$(D);"="
      ;A$(C);" C/R"
740: GOTO 610
750: CLEAR : DIM A$(
      40)*10: FOR A=1
      TO 20: A$(A)="
      ": A$(20+A)=" "
      : NEXT A: RETURN
STATUS 1 508
```


for recording data such as instrument readings, time accounting, or whatever else needs to be recorded for later recall.

I'll bet you can find all kinds of uses for your electronic memo pad. Can you think of ways to enhance the program? If so, try them out! You can always go back to the original version if things don't work out as planned. Remember, if at first you don't succeed, try and try again.

5.5 WEATHER FORECASTER

Predict the weather with a tiny pocket computer? Is it actually possible? Sure—but with a few words of caution. There is no law that says weather predictions will always turn out to be absolutely correct!

Actually, there really isn't any magic involved. This program, prepared by Emerich Auersbacher, is designed to predict local short-term weather (12 to 24 hours) for any region in the Northern Hemisphere above the Tropic of Cancer. It is based on meteorological rules of thumb. To obtain the highest possible accuracy, it is best to use the program at frequent intervals. Atmospheric conditions are constantly changing and trends or tendencies are often more significant than isolated forecasts.

To use the program, load it into your PC and start it up by putting the computer into its normal program execution (RUN) mode.

The process of making a weather prediction is started by obtaining the local barometric pressure reading in inches. Try to interpolate readings from a home or office barometer to the nearest hundredth of an inch.

You must also be able to provide the latest barometric tendency. This is obtained by observing barometric movements over a period of several hours. For the purposes of using this program, the following definitions apply: the barometric tendency is considered to be changing fast if the pressure is varying by 0.05 or more inches per hour; it is considered to be changing slowly if its movements are in the range 0.02 to 0.05 inches per hour; it is to be considered as steady if changes are less than 0.02 inches per hour.

Finally, you must input the direction from which the wind is blowing. Note that this is the direction the wind is coming from, not the direction that it is blowing toward! Enter the wind direction in abbreviated form to the closest of the following eight major compass points: N, NE, E, SE, S, SW, W, or N.

That is all there is to it! Enter that information as prompted by the PC, and in a few seconds the PC will beep and display its prediction! Write it down. Keep records. See how accurate it is. Chances are good that you can make better forecasts than a lot of those "experts" on radio and television do!

Here is a sample run-through of the program in operation, so that you can get a first-hand feel for the procedure:

```
WEATHER FORECASTER
BAROMETER (IN): 29.85
(R)ISE, (F)ALL, (S)TEADY?F
FALLING (F)AST, (S)LOW?F
WIND FROM THE:SE
FORECAST -->
RAIN AND HIGH WINDS.
```

Whoops—I hope most of your forecasts predict better weather than that one did!

Program listing: WEATHER FORECASTER

10: BEEP 1: PAUSE "	S=4
WEATHER FORECASTER"	80: INPUT "(R)ISE,
20: A\$="S": B\$="SW"	(F)ALL, (S)TEAD
: C\$="W": D\$="NW"	Y?"; T\$: X=0
: E\$="N": F\$="N"	90: IF T\$="R" INPUT
: G\$="E": H\$="SE"	"RISING (F)AST
30: INPUT "BAROMETER (IN): "; P: IF	, (S)LOW?"; R\$:
P>30.2 LET S=0	IF R\$="F" LET X
40: IF P<30.2 LET S=1	=1: GOTO 150
50: IF P<30.1 LET S=2	100: IF T\$="R" LET X
60: IF P<30.0 LET S=3	=2: GOTO 150
70: IF P<=29.8 LET	110: IF T\$="F" INPUT
	"FALLING (F)AST
	T, (S)LOW?"; R\$:
	IF R\$="F" LET X
	=3: GOTO 150
	120: IF T\$="F" LET X
	=4: GOTO 150

```

130: IF T$="S"GOTO
    150
140: GOTO 80
150: INPUT "WIND FR
    OM THE: ";W$:
    GOSUB 200
160: IF Y=0BEEP 3:
    PAUSE "ERROR":
    GOTO 150
170: GOTO 300
200: Y=0: FOR Z=1 TO
    8
210: IF W$=Q$(Z)LET
    W=Z: Z=8: Y=1:
    GOTO 220
220: NEXT Z: RETURN
300: T=600: Y=ABS (W
    -3)<2
310: IF Y*(S<3)*(X=
    0)LET Z=1: GOTO
    T
320: IF Y*(S=1)*(X=
    1)LET Z=6: GOTO
    T
330: IF Y*(S=0)*(X=
    4)LET Z=3: GOTO
    T
340: Y=(W=1)+(W=8)
350: IF Y*(S=1)*(X=
    4)LET Z=4: GOTO
    T
360: IF Y*(S=1)*(X=
    3)LET Z=5: GOTO
    T
370: Y=ABS (W-7)<2
380: IF Y*(S=1)*(X=
    4)LET Z=4: GOTO
    T
390: IF Y*(S=1)*(X=
    3)LET Z=5: GOTO
    T
400: IF Y*(S>2)*(X=
    4)LET Z=6: GOTO
    T
410: IF Y*(S>2)*(X=
    3)LET Z=7: GOTO
    T
420: Y=(W=6)+(W=7)
430: IF Y*(S<2)*(X=
    4)LET Z=8: GOTO
    T
440: IF Y*(S<2)*(X=
    3)LET Z=9: GOTO
    T
450: IF ((W=1)+(W=2
    ))*(S>2)*(X=2)
    LET Z=10: GOTO
    T
460: IF ((W<2)+(W>6
    ))*(S=4)*(X=3)
    LET Z=11: GOTO
    T
470: IF (ABS (W-6)<
    2)*(S=4)*(X=3)
    LET Z=12: GOTO
    T
480: IF (ABS (W-3)<
    2)*(S=4)*(X=1)
    LET Z=13: GOTO
    T
490: IF (X<3)*(ABS
    (W-5)<3)LET Z=
    2: GOTO T
500: Z=1: IF X=3LET
    Z=8
600: BEEP 3: PAUSE "
    FORECAST -->"
    :GOSUB T+10*Z:
    GOTO 30
610: PRINT "FAIR, L
    ITTLE CHANGE."
    :RETURN
620: PRINT "FAIR, A
    LITTLE COOLER
    .":RETURN
630: PRINT "FAIR, A
    LITTLE WARMER
    .":RETURN
640: PRINT "RAIN WI
    THIN 24 HRS.":
    RETURN
650: PRINT "WINDY,
    RAIN BY 12 HRS
    .":RETURN
660: PRINT "INCREAS
    ING CLOUDS, RA
    IN.":RETURN
670: PRINT "RAIN AN
    D HIGH WINDS."
    :RETURN
680: PRINT "UNSETTL
    ED, CHANCE RAI

```

```

N.":RETURN
690:PRINT "RAIN/SN
OW, WINDY.":
RETURN
700:PRINT "CLEARIN
G, BECOMING FA
IR.":RETURN
710:PRINT "SEVERE
STORM IMMINENT

```

```

.":RETURN
720:PRINT "HEAVY R
AIN/SNOW, WIND
S.":RET
730:PRINT "CLEARIN
G AND COOLER."
:RETURN
STATUS 1      !608

```

5.6 WIND CHILL

This program, by Emerich Auersbacher, calculates the wind chill equivalent temperature, which is the temperature value a person actually feels when exposed to various combinations of temperature and wind velocity. If you ski or have to be outside in cold weather for any other reason, this program can give you an indication of how much warm clothing you will need to wear.

The program is based upon heat loss equations originally developed by Siple and Passel from wind chill experiments performed at the South Pole icecap in 1941. The equation takes into account heat loss from radiation, conduction, and convection, assuming an average skin temperature of 91.4° F. 27.5°C

To use the program, load it into memory, and then place the PC in its program execution (RUN) mode. Start the program with a *RUN* command, and respond to the prompts.

All you need to input is the current temperature and your estimate of the present wind velocity (use the Wind Speed Table for the latter).

WIND SPEED TABLE

WIND	VELOCITY (MPH)	PHYSICAL INDICATORS
Calm	<1	Smoke rises straight up.
Light	1-3	Smoke drifts, wind vanes still.
Slight	4-7	Felt on face, leaves rustle.
Gentle	8-12	Leaves, twigs in constant motion.
Moderate	13-18	Dust, paper raised. Small branches sway.
Fresh	19-24	Small trees sway, wavelets form on water.
Strong	25-31	Large branches sway, phone lines whistle.
High	32-38	Large trees in motion, hard to walk.
Gale	39-46	Twigs break off trees.

The program includes a frostbite alarm or alert feature. An appropriate cautionary message is issued when the wind chill equivalent temperature falls below -10° F. (alert) or -20 (warning).

Here is a sample run of the program that will give you values you can use to ascertain that your copy of the program has been properly loaded into your PC:

WIND CHILL FACTOR

WIND SPEED (MPH): 35

TEMPERATURE (F): 19

BEEP *BEEP* *BEEP* *BEEP* *BEEP* (series of 5 beeps)

DANGER: FROSTBITE ALERT (warning flashed 4 times)

DANGER: FROSTBITE ALERT

DANGER: FROSTBITE ALERT

DANGER: FROSTBITE ALERT

WIND CF= -21.5F (-29.7C)

You can get the program to cycle back for another series of inputs by pressing the ENTER key at the conclusion of a program run.

Program listing: WIND CHILL

```

5: BEEP 1: PAUSE "
  WIND CHILL FAC
  TOR"
10: INPUT "WIND SP
  EED (MPH): "; U
  U = .44704 * U
15: IF U < 1.78816
  LET U = 1.78816
20: INPUT "TEMPERA
  TURE (F): "; T
  T = 5/9 * (T - 32)
30: A = 10.45: B = 10: C
  = -1
40: H = (A + B * J * U + C * U)
  * (33 - T)
50: T = 33 - (H / 22.034
  ): T = 9/5 * T + 32
60: U = 5/9 * (T - 32): U
  = INT (10 * U + .5)
  / 10: T = INT (10 *
  T + .5) / 10
70: IF T < -19: BEEP 5
  : FOR Z = 1 TO 4:
  PAUSE "DANGER:
  FROSTBITE ALE
  RT": PAUSE " ";
  NEXT Z: GOTO 90
80: IF T < -10: BEEP 3
  : FOR Z = 1 TO 4:
  PAUSE " FROST
  BITE WARNING":
  PAUSE " "; NEXT
  Z
90: PRINT "WIND CF
  = "; T; "F ("; U;
  "C)"
100: GOTO 10
STATUS 1 424

```


5.7 TEMPERATURE/HUMIDITY INDEX

Just as excessive cold can be dangerous in the winter, too much heat and humidity can be unhealthy in the summer. As Emerich Auersbacher demonstrates with this program, your PC can help you guard against this condition, too.

To use this program you will need to have a direct reading hygrometer. This instrument is commonly available and is sometimes mounted along with a thermometer on a board. It indicates the moisture in the air as a percentage of the total amount of moisture required to reach saturation. You will also need to have a thermometer calibrated in degrees Fahrenheit.

Load the program into your PC, and place it in the program execution (RUN) mode. Start the program by issuing a *RUN* command. Respond to the prompts for temperature and humidity. The program will respond with an alert or warning if the temperature/humidity index (THI) is calculated to be at a possibly dangerous level. The program will then provide the calculated THI and the comfort (or discomfort) level to be expected. Pressing the ENTER key after a program run allows you to input another set of values.

Here is a sample run for checking purposes:

```
TEMP-HUMIDITY INDEX
TEMPERATURE (F): 92
HUMIDITY (%): 63
*BEEP* *BEEP* *BEEP*
HEAT STRESS ALERT
HEAT STRESS ALERT
HEAT STRESS ALERT
HEAT STRESS ALERT
T.H. INDEX= 85F ( 29C)
** ACUTE DISCOMFORT **
```

(Series of 3 beeps)
(Warning flashed 4 times)

*Index better to give ;
Please enter to next time.*

In addition to warning you when THI conditions become dangerous, this program can help you increase the comfort of your home or business. During the winter, you can save money on fuel bills by turning down the thermostat while using a humidifier to increase the humidity. During the summer you can feel just as comfortable at a slightly higher temperature if you

decrease the humidity by using a dehumidifier. Try running this program with various values on a trial-and-error basis until you find a combination of temperature and humidity that provides comfort while saving money.

Here are a few other rules of thumb relating to THI values:

1. Air conditioning may be required if the THI exceeds 73; about half the population feels uncomfortable when the THI exceeds 75.
2. Mistakes and accidents increase as the THI rises into the 80s.
3. Many kinds of animals may suffer or be in danger of dying at values of 84 and up; THI values above 90 are dangerous for most people.

By executing this program your PC can help you stay comfortable.

Program listing: TEMPERATURE/HUMIDITY INDEX

```

18 10: BEEP 1: PAUSE "          PAUSE " ": NEXT
    TEMP-HUMIDITY          Z
    INDEX"
20 20: INPUT "TEMPERA          32 70: PRINT "T.H. IN
    TURE (F): "; T          DEX= "; I; "F ("
30 30: INPUT "HUMIDIT          34 80: IF I>84 PRINT "
    Y (%): "; H: H=H          ** ACUTE DISCO
    /100                      MFORT **": GOTO
40 40: I=T-(.556-.556          20—20
    *H)*(T-58.1)            26 90: IF I>79 PRINT "
50 50: T=5/9*(I-32): I          ** DISCOMFORT
    =INT I: T=INT T          **": GOTO 20
28 60: IF I>=90 BEEP 5          38 100: IF I>75 PRINT "
    :FOR Z=1 TO 4.          ** SOME DISCOM
    PAUSE "DANGER:          FORT **": GOTO
    HEATSTROKE AL          20
    ERT": PAUSE " "          40 110: IF I>60 PRINT "
    :NEXT Z: GOTO 7.          ** COMFORT ZON
    0 32                      E **": GOTO 20
30 65: IF I>=84 BEEP 3          42 120: PRINT "** FEEL
    :FOR Z=1 TO 4:          S COOL **"
    PAUSE "HEAT ST          44 130: GOTO 20
    RESS WARNING":          STATUS 1          494

```

with new numbering?

5.8 MORSE CODE

Radio amateur operators can bone up on their Morse code skills by using this program. It generates audible Morse code signals at speeds of about two to seven or eight 5-character ciphers per minute as selected by the user. You can also vary the tone or pitch over a modest range to suit your eardrums.

Practicing the receipt of Morse code that is sent as cipher groups is just about the best method of training known. This is because you cannot guess or deduce characters as is the case when receiving plain text messages. While it is the best method, it is also the most difficult. Once you learn to copy cipher at a given speed, however, you are likely to find that copying ordinary text sent at the same rate seems amazingly easy.

Of course, it would not do much good to practice copying code if you could not check your work. This program has a feature that enables you to do that. Each time it sends a block of code it keeps track of the characters until it has stored a block of 200. Then it stops so that you may review what you wrote down against what it stored in its memory.

If you do not know Morse code well enough to start writing down characters as you hear them, then watch the liquid-crystal display of the PC as you listen. The program briefly flashes the character on the PC screen as it is sent. By watching the screen and listening at the same time, you will be able to learn Morse code!

Using the program is a lot easier than learning Morse code. Once the program has been loaded into memory, put the PC into the RUN mode. You may start the program by executing a *RUN* command or by pressing the *DEFine* key and the letter *C* (for code). When this is done, the program will prompt for the tone and the speed at which code is to be sent. Each of these parameters must be specified by a single digit in the range of 1 to 5 (low tones to high tones and slow speed to high speed). To get started try something in the middle, such as 3. You can then experiment with other settings until you find a tone and pace that is comfortable (or instructive). Once these parameters have been specified, the PC will start sending code. There is a longer

than normal space after each five characters, which separates the code into 5-character blocks or ciphers.

After 200 characters have been sent, the program will stop and display the message:

USE DEF/L TO VERIFY

If you have been practicing receiving the code without looking at the display, you can now check your work by pressing the DEFine key and the letter L for list. Doing this will cause the program to display the 200 characters just sent. It does this in groups of five characters, with four such groups to a line. After each line has been displayed, the PC waits for you to press the ENTER key before outputting the next line. After all 200 characters have been displayed, pressing the ENTER key will cause the program to start over (again prompting for tone and speed) so that you may start another practice session.

If you do not want to bother checking what has just been sent, you can resume your Morse code receiving practice by pressing the DEFine key and the letter C. That will take you back to the start of the program.

By the way, if you want to restrict the program so that it sends only letters of the alphabet (sans numbers and punctuation), change line 70 to read:

70 R=RND 26-1

On the other hand, if you would like to practice receiving just digits, then change line 70 to read:

70 R=25+RND 10

To restore the program to being able to send all the characters as originally provided, line 70 must read:

70 R=RND 40-1

If you are the ambitious type, you may want to consider adding more punctuation codes to the program. Studying lines

200 to 290 should give you a pretty good idea of how this may be done.

Learning the Morse code can be a lot of fun. It can open up a whole new world to radio amateurs. Of course, it can also be somewhat frustrating, especially at the beginning. After a week or so of dedicated practice with this program you will probably either be nuts or you will know the Morse code!

Program listing: MORSE CODE

