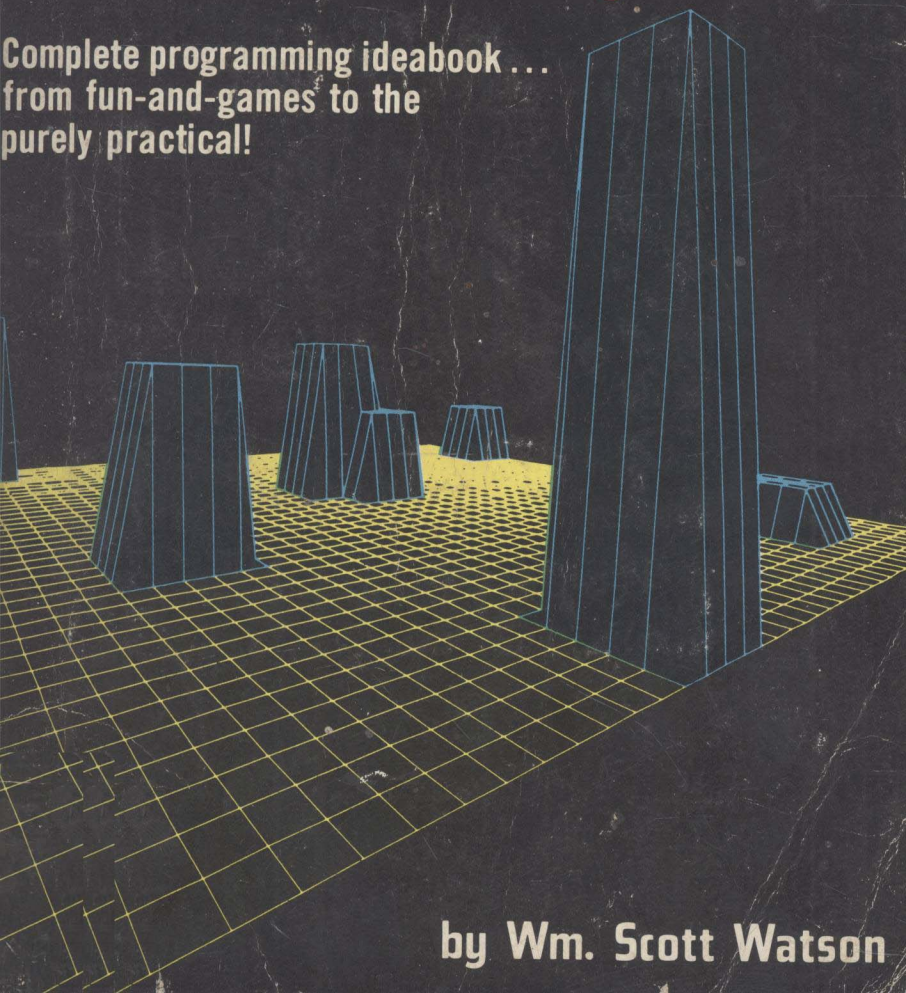
TAB 1195

$6.95

67 Ready-To-Run Programs in BASIC: graphics, home & business, education, games

Complete programming ideabook . . . from fun-and-games to the purely practical!



by Wm. Scott Watson

67 Ready-To-Run Programs In BASIC: graphics, home & business, education, games

Acknowledgements

Dianne Wiles-Watson for the artwork

Radio Shack- Thanks to Hy Siegel and Terry Phelps for the loan of the Screen Printer, as well as the entire company for giving us a darned good computer at a reasonable price.

John Watson - Who helped out with many of the business applications programs.

My Family - For tolerating me (most of the time) during this venture.

And finally,
Computer Center of the Central Missouri State University

67 Ready-To-Run Programs In BASIC: graphics, home & business, education, games

by Wm. Scott Watson

TAB BOOKS Inc.
BLUE RIDGE SUMMIT, PA. 17214

FIRST EDITION

SECOND PRINTING

Printed in the United States of America

Library of Congress Cataloging in Publication Data

Watson, William Scott
67 ready-to-run programs in BASIC
Includes index.
1. Computer programs. 2. Basic (Computer program language) I. Title.
QA76.6.W384 002.64'25 80-28282
ISBN 0-8306-9660-1
ISBN 0-8306-1195-9 (pbk.)

Contents

Introduction

Perhaps the most frustrating chore for the average microcomputer owner is finding an adequate supply of software. For many owners, the only apparent alternative to hiring a professional programmer is to rely on the preprogrammed or "canned" software available from the manufacturer and various mail-order companies. However, with a majority of preprogrammed software costing from two to twenty dollars for simple games and up to one hundred dollars for complex business management packages, it is easy to see why many owners simply cannot *afford* to build a large enough software library to fully utilize the potentials of their computers.

This book will help you build a satisfactory software library at a reasonable cost. Each program uses a low level form of BASIC and is designed to run on 4k of memory or less. Ultimately, the programs in this book should provide the microcomputer owner with many hours of use in various applications, for approximately the average price of a single canned program. You can bring Las Vegas to your home and gamble with the best without ever risking your paycheck, or, if you choose, race out to the far reaches of the galaxy to battle deadly enemies and still return home in time to balance your checkbook.

