Coco Sleuth3
Program Analysis and Debugging Tool

by Edgar M. (Bud) Pass, Ph.D.

With Updates for OS9 Level 2 1990-1995
M. E. (Gene) Heskett
291 Garton Avenue
Weston, WV 26452

Updates for NitrOS9 Level 2 2016
Bill Pierce
570 Sandy Bend Rd.
Rocky Point, NC 28457
ooogalapasooo@aol.com

Copyright Notice

Limited Warranty Statement

Computer Systems Consultants, Inc., and it's agents, makes no express or implied warranties concerning the applicability of Coco Sleuth3 to a particular purpose. Liability is limited to the original license cost. This warranty is presented expressly in lieu of all other warranties, expressed or implied, including those of merchantability and fitness for use and of all other obligations on the part of Computer Systems Consultants, Inc. and it's agents.

Problems and Improvements

Users are encouraged to submit problems and to suggest or to provide improvements for Coco Sleuth3. Such input will be processed on a best effort basis. Computer Systems Consultants reserves the right to make program corrections or improvements on an individual or wholesale basis, as required. The company is under no obligation to provide corrections or improvements to all users of Coco Sleuth3. In the case of specific situations requiring extensions to Coco Sleuth3 or major assistance in it's use, consulting on a pre-arranged, for-free basis.

Table Of Contents

Coco Sleuth3--An Overview

Coco Sleuth3 is a collection of three programs which enables the user to examine and/or modify binary program files on disk or in memory, on Tandy TRS-80 Color or TDP-100 or similar computers, with at least 32k bytes of memory and at least one disk drive.

Programs may be disassembled into source code format and the source may be displayed, printed, or saved on disk. Labels produced by Sleuth3 may be changed globally to labels of your own preference. Cross reference listings of labels may be produced to aid in debugging or modifying the program. Programs in ROM or on disk may be "altered" with the altered program being saved on a disk file; the resultant file could then be used to program a new ROM, ect.

The three programs are named Sleuth3, Chngnam3, and XRef3. These are the Disassembler, Name-Changer, and Label Cross Reference generator, respectively. The programs are supplied as 6809 object files for the Tandy Color or TDP-100 or similar computers. The processors which may be analyzed are 6800, 6801, 6803, 6805, 6808, 6809, and 6502.

NOTE: This copy of the "Coco Sleuth3" manual was hand copied from the original "Coco Sleuth" manual for RSDOS. It is being distributed with the OS-9 version of "Coco Sleuth3" since no copies of the OS9-9 version of the "Coco Sleuth" manual or the RSDOS vesion of the "Coco Sleuth" program can be found.
Several edits may or may not appear in order to change the RSDOS loading and running instructions to match the OS-9 instructions. The instructions for the actuall program operations are exactly the same in both versions of "Coco Sleuth". If you find any mis-spellings or incorrect text, chalk it up to "too many late nights typing from a bad PDF scan of the original manual".

This edition of "Coco Sleuth3" has been released to "Public Domain" by "Edgar M. (Bud) Pass" (original author). The original source code was modified by myself and Gene Heskett (see "Gene's Notes.txt") to faciliate use with NitrOS9 Level 2 from it's original OS-9 Level 1 status. Conditional code is provided to allow Sleuth3 to be compiled for use with NitrOS9 Level 1 as well. The name of the program has been changed to "Sleuth3" to distinguish it from the original.
Several disassemblies of OS-9 modules were made to make sure Sleuth3 would disassemble the modules to match the original sources of the modules. No problems were found, how ever, Sleuth3 does have a couple of "quirks" in it's disassembly. These are listed below:

psh-pul display: When disassembling a "pshs", "puls", "pshu", or "pulu" instruction, Sleuth3 will list the register names in reverse order from standard OS-9 convention. The OS-9 standard is to display the registers in order from low to high (with the exception of the "pc" reg as it is always last). Example:

```
pshs cc,a,b,x,y,u,pc
```

Sleuth3 displays this as:

```
pshs pc,u,y,x,b,a,cc
```

In consulting the OS-9 programming manual, the manual states that the order of the registers do not matter and that the assembler will place then in the needed order during assembly, so this should not be a concern. Assemblers other than the standard OS-9 "asm" assembler may differ in this respect.

