MetaBASIC

The Programmer's Assistant

"MetaBASIC" for the Commodore 64 is probably the most popular programming utility we've ever published in the GAZETTE. *So popular, in fact, that we're publishing it again for those who may have missed it in the April 1985 issue. If you already have the program, you'll want to try "MetaBASIC Plus," which adds 11 valuable, new commands. In addition, the version of MetaBASIC published here includes some minor modifications to the original. And for Commodore 128 owners, we've included "MetaBASIC 128." Originally published in* COMPUTE!'s First Book Of Commodore 128, *the version appearing here includes some modifications and corrections, courtesy of Jim Butterfield.*

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?

If you take it to your favorite mechanic, 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 Tuneup

You can add new programming commands to your 128 or 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. You keep the BASIC engine, but add some additional parts which make it work faster or more efficiently.

The third way is like a tuneup 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 tuneup machine.

Following are two versions of MetaBASIC—one for the 64, the other for the 128. Also, if you're using "MetaBASIC 64," be sure to read "MetaBASIC Plus" on page 77. "MetaBASIC 128" appears on page 79.

MetaBASIC 64

Kevin Mykytyn

This utility 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.

An Introduction To MetaBASIC 64

"MetaBASIC 64" commands use 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 are available. You can use CHANGE, DELETE, FIND, RENUM, and VCHANGE to examine and alter programs. 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 64, 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.

Special Notes

Always type NEW after loading MetaBASIC 64. 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 MetaBASIC command per line (although you can still use multistatement lines inside your programs). Unlike ordinary BASIC commands, there are no abbreviations. You must type out the entire MetaBASIC 64 command. If it seems to be working incorrectly, make sure the syntax is correct.

Machine language (ML) programmers should remember that MetaBASIC 64 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 64 *before* loading any other programs.

Typing It In

MetaBASIC 64 is written entirely in machine language, and "MLX," the machine language entry program found elsewhere in this issue, 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. Then, load MLX and run it. You'll first be asked for a starting address and an ending address for the data you'll be entering. For MetaBASIC 64, use the following values:

Starting address: 9000
Ending address: 9F67

Next, following the MLX instructions, enter the data for MetaBASIC 64 and save a copy.

To use MetaBASIC 64, follow these steps:

1. Load the program with a command of the form LOAD"METABASIC 64",8,1 (for disk) or LOAD"METABASIC 64",1,1 (for tape). Of course, you should replace METABASIC 64 with whatever name you used when you saved the MetaBASIC data
2. Type NEW
3. Activate the program with SYS 36864 (or SYS 9*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 has changed. But MetaBASIC 64 is active, and you now have 32 new commands to help you write and debug programs.

MetaBASIC 64 Commands

Here's an alphabetical list of the new commands and how to use them, with examples. In the descriptions of syntax, MetaBASIC 64 commands and mandatory parameters appear in boldface. String parameters appear in *italics*. Optional parameters appear in normal printing.

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.

@

Use: ML programming (see also MEMORY)
Syntax: **@ starting 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 colors 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, display the contents of a series of locations using MEMORY. Then change the information there by putting @ before each line you want to change. Cursor over to the numbers you want to change, change them, and press RETURN.

AUTO

Use: BASIC programming
Syntax: **AUTO** starting line number, increment

AUTO can take some of the drudgery out of writing a program. It automatically numbers a program, starting at the first number and incrementing by the second. Separate the numbers with a comma. If you do not specify a starting line number or increment, numbering will start at 5 and increment by 5 for each additional line. If you specify only a starting line number, then that value will also be used for the increment. 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: **AUTO 100,10** starts at 100 and numbers by 10.

BSAVE

Use: disk command (see also RESAVE)
Syntax: **BSAVE *"filename"*, starting address, ending 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-resolution screens.

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

After you BSAVE the contents of an area of memory to disk, you can load the data back in with LOAD *"filename"*,8,1.

