Learning the 6809

Micro Language Lab

Dennis Bathory Kitsz

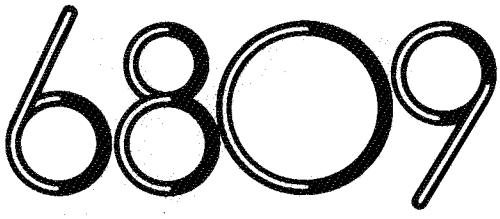
First Edition
Second Printing

Green Mountain Micro

Roxbury, Vermont

The Micro Language Lab:

Learning the



First Edition
First Printing
Printed in the United States of America.

ISBN 0—916015—00—9

Recording: Steve Lusk, Claire Manfredonia, JoAnn Trottier
Recording Supervision and Tape Editing: Dennis Bathory Kitsz
Illustrations: Jim (Doc) Holliday
Cover, Layout and Design: Dennis Kitsz
Typesetting: Northfield News; IJG Inc.
Additional Design and Preparation: M. J. Rufino Associates, Marie Lapre' Grabon
Sherlock Holmes: Peter Clarke
Dr. Watson: Kalvos Gesamte
Automobile: Steve's Honda
Marge: Claire Manfredonia
Cook: Steve Lusk
Mac: RB2—3
The Drivers: Kalvos Gesamte and JoAnn Trottier
Chocolate Cream Pie: Bev Fischer

Thanks to Jane and Ed Pincus, Chuck Trapp, Harv Pennington, Bruce Stuart, RB 2—3, Jim and Ingrid Wilson, Tom Bentley, Mary Bocage, Michael Rufino, Matthew and Gabriel and Beth Ann Betit, Paul Wiener

Notebooks by Mid-America Plastics

Cataloging Information:
Kitsz, Dennis Bathory, 1949
Learning the 6809. (The Micro Language Lab). viii, 222 pp.
Roxbury, Vermont, Green Mountain Micro: 1983.

Preface

When IBM introduced its Personal Computer with grand gestures and flourishes, the reviewers and the public seemed overwhelmed, as if in the presence of royalty. The PC's 16-bit microprocessor was revered and its BASIC praised, while its operating flaws were forgiven. Everyone seemed to say, "Good show, IBM. Wish we'd thought of that!"

Tandy Corporation doesn't have that classy IBM image. When Radio Shack introduced its Color Computer, hardly anyone noticed. It looked for all the world like another toy, said the critics.

Maybe Radio Shack needs to work on its grand gestures and flourishes a little harder. That toylike Color Computer appeared more than a year before that IBM PC. So although the microcomputing press pointed to the PC as innovative for including line, circle, draw and paint commands, they had conveniently overlooked that these same BASIC commands were actually introduced a year earlier on the Color Computer. And while critics talked about 16-bit processing power in the IBM machine, they had conveniently overlooked that both the PC and the Color Computer contain powerful 16-bit "internal" -- but 8-bit "external" -- microprocessors.

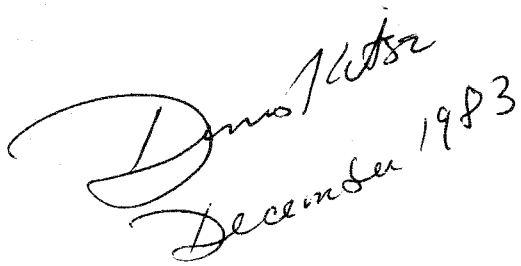
As I said, it's an image problem. The Color Computer, at one-quarter or less the cost of IBM's pricey PC, is the computing bargain of the early 1980s. And the heart of the bargain lies in the heart of the computer: the 6809 processor.

The 6809 is the Maserati of the 6800 family. It's fast, sleek and powerful. Almost anything any processor can do, the 6809 can do better. Its software capability is almost unrivaled in the 8-bit world, and its hardware features are stable and easily applied. Combined with its cousins -- the 6883 address processor, the 6847 video processor, and the 6821 interface circuit -- the 6809 creates a simple yet versatile personal computer. The Color computer is actually a practical computer application suggested by Motorola, the 6809's manufacturer.

"Learning the 6809" was created to fill a knowledge gap. The 6800 family hasn't produced any real "pop" processors. The 6502 achieved its glamour in the Apple, the Z80 became known through its presence in so many different TRS-80 computers. The 6809 looks different. It works in powerful ways which are, unfortunately, alien to users of 6502, Z80 or IBM-PC-style 8088 computing.

