

SAM76 Technical Note 9 - August 1981

Publication of Technical Note No. 9 marks the fifth anniversary of the formal publication and placement of the SAM76 language in the public domain. The original specification of the language was placed in the repository of the IEEE Computer Society and then published in full in Doctor Dobbs' Journal

The original version of the SAM76 language was implemented in an 8080 system, this was rapidly followed by a Z80 version - both of these paper tape or ROM oriented. Shortly thereafter (tm) operation within the framework of the Digital Research CP/M (tm) operating system was achieved.

A version of the SAM76 language was then implemented to function on the TRS-80 as a standalone system. Another version was coded for the 6502 microprocessor; the 6502 version does not include the file management functions. A very much larger version designed to operate on DEC 10 systems was completed and placed in the DECUS library.

The latest, and probably the most interesting, version was implemented to operate on the APPLE II (tm) computer equipped with the MicroSoft Co. Z80 SoftCard (tm) This APPLE II version is significant in that it contains as a matter of course good high resolution graphics capability both on the APPLE screen and on the EPSON MX-80 printer (equipped with the "GRAFTRAX" (tm) option). The APPLE II version was also found useful in generating graphics for Video Tape systems.

The SAM76 language was used to implement a version of the "ORIGINAL ADVENTURE" game. Great care was taken to make it as similar as possible to the version found on the DEC 10 computer system. The SAM76 version was augmented to make it BI-LINGUAL (English and French). In order to demonstrate some of the possible features available in the language this latter version was further augmented with the incorporation of some 65 erotic limericks concealed between the magnetic dots on the disk sold by Creative Computing. Location of this data base was known by SAM76 but not by CP/M making it difficult to copy the disk. The APPLE II version plots a strip map of the adventurer's peregrinations.

This Technical Note provides descriptions of a number of changes in the organization of the current (Version 25) distribution disk of the SAM76 language interpreter for the 8080 and Z80 microprocessors. This note also defines new as well as enhanced old functions of the language.

```
-----
!!                                     !!
!!          SAM76 Language Contest          !!
!!                                     !!
!!-----
```

As may have been noticed there is a severe lack of SAM76 related publications, descriptions, blurbs, notices, application listings &c.....

In the hopes of encouraging the generation of material suitable for publication in any magazine or journal which may accept such material an open ended CONTEST is hereby declared.

CATEGORY 1. Explanation and or documentation of some of the application scripts to be found on the SAM76 distribution disk. In particular "SLED.SAM" the String Language Editor, and "TOP.SAM" the Text Output Processor deserve to have their status changed from carefully undocumented to carefully documented.

Likewise for users of the APPLE II graphics version the application scripts "DRAW.SAM", and other graphic oriented material should be exposed to the full light of publication.

CATEGORY 2. New or other interesting applications should be revealed and published with appropriate documentation.

CATEGORY 3. Some of the more esoteric functions of the SAM76 language should be given considerably more explanation than is to be found in the language manual. In particular the following function groups which are unique to the SAM76 language.

The "MULTIPARTITION" stuff - MT and MC  
The "pipeline" concepts of LIC, LOC, SIC, SOC  
The "Neutral Implied" function  
Clever ways of using the file streaming capabilities.  
Clever ways of using the "LW" List Where function.

CATEGORY 4. Anything else relating to SAM76.

Entries to this contest may be made either before or after publication of SAM76 language related material. Entries may be either unpublished manuscripts - preferably in machine readable form in SAM76 format or in printed form after publication in a magazine of the author's choosing. Needless to say that material already published will rate higher than unpublished material in this contest.

There is no deadline for this contest, and there is an indeterminate number of prizes. Prizes in general shall consist of interesting pieces of computer equipment guaranteed to be more than twenty years old.

	Description of changes	

1. It is not expected that the changes made in the arrangement of the software will have any impact on the great majority of users of previous versions. They are explained solely for the benefit of users who may have made machine language patches to the I/O vector jump tables. These tables have been rearranged to be in the same sequence as required by the BIOS of operating systems such as CP/M or TPM. The new arrangement may be found in the source code file named BOILER.INS.

