WORKING SPECIFICATION
MATTEL ADVANCED GRAPHIC INTERFACE CIRCUIT

0.0 PREVIEW

The MAGIC chip will be the core of Mattel's second generation TV game. As one of two custom ICs planned for this design, it will be responsible for generating composite video, controling the dynamic RAM memory, providing system clocking and integrating miscellaneous SSI and MSI functions for minimum chip count. Since the CPU selection has not been made at this time, this paper assumes the author's favorite, the Motorola 68000.

1.0 POWER

MAGIC will operate with only a +5 volt supply. It is expected that a number of power and ground pins and carefull chip layout will be necessary to allow proper decoupling of the video D/A converter.

2.0 RAM INTERFACE

The MAGIC chip will require direct access to a minimum of 16K words (16 bit) of RAM. Since the base system is projected to have 16K words of RAM, the CPU must also access this same RAM. To provide maximum CPU speed, the RAM data and address buses will be isolated from the CPU buses except during CPU accesses. To insure maximum CPU speed the MAGIC chip will interleave it's accesses with those of the CPU and internally pipeline data to allow immediate CPU access. In addition, the MAGIC chip will provide all RAM control signals (RAS, CAS, Enable, and address multiplex) and refresh management. The control and refresh design will accommodate 16K x 4 dynamic RAMs; the selected type should be dual sourced and is subject to Mattel Electronics approval. If possible, the design should also handle 64K x 4 (256k) RAMs.

3.0 CPU INTERFACE

MAGIC will provide the clock for the CPU. This clock will be at twice the color subcarrier frequency; 7.16 MHz for NTSC and 8.86 MHz for PAL and SECAM. These frequencies are the system crystal frequency divided by 2.

MAGIC will provide two different types of interupt signal to the CPU over a single line (-IPL2). The first of these will occur at the horizontal line rate divided by 256. Since both 525 and 625 line systems use nearly the same line rate

this will result in an interupt that occurs at nearly 60 Hz regardless of TV system. This interupt will be used for system timing functions to make games that play at the same speed on both 525 and 625 line systems. The second type of interupt will occur when the active scanning line on the TV screen passes a line number stored by the CPU in a MAGIC register. This interupt is intended primarily to allow the moving objects to be reused (details in the moving object section). Unless the second type of interupt is disabled, it will be necessary for the CPU to read a register to determine which has occured. Note that the timing interupt is asynchronous with both 525 and 625 line vertical field rates; this means that often the timing interupt will occur during the active picture and occasionaly both interupts will occur at the same time.

How VBI?

Communication between the CPU and MAGIC is through 6 control lines, 16 data lines and 13 of the 19 system address lines. The control lines are -AS, R/-W, -DTACK, -UDS, -LDS and -RESET. The functions of these lines are fully described in the 68000 literature. RAM data in and out of MAGIC and data communication between the CPU and MAGIC through the external bidirectional buffer use the same 16 data lines. In addition, data communication between the RAM and the CPU use the same bidirectional buffers. An output from MAGIC enables the buffers, while the R/-W line controls the direction. Obviously, the RAM interface mentioned above must work closely with the graphics generators and the CPU interface to prevent conflict in the use of data lines and buffers. The 13 address lines are split into two groups; 8 low order (A1 through A8) and 5 high order (A15 through A19). the remaining 6 mid order lines (A9 through A14) are not available to MAGIC; they are, however, routed to the RAM through an external buffer (further description in the RAM interface section). The 8 low order lines allow selection within MAGIC of 256 control registers and are also passed through to the RAM during CPU accesses, reducing external multiplexing. The 5 high order lines allow decoding to 16K word blocks to enable the MAGIC and RAM as shown in the system address map. Outputs from this decoder also go external to the sound chip (1-16K space), cartridge (1-32K space), Exec ROMs (2-32K spaces) and language ROMs (2-32K spaces). These outputs would be the first to go in a pin-out crunch, since they are only saving a simple TTL decoder.

4.0 TV SYNC TIMING AND CONTROL

All system timing is derived from a 14.32 MHz fundamental mode crystal (17.72 MHz for 625 line systems). This crystal will be trimmed to 4 times the chroma subcarrier frequency

in NTSC and PAL systems. To allow locking the system to an
external video source, a connection will be available to
allow slight changes to the crystal frequency under phase
lock control. The oscillator divided by 2 forms the CPU
clock and divide by 2 and divide by 4 signals go to the
video output section to generate the chroma subcarrier.

Dividing the oscillator by 2.5 produces the pixel clock. A
discussion of pixel size is in the graphics generator
section, but I mention here that we are generating a full
interlace TV signal complete with equalizing pulses and
serrated vertical sync pulse. This is being done for two
primary reasons. First, it will enable mixing external
video with MAGIC generated video and second, it is expected
that non-standard signals will suffer varying degrees of
performance degradation as TVs use more and more digital
signal processing. It is even possible that complete
incompatability could result. In addition to full interlace
the picture will be 24 rows of 40 alpha-graphic cards. Each
card will be 6 pixels wide and 8 pixels high (one pixel is
two interlaced lines high). A non-interlaced mode will be
available under software control to improve the appearance
of alpha characters.

The horizontal line is 364 pixels long; of these the active
picture is 240 pixels, blanking and sync require 75 pixels,
the left border is 28 pixels and the right border is 21
pixels. In 625 line systems the left border has 73 and the
right border has 66 pixels. Note that the left border is 7
pixels wider that the right; this allows for the tendancy of
TVs to cut off more of the left side of the picture and
results in a better centered picture. To allow simplicity
and economy in the picture generation circuits, the
horizontal counter must have the same counter outputs for
the active portion of the line for both 525 and 625 line
systems. It is also desirable that the counter output
during the blanking, sync, etc. are the same in both
systems. This can be achieved by resetting the counter to
zero 16 pixels before the end of the left border. At the
count of 256, the active picture ends and the right border
starts. For 625 line systems, the start of the blanking is
at 322 and the end of blanking is at 397. At 454 the
counter is reset to zero completing the 454 pixel line. For
525 line systems, the counter is forced to skip from 277 to
322; shortening the border by the required 45 pixels and
allowing the decoders to generate blanking, sync, etc. When
the count reaches 409 it is reset to zero completing the 364
pixel line (45 counts were skipped at end of right border,
409-45=364). Starting the count 16 pixels under the border
allows 16x16 pixel moving objects to completely "hide" under
the left border and leave only one pixel exposed on the

right border; using only an 8 bit X location. When the background is in the scroll mode, the left border will cover an additional 6 pixels (one card width).

The output of the horizontal counter is divided by 2 as a line alternation signal to the video output section to generate PAL and SECAM. This output is further divided by 128 to generate the timer interupt at 61.4 Hz (60.99Hz in 625 line systems).

The vertical counter counts 525 or 625 depending the system being generated. As in the horizontal counter, it is convient to arrange the start of the counter and skip counts to simplify the decoding. In this case, the counter is forced to skip once during the frame for 625 line systems and three times for 525 line systems; fortunately, two of the three 525 line system skips are exactly alike from decode and preset standpoints.

First, the 625 line system. Vertical blanking is 21 lines in each field. The top border is 51 lines in field 1 and 50 lines in field 2, the bottom border in each field is 49 lines and the active picture is 192 lines in each field. The lower 9 bits of the 10 bit vertical counter must be the same during the active picture lines since the same information is presented during each field (each pixel is 2 interlaced lines high). Again the counter is reset to zero 16 lines before the end of the top border allowing the moving objects to "hide" under the border. At line count 208, the bottom border starts, at count 257, the blanking for field 2 starts, at count 278, the top border for field 2 starts. At count 312 the counter is skipped to 512; resetting the low 9 bits to zero, as desired to scan the active lines for field 2. Count 720 brings the end of the active lines and start of the border. Count 769 ends the bottom border of field 2 and starts the blanking and sync for field 1. Field 1 blanking ends and the top border starts at 790. At 825 the counter is reset to zero completing the frame (200 counts were skipped, 825-200=625).

For 525 line systems the skip at 312 becomes a skip from 287 to 512 and the reset to zero happens at 800, causing 25 additional border lines to be skipped each place. 25 border lines are also skipped in each field at the end of the bottom border by skipping from 232 to 257 and 744 to 769.

When the background is in the scroll mode, 8 additional lines (1 card height) become border at the top of the active picture in each field. Non-interlace mode is achieved by forcing bit 10 to remain high, causing field 2 picture and field 1 blanking and sync to repeat (313 lines/625 system,

263 lines/525 system). MAGIC will syncronize to an external
source by providing composite sync to an input pin.

Decoders are provided to generate composite sync, composite
blanking, burst gating, SECAM "bottle" and PAL "meander
gate" burst blanking signals from the combined outputs of
the horizontal, vertical and line alternation counters.

VIDEO OUTPUT

The video output section receives signals from the graphic
and alpha generators, a pixel clock, Fsc and 2 Fsc, line
alternation (for PAL and SECAM) and PAL/NTSC/SECAM control.
Its output, from a primary and a secondary D/A converter,
are composite video for NTSC/PAL and Y and composite color
difference for SECAM. There is also an output for use with
an external video mixing gate.

Two video outputs allow spliting the graphic and alpha
generator outputs to two televisions. An application for
this split output is two player games where the players each
have their own display with both common and private
information. Another application would be dual user systems
where one group uses the graphics for a game while someone
else uses the alpha for kitchen, den or study applications.
An optional dual output modulator will be necessary to
implement this function. Alternately, the optional unit may
contain only the second modulator, with the original
modulator plugging into it to provide dual outputs. Since
SECAM uses both DACs to produce a single video output, the
split graphics/alpha mode is not available.

A 16 bit register controls the spliting function. The
following assignments are made:
   5 bit Graphic split
   5 bit Alpha split
   1 bit H or V split
   1 bit Graphic overlap on/off
   1 bit Alpha overlap on/off
   1 bit Graphic split on/off
   1 bit Alpha split on/off
   1 bit Alpha flip on/off

The 5 bits allow selection of any of the 24 rows or column 4
through 35, depending on the sense of the H/V bit. With the
graphic overlap off, the primary output displays the graphic
generator output from the top down to the selected row or
from the left across to the selected column. The secondary
output displays the remaining graphic output. When the
graphic overlap bit is on, the primary output displays to
the lower (or rightmost) split and the secondary output

displays from the upper (or leftmost) split. Normally the primary output displays alpha to the alpha split and the secondary output displays from there. With alpha overlap on, the primary displays to the lower (or rightmost) split and the secondary displays from the upper (or leftmost) split. The use of upper/lower for overlap allows a display that has a graphic overlap, but the alpha in the overlap area is switched between the primary and secondary outputs. This alpha could be an overlay of a map graphic showing private information alternatly to each player. The graphic and alpha split on/off enables the spliting functions; when off, full graphics and/or alpha are on both outputs. A one in the alpha flip bit causes the split alpha to swap outputs (primary to secondary and secondary to primary). This allows displays with the top half graphics and bottom half alpha from the primary output and the reverse from the secondary output.

The primary and secondary DACs will be four binary ratioed MOS transistors connected to sink current. Inputs to the DACs will also include sync, blanking, burst gate and NTSC/PAL control. The external gating output is asserted when the display color code is zero. A suitable delay is inserted in this signal to match it to the processing delay that the video generation will require.

Background and display color outputs from the graphic generators select outputs from the color map registers. There are 16 color map registers, each 12 bits wide. Four bits of each register describe the Y (luma) level. Four bits each describe the two color difference (R-Y and B-Y) levels. These are encoded as a sign bit with 3 maginitude bits. First the Y levels. For the background color, only the Y value is selected and is combined with the display Y value when the pixel is a moving object edge. One bit of the Y adjust signal indicates an edge; the remaining bits are data used to adjust the Y value. The adjustment method varies depending on the mode of the graphic generator; for more information see the graphic generator section.

The selected color difference signals are passed through the 4 pixel averagers to limit the bandwidth. The averaged color difference signals are then assembled with the Y signal to generate the required composite signal. This assembly is performed by dividing the color cycle into four quadrants. In the first quadrant, the R-Y is added to the Y; in the second quadrant, B-Y is added to Y; in the third quadrant R-Y is subtracted from Y; in the fourth quadrant, B-Y is subtracted from Y. The MUX and adder/subtractor that assemble this composite signal are clocked from the phase continuous Fsc and 2 Fsc signals from the timing section.

When PAL is being generated, the action in quadrants 1 and 3 are swaped on alternate lines.

GRAPHIC GENERATORS

First a word about pixel size. The 5.7 MHz (4Fsc/2.5) pixel rate was originally choosen to give a square pixel. A square pixel balances the resolution horizontaly and vertically. This balance allows moving objects to be rotated without distortion or designer intervention. It also allows simpler software for drawing lines and circles since the construction technique used can be mirrored on the 45 degree axis. Antialiasing algorithms can also be applied with ease when both axes are the same resolution. Other advantages are based on comparsion to the leading alternative, a 7.16 MHz (2 Fsc) pixel rate. With this higher pixel rate the pixels are no longer square, although this has not been a significant problem with our current system which has pixels with the same aspect ratio. With a goal of 40 characters per line, the faster pixel rate would use 8x8 pixel characters (320 pixels/line) and the slower rate would use 6x8 characters (240 pixels/line). The most obivous difference is the 33% larger data base necessary to support the smaller pixels. This data problem continues to the on chip character ROM requiring 40% more bits for a 7x8 matrix vs a 5x8 matrix. The 40 6x8 cards fit within the SMPTE "safe title area" making all cards usable vs the leftmost two cards being left unused for critical information or action. For moving object positioning, 240 pixels plus 16 pixels "under" the border is 256 X locations with the square pixels. This allows an 8 bit X register. Coupled with an 8 bit Y register, a 16 bit register conviently locates an object. The alternate is a 9 bit X location; requiring two register ,addresses and two writes to move the object. The finial advantage relates to "false" color generation. In the current Keyboard Component the high resolution alphanumerics, required careful pattern selection to avoid pixel combinations that directly or through harmonics created 3.58 MHz energy. The proposed 5.7 MHz pixel rate generates 3.58 MHz energy only on the 5th harmonic of an 8 pixel pattern (1 on 7 off, 2 on 6 off, 3 on 5 off, 4 on 4 off, and the inverse patterns). I have not checked, but I suspect that none of those patterns have large amounts of 5th harmonic energy. The result is a great freedom of choice in font patterns for the alphanumeric characters. The same freedom applies to the background and moving objects. With the smaller pixels, even a grey and white textured background can result in "false" color; it happens all the time on broadcast TV with striped ties and jackets, automobile grills and buildings with vertical decorations or windows.