Be prepared to work hard; this course isn't an information giveaway. If you want to find out how to copy Joe's Lumbergrunters game, forget it; the answer won't be here. But you will be able to answer the question yourself by applying the knowledge, tools and techniques I present. This isn't "Using the 6809 to Learn the Color Computer" -- it's "Learning the 6809", where the Color Computer is the practical example. When you finish this series of tapes, you'll have the tools to explore the programming limits of the Color Computer, you'll be prepared for programming other 6809-based machines, and you'll be ready for the programming concepts and principles of the 68000 family of full 16-bit processors.

Work hard. With concentrated listening, by working out each example and by answering every question, these 24 half-hour lessons should take you anywhere from 50 to 100 hours to complete. By then, you'll be speaking 6809. Work, enjoy, and good luck.



Acknowledgments

It was midway through a long, bleak Vermont winter day spent with an incomprehensible microprocessor data book that I conceived of the Micro Language Lab. The data book made no sense to me. Engineers, I thought, don't speak English. No, I reconsidered, that's wrong. Engineers speak eloquently, but in an English far different from the rest of us. Just like musicians. And typographers. And artists. And priests.

A book was needed for 6809 users, and Color Computer owners in particular. I glanced at my library of programming books, looking desperately for ideas and inspiration. Nothing there. I couldn't think like Adam Osborne and I couldn't write like Bill Barden.

But talking was something fluid. Ideas that came to me easily when I was speaking would choke and gasp at my typing fingertips. Perhaps if I took microphone to hand, I could close my eyes and imagine a circle of anxious faces around me -- hanging on every word -- and the eloquence would begin...

The project got down to business at the same time Green Mountain Micro was established as my full-time occupation. I sat across from my old friend and business partner, RB2—3 (born with that name -- really!), and presented the idea. Sure, talk, great, he said, do it.

That was the easy part. The talking came quickly. But with me a musician and RB an artist, we found ourselves as babes in the business woods. We needed pretty notebooks, crates of cassettes, someone to print cassette labels and stick them on, a good and accurate typesetter, a nearby printer, recording and editing facilities, a duplicator, and a hundred sundries.

Everyone went to work. RB, our friends and new employees JoAnn and Steve, and my wife did the recording in my music studio. I edited the tapes onto the floor in a two-foot heap of gutteral stumbles and flubbering stutters. Graphics designers visiting from New York were ingloriously put to work on the layout. The typesetting was done very efficiently by computer connection to California, but on the trip back, the shiny (and expensive) new strips of typeset

got lost -- twice! -- in the back rooms at Federal Express. People (specifically me) got sick, the printer went on Christmas vacation, and our New York visitors escaped in the dark of the night.

Meanwhile, advertisements placed three months ahead of time began to appear. Faithful customers had placed orders for the holidays. We worked round-the-clock, only to have the last few weeks tumble into an abyss of chaos and exhaustion. We blew our deadline. As I write this, the final pieces fit together. The result is Learning the 6809, what I consider my -- and Green Mountain Micro's -- finest work.

During the craziness of preparation, our combination home and office took on the look of a factory as dedicated people traipsed in and out, crossing paths at 3:30 a.m. in 25-below winter weather. Those deserving my sincerest thanks:

-- RB2-3, for going along with the Micro Language Lab idea and for leaving me alone and phone-free for a whole month.

-- Jim (the Doctor) Holliday, for completing three hundred illustrations in a record two weeks; and Lynda, for not holding those all-nighters against us.

-- Mary Bocage and Michael Rufino, who escaped in the night leaving it all under control; Marie Lapre Grabon for finding it under control.

-- Chuck Trapp, for controlling those typesetting codes for three straight weeks and through two lost shipments; Harv Pennington, for delivering on the promise; Bruce Stuart for remaining cool; and Paul Wiener for half-duplex.

-- Jim Wilson, Tom Bentley, and M. Dickey Drysdale, all of whose last-minute cooperation alleviated the typesetting-in-Vermont syndrome.

-- JoAnn Trottier and Steve Lusk, who realized too late the meaning of "going on salary".

