V9938 MSX-VIDEO

Technical Data Book Programmer's Guide

ASCII CORPORATION / NIPPON GAKKI CO., LTD.

YAMAHA

NIPPON GAKKI CO., LTD.

This page is intentionally left blank

NOTE FROM THE EDITOR

This important technical book – including preface, dated back to 1985, had been reworked and modified appropriately in order to give more detailed explanation of how V9938 chip works and how to effectively program the system based on it.

While MSX gave up to the IBM PC standard as a computing platform in early 1990s, it is still one of the powerful specimens in the field of 8-bit home entertainment and hobbyist platforms. It is argued that modern PC architecture origins at the MSX standard; and it is clear that initiative, taken by Microsoft and ASCII by institutionalizing software and hardware compatibility was taken in further developments on the IT market.

V9938 is one of the most popular video-processors in the family of Texas Instruments original chips, improved by a group of companies from Japan and US. Its follower, V9958, was a VDP chip for limited release of MSX2+ and MSX Turbo-R machines, and the last VDP in the family, V9990, can only be found in add-on cartridges.

The book is provided in PDF format and is searchable to allow internet search engines indexing it so that potential readers would easily find it.

December, 2010
Eugeny Brychkov

Rev. 1.00 (Jan 06, 2011): Initial release
Rev. 1.01 (Jan 27, 2011): http://www.msx.org/forumtopic12176.html per Daemos et al.
Rev. 1.01a (Mar 06, 2011): http://www.msx.org/forumtopic12176.html per ARTRAG et al.

This page is intentionally left blank

PREFACE

The V9938 introduced in this manual is a Very Large-Scale Integrated Circuit (VLSI) that was developed as a video Display Processor (VDP) for the MSX2. The MSX personal computer standard was introduced in 1983 by ASCII Corporation and Microsoft Incorporated. At present, the MSX is manufactured and marketed worldwide. In order to strengthen some of the functions of the original MSX, the MSX2 standard was developed in 1985. In addition to being software-compatible with the MSX, the MSX2 supports new media and has video processing capabilities that are not available on conventional 8-bit personal computers.

To make the MSX2 a reality, two requirements for the Video Processor were: upward compatibility with the existing TMS9918A (the VDP for the MSX) software and increased number of functions. The V9938 was developed through the joint efforts of ASCII Corporation, Microsoft Incorporated, and YAMAHA.

The following functions are supported on the V9938:

- Full bit-mapped mode
- 80-column text display
- Access using X- and Y-coordinates for easier programming; the X-Y coordinates are independent of the screen mode
- Fundamental hardware commands to decrease the processing time and reduce programming complexity
- Digitize and external synchronization
- Color palette (9 bits x 16 patterns)
- Linear RGB video output
- More sprites per horizontal line

Because the V9938 has the above functions, it provides superior video capabilities that make it possible for its use in a variety of applications, including the MSX2. CAPTAIN terminals and NAPLPS terminals using the V9938 have already been developed. We hope that the V9938 will be a standard video processing device on a worldwide basis.

This manual was written to explain how to set parameters of the v9938 and is a reference for developing applications and system software for it.

We are pleased that you have chosen to develop software for the V9938 and that you have referred to this manual for assistance.

Finally, we would like to express our deep gratitude to the people at NTT as well as other related manufacturers for their valuable opinions which contributed to the development of the V9938.

August, 1985
ASCII Corporation

This page is intentionally left blank

CONTENTS

DEFINITIONS

A

Attribute

The property of an object, which controls how object looks like on the screen. Attribute can be a color, position of an object, or control which pixel should have specific color

B

Background

Is an *object* or *property* which is perceived to be in the background to another property or object. For example, for a character displayed on the screen the pixels of its image is said to have foreground color, while other pixels to have background color. In case of sprites, they may be said to appear in the foreground to the font patterns, as sprites overlap images of the font. See also *Foreground*

C

Collision

Sprites are said to collide when their dots having color code 1 (simply saying – dots identified with binary 1s in their sprite pattern generator table) overlap. In some circumstances such behavior may be changed in favor or mixing sprite colors to have pseudo-multi-colored sprites

Color

A property of the pixel on the screen. Color of the pixel may come from various sources: from global color register, from pattern color table or from sprite color table. Colors can also be coded in the palette registers through setup of red, green and blue components – in this case, if color table of the patterns and sprites remain unchanged, actual colors, associated with them, may be different

Command

A special sequence of VDP operations, a kind of hardware acceleration. Command is expected to streamline CPU-VDP-VRAM operations, unload CPU and increase data transfer speed.

