SOFTWARE MANUAL

AlphaFIX

USER'S MANUAL

DWM-00100-69

REV. A01

alpha micro

AlphaFIX User's Manual

April 1981
DWM-00100-69
Revision A01

First Printing: May 1980
Second Printing: April 1981

ALPHA MICROSYSTEMS
17881 Sky Park North
Irvine, CA 92714

This document reflects AMOS Versions 4.5 and later.

2MD-4/81

Table of Contents

CHAPTER 4 COMMAND MODE

CHAPTER 5 FIX OVERLAY COMMANDS

QUICK REFERENCE GUIDE

DISPLAY MODE

Disassembly Mode:

down.....(Control-J)
up.....(Control-K)
down page.....(Control-T)
initial page.....(Control-R)
home (to PC).....(Control-^)

toggle breakpnt..(Control-P)
proceed.....(Control-X)
single step.....(RETURN)
exit subroutine..(Control-E)
proceed to here..(Control-G)
no subr mode.....(Control-\)

Hex/ASCII Mode:

right.....(Control-L)
left.....(Control-H)
up.....(Control-K)
down.....(Control-J)
home page.....(control-^)
page down.....(control-T)
page up.....(control-E)
indirection.....(TAB)

COMMAND MODE

S)earch {symbol} ..... search for symbol
D)ump {address} ..... dump memory
Q)uit ..... leave FIX
F)inish ..... leave FIX without deleting module
R{n} {value} ..... modify register
Rel {mode} ..... set address base mode
Spec {string} ..... set up command string
Go {string} ..... execute AMOS command
Proceed ..... proceed to breakpoint
Attach {terminal}..... attach debugging terminal
Set {radix} ..... set radix (hex/octal)
Break {symbol} ..... set breakpoint
Clear {symbol} ..... clear breakpoint
Ini {filename} ..... process FIX command file
Fxo ..... list overlay commands

OVERLAY COMMANDS

```
HELP ..................  list FIX commands
MAP ...................  list memory modules
NEW ...................  delete memory modules
JCB {jobname} .........  display JCB status
TRMDEF {termname} .....  display terminal status
EA {effective addr}....  display effective address
RAD50 {effective addr}.  display RAD50 format
DDB {effective addr} ..  display DDB contents
DIR {Filespec} ........  display directory
TYPE {Filespec} .......  display ASCII file
LABELS ................  list labels in symbol table
RPN ...................  hex/octal calculator
```

RPN Operators

^	(change base)
@	(clear stack)
ESC or Q	(return to FIX command mode)
+	(add)
-	(subract)
*	(multiply)
/	(divide)
\	(modulo)
&	(and)
|	(or)
:	(exclusive or)
<	(shift top of stack left one bit)
>	(shift top of stack right one bit)
=	(negate top of stack)
~	(swap top two stack elements)
L	(roll stack elements left)
R	(roll stack elements right)
P	(pop stack)

CHAPTER 1

INTRODUCTION

AlphaFIX is an interactive screen oriented debugging program. It allows the user to examine and run AM-100 machine language programs. Users of the VUE screen editor will find operation of FIX familiar. FIX uses many of the same screen display concepts to give the user a two dimensional visual interaction with his/her program.

This manual assumes that the reader is familiar with AM-100 assembly language programming and the AMOS runtime environment. Related documents are: the WD16 Microcomputer Reference Manual, (DWM-00100-04); the AMOS Assembly Language Programmer's Manual, (DWM-00100-43); and the AMOS Monitor Calls Manual, (DWM-00100-42).

1.1 FIX AND DDT

FIX is designed to be an alternative to the debugger called DDT. FIX provides most of the DDT functions (often expanding those functions), while adding a number of unique features. DDT users will note that FIX -

1. Disassembles and single-steps through AMOS supervisor calls.

2. Allows the program under test to use a separate terminal for I/O. This is especially useful when debugging programs that are themselves screen oriented, since the debugger does not interfere with the screen display (and vice versa).

3. Allows BASIC assembly language subroutines and AMOS drivers to be tested.

4. Allows formatted display of AMOS data structures, such as DDBs, JCBs, files and directories.

One important restriction of FIX is that the current version does not support program patching in assembly language. This will be added in a future release.

1.2 INVOKING FIX

FIX is invoked from AMOS command level by entering FIX followed by the name of the program to be debugged. Any extension may be specified. The default is .PRG. FIX attempts to locate the desired file, and, if found, FETCHes it. FIX does not delete the module if it already exists before doing the FETCH. FIX then attempts to locate a corresponding SYM file. If found, it is loaded in and the user may reference locations by their symbolic names as well as by numeric value.

Example:

```
.fix user (RET)
AlphaFix Version 1.0
```

FIX then enters Command Mode (See Chapter 4, "COMMAND MODE"), and the user may commence debugging.

1.2.1 The FIX Initialization File

When FIX is invoked it looks for an initialization file. It first looks for a file FILENAME.FIX, where FILENAME is the name of the file being debugged. If it does not find this file, it looks for a file named INI.FIX. If found, this file is read in and processed. The file consists of a series of commands identical to the commands the user would enter in Command Mode (See Chapter 4, "COMMAND MODE"). These commands can be used to initialize FIX with commonly used commands.

