HUMBUG

HUMBUG is a machine language monitor and debugging system available for many 6800 and 6809 based computers. This manual describes the specific version for the Color Computer.

As it comes out of the factory, the Color Computer is set up to run programs mostly written in the Basic language. That is the only language you can use directly from the keyboard. But the Motorola 6809E processor inside the Color Computer is perhaps the most powerful 8 bit processor ever made. Programs written in Basic simply do not allow you to access the full power of this fantastic processor.

With HUMBUG, on the other hand, you can get into the machine and do things which Basic can never do. With HUMBUG you can enter or save machine language programs, read out the contents of memory in various forms, interface to a remote printer or terminal, start and stop programs, even step through programs one instruction at a time and observe how they work. In short, HUMBUG allows you to access the awesome power of the 6809 processor hidden beneath that pretty case.

How to Load HUMBUG

HUMBUG is a machine language program which is supplied on either cassette, disk, or ROM-pack (cartridge). It takes up 4K of memory, and can be used with any 16K (or larger) Color Computer, with or without Extended Basic or Disk Basic.

The cassette and disk versions of HUMBUG load into addresses $3000 through $3FFF (the prefix $ signifies that these are hexadecimal addresses.) Hence HUMBUG takes up the top 4K of a 16K machine. The program is, however, written in what is called PIC or Position Independent Code. That means that it can be placed anywhere in memory where there is RAM (read-write memory) which does not conflict with other needs. It can be moved into another area of memory either during loading (by specifying an offset in the CLOADM or LOADM command), or it can move itself once it is started.

The ROM-pack version of HUMBUG is supplied in a cartridge which plugs directly into the cartridge connector on the side of the computer. This version has the advantage that it cannot be accidentally erased from memory if you do something wrong, but it has some disadvantages as well. Since it uses the cartridge slot, it cannot be used at the same time as another cartridge or a disk interface. The ROM-pack version occupies addresses $D000 through $DFFF, but it uses a small amount of RAM (about 256 bytes) at the very top of your usable RAM for its internal storage.

a. To load the cassette version: With the computer running and the OK prompt on the screen, type CLOADM and start reading the tape. When the tape is loaded, you will again get the OK message. HUMBUG is now loaded into locations $3000 through $3FFF, and the starting address (the very first location, or $3000 in this case) has been noted by the computer. Simply type EXEC to start HUMBUG, and you will get the message

```
HUMBUG (C) 1982 P. STARK
:
```

The colon (:) is the standard HUMBUG prompt, and tells you that HUMBUG is active and waiting for commands.

b. To load the disk version: Turn on the computer, insert the disk into your disk drive, and type LOADM HUMBUG. Once the program is loaded and you get the OK prompt, type EXEC to start it. You will now get the standard HUMBUG

```
sign-on message
  HUMBUG (C) 1982 P.  STARK
  :
```

The : prompt means that HUMBUG is ready to be used.

c. To load the ROM-pack version: Before inserting or removing the ROM-pack cartridge, make sure that you always turn off the power first. Once the power is off, you can load HUMBUG by plugging in the ROM-pack cartridge and turning the computer back on. Unlike other ROM-pack programs, HUMBUG does not automatically start, so you will get the standard Basic sign-on message and its OK prompt. To start HUMBUG, type the command

```
  EXEC 53248
     or
  EXEC &HD000 (if you have Extended Basic)
```

to start. You will again get the standard HUMBUG sign-on message.

Caution

When HUMBUG starts for the first time, it automatically does the equivalent of a CLEAR statement to reserve 256 bytes of RAM for its own use at the very top of your RAM. As you will note in the Basic manual, a CLEAR erases all variables in your Basic program. Thus do not start HUMBUG for the first time if you have some Basic program variables in memory which you do not want to lose.

A Quick Introduction to Using HUMBUG

HUMBUG commands are two-letter combinations such as DE or HD. The letters used stand for the function being performed. To get a list of available commands, type HE (which stands for HELP!), and you will get the following:

```
 AD AI AO AT BA BP BR CO CS DE
 EN EX FI FM HD HE JU LO MC ME
 MH MM MT MV PR PU RC RE RT SA
 SA SS ST TF TH WH !!
 :
```

This shows the HUMBUG commands, followed by the : prompt to show that HUMBUG is waiting for the next command. Let's try out a few commands just to get the feeling for HUMBUG and its operation.

Type WH (which stands for Where is Humbug?) and you will get the answer 3000 (except for the ROM-pack version, which will answer D000). Since HUMBUG can be positioned anywhere in memory, it is sometimes easy to forget just where you put it last. Typing WH will always tell you where the currently used copy begins.

Let's move HUMBUG to another place in memory with the MH (or Move Humbug) command. HUMBUG will now ask TO: and wait for you to enter a new address. Enter the number 2000. In a moment, the screen will clear and the new

```
 HUMBUG (C) 1982 P.  STARK
 :
```

message appears. HUMBUG has moved itself to lie at $2000 through $2FFF, and restarted. Typing WH (WHere) will now result in the number 2000.

Note

The MH command in the ROM-pack version behaves differently from the other versions. Read the explanation of the MH command later for more details.

The next sections discuss all the HUMBUG commands, and you may wish to try them out as you read. For now, let us leave HUMBUG and return to your Basic by typing EX (which stands for Exit to Basic). The screen will now clear, and you will get the standard OK prompt from Basic.

Once you are in Basic, you may return to HUMBUG as long as you do not do something to erase it from memory. Typing EXEC will return you back to the last copy of HUMBUG that you used; in this case, it would return to the copy which starts at $2000. We could also specify a starting address as follows:

```
  To return to the copy at $2000:
     EXEC 8192 (since $2000 is equal to decimal 8192); or
     EXEC &H2000 (only if you have Extended Basic).
  To return to the copy at $3000:
     EXEC 12288 (since $3000 is equal to decimal 12288); or
     EXEC &H3000 (only if you have Extended Basic).
```

HUMBUG Command Summary

HUMBUG functions are called by two-letter commands. The following table lists the commands available.

