# 68010 SUPER SLEUTH

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

## Copyright Notice

### *Limited Warranty Statement*

### *Problems and Improvements*

Users are encouraged to submit problems and to suggest or to provide improvements for 68010 SUPER SLEUTH. 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 68010 SUPER SLEUTH. In the case of specific situations requiring extensions to 68010 SUPER SLEUTH or major assistance in its use, consulting is available on a pre-arranged, for-fee basis.

## 68010 SUPER SLEUTH -- INTRODUCTION

A disassembler performs a function opposite to that of an assembler. That is, it allows its user to process an object file, for which no source program is readily available, and to re-create the essentials of the original source file. The resultant source file may then be studied, commented, modified, reassembled, etc. A trivial disassembly may be performed rather simply manually or with the use of a computer by translating the successive bytes of object code into the equivalent mnemonics for a target processor. A really useful disassembly requires the assignment of labels and the ability to determine whether given bytes in the object file represent data, addresses, garbage, or instructions. Since a computer can assign labels but cannot classify object file contents, disassembly must be interactive to be meaningful.

68010 SUPER SLEUTH is a collection of programs which enables the user to examine and process 68000, 68008, and 68010 object files into more meaningful assembler source code, with labels, which may be examined, modified, and reassembled into object files.

The programs include DIS68K (the Disassembler), VARIABLE (the Label-Changer), XREF (the Cross Reference Generator), and various utility programs to assist in preprocessing the object files into formats acceptable to DIS68K. The functions of all of them are described below.

## DIS68K -- DISASSEMBLER

This program is the heart of the 68010 SUPER SLEUTH package. Some of its functions, which may be requested from the command line or interactively, include the following:

- Disassemble object files and write the source to a file.
- Disassemble object files and write the listing to a file.
- Disassemble object files and display the source.
- View object files in instruction format.
- Dump object files in hex-ascii format.

The object files are never loaded into memory in their entirety. Thus, there is no inherent limitation on the size of the object files which may be processed. The primary limitation is the number of labels encountered in the processing of the object files.

Once object files are loaded, it is necessary to classify all address ranges according to usage before they may be meaningfully disassembled. That is, each byte must be identified as data, variable, text, instruction, etc., although all bytes are assumed to represent instructions by default. Commands are provided in DIS68K to classify memory, as described above.

Since the process of classifying all portions of a large set of object files can be very tedious and time-consuming, DIS68K supports storing most descriptive information in a parameter file. If it is necessary to run DIS68K several times, which is quite likely when processing large sets of object files, it is not necessary to manually reclassify the various data or variable areas each time. They can be recalled immediately from the parameter file.

Very large sets of object files, such as system programs, generate source which is quite long -- sometimes too long to conveniently process as one large file. To alleviate this problem, large sets of object files may be segmented. This is accomplished by first classifying the entire set of object files, as described above. Several trial disassemblies are normally necessary to verify that all of the object files' contents have been classified properly. Once this is true, new disassembly limits may be specified such that only part of the object files are being disassembled. Each segment can then be generated as separate source files. By changing the disassembly limits, the entire set of object files can be disassembled into source segments of manageable size. Each segment will have all the necessary equates to link it to the other segments. However, it is quite possible that not all equates generated will appear in the same part of the listing.

Labels generated by DIS68K have the form "z[address]" for absolute addresses and "x[address]" for PC-relative addresses. Thus, all labels will start with a "z" or "x", followed by the hexadecimal address at which the label is defined. If the set of object files as been disassembled properly, all labels should reassemble to their corresponding addresses. This may vary, however, when working with different assemblers. For example, some assemblers may use a 16-bit offset when an 8-bit offset was used in the original code, or vice versa. This will cause a slight displacement of labels in the reassembled program and the displacement may change with higher addresses; however, the code should execute correctly.

