INTRODUCTION

*** REVAS ***
REVERSE ASSEMBLER FOR Z80 OBJECT CODE PROGRAMS

REVAS is an interactive reverse assembler (disassembler) designed to translate Z80 or 8080 machine language code into an assembly-like listing. It is written in Z80 code and can be used in any system that uses a Z80 central processor. It supports a punch and line printer as well as a CRT or TTY.

With 22 commands, REVAS can help you:

**Analyse undocumented programs

**Document your machine language patches

**Document your special I/O routines

**Debug developmental programs

**Modify and relocate your software

Here are some of REVAS' features:

**Assembly format listings

**Output suitable for reassembly

**Generates synthetic labels

**Accepts your choice of real labels

**Prints tables in data format

**Displays alphanumeric equivalents of the machine code

**Displays symbol table at any time

**Cross reference listing shows where and how each symbol is used

**Up to 3 output devices can be used

**You are always in complete control of the disassembly process..stop and restart, return to monitor, or return to command mode at will

The instruction mnemonics produced by REVAS are the same as those used by the Technical Design Labs' Z-80 Relocating Macro Assembler, and by Intel for the 8080.

The remainder of this manual shows you how to make REVAS work for you.

The 'A LITTLE INSIGHT' section introduces the general algorithm by which REVAS performs a disassembly. The use of tables is explained and related to the commands that use these tables. You will need to understand this subject in order to make most effective use of the REVAS capabilities. In particular, the two most frequently used commands are introduced in that section.

In the 'REVAS COMMANDS' section, you will find a detailed description of the syntax and operation of each command. Until you become thoroughly familiar with the command set, you will have frequent use for this section. A careful reading of the formal command descriptions will reveal the freedom of format that is designed into these commands. A list of REVAS COMMANDS appears on the back cover for ready reference.

The IMPLEMENTATION section contains the information you need to load REVAS into your system and properly interface to your I/O devices. The I/O jump vectors are explained there, as well as the register usage associated with I/O. Some of the addresses and their functions within REVAS are given to permit minor changes to be made in output format if you wish.

Read the manual clear through before trying to operate REVAS, then refer to it frequently. Then go ahead and disassemble something!

One caution!

Think carefully before you use the 'A' command or the 'G' command. They can cause a system crash; the 'A' command by assigning tables in a program area, and the 'G' command by calling an address that is not the subroutine you intended. You have control of such situations because you are the one who specifies the address for these commands.

It is my intent to furnish software and documentation that is as useful and free of errors as possible. The REVAS program has been in constant use during its own development(!) and for many months by several users before the first version was released. All known bugs have been exterminated. I am interested in improving wherever possible

the quality of the program and its documentation. Thus, I will welcome and respond to comments and recommendations sent to the address below. (accolades are also welcome!) Please include a stamped self-addressed envelope if you wish a reply.

Al Hawley
6032 Chariton Ave
Los Angeles, CA
90056

HOW THE DISASSEMBLER WORKS

Here is a brief description of REVAS:

Program size-- 4k bytes
Symbol table location--end of REVAS or as assigned
Symbol table usage--
    synthetic symbols 4 bytes/symbol
    assigned labels 6 bytes/label
Symbol table length is initially zero, increasing as required to accomodate symbols and up to 682 labels.

The object program to be disassembled must be in memory at it's normal location. When control is passed to REVAS, the prompt character(#) will appear and you may respond with one of the commands described in the next section. Let's start with the 'D' (display disassembly) command:

REVAS will examine the byte located at the start address and analyse it as the first byte of an instruction by finding it in internal lookup tables. The operator mnemonic is obtained from the tables, as well as the number of bytes in the current instruction.

The operand field contents, if any, are next determined by a combination of logical operations and table lookup.

The operator and operand are stored in appropriate fields of the line buffer(LB).

Next, the address of the instruction and the object code are stored in the LB as hex characters, and the object code is converted and placed in the comment field of the buffer for printout as ASCII data.

The symbol tables are searched for a label assigned to the address just defined. If a label is found, it is inserted in the label field of the line buffer. If there is no label, then the tables are searched for a synthetic label to insert. If none exists, then the field is left blank. Labels can be right or left justified (see Patch Locations below).

Now the contents of LB are printed on the output devices, the console is checked for any pending commands, and the process is repeated until terminated by reaching the last address or by a command from the console.

The symbol tables comprise two tables: an index table and a label table. The index table is constructed during execution of the 'B' command. It contains the hex value of each 16 bit argument encountered in the address ranges that

have been disassembled. It also contains flags which indicate for each entry the presence of an assigned label, the mode of the label (instruction mode or table mode), and a pointer to the location of the assigned label in the second (label)table. The 'K' command deletes entries from the index. The 'M' command changes the flag which indicates label mode. The 'F' command adds it's argument to the index table if it is not already there.

The label table (assigned labels) is constructed during execution of the 'L','S', or 'T' commands. When one of these commands is given, the label specified in the command is added to the label table. The index table is then searched for the corresponding hex value (address of the label in the object program) and a pointer is entered in the index table that points to the label entry. If there is no corresponding entry in the index table, then one is created. Thus, these commands also act to build the index table. The 'S' command resets a flag to indicate that this label belongs to an instruction. The 'T' command sets the flag to indicate that this label belongs to a byte in a table of data. The 'L' command leaves the mode flag unchanged.

A description of the flag and pointer words for the symbol tables is included in the implementation section. If you choose to store the tables (on tape or disc, for example) for future use, then you must be sure to also record these locations and restore them when you restore the tables.

Mode, Mode Character, and Mode Control

The character immediately following the 'D' or 'B' in those commands is the mode character. Two modes are possible: instruction mode and table mode. Table mode is specified if the mode character is a 'T'; any other specifies instruction mode.