AD	Formatted ASCII Dump
AI	ASCII Input
AO	ASCII Output
AT	Analyze Tape
BA	Change Baud Rate
BP	Print Breakpoints
BR	Set/Reset Breakpoints
CO	Continue
CS	Checksum memory
DE	Desembler dump
EN	Punch end-of-tape
EX	Exit (back to Basic)
FI	Find 1, 2 or 3 bytes
FM	Fill Memory
HD	Hex dump
HE	Print command codes
JU	Jump to program
LO	Load tape
MC	Memory Compare
ME	Memory exam/change
MH	Move Humbug and restart
MM	Move memory
MT	Memory test
MV	Memory View
PR	Printer
PU	Punch (write) tape
RC	Register Change
RE	Register Examine
RT	Remote Terminal
SA	Save to cassette
SI	Serial Input
SS	Single-step
ST	Start Single-stepping
TF	Terminal - Full duplex
TH	Terminal - Half Duplex
WH	Where is HUMBUG?

!! - Force complete monitor reset

Complete Description of HUMBUG Commands

Many of the HUMBUG commands, such as memory dump commands, require a starting and ending address for proper operation. These commands prompt for this pair of addresses with a FROM ... TO ...; the pair of addresses entered by the user is called the "FT" pair, and is stored in monitor RAM locations BEGA and ENDA.

Any valid hex address is acceptable as a response to the FROM ... TO ... prompt. If ENTER is typed instead of the FROM address, then the current command will use the last FT pair previously entered. Any other character will cancel the execution of the command and return to HUMBUG command entry mode.

The following covers each of the HUMBUG commands.

AD

ASCII dump. The area of memory specified by the FT (FROM ... TO ...) addresses is dumped to the screen or output device, sixteen bytes to a line. Each line is identified with its starting address. ASCII codes of 7E, 7F, and 00-1F are printed as a period, and the most significant bit (parity bit) is ignored. This command is useful in scanning through memory to look for text strings.

AI

ASCII Input. The AI command allows the direct input of ASCII data from the keyboard into any area of memory. All text following the AI is inserted into the memory area defined by the FT pair. If the memory area set aside is too small to hold all the text entered, or if the text is not properly stored (due to nonexistent or defective memory), the display will output the word ERROR immediately after the last possible character has been stored. The only way to get out of the AI mode is by typing shift - @ / BREAK (see more details later), or by pushing RESET. When this is done, the FT pair will be changed to reflect the amount of memory actually filled by the AI, so that a following AO or HD command would output exactly the same data as entered by AI.

AO

ASCII Output. Following this command, the contents of the memory area defined by the FT pair is output to the screen as ASCII characters. This command is the opposite of the AI command - AI enters ASCII text into memory, while AO outputs ASCII text.

AT

Analyze Tape. This command allows you to analyze a Color Computer cassette tape to determine (1) the program name, (2) whether it is a Basic program, data, or machine code, (3) where in memory it is supposed to load, (4) how many bytes of data it contains, (5) how much memory it occupies, and (6) whether it is in ASCII format.

To use the AT command, place the tape you wish to analyze into the cassette player and press PLAY. Then type AT. The screen will clear, and the recorder motor will start.

As the tape is being read, HUMBUG will use the top half of the screen to display tape data, and will print pertinent tape information on the bottom half of the screen.

A typical display might look as follows:

```
    BIN:     PROG
    3000,3EEB,3000
    :
```

This means that this is a binary tape and the file name is PROG. The program occupies addresses 3000 to 3EEB, and the starting address is 3000, in that order. (An ASCII tape would display the word ASCII after the addresses.)

Basic program and data files also show starting and ending addresses, but these are not significant.

HUMBUG loads tape contents into the screen memory area, locations $0420 through $051F. This area is visible on the screen as the tape is being read by HUMBUG, thus giving you an idea of what is on the tape. In addition, by examining these locations you may also read the hex contents of the last data block read from the tape. This may be useful in analyzing some tapes, especially so-called "uncopyable" tapes. (Since the screen will be scrolled when you start to examine the contents of these locations, it may be a good idea to transfer the screen memory - locations $0400 through $05FF - to another area of memory with the MM - Move Memory - command as soon as the AT command finishes, and examining the moved copy.)

BA

Change Baud rate. This command sets the speed at which HUMBUG communicates over the RS-232 serial connector at the rear of the machine. You must enter a four digit baud rate; allowable baud rates are 0110, 0300, 0600, 1200, or 2400 baud, and HUMBUG defaults to 0300 baud if no other baud rate is specified.

This baud rate setting is independent from Basic's own baud rate, which is normally 600 baud. You may change HUMBUG's default baud rate as follows: (1) Load HUMBUG into memory at address $3000 as described above. (2) Refer to the source listing at the back of this manual, and find the location which reads BAURAT FDB 182, which is near the end of the listing. (3) Using HUMBUG itself, change the number in this location from the current 00B6 to one of the following numbers:

110 baud	- 01F8
300 baud	- 00B6
600 baud	- 0059
1200 baud	- 002A
2400 baud	- 0013

(4) Using HUMBUG's SA command, save HUMBUG to cassette. Use addresses from $3000 up through the next location past the BAURAT location used in step 2, and use $3000 as the starting address.

BP

Print Breakpoints. Breakpoints are instructions which may be inserted into a program being debugged to halt the program and return to HUMBUG. When the computer encounters a breakpoint and returns to HUMBUG, it prints a register dump to indicate what was happening at the interruption.

