COMMODORE 64 COLOR GRAPHICS: AN ADVANCED GUIDE

By Shaffer & Shaffer Applied Research & Development

APPENDIX G: TOOL KIT REFERENCE CARD

Under "How To Use" you will find that all variables are set equal to "#". See the back of this card for the value ranges allowed for each variable. Note also that most of the tools listed below require a DIM statement of:
"DIM P%(99,2),R%(99,2),L%(99,1), C(2,2),T(2,2),W(2,2)".

Tool	Description	How To Use
10	TURN ON GRAPHICS	MU=#: GOSUB 10
20	RETURN TO TEXT	GOSUB 20
30	CLEAR HIRES/MULTI	C=#: GOSUB 30
40	PLOT A POINT	X=#: Y=#: C=#: GOSUB 40
50	PLOT A LINE	X1=#: Y1=#: X2=# Y2=#: C=#: GOSUB 50
60	PAINT A SHAPE	X=#: Y=#: C=#: GOSUB 62 *or* PP=#: C=#: GOSUB 60
70	CLIP A SHAPE	(SEE TOOL 90)
80	DRAW A SHAPE	C=#: GOSUB 90 (SEE TOOL 800)
90	DRAW A SHAPE	ND=#: NL=$: C=#: MU=#. SEE TOOL 800.
100	APPLY TRANSFORMS	(SEE TOOL 90)
110	CLEAR C MATRIX	GOSUB 110
120	CLEAR T MATRIX	GOSUB 120
130	COMBINE MATRICES	(SEE TOOLS 140,150,160)
140	TRANSLATE A SHAPE	XT=#: YT=#: GOSUB 140
150	SCALE A SHAPE	XS=#: YS=#: GOSUB 150
160	ROTATE A SHAPE	RO=#: GOSUB 160
170	ZAP!	(DON'T USE WITHIN PROGRAM) *Type RUN 172*
180	TURN ON SPRITE SP	SP=#: GOSUB 180
190	TURN OFF SRITE SP	SP=#: GOSUB 190
200	X EXPAND SPRITE SP	SP=#: GOSUB 200
210	X UNEXPAND SPRITE SP	SP=#: GOSUB 210
220	Y EXPAND SPRITE SP	SP=#: GOSUB 220
230	Y UNEXPAND SPRITE SP	SP=#: GOSUB 230
240	SP PRIORITY OVER SHAPES	SP=#: GOSUB 240
250	SHAPE PRIORITY OVER SP	SP=#: GOSUB 250
260	SET SPRITE TO COLOR C	SP=#: C=#: GOSUB 260
270	PLACE SPRITE AT X,Y	X=#: Y=#: SP=#: GOSUB 270
280	MOVE SP FROM X1,Y1 TO X2,Y2	X1=#: Y1=#: X2=#: Y2=#: SP=#: SD=#: GOSUB 280
290	HOOK UP ACTION SPRITES	KB=#: P1=#: P2=#: M1=#: M2=#: T1=#: VE=#: GOSUB 290
300	COLLISION DETECTION	GOSUB 300
310	RESET COLLSION REGISTER	GOSUB 310
320	SUSPEND GAME	GOSUB 320
330	RESTART GAME	GOSUB 330
340	CRASH SOUND ON	GOSUB 340
350	SOUND OFF	GOSUB 350

360	COLLISION PUNISHMENT	SP=#: GOSUB 360
800	RETRIEVE A SHAPE	SE$=#: GOSUB 800
810	RETRIEVE A SPRITE	SE$=#: SP=#: GOSUB 810

Variable List

The following variables are commonly needed by this book's subroutine tools:

Variable	*Description*	*Value Range*
MU	Multicolor Indicator	0 = Hi-Res, 1 = Multicolor
C	Color	0 To 15
X	X Coordinate	0 To 319
Y	Y Coordinate	0 To 199
X1	X Coordinate Endpoint 1	0 To 319
Y1	Y Coordinate Endpoint 1	0 To 199
X2	X Coordinate Endpoint 2	0 To 319
Y2	Y Coordinate Endpoint 2	0 To 199
PP	Paint point coordinates' position in P% array	0 Based
XT	Translate Along X	————
YT	Translate Along Y	————
YS	Scale Along X	————
YS	Scale Along Y	————
RO	Rotation Degrees	————
SP	Sprite Number	0 To 7
KB	Keyboard Enable	0 = Joysticks, 1 = Keyboard
P1	Player 1 Enable	0 = Disable, 1 = Enable
P2	Player 2 Enable	0 = Disable, 1 = Enable
M1	Missile 1 Direction	0 = Disable, 1 = Up, 2 = Down, 4 = Left, 8 = Right, 5 = ↖, 6 = ↙, 9 = ↗, 10 = ↘.
M2	Missile 2 Direction	0 = Disable, 1 = Up, 2 = Down, 4 = Left, 8 = Right, 5 = ↖, 6 = ↙, 9 = ↗, 10 = ↘.
T1	Target Direction	0 = Disable, 1 = Up, 2 = Down, 4 = Left, 8 = Right, 5 = ↖, 6 = ↙, 9 = ↗, 10 = ↘.
VE	Game Speed	0 = Fastest, 65 = Slowest
SE$	Search String	————-

COMMODORE 64 COLOR GRAPHICS: AN ADVANCED GUIDE

By:
Shaffer & Shaffer Applied Research & Development

RESTON PUBLISHING COMPANY, INC.
A Prentice-Hall Company
Reston, Virginia

ACKNOWLEDGEMENTS

Special thanks and appreciation are extended to Penelope Semrau for developing the instructional concepts and graphic designs, to Jeffrey Young for creating the Commodore 64 advanced color graphics tool kit, and to Tamara L. Sullivan for writing the manuscript. This development team was supported by Sandra Locke, who produced the artwork; and Andrew Whitman, who tested and edited the manuscript. Thanks also goes to Kathy Planton for her contributions in the typing of the manuscript. All of us hope you'll enjoy learning more about the Commodore 64.

Daniel N. Shaffer
President,
Shaffer & Shaffer,
Applied Research &
Development, Inc.

General Editor
Robert P. Wells, Ph.D.

Graphics Production
Estela Montesinos
Steve Gunn

ISBN 0-8359-0787-2

TABLE OF CONTENTS

INTRODUCTION

Welcome to the world of Commodore 64 advanced graphics—a dynamic world of colors, imagination, intrigue, and, of course, fun. The fact that you purchased an advanced book suggests that you are already familiar with computer art, or the Commodore 64, or both. As we explain later in this section, some prior computer experience will be necessary.

If you got this book to learn advanced programming techniques for color graphics, you will appreciate the programs covered in these chapters. Programming code that can rotate, scale, and translate (move) images on the screen has been included for use in both simple and complex compositions. Also included are instructions on creating multi-colored images, as well as advanced methods of sprite manipulation. Carefully planned lessons will help you understand how and when you should use each new-found skill. The result will be pictures for school, work, entertainment—*anything.*

If your interest lies in learning more on the use of colors, tones, repetition, patterns, and other artistic techniques, you won't be disappointed. We've provided you with useful tips, suggestions, and facts to help you effectively put your ideas on the screen. This includes over 30 sketched designs illustrating how you can better take advantage of each graphic concept introduced.

The idea of this book is to go one step beyond the technical aspects of computer art. After answering the question "*How can I rotate a figure?*", it is just as important to answer the question "*What can I do with rotation?*" Exactly what doors are opened once the programs have been entered? That is what you will explore in the coming chapters.

What You Should Know

In order to write an advanced book, we've made some assumptions about the experience level of our readers. If you have already gone through our *Commodore 64 Color Graphics: A Beginner's Guide,* you are ready for this book.

If not, the first requirement is that you have a good feel for the Commodore 64 keyboard and its collection of special keys. In this text, special keys are printed in **boldface** to distinquish them from the rest of the text. So, for example, when you read "Press the **RETURN** key," you know to press the key marked RETURN on your keyboard (do *not* type R-E-T-U-R-N).

