Appendix 7 KERNAL Functions

System Interrupt Table for the VC-3 system, running on the SD-8516.

Introduction

The following is a list and short explanation of the System Interrupt Table; the user-facing part of the KERNAL that you can interact with.

For more information and examples, please see the SD-8516 Programmer's Reference Guide. Over time, more information will be added here but it is intended solely as a catalog of the system calls.

Example Code

```
    LDAL #'A'       ; AL has the variable name
    LDB #1234       ; B has the value to set
    LDAH $11        ; VAR_SET
    INT $05

    LDELM @string   ; address of string containing a number
    LDAH $62        ; IO_GETNUM
    INT $05
    ; B now contains the number from the string
```

INT 05h - CAM/IL

```
;
================================================================================
; INT 05h - "TinyBASIC" Interpretive Language
; Complete implementation of a TinyBASIC IL + Stellar BASIC extensions
;
================================================================================
; Function dispatch via AH register:
;
; === EXECUTION CONTROL (AH=$00-$0F) ===
;   AH=$00: RUN_PROGRAM    - Start execution from line number or beginning
;   AH=$01: STOP_PROGRAM   - Stop execution
;   AH=$02: CONTINUE       - Continue from current position
;   AH=$03: EXEC_IL_STEP   - Execute one IL instruction
;   AH=$04: EXEC_IL_CONT   - Execute IL until completion
;   AH=$05: GET_STATUS     - Get interpreter status
;   AH=$06: SET_STATUS     - Set interpreter status flags
;   AH=$07: RESET_INTERP   - Reset interpreter state
```

```
;
; === VARIABLE MANAGEMENT (AH=$10-$1F) ===
;   AH=$10: VAR_GET        - Get variable value by letter
;   AH=$11: VAR_SET        - Set variable value by letter
;   AH=$12: VAR_CLEAR_ALL  - Clear all variables to 0
;   AH=$13: VAR_GET_ADDR   - Get address of variable storage
;
; === NAMED VARIABLE SERVICES (AH=$14-$1F) ===
;   AH=$14: NVAR_GET    - Find variable by name
;   AH=$15: NVAR_SET    - Insert or update variable
;   AH=$16: NVAR_DELETE - Remove variable
;   AH=$17: NVAR_CLEAR  - Clear all named variables
;
; === EXPRESSION STACK (AH=$20-$2F) ===
;   AH=$20: EXPR_PUSH      - Push value to expression stack
;   AH=$21: EXPR_POP       - Pop value from expression stack
;   AH=$22: EXPR_PEEK      - Peek at top of stack
;   AH=$23: EXPR_CLEAR     - Clear expression stack
;   AH=$24: EXPR_DEPTH     - Get current stack depth
;
; === STACK MANIPULATION (AH=$28-$2F) ===
;   AH=$25: EXPR_DUP        - Duplicate top of stack
;   AH=$26: EXPR_SWAP       - Swap top two values
;   AH=$27: EXPR_DROP       - Drop top value
;   AH=$28: EXPR_OVER       - Copy second value to top
;   AH=$29: EXPR_ROT        - Rotate top 3 values
;
; === PROGRAM LINE MANAGEMENT (AH=$30-$3F) ===
; $30-$37: Search/Navigate
;   $30: LINE_FIND          - Find line by number (forward)
;   $31: LINE_NEXT          - Get next line
;   $32: LINE_FIND_REVERSE  - Find line by number (backward)
;   $33: LINE_PREV          - Get previous line
; $36-$39: Modify
;   $36: LINE_INSERT        - Insert or replace line
;   $37: LINE_DELETE        - Delete line by number