Another quirk is the module header naming convention. The names used by Sleuth3 in the creation of the module header code is slightly different from standard OS-9 naming convention, but the resulting sources will compile properly using the Sleuth3 naming convention.
Bill Pierce

Sleuth3--A Disassembler/File Editor

This program is the "work-horse" of the Coco Sleuth3 package. Some of the functions of the program include the following:

1. Disassemble a program from a disk file and write the source to another disk file.
2. Disassemble a program in memory and write the source to a disk file.
3. Dump the object code of a binary file in memory-dump format and allow modifications to the file. The modified file can then be written back to disk.
4. Dump the object code from memory, allow modifications, and write modified program to disk.

In all the above cases, any modifications made by the user do not actually change the original object code. Rather the changes are stored in a table and overlaid into the original code when the output file is written to disk. In the case of operating on files from disk, the object program is never actually loaded into memory. Instead, tables are set up in memory describing various aspects of the program and the file is read, one sector at a time, as needed. These tables are used to build the display when an object dump or disassembly is performed. The net effect of this is that the original program, either on disk or in memory, is undisturbed. This means that the operating system code can be analyzed and "changed."

Once a program is loaded, it is usually necessary to classify all parts of the program according to usage. That is, each byte must be identified as data, variable, text, instruction, ect. This is so that when a disassembly is performed, the source code generated will correctly represent the program as it was originally written. Commands are included in Sleuth3 to classify memory. Once memory has been classified as to usage, the object code dump will indicate how each byte has been dedicated.

If the program is being modified, a screen edit capability is included to make the changes easier. The program is loaded and an object code dump is made of the sections where changes are necessary. The screen-edit mode is then entered and the cursor positioned to the location that needs to be changed. The new value is entered at the cursor location and is recorded. Any or all of up to 256 bytes within a given dump may be changed while in the screen-edit mode. If the cursor is in the hexadecimal area of the dump, the new values are entered in hex. If the cursor is positioned in the ASCII area of the dump, the new values are entered as ASCII characters.

Since the process of classifying all portions of a large program can be very tedious and time consuming, provision is made for storing all descriptive information about the program in a disk file. If it necessary to run Sleuth3 several times (quite likely when working on large programs such as “ShellPlus”), it is not necessary to reenter the various data or variable areas each time. They can be recalled immediately from the parameter file on disk.

In order to take advantage of position-independent coding which is supported by the 6809 microprocessor (and required by OS-9), a 6801/2/3/8/9 program processed by this system may be converted into position-independent source by setting the "6809 position-independence switch." Some careful checking must be done after this conversion to verify that the 6809 instructions are addressing data correctly. Some manual changes will usually be necessary to complete the conversion.

Very large programs or other system programs, generate source which is quite long--sometimes too long to fit on the disk. To accommodate this problem, source programs can be broken up into several parts or "segmented." This is accomplished by first by classifying the entire program as described above. Several trial disassemblies will, no doubt, be necessary to verify that all of the program has been classified properly. Once you are satisfied, new disassembly limits may be specified such that you are now only disassembling part of the program.

Each segment can be written to a disk file. By changing the disassembly limits, the entire program can be disassembled into source segments of manageable size. Each segment will have all the necessary equates to link it to the other segments.

When a program is disassembled by Sleuth3, it is very possible that not all the equates generated will appear in the same part of the listing. The source listing is output with all component parts in memory-address order. Consequently, low address equates will appear first in the listing while high address equates (most system and I/O calls) will appear at or near the end of the program. After the source has been written to disk, a text editor with block-move capability can be used to group the equates.

Labels generated by Sleuth3 will be of the form "Z[address]". This means that all the labels will start with a "Z" followed by a 4-digit hex number which represents the address at which the label was defined. If the program has been disassembled properly, when reassembled on the same type processor, all the labels should assemble at their corresponding addresses. This may vary however, when working with 6809 code. Not all the 6809 assemblers follow the same rules for defining the offset used with PC-relative addressing. Some assemblers may force a 16-bit offset when an 8-bit offset was used in the original code. This will cause a slight displacement of labels in the reassembled program and the displacement will increase as the program is processed. This problem can be alleviated by setting the Cross-Assembler flag "on" (see "B" command.) This combination should produce the correct length PCR code.