Color
artifact
(hi-freq
luma →
chroma)

BORDER

The border is divided into four quadrants. Each quadrant fills a corner and has its own color assignment. The division between the upper left and upper right corner color is a vertical line starting from any X location in the active picture area. A similar division is made along the other three sides of the picture. Each of the four dividing points are independently variable. One 16 bit register specifies the colors (4 bits/color) and two more 16 bit registers set the dividing points (8 bits/divider).
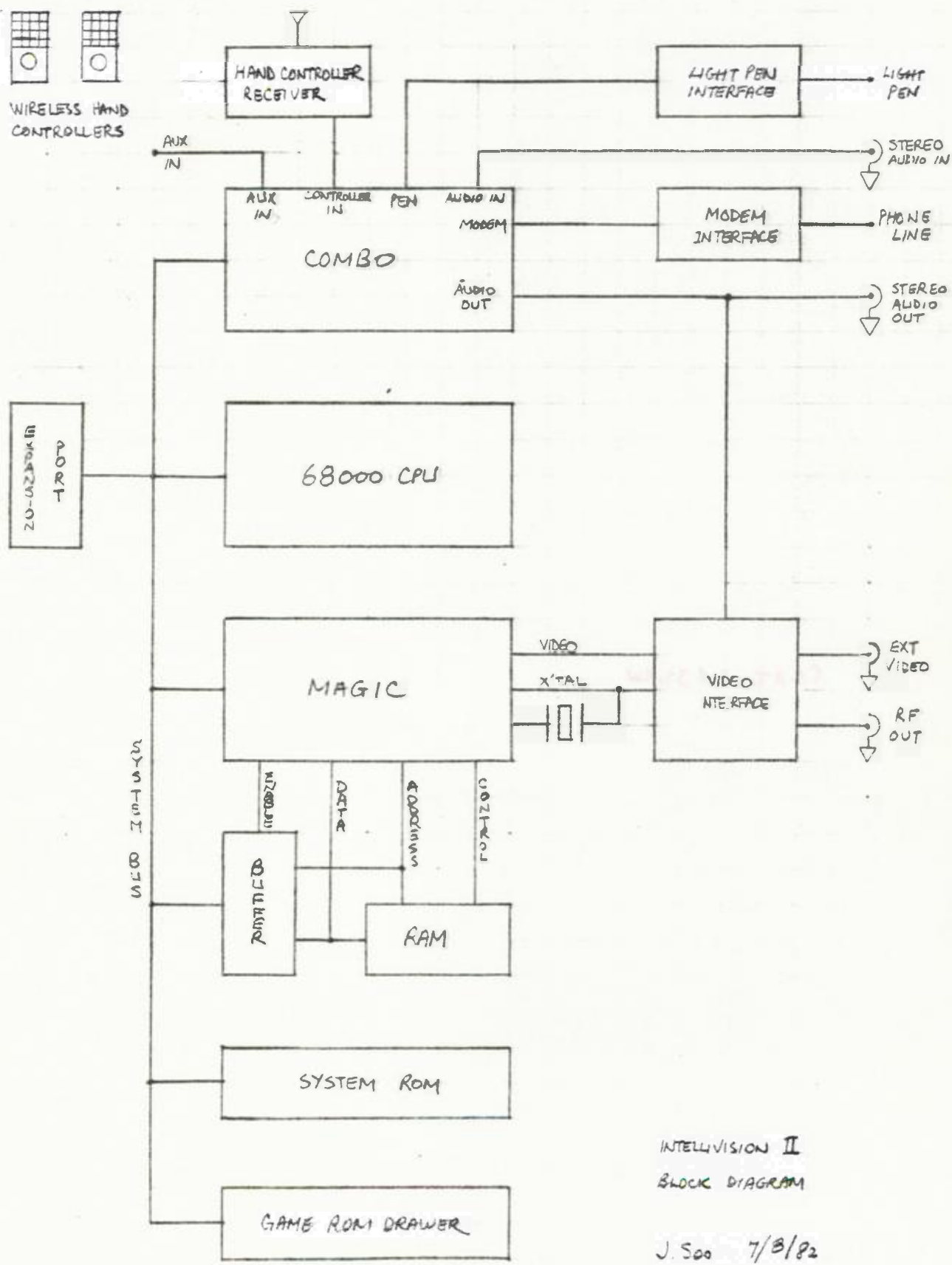
BACKGROUND

The background can be displayed in 3 modes: bit map, cards, and high resolution cards.

CONFIDENTIAL

WORKING SPECIFICATION
MATTEL COMBO INTEGRATED CIRCUIT

0.0 Introduction

The Mattel COMBO circuit contains a digital signal processor

INTELLIVISION II
BLOCK DIAGRAM

J. Soo 7/8/82

A19 – A1	A20-A22	O	1	2 — E	F
FFFFE	512K	256 MAGIC REG	SWAPPED		
F8000	496K	256 COMBO REG			
F0000	480K	32K OPERATING SYSTEM			
E0000	448K	64K GAME ROM CARTRIDGE	NOT USED	NOT USED	NOT USED
C0000	384K				
80000 32K		64K LANGUAGE ROM	LANGUAGE	ROM	EXPANSION
A0000	320K				
80000	256K	192K EXPANSION PORT	OTHER	EXPANSION	ADDRESS
40000	128K				
20000	64K	112K RAM EXPANSION	NOT USED	NOT USED	NOT USED
10000	32K				
08000	16K				
00000	0	16K RAM			
HEX ADDR.	WORD ADDR.				

INTELLIVISION II
ADDRESS MAP
J See

MAGIC Registers

RBS 7

8/23/82
Dave Walden

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	#	
F00FE	Time-Out Regs (MSB)																15	
F00FC	Time-Out Regs (LSB)																14	
F00FA	RAM Sy.	Pel Den.	No. Lns.	Chrom code		RTC En	Ln En	Interrupt Line Count									13	
F00F8					RTC Int	Ln Int		Current Line Count									12	(read-only)
F00F6	GS EN	GO EN	H O R	Screen Split 1					AS EN	AO EN	AS RV	Screen Split 2					11	
F00F4	Al Xt	Alpha Y Shift		Alpha X Shift					Y Xt	BG Y Shift		X Xt	BG X Shift				10	
F00F2					Priority 2			Priority 1			Priority 0						9	Background Card Priorities
F00F0	Character Read-Back Pattern																8	(read-only)
F00EE					Read-Back Phase					ASCII Code							7	
F00EC	Border Color 3 index			Border Color 2 index				Border Color 1 index				Border Color 0 index					6	Border Palette
F00EA	Left Border Divide Position								Right Border Divide Position								5	
F00E8	Top Border Divide Position								Bottom Border Divide Position								4	
F00E6	Bit	Int 1	Extvid		Background Excess				Al En				Alpha Excess				3	
F00E4	Alpha Table Address																2	
F00E2	Background Address																1	
F00E0	Background Pel Pattern Base Address								Moving Object Pel Pattern Base Address								0	

(READS from MAGIC registers must be done a word at a time. WRITES to MAGIC registers may be on a byte or word basis.)

MAGIC Registers

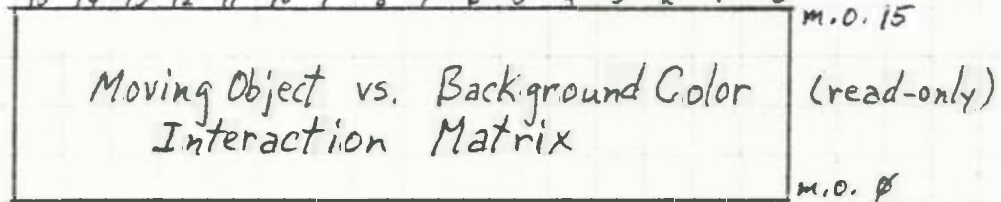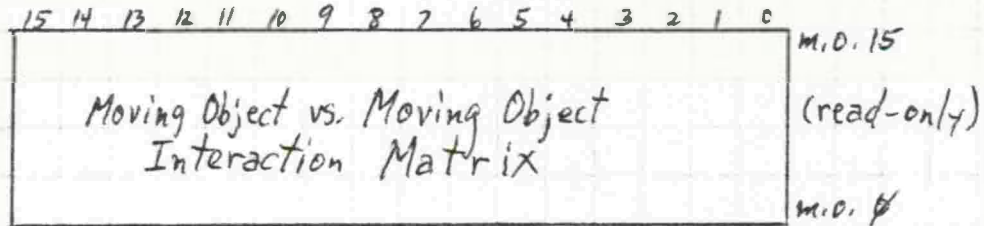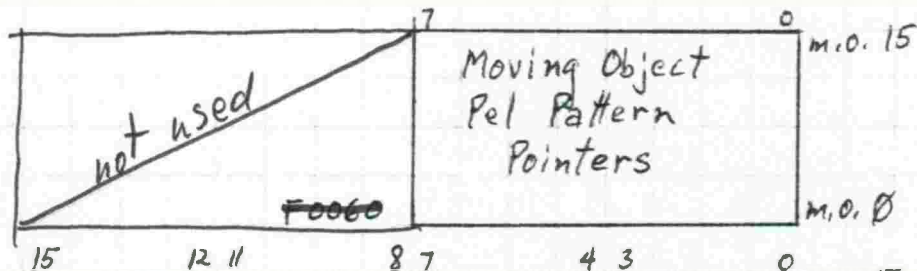**RBS6**

	11	8	7	4	3	0	color 15
not used F00C0		Y		BY B−Y		RY R−Y	color 0

Master Palette

RBS5

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	m.o. 15

Moving Object vs. Background Color Interaction Matrix (read-only)

F00A0 — m.o. 0

RBS4

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	m.o. 15

Moving Object vs. Moving Object Interaction Matrix (read-only)

F0080 — m.o. 0

RBS 3

	7	0	m.o. 15
not used F0060		Moving Object Pel Pattern Pointers	m.o. 0

RBS2

15	12	11	8	7	4	3	0	m.o. 15
moving obj. Color 3 index		moving obj. Color 2 index		moving obj. Color 1 index		moving obj. Color 0 index		m.o. 0

F0040

Moving Object Palettes

RBS1

15	13		9	8	7	6	5	1	0	m.o. 15
HIRES		X Scale Factor		XMIR	INVIS	INTER	Y Scale Factor		YMIR	m.o. 0

F0020

Moving Object Parameters

RBS0

15	8	7	0	m.o. 15
X Position		Y Position		m.o. 0

F0000

Moving Object Positions
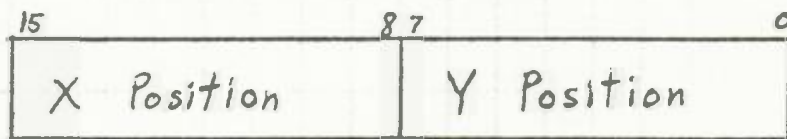
READS: on Word basis

WRITES: on Word _or_ Byte basis

MAGIC Registers

RBS∅ - Moving Object Position

Register Bank Select ∅ is a set of 16 registers, one for each of the 16 moving objects. Each register contains the position of the upper left-most pel of the associated moving object. The format of each register is as follows:

15	8 7	0
X Position	Y Position	

Bits 0-7: Y Position ($\emptyset \leq Y \leq \overset{255}{\cancel{208}}$)

This 8-bit field contains the ~~value~~ vertical position of the moving object in number of lines from the top of the screen. Since the "active picture" starts on line 16 and there are 16 lines per moving object, Y position ∅ places the moving object totally "under" the top border. Similarly, Y position 208 places the moving object totally under the bottom border
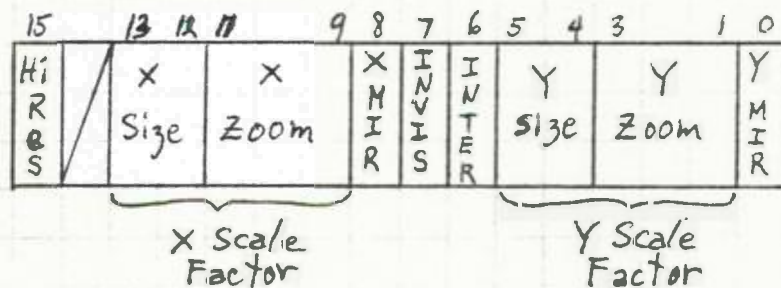
Bits 8-15: X Position ($\emptyset \leq X \leq 255$)

This 8-bit field contains the horizontal position of the moving object in number of pels from the left of the screen. Since the "active picture" starts on pel 16 and ends on pel 255, and since a moving object is 16 pels wide, X position ∅ ~~and 255~~ places it totally under the left ~~and right~~ ~~borders respectively.~~ border and 255 leaves just the left-most pels exposed at the right border.