HUMBUG allows up to four breakpoints to be set at the same time. The BP command prints out the numbers and addresses of the current breakpoints, and the operation codes of the instructions at those breakpoints, so that the user does not forget their locations.

Breakpoints use the SWI instruction, which replaces the program instruction normally at that place in the program. HUMBUG keeps track of the instructions replaced, and their operation codes are displayed when the BP command is executed. Breakpoints remain in a program even when the RESET button is pressed or HUMBUG is reentered from Basic, and the BP instruction will keep track of them.

BR

Breakpoint set/reset. The four possible breakpoints (see the BP command described above) are numbered 1 through 4, and can be individually set or reset. The typical BR command has the following form:

```
        BR NO: n NEW ADDR: addr
           ---   ---------
```

where the computer's responses are underlined. n is the number of the breakpoint you wish to set or reset; addr is the new address of that breakpoint. Entering a new address, or hitting ENTER or any invalid entry for addr, will cancel the old breakpoint number n.

CO

Continue. After a breakpoint is encountered in a program, or after a single-step execution, the program being tested may be continued with the CO command. After a breakpoint, the breakpoint should be removed with the BR command before hitting CO; otherwise the break will be executed again and the program will not go on.

When the CO command is given, the program being restarted will continue to the end, unless another breakpoint is encountered, or unless there is an error in the program. You may wish to insert more breakpoints before continuing, or perhaps using single-stepping instead.

Using the CO command if no previous breakpoints or single stepping has been performed will probably cause the system to crash, as HUMBUG will attempt to continue a non-existent program.

CS

Checksum. This command prints a 16-bit checksum of the memory area defined by the FT pair. The CS command is useful for doing "before and after" comparisons on memory, perhaps to see whether a program has been properly loaded, or to check whether some data in memory has been modified since some previous time.

The checksum is a simple 16-bit sum of the locations being checked.

DE

Desemble. The DE causes HUMBUG to do a memory dump of the memory area defined by the FT pair, in a semi-disassembled form. Memory contents are grouped in one- to four-byte groups as if they were variable length instructions. Each group is preceded by its address in memory.

For example, a section of memory starting at location 1000 which contains the numbers 86 0C 17 02 85 AE 8D 0D 91 would be interpreted as the instructions

```
   1000  86 0C
   1002  17 0285
   1005  AE 8D 0D91
```

and displayed in that format. Though assembly mnemonics are not shown, the DE command is very useful in displaying machine code in a more readable and recognizable form.

EN

End-of-tape. The EN command is one of three commands which support the traditional tape format (sometimes called the S1-S9 or "MIKBUG" format) used by other 68xx based computers. It transmits the end-of-tape code (S9) over the RS-232 serial port at the back of the computer at the baud rate chosen by the BA command. (See the comment under the PU command for details.)

EX

Exit (return to Basic). Clears the screen and gives the normal Basic OK prompt. EX will return to whichever Basic is in the system - the basic Color Basic, Extended Basic, or Disk Basic. You may return from Basic back to HUMBUG by typing EXEC, unless you erase HUMBUG after returning to Basic.

FI

Find. FI is used to search memory for a specified one-, two- or three-byte number, and prints out the addresses of all locations (in the area defined by the FT pair) which contain that number.

The typical command sequence is

```
  FI HOW MANY?  n WHAT?  dd FROM addr TO addr  .
     ----------     -----     ----      --
```

where computer responses are underlined. n is the number of bytes to be found, dd are 2, 4, or 6 hex digits representing the 1, 2, or 3 bytes to be found, and addr are the two FT addresses specifying the address range to be searched.

In addition to printing the address where the number was found, FI also prints the contents of the four locations beginning just before that address. For example, if location 1002 contains the number 17, then the command

```
  FI HOW MANY?  1 WHAT?  17 FROM 1000 TO 1010
```

would produce the printout

```
  1002  CC 17 02 85
```

These are the four bytes located at addresses 1001 through 1004; the second byte is the one at location 1002. In general, then, this command displays the desired byte as well as the byte before and several bytes after the desired one.

FM

Fill memory. This command allows a specified area of memory, defined by the FT pair, to be filled with a specified byte. For example, all of memory could be filled with SWI instructions (3F) prior to executing a faulty program so that control would safely return to HUMBUG if a program goes astray.

For example, the command

```
 FM FROM 0400 TO 05FF WITH 3F
```

would fill locations 0400 through 05FF with the SWI instruction 3F. (Incidentally, since locations 0400 through 05FF are the screen memory of the Color Computer, and 3F is the ASCII Code for a question mark, this command would also fill the entire screen with question marks.)

HD

Hex Dump. Prints a hexadecimal dump of the area of memory defined by the FT pair. Eight bytes are printed per line, with each line preceded by the address.

HE

Help. Prints a listing of all HUMBUG commands. This is a useful command if you forget what other commands HUMBUG has.

JU

Jump. The JU command is used to execute other programs by jumping to their starting address. The normal command is to type JU xxxx, where xxxx stands for the hexadecimal starting address to jump to.

In executing the JU command, HUMBUG resets the stack pointer to its user stack (see the USTACK description later), and uses a JSR instruction to enter the user program. This places a return address above the stack so that the subroutine will return to the monitor when done.

LO

Load tape. The LO command is one of three commands which support the traditional tape format (sometimes called the S1-S9 or "MIKBUG" format) used by other 68xx based computers. It loads data from the serial RS-232 port at the back of the computer at the baud rate chosen by the BA command. (See the comment under the PU command for details.)

MC

Memory compare. This command compares two specified memory areas byte-by-byte, and prints out memory contents for each byte which is different in the two areas. Prompts ask for the FT pair for the first area, and for the starting area of the second.