The foregoing change was made in order to simplify the initialization process and to release a great deal of space tied up permanently, yet used only once. Additionally some of the table locations reserved for use by certain functions have been made available for joint use by other functions.

2. Exit from or entry into the SAM76 language system no longer affects the setting of the IOBYTE (memory location 3). It is now incumbent on the user to use the SIO function if it is desired to change the IO conditions whenever exiting or reentering.

3. Because of the rearrangement of the I/O vector tables and the generally universal acceptance of certain operating systems it is no longer necessary to go through a two step initialization process using ILRAW or ZLRAW and then CPMSETL or ZAPSETL. The files ILRAW.COM and ZLRAW.COM are initialized for the operating system BIOS when started up. If the user wishes to make patches to the I/O vector tables DDT should be used to bring the file into machine memory for purposes of patching and a new save file created.

4. The files IHRAW.COM and ZHRAW.COM no longer appear on the standard distribution disk to make room for other more useful information or scripts. These files provided the SAM76 language interpreter assembled for use in a high portion of memory for incorporation into Read Only Memory. Purchasers of the distribution disk may request these to be included at time of purchase or obtain same by arrangement.

5. Functions IT, IDT, and OT are no longer resident in the main body of the SAM76 object code. Their use was primarily intended for transfer of SAM76 language scripts in the form of paper tape or cassette. For obvious reasons this capability is no longer generally required. The functions are available and may be brought in, if required, through the use of the overlay file functions, and the source code for the functions removed may be found on the "distribution disk".

6. Distribution of SAM76 language object code or scripts is no longer made in paper or cassette tape formats.
7. Because of the wide availability of 8 inch disk systems orders for media suitable for NORTH STAR, MICROPOLIS and the like will be resisted unless the purchaser can honestly indicate that he would incur greater hardships than SAM76 Inc. would in transferring the files from 8 inch soft sector medium.
8. The arrangement of the files created on disk by the SAM76 interpreter has been extended to include a number of parameters which are accessed through the use of the "QFA - Query File Attribute" function. Additionally Date and Time are automatically included in the file header. Desired date may be set using the SDA function unless a real day calendar is part of the system.
9. The IEEE Computer Society is currently sponsoring a development effort aimed at generating a standard programmatic interface between user programs and operating systems. This activity is being carefully monitored with the hope of maintaining compatibility with a variety of operating systems while continuing the philosophy of full machine independence of the SAM76 language.
10. The sequence of the arguments or values of the functions related to TIME and DATE have been rearranged to conform with the provisions of ANSI standard X3.43-1977, "Representations of local time of the day for information interchange".
11. Use of the LF (line feed) and CR (carriage return) codes is precisely as defined in ANSI standard X3.4-1977 and X3.64-1979. The initialized state of the SAM76 language processors is with the LNM (Line Feed New Line Mode) SET (See section 5.55 of X3.64-19). Users of keyboards which do not provide the LF code should use "CONTROL J" to go to the beginning of the next line.



```

!!-----!!
!!               New and Extended Functions               !!
!!-----!!

```

Some of the following functions are not strictly new since they are defined in the SAM76 language manual, however they were not incorporated in the early releases of the language and some of these may have either expanded or different arguments.

`%ii,s1,s2,t,f,s3,t3,s4,t4,.../` If Identical extended

Additional arguments have been added to the "II" function so that branching to more than two places based on two arguments is possible without having to nest a series of "ii" expressions. {135}

`%ig,s1,s2,t,f,s3,t3,s4,t4,.../` If Greater extended

Additional arguments have been added to the "IG" function so that branching to more than two places based on two arguments is possible without having to nest a series of "ig" expressions. {136}

`%ge1,d1,d2,t,eq,/` Greater Equal Less branch

This function compares the values of the numbers symbolized by d1 and d2 in the current number base. If the number symbolized by d1 is greater than d2, then the value of the expression is the string symbolized by "t". If the numbers are equal then the value is the string symbolized by "eq", else the value is that string symbolized by "f".

