– CHAPTER 8 –

LINE-A

Overview

The **Line-A** portion of the operating system is so named because it uses a special exception vector of 680x0 processors triggered when the first nibble of the a command word is $A. On Atari systems this vector is routed to the operating system ROMs and provides a low-level yet high-speed graphics interface.

The **Line-A** system is included in this document for completeness only. It is recommended that its use be avoided and that the counterpart **VDI** calls be used instead. Atari has not guaranteed that it will maintain **Line-A** compatibility in future systems. Its functionality has already been limited as video capabilities have advanced beyond its design.

The Line-A Variable Table

The **Line-A** opcode $A000 will return a pointer to an internal variable table in D0 and A0. This table is used by the **Line-A** functions as a parameter passing mechanism as opposed to using the stack or internal registers.

Members of the **Line-A** variable table are accessed via offsets from the base address. The function, location, and size of documented variables are as follows:

Name	Offset	Size	Contents
RESERVED	-910	**LONG**	Reserved for future use.
CUR_FONT	-906	**LONG**	Pointer to the current font header.
RESERVED	-902	92 **BYTE**s	Reserved for future use.
M_POS_HX	-856	**WORD**	X Offset into the mouse form of the ‘hot spot’.
M_POS_HY	-854	**WORD**	Y Offset into the mouse form of the ‘hot spot’.
M_PLANES	-852	**WORD**	Writing mode for the mouse pointer (1 = **VDI** Mode, -1 = XOR Mode). Defaults to **VDI** mode.
M_CDB_BG	-850	**WORD**	Mouse pointer background color.
M_CDB_FG	-848	**WORD**	Mouse pointer foreground color.
MASK_FORM	-846	32 **WORD**s	Image and Mask for the mouse pointer. Data is stored in the following format: Line 0 Mask Line 0 Image Line 1 Mask Line 1 Image etc.
INQ_TAB	-782	46 **WORD**s	This area contains 45 **WORD**s of information returned from a **vq_extnd()** of the physical screen workstation plus one extra reserved **WORD**.
DEV_TAB	-692	46 **WORD**s	This area contains the first 45 **WORD**s of information returned from a **v_opnwk()** of the physical screen workstation plus one extra reserved **WORD**.
GCURX	-602	**WORD**	Current mouse pointer X position.
GCURY	-600	**WORD**	Current mouse pointer Y position.

M_HID_CT	-598	**WORD**	Current mouse 'hide' count (number of times mouse has been hidden, 0 = visible).
MOUSE_BT	-596	**WORD**	Bitmap of the current mouse button status.
REQ_COL	-594	48 **WORD**s	Contains 48 **WORD**s of RGB data for the first 16 **VDI** color registers as would be returned by **vq_color()**.
SIZ_TAB	-498	15 **WORD**s	This table contains the final 12 **WORD**s of information returned from a **v_opnwk()** of the physical screen workstation plus 3 reserved **WORD**s.
RESERVED	-468	**WORD**	Reserved for future use.
RESERVED	-466	**WORD**	Reserved for future use.
CUR_WORK	-464	**LONG**	Pointer to the current **VDI** workstation attribute table.
DEF_FONT	-460	**LONG**	Pointer to the default font header.
FONT_RING	-456	4 **LONG**s	This area contains three pointers and a **NULL**. The first two pointers point to linked lists of system font headers. The third pointer points to the linked list of **GDOS** based fonts.
FONT_COUNT	-440	**WORD**	Total number of fonts pointed to by the ***FONT_RING*** pointers.
RESERVED	-438	90 **BYTE**s	Reserved for future use.
CUR_MS_STAT	-348	**BYTE**	Bitmap of mouse status since the last interrupt as follows: **Bit** — **Meaning** 0 — Left mouse status (0=up) 1 — Right mouse status (0=up) 2 — Reserved 3 — Reserved 4 — Reserved 5 — Mouse move flag (1=moved) 6 — Right mouse status flag (0=hasn't changed) 7 — Left mouse status flag (0=hasn't changed)
RESERVED	-347	**BYTE**	Reserved for future use.
V_HID_CNT	-346	**WORD**	Number of times the text cursor has been hidden (0 = visible).
CUR_X	-344	**WORD**	X position where mouse pointer will be drawn.
CUR_Y	-342	**WORD**	Y position where mouse pointer will be drawn.
CUR_FLAG	-340	**BYTE**	Mouse redraw flag (if non-zero, mouse pointer will be redrawn at the next vertical blank interrupt).
MOUSE_FLAG	-339	**BYTE**	Mouse interrupt flag (0=disable interrupts)
RESERVED	-338	**LONG**	Reserved for future use.
V_SAV_XY	-334	**2 WORD**s	X and Y position of the text cursor as saved by the VT-52 emulator.
SAVE_LEN	-330	**WORD**	Height of the form saved in ***SAVE_AREA*** in pixels.
SAVE_ADDR	-328	**LONG**	Address of the first **WORD** of screen data contained in ***SAVE_AREA***.
SAVE_STAT	-324	**LONG**	Save status flag as follows: **Bit** — **Meaning** 0 — Save buffer valid? (0=no) 1 — Width of save (0=16 bits, 1=32 bits)
SAVE_AREA	-322	256 **BYTE**s	Save buffer for the mouse pointer,

