Note: this document is a work in progress. I intend to reorganize it.

* General code conventions:
  + Division and multiplication operators take precedence over addition and subtraction operators.
  + Lines including white space cannot be longer than 120 characters.
  + Numeric literals can only contain the decimal digits (0-9).
  + Variables are implicitly declared by initializing them. Uninitialized variables are invalid.
  + White space made up of tabs, spaces and empty lines is ignored.
* Predefined environment variables:
  + Bell$ (the control character used to generate a beep)
  + ColorCard (3 = VGA, 4 = EGA, 6 = Tandy, 7 = CGA/PCjr, 10 = Hercules)
  + CopyError (Copy statement error code (0 = success, nonzero = failure))
  + CriticalError (Unknown purpose.)
  + CurrentPath$ (current path)
  + CurrentDrive$ (current drive)
  + DosVersion (MS-DOS version)
  + Floppies (The number of floppy drives plus one with bitmask 0xC0 at address: 0x0040:0x0010)
  + JoySticks (The game port flag with bitmask 0x10 at address: 0x0040:0x0011)
  + MachineType (The machine identification byte at address 0xF000:0xFFFE)
  + MemorySize (base memory in kilobytes)
  + TandyID (0 = no Tandy, 1 = Tandy)
  + VideoMode (The video mode byte at address 0x0040:0x0049)
  + VolName$ (drive volume name)
  + Note: the script can override these values at any time.
* Data types:
  + 32 bit signed long integers.
  + Strings with a maximum length of 118 characters.
* Variable naming rules:
  + Names must start with a letter ("A-Z", "a-z") or underscore ("\_").
  + Numeric characters ("0-9") must be prececed by a letter or underscore.
  + Names are case sensitive.
  + Names (including any sigils) cannot be longer than 19 characters.
* Special characters:
  + Addition is done using the plus sign ("+").
  + Equality is checked using two equal signs ("==") and ("!=") for inequality.
  + Conditions must be enclosed in parentheses ("(" and ")").
  + Division is done using the forward slash ("/")
  + Greater than and less than signs are used for comparisson (">" and "<").
  + Keywords are separated from other code elements using spaces (" ").
  + Labels start with a colon (":").
  + Lines of code are separated using linebreaks (0x13 character (carriage return)).
  + Multiplication is done using the asterisk sign ("\*").
  + Parameters are delimited using a comma (",").
  + String concatenation is done using the plus sign ("+").
  + String literals are enclosed in quotes (""").
  + String variable names must have a "$" sigil.
  + Subtraction is done using the minus sign ("-").
  + The default operator precedence can be overruled using parentheses ("(" and ")").
  + Variable assignment is done using the equal sign ("=").
* Character functions:
  + *result* (ASCII) = chr\_to\_int *character$*
  + *result* (string) = int\_to\_chr$ *ascii*
* Dialogs:
  + Coordinates have the following range: x: 0-79, y: 0-24.
  + atdialog y*,x,input$, (returns the input and cannot be a literal),length (0 = return a keystroke),prompt$*
  + atsay y*,x,message$*
  + colorf y,x (unknown purpose)
  + dialog *input$, (returns the input and cannot be a literal),length (0 = return a keystroke),prompt$*
  + say message$
  + Predefined keynames:
    - CR Enter.
    - CURDN Down arrow.
    - CURLF Left arrow.
    - CURRT Right arrow.
    - CURUP Up arrow.
    - DEL Delete.
    - END End.
    - ESC Escape.
    - HOME Home.
    - INS Insert.
    - PGDN Page down.
    - PGUP Page up.
* Display:
  + color foreground (0-15), background/blink (0-7 just background, 8-15 background and enables the specified foreground color to blink)
  + cls
* Quitting:
  + exit *exitcode*
* Control flow:
  + gosub *label*
  + goto *label*
  + return
* Conditional code:
  + Any expression resulting in a non-zero value is treated as "true".
  + Any expression resulting in zero is treated as "false".
  + if (condition) *code*
  + if (*condition*) *code* [else] *code* endif
* File I/O:
  + copy *source*$, *destination$*
  + *handle =* fopen *path$*
  + fclose *handle*
  + fsay *handle, output$*
  + *result* (0 = success, 2 = failure) = delete file$
* Disk functions:
  + *result* (0 = failure, 1 = success)
  + *result* (0 = failure, 1 = success) *=* chdrive *drive$*
  + *result* (0 = failure, 1 = success) *=* drivevalid *drive$*
  + result (0 = does not exist, 1 = file exists, 2 = directory exists) = exists *path$*
  + *result* (0 = failure, 1 = success) *=* makedir *drive$*
  + *result* (bytes free) *=* diskspace *drive$*
  + *volume =* getVolume$
* Arcive functions:
  + unpack *source$, destination$, quietflag$ ("q" or "Q" = quiet)*
* Reserved keywords:
  + then (unknown purpose)