RBS1: Moving Object Parameters

Register Bank Select 1 is a set of 16 registers, one for each of the moving objects. Each register has the following format:

15		13	12	11		9	8	7	6	5	4	3		1	0
Hi RES		X Size		X Zoom			X MIR	I N V I S	I N T E R	Y Size		Y Zoom			Y MIR

X Scale Factor   Y Scale Factor

Bit 0: Y Mirror

A "1" means the moving object's pels are "flipped", top for bottom about an axis running horizontally through its center.

Bits 1-5: Y Scale Factor (see discussion of Scale Factor)

This 5-bit field contains information for the vertical scale factor.

Bit 6: Interaction Enable

A "1" enables interaction detection and the reporting thereof in the interaction matrices RBS4 and RBS5.

MAGIC Registers

<u>RBS1: Moving Object Parameters</u> (cont'd)

<u>Bit 7: Invisible Color Enable</u>

<u>Lo-Res mode</u>

A "1" enables pel color code 01 to indicate the "Invisible Color", i.e. transparent but able to interact.

A "0" allows the normal scheme whereby color code 01 selects Color 1 from the moving object's palette in RBS2.

<u>Hi-Res mode</u>

A "1" enables pel luminance code 01 to indicate the "Invisible Color" and to override the pel's color code.

A "0" allows the normal scheme whereby luminance code 01 selects ¼ Adjustment and lets the pel color code select the color.

RBS1 (cont'd)

Bit 8: X Mirror

A "1" means the moving object's lines are "flipped", left for right about an axis running vertically through its center.

Bits 9-13: X Scale Factor (see discussion of Scale Factor)

This 5-bit field contains information for the horizontal scale factor.

Bit 15: High Resolution Enable

A "1" means the moving object's pel pattern is coded for the High Resolution display mode and that it will be displayed as such.

RBS 2: Moving Object Palettes

Register Bank Select 2 is a set of 16 registers, one for each of the 16 moving objects. Each register acts as a palette, or color menu, from which each pel of the associated moving object is assigned a color according to its 2-bit color code. In most situations, this 2-bit code acts as an index into the moving object's palette. (The palette in turn merely contains ~~some~~ indices into the Master Palette.) The format of the Moving Object Palettes is as follows:

15 12 11	moving obj. 8 7	moving obj. 4 3	moving obj. 0
moving obj. color 3 index	moving obj. color 2 index	moving obj. color 1 index	moving obj. color 0 index

Each 4-bit field contains an index into the 16-color Master Palette. Under some conditions, the contents of fields 0 and 1 may be ignored in the assignment of color to a pel. For details, see the discussion of moving objects.

MAGIC Registers

RBS3: Moving Object Pel Pattern Pointers

Register Bank Select 3 is a set of $\overset{16}{\wedge}$ single-byte registers { (whose high order bytes do not exist).} They contain pointers to the pel patterns of each moving object. These pointers are combined with the Moving Object Pel Pattern Base Address to form the addresses of each pel pattern. (See discription of register $\emptyset$ of RBS7 for details of this scheme.)

MAGIC Registers

RBS 4: Moving Object vs. Moving Object Interaction Matrix (Read-only)

    Register Bank Select 4 is a set of 16 registers, one for each of the 16 moving objects. Each bit position also corresponds to a moving object. The set of 16 registers, then, forms a 16×16 matrix which records interactions between moving objects. A "1" means the moving object of the register number and the moving object of the bit position number interacted within the "active picture" area (i.e. not under the borders).

    The diagonal entries will always be "∅". This means that moving objects do not self-interact (e.g. they do not stumble over their own feet).

    (The effect of a read operation is to reset all the bits of the register to "∅".)

MAGIC Register

## RBS5 : Moving Object vs. Background Color Interaction Matrix (Read-only)

Register Bank Select 5 is a set of 16 registers, one for each of the 16 moving objects. Each bit position corresponds to one of the background colors from the 16-Color Master Palette. The set of 16 registers, then, forms a 16×16 matrix which records interactions between moving objects and colors forming the background. A "1" means that an interaction occurred in the "active picture" area between a moving object of the register number and a background color of the bit position number. (The effect of a read operation is to reset all the bits of the register to "0".)

MAGIC Registers

RBS6: Master Palette

Register Bank Select 6 is a set of 16 12-bit registers, each containing chrominance information for one of the 16 colors (including tints and shades) that the programmer has chosen to be available for simultaneous display. The format of each register is as follows:

11	8	7		4	3		0
	Y		RY			BY	
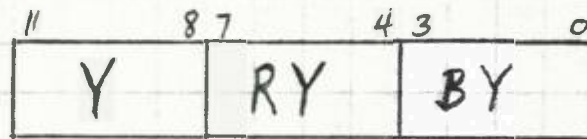
The attached table shows examples of values of Y, RY and BY for various colors. Y can range from 0 to 15, and it is coded in normal binary notation. RY and BY may range from -8 to 7, and they are coded in excess-8 binary notation. To find the excess-8 form of a binary number, just add 8 to the number and discard any carry-out bit.

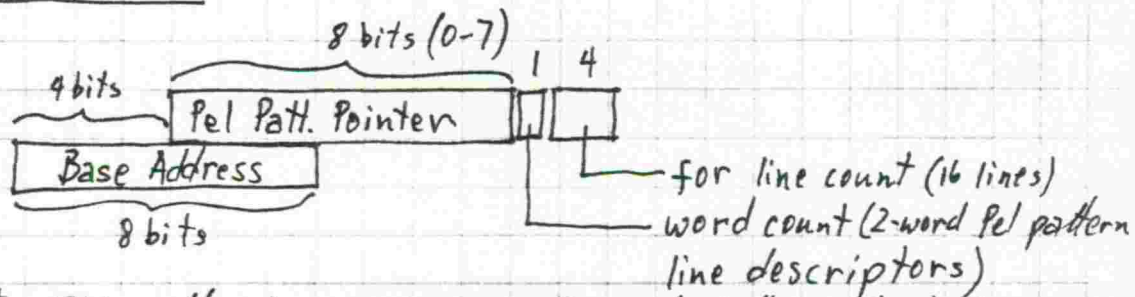| COLOR	Y	RY	BY	Y+RY	Y+BY	Y-RY	Y-BY
BLACK	4	0	0	4	4	4	4
BLUE	6	-3	6	3	12	9	0
RED	7	6	-3	13	4	1	10
TAN	11	1	-3	12	8	10	14
GREY GREEN	7	-4	-2	3	5	11	9
GREEN	8	-5	-3	3	5	13	11
YELLOW	11	2	-4**	13	6	9	15**
WHITE	14	0	0	14	14	14	14
GRAY	10	0	0	10	10	10	10
CYAN	9	-6	1	3	10	15	8
ORANGE	10	5	-5	15	5	5	15
BROWN	6	-1	-2	5	4	7	8
MAGENTA	9	6*	2	15*	11	2	7
LIGHT BLUE	10	1	5***	11	15***	9	4
YELLOW GREEN	10	-5	-5	5	5	15	15
PURPLE	6	5	2	11	8	1	4

* RY actually 7 but forced to 6 so that Y+RY < 16
** BY actually -5 but forced to -4 so that Y-BY < 16
*** BY actually 6 but forced to 5 so that Y+BY < 16
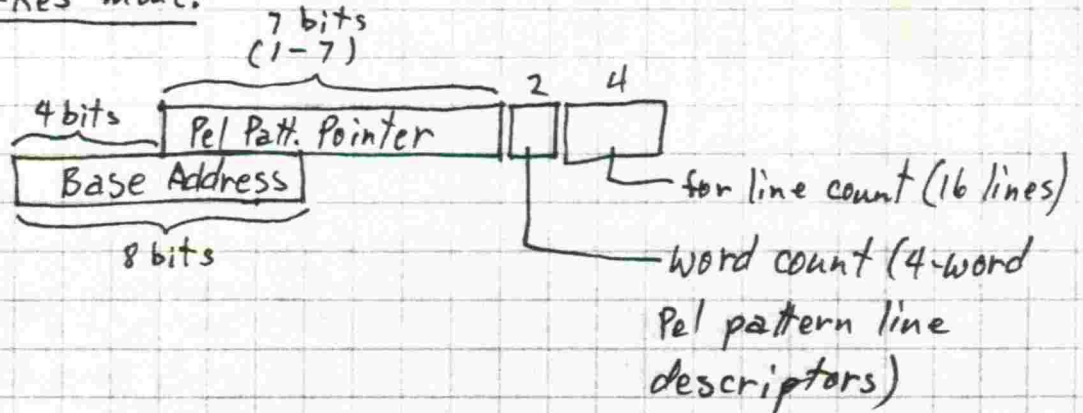
MAGIC Registers

RBS7, Register 0 ~~or~~
~~located at 002~~

Bits 0-7: Moving Object ~~X~~ Pel Pattern Base Address ~~(of the object's)~~

This 7or 8-bit field is ~~combined with the~~ added to each moving object's Pel Pattern Pointer ~~for each moving object to form~~ (from RBS3) to form the word address of the first word of ~~the~~ each moving object's pel pattern. These word addresses ~~are~~ are formed as follows: ~~as word addresses~~
in Lo-Res mode:

        8 bits (0-7)   1   4
  4 bits ┌──────────────┐ ┌─┐ ┌──┐
  ┌─────┤ Pel Patt. Pointer │ │ │ │
  │ Base Address   │ └─┼──┘── for line count (16 lines)
  └────────────────────────┘   └───── word count (2-word Pel pattern
        8 bits   line descriptors)

Note: This addressing requires Lo-Res pel patterns to begin on 32-word boundaries.

in Hi-Res mode:

              7 bits
             (1-7)   2   4
  4 bits ┌──────────────┐ ┌─┐ ┌──┐
  ┌─────┤ Pel Patt. Pointer │ │ │ │
  │ Base Address   │ └─┼──┘── for line count (16 lines)
  └────────────────────────┘   └───── word count (4-word
        8 bits   Pel pattern line
   descriptors)

Note: This addressing requires Hi-Res pel patterns to begin on 64-word boundaries.

RBS7, Register Int(dd)

Bits 8-15: Background Pel Pattern Base Address

This 8-bit field is ~~combined with~~ added to each background card's pel pattern pointer (from entries in the Background Table) to form the word address of each background card's pel pattern. These word addresses are formed as follows:
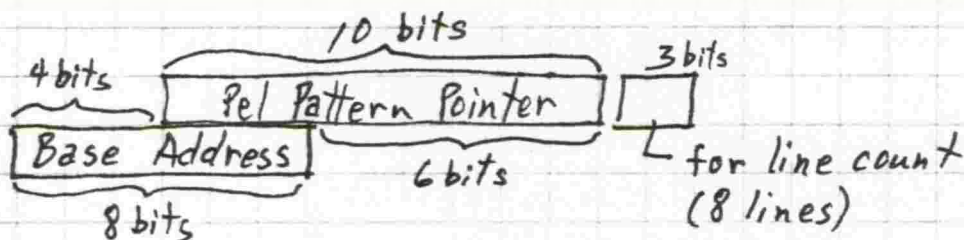
Note: This addressing requires background card pel patterns to begin on 8-word boundaries.

MAGIC Registers

RBS7, Register 1: Background Address

~~mark~~ This register contains ~~the~~ a 16-bit word address.

In Card mode, this is the address of the

Background Table. In Bit Map mode, this is

the address of a ~~11,520~~ 11,520 -word whole-screen pel

pattern.