ME

Memory Examine and change. This is one of the more often used commands of any monitor, and is used to examine or change the contents of memory. Follow the command ME with the address of the memory location you wish to examine or change. HUMBUG will then display, on the next line, the address and contents of that location, and wait for a new value if you wish to change the contents. You now have the following options:

1. If you enter a new hexadecimal value, that value will be placed into memory, and HUMBUG will display the address and contents of the next location. (If the location cannot be changed, as when you try to change the contents of ROM or read-only memory, HUMBUG will print a question mark.)
2. If you enter an up arrow, HUMBUG will back up to the previous location and display its contents.
3. If you hit the ENTER key, HUMBUG will return to the : prompt.
4. If you type any other key, HUMBUG will leave the displayed location unchanged and go to the next location.

MH

Move HUMBUG. This command is used to relocate all of HUMBUG to another 4K block of memory. After you type MH, HUMBUG will respond with TO: and wait for a new address, which you must provide as a four digit hexadecimal number. If the address is valid, HUMBUG will copy itself to start at the specified location, and then jump to the beginning of the new copy. It will retain all existing data and parameters. If the address is invalid (for example, the new copy of HUMBUG would overlap the existing copy, or there is no RAM at the new location), HUMBUG will print the word NO! and refuse to perform the command.

In general, moving HUMBUG with the MH command retains all of HUMBUG's data and pointers, but there are three instances where moving HUMBUG may cause problems.

1. If HUMBUG is in the process of single-stepping through a user program, the user stack will not be moved unless the user program is restarted with the JU command.
2. If the remote terminal (RT) option is being used to control the Color Computer from a remote terminal, the old copy of HUMBUG will still be used for

the remote I/O routines. If the RT command is executed after the MH, then the new I/O routines will be used instead.

3. In cassette and disk versions, the MH moves both the HUMBUG program and also all of its data to a new location. In the ROM-pack version, however, MH moves only the program - the data area is always at the last 256 bytes of available RAM.

In a 16K system, the data uses locations $3F00-3FFF, in a 32K system the data uses locations $7F00-7FFF. Since the ROM-pack version of HUMBUG requires a full 4K, it should only be moved to $2F00 in a 16K system, or $6F00 in a 32K system, exactly 4K under the data area. (Any other move would result in the program being in one place and the data in another, a situation which might cause Basic to erase part of it.)

Once ROM-pack HUMBUG is moved to its new location, it should be protected by giving Basic the appropriate CLEAR command:

```
     In a 16K system use either
        CLEAR ..., 12031
              or
        CLEAR ..., &H2EFF (if you have Extended Basic)
     In a 32K system use either
        CLEAR ..., 28415
              or
        CLEAR ..., &H6EFF (if you have Extended Basic)
```

(the periods should be replaced by the number of bytes you wish to reserve for strings. If you are not going to use Basic, reserve just one or two bytes, if any.)

MM

Move memory. This command allows the contents of the memory area specified by the FT pair to be moved (copied) to another memory area. Memory data can be moved to higher or lower addresses, and the new area can overlap the original area. Moving is done in the correct way so that no data is lost even on overlaps.

HUMBUG prompts for the old and new addresses. For example, to copy Color Basic into locations 4000 through 5FFF, the command would be

```
 MM OLD ADDR: FROM A000 TO BFFF
    NEW ADDR: 4000
```

Note, however, that this command should not be used to move HUMBUG itself. Use the MH command instead.

MT

Memory Test. Does a simple (rotating bit) memory test on the memory area defined by the FT pair. If memory is OK, it prints OK and returns to HUMBUG. If memory is bad, it prints the address of the bad location, a hex number representing the bad bit, and the actual contents of that location at the time it failed the test. (This is a non-destructive test of memory since the previous contents of each location are restored. If, however, a memory test is done of I/O or SAM locations, or locations used by HUMBUG itself, the system may crash.)

MV

Memory View. This command allows you to view a 512-byte section of memory on the screen at one time by instructing the 6883 SAM integrated circuit to substitute a different 512-byte video memory buffer instead of the one normally assigned for alphanumeric use at $0400 through $05FF. When you type the MV command, HUMBUG asks for a STARTING ADDRESS: and expects a four-digit hexadecimal address. When you enter the address, the 512-byte block of memory containing that address (beginning on an even 512-byte boundary) will be displayed on the screen. The display will be terminated when you hit any key, and the normal screen display will reappear.

Because of the way the 6883 SAM integrated circuit is wired, only part of the computer's memory may be accessible by the MV command. In general, any memory above $8000 cannot be accessed; moreover, in some 32K machines the upper 16K may not be properly addressed, with the result that only the lower 16K to $3FFF may be used. If an invalid memory address is specified the display will show meaningless garbage.

PR

Printer. The RS-232 Serial port on the Color Computer may be used in two different ways: either for bi-directional communications with a modem or terminal, or for one-directional output to a printer, with the input line used for handshaking. The former method is that used by Radio Shack's Videotex and other programs; the latter is that used by Basic itself. The PR command switches to and from the 'printer with handshaking mode'. The function of this command is explained more in a separate section dealing with serial I/O.

The PR command is included to allow full compatibility with Basic.

PU

Punch tape. The PU command is one of three commands which support the traditional tape format (sometimes called the S1-S9 or "MIKBUG" format) used by other 68xx based computers. It outputs data to the serial RS-232 port at the back of the computer at the baud rate chosen by the BA command. (The EN, LO, and PU commands use the serial RS-232 port, and are intended for communication with other 6800 or 6809 computers or for saving and loading data on cassettes intended for those systems. Use the SA command for saving machine language programs for use with the Color Computer.)