```

10:"C"CLEAR :DIM
   A$(39),B(5),C$
   (199)*1:GOSUB
   200:GOTO 20
20:INPUT "TONE? "
   ;T:T=ABS (INT
   (T)):IF (T<1)+
   (T>5)THEN BEEP
   1:GOTO 20
30:T=11-(T+3)
40:INPUT "SPEED?
   ";S:S=ABS (INT
   (S)):IF (S<1)+
   (S>5)THEN BEEP
   1:GOTO 40
50:P=(11-(S+3))/4
   ;S=((10/S)^(T/
   48))*300/S:T=T
   *10:Q=3*S
60:CLS :L=0:
   RANDOM
70:R=RND 40-1
80:WAIT 0:CURSOR
   12:C$(L)=LEFT$
   (A$(R),1):
   PRINT C$(L)
90:C=VAL (MID$ (A
   $(R),2,1))
100:FOR J=3TO C+2:
   Q(J+18)=VAL (
   MID$ (A$(R),J,
   1)):NEXT J
110:FOR J=21TO 20+
   C
120:IF Q(J)=1THEN
   140
130:BEEP 1,T,S:
   WAIT P:PRINT "
   ":GOTO 150
140:BEEP 1,T,Q:
   WAIT P:PRINT "
   "
150:NEXT J:WAIT P*
   20:PRINT "":L=
   L+1:IF (L)/5=
   INT ((L)/5)
   THEN WAIT P*60
   :PRINT ""
160:IF L=200THEN
   CLS :WAIT :
   PRINT " USE
   DEF/L TO VERIF
   Y"
170:GOTO 70
200:A$(0)="A201":A
   $(1)="B41000":
   A$(2)="C41010"
   :A$(3)="D3100"
   :A$(4)="E10"
210:A$(5)="F40010"
   :A$(6)="G3110"
   :A$(7)="H40000
   ":A$(8)="I200"
   :A$(9)="J40111
   "
220:A$(10)="K3101"
   :A$(11)="L4010
   0":A$(12)="M21
   1":A$(13)="N21
   0":A$(14)="O31
   11"
230:A$(15)="P40110
   ":A$(16)="Q411
   01":A$(17)="R3

```

```

010":A$(18)="S
3000":A$(19)="
T11"
240:A$(20)="U3001"
:A$(21)="U4000
1":A$(22)="W30
11":A$(23)="X4
1001"
250:A$(24)="Y41011
":A$(25)="Z411
00":A$(26)="15
01111":A$(27)="
"2500111"
260:A$(28)="350001
1":A$(29)="450
0001":A$(30)="
5500000":A$(31
)="6510000"
270:A$(32)="751100
0":A$(33)="851
1100":A$(34)="
9511110":A$(35
)="0511111"
280:A$(36)=" .60101
01":A$(37)=" ,6
110011":A$(38)
="76001100":A$
(39)="/510010"
290:RETURN
300:"L"WAIT 0:CLS
310:FOR I=0TO L-1
320:PRINT C$(I);
IF (I+1)/5=INT
((I+1)/5)THEN
PRINT " ";
330:IF (I+1)/20=
INT ((I+1)/20)
THEN WAIT :
PRINT " ";WAIT
0
340:NEXT I
350:GOTO 10
STATUS 1 1319

```

5.9 SIMPLE INTEREST

Ken Slaughter designed this program with a nicely formatted output. It calculates simple interest on noncompounding investments such as six-month treasury bills. The program then uses the PC's printer/plotter unit to list the monthly interest as it accrues. It then sums the total interest earned over a stated income period and provides that value as a grand total.

To use the program, load it into memory, and then place the PC in the RUN mode. The program may be started by typing RUN and pressing the ENTER key. Alternatively, pressing the DEFine key and then the SPACE bar (key) will also put the program into operation.

Respond to the prompts for the amount of principal, the interest rate (expressed as an annual percentage), the number of weeks of data to be calculated, and the date the investment starts to earn income. The date is asked for in three parts: the month (expressed numerically in one or two digits, such as 1 for January, 10 for October, etc.), the day of the month (1 to 31), and the year (in four digits, such as 1983).

The program will then summarize the original input criteria on the printer. It will follow this with a list of the daily accumulation of interest for each month over the projection period. It concludes by providing a grand total of interest earned over the specified period. A sample output produced by the program is provided for testing purposes.

If you are interested in enhancing this program, you might want to try making a second version that would compound interest on a daily basis.

Start saving your money. You can use the interest earnings to help finance the purchase of your next PC. At the rate the capabilities of these little marvels is increasing, having the power of one in hand may soon be worth more, in terms of personal earning power, than a modest amount of hard cash earning interest in a bank!

Program listing: SIMPLE INTEREST

```

10: " "CLEAR :
    USING "#####
    #.##":PAUSE "
        SIMPLE IN
    TEREST"
20: INPUT "PRINCIP
    AL? ";P
25: INPUT "% INTER
    EST? ";I
26: INPUT "# OF WE
    EKS? ";W
30: INPUT "MONTH?
    ";M
35: INPUT "DAY? ";
    D
40: INPUT "YEAR? "
    ;Y
45: CSIZE 2: COLOR
    1
50: LPRINT "      IN
    VESTMENT"
52: LPRINT "
    INCOME"
54: LPRINT "      PR
    OJECTION"
56: LPRINT " "
57: LPRINT ">SIMPL
        E INTEREST"
58: LPRINT "DATA: "
59: LPRINT "PRINCI
    PAL= ",P
60: LPRINT "INTERE
    ST= ",I
65: LPRINT " ";
    USING :LPRINT
    "DATE: ";M;D;Y
70: LPRINT " "
90: LPRINT "DATE
        AMOUNT"
100: I=I/100/360
110: L=7*W: A=P*I
115: GOSUB 305
120: FOR Z=1TO L
121: T=T+A: G=G+A
122: IF (M>9)*(D>9)
    GOTO 132
123: IF (M>9)+(D>9)
    GOTO 130
125: USING :LPRINT
    M;D;USING "###
    #####.##";T
    :GOTO 134
130: USING :LPRINT
    M;D;USING "###

```

```

#####.##";T;          +1
GOTO 134                301:LPRINT " "
132:USING :LPRINT        304:T=0
M;D;USING "###          305:GOTO 305+M
#####.##";T;          306:E=31:RETURN
134:D=D+1:IF D>E        307:E=28:RETURN
GOSUB 300                308:E=31:RET
135:NEXT Z               309:E=30:RET
140:LPRINT " ";H$=       310:E=31:RET
"*****"                311:E=30:RETURN
142:GOSUB 400            312:E=31:RETURN
145:LPRINT "GRAND        313:E=31:RETURN
TOTAL=",G               314:E=30:RETURN
150:GOSUB 400            315:E=31:RETURN
155:FOR Z=1TO 3:         316:;E=30:RET
LPRINT " ";              317:E=31:RETURN
NEXT Z                   400:LPRINT H$;"
156:END                  ";H$:RETURN
300:M=M+1:D=1:IF M       STATUS 1          942
=13LET M=1:Y=Y

```

Sample output

INVESTMENT INCOME PROJECTION

>SIMPLE INTEREST

DATA:

PRINCIPAL=

1250.00

INTEREST=

6.75

DATE: 9 28 1983

DATE	AMOUNT
9 28	0.23
9 29	0.46
9 30	0.70
10 1	0.23
10 2	0.46
10 3	0.70

■ PERSONAL PROGRAMS ■

10 4	0.93
10 5	1.17
10 6	1.40
10 7	1.64
10 8	1.87
10 9	2.10
10 10	2.34
10 11	2.57
10 12	2.81
10 13	3.04
10 14	3.28
10 15	3.51
10 16	3.75
10 17	3.98
10 18	4.21

***** *****
GRAND TOTAL=
4.92
***** *****

Programs for Entertainment

6.1 BLACKJACK

Blackjack is an extremely popular card game that is readily computerized, even on a PC. One nice thing about it is that it is a fast game that is conducive to play at any time and can be played for just a few moments to an hour or longer. Planning on hitting a casino in the near future? Perhaps you can sharpen your Blackjack skills through the use of this program on your PC?

The rules and the object of Blackjack are pretty simple. You and the dealer are initially dealt two cards apiece. One of the dealer's cards is kept hidden until you are through playing. The object is for you to come as close as possible to a point-count of 21 with the cards in your hand, *without going over 21*. Face cards count as 10 points, aces count as 1 or 11 (your choice), and other cards count as their face value. If you go over 21 points, you "bust" and lose automatically. You can draw cards (take "hits") until you reach 21 points or go over. You can stop at any time, with 21 points or less, after which the dealer's "hole" card will be shown. The dealer then draws cards (done automatically by the PC program) to try to tie or beat your score without going over 21, in which case the dealer wins. Of course you win, if the dealer goes over 21. This program actually gives a considerable break to players by counting dealer aces as 1 rather than 11 (in all but the Blackjack situation). This kind of makes up for the relentless accuracy of the PC's play. Watch out, though, because you can still lose your bankroll pretty fast!

There are a few other fine points of play in this version of Blackjack. First, of course, a "natural" 21 dealt to the player pays

1.5 to 1 instead of the normal 1 to 1, as long as the dealer doesn't steal the hand with a natural (2-card) 21 tie.

If the dealer's "up" card (after the first two cards have been dealt) is worth 10 points or is an ace, the player is offered the opportunity to obtain "insurance" against a natural 21 by the dealer. You lose half of your original bet if the dealer does not have 21 at that time, but protect your bet (which you would ordinarily lose) if the dealer has a Blackjack.

Third, if the initial count of your hand after the first two cards have been dealt is 10 or 11, you are given the opportunity to double your bet. If you elect this option, you can win or lose twice your original bet. However, you are only dealt one more card and must rest with the final count of just three cards.

Can you turn your initial \$100.00 bankroll into a grand? Give it a try, but watch out if you are a card "counter." This computerized dealer likes to shuffle often!

Here is what a typical hand of Brian Peterson's version of Blackjack might look like if it was presented on the display of your PC (All you have to do to put it into action is load the program, and then, with the PC in the RUN mode, type RUN.):

```
SHUFFLING....
BANK $ 100
BET= 10
DEALING....
YOU: 72          DEALER: ^4
HIT? Y
YOU: 722        DEALER: ^4
HIT? Y
YOU: 7222       DEALER: ^4
HIT? Y
YOU: 72227      DEALER: ^4
HIT? N
YOU: 72227      DEALER: 64
YOU: 72227      DEALER: 645
YOU: 72227      DEALER: 6459
DEALER BUSTS WITH 24
YOU WIN $ 10
BANK # 110
BET=
```

.

Good luck.

Program listing: BLACKJACK

```

1: DIM A(12):
  RANDOM : W=0: X=
  100: R=-1
2: BEEP 2: IF -R
  PAUSE "SHUFFLI
  NG...": FOR R=
  0 TO 12: A(R)=5:
  NEXT R: IF W=0
  GOTO 25
3: PRINT "YOU: "; A
  $; B$; C$; D$; E$;
  F$; "DEALER: "; G
  $; H$; I$; J$; K$;
  L$; M$: RETURN
4: W=1: N=(RND 13)
  -1: A(N)=A(N)-1
  : IF A(N) LET R=
  R-1: GOTO N+6
5: GOTO 4
6: Q$(P)="A": U=1:
  O=10: RETURN
7: Q$(P)="2": U=2:
  RETURN
8: Q$(P)="3": U=3:
  RETURN
9: Q$(P)="4": U=4:
  RETURN
10: Q$(P)="5": U=5:
  RETURN
11: Q$(P)="6": U=6:
  RETURN
12: Q$(P)="7": U=7:
  RETURN
13: Q$(P)="8": U=8:
  RETURN
14: Q$(P)="9": U=9:
  RETURN
15: Q$(P)="T": U=10
  : RETURN
16: Q$(P)="J": U=10
  : RETURN
17: Q$(P)="Q": U=10
  : RETURN
18: Q$(P)="K": U=10
  : RETURN
19: INPUT "INSURAN
  CE? "; N$: IF N$
  ="N" RETURN
20: N=INT (.5*Y): X
  =X-N: PAUSE "YO
  U LOSE $"; N: IF
  U+O<>21 LET U=0
  : GOSUB 31:
  RETURN
21: GOSUB 2: U=0:
  RETURN
25: FOR Y=1 TO 13: Q
  $(Y)=" ": NEXT
  Y: O=0: T=0: U=0
26: BEEP 1: PAUSE "
  BANK $"; X: IF X
  INPUT "BET="; Y
  : IF Y-X GOTO 26
27: IF X<=0 INPUT "
  AGAIN? "; N$: IF
  N$="Y" LET X=10
  0: GOTO 26
28: IF X<=0 END
29: PAUSE "DEALING
  ....": P=7:
  GOSUB 4: U=U+U:
  P=8: GOSUB 4: Z=
  U: U=U+U: P=1:
  GOSUB 4: T=T+U:
  P=2
30: GOSUB 4: T=T+U:
  P=3
31: N$=G$: G$="^":
  GOSUB 2: G$=N$:
  IF U=0 RETURN
35: IF P=3 IF O=10
  IF T=11 PAUSE "
  BLACKJACK!": Y=
  1.5*Y: GOTO 63
40: IF P=3 IF (Z=1)
  +(Z=10) GOSUB 1
  9: IF U=0 GOTO 2
  5
42: IF P=3 IF O=0 IF
  (T=10)+(T=11)
  INPUT "DOUBLE?
  "; N$: IF N$="Y
  " GOSUB 4: Y=2*Y
  : T=T+U: GOTO 50

```



