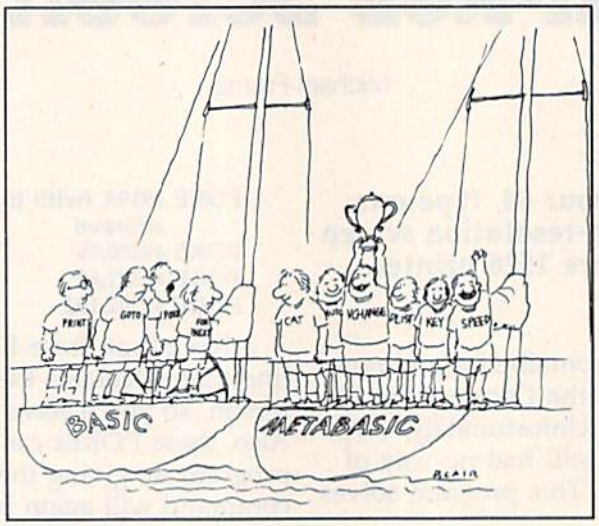
# MetaBASIC:
## Programmer's Problem Solver

Kevin Mykytyn, Editorial Programmer




**Here's a utility that will change the way you program. It adds 32 new debugging and testing commands to Commodore 64 BASIC, working by itself or in conjunction with a machine language monitor/assembler.**

You've bought your first car and it runs well. But when you take it out on the highway, you're dismayed to find that it won't go faster than 45 miles per hour. What do you do?

Take it to your favorite mechanic and he might give you three options: Remove the engine and replace it with a brand new one. Or add some fancy turbo-charging fuel-injected doohickeys to the engine you already have. Or, without adding anything, you could tune it up, using a special machine that measures the engine's performance.

### A BASIC Tune-Up

You can add new programming commands to your 64 in three similar ways. The first is to toss out BASIC and create a whole new language (a more powerful engine) based on your ideas of what a programming language should do.

The second method, a language extension, keeps BASIC but adds some new programming commands (for sound, high-res graphics, or other specialized functions). You keep the BASIC engine, but add some additional parts which make it work faster or more efficiently. *Simons' BASIC* and the *Super Expander 64* are examples of an extension.

The third way is like a tune-up which doesn't change the engine. You add direct mode commands for debugging. This is not a new language or even an extension of BASIC, it's more properly called a *development system* or *writing/debugging tool.* The new commands you add cannot be used inside a program, they work only in immediate mode.

New languages and extensions have several advantages. But they also have a major drawback: You have to load the language or extension *before* you load the main program, or the program just won't work.

The nice thing about a development system like "MetaBASIC" is that it's there when you need it, during the time you're writing and tuning up a program. But once you've finished the program, you don't need MetaBASIC to run it—you can disconnect the tune-up machine.

### An Introduction To MetaBASIC

MetaBASIC uses English mnemonics, so you don't have to memorize a lot of SYS numbers. And if you forget the new words, you can either refer back to this article or type HELP.

BASIC programmers have 12 new commands at their fingertips. For writing programs, AUTO, KEY, and UNNEW. For examining and altering programs, CHANGE, DELETE, FIND, RENUM, and VCHANGE. And DUMP, SPEED, TRACE, and TROFF help during debugging sessions.

If you're writing in machine language, you can use some of the BASIC problem solvers, as well as MEMORY, MONITOR, NUMBER, and @.

To control MetaBASIC, you have DEFAULT, HELP, INT, and QUIT.

Disk commands include BSAVE, CAT, DLIST, ERR, MERGE, READ, RESAVE, SCRATCH, SEND, and START.

Finally, there's LLIST if you have a printer, and TERMINAL if you have a modem.

## MetaBASIC Commands

Here's an alphabetical list of the new commands and how to use them, with examples. MetaBASIC commands and strings appear in **boldface** and numbers appear in *italics*. Anything enclosed in parentheses is optional.

If something is described as a disk command, it won't work unless you have a disk drive. However, some of the ML programming aids can be useful in BASIC and vice versa.

---

**AUTO**—BASIC Programming

Syntax: **AUTO** *startnum, increment*

AUTO can take some of the drudgery out of writing a program. It automatically numbers a program, starting at the first number, incrementing by the second. Separate the numbers with a comma. After you press RETURN over a line, the next number is automatically printed. The current line number can be changed by using the INST/DEL (delete) key and replacing it with another number.