## USING DIS68K

To use DIS68K, enter a command line in a format similar to the following:

```
dis68k [-options] [filename-list]
```

If this command line format is forgotten, enter just DIS68K and it will provide the following information:

```
dis68k disassembles 68010 and similar machine code
     so that the new code may be correctly reassembled.

Usage: dis68k [-options] [object file list]
       -d  (debug mode)
       -g generated code filename
       -i  (interactive mode)
       -l listing filename
       -o1 (object file type S1, S2, S3 (default))
       -p program profile filename
       -q  (view object file in hexadecimal)
       -r relocation address
       -t transfer address
       -v  (view object file without labels)
       -w  (label 16-bit immediates)
       -x  (make relative labels absolute)
E.g.: dis68k -llisting -w -x objfile.mxt
```

## DIS68K COMMANDS

DIS68K accepts commands, options, and file names from the command line or interactively from the terminal.

The command line parameters are as follows:

```
          object file list

       -d  (debug mode)
       -g generated code filename
       -i  (interactive mode)
       -l listing filename
       -o1 (object file type S1, S2, S3 (default))
       -p program profile filename
       -q  (view object file in hexadecimal)
       -r relocation address
       -t transfer address
       -v  (view object file without labels)
       -w  (label 16-bit immediates)
       -x  (make relative labels absolute)
```

The commands and options are described later in this manual. The object file list is the set of object files to be processed during the current execution. This set may be defined or redefined by interactive commands. Note that not all of these parameters are available on systems with restricted memory.

The interactive mode commands are as follows:

```
       Interactive Mode:

        a? [start end]  request [set] memory attributes:
          1,2,4 hex bytes (dc.b,dc.w,dc.l)
          c     ascii bytes (dc.b)
          i     instructions
          l,w   extended addresses (dc.l,dc.w)
          s     skipped bytes (ds.b)
        c [command]  send command to O/S
        d[-] [start end]  disassemble file [w/o labels]
        g [filename]  set generated code filename
        l [filename]  set listing filename
        m  enter manual mode
        o? [filename]  set object type [and filename]:
          1 S1, S2, S3 (default)
        p [filename]  read profile
        q [start end]  hex dump file
        r [address]  set relocation address
        t [address]  set transfer address
        u [filename]  update profile
        w[-]  [do not] label 16-bit immediates
        x[-]  make relative labels absolute [relative]
        + -  change debugging level
        *  exit
```

The interactive mode commands are described later in this manual. Note that not all of these commands are available on systems with restricted memory.

The manual mode commands are as follows:

```
       Manual Mode:

        code,...,code  opcode to disassemble (in hex)
        =program-ctr   change program counter
        + -            change debug level
        *              exit
```

The manual mode commands are described later in this manual. Note that not manual mode is not available on systems with restricted memory.

## *COMMAND LINE PARAMETERS*

Command line commands and options may be coded in either upper or lower case. File name case is critical if the operating system under which DIS68K is being run distinguishes between case in file names.

If neither "-i" nor "-p" parameter is coded, all commands, options, and file names must be provided on the command line.

-d (debug mode)
The "-d" command increments the debug indicator. Each increment of the debug counter (up to 3) provides more detailed description of the actions of DIS68K.

-g generated code filename
The "-g" command provides the name of the file which will contain the disassembled source code. No space may appear between the "g" and the file name.

-i (interactive mode)
The "-i" command causes DIS68K to enter interactive mode (described later in this manual) after all parameters on the command line have been processed.

-l listing filename
The "-l" command provides the name of the file which will contain the disassembled source code listing. No space may appear between the "l" and the file name.

-o1 (object file type S1, S2, S3 (default))
The "-o" option sets the type of object file being processed. No space may appear between the "o" and the option value. The only option value currently accepted is "1", which specifies an object file type of S1, S2, or S3.

-p program profile filename
The "-p" command provides the name of the file which contains the program parameter file, or "profile". No space may appear between the "p" and the file name. The program profile, if specified, is read after after all parameters on the command line have been processed.

-q (view object file in hexadecimal)
The "-q" command provides a set of windows through each of which may be viewed a portion of an object file in a convenient display format. This window consists of 16 lines of 16 bytes per line for a total of 256 bytes of code. A CRT terminal with a minimum of 24 lines of 80 characters per line is required for proper display.

-r relocation address
The "-r" command sets the relocation address, which is a 32-bit value added to all PC-relative addresses encountered in the disassembly. No space may appear between the "r" and the address, which must be coded in hexadecimal. The default relocation address is zero.