```

45: IF P<7 IF T<=21
    INPUT "HIT? ";
    N$: IF N$="Y"
    GOSUB 4: T=T+U:
    P=P+1: GOTO 31
48: IF T>21 PAUSE "
    YOU BUST WITH
    "; T: GOTO 65
50: GOSUB 2: O=0: P=
    8: IF G$="A" LET
    O=10
51: IF H$="A" LET O
    =10
53: N=U+O: IF (U<>6
    )+(O<>10) IF N<
    =21 IF N>=17+(P
    >8) LET U=N:
    GOTO 60
54: IF U>21 PAUSE "
    DEALER BUSTS W
    ITH "; U: GOTO 6
    3
55: IF U>=17+(P>8)
    GOTO 60
57: P=P+1: GOSUB 4:
    U=U+U: IF U=1
    LET O=10
59: GOSUB 2: GOTO 5
    4
60: PAUSE "DEALER
    STAYS WITH "; U
    : FOR N=1 TO 6:
    IF G$(N)="A"
    LET O=10: N=6
61: NEXT N: IF U>=T
    +O*((T+O)<=21)
    GOTO 65
63: PAUSE "YOU WIN
    $"; Y: X=X+Y: IF
    X<1000 GOTO 25
64: PRINT "YOU WON
    $"; X: END
65: PAUSE "YOU LOS
    E $"; Y: X=X-Y:
    GOTO 25
STATUS 1 1486

```

6.2 HANGMAN

Gary Heidbrink provides this pocket computer version of the classic old word game Hangman. This implementation is designed to be played with two people. One person thinks of a word having up to 10 letters. These are put into the PC one character at a time. This process is concluded by pressing the ENTER key without having inputted any other character. Once the word has been set up, the other player tries to discern the word in as few tries as possible by guessing the letters of which it is composed. Each time a letter is selected, the computer shows whether it appears one or more times in the word and the position(s) in which it appears. If the letter does not occur in the word at all, the player is docked by having a segment added to the dreaded "hanging man." Six wrong guesses and you are finished (the man is hanged)!

To give it a whirl yourself, load the program and execute a *RUN* command with the PC in the RUN mode. Here is a sample run of the program just to show you how the computer does its stuff:

```

-Ø->=<  HANGMAN  -Ø->=<
NEED INSTRUCTIONS? (Y/N)N
PLAY OR QUIT? (P/Q):P
PLAYER1 ENTER LETTER 1:N
NEXT LETTER:I
NEXT LETTER:C
NEXT LETTER:E
NEXT LETTER:  (ENTER alone pressed to terminate word input.)
----
PLAYER #2 GUESS LETTER:A
----
A          Ø
PLAYER #2 GUESS LETTER:I
-I--
A          Ø
PLAYER #2 GUESS LETTER:E
-I-E
A          Ø
PLAYER #2 GUESS LETTER:D
-I-E
AD        Ø-

PLAYER #2 GUESS LETTER:C
-ICE
AD        Ø-
PLAYER #2 GUESS LETTER:N
NICE
NICE / CORRECT
PLAY OR QUIT? (P/Q):P
PLAYER2 ENTER LETTER1: (Now second player enters a word,
.                        and the first player tries to guess it.)
.

```

Okay, that is how it goes. So, break out your dictionary and see if you can come up with a real stumper for that special playmate of yours!

Program listing: HANGMAN

```

1: CLEAR : DIM A$(
  40)
5: "H"A=1: G=0: H=0
  : A$(0)="": A$(1
  )="O": A$(2)="O
  -": A$(3)="O->"
10: A$(4)="O->=": A
                                $(5)="O->=<": A
                                $(6)="O->=<"
15: WAIT 100: PRINT
  " "; A$(6); "  H
  ANGMAN  "; A$(6
  )
20: WAIT : INPUT "N

```

```

EED INSTRUCTIO
NS?(Y/N)";B$:
IF B$="N"GOTO
40
25:PRINT "2 PLAYE
RS TAKE TURNS
TO"
30:PRINT "CHOOSE
AND GUESS WORD
S"
35:PRINT "MAX 10
LETTERS 6 MISS
ES"
40:INPUT "PLAY OR
QUIT? (P/Q): "
;B$: IF B$="Q"
GOTO 185
45:FOR C=11TO 36:
A$(C)="":NEXT
C:F=0
50:IF A=1INPUT "P
LAYER1 ENTER L
ETTER 1: ";A$(1
1):A$(21)="-":
GOTO 65
55:IF A=2INPUT "P
LAYER2 ENTER L
ETTER1: ";A$(11
):A$(21)="-":
GOTO 65
60:GOTO 50
65:FOR C=12TO 20:
IF A$(C-1)=""
LET C=20:GOTO
80
70:INPUT "NEXT LE
TTER: ";A$(C):A
$(C+10)="-"
80:NEXT C:GOTO 13
0
85:IF A=1INPUT "P
LAYER #2 GUESS
LETTER: ";D$:
GOTO 100
90:IF A=2INPUT "P
LAYER #1 GUESS
LETTER: ";D$:
GOTO 100
95:GOTO 85
100:FOR C=11TO 20
105:IF A$(C)=D$LET
A$(C+10)=D$:E=
1
110:NEXT C
115:IF E=1LET E=0:
GOTO 160
120:FOR C=31TO 36:
IF A$(C)=""LET
A$(C)=D$:F=C-3
0:C=36:NEXT C:
GOTO 130
125:NEXT C
130:IF A$(36)<>""
GOTO 135
132:PRINT A$(21);A
$(22);A$(23);A
$(24);A$(25);A
$(26);A$(27);A
$(28);A$(29);A
$(30)
135:IF A$(F)="-O->
=<"GOTO 145
137:IF A$(31)=""
GOTO 85
140:PRINT A$(31);A
$(32);A$(33);A
$(34);A$(35);"
";A$(F)
:GOTO 85
145:PRINT A$(11);A
$(12);A$(13);A
$(14);A$(15);A
$(16);A$(17);A
$(18);A$(19);A
$(20);
150:PRINT " ";A$(
06);" HUNG":
IF A=1LET A=2:
G=G+1:GOTO 40
155:A=1:H=H+1:GOTO
40
160:FOR C=11TO 20:
IF A$(C)=A$(C+
10)NEXT C:GOTO
170
165:C=20:NEXT C:
GOTO 130
170:PRINT A$(11);A
$(12);A$(13);A
$(14);A$(15);A
$(16);A$(17);A
$(18);A$(19);A

```

```
$(20);  
175:PRINT " / CORR  
ECT":IF A=1LET  
A=2:H=H+1:GOTO  
40  
180:A=1:G=G+1:GOTO
```

```
40  
185:PRINT "PLAYER  
#1:";G;" PLAYE  
R #2:";H:GOTO  
.1  
STATUS 1 1404
```

6.3 SUPER-WUMPUS

Gary Heidbrink, who submitted this entertaining PC program, says that the computer game Wumpus was originally developed in 1973 by Gregory Yob. The game quickly caught on and became one of the early personal computer classics.

The objective of the game is to search through a complex of caves containing twenty caverns that are connected by dark tunnels. You are seeking to locate and destroy the ferocious Wumpus creature. Two bats also occupy the cave complex. The bats are huge and are easily capable of carrying a person. Indeed, should you enter a cavern containing a bat, you will suddenly find yourself (unwillingly) transported to another randomly selected location. Two other caverns contain bottomless pits. Alas, should you slip into such a pit, you can kiss yourself goodbye! There is only one worse fate: meeting face to face with the mighty Wumpus. Don't even think about it!

When you are in a cavern adjacent to one containing a hazard you will receive a warning such as:

```
YOU FEEL A DRAFT.  
YOU HEAR WINGS.  
YOU SMELL A WUMPUS!
```

If you do receive one of these messages, take heed!

Remember, though, that what you really want to do is destroy the Wumpus before meeting the creature in a cavern. To do this, you use arrows that have the capability of darting through one to five caves in a single shot. Find the Wumpus. Give it a taste of your arrow. You will be a winner!

In case you think that an interesting and challenging game of Wumpus is next to impossible to achieve on a tiny pocket computer, be prepared to reconsider, for this version of the game has several variations to pique your interest. First of all, you can start the game with one to four arrows, depending on the skill level (there are twenty possible) selected. Furthermore, in all but the most difficult level there is a chance that you will discover new arrows as you venture through the caverns.

Then, there are the earthquakes! At game level three and up, there is the possibility that earthquakes will alter the locations of those nasty pits. The warning message:

I FEEL AN EARTHQUAKE!

indicates that pits have been randomly relocated.

You must be careful with your arrows. Should you try to shoot an arrow into a cavern that is not within range or that passes through your present location, guess what? You will shoot yourself! Tsk, tsk, what a poor way to go.

An arrow will only destroy the Wumpus if it is in the final cave of the arrow's flight. If the arrow merely passes through a cave in which the creature is residing, chances are good that the Wumpus will get suspicious and move to another cave. It might just move into your present cave (if you are adjacent to it) and eat you alive! Be cautious when approaching the Wumpus. It is advisable to stay well out of its range.

Refer to the accompanying cave diagram (Fig. 6.3) to see how the caves are interconnected. You may want to keep track of the progress of a game by drawing your own movements as you proceed on a mission.

Also provided here is a table showing the chances of finding another arrow and the likelihood of earthquakes occurring at the various skill levels (see Table of Game Factor Probabilities).

To play the game, load the program into your PC and place it in the normal RUN mode. You may then merely execute the *RUN* command or select the DEF/S (press the DEFine key and

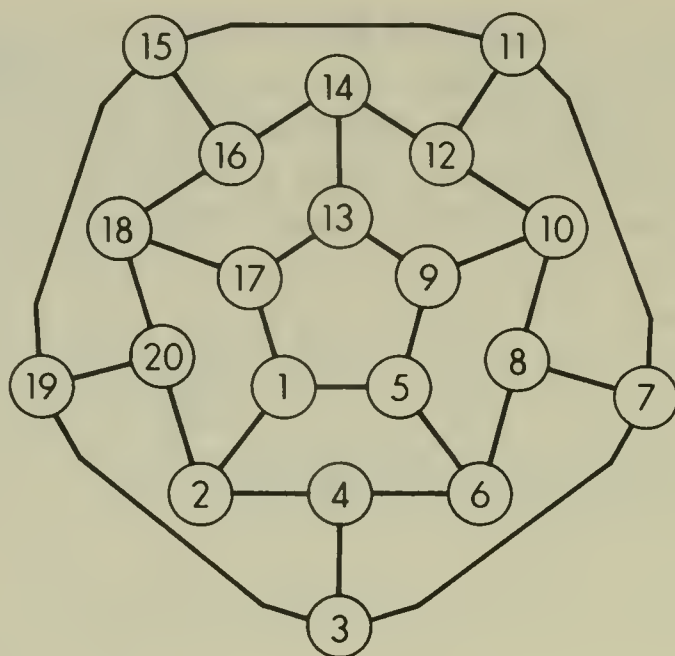


Fig. 6.3 Cave location map.

TABLE OF GAME FACTOR PROBABILITIES

LEVEL OF PLAY	STARTING ARROWS	CHANCE TO FIND ARROW	CHANCE OF QUAKE
1	4	95%	—
2	4	90%	—
3	4	85%	05%
4	4	80%	05%
5	4	75%	10%
6	3	70%	10%
7	3	65%	15%
8	3	60%	15%
9	3	55%	20%
10	3	50%	20%
11	3	45%	25%
12	2	40%	25%
13	2	35%	30%
14	2	30%	30%
15	2	25%	35%
16	2	20%	35%
17	2	15%	40%
18	1	10%	40%
19	1	05%	45%
20	1	—	45%

then the letter S) option. After announcing the title, the program prompts for the desired skill level. Start at level one until you get the idea and a few wins under your belt.

Here is how a game might start off:

```

****  SUPER-WUMPUS  ****
LEVEL (1-20) 1
YOU ARE IN ROOM 10
YOU FEEL A DRAFT.
TUNNELS GO TO:  9 12  8
MOVE OR SHOOT? (M/S): M
ENTERING WHICH CAVE? 12
YOU ARE IN ROOM 12
TUNNELS GO TO:  11 14 10
MOVE OR SHOOT? (M/S):
.
.
.

```

You can take it from there

Program listing: WUMPUS

```

1:RANDOM :GOTO "
  S"
5:H=(RND 21)-1:
  RETURN
10:X=W:Y=W:Z=W:IF
  W/2<>INT (W/2)
  LET X=X+3:
  GOSUB 15
15:X=X-1:IF X<1
  LET X=X+20
20:Y=Y+2:IF Y>20
  LET Y=Y-20
25:Z=Z-2:IF Z<1
  LET Z=Z+20
30:RETURN
40:"S"WAIT 100:
  PRINT " ****
  SUPER-WUMPUS
  ****":WAIT
45:U=6:INPUT "LEV
  EL (1-20)";U:
  IF (U<1)+(U>20
  )GOTO 45
50:J=0:K=0:I=4-
  INT (U/6):
  GOSUB 5:M=H
55:GOSUB 5:N=H:IF
  N=MGOTO 55
60:GOSUB 5:O=H:IF
  (O=M)+(O=N)
  GOTO 60
65:GOSUB 5:P=H:IF
  (P=M)+(P=N)+(P
  =O)GOTO 65
70:GOSUB 5:Q=H:IF
  (Q=M)+(Q=N)+(Q
  =O)+(Q=P)GOTO
  70
75:GOSUB 5:W=H:IF
  (W=M)+(W=N)+(W
  =O)+(W=P)+(W=Q
  )GOTO 75
80:BEEP 1:PAUSE "
  YOU ARE IN ROO
  M ";W
85:GOSUB 5:IF (I=
  0)IF (H>U)LET
  I=1:BEEP 1:

```