E

Expansion RAM

This random access memory is used to store non-displayed data or register information, and is not necessary for proper operation of the VDP. Maximal size or expansion RAM is 64K bytes. Due to specific purposes of this RAM, it is rarely used in applications.

L

Layout

A map of patterns or sprites which identify where to display specific object or which object should be displayed in specific position. In case of patterns (font), Pattern generator Table identifies the appearance of font, but in order to display these patterns in specific position, programmer should put its number into Pattern Layout Table in respective location

O

Object

Font patterns or a sprite.

P

Pattern

A property of an object identifying how object looks like. Object can be a font pattern or a sprite pattern. Pattern may be represented by one byte or 8 bytes in different modes; its contents codes colors of the pixels displayed. For some modes, 8 bits (one byte) can code 8 pixels – 0 for background color and 1 for foreground color, in other modes 8 bits can code 4 pixels – with colors 0, 1, 2, and 3. To display font patterns on the screen it may not be enough to set its *pattern*, but also put its pattern number into the *layout* map. VDP, when displaying the picture, reads the number of pattern to display, and then refers to its actual image to font pattern generator table

Port

Is a physical latch with specific system address for CPU reads and writes to communicate with VDP. VDP has four ports, port #0 is a read/write data port, port #1 is write register set-up port, port #2 is write palette port, and port #3 is write register data port

R

Register

Register is a static place within VDP for control information about VDP's mode, screen property etc. Registers can be Status, Video or Command. They can be accessed directly or indirectly. Access to the registers is made through consecutive writes to specific VDP *ports*, thus in order not to break the order and thus successful completion of specific operation, programmer should disable interrupts.

RGB

Abbreviation for base colors Red, Green and Blue. It may be used to describe hardware wiring with three analog signals; or to describe a color composition for the pixel. Note that when coding RGB in VDP palette registers programmer uses 3 bits for every base color, and when coding RGB in GRAPHIC 7 mode red and green occupy 3 bits each, but blue occupies only 2 bits

T

Tile

Same as *font pattern*

V

VRAM

Video Random Access Memory is a set of memory cells used by VDP to keep information about picture displayed on the screen. VRAM is accessed for picture displaying purposes as well as for picture modifications. Picture displaying occurs continuously when VDP is enabled. V9938 may have 16K to 128K VRAM, and depending on the memory organization and size may not be able to function properly in specific modes. See description of register R#8's bit VR for more information

This page is intentionally left blank

1. BASIC INPUT AND OUTPUT

1.1. Accessing the Control Registers

V9938 has 4 ports: port #0 – port #3; port number is selected by VDP address lines A0 and A1. Table below also shows port address allocation for MSX compatible machine.

	A1	A0	Operation	Primary MSX port (Hex)
Port #0	0	0	VRAM Data (R/W)	98h
Port #1	0	1	Status Register (R) VRAM Address (W) Register set-up (W)	99h
Port #2	1	0	Palette registers (W)	9Ah
Port #3	1	1	Register indirect addressing (W)	9Bh

There are two ways to set data in the MSX-VIDEO control registers (R#0 to R#46).

1.1.1. Direct access to VDP registers

Output the data and the register number in sequence to port #1. The order of reads and writes to/from VDP ports is vitally important, thus you should keep in mind that this order can be potentially interrupted by CPU interrupt routine which can write to or read from VDP port(s) and thus break the proper sequence. In case of Z80 CPU, use **DI** (disable interrupts) at the start and **EI** (enable interrupts) at the end of VDP your access code.

Data byte is written first (bits D0-D7), and register number is written next to data byte (bits R0-R5). If interrupt involving VDP operations will occur between these two operations, it may cause unpredictable results.