Press RUN/STOP to escape from AUTO.

Example: **AUTO100,10** starts at 100 and numbers by 10.

---

**BSAVE**—Disk Command (see also RESAVE)

Syntax: **BSAVE "filename",** *start address, end address + 1*

BSAVE (Binary SAVE) saves a chunk of memory to disk, from the starting address to the ending address. Put the program name inside quotation marks, and use commas to separate the name, starting address, and ending address. It's important that you add one to the actual ending address. You can use this command to make backups of machine language programs, as long as you know the starting and ending addresses. BSAVE can also function to save sections of screen memory, custom character sets, or high-res screens.

The numbers should be in decimal. If you need to translate from hexadecimal to decimal, see NUMBER (below).

After you BSAVE to disk, you can load it back with LOAD *"filename"*,8,1.

Example: **BSAVE "METABASIC" ,36864 ,40961** to make a customized backup of MetaBASIC. By saving 100 bytes past the actual end of the program, you conveniently save all previously entered DEFAULT and KEY definitions. The next time you load the BSAVEd MetaBASIC, type INT and DEFAULT to regain them.

---

**CAT**—Disk Command (see also DLIST, READ)

Syntax: **CAT**

Anytime you want to look at the entire disk directory, use CAT (for CATalog). The BASIC program currently in memory will remain undisturbed. To see specific portions of the directory, see DLIST.

---

**CHANGE**—BASIC Programming (see also FIND, VCHANGE)

Syntax: **CHANGE @OLD@NEW@** (,*startnum, endnum*)

or **CHANGE @"OLD"@"NEW"@** (,*startnum, endnum*)

CHANGE searches through the program in memory, changing every occurrence of the old string to the new one. The strings can be up to 30 characters long, and must be bracketed by the commercial at sign (@). All lines in which changes are made are listed to the screen.

The first format will change BASIC commands and variable names. The second format should be used to change strings. If you omit the line numbers, CHANGE affects the whole program. If you want to change only one section, add the starting and ending line numbers, marked off by commas.

Example: **CHANGE @X@QQ@,1,200** changes the variable X to QQ in lines 1–200. To change the name Charles to John throughout the program, **CHANGE @"CHARLES"@"JOHN"@.**

---

**DEFAULT**—MetaBASIC Command (see also INT, QUIT)

Syntax: **DEFAULT** *border, background, text, device#*

When you hit RUN/STOP–RESTORE, the screen reverts to the default colors of light blue characters on a dark blue screen, whether you like it or not. And several commands like LOAD and SAVE default to tape. DEFAULT lets you change these values to whatever you prefer.

If you have a disk drive, you can change the device number to 8. If you want to use your second drive (device nine) for SAVEs, change the default to 9. If your 64 is hooked up to a black-and-white TV, change the character/background color to a more readable combination.

Note: You cannot use any of the new disk commands once you change the default device number to 1 (tape). To disable DEFAULT (and go back to normal), use the MONITOR

command below.

Example: **DEFAULT1,1,0,8** changes border and background to white, characters to black, and device number to 8. If you press RUN/STOP–RESTORE, you'll see black characters on a white background. And you'll be able to type SAVE"*filename*" (without adding a ,8).

---

**DELETE**—BASIC Programming

Syntax: **DELETE** *startnum-endnum*

DELETE removes a range of lines from your program. Separate the starting line number from the ending number with a dash (–).

Example: **DELETE200-250** erases lines 200–250.

---

**DLIST**—Disk Command (see also CAT, READ)

Syntax: **DLIST "filename"**

This command lists a BASIC program from disk to the screen, without affecting what's currently in memory. The program name must be enclosed in quotation marks. DLIST enables you to look at a program before using MERGE or SCRATCH.

It also allows you to read portions of the directory. DLIST "$:A*" displays all disk files beginning with the letter A.

Example: DLIST "BASICPROGRAM" reads the file from disk and lists it to the screen.

---

**DUMP**—BASIC Programming

Syntax: **DUMP**

Use DUMP to examine the current values of all non-array variables in a program. If the program is running, press RUN/STOP and type DUMP. To resume, type CONT.

---

**ERR**—Disk Command

Syntax: **ERR**

ERR reads the disk error channel. Use it when the red light on the disk drive starts blinking.