-t transfer address
The "-t" command sets the relocation address, which is a 32-bit value placed on the "end" statement of the disassembly. No space may appear between the "t" and the address, which must be coded in hexadecimal. The default transfer address is zero, unless one is coded on a S7, S8, or S9 record.

-v (view object file without labels)
The "-v" option causes the disassembly to ignore any label references. This is often used in the early steps of disassembling a set of object files, when many label references are false and obscure the structure of the object code.

-w (label 16-bit immediates)
The "-w" option causes 16-bit immediate addresses to be converted to absolute labels. By default, this action does not occur.

-x (make relative labels absolute)
The "-x" option causes relative address references to be converted to absolute address references. This is useful when the object files being disassembled are known to be written using absolute addressing, not relative addressing, as the two types of labels will then be merged.

## *INTERACTIVE MODE COMMANDS*

Interactive mode commands may be coded in either upper or lower case. File name case is critical if the operating system under which DIS68K is being executed distinguishes between cases in file names. Each command must each be terminated with a new line character.

```
a? [start end]  request [set] memory attributes:
  1,2,4 hex bytes (dc.b,dc.w,dc.l)
  c     ascii bytes (dc.b)
  i     instructions
  l,w   extended addresses (dc.l,dc.w)
  s     skipped bytes (ds.b)
```

The "a" command allows the user to request or set memory attributes. If "a" is entered without a parameter, a list representing all memory attributes and their corresponding address ranges is output. If "a" is entered with a parameter but without an address, a list representing only that memory attribute and its corresponding address ranges is output. If "a" is entered with a parameter and address range, an entry in the list is made for that memory attribute and 32-bit address range. If an address range is entered, the starting address must be entered before the ending address. The last definition of an address overrides previous definitions. By default, all memory addresses have type instruction.

```
c [command]  send command to O/S
```

The "c" command causes the string of characters (if not whitespace) to be sent to the operating system under which DIS68K is being executed. It is the user's responsibility to ensure that nothing performed by this command interferes with the operation of DIS68K.

```
d[-] [start end]  disassemble file [w/o labels]
```

The "d" command disassembles the current set of object files, using the current list of memory attributes. If a 32-bit address range is specified, the disassembly is restricted to that range. If "-" is entered, the existence of labels in the disassembled program is ignored; this is useful in the early stages of disassembling a new set of object files, in that many label references are misleading because the list of memory attributes has not been completed.

```
g [filename]  set generated code filename
```

The "g" command specifies the name of the file into which DIS68K will place the disassembled source code whenever the "d" command is issued. No space should appear between the "g" and the file name. If no file name is provided for the disassembled source code, DIS68K will not write the disassembled source code to a file.

```
        l [filename]  set listing filename
```

The "l" command specifies the name of the file into which DIS68K will place the disassembled source listing whenever the "d" command is issued. No space should appear between the "l" and the file name. If no file name is provided for the disassembled source listing, DIS68K will not write the disassembled source listing to a file.

```
        m  enter manual mode
```

The "m" command causes DIS68K to enter manual mode, which allows the entry and disassembly of individual instructions. This mode is described later in this manual.

```
        o? [filename]  set object type [and filename]:
          1 S1, S2, S3 (default)
```

The "o" command allows the specification of the type and filenames of the set of object files. The only type currently supported is "1", which specifies S1, S2, and S3 records. If type or filename list is not specified, the current type or filename list is used.

```
        p [filename]  read profile
```

The "p" command causes DIS68K to read the specified file and interpret the records in that file as interactive mode commands. This facility allows the user to create parameter files to save and to restore various operational parameters in a disassembly session.

```
        q [start end]  hex dump file
```

The "q" command provides a set of windows through each of which may be viewed a portion of an object file in a convenient display format. If a 32-bit address range is specified, the dump is restricted to that range. This window consists of 16 lines of 16 bytes per line for a total of 256 bytes of code. A CRT terminal with a minimum of 24 lines of 80 characters per line is required for proper display.

```
        r [address]  set relocation address
```

The "r" command sets the relocation address, which is a 32-bit value added to all PC-relative addresses encountered in the disassembly. No space may appear between the "r" and the address, which must be coded in hexadecimal. If no relocation address is specified, zero is assumed.

```
        t [address]  set transfer address
```