`%rp,c,d,s1/` Return Padded

The value of this function is the string denoted by "s1" rearranged in such a manner that a "NEW LINE" code is inserted at appropriate intervals so that no line exceeds a length of "d" characters. The optional character symbolized by "c" is appended in sufficient number to each line so that all lines will then be of the same length. {248}

`%rj,d,s1/` Return Justified

This function is similar to the "RP" function, except that spaces are distributed in sufficient number throughout any given line to the effect that right column justification is achieved. If the special "justify table switch" is set, then justification is achieved using a character width table and filling may be spread out in fractions of normal spaces. {247}

**%sds,d/****Set Display Speed**

This null valued function sets the rate at which characters will be displayed as a function of the value of the decimal number symbolized by "d".

**%sfl,X/****Set Flags.**

This null valued function may be used to set or clear as desired the bits in the "flag" byte which is described under the "qfl - query flag" function. Bit positions in the "flag" byte and their meanings are enumerated below:

BIT	HEX	Meaning.
0	0	
1	02	Shift Out - (Alternate Graphics).
2	04	Pictorial Graphics if bit 1 is on.
3	08	Alternate Keyboard layout (eg: DVORAK).
4	10	Display Upper Case only.
5	20	Keyboard "CAPS LOCK".
6	40	Cursor Indicator Off.
7	80	Page Mode on (Scroll mode off).

**%qfl/****Query Flags.**

The value of this function is the hex representation of a byte which indicates the current condition of various user controllable features of the keyboard and display software. These bits may be set either by use of the %sfl,X/ function or through appropriate operation of "CONTROL char" or recognition by the display software of certain "ESC char" sequences.

**%xu,xtb,x0,x1,x2,...,xn/****XU X to Binary.**

The value of this function is the representation in base 2, of the "x-base" data symbolized by "x0,x1,x2,...,xn". The normal representation is in groups of eight digits each preceded by a space. The actual digits "0" and "1" may be changed to other characters if desired.

**%xu,wm,X,x0,x1,...,xn/****XU Write in Memory.**

This null valued function writes into successive memory locations starting with that symbolized by "X" the binary data symbolized by "x0,x1,...,xn".

**%nav,d/****Not Available switch**

This function enables the user to stipulate the manner in which the "<nav-xxx-yyy>" warnings are produced. The decimal number symbolized by "d" may be set to be 0, 1 or 2 with the following results when an "nav" warning is generated.

- 0 - The warning is returned as a value for user analysis in the user script.
- 1 - The warning is a null valued string which is displayed on the console during the course of script execution with no other interruption.
- 2 - The warning forces the execution of the user trap (%ut,xx/) function.

```

!!-----!!
!!           Disk file related functions           !!
!!-----!!

```

`%ex,filn/` Exit

When used as shown above, this null valued function will cause the work space to be shrunk as much as possible and a file whose name is symbolized by the string "filn" will be created. This file may subsequently be executed and the action will continue with whatever expressions followed the "%ex,filn/" expression. A typical use is illustrated below:

`%ex,game.com/%bf,%qcs/.sam/%A/`

`%bop,filename,X1,X2/` Bring Overlay Program.

This null valued function loads the designated overlay program starting at location symbolized by "X1"; upon completion of the load process a subroutine call is made to the address symbolized by "X2". Omission of either "X1" or "X2" causes load point or call to be made as specified in the file itself. (See UDF).

`%uop,filn,verb,X1,X2,X3/` Update Overlay Program.

This null valued function creates a file containing the data found in memory between the addresses symbolized by "X1" and "X2" inclusive; "X3" symbolizes a machine location which is to be called upon loading of this file - omission of "X3" signifies that no call is required, unless stipulated as part of the "bop" function.