-- and for things many and varied: Claire, Peter Clarke, Deb Marshall, Charlie Freiberg, Claire, N. Spike Maggio, Gerald and Susan D'Amico, Cornelius ("the burritos are in") Murray, Claire, Tom Hardy of Motorola, Greg Keilty, those first faithful 80 customers, and Claire.

Contents:

1.

Hello. I'm Dennis Kitsz, your guide through the subminiature world of assembly language programming for the 6809 microprocessor. As you move with me through these new software concepts, I believe you'll constantly have mixed emotions. You'll likely find it rewarding . . . frustrating . . . enlightening . . . tedious — as well as very fast and powerful.

You probably know Color BASIC or Extended Color BASIC. But please start off learning with a blank slate; clear BASIC from your mind. Except for a few early examples, BASIC won't help you to learn 6809 assembly language. And, if you haven't found out already, you'll be surprised to discover how slowly BASIC really does work for you. On the other hand, it *is* a language that spoils you, with many convenient features, error messages, and programming prompts. By contrast, assembly language will at first seem the height of tedious absurdity. "All that just to clear the screen?", you will ask.

Don't worry. The feeling is almost universal. I'll admit right here that the breakthrough in learning assembly language for me took almost a year. There was no one to guide me. And because I remember that sense of frustration, I want to guide you.

If you're a newcomer to 6809, but know other processors, be prepared for some major differences in concept and approach. These are different languages we'll be working with. So whether you're a seasoned programmer or discovering assembly language for the first time, don't rush through these tapes; work with each one. Try every program. I've organized each lesson carefully so I won't waste your time, but even so, every concept will be presented and reinforced; most demonstration programs are provided on tape to save you the typing. So turn off the TV or radio, send the kids to bed, unhook the telephone, and pack the spouse off to bowling or a movie. More than anything else, assembly language takes concentration, the elimination of distractions, and — occasionally — the ability to suspend time and reality. Let me say part of that

This is the programmed learning section of the Micro Language Lab. In this column you will find questions and answers about the accompanying text in the form of quick questions. Also, your regular exercises and self-tests appear in this column. To make best use of these questions, start at the top of the page, and use a card to reveal each question but to cover the answer. Try to answer the question, and immediately compare your answer to the answer in the book.

For full use of the Micro Language Lab, follow these steps for each lesson: First, listen to the cassette tapes and follow along. Second, read the text and attempt the accompanying questions as you go along. Third, start over and attempt the questions by themselves. Repeat the second and third steps until you can answer all the questions without reference to the text. Then you are ready for the next lesson.

It works like this:

* How many steps are involved in using the Micro Language Lab programmed learning?

Three steps are involved in the programmed learning.

Requirements

*** What is the first of the three steps in the Micro Language Lab programmed learning?**

The first step is to listen to the cassette tapes.

*** What is the second of the three steps in the Micro Language Lab programmed learning?**

Read the text and try the questions.

*** What are the first two steps in the Micro Language Lab programmed learning?**

1. Listen to the cassette tapes. 2. Read the text and try the question.

*** What is the last of the three steps in the Micro Language Lab programmed learning?**

The third step is to learn the answers to the questions without referring to the text.

*** What are the three steps in the Micro Language Lab programmed learning?**

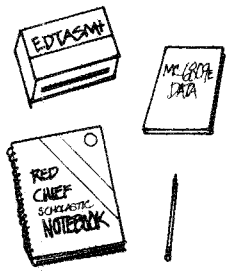
1. Listen to the cassette tapes. 2. Read the text and try the questions. 3. Learn the answers to all the questions.

So that's how it goes.

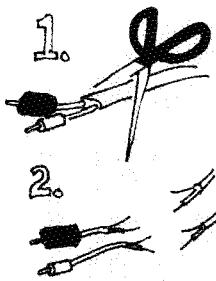
again. Assembly language takes concentration and the elimination of distractions.

There are also some things you will need for this course. You can't get along without an Editor/Assembler, so please don't try. Get one. Radio Shack calls this program EDTASM+, and it's available in a ROMpack cartridge for all the Color Computers. It contains an Editor/Assembler system, which I'll help you learn to use, a rundown of the 6809 instructions, and other pertinent information. All the sample programs are compatible with EDTASM+.

You will also need a machine-language software monitor. That's part of the EDTASM+ cartridge, but if as you progress you feel you need more features, then there are several excellent commercial programs available.

