# Axe Parser v0.2.0

## Command List Index



### System

| Command | Description |
|---|---|
| – | Spaces are ignored in most situations. They mainly just help for code organization and readability. |
| : | The colon and enter key end a line of code. |
| . | The period is a single line comment. Whatever follows will be ignored until the next newline. Must be the first character on the line. |
| DiagnosticOn | Turns on the run indicator (marching ants). Program will display "done" after finishing. |
| DiagnosticOff | Turns off the run indicator. Program will not display "done" after finishing. |
| Full | Full speed mode is activated if supported, making it 3 times faster on newer calculators. Returns 0 if not supported. |
| Normal | Full speed mode is deactivated. |
| Pause EXP | Pause for the given amount of time in milliseconds. |
| getKey | Returns the last key pressed or zero if no keys are pressed. Its just like the BASIC getkey, but with different codes. |
| getKey(KEY) | Returns 1 if the key is held down this instant and 0 otherwise. The key code must be a single constant. |
| getKey(0) | Returns a non-zero number if any key is held down and 0 otherwise. |
| SinReg WAVE,TIME | Sound is played out of the link port. Wave must be between 1-255 inversely proportional to frequency. Time is in the order of microseconds. |
| Asm(HEX) | Native assembly code written in hexadecimal is inserted at the current |

| | |
|---|---|
| | position. |

## Screen and Buffer

| Command | Description |
|---|---|
| ClrHome | Erases the screen and text shadow and moves the cursor to the upper left corner. |
| ClrDraw | Erases the buffer. |
| DispGraph | Draws the buffer on the screen. |
| DispGraph$^{r}$ | Draws the buffer on the screen over an alternating crosshatch of the back buffer. Used for grayscale. |
| StoreGDB | Copies the screen to the buffer. |
| StorePic | Copies the buffer to the back-buffer. |
| RecallPic | Copies the back-buffer to the buffer. |
| DrawInv | The colors on the buffer are inverted. |
| Horizontal +<br>Horizontal - | The buffer is shifted right (+) or left (-) by 1 pixel. White pixels are shifted in. |
| Vertical +<br>Vertical - | The buffer is shifted down (+) or up (-) by 1 pixel. New pixels are not shifted in, that row remains the same. |
| Shade(EXP) | Sets the contrast. 0 is lightest, 63 is darkest. |

## Control Blocks

| Command | Description |
|---|---|
| If EXP<br>code1<br>End | If the expression is true, code1 will be executed. |
| If EXP<br>code1<br>Else<br>code2<br>End | If the expression is true, then only code1 is executed. Otherwise, only code 2 is executed. |
| !If EXP<br>code1<br>End | If the expression is false, code1 will be executed. |
| !If EXP<br>code1<br>Else<br>code2<br>End | If the expression is false, then only code1 is executed. Otherwise, only code 2 is executed. |

| | |
|---|---|
| While EXP<br>code1<br>End | The expression is checked first. If its true, code1 will be executed over and over until its false. |
| Repeat EXP<br>code1<br>End | The expression is checked first. If its false, code1 will be executed over and over until its true. |
| For(VAR,EXP1,EXP2)<br>code1<br>End | The variable is initialized with expression1. Until the variable is greater than expression2, code1 is executed and the variable is incremented by 1. |

## Labels and Subroutines

| Command | Description |
|---|---|
| Lbl LBL | Creates a label at the current position. |
| Goto LBL | Jumps to the label. |
| DelVar LBL | Frees the label name from memory. The name can then be reused somewhere later in the code. |
| Sub(LBL) | Calls the subroutine. All subroutines should end with a Return. |
| Return | Returns from a subroutine. If not in a subroutine, the program will end. |
| ReturnIf EXP | Returns only if the expression is true. |
| Return!If EXP | Returns only if the expression is false. |

## Basic Math

| Command | Description |
|---|---|
| VAR | Returns the variable. Uppercase A through Z are variables. |
| EXP→VAR | Stores the expression into the variable. |
| 'CHAR' | Converts an ASCII constant into an integer. |
| -EXP | Returns the negative of the expression. That's a negative sign, not a minus sign! |
| EXP1+EXP2<br>EXP1-EXP2 | Expression2 is added to or subtracted from expression1. |
| EXP1*EXP2<br>EXP1/EXP2<br>EXP1^EXP2 | Expression1 is multiplied, divided, or the modulus of expression2. |
| $EXP^2$ | The expression is multiplied by itself. |
| EXP1=EXP2<br>EXP1≠EXP2 | Returns 1 if the statement is true or 0 if its false. This is an unsigned comparison. |

| | |
|---|---|
| EXP1<EXP2<br>EXP1≤EXP2<br>EXP1>EXP2<br>EXP1≥EXP2 | |
| EXP1 or EXP2<br>EXP1 and EXP2<br>EXP1 xor EXP2 | Returns the bitwise operation of the expressions. You need parenthesis to use this with truths. |
| abs(EXP) | Returns the absolute value of the expression. |
| √(EXP) | Returns the square root of the expression. |
| sin(EXP) | Returns the sine of the expression. One Period is 255 and the value returned ranges from -127 to 127. |
| cos(EXP) | Returns the cosine of the expression. One Period is 255 and the value returned ranges from -127 to 127. |
| rand | Returns a random 16 bit number. |

## Advanced Math

| Command | Description |
|---|---|
| EHEX | Converts a 4 digit hexadecimal number into an integer. That E is the scientific notation E. |
| EXP1<<EXP2<br>EXP1≤≤EXP2<br>EXP1>>EXP2<br>EXP1≥≥EXP2 | Signed comparisons for numbers that aren't always positive. Returns 1 if the statement is true or 0 if its false. |