FIX searches for the initialization file in three accounts. Assuming the user is currently logged on [P,PN] the search order is:

1: FILENAME.FIX (special INI file for this program)

2: INI.FIX[P,PN] (user's account)

3: INI.FIX[P,0] (project library account)

4: DSK0:INI.FIX[7,0] (system library)

If desired, a :T may be placed into the INI.FIX file. This causes all following lines to be echoed to the console as they are processed, just as with .CMD and .DO files. Any line beginning with a semicolon is ignored.

A typical INI.FIX file might look something like this:

```
:T                  ; enable echoing
set hex             ; choose your favorite radix
attach term2        ; attach debugging terminal
break cmloop
break getesc
break finish
break xeq
break value         ; set up some standard breakpoints
```

For details on the command format, see Chapter 4, "COMMAND MODE."

1.3 FIX MODES

FIX operates in two basic modes, called Display Mode and Command Mode. You toggle between the two modes by pressing the ESC key. When in Command Mode, FIX displays the current status of FIX and your program, and allows you to modify the debugging environment. For instance, you can set register values or delete memory modules.

Display Mode is further subdivided into two submodes: Disassembly Mode and Dump Mode. In Disassembly Mode, FIX displays a screenful of disassembled code. You may move the cursor through memory to examine it, or you may single-step and proceed through your program.

In Dump Mode, FIX displays a screenful of memory in hex and ASCII formats. You may move the cursor through memory, and you may modify (patch) the contents of memory in either hex or ASCII.

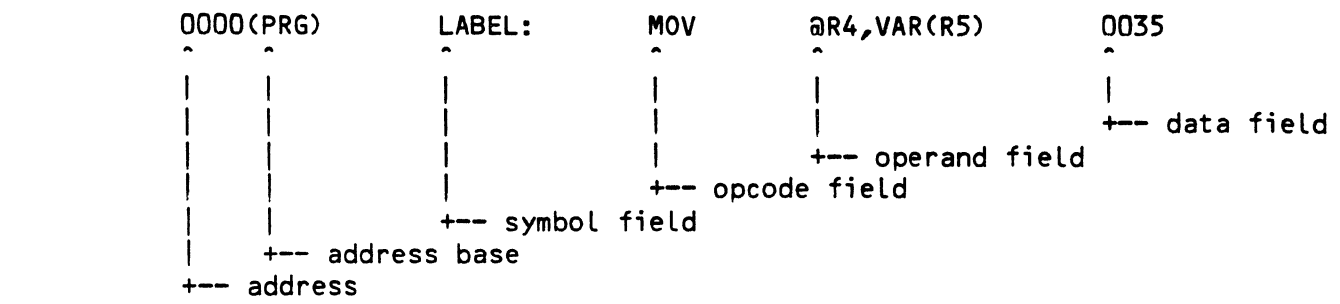
One of these two submodes (Disassembly or Dump) is the active Display Mode. Pressing ESC toggles you between Command Mode and the currently active Display Mode. The active Display Mode is indicated on the FIX status line while in Command Mode. To change the active Display Mode, use the Search and Dump commands. The Search command makes Disassembly Mode the currently active Display Mode, while the Dump command makes Dump Mode the active Display Mode.

FIX keeps a separate cursor and parameters for the Disassembly and Dump Modes. Therefore, you can change the addressing mode or move the cursor in one Display submode without affecting operation when you return to the other submode.

CHAPTER 2

DISASSEMBLY MODE

In the Disassembly mode, FIX displays memory as disassembled AM-100 op codes. This mode is a submode of Display Mode. The format is as follows:



The first field is the address. The address is displayed in the current radix (4 digits if hex, 6 digits if octal) and is an offset from the current address base. In the example line above, this base is (PRG), the base of the program being debugged. Legal address bases are:

```
(R0) - (R7)  : relative to AM-100 register
(PRG)        : relative to program base
(ABS)        : absolute
(JCB)        : relative to user's job control block
(SCA)        : relative to system communication area
```

The next field is the symbol field. If the address being displayed corresponds to a symbol, that symbol is displayed followed by a colon.

The next two fields contain the disassembly of the instruction in standard AM-100 MACRO format. FIX properly disassembles supervisor calls as well as standard machine codes. Like all disassemblers, FIX can get out of sync when displaying unusual code, such as imbedded data tables.

The final field is the data field. This field is used when single-stepping to display the result of an instruction.

Lines at which breakpoints are set are displayed in reduced intensity as they are disassembled.

2.1 DISASSEMBLER NOTES

This section describes some of the features/quirks of the FIX disassembler. The disassembler disassembles all AM-100 opcodes and AMOS supervisor calls as of AMOS release 4.4. If a symbol file (.SYM) is present for the program being debugged, it loads it and attempts to use symbolic values wherever possible. FIX only uses labels; equated symbols are ignored. If a symbol is defined twice (for instance in two different modules) only the first definition is used.

If an instruction uses absolute addressing (@#label), FIX checks for system variables (as defined in SYS.MAC).

Example:

```
        MOV     @#TIME,R1
```

FIX also checks for valid JCB symbols in job index supervisor calls.