Wm. Scott Watson

Chapter 1
Book Basics

The programs in this book are arranged in four categories;

- Games,
- Graphics,
- Home and Business Management, and
- Educational.

In order to provide full aid to debugging and modification, each program is supported with full documentation. For each program listing, there is a section of REMARKS. The remarks section will list any specific information needed to use the program that is not mentioned in the program listing.

Also included in each program's documentation is a VARIABLE LIST. This list identifies all important variables, as well as their specific function. This section will be particularly useful to users who wish to study, modify, or simply change the level of difficulty of a program.

For users with intermediate programming skills, I have included a section of SUGGESTED VARIATIONS. Although this section is by no means comprehensive, there are enough modifications mentioned to increase the actual number of programs in this book to well over two hundred.

Finally, there is a SAMPLE RUN. If the user has typed a program in correctly, the sample run listed with each program will give a good indication of what to expect. Please take note that because of the lengthy repetition of many of the programs, the

sample runs are often not complete. Again, they should give a good indication of what to expect.

These programs should give you hours of practical use and entertainment. Doublecheck each program with the listing to insure that you have not made a typographical error. If all else fails and a program still will not run correctly, relax. Go get a drink of water, take the dog for a walk, and start again from scratch.

Once you have a program "up and running," you will probably want to save it on cassette tape (or whatever storage method you use). Use a good quality cassette, make two copies of each program, and you should not experience any retrieval problems.

Chapter 2
Program Language

All of the programs listed in this book are written using Radio Shack's TRS-80 Level 1 BASIC (*B*eginner's *A*ll-purpose *S*ymbolic *I*nstruction *C*ode). This version of BASIC was chosen because of its universality, as well as the ease in which this low level BASIC may be modified to run on other BASIC speaking computers. This chapter will deal with Level 1 BASIC in detail. Using the Level 1 BASIC guide below, the user can modify the listed programs to run on any of the popular BASICS with relatively little difficulty. If your computer does not use Level 1 BASIC, consult your user's manual to determine what modifications are necessary.

Although the programs in this book are listed using Radio Shack's Level 1 BASIC, the programs are by no means strictly limited to the TRS-80 (Fig. 2-1), Dartmouth BASIC, the father of all BASICS, including Radio Shack's Level 1, is a comprehensively defined computer language. If you were to study the BASICS utilized by various microcomputer brands, you would find that nearly all of the dissimilarities would fall into one of two categories:

- special commands or statements not included in Dartmouth BASIC, or
- differences of syntax usage in the various commands and statements.

All of the essential commands and statements used in the programs in this book are consistant with Dartmouth BASIC. Therefore, the

Fig. 2-1. Radio Shack TRS-80 Microcomputer System.

first category of conversion problems has been resolved. The following statement definition guide explicitly defines the syntax usage for commands and statements used in Level 1 BASIC. Therefore, the user, understanding the style of syntax used by their machine (which is usually described by materials supplied with the computer), will be able to adapt the syntax or "convert" the programs in this book for use on *any* BASIC speaking microcomputer on the market.

Only the program statements and intrinsic (library) functions used in the programs in this book are listed below. A complete summary of Level 1 BASIC can be found in Radio Shack's Level 1 BASIC User's Manual.

Statements

PRINT—Prints the value of a variable or whatever is contained within quotes. Example—10 PRINT "GAMES WON= ";A
INPUT—Allows data to be entered from the keyboard. Example—10 INPUT A. A PRINT statement may also be placed within an INPUT. Example—10 INPUT "HOW MANY GAMES DO YOU WISH TO PLAY";A
LET—(optional) Assigns a number to a variable. Example—10 LET A=5.

GOTO—Branches unconditionally to specified line number. Example—10 GOTO 150.
IF-THEN—Conditional test. Example—10 IF A=1 THEN B=B+1
IF-GOTO—Branches to specified line number only if conditional is true. Example—10 IF A=1 GOTO 150
IF-GOSUB—Branches to specified subroutine only if conditional is true. Example—10 IF A=1 GOSUB 150
FOR-NEXT—Performs a multi-statement loop. Example—10 FOR X=1T05 20 NEXT X
STEP—Increments a FOR-NEXT loop. Example—10 FOR X=1T05 STEP 2
END—Ends program execution. Example—10 END
GOSUB—Branches unconditionally to a specified subroutine. Example—10 GOSUB 150
RETURN—Ends subroutine and branches to statement following GOSUB.
ON-GOTO—Multi-statement branch. Example—10 ON X GOTO 10,20,30
ON-GOSUB—Multi-subroutine branch. Example—10 ON X GOSUB 10,20
CLS-Clears screen. Example—10 CLS
SET—Lights up graphic block at specified x,y coordinates. Example—10 SET (10,20)
RESET—Turns off graphic block at specified x,y coordinates. Example—10 RESET (10,20)
POINT—Interrogates graphic point at specified x,y coordinates. Returns a 1 if point is lit, or 0 if point is not lit. Example—10 IF POINT (10,20)=1 GOTO 150
PRINT AT—Begins printing at specified address. Example—10 PRINT AT 100, "100"
PRINT TAB (X)—Begins printing X spaces from left margin. Example—10 PRINT TAB (10); "10"

Intrinsic (Library) Functions

INT(X)—Returns the integer value of X—Example 10 X=INT(X).
ABS(X)—Returns the absolute value of X—Example 10 X=ABS(X).
RND(X)—When X is greater than zero, returns a number between 1 and X. When X equals zero, returns a number between 0 and 1. Example 10 X=RND(5) .