GETTING STARTED

Insert the disk containing the file Sleuth3 into the drive assigned as the default drive (normally "/d0") and copy Sleuth3 to the CMDS directory of your work disk or hard drive system. Example:

```
Copy /d0/cmds/sleuth3 /dd/cmds/sleuth3
```

Then, enter

```
tmode pau=1
```

This will make listing readable instead of just scrolling by too fast to read. To run Sleuth3, you type:

```
Sleuth3 #20k
```

This should give Sleuth3 enough ram to load most modules. If you find Sleuth3 is running out of memory on larger modules, then more memory can be assigned. A title and heading will be displayed. The system prompt "?" is then displayed, indicating that the system is ready to accept commands from the keyboard. For a list of commands (MENU), enter a "?" from the keyboard.

If you plan to use input or output disk files, insert the appropriate diskette into a disk drive. The disk containing Sleuth3 may be removed as it is no longer required. You are now ready to proceed with the operating session.

GENERAL NOTES ON OPERATION OF Sleuth3

At any time Sleuth3 is waiting for input from the keyboard, hitting BREAK will abort the operation in progress and return to the "?" prompt. At any time a non-hexadecimal number is entered when Sleuth3 is expecting a hexadecimal number, the current operation is aborted and the "?" prompt is displayed.

The operation of the space bar while in Sleuth3 is somewhat different from normal Coco operations. When Sleuth3 is outputting to the screen or printer, hitting the space bar will stop output temporarily. However, the operation of the break key after using the space bar varies, depending on the operation in progress. If you are doing a disassembly using the "D" command and no output file is used, hitting return terminates the operation and displays the "?" prompt. If an output file is in use while concurrently outputting to the screen and/or printer, hitting return will terminate the screen or printer output, but the disk output will continue. Once this has been done, the output to the screen or printer cannot be restarted until the disassembly is completed. In all other operations in Sleuth3, hitting return has the same effect as hitting the space bar a second time. If you want to terminate an operation, hit the break key until the operation is terminated; because of the unbuffered nature of the keyboard on the Coco, this will normally require several attempts.

During certain operations in Sleuth3, there may be times when nothing seems to be happening and the machine seems to have "died". This happens during a disassembly or when writing a new object file and a large section of "ignored: code is encountered. (See the "K" and "R" commands for information on ignored code). When this happens, and you think the system is "hung", wait at least five minutes before attempting any corrective action.

Sleuth3 COMMANDS

The Disassembler Command set is divided into the following four categories:

- Address Range Commands
- Mode Commands
- Operation Commands
- Miscellaneous Commands

Address range commands are used to classify memory as described in the overview or to change the disassembly range. Each command of this type will prompt for a starting and ending address. Thus a single byte or a group of bytes may be classified with one command. Mode commands are used to change the operating mode of Sleuth3. There are four different mode switching commands. Operation commands initiate some major operation, such as disassembling a program, executing an object code dump, making changes to the program, simulating a RESET operation, ect. Each of the commands will now be discussed in detail.

ADDRESS RANGE COMMANDS

Note--These commands can be used in any order, at any time that the "?" prompt is displayed. Each command will define one address range (Start-End) per use. Address range entries are terminated with a <return>. Any given address may be classified more than once. In this case, the Last classification entered for a particular byte or address range is the one that will be used by the disassembler.

A--set FDB Address Range

This command is used to define sections of code containing two-byte data items. These are usually 16-bit addresses in a table. Each FDB defined by this command will be assigned a label by Sleuth3. In 6502 mode, the 2-byte pairs will be reversed in sequence. Each byte within a specified FDB range will be indicated in an object code dump by placing the symbol "l" immediately after the hex value.

C--set FCC Address Range

This command is used to define sections of code containing text or ASCII data. Any code within the specified range which does not have an ASCII equivalent will automatically be marked as FCB's. Each byte within a specified FCC range will be indicated in an object code dump by placing the symbol "#" immediately after the hex value.