```

    PAUSE "YOU FOU
    ND AN ARROW."
90: IF H<V/2BEEP 2
    :PAUSE "YOU FE
    EL AN EARTHQUA
    KE!":GOSUB 5:R
    =N:N=H:IF F/2=
    INT (F/2)LET O
    =R
95: IF W=MBEEP 4:
    PRINT "WUMPUS
    ATE YOU.":GOTO
    40
100: IF (W=N)+(W=O)
    BEEP 4:PRINT "
    YOU FELL INTO
    A PIT.":GOTO 4
    0
105: IF KGOTO 120
110: IF W=PLET P=Q:
    O=W
115: IF W=QBEEP 2:
    PAUSE "BAT GRA
    BBED YOU.":K=1
    :GOSUB 5:Q=H:W
    =H:GOTO 80
120: GOSUB 10
125: IF (X=M)+(Y=M)
    +(Z=M)BEEP 3:
    PAUSE "YOU SME
    LL A WUMPUS!"
130: IF (X=N)+(Y=N)
    +(Z=N)+(X=O)+(
    Y=O)+(Z=O)BEEP
    1:PAUSE "YOU F
    EEL A DRAFT."
135: IF (X=P)+(Y=P)
    +(Z=P)+(X=Q)+(
    Y=Q)+(Z=Q)BEEP
    1:PAUSE "YOU H
    EAR WINGS."
140: PRINT "TUNNELS
    GO TO: ";USING
    "###";X;Y;Z
145: INPUT "MOVE OR
    SHOOT? (M/S):
    ";L$:J=J+1:IF
    L$="M"GOTO 160
150: IF L$="S"GOTO
    170
155: J=J-1:GOTO 140
160: INPUT "ENTERIN
    G WHICH CAVE?
    ";H:IF (H=X)+(
    H=Y)+(H=Z)LET
    W=H:K=0:GOTO 8
    0
165: PAUSE "YOU CAN
    NOT GO THERE."
    :GOTO 155
170: IF I=0PAUSE "S
    ORRY, NO ARROW
    S.":GOTO 155
175: INPUT "THRU HO
    W MANY CAVES?
    ";S:IF (S>5)+(
    S<1)PRINT "CAN
    SHOOT TO 1-5
    CAVES.":GOTO 1
    5
180: I=1-1:FOR T=1
    TO S:INPUT "TH
    RU WHICH CAVE?
    ";R:Q(T)=R:
    NEXT T
185: R=W:FOR T=1TO
    S:U=Q(T):IF R<
    >UIF (X=U)+(Y=
    U)+(Z=U)LET W=
    U:GOSUB 10:
    NEXT T:GOTO 19
    5
190: BEEP 4:PRINT "
    MISFIRE! SHOT
    SELF.":GOTO 40
195: IF W=MBEEP 5:
    PRINT "SHOT WU
    MPUS IN";J;" T
   URNS.":GOTO 40
200: W=M:GOSUB 10:
    GOSUB 5:T=INT
    (27-H/5):M=Q(T
    )
205: W=R:GOTO 80
STATUS 1          1546

```


6.4 ROCK, PAPER, SHEARS

Do you remember that old school yard game? We used to play it by holding one hand behind our back. Your opponent and you would simultaneously bring the playing hand around front in either a clenched fist (rock), open palm (paper), or two fingers extended (shears) option. Usually this was done on the count of three. The rules of the game were that paper (an open palm) covered rock, that shears (two fingers extended to represent scissor blades) would cut paper, and that rock (clenched fist) would break the shears. In my school the loser was subjected to the pain (torture?) of having to absorb a strike of two fingers (delivered by the winner) across the wrist part of the back of a hand. If you lost a lot of matches against an unfriendly opponent, you could end up with a pretty sore wrist!

Ah yes, this computer version is much better. The object of the game is the same. You try to outguess your opponent (the mighty pocket computer) so that the object you select (rock, paper, or shears) will beat the one randomly selected by the computer. But all the PC does is keep score. If you win, the computer won't attack you. Of course, if you lose, you could always toss the PC down a flight of stairs. But then it might not play with you any more!

Load the program into your PC, and place the PC in its usual RUN mode. Start the program by executing a *RUN* command. Here is how a typical game might proceed for a few rounds:

```
* ROCK, PAPER, SHEARS *
NEED INSTRUCTIONS?(Y/N)N
YOUR CHOICE IS?(R/P/S): R
COMP: ROCK  YOU: ROCK
TIE, TRY AGAIN
YOUR CHOICE IS?(R/P/S): R
COMP: PAPER  YOU: ROCK
COMPUTER WINS
COMP: 1      YOU: 0
PLAY OR QUIT? (P/Q):P
YOUR CHOICE IS?(R/P/S): R
COMP: SHEARS  YOU: ROCK
```

```

YOU WIN
COMP: 1      YOU: 1
PLAY OR QUIT? (P/Q):
.
.
.

```

Okay? Watch out, though; it is possible to get so involved with a game that you end up getting a sore wrist just from pushing the buttons on your PC!

Program listing: ROCK, PAPER, SHEARS

```

10: " "RANDOM :C=0
    :P=0
15: PAUSE " * ROCK
    , PAPER, SHEAR
    S *"
20: INPUT "NEED IN
    STRUCTIONS?(Y/
    N)";X$
25: IF X$="N"GOTO
    65
30: PRINT "CHOOSE
    ROCK(R) PAPER(
    P)"
35: PRINT "OR SHEA
    RS(S). THE"
40: PRINT "COMPUTE
    R WILL CHOOSE"
45: PRINT "ALSO AN
    D COMPARE THEM
    "
50: PRINT "ROCK BR
    EAKS SHEARS, "
55: PRINT "PAPER C
    OVERS ROCK, "
60: PRINT "SHEARS
    CUT PAPER. ";
    PAUSE ""
65: INPUT "YOUR CH
    OICE IS?(R/P/S
    ); :A$
70: IF A$="R"LET A
    $="ROCK"
75: IF A$="P"LET A
    $="PAPER"
80: IF A$="S"LET A
    $="SHEARS"
85: Z=RND 3:N=N+7
90: IF Z=1LET B$="
    ROCK"
95: IF Z=2LET B$="
    PAPER"
100: IF Z=3LET B$="
    SHEARS"
105: PRINT "COMP: "
    ;B$;" YOU: ";
    A$
110: IF A$=B$PAUSE
    "TIE, TRY AGAI
    N";GOTO 65
115: IF B$="ROCK"
    GOTO 155
120: IF B$="PAPER"
    GOTO 165
125: IF A$="ROCK"
    GOTO 150
130: PRINT "COMPUTE
    R WINS";C=C+1
135: PRINT "COMP: "
    ;C;" YOU: "
    ;P
140: INPUT "PLAY OR
    QUIT? (P/Q)";
    ;X$; IF X$="Q"
    PAUSE "BYE-BYE
    !";END
145: GOTO 65
150: PAUSE " YOU WI
    N";P=P+1;GOTO

```

```

135
155: IF A$="PAPER"
      GOTO 150
160: GOTO 130

165: IF A$="SHEARS"
      GOTO 150
170: GOTO 130
STATUS 1      816

```

6.5 WORD SQUARES

There is a game known as *Boggle* and David Motto, developer of this program, calls himself a "Boggle-maniac." He developed this PC program so that he could indulge in his passion anytime he had a few moments to spare.

For those of you who have not played the traditional game, here is how it operates. The game comes with an egg timer and a small translucent cube that contains sixteen dice. On each side of each die there is a letter of the alphabet (traditionally, the side with a "q" also contains a small "u"). You shake the translucent cube with all its dice, and then allow the dice to settle into an area of square depressions, thereby forming a matrix of dice. The top of each die is thus exposed revealing a random assortment of letters. You then turn over the egg timer, examine the array of exposed letters, and race to form as many words as possible in a limited time span.

You may form words by starting at any position in the matrix and traveling to an adjacent die. You may move up, down, right, left, and diagonally. You write down the letters as you go. You are not permitted, however, to backtrack to the same letter in any one word, but you may reuse letters again when you form other words. This process is repeated until the timer runs out. How many words can you come up with?

With this program, your pocket computer takes the place of the translucent cube and its dice. The PC will generate an array of letters. Indeed, it uses quite a sophisticated method that even takes into account the typical usage frequency of each letter in the alphabet.

To begin the game, load the program, place the PC into the RUN mode, and execute a RUN command. Respond to the prompt for *SIZE?* by indicating the size of the square (from a 1 by 1 to 10 by 10) in which you want to have letters arranged. The computer will then output a series of rows containing letters that

may be arranged to form the letter matrix. Here is how a sample run of the program might appear:

```

SIZE? 6
TBMVTC
EHOTIW
ETOSTS
NTEOEH
GOYTBP
EEKGAB

```

As soon as one set of letters has been outputted, the program will loop back to ask for the size of the next matrix. You can copy down a whole series of matrices, if you like, so that you have a ready supply of games. You might want to consider changing the *PRINT* statements within the program to *LPRINT* statements so that you can make hardcopies directly on your printer unit. You might even want to get fancy and draw a block of squares, and then place the letters selected into the center of each square—just as in the version that uses actual dice.

Program listing: WORD SQUARES

1: " "GOTO 29	12: @\$(P)="D":
2: Z=9821*Z+.2113	RETURN
27: Z=Z-INT Z:	13: @\$(P)="L":
RETURN	RETURN
3: @\$(P)="E":	14: @\$(P)="F":
RETURN	RETURN
4: @\$(P)="T":	15: @\$(P)="C":
RETURN	RETURN
5: @\$(P)="A":	16: @\$(P)="M":
RETURN	RETURN
6: @\$(P)="O":	17: @\$(P)="U":
RETURN	18: @\$(P)="G":
7: @\$(P)="N":	RETURN
RETURN	19: @\$(P)="Y":
8: @\$(P)="R":	RETURN
RETURN	20: @\$(P)="P":
9: @\$(P)="I":	RETURN
RETURN	21: @\$(P)="W":
10: @\$(P)="S":	RETURN
RETURN	22: @\$(P)="B":
11: @\$(P)="H":	RETURN
RETURN	23: @\$(P)="V":

```

RETURN
24: Q$(P)="K":
RETURN
25: Q$(P)="X":
RETURN
26: Q$(P)="J":
RETURN
27: Q$(P)="Q":
RETURN
28: Q$(P)="Z":
RETURN
29: CLEAR :RANDOM
:Z=RND 100
30: T=13108:U=2357
9.01:V=66104.0
26:W=73166.08:
X=89233.096:Y=
98163.155
31: INPUT "SIZE? "
:S
32:FOR Q=1TO S
33:FOR P=1TO S
34:GOSUB 2
35:O=100000*Z:L=2
0.6
36:FOR M=20TO 25:
IF O<Q(M)THEN
LET L=100*(Q(M
)-INT Q(M)):M=
25
37:NEXT M
38:GOSUB 2:L=INT
L+10*Z*(L-INT
L)
39:GOSUB 3+L
40:NEXT P
41:BEEP 1:PRINT A
$:B$:C$:D$:E$:
F$:G$:H$:I$:J$
:NEXT Q
42:GOTO 31
STATUS 1 761

```

6.6 VEGAS CRAPS

This is the game of Craps as typically played in Las Vegas. It is designed to be fast and fun for a single player against the PC as the “house.” This version allows most “right way” or so-called “bet to pass” options with the exception of the “coming out” bet. Here is a synopsis of the various bets permitted:

1. **Pass Bet:** This is the initial bet. It lets you into the game. Bet as much as you like. The bet is won if the initial roll of the dice is 7 or 11. The bet is lost if the initial roll is a 2, 3, or 12 (“craps”). Any other number becomes known as the “point” number. This is the number the player shoots for on subsequent rolls of the “bones” (dice). This number must be made prior to a 7 being rolled in order to win the original pass bet.

2. **Free Odds Bet:** A bet of any amount up to the original value of the initial “pass bet” may be made (after a point has been identified on a come-out roll). The name of this bet is derived from the fact that it is the only bet giving even odds to the player.

3. **Field Bet:** A bet of any amount that is won if a 2, 3, 4, 9, 10, 11, or 12 is rolled on the next toss of the dice. A 2 or 12 pays

double money. The remaining “field” numbers pay even money. The bet is lost if a 5, 6, 7, or 8 is rolled.

4. Big 6: A poor percentage bet that the number 6 will be rolled before a 7 out.

5. Big 8: A similar bet for the number 8.

6. Bet the Hard Way: A bet of any amount that any of the four possible “hard way” bets (4, 6, 8, or 10) will be rolled. The “hard way” bet is that the number will be rolled the hardest way—by rolling doubles—before the number is rolled some other way or before the player rolls a 7. Pay offs for the hard way bets are shown in parentheses by the program.

7. Place to Win: A bet of any amount that the number selected (4, 5, 6, 8, 9, or 10 permitted) will be rolled before a 7 comes up. The payoffs for place-to-win bets are shown in parentheses by the program.

To get the computerized dice rolling, load the program into memory, and then place the PC into its standard RUN mode. Now start the program by executing a *RUN* command or pressing the *DEFine* key followed by the letter *V* (*DEF/V*). Here is an illustration of the program in operation:

```

                VEGAS CRAPS
BET OR QUIT? (B/Q):B
PASS BET:100
  DICE ARE BEING ROLLED
  DICE ARE:   5   AND   4
POINT # IS: 9
YOU WIN $ 0
CHANGING BETS? (Y/N):Y
FREE ODDS BET:100
FIELD BET:                (No bet, so just press ENTER.)
BIG 6:                    (No bet . . .)
BIG 8:                    (No bet . . .)
BET THE HARDWAY? (Y/N):N
PLACE TO WIN? (Y/N):Y
4 (9/5):50
5 (7/5):50
6 (7/6):50
8 (7/6):50
9 (7/5):50
10 (9/5):50
  DICE ARE BEING ROLLED

```

```

DICE ARE:      5  AND  6
YOU WIN $ 0
CHANGING BETS? (Y/N):N
DICE ARE BEING ROLLED
DICE ARE:      4  AND  4
YOU WIN $ 58
CHANGING BETS? (Y/N):N
DICE ARE BEING ROLLED
DICE ARE:      6  AND  1
YOU LOSE $ 450
BET OR QUIT? (B/Q):Q
YOU LOST $ 392

```

Gary Heidbrink is the creator of this Vegas Craps game. Roll those dice, and may all your hard-way bets come easy!

Program listing: VEGAS CRAPS

```

5: "V"RANDOM
10: FOR V=1 TO 18: Q
    (V)=0: NEXT V:
    PAUSE "      U
    EGAS CRAPS"
15: IF P>0 GOTO 35
20: INPUT "BET OR
    QUIT? (B/Q): ";
    Z$: IF Z$="Q"
    GOTO 280
25: INPUT "PASS BE
    T: "; A: Q=A: GOTO
    120
30: GOTO 20
35: INPUT "CHANGIN
    G BETS? (Y/N):
    "; Z$: IF Z$="N"
    GOTO 120
40: INPUT "FREE OD
    DS BET: "; B: IF
    B>A LET B=A
45: INPUT "FIELD B
    ET: "; C
50: INPUT "BIG 6: "
    ; D
55: INPUT "BIG 8: "
    ; E
60: INPUT "BET THE
    HARDWAY? (Y/N
    ): "; Z$: IF Z$="
    N" GOTO 85
65: INPUT "4 (7/1)
    "; F
70: INPUT "6 (9/1)
    "; G
75: INPUT "8 (9/1)
    "; H
80: INPUT "10 (7/1
    ): "; I
85: INPUT "PLACE T
    O WIN? (Y/N): "
    ; Z$: IF Z$="N"
    GOTO 120
90: INPUT "4 (9/5)
    "; J
95: INPUT "5 (7/5)
    "; K
100: INPUT "6 (7/6)
    "; L
105: INPUT "8 (7/6)
    "; M
110: INPUT "9 (7/5)
    "; N
115: INPUT "10 (9/5
    ): "; O
120: PAUSE " DICE A
    RE BEING ROLLE
    D"
125: GOSUB 290: T=W:
    GOSUB 290: U=W:

```