`%xzs,unit,trk,sect,X/` Xperimental Zeroize Sector

This null valued function fills the sector on the specified track and unit with the binary character symbolized by "X". If "X" is the null string then the sector is filled with zeroes. {270}

`%xfs,unit,trk,sect/` Xperimental Fetch Sector

The value of this function is the string of characters found in the designated sector. {269}

`%xrr,s0/` Xperimental Read Record

The value of this function is the binary content in the current "X" base of the next available record to be read from a disk file previously opened using the "DIF" function. Each binary value is preceded by the string symbolized by "s0".

`%xwr,x0,x1,...xn/`

Xperimental Write Record

This null valued function writes the sequence of binary numbers symbolized by "x0, x1, .. xn" in the next available record of a disk file previously opened using the "DOF" function.

`%xss,seccsize,bufad/`

Xperimental Specify Sector

This null valued function permits the user to specify a desired sector size, and a buffer location for subsequent disk reads and writes using the "XRS", "XWS", "XFS", and "XZS" functions.

`%rnf,newname,oldname/`

Rename File

This null valued function renames in the current selected disk unit an existing file whose name is symbolized by "oldname" to be that symbolized by "newname".

`%qfa,filename,s0/`

Query File Attributes

The value of this function is a list of the file attributes, each preceded by the string symbolized by "s0". These attributes are as follows:

- 2 bytes - File category - File mode
- 2 bytes - Exact file size in bytes
- 2 bytes - File size rounded up modulo 128
- 2 bytes - Address of last byte of saved data
- 2 bytes - Address of first byte of saved data
- 2 bytes - Initialization reentry (0 means none required)
- 2 bytes - Reserved for future
- 3 bytes - YEAR, MONTH, DAY of creation
- 1 byte - Meridian of file creation
- 4 bytes - HOUR, MINUTE, SEC.Fract. - Time of creation
- 4 bytes - reserved for future
- 4 bytes - reserved for user needs

`%dos,d1,arg1,.../`

DOS function

This function is used to access certain of the Disk Operating System special calls. Some of them are used to set values in the operating system, while other calls are used to return values. The call number symbolized by "d1" is described in the appropriate Disk Operating System manual.

```

--
!!                                     !!
!!          PLOT or GRAPHICS FUNCTIONS          !!
!!                                     !!
--

```

The following functions are associated with the "plot" or graphics functions. Some of these functions are peculiar to the APPLE II computer or other display control systems such as the "GODBOUT SPECTRUM" board.

The user is reminded that in all versions of the SAM76 language system which include the plot functions it is possible to drive any type of incremental plotter through one of the peripheral output channels - "LST" "COM" or "USR". This is done by executing the "PL,NM" function, then execution of the "WS" function will generate the proper output to the selected channel. The low order six bits of the output stream are as required by the CALCOMP plotter convention.

```

-----
Xpl,ibr,t,f/                                If Blank Raster

```

This function returns the argument symbolized by "t" if the top raster line of the current display screen is blank (all white, or all black), else the value is that argument symbolized by "f".

```

Xpl,inv/                                Invert display screen

```

This null valued function inverts the screen, that is to say all that is black becomes white and vice versa.

```

Xpl,su,d/                                Scroll Up

```

This null valued function is used to initiate and control the vertical scrolling of the display screen. The number of raster lines moved is specified by the decimal number symbolized by "d". If "d" is omitted then continuous scrolling takes place. The active form of this function actually causes "rolling" around the defined screen, rather than "scrolling". The neutral form of the function causes scrolling off the top of the screen, the bottom being blanked out.

```

Xpl,yad,d/                                Y Address

```

This function returns the address in the current "X" base of the first pixel of the raster line specified by the decimal number symbolized by "d". Note that in all PLOT functions the motto "READ RIGHT UP" applies, that is to say that the origin of plotting is the lower left corner of the display screen unless the Xpl,mq,d/ function has been used. In the Apple II computer

the range for "d" is zero to 191.

%bfp,filename,FROM,TO/                      Bring File Picture.