For example, to see the S1S9 format, execute the command

```
  PU FROM A000 TO A003
```

The screen (and serial output) will be

```
  S107AC00A1C1A282D2
```

In this example, S1 signifies the start of data, 07 is the number of bytes following (each byte is shown as a two-digit number), A000 is the starting address of the data, A1 C1 A2 82 are the four bytes at locations AC00-A003, and D2 is a checksum used for error detection.

RC

Register Change. This command is used in conjunction with the RE or Register Examine command discussed next. After the RC command is used, HUMBUG will ask for the name of a register to be changed. You should respond with one letter, which may be C (condition code register), A or B (accumulator), D (direct page), X or Y (index register), U (U stack pointer) or P (program counter). HUMBUG will then find the memory location which holds that register, and switch to the ME mode to allow examination or change of that register.

RE

Register examine. Prints the contents of all registers as maintained in the user's stack following a breakpoint or single-step. A typical display

Format is as follows:

```
 EFHINZVC A  B  DP  X    Y    U
 11010000 11 22 33 4444 5555 6666
  PC   SP
 7777 8888
```

The register contents printed are the condition code register (with each bit listed separately under the letters EFHINZVC), accumulators A and B, direct page register DP, index registers X, Y, and U, program counter PC, and stack pointer register SP.

The stack pointer address printed out is the SP as it existed in the user's program just prior to the break, rather than as it exists after return to the monitor. Register contents are undefined prior to execution of the first breakpoint or single-step. An RE printout is automatically performed following a single-step or upon encountering any breakpoint (SWI) instruction.

RT

Remote Terminal. The RT command reconfigures the I/O routines of your Color Computer to allow you to control the computer from a terminal connected to the serial I/O port. The connection can be direct, or it could be through a modem and telephone line. The RT command is interlocked with several other commands; see the complete discussion of serial I/O functions later.

SA

Save to cassette. This command is functionally identical with the CSAVEM command of Extended Basic, and is provided as a convenience for those systems which do not have Extended Basic. The command prompts for the beginning and ending addresses of memory to be saved to tape, the transfer address, and the program name. Unlike Basic's CSAVEM, all addresses must be given as four-digit hexadecimal numbers.

The SA command may be used to make backup or relocated copies of HUMBUG (but please do not use it to make copies for your friends, as that is a violation of copyright laws and we will sue you if we find out.)

In order to make a copy of HUMBUG, you will need to know the last address it uses. You may obtain this by doing an AT test on the cassette, or by referring to the program listing in the back of this manual. (The program size may vary if we revise HUMBUG, so be sure to refer to your manual for this information on the copy you have.) Suppose the listing at the back of this manual says that HUMBUG occupies addresses 3000 through 3EEB. The correct command sequence would then be as follows:

```
 SA   FROM 3000 TO 3EEB
   START ADDR: 3000
   NAME: HUMBUG
```

SI

Serial Input. HUMBUG normally uses the Color Computer's own keyboard for all input. It is possible, however, to connect either a standard terminal, or another computer, to the Color Computer via its RS-232 connector, and control HUMBUG from there. The SI command switches back and forth between the Color Computer's own keyboard and the external serial input. See the serial I/O section later for more details.

SS

Single-Step. The SS command is a unique feature of HUMBUG which allows you to step through a program one instruction at a time for debugging purposes.

It does this by automatically inserting breakpoints (SWI instructions) after each instruction (and then automatically removing them) so that just one instruction is done at a time.

SS uses the register contents printed by the RE command; hence the SS command cannot be used to start single-stepping until after a prior breakpoint or single-step has been performed. When an SS is performed, HUMBUG prints out five lines: the first line prints out the address and code of the instruction to be performed, while the other four lines print out the RE dump after the instruction has been performed. (Not all 6809 instructions can be single-stepped; see the section on Single-Step Limitations later for further information.)

ST

Start single-stepping. Since SS cannot be performed until after a breakpoint or previous SS, the special ST command is included to perform an initial single-step if the breakpoint is not used. ST prompts for the address of the first instruction to be single-stepped, and then executes it in exactly the same way as the SS instruction. The ST command always enters user programs with the DP register set to 0.

Note

Breakpoints and single-stepping use the SWI instruction, which in turn uses the SWI vector in location 0106. If SWI is changed by any user program, then subsequent attempts to use breakpoints or single-stepping may not work, and may in fact cause the current program to self-destruct.

TF
TH

TF is Terminal, Full duplex; TH is Terminal, Half Duplex. By using the TF or TH commands, the Color Computer can be used as a dumb terminal via its serial RS-232 port. All data typed on the keyboard will be sent out over the serial port, and serial data received from the RS-232 line will be displayed on the screen. Before using the terminal modes, you should specify a baud rate with the BA command (unless the default value of 0300 baud is acceptable.) When outputting from the keyboard to the serial port, the Color Computer can generate both upper and lower case letters (using the Shift-0 sequence described in the Color Computer manual), numbers, and control characters. To generate control characters, first push the right arrow key and then a second key. For example, to generate control-G, push first the right arrow and then the letter G.

The TH and TF commands are interlocked with several other serial I/O commands; see the separate section on serial I/O later for more details.

WH

Where is HUMBUG? This command prints out the starting address of the current copy of HUMBUG being executed. Since HUMBUG is relocatable and could be anywhere in memory, this command is used to tell you where HUMBUG is currently located.

!!

Monitor reset command. HUMBUG does not normally erase breakpoints except at the first power up; other resets omit this step. The !! command does a complete reset, exactly the same as at power up. In general, this is a command which will not be commonly used, and hence has been assigned a non - standard command code.

I/O Control via Shift-@

Just as in normal Basic operation, HUMBUG I/O can be halted and restarted by typing Shift-@. The use of this combination in HUMBUG is, however, substantially different and more powerful than in Basic.