Example:

```
        JOBIDX @R0,JOBTRM
```

Currently, conditional jumps are disassembled as a branch followed by a jump. Therefore

```
0000            JEQ     TARGET
0006            NOP
```

is disassembled as

```
0000(PRG)       BNE     6
0002(PRG)       JMP     TARGET
0006(PRG)       NOP
```

FIX attempts to disassemble OFFSET lists following TJMP and TCALL instructions. One branch is assumed after a TCALL. The list is terminated by the first label after the TJMP or TCALL. This feature is disabled if there is no symbol table or if you are currently disassembling outside of the program being tested.

Example:

```
0000(PRG)         TCALL   @R0
0002(PRG)         BR      CMDRET
0004(PRG)         OFFSET  SUB1
0006(PRG)         OFFSET  SUB2
0008(PRG)         OFFSET  SUB3
000A(PRG) CMDRET: MOV     R0,LST
```

If a BPT instruction appears in a data area, it is disassembled in low intensity. If FIX does not recognize an instruction, it is displayed as a WORD opcode followed by the value of the word in the current radix.

2.2 CURSOR COMMANDS

These commands are used to move the cursor through the program being debugged. Currently, it is not possible to modify the program while in Disassembly Mode.

2.2.1 Cursor Down : Control-J

This command moves the cursor down one instruction. If the cursor was already at the bottom of the screen, the screen is scrolled up one line and the next line is disassembled.

2.2.2 Cursor Up : Control-K

This command moves the cursor up one instruction. If the cursor was already at the top of the screen, this command is ignored (since the disassembler has no way of determining how far to back up to the last instruction).

2.2.3 Page Down : Control-T

This command moves the cursor down one page (24 instructions).

2.2.4 Initial Page : Control-R

This command moves the cursor to its initial position upon entry to Disassembly Mode. For instance, if the command

```
>s abcd: RET
```

was used to get into Disassembly Mode, pressing Control-R moves the cursor to the symbol ABCD and redisplays the page following it.

2.2.5 Home : Control-^

This command moves the cursor to the current PC. This command is useful when you have executed part of a program, then move the cursor to look at something, and then wish to restart the program.

2.3 EXECUTION COMMANDS

These commands are used to test the execution of the program being debugged. They perform single-stepping and breakpoint control.

If a second terminal is attached for debugging using the ATTACH command (See Chapter 4, "COMMAND MODE"), it is used for I/O during single-stepping and while proceeding.

If a buss error occurs while single-stepping or proceeding, the message "BUSS ERROR" is displayed. After a short delay, FIX returns to Disassembly Mode with the cursor just after the instruction that caused the buss error.

2.3.1 Toggle Breakpoint : Control-P

This command toggles a breakpoint at the current cursor location. If no breakpoint is set at the current location a breakpoint is added at this location. The line the cursor is on is redisplayed in reduced intensity. Up to 16 breakpoints may be set at one time. If 16 breakpoints are already set, this command is ignored.

If there is already a breakpoint at this location, this command clears the breakpoint and redisplays the screen in full intensity.

2.3.2 Proceed to Breakpoint : Control-X

This command starts the user program at the current cursor location. The program executes until a breakpoint is encountered (or until an EXIT supervisor call is encountered). When a breakpoint is encountered, the cursor is left at the breakpoint location in Disassembly Mode.

2.3.3 Single-step : RETURN

This command executes the instruction at the current cursor location. The cursor is updated to point to the new PC (it may be on a new page if the instruction was a transfer instruction). FIX single-steps properly through any instruction, including AMOS monitor calls. FIX prints the destination operand in the data field if it is applicable. As you single-step, the cursor "walks" through the program just as you would if you were walking through a listing.

```
000000(PRG)           MOVI   1,R1
000002(PRG)    LOOP:  DCVT   0,2
000006(PRG)           CRLF
000010(PRG)           ASL    R1
000012(PRG)           BNE    LOOP
000014(PRG)           EXIT
```

The example shown above is a sample FIX session. The cursor appears to loop on the screen until the conditional branch fails. It then falls through to the EXIT statement. (If you single-step through the EXIT, you will end up in AMOS command level.)

Hints: Not all code can be single-stepped properly, due to timing or other considerations. For instance, other users on your system could get annoyed if you single-step through a JLOCK! (NOTE: FIX will not lock the processor when single-stepping through a LOCK (IDS) instruction.)

2.3.4 Proceed from Subroutine : Control-E

This command sets a temporary breakpoint at the value on the top of the stack, and proceeds to it (proceed until PC = @SP). Any other breakpoint encountered is ignored. In a normal subroutine, this command has the effect of executing the rest of the subroutine, returning control to the user just after the original CALL instruction. However, if the subroutine plays games with the stack, this command will have undefined results.

2.3.5 Proceed to Cursor : Control-G

This command sets a temporary breakpoint at the current cursor location and proceeds to it. All other breakpoints are ignored. The terminal bell rings to indicate the operation is successful, since the cursor does not move.

2.3.6 Toggle Skip Subroutine Mode : Control-\

This command turns Skip Subroutine Mode on/off. When this mode is on, an uppercase S is displayed in the upper right-hand corner of the screen.