H--set FCB Address Range

This command is used to define sections of code which are used to store single byte hex data. Each byte within a specified FCB range will be indicated in an object code dump by placing the symbol "(" immediately after the hex value.

I--set INSTRUCTION Address Range

This command is used to identify sections of code which contain program instructions. This is the default classification for all of memory. Each byte within a specified INSTRUCTION range will be indicated in

an object code dump by placing the symbol ")" immediately after the hex value.

J--set INSTRUCTION & ASCII Range

This command is very similar to the "I" command with the difference that when a disassembly is performed, code marked with the "J" command will have the ASCII character equivalent of each byte of the instruction displayed to the right of the instruction. Each byte of code marked with the "J" command will be displayed in an object code dump by placing the symbol "*" immediately after the hex value.

K--set IGNORED Address Range

Frequently, an object program to be analyzed will contain more than one contiguous segment of code. These segments may be in completely different areas of memory. It is desirable to have Sleuth3 "ignore" anything between segments. When disassembling from disk, any address range not defined but between the start and end addresses are implicitly ignored by Sleuth3. The "K" command marks sections of memory which should be IGNORED by Sleuth3. Bytes marked by the "K" command are indicated in an object code dump by placing the symbol "+" immediately after the hex value.

R--set RMB Address Range

Frequently, an object program to be analyzed will contain more than one contiguous segment of code and may contain logically reserved areas of memory not represented by the binary object file. It is desirable to have Sleuth3 "ignore" anything between segments of code, yet it is desirable to be able to define these areas as RMB areas so that Sleuth3 will output a better representation of the program. Values of bytes found in RMB areas are ignored. Bytes marked by the "R" command are indicated in an object code dump by placing the symbol """ immediately after the hex value.

MODE COMMANDS

These commands are used to change the operating modes of Sleuth3. The current operating mode can be determined by using the "L" command described later in this manual.

B--Flip Cross-Assembler Switch

The disassembled code for the 6502 and 6805 options is oriented toward a 6809 macro assembler, rather than a 6502 or 6805 assembler. This is for the convenience of those who wish to do program development work for the 6502 or 6805 on a 6809. (Computer Systems Consultants markets cross-assembly macro sets for the 6800,6801,6805, and 6502 which run on a 6809 macro assembler.) When the "B" option of Sleuth3 is turned on, page zero addresses for 6800, 6801, 6805, and 6502 are indicated by the use of "<" prefixing the operand and extended addresses are indicated by ">" prefixing the operand. For the 6805 or 6809, eight-bit index offsets are indicated by "<" prefixing the operand while the sixteen-bit offsets are indicated by ">" prefixing the operand.

E--Flip Separate Label Switch

Many programmers prefer to assign the labels as an equate to the current program counter value rather than associate the label with a program instruction (e.g. LABEL EQU *). If the "E" switch is on, all program labels produced by Sleuth3 will be equated to the current pc value. If the "E" option switch is off, labels will be assigned to the current program instruction whenever possible.

P--Flip 6809 Position-Independence Switch

The "P" command is use to assist in the production of 6809 Position-Independent code. It makes the following changes in the output text:

1. All extended and direct addressing references to addresses within the program area are changed to program-counter-relative by adding ",PCR" after the operand.

2. All three-byte immediate instructions are changed to the corresponding PCR LEA instruction.

You must make the following changes to complete the conversion:

1. All references to FDB's within the program must be rewritten to be relocatable, perhaps through the use of program-counter-relative LEA instructions.
2. All old immediate and new LEA instructions must be reviewed to insure that the correct values are still loaded into various registers.
3. All out-of-program references must be reviewed to insure that they refer to truly constant addresses and not simply to program variable storage areas, which should be changed to PCR within the program.

6800/1/3/8 code may be converted to 6809 position-independent code in a similar manner but the resultant code must be checked very carefully to ensure the program logic has not been changed. 6805 and 6502 object code may not be processed in this manner. Any attempt to do so will be ignored.

