SECTION 1

VDP RAM USAGE IN ARMADILLO BASIC

Armadillo BASIC will differ from its predecessors (99/4A BASIC and TI Extended BASIC) in several ways. It will be able to use all of the 9918A modes: Graphics I, Graphics II, Multicolor and Text. Sprites will be supported in all modes, except the Text mode. User-controlled left, right, top and bottom margins will be supported. The default mode will be Graphics I, for consistency with the current products. Whenever the mode is changed, the screen will be cleared and all tables (pattern generator, color, sprite, etc.) will be initialized. The screen will revert to pattern mode for editing.

1.1 Errors

Several operations will result in errors, such as attempting to draw lines in text mode or attempting an INPUT statement in Graphics II mode or Multicolor mode. These will cause the error message "WRONG SCREEN MODE". Naturally, the interpreter will revert to Graphics I mode for error display and editing.

Previous BASICs had the pattern name table (screen image) and pattern generator table (fonts) overlapping. This limited the number of characters available and caused characters to be offset on the screen from their standard ASCII codes. Armadillo BASIC will maintain completely separate tables. Characters on the screen will have their standard ASCII codes.

1.2 Compatibility

NOTE

Assembly Language subprograms called from BASIC that access the screen will have to be modified.
PARTIAL FIX: The default character set will be loaded twice: once at zero offset and again at >60 offset; so that either at >41 or

a >A1 in the pattern name table will display an "A". This will at least help screen output.

A BASIC program will be able to define and modify characters 0 through 255, instead of the current 32 through 159 (32 through 143 for Extended BASIC). In Graphics I mode there will be 32 color sets instead of the current 16.

1.2.1 Compatibility Problem. How do we number the color sets? In current BASIC, the set including the space character is numbered 1. It should be either 5 in the new BASIC. One solution would be the following:

Codes	Set	Codes	Set
0-7	29	128-135	13
8-15	30	136-143	14
16-23	31	144-151	15
24-31	32	152-159	16
32-39	1	160-167	17
40-47	2	168-175	18
48-55	3	176-183	19
56-63	4	184-191	20
64-71	5	192-199	21
72-79	6	200-207	22
80-87	7	208-215	23
88-95	8	216-223	24
96-103	9	224-231	25
104-111	10	232-239	26
112-119	11	240-247	27
120-127	12	248-255	28

1.3 Sprites

The automotion routines in the console ROM will be modified so that they do not depend on fixed table locations. Two words in the new part of the scratchpad RAM will be allocated for the pointers. The power up code will set these to >300 and >780.

1.4 Graphics I Mode

* >0000 to >02FF Pattern name table
* >0300 to >037F Sprite attribute table for 32 sprites
* >03C0 to >03DF Color table
* >03E0 to >03EF Sound list (single chord)
* >0780 to >07FF Sprite velocity table
* >0800 to >0FFF Pattern generator table (character table)
  (Sprite pattern table coincident with character table)

Access from BASIC:

- PRINT, DISPLAY, INPUT, ACCEPT for text
- CALL HCHAR, CALL VCHAR for direct writes to PNT
- CALL GCHAR to read PNT
- CALL CHAR, CHARPAT to access PGT
- CALL SPRITE, etc. for sprites
- CALL COLOR sprite colors and character colors
- CALL SCREEN for background color
- CALL SOUND for chord

1.5 Text Mode

* >0000 to >03BF Pattern name table
* >03E0 to >03EF Sound list
* >0800 to >0FFF Character table

Access from BASIC:

- PRINT, DISPLAY, INPUT, ACCEPT for text
- CALL HCHAR, CALL VCHAR for direct writes to PNT
- CALL GCHAR to read PNT
- CALL CHAR, CHARPAT to access PGT
- CALL SCREEN for foreground and background colors
- CALL SOUND for chord

1.6 Multicolor Mode

* >0000 to >02FF Pattern name table
* >0300 to >037F Sprite attribute table for 32 sprites
* >03C0 to >03DF Color table
* >03E0 to >03EF Sound list (single chord)
* >0780 to >07FF Sprite velocity table
* >0800 to >0FFF Pattern generator table (character table)
  (Sprite pattern table coincident with character table)

Access from BASIC:

- CALL DOT for colored square
- CALL LINE for low-resolution line
- CALL GCHAR returns color of square
- CALL SCREEN for background color
- CALL SOUND for chord

1.7 Graphics II Mode

* >0000 to >17FF Pattern generator table (Character table)
* >1800 to >1AFF Pattern name table
* >1B00 to >1B7F Sprite attribute table
* >1B80 to >1BFF Sprite velocity table
* >1C00 to >1C0F Sound list (single chord)
* >2000 to >37FF Pattern color table
* >3800 to >3FFF Sprite generator table (initialized to standard text font, just to have it in memory)

NOTE

This assumes that no peripheral device (such as disk) is trying to use VDP ram.

Access from BASIC:

- PRINT, DISPLAY for text (copies patterns from SGT to PGT) (NO input from keyboard with echo) (insert character would be horrendous)
- CALL CHAR, CHARPAT for access to SGT
- CALL SPRITE, etc. for sprites
- CALL COLOR sprite colors
- CALL SCREEN for background color
- CALL DOT for colored dot
- CALL LINE for colored lines
- CALL ARC for arcs, if we can find a reasonable arc-drawing routine.

- CALL SOUND for chord