---

**FIND**—BASIC Programming (see also CHANGE, VCHANGE)

Syntax: **FIND @string@** (,*startnum*, *endnum*)
or **FIND @"string"@** (,*startnum*, *endnum*)

This allows you to find any word, variable, or other string within a program. Each line containing the search string is listed to the screen. If you wish to search just one section of the program, add the starting and ending line numbers, separated by commas.

If you're trying to find BASIC keywords (like PRINT or REM), use the first format. It also works for variables and numbers. The second format should be used when you're looking for strings or items inside quotation marks.

Example: **FIND @A=@** searches for lines where variable A is defined.

---

**HELP**—MetaBASIC Command

Syntax: **HELP**

Whenever you are unsure of the commands available in MetaBASIC, type HELP for a complete list.

---

**INT**—MetaBASIC Command (see also DEFAULT, QUIT)

Syntax: **INT**

Some features of MetaBASIC are interrupt-driven. If you reset the interrupts (with the MONITOR command), the function keys and the SPEED function may no longer work. INT puts the interrupts back in place.

---

**KEY**—BASIC Programming (see also INT)

Syntax: **KEY** *function#*, **"command or string"**

This command adds a lot of flexibility to MetaBASIC, allowing you to define each of the eight function keys as a different command or string.

The command, up to ten letters in length, must be inside quotation marks. There are two special characters. The back arrow acts as a carriage return, so you don't have to press RETURN after BASIC commands. Also, the apostrophe (SHIFT-7) counts as a double quotation mark.

Using KEY, you can load other utilities you may own, and SYS to them with a tap of a function key. Or you can do a one-key RUN or LIST.

If you want to permanently define the function keys and screen/text colors, you can use KEY and DEFAULT and then BSAVE "MetaBASIC" using the starting and ending addresses above. The definitions will be saved along with the program.

If the interrupts are accidentally reset, you may have to use the INT command to re-enable the KEY function.

Examples:

**KEY1,"{CLR}LIST100-←"** clears the screen and lists from line 100 on whenever you press f1 (the back arrow means RETURN will happen automatically). You could also abbreviate LIST with L SHIFT-I.

**KEY7,"DATA"** could be useful with automatic line numbering (see AUTO) if you're writing a program with a lot of DATA state-

ments. After entering a line, press RETURN and you'll see the next line number. Then press f7 and the word DATA automatically appears.

**KEY2,"VERIFY'*'←"** defines f2 to print VERIFY"*" plus a RETURN (note the apostrophes have been changed to quotation marks). If you've used DEFAULT to change the device number to 8, pressing f2 will automatically verify the program most recently saved to disk.

---

**LLIST**—Printer Command

Syntax: **LLIST** (*startnum-endnum*)

This command lists a program, but the listing is sent to a printer rather than to the screen. Line numbers are optional. The syntax for LLIST is identical to the regular LIST.

Example: **LLIST10-20** to list lines 10–20 to the printer.

---

**MEMORY**—ML Programming (see also @)

Syntax: **MEMORY** *start address (-end address)*

You can examine any section of memory with this command. Use decimal numbers (not hex) for the starting and ending addresses. The values in memory are displayed, six bytes per line, in decimal. In addition, the equivalent ASCII characters are printed in reverse to the right (if there's no corresponding ASCII character, a period is printed).

If you omit the ending address, MEMORY 43 for example, you'll see the contents of two bytes (43 and 44). This makes it easier to look at two byte pointers—like 43 and 44 which point to the beginning of BASIC memory.

To change memory, you can use the @ command, described below.

Example: Enter **MEMORY 41374-41474** and you'll see the first few error messages (note that the ASCII value of the last character is always added to 128). Or, load a BASIC program, and type **MEMORY 2048-2148** to see how programs are stored in memory.

---

**MERGE**—Disk Command

Syntax: **MERGE "program name"**

MERGE reads a program from disk, lists each line to the screen, and adds the line to the program in memory. If the programs have common line numbers, the program on disk takes precedence. Say they both contain a line 250. The line 250 from the disk program will replace line 250 in memory.

Before using this command, you may want to use DLIST to make sure you're merging the right program. And if there are conflicting line numbers, you can use RENUM to renumber one of the two programs. If you want to merge just part of one program, use DELETE to eliminate the unwanted lines.