;   $38: LINE_REMOVE_SPACE  - Add space for new program line
;   $39: LINE_MAKE_SPACE    - remove unused line space
; $3A-$3F: Utility
;   $3A: LINE_FIRST         - Get first line
;   $3B: LINE_CLEAR_ALL     - Clear entire program (NEW)
;   $3C: LINE_COUNT         - Count total lines
;   $3F: PROGRAM_END        - ELM points to end of program
;
; === GOSUB/RETURN STACK (AH=$40-$4F) ===
;   AH=$40: GOSUB_PUSH     - Push return address for GOSUB
;   AH=$41: GOSUB_POP      - Pop return address for RETURN
;   AH=$42: GOSUB_CLEAR    - Clear GOSUB stack
;
; === FOR/NEXT STACK (AH=$50-$5F) ===
;   AH=$50: FOR_PUSH       - Push FOR loop context
```

```
;   AH=$51: FOR_POP        - Pop FOR loop context
;   AH=$52: FOR_PEEK       - Peek at current FOR loop
;   AH=$53: FOR_CLEAR      - Clear FOR stack
;   AH=$54: FOR_FIND_VAR   - Find FOR loop by variable letter
;
; === INPUT/OUTPUT HELPERS (AH=$60-$6F) ===
;   AH=$60: IO_GETCHAR     - Get next character from input buffer
;   AH=$61: IO_PUTCHAR     - Output character to screen
;   AH=$62: IO_GETNUM      - Parse number from input
;   AH=$63: IO_PUTNUM      - Output number to screen
;   AH=$64: IO_NEWLINE     - Output CR/LF
;   AH=$65: IO_GETLINE     - Get line of input from use/publir
;   AH=$66: IO_PRINT_STR   - Print string at pointer
;   AH=$67: IO_PROCESS_STRING - Process a string in quotes for printing.
;   AH=$68: IO_INPUT - for the INPUT command. Simple string input.
;
; === STRING/TOKEN HELPERS (AH=$70-$8F) ===
;   AH=$70: STR_COMPARE    - Compare strings for keyword matching
;   AH=$71: STR_TO_UPPER   - Convert string to uppercase
;   AH=$72: STR_SKIP_SPACE - Skip whitespace in ELM string
;   AH=$73: STR_IS_DIGIT   - Check if character is digit
;   AH=$74: STR_IS_ALPHA   - Check if character is letter
;   AH=$75: STR_LENGTH     - Get string length
;   AH=$76: STR_COPY       - Copy string from src to dst
;   AH=$77: STR_CONCAT     - Concatenate two strings
;   AH=$78: STR_LEFT       - Extract left N characters (LEFT$)
;   AH=$79: STR_RIGHT      - Extract right N characters (RIGHT$)
;   AH=$7A: STR_MID        - Extract middle substring (MID$)
;   AH=$7B: STR_CHR        - Convert byte to character (CHR$)
;   AH=$7C: STR_ASC        - Convert character to byte (ASC)
;   AH=$7D: STR_VAL        - Convert string to number (VAL)
;   AH=$7E: STR_STR        - Convert number to string (STR$)
;   AH=$7F: STR_FIND       - Find substring within string (INSTR)
;   AH=$80: STR_LTRIM      - Trim leading whitespace
;   AH=$81: STR_RTRIM      - Trim trailing whitespace
;   AH=$82: STR_TRIM       - Trim both leading and trailing whitespace
;   AH=$83: STR_REVERSE    - Reverse a string
;   AH=$84: STR_REPEAT     - Repeat character N times (STRING$)
;   AH=$85: STR_SPLIT      - Split string at delimiter
;   AH=$86: STR_REPLACE    - Replace substring with another
;   AH=$87-$8F: (Reserved for future string operations)
;
; === ARITHMETIC/LOGIC HELPERS (AH=$90-$9F) ===
;   AH=$90: MATH_ADD       - Pop two values, push sum
;   AH=$91: MATH_SUB       - Pop two values, push difference
;   AH=$92: MATH_MUL       - Pop two values, push product