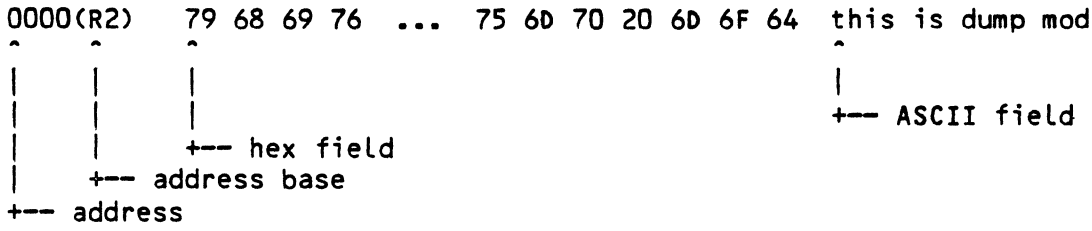
When Skip Subroutine Mode is on, the single-step command treats a CALL as a single instruction. In other words, execution of code in subroutines proceeds at full speed until the return instruction.

Warning: Skip Subroutine Mode assumes that no data follows a CALL instruction (i.e., the return address of a subroutine is the instruction following the subroutine). If a subroutine modifies the return address, this mode may crash FIX.

CHAPTER 3

DUMP MODE

In the Dump mode FIX displays memory in hex and ASCII. This mode is a submode of Display Mode. The other submode is Disassembly Mode, described in the last chapter. Each screen consists of 24 lines displaying 384 bytes. The format of each line is as follows:



The first two fields are identical to the corresponding fields in Disassembly Mode (See Chapter 2, "DISASSEMBLY MODE"). The hex field contains the hex representation of the following 16 bytes of memory. The ASCII field contains the same 16 bytes displayed as text. If a byte does not contain a legal printing character, it is displayed as a dot in the ASCII field.

While in Dump Mode, the user may move the cursor through memory to examine data, or memory may be modified in either hex or ASCII.

3.1 TOGGLE HEX/ASCII MODE : RUBOUT

When in Dump Mode, the cursor normally points to a hex number within the hex field. However, the cursor can be moved to the ASCII field by pressing RUBOUT. The cursor can be returned to the hex field by pressing the RUBOUT key again.

3.1.1 Hex Entry

When in Hex Mode, the user can modify the contents of memory by entering hex digits. If a valid hex digit is entered (0-9,A-F), the memory corresponding to the current cursor position is modified to hold that data. Both the hex and ASCII fields of the display are updated, and the cursor advances one nibble.

3.1.2 ASCII Entry

When in ASCII Mode, the user can modify memory by entering ASCII characters. If a valid ASCII printing character is entered, the current memory location is modified and the display is updated. The cursor is advanced by one byte.

3.2 CURSOR ADVANCE : CONTROL-L

This command moves the cursor forward by one nibble (4 bits) if in Hex Mode, or one byte if in ASCII Mode. If the cursor reaches the end of a line, it wraps around to the next line. If it reaches the end of the screen, the screen is scrolled up by one line.

3.3 CURSOR BACKSPACE : CONTROL-H

This command moves the cursor backward by one nibble (4 bits) if in Hex Mode, or one byte if in ASCII Mode. If the cursor reaches the beginning of a line, it wraps around to the end of the preceding line. If it reaches the beginning of the screen, the screen is scrolled down by 1/2 page and the cursor is wrapped around.

3.4 NEXT BYTE : CONTROL-W

This command moves the cursor foward to the next byte. If the cursor reaches the end of a line, it wraps around.

3.5 LAST BYTE : CONTROL-A

This command moves the cursor backwards to the preceding byte. If the cursor reaches the beginning of the line, it wraps around.

3.6 BEGINNING OF LINE : CONTROL-U

This command moves the cursor to the first byte in the current line.

3.7 END OF LINE : CONTROL-N

This command moves the cursor to the last byte in the current line.

3.8 CURSOR DOWN : CONTROL-J

This command moves the cursor down one line (16 bytes). If the cursor was already at the bottom of the screen the display is scrolled up by one line.

3.9 CURSOR UP : CONTROL-K

This command moves the cursor up one line (16 bytes). If the cursor was already at the top of the screen the display is scrolled down by 1/2 page.

3.10 RETURN

This command moves the cursor to the beginning of the next line. If the cursor was already at the bottom of the screen the display is scrolled up by one line.

3.11 CURSOR HOME : CONTROL-^

This command moves the cursor to address zero relative to the current base, and redisplays the screen.

3.12 PAGE DOWN : CONTROL-T

This command moves the cursor down one page (24 lines - 384 bytes) and redisplays the screen.

3.13 PAGE UP : CONTROL-R

This command moves the cursor up one page (24 lines - 384 bytes) and redisplays the screen.

3.14 CURSOR INDIRECTION : TAB

This command does indirection. The value of the word pointed to by the cursor is made the new cursor location. The new screen is displayed, with the cursor pointing to the address found at the old cursor. In other words:

```
CURSOR := @CURSOR
```

If the cursor is originally pointing to an odd location, this command is ignored. (This command will appear to fail on even addresses if the current base value is odd.)

CHAPTER 4

COMMAND MODE

4.1 INTRODUCTION

This chapter describes the Command Mode of FIX. Command Mode is entered when FIX initially starts running, and may be entered by pressing ESC while in Display Mode. Command Mode is used to set up the debugging environment.