In the instruction mode, bytes from the object program are interpreted as Z-80 and 8080 instructions.

In the table mode, bytes from the object program are interpreted as single-byte constants which are part of a table of data.

There are two flags associated with mode control. The mode control flag is set (or reset) by the mode control character when the 'D' or 'B' command is issued to REVAS. The second flag, the mode bit, is part of the data stored for each entry in the index table. The mode bit is set or reset during execution of the 'B','M','S',and 'T' commands. The function of the 'M' command is to define the state of

the mode bit for a particular (address) entry in the index table, creating a dummy entry if none is present when the command is given.

Operation of the 'D' Command

The 'D' command displays the disassembly on the selected output device(s), using the mode control flag to determine the format of the output.

If an index table entry matching the current instruction or data byte address is encountered, then the mode bit from that table entry replaces the mode control flag; the output format (or mode) is controlled for this and subsequent bytes by this new mode control flag.

Clearly, when the index table is empty (at the start of a session or after the 'I' command) all output format is specified by the mode control character. After any of the table building commands (B,L,M,S,T) have been executed, mode information from the index table entries will be used as appropriate.

Operation of the 'B' Command

The 'B' command functions much like the 'D' command. One difference is in the use of the mode flags.

When, during disassembly, an index table entry is found which matches the current instruction or byte address, the mode bit of the entry is changed to correspond with that specified by the current mode control flag.

Another difference is the table building function. When a 16 bit argument is found in the current operand field, it is replaced by a synthetic symbol formed by the concatenation of an 'S' or 'T' and the hex representation of the argument. The first letter will be an 'S' if there is no index table entry. It will be either 'S' or 'T' (depending on the state of the mode bit) when an entry already exists. If this is the first occurence of the argument value, then an entry is created in the index table whose mode bit specifies instruction mode ('S'). Later, when tables of data are being disassembled with the 'BT' command, those arguments in the index table that refer to labels in the object-program table area will have their mode bits changed to specify table ('T') mode. Incidentally, the index table itself can be listed using the 'DT' command.

Note that if a table of data in the object program is

disassembled using the 'B?' (? not a 'T') command, many spurious arguments will be generated and stored in the index table with curious effects during later listing. For example, the hex code sequence 20 20...(ASCII blanks) would be interpreted as a relative jump from the Z-80 instruction set, and the destination of the jump would be stored in the index as the current address plus 20H. similar situations exist for code sequences that look like LXI, SHLD, LDA, etc., from the 8080/Z-80 instruction sets.

Because of the above considerations, it is usually best to analyse object code initially with the 'D' command, reserving the use of the 'B' command until the instruction and table areas have been located. Then the 'B' command can be used to build tables (and assign synthetic symbols), first to the instruction areas, and then to the tables of data.

Command Syntax

Portions of a command are separated by a delimiter in most cases. The delimiter is represented in the command descriptions by '%', which implies either comma or space.

Numeric values (addresses or symbol values, for example) are expected to be in hexadecimal notation. When entering the hex number, as many hex characters as desired may be entered; only the last four will be used by REVAS. If you type the wrong number, simply retype it without intervening keyboard entries.

If a non-hex character is entered then REVAS simply returns to the command mode and you may re-enter the command or change to another command. It's a good way to escape a command sequence when you change your mind..

Spaces in the formal command descriptions are present for clarity of presentation only; they are not a part of the keyboard input.

REVAS accepts commands in either upper or lower case. Upper case is used in the command descriptions only for clarity.

Definition Syntax

/../ Text enclosed by slashes is typed by REVAS. Other parts of the commands are typed by the user.

^ (up-arrow) means "depress the ctrl key and keep it depressed while typing the next character".

+ The logical "inclusive-or" function. "a+b+c" means "one or more of the parameters listed".

! The logical "exclusive-or" function. "a!b!c" means "one and only one of the parameters listed".

% Means "enter a space or comma from the console keyboard". I.e., %=<space>!<comma>

@ Means "enter a carriage return by depressing the console keyboard return key".

<..> Text enclosed by "<" and ">" is a symbolic representation of a keyboard entry. The actual entry, if not self evident, is explained in the command description.

[..] The expression(s) enclosed by square brackets may be included in or excluded from the command at the user's option. The command processor in REVAS will recognize the intent of the command either way. Furthermore, the brackets also imply that the contents may be repeated an indefinite number of times.

? Means "Enter any printable character". A space is considered a printable character.

(..) Parentheses are used to group elements of a command in the command descriptions to avoid ambiguity of interpretation. The parentheses are not part of the actual command.

Immediate Commands

The next three commands may be used at any time during a disassembly activity (even when printout has been suppressed).

R

Return to command mode. This command causes an abort of the current listing and an immediate return to the command mode. The prompt character will be printed and a new command sequence is expected.

S

Suspend printout at the end of this line. This command causes the disassembly to pause at the end of the current line. Escape from this pause (or wait) state is via any keyboard entry.

The next command may be used at any time:

^C

(control/C) this command causes an immediate trap to the monitor. It may be used at any time during the disassembly or command phases. If you are using a monitor from Technical Design Labs, then a return via the monitor 'G' command without an argument will result in resumption of the interupted activity if none of the registers has been changed. Note that if ^C is executed during the pause after the 'S' command, return will be to the pause state and a keyboard entry will be needed to continue the disassembly.

Disassembly Display & Analysis Commands

/#/ D? /addr range=/ <start addr> % <end addr> @

'D' means "display the object code as mnemonic assembly instructions". This command results in disassembly of the object code specified by the start and end addresses.

If the <?> character is a 'T' then the object code is interpreted as data tables and printed in .BYTE format. When mode information is encountered in the Index table, that mode automatically takes precedence and subsequent printout is controlled by the flag in the Index table.