The "t" command sets the relocation address, which is a 32-bit value placed on the "end" statement of the disassembly. No space may appear between the "t" and the address, which must be coded in hexadecimal. The default transfer address is zero, unless one is coded on a S7, S8, or S9 record.

```
        u [filename]  update profile
```

The current memory address ranges and types and the current option settings are written to the specified file name. No spaces should appear between the "u" and the file name. This file may be reloaded with the "p" command at a later time to re-establish the environment at the point at which the "u" command was given.

```
        w[-]  [do not] label 16-bit immediates
```

The "w" command causes 16-bit immediate addresses to be converted to absolute labels. By default or if "-" immediately follows the "w", this action does not occur.

**`x[-]  make relative labels absolute [relative]`**

The "x" command causes relative address references to be converted to absolute address references. This is useful when the object files being disassembled are known to be written using absolute addressing, not relative addressing, as the two types of labels will then be merged. By default, or if "-" immediately follows "x", relative address references are considered separate from absolute address references.

**`+ -  change debugging level`**

The "+" and "-" commands cause the debugging level to be incremented or decremented, respectively. Incrementing the debugging level (up to 3) causes DIS68K to output increasing amounts of information about its internal operations.

**`*  exit`**

The "*" command causes DIS68K to terminate. To help prevent a premature exit, it must be entered twice in a row. The first entry causes a prompt to be sent to the terminal, and the second entry causes DIS68K to terminate.

## *MANUAL MODE*

The "m" command causes DIS68K to enter manual mode. In this mode, the user enters hex opcodes and operands and DIS68K immediately disassembles them. Each command must each be terminated with a new line character. It does not write the disassembled code to the output or listing files, nor does it affect the contents of memory, nor is it affected by memory ranges and types.

**`code,...,code  opcode to disassemble (in hex)`**

A sequence of 16-bit hex codes, separated by commas, is processed as an instruction. If an insufficient number of bytes is entered, the remainder of the instruction is assumed to be zero. The pseudo program counter is incremented by the length of the disassembled instruction.

**`=program-ctr   change program counter`**

The "=" command causes the pseudo program counter to be set to the 32-bit address following it.

**`+ -            change debug level`**

The "+" and "-" commands cause the debugging level to be incremented or decremented, respectively. Incrementing the debugging level (up to 3) causes DIS68K to output increasing amounts of information about its internal operations.

**`*              exit`**

The "*" command causes an immediate return to interactive mode.

## EXAMPLE OF USE OF DIS68K

As a simple example of the use of DIS68K, consider the following short example of a 68010 program:

```
00000000                             org.l   $00000800
00000000 48E71800        decconv  movem.l d3-d4,-(sp) save regist
00000004 7200                     moveq   #0,d1 clear answer
00000006 74FF                     moveq   #-1,d2 no decimal to st
00000008 0C190020        skipws   cmp.b   #' ',(a1)+ look for spa
0000000C 67FA                     beq.s   skipws skip on...
0000000E 1F21                     move.b  -(a1),-(sp) save (possi
00000010 0C11002B                 cmp.b   #'+',(a1) look for pos.
```

```
00000014 6706                    beq.s   skipsign skip sign
00000016 0C11002D                cmp.b   #'-',(a1) look for neg.
0000001A 6602                    bne.s   convert ready for first
0000001C 5249           skipsign addq    #1,a1 inc string pointe
0000001E 1619           convert  move.b  (a1)+,d3 into data reg
00000020 0C03002E                cmp.b   #'.',d3 look for decima
00000024 6604                    bne.s   digit nope
00000026 7400                    moveq   #0,d2 clear count
00000028 60F4                    bra.s   convert continue digit
0000002A 0C030030       digit    cmp.b   #'0',d3 low end numeral
0000002E 6500002C                bcs     scandone less than '0'
00000032 0C030039                cmp.b   #'9',d3 high end
00000036 62000024                bhi     scandone greater than
0000003A 02830000000F            and.l   #$0f,d3 make binary
00000040 2801                    move.l  d1,d4 get high word...
00000042 4844                    swap    d4 ... into low word
00000044 C2FC000A                mulu    #10,d1 previous count,
00000048 C8FC000A                mulu    #10,d4 high word also
0000004C 4841                    swap    d1 high word into low,
0000004E D244                    add.w   d4,d1 add high to d4/lo
00000050 4841                    swap    d1 restore order
00000052 D283                    add.l   d3,d1 add new digit
00000054 4A42                    tst     d2 past decimal?
00000056 6BC6                    bmi.s   convert not yet
00000058 5202                    addq.b  #1,d2 yes, one more pas
0000005A 60C2                    bra.s   convert again
0000005C 5389           scandone subq.l  #1,a1 align on first
0000005E 161F                    move.b  (sp)+,d3 get sign
00000060 0C03002D                cmp.b   #'-',d3 was it neg?
00000064 6602                    bne.s   unneg no...
00000066 4481                    neg.l   d1 yes...
00000068 4CDF0018       unneg    movem.l (sp)+,d3-d4 restore wor
0000006C 4E75                    rts
0000006E                         end
```