Whenever the Shift-@ is typed (either when HUMBUG is outputting, or even when HUMBUG is just waiting for input), HUMBUG halts all I/O and waits for one more character which is used for controlling the I/O process. This control character can be one of the following:

BREAK - this cancels the current program and forces a return to HUMBUG's : prompt.

O - the letter O turns the serial output port on and off. (See the serial I/O section following for details.)

P - turns the pause mode on and off. When the pause mode is on, output will stop every 15 lines to allow it to be read on the screen.

Any other character is ignored.

Since these characters are not echoed or returned to calling programs, the serial port can be turned on and off in the middle of input or output.

When an external terminal is used to control the Color Computer, hitting any key can be used to stop output (instead of the Shift-@), although it may sometimes be necessary to hit the key several times. A control-C is used (instead of BREAK) to return to HUMBUG (unless Basic programs are running in the RT mode, in which case a control-C will return to Basic's OK prompt).

Serial I/O

The Color Computer has an RS-232 serial connector on the back; HUMBUG has several commands which allow this connector to be used with HUMBUG or even Basic. There are several different operating modes, and this section describes how they work in more detail.

In general, the serial port can be used for two tasks:

1. As a bi-directional I/O port for a terminal.
2. As a one-directional output port for a printer.

The job is complicated by the fact that there is some confusion about how this port is used.

The RS-232 connector uses a four-pin DIN connector, rather than the standard 25-pin DB connector. The pins are labelled as follows:

1. CD - a signal status line
2. Data Input
3. Ground
4. Data Output

Theoretically, pin 1 is used to send READY or BUSY status to the computer. In reality, however, when a printer is used with the Color Computer, Basic assumes that a READY signal is provided on pin 2, not pin 1. Hence pin 1 is unused, and pin 2 is used for two purposes.

Since HUMBUG users may have a Basic-compatible printer, HUMBUG is forced to be compatible with this use. In general, HUMBUG allows four types of devices to be connected to the serial port:

TYPE 1. A normal serial terminal, connected to send data to the computer on pin 2 and take data on pin 4. The terminal would be switched to full duplex and operating on a baud rate between 110 and 2400 baud. In this mode, the remote terminal can operate the Color Computer remotely, and either operate

HUMBUG or run Basic programs. (The terminal may be connected either directly, or through a modem.)

TYPE 2. A remote computer, again connected to send data to the Color Computer on pin 2 and take data from pin 4. In the most common situation, the Color Computer would be used as a terminal to the remote computer, although the situation could also be reversed. The connection can be through a modem, if desired.

There is a baud rate limitation when a remote computer is used. Because of the time it takes the Color Computer to scroll its display, a maximum baud rate of 110 baud may be used unless the remote computer is programmed to insert a pause after every character which might cause a scroll on the Color Computer's screen. This includes not only carriage return and line feed codes, but also any line longer than 32 characters which causes a wrap-around on the screen.

TYPE 3. A printer configured to be compatible with Color Computer Basic. The printer should accept data from pin 4 of the serial connector, and return a READY/BUSY signal to pin 2 which is negative (-3 to -15 volts, or even 0 volts) when the printer is busy and cannot accept output, and positive (+3 to +15 volts) when the printer is ready and can accept output. To be compatible with Basic, the printer must do an automatic line feed after every carriage return, and should ignore any line feed characters received.

TYPE 4. A printer not compatible with Color Computer Basic. The printer should accept data on pin 4, but not return any READY/BUSY signal. For this to work properly, the baud rate must be set below that rate which the printer can accept in a continuous stream of characters, or else the printer must have an internal buffer which is capable of holding all the data which will be fed to it. In this mode, the printer should not provide automatic line feeds, but should do a line feed each time an LF (hex 0A) code is fed to it.

To support these four types of devices, HUMBUG has four commands and one Shift-@ code. In addition, several other commands affect the serial I/O indirectly.

BA - changes the baud rate for all serial I/O functions except for Basic's own printer routines (such as LLIST).

EN, LO, and PU - the S1S9 format tape commands automatically turn the serial port on and off as needed, and use this port.

SI - Serial Input. This command enables external input to control HUMBUG (not Basic). This mode would be used most often for a TYPE 1 terminal, although it could also allow control of the Color Computer from a remote TYPE 2 computer. The Color Computer keyboard remains active, so that HUMBUG commands can be entered from there as well. Note that the SI command enables input but not output.

Shift-@ followed by the letter O enables serial output from HUMBUG to either a type 1 terminal, type 2 computer, or type 4 printer. This output runs in parallel with the Color Computer's own screen. The most common application of this function would be to print selected segments to a printer. By accessing this function through a Shift-@ function rather than a separate command, we can turn the printer on and off in the middle of other output, as needed.

PR - this command switches the above functions from a type 4 printer to a type 3 printer and vice versa. In the type 3 printer mode, line feeds are not sent and handshaking is required on pin 2 of the serial connector.

TN and TF allow the Color Computer to act as a dumb terminal to either a remote computer, or for communication with another terminal (or Color Computer) via the serial connector. In TF, characters typed on the Color Computer keyboard do not display on the screen unless they are echoed by the remote computer; in TH they are displayed at the same time as they are sent to the remote computer. The only way to get out of the TH or TF modes is by pushing the RESET. In this mode, the Color Computer can send upper and lower case letters (by using the Shift-0 key to switch to upper/lower mode). It can also send control characters by preceding them with the right arrow key. For example, to send control-C, push the right arrow and then C.

RT allows a remote Type 1 terminal, or Type 2 computer, to control the Color Computer. Though this mode will work with HUMBUG (but skip extra lines), it is intended for running Basic programs.

The RT mode works with all Color Basic programs and also with all Extended Basic programs which do not use graphics. It works with some Disk Basic programs but not others, and use with Disk Basic is not recommended.