Z--set Processor Type

The "Z" command specifies for which processor the current input file is written. The "Z" command will prompt for one of five choices. A "0" selects 6800, 6802, or 6808. A "1" selects 6801 or 6803. A "2" selects 6502. A "5" selects 6805. A "9" selects 6809. The default processor selected will be 6809

OPERATIONAL COMMANDS

These commands perform specific operations on memory or the input file.

D--Disassemble Program

The "D" command initiates the disassembly process. The user is asked for an output file name. If none is desired hit the return key. The following prompt will appear:

P(printer),B(both),T(terminal),N(none)

The "P" subcommand causes disassembler output to be sent to the printer. The "T" subcommand causes the output to be sent to the screen, while the "B" subcommand does both. If no output disk file is specified, the "N" subcommand causes the output to be sent to the screen. If an output file is used, the "N" subcommand will suppress the display of Sleuth3 output. If an output file has been specified, Sleuth3 will prompt for a title and assembler options. If a message of the form "TABLE OVERFLOW" appears, the input file has too many labels to process and must be redefined for smaller ranges of addresses and processed in parts.

F--Exit Coco Sleuth3

The "F" command terminates the current operation, closes any open files, and prepares for a RESET operation. It must be followed with a return. If you desire to save the current operating parameters or update the working file, these operations must be performed BEFORE the "F" command is used. Once the "F" is typed, all current parameters and work file are lost.

M--Examine and/or Change Program Code

The “M” command operates in a similar manner to the memory change function of most system monitors. The user is prompted for a starting address. The address entered is displayed followed by the hex value of the current contents of that location. To change the value, enter the new

2-digit hex value. The next memory location will then be displayed. Hitting any character except "^" or return or a valid hex digit causes the next sequential location to be displayed. Entering "^" causes the previous memory location to be displayed. Return terminates the examine/change mode.

Q--Object Code Display (Dump)

The "Q" command is used to invoke the object code display function. If an input file has been specified with the "S" (described later) command, the first 128 bytes of the input file will be displayed. If no input file has been specified, then a disassembly range must first be entered with the "N" command. More details on the use of the Object Code Dump are provided later.

T--Fill Program Code with a Specified Byte

This command will cause the specified byte to replace every byte within the address range specified. You will be prompted for starting address, ending address, and the byte you want to insert.

U--Display Directory of a Mounted Diskette

This command allows the display of the directory of a mounted diskette while still in Sleuth3. Either a drive number or return must be entered. If return is entered, drive zero is assumed.

V--View Object Code and Perform Absolute Disassembly

The "V" command will perform and absolute disassembly of the object code within the disassembly limits previously established. No labels are generated. All relative address instructions are resolved to absolute addresses. This provides a quick method of examining a section of code without doing a formal disassembly of the entire program. Memory classification is supported by the "V" command.

W--Write Modified Binary File to Disk

This command causes a new Coco binary file to be written to the disk. All changes made up to this point will be inserted into the file as it is

written, and all multiple byte definitions are resolved. The name of the output file is requested prior to writing the file. “W” also displays a memory map on the screen for the object file being written.

Y--Find Hexadecimal String in Program Code

This command finds all occurrences of a string up to 31 hex bytes within the current disassembly limits. The user is prompted for the search limits (start and end addresses) and the hex string for which to search. The hex string must be entered as continuous pairs of hex digits with no spaces intervening (e.g. 23DB2390A4). The starting address of all occurrences of the string are displayed on the screen.

MISCELLANEOUS COMMANDS

G--Specify Auxiliary Parameter File (Input or Output)

When the "G" command is entered, the program will prompt for an input file name. If only return is entered, the program will prompt for an output file name. The parameter file is used to store information on the current operating mode, input file name, classification of memory, and other operating parameters. As described earlier, it is sometimes necessary to repeat the memory classification and trial disassembly process several times before the disassembly comes out correctly. To avoid retyping the various memory classification commands each time, use the "G" command to save everything up to this point. Once the parameters have been saved, you may return to the exact point after restarting using the "G" command to retrieve the parameters previously saved.