Your programming skills do not have to be extensive, but should include a first-hand knowledge of each item listed below (recommended reading is listed in the right-hand column):

	Commodore 64 User's Guide
—variables (e.g., A, B$, T%)	95-103,112-113
—PRINT statement	23-29,123-124
—GOTO statement	32-34,120
—GOSUB/RETURN statements	120,124

	Commodore 64 User's Guide
—GET statement	47-48,119-120
—POKE statement	60-61
—IF/THEN statements	37-39,120-121
—REM statement	124
—computer memory	2-4,62-65,142
—arrays (computer lists)	95-103
—FOR/NEXT loops	39-40,43-45,119,121-122
—SAVE statement (saving programs)	21,116
—LOAD statement	18-20,115

Finally, you should have some experience with high resolution graphics. You should know about foreground/background colors, screen color blocks, screen memory versus color memory, and the X,Y coordinate system. If you need a review of beginner's graphics, try our *Commodore 64 Color Graphics: A Beginner's Guide*, or the *Commodore 64 Programmer's Reference Guide.*

How to Use This Book

To use this book, you will need the following equipment:

—A Commodore 64 computer;

—A video monitor or TV screen (preferably color);

—A Commodore 64 disk drive with a properly formatted diskette, *or* a cassette recorder for the Commodore 64 with a blank cassette tape; and

—Some graph paper to work out your own designs (optional).

Each time you sit down to use this book, you should be at your computer. All equipment should be properly set up and turned on. Information on connecting your computer and monitor is provided in the Commodore 64 User's Guide. Disk drive installation is covered in the manual(s) provided with the disk drive itself. (This manual also covers formatting a diskette.) When the system is turned on, your screen should display "**** COMMODORE 64 BASIC V2 ****" at the top. Only then will you be ready to begin a session with this book.

A "session" can be as long or a short as you like. That is the beauty of working with programs. At the end of each session, simply save the current form of your program. You can then turn the computer off and take a nap, watch TV, or visit your friends. Later, you can easily return to your work by loading the program back into memory, and then picking up in the chapter where you left off.

Each chapter ends at a logical breaking point. This makes it easy for you to read a chapter, SAVE your program, take a break, and then continue later. For this reason, we ask you to SAVE your program at the end of each chapter. When you begin reading the next chapter, you are asked to LOAD the program back into memory. As a general rule, it is a good idea to SAVE and LOAD your program whenever instructed.

The general format of each chapter is as follows:

—New graphics and design concepts are introduced;

—New program lines are typed;
—The program is RUN and discussed in depth;
—Any additional design ideas and sketches are introduced where appropriate;
—All key technical and artistic points are summarized.

In the chapters, each programming technique is packaged as a useful subroutine "tool" that can be inserted and used in any picture-drawing program you create. In fact, by the end of Chapter 8 you will have a complete "tool kit" containing over 20 graphics subroutines. Need to draw a line? No problem. Just pick up the DRAW A LINE tool, specify where you want the line drawn, and the job is done. (This will become clear in Chapter 1.)

Another important aspect of this book is that it concentrates on teaching *how* pictures can be drawn on the Commodore 64. Often, knowing *why* things work is not essential to creating the picture.

Think of using your radio. You may not care why it works, just how it works (where the switch is). Beginning in Chapter 1, any "why" that is not necessary to understand has been placed in a box. These technical descriptions can be read or passed over, as you please. Passing over a technical description will in no way keep you from learning how to create your graphics displays.

As a final comment, practice what you learn before moving on from one chapter to the next, and do not skip chapters. If you have difficulty with some of the material, read through it again and re-try each example you are given. It will be through repetition that your skills are retained and refined.

What You Can Expect to Learn

This book sets out to accomplish two things:

(1) Provide you with advanced programming techniques for color graphics; and

(2) Show you how and where these techniques can be applied to produce more professional looking pieces of artwork.