```

      S=T+U
130: PRINT " DICE A
      RE: ";T;" AN
      D ";U
135: IF P>0GOTO 170
140: IF S=7LET Q=Q+
      A:FOR U=1TO 15
      :Q=Q-Q(U):Q(U)
      =0:NEXT U:GOTO
      240
145: IF S=11GOTO 24
      5
150: IF S=2GOTO 250
155: IF S=3GOTO 250
160: IF S=12GOTO 25
      0
165: P=13:PRINT "PO
      INT # IS: ";S:Q
      =0
170: IF T=UGOSUB 25
      5
175: IF S=2LET Q=Q+
      C*2
180: IF S=3LET Q=Q+
      C
185: IF S=4LET Q=Q-
      F+C+INT (J*9/5
      ):F=0:J=0:IF P
      =SLET Q=Q+B*2
190: IF S=5LET Q=Q-
      C+INT (K*7/5):
      K=0:IF P=SLET
      Q=Q+INT (B*3/2
      )
195: IF S=6LET Q=Q-
      C-G+D+INT (L*7
      /6):D=0:G=0:L=
      0:IF P=SLET Q=
      Q+INT (B*6/5)
200: IF S=7FOR U=1
      TO 15:Q=Q-Q(U)
      :Q(U)=0:NEXT U
      :P=0
205: IF S=8LET Q=Q-
      C-H+E+INT (M*7
      /6):E=0:H=0:M=
      0:IF P=SLET Q=
      Q+INT (B*6/5)
210: IF S=9LET Q=Q+
      C+INT (N*7/5):
      N=0:IF P=SLET
      Q=Q+INT (B*3/2
      )
215: IF S=10LET Q=Q
      -I+C+INT (O*9/
      5):I=0:O=0:IF
      P=SLET Q=Q+B*2
220: IF S=11LET Q=Q
      +C
225: IF S=12LET Q=Q
      +C*2
230: C=0:IF S=PLET
      Q=Q+A:P=0
235: IF P=13LET P=S
240: IF Q<0LET Q=-Q
      :GOTO 250
245: R=R+Q:PRINT "Y
      OU WIN $";Q:Q=
      0:GOTO 15
250: R=R-Q:PRINT "Y
      OU LOSE $";Q:Q
      =0:GOTO 15
255: IF S=4LET Q=Q+
      F*7:F=0
260: IF S=6LET Q=Q+
      G*9:G=0
265: IF S=8LET Q=Q+
      H*9:H=0
270: IF S=10LET Q=Q
      +I*7:I=0
275: RETURN
280: IF R<0LET R=-R
      :PRINT "YOU LO
      ST $";R:GOTO 1
      0
285: PRINT "YOU WON
      $";R:GOTO 10
290: W=RND 6:RETURN
      STATUS 1      1557

```

6.7 JACKPOT SLOTS

Gamester Gary Heidbrink serves up another fun package with this emulation of the old One-Arm Bandit.

It is a cinch to play. (What slot machine isn't?) Why, you don't even have to waste time depositing coins! Just load the program into your trusty PC, put your portable slot machine in the RUN mode, and press the DEFine key followed by the letter J. Alternatively, just type in the command *RUN*. Now answer the prompts and start playing! Here is what a few typical cranks of the long handle might show you:

```

** JACKPOT SLOTS **
NEED INSTRUCTIONS?(Y/N) N
PLAY OR QUIT? (P/Q):P
LEVER HAS BEEN PULLED!
777 APPLE APPLE
YOU LOSE!!
PLAY OR QUIT? (P/Q):P
LEVER HAS BEEN PULLED!
CHERRY APPLE BELL
YOU WIN!! $ 0.5
PLAY OR QUIT? (P/Q):Q
HOUSE OWES YOU $ .25

```

By the way, the program has been arranged so that you can just hit the ENTER key each time the *play or quit?* query comes up. That will automatically cause another "spin of the wheels." Good luck!

Program listing: JACKPOT SLOTS

```

5:"J":Z=3254:H=0          "TAKE:$";J;" B
   :O=0:J=0              AL:$";O:M$="Y"
10:A$=" CHERRY"          :GOTO 45
15:B$=" APPLE"           60:PRINT " EACH P
20:C$=" PLUM"             LAY COSTS $ .2
25:D$=" BELL"            5"
30:E$=" BAR"             65:PRINT " PAYOFF
35:F$=" 777"             S ARE:"
40:G$=" ANY"             70:PRINT A$;G$;G$
45:PAUSE " ** J          ;" $.50"
   ACKPOT SLOTS *       75:PRINT A$;A$;G$
   *".W=0               ;" $1.25"
50:INPUT "NEED IN        80:PRINT B$;B$;B$
   STRUCTIONS?(Y/       ;" $2.50"
   N) ";M$:IF M$=       85:PRINT C$;C$;C$
   "N"GOTO 105          ;" $3.50"
55:IF M$="J"PRINT        90:PRINT D$;D$;D$

```

```

        ;" $6.75"
95: PRINT E$;E$;E$
        ;" $50.00"
100: PRINT F$;F$;F$
        ;" $50.00":
        PAUSE ""
105: RANDOM :Z=RND
        100
110: INPUT " PLAY O
        R QUIT? (P/Q):
        ;M$: IF M$="Q"
        GOTO 290
115: PAUSE " LEVER
        HAS BEEN PULLE
        D!": J=J+.25:H=
        H+.05:W=W-.25:
        GOTO 125
120: Z=439147*Z+J:Z
        =23*Z-INT (Z*.
        23)*100:Y=INT
        (Z*.2)+1:
        RETURN
125: GOSUB 120
130: IF Y<8LET I$=C
        $:GOTO 160
135: IF Y<14LET I$=
        B$:GOTO 160
140: IF Y<17LET I$=
        E$:GOTO 160
145: IF Y<19LET I$=
        A$:GOTO 160
150: IF Y=19LET I$=
        D$:GOTO 160
155: I$=F$
160: GOSUB 120
165: IF Y<7LET K$=A
        $:GOTO 195
170: IF Y<12LET K$=
        D$:GOTO 195
175: IF Y<16LET K$=
        B$:GOTO 195
180: IF Y<19LET K$=
        C$:GOTO 195
185: IF Y=19LET K$=
        F$:GOTO 195
190: K$=E$
195: GOSUB 120
200: IF Y<10LET L$=
        D$:GOTO 225
205: IF Y<15LET L$=
        B$:GOTO 225
210: IF Y<19LET L$=
        C$:GOTO 225
215: IF Y=19LET L$=
        E$:GOTO 225
220: L$=F$
225: BEEP 1:WAIT 10
        0:PRINT I$;" "
        ;K$;" ";L$:
        WAIT
230: IF I$=K$GOTO 2
        45
235: IF I$=A$LET X=
        .5:GOTO 285
240: PAUSE " YOU LO
        SE!!":GOTO 110
245: IF I$=A$LET X=
        1.25:GOTO 285
250: IF I$=L$GOTO 2
        60
255: GOTO 240
260: IF I$=B$LET X=
        2.5
265: IF I$=C$LET X=
        3.5
270: IF I$=D$LET X=
        6.75
275: IF I$=E$LET X=
        50
280: IF I$=F$LET N=
        INT (H/.25)*.2
        5:X=N+50:H=H-N
285: PRINT " YOU WI
        N!! $";X:W=W+X
        :GOTO 110
290: IF W<0GOTO 300
295: PRINT " HOUSE
        OWES YOU $";W:
        O=O-W:GOTO 45
300: W=W*-1:PRINT "
        YOU OWE $";W:
        O=O+W:GOTO 45
STATUS 1 1407

```

6.8 TIC-TAC-TOE

How good can a computer be at playing a game? Well, with the right program, it can be unbeatable. And, that is just what Norlin Rober has created with this crafty implementation of the old chalkboard game. Play this game for awhile and you will know exactly what it feels like to be a loser—frustrating!

Want to try your hand at it anyhow? Okay, then load the program into memory, place your pocket computer in the program RUN mode, and put the game into action with the *RUN* command. Here is a sample of how the game might transpire:

```
YOUR MOVE: 5
MY MOVE, 1
YOUR MOVE: 9
MY MOVE, 7
YOUR MOVE: 4
MY MOVE, 6
YOUR MOVE: 3
MY MOVE, 2 TO KITTY.
YOU  Ø, ME  Ø, KITTY  1
```

Try as hard as you may, the best you will be able to do is tie with your mighty PC. Of course, you can always look at the situation from a different point of view—you always have a winner in your pocket when this program is in your PC's memory!

Just in case you want to give this version of the game a real workout, the accompanying diagram (Fig. 6.8) illustrates the

7	8	9
4	5	6
1	2	3

Fig. 6.8 TIC-TAC-TOE playing positions.

numbered positions referred to by the computer. Just tell the computer where you want to place your X.

Say, maybe you could modify the program so that the computer sometimes loses. I'll bet that would make you feel better!

Program listing: TIC-TAC-TOE

```

1: E=34: F=24: G=72
   : I=23: J=42: K=4
   : L=K: M=E: O=16
   : P=13: R=31: S=6
   : T=17: U=K: V=0
   : W=0
2: H=0: Q=0: GOSUB
   11: IF X=5 GOTO
   20
3: A=X: B=X-2*INT
   (X/2): GOSUB 39
   : IF X+Y<>10
   GOTO 13
4: C=N-10*INT (.1
   *N): GOSUB 7:
   GOSUB 10: IF X+
   Y<>10 GOTO 13
5: IF B=0 IF D=8
   LET C=6: GOSUB
   7: GOTO 13
6: C=8+B*(D<9):
   GOSUB 7: GOTO 1
   4
7: Y=INT ((C+2)/3
   ): Z=5-ABS (5-A
   ): IF Z=3 LET C=
   6*Y-C-2
8: IF Z=4 LET C=3*
   C-8*Y+6
9: Y=ABS (C-10*(A
   >5)): RETURN
10: USING "##":
   PAUSE "MY MOVE
   , "; Y: Q=10*Q+Y
11: X=0: INPUT "YOU
   R MOVE: "; X: H=
   10*H+X: RETURN
12: GOSUB 30: GOTO
   11
13: W=W+U+1: U=0: Y=
   10-Y: Q=10*Q+Y:
   BEEP 1: PAUSE "
   I WIN IN SQUAR
   E"; Y: GOTO 15
14: U=U+1: Q=100+Y:
   PAUSE "MY MOVE
   , "; Y: ", TO KIT
   TY. "
15: PAUSE USING "#
   ##"; "THE SCORE
   : ": PRINT "YOU
   0, ME"; W: ", K
   ITTY"; U: GOTO 2
20: Y=1: GOSUB 10: A
   =X: Y=10-X+6*(X
   =9): GOSUB 10: B
   =X: IF ABS (A-3
   )=1 GOTO 25
21: Y=9-A+6*(A>6):
   IF X+Y<>10 GOTO
   13
22: GOSUB 10: Y=14-
   A: IF B/2>INT (
   B/2) IF X+Y<>10
   GOTO 13
23: Y=6-B: IF X/2=
   INT (X/2) LET Y
   =10-X
24: GOTO 14
25: Y=10-B+(5-A)*(
   B=9): GOSUB 10:
   IF ABS (A-B+2)
   =1 LET Y=1+(A+3
   )*(X=9): GOTO 1
   3
26: IF A+Y=6 IF 1<>
   ABS (A+X-8) LET
   Y=2*A-1: GOTO 1
   3
27: Y=10-X+(X=9)*(
   1+ABS (B-3)):
   GOTO 14

```



```

30:PRINT USING "#
    ####";"YOU:";H
    ;"    MR:";Q:
    USING "##":
    RETURN
31:"D"GOSUB 30:
    GOTO 2
39:Y=5:GOSUB 10:C
    =X:GOSUB 7:D=Y
    ;N=Q(D+12-9*B)
    ;C=INT (.1*N):
    GOSUB 7:GOSUB
    10:RETURN
STATUS 1          1005

```

6.9 MUSIC

Can you really play music on a pocket computer? I will let you be the judge.

This program gives a rendition of Johann Sebastian Bach's Prelude Number II from the Well Tempered Clavier. The arrangement was programmed by Brian Peterson. I don't know how he did it, but the results are absolutely amazing.

You do need at least a 4K memory expansion module in your PC in order to use this program. The program is rather lengthy, consisting mostly of data statements that specify the notes to be played. If you have the fortitude to load this masterpiece into your machine, I think you will be rewarded with a novel performance that will certainly impress your friends and associates (you know, those other people who don't understand why you waste so much time fooling around with your computer)!

Program listing: MUSIC

```

10:DIM A(51),B(51
    )
15:FOR X=1TO 51
20:READ A(X):B(X)
    =10^(X/40)*8
25:NEXT X
30:WAIT 25:PRINT
    "ONE":PRINT "A
    ND":PRINT "TWO
    ":PRINT "AND":
    CLS
50:READ N:IF N
    BEEP 1,A(N),B(
    N)*1.2:GOTO 50
51:BEEP 1,A(11),B
    (11)*4
52:READ N:IF N
    BEEP 1,A(N),B(
    N)*2:GOTO 52
53:READ N:IF N
    BEEP 1,A(N),B(
    N)*1.5:GOTO 53
54:BEEP 1,A(11),B
    (11)*4
55:READ N:IF N
    BEEP 1,A(N),B(
    N)*2:GOTO 55
56:READ N:IF N
    BEEP 1,A(N),B(
    N)*1.5:GOTO 56
57:BEEP 1,A(11),B
    (11)*4

```

```

58: READ N: IF N
    BEEP 1, A(N), B(
N)*2: GOTO 58
59: READ N: IF N
    BEEP 1, A(N), B(
N)*1.5: GOTO 59
60: READ N: IF N
    BEEP 1, A(N), B(
N): GOTO 60
61: BEEP 1, A(16), B
(16): BEEP 1, A(
23), B(23): BEEP
1, A(26), B(26):
BEEP 1, A(28), B
(28)
62: BEEP 1, A(32), B
(32)*20
63: READ N: IF N
    BEEP 1, A(N), B(
N)*2: GOTO 63
64: READ N: IF N
    BEEP 1, A(N), B(
N)*1: GOTO 64
65: READ N: IF N
    BEEP 1, A(N), B(
N)*4: GOTO 65
66: READ N: IF N
    BEEP 1, A(N), B(
N)*2: GOTO 66
67: BEEP 1, A(16), B
(16): BEEP 1, A(
21), B(21): BEEP
1, A(24), B(24):
BEEP 1, A(28), B
(28)
68: BEEP 1, A(33), B
(33)*20
69: READ N: IF N
    BEEP 1, A(N), B(
N)*2: GOTO 69
70: READ N: IF N
    BEEP 1, A(N), B(
N)*1: GOTO 70
71: READ N: IF N
    BEEP 1, A(N), B(
N)*1.2: GOTO 71
72: READ N: IF N
    BEEP 1, A(N), B(
N)*2: GOTO 72
73: READ N: IF N
    BEEP 1, A(N), B(
N)*4: GOTO 73
74: READ N: IF N
    BEEP 1, A(N), B(
N)*6: GOTO 74
75: READ N: IF N
    BEEP 1, A(N), B(
N)*8: GOTO 75
76: BEEP 1, A(32), B
(32)*2: BEEP 1,
A(30), B(30)*2:
BEEP 1, A(32), B
(32)*30
77: END
100: DATA 255, 244, 2
29, 219, 204, 195
, 184, 172, 163, 1
51
101: DATA 143, 133, 1
25, 118, 111, 105
, 99, 94, 88, 82, 7
7
102: DATA 72, 68, 64,
59, 55, 52, 49, 45
, 43, 40, 37, 35
103: DATA 32, 30, 28,
26, 24, 22, 21, 19
, 18, 16, 15, 14
104: DATA 12, 11, 10,
9, 8, 7
200: DATA 40, 31, 30,
31, 28, 31, 30, 31
201: DATA 40, 31, 30,
31, 28, 31, 30, 31
202: DATA 36, 33, 32,
33, 28, 33, 32, 33
203: DATA 36, 33, 32,
33, 28, 33, 32, 33
204: DATA 39, 33, 31,
33, 30, 33, 31, 33
205: DATA 39, 33, 31,
33, 30, 33, 31, 33
206: DATA 40, 35: 33,
35, 31, 35, 33, 35
207: DATA 40, 35, 33,
35, 31, 35, 33, 35
208: DATA 43, 36, 35,
36, 31, 36, 35, 36
209: DATA 43, 36, 35,
36, 31, 36, 35, 36
210: DATA 42, 34, 32,
34, 30, 34, 32, 34

```

211: DATA 42, 34, 32,	31, 33, 31, 30, 31
34, 30, 34, 32, 34	237: DATA 21, 31, 30,
212: DATA 42, 35, 34,	31, 33, 31, 30, 31
35, 30, 35, 34, 35	238: DATA 22, 28, 27,
213: DATA 42, 35, 34,	28, 31, 28, 27, 28
35, 30, 35, 34, 35	239: DATA 22, 28, 27,
214: DATA 40, 32, 30,	28, 31, 28, 27, 28
32, 28, 32, 30, 32	240: DATA 31, 28, 27,
215: DATA 40, 32, 30,	28, 23, 28, 27, 28
32, 28, 32, 30, 32	241: DATA 31, 28, 27,
216: DATA 40, 33, 32,	28, 23, 28, 27, 28
33, 28, 33, 32, 33	242: DATA 34, 28, 27,
217: DATA 40, 33, 32,	28, 25, 28, 27, 28
33, 28, 33, 32, 33	243: DATA 34, 28, 27,
218: DATA 38, 33, 31,	28, 25, 28, 27, 28
33, 30, 33, 31, 33	244: DATA 35, 28, 27,
219: DATA 38, 33, 31,	28, 30, 28, 27, 28
33, 30, 33, 31, 33	245: DATA 35, 28, 27,
220: DATA 38, 35, 33,	28, 30, 28, 27, 28
35, 31, 35, 33, 35	246: DATA 36, 28, 27,
221: DATA 38, 35, 33,	28, 30, 28, 27, 28
35, 31, 35, 33, 35	247: DATA 36, 28, 27,
222: DATA 36, 35, 33,	28, 30, 28, 27, 28
35, 31, 35, 33, 35	248: DATA 0, 15, 18, 2
223: DATA 36, 35, 33,	1, 0, 24, 21, 20, 2
35, 31, 35, 33, 35	1, 27, 21, 30, 27,
224: DATA 36, 30, 28,	24, 21, 20, 21
30, 26, 30, 28, 30	249: DATA 0, 16, 19, 2
225: DATA 36, 30, 28,	3, 0, 28, 23, 22, 2
30, 26, 30, 28, 30	3, 31, 28, 35, 31,
226: DATA 35, 26, 24,	28, 24, 23, 24
26, 31, 26, 24, 26	250: DATA 0, 13, 22, 2
227: DATA 35, 26, 24,	8, 0, 31, 28, 27, 2
26, 31, 26, 24, 26	8, 34, 28, 37, 34,
228: DATA 33, 28, 26,	31, 28, 27, 28
28, 25, 28, 26, 28	251: DATA 11, 0, 42, 4
229: DATA 33, 28, 26,	0, 42, 43, 40, 39,
28, 25, 28, 26, 28	40, 37, 40, 39, 40
230: DATA 33, 30, 28,	, 42, 39, 37, 39
30, 27, 30, 28, 30	252: DATA 35, 39, 37,
231: DATA 33, 30, 28,	39, 40, 37, 35, 37
30, 27, 30, 28, 30	, 34, 37, 35, 37, 3
232: DATA 33, 30, 28,	9, 35, 34, 35
30, 27, 30, 28, 30	253: DATA 30, 47, 45,
233: DATA 33, 30, 28,	47, 48, 45, 43, 45
30, 27, 30, 28, 30	, 42, 45, 43, 45, 4
234: DATA 31, 28, 27,	7, 43, 42, 43
28, 23, 28, 27, 28	254: DATA 40, 43, 42,
235: DATA 31, 28, 27,	43, 45, 42, 40, 42
28, 23, 28, 27, 28	, 39, 42, 40, 42, 4
236: DATA 21, 31, 30,	3, 40, 39, 40