MSB	7	6	5	4	3	2	1	0	LSB
Port #1 first byte	D7	D6	D5	D4	D3	D2	D1	D0	Data
Port #1 second byte	1	0	R5	R4	R3	R2	R1	R0	Register #

1.1.2. Indirect access to registers through R#17 (Control Register Pointer)

Set the register number in R#17 using direct addressing and then send data to Port #3. MSB of the value written to R#17 (AII) controls auto-incrementing of the register number. If auto-incrementing is enabled, after each data read or write control register pointer is incremented; if auto-incrementing is disabled then pointer value in R#17 remains unchanged. Auto-increment mode is useful for bulk read or update of VDP registers.

Note: data in register R#17 can not be changed by indirect addressing.

	MSB	7	6	5	4	3	2	1	0	LSB	
Register #17		AII	0	R5	R4	R3	R2	R1	R0		Register #

AII:
0: Auto-increment is enabled
1: Auto-increment is disabled

Port #3 first byte	D7	D6	D5	D4	D3	D2	D1	D0	Data
Port #3 second byte	D7	D6	D5	D4	D3	D2	D1	D0	Data
…									
Port #3 nth byte	D7	D6	D5	D4	D3	D2	D1	D0	Data

1.2. Accessing the Palette Registers

To set data in the MSX-VIDEO palette registers (P#0 to P#15) you must first set the palette register number in register R#16 (Color palette address pointer) and subsequently write two bytes of data (in specific order) into port #2. Every color consists of 3 sets of 3 bits: red, green and blue component (value 0…7).

Note: after writing pair of data to port #2 palette register number (pointer) in register R#16 auto-increments.

	MSB	7	6	5	4	3	2	1	0	LSB	
Register #16		0	0	0	0	C3	C2	C1	C0		Palette #
Port #2 first byte		0	R2	R1	R0	0	B2	B1	B0		Data 1
			Red data				Blue data				
Port #2 second byte		0	0	0	0	0	G2	G1	G0		Data 2
							Green data				

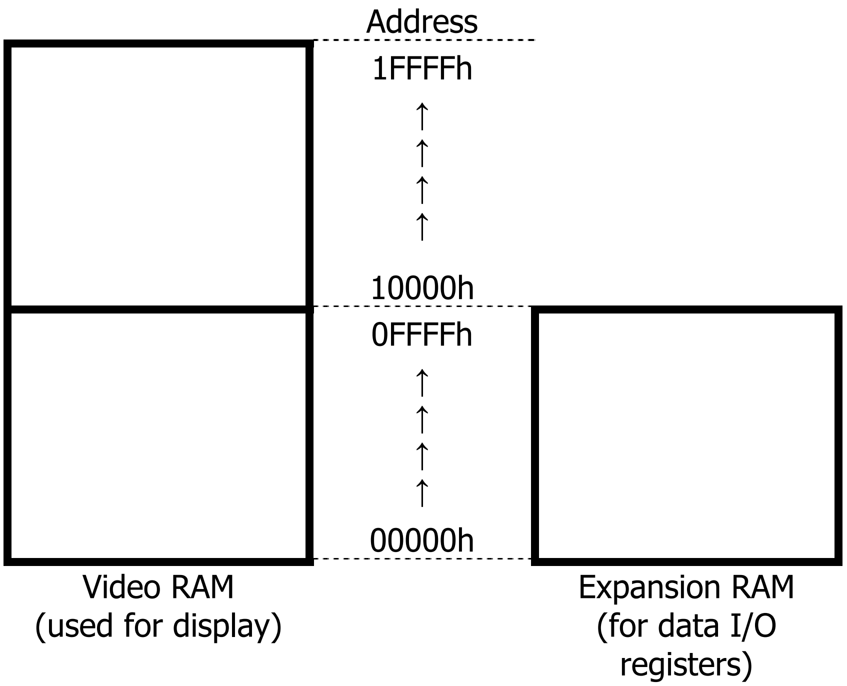
1.3. Accessing the status registers

To read the status registers of MSX-VIDEO (S#0 to S#9) you must first set the register number in R#15 (Status register pointer) and then read data from port #1.

	MSB	7	6	5	4	3	2	1	0	LSB	
Register #15 (write)		0	0	0	0	S3	S2	S1	S0		Stat Reg #
Port #1 data (read)		D7	D6	D5	D4	D3	D2	D1	D0		Data read

1.4. Accessing the Video RAM (VRAM)

A video RAM of 128K bytes plus an expansion RAM of 64K bytes can be attached to the VDP. Memory map is shown below.