Arithmetic Operators

+	Addition
–	Subtraction
*	Multiplication
/	Division
=	Equal to
<>	Not equal to
<=	Less than or equal to
>=	Greater than or equal to
>	Greater than
<	Less than

Logical Operators

+	Or
*	And

Variables

Numeric—Letters A through Z.
String—A$, B$.
Array—A(X).

Chapter 3
Program Size

In keeping with the goal of universality, all of the programs in this book have been written to run (either directly or in some cases indirectly) on 4K or less of memory. If a program listing exceeds your computer's memory capacity, the following modifications should be made to the program. First, delete all unnecessary REM or remark statements you might have added to the program. Next, delete all spaces in the program statements that are not vital for syntax. This may make the program difficult to read and debug, but it does free a lot of memory space in many cases. Finally, if the above steps don't help, use abbreviations wherever possible. I have used abbreviations in many of the programs for such repetitive commands such as PRINT and GOTO, and if your computer will support abbreviations, it would be wise to get in the habit of using them.

If your computer will not support abbreviated statements, use the following list to modify the abbreviations in the program listings back to the original statement.

PRINT—P.
END—E.
THEN—T.
GOTO—G.
INPUT—IN.
FOR—F.
NEXT—N.
STEP—S.

TAB—T.
INT—I.
GOSUB—GOS.
RETURN—RET.
ABS—A.
RND—R.
SET—S.
RESET—R.
POINT—P.
PRINT AT—P.A.

Occasionally you might find that if you dress up a program with too many visuals, you will run out of memory quite rapidly. I have attempted to make these programs as eye appealing as feasible. If your computer has the extra memory capacity, don't be afraid to add some fancy graphics.

Chapter 4
Programming Tips, Inspirations, and Proverbs

You're on thin ice reusing cassettes. Cassettes are cheap. The time you spend retyping a lengthy program due to misalignment of your recorder's erase head isn't.

It may be trite, but its true—a program is never finished. There is always a better way to do what you have done.

If there is a users group in your area, I would recommend that you join it. If there isn't, see if you can't get one started. The amount of information floating around in these groups will make them a profitable experience for any computer owner. If you plan to start one, contact the manager of your local computer outlet. He should be more than willing to help, since an increased awareness of his product will ultimately mean more customers.

If you smoke while you use your computer, be careful that ashes do not fall between the keys. Ashes can cause a nasty keybounce.

Keybounce causes multiple characters to appear on the screen after typing only one key. It can be cured quite easily in most cases. First, remove the cap of the offending key(s). This usually can be done using a strong paper clip bent in the shape of a "J". Next, wipe the key's switch contacts gently with a cotton swab sprayed with a good television tuner cleaner/lubricant. Replace the key's cap and give 'er a try.

It is even easier to prevent keybounce than to cure it. A folded pillowcase makes an excellent keyboard cover.

An experienced programmer once told me: "A program should never run correctly the first time. If it does, there's something wrong that you've missed." Sounds logical to me.

If you store more than one program per cassette side, be sure to leave plenty of space between programs.

It isn't a good idea to copy or otherwise reproduce a copyrighted program except for your own use. Sitting in jail for selling a bootlegged Spacewars program is no way to waste your time.

Avoid using low quality tapes to store your programs on. Many times these cassettes will have dropouts in the ferrous recording material that will not store data properly.

When storing a program on tape, its a good idea to make two copies in case one doesn't record properly or is later damaged.

If you are in danger of power surges or interruptions as you program, make frequent copies as you progress. This will probably make more sense after you have lost a lengthy program due to a power failure.

Know when to stop. If you just can't seem to figure out a problem after an hour or so, work on something else for a while. The answer will probably come to you in the middle of the night.

Always trust hunches when you wake up in the middle of the night.

You really shouldn't discard any work you have done because of a seemingly unsolvable problem.

Even if your practical knowledge of computing is near zilch, chances are that your friends will think you avate garde for owning one.

Repeat after me: "Its only a stupid machine.. Its only a stupid machine...Its only a stupid machine."

Computer technology is a young science. There is no such thing as a computer expert, only men with various degrees of understanding. This is true no matter what the experts tell you.

If it is possible, don't sell your programs, barter them. Not only do both parties profit from a barter, its untaxable!

Strive to understand *all* of your system's capabilities. It is beyond the scope of this book to teach BASIC or programming skills. There are many fine books on the market to pick up where your instruction manual leaves off.

You should plan your program to achieve the desired result using as little memory as possible. This leaves plenty of room for modification and "dressing up" later.