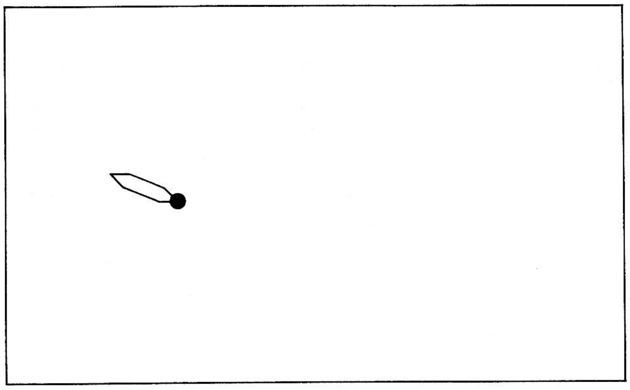
To accomplish the first of these goals, you will learn how to:

—plot points and lines*
—store and retrieve shapes
—draw shapes
—paint shapes
—translate shapes
—rotate shapes
—scale shapes

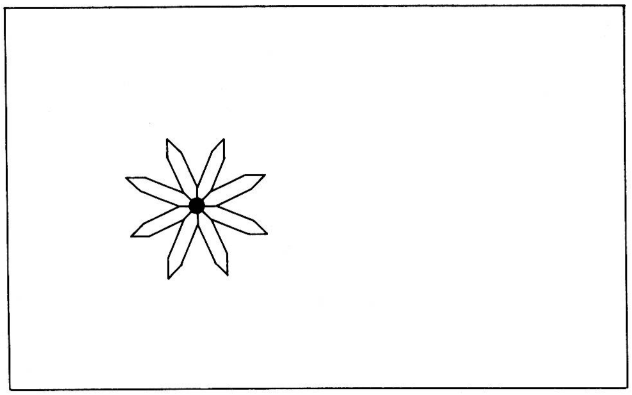
If you already have experience making and moving sprites but want to learn more about them, we also cover connecting sprites to joysticks, and sprite collision detection. Sprites are small, arcade-like figures that can move around on your screen. The ability to create moving designs is just one of the advantages computer art has over sketchpads and canvases.

*These are beginning graphics concepts, and are not discussed in as much detail as the others.

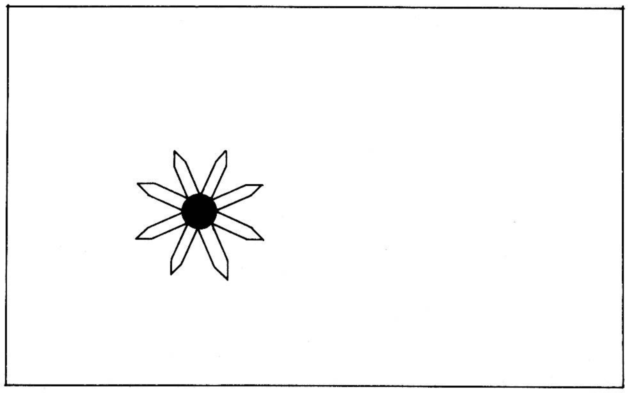
To give you an idea of what some of this means, consider the two basic shapes sketched below.



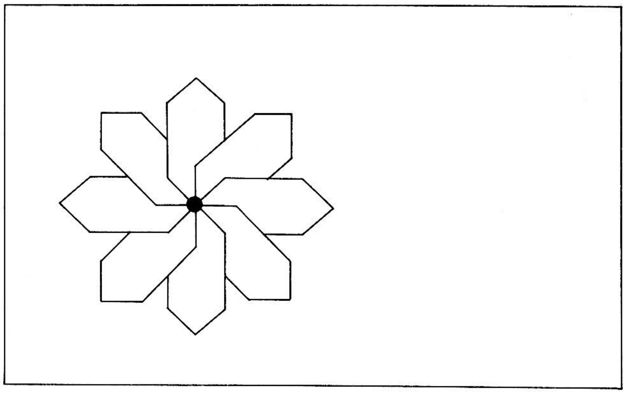
By rotating the petal shape, you arrive at a flower:



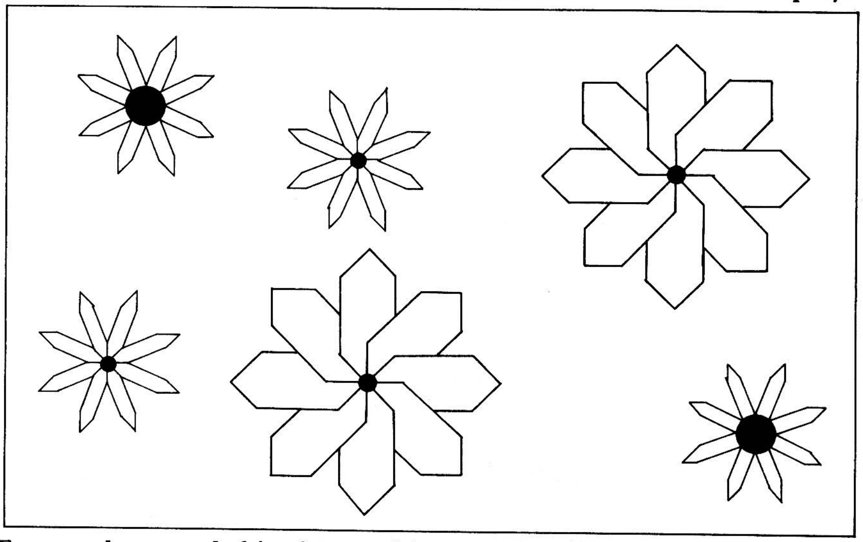
By scaling the center piece, you change the appearance of the flower:



By scaling the petals instead, you achieve another form of the flower:



Finally, by translating all of the flower types, you arrive at a floral display:



To meet the second objective, teaching art concepts, we gave special consideration to those art ideas that related specifically to our programs. Some of the topics discussed include:

—patterns
—repetition
—tone or "value" variation
—the illusion of depth
—the use of horizon lines
—variety through scaling
—using shapes to create other shapes
—the effect of shape placement/size.

For most people, drawing does not come naturally. Fortunately, there are most specific guidelines, "tricks of the trade" if you will, that are easy to learn, understand and apply. For example, you will see how a "horizon line" can significantly add to the feeling of depth in a picture. You will learn about "negative space," and why it is an important consideration in each of your designs. These and other simple facts about design control are discussed and illustrated as you proceed through the chapters.

Chapter One

SETTING UP THE PROGRAM

In order to work on advanced graphics, you have to start with some basic graphics tools. As fundamental as "plot a point" might be, there really can be no advanced graphics without it. In this chapter, we will set you up with tools that can do the following:

—"turn on" graphics mode
—"turn off" graphics mode
—clear the graphics screen and set the background color
—plot a point
—plot a line
—erase the main routine

There are several approaches to placing these tools in a program. We have taken the approach of creating a *subroutine* for each. This saves you the trouble of re-typing them every time you need one in your program. Instead, you set a few variables and insert a GOSUB. By the end of this book you will have a whole range of subroutine tools, ranging from the very simple to the very complex. The "main routine" of your program will vary from picture to picture, but the subroutines will remain the same.

We also chose to take advantage of *machine language*. If you've ever written BASIC programs that draw pictures, you are no doubt aware of the time it can take to run the finished program. This is because BASIC is not a language the computer immediately understands. Instead, it must first "translate" each BASIC statement into machinge language. Only then can it carry out the instructions it finds.

We felt that the time it takes to convert BASIC into machine language was too long for an advanced book. So, in the next section, you will enter some machine language as data statements to streamline and speed up a few of the slower tools. The result will be dramatic.

You will find that the main thrust of this chapter is to set you up for advanced graphics. This involves getting some beginning graphics programming and some machine language typing out of the way.