To access memory, use the following procedure:

1. Switch respective bank (VRAM or expansion RAM)
2. Set the address counter A16 to A14
3. Set the address counter A7 to A0
4. Set the address counter A13 to A8, and specify if following data command will be read or write
5. Read or write data to the memory

<u>Step 1:</u> Switching banks (VRAM to expansion RAM)

Applications are used to work with Video RAM, thus re-specification of the bank is rarely necessary. It will be required if your application will need to access expansion RAM. After performing necessary operations on expansion RAM, ensure that you map the Video RAM back.

	MSB	7	6	5	4	3	2	1	0	LSB
Register #45		0	MXC	MXD	MXS	DIY	DIX	EQ	MAJ	Argument register (ARG)

0: Video RAM
1: Expansion RAM

Step 2: Setting the address counter A16 to A14

VDP can logically address 128K bytes in the address range of 00000h-1FFFFh through 16 address bits A16…A0. At this step we set up bits A16…A14 writing them into register R#14 (VRAM access base address register).

MSB	7	6	5	4	3	2	1	0	LSB
Register #14	0	0	0	0	0	A16	A15	A14	Base reg

Step 3: Setting the address counter A7 to A0

Set the low-order eight bits A7…A0 of the address counter by writing data to port #1.

MSB	7	6	5	4	3	2	1	0	LSB
Port #1	A7	A6	A5	A4	A3	A2	A1	A0	A7…A0

Step 4: Setting the address counter A13 to A8 and operation mode

Set the remaining six bits A13…A8 of the address counter by writing data to the port #1.

You also should specify which memory operation will follow – read or write. It is very important, as if you specify that next operation will be a "read", VDP will pre-fetch value from the memory (specified by the address set up earlier) and will get ready for CPU data read. If you will not do so and issue read command, VDP may not get enough timeslot to read data from the VRAM and CPU may get invalid data.

If you specify that next command will be a "write", then VDP does not do pre-fetch and waits for write instead.

MSB	7	6	5	4	3	2	1	0	LSB
Port #1	0	W	A13	A12	A11	A10	A9	A8	A13…A8

0: Next command is "Data Read"
1: Next command is "Data Write"

Step 5: Reading or writing data to memory

It is important to know that after every data read or data write operation from port #0 address counter is being incremented. It is very useful when you need to read from or write to the memory in sequential manner. However you should mind the timing, ensuring that VDP has enough time to write cached data or read requested data. Please refer to the data sheet for timings.

MSB	7	6	5	4	3	2	1	0	LSB
Port #0	D7	D6	D5	D4	D3	D2	D1	D0	Data Addr ++

2. REGISTER FUNCTIONS

2.1. Control registers

#0 to #23
#32 to #46

2.1.1. Mode registers

	MSB	7	6	5	4	3	2	1	0	LSB
R#0		0	DG	IE2	IE1	M5	M4	M3	0	Mode R#0
R#1		0	BL	IE0	M1	M2	0	SI	MAG	Mode R#1
R#8		MS	LP	TP	CB	VR	0	SPD	BW	Mode R#8
R#9		LN	0	S1	S0	IL	E0	*NT	DC	Mode R#9

* Indicates negative logic

R#0	DG	*Digitize mode:* sets the color bus to the input or output mode
	IE2	Enables interrupts from Light pen
	IE1	Enables interrupt from horizontal retrace
	M5	Screen mode flag (see Screen Modes chapter)
	M4	Screen mode flag (see Screen Modes chapter)
	M3	Screen mode flag (see Screen Modes chapter)

R#1	BL	Blank screen: if set to 1, screen display is enabled. If set to 0, screen display is disabled and no VRAM read operations are performed.
	IE0	Enables interrupt from vertical retrace
	M1	Screen mode flag (see Screen Modes chapter)
	M2	Screen mode flag (see Screen Modes chapter)
	SI	*Sprite size:* when set to 1, sprite size is 16*16. If set to 0, sprite size is 8*8
	MAG	*Sprite enlarging:* If set to 1, sprites are enlarged (double size)