The Command Mode screen displays the current status of FIX and the program under test. The first line of the screen displays the current FIX status. Status displayed includes:

current Display Mode (disassembly/dump)
current base address and mode
current radix
attached terminal (if any)

The name of the program being debugged is displayed on line 3. The prompt appears on line 5. The current AM-100 user program registers are displayed on lines 9 and 10. If any breakpoints are set they are displayed beginning on line 14.

Example:

```
AlphaFix 1.0 Status: Disassembly  (PRG)=021360  octal

Debugging USER.PRG

>

  Registers:

R0 = 000000   R1 = 000000   R2 = 020678   R3 = 000000            N Z V C
R4 = 000000   R5 = 000000   SP = 004360   PC = 021360   FLAGS = 1 0 0 1

  Breakpoints:

000026  LOOP:
000040  SUBR:
000060  LOOP3:
```

The user may leave Command Mode and return to Display Mode by pressing the ESC key.

4.2 COMMANDS

The basic format of FIX commands is the command name followed by a parameter list. Where the command name is listed with a parenthesis, the command can be abbreviated with a single letter.

4.2.1 S)earch

This command looks up a symbol, and, if the symbol is found, displays the page following the symbol in Disassembly Mode. If the symbol does not exist, or if no symbol file exists, the mesage "What?" is displayed. This command leaves the user in Disassembly Mode.

Example:

```
>s loop: RET
```

This example searches for the symbol LOOP. The colon is optional.

Note that this command is not a text search command, as in VUE. Searches may only be made for symbols.

A value may be specified instead of the symbol. This moves the cursor to the specified relative address.

Example:

>s 332 RET

If no argument follows the Search command, FIX returns to Disassembly Mode at the last location examined. The Search command always makes Disassembly Mode the active Display Mode.

4.2.2 D)ump

This command causes FIX to enter Dump Mode. There are several formats which may be used:

D address

This sets the cursor to the relative address specified. The current relative base is not affected.

Example:

>d 20 RET

D @ mode

This sets the cursor to relative zero, and changes the current base mode to the mode specified. Usually, this is an AM-100 register.

Example:

>d @r5 RET

D @ address (mode)

This sets the cursor to the address specified, and changes the current base mode to the mode specified.

Example:

>d @4(sp) RET

D

This returns you to Dump Mode without changing the cursor or base mode.

The Dump command always makes Dump Mode the currently active Display Mode.

4.2.3 Q)uit

This command is the normal method of leaving FIX. All modules in memory are deleted, and the user returns to AMOS.

4.2.4 F)inished

This is an alternate method of leaving FIX. It is identical to Quit except that the memory module being debugged is not deleted. After returning to AMOS, the user may SAVE the module.

4.2.5 Register Modification

These eight commands allow the user to modify the AM-100 registers. The register name is followed by the new value of the register.

Example:

>R0 2356 RET

set R0 to 2356

The user should be careful when modifying SP. Any data below SP is invalid, since the first thing FIX does upon acquiring control from the user program is to save the PS and PC on the stack. In other words, it is OK to increase the SP but not to decrease it.

The program counter can also be modified by moving the cursor in Disassembly Mode and either single-stepping or proceeding.

4.2.6 Set Address Base Mode

This command sets the current Base Mode. The form of this command is:

>rel address mode RET

The legal modes are:

```
R0  - relative to R0
R1  - relative to R1
R2  - relative to R2
R3  - relative to R3
R4  - relative to R4
R5  - relative to R5
R6  - relative to R6
SP  - relative to R6
R7  - relative to R7
PC  - relative to R7
PRG - relative to base of program
ABS - absolute mode
JCB - relative to user's JCB
SCA - relative to AMOS System Communication Area
```

FIX keeps separate modes and cursors for Disassembly and Dump Modes. The address mode is set for the Display Mode used most recently. The address mode can also be modified by the Dump command (see above).

Example:

```
>rel r5 RET
```

Set address base mode to R5.

4.2.7 Spec Command

This command stores a line of text and leaves the user's R2 indexing the stored line. It is used to simulate the set up of command lines by AMOS on entry to a user program.

Example:

```
>spec =dsk0:[20,4]fix.prg,*.fix RET
```

The example sets up a command line for testing a wildcard program.

If no argument is specified, SPEC lists the last line SPEC'd.

Example:

```
>spec RET
=DSK0:[20,4]FIX.PRG,*.FIX
```

4.2.8 Go Command

This command is used to fetch and execute another AMOS program. It is used to debug drivers or BASIC assembly language subroutines. The subroutine or driver is loaded and breakpoints set. The GO command is then used to start a program that calls the driver or subroutine. When a breakpoint is encountered, control returns to FIX and the user may debug the subroutine or driver.

Example:

```
AlphaFix 1.0 Status: Disassembly  (PRG)=021136  octal

Debugging INPUT.SBR

>go run menu  ; execute basic interpreter (RET)
```

In the example shown, the user is debugging INPUT.SBR, an XCALL subroutine for BASIC applications. The BASIC program MENU.RUN contains an XCALL INPUT. If a breakpoint is set in INPUT.SBR, FIX gains control when BASIC calls INPUT.SBR, allowing INPUT.SBR to be debugged under FIX.

4.2.9 Proceed

This command starts program execution at the current user PC location. It is equivalent to entering Disassembly Mode and pressing Control-X. When a breakpoint is encountered, you are left in Disassembly Mode with the cursor at the new PC location.