Example: **BSAVE"METABASIC 64",36864,40805** makes a backup of MetaBASIC 64. To copy the first five lines of screen memory (locations 1024–1223) to disk, **BSAVE "SCREEN",1024,1224**. Screen memory does not include color information—that is stored in color memory and would have to be handled separately.

CAT

Use: 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

Use: BASIC programming (see also FIND, VCHANGE)
Syntax: **CHANGE @*old string*@*new string*@**, starting line, ending line
CHANGE @*"old string"*@*"new string"*@, starting line, ending line

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 keywords 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

Use: MetaBASIC 64 command (see also INT, QUIT)
Syntax: **DEFAULT** border color, background color, text color, device number

When you press RUN/STOP–RESTORE, the screen always reverts to the default colors of light blue characters on a dark blue screen. 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 have a second drive addressed as device 9 that you want to use for SAVEs, change the default to 9. If your 64 is hooked up to a black-and-white TV, change the character and background colors to a more readable combination.

Note: If you change the default device number to 1 (tape), you will be unable to use any of the new MetaBASIC disk commands. To disable the DEFAULT device number setting and go back to normal, use the MONITOR command below. Also, the TERMINAL command will not operate properly after DEFAULT has been used to change the device number. If you use DEFAULT, be sure to issue a MONITOR command before trying to use the TERMINAL command.

Example: **DEFAULT 1,1,0,8** changes the border and background colors to white, the character color to black, and the 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 save to disk by typing just SAVE*"filename"* (without adding a ,8).

DELETE

Use: BASIC programming
Syntax: **DELETE starting line–ending line**

DELETE removes a range of lines from your program. Separate the starting line number from the ending number with a dash (–). Use this command with extreme caution, since it is impossible to recover deleted program lines.

Example: **DELETE 200–250** erases all lines with line numbers in the range 200–250, including lines 200 and 250.

DLIST

Use: 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 "$0:A*" displays all disk files beginning with the letter A.

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

DUMP
Use: BASIC programming
Syntax: **DUMP**

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

ERR
Use: disk command
Syntax: **ERR**

ERR reads the disk drive error channel and displays the DOS error number and error message from the drive. Use it when the red light on the disk drive starts blinking to determine what caused the problem.

FIND
Use: BASIC programming (see also CHANGE, VCHANGE)
Syntax: **FIND @*string*@**, starting line, ending line
FIND @"*string*"@, starting line, ending line

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
Use: MetaBASIC 64 command
Syntax: **HELP**

Whenever you're unsure of the commands available in MetaBASIC 64, type HELP for a complete list.

INT
Use: MetaBASIC 64 command (see also DEFAULT, QUIT)
Syntax: **INT**

Some features of MetaBASIC 64 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 MetaBASIC interrupts back in place.

KEY
Use: BASIC programming (see also INT)
Syntax: **KEY key number, "*command or string*"**

This command adds a lot of flexibility to MetaBASIC 64, allowing you to define each of the eight function keys as a different command or string. (However, due to a minor bug in MetaBASIC, any definition you assign to the f8 key will be garbled whenever you use the RENUM command.) The command, up to ten letters in length for each key, 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 to set up the desired configuration, then save a copy of your customized version of MetaBASIC using BSAVE "METABASIC",36864,40960. The definitions will be saved along with the program.

If the interrupts are accidentally reset, you'll have to use the INT command to reenable the KEY function.

Examples:

KEY 1,"{CLR}LIST100-←" clears the screen and lists from line 100 on whenever you press f1 (the back arrow means RETURN will happen automatically). You can also abbreviate LIST with L SHIFT-I.

KEY 7,"DATA" could be useful with automatic line numbering (see AUTO) if you're writing a program with a lot of DATA statements. After entering a line, press RETURN and you'll see the next line number. Then press f7, and the word DATA automatically appears.