This program may be compiled into the following S3 records:

```
S31B0000080048E71800720074FF0C19002067FA1F210C11002B670615
S31D000008160C11002D6602524916190C03002E6604740060F40C0300309A
S31B0000082E6500002C0C03003962000024028300000000F2801484406
S31D00000844C2FC000AC8FC000A4841D2444841D2834A426BC6520260C250
S3170000085C5389161F0C03002D660244814CDF00184E7504
```

DIS68K produces the following source code file in response to the "d" command:

```
xf EQU $f
 ORG.L $800
 MOVEM.L D3/D4,-(A7)
 MOVEQ #$0,D1
 MOVEQ #-$1,D2
r808 EQU *
 CMPI.B #$20,(A1)+
 BEQ.S r808
 MOVE.B -(A1),-(A7)
 CMPI.B #$2b,(A1)
 BEQ.S r81c
 CMPI.B #$2d,(A1)
 BNE.S r81e
r81c EQU *
 ADDQ.W #1,A1
r81e EQU *
 MOVE.B (A1)+,D3
 CMPI.B #$2e,D3
 BNE.S r82a
 MOVEQ #$0,D2
 BRA.S r81e
r82a EQU *
 CMPI.B #$30,D3
 BCS r85c
 CMPI.B #$39,D3
 BHI r85c
 ANDI.L #xf,D3
 MOVE.L D1,D4
 SWAP D4
 MULU #$a,D1
 MULU #$a,D4
 SWAP D1
 ADD.W D4,D1
 SWAP D1
 ADD.L D3,D1
 TST.W D2
 BMI.S r81e
 ADDQ.B #1,D2
 BRA.S r81e
r85c EQU *
 SUBQ.L #1,A1
 MOVE.B (A7)+,D3
 CMPI.B #$2d,D3
 BNE.S r868
 NEG.L D1
r868 EQU *
 MOVEM.L (A7)+,D4/D3
 RTS
 END
```

DIS68K provides the following disassembled listing file:

```
0000000f                      xf         EQU      $f
00000800                                 ORG.L    $800
00000800 48e71800                        MOVEM.L  D3/D4,-(A7)
00000804 7200                            MOVEQ    #$0,D1
00000806 74ff                            MOVEQ    #-$1,D2
00000808                      r808       EQU      *
00000808 0c190020                        CMPI.B   #$20,(A1)+
0000080c 67fa                            BEQ.S    r808
0000080e 1f21                            MOVE.B   -(A1),-(A7)
00000810 0c11002b                        CMPI.B   #$2b,(A1)
00000814 6706                            BEQ.S    r81c
00000816 0c11002d                        CMPI.B   #$2d,(A1)
0000081a 6602                            BNE.S    r81e
0000081c                      r81c       EQU      *
0000081c 5249                            ADDQ.W   #1,A1
0000081e                      r81e       EQU      *
0000081e 1619                            MOVE.B   (A1)+,D3
00000820 0c03002e                        CMPI.B   #$2e,D3
00000824 6604                            BNE.S    r82a
00000826 7400                            MOVEQ    #$0,D2
00000828 60f4                            BRA.S    r81e
0000082a                      r82a       EQU      *
0000082a 0c030030                        CMPI.B   #$30,D3
0000082e 6500002c                        BCS      r85c
00000832 0c030039                        CMPI.B   #$39,D3
00000836 62000024                        BHI      r85c
0000083a 02830000000f                    ANDI.L   #xf,D3
00000840 2801                            MOVE.L   D1,D4
00000842 4844                            SWAP     D4
```

```
00000844 c2fc000a                     MULU     #$a,D1
00000848 c8fc000a                     MULU     #$a,D4
0000084c 4841                         SWAP     D1
0000084e d244                         ADD.W    D4,D1
00000850 4841                         SWAP     D1
00000852 d283                         ADD.L    D3,D1
00000854 4a42                         TST.W    D2
00000856 6bc6                         BMI.S    r81e
00000858 5202                         ADDQ.B   #1,D2
0000085a 60c2                         BRA.S    r81e
0000085c                   r85c       EQU      *
0000085c 5389                         SUBQ.L   #1,A1
0000085e 161f                         MOVE.B   (A7)+,D3
00000860 0c03002d                     CMPI.B   #$2d,D3
00000864 6602                         BNE.S    r868
00000866 4481                         NEG.L    D1
00000868                   r868       EQU      *
00000868 4cdf0018                     MOVEM.L  (A7)+,D4/D3
0000086c 4e75                         RTS
0000086e                              END
```