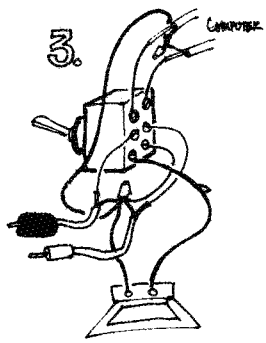


Blank cassettes are necessary only for saving original programs as you write them. You won't need blanks with this package to do any of the demonstration programs since everything is typed for you. But as you develop software, you may find that you like what you've done enough to keep it. For this you *will* need blank tapes.



Keep your Extended Color BASIC manual handy for reference, have paper and pencil ready, and take out the enclosed MC6809E data booklet and leave it nearby.

Finally, you will soon find that unplugging cables from your cassette player is no fun. Both my voice and all the programs are recorded together on these cassettes. Enclosed in this package are plans for a simple switch box so you can flip between listening to me and loading programs into your computer.

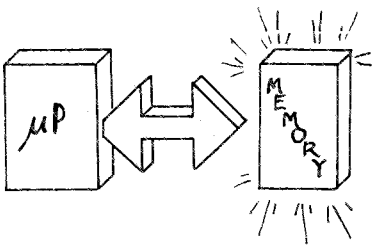


Support materials:

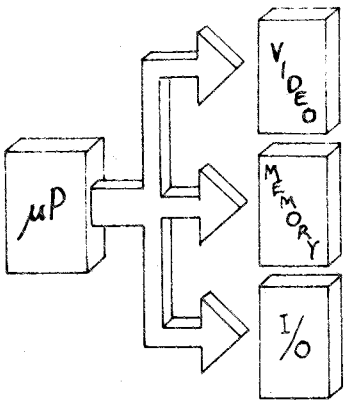
EDTASM+ and manual	RS Cat. No. 26-3250
Color Computer Technical Manual	RS Cat. No. 26-3193
Technical Manual Supplement	RS Parts No. 8749420
MC6809E data booklet (included)	Motorola DS9846-R1
MC6821 data booklet (included)	Motorola DS9435-R3
MC6847 data booklet (included)	Motorola DS9823
MC 6883 data booklet (included)	Motorola ADI-595R1

Now I want to tell you what you will be learning in this course. You will discover that assembly language is nothing like BASIC, but also that there are real advantages and disadvantages to using either one on the computer. You will learn binary and hexadecimal number systems, why they are needed at all, the ASCII codes, the job of the microprocessor, its architecture and timing, data flow, a little about how hardware relates to all of this, and lots of jargon. There will be lessons on memory maps, CPU control, input and output techniques, instruction sets, operation codes, instruction names, the inside and outside of the processor's world, and more jargon. Lots of demonstration programs will be provided, and in trying them you will learn how to use machine language monitors, editors, assemblers, and debugging techniques. Midway through the course, you will be learning all the different types of assembly language commands and their operation, how to use some subroutines already written for you in BASIC, the pitfalls of depending on that option, and more jargon. By the end of these tapes, you will be writing your own keyboard and screen subroutines, hopscotching data through memory, doing graphics and sound, and interfacing fast machine language with the simplicity of BASIC. And, of course, you'll be able to intimidate your friends with all the jargon you will use with such ease.

So now take some time to relax, clear your mind, and get set to begin learning 6809 assembly language programming. By the way, Claire is here to tell you exactly when to turn this tape on and off, when to load programs, and where to look in your booklet for your next instructions.



Let's get started. I've already said that the microprocessor's language is not BASIC. So what is it? Theoretically, that answer is simple. The microprocessor's language — the machine's language — is a set of binary signals which causes predictable electronic events to take place within a microprocessor and in relationship to its external memory, events which can be combined and expanded into control signals, mathematical calculations, video displays, and high-level languages like BASIC itself.



However, I'm not sure this definition is very a useful start. Let me try it from a different angle. Imagine your car is a computer. You unlock the door, open it, sit down, put on the seat belt, insert the key, start the ignition, release the brake, put the car in gear, let up on the clutch, step on the accelerator, turn the wheel, and off you go. That's BASIC.

Machine language takes you inside. You unlock the door by inserting a key whose ridges lift tumblers to specific heights, enabling a cylinder to turn inside a shell, releasing certain mechanical barriers. Open the door by pressing a button which engages some levers, slides and springs, allowing the door to be pulled out on hinges. The seat belt unrolls from a spring-loaded coil, perhaps turning off a small switch as it is pressed into a latch. Another key is for

* What is the first thing you will discover in this course?