4.2.10 Attach

This command allows the user to use separate terminals for FIX and user program I/O. The terminal specified is used for all program output.

Example:

```
>attach term2 (RET)
```

If the program leaves FIX via an EXIT supervisor call (instead of via Quit or Finish), the second terminal will now be attached to your job.

This feature is useful when debugging programs that do screen formatting. If another terminal is attached, FIX and the user program do not interfere with each other's screen display.

4.2.11 Set

This command sets the radix used by FIX for display and input. Note that Dump Mode always uses hex. The Set command is followed by the desired radix.

Example:

```
>set octal (RET)
```

FIX always starts up in the radix the user's job is set for.

4.2.12 Break

This command sets a breakpoint. It accepts a single argument, either a symbol or a hex value. A breakpoint is set at the specified value.

Example:

```
>break cml2: (RET)
```

This example sets a breakpoint at the label CML2 if it exists. The colon is optional.

4.2.12.1 Pass Points - In addition to simple breakpoints, you may also set Pass Points in your program. Pass Points are a special form of breakpoint. When the program reaches a Pass Point, it increments a counter (the pass counter) associated with that point. If the pass counter has reached a certain predefined value (the pass count), a breakpoint is executed and you regain control in Disassembly Mode. Otherwise, the program continues normal operation.

Pass Points are set using the Breakpoint Command with an additional argument specifying the pass count. While in Command Mode, the breakpoint is now displayed with two additional values. The first, displayed in normal intensity, is the current pass counter. The second value, displayed in reduced intensity, is the pass count.

Example:

```
>break loop 4 (RET)
```

The example sets a Pass Point at LOOP with a pass count of 4. (Note that the pass count is expressed in the current radix.) This means the program will not break until it reaches LOOP for the fourth time.

A secondary use of Pass Points is a simple histogram generator. If the pass count is set to OFFFF (177777 octal), the pass counter will always contain the number of times that point has been reached during execution. By setting pass points at crucial subroutines it is possible to determine how many times they are called during a program run.

Pass Points do impose a speed overhead on the program being debugged. Normally, this overhead is under 50 instructions/pass point. Putting a Pass Point on a supervisor call increases the overhead significantly.

4.2.13 Clear

This command clears a breakpoint. It accepts a single argument, either a symbol or a hex value. If a breakpoint is set at the specified location, it is cleared, otherwise, this command is ignored. If a * is specified, all breakpoints are cleared.

Examples:

>clear cml2 RET

clear breakpoint (if any) set at cml2

>clear * RET

clear all breakpoints

4.2.14 INI

This command processes a file which contains FIX commands. Any FIX command (except INI) may be used in the command file. The default extension is .INI.

Example:

>ini breaks RET

process command file BREAKS.INI.

4.2.15 List Overlays (FXO)

This command lists the Overlay Commands available. For more information on Overlay Commands, see the next chapter.

Example:

```
>fxo (RET)
Fix Overlay Files

DDB     DIR     EA      HELP    JCB     LABELS  MAP     NEW
RAD50   RPN     TRMDEF  TYPE
```

CHAPTER 5

FIX OVERLAY COMMANDS

5.1 INTRODUCTION

In addition to the commands described in the last chapter, a number of additional commands are available in FIX Command Mode. These commands, however, are not permanently resident, but are overlays. If an overlay command is entered, the overlay is fetched into memory from DSK0:[7,0] and executed. FIX overlay files have an extension of .FX0. To get a list of the overlay commands, type FX0 followed by RETURN. FIX lists the overlay commands available.

5.2 HELP

This command displays a menu of FIX commands.

5.3 MAP

This command displays the current memory map of your partition. The name (if any), size and address of each module are listed, along with the free space. If a module is permanently loaded, it is displayed in reduced intensity.

```
AlphaFix 1.0 Status  Disassembly (PRG)=101406  octal

Debugging LOOP.PRG with SYM

>map (RET)

User Memory Map

Module          Size    Address
FIX     PRG     16078   040012
IMPURE  FIX     1070    077330
LOOP    PRG     284     101406
LOOP    SYM     60      102042
IOBUF   FIX     522     102136
FREE            30880   103136
```

The example above is a typical memory map. The modules shown are:

FIX.PRG	AlphaFIX debugger
IMPURE.FIX	scratch space used by FIX
LOOP.PRG	program being debugged
LOOP.SYM	symbol table of program being debugged
IOBUF.FIX	FIX disk I/O buffer
FREE	non-allocated memory space

If the program being tested allocates any memory modules, they appear after IOBUF.FIX. Usually, they do not have any names. A common module is a 522 byte disk I/O buffer allocated by an INIT call.

5.4 NEW

This command deletes memory modules. If no argument is specified, the highest memory module is deleted. If a * is specified all memory modules allocated during the debugging session are deleted.

The following examples refer to the memory map in the last example.

```
>new (RET)
```

This command would delete the module IOBUF.FIX. Since this module is used by FIX for disk I/O, this module should not normally be deleted.

```
>new * (RET)
```

This command has no effect, since no modules have been allocated during the session.