L--List Current Control Information

The "L" command displays the current operating mode, disassembly limits, offset load values, and all memory classification ranges and types currently in effect. It also shows the memory changes that have been made with the "M", "Q", and "T" commands.

N--Set New Disassembly Range

This command defines the range of code that will be disassembled when the certain commands are used. The "N" command prompts for start and end addresses and for a transfer address. If no transfer address is desired, enter "FFFF".

O--Set Offset Load Value

The "O" provides an offset value which is added to each address in the program being processed. If the program is being processed from disk, the offset value is applied when the input file is loaded. If the program is being processed from main memory, the offset value may be changed as often as desired, since the offset value is applied during the actual process of acquiring data from memory.

S--Specify New Input File and do Partial Restart

The “S” command prompts for the file name of an input file to be used by Sleuth3. The file must be a Coco binary type of file. Any address range commands, separate-label switch, or position-independence switch settings previously in effect are cleared. The previous operating mode and address offset are preserved. If an input file is specified, the disk may not be removed until another “S” command is entered or Sleuth3 is terminated.

X--Specify New Transfer Address

The “X” command changes the transfer address or adds a transfer address to a file which previously did not have one. The transfer address is the initial program execution address.

OBJECT CODE DUMP AND SCREEN EDIT FUNCTIONS

The object code dump provides a "window" through which you can view a portion of memory or an object program in a convenient display format. For the Coco, this window consists of 16 lines of 8 bytes per line for a total of 128 bytes of code; special versions of Sleuth3 are available from CSC for smaller and larger displays.

If an input file has been specified, anything outside the disassembly limits will show as zeros. If an input file has not been specified, the current contents of memory will be displayed incorporating any changes made. As indicated previously, object code on disk or in memory is not actually changed, but the object code dump will show the changes entered.

After each byte in the hexadecimal display, there is a symbol indicating the classification of that byte. In the legend in the right margin of the display is a table showing these symbols and the command used to perform the classification. The current disassembly limits are displayed in the upper right corner of the display while a list of available commands is in the lower right corner.

SUB-COMMANDS

N--Display Next Page of Memory or File

This sub-command causes the next page of memory or the input file to be displayed. Hitting the return key will cause the same operation If the next page is outside of the disassembly range, the "Q" command terminates.

P--Display Previous Page of Memory or File

This sub-command causes the preceding page of memory or input file to be displayed. If the next page is outside of the disassembly range, the next page will be displayed.

(Hex Byte)--Display a Specified Page of Memory or File

If a two-digit hexadecimal value is entered, the corresponding page of memory will be displayed. For instance, if "4E" is entered, the page of memory at $4E00 will be displayed. The hex value entered should be within the disassembly limits. After using a hex-byte address, the "N", "P", and "S" sub-commands may be used as desired. If the requested page is outside the disassembly range, the next page will be displayed.

Q--Quit and Return to Main Command Mode

The "Q" sub-command returns control to the Sleuth3 main command interpreter. If the next page is outside of the disassembly range, control is also returned to the Sleuth3 command interpreter.

S--Full Screen Edit Mode

The "S" sub-command place the object code dump into a full screen edit mode of operation. The cursor will be positioned on the first nibble (digit) of the first byte in the upper left corner of the hex display. At this point, the cursor may be moved, using the cursor control keys you have defined, to any point within the hex or ASCII display fields. Note that the cursor will always point to valid data and cannot be moved outside the hex or ASCII display fields. Also the cursor will never point to a space between data. When the cursor has been positioned to the desired location, simply type the desired new data. If the cursor is in the hex display field, enter one or both nibbles of the new value, depending on whether the cursor is pointing to the first or second nibble of the hex byte. If the cursor is in the ASCII field, enter an ASCII character. Any or all bytes within the current page may be changed while in screen edit mode. After making the last change, hit return. The screen edit mode is exited and the current page is redisplayed showing the changes just made. To edit a different page of memory, use the "N", "P", or "Hex-byte" sub-commands to select the desired page and then "S" to enter the screen edit mode again. While in screen edit mode, none of the object code dump sub-commands may be used.

DISK FILES USED BY Sleuth3

During an operating session, Sleuth3 may use one to four different disk files. These include: Input file (binary), Output file (binary), Output file (text), and a Parameter file (text).