That assembly language is nothing like BASIC.

* Name three other things you will learn in this course (there are several answers to this question)?

Number systems; architecture and timing; data flow ... or
Memory maps; instruction sets; operation codes ... or
Graphics; sound; jargon.

* Again, the first thing you will learn in this course is...

...that assembly language is nothing like BASIC.

* When you hear Claire's voice, she will tell you one of three things. What is the first one?

When to turn the tape on and off.

* Claire will tell you when to turn the tape on and off. What is another thing she may tell you?

When to load programs.

* Claire will tell you when to turn the tape on and off and when to load programs. What else may she tell you?

Where to look in your book for your instructions.

* What is another name for the microprocessor's language?

Another name for the microprocessor's language is machine language.

* How is knowing BASIC like driving a car?

Because both are simple to use but cause complex operations inside a machine.

Memory Map

* What do you call the description of how the computer's designers have arranged its memory?

A memory map.

* How many characters of memory does the normal display screen use?

512 characters.

* At what memory location does the normal display screen begin on the Color Computer?

At memory location 1024.

* How many memory locations are there in the Color Computer?

There are 65,536 memory locations in the Color Computer.

* What is the arrangement of these memory locations called?

The memory map.

* Where does the normal display screen begin in the memory map?

At location 1024.

* Where does the normal display screen end in the memory map?

At location 1535.

* How many memory locations does the normal Color Computer display screen use?

The screen uses 512 locations.

* How many memory locations are there altogether in the Color Computer memory map?

There are 65,536 locations in the memory map.

* What is the number of the first memory location?

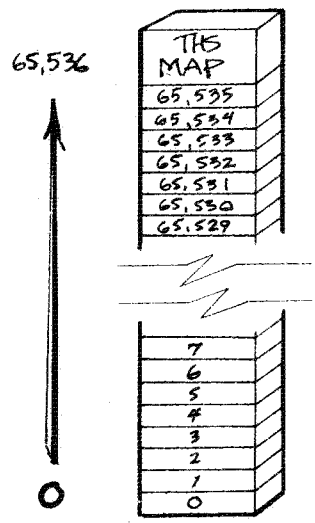
It is number 0 (zero).

another set of tumblers which releases a clamp on the steering wheel and permits electrical current to flow through engine components. Turning the key further sends electricity to an electromagnet, pulling a starter motor into position, rotating the starter motor, spitting high voltage through rotors, wires and spark plugs in a very precise order, sucking gasoline and air into engine cavities, consequently igniting the gasoline and air mixture, pushing pistons which, through mechanical linkages, rotate the engine's crankshaft. The rotation also activates a generator which, combined with those explosions, causes a self-sustaining repetition. Electrical and monitoring circuits are activated. You release the key and prepare to put the car in gear.

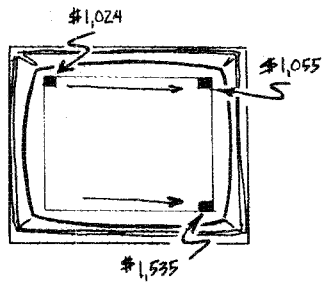
By now you get the idea. Getting into a car and driving away is a simple task for a modern American. Yet the number of machine-level activities that take place in that short span is enormous. When you enter "PRINT 3 + 4" and BASIC responds "7", that simple action represents an equally astounding number of machine-level activities: checking the entire keyboard for your typing, displaying your typing in the correct screen position, interpreting your commands and checking them for correctness, calculating the results, displaying the results, and returning for your next input. That's a summary of the thousands of steps involved. Machine language is working for you at all times.

Where is the machine language? How do you get to it? And how does it work? Some folks tell me that the "dot on the screen" example is shopworn. Well, get ready. Here it is again. For me, an intellectual understanding of a concept is seldom as effective as seeing or hearing something concrete. Throughout this course, visual and sonic examples will be used frequently — so you know you've "done something". So, putting a dot on the screen is the place to start.

To put that dot on the screen, you have to know where the screen is. The "where" is what's known as the computer's memory map. This map is a description of how the computer's designers have arranged its memory. I'll talk a great deal about memory maps later in this course, but for the moment let me tell you that the normal Color Computer screen occupies a block of memory 512 characters long beginning at memory location 1024 and running through memory location 1535. That's where it lies in the overall map of 65,536 memory locations.