KEY 2,"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
Use: printer command
Syntax: **LLIST** starting line–ending line

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. As written, LLIST does the equivalent of OPEN 4,4,4 to open a file for output to the printer. Some printers may require a different secondary address (the last number in the OPEN statement)—OPEN 4,4,7, for example. To change the secondary address, POKE the desired value into location 40341. If you are using a printer with a different device number (5, for example) or a plotter (device 6), you can change the device number for LLIST by POKEing the desired value into location 40339. To make the changes permanent, follow the instructions for saving a new copy of MetaBASIC given above in the discussion of the KEY command.

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

MEMORY
Use: ML programming (see also @)
Syntax: **MEMORY starting address–ending 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 above.

Example: Enter **MEMORY 41374–41474** and you'll see the first few error messages in BASIC ROM (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
Use: 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 both programs 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
Use: 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 when a BRK instruction is executed). To use MetaBASIC 64 with a monitor, you must load MetaBASIC 64, type NEW, and activate the program with SYS 36864. Next, load the monitor, type NEW, and SYS to the starting address for the monitor (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 resets the default device number and 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
Use: ML programming
Syntax: **NUMBER $hexadecimal number**
NUMBER decimal number

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

Examples:
NUMBER $100
256
0 1
%100000000

NUMBER 34
$22
34 0
%100010

QUIT
Use: MetaBASIC 64 command
Syntax: **QUIT**

This resets all vectors and disables all MetaBASIC 64 commands. The one thing it does *not* do is restore the top-of-memory pointer. MetaBASIC 64 is still protected from BASIC. Reactivate MetaBASIC with SYS 36864 or SYS 9*4096.

READ
Use: disk command (see also CAT, DLIST)
Syntax: **READ *"sequential 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 statement in direct mode (to see if all DATA statements have been read, for example), you can precede it with a colon.

RENUM
Use: BASIC programming
Syntax: **RENUM** starting line, increment

This command renumbers the entire BASIC program in memory (you can't renumber just part of the program). The first line of the renumbered program will be given the specified starting line number. If you omit the starting line number, the renumbered program will begin with line 10. The increment value specifies how much the starting value will be incremented for each succeeding line. If no increment value is provided, the value defaults to 10.

In addition to renumbering BASIC lines, all references in GOTOs, GOSUBs, ON-GOTOs, ON-GOSUBs, IF-THENs, and so forth 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
Use: disk command (see also BSAVE)
Syntax: **RESAVE *"filename"***

The save-with-replace disk command (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. The save-with-replace command is also sometimes unreliable and should be avoided.

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
Use: 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*. However, you should use such commands with care to avoid accidentally deleting important programs.

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

SEND
Use: disk command
Syntax: **SEND *"command string"***

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
Use: BASIC programming
Syntax: **SPEED number**

SPEED changes the rate at which the 64 prints to the screen. The number supplied with the command must be in the range 0–255. The higher the number, the slower the printing 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 useful when you're using the TRACE command.

START
Use: 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. If this command returns the value 2049, the file you are checking is probably BASIC rather than machine language (or it at least has a single line of BASIC, like *SpeedScript*).

Example: **START "METABASIC 64"** should display 36864 on the screen.

TERMINAL
Use: 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, 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 ($CE00–$CFFF) are used for buffers, so any program in this area will be overwritten.

TRACE
Use: BASIC programming (see also TROFF)
Syntax: **TRACE**

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 keys to temporarily halt the program. SPEED controls the speed of execution, and TROFF turns off TRACE.

TROFF
Use: BASIC programming (see also TRACE)
Syntax: **TROFF**

This command turns off the TRACE function.

UNNEW
Use: 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
Use: BASIC programming (see also CHANGE, FIND)
Syntax: **VCHANGE @*old string*-@*new string*@**, starting line, ending line

VCHANGE @"*old string*-"@"*new string*"@, starting line, ending line

VCHANGE (Verify CHANGE) works just like CHANGE (see above), except you get to choose whether or not each 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 occurrence of the old string.

See program listing on page 99.

