Draft version

NEW COMMAND SYNTAX for version 5

Updated 14 Dec 2023

Commands have one or more data Items, which in some cases may have nested sub-items. Each item contains option definitions, which is a letter with optional following parameters.

Key

<D>	is a decimal parameter
<X>	is a hexadecimal (optionally with 5th digit for bank)
<R>	is a register (hex), limited to 0-0xff (8061) or 0-0x3ff (8065)
<F>	is a floating point number

Full command syntax is -

<command> <pars> “name” $[item] [item] [item] …..

where -

<command>	is a text string, minimum 3 chars, followed by up to 4 parameters.
“name”	is an optional symbol name, always in quotes
[$]	is an optional set of global options (e.g. print layout)
[ ]	is one data item definition. An item defines print options for one piece of data, as a byte, word, triple or long. Items may be further split with one or more sub-items for multiple print options.
[Item [sub-item] [sub-item]]	For multiple prints of same data item, field definitions etc. Print options apply within each sub-item.
[sub-item]	Defines print options for a data item within a data item, up to and including the item itself (so that the item can be printed multiple ways).

Some commands allow a separate global data item, marked with a [$] .
Commands have different detail options and parameter arrangements allowed, as makes sense for that command.

A data item consists of one or more print options, one of which must be a size definition. Some options are not allowed in the top level item, and some options are sub-item only.

Command list. Minimum 3 chars for command string.
Additional data items are always optional, details defined in next section

command	description
args	Subroutine argument data (i.e. input parameters)
bank	define a ROM bank within the .bin file
byte	Define one or more 8 bit bytes
calc	Define a math calculation
code	Define a block of code
fill	Define an unused area (typically all set to 0xff)
func	Define a 1 dimension lookup structure, used to convert a single value into a scale, etc (i.e. x = f(x))
integer	Define one or more 16 bit words
long	Define one 32 bit long (2 x16 bit words)
pswset	Define where the PSW (Processor Status Word) is set for a conditional jump
scan	Define a point to start a Code scan
struct	Define a generic data structure
subr	Define a subroutine
sym	Define a symbol name
table	Define a 2 dimension lookup structure , for example RPM x Throttle position, etc. x = f(x,y)
text	Define a text area (treated as ASCII characters)
triple	Define a 24 bit ‘triple’ (1 x 16 bit word, 1 x 8 bit byte)
vect	Define one or more ‘vector’ pointers , which point to another address.
xcode	Define this area as data only, no code allowed in here
setopt	Set attributes for SAD default behavior, covering print, code analysis and automatic naming features
clropt	Set attributes for SAD default behavior, covering print, code analysis and automatic naming features

Command parameters

These are mainly start and end addresses as relevant to the command.

Command Option List - (global options are below)

Option letter and format	Parameter limits	Description
B <D> <D>	1-31	A field, beginning bit to end bit
O <D>	2 - 31	repeat Count
K <D>	0,1,8,9	Set bank number. Affects addresses (R, D, Vect command)
L		Item is Long (32 bits)
N		Look for symbol Name for this item (by address)
P <D>	(2-31)	Print field width
R		Item is a Reference pointer (= address). May need ‘K’ in multi-banks
S		Signed value (any size or field can be signed or unsigned)
T		Item is Triple (24 bit)
U		Unsigned (default)
W		Item is integer size (=16 bits, commonly called a ‘Word’)
W		Symbols only. This is a Write Symbol.
X <D>	2,10,16	Print radix = bin, decimal, hex X <D>.<D> for decimal places
Y		Item is byte size
|		A newline marker, to split data structure printouts across lines
=		Define a calculation (two forms, see below)

GLOBAL options

A	Use ‘Subroutine arguments’ layout (each item printed on its own line).	Default for subroutines

C	Use ‘column’ layout, each item in its own column, organised by ‘row’	Default for data structures
F <d> <x> <x>	Subroutine Special function. Used to define that subroutine has a special function.	
Q <D>	Terminator (‘quit’) bytes, (1 or 2). Used to define a terminator at end of a structure or list	For data structures
= <X>	defines a subroutine answer for printout. Can be any valid address.	(size ? B,I,T,L)

NOTE – long and triples probably need Two start addresses as they don’t have to be contiguous. This is not finalised yet.

Examples and Notes -

Simple commands have only one item attached, e.g. [W = flo(x/256)] means a single float (Word) printed after division by 256.

A pointer used for multiple lookup lists (this occurs in some binaries) could be specified as [W [= add(x+2323)] [=add(x+3456)]] which will print TWO addresses from the single integer.

The ‘top’ or ‘master’ level of the command item must have a size defined (byte, integer etc), but will not be printed if it has sub-items defined. If it is desired to print this ‘top’ level as well, then it must have a radix option as well as the size.