---

**MONITOR**—ML Programming (see also INT)

Syntax: **MONITOR**

If you have a machine language monitor in memory, you can enter it with MONITOR (providing it is enabled with a BRK). To use MetaBASIC with a monitor, you must load MetaBASIC, type NEW, and SYS36864. Next, load the monitor, type NEW, and SYS to the starting address (which will set up the BRK vector to point to the monitor).

MONITOR does several other things, as well. It changes border, background, and text colors back to their default values (light blue on dark blue). It also sets interrupts to normal, which disables the function key definitions (see KEY) and SPEED command. You can get them back with the INT command.

---

**NUMBER**—ML Programming

Syntax: **NUMBER $***hexnum*
or **NUMBER** *decnum*

NUMBER allows you to convert back and forth between decimal and hexadecimal. Put a dollar sign ($) in front of hex numbers. In addition, the number is converted to low-byte/high-byte format (in decimal) and the equivalent binary number (marked by a percent sign).

Examples:

```
NUMBER $100
256
0 1
%100000000
NUMBER 34
$22
34 0
%100010
```

---

**QUIT**—MetaBASIC Command

Syntax: **QUIT**

This resets all vectors and disables all MetaBASIC commands. The one thing it does *not* do is restore the top of memory pointer. MetaBASIC is still protected from BASIC. Reenter the program with SYS36864 or SYS9*4096.

---

**READ**—Disk Command (see also CAT, DLIST)

Syntax: **READ "seq filename"**

READ allows you to examine sequential disk files. The information in the file is displayed to

the screen, without altering whatever program is in memory.

In the rare case that you want to use the BASIC READ command from direct mode (to see if all DATA statements have been read), you can precede it with a colon.

---

**RENUM**—BASIC Programming

Syntax: **RENUM** (*startnum*)(,*increment*)

This command renumbers the entire BASIC program in memory (you can't renumber just part of the program), starting at the specified line number. The increment size is optional; RENUM defaults to 10. If you omit the starting number, it will start at line 10.

In addition to renumbering BASIC lines, all references in GOTOs, GOSUBs, ON-GOTOs, ON-GOSUBs, IF-THENs, etc. are taken care of. One word of caution: GOTO is covered, but GO TO (with a space in the middle) is not. Use FIND before renumbering to look for occurrences of GO TO.

Example: **RENUM 100,20** renumbers a program, starting at line 100, counting up by 20s.

---

**RESAVE**—Disk Command (see also BSAVE)

Syntax: **RESAVE "filename"**

The disk command save-with-replace (SAVE "@:*filename*") first saves the program and then scratches the older version, so there must always be enough free space on the disk for the new version of the program. This can cause problems if you don't have enough available space.

Save-with-replace is also sometimes unreliable and should be avoided (although some experts dispute this).

RESAVE reverses the order—first it scratches the old version of your program from disk, and then does a regular SAVE, solving both of the above problems.

---

**SCRATCH**—Disk Command

Syntax: **SCRATCH "filename"**

SCRATCH does the same thing as OPEN 15,8,15: PRINT#15,"S0:*filename*": CLOSE 15, but it's easier to type. It scratches a file from the disk. If you have just inserted the disk into the drive, it's a good idea to initialize it first (see SEND). You can use wildcards to scratch more than one program—SCRATCH "A*" will get rid of all files beginning with the letter A.

Example: **SCRATCH "SPACEGAME"** removes the program named SPACEGAME from the disk.

---

**SEND**—Disk Command

Syntax: **SEND "disk command"**

This is a convenient way to send disk commands to channel 15. SEND"I0" initializes the drive, SEND"V0" validates the disk, SEND "R0:*newname*=*oldname*" renames a disk file, and so on. For more information about disk commands, see the 1541 User's Manual.

---

**SPEED**—BASIC Programming

Syntax: **SPEED** *number*

SPEED followed by a number from 0 to 255 changes the printing speed. The higher the number, the slower the speed. Try typing SPEED 255 (the slowest you can make it) and then list a program. You can get back to normal with SPEED 0. If it doesn't work, try using INT (see above) to correct the interrupts.

SPEED is also useful when you're using the TRACE command.

---

**START**—Disk Command

Syntax: **START "filename"**

If you forget where a machine language program begins, put the disk in the drive and use this command. This can help when you have forgotten the SYS that starts a program.

Example: **START "METABASIC"** should display 36864 on the screen.