Whenever a remote terminal or keyboard is used for controlling the Color Computer, the Shift-@ character is replaced by a Control-S, and the BREAK key is replaced by a Control-C. For example, when a program is running and outputting to the remote terminal, hitting any key once or twice will generally stop it, and then typing control-C will stop the program and return to the main prompt (in the case of HUMBUG, control will return to the : prompt; in the case of Basic programs, control will return to Basic's OK prompt.)

HUMBUG's Memory Requirements

In general, HUMBUG will be used to examine or debug machine language programs at a time when no Basic program is running. If, however, you want to use HUMBUG in conjunction with a Basic program (as in, for example, using the remote terminal feature), then this section discusses how to ensure that HUMBUG can coexist with Basic in relative peace and harmony.

Cassette and disk versions of HUMBUG take up 4K of RAM, from $3000 through $3FFF. (They are fully relocatable, however, so you may move HUMBUG to anywhere where there is room.)

Although HUMBUG may conceivably change with time, the 4K of RAM used by HUMBUG is divided approximately as follows:

```
     $3000-$3EC0 - Main program
     $3EC0-$3F00 - Fixed data storage
     $3F00-$3F80 - Monitor stack
     $3F80-$3FFF - User stack
```

The cassette or disk file only contains the main program area (approximately to $3EC0); the data storage is not on the tape, but is initialized by HUMBUG once it is used.

The fixed data storage area contains data which HUMBUG uses while running. If you exit from HUMBUG to Basic and then return, this data should hopefully remain intact and not be wiped out by Basic.

The monitor stack area is re-initialized each time you re-enter HUMBUG from either Basic or a user program, and hence does not have to be protected from Basic.

The user stack area is only used when you exit HUMBUG into a user program, and if you return back to HUMBUG from a breakpoint or single step. Hence this area too does not generally have to be protected from Basic.

(In the ROM-pack HUMBUG, the main program area is in the cartridge ROM, and only the fixed storage and stack areas are in RAM.)

Basic, in turn, requires four areas of memory:

1. Various operational constants, pointers, screen memory, etc. begin at location $0000. How much memory is used up depends on which Basic it is and, in the case of Extended or Disk Basic, how many pages are set aside for

graphics with the PCLEAR command.

The basic Color Basic system requires memory up to $0600. Extended and Disk Basic (neglecting Disk Basic programs which use random files or multiple disk files) require memory up to the following addresses:

Number of pages PCLEARed	Extended Basic	Disk Basic
0*	$0600	$0E00
1	0C00	1400
2	1200	1A00
3	1800	2000
4	1E00	2600
5	2400	2C00
6	2A00	3200
7	3000	3800
8	3600	3E00

(* The PCLEAR statement cannot be used to PCLEAR 0, but there are other ways of achieving the same end with some POKEs.)

As you can see, the amount of memory used up for graphics pages can be substantial.

2. Immediately after the space set aside in 1. above comes the Basic source code for whatever program you are executing. Whenever you do a PCLEAR instruction, Basic moves the program up or down, as necessary, to put it right after the area set aside for graphics pages. The source code takes up as much room as is necessary for the program entered.

3. Once a Basic program is RUN, Basic uses any available space above the source code for the data used by the program itself.

4. At the top of RAM, Basic sets aside space for a stack and for string variables. The amount of space used up here depends to a large extent on the CLEAR statement.

Hence Basic manages to pretty well use up available RAM when it is running a program. In fact, even fairly short Basic programs can use up all of the available memory even in a 32K system if they use a lot of data and array storage.

One the other hand, when it is not running a Basic program, Basic may leave quite a bit of space available for HUMBUG somewhere in the middle, above the Basic data (item 1. above) and below the Basic stack (item 4. above).

In general, then, it is fairly safe to load HUMBUG into memory at location $3C00, even in a 16K system whose RAM memory ends at $3FFF, as long as a Basic program is not run while HUMBUG is in memory, without any special precautions. If a Basic program is run, however, it is quite possible that HUMBUG might be erased by Basic's program, data, or stack.

The best way to protect HUMBUG and allow it to remain in memory while a Basic program is being executed is to use the CLEAR statement to set aside memory at the top of the available RAM.

The syntax for the CLEAR statement is to type

    CLEAR stringspace, memend

where stringspace tells Basic how many bytes of memory to allow for storage of strings, and 'memend' or memory end specifies the top of allowable RAM to use. Both of these numbers may be specified in decimal, or (with Extended Basic) in hexadecimal.

When the computer is first turned on, Basic automatically allows 200 bytes for the string space, and sets the memend pointer to point to the last location of available RAM. This works out as follows:

Memory Size	Last available RAM address Decimal	Hexadecimal
4K	4095	$0FFF
16K	16383	$3FFF
32K	32767	$7FFF

The string space lies just at the limit of memend, and Basic's stack is just before it. For example, with a string size of 200 bytes and 16K of RAM, locations 16183 through 16383 are used for 200 bytes of strings, and the stack goes from just below 16183 down to about 16146, or about 35 additional bytes.

The stack is constantly changing in size, but the bottom of the stack, at approximately decimal location 16146 or hexadecimal location $3F12, is almost on the verge of stepping on the very last data location of HUMBUG.

We could, of course, cut down the string space and thereby move the stack higher. For example,

```
 CLEAR 100
```

would reduce string space by 100 bytes and therefore move the stack up by 100 locations. But, though this will keep Basic's stack from stepping on the top of HUMBUG, running a Basic program with a large amount of data is still likely to creep up on HUMBUG from the bottom.

The best approach, then, is to use the CLEAR statement to specify a memend location which is below HUMBUG. For example, specifying a memend location of $2FFF (in a 16K system) would leave the entire 4K beginning at $3000 reserved for HUMBUG, and would guarantee that no data, string, or stack would step on HUMBUG.

