12 January 1984

This document describes the enhancements made to VAMP.PR for MP/AOS Revision 2.00. All enhancements are valid for the cross-development MVAMP.PR programs also released in Revision 2.00. All previous VAMP syntax/features are still viable.

Expression Syntax

(1) Arithmetic expressions may contain the following:

   (a) Spaces
   (b) The multiplication operator, '*'.
   (c) Nested subexpressions (up to 9 nest-levels of parentheses).
   (d) The extraction operator, '@', within a sub-expression.

(2) A period may be appended to any number as an attribute operator to direct VAMP to interpret the number as a decimal number, instead of as an octal number (default).

(3) The ASCII-operator syntax may contain "<=> as a byte-mask to provide 2 new capabilites in a patchline. In both cases, the syntax directs VAMP to effectively 'ignore a particular byte' in its processing. The new uses:

   (a) To compare a value against the current content of a particular byte (of a particular word). Example:

```
76000, "<=>"<361>, 12345
```

   This directs VAMP to compare the value 361 against the low-order byte (bits 8-15) of word 76000. The '"<=>' instructs VAMP to ignore this byte in the comparison process. Example:

```
76000, "<=>"<=>, 12345
```

   This directs VAMP to not perform any comparison. This syntax is used in the EJMP patchlines automatically synthesized by the new %INSERT and %RETURN commands (described below).

   (b) To change a particular byte of a particular word, while leaving the other byte of that word intact. Example:

```
76000, 60542, "x"<=>
```

   This directs VAMP to leave the low-order byte as 142 (the character 'b') and replace the high-order byte, 141 (the character 'a') with the character 'x'.

Next-Address Symbol

When a patchline is entered in interactive mode, and the new value (argument 3) is an Eclipse instruction which assembles to 2 words, the next_address symbol, '&', is automatically incremented by 2 (instead of just 1).

Help Files

All help files have been upgraded to reflect enhancements to syntax, interactive commands, and other features.

%DISPLAY Command

Any one of eight switches may be appended to the mnemonic:

none	Redisplay word content as an Eclipse instruction, using LEF mnemonics instead of I/O mnemonics.
/A	Redisplay word content as a 2-characters/ASCII-code.
/2	Redisplay word content as a binary integer, 3-bit grouped, zero-filled.
/10	Redisplay word content as a decimal integer.
/16	Redisplay word content as a hexidecimal integer.
/IO	Redisplay word content as an Eclipse instruction, using I/O mnemonics instead of LEF mnemonics.
/P	Display current contents of the patch file.
/E	Display current contents of the error file.
/L	Display current contents of the log file.

%EVAL Command

Any one of five switches may be appended to the mnemonic, to specify the format of the display:

none	Display result as an octal number
/2	Display result as a binary number, zero-filled, 3-bit grouped
/10	Display result as a decimal number
/16	Display result as a hexidecimal number
/A	Display result as 2 characters/ASCII representation
/O	The argument must contain at least 1 overlay designator, and the associated overlay file must be opened. VAMP treats evaluated expression as an overlay file address, converts the address to its program-relative value (i.e., as if the overlay were loaded), and displays this value as an octal number.

Automatic Patchline Insertion Features

These features will aid the user in building a patchfile which requires a patch area, during an interactive 'E'-mode session. They provide a write-protected patch area within the root target file, and greatly simplify the chore of jumping to and from the patch area from the main code path. The %STATUS command display has been updated to provide information about the patch area, such as start and end addresses, pointers to the next available patch area word, and the number of available words remaining (these display changes are described below).

(1) Establishing a binder-reserved patch area

Only a program file (.PR), kernel intermediate file (.KNL), or superviser intermediate file (.SUP) may have a binder-reserved patch area. The patch area is allocated when these files are bound for MP/AOS, by including the /PATCH syntax on the BIND command line to specify patch area word size and segment location. The following syntax is entered on the command line as a psuedo-module name (in fact, the binder creates a dummy module as the patch area, listed on the program load map), anywhere among the other program modules:

```
   <segment>/PATCH=<size of patch area>
```

 Any one of the following:

ZR	--	ZREL
IC	--	Impure code
ID	--	Impure data
PC	--	Pure code
PD	--	Pure data

<size of patch area> -- Word size; octal radix unless a decimal point is suffixed, indicating decimal radix.

Example:

```
x bind/n/liblist/mpaos/p==putz/tasks=3/rev=[:ks:ozmos:rev]    &
      PC/PATCH=100    main asm encode interface lookup memory &
      parse pukool respond res_comm utils &
      !* ovly_1 dasm decode &
      !  ovly_2 install ovly_comm &
      syspascal.lb mmsl.lb urt.lb syspascal_1.lb
```

The start address, permanent avail-pointer (described below), and end address get stored in consecutive locations in the file header, beginning at location <last-file-address>-176r8. If no patch area is reserved, all 3 words are set to 177777r8.