This null valued function loads into a portion of memory designated for use as a High Resolution screen display a picture file ready for display. Arguments "FROM" and "TO" may be used to control the amount of the picture file loaded and its location.

%sfp,filename,FROM,TO/                      Save File Picture.

This null valued function stores a file which is found in that portion of memory designated for use as a High Resolution Screen display. The Arguments "FROM" and "TO" may be used to control the amount and the source address of the picture file which is to be saved.

%pl,dis,x1,x2,.. /                      Plot Display control

This is a null valued function which sets parameters for, or controls the operation of the display screen.

In the APPLE II computer, the "x1" symbolizes the decimal number described in the APPLE reference manual to control the screens. The neutral form of this function (%pl,dis/) is used to control the text screen, and (%pl,dis/) without argument "x1" switches back to the mode previously set.

With respect to the GODBOUT SPECTRUM display board, "x1" if specified is used to select the port address and "x2" to specify the location in the memory continuum of the display screen. Active and neutral execution of this function serves to enable or disable the flicker control software.

%rgc,d/                      Read Game Controller.

The value of this function is the setting of a "game controller" whose identity is symbolized by "d".

%ipb,d,t,f/                      If Push Button.

The value of this function is "t" if the push button whose identity is symbolized by "d" is actuated, else the value is "f".

%san,d/ or %san,d/                      Set or Reset Annunciator.

This null valued function is used to "turn on" or turn "off" the annunciator whose identity is symbolized by "d". Active use "%.." turns on, and neutral "&.." turns off.

%pl,res,d/                      Resolution specification

This is a null valued function which is used to change the

characteristic resolution for plot purposes of the APPLE display screen. Experimentally, at this time, available modes symbolized by "d" are:

0 - mode 0

1 - mode 1

2 - mode 2

3 - All 8 bits in each byte are used in plotting. Furthermore in this mode the screen is assumed to be, for purposes of plotting, 128 bytes or 968 dots wide.

Introducing  
SAM76 graphics  
and a sampler of the

*Hershey Fonts*

Font "P2UR"

!"#\$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN  
OPQRSTUVWXYZ[\]^\_`abcdefghijklmnopqrstuvwxyz{|}~

Font "P1UR"

!"#\$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN  
OPQRSTUVWXYZ[\]^\_`abcde  
fghijklmnopqrstuvwxyz{|}~

Font "P1IR"

!"#\$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN  
OPQRSTUVWXYZ[\]^\_`abcde  
fghijklmnopqrstuvwxyz{|}~



Functions specialized for the EPSON MX-80

The functions listed below are associated with the use of the MX80 printer made by EPSON, some of them being peculiar to the use of the APPLE II computer. In order to achieve the capability of printing in a single continuum of 960 dots across the paper it is necessary that the plot program treat the screen as if it were in fact a single line 960 dots wide by either 32 or 64 dots high folded over merely to allow the user to view the material to be printed.

-----

%eps/

EPSON print and control

This null valued function causes a graphics print out of the display screen as previously defined and parametrized. See all of the "eps" subfunctions for details.

%eps,lf,s0/

List {eps} functions

The value of this function is a list of subfunctions of "EPS", each mnemonic in the list is preceded by the string symbolized by "s0".

%eps,dpb,d/

Dots Per Byte

This null valued function is used in defining the characteristics of the screen display for purposes of printing. In the Apple computer the normal mode is "7 dots per byte", the eighth bit being used as a color control. However when printing in black and white better use is made of the screen memory by specifying "8 dots per byte". This function should be used in conjunction with a setting, when plotting, of "%pl,res,3/".

%eps,lps,d/

Lines per screen

This function is used as part of the screen definition for use by the MX80 printer. The lines are "logical" lines where "d" times the number of bytes per line cannot exceed the size of the screen memory.

%eps,pag/

Page (graphics)

This function causes a page feed out equal to the number of vertical dots printed in graphics mode subtracted from the length of the page also measured in dots.

**%eps,qvd/****Query Vertical Dots**

This function returns as its value the current number of dot spaces actually advanced along the length of the page.