R#8	MS	*Mouse:* when set to 1, sets the color bus into input mode and enables mouse. If set to 1, sets color bus into output mode and disables mouse
	LP	*Light pen:* when set to 1, enables light pen
	TP	Sets the color of code 0 to the color of the palette
	CB	*Color bus:* when set to 1, sets color bus into input mode. If set to 0, sets color bus into output mode
	VR	Selects the type and organization of VRAM. If set to 1, VRAM is 64Kx1Bit or 64Kx4bits. If set to 0, VRAM is 16Kx1Bit or 16Kx4Bits. Affects how VDP performs refresh on DRAM chips
	SPD	*Sprite disable:* if set to 1, sprites are not displayed and related VRAM reads are not performed.
	BW	Black/White: if set to 1, output is grayscale in 32 tones

R#9	LN	*Line:* if set to 1, vertical dot count is set to 212. If set to 0, horizontal dot count is 192
	S1	Selects simultaneous mode
	S0	Selects simultaneous mode
	IL	*Interlace:* if set to 1, interlace; if set to 0, non-interlace mode
	EO	*Even/Odd screens:* When set to 1, displays two graphic screens interchangeably by even/odd field; if set to 0, displays same graphic screen by even/odd field
	*NT	(RGB output only) If set to 1, PAL mode (313 lines, 50Hz); if set to 0, NTSC mode (262 lines, 60Hz)
	DC	*Dot clock:* If set to 1, *DLCLK is in input mode; if set to 0, *DLCKL is in output mode

2.1.2. Table Base address registers

When displaying information on the screen, VDP uses color, pattern, sprite and other information from video RAM. It is important to set proper starting addresses of such VRAM locations by writing to specified table base address registers.

Note: you should ensure that unused bits are set to 0. Further in the book bit set to "0" will mean that this bit has to be set to 0, "1" will mean that this bit has to be set to 1, and "*" will mean that value of the bit does not matter.

	MSB	7	6	5	4	3	2	1	0	LSB
R#2		0	A16	A15	A14	A13	A12	A11	A10	Pattern layout table
R#3		A13	A12	A11	A10	A9	A8	A7	A6	Color table low
R#10		0	0	0	0	0	A16	A15	A14	Color table high
R#4		0	0	A16	A15	A14	A13	A12	A11	Pattern generator table
R#5		A14	A13	A12	A11	A10	A9	A8	A7	Sprite attribute table low
R#11		0	0	0	0	0	0	A16	A15	Sprite attribute table high
R#6		0	0	A16	A15	A14	A13	A12	A11	Sprite pattern generator table

2.1.3. Color registers

Color registers are used to control MSX-VIDEO text and background screen colors, blinking and other functions.

	MSB	7	6	5	4	3	2	1	0	LSB	
R#7		TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0		Text and screen margin color
		Text color in TEXT1 and TEXT2 modes				Screen margin / backdrop color					

	MSB	7	6	5	4	3	2	1	0	LSB	
R#12		T23	T22	T21	T20	BC3	BC2	BC1	BC0		Text and background blink color
		Color part 1				Color part 0					

In TEXT2 mode, if attributes for blinking are set, color set in this register R#12 and in register R#7 are displayed alternatively (blinked).

	MSB	7	6	5	4	3	2	1	0	LSB	
R#13		ON3	ON2	ON1	ON0	OF3	OF2	OF1	OF0		Blinking period register
		Display time for even page				Display time for odd page					

In the TEXT2 mode and in bit map modes of GRAPHIC4 to GRAPHIC7 two pages can be alternatively displayed (blinked). Write to this register R#13 in order for blinking to start.

	MSB	7	6	5	4	3	2	1	0	LSB	
R#20		0	0	0	0	0	0	0	0		Color burst register 1
R#21		0	0	1	1	1	0	1	1		Color burst register 2
R#22		0	0	0	0	0	1	0	1		Color burst register 3

The above values of color burst registers are preset on power-on. If all the bits in all three registers are set to 0, color burst of the composite video is not performed. If values are returned to above values, VDP will start generating normal color burst signal.

2.1.4. Display registers

The display registers are used to control display position on the screen.

	MSB	7	6	5	4	3	2	1	0	LSB
R#18		V3	V2	V1	V0	H3	H2	H1	H0	Display adjust register

Register #18 controls horizontal and vertical alignment on the screen. Please refer to the table below.