NOTE: The /PATCH syntax may only be used when binding programs for MP/AOS systems.

(2) Maintaining the patch area

A variety of pointers and counters are maintained for use of the patch area (see the new %STATUS command display, below). In particular are the permanent and current avail-pointers, and the permanent and current words-remaining counters. The current pointer and counter get updated each time the user interactively enters a patchline destined to install a word in the patch area, at an address equal to or greater than the present pointer value. They provide current patch area status, varying even if in Non-Append mode (both get reset to their permanent counterparts when switching from Non-Append to Append modes).

The permanent pointer and counter get updated ONLY upon actual installation of the patch area patches. The permanent pointer is read from the file header when VAMP is invoked in any mode (the current pointer is then initialized to this value) and written back to the header when VAMP terminates.

Any time the root target file is bound with the /PATCH syntax, a blank (all zeroes) area is allocated; this clears any previous patch area, and resets the file header values accord-ingly.

(3) Using the patch area -- The new %INSERT command

Whenever the user must make insertion patches (i.e., not simply replacing exisitng code with new code on a 1:1 basis, but instead ADDING more lines of code), he must install at the right location a jump instruction to some unused pure area. He must install the new instructions beginning at the jumped-to location, then install a jump instruction to return to the main code path. He may decide to install, before this return-jump, those instructions overwritten by the first jump. And he must be very careful to select a patch area which does not overwrite necessary pure code or have itself overwritten.

The %INSERT command directs VAMP to do the following:

(a) Perform validity checking on the address;
(b) Examine the instructions at and around the address (discussed below);
(c) Synthesize a comment patchline;
(d) Synthesize a patchline to install an EJMP instruction at the address, directing a jump to the next available word in the patch area (based on the current avail-pointer).
(e) Save the 2 overwritten instructions if the /D switch is not appended to the %INSERT mnemonic.

Command format:

```
%INSERT[/D]   <address specifying point of EJMP installation>
```

%INSERT may be entered in Append or Non-Append edit modes, as may %RETURN. If %INSERT is entered in Non-Append mode, then the user switches to Append mode and all activity in Non-append mode gets ignored: nothing has been written to the patchfile, the current avail-pointer is reset to the permanent next-avail-pointer, the current words-remaining is reset to the permanent words remaining, and the user need not enter %RETURN if he had previously entered %INSERT in this mode. A special message is displayed to remind the user that the current avail-pointer and current words-remaining have been reset (as would be evident by a subsequent %STATUS display). When switching from Non-Append to Append mode, an informatory message is displayed:

```
*  %ED/A  <nl>

   Address of next available patch word for
      current edit session reset to 25000

   Comment and correct patch lines will be appended
```

The EJMP instruction will get installed beginning at the specified address. The specified address may be an address within the program space or associated overlay file. When the %INSERT command is entered, a synthesized comment is displayed and written to the patchfile (if in Append mode), and the EJMP-instruction patchline is synthesized, displayed, and -- if in Append mode -- written to the patchfile. Example:

```
* %INSERT/D  50  <nl>

; *****  Jump to next available word in patch area
50, "<=>"<=>, EJMP 25034 0
```

When %INSERT entered, the current-address symbol, '.', is set to the next available address in the patch area, and the next-address symbol, '&', is set to this value plus 1.

If the user enters the current-address or next-address symbols within patchlines whose address fields refer to locations within the patch area, these symbols are automatically replaced by their numeric values before writing the patchline to the patchfile. The new version of the patchline is displayed to the user. The anchor-address symbol, entered in ANY patchline, always gets replaced by its numeric value.

(4) Using the patch area --- the new %RETURN command

When the user is finished interactively building patchlines which will install instructions into the patch area, he must enter the %RETURN command. This directs VAMP to do the following:

(a) Perform validity checking on the address;
(b) Synthesize 2 patchlines to re-install the code over-written by the %INSERT EJMP instruction if the /D switch was NOT appended to the %INSERT mnemonic;
(c) Synthesize a relevent comment patchline if (b) was performed;
(d) Synthesize a patchline to install an EJMP instruction directing a jump back to the main code path (at the address, or at the return address saved by the %INSERT command).;
(e) Synthesize a comment patchline.