RBS7, Register 2: Alpha Table Address

This register contains the 16-bit word address of the first

of a series of 480 words ~~~~

which form a table 20 words wide by 24 words high. This table

~~of ASCII characters.~~ ~~These 480 words will contain,~~

contains 960 ASCII characters which will appear on the screen

in left to right sequence, 40 characters per row, 24

~~line~~ rows per screen display. ~~~~

~~~~ Note that this table will be larger by an amount

2 4 × 16 n words, where n is the number in the Alpha Excess

field.

MAGIC Registers

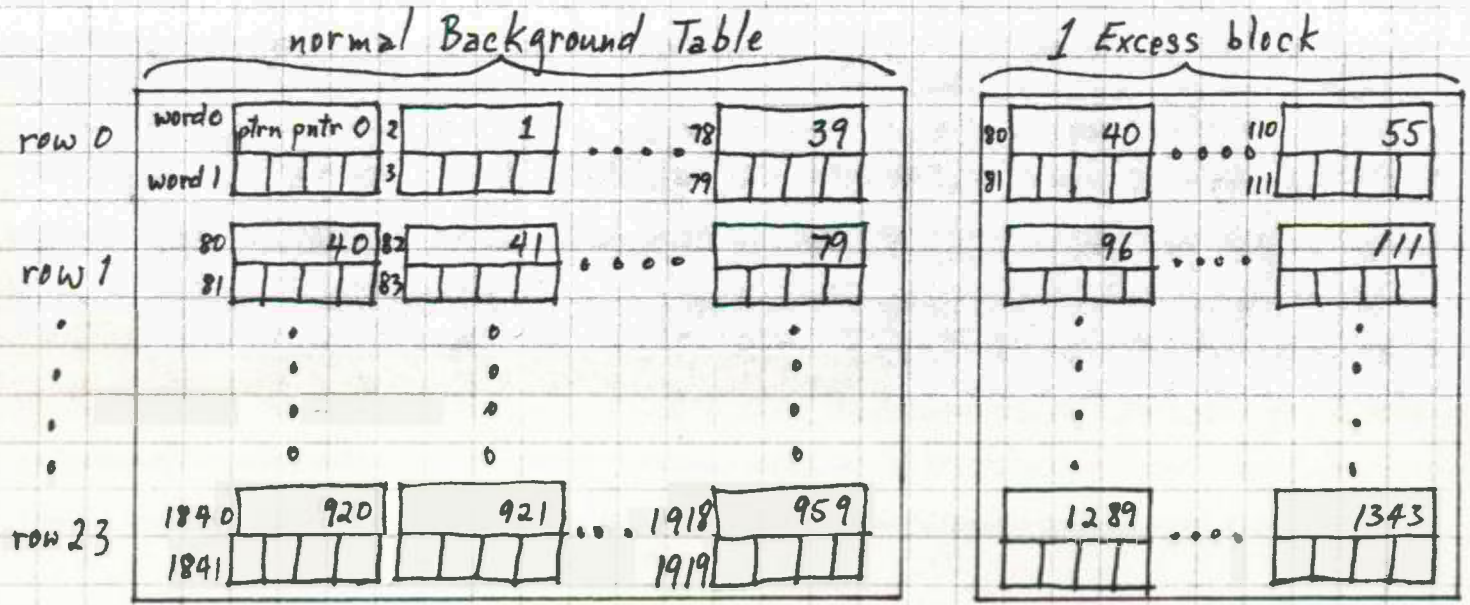Bit 7: ~~crossed out~~ Alpha ~~Display~~ Enable

A "1" enables display of the contents of the Alpha Table. A "0" ~~disables~~ the ~~Alpha~~ display ~~mode~~. The Alpha display will be superimposed over any other display on the screen.

Bits 8-11 ~~crossed out~~ – Background Excess

Card mode:

In card mode, this 4-bit field contains the number of 16-entry (32-~~character~~ word) ~~blocks~~ segments that MAGIC will assume have been added to each row of ~~cards~~ 2-word entries in the Background Table. After accessing 40 entries to display 40 cards on the screen, MAGIC will skip the designated number of 16-entry ~~blocks~~ (32-words) segments before displaying the contents of the next ~~word segment~~ (40-entry) row.

This effectively widens the Background Table (without increasing the number of cards displayed at any one time), and it allows for limited horizontal scrolling without altering the table contents. In other words, each Excess count enables horizontal scrolling for an extent of 16 cards.
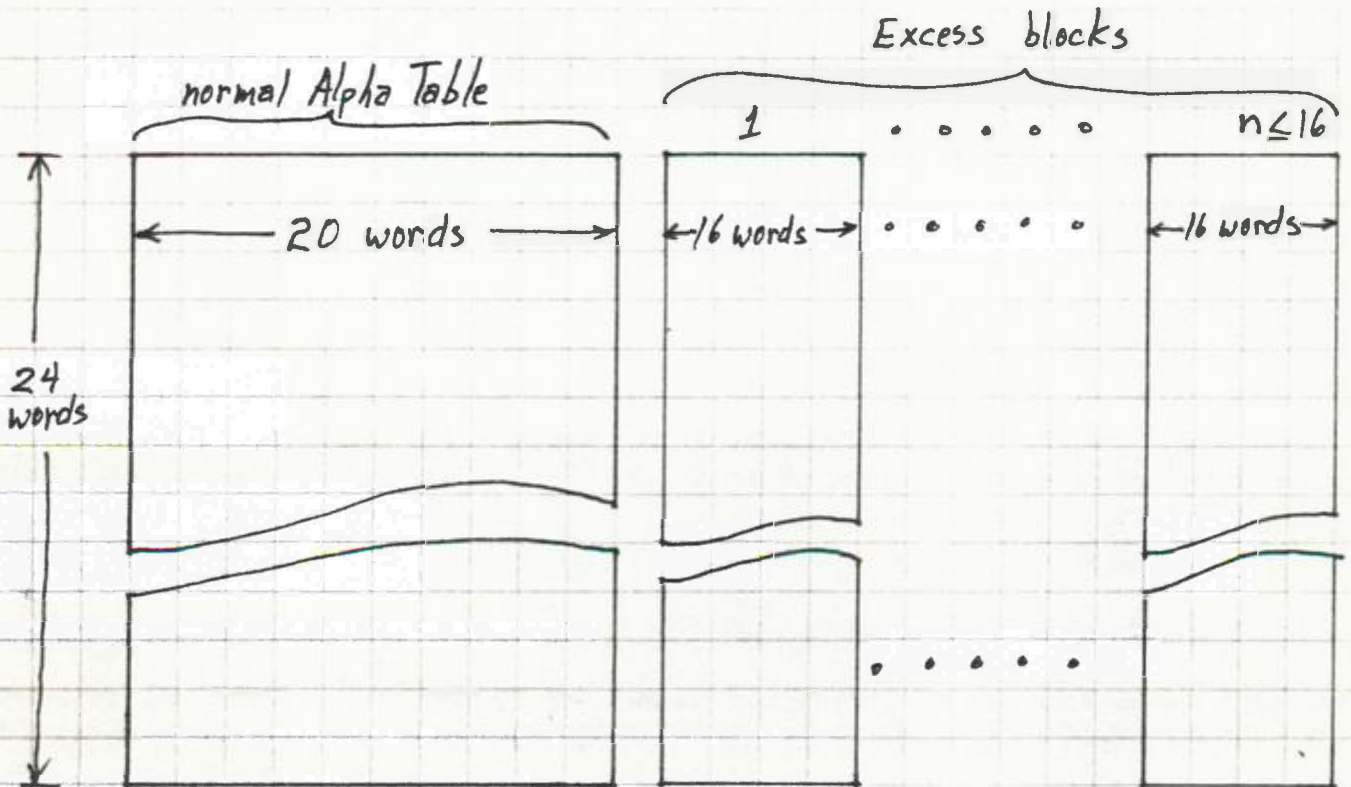

normal Background Table   1 Excess block

row 0	word 0: ptrn pntr 0, 2 ... 1 ... 78 ... 39 / word 1: 3 ... 79	80 ... 40 ... 110 ... 55 / 81 ... 111	
row 1	80 ... 40, 82 ... 41 ... 79 / 81 ... 83	96 ... 111	
row 23	1840: 920 ... 921 ... 1918: 959 / 1841: ... 1919	1289 ... 1343	

(up to 15 excess blocks allowed)

MAGIC Registers

RBS7, Register 3 ~

Bits 0-3: ~~EXAMPLE~~ Alpha Excess

This 4-bit field contains the number of 16-word (32-character) blocks that MAGIC will assume have been added to each row of the Alpha Table. After displaying the contents of 20 words to form a row of 40 characters on the screen, MAGIC will skip the designated number of 16-word blocks before displaying the contents of the next 20-word segment. This effectively widens the Alpha Table (without increasing the displayed information), and it allows for limited horizontal scrolling without moving the table contents.
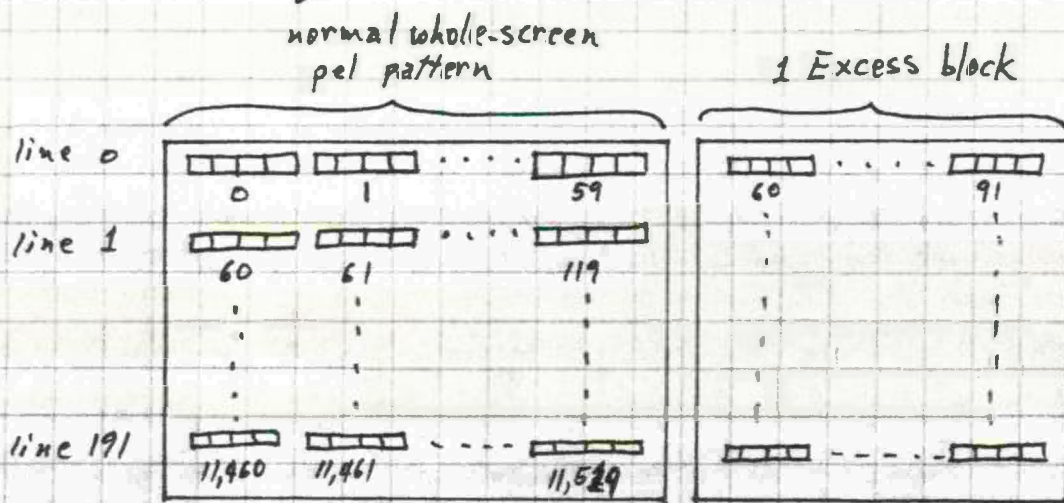
RBS7, Register 3 (cont'd)

Bits 8-11: ~~Background Excess~~ Background Excess (cont'd)

Bit Map mode:

In Bit Map mode, this 4-bit field contains the number of 32-word segments that MAGIC will skip after displaying the 60 words (240 pels) that describe one line of the whole-screen pel pattern. This effectively widens the pel pattern by 128 pels for each Excess count, and it enables scrolling to the it extent. Each Excess count implies an *additional* memory requirement of 6,144 words (32 × 192).

normal whole-screen pel pattern   1 Excess block

line 0   0   1 ····· 59   60 ····· 91

line 1   60   61 ····· 119

line 191   11,460   11,461 ····· 11,519

(up to 16 Excess blocks allowed)

RBS7, Register 3 (cont'd)

Bit 13: ~~xxxxxxxxxxxx~~ External Video Enable

A "1" enables MAGIC to synchronize its internal clock with an external compositie video signal.

Bit 14: ~~xxxxxxxxxxxx~~ Interlace Disable

A "1" disables video display field interlacing.

Visually, the effect is to halve the ~~xxxxx~~ line density a n d to improve the appearance of the Alpha mode display.

Bit 15: ~~xxxxxxxxxxxx~~ Background Bit Mode Enable

A "1" enables the Bit Map mode (for background display).

A "0" enables the Card mode.

MAGIC Registers

RBS7, Register 4

Bits 0-7: ~~Unreadable crossed-out text~~ Bottom Border Divide Position

This 8-bit field contains the number of the left-most pel that has the color of the right-side portion of the bottom border (i.e. the first pel of ~~the second~~ color). [position] [3]

Bits 8-15: ~~Unreadable crossed-out text~~ Top Border Divide Position

This 8-bit field contains the number of the left-most pel that has the color of the right-side portion of the top ~~bottom~~ border (i.e. the first pel of ~~this second~~ color). [position] [1]

~~See~~ (See the diagram in the description of register 5.)

RBS7, Register 5

Bits 0-7: ~~Unreadable crossed-out text~~ Right Border Divide Position

This 8-bit field contains the number of the upper-most line that has the color of the lower portion of the right border (i.e. the first line of ~~the second~~ color). [position] [3]

Bits 8-15: ~~Unreadable crossed-out text~~ Left Border Divide Position

This 8-bit field contains the number of the upper-most line that has the color of the lower portion of the left border (i.e. the first line of ~~the second~~ color). [position] [2]
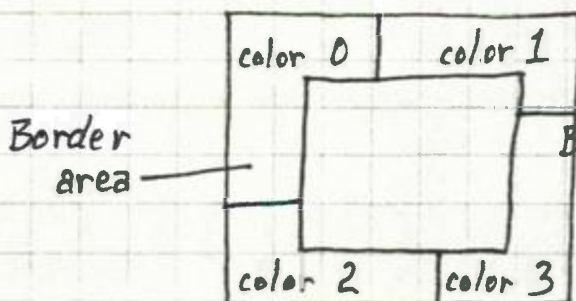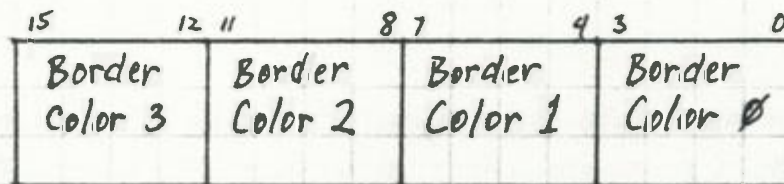

Illustration of Border Divides

color 0 | color 1

Border area

color 2 | color 3

MAGIC Registers

RBS7, Register 6: Border Palette

This register consists of four 4-bit fields, each of which contains an index to one of the 16 colors of the Master Palette. (RBS6) Each field corresponds to a portion of the border and it selects the color for that portion. The register is formatted as follows:

15	12 11	8 7	4 3	0
Border Color 3	Border Color 2	Border Color 1	Border Color 0

(See description of register 5 for a diagram of the arrangement of the colors.)

MAGIC Registers

RBS7, Register 7

Bits 0-6: ~~transmit address format~~ ASCII Code

This 7-bit field contains the ASCII code
of the character whose pel pattern will be returned
in register 8, word by word, in a sequence determined
by the ReadBack ~~Mode~~ Phase (see following field).

Bits 8-12: ~~has the address FOO and~~ Read-Back ~~Mode~~ Phase

This 5-bit field may only take on ten values, each
corresponding to one phase in the four possible Alpha read-back
modes:

mode	phase name	phase hex code
Card Normal	Card Normal	0
Card DoubleWidth	Left	4
	Right	F
Bit Map Normal	Left ⅔	1
	Right ⅓	9
	Left ⅓	19
	Right ⅔	11
Bit Map Double Width	Left Third	5
Middle		15
	Right Third	D

MAGIC Registers

RBS7, Register 8: Character Read-Back Pattern (Read-only)

~~(character address found)~~
~~Read Only~~

This register contains ~~the~~ pel pattern information for displaying the ASCII character whose code was placed in ~~register~~ the ASCII Code field of register 7. Eight sequential reads will provide all the information for the phase whose code resides in the Read-Back Phase field of register 7.
(See discussion of Character ~~Read-Back~~ Generation.)

~~RBS7, Register 9, Background Color Priority~~
~~(index byte address FX0F2)~~

RBS7, Register 9

Bits 0-11: — Background Card ~~Color~~ Priorities
~~(index byte address FX0F2)~~

This 12-bit field consists of three 4-bit subfields, each containing a display priority level. The 2-bit Card Priority Index ~~Priority Number~~ in each entry of the Background Table will index one of these fields to get a priority level available to colors on its particular card. (Priority Card Index 3 is an exception as it specifies "no priority level" for colors on its card.)

MAGIC Registers

RBS7, Register 10

Bits 0-2: ~~[struck out]~~ Background X ~~Scroll~~ Shift

This 3-bit field contains the number of pel positions that the background will be shifted to the right ~~[struck out]~~ on the screen. This value may range from 0 to 5 (i.e. to 1 pel less than a card width).

Bit 3: ~~[struck out]~~ X Border Extend

A "1" causes the ~~vertical~~ left borders to extend inward ~~extend~~ by 6 pels (i.e. a card width). This hides the junk "exposed" by horizntal background shifts.

Bits 4-6: ~~[struck out]~~ Background Y Shift

This 3-bit field contains the number of line positions that the background will be shifted downward ~~[struck out]~~ on the screen. This value may range from 0 to 7 (i.e. to 1 line less than a card height).

Bit 7: ~~[struck out]~~ Y Border Extend

A "1" causes the ~~horizontal~~ top borders to extend inward by 8 lines (i.e. a card height). This hides the junk "exposed" by vertical background or Alpha shifts.

RBS7, Register 10 (cont'd)

Bits 8-11: ~~××××××××××~~ Alpha X Shift ~~Scroll~~

This 4-bit field contains the number of pel positions that the Alpha display will be shifted to the right on the screen. This value may range from 0 to 11 (i.e. to 1 pel less than it takes to display the two ~~words~~ ASCII characters contained in a word).

Bits 12-14: ~~××××××××××~~ Alpha Y Shift

This 3-bit field contains the number of ~~p~~ line positions that the Alpha display will be shifted downward on the screen. This value may range from 0 to 7 (i.e. to 1 less than the ~~width~~ height of a row of Alpha characters).

Bit 15: ~~××××××××××~~ Alpha Border Extend

A "1" causes the left border ~~××××××××××~~ to extend inward by 12 pels (i.e. 2 character widths). This hides the junk "exposed" by ~~Alpha~~ horizontal Alpha shifts.

MAGIC Register

RBS7, Register 11   (See discussion of screen splits.)

Bits 0-4: Screen Split #2

This 5-bit field contains the line or pel position (divided by 8) of ~~the~~ Screen Split #2.

Bit 5: Alpha Screen Reverse

A "1" means that the roles of the primary and secondary screens are reversed for Alpha display.

Bit 6: Alpha Overlap Enable

A "1" causes Alpha display to end at Split #2 on the primary screen and to begin at Split ~~#~~ 1 on the secondary screen. (Alpha split must be enabled.)

Bit 7: Alpha Split Enable

A "1" causes the Alpha display to end on Split #2 on the primary screen and to begin at Split #2 on the secondary screen.

RBS7, Register11 (cont'd)

Bits 8-12: — Screen Split #1

This 5-bit field contains the line or pel position (divided by 8) of Screen Split #1.

Bit 13: — Horizontal Split

A "1" means the lines splitting the Graphic and/or Alpha screen display(s) between primary and secondary screens run horizontally. A "0" means the split runs vertically.

Bit 14: — Graphic Overlap Enable

A "1" causes Graphic display to end at Split #2 on the primary screen and to begin at Split 1 on the secondary screen. (Graphic split must be enabled.)

Bit 15 — Graphic Split Enable

A "1" causes the Graphic display to end on Split #1 on the primary screen and to begin on Split #1 on the secondary screen.

MAGIC Registers

RBS7, Register 12 (Read-only)

Bits 0-8: — Current Line Count

This 9-bit field contains the number of the display line which ~~the~~ is currently being refreshed on the screen.

Reading this ~~&~~ register resets the interrupts, but the line count will not be affected.

Bit 9 ~~8~~ — Line Count Interrupt Flag

A "1" means that the interrupt that MAGIC generated for the 68000 was due to the current Line Count matching the value in the Interrupt Line Count field. This bit will be reset to "0" when the register is read.

Bit 10: — Real Time Clock Interrupt Flag

A "1" means that the interrupt that MAGIC generated for the 68000 was ~~due to the~~ a "tick" of the Real Time Clock. This bit will be reset to "0" when the register is read.

MAGIC Registers

RBS7, Register 13

Bits 0-8: Interrupt Line Count