5.5 JCB

This command displays the current status of a job. If no argument is specified, your job status is displayed. If a job name is specified, that job's status is displayed.

```
>jcb system (RET)

Job Name = SYSTEM      Logged on DSK0:[21,1]      Job Priority = 0

Job Status = RN
Job Type = J.USR J.LPT J.GRD
Job Memory = 1:04000-163312
JOBTRM = 030000 WIZARD
```

The JCB command may also be used to display a summary of the system status, similar to the AMOS command SYSTAT. To display the system status, enter JCB with an argument of *.

```
>jcb * (RET)

BILL    TERM1   DSK0:[1,4]    ^C  MEMORY
SYSTEM  WIZARD  DSK0:[21,7]   RN  FIX
BCKGN1  PTY1
SPOOL   SPLT    DSK0:[1,2]    EW  LPTSPL
```

5.6 TRMDEF

This command displays the current status of a terminal. If no argument is specified, your terminal's status is displayed. If you are running with a terminal attached for debugging, the attached terminal's status is displayed. If a name follows the TRMDEF, that terminal's status is displayed.

```
>trmdef wizard (RET)
Terminal = WIZARD

Terminal Status  = 000363
Interface Driver = 025546 AM300
Terminal Driver  = 030106 SOROC
Inp Char Count   = 000000
Echo Char Count  = 000000
Brk Char Count   = 000000
Input Buf Link   = 030760
Input Buf Size   = 000144
Output Buf Link  = 031124
Output Buf Size  = 000144
Output Buf Index = 000004
Beg Position     = 000130
Current Position = 000022
Last Char Input  = 000015
```

The TRMDEF command can also be used to display a summary of all terminals (similar to the AMOS TRMDEF command) by using an argument of *.

```
>trmdef * (RET)

TERM1   BILL     025440  AM300   000001  BEE     100,100,100
WIZARD  SYSTEM   030000  AM300   000002  SOROC   100,100,100
P                031442  AM300   000004  SOROC   2,2,10
PTY1    BCKGN1   031600  PSEUDO  000000  PSEUDO  100,100,100
SPL1    SPOOL    032526  PSEUDO  000000  NULL    9,9,2
```

5.7 EA

This command computes an Effective Address. It accepts one standard argument.

```
>ea @r2 (RET)

        Effective Address = 020000

>ea 4(r2) (RET)

        Effective Address = 020004

>ea symbol(prg) (RET)

        Effective Address = 161552

>ea symbol (RET)

        Effective Address = 161552
```

5.8 RAD50

This command dumps 2 words in memory in RAD50 format. It accepts one standard argument.

Example:

```
>rad50 @r2 (RET)
DATL$Z
```

5.9 DDB

This command displays the contents of a DDB. The address of the DDB is specified by a standard argument.

```
>ddb infile(r5) RET

File Name              = SEQIO.MAC
Flags                  = 000100  inited
Error                  = 000000

Buffer Link            = 102742
Record Size            = 001000
Buffer Index           = 001000  103742(ABS)
Record Number          = 010574
Queue Link             = 022166
JCB Address            = 000000
Open Code              = 000001  sequential input

File Size              = 000011
Active Bytes           = 000247
First Record           = 010574
```

5.10 DIR

This command displays directories. Wildcards are allowed in the filename and extension but not in the drive name or PPN. The default Filespec is *.*.

```
>dir *.prg RET

FIX    PRG  KBD    PRG  OUT    PRG  LOOP    PRG
```

```
>dir xeq.*[21,1] RET

XEQ    MAC  XEQ     OBJ  XEQ     PRG
```

5.11 TYPE

This command displays a file on the terminal.

```
>type l.cmd RET
:T
LINK MAIN,SUB1,SUB2,UTIL

Enter CR to return to Command Mode -
```

5.12 LABELS

This command displays the symbol table of the program being debugged.

5.13 RPN

This overlay is an RPN calculator useful for scratchpad computations. It operates in any base from 2 to 16, and can be used for base conversion. Its operation is similar to calculators using Reverse Polish Notation.

When you enter RPN, the initial display appears as follows:

```
AlphaFix 1.0 Status  Dump  (R4)=08A3  hex

Debugging USER.PRG

>rpn (RET)

AlphaFIX RPN calculator
                                    Current Base is 10
))

Stack Empty
```

The double parenthesis is the RPN prompt symbol. You may now enter a line of text.

The standard form of an RPN command is a series of 0 to 9 numbers optionally followed by an operator:

))⟨number⟩ ⟨number⟩ .. ⟨number⟩ ⟨operator⟩ (RET)

Any operator terminates input immediately and causes RPN to process the line.

5.13.1 The RPN stack

RPN processes numbers on a 10-level arithmetic stack. Before a number can be processed, it must be pushed onto the stack. All operators manipulate values on the stack, usually the top two levels. RPN displays the contents of the stack below the prompt line.

The RETURN key corresponds to the ENTER key on a calculator. It pushes the values on the current prompt line (if any) onto the stack.

OPERATION		RESULT
))2 RET	--->	))
Stack Empty		2
))4 RET	--->	))
2		4 2
))9 8 7 RET	--->	))
4 2		7 8 9 4 2

5.13.2 RPN operators

There are 18 operators available in RPN. They are:

^	(change base)
@	(clear stack)
ESC or Q	(return to FIX Command Mode)
+	(add)
-	(subtract)
*	(multiply)
/	(divide)
\	(modulo)
&	(and)
|	(or)
:	(exclusive or)
<	(shift stack left one bit)
>	(shift stack right one bit)
=	(negate top of stack)
~	(swap top two stack elements)
R	(roll stack elements right)
L	(roll stack elements left)
P	(pop top of stack)

These operators may either be used alone or following a string of numbers. If the operator follows a string of numbers, the numbers are pushed onto the stack before the operation is performed.