```

255: DATA 35, 40, 39,      8, 21, 24, 23, 21,
    40, 36, 45, 43, 45      27, 21, 30, 21, 27
    , 35, 43, 42, 43, 3    , 24, 23, 21
    3, 42, 40, 42          260: DATA 20, 29, 26,
256: DATA 31, 40, 39,      23, 28, 24, 21, 24
    40, 36, 33, 31, 33      , 23, 26, 23, 20, 2
    , 35, 31, 30, 31, 3    4, 21, 18, 21
    3, 30, 28, 30, 0        261: DATA 20, 23, 20,
257: DATA 28, 30, 32,      16, 21, 18, 15, 18
    0, 33, 35, 36, 38,      , 0, 4, 11, 16, 18,
    40, 38, 36, 35, 0,      20, 23, 26, 23
    33, 0, 35, 32, 0        262: DATA 24, 28, 33,
258: DATA 35, 33, 32,      30, 33, 36, 0, 40,
    33, 35, 36, 35, 0,      39, 40, 0, 35, 33,
    33, 31, 30, 31, 33      0, 30, 0, 0
    , 30, 31, 33, 0        263: DATA 0
259: DATA 27, 4, 15, 1    STATUS 1          3403

```

Once you have the program loaded, the performance is started by executing a *RUN* command with the PC in the RUN mode. It takes about 10 seconds after you issue the *RUN* directive (for the tiny conductor inside the PC to get the orchestra composed, of course) before the concert begins. You will know it is time to dim the lights for the performance when the display flashes:

```

ONE
AND
TWO
AND

```

This program is purely for entertainment. Enjoy!

6.10 LCD GRAPHICS

Say, do you like to doodle? How about drawing pictures on the LCD of your PC? Well, David Motto likes to, and he has provided a series of simple designs to challenge your imagination.

To take a look at how they appear on your LCD, load the source listing provided into your pocket machine. Place the unit in the RUN mode, and issue a *RUN* command. The pictures will be displayed one by one for a brief period of time (you can increase the length of time each diagram is left on the screen by changing the *WAIT 100* statement in line 100).

See your PC owner's manual for an explanation of how graphics are coded on your unit. The column-by-column codes are stored within DATA statements in this program. A very simple program (consisting of lines 1 through 100) is used to access each data statement in succession and display the pattern each represents. You can extend the concept to produce your own creations and incorporate them into your own programs. Note that lines 1 through 100 are simply provided as a means for you to view the patterns. When the program runs out of data statements, you will see an error message appear on the display. Just start the program over if you want to view the pictures again.

Of course, you can substitute or append your own creations to the list provided, as desired.

By the way, note that David has included a complete set of graphics representing chess pieces. Now, is someone going to try to program a game of chess on a PC that could use those graphics? What a challenge!

Program listing: GRAPHICS

<pre> 1:RESTORE 1000 5:DIM X\$(0)*80 100:CLS :WAIT 100: GCURSOR 0:READ X\$(0):GPRINT X \$(0):GOTO 100 1000:DATA "081436 49361408" 1010:DATA "776355 08556377" 1020:DATA "553677 00773655" 1030:DATA "1C0055 4955001C" 1040:DATA "080014 002A0055" 1050:DATA "775577 00775577" 1060:DATA "7F2214 0814227F" 1070:DATA "63411C 141C4163" 1080:DATA "493622 49223649" </pre>	<pre> 1090:DATA "775D77 22775D77" 1100:DATA "364949 36494936" 1110:DATA "7F415D 555D417F" 1120:DATA "1C2A49 7F492A1C" 1200:DATA "0E1121 4221110E": REM HEART 1210:DATA "081C4A 7F4A1C08": REM CLUB 1220:DATA "081422 41221408": REM DIAMON D 1230:DATA "183C1E 7F1E3C18": REM SPADE 1300:DATA "1D2222 4122221D": REM TAURUS </pre>
--	--

```

1310: DATA "21322C
      20207C20":
      REM JUPITE
      R
1320: DATA "062979
      2906": REM
      FEMALE
1330: DATA "304848
      48350307":
      REM MALE
1340: DATA "434C34
      2A161961":
      REM SPIRAL
1350: DATA "04324A
      14292610":
      REM SPIRAL
1360: DATA "1C3E7F
      7F7F3E1C":
      REM BALL
1370: DATA "1C2241
      4141221C":
      REM CIRCLE
1380: DATA "63594F
      454F5963":
      REM BELL
1400: DATA "0E7C7E
      7C0E": REM
      ROOK (B)
1410: DATA "080C4E
      677E": REM
      KNIGHT (B)
1420: DATA "4C5E3F
      5E4C": REM
      BISHOP (B)
1430: DATA "427C7F
      7C42": REM
      QUEEN (B)
1440: DATA "407A7F
      7A40": REM
      KING (B)
1450: DATA "1C1E1F
      1E1C": REM
      PAWN (B)
1500: DATA "0E7C42
      7C0E": REM
      ROOK (W)
1510: DATA "080C4A
      257E": REM
      KNIGHT (W)
1520: DATA "4C5233
      524C": REM
      BISHOP (W)
1530: DATA "427C43
      7C42": REM
      QUEEN (W)
1540: DATA "407A4F
      7A40": REM
      KING (W)
1550: DATA "1C1211
      121C": REM
      PAWN (W)
1600: DATA "3F0B7F
      1F1F1F7F":
      REM ELEPHA
      NT
1610: DATA "40787E
      7F7E7840":
      REM BUILDI
      NG
1620: DATA "3F7F79
      79393F3F3838
      787C7C3838":
      REM LOCOMO
      TIVE
1630: DATA "3C7C7C
      3C3C7C7C3C10
      ": REM BOXC
      AR
1640: DATA "020302
      1C0C5C7C6C08
      10204040":
      REM KANGAR
      OO
1650: DATA "010101
      0909191F1919
      18181C1E1A06
      060706060202
      02": REM EN
      TERPRISE
1660: DATA "1A1E1F
      1C0C04040404
      04040E0E0E6F
      6F7F7B7B6860
      60": REM KL
      INGON
1670: DATA "081412
      122448102040
      000000402010
      102040000000
      402010102040
      0000004020"
STATUS 1 1556

```


6.11 NOTES

Musical notes will help you pass your idle hours away. This short routine converts your PC into a sing-along companion. The digits keypad springs into new life as a musical scale covering the middle C octave. Credit goes to Brian Peterson for deriving the parameter values used to generate the musical scale via *BEEP* statements.

Program listing: NOTES

100: X\$=INKEY\$: IF	GOTO 100
X\$="" THEN 100	240: BEEP 1, 35, 98:
110: IF (X\$<"0")+ (X	GOTO 100
\$>"9") THEN 100	250: BEEP 1, 30, 104:
120: X=(ASC (X\$)-47	GOTO 100
)*10+190: GOTO	260: BEEP 1, 26, 118:
X	GOTO 100
200: BEEP 1, 52, 72:	270: BEEP 1, 22, 130:
GOTO 100	GOTO 100
210: BEEP 1, 49, 76:	280: BEEP 1, 21, 136:
GOTO 100	GOTO 100
220: BEEP 1, 43, 84:	290: BEEP 1, 18, 150:
GOTO 100	GOTO 100
230: BEEP 1, 37, 92:	STATUS 1 274

Try it. Load the program, place the PC in the RUN mode, and issue a *RUN* directive. Press any of the digits 0 through 9. You should hear a brief musical note. Hold the key down and the note will repeat automatically. The digit 0 gives the lowest tone (the musical note B, just below middle C). The digit 9 provides the highest tone (the note D one octave above middle-C). The range (a little over an octave) is sufficient to play a variety of popular melodies.

If you think the notes are a little short for your tastes, you can extend their duration. Just multiply the third parameter in each *BEEP* statement (lines 200–290) by a common factor. Try a multiplier of two or three and then see what you think.

Say, maybe you can come up with a new melody that will make you rich and famous!

6.12 AMERICAN ROULETTE

Here is a pocket computer version of the popular Las Vegas gambling game known as American Roulette. American Roulette is based on a betting board having the numbers zero through 36 plus the dreaded double zero (00). This version of the game permits the following bets:

Double zero A bet that the marble will fall in the 00 slot.

Single number bet Enter the number desired (0 to 36) followed by the amount of the wager.

Red The numbers 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30, 32, 34, and 36 are red.

Black The numbers 2, 4, 6, 8, 10, 11, 13, 15, 17, 20, 22, 24, 26, 28, 29, 31, 33, and 35 are this color.

Even Any even number (excluding zero or double zero).

Odd Any odd number (excluding zero or double zero).

1 to 18 Any number within this range.

19 to 36 Any number within this range.

1 to 12 Inclusive.

13 to 24 Inclusive.

25 to 36 Inclusive.

Column 1 to 34 A bet that the number will be one of these: 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, or 34.

Column 2 to 35 A bet that the number will be one of these: 2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, or 35.

Column 3 to 36 If a 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, or 36 comes up, you win.

Bets continue to ride until they are removed by betting zero or you change the value of the bet. If you want to clear all bets, it is easiest to simply quit the game and start over.

What are the payoff odds? Betting on red, black, even, odd, 1 to 18, or 19 to 36 pays even money for a win. Betting on 1 to 12, 13 to 24, 25 to 36, or a column bet pays 2 to 1. Single numbers, including 0 and 00, pay 35 to 1.

Gary Heidbrink, the programmer of this version of American Roulette for a PC, wishes everyone the best of luck!

Program listing: AMERICAN ROULETTE

```

10: CLEAR : DIM A(5
    7): FOR E=0 TO 5
    7: A(E)=0: NEXT
    E: PAUSE "
    AMERICAN ROULE
    TTE"
15: INPUT "BET OR
    QUIT? (B/Q): "
    ;D$: IF D$="Q"
    GOTO 295
20: INPUT "CHANGIN
    G BETS? (Y/N):
    ";D$: IF D$="Y
    "GOTO 25
22: GOTO 95
25: INPUT "DOUBLE
    00: ";A(8)
30: INPUT "ENTER S
    INGLE BET #: "
    ;E: A(6)=E+21:
    INPUT "AMOUNT
    BET: ";A(A(6))
    : GOTO 30
35: INPUT "RED: ";
    A(9)
40: INPUT "BLACK:
    ";A(10)
45: INPUT "EVEN: "
    ;A(11)
50: INPUT "ODD: ";
    A(12)
55: INPUT "1-18: "
    ;A(13)
60: INPUT "19-36:
    ";A(14)
65: INPUT "1-12: "
    ;A(15)
70: INPUT "13-24:
    ";A(16)
75: INPUT "25-36:
    ";A(17)
80: INPUT "COLUMN
    1-34: ";A(18)
85: INPUT "COLUMN
    2-35: ";A(19)
90: INPUT "COLUMN
    3-36: ";A(20)
95: PAUSE "          W

                                HEEL IS SPUN"
100: RANDOM : A=(RND
    38)-1
105: IF A=37 LET A(6
    )=A(8)*36: GOTO
    270
110: IF A=0 LET A(6)
    =A(21)*36: GOTO
    270
115: FOR E=1 TO 36:
    IF A=E LET A(6)
    =A(E+21)*36
120: NEXT E
125: IF A=2 GOTO 220
130: IF A=4 GOTO 220
135: IF A=6 GOTO 220
140: IF A=8 GOTO 220
145: IF A=10 GOTO 22
    0
150: IF A=11 GOTO 22
    0
155: IF A=13 GOTO 22
    0
160: IF A=15 GOTO 22
    0
165: IF A=17 GOTO 22
    0
170: IF A=20 GOTO 22
    0
175: IF A=22 GOTO 22
    0
180: IF A=24 GOTO 22
    0
185: IF A=26 GOTO 22
    0
190: IF A=28 GOTO 22
    0
195: IF A=29 GOTO 22
    0
200: IF A=31 GOTO 22
    0
205: IF A=33 GOTO 22
    0
210: IF A=35 GOTO 22
    0
215: A(6)=A(6)+A(9)
    *2: GOTO 225
220: A(6)=A(6)+A(10)

```