USER_TIM	-66	**LONG**	Pointer to a routine which occurs at each timer tick. (use **vex_timv()** instead). Routine ends by jumping to function pointed to by ***NEXT_TIM***.
NEXT_TIM	-62	**LONG**	See above.
USER_BUT	-58	**LONG**	Pointer to a routine called each time a mouse button is pressed (use **vex_butv()** instead).
USER_CUR	-54	**LONG**	Pointer to a routine called each time the mouse needs to be rendered (use **vex_curv()** instead).
USER_MOT	-50	**LONG**	Pointer to routine called each time the mouse is moved (use **vex_motv()** instead).
V_CEL_HT	-46	**WORD**	Current text cell height.
V_CEL_MX	-44	**WORD**	Number of text columns – 1.
V_CEL_MY	-42	**WORD**	Number of text rows – 1.
V_CEL_WR	-40	**WORD**	Number of bytes between character cells.
V_CEL_BG	-38	**WORD**	Text background color.
V_COL_FG	-36	**WORD**	Text foreground color.
V_CUR_AD	-34	**LONG**	Text cursor physical address.
V_CUR_OF	-30	**WORD**	Offset (in bytes) from physical screen address to the top of the first text character.
V_CUR_XY	-28	**2 WORD**s	X and Y character position of the text cursor.
V_PERIOD	-24	**BYTE**	Current cursor blink rate.
V_CUR_CT	-23	**BYTE**	Countdown timer to next blink.
V_FNT_AD	-22	**LONG**	Pointer to system font data (monospaced).
V_FNT_ND	-18	**WORD**	Last ASCII character in font.
V_FNT_ST	-16	**WORD**	First ASCII character in font.
V_FNT_WD	-14	**WORD**	Width of the system font form in bytes.
V_REZ_HZ	-12	**WORD**	Horizontal pixel resolution.
V_OFF_AD	-10	**LONG**	Pointer to font offset table.
RESERVED	-6	**WORD**	Reserved for future use.
V_REZ_VT	-4	**WORD**	Vertical pixel resolution.
BYTES_LIN	-2	**WORD**	Bytes per screen line.
PLANES	0	**WORD**	Number of planes in the current resolution.
WIDTH	2	**WORD**	Width of the destination form in bytes.
CONTRL	4	**LONG**	Pointer to the *CONTRL* array.
INTIN	8	**LONG**	Pointer to the *INTIN* array.
PTSIN	12	**LONG**	Pointer to the *PTSIN* array.
INTOUT	16	**LONG**	Pointer to the *INTOUT* array.
PTSOUT	20	**LONG**	Pointer to the *PTSOUT* array.
COLBIT0	24	**WORD**	Color bit value used for plane 0.
COLBIT1	26	**WORD**	Color bit value used for plane 1.
COLBIT2	28	**WORD**	Color bit value used for plane 2.
COLBIT3	30	**WORD**	Color bit value used for plane 3.
LSTLIN	32	**WORD**	Last pixel draw flag (0=draw, 1=don't draw). Used to prevent the last pixel in a polyline segment drawn in XOR mode from overwriting the first pixel in the next line.
LNMASK	34	**WORD**	Line draw pattern mask.
WMODE	36	**WORD**	VDI writing mode.
X1	38	**WORD**	X coordinate for point 1.
Y1	40	**WORD**	Y coordinate for point 1.
X2	42	**WORD**	X coordinate for point 2.
Y2	44	**WORD**	Y coordinate for point 2.
PATPTR	46	**LONG**	Fill-pattern pointer.