**%eps,rpl,d/****Rasters Per Line**

This null valued function is used to specify to the printer the number of display raster lines (or vertical dots) to be printed as part of a single logical line. Typically if the Apple display screen is viewed as a screen 960 dots wide by 1 line high, then "d" would be 64.

**%eps,svd,d/****Set Vertical Dot**

This null valued function is used to specify the length of the page in terms of dot spacings. This is used by the %eps,pag/ function in determining the amount of space to be advanced on a "graphics page feed".

**%eps,bpl,d/****Bytes Per Line (screen)**

This null valued function is used to define the number of bytes of memory which will when printed correspond to one line on paper. For instance if one line on paper contains 960 dots, then at the rate of 8 dots per byte, "d" would be 120.

**%eps,wid,d/****Width**

This null valued function provides a convenient method of presetting the assortment of parameters needed to specify a particular print mode and screen specification. The table below lists available options using this function:

"d"	RPL	LPS	DPB	BPL	ESC	Comments
0	192	1	7	40	K	Screen Dump
1	32	1	8	120	L	1 line 32 by 960
2	32	2	8	120	L	2 lines 32 by 960
3	64	1	8	120	L	1 line 32 by 960
4	96	1	8	120	L	1 line 96 by 640
5	32	1	8	240	L	2 line 32 by 840
6	32	2	7	120	L	2 line 32 by 840
7	192	1	8	40	L	Compressed screen

**%eps,esc,Cx/****Escape + Char + argument**

This null valued function sends to the MX80 an "ESC" character followed by the character symbolized by "C" and optional arguments - numeric or otherwise symbolized by the string "x". A null is automatically transmitted if required at the end of the string "x".

**%eps,com/****Compressed print mode**

This null valued function sets the EPSON MX80 into compressed mode. Reset to uncompressed is accomplished through use of the neutral form of the function, to wit (%eps,com/).

**%eps,dbl/****Double print mode**

This null valued function sets the EPSON MX80 into double wide print mode. The neutral form of the function resets to normal printing.

**%eps,emp/****Emphasized printing**

This null valued function sets the MX80 printer into emphasized print mode. Normal mode is resumed through use of the neutral form of the function - (%eps,emp/).

**%eps,vhr/****Very High Resolution**

This null valued function conditions the system to control the EPSON MX80 printer in such a manner as to print along the length of the paper in increments of 1/216 inches.

**Font "C261"**

!'"#\$%&'()\*+,-./0123456789:;<=?@ABCDEFGHIJKLMNO  
PQRSTUVWXYZ[~`abcbrdfghijklmnopqrstuvwxyz

**Font "KANJ2"**

算管籠米料粒粘糸綴積穴究空罌立竹第等筆礪磁示利私和秒科秋秤称程稲種黑墨

*This demonstrates the use of the test draw program written using the SAM76 language. The character sets used were designed and digitized by Dr. A. V. Hershey for use in his work at the Navy Weapons Research Laboratory in Dahlgren, Virginia. More recently his work was adapted for use in small microcomputer systems, in this case an APPLES, and printed on an EPSON printer.*

**Font "P1IR"**

!'"#\$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO  
PQRSTUVWXYZ[~`abcdfghijklmnopqrstuvwxyz{|}

**Font "P2IR"**

!'"#\$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN  
OPQRSTUVWXYZ[~`abcdefghijklmnopqrstuvwxyz{|}

```

-----
!!                               !!
!!           Future Functions under consideration           !!
!!                               !!
!!-----!!

```

This section identifies a series of functions which are currently being developed or being considered for augmenting the capabilities of the SAM76 language. Comments as to potential value or suggested changes in definitions are earnestly requested.

-----

**%wtd,Cdx,dy,dz,dx1,dy1,dz1,.../ Write Three Dimensions**

This function is intended to function precisely in the same manner as the "WS - Write Straight Line" function except that the arguments are in groups of three. In the example above "C" symbolizes a control modifier such as "U" for "pen up", or "S" for scaling. The other arguments "dx", "dy", and "dz" are respective values of X, Y and Z either incrementals or absolutes depending on the current plot mode.