```

)*2
225: IF A/2=INT (A/
2)LET A(6)=A(6
)+A(11)*2:GOTO
235
230:A(6)=A(6)+A(12
)*2
235: IF A>24LET A(6
)=A(6)+A(17)*3
+A(14)*2:GOTO
255
240: IF A>18LET A(6
)=A(6)+A(16)*3
+A(14)*2
245: IF A>12LET A(6
)=A(6)+A(16)*3
+A(13)*2:GOTO
255
250:A(6)=A(6)+A(15
)*3+A(13)*2
255: IF A/3=INT (A/
3)LET A(6)=A(6
)+A(20)*3:GOTO
270
260: IF (A+1)/3=INT
((A+1)/3)LET A
(6)=A(6)+A(19)
*3:GOTO 270
265:A(6)=A(6)+A(18

```

```

)*3
270:FOR E=8TO 57:A
(6)=A(6)-A(E):
NEXT E:A(7)=A(
7)+A(6)
275:BEEP 1:IF A=37
PRINT "BALL LA
NDS ON: 00":
GOTO 285
280:PRINT "BALL LA
NDS ON: ";A
285: IF A(6)<0LET A
(6)=-A(6):
PRINT "YOU LOS
E $";A(6):GOTO
15
290:PRINT "YOU WIN
$";A(6):GOTO
15
295: IF A(7)<0LET A
(7)=-A(7):
PRINT "YOU LOS
T $";A(7):GOTO
10
300:PRINT "YOU WON
$";A(7):GOTO
10
STATUS 1          1571

```

Programmer's Aids

7.1 CONVERTING BASES 2 to 16

Computer programmers often like or need to be able to convert rapidly between the decimal numbering system and the binary, octal, and hexadecimal systems. This program makes such conversions easy, particularly since it allows you to specify digits above 9 (those that represent the values 10 through 15) as the letters A through F. This is the type of nomenclature generally used by computer operators.

To use the program, load it into your PC, place your unit in the program execution (RUN) mode, and execute a *RUN* command. Respond appropriately to the initial prompt:

```
B(N)-10 or B10-BN 1or2? 2
```

As an example, the option of going from a decimal number to the hexadecimal system will be selected. Having chosen the direction of the conversion, the program now asks which base the decimal input is to be converted to:

```
BASE (N=?) (2-16) 16
Base 10 to Base 16
```

Note that the program confirms the base selected. It then asks for a decimal number to be inputted. Once inputted, it calculates the answer:

```
DECIMAL No.= 65535
FFFF
```


If you press ENTER after the answer has been displayed, the program will loop back to perform another conversion using the same base:

```
Base 10 to Base 16
DECIMAL No.= 65536
10000
```

When you want to switch to another base, press the BREAK key and start the program over again.

Going from a selected base (in the range 2 through 16) to decimal is just as straightforward. Here is a sample run of the program for such a conversion in the reverse direction:

```
Base Conversions (2-16)
B(N)-10 or B10-BN 1or2? 1
BASE (N=?) (2-16) 16
BASE 16 TO BASE 10
?FFFF
65535
```

Again, hitting ENTER alone after the result is displayed will cause the program to loop back in order to perform another similar conversion.

This conversion routine was created by Thomas S. Cox.

Program listing: BASES 2 to 16 CONVERSION

```
10:"BC"CLEAR :                )PRINT "BASE 0
    WAIT 50:PRINT              UT OF RANGE ";
    "Base Conversions (2-16)"  CLS :GOTO 30
12:A$="0123456789ABCDEF"      50:CLS
15:DIM A0$(0)*50,             60:ON LGOTO 100,2
    A1$(0)*50                  00
20:INPUT "B(N)-10 or B10-BN 1or2? ";L
25:IF (L=1)+(L=2)             100:WAIT 50:PRINT
    <>1CLS :GOTO 2              "BASE ";N;" TO
    0                          BASE 10":A0$(
30:INPUT "BASE (N              0)="":A1$(0)="
    =?) (2-16) ";N            "
40:IF (N<2OR N>16             105:WAIT 0:PRINT "
                                # IN BASE";N;"
                                ";INPUT A0$(0)
                                )
110:CLS :A=LEN A0$
    (0):B=0
```

```

115:FOR I=0TO A-1:
    C$=MID$(A$(0
    ),A-1,1):G=ASC
    C$:IF (G)=48)+
    (G<=57)=2LET G
    =G-48:GOTO 130
120:IF (G)=65)+(G<
    =70)=2LET G=G-
    55:GOTO 130
125:BEEP 1:CLS :
    WAIT 20:PRINT
    "INVALID ENTRY
    : TRY AGAIN":
    CLS :WAIT 0:
    GOTO 100
130:IF G>=NWAIT 50
    :PRINT "DIGIT
    >= BASE":CLS :
    WAIT 0:GOTO 10
    5
140:B=B+G*N^I:NEXT
    I:WAIT :PRINT
    B:GOTO 100
200:WAIT 50:PRINT
    "Base 10 to Ba
    se ";N:A$(0)=
    "":A1$(0)="" :B
    $=""
205:WAIT 0:INPUT "
    DECIMAL No., = "
    :D:E=D:IF D>9,
    999E10WAIT 50:
    PRINT "TOO BIG
    " :CLS :WAIT 0
    :GOTO 200
215:IF (N=2)+(D>2^
    26-1)=2WAIT 50
    :PRINT "TOO LA
    RGE FOR DISPLA
    Y":CLS :WAIT 0
    :GOTO 200
220:WAIT 0:F=0:CLS
    230:FOR I=1TO 26
    235:B=(E/N)-INT (E
    /N):C=INT ((B*
    N)+.5):IF INT
    (E/N)=0LET F=1
    240:E=INT (E/N)
    250:B$=MID$(A$,C+
    1,1):A$(0)=A0
    $(0)+B$
    260:IF F=1GOTO 300
    270:NEXT I
    280:WAIT 100:PRINT
    "TOO LARGE FOR
    DISPLAY":CLS
    :GOTO 200
300:WAIT 0:FOR I=
    LEN A$(0)TO 1
    STEP -1:B$=
    MID$(A$(0),I
    ,1)
    310:A1$(0)=A1$(0)+
    B$:NEXT I:WAIT
    :CLS
    320:PRINT A1$(0):
    GOTO 200
STATUS 1 1090

```

Another program for converting from one number base to another is included in this publication. While it has the advantage of being able to convert between any base in the range 2 through 99, it utilizes two digits to represent numbers above base 10. This can be somewhat cumbersome if one has to make frequent conversions, as is the case for computer programmers working in decimal and hexadecimal. The program shown here is convenient to use for the range of bases that it covers. Both programs were included in this manual to serve the diversified needs of PC users.

7.2 MEMORY DUMP

Did you know that your PC has PEEK and POKE capability? That is, you can peek into memory locations to examine their contents, and you can poke values into random access memory locations. Well, it is true, even if not advertised.

This program may therefore be used to “dump” the contents of any desired section of memory. You can use it to see how source code is stored in user memory sections or to examine the BASIC interpreter stored in ROM. The addresses for the 16K BASIC ROM are &C000 through &FFFF inclusive.

The program displays the contents of memory locations (addresses) as a hexadecimal value in the range 00 through FF. Eight bytes of memory are displayed on a line. The line is prefixed by the hexadecimal address of the first byte of memory whose contents are being displayed on that line. A second line is used to display the ASCII-equivalent characters for all codes that have printable symbols. A blank line is left after each group of eight bytes (and their ASCII-equivalents) to provide room for making handwritten notes when analyzing code.

To use the program, load it into memory, and then place the PC in its normal program RUN mode. Execute a *RUN* directive, and then answer the initial query for the starting address of the block of memory that you want to examine. Be sure to specify this address using hexadecimal notation. (Remember, hexadecimal notation is indicated to the PC by prefixing the value with an ampersand (&).) Press ENTER after specifying the address at which the dump is to start. Be sure your PC is plugged into a printer/plotter before starting the program. To stop the program, press the BREAK key. If you want to examine portions of memory without using a printer, then change the *LPRINT* statements in lines 29, 40, and 45 to plain *PRINT* statements.

By the way, user program storage starts at address &40C5 in a standard PC or one equipped with a 4K RAM module. If you have an 8K memory module installed, then BASIC source storage starts at address &38C5 (it can be quite enlightening to study a few lines of BASIC source code).

It can be even more interesting to dump and study the BASIC ROM. Indeed, one group of PC enthusiasts was able to determine virtually the entire machine language instruction set of the PC simply by exhaustively studying the ROM coding.

Have fun exploring!

Program listing: MEMORY DUMP

```

10: CSIZE 1: INPUT          , 1); " ";
    "STARTING ADDR        30: WAIT 0: FOR B=0
    ESS? "; A              TO 7: C=(PEEK (
20: A$="0123456789        A+B))AND (&F0)
    ABCDEF"                :D=1+INT (C/16
25: WAIT 0: W=INT (        );E=(PEEK (A+B
    A/4096)                ))AND (&0F)
26: X=INT ((A-(W*4        40: LPRINT MID$ (A
    096))/256)              $, D, 1);MID$ (A
27: Y=INT ((A-(W*4        $, E+1, 1); " ";
    096)-(X*256))/         NEXT B
    16)                    45: LPRINT :LPRINT
28: Z=INT ((A-(W*4        " ";:FOR
    096)-(X*256)-(        B=0TO 7:LPRINT
    Y*16)))                (CHR$ (PEEK (A
29: LPRINT MID$ (A        +B))); " ";
    $, W+1, 1);MID$        NEXT B:LPRINT
    (A$, X+1, 1);          :LF 1
    MID$ (A$, Y+1, 1      50: A=A+8: GOTO 25
    );MID$ (A$, Z+1       STATUS 1          416

```

Sample output: STARTING ADDRESS &C000

```

C000  55 FD 2A 65 FD A8 2A 68
      U   * e   * h
C008  78 84 81 84 2E 9A A5 7A
      x   a   .   z
C010  07 AE 78 85 24 AE 78 84
      x   *   x
C018  A4 AE 78 83 9A C4 AF FF
      x

```

7.3 RENUMBER

If you do a substantial amount of program development, you will certainly appreciate this powerful and valuable utility program developed by Milt Sherwin. It permits you to renumber lines in a BASIC program. In order to utilize it, however, you will need at

least a 4K memory expansion module in your PC, but it wouldn't hurt to have an 8K module (in case you need some incentive to save your money a little longer).

Remember, this program (that performs the renumbering) as well as the program that is going to be renumbered must reside in memory (together) at the same time.

There is also a very important restriction to keep in mind when using this program. The number of lines that can be renumbered at one time is limited to a maximum of 88. Should you want to renumber a program that has more than 88 lines, you may be able to accomplish your objective by sectioning the program; that is, have the renumbering program work over, say, half of the program at a time. This restriction is based on the fact that the string variables area normally used to hold string variables E\$ through Z\$ is used to store a "line numbers relationship table" when the renumbering program is in operation. If the space allotted to this function is exceeded, by having more than 88 sets of line numbers (old and new), then the program is likely malfunction.

In order to use this program on a regular basis, you will first want to prepare a copy of the program on a tape cassette. To do this, clear out memory using the command *NEW0* (while in the PRO mode). Load in the program as shown in the listing. Make a copy on tape using the *CSAVE* command. You might want to name the program "RENUMBER." Once you have a copy (or several, for backup) saved on tape, you will be ready to use it in the manner prescribed below.

You must adhere to the following series of instructions when using this program. Any deviation from the sequence prescribed can result in an erroneous performance of the renumbering program as well as possible alteration or destruction of the program that one wants to have renumbered.

1. Clear memory (using *NEW0*) and load the renumbering program from that tape cassette you made as suggested above.
2. If you wish to renumber a previously developed program that currently resides on a tape cassette, then use the tape *MERGE* command to merge the program into memory. This

is done immediately after loading the renumbering program and before trying to run the renumbering program!

3. If you plan to develop (key in) a program that you may eventually wish to renumber using the currently loaded renumber program, then you must immediately execute the renumbering program. Do this, before starting to develop your new program, by placing the PC in the RUN mode and issuing a *RUN* command. This initial execution of the program in this situation will cause the renumbering program to simulate a tape merge operation and effectively protects the program from renumbering itself.

4. Develop the program that is to be renumbered (this is done with the PC in the PRO mode). Assign a label to the start of the program under development (or being modified), so that you can access and test it by reference to that label. This is necessary because the renumbering program is protected by the merge or simulated merge operation, and the second program in memory cannot be accessed except by reference to labels.

5. When you want to renumber the program under development, go to the RUN mode, and issue a regular *RUN* command. This will activate the renumbering program. Now, respond to the prompt:

ST, IC, EN, LN

with the following parameters that tell the computer how to go about renumbering your program:

- ST** Starting line number (defaults to the first line of the program currently under development). This is the first line in the program that you want to have renumbered. Note that it doesn't have to be the lowest numbered line in the program.
- IC** Line increment value (defaults to a value of 10). This is the line number spacing value that the renumbering program will provide between each line number it assigns.
- EN** Ending line number (defaults to the end of the program being renumbered). Of course, you don't have

to renumber an entire program. This parameter is used to specify where you want the renumbering program to stop assigning new numbers.

LN Initial line number assignment (defaults to 100). This is the line number that you want the renumbering program to assign to the first line that it renumbers. Successive lines follow this at the spacing specified by the line increment value.

A comma is used to separate each parameter value, or to indicate that you want to use the default value. Thus, entering three commas `<,,,>` would result in the renumbering program using all of its default values. Use the ENTER key to terminate the parameter-specifying sequence. The program will then do its stuff while displaying the message:

WORKING . . .

6. At the conclusion of the renumbering process, the program will ask:

UNLINK RENUM?

Input a Y for “yes” if you are all through using the renumbering program. Specify N for “no” if you plan further development work. If you select the latter option, you may then go back to the PRO mode to examine the results of the renumbering operation, make further refinements to your program, etc. When you want to renumber again, repeat from step 5 of this operating sequence.

7. Once the renumbering program has been unlinked, you may *LIST* or *CSAVE* your renumbered program in the normal manner. The renumbering program, however, is no longer accessible.

8. When you are all through with a session, you should issue a *NEW0* command to your PC. This will restore various pointers used by the operating system that are altered by the renumbering package. Remember, the *NEW0* command will also clear out memory, so be sure you have saved a copy of your developed program.

Program listing: RENUMBERING