This 9-bit field contains the number of the display line for which MAGIC (if enabled) will generate an interrupt to the 68000 when it is ~~refreshed~~ matched by the value of the Current Line Count.

Bit 9: Line Interrupt Enable

A "1" means that MAGIC is enabled to ~~will~~ generate an interrupt to the 68000 when the Current Line Count matches the Interrupt Line Count.

Bit 10: Real Time Clock Interrupt Enable

A "1" means that MAGIC is enabled to ~~will~~ generate an interrupt to the 68000 upon every "tick" of the Real Time Clock.

Bit 11-12: Chrominance Encoding Method

This is 2-bit field ~~contains~~ specifies which ~~chroma~~ chrominance encoding method is being used. This is ~~RGB, SECAM, PAL, NTSC~~ or coded as follows:

chrominance encoding	code
SECAM	0
RGB	1
PAL	2
NTSC	3

RBS7, Register 13 (cont'd.)

Bit 13: — Number of Lines

A "1" means that the screen display consists of 625 lines. A "0" means 5.25 lines comprise the display.

Bit 14: — Pel Density

A "1" means that the higher of two pel densities will be used and that the resultant active picture will be slightly smaller and the borders slightly wider. A "0" means that the lower pel density will be used.

Bit 15: — RAM Size

A "1" means that RAM is composed of 64K chips. A "0" means that the chip size is 16K.

MAGIC Registers

RBS7, Registers 14 & 15: Time-Out Registers

These two registers contain sixteen 2-bit fields, register 14 containing the low order bits and register 15 the high order bits. Each 2-bit field contains a code specifying the number of WAIT states normally involved in accessing each 32K portion of memory. Bits in the zero bit position pertain to the first (low) 32K, bits in the next position to the next 32K, etc. If the number of WAIT states exceeds the number required for normal access, MAGIC will generate the DTACK signal so as to avoid hanging up the 68000. The number of WAIT states allowed are coded as follows:

high order bit:	0	0	1	1
low order bit:	0	1	0	1
no. WAIT states:	8	2	1	0

[Since these values affect the basic functioning of the system, the RESET initialization sequence should write them into registers 14 and 15 as soon as possible.

A background screen display consists of 192 lines, each line made up of 240 pels. The screen display may be logically divided into "cards", each card comprised of 8 lines, each line containing 6 pels. In this "Card mode", each pel is assigned a color and each card can have a selection of attributes, including degree of resolution and display priority. Alternately, in "Bit Map mode", the screen display may be logically monolithic, without degrees of resolution or priority, but with greater variety and complexity of coloration.

Each background picture (which may be larger than what can be displayed on the screen at once) is described ~~divided~~ in memory by a "pel pattern". A pel pattern describes the color (and sometimes luminance) of each pel or trio of pels in the picture. These pel patterns may be on a card-by-card basis, as in the Card mode, or they may be for the entire background picture, as in the Bit Map mode.

CARD MODE
~~Card Mode~~

In the Card mode, the displayed picture consists of 24 rows of 40 cards per row. The pel pattern for each card has a corresponding pointer which is part of the entry for each card in the Background Table. The pel patterns may encode pel information in the form required for either High or Low Resolution display, and colors may be assigned a display priority level which determines whether a given moving object ~~will~~ would pass "behind" or "in front" of it.

BACKGROUND GENERATION

~~Card Mode~~

Background Table —

This table resides in memory ~~at. A~~ and (without an Excess count) ~~word~~ it consists of 960 2-word entries, each entry containing information ~~attributes~~ for a particular card in the background display. The first 40 ~~words~~ entries pertain to the 40 cards of the first row of the display (ordered left to right), and the next 40 entries ~~pertains only~~ pertain to the 40 cards of the second row, ~~et cetera until~~ and so on for all 24 rows of the display. (See the description of Register 3 of BB57 for details of the Background Excess count.)

Each 2-word entry is formatted as follows:

	15 14 13	12	11	10	9 0	
word 0	card Pri. 2 PRI	HI RES	X MIRROR	Y MIRROR	P e l Patten Pointer	
word 1	card color 3 index	card color 2 index	card color 1 index	card color 0 index		~~4 color Card~~ ~~Color Index~~ Palette

BACKGROUND GENERATION

where the fields are defined as follows:

<u>Pel Pattern Pointer</u> — This value is ~~combined with~~ <u>added to</u> the Background Pel Pattern Base Address (i.e.g. Dot RBS7) to form the address of the first word of ~~the~~ a card's pel pattern. These <u>word</u> addresses are formed as follows:

note: This addressing requires card pel patterns to begin on 8-word boundaries

<u>Y Mirror</u> — A "1" inverts the order of displaying the rows of the pel pattern, i.e. they are "flipped" ~~top~~ top for bottom.

<u>X Mirror</u> — A "1" ~~inver~~ reverses the order of displaying the pels of each row of the pel pattern, i.e. they are "flipped" left for right.

BACKGROUND GENERATION

<u>HiRes</u> — A "1" means the pel pattern is coded
for High Resolution ^{card} mode display. A "∅" means
it is coded for Low Resolution ^{card} mode.
(High and Low Resolution cards may be
mixed in a display.)

<u>2 Pri</u> — A "1" means ^{just} card colors 3 and 2 take

~~one the Card Priority as their priority~~

on a priority level determined by the
Card Priority Index for this card. A "∅"
means card colors 3, 2 and 1 take on
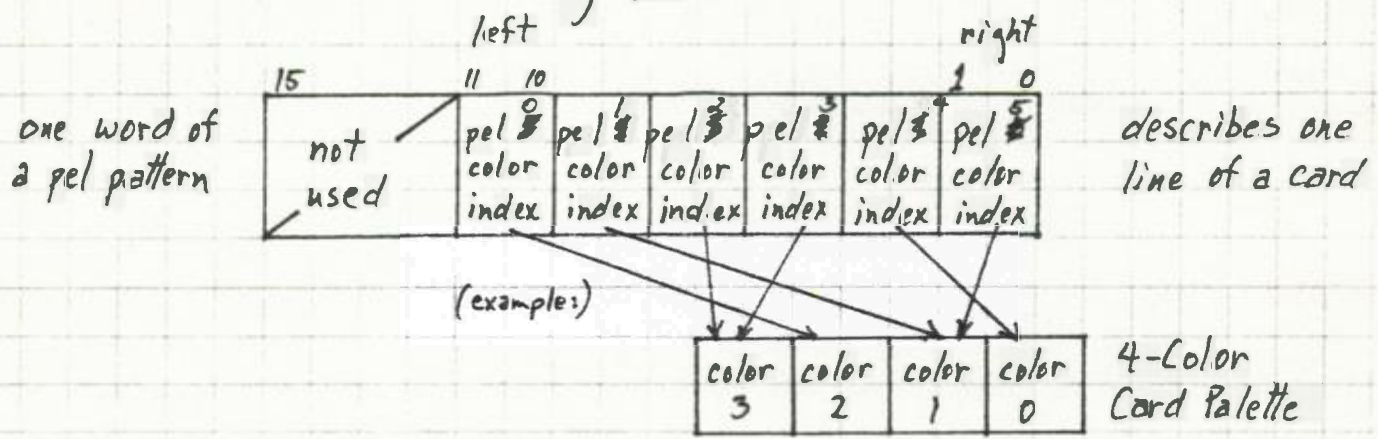a priority level.

<u>Card Pri</u> — This is the Card Priority Index.
Values 0, 1 and 2 act as indices to select one
of the three priority levels available in register 9
of RBS7. The selected priority level becomes the
Card Priority Level and is then available to two or
three of the card colors as their priority levels.
Index value 3 means no priority level is available for colors
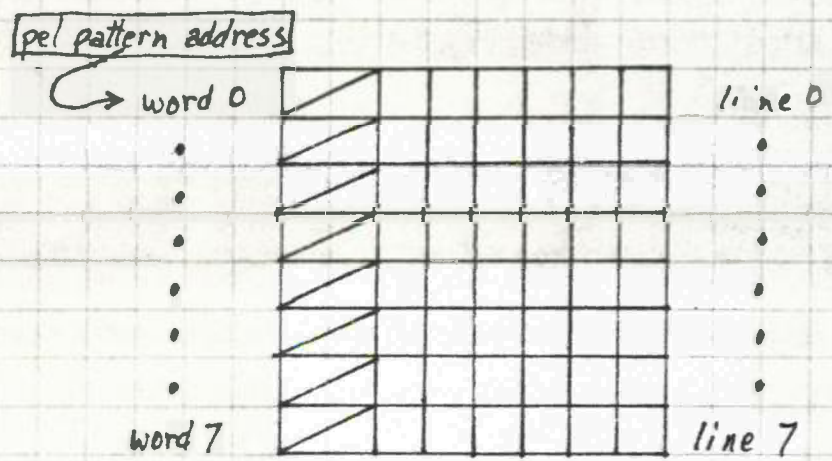this card.

BACKGROUND GENERATION

Lo-Res Pel Patterns –

The pel patterns in Low Resolution mode provide for selection of a color for each pel in a line. The colors available for selection are those specified in the ~~Color~~ Card ~~Color Card~~ Palette of the associated entry in the Background Table. Each word has the following format:

one word of a pel pattern

15		left 11 10					right 1 0		describes one
not used		pel 0 color index	pel 1 color index	pel 2 color index	pel 3 color index	pel 4 color index	pel 5 color index		line of a card

(example:)

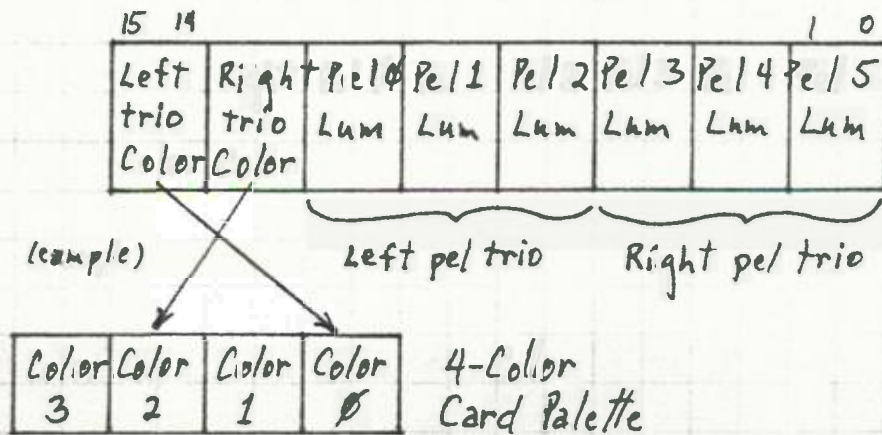| color 3	color 2	color 1	color 0	4-Color Card Palette

Pel 0 is the left-most pel of a card & line, and pel 5 is the right-most. Each 2-bit field indexes one of the four colors in the ~~4-Color card~~ Card Palette. Each subsequent word of the pel pattern describes the next lower line of the background card. They are arranged as follows:

pel pattern address

→ word 0   line 0

                  .
.   .
.   .
.
.   .
.   .
.   .

word 7   line 7

a pel pattern for a
Lo-Res card

Hi-Res Pel Patterns

The pel patterns in High Resolution mode provide _luminance_ information for each pel in a line and _color_ information for horizontal groups of three pels at a time. The colors available are those specified in the Palette of the associated entry in the Background Table. The luminance specified may be one of four levels (or "adjustments"), the absolute value being dependent on the original luminance Y specified in the color's unadjusted chrominance code. (These chrominance codes are the Y/BY/RY values given in the Master Palette, RBS 6. Each word of the pel pattern describes one line of the Hi-Res card, and it has the following format:

15 14							1 0
Left trio Color	Right trio Color	Pel 0 Lum	Pel 1 Lum	Pel 2 Lum	Pel 3 Lum	Pel 4 Lum	Pel 5 Lum

(example)   Left pel trio   Right pel trio

Color 3	Color 2	Color 1	Color 0

4-Color
Card Palette

Pel 0 is the left-most pel of a card's line, and pel 5 is the right-most. Each 2-bit color field indexes one of the four colors in the Card Palette and it does so for a trio of pels. Each subsequent word of the pel pattern describes the next lower line of the background card, just as in Lo-Res mode.

The following table maps the luminance code to the luminance adjustment according to the Y value in the chrominance for the pel's color:

Luminance code	range of Y value			
	4-6	7-9	10-12	13-14
00	0	0	0	0
01	+2	-2	-4	-6
10	+4	+2	-2	-4
11	+6	+4	+2	-2

luminance adjustment

Color Priority

The relative priority levels of a moving object and a background pel determine whether the moving object passes "behind" or "in front" of that pel. When a moving object encounters a background pel, the item with higher priority level ~~passes~~ remains "in front". If both items have the same priority level, the background pel stays "in front".

The priority level of the moving object is simply equal to the moving object's ordinal number. Moving Object 0 has low priority level, and Moving Object 1 has a higher priority level.
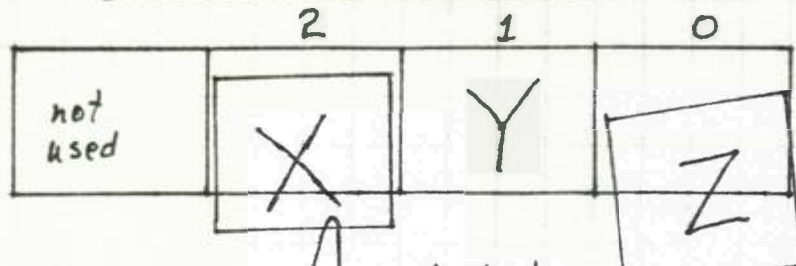
The priority level of a background pel, on the other hand, results from a complex selection process which may even result in no priority level (which is less priority than zero). For a pel to have a priority level, it must:

1) be part of a Card mode display,
2) have Card Priority Index unequal to 3, and
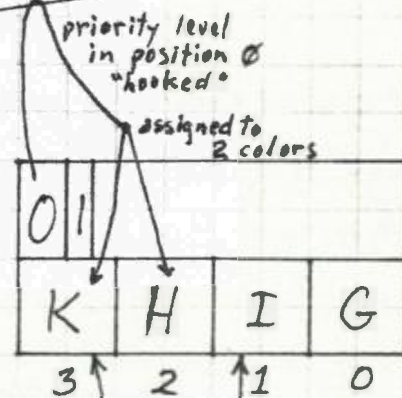3) have a color from the ~~4 color card~~ Palette that can have a priority level.

If a card's Card Priority Index in its Background Table entry is 3, its colors have no priority associated with them. If the Card Priority Index is 2, 1 or 0, it will index and select a priority level from register 9 of RBS7 and associate it with colors 3 and 2 of the card's ~~Card Color~~ Palette. If the 2-Pri bit is OFF in the card's Background Table entry, the selected priority level will also be associated with color 1 of the ~~Card Color~~ Palette. The following diagrams attempt to illustrate this scheme.

RB37
Register 9

Background
Card Priorities

	2	1	0
not used	X	Y	Z

no priority

priority level in position 2 "hooked"

assigned to 3 colors

priority level in position 0 "hooked"

assigned to 2 colors

Background Table entries

card Palette word

Background Table entry

3		

2 0		
G	H	I	J

3 2 1 0

0 1		
K	H	I	G

3 2 1 0

Pel Patterns
(selected words illustrated)

		11		00	

5 4 3 2 1 0

		11	01		

5 4 3 2 1 0

pel 5 : color G
priority level X

pel 2 : color J
no priority level

pel 4 : color K
priority level Z

pel 3 : color I
no priority level

BACKGROUND GENERAT.ION

BIT MAP MODE

In the Bit Map mode, the background picture is not sub-divided into cards, each having a pel pattern. The address in register 1 of RBS7 does not point to an intermediate Background Table, but instead directly to one huge pel pattern. Each word of this pel pattern describes four pels. It takes 60 words to describe each of the 192 lines in a screen-size display, 11,520 words in all. The pel pattern words and the pel fields within them describe pels sequentially from left to right and then from top to bottom of the picture (although pels are arbitrarily numbered from right to left). The 4-bit pel fields contain an index into RBS6, the "16-Color ~~Master~~ Palette", and therefore, in Bit Map mode, each pel may be assigned any one of sixteen colors, regardless of position. The format for a Bit Map pel pattern is as follows:

word 0	pel 0 color	pel 1 color	pel 2 color	pel 3 color
word 1	4	5	6	7
word 59	236	23 7	238	239

60 words describe 1 line out of 192 lines

CHARACTER GENERATION

Background

The purpose of Character Generation is to allow generation of pel patterns for ASCII characters at runtime. The programmer writes the ASCII code of the character whose pel pattern he wants into the ASCII Code field of register 7 of RBS 7. MAGIC then passes pel pattern information back in register 8 (the Character Read-Back Pattern) of RBS 7.

Four Character Read-Back modes are available which provide pel pattern information for character displays of two sizes in both Card and Bit Map display modes. These Character Read-Back modes are:

1) Card Normal
2) Card Double Width
3) Bit Map
4) Bit Map Double Width

Each mode consists of one or more phases. It takes eight consequtive reads of the Character Read-Back Pattern to acquire all the pel pattern information for a particular phase. The programmer may then change the Read-Back Phase code, for ~~get~~ more information on the same character, or he may change the ASCII Code for pel pattern information ~~of another~~ about a different character.

(in register 7 of RBS 7)

No color control.
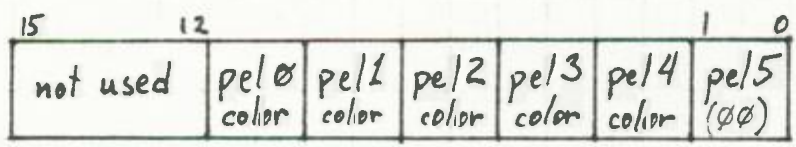Display as a lighter version of background.
No inverse video.
(used for displaying VIDEOTEX)

1. Card Normal mode

This mode consists of one phase which provides information for a card-size (6 pels wide) pel pattern containing the image of one ASCII character.

Phase Code φ ("Card Normal")

Each successive read of the Character Read-Back Pattern provides a word containing the next lower line of pel pattern in the form ~~required~~ required ~~required~~ for Background cards:

15 12	pel φ color	pel 1 color	pel 2 color	pel 3 color	pel 4 color	1 0 pel 5 (φφ)
not used	pel φ color	pel 1 color	pel 2 color	pel 3 color	pel 4 color	pel 5 (φφ)

Each 2-bit field corresponding to a pel will be either 11 or φφ, where 11 indicates a portion of the character image. The right pel, pel 5, will always represent the space between characters, and it's pel field will therefore contain φφ.
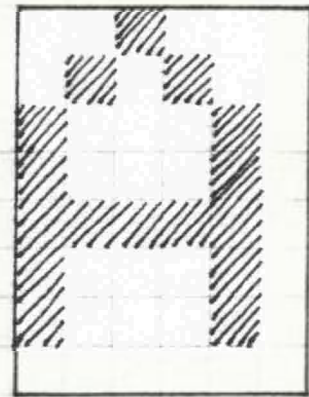
Luminance shifts: 00 & 11.

Card NormalMode, example

pel pattern word pos. | words returned from read-back register | corresponding card display

Phase = Card Normal, ASCII code = A

pos	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+0									1	1						
+1							1	1			1	1				
+2		not			1	1							1	1		
+3					1	1							1	1		
+4			used		1	1	1	1	1	1	1	1	1	1		
+5					1	1							1	1		
+6					1	1							1	1		
+7																

Phase = Card Normal, ASCII code = B

pos	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+0					1	1	1	1	1	1	1	1				
+1					1	1							1	1		
+2		not			1								1	1		
+3					1	1	1	1	1	1	1	1				
+4			used		1	1							1	1		
+5					1	1							1	1		
+6					1	1	1	1	1	1	1	1				
+7																

2. Card Double Width mode

This mode consists of two phases. Together, they provide pel pattern information to display the character images two cards wide (12 pels). By doubling each line of the pel patterns, the programmer can restore the normal proportions of the character.

Phase code 4 ("Left")

Eight consequtive reads in this phase provide the information for the left half of the character image. Each read provides a word in the format for Background cards:

15 12	pel 0 color	pel 1 color	pel 2 color	pel 3 color	pel 4 color	1 0 pel 5 color
not used						

Each of the three pel fields pairs contain either 1111 or 0000, where 1111 indicates two pels that are part of the character image.

Phase code F_{16} ("Right")

Eight consequtive reads in this phase provide the information for the right half of the character image. The returned words are formatted identically to those for the left half of the character except that pel fields 4 and 5 will _always_ contain 0000. This is because the right two pels represent the space between characters.

15 12	pel 0 color	pel 1 color	pel 2 color	pel 3 color	pel 4 (00)	1 0 pel 5 (00)
not used						

3. Bit Map Normal mode

This mode consists of four phases. Together, they provide pel pattern information to display characters 6 pels wide (which includes 1 pel for a space). Each word of pel pattern contains a line two thirds of the width of a character. Thus, three words of pel pattern will span two characters, and the center word will contain the right ⅓ of the first character and the left ⅓ of the second character. The programmer must do the combining of character thirds by ORing the pel pattern words containing those two portions. Since this is for the Bit Map display mode, one must remember that words describing horizontal lines of pels are consecutive in memory, whereas words describing vertical lines of pels are ~~spaced~~ spaced 60 words apart.

Phase code 1 ("Left ⅔")

The eight words of pel pattern information returned by this phase is formatted as required by Bit Map background display mode:

			3 0
pel x color	pel x+1 color	pel x+2 color	pel x+3 color

The pels represent the left ⅔ of the requested character's image. Each 4-bit field contains 1111 or 0000, where 1111 represents a portion of the character.

Phase code 9 ("Right ⅓")

The eight words from this phase represent just the right ⅓ of the requested character's image. The remainder of each word represents a space or null (which are identical). They are formatted as follows:
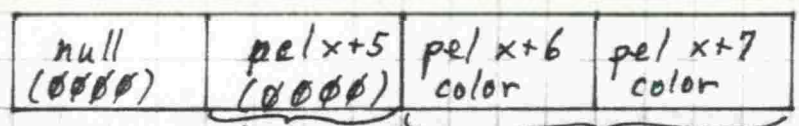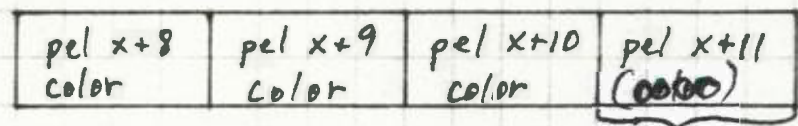
		3	0
pel x+4 color	pel x+5 (0000)	null (0000)	null (00000)

char. rt. ⅓   space

Pels representing a portion of the character have color code 1111.

Phase code 19₁₆ ("Left ⅓")

The eight words from this phase represent just the left ⅓ of the requested character's image. The remainder of each word represents a space or null. The format is as follows:

null (0000)	pel x+5 (0000)	pel x+6 color	pel x+7 color

space   char. lft. ⅓

Pels representing a portion of the character have color code 1111.

Phase code 11₁₆ ("Right ⅔")

The eight words of this phase represent the right ⅔ of the requested character's image. They are formatted as required by Bit Map background display mode:

pel x+8 color	pel x+9 color	pel x+10 color	pel x+11 (0000)

space

Pels representing a portion of the character have color code 1111.

pel pattern word pos ↓ words returned from read-back register

corresponding display segment

Phase = Left 2/3, ASCII code = A

word pos	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+0									1	1	1	1				
+60					1	1	1	1					1	1	1	1
+120	1	1	1	1												
+180	1	1	1	1												
+240	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
+300	1	1	1	1												
+360	1	1	1	1												
+420																

Phase = Right 1/3, ASCII code = A

word pos	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+1																
+61																
+121	1	1	1	1												
+181	1	1	1	1												
+241	1	1	1	1												
+301	1	1	1	1												
+361	1	1	1	1												
+421																

Phase = Left 1/3, ASCII code = B

word pos	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+1									1	1	1	1	1	1	1	1
+61									1	1	1	1				
+121									1	1	1	1				
+181									1	1	1	1	1	1	1	1
+241									1	1	1	1				
+301									1	1	1	1				
+361									1	1	1	1	1	1	1	1
+421																

Phase = Right 2/3, ASCII code = B

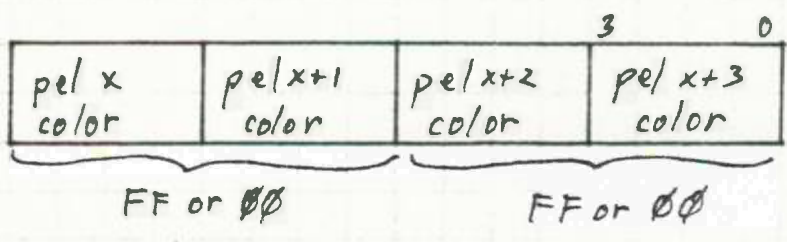| word pos	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+2	1	1	1	1	1	1	1	1								
+62									1	1	1	1				
+122									1	1	1	1				
+182	1	1	1	1	1	1	1	1								
+242									1	1	1	1				
+302									1	1	1	1				
+362	1	1	1	1	1	1	1	1								
+422																

4. Bit Map Double Width mode

This mode consists of three phases. Together, they provide pel pattern information to display characters 12 pels wide (which includes 2 pels for a space). Each word of pel pattern information describes a line one third of the width of a character. Three words will describe a line one character wide. As in all Bit Map displays, words describing horizontal lines of pels are consecutive in memory, and words describing vertically adjacent pels are spaced 60 words apart.

Phase code 5 ("Left Third") ~~(scribbled out)~~

Each of the eight words provided by ~~each of~~ this ~~the three three~~ phase has the following format:

pel x color	pel x+1 color	pel x+2 color	$\overset{3}{\text{pel}}$ x+3 $\overset{0}{\text{color}}$

FF or 00 (under first two)   FF or 00 (under last two)

Pel pairs representing a portion of the character will have combined color fields of hex FF.

Phase code 15_{16} ("Middle") – similar to above

Phase code D_{16} ("Right Third")

Each of the eight words provided by this phase has the following format:

pel x+8 color	pel x+9 color	pel x+10 (0000)	pel x+11 (0000)

FF or 00 (under first two)   space (under last two)

Moving Object Pel Patterns and Display Priority

Images of moving objects are composed of pels within a 16×16 movable grid of pels. The pels defining the object are differentiated from neighboring pels within the grid by their ~~color~~ "opacity" (pels outside the object are made transparent). RBS 1 through RBS 3 in MAGIC and portions of memory (called "pel patterns") define the images of the 16 moving objects. The registers in RBS 3 contain pointers to each of the pel patterns. MAGIC adds these pointers to the Moving Object Base Address in register 0 of RBS 7 to get the beginning address of each pel pattern (see description of that register for details). Given appropriately coded pel patterns, each moving object may be displayed in one of two modes: Low Resolution or High Resolution. When the value of the Hi-Res bit in the moving object's parameter register in RBS 1 is "0", the moving object's display mode is Low Resolution ("Lo-Res"). When it is "1", the display is High Resolution ("Hi-Res").

Lo-Res mode

Pel patterns in the Lo-Res mode allow for two or three choices of color (and only color) for each of the pels forming the moving object image. Thirty-two words comprise each Lo-Res pel pattern, two words per line of the 16×16-pel grid, starting at the top line. Because of the way MAGIC steps through it, the pel pattern must begin on a 32-word boundary. The format for each two words of pel pattern is as follows:

	15	14											2	1	0	
word 0	0														0	MSB of Color
word 1	0														0	LSB of Color

└ color code for left pel   └ color code for right pel

Corresponding bits from both registers form a 2-bit color code for each pel. The following table maps the color code:

color code	color assigned
00	"Absent" color
01	Color 1 if INVIS = $\emptyset$
	"Invisible" color if INVIS = 1
10	Color 2
11	Color 3

Colors 1, 2 and 3 are those having indices in like-numbered fields of the moving object's palette in RBS 2 (i.e. Color 1 refers to the Master Palette color whose index is in field 1 of the moving object's Palette in RBS 2).