MetaBASIC Plus

John Brox Shadle

"MetaBASIC Plus" is a companion program to "MetaBASIC 64." It adds 11 new commands and modifies HELP to print the new MetaBASIC Plus commands in addition to the original MetaBASIC commands. To create MetaBASIC Plus, you must have a working copy of MetaBASIC 64. If you don't already have a copy, see the "MetaBASIC 64" article on the preceding pages. After you've entered MetaBASIC 64, return to this article for instructions for adding the enhanced commands.

Creating MetaBASIC Plus

MetaBASIC Plus is a collection of routines to add new commands to MetaBASIC 64. Like the original program, the MetaBASIC Plus routines are written in machine language and must be entered using the "MLX" machine language entry program found elsewhere in this issue. When you run MLX, you'll be asked for a starting address and an ending address for the data you'll be entering. For MetaBASIC Plus, use the following values:

Starting address: 8936
Ending address: 8F15

After you've entered all the data for MetaBASIC Plus and saved a copy, you're ready to create a new copy of MetaBASIC that includes the additional commands. Follow these steps carefully:

1. Load MetaBASIC 64 using a command of the form LOAD "METABASIC 64",8,1 (for tape, use ,1,1 instead).
2. Load MetaBASIC Plus using a command of the form LOAD "METABASIC PLUS",8,1 (again, use ,1,1 for tape).
3. Enter a NEW command to reset memory pointers.
4. Type SYS 35126 and press RETURN. This calls a short (19-byte) routine at the beginning of MetaBASIC Plus which patches MetaBASIC Plus into the original MetaBASIC and activates the combined programs.

Now you're ready to save a copy of the new version of MetaBASIC, which has the additional MetaBASIC Plus commands. Before doing so, however, you might want to use the KEY command to set up some default function-key definitions that will be enabled whenever you activate the new MetaBASIC. When you're ready to save a new copy to disk, simply use a command of the form

```
BSAVE "METABASIC+",35145,40960
```

For tape, the procedure is a bit more complicated. Use the following statements:

```
POKE 43,73: POKE 44,137: POKE 45,0:
  POKE 46,160: SAVE "METABASIC+"
  ,1,1
```

Once you've saved a copy of the combined file, load and activate the new version of MetaBASIC just like you did the old version, with LOAD "METABASIC+",8,1 (or ,1,1) and SYS 36864 (or SYS 9*4096).

MetaBASIC 128

Kevin Mykytyn

"MetaBASIC 128," will change the way you program. It adds 11 new debugging and testing commands to BASIC 7.0—and these commands are instantly at your fingertips for programming sessions.

Using MetaBASIC 128

MetaBASIC 128 commands use English mnemonics, so you don't have to memorize a lot of SYS numbers. Once MetaBASIC 128 is active, you'll have these 11 additional commands: AID, CHANGE, DEFAULT, DLIST, FIND, MERGE, QUIT, READ, RESAVE, START, and UNNEW.

The commands work only in direct mode; you cannot add them to programs. Also, you're limited to one MetaBASIC command per line (although you can still use multistatement lines inside your programs). Unlike ordinary BASIC commands, there are no abbreviations. You must type out the entire MetaBASIC 128 command. If you wish to stop the execution of a command, press the RUN/STOP key (*not* RUN/STOP–RESTORE). If it seems to be working incorrectly, make sure the syntax is correct.

Machine language programmers should remember that MetaBASIC 128 occupies memory locations $1300–$18BF (4864–6335) and uses zero-page locations $FB–$FE (251–254) and $AC–$AF (172–175).

Typing It In

MetaBASIC 128 is written entirely in machine language, and "MLX," the machine language entry program found elsewhere in this issue, is required to type it in. Be sure to read and understand MLX before typing in MetaBASIC 128. After loading and running MLX, you'll be asked for a starting and ending address. The correct values for MetaBASIC 128 are:

Starting address: 1300
Ending address: 188F

Next, following the MLX instructions, type in MetaBASIC 128 and save a copy.

To use MetaBASIC 128, follow these steps:

1. For disk, load the program with a statement of the form BLOAD "METABASIC128". For tape, use LOAD"METABASIC128",1,1. After a tape load, you must also type NEW to reset memory pointers.
2. Type SYS 4864 to activate MetaBASIC 128.