---

**TERMINAL**—Modem Command

Syntax: **TERMINAL**

If you own a Commodore modem (and it's plugged into your 64), TERMINAL transforms your computer into a 300 baud "dumb" terminal you can use to talk to standard-ASCII bulletin boards or information services like CompuServe. You can't change any of the default parameters (like full-duplex), nor can you upload or download text or programs.

To return to BASIC, press the £ (English pound) key; do not press RUN/STOP–RESTORE. A note of caution: Memory locations 52736–53247 are used for buffers, so any program in this area will be overwritten.

---

**TRACE**—BASIC Programming (see also TROFF)

Syntax: **TRACE** followed by RUN.

If you're debugging a BASIC program, TRACE helps you see what's happening. As each line is executed, its line number is printed on the screen. Use the SHIFT or CTRL key to temporarily halt the program. SPEED controls the speed of execution, and TROFF turns off TRACE.

---

**TROFF**—BASIC Programming (see also TRACE)
Syntax: **TROFF**
This command turns off the TRACE function.

---

**UNNEW**—BASIC Programming
Syntax: **UNNEW**
You may never need this command, but it's nice to have it available. If you accidentally type NEW and you want to retrieve the program, use UNNEW to get it back.

---

**VCHANGE**—BASIC Programming (see also CHANGE, FIND)
Syntax: **VCHANGE @OLD@NEW@** (,*startnum*, *endnum*)
or **VCHANGE @"OLD"@"NEW"@** (,*startnum*, *endnum*)
VCHANGE (Verify CHANGE) works just like CHANGE (see above), except you get to choose whether or not the change is made. Each line containing the old string is displayed, with each occurrence of the string marked with a filled-in circle. If you press Y, the change is made. Press N if you want to skip to the next one.

---

**@**—ML Programming (see also MEMORY)
Syntax: **@** *start address*, *number*, *number*....
This works like POKE, except it allows you to put a series of numbers into consecutive memory locations. For example, if you want to change border and background color to white, you would use @53280,1,1. The first 1 goes into 53280, the second into 53281. If you add more numbers, separated by commas, they are POKEd into the next locations: 53282, 53283, and so on.

You can also use this in conjunction with MEMORY. First, PEEK at a series of locations using MEMORY. Then change the information there by putting @ before each line you want to change. Cursor over to the number you want to change, change it, and press RETURN.

---

## Typing It In

MetaBASIC is written entirely in machine language, and MLX is required to type it in.

If you don't already have a copy of MLX for the 64, type it in and save it to tape or disk.

The program resides at the top of memory, where BASIC programs (including MLX) store dynamic strings. To protect this section of memory, you must enter POKE644,144:SYS58260 before loading MLX. Otherwise, the variables will overwrite MetaBASIC. Then, load MLX and run it. Give it the following information:

Starting Address: 36864
Ending Address: 40805

Next, following the MLX instructions, enter MetaBASIC and save it.

To use MetaBASIC, follow these steps:

1. LOAD"MetaBASIC",8,1 (for disk) or LOAD"MetaBASIC",1,1 (tape).
2. Type NEW
3. SYS36864 (or SYS9*4096)

The program uses 4K at the top of BASIC memory (which leaves you with 35K for your programs). The first thing it does is move the top of BASIC pointer down, to protect itself from variables. After the SYS, it may seem that nothing is happening. But MetaBASIC is running in the background, and you now have 32 new commands to help you write and debug programs.

## Special Notes

Always type NEW after loading MetaBASIC.

One feature that works automatically is LIST Pause. When you're listing a program, hold down CTRL, SHIFT, or the Commodore key to temporarily halt it.

RUN/STOP–RESTORE is available in both program mode and direct mode. But if you want to interrupt any of the utilities like RENUM, use the RUN/STOP key by itself (not RUN/STOP–RESTORE).

The commands work only in direct mode; you cannot add them to programs. Also, you're limited to one command per line (although you can still use multi-statement lines inside your programs). Unlike ordinary BASIC commands, there are no abbreviations. You must type out the entire MetaBASIC command. If it seems to be working incorrectly, make sure the syntax is correct.

Machine language programmers should remember that MetaBASIC occupies memory locations $9000–9FFF. The 4K which begins at $C000 is available for programs like Micromon or for your own ML programs. Be sure to load and run MetaBASIC *before* loading any other programs. *See program listing on page 141.*