;   AH=$93: MATH_DIV       - Pop two values, push quotient
;   AH=$94: MATH_NEG       - Negate top of stack
;   AH=$95: MATH_RAND      - Generate random integer
;   AH=$96: MATH_RND       - Generate random number between 0 and 1
;
```

```
; === ADVANCED MATH (AH=$96-$9F) ===
;   AH=$9A: MATH_MOD        - Modulo (remainder)
;   AH=$9B: MATH_ABS        - Absolute value
;   AH=$9C: MATH_MIN        - Minimum of two values
;   AH=$9D: MATH_MAX        - Maximum of two values
;   AH=$9E: MATH_SGN        - Sign (-1, 0, or 1)
;
; === COMPARISON HELPERS (AH=$A0-$AF) ===
;   AH=$A0: CMP_EQ         - Compare equal
;   AH=$A1: CMP_NE         - Compare not equal
;   AH=$A2: CMP_LT         - Compare less than
;   AH=$A3: CMP_GT         - Compare greater than
;   AH=$A4: CMP_LE         - Compare less than or equal
;   AH=$A5: CMP_GE         - Compare greater than or equal
;   AH=$A6-$AF: (Reserved)
;
;
================================================================================

;
================================================================================
; INT 05h - PALO ALTO TINY BASIC SYSTEM
; Memory map and constants
;
================================================================================

.equ PATB_TBUF          $01DB00   ; tokenizer buffer (as listed in memory
map) CHECKME: not really being used?
.equ PATB_VARIABLES     $01EE00   ; 26 vars × 2 bytes = 52 bytes ($01EE00-
$01EE33)
.equ PATB_EXPR_STACK    $01EE34   ; Expression stack, 64 bytes ($01EE34-
$01EE73)
.equ PATB_EXPR_SP       $01EE74   ; Expression stack pointer (2 bytes)
.equ PATB_GOSUB_STACK   $01EE76   ; GOSUB stack, 48 bytes (16 levels × 3
bytes)
.equ PATB_GOSUB_SP      $01EEA6   ; GOSUB stack pointer (2 bytes)
.equ PATB_FOR_STACK     $01EEA8   ; FOR stack, 40 bytes (8 levels × 5 bytes)
.equ PATB_FOR_SP        $01EED0   ; FOR stack pointer (2 bytes)
.equ PATB_GOTO_TARGET   $01EED2   ; 2 bytes: 0 = no goto, nonzero = target
line#
.equ PATB_STATUS        $01EED4   ; 1 byte
                                  ; Bit 0: Running (1) / Stopped (0)
                                  ; Bit 1: Error flag
                                  ; Bit 2: Break flag
.equ PATB_RANDSEED      $01EED5   ; 2 bytes
.equ PATB_PROGRAM_PTR   $01EED7   ; 3 bytes
.equ PATB_IL_PTR        $01EEDA   ; 3 bytes
.equ PATB_INPUT_PTR     $01EEDD   ; 3 bytes
.equ PATB_BANK          $01EEE0   ; 1 byte
.equ PATB_DIM_NAMEPTR   $01EEE1   ; 3 bytes - saved var name pointer for DIM
loop
```

```
.equ PATB_VARTOP        $01EEE4   ; 3 bytes - bottom of variable storage
(grows down)
.equ PATB_DATA_PTR      $01EEE7   ; 3 bytes - for DATA and READ statements
.equ PATB_MS_PTR        $01EEEA   ; 3 bytes - multiline statement pointer
.equ PATB_LINE_NO       $01EEED   ; 2 bytes - current line no.

; RESERVED: $01EEED-$01EEFF = 17 bytes free

; Refer to memorymap.sda if these change
.equ PATB_PARSEBUF      $0000D0   ; 16 bytes
.equ PATB_LET_NAMEBUF   $0000E0   ; 32 bytes (until $0000FF) -- just before
BASIC space.
.equ PATB_PROGRAM_BASE  $000100   ; Start of program storage (Bank 3, offset
$100)
.equ PATB_PROGRAM_END   $00FFFF   ; End (almost 64KB for programs)
```