So when you ask BASIC to PRINT on the screen, evaluations are made to determine the exact screen location that is available, and the information is subsequently placed in screen memory for you to see and read. We can emulate this process. Turn your computer on, and when "OK" appears, type POKE 1024,110. (Repeat) Press ENTER. Your screen should show a black dot in the upper left hand corner — an ordinary period, actually. You could just as easily PRINT this from BASIC. But now try this. Type POKE 1024,46 (repeat), and press ENTER.



Now there's a black box with a white dot — a reverse-video period. There's nothing you can PRINT from BASIC to produce that, because it's one of BASIC's non-printable codes.

Simple as that seems, this example represents just one of the hundreds of capabilities that machine language offers. In fact, there are 32 characters BASIC doesn't let you see. Have a look in this next example.

Program #1, a BASIC program. Turn on the power of your Extended Color BASIC computer. When the cursor appears, type CLOAD and press ENTER. The computer will search (S) and find (F). When the cursor reappears, LIST this program. If the program is not similar to the listing, or if an I/O error occurs, rewind to the start of the program and try again. For severe loading problems, see the Appendix.

```
10 CLS
20 PRINT"BASIC'S CHARACTER SET:"
30 FOR X = 0 TO 127
40 PRINT CHR$(X);
50 NEXT
60 PRINT:PRINT"THE WHOLE THING:"
70 FOR X = 0 TO 127
80 POKE 1216+X,X
90 NEXT
100 PRINT@448,"";
```

Run this program. You will see the 96 numbers, letters and symbols that BASIC can print. Below them you will see all 128 numbers, letters and symbols that your computer actually has available.

To summarize this program: BASIC prints its available characters, whereas the POKE statement manipulates memory to contain exactly what you wish.

The first advantage of machine language, then, will be to give you access to everything your computer has built into it, with no exceptions. Before I turn to another advantage, you should note now that the two sets of characters in the previous example are not displayed in the same order. I'll explain why later.

* Can you PRINT a reverse-video period on the screen using BASIC?

No, you can't PRINT a reverse-video period.

* What BASIC command do you use to display a reverse-video period?

POKE.

* What does POKE do?

POKE places a value directly in memory.

* How many characters can BASIC not display using PRINT?

32 characters cannot be displayed with PRINT.

* How many characters are available in the Color Computer?

128 characters are available.

* What command can display all 128 characters?

POKE.

* How does it display all 128 characters?

By directly manipulating display memory.

* What is the arrangement of memory locations called?

The memory map.

* Where does the normal Color Computer display screen start in this memory map?

At location 1024.

* What is the command for displaying value #111 at the first location in display memory?

POKE 1024,111

Program #2, a BASIC program. Turn on the power of your Extended Color BASIC computer. When the cursor appears, type CLOAD and press ENTER. The computer will search (S) and find (F). When the cursor reappears, LIST this program. If the program is not similar to the listing, or if an I/O error occurs, rewind to the start of the program and try again. For severe loading problems, see the Appendix.

```
10 CLS
20 INPUT"CHARACTER";A$
30 PRINT"PRINTING..."
40 GOSUB 440
50 CLS : GOSUB 440 : TIMER = 0
60 FOR X = 1 TO 511
70 PRINT A$;
80 NEXT
90 A = TIMER : GOSUB 440
100 GOSUB 460
110 GOSUB 440 : CLS
120 PRINT"PRINTING STRINGS..."
130 GOSUB 440 : CLS : TIMER = 0
140 FOR X = 1 TO 15
150 PRINT STRING$(32,A$);
160 NEXT
170 PRINT STRING$(31,A$);
180 A = TIMER : GOSUB 440
190 GOSUB 460
200 GOSUB 440
210 CLS
220 PRINT"POKING CHARACTERS..."
230 A = ASC(A$)
240 GOSUB 440 : CLS : TIMER = 0
250 FOR X = 0 TO 511
260 POKE 1024+X,A
270 NEXT
280 A = TIMER : GOSUB 440
290 GOSUB460
300 GOSUB 440
310 CLS
320 PRINT"MACHINE LANGUAGE..."
330 DATA BD,B3,ED,8E,04,00,E7,80,8C,06,00,26,F9,39
340 FOR X = 16000 TO 16013
350 READ B$ : A = VAL ("&H"+B$)
360 POKE X,A
370 DEFUSR0=16000
380 NEXT : TIMER = 0
390 A = USR0(ASC(A$))
400 A = TIMER : GOSUB 440
410 GOSUB460
420 GOSUB 440
430 END
440 FOR N = 1 TO 500 : NEXT
450 RETURN
460 CLS :PRINT"TIMER READS"A
470 GOSUB440
480 RETURN
```

*** What is the purpose of program #2?**

To fill the screen with a display 512 identical characters.