The "Invisible" color is available as a result of the INVIS bit being ON in the moving object's Parameters register in RBS 1. It is transparent (i.e. colors behind it show through), but it can interact. The "Absent" color is also transparent, but it cannot interact (it "isn't there"). The Absent color is used for "carving away" portions of the 16×16 grid which do not form part of the moving object's image. The Invisible color is useful for extending small speedy objects, such as bullets, to assure interaction detection although the "contact" may have occurred between frame displays.

"wire frames"

Hi-Res mode

Pel patterns in the Hi-Res mode allow for four choices of color for each _pair_ of pels and for several levels of luminance for each pel. The way the luminance information is expressed depends on whether the pel is an Edge pel or a Body pel. A Body pel has its BODY bit equal "1", and its luminance code modifies the color's normal luminance Y that is specified by its Y/BY/RY chrominance code. An Edge pel has its BODY bit equal "0", and its luminance code results in various ratio's of blending of the luminance of the pel with the luminance of the pel "behind" it.

Sixty-four words comprise each Hi-Res pel pattern, four words per line of the 16x16-pel grid, starting at the top line. Due to the way MAGIC addresses each word, the pel patterns must start on a 64-word boundary. The format of set of four words of pel pattern is as follows:

word 0 — MSB of Luminance
1 — LSB of Luminance
2 — BODY bit
3 — Color

color code for left pel pair

luminance code for left-most pel

The following table maps the color code for each of the 8 pel _pairs_:

color code	color assigned from Palette
00	Color 0
01	Color 1
10	Color 2
11	Color 3

The following tables map the luminance code for each pel:

Body pel (BODY = 1):

luminance code	range of Y value of pel's color				
	4-6	7-9	10-12	13-14	
00	0	0	0	0	value added to pel's Y value
01	+2	-2	-4	-6	
10	+4	+2	-2	-4	
11	+6	+4	+2	-2	

Edge pel (BODY = $\emptyset$):

luminance code	luminance assigned
00	"Absent" luminance
01	$\frac{1}{4}Y + \frac{3}{4}Y'$ if INVIS = $\emptyset$
	"Invisible" luminance if INVIS = 1
10	$\frac{1}{2}Y + \frac{1}{2}Y'$
11	$\frac{3}{4}Y + \frac{1}{4}Y'$

where Y' is the Y value in the chrominance code of the "rear" pel.

The Absent and Invisible luminances are analogous to the Absent and Invisible Colors in Lo-Res mode and they serve the same purposes for Hi-Res. The "rear" pel is a pel that is being "covered" by an Edge pel of a Hi-Res moving object. It may be part of the background or part of a moving object of a lower priority level.

Priorities

When one moving object encounters the image of another moving object on the screen, the moving object having the higher priority level passes "in front" of the other. Or when a moving object with priority level higher than ~~that of~~ that of a certain background color encounters that background color, the moving object passes "in front" of that background color. This means that the pels with higher priority level are displayed instead of pels of lower priority level whenever MAGIC must make a choice. For moving objects, all pels of the object have a priority level equal to the number of the moving object, the higher number having the higher priority, and thus the terms "front" pel and "rear" pel refer to pels of higher and lower priority levels, respectively.

Scale Factor

The size of a moving object's image on the screen may be increased or diminished without altering or switching pel patterns. The facilities for doing this are the X and Y Scale Factor fields in the Moving Object Parameters registers of RBS 1. MAGIC will repeat or delete certain lines or pels of the image to expand or shrink it by a scale factor which may range between 0.5626 and 8.0 in 32 unequal steps. The change of size may be in the X (horizontal) or Y (vertical) directions or both at once. By varying the scale factor in time, one may cause the moving object to "approach" or "recede" on the screen.

Each Scale Factor field is composed of two subfields called SIZE (2-bits) and ZOOM (3-bits):

SIZE	ZOOM

By entering values from these two subfields into the following formula, one may calculate the scale factor:

$$SF = (z + 9)\, 2^{s-4}$$

where
   SF is the Scale Factor
   Z is the ZOOM value
   S is the SIZE value

(2-bits) SIZE	(3-bits) ZOOM	Scale Factor		No. of pels in one dimension
3	7	8.0		128
3	6	7.5		120
3	5	7.0		112
3	4	6.5		104
3	3	6.0		96
3	2	5.5		88
3	1	5.0		80
3	0	4.5		72
2	7	4.0		64
2	6	3.75		60
2	5	3.5		56
2	4	3.25		52
2	3	3.0		48
2	2	2.75		44
2	1	2.5		40
2	0	2.25		36
1	7	2.0		32
1	6	1.875		30
1	5	1.75		28
1	4	1.6.25		26
1	3	1.5	2	4
1	2	1.37.5		22
1	1	1.25		20
1	0	1.125		18
0	7	1.0		16
0	6	.9375		15
0	5	.875		14
0	4	.8125		13
0	3	.75		12
0	2	.6875		11
0	1	.625		10
0	0	.5625		9

Word Addressing

The Motorola 68000 addresses bytes of memory. That is, it assumes consecutive bytes differ in address by one and that consecutive words differ in address by two. MAGIC, on the other hand, addresses words of memory. It assumes that consecutive words differ in address by one. The programmer can access MAGIC registers on a byte basis, treating them as words in memory, but when supplying addresses to MAGIC (or data that MAGIC will use to construct addresses), the programmer must be sure to allow for MAGIC's method of word addressing.

The fields affected by this consideration are:

1) Pel Pattern Pointers (both moving object and background)

2) Pel Pattern Base Addresses ("   "   "   ")

3) Background Address

4) Alpha Table Address

When supplying data for these fields, the programmer need only first shift the data right one bit position within the field before writing it into the MAGIC register or entering it in a Background Table entry.

Screen Splits

Screen splits are for the purpose of dividing Alpha and/or Graphic displays between two screens so that each screen may contain information not available on the other. The programmer controls this feature through the fields of register 11 in RBS 7. Normally, this entire register would be zeroed out, and the primary and secondary screens would have identical displays. The following examples illustrate the extents of the Alpha and Graphic displays on both screens when various combinations of fields in register 11 are set to non-zero. An "A" and an arrow represent the extent of Alpha display, and a "G" and an arrow represent the extent of Graphic display. The Horizontal Split flag is set (HOR=1) in all examples. Otherwise, all unlisted register 11 fields are assumed to contain zero.
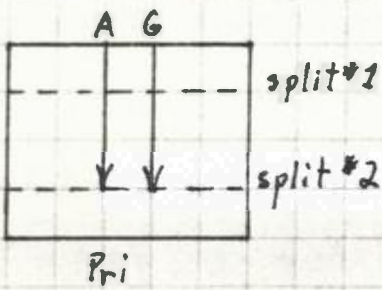
Normal

Pri   Sec

ASEN=1
GSEN=1

split #1
split #2

Pri   Sec

ASEN=1   ASRV=1
GSEN=1

split #1
split #2

Pri   Sec

ASEN=1   AOEN=1
GSEN=1   GOEN=1

split #1
split #2

Pri   Sec

ASEN=1   AOEN=1   ASRV=1
GSEN=1   GOEN=1

split #1
split #2

Pri   Sec

ASEN=1   AOEN=1

split #2
split #1

Pri   Sec

A S E N = 1
GSEN=1

split 1&2

Pri   Sec

Best Image W/Hi Res Cards Mode.

16 Moving Objects On One Line
    Hardware Capability
    CPU Limitations

1 Moving Object Any Color Move Against Rainbow Of Background.

Check Antialiasing Of Moving Object.

Hi Res Cards Mode - Move Objects (lines) Relative To Card Locations.

Interactive Input Device.

Superimpose Alpha On Screen Of Background & Zoom Background With Moving Objects That Are Moving.

Demonstrate Moving Object Zoom.

Demonstrate Animation (With Antialiasing).

3 Dimensional Objects - Rotations.

3 Dimensionalty W/Shading, Etc.

Interactions

Split Screen

Borders

Scrolling In Larger Field.

Square Pixel - Rotation Of Object.

Normal Cards Mode.

Bit Map Mode.

ADDED BY DAVE CHANDLER:

Split Screen
    A La Auto Race
    Alpha, Graphics

Separate TV'S

This is a very rough draft describing the software required to evaluate MAGIC.

cc: Dave Chandler
     Dave Hostetler
     Gary Lorenc
     ~~Barry Humphrey~~
     Jason Soo

1. Display a background scene in high resolution cards mode that uses a large number of colors.

   This should give us some feel for the picture quality that we can expect to see in the new unit.

2. Have a moving object under user control (graphics tablet or ~~hand controller~~) *or keyboard* and move the object over the background that was created above.

   This will show how well antialiasing works with moving objects and different colors.

3. Animate the moving object above. Test antialiasing again.

4. Draw several slanted parallel lines across the background so that they are not an integer number of card widths apart. See if there are any noticable bulges in the lines caused by the color limitations in high resolution cards mode.

5. Add external video to the system. This is just to show if we have mechanized this properly.

6. Place 16 moving objects on one horizontal scan line. This is to make sure that Magic has the time to handle all of the moving objects.

7. Try to reuse each moving object several times. This should show our maximum apparent moving objects and should give us a feel for how many we really need. This will also test the programmable interrupt.

8. Reuse the moving objects while changing the color base for the moving objects. This increases the amount of work to be done by the processor. Once again, we can find out what our limits are.

9. Perform a two-dimensional rotation on a moving object. This will show us if square pixels will do what we hope they can for us.

10. Rapidly change the background through a sequence of several pictures. This should give us a feel for the band width of the RAM.

11. Try to draw some good looking three dimensional objects. See if shading gives us the effect that we want.

12. Zoom a moving object in the X and Y direction, the X direction only, and the Y direction only. See if it looks good enough.

13. Use the split screen option to display half alpha and half graphics.

14. Send different displays to separate t.v's.

15. In addition to the above tests, we need to be sure that software
    is developed to facilitate hardware debug. Every feature must be
    tested. A few of these features are listed below:

              - Normal Cards Mode
              - Total Color Spectrum
              - Interaction (If Done In Hardware)
              - Scroll (Both Full And Half Screens

ADDITIONS:

KM/bb

From: Mike Spak   Date: June 15, 1982

Mail Drop: M2880

Phone: 6624

To: B.Wieder   Subject: Mattel Custom (MAGIC)
                                     Die Size Estimate

cc: G.Daniels
             S.Groves
             T.Gunter
             D.McAlister
             H.Scales
             G.Schriber
             G.Walker

The initial die size estimate for Mattel's custom design,
which is referred to as MAGIC (Mattel Advanced Graphics
Interface Circuit), has been completed. The information for
the estimate has been accumulated over approximately three
days of engineering meetings with Mattel and from incomplete
schematics of the emulator which Mattel is currently
designing. This estimate is 87,600 square mils, which is
equivalent to a 296 x 296 mil die. It corresponds to a
rough transistor count of 54,000 devices. The size is
referenced to an "as drawn" dimension according to HMOS
layout rules with a minimum channel length of 3.5 microns.
This estimate is made with the understanding that not all
information (in the form of schematics or detailed block
diagrams) has been received from Mattel and, therefore,