INT 10h - Terminal Services

```
;
================================================================================
; INT 10h - KEYBOARD / TERMINAL SERVICES
; PC BIOS-style keyboard interrupt handler
;
================================================================================
; Function dispatch via AH register:
;   AH=00h: Read character (non-blocking)
;   AH=01h: Check if key available (in keyboar buffer)
;   AH=02h: Read character (blocking)
;   AH=03h: Get keyboard flags/shift state
;   AH=04h: Clear keyboard buffer
;   AH=05h: Push character into buffer

;   AH=10h: Clear screen (blank characters only)
;   AH=11h: Write character at cursor location
;   AH=12h: Read character at cursor location
;   AH=13h: Set color data at cursor location
;   Ah=14h: Get color data at cursor location
;   AH=15h: Set cursor position
;   AH=16h: Get cursor position
;   AH=17h: Print character at cursor (teletype)
;   AH=18h: Print string at cursor
;   Ah=19h: Read string between X,Y and cursor
;   AH=1Ah: Carriage Return
;   AH=1Bh: Issue Linefeed
;   AH=1Ch: Scroll screen up (linefeed_push forced linefeed)

;   AH=20h: Cursor Left (move cursor left one column)
;   AH=21h: Cursor Right (move cursor right one column)
;   AH=22h: Cursor Up (move cursor up one row)
```

```
;   AH=23h: Cursor Down (move cursor down one row)
;   AH=24h: Bell (audible beep - optional)
;   AH=25h: Backspace (move left and erase character)
;   AH=26h: Delete (pull line left from cursor to end of line)
;   AH=27h: Tab (move to next tab stop)
;   AH=28h: Home (move cursor to 0,0)
;   AH=29h: Advance Cursor with Line Link Hook (links to term.sda)
;   links to term.sda:
;   AH=$2A: int10_cursor_blink -- Blink cursor (main loop)
;   AH=$2B: int10_cursor_on -- Turn cursor on
;   AH=$2C: int10_cursor_off -- Turn cursor off
;   AH=$2D: int10_cursor_toggle -- Toggle cursor
;   AH=$2E: int10_set_cursor_colors -- Set char/cursor colors

;   AL=30h: Clear to End of Line (erase from cursor to end of line)
;   AL=31h: Clear to End of Screen (erase from cursor to end of screen)
;   AL=32h: Insert Line (insert blank line at cursor row)
;   AL=33h: Delete Line (delete line at cursor row)
;   AL=34h: link line
;   AL=35h:
;   AL=36h:
;   AL=37h:
;   AH=38h: Scroll Line Links

;   AH=40h: Set video mode
;   AH=41h: Get current video mode
;   AH=42h: Get MAX ROWS (mode 0:23, 1:25, 2:25, 3: 30)
;   AH=43h: Get MAX COLS (mode 0:22, 1:40, 2:80, 3: 120)
;   AH=44h: Get pointer to VIDEO_TEXT_BASE for current mode (or start of
framebuffer for graphics modes)
;   AH=45h: Get pointer to VIDEO_COLOR_BASE for current mode (or palette
space for graphics modes)
;   AH=46h: is text mode?
;   AH=47h: get char rom base
```