**%pl,rot,Ax,Ay,Az/ Plot rotate**

This null valued function sets desired rotational parameters for subsequent uses of the "WS" or "WTD" functions. In the example "Ax", "Ay", and "Az" symbolize the prescribed angles of rotation in the "X", "Y", and "Z" planes.

<b>%fad,n1,n2,n3,...,n/</b>	<b>Floating Add</b>
<b>%fsu,n1,n2,...n/</b>	<b>Floating Subtract</b>
<b>%fmu,n1,n2,vz/</b>	<b>Floating Multiply</b>
<b>%fdi,n1,n2,vz/</b>	<b>Floating Divide</b>
<b>%fig,n1,n2,vt,vf,..../</b>	<b>Floating If Greater</b>

This group of functions is identical to the corresponding group of integer arithmetic functions excepting that the numbers symbolized by n1, n2, &c. are floating point numbers of arbitrary precision where a "." separates the whole number from the fraction rather than being treated as a "non numeric". The precision of computation is set to correspond to the longest fractional part of the arguments in the expression - arguments whose fractional parts are shorter are padded with zeroes.

Optionally there may be a set arithmetic mode function which would permit the user to specify either pure integer or floating point action on the normal set of arithmetic functions - "AD", "SU", "MU", "DI", and "IG".

```

||-----||
||          RELATIONAL System Functions          ||
||-----||

```

The following family of functions are intended to extend the SAM76 language with a logical inference capability by assuming, or simulating an associative memory with a superimposed relational structure. The basis for and details of these functions may be found in University of Michigan Technical Report No. 5, authored by William Ash and Edgar Sibley, "TRAMP: A Relational Memory With An Associative Base" dated May 1968.

-----

**%dr,Attribute,Object,Value/      Define Relationship**

This is the associative storage function - the function that inserts the data into the structure.

The three arguments "Attribute", "Object" and "Value" (abbreviated as A, O and V) are each non empty sets. The set element delimiter is the circumflex (^). The triple is ordered and interpreted as meaning  $A(O) = V$ . Each element of each set is grouped with each pair of elements of the other two sets, and the resulting triple is stored, i.e., each point in the cartesian product is stored. The three sets are ordered sets only inasmuch as the order in which they appear in the storage declaration is retained.

**%er,Attribute,Object,Value/      Erase Relationship**

This function undoes what the "DR" define relationship established, namely erase an association from memory.

**%lr,sØ,Attr.,Object,Value/      List Relationships**

The value of this function is a list, each term being preceded by the string symbolized by "sØ".

Further description of the syntax and semantics of this function is withheld at this time.

**%lrr,sØ,Attr.,Object,Value/      List Redundant Relationships**

This function is identical to "LR" except that any redundancies are reported.

**%lir,sØ,Attr.,Object,Value/      List Intersecting Relation.**

This function has the same syntax as the one variable use of "LR". "LR" generates the UNION of the value sets, while "LIR"

generates the INTERSECTION.

%drc, SET1, SET2, NAME/ Define Relative Complement

This function computes the relative complement of two relational sets.

%dsd, SET1, SET2, NAME/ Define Symetric Difference

This function computes the symetric difference of two sets.

%dri, SET1, SET2, NAME/ Define Relation Intersection

Definition of this function is being withheld at this time.

%qna, s1/ Query Number of associations

The value of this function is the number of explicit associations that the string symbolized by "s1" is used in.

%lat, X, s0/ List Attribute Table

Definition of this function is withheld at this time.

Font "ASTRO"

astronomy: Ω♄⊕♃♂(♂♂♂\*⊙♂♂ astrology: ≈Υ♂♂⊕Ω♂♂♂♂♂♂

SAM76 INC.

BOX 257 RRI

PENNINGTON, NJ 08534

TELEPHONE (609)-466-1129