If the character following the 'D' or 'DT' is a HEX digit then the prompt message (ADDR RANGE=) is suppressed, and you may continue with the address entries. Any other character in this position causes printing of the prompt message.

/#/ F [?]%<arg>/addr range=/ <start addr>%<end addr>@

Find all statements in the address range specified that reference <arg> as a 16 bit number. These would be calls, jumps, LXI statements, etc. Each such statement is printed out in the normal disassembly format. The starting address must be the start of an instruction to avoid an initial phase problem and an inaccurate disassembly. This instruction loads the HEX number into the index table before starting its search, so the command may be used even before tables have been built. You can remove the number from the tables later on with the 'K' command if you wish. You could use this command to find all the locations in the object program that call a particular subroutine.

/#/ P @

Print out the symbol table. Only those symbols (and their addresses) to which you have assigned a name will be printed. Synthetic symbols are not printed, since their addresses are part of the symbol.

/#/ X /addr range=/ <start addr> % <end addr> @!%
/sym val range=/ <first addr> % <last addr> @

This command searches repeatedly through the address range specified, printing out those instructions that reference symbols whose addresses are in the range of addresses requested by /sym val range/. The instructions that reference the smallest address value are listed first; then the next value is used; the process is repeated for each value in the range. This command uses the information in the symbol tables, so it will only be useful after the symbol address values have been entered by the B,L,S, or T commands.

EXAMPLE: /#/XADDR RANGE=100 51E SYM VAL RANGE=100 115

Index & Symbol Table Control Commands

/#/ B? /addr range=/ <start addr> , <end addr> @

The form of this command is exactly the same as that of the 'D' command. You may avoid the prompt message the same way.

'B' means 'build the Index table'. When the instruction mode is specified(B? is anything except BT), each 2-byte argument encountered during the disassembly is assigned a synthetic symbol name which is the hex value preceeded by 'S'. This synthetic symbol is placed in the label field of the listing when the corresponding address is encountered in the disassembly. When the table mode is specified (BT) the object code is listed in ".BYTE" format as data. The mode of any labels encountered is changed to 'T' and the synthetic symbol prefix is changed from 'S' to 'T'. When building tables, the 'B?' command should be used until the instruction code sequences have been disassembled. Then use the BT command to identify the table area labels. This will avoid the need to re-build tables for some areas of the object program. Note that there is no problem with using this command repeatedly on the same or overlapping address ranges. Symbols already entered are retained; only the mode flag associated with the symbol is affected.

/#/ L [%] <address> / =/ <alpha string> @!%

Create a label in the symbol table. The alpha string specifies the label name. If more than 6 characters are typed, only the first 6 will be stored and used. the mode flag is not affected.

/#/ S [%] <address> / =/ <alpha string> @!%

Same as 'L', except mode flag is reset to indicate that this is an instruction.

/#/ T [%] <address> / =/ <alpha string> @!%

Same as 'L', except mode flag is set to indicate that this is a label for data.

L,S, and T may be used to replace a label and change its mode as often as required. These commands may be used even before the symbol tables have been built with the 'B' command.

/#/ M [%] <address> % (0 ! 1) (@!%)

0 means 'the address specified is an instruction'. 1 means 'the address specified is a data byte'.

This command permits marking of data or instructions in the program for which there is no label. It is typically used to mark the beginning of a table of data whose first byte is not referenced directly. Likewise, it might be used to mark the first instruction following a table, where no direct reference is made (reference might be by means of a jump table, for example).

When the delimiter (%) is used to terminate the L,S,T, or M commands, the next prompt (#) will be on the same line as the last one. These commands can be 'strung out' across the page using this feature.

/#/ K [[%] <address> %] @

Kill a symbol table entry. This command removes all reference to the address given from the symbol tables. It's most important use is to remove 16 bit constants from the tables so that they will print out during disassembly as constants (numbers) rather than synthetic labels. You may also use 'K' to remove labels assigned by the L,S, and T commands.

When a delimiter is used after the address, another address may be entered, and another, etc until a carriage return is entered to terminate the Kill mode. For example, "/#/K0,1,2,4,8,A00@" would result in deletion of 0000, 0001, 0002, 0004, 0008, and 0A00 from the tables.

Utility Commands

/#/ IΛFΛF

This command initializes the symbol tables by assigning initial values to the Symbol Table Pointers corresponding to empty tables located at the end of REVAS. It is typically used to start a new disassembly.

```
**    DO NOT USE THIS COMMAND      **
** IF YOU WANT TO SAVE THE SYMBOLS **
**   YOU HAVE ALREADY ENTERED!     **
```

/#/ AΛFΛF [%] <address> @

Assign the start of the symbol tables to the memory address specified. The tables are moved to the new location and the Symbol Table Pointers are adjusted to correspond to the new table location. The Symbol Table Pointers are the sole link between REVAS and the tables, and their location is not changed. Copies of REVAS at two different locations could use the same tables if you were to copy the ST Pointers from one REVAS copy to the other. Normally you would use this command at the start of a session to place the tables advantageously in your memory space. Tables are built at the end of REVAS if not otherwise assigned with this command.

NOTE

The two ΛF characters immediately following the 'I' and 'A' commands are included as a safety feature to prevent inadvertant issuance of these commands. You can change these 'lockout' characters by changing the contents of two memory locations. (see Patch Locations below) Note that the parity bit of the second byte must be SET (=1). Thus, if you selected 'PE' as the two (ASCII) characters, the entries would be 50 and C5.

/#/ O [[?] (C!P!L) [?] %] @

The output device(s) for the disassembly listing are determined by this command. Note that all but the CR(@) may be ommitted. In that case, only the command dialog will be printed on the console. This option is useful when you wish to build ('B' command) tables without wasting time with the listing.