5.13.2.1 Change Base (^) - This operator changes the current base. It must be preceded by a valid decimal number from 2 to 16. This number becomes the new base. If the number is preceded with an N, numbers are interpreted as 2's complement, and numbers greater than 32768 are displayed as negative.

OPERATION		RESULT
))16^ RET	--->	))
27 65534		1B FFFE
))n10^ RET	--->	))
1B FFFE		27 -2

5.13.2.2 Clear Stack (@) - This operator clears the entire arithmetic stack, leaving it empty.

OPERATION		RESULT
))@ RET	--->	))
3 5 7 11		Stack Empty

5.13.2.3 Quit (ESC or Q) - This operator leaves RPN and returns you to FIX Command Mode. If Q is used it must be the first character on the line.

5.13.2.4 Add (+) - This operator adds the top two elements on the stack, leaving the result on the top of the stack.

OPERATION		RESULT
))+ RET	--->	))
3 5		8

5.13.2.5 Subtract (-) - This operator subtracts the top of stack from the next to top of stack, leaving the result on the top of the stack.

OPERATION		RESULT
))- RET	--->	))
3 5		2

5.13.2.6 Multiply (*) - This operator multiplies the top two elements on the stack.

OPERATION		RESULT
))* RET	--->	))
2 8		16

5.13.2.7 Divide (/) - This operator divides the next to top of stack by the top of the stack.

OPERATION		RESULT
))6 3/ RET	--->	))
Stack Empty		2

5.13.2.8 Modulo (\) - This operator divides the next to top of stack by the top of the stack, and leaves the remainder on the top of the stack.

OPERATION		RESULT
))\ RET	--->	))
3 11		2

5.13.2.9 And (&) - This operator ANDs the top two elements on the stack.

OPERATION		RESULT
))& RET	--->	))
36 12		4

5.13.2.10 Or (|) - This operator ORs the top two elements on the stack.

OPERATION		RESULT
))| RET	--->	))
36 12		40

5.13.2.11 Exclusive Or (:) - This operator Exclusive ORs the top two elements on the stack.

OPERATION		RESULT
)): RET	--->	))
36 12		40

5.13.2.12 Shift Left (<) - This operator shifts the top of the stack left one bit.

OPERATION		RESULT
))< RET	--->	))
36		72

5.13.2.13 Shift Right (>) - This operator shifts the top of the stack right one bit.

OPERATION		RESULT
))> RET	--->	))
36		18

5.13.2.14 Negate (=) - This operator negates the top of the stack.

OPERATION		RESULT
))= RET	--->	))
36		65500

5.13.2.15 Swap Stack (~) - This operator swaps the top two elements of the stack with each other.

OPERATION		RESULT
))~ RET	--->	))
1 7 14 21		7 1 14 21

5.13.2.16 Roll Stack Elements Right (R) - This operator rolls the stack elements right one position.

OPERATION		RESULT
))R RET	--->	))
1 3 5 7		7 1 3 5

5.13.2.17 Roll Stack Elements Left (L) - This operator rolls the stack elements left one position.

OPERATION		RESULT
))L RET	--->	))
2 4 6 8		4 6 8 2

5.13.2.18 Pop Stack (P) - This operator pops the top element of the stack and throws it away.

OPERATION		RESULT
))P RET	--->	))
1 4 9 16		4 9 16

Index

VARE PUBLICATIONS FILE REFERENCE NUMBER: **ALPHA FIX USER'S MA**

SOFTWARE DOCUMENTATION READER'S COMMENTS

preciate your help in evaluating our documentation efforts. Please feel free to attach additional comments. If you require a written response, check h

NOTE: This form is for comments on software documentation only. To submit reports on software problems, use Software Performance Reports (SPRs), available from Alpha Micro.

comment on the usefulness, organization, and clarity of this manual:

u find errors in this manual? If so, please specify the error and the number of the page on which it occurred.

kinds of manuals would you like to see in the future?

indicate the type of reader that you represent (check all that apply):

- ☐ Alpha Micro Dealer or OEM
- ☐ Non-programmer, using Alpha Micro computer for:
  - ☐ Business applications
  - ☐ Education applications
  - ☐ Scientific applications
  - ☐ Other (please specify):

_____

- ☐ Programmer:
  - ☐ Assembly language
  - ☐ Higher-level language
  - ☐ Experienced programmer
  - ☐ Little programming experience
  - ☐ Student
  - ☐ Other (please specify):

_____

: _____ DATE: _____

: _____ PHONE NUMBER: _____

NIZATION: _____

ESS: _____

_____ STATE: _____ ZIP OR COUNTRY: _____

TAPLE

STAPLE

FOLD

PLACE
STAMP
HERE

alpha
micro

17881 Sky Park North
Irvine, California
92714

ATTN: SOFTWARE DEPARTMENT

FOLD

CUT ALONG LINE