In a 16K system, with HUMBUG at $3000, the CLEAR statement should be either CLEAR 200, 12287 or CLEAR 200, &H2FFF (with Extended Basic). This statement reserves 200 bytes for strings, but this can be changed.

In a 32K system, HUMBUG could be placed at $7000, and then the top 4K protected with either CLEAR 200, 28671 or CLEAR 200, &H6FFF.

There are several ways of extending RAM memory above 32K, either in the form of RAM on a disk controller (as in the Exatron disk), or on a cartridge which plugs into the ROM slot on the side of the Color Computer. When one of these is used, then it is possible to place HUMBUG into this higher memory, which is generally at address $C000 or above, and keep it far away from Basic altogether. (There is another way of increasing memory past 32K, which was described in the February 1982 issue of Color Computer News, and the March 1982 issue of 68 Micro Journal. This method, however, disables all of the Color Basic or Extended Basic ROMs, and hence completely eliminates all the standard I/O routines from the machine. The current version of HUMBUG will not work with this memory expansion, though we plan to offer a different version of HUMBUG at a later time.)

The ROM-pack version of HUMBUG avoids many problems by using the cartridge memory space at $D000, although it does mean that it cannot be used with any other device plugged into the cartridge slot. This Humbug does, however, require 256 bytes of RAM at the very high end of memory for its data and stack. This area is automatically reserved by HUMBUG, because HUMBUG subtracts 256 bytes from Basic's memend pointer and forces Basic to ignore the top 256 bytes of RAM.

The Stack in HUMBUG

HUMBUG treats the stack in a completely different way from most other monitors.

HUMBUG always sets up two different stack areas called USTACK for user stack, and MSTACK for the monitor stack.

Assuming that HUMBUG is located to start at address X000 (where X could be any hexadecimal digit corresponding to existing RAM), the monitor stack MSTACK lies just under location XF80, and the user stack USTACK lies just under location XFFF. In other words, the top 200 bytes or so of the 4K RAM segment used by HUMBUG are dedicated to the two stacks. Separation of the two stacks in this way guarantees that user programs will not interact with HUMBUG.

When HUMBUG is relocated with the MH command, the stacks are moved along with HUMBUG, except that any user stack which is currently being used by a user program being breakpointed or single-stepped remains in its old location until the next JU command is executed.

Whenever HUMBUG is restarted for any reason (with the sole exception of running a single subroutine from another program), HUMBUG reinitializes the S stack pointer to use MSTACK. On the other hand, whenever a JU is made to a user program, HUMBUG always sets the S stack pointer to USTACK. (The U stack pointer is not initialized). When a return is made to a user program with the SS or CO commands, then the previously used user stack is again established.

Making this distinction between user stack and monitor stack decreases the possibilities of harmful interaction between the two.

In ROM-based versions of HUMBUG, the stacks (and other variables) are instead located at the very top of available RAM (just under $3FFF in a 16K system, under $7FFF in a 32K system), and Basic's RAM vectors are modified so that Basic will not interfere with this area.

Single - Step Limitations

Although most 6809 instructions can be single - stepped by HUMBUG, there are several types of instructions which cannot. This includes RTI, SWI, SYNC, and CWAI instructions, as well as certain kinds of jumps and branches. In addition, single - stepping cannot be done on programs in ROM.

All branch instructions can be single - stepped except those for which the offset is - 2, - 1, or 0. This includes instructions such as

```
    HERE BRA HERE
```

or

```
    HERE BRA NEXT
    NEXT ...
```

The same limitation applies to jump instructions.

In addition, indexed jump instructions of the form JMP (or JSR) n,R are allowed only when n is a number from 0 to 15, and R is either the X, Y, U, or S register. In other words, indexed jumps with negative offsets or with offsets greater than 15 are not allowed; neither are indexed jumps which are PC relative. If an attempt is made to single - step an instruction which is not implemented, HUMBUG will print the message "NO!"

HUMBUG Routines and Entry Points

In addition to its own commands and functions, HUMBUG has a number of routines which may be useful to you in other programs. All of the HUMBUG routines are vectored through a jump table at the beginning of HUMBUG. Assuming that HUMBUG begins at location X000, the following table lists the documented entry points. Any other entry points should be avoided, as they may change in future releases of HUMBUG.

HUMBUG	X000	HUMBUG beginning
NXTCMD	X003	HUMBUG monitor command processor
INEEE	X006	Input character from active keyboard
INCHCC	X009	Input character from CC keyboard
INCHKB	X00C	Input character from serial keyboard
INHEX	X00F	Input a Hexadecimal digit
BADDR	X012	Input 4-digit Hex number into X reg
OUTEEE	X015	Output character to current device
OUTCHC	X018	Output character to CC screen
OUTCHS	X01B	Output to serial output
OUTCHL	X01E	Output to serial output w/o line feeds
PDATA	X021	Print output string
PCRLF	X024	Print carriage return and line feed
OUTS	X027	Output a space
OUTHR	X02A	Output one hex digit
OUT2HS	X02D	Output 2 hex digits and space
OUT4HS	X030	Output 4 hex digits and space

These entry points are to be accessed with JSR instructions.

All INxxx routines leave the data entered in the A accumulator and preserve registers. BADDR leaves the address in the index register. INHEX and BADDR return to HUMBUG if an invalid hexadecimal digit is entered.

OUTEEE, OUTHR, and all OUTCHx routines assume that the byte to be printed is in the A accumulator, and preserve registers. OUT2HS and OUT4HS print the data which is pointed to by the index register, and increment the register to point to the next location after the data when done.

OUTEEE and all OUTCHx routines recognize hex 0C as a clear screen character.

PDATA is used to print a string to the current output device (which is always the Color Computer screen and could also include the serial port). The index register should point to the first character of the string, and the string should end with a 04 delimiter.