*** What are the four ways this program fills the screen with characters?**

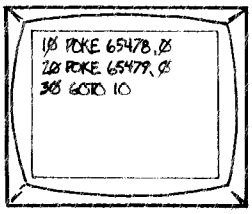
By PRINTing characters; by PRINTing strings; by POKEing values; by using machine language.

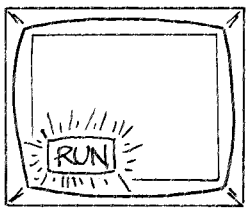
Welcome back. The program demonstrates the speed of 6809 assembly language. Its purpose is simply to fill the screen with 512 identical characters, which can be done in at least four ways: by printing 512 characters through BASIC, by printing strings of characters, by POKEing 512 characters from BASIC directly into screen memory, and by handing control over to a 6809 machine language program. RUN this program now.

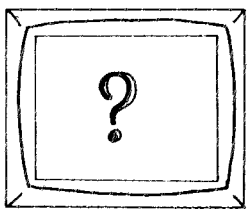
First, enter any uppercase letter from A to Z you wish displayed. Observe the BASIC printing technique. Notice the string printing method, which is quite fast. Now watch the BASIC POKEing technique. And finally, the machine language routine seems instantaneous.

Now there are three important things to notice. The first is the speed of the machine language program; don't miss that final display. Run the program again. This time, enter a number or punctuation mark as the character to be printed instead of a letter. Observe carefully as the printing and string printing finish that the LAST (512th) letter is missing. In BASIC, if you print in that 512th screen position, the screen automatically scrolls to the next line. But characters can be POKEd anywhere in memory, even in the last screen space. The machine language program is a fast way of doing that POKEing.

Yet there's something else. This time, the characters printed are not the same as those POKEd into memory or displayed by the machine language program. Recall the first program in this lesson — the characters weren't in the same order when printed and POKEd into memory. The reason is the hardware chosen to perform the video display. This hardware is limited to displaying only 64 characters — numbers, symbols, and uppercase letters. The Color Computer uses reverse (also called inverted) letters to represent lowercase. The BASIC software knows how to switch all these around to get the standard order — the order of ASCII, the American Standard Code for Information Interchange. This first, short machine language program doesn't do that. But it can be expanded. We'll return to that later.







The final lines of the BASIC program contain data statements and other commands which set up and execute a machine language program. Although you may examine these now, I'll hold back the detailed explanation of these for the moment.

So far, I've only played around with screen memory by putting some things on it. Now enter a three-line program; I'll read it to you. Line 10. POKE 65478,0. Line 20. POKE 65479,0. Line 30. GOTO 10. I'll repeat that; you can glance in the manual and check Program #3 to double-check.

```
10 POKE65478,0
20 POKE65479,0
30 GOTO10
```

RUN this program. What's that? It's delving into the heart of the computer, manipulating its control signals. It's video screen position information masquerading as computer memory. And that's the subject of the next lesson.

*** What does ASCII mean?**

American Standard Code for Information Interchange.

*** How does the Color Computer represent lowercase letters?**

Lowercase is represented by reverse video (white on black).

*** Are the internal (hardware) Color Computer characters in ASCII order?**

No.

*** Does PRINT display the characters in ASCII order?**

Yes.

*** Does POKE display the characters in ASCII order?**

No.

*** Why does PRINT display the characters in ASCII order?**

Because the BASIC software switches them.

*** What BASIC command is used to show the internal order of the characters?**

POKE.

*** What does POKE do?**

It places a value into memory.

*** What locations in the memory map does the normal Color Computer display screen use?**

From locations 1024 to 1535.

*** Of the four methods in Program #2 — PRINTing characters, PRINTing strings, POKEing, and machine language — which is fastest?**

Machine language is the fastest method.