DIS68K produces the following object file in response to the "q" command:

```
0000|00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f  0123456789abcdef
---- -- -- -- --  -- -- -- --  -- -- -- --  -- -- -- --  ----------------
0800 48 e7 18 00  72 00 74 ff  0c 19 00 20  67 fa 1f 21  H...r.t.... g..!
0810 0c 11 00 2b  67 06 0c 11  00 2d 66 02  52 49 16 19  ...+g....-f.RI..
0820 0c 03 00 2e  66 04 74 00  60 f4 0c 03  00 30 65 00  ....f.t.`....0e.
0830 00 2c 0c 03  00 39 62 00  00 24 02 83  00 00 00 0f  .,...9b..$......
0840 28 01 48 44  c2 fc 00 0a  c8 fc 00 0a  48 41 d2 44  (.HD........HA.D
0850 48 41 d2 83  4a 42 6b c6  52 02 60 c2  53 89 16 1f  HA..JBk.R.`.S...
0860 0c 03 00 2d  66 02 44 81  4c df 00 18  4e 75 00 00  ...-f.D.L...Nu..
```

## FILES USED BY DIS68K

During an operating session, DIS68K may use several classes of files. These classes include the following: Input file, Output file, Listing file, Control file, and Parameter file.

The Input file must be a set of S1, S2, or S3 text files containing machine language object code for the object files to be disassembled or examined. Although not absolutely essential, it should be in order of increasing starting addresses. Utilities are provided to convert several common formats into S1, S2, or S3 format, and are described below.

The Output file contains the source code produced by DIS68K. This file may be processed by an appropriate assembler and should produce no syntax or logical errors, if it has been disassembled properly.

The Listing file contains the source code produced by DIS68K, formatted as if it were assembled. It contains the same output as is produced to the terminal during a disassembly operation.

The Control file contains a table used by DIS68K read once per execution. This file controls many of the aspects of the disassembly process. By default, it must be in the default directory when required, and is named DIS68K.OPC.

The Parameter file contains the operating parameters in effect at the time the file was produced. The data is stored in the file exactly as commands would have been entered from a keyboard. It may be recalled at any time during the execution of DIS68K.

## VARIABLE -- LABEL CHANGER

VARIABLE is essentially a word substitution program. A table of words and the desired substitutes is read into memory and then the input file is read. All 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 DIS68K to standard labels that are more meaningful. Since DIS68K always produces the same label name for the same absolute or evaluated relative address, consistent files of labels may be generated and used.

### *USING VARIABLE*

To use VARIABLE, enter a command line in a format similar to the following:

```
VARIABLE input wordlist output -n
```

If this command line format is forgotten, enter just VARIABLE and it will provide the appropriate command line format for entering the following file names: the input file (file to be processed), the wordlist file (substitution table), and the output file (new file with changes made).

The input and output file names must not be the same file name, or the input file contents will almost certainly be destroyed. The wordlist file is a text file containing a table of text substitutions with records of the following format:

```
old-name new-name new-line
```

Frequently in assembly language programming, references to individual bytes of a multi-byte sequence of code or data are made. To do this, the first byte of the sequence is normally assigned a label and succesive bytes are addressed as that label plus an offset. DIS68K and most other disassemblers have no method of recognizing this convention and will assign a separate label to each byte so referenced, which is acceptable to most assemblers.

When using VARIABLE, it may be desirable for increased clarity to restore the original labeling, without the offset values. This is done by substituting the desired labels 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 may arise which must be dealt with prior to reassembling the source file. Assemblers will not permit expressions in the label field of the source file. Consequently, after making the changes described above, it may be necessary to use a text editor to delete the equates with the offset labels, but not the equate that defines the original label. Offset labels are permitted in the operand field since most assemblers allow and evaluate expressions in that field.

## XREF -- CROSS-REFERENCE GENERATOR

XREF processes an assembly language source file and produces a sorted list of labels found in that file and the line numbers of all lines in the file containing that label. Any source file which follows the Motorola source code format may be processed with XREF.

### *USING XREF*

To use XREF, enter a command line in a format similar to the following:

```
XREF -a filename-list
```

If this command line format is forgotten, enter just XREF and it will provide the following information:

```
xref cross-references a c, basic, assembler, or text file.
use: xref [-option [-option ...]] filename-list
```

```
-a  for non-intel assembler program.
-ai  for intel format.
-b  for basic program.
-bn  for xref by statement rather than line number.
-c  for c program.
-f  for inclusion of file names in xref.
-kfilename for kill-list file.
-l  for listing of input files with line numbers.
-pxxx  for overriding page depth (xxx = lines).
-t  for text file (default).
-u  for forcing upper to lower case.
-wxxx  for overriding page width (xxx = columns).
-x  for insertion of line after each xref item.
```

## UTILITIES

Following is a partial list of the utilities provided with 68010 SUPER SLEUTH, along with a brief description of each.

| | |
|---|---|
| S1FLEX | converts S1, S2, S3 records to FLEX binary format |
| S1INTEL | converts S1, S2, S3 records to INTEL format |
| S1LOAD | converts S1, S2, S3 records to straight binary format |
| S1UNFLEX | converts FLEX binary format to S1 records |
| S1UNLOAD | converts straight binary format to S1 records |
| S2FMBIN | converts binary format from S1, S2, S3 records |
| S2TOBIN | converts binary format to S1, S2, S3 records |

### *USING THE UTILITIES*

To use a given utility program, enter a command line consisting only of its name, selected from the list above, and it will provide a more complete description of its use, in addition to complete command line format information.

## DIS68K USAGE SUMMARY

| |
|---|
| **Command Line Parameters:** |
| |
| object file list |
| |
| -d  (debug mode) |
| -g generated code filename |
| -i  (interactive mode) |
| -l listing filename |
| -o1 (object file type S1, S2, S3 (default)) |
| -p program profile filename |
| -q  (view object file in hexadecimal) |
| -r relocation address |
| -t transfer address |
| -v  (view object file without labels) |
| -w  (label 16-bit immediates) |
| -x  (make relative labels absolute) |
| |
| **Interactive Mode:** |
| |
| a? [start end]  request [set] memory attributes: |

```
   1,2,4 hex bytes (dc.b,dc.w,dc.l)
   c     ascii bytes (dc.b)
   i     instructions
   l,w   extended addresses (dc.l,dc.w)
   s     skipped bytes (ds.b)
 c [command]  send command to O/S
 d[-] [start end]  disassemble file [w/o labels]
 g [filename]  set generated code filename
 l [filename]  set listing filename
 m  enter manual mode
 o? [filename]  set object type [and filename]:
   1 S1, S2, S3 (default)
 p [filename]  read profile
 q [start end]  hex dump file
 r [address]  set relocation address
 t [address]  set transfer address
 u [filename]  update profile
 w[-]  [do not] label 16-bit immediates
 x[-]  make relative labels absolute [relative]
 + -  change debugging level
 *  exit

Manual Mode:

 code,...,code  opcode to disassemble (in hex)
 =program-ctr   change program counter
 + -            change debug level
 *              exit
```