/#/O C Assigns disassembly listing to the console

/#/OL Assigns listing to the list device
(56 lines per page)

/#/OP Lists the label, operator, and operand fields on the punch device. (suitable for reassembly!)

/#/Out Console,List,Punch

This is an acceptable command which will result in listing on all 3 devices. Note that, for this command, words may be substituted for the single letters that REVAS recognizes. The first letter should be O,C,L, or P as implied in the example.

G [?] <addr>@

This is the 'GO' command that allows you to transfer to your own subroutine. If the subroutine terminates with a 'RET' statement and the stack pointer has the same value as it had at the start of your routine, then return will be to the REVAS command processor after your routine has done it's job. One use for this command in my system is to run a routine that closes a disc file after REVAS has finished writing a disassembly listing to it.

/#/ C

Comment field control...This command switches the comment field on and off. If the last output included the comment field, then execution of this command will inhibit printing of the comment field until the command is given again. If you have already made use of the data in the comment field, then you can inhibit it's printout with the 'C' command and considerably increase the printout speed and get cleaner copy in the bargain. (The printer doesn't have to print all those spaces between the operand field and the comment field)

/#/ H

HALT at the top of the each page of the logical list device. This command may be used at any time, even while listing is in progress. Listing is resumed when any keyboard entry is detected. If the entry is a ^C, then that function is performed first and listing will resume immediately on return from the monitor.

By assigning the logical list device to a CRT or TV display and setting the top margin, lines/page, and bottom margin appropriately, the HALT function can be used to step through a complete disassembly one screen-full at a time.

/#/^H

The ^H character cancels the HALT mode, resulting in continuous paginated listing without pause at page boundaries. This is the default mode for REVAS.

/#/ E

The 'E' command causes the '.END' Pseudo-op to be output to the active (assigned by 'O') output devices. This Pseudo-op is required by most assemblers at the end of the source listing.

/#/ ^L

Control-L, the ASCII Form Feed, causes the logical list device to space to the top of the next page. It may be used at any time, even when the list device is unassigned by the 'O' command.

This completes the description of the REVAS commands. Experiment with them until their nuances become familiar to you, and you will then get the most benefit from your REVAS disassembler.

Loading REVAS From Cassette

The standard REVAS cassette is recorded in straight Tarbell format. It starts with a 25 second sync stream that you can use to adjust your interface. Following the sync stream is 15 seconds of carrier, and then the first load module. The cassette contains 3 load modules; the first module is 1400H bytes long and is named REVAS. If you do not have the CPM system, this is the only module of interest to you. The second module is 400H bytes long, and is named REVAS.COM. The third module is 1400H bytes long and is named REVAS.LOD.

REVAS (the first module) may be loaded anywhere you wish in memory and executed by jumping to the load address. During first execution, the relocating code is destroyed and the resulting copy of REVAS is no longer relocatable. Thus, to make I/O patches to the jump vectors, the modifications must be made to the copy immediately after loading. The modified copy can be saved (on tape, for example, or on disc); the relocatability feature will still be valid if you have done no more than change the arguments of the I/O jump vector. Once executed, REVAS is exactly 1000H bytes long.

REVAS.COM and REVAS.LOD should be loaded into memory one at a time and saved from location 100H (the CPM tpa) using CPM commands. More details are given in the appended REVAS/CPM guide.

Before loading REVAS, insert the cassette in your recorder with the interface disconnected so you can hear the data. Play the tape from the beginning. The first sound you hear will be that of the sync stream, then the carrier tone. Note the places where the steady tone of the carrier is replaced by the 'noise' sound of the recorded data. The programs are recorded in the order listed above, with about 15 seconds between copies. Now position the tape a few seconds ahead of the copy you want, reconnect the Tarbell interface, and copy the program into memory.

I/O Interface Description

REVAS is designed to support one logical input device (the console) and 3 logical output devices (console, punch, and printer).

The physical devices referenced by the logical names (console, punch, and printer) are determined by your driver routines and the jump vectors in REVAS that point to them. You could, for example, have the punch actually write on a disc file.

All I/O transactions take place through the jump vectors located near the beginning of the REVAS program. These vectors are shown in the listing below. You must verify that these jumps point to the proper driver routines in your system. If you are using a monitor from Technical Design Labs and it is located at 0F000H, then no changes will be necessary. Otherwise, you must change the jump arguments so that they point to your own driver routines.

The driver routines must observe the following register usage conventions:

A byte to be output is transmitted in the 'C' register and will be in the 'C' and 'A' registers on return from the output driver. An input byte (from the console) is expected to be in the 'A' register. The content of all other registers must be returned unchanged during an I/O operation.

REVAS Entry & I/O Vector

```
ADDR  CODE      LABEL    OPR   OPA        COMMENTS

0020 31 XXXX      REVAS: LXI   SP,STACK ;LOCATE STACK

       ;ADDR is the address relative to the load
       ;address BEFORE execution. During execution,
       ;all of this code is moved down 20H bytes,
       ;so that the instruction labeled REVAS is
       ;located at relative address 0000.
       ;XXXX depends on Version number of REVAS


0023 C3 XXXX            JMP   START    ;GO TO WORK
0026 C3 12F0      CSTS: JMP   0F012H   ;CONSOLE STATUS

       ;RETURNS WITH 0FFH IN ACCUMULATOR IF THERE IS
       ;CONSOLE INPUT WAITING, 00 IF NOT.
```

```
0029 C3 03F0     CNSLIN: JMP   0F003H   ;CONSOLE INPUT
002C C3 09F0     CSLOUT: JMP   0F009H   ;CONSOLE OUTPUT
002F C3 0FF0      LPOUT: JMP   0F00FH   ;PRINTER OUTPUT
0032 C3 0CF0       POUT: JMP   0F00CH   ;PUNCH OUTPUT
0035 C3 1EF0       TRAP: JMP   0F01EH   ;RETURN TO MONITOR

       ;RETURN FROM THE MONITOR WITH ALL REGISTERS
       ;(INCLUDING THE STACK PTR AND THE PC)
       ;RESTORED TO THEIR STATES AT THE TIME OF THE
       ;JUMP TO TRAP WILL PERMIT CONTINUED EXECUTION
       ;OF REVAS WHERE IT LEFT OFF. IF YOUR MONITOR
       ;ROUTINES DO NOT INCLUDE THIS FACILITY, THEN
       ;RETURN SHOULD BE THROUGH A JUMP TO REVAS
       ;(I.E. JUMP TO RELATIVE LOCATION 0000)
```