INT 11h - Sound Services

Please see the interrupt table and function discussion in Appendix 5 Sound System of the SD-8516 Programmer's Reference Guide.

INT 12h - String Services

- INT 12h String Services

INT 13h - Math Services

```
;
;===============================================================================
; INT 13h - MATH SERVICES
; Stellar BIOS Math operations interrupt handler
;
;===============================================================================
; Function dispatch via AH register:
;   AH=00h: Random number (16-bit)
;   AH=01h: Random number in range
;   AH=02h: Seed random number generator
;   AH=03h: Square root (integer)
;   AH=04h: Power (base^exponent)
;   AH=05h: Sine (8-bit fixed point)
;   AH=06h: Cosine (8-bit fixed point)
;   AH=07h: Tangent (8-bit fixed point)
;   AH=08h: Arctangent2 (for angles)
;   AH=09h: Absolute value
;   AH=0Ah: Min of two values
;   AH=0Bh: Max of two values
;   AH=0Ch: Clamp value to range
;
;
;===============================================================================
```

INT 15h - Filesystem Services

```
;
;===============================================================================
; INT 15h - FILE SERVICES
; Stream-oriented file I/O for the SD-8516 / VC-3
;
;===============================================================================
;
; Provides file and directory operations for programs.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
; NOTE: INT $15 uses a JZ-style calltable.
;       This means you cannot CALL into these functions by name.
;       You must use INT 0x15 to call them (even from the kernal).
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;
; Function dispatch:
;
; Group A: File Handle Operations ($00–$0F)
```

```
;   AH=$00: FOPEN    - Open a file (returns handle)
;   AH=$01: FCLOSE   - Close a file handle
;   AH=$02: FREAD    - Read bytes from a file
;   AH=$04: FWRITE   - Write bytes to a file
;
; Group B: Directory Listing ($10–$1F)
;   AH=$12: FLIST    - List directory contents to buffer
;
; Group C: Directory Navigation ($20–$2F)
;   AH=$20: FGETCWD  - Get current working directory
;   AH=$21: FCHDIR   - Change directory
;   AH=$22: FMKDIR   - Create directory
;   AH=$23: FRMDIR   - Remove directory
;
;
================================================================================

; --- File Command Block (FCB) ---
; A shared memory region used to pass parameters to/from JavaScript.
; Located in bank 0 scratch space. 16 bytes.
;
.equ FS_CMD_BLOCK      $00FD00     ; Base address of command block

.equ FS_CMD            $00FD00     ; +0: Command code (1 byte)
.equ FS_HANDLE         $00FD01     ; +1: File handle (1 byte, 0-7)
.equ FS_MODE           $00FD02     ; +2: Mode (1 byte, for fopen: 0=read,
1=write)
.equ FS_STATUS         $00FD03     ; +3: Result status (1 byte)
                                   ;     $00=ok, $01=error, $02=eof
.equ FS_COUNT          $00FD04     ; +4: Byte count (2 bytes,
requested/returned)
.equ FS_ADDR           $00FD06     ; +6: 24-bit address (buffer or filename
ptr)
.equ FS_DONE           $00FD09     ; +9: Completion flag (1 byte, 0=pending,
1=done)
.equ FS_EXTRA          $00FD0A     ; +10: Reserved (3 bytes) and alias for:
.equ FS_ADDR2          $00FD0A;    ;  +A: second 24-bit address (3 bytes)

; --- Constants ---
.equ FS_MAX_HANDLES    8           ; Maximum simultaneous open files

.equ FS_MODE_READ      0
.equ FS_MODE_WRITE     1

.equ FS_OK             0
.equ FS_ERR            1
.equ FS_EOF            2
```

INT 18h - Graphics Services

```
;
================================================================================
; INT 18h - GRAPHICS SERVICES
; Stellar BIOS Graphics operations interrupt handler
;
================================================================================
; Function dispatch via AH register:
;   AH=00h: reserved (for set graphics mode see int 10h ah=40h)
;   AH=01h: Plot MODE 3 pixel (4bpp)
;   AH=02h: Plot MODE 3b pixel (8bpp)
;   AH=11h: Get MODE 3 pixel (4bpp)
;   AH=12h: Get MODE 3b pixel (8bpp)

;   AH=20h: Clear screen (fill with color)
;   AH=21h: Draw line
;   AH=22h: Draw rectangle (outline)
;   AH=23h: Fill rectangle
;   AH=24h: Draw circle (outline)
;   AH=25h: Fill circle
;   AH=26h: Draw ellipse
;   AH=30h: Blit sprite/image

;   AH=40h: Set color mode for video mode
;   AH=41h: Get current color mode
;   AH=42h: Set palette entry (AL=0, use 0-15. Anything else, use 0-255)
;   AH=43h: Get palette entry (AL=0, use 0-15. Anything else, use 0-255)
;   AH=44h: Load palette (AL=0, use 16color, anything else, all 256.
ELM=source for palette data)
;   AH=45h: Set draw color (deprecated for now)
;   AH=46h: Flood fill (PAINT) (deprecated for now)

;   AH=50h: Scroll region
;   AH=51h: Copy region
;
================================================================================
; Graphics Mode 3: 320x200x16 or 320x200x256
;   Framebuffer: Bank 2, $020000-$027CFF (32,000 bytes)
;   Framebuffer: Bank 2, $020000-$02F9FF (64,000 bytes) (3b)
;   Palettes: Bank 2, $02FA00-$02FCFF (768 bytes, RGB triplets)
;   Each pixel = 4 bits (colormode palettes)
;   Each pixel = 1 byte (palette index 0-255) (3b)
;   Each palette entry = 3 bytes (R, G, B, 0-255 each)
;
================================================================================
```