could increase if any additional functionality is deemed
necessary. However, the estimate is submitted with the
thought that it accounts for all functionality. The
"window" feature is not included in this estimate since it
is not clear what exact implementation is desired.

Sufficient information does exist to make an accurate
estimate despite the the fact that detailed design is still
ongoing. This estimate reflects the particular
architectural design which Mattel has explained. Little
deviation has been made from their functional description
except where thought was given to ease MOS layout or
implementation peculiar to MOS technology. For any "black
box" function (ie., where PROM's are used to minimize

- 1 -

package count in the emulator) time was taken to derive a first pass logical MOS implementation to make a more accurate transistor count. This engineering estimate has been derived prior to actually doing any detailed MOS logic design. Estimates are normally refined on two occasions prior to actually beginning extensive layout design. Refinements occur when the logic diagram is complete after various timing problems are more thoroughly considered and then by layout designers in the process of doing detailed chip planning just prior to actually starting transistor layout.

The methodology used in the estimate was to accumulate active die area based on actual transistor count. This has been done in this case since no previous layout exists from a prior chip development. The transistor count for each particular function is then multiplied by a density factor to reflect how compact the function can be layed out in HMOS. This cumulative active area is then added to a routing overhead which is determined by multiplying a portion of the active area by a certain factor. Large areas such as a general purpose RAM or a large shift register structure have been assumed to not contribute to the routing overhead since they are a fairly closed structure and the functional circuitry around the large structure is well defined and contributes its own routing overhead. Attached you will find a listing of various functions and the compiled transistor count and active area that is believed associated with that function. More detail can be provided as needed. To this cumulative figure is then added the area requirements for the I/O buffers corresponding to the particular pin configuration and for internal power bussing which is typically associated with this size of chip.

The problem with trying to make a realistic estimate with this type of architecture, where there are roughly 130 addressable registers by the CPU which contain control and data information for distribution throughout the die, is that excessive routing can blow out the chip size. That is why a good layout plan for this chip is considered critical in order to minimize routing. The routing factor and density factors used in this estimate are ones typically found in good layouts.

I welcome and encourage the opportunity to meet with Mattel to go over the estimate in detail for the purpose of making sure there are not any errors and that all circuitry is accounted for in the estimate.

Sincerely,
Mike Spak

Die Area & Transistor Count vs. Function

** Area does not contribute to routing overhead.

	# xtrs	(mil^2) area
1. Video Generator		
a. YADJ ckt. w/ PLA	490	726
b. Digital color filter + delay	700	1232
c. Blue/red adder w/excess 8 adj.	525	893
d. DAC input logic & mux	117	225
e. Color palette ram + decoders	1400	1103 **
sense amps	192	369
SUBTOTAL	3430	4550
2. Address Generation + Interrupt		
a. Line interrupt + IPL2 logic	179	252
b. AU, output latch, temp reg	830	1138
c. Table ptr, base, excess reg	1728	1731
d. Decoders, drivers, misc logic	530	748
e. Bus precharge + control	160	205
f. M.O. barrel shift + control	108	207
SUBTOTAL	3540	4280
3. CPU Interface + DRAM Control		
a. DTACK wait reg + related	388	460
b. 68000 interface + addr PLA	253	295
c. Clock generation	155	1596
d. RAS/CAS + refresh ctr	316	481
SUBTOTAL	1110	2830
4. Horizontal/Vertical Timing		
a. Vert ctr + PLA detect	590	672
b. Horiz ctr + PLA + random logic	719	975
c. Vert/horiz scroll + contr bits	944	1303
d. Vert/horiz split + contr bits	676	1094
e. Border control bits + logic	747	1064
SUBTOTAL	3680	5110
5. Background Generator		
a. Buf shift register (1280 bits)	7680	4792 **
b. Shift mux + I/O contr	411	749
c. FIFO + output mux logic	508	786
d. Data bus reg + decode	106	110
e. Bgnd priority bits + logic	182	256
SUBTOTAL	8890	6690

6. Alpha Generator

a. Vert ctr + PLA detect	590	672
b. Horiz ctr + PLA + random logic	719	975
c. Vert/horiz scroll + contr bits	944	1303
d. Vert/horiz split + contr bits	676	1094
e. Border control bits + logic	747	1064
SUBTOTAL	3680	5110

5. Background Generator

a. Buf shift register (1280 bits)	7680	4792 **
b. Shift mux + I/O contr	411	749
c. FIFO + output mux logic	508	786
d. Data bus reg + decode	106	110
e. Bgnd priority bits + logic	182	256
SUBTOTAL	8890	6690

6. Alpha Generator

a. Char rom + decode/sense amps	2216	1557 **
b. Char rom in addr + shift bits	17.2	255
c. Buf shift register (640 bits)	3840	2396 **
d. Shift mux + I/O contr	100	192
e. Readback bits + encoder	291	450
SUBTOTAL	6620	4850

7. Moving Object Generator

a. One moving object in array		
(1) Shift register (80 bits)	560	408
(2) Reg bits + decode/compare	470	512
(3) Clock/priority/out logic	240	310
(4) Routing overhead = .2x310	-	60
SUBTOTAL (per object)	1270	1290 **
SUBTOTAL (16 objects)	20320	20640 **
b. Data Encoder	260	540
c. Y zoom/mirror AU + logic	278	415
d. Interaction matrix	4222	2945
SUBTOTAL	25080	24540

8. I/O Pin Buffers   1470   23502

TOTAL ACTIVE AREA	52350	52850	
ROUTING (.3 x 22370)	-	6710	
TOTAL I/O + PADS	1470	23502	
FINAL TOTAL	53820	83060	(288x288)

Estimate with 8 mils internal power bussing =   296 x 296

handwritten notes: 786 ; ←700 ; 12 moving objects → 260x260 mils

HIGH RESOLUTION CARDS MODE   10/30/81

Note: References to this problem are on pages in my notebooks dated page before 10/30/81, 11/17/81, 12/2/81 and several pages 12/17/81.

The NTSC composite color video signal takes advantage of the eyes limited (compared to luminance) ability to resolve color. To this end the chroma components are limited to aproximately 0.5 MHz while the luminance uses about 3 MHz. These are bandwidths that are currently realized in practice; the standards allow slightly greater bandwidths. If I might make an analogy, the results are very much like using a fine brush to paint a detailed picture in shades of gray and then using a much broader brush to add the color. On a TV again, even though the color information is lacking detail, the picture looks sharp and clear because the uses the luminance to decide edges and detail. When the color spills over the edges and smears the eye ignores it, thinking that it is correcting its own problems and not those of the TV.

What has all this to do with TV games? As we improve the resolution of the TV games, the pixel size gets smaller than the chrominance bandwidth can pass. Thus, if each of the pixels is a different color much of the information that is stored and sent to the TV is lost in the process of being converted into and out of the NTSC signal. If we can make a data structure that stores the color information in less detail and the luminance information in greater detail, we can optimize the data storage requirements and better match the information carrying ability of the NTSC (and PAL and SECAM) signals. All of this adds up to better pictures at less cost.

The basic plan is to group the pixels and give a color to each group. To put the detail in, the luminance level of each pixel will be specified to add shadows and highlights, perform antialiasing and define the edges for the eye. Since we are going to arbitrarily group the pixels, there are going to be cases where more than one color is present in a group. In this case the color assigned to the group will be that of the majority (except in a few special cases, primarily involving narrow lines against a different color background). How does the eye know that the few pixels are of a different color? Very likely, there will be other pixels of the same color next to, above or below them. The plan is to make their luminance levels match that of their friends and the eye will do the rest. The number of pixels grouped together should bear some relationship to the color resolution; certainly, it would be wasteful to use more than

twice the resolution and using less than the color resolution will degrade the picture. The exact ratio will likely be determined some other system requirements.

A specific implementation of this method is the High Resolution Cards Mode proposed for MAGIC (Mattel Advanced Graphic Interface Circuit). For MAGIC the normal resolution cards mode will consist of 24 rows each 40 cards long. Each card will contain 8 lines of 6 pixels each. Each card will be capable of displaying 1 of 4 colors at any pixel location; these colors will be 4 out of 16 stored in the color palette RAM map. Since we will be using a 16 bit wide data bus to the RAM, it will be easiest to get 16 bits for each line of each card. Each line of a card is 6 pixels wide; the 16 bits are taken two at a time to select 1 of the 4 colors from the card color map. For 6 pixels this only requires 12 bits leaving 4 bits unused.

In High Resolution Cards Mode, we find a use for those extra 4 bits. By grouping the pixels into two groups of three, the color for both groups can be set using the extra 4 bits. This grouping results in a chrominance sample rate approximately four times the NTSC chrominance bandwidth. The remaining 12 bits in the 16 bit word are used, at two bits per pixel, to modify the luminance level. In MAGIC the output DAC has four bits of input and hence 16 levels output. Eleven of these are used for gray scale. The two bits assigned for luminance adjustment define four adjustment levels. One of these is no change from the luminance specified in the color palette RAM map. The remaining codes functions depend on the luminance level specified in the color palette (4 through 14 are the codes for the 11 gray scale levels):

Level   Adjustment Available (gray scale levels)
4 to 6   +2, +4, +6
7 to 9   -2, +2, +4
10 to 12   -4, -2, +2
13 and 14 -6, -4, -2

In this specific case the technique uses the same amount of data more efficiently, allowing better pictures; in other cases the amount of data for the same picture quality could be reduced. While in this case a relatively complex method is used to adjust the luminance level, other methods including, but not limited to, fixed adjustments up and down and absolute luminance specification could be used.

A software/hardware simulation of the High Resolution Cards Mode as described above has been completed with excellent results.

VERTICAL COUNT

525 LINES

	FIELD 1	FIELD 2	
	FIELD 1	FIELD 2	
VERTICAL BLANKING	21	21	
TOP BORDER	26	25	
ACTIVE PICTURE	192	192	
BOTTOM BORDER	24	24	

625 LINES

FIELD 1	FIELD 2	FIELD 1	FIELD 2
256			256
336			336
337			337
324			311
0			512
15			513
16			528
207			319
208			320
255			143

HORIZONTAL COUNT

	HORIZONTAL BLANK	RIGHT BORDER	ACTIVE PICTURE	LEFT BORDER
14 MHz	322	396 397	409 0 1516	255 301
17 MHz	322	396 397	453 0 1516	255 256 321