The Joy of Machine Language

This section's title expresses a mixture of both admiration and sarcasm. There's no doubt about it, nothing beats machine language for speed. Unfortunately, it is not nearly as simple to learn or understand as it is fast. This section does not attempt to explain any part of machine language to you. Instead, you will learn what to type, why it will help you, and how to check it.

1 SETTING UP THE PROGRAM

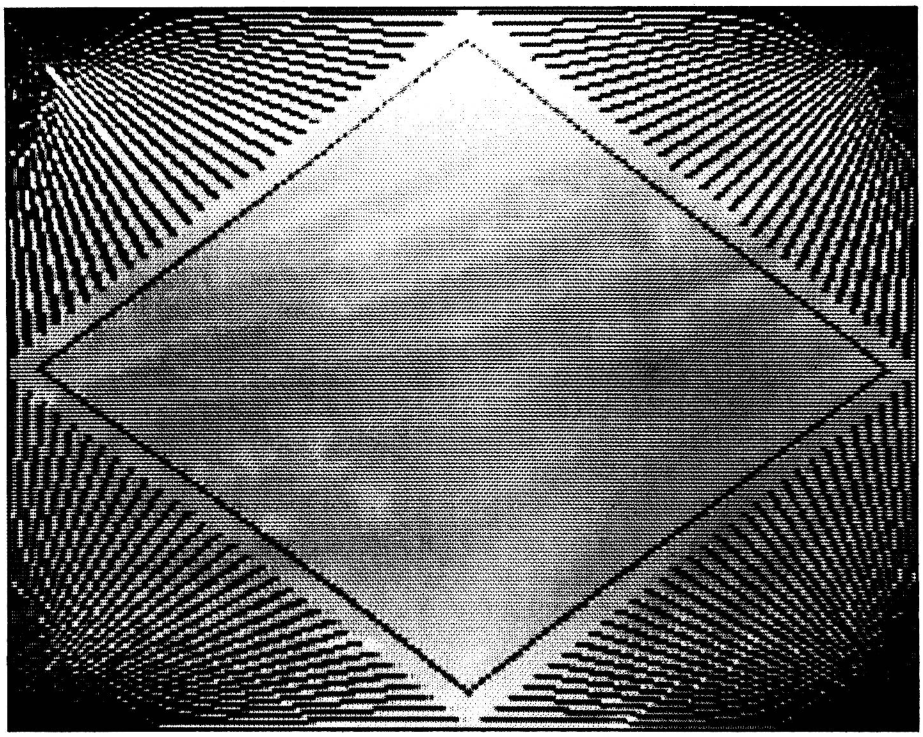
You will perform three steps to enter the machine language data. The steps are:

(1) Enter a small program to help you type the machine language.

(2) Entering the machine language data.

(3) Enter a program that double-checks the machine language for accuracy.

Initially, this may seem like a lot of work. However, spending 45 minutes of typing time now can save you hours of plotting time in the future. To instill enough incentive to get you through the next few pages, we have provided the picture below. You may need to flip back to it from time to time to keep yourself going. This relatively simple picture took 28 *minutes* to plot using BASIC alone, while taking only 41 *seconds* to plot using machine language.



The HELPER Program

Several hundred numbers need to be accurately POKEd into memory. This, needless to say, is quite a task. In addition, there will be many occasions when you need to PEEK into memory to check your entries, more typing.

To aid you in this process, we provide a HELPER program on page 13 that will do all of the repetitious typing for you. This will save you time and also reduce the possibility of typing errors. In addition, the HELPER program produces a "check number" after every eight pieces of data are entered. By comparing this number to one in our text, you can check to make sure you are entering the data correctly.

You will be told to SAVE this HELPER program after typing it. Be prepared with a formatted diskette or blank tape on hand. When you are ready, read the list of instructions below and then type the HELPER program on your Commodore.

—If you own a machine language monitor program that is easy to use *and* you understand how it works, you may use it instead of the HELPER program. If you don't know what a machine language monitor program is, it probably won't help you.

—Type slowly: accuracy is far more important than speed.