PATMSK	50	**WORD**	This value is AND'ed with the value in Y1 to give an index into the current fill pattern for the current line.
MFILL	52	**WORD**	Multiplane fill pattern flag (0=Mono).
CLIP	54	**WORD**	Clipping flag (0=disabled).
XINCL	56	**WORD**	Left edge of clipping rectangle.
XMAXCL	58	**WORD**	Right edge of clipping rectangle.
YMINCL	60	**WORD**	Top edge of clipping rectangle.
YMAXCL	62	**WORD**	Bottom edge of clipping rectangle.
XDDA	64	**WORD**	Text scaling accumulator (set to $8000 prior to blitting text).
DDAINC	66	**WORD**	Scaling increment. If *SIZE1* is the actual point size and *SIZE2* is the desired point size then to scale up use: $DDAINC = 256 * \frac{(SIZE\,2 - SIZE\,1)}{SIZE\,1}$ To scale down use: $DDAINC = 256 * \frac{SIZE\,2}{SIZE\,1}$
SCALDIR	68	**WORD**	Text scaling direction (0=down, 1=up).
MONO	70	**WORD**	Monospaced font flag.
SOURCEX	72	**WORD**	X coordinate of character in font form.
SOURCEY	74	**WORD**	Y coordinate of character in font form.
DESTX	76	**WORD**	X position on screen to output character at.
DESTY	78	**WORD**	Y position on screen to output character at.
DELX	80	**WORD**	Width of the character to output.
DELY	82	**WORD**	Height of the character to output.
FBASE	84	**LONG**	Pointer to the font character image block.
FWIDTH	88	**WORD**	Width of the font form in bytes.
STYLE	90	**WORD**	Special effects flag bitmap as follows: **Bit** — **Meaning** 0 — Thickening 1 — Lightening 2 — Skewing 3 — Underlining (not supported by Line-A) 4 — Outlining
LITEMASK	92	**WORD**	Mask to lighten text (usually $5555).
SKEWMASK	94	**WORD**	Mask to skew text (usually $5555).
WEIGHT	96	**WORD**	Width to thicken characters by.
ROFF	98	**WORD**	Offset above baseline used for italicizing.
LOFF	100	**WORD**	Offset below baseline used for italicizing.
SCALE	102	**WORD**	Text scaling flag (0=no scale).
CHUP	104	**WORD**	Character rotation angle in tenths of degrees (supported only in 90 degree increments).
TEXTFG	106	**WORD**	Text foreground color.
SCRTCHP	108	**LONG**	Pointer to two contiguous scratch buffers used in creating text special effects.
SCRPT2	112	**WORD**	Offset from first buffer to second (in bytes).
TEXTBG	114	**WORD**	Text background color.
COPYTRAN	116	**WORD**	Copy raster mode (0=Opaque, 1=Transparent).

SEEDABORT	118	**LONG**	Pointer to a routine called by the seedfill routine at each line. If not needed during a seed fill you should point it to a routine like the following:
			
			`seedabort:`
			`        sub.l    d0,d0`
			`        rts`

Line-A Font Headers

Raster system and **GDOS** fonts are linked to form a list of font headers which contain the information needed to render text. Outline text generated by **FSM** is inaccessible in this manner.

Each monospaced font contains a font header, character and horizontal offset table, and font form. All data types are in “Little Endian” (Intel format) and as such must be byte-swapped before use.

The font form is a raster form with each character laid side-by-side on the horizontal plane. The first character is **WORD** aligned but padding within the form only occurs at the end of a scanline to force the next scanline to be **WORD** aligned.

Each font header contains a pointer to the next font in the list. The list is terminated by a **NULL** pointer. The font header format is as follows:

Name	Offset	Type	Contents
font_id	0	**WORD**	Font ID number (must be unique).
point	2	**WORD**	Point size of font.
name	4	**32 BYTE**s	ASCII Name of font.
first_ade	36	**UWORD**	First ASCII character in font.
last_ade	38	**UWORD**	Last ASCII character in font.
top	40	**UWORD**	Distance from the top line of the font to the baseline.
ascent	42	**UWORD**	Distance from the ascent line of the font to the baseline.
half	44	**UWORD**	Distance from the half line of the font to the baseline.
descent	46	**UWORD**	Distance from the descent line of the font to the baseline.
bottom	48	**UWORD**	Distance from the bottom line of the font to the baseline.
max_char_width	50	**UWORD**	Width of the widest character in the font.
max_cell_width	52	**UWORD**	Width of the widest character cell in the font.
left_offset	54	**UWORD**	Amount character slants left when skewed.
right_offset	56	**UWORD**	Amount character slants right when skewed.
thicken	58	**UWORD**	Number of pixels to smear for thickening.
ul_size	60	**UWORD**	Size of an appropriate underline for the font.
lighten	62	**UWORD**	Mask for character lightening.
skew	64	**UWORD**	Mask for character skewing.
flags	66	**UWORD**	Font type flags.
hor_table	68	**LONG**	Pointer to the horizontal offset table. The horizontal offset table is an array of bytes with one entry per character denoting the pixel offset to the character.

off_table	72	**LONG**	Pointer to the character offset table. The character offset table is an array of **WORD**s with one entry per character denoting the byte offset into the font form of the character.
dat_table	76	**LONG**	Pointer to the character data.
form_width	80	**UWORD**	Width of the font form in bytes.
form_height	82	**UWORD**	Height of the font form in pixels.
next_font	84	**LONG**	Pointer to the next font in the list (0=no more fonts).
reserved	88	**UWORD**	Reserved for future use.

Line-A Function Calling Procedure

Line-A functions are called by simply inserting the opcode into the instruction stream. For example, the 'Hide Mouse' function is called with the following assembly language instruction:

```
dc.w    $A00A
```

Generally, the **Line-A** initialization function is called ($A000) and the address of the variable and/or font header tables are stored. Prior to each **Line-A** call variables are set as explained in the ***Line-A** Function Reference* and the function is then called. There is no method of error reporting available.