	Value						
	7	…	1	0	15	…	8
H	Left	…	…	Center	…	…	Right
V	Top	…	…	Center	…	..	Bottom

	MSB	7	6	5	4	3	2	1	0	LSB
R#23		DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0	Vertical offset register

This register R#23 sets the value of the first line to display on the screen. Virtual screen size is 256 lines, visible vertical screen size can be 192 or 212 depending on LN bit of register R#9. Setting R#23 to value other than 0 may display un-initialized parts of the memory which may look as garbage. Display of virtual screen is performed in cycle, meaning that when increasing value of R#23 top of virtual screen appears at the bottom of visible screen. Please see pictures below.

Original screen* in GRAPHIC1 mode

Offset screen*, horizontal scan line is 256 dots, GRAPHIC1 mode, top appears at the bottom

```
20 out&h99,i:out&h99,&h97
30 a$=input$(1)
40 next i
run
Ok

run
█









color  auto   goto   list   run
èP Pè    èèÿh█°   °+ @° r  @  Γ
 +     @¿+        P°     MSX BASIC vers
Ion 2.I                  Copyright 1986
 by Microsoft
                         Disk BASIC ver
sion 1.0                 Ok
```

TEXT1 mode, horizontal scan line is 240 dots, garbage in un-initialized memory space

---

* “Auf Wiedersehen Monty” image, a property of Gremlin Graphics. Used in this book for educational purposes only

	MSB	7	6	5	4	3	2	1	0	LSB
R#19		IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0	Interrupt line register

VDP generates interrupt when it starts to display respective scan line if bit 4 "IE1" of register R#0 is set to 1. Write a value to this register R#19, and when VDP will start displaying the specified line, it will set bit 0 "FH" of status register S#1 to 1.

2.1.5. Access registers

Access registers is the set of registers used for accessing other VDP registers or VRAM. These registers include R#14, R#15, R#16 and R#17.

	MSB	7	6	5	4	3	2	1	0	LSB
R#14		0	0	0	0	0	A16	A15	A14	VRAM access base register

R#14 contains three senior bits of VRAM access address. In all modes, except GRAPHIC1, GRAPHIC2, MULTICOLOR and TEXT1, if there's a carry flag from A13 the value in this register is automatically incremented.

	MSB	7	6	5	4	3	2	1	0	LSB
R#15		0	0	0	0	S3	S2	S1	S0	Status register pointer

R#15 points to the respective status register (S#0…S#9) to be read.

	MSB	7	6	5	4	3	2	1	0	LSB
R#16		0	0	0	0	C3	C2	C1	C0	Color palette address register

R#16 points to the respective color palette register (P#0…P#15) to be accessed.

	MSB	7	6	5	4	3	2	1	0	LSB
R#17		AII	0	RS5	RS4	RS3	RS2	RS1	RS0	Control register pointer

R#17 is a register used in indirect access to other VDP registers. It also has auto-increment flag (AII) which is used to control increment of value in this register.

2.1.6. Command registers

The following command registers are used when executing a command on the MSX-VIDEO. Details on the use of these command registers will be presented in later chapter.

	MSB	7	6	5	4	3	2	1	0	LSB
R#32		SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	Source X low register
R#33		0	0	0	0	0	0	0	SX8	Source X high register
R#34		SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	Source Y low register
R#35		0	0	0	0	0	0	SY9	SY8	Source Y high register

	MSB	7	6	5	4	3	2	1	0	LSB
R#36		DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	Destination X low register
R#37		0	0	0	0	0	0	0	DX8	Destination X high register
R#38		DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	Destination Y low register
R#39		0	0	0	0	0	0	DY9	DY8	Destination Y high register

	MSB	7	6	5	4	3	2	1	0	LSB
R#40		NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	Number of dots X low register
R#41		0	0	0	0	0	0	0	NX8	Number of dots X high register
R#42		NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	Number of dots Y low register
R#43		0	0	0	0	0	0	NY9	NY8	Number of dots Y high register

	MSB	7	6	5	4	3	2	1	0	LSB
R#44		CH3	CH2	CH1	CH0	CL3	CL2	CL1	CL0	Color register
R#45		0	MXC	MXD	MXS	DIY	DIX	EQ	MAJ	Argument register
R#46		CM3	CM2	CM1	CM0	LO3	LO2	LO1	LO0	Command register

2.2. Status registers #0 to #9

The following status registers are read-only registers for VDP status reporting. Let's consider each register.