—Type in lower-case. This makes it easier to spot errors. To change to lower-case, hold down a **SHIFT** key and press **C=** (located on the lower, left-hand side of your keyboard).

—If you have trouble seeing your typing, press **CTRL** and **2** at the same time. This changes your typing to white.

—If you have a habit of typing oh's for zeroes, or small L's for ones, you must break that habit now. The computer expects numbers typed where numbers are intended.

—Carefully check over your typing when you are done.

Begin typing:

```
2000 REM :::::: HELPER PROGRAM
2010 PRINT CHR$(147) CHR$(18) SPC(15) "HELPER"
2020 A$="": INPUT "MEM/DATA"; A$: IF A$="" THEN END
2030 I=0: J=7: GOSUB 4030: REM GET ADDR
2040 ADDR=T: IF T<49152 OR T>50504 THEN PRINT
     "ERROR. TRY AGAIN.": GOTO 2020
2050 IF LEN(A$) = 28 THEN 3070: REM POKER
2060 IF LEN(A$)>4 THEN PRINT "ERROR. TRY AGAIN.":GOTO 2020
2070 CK=0
2080 FOR I = 0 TO 7
2090 PRINT " ";
3000 P%=PEEK(ADDR+I): CK=CK+P%
3010 PH% = P%/16: PL% = P%-PH%*16
3020 IF PH%>9 THEN PH%=PH%+7
3030 IF PL%>9 THEN PL%=PL%+7
3040 PRINT CHR$(PH%+48) CHR$(PL%+48);
3050 NEXT I:PRINT:PRINT"SUM FOR THIS ROW:" CK:PRINT
3060 GOTO 2020
3070 CK=0
3080 FOR J = 0 TO 7
3090 GOSUB 4030: CK=CK + T
4000 POKE AD+J,T
4010 NEXT J: PRINT"SUM FOR THIS ROW:" CK:PRINT
4020 GOTO 2020
4030 T=0
```

```
4040 I=I+1
4050 IF I>LEN(A$) AND J=7 THEN RETURN
4060 A=ASC(MID$(A$,I))
4070 IF A=32 THEN RETURN
4080 A=A+48*(A<58)
4090 A=A+55*(A>64)
5000 IF A<0 OR A>15 THEN PRINT"ERROR. TRY
     AGAIN.":GOTO 2020
5010 T=T*16+A
5020 GOTO 4040
```

Carefully double-check your typing when you are done, and then SAVE this program under the name "HELPER". After saving any program, always use the VERIFY command to make sure that the program *did* get saved. A summary of the SAVE, VERIFY, LOAD, and LIST commands is given below.

To SAVE on disk, type:	SAVE "*filename*",8
To SAVE on tape, type:	SAVE "*filename*",1
To VERIFY on disk, type:	VERIFY "*filename*",8
To VERIFY on tape, type:	VERIFY "*filename*",1
To LOAD from disk, type:	LOAD "*filename*",*8*
To LOAD from tape, type:	*LOAD "filename",1*
*To re-SAVE a program on disk, type:	*SAVE "@O:filename"*,8
to re-SAVE on tape:	*N/A. Save the revised program at the end of the tape.*

* The @0 command is one that allows you to erase and replace a program on diskette, using the same filename. This command has a history of problems, and we therefore do not recommend using it. An alternative is to re-name each modified or corrected program with a filename similar to the original program (i.e., "HELPER", "HELPER.1", "HELPER.2", etc.).

To use the above commands now, be sure to replace "*filename*" with "HELPER" (including quotes). When working with programs of your own, *filename* can be replaced with any 16-character name you wish to assign to the program.

If you are working with a cassette recorder, you will have to make use of your COUNTER with every SAVE, VERIFY, and LOAD command. In addition, the screen will present you with several messages as the commands are executed (e.g., PRESS PLAY AND RECORD). Follow the screen's instructions at all times. If nothing seems to be happening, try pressing C=. This keypress is necessary at certain times in the LOAD and VERIFY commands.

With the HELPER program safely stored on disk/tape, you can now try it out to see just exactly how helpful it is.