Command format:

```
%RETURN    [ <return address> ]
```

If the return address is not specified, the EJMP instruction will direct a jump to the the location 2 words past the location where the %INSERT EJMP instruction is to be installed. Example:

```
* %RETURN

; *****  Re-install the 2 words overwritten by %INSTALL
25035, "<=>"<=>, 105710
25036, "<=>"<=>, 101000
; *****  Jump back from patch area to main code path
25037, "<=>"<=>, EJMP 52 0
```

If the user is in Append mode and has entered the %INSERT command, then enters %BYE to terminate the interactive session before entering the complementary %RETURN command, VAMP displays a warning message, then performs the %RETURN processing as if the user had entered %RETURN with no address argument. Example:

```
* %BYE  <nl>

WARNING:  %RETURN automatically executed

; *****  Jump back from patch area to main code path
25037, '<=>'<=>, EJMP 52 0
```

(4) Special Validity Features/Considerations

(a) The user must not install his own jump instructions to/from the patch area; he must use %INSERT and %RETURN to maintain the integrity of the patch area pointers and counters. He may install patchlines which change any word within the patch area without having entered the %INSERT and %RETURN commands; in this context, the pointers and counters do not change until he enters a patchline the address field of which references a word in the patch area AT or BEYOND the current avail-pointer value.

An error message is displayed if:

(1) The user enters the %INSERT or %RETURN command and there is no binder reserved patch area.

(2) The user enters the %INSERT command, after having previously entered an %INSERT command in the same edit mode, without an intervening %RETURN command.

(3) The user enters the %INSERT or %RETURN command with an argument evaluating to an address in which the the value <address+1> lies beyond the file in which the %INSERT EJMP instruction is to be installed.

(4) The user enters the %INSERT command and there is not at least 1 available word in the patch area (according to the current avail-pointer), beyond the words committed via the %RETURN command.

(e) The user enters the %RETURN command, after having previously entered a %RETURN command in the same edit mode, without an intervening %INSERT command.

(b) There are some contexts in which a 2-word EJMP instruction may not be installed via the %INSERT command. An error message is displayed if any of the following is true (the user furnishes <address>):

(1) A 2-word instruction begins at <address> + <-1, 1>

(2) An ALC instruction with skip option begins at <address> + <-1, 0, 1 >.

(3) A CLM instruction with identical accumulator arguments begins at <address> + <-2, -1, 0, 1>.

(4) An SYC 0 3 system call instruction begins at <address> + <-2, -1, 0, 1>.

Example:

```
   * %INSERT  76502
     ^
     Not valid at this address
```

(c) The user may not wish to save the overwritten code to be reinstalled in the patch area via the %RETURN command if the code involves PC-, AC2-, or AC3-relative addressing. A warning message is displayed, followed by a query allowing the user to abort the %INSERT process, if the /D switch is NOT specified on the the %INSERT command, and there is an instruction involving PC-, AC2-, or AC3-relative addressing at <address> + <0, 1>. This occurs for each of the 2 overwritten instructions. Example:

```
   * %INSERT  52  <nl>

     WARNING:  Address 53 contains the instruction
                          LDA  1  2  3
               which is AC3-relative.  This word will be
               automatically re-installed in the reserved
               patch area via %RETURN.

   Do you want this %INSERT command completed ? (Y/N) [N]:
```

(d) When the user interactively enters a patchline directing installation of a patch to the patch area, a warning message is displayed if there are eight or fewer remaining available words (after subtracting patch words already committed). Example:

```
   * &, MOVZL# 1 1 SNR, MOVZL# 0 0 SNR  <nl>

     WARNING:  Only 12. words left in patch area now,
               of which 4. words are committed via %RETURN

     34063, MOVZL# 1 1 SNR, MOVZL# 0 0 SNR
```

%STATUS Command

(1) This command may now take an optional switch, /R, and if the switch is used, may take an optional address argument. New format:

```
%STATUS[/R  [<new next-patch area-avail. address>] ]
```

If the /R switch is used, the permanent and current avail-pointers are reset to the start address of the patch area, allowing the user to re-use all of the area.

If an address argument is supplied, the pointers area reset to this address, which must lie within the patch area.

(2) In addition to the former information, the %STATUS command now displays information about the binder reserved patch area. If no patch area has been reserved by the binder command line (when binding the root target program), the following is displayed:

```
Binder-reserved patch area:    None
```

The following is an example display, with the user having entered an %INSERT command and patchlines to patch 2 words of the patch area:

```
Binder-reserved patch area:
   Start address:     25000
   Next address:      25036        (after last installation)
   End address:       25077

   Total patch area:  64. words
   Patch words left:  34. words    (after last installation)

   Next patch word:   25040        (current edit session)
   Patch words left:  32. words    (current edit session)
                      -- 4. words committed via %RETURN
```

As described above, the first 'Next address" value is the permanent avail-pointer, and the second is the current avail-pointer. The first 'Patch words left' value is the permanent words-remaining counter, and the second is the current words--remaining counter.

The display indicates that 4 words of the patch area are already committed because the %RETURN command must eventually be entered (or automatically simulated if the user enters %BYE first): 2 words will get used to replace code overwritten by the %INSERT command, and another 2 words will get used for the EJMP command to return to the main code path.

The /R switch will change both avail-pointers and words-remaining counters to 25000 (in the above example), or to a specified address.