	MSB	7	6	5	4	3	2	1	0	LSB
S#0		F	5S	C	5SN					Status register 0

7	F	*Vertical scan interrupt flag.* When S#0 is read, this flag is reset
6	5S	*Flag for 5th sprite.* Five (or nine in G3…G7 modes) sprites are aligned on the same horizontal line
5	C	*Collision flag.* Two sprites have collided
4…0	5SN	The number of 5th (or 9th in G3…G7 modes) sprite

	MSB	7	6	5	4	3	2	1	0	LSB
S#1		FL	LPS	ID #					FH	Status register 1

7	FL	*Light pen.* Is set if light pen detects light. If IE2 is set, interrupt is generated. Reset when S#1 is read. *Mouse 2.* Is set if second button of mouse was pressed. This flag is not reset when reading status register S#1
6	LPS	*Light pen button.* Is set when light pen button is pressed *Mouse 1.* Is set if first button of mouse was pressed This flag is not reset when reading status register S#1 in both set-ups
5…1	ID #	The identification number of the VDP chip
0	FH	*Horizontal scan interrupt flag.* Is set if VDP scans line specified in register R#19. If IE1 is set, interrupt is generated. FH is reset when S#1 is read

	MSB	7	6	5	4	3	2	1	0	LSB
S#2		TR	VR	HR	BD	1	1	EO	CE	Status register 2

7	TR	*Transfer ready flag.* If set to 1, indicates to the CPU that VDP is ready for next transfer. Value of 0 means that VDP is not ready
6	VR	*Vertical retrace flag.* Is set during scanning of VBLANK area of the screen, i.e. during vertical retrace plus while lower and upper borders of the screen is drawn
5	HR	*Horizontal retrace flag.* Is set during scanning of HBLANK area of the screen, i.e. when right and left borders of the screen are drawn
4	BD	*Color detect flag.* When the search command is executed, this flag is set if specified color was detected
1	EO	*Display field flag.* If set to 0, indicates the first field. If set to 1, indicated the second field
0	CE	*Command execution flag.* If set to 1, indicates that VDP is busy executing a command

	MSB	7	6	5	4	3	2	1	0	LSB
S#3		X7	X6	X5	X4	X3	X2	X1	X0	Column register low
S#4		1	1	1	1	1	1	1	X8	Column register high
S#5		Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Row register low
S#6		1	1	1	1	1	1	Y9	Y8	Row register high

The above registers S#3…S#6 contain coordinate information about collision location of the sprites, or location of light pen, or relative movement of the mouse.

	MSB	7	6	5	4	3	2	1	0	LSB
S#7		C7	C6	C5	C4	C3	C2	C1	C0	Color register

This color register is used when executing commands “POINT” and “VRAM to CPU” and contains VRAM data.

	MSB	7	6	5	4	3	2	1	0	LSB
S#8		BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0	Coded color X register low
S#9		1	1	1	1	1	1	1	BX8	Coded color X register high

When the search command is executed and coded color had been detected (see R#2), this register contains its X coordinate.

3. SCREEN MODES

3.1. TEXT1 mode

Characteristics	
Pattern size (w*h)	6 dots * 8 dots
Patterns	256 types
Screen size (w*h)	40 * 24 patterns
Pattern colors	Two colors out of 512 (per screen)
VRAM area per screen	4K bytes

Controls	
Pattern font	VRAM pattern generator table
Screen pattern location	VRAM pattern name table
Pattern color code 1	High-order four bits of R#7
Pattern color code 0	Low-order four bits of R#7
Background color code	Low-order four bits of R#7

Mode flags					
Bit	M5 (R#0)	M4 (R#0)	M3 (R#0)	M2 (R#1)	M1 (R#1)
Value	0	0	0	0	1

MSX system default values		
BASIC SCREEN number	Pattern generator	Pattern layout
0, width 1 … 40	00800h … 00FFFh	00000h … 003BFh

3.1.2. Pattern Generator Table

The pattern generator table is a location in VRAM that stores patterns (font). Each pattern has number from PN0 to PN255. The font displayed on the screen for each pattern is