After the SYS, it may seem that nothing has changed. But MetaBASIC 128 is active, and you now have 11 new commands to help you write and debug programs.

MetaBASIC 128 Commands

Here's an alphabetical list of the new commands and how to use them, with examples. MetaBASIC 128 commands and mandatory parameters appear in boldface. String parameters appear in italics. Optional parameters appear in normal print.

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.

AID

Syntax: **AID**

Lists all available MetaBASIC 128 commands.

CHANGE

Syntax: **CHANGE @***old string***@***new string***@**, starting line, ending line
CHANGE @*old string***@***new string***@**,**,**starting line
CHANGE @*old string***@***new string***@**,,ending line
CHANGE /*old string***/***new string***/**, starting line, ending line
CHANGE /*old string***/***new string***/**, starting line
CHANGE /*old string***/***new string***/**,,ending line

See also FIND.

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 (@) or the slash (/). All lines in which changes are made are listed to the screen. The format with @ is the tokenized form and should be used to change BASIC commands and variable names. The ASCII form (the slash format) is useful when you want to change a word in a string without changing keywords. For example:

CHANGE /PRINT/WRITE/

changes all occurrences of the word *PRINT* within quotation marks without changing any PRINT statements.

Use the slash format to change anything inside quotation marks or after a REM statement; use the at sign format to change anything not inside quotation marks or after a REM statement. Remember that mathematical operators within programs such as +, −, *, /, >, <, and = are stored as tokens, not characters, so you must use the @ format when searching for one of these.

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@ „200** changes the variable X to QQ in all lines up to and including 200. To change the name Charles to John throughout the program, **CHANGE /CHARLES/JOHN/**.

DEFAULT
Syntax: **DEFAULT border color, background color, text color**

See also QUIT.

When you press RUN/STOP–RESTORE, the screen reverts to the default colors light green and black. DEFAULT lets you change these values to whatever you prefer. If your 128 is hooked up to a black-and-white TV, change the character and background colors to a more readable combination. The border- and background-color changes affect only the 40-column screen; the text-color change affects both the 40- and 80-column displays.

To disable DEFAULT (and go back to normal colors), use the QUIT command.

Example: **DEFAULT 1,1,0** changes border and background to white, and characters to black. If you press RUN/STOP–RESTORE, you'll see black characters on a white background.

DLIST
Syntax: **DLIST** *"filename"*

See also READ.

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.

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

FIND
Syntax: **FIND @*string*@**, starting line, ending line
FIND @*string*@, starting line
FIND @*string*@„ ending line
FIND /*string*/, starting line, ending line
FIND /*string*/, starting line
FIND /*string*/„ ending line

See also CHANGE.

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 with the @ symbols. 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.

MERGE
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 both programs 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 RENUMBER to renumber one of the two programs. If you want to merge just part of one program, use DELETE to eliminate the unwanted lines.

QUIT
Syntax: **QUIT**

This resets all vectors and disables all MetaBASIC commands. MetaBASIC is still protected from BASIC. Reactivate MetaBASIC with SYS 4864.

READ
Syntax: **READ** *"filename"*

See also DLIST.

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 statement in direct mode (to see if all DATA statements have been read, for example), you can precede it with a colon to distinguish it from MetaBASIC 128's READ command.

RESAVE
Syntax: **RESAVE** *"filename"*

The save-with-replace disk command (SAVE "@0:*filename*") first saves the new version of the program and then scratches the older version, so there must always be enough free space on the disk for both versions. Thus, the command can cause problems if you don't have enough available disk space for the new version. The save-with-replace command also has other problems and is best avoided.

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

START
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. If the command returns a value of 7169, the program is probably BASIC or a machine language program with a single BASIC line so that it starts with RUN rather than SYS.

Example: **START "METABASIC128"** should display 4864 on the screen.

UNNEW
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.

See program listing on page 102.