The Input file must be a Coco binary type of file containing machine language object code for the program to be disassembled or modified. BASIC programs, Text files, data files, and other non-binary files will not work. If an attempt is made to use a non-binary input file, the error message "Bas format!" is usually displayed.

The Output file produced by the "W" command is another binary file similar to the input file. Any changes made during the operating session will be included in this file. If a transfer address has been set, it will be recorded in the file.

The Output file produced by the "D" command is a Text file containing the source code produced by Sleuth3. This file may be immediately assembled by an appropriate assembler and should produce no errors. The file may be edited and/or modified as the user desires.

The Parameter file used by the "G" command is a text file containing the various operating parameters in effect at the time the file was produced. The data is stored in the file exactly as it would have been entered from the keyboard.

The disk error messages produced by Sleuth3, Chngnam3, and XRef3 are of the following form:
error XXX
where the XXX is one of the numeric codes documented in the OS-9 User's Manual and Programming Guide.

NAME CHANGER (Chngnam3)

The Name Changer is essentially a word substitute program. A table of words and desired substitutes is read into memory and then the file is read. All the words in the input file are checked against the substitution table and, if a match is found, the appropriate substitution is made. The principal use of this is in changing the machine-generated labels produced by Sleuth3 to standard labels that are more meaningful. Since Sleuth3 always produces the same label name for the same address, a standard file of labels may be maintained and used whenever desired.

GETTING STARTED

To start the program, simply insert the disk containing the program Chngnam3 into the system drive and enter:

```
Copy /dx/cmds/chgnam3 /dd/cmds/chgnam3
```

To run the program, just type:

```
Chgnam3 <oldpath> <newpath> <wordlist>
```

Operation is entirely automatic with no operator intervention required.

DISK FILES USED BY NAME CHANGER

Name Changer uses three disk files.

The first file is the control file (wordlist). This is a text file containing a table of text substitutions in the following format:

<delimiter> STRING <delimiter> NEWSTRING <delimiter><return>

Where <delimiter> may be any special character which does not appear in either the STRING or the NEWSTRING string and must be the same in all three locations; STRING may not be null, but NEWSTRING may be null; the total length is limited to 32 characters. Following are examples:

/ZD66C/DSKCON/
.THIS STRING WILL BE DELETED..

Since this file is placed into memory, the diskette containing it may be removed after it is read.

The second file is the Input file. This will usually be an assembly source file produced by Sleuth3; however any text file may be processed, including data files.

The third file is the Output file. This is the text file that will receive the modified text from the Input file.

OPERATING HINTS

If the message “MEMORY OVERFLOW” appears, too many entries are present in the control file. To expand Chgnam3's memory capacity, use the OS-9 memory modifier as the last item on the command line:

```
Chgnam3 <oldpath> <newpath> <wordlist> #xxk
(xx being the amount of memory requested)
```

The only limitation on the size of input and output text files which may be processed is imposed by the size of one disk drive each. Even in this case, large text files may be processed as smaller sub-files.

Frequently, in assembly language programming, reference is made to individual bytes of a multi-byte sequence of code or data. To do this, the first byte of the sequence is normally assigned a label and successive bytes are addressed as that label plus an offset (e.g. LABEL+1). Sleuth3 and most other disassemblers have no way of recognizing this convention and will always assign a separate label to each byte so referenced.

When using the Name Changer, it may be desirable, for increased clarity, to restore the original labeling convention. This is done by substituting the desired label for the first byte of the sequence and then substituting the same label plus the appropriate offset for the labels that the disassembler assigned to the other bytes. A problem arises here, however, which must be dealt with prior to reassembling the program.

Most assemblers will not permit the form “LABEL+1” in the label field of the source program. Consequently, after making the changes described above, it may be necessary to use a text editor to delete the equates with the offset labels. Do not delete the equate that defines the original label. The offset labels are permitted in the operand field since most assemblers allow and evaluate expressions in that field.

Several examples of using the wodlist file are in the "EXAMPL" directory on the disk.