## Drawing

| Command | Description |
|---|---|
| Disp EXP | The string that is pointed to is displayed at the cursor position. The cursor moves with the string. If it reaches the end of the screen, it will loop around to the next line. |
| Disp EXP▸Dec | The expression is displayed as a decimal at the cursor position. The cursor is then advanced 5 spaces. |
| Disp EXP▸Frac | The ASCII character of the expression is displayed at the cursor position. The cursor is advanced 1 space. A new line is added if it hits the edge. |
| Disp "" | The string is displayed at the cursor position. |
| Disp i | The cursor moves to the next line down. This is the imaginary, not lowercase 'i'. |

| | |
|---|---|
| Output(X) | The cursor moves to the cursor position (X/256,X%256). |
| Output(X,Y) | The cursor moves to the cursor position (X,Y). |
| Output(X,Y, | The cursor moves to the cursor position (X,Y) and whatever follows is displayed at that position. |
| Pxl-On(X,Y) | A pixel becomes black on the buffer at (X,Y). |
| Pxl-Off(X,Y) | A pixel becomes white on the buffer at (X,Y). |
| Pxl-Change(X,Y) | A pixel will change color on the buffer at (X,Y). |
| pxl-Test(X,Y) | Returns 1 if pixel is black and 0 if pixel is white on the buffer at (X,Y). |
| Pt-On(X,Y,PIC) | The 8x8 sprite that expression points to is drawn to the buffer at (X,Y). Does not clear the area behind it. |
| Pt-Off(X,Y,PIC) | The 8x8 sprite that expression points to is drawn to the buffer at (X,Y) but clears the area behind it first. |
| Pt-Change(X,Y,PIC) | The 8x8 sprite that expression points to inverts its pixels on the buffer at (X,Y). |
| Text(X,Y,EXP) | The text pointed to is drawn at (X,Y). See below for drawing details. |
| Fix CODE | Changes how text is drawn. Code must be a constant.<br>0 = Small size. Calculator should exit in this mode if changed!<br>1 = Large size.<br>2 = Normal font. Calculator should exit in this mode if changed!<br>3 = Inverted font.<br>4 = Draw to screen. Calculator should exit in this mode if changed!<br>5 = Draw to buffer. |

## Data and Storage

| Command | Description |
|---|---|
| "" | Adds the string to program memory, but without the ending character. |
| [HEX] | Adds the hex to the program memory. |
| [PICVAR] | Absorbs the 96x63 picture from RAM into the program (756 bytes). Only the source needs the pic, not the executable. |
| ΔList(NUM,...) | Adds the byte to program memory. Numbers ending with $^{r}$ are added as 2 byte numbers. |
| det(SIZE) | Adds Size bytes of zeros to program memory. |
| DATA→NAME | Saves the data's pointer to a static variable. Also terminates current string if applicable. |
| NAME | Returns a pointer to the start of the data. |
| $L_1$ | Returns a pointer to some free memory. |

| | |
|---|---|
| $L_2$<br>$L_3$<br>$L_4$<br>$L_5$<br>$L_6$ | $L_1$ = 714 bytes (saveSScreen+54) Volatility: **LOW**<br>$L_2$ = 531 bytes (statVars) Volatility: **LOW**<br>$L_3$ = 768 bytes (appBackUpScreen) Volatility: **MED** (Saving to back-buffer will corrupt)<br>$L_4$ = 323 bytes (tempSwapArea) Volatility: **MED** (Corrupt when archiving/unarchiving in program)<br>$L_5$ = 128 bytes (textShadow) Volatility: **MED** ("Disp","Output", and "ClrHome" will corrupt)<br>$L_6$ = 768 bytes (plotSScreen) Volatility: **HIGH** (Any buffer drawing will corrupt) |
| {EXP}$^r$ | Returns the 2 byte data the expression points to. |
| {EXP} | Returns the single byte the expression points to. It will be in the range 0 to 255. |
| int(EXP) | Returns the single byte the expression points to. It will be in the range -128 to 127. |
| EXP1→{EXP2}$^r$ | The full 2 bytes of Expression1 is stored to where Expression2 points. |
| EXP1→{EXP2} | The single byte of Expression1 is stored to where Expression2 points. |
| Fill(PTR1,SIZE) | The byte already at Ptr1 is copied to all the bytes after it until Size bytes have been filled with that value. Zero is not a valid Size. |
| conj(PTR1,PTR2,SIZE) | Size bytes starting from Ptr1 are copied to Size bytes starting at Ptr2. Zero is not a valid Size. |
| expr(PTR1,PTR2,SIZE) | Size bytes starting from Ptr1 are exchanged with Size bytes starting at Ptr2. Zero is not a valid Size. |

## File Management

| Command | Description |
|---|---|
| GetCalc(PTR) | Finds the object who's name is pointed to and returns a pointer to the start of its data, or zero if it was archived or not found. |
| GetCalc(PTR,SIZE) | Creates an application variable in RAM, with the name pointed to, and makes it Size bytes. Returns a pointer to the start of data, or zero if there was not enough RAM. Overwrites existing appvar, even if it was in archive. |
| Unarchive PTR | Tries to unarchive the object who's name is pointed to. Returns 1 if it could unarchive and 0 otherwise. Gives a memory error if not enough RAM. |
| Archive PTR | Tries to archive the object who's name is pointed to. Returns 1 if it could archive and 0 otherwise. Gives a memory error if not enough Flash Memory. |

New In This Version

Changed From Last Version

Existing Command

Documentation for Axe Parser