Subfields themselves cannot be nested but can have duplicate option sets (e.g. offsets) as in

[W [B 10 15 [=add(2323)] [= add(x+3456)]] ...]
defines a field within an integer , starting at bit 10 and ending at bit 15 (the top 5 bits) and having two printed offset values. This would print the addresses 2323+x and 3456+x, where x is value in top 5 bits.

This format would also allow a single item to be printed different ways as in

[WU [=flo(x/256)]] would print the integer as divided by 256, and as a plain unsigned value (two numbers).

[WN [N B 10 15 [= add(x+2323) N] [=add(x+3456) N]]] would print 4 symbol names, offset, offset, field, integer (if symbols found)

NOTE - UNRESOLVED - Triple and Long – If top byte/integer is not contiguous with low integer, I assume we must allow normal address start,end to become startL, startH to define lower word and upper byte/word. But how to print and represent these is still unclear. Assume this will never happen in a data structure ??

Other rules (incomplete, more to define) -

Sub-items must be single, i.e. cannot have a ‘O’ for multiple counts ?
Fields with one value are single bits, a one bit wide unsigned subfield.

Symbol names attached to subfields would still be SYM <add> : B x where x is the start bit.
Probably have to limit the nesting for safety and readability (3 levels max)?
A subitem '[...]' must be smaller or equal in size to its 'parent' item.

CALC format

= calctype (formula) short form

or CALC "name" = calctype(formula)

and linked via "= name" in additional data

calctype is integer, address,float. Address wraps in 16 bits and keeps the bank

formula is as maths, each term in round brackets and can be nested eg (x/128), x/(4+5)) , (x+3.04)/(9/5)
'X' is the value contained in the current data item.

PROPOSED COMMENT SYNTAX

‘\’ sequences can appear anywhere in the comment text, and cover layout requests

\n	insert a newline at this point
\w	Wrap. Pad out (in spaces) to comments start column on next line.
\\	print a single ‘\’ character
\[	print a single ‘[‘ character
\t <d>	Tab to column <d> (in spaces)
\m	Tab to next position (in sets of 4 spaces)
\r <c> <d>	Runout char <c> <d> times.
|	Special ‘split’ marker for multiple items in subroutine arguments – see below (change this….?)

To print items related to an opcode line, items are enclosed in square brackets, mimicing the main command syntax.

[Item] prints one item. Unlike commands, these item cannot be nested. These items are mostly about embedding symbol names and values, but can be used for other printouts too.

Options -

N <X>	Look for Symbol name	As ‘N’ in command options
V <X>	Immediate value	Print a raw number/address
R <X>	Address	As ‘R’ Value <X> is printed as “[X]”
A <D>	Operand reference (0,1,2,3)	See Note below

Operand Notes -
If an Operand is not specified, a hex value must be defined after after one of ‘N’,’I’,’R’ options to give a ‘base address’ to work from. This value can be reused once specified for multiple prints (e.g. symbol followed by its address). Examples -
[N2222 R] will print “Symbol_name [2222]”, and [R2222 N] will print “[2222] Symbol_name”

Same rules apply for Operands, so [A1 N] or [N A1] will print “symbol_name” for operand 1 (by its address), and
[A1 R N] will print “[address] sym_name” for operand 1.

Internally, SAD calculates a fourth operand for indexed address mode, this ‘operand 0’ is the ‘base address’ or ‘offset address’ as in LDW, Ry, [Rx + base]. This is included as it may be useful for comments.

NB. Using V with an operand (to print its value instead of its address) works for immediate address modes, and for indexed modes above 0x2000, but not for RAM or Register operand addresses, as SAD does not scan the code in the same order as it is executed.

Additional options

B	Only with N. Look for a byte symbol
D <F>	Divisor (floating point)
F <D> <D>?	Bit field start→end. End is optional for single bit
G <X>	Range address (only with N) (defaults to current comment address)
I	Only with N. Look for an integer (word) symbol
O <X>	Offset Address
W	Only with with N. Look for RITE symbol. not required if using operands (Ax) as SAD knows already.
X <D>	Print radix (2,10,16)

An [item] sequence cannot have duplicate option letters. To print multiple offsets for the same address for example, this is two items [A1 N O 1234] [A1 N O 4567] to print two different offset symbols for operand 1

** My notes
- coding – If we can stick fairly closely to command type syntax, can reuse my ‘additional’ structures in SAD code to cover comments as well, which would be neat, and use same code to print and parse them (= less errors).

Not sure about subfields vs. ‘E’ (encoded) in arguments yet either, as currently ‘E’ displays final answer (=calculated address) could keep this I guess to add to arguments and even comments if required.