CROSS-REFERENCE GENERATOR (XRefs3)

XRefs3 processes an assembly language source file and produces a sorted list of labels found in that file, the line number where the label is defined, and the line numbers of all lines in the program that refer to that label. Any source file that follows the Motorola source code format may be processed with this program. Labels are restricted to 8 characters.

GETTING STARTED

To start the program, simply insert the disk containing the program XRefs3 into the system drive and enter:

```
Copy /dx/cmds/XRefs3 /dd/cmds/XRefs3
```

To run the program, just type:

```
XRefs3 <infile >outfile
```

Entering no output file name will send the output to the screen. An output disk file contains the cross reference listing, which may be saved for later reference and/or printed.

Program operation, with minor exceptions, is entirely automatic.

OPERATING HINTS

If the message “MEMORY OVERFLOW” appears, too many entries are present in the input file. To expand Xrefs3's memory capacity, use the OS-9 memory modifier as the last item on the command line:

```
XRefs3 <infile >outfile #xxk
(xx being the amount of memory requested)
```

The only limitation on the size of input and output text files which may be processed is imposed by the size of one diskette each. However, the input diskette may be removed when the output file name is requested.

ADAPTING Sleuth3, Chngnam3, AND XRef3 TO YOUR SYSTEM

The first few bytes of the object files of Sleuth3, Chngnam3, and XRef3 contain the information which may be required to adapt them to your system. The object file editing capabilities of LSEUTH may be used to perform the modifications, as required. Following are the addresses and contents of this information:

Address Offset	Contents
0002	Program version number
0003-4	Serial port (printer) baud rate
	01CA = 110 baud
	00BE = 300 baud
	0057 = 600 baud (default)
	0029 = 1200 baud
	0012 = 2400 baud
0005	Bits per byte on serial port
	07 = 7 bits/byte
	08 = 8 bits/byte (default)
0006-9	Disk step rate (drives 0-3)
	00 = 06 millisec.
	01 = 12 millisec.
	02 = 20 millisec.
	03 = 30 millisec.
000A-B	Table start address (default 0000)
000C-D	Table end address (default 1F00 for Sleuth3)

When modifying this information, be careful not to modify any other of the contents of Sleuth3, Chngnam3, or XRef3. Also, be sure to keep the original versions of the programs on the original disk in case you make an error in modifying them or need them to run on slower disk drives in the future.

Sleuth3 does not actually modify memory with the “Q”, “M”, and “T” commands; rather, it records the changes in a table and applies them when the object program is written with the “W” command or disassembled with the “V” or “T” command. Thus, before the change will be effective, the program must be re-executed from the new object file.

The source files processed by Coco Sleuth3 contain a carriage return following each line of text. This is also the format required by most of the current Coco assemblers and editors. If your assembler or editor requires a different format, it should be very simple to write a BASIC program to reformat the file produced by Sleuth3 to be compatible with your assembler or editor, or vice versa.

If your printer will not work properly with Sleuth3, contact CSC. There is a tremendous variation among printers which may be attached to the Coco. A standard printer driver is provided, but it may be possible to easily modify it to drive your printer. Luckily, a printer is not essential to the use of Sleuth3.

Coco Sleuth3 COMMAND SUMMARY

OPERATIONAL COMMANDS

D--Perform full disassembly
F--Exit Coco Sleuth3
Q--Edit object code dump
M--Query/modify object code
T--Fill address range with hex value
U--List directory of a mounted diskette
V--View object code and perform absolute disassembly
W--Write new object code file
Y--Find hex string in object code

ADDRESS RANGE COMMANDS

A--Classify as FDB
C--Classify as FCC
H--Classify as FCB
I--Classify as Instruction
J--Classify as Instruction + ASCII
K--Classify as Killed or Ignored
R--Classify as RMB

MODE CHANGE COMMANDS

B--Flip cross-assembler switch
E--Flip separate-label switch
P--Flip position-independent switch
Z--Select CPU mode

MISCELLANIOUS COMMANDS

G--Specify auxiliary input/output file
L--List control information

N--Set new disassembly range
O--Set offset load value
S--Specify input file name
X--Set transfer address