```

100: IF PEEK 30821<
    >PEEK 30825AND
    PEEK 30822<>
    PEEK 30826THEN
    150
110: IF PEEK 30824+
    1=256THEN POKE
    30826, PEEK 308
    24-255: POKE 30
    825, PEEK 30823
    +1: GOTO 130
120: POKE 30826,
    PEEK 30824+1:
    POKE 30825,
    PEEK 30823
130: POKE 30823,
    PEEK 30825:
    POKE 30824,
    PEEK 30826
140: POKE PEEK 3082
    3*256+PEEK 308
    24, 255: GOTO 52
    0
150: CLEAR : DIM A$(
    0)*26: C=1: P=&7
    050: R=&7150: M=
    PEEK 30825*256
    +PEEK 30826
160: INPUT "ST, IC, E
    N, LN "; A$(0)
170: FOR J=1 TO LEN
    A$(0)
180: C$=MID$ (A$(0)
    , J, 1)
190: IF C$="," THEN
    LET C=C+1: GOTO
    250
200: ON CGOTO 210, 2
    20, 230, 240
210: A$=A$+C$: GOTO
    250
220: B$=B$+C$: GOTO
    250
230: D$=D$+C$: GOTO
    250
240: E$=E$+C$
250: NEXT J
260: S=VAL A$: F=0: N
    =100
270: I=VAL B$: IF I=
    0 THEN LET I=10
280: E=VAL D$: IF E=
    0 THEN LET E=65
    279
290: L=VAL E$: IF L<
    0 THEN LET F=
    INT (L/256): N=
    L-F*256
300: PAUSE "WORKING
    . . ."
310: IF S=0 THEN 370
320: GOSUB 530: IF B
    =S THEN 350
330: IF PEEK (M+C+3
    )=255 THEN 520
340: M=M+C+3: GOTO 3
    20
350: IF L=0 THEN LET
    F=A: N=D
360: GOTO 380
370: GOSUB 530
380: POKE M, F: POKE
    (M+1), N
390: POKE P, A: POKE
    (P+1), D: P=P+2:
    POKE R, F: POKE
    (R+1), N: R=R+2
400: IF PEEK (M+C+3
    )=255 THEN 460
410: IF B=ETHE 450
420: N=N+1: IF N<256
    THEN 440
430: N=N-256: F=F+1
440: M=M+C+3: GOTO 3
    20
450: M=M+C+3: GOSUB
    530: IF PEEK (M
    +C+3)<>255 THEN
    450
460: P=P-&7050: N=M:
    M=PEEK 30825*2
    56+PEEK 30826
470: C=PEEK (M+2):
    GOSUB 570
480: M=M+C+3: Z=Z-1:
    IF Z<>0 THEN 47

```



```

0
490: INPUT "UNLINK
      RENUM? ";A$
500: IF LEFT$ (A$,1
      )<>"Y" THEN 520
510: POKE 30821,
      PEEK 30825:
      POKE 30822,
      PEEK 30826
520: END
530: C=PEEK M: D=INT
      (C/16): B=D*409
      6+(C-D*16)*256
      :A=C
540: C=PEEK (M+1): D
      =INT (C/16): B=
      B+D*16+(C-D*16
      ): D=C
550: C=PEEK (M+2): Z
      =Z+1
560: RETURN
570: A$="": J=M+3: K=
      0
580: A=PEEK J: G=
      PEEK (J+1)
590: IF A=&0D THEN 7
      80
600: IF A=&F1 AND G=
      &920R G=&940R
      G=&AEOR G=&AB
      THEN LET J=J+1
      :GOTO 620
610: J=J+1: GOTO 580
620: J=J+1: A=PEEK J
630: IF A<&300R A>&
      39 THEN 660
640: IF K=0 THEN LET
      K=J
650: A$=A$+CHR$ A:
      GOTO 620
660: L=LEN A$: IF L=
      0 THEN 580
670: D=VAL A$
680: FOR X=0 TO P
      STEP 2
690: E=PEEK (&7050+
      X)*256+PEEK (&

```

```

      7050+X+1): IF E
      =D THEN 720
700: NEXT X
710: GOTO 760
720: D=PEEK (&7150+
      X)*256+PEEK (&
      7150+X+1)
730: D$=STR$ D: R=
      LEN D$: IF R-L>
      0 GOSUB 790:
      GOTO 760
740: IF R-L<0 GOSUB
      830: GOTO 760
750: GOSUB 880
760: A$="": K=0: IF A
      =&2C THEN 620
770: GOTO 580
780: RETURN
790: I=PEEK (30823)
      *256+PEEK 3082
      4: H=I
800: POKE (I+R-L),
      PEEK I
810: I=I-1: IF I>K
      THEN 800
820: GOTO 860
830: I=K: H=PEEK (30
      823)*256+PEEK
      30824
840: POKE I, PEEK (I
      -R+L)
850: I=I+1: IF I<H
      THEN 840
860: C=C+R-L: POKE (
      M+2), C: J=J+R-L
870: H=H+R-L: POKE 3
      0823, INT (H/25
      6): POKE 30824,
      H-INT (H/256)*
      256
880: FOR X=1 TO R
890: A$=MID$ (D$,X,
      1): POKE K, ASC
      A$: K=K+1
900: NEXT X
910: RETURN
STATUS 1 1935

```

7.4 SIDELISTER

Mel Beckman has come up with an interesting program that permits you to list programs sideways! Sideways? Yep! Instead of having the printer/plotter unit list programs in its usual fashion, which results in many eye-jogging breaks in a long program line (as the printer goes to a new line every 12 to 14 characters), this special program results in the outputting of nice long lines that are easier to read.

How is it done? By having the printer “rotate” the output so that it displays lines going down a length of printer paper instead of across. Thus the listing of a program occurs in blocks that are actually printed sideways—hence the derivation of the name of this program.

This program is designed to be merged with a program already in memory. It has line numbers starting at 64000. Thus, any program that you want to list “sideways” should have line numbers below this value.

In order to effectively utilize this program you will first want to load it into memory, and then save a copy of it on a tape cassette. This is done so that you can merge the program back into memory when you want to use it to list another program.

Assuming that you have made a copy of the program on tape, you would apply it in the following manner.

After loading a program that you want to list, the Sidelister program is merged into memory from tape using the tape *MERGE* directive. Then place the PC into its normal RUN mode. Be sure the PC is properly attached to its printer/plotter accessory. Execute a *RUN “L”* command to start the sideways listing operation. The program will automatically list out your original program in a sideways format (note that the program being listed should not contain a label referenced “L”).

If you execute this program exactly as shown in the accompanying listing, it will print up to 42 characters of *CSIZE 2* per line. It will continue lines longer than this onto the next line, properly indented to maintain a neat format. It will also automatically start a new “page” after each group of 10 lines has been printed.

Sample operation of program SIDELISTER when Z = 2 (line 64025)

```

*TOP* 21116.5821
64000:"L"REM  SIDELISTER
64025:Z=2:X=212-Z*6
64050:GRAPH :GLCURSOR (X,0):SORGN :
        ROTATE 1:CSIZE Z
64100:X=0:L=0:C=0:M=0:Q=STATUS 2-STATUS 1:
        LPRINT "*TOP*";TIME
64105:GOSUB 64200
64110:IF PEEK (Q)=&FFEND
64115:LPRINT USING "#####";PEEK (Q)*256+
        PEEK (Q+1);": ";
64120:Q=Q+3:C=C+7
64125:IF PEEK (Q)=&0DLET Q=Q+1:GOTO 64105
64130:GOSUB 64400
64135:GOTO 64125
64200:IF C>MLET M=C:IF M>(84/Z)LET M=84/Z
64205:C=0:X=X-Z*10:GLCURSOR (X,0)
64210:L=L+1:IF L=INT (20/Z)GOSUB 64300
64215:RETURN
64300:L=0:X=0:Y=-(Z*M*6.4):GLCURSOR (X,Y):
        SORGN :M=0:RETURN
64400:IF PEEK Q>&E5GOTO 64450
64405:IF C>(84/Z)GOSUB 64500
64410:LPRINT CHR$ PEEK Q:C=C+1:Q=Q+1:
        RETURN
64450:V=PEEK (Q):W=PEEK (Q+1):U=&C053
64460:N=PEEK UAND &0F
64470:IF (PEEK (U+N+1)=V)AND (PEEK (U+N+2)
        =W)GOTO 64490
64475:U=U+N+5:IF U>&C348LET U=&B054:GOTO 6
        4460
64480:IF (U<&C053)AND (U>&B0E2)LPRINT "? "
        ;:Q=Q+2:C=C+2:RETURN
64485:GOTO 64460
64490:IF (C+N+1)>(84/Z)GOSUB 64500
64495:FOR J=1TO N:LPRINT CHR$ (PEEK (U+J))
        ;:NEXT J
64498:LPRINT " ";:C=C+N+1:Q=Q+2:RETURN
64500:GOSUB 64200:LPRINT " ";:C=C+7:
        RETURN

```

You may, however, elect to change the printing size to some other value. This is accomplished by changing the value for Z given in line 64025 of the program to the desired CSIZE value. For instance, changing Z to a value of 1 will result in CSIZE 1 being used for a listing. This size will permit up to 84 characters to

be printed per line and 20 lines to a "page." This size is particularly useful for archive listings.

By the way, when Sidelister starts to list a program, it will initially print "**TOP**" followed by the current date and time (as indicated by the TIME function value of the PC). This provides a means of keeping listings in chronological order in your files.

You will notice that the program is rather slow in operation. This is because it must translate machine code token values to their BASIC keyword equivalents. Each such translation can take a number of seconds, but many people find the wait worthwhile.

Program listing: SIDELISTER

```

64000: "L"REM  SIDE                               : GLCURSOR (X
      LISTER                                     , 0)
64025: Z=2: X=212-Z*                             64210: L=L+1: IF L=
      8                                           INT (20/Z)
64050: GRAPH :                                   GOSUB 64300
      GLCURSOR (X,                             64215: RETURN
      0): SORGN :                               64300: L=0: X=0: Y=- (
      ROTATE 1:                                Z*M*6.4):
      CSIZE Z                                  GLCURSOR (X,
64100: X=0: L=0: C=0:                            Y): SORGN : M=
      M=0: Q=STATUS                            0: RETURN
      2-STATUS 1:                             64400: IF PEEK Q>&E
      LPRINT "*TOP                             5GOTO 64450
      *"; TIME                                64405: IF C>(84/Z)
64105: GOSUB 64200                             GOSUB 64500
64110: IF PEEK (Q)=                             64410: LPRINT CHR$
      &FFEND                                   PEEK Q: C=C+1
64115: LPRINT USING                             : Q=Q+1:
      "#####";                               RETURN
      PEEK (Q)*256                             64450: U=PEEK (Q): W
      +PEEK (Q+1);                             =PEEK (Q+1):
      ": ";                                   U=&C053
64120: Q=Q+3: C=C+7                             64460: N=PEEK UAND
64125: IF PEEK (Q)=                             &0F
      &0DLET Q=Q+1                             64470: IF (PEEK (U+
      : GOTO 64105                             N+1)=U)AND (
64130: GOSUB 64400                             PEEK (U+N+2)
64135: GOTO 64125                             =W)GOTO 6449
64200: IF C>MLET M=                             0
      C: IF M>(84/Z                             64475: U=U+N+5: IF U
      )LET M=84/Z                             >&C348LET U=
64205: C=0: X=X-Z*10                           &B054: GOTO 6

```



```

        4460                                LPRINT CHR$
64480: IF (U<&C053)                        (PEEK (U+J))
        AND (U>&B0E2)                        ;:NEXT J
        )LPRINT "?"                          64498: LPRINT " ";:
        ;:Q=Q+2:C=C+                          C=C+N+1:Q=Q+
        2:RETURN                             2:RETURN
64485: GOTO 64460                          64500: GOSUB 64200:
64490: IF (C+N+1)>(                          LPRINT "
        84/Z)GOSUB 6                          ";:C=C+7:
        4500                                RETURN
64495: FOR J=1 TO N:                        STATUS 1          739

```

7.5 SORT

Norlin Rober contributed this version of a Shell/Walters sort routine for use on a PC. While this implementation is intended strictly for arranging numeric values, you could alter the program to sort character strings.

Using the program is a cinch. After it has been loaded into memory, place the PC into its RUN mode and start the program, using the *RUN* directive. Respond to the prompt for the total number of items to be sorted, and then enter the values to be sorted. When all the values have been inputted, the program will perform the sort operation. At its conclusion it will display each value in ascending order. Here is a sample run of the program for illustration :

```

NUMERIC SORT PROGRAM
# OF ITEMS TO SORT? 5
ITEM # 1> 18
ITEM # 2> 4
ITEM # 3> 99
ITEM # 4> 52
ITEM # 5> 1
SORT IN PROGRESS....
N( 1)= 1
N( 2)= 4
N( 3)= 18
N( 4)= 52
N( 5)= 99
END OF SORTED LIST

```

The maximum number of items you can sort is dictated by the *DIM*ension statement in line 10. You can increase the

number of array elements by changing this *DIMension* statement. Of course, the true limiting factor is the amount of memory you have in your PC. However, with an 8K memory module installed, you could conceivably sort on the order of a thousand or so numeric entries.

If you were to change the sort array (named "A") in the program to a string array (say, "A\$"), then you could use the program to arrange entries in alphabetical order. Of course, you would need to change every occurrence of A() to A\$() in the program. Furthermore, you would most likely want to define the number of characters permitted in each element of the string array. To define, say, 20 characters (maximum) per element, the *DIMension* statement would read as follows:

DIM A\$(100)*20

That would create a string array named A\$, having 100 usable elements, with 20 character positions allotted to each element. (While the array would have element numbers 0 to 100, the sort program would only utilize elements 1 to 100.) You would, in this instance, need at least a 4K memory module in order to store such a sizeable array.

Program listing: SORT

```

10: CLEAR : DIM A(1
    00): WAIT :
    PAUSE "NUMERIC
    SORT PROGRAM"
    : INPUT "# OF I
    TEMS TO SORT?
    "; N
20: FOR I=1 TO N:
    WAIT 0: PRINT "
    ITEM #"; I; :
    INPUT "> "; B: A
    (I)=B: CLS :
    NEXT I
30: PRINT "SORT IN
    PROGRESS...."
    ;
40: BEEP 1: C=N
50: C=INT (C/3):
    FOR I=1 TO N-C:
        IF A(I)<=A(I+C
        ) THEN 100
60: T=A(I+C): B=I
70: A(B+C)=A(B): B=
    B-C
80: IF B>0 IF T<A(B
    ) THEN 70
90: A(B+C)=T
100: NEXT I: IF C>1
    THEN 50
110: CLS : BEEP 3:
    WAIT : FOR I=1
    TO N: S=A(I):
    PRINT "N("; I; "
    )="; S: NEXT I
120: PRINT "END OF
    SORTED LIST"
130: END
STATUS 1 376

```

Notes ... Panel - ...

... of the program

Panel ...

130 ... to ... figures ...

... of the program

POCKET COMPUTER PROGRAMS

Nat Wadsworth

Use your pocket computer to get the edge in business, keep track of home activities, make engineering calculations — or while away a quiet moment with a game of pocket Blackjack or Hunt the Wumpus. You can even teach your pocket computer to play Bach.

Here are 50 thoroughly tested, ready-to-run programs written by experts throughout the world and compiled by the editor of the *Pocket Computer Newsletter*. Here's just a sample: Business programs keep personal calendars, calculate amortizations, monitor trip expenses, calculate payroll taxes. Math programs convert numbers to any base, solve polynomials and simultaneous equations... even do regression analyses. Engineering programs do fast Fourier transforms, design electronic filters, fit curves, plot histograms. Home and entertainment programs keep shopping lists straight, take memos, and play Hangman or Tic-Tac-Toe — or even Prelude II from the Well Tempered Clavier. And much, much more.

Each application includes an introduction by the author, a complete program listing, and a sample run. These programs have been tested on the Sharp PC-1500 and Radio Shack PC-2. Some programs may require a printer/plotter or a memory expansion module.

Also of Interest...

BASIC Computer Programs in Science and Engineering

Jules H. Gilder

Save time and money with this collection of 114 concise, ready-to-run BASIC programs for the hobbyist and engineer. Includes programs in statistics, differential and integral calculus, matrix mathematics, and data analysis. The fundamentals of electricity and electronics are covered by programs that calculate resistance, inductance, capacitance, and other factors for a wide range of common applications. For more advanced users, there are programs for designing amplifiers, power supplies, active and passive filters, attenuator pads, and communication system parameters. #0761-2, paper, 256 pages.



HAYDEN BOOK COMPANY, INC.

Rochelle Park, New Jersey

ISBN-8104-6283-4