Symbol Table Pointers after execution

REL ADDR	CONTENT DESCRIPTION
0FF5	TFLAG: 0=EMPTY TABLE; 1=NOT EMPTY
0FF6	A1: ADDR OF FIRST ENTRY OF INDEX TABLE
0FF8	A3: ADDR OF LAST ENTRY OF INDEX TABLE
0FFA	A4: ADDR OF FIRST ENTRY IN LABEL TABLE
0FFC	A2: POINTER TO NEXT AVAILABLE LOCATION FOR LABEL ENTRY (RELATIVE TO FIRST ENTRY)
0FFE	A5: POINTER TO LAST LOCATION IN THE LABEL TABLE RELATIVE TO THE FIRST ENTRY
1000	DEFAULT LOCATION OF INDEX TABLE (THIS ADDR WOULD BE IN A1: ABOVE)

Patch Locations after execution

REL ADDR	CODE	REMARKS
0850	3B	Comment Field Delimiter
0908	AE	Right Justify Label Field EE=Left Justify Lbl Field
091C	3A	Label Terminator
0EA1	2E 42 59 54 C5	'.BYTE' Pseudo-Op For Tables
0EA6	2E 45 4E C4	'.END' Pseudo-Op
0EAA	0D 0A 00 80 00	CRLF For List Device
0EAF	0D 0A 00 80 00	CRLF For Other Devices
0ECE	06 86	Lockout bytes, ^F^F

(To change an ASCII string such as this to some other string such as 'DB', observe the following two rules:

1. You must use the same number of characters, filling out with blanks if necessary.
2. The last byte in the string must have bit 7 set. I.e., the parity bit must be equal to one.)

0037	38	LC: Line Counter
0038	5	TM: Top Margin
0039	5	BM: Bottom Margin
003A	38	LP: Text Lines/Page

Immediate command characters

0445	52	R=RET to Command Mode
0187	53	S=Suspend Disassembly

Line Format Modification

The line buffer is 60 bytes long and is divided into 6 fields. The length of each field is specified in a format list starting at 0018H. It contains 7 words, each of which is the absolute address of the start of a field. The field starting with the seventh address is not used. The first address in the list is the start of the line buffer after execution.

Format List

REL ADDR	HEX WORD	SYMBOL	FIELD	FIELD DESCRIPTION
0018	D40E	LB	1	address field
001A	D90E	LB+5	2	object code
001C	E50E	LB+17	3	label field
001E	ED0E	LB+25	4	operator field
0020	F30E	LB+31	5	operand field
0022	080F	LB+52	6	comment field
0024	0D0F	LB+57	defines	end of field 6

Page Format Control

Page format for the list device output is controlled by TM, LP, and BM. TM specifies the number of blank lines at the top of the page, LP specifies the number of text lines per page, and BM specifies the number of blank lines at the bottom of the page. Total page length is thus the sum of these three constants. You can change TM or BM to any 8 bit value, including zero. LP may be assigned any 8 bit value except zero. These assignments are made by modifying the appropriate patch locations.

Nulls After CRLF

The CRLF-Nulls sequence is separately specified for the List Device and for all other output devices. (See Patch Locations) As supplied, REVAS outputs 2 nulls after a line feed. You can change this to from zero to three nulls by changing the byte in the sequence for which bit 7 is set. For zero nulls, the sequence must be changed to 0D 8A 00 00 00; for 3 nulls change to 0D 0A 00 00 80.

REVAS FOR CPM

The CPM environment requires some changes in the command structure of REVAS, as well as the way in which REVAS is implemented. The following description of new and changed features is an appendix to the REVAS User's Manual, V2; please refer to that manual if you are not already familiar with it's contents.

In the CPM environment, REVAS is present as TWO files: REVAS.COM and REVAS.LOD. The first file, REVAS.COM, is an executive program which manages the loading of the target program, REVAS, and the symbol tables. This file is 4 blocks (pages) long and runs at a location just below CPM's CCP section. You can, if you wish, rename this file using CPM's REN command, but it must remain a .COM file. This file name defines the transient command used to invoke a disassembly with REVAS. The second file, REVAS.LOD, contains the actual REVAS program embedded in a relocating loader. It is loaded into memory and executed by REVAS.COM, which contains a reference by filename.ext to REVAS.LOD. Thus, the REVAS.LOD file must not be renamed. REVAS.LOD is 20 blocks (pages) long. (a block is 256 bytes) All files are assumed to be on the currently selected disc.

The program to be disassembled (the target program) must coexist with REVAS in your computer's memory space. When REVAS is invoked, the target file is accessed and loaded at the CPM tpa (address 100 Hex). The REVAS disassembler is then loaded above the target pgm. If a symbol table file for the target program exists on the disk, it is loaded at the end of REVAS. If no symbol table file exists, then one is automatically created. During the loading process, messages will be typed on the console to let you know the results of each of these steps.

REVAS always uses two files: FN.TBL, and FN.ASM. FN.TBL is the file to which REVAS writes its symbol tables (see the 'W' command). FN.ASM is the file to which REVAS writes assembler mnemonics for editing and/or reassembly. When REVAS is invoked it searches the current disc directory for these files. If not found, they are automatically created. If FN is not specified in the invoking command, then FN=### is assumed. If the .EXT is not specified for the target program, then EXT=COM is assumed.

A disassembly is invoked by typing a standard CPM transient command (after the CPM prompt) of the following form:

REVAS [ufn]

'ufn' is an unambiguous file name, as defined in your CPM manual. Four possible forms of this command and the resulting file names involved are shown in the table below. In this table, 'EXT' means any file extension except 'TBL'. 'FN' stands for any file name.

Command	Response
REVAS	REVAS is loaded at the tpa. ###.TBL and ###.ASM are used.
REVAS FN	FN.COM is loaded at the tpa followed by REVAS. The files FN.TBL and FN.ASM are used.
REVAS FN.EXT	FN.EXT is loaded at the tpa, followed by REVAS. The files FN.TBL and FN.ASM are used.
REVAS FN.TBL	No target program is loaded. REVAS is loaded at the tpa, followed by the symbol table file FN.TBL. If FN.TBL does not exist, then it is created. FN.ASM is used.

After the loading process is completed, control is passed to REVAS, as indicated by display of the REVAS prompt (#) on your console. The REVAS command set is now at your disposal.

REVAS/CPM Special Commands

The 'A' and 'I' commands described in the User's Manual are not needed in the CPM environment, and have been deleted in the CPM version of REVAS. The 'W' command in the CPM version saves the current symbol tables on disk in a file named FN.TBL. The 'O' (Output channel control) command and the 'E' (.END pseudo-op) commands have been modified for the CPM environment to OPEN and CLOSE the FN.ASM file. The Punch output option is no longer implemented in REVAS/CPM, since that utility is available through PIP.

^C Control-C re-boots the CPM system.

E Inserts the pseudo-op '.END' into the output stream and properly closes the FN.ASM file. A FN.ASM file that is not closed with this command will not contain the final record with the end-of-file mark required by the Editor.

O The key letter 'P' has been replaced by 'A' (for .ASM). Otherwise, the command format is unchanged from that in the user's manual. The command 'OPEN C,A@' results in normal disassembly output at the console and label, opcode, and argument output to the (now open) .ASM file. A subsequent OC command DOES NOT CLOSE THE FN.ASM FILE. File closure MUST be accomplished with the 'E' command. The Console and List devices ARE deselected when their key letters are ommited from an 'O' command argument list.

W Write the symbol tables into the FN.TBL file. This command opens the file, writes to it, then closes the file. It will not execute if the FN.ASM file is currently open, and will print a reminder to close the .ASM file if it is open. The command may be repeated as often as you wish during a disassembly, so you can always have saved the latest version of the label assignments.

A <Switch>%<Input>@ The (new) 'A' command is used for assigning new values to the parameters listed in the User's manual under the heading 'Patches'. The table which follows gives the expected Input for each <switch> value. 'HEX' means a hexadecimal value in the range 0 to FF; 'CH' means any keyboard character, including lower case and control characters.

SWITCH	INPUT	FUNCTION
0	HEX	Number of nulls to send after a carriage return, line feed to the list device.
1	HEX	Number of null to send after a carriage return, line feed to the console or punch devices.
2	HEX	Number of lines in the Top Margin of the List device page.
3	HEX	Number of lines in the Bottom Margin of the List device page.
4	HEX	Number of lines of text per List page.
5	AE	Right justify labels in label field.
5	EE	Left justify labels in label field.
6	CH	Replace 'S' for the immediate command which suspends printout and disassembly.
7	CH	Replace 'R' for the immediate command which returns to REVAS command mode.
8	CH	Replaces the ':' label terminator
9	CH	Replaces the ';' comment field delimiter
A	1 to 4 CH	These characters replace the '.BYTE' pseudo-op which defines data storage bytes.

The 'A' command has several restrictions. First, it only operates when no other commands have been previously executed. Second, it only operates after the CPM command:

REVAS REVAS.LOD

The changes which are made by the 'A' command occur only in the copy of REVAS.LOD which is now located at the tpa. After all the changes have been made, the new REVAS.LOD is saved by executing a re-boot of the CPM system (^C) and using the CPM command:

SAVE 20 REVAS.LOD

REVAS, on subsequent invocation, will contain the changes.

All other commands are as described in the REVAS user's manual.

```
                        .TITLE *NORTH STAR HORIZON I/O FOR REVAS*
                        ;THESE ROUTINES CONVERT REVAS I/O
                        ;FORMAT TO NORTH STAR FORMAT, LEAVING
                        ;DOS OPERATION AS SPECIFIED BY NORTH STAR.
                        ;
0002            CIBIT=02        ;CONSOLE INPUT STATUS IN BIT 1
0001            LBIT=01H        ;LIST OUTPUT STATUS BIT
0001            PBIT=01H        ;PUNCH OUTPUT STATUS BIT
0003            CSTAT=03H       ;CONSOLE STATUS PORT
0009            LSTAT=09H       ;LIST DEVICE STATUS PORT
0008            LDATA=08H       ;LIST DEVICE DATA PORT
000B            PSTAT=0BH       ;PUNCH DEVICE STATUS PORT
000A            PDATA=0AH       ;PUNCH DEVICE DATA PORT
200D            CHOUT=200DH     ;CONSOLE OUTPUT
                        ;
                        ;THE 'RXXXX' ROUTINES ARE USED BY REVAS FOR
                        ;OUTPUT AND CONSOLE STATUS TESTING. FOR INPUT,
                        ;REVAS USES THE DOS CONSOLE INPUT ROUTINE
                        ;
                        ;THE RLOUT AND RPOUT ROUTINES ARE FOR THE LIST
                        ;AND PUNCH LOGICAL DEVICES WHICH REVAS CAN
                        ;SPECIFY FOR OUTPUT. AS SUPPLIED, BOTH ROUTINES
                        ;JUMP TO THE CONSOLE FOR OUTPUT; YOU MUST
                        ;CHANGE THEM TO FIT YOUR OWN SYSTEM PORT AND
                        ;STATUS BIT REQUIREMENTS. IF INITIALIZATION
                        ;IS REQUIRED, YOU MUST ADD IT TO THE INIT
                        ;ROUTINE WHICH STARTS AT 2925H. SPACE HAS BEEN
                        ;LEFT THROUGH 29C8 FOR SUCH PATCHES.
                        ;
                        ;THE 'MOVE' ROUTINE TRANSFERS THE I/O ROUTINES
                        ;(RCSTS TO 'END') TO LOCATION 29C9. THIS ENTIRE
                        ;PROGRAM IS TITLED 'NSHIO' ON YOUR DISC AND CAN
                        ;BE EXECUTED BY THE DOS COMMAND 'GO NSHIO'
                        ;TO AUTOMATICALLY CONFIGURE THE IO FOR REVAS
                        ;COMPATIBILITY. SINCE THE CODE IS SELF-RELOCATABI
                        ;YOU CAN TRANSFER IT TO SOME OTHER LOCATION BY
                        ;CHANGING THE ARGUMENT OF THE 'LXI D....' IN THE
                        ;MOVE ROUTINE...REMEMBER, THOUGH, THAT YOU
                        ;MUST THEN MAKE THE SAME BIAS ADJUSTMENT IN THE
                        ;IO JUMP VECTOR AT THE BEGINNING OF EACH COPY
                        ;OF REVAS.
                        ;
29B1                    .LOC 29B1H      ;FOR CONVENIENCE ONLY
                ;TO MAKE ASSY ADDRESSES SAME AS FINAL ONES..
                        ;
29B1  21 E9E1   MOVE:   LXI H,0E9E1H
29B4  22 29C9           SHLD 29C9H      ;STORE 'POP H--PCHL'
29B7  CD 29C9           CALL 29C9H      ;RET WITH 'LOC' IN HL
29BA  11 000F   LOC:    LXI D,RCSTS-LOC ;OFFSET
29BD  19                DAD D           ;HL POINTS TO RCSTS
29BE  11 29C9           LXI D,29C9H     ;DESTINATION
29C1  01 0029           LXI B,END-RCSTS ;NUMBER OF BYTES
29C4  EDB0              LDIR            ;MOVE INTO PLACE
29C6  C3 2028           JMP 2028H       ;RETURN TO DOS
29C9  DB03      RCSTS:  IN CSTAT        ;CHECK CONSOLE STATUS
```

```
29CB  E602            ANI CIBIT
29CD  3E00            MVI A,0         ;FOR 'NO INPUT WAITING'
29CF  C0              RNZ
29D0  2F              CMA             ;CHANGE TO FF FOR INPUT
29D1  C9              RET             ;..WAITING
29D2  AF        RCHOUT: XRA A
29D3  C5              PUSH B          ;CONSOLE OUTPUT
29D4  41              MOV B,C
29D5  CD 200D         CALL CHOUT
29D8  C1              POP B           ;RECOVER BC
29D9  C9              RET
29DA  18F6      RLOUT:  JMPR RCHOUT     ;REPLACE WITH NOPS TO USE LOUT
29DC  DB09      LOUT:   IN LSTAT        ;LIST DEVICE OR LINE PRINTER
29DE  E601              ANI LBIT        ;GET OUTPUT STATUS BIT
29E0  20FA              JRNZ LOUT       ;LOOP IF NOT READY FOR OUTPUT
29E2  79                MOV A,C
29E3  D308              OUT LDATA
29E5  C9                RET
                        ;
                        ;YOU MAY WISH TO SUBSTITUTE ROUTINES FOR
                        ;RPOUT THAT WRITE TO A MEMORY BUFFER (ONE
                        ;BYTE AT A TIME) AND THEN STORE THE OUTPUT
                        ;IN A DISC FILE WHICH YOU HAVE CREATED FOR
                        ;THIS PURPOSE...
                        ;
29E6  18EA      RPOUT:  JMPR RCHOUT     ;CHANGE TO NOPS TO USE POUT
29E8  DB0B      POUT:   IN PSTAT        ;LOGICAL PUNCH DEVICE
29EA  E601              ANI PBIT
29EC  20FA              JRNZ POUT
29EE  79                MOV A,C
29EF  D30A              OUT PDATA
29F1  C9                RET
29F2                    END=.
                        .END
```


+++++ SYMBOL TABLE +++++

CHOUT	200D	CIBIT	0002	CSTAT	0003	END	29F2
LBIT	0001	LDATA	0008	LOC	29BA	LOUT	29DC
LSTAT	0009	MOVE	29B1	PBIT	0001	PDATA	000A
POUT	29E8	PSTAT	000B	RCHOUT	29D2	RCSTS	29C9
RLOUT	29DA	RPOUT	29E6				

```
                        .TITLE *NORTH STAR DOS I/O FOR REVAS*
                        ;THESE ROUTINES CONVERT REVAS I/O
                        ;FORMAT TO NORTH STAR FORMAT, LEAVING
                        ;DOS OPERATION AS SPECIFIED BY NORTH STAR.
0001            CIBIT=01H       ;CONSOLE INPUT STATUS IN BIT 0
0080            LBIT=80H        ;LIST OUTPUT STATUS BIT
0080            PBIT=80H        ;PUNCH OUTPUT STATUS BIT
0000            CSTAT=00H       ;CONSOLE STATUS PORT
0004            LSTAT=04H       ;LIST DEVICE STATUS PORT
0005            LDATA=05H       ;LIST DEVICE DATA PORT
0002            PSTAT=02H       ;PUNCH DEVICE STATUS PORT
0003            PDATA=03H       ;PUNCH DEVICE DATA PORT
200D            CHOUT=200DH     ;DOS CONSOLE OUTPUT
                        ;
                        ;THE 'RXXXX' ROUTINES ARE USED BY REVAS FOR
                        ;OUTPUT AND CONSOLE STATUS TESTING. FOR INPUT,
                        ;REVAS USES THE DOS CONSOLE INPUT ROUTINE
                        ;
                        ;THE RLOUT AND RPOUT ROUTINES ARE FOR THE LIST
                        ;AND PUNCH LOGICAL DEVICES WHICH REVAS CAN
                        ;SPECIFY FOR OUTPUT. AS SUPPLIED, BOTH ROUTINES
                        ;JUMP TO THE CONSOLE FOR OUTPUT; YOU MUST
                        ;CHANGE THEM TO FIT YOUR OWN SYSTEM PORT AND
                        ;STATUS BIT REQUIREMENTS. IF INITIALIZATION
                        ;IS REQUIRED, YOU MUST ADD IT TO THE INIT
                        ;ROUTINE WHICH STARTS AT 2925H. SPACE HAS BEEN
                        ;LEFT THROUGH 29C8 FOR SUCH PATCHES.
                        ;
                        ;THE 'MOVE' ROUTINE TRANSFERS THE I/O ROUTINES
                        ;(RCSTS TO 'END') TO LOCATION 29C9. THIS ENTIRE
                        ;PROGRAM IS TITLED 'NSDIO' ON YOUR DISC AND CAN
                        ;BE EXECUTED BY THE DOS COMMAND 'GO NSDIO'
                        ;TO AUTOMATICALLY CONFIGURE THE IO FOR REVAS
                        ;COMPATIBILITY. SINCE THE CODE IS SELF-RELOCATAB
                        ;YOU CAN TRANSFER IT TO SOME OTHER LOCATION BY
                        ;CHANGING THE ARGUMENT OF THE 'LXI D....' IN THE
                        ;MOVE ROUTINE...REMEMBER, THOUGH, THAT YOU
                        ;MUST THEN MAKE THE SAME BIAS ADJUSTMENT IN THE
                        ;IO JUMP VECTOR AT THE BEGINNING OF EACH COPY
                        ;OF REVAS.
                        ;
29B1                    .LOC 29B1H      ;FOR CONVENIENCE ONLY
                ;TO MAKE ASSY ADDRESSES SAME AS FINAL ONES..
                ;
29B1  21 E9E1   MOVE:   LXI H, 0E9E1H
29B4  22 29C9           SHLD 29C9H      ;STORE 'POP H--PCHL'
29B7  CD 29C9           CALL 29C9H      ;POP RET ADDR, THEN RETURN
29BA  11 000F   LOC:    LXI D,RCSTS-LOC ;OFFSET IN DE
29BD  19                DAD D           ;HL POINTS TO RCSTS
29BE  11 29C9           LXI D,29C9H     ;DESTINATION
29C1  01 0029           LXI B,END-RCSTS ;NUMBER OF BYTES
29C4  EDB0              LDIR            ;MOVE INTO PLACE
29C6  C3 2028           JMP 2028H       ;RETURN TO DOS
29C9  DB00      RCSTS:  IN CSTAT        ;CHECK CONSOLE STATUS
29CB  E601              ANI CIBIT
```

```
29CD  3E00            MVI A,0         ;FOR 'NO INPUT WAITING'
29CF  C0              RNZ
29D0  2F              CMA             ;CHANGE TO FF FOR INPUT
29D1  C9              RET             ;..WAITING
29D2  AF        RCHOUT: XRA A
29D3  C5              PUSH B          ;CONSOLE OUTPUT
29D4  41              MOV B,C
29D5  CD 200D         CALL CHOUT
29D8  C1              POP B           ;RECOVER BC
29D9  C9              RET
29DA  18F6      RLOUT:  JMPR RCHOUT     ;REPLACE WITH NOPS TO USE LOUT
29DC  DB04      LOUT:   IN LSTAT        ;LIST DEVICE OR LINE PRINTER
29DE  E680              ANI LBIT        ;GET OUTPUT STATUS BIT
29E0  20FA              JRNZ LOUT       ;LOOP IF NOT READY FOR OUTPUT
29E2  79                MOV A,C
29E3  D305              OUT LDATA
29E5  C9                RET
                        ;
                        ;YOU MAY WISH TO SUBSTITUTE ROUTINES FOR
                        ;RPOUT THAT WRITE TO A MEMORY BUFFER (ONE
                        ;BYTE AT A TIME) AND THEN STORE THE OUTPUT
                        ;IN A DISC FILE WHICH YOU HAVE CREATED FOR
                        ;THIS PURPOSE...
                        ;
29E6  18EA      RPOUT:  JMPR RCHOUT     ;CHANGE TO NOPS TO USE POUT
29E8  DB02      POUT:   IN PSTAT        ;LOGICAL PUNCH DEVICE
29EA  E680              ANI PBIT
29EC  20FA              JRNZ POUT
29EE  79                MOV A,C
29EF  D303              OUT PDATA
29F1  C9                RET
29F2                    END=.
                        .END
```


+++++ SYMBOL TABLE +++++

CHOUT	200D	CIBIT	0001	CSTAT	0000	END	29F2
LBIT	0080	LDATA	0005	LOC	29BA	LOUT	29DC
LSTAT	0004	MOVE	29B1	PBIT	0080	PDATA	0003
POUT	29E8	PSTAT	0002	RCHOUT	29D2	RCSTS	29C9
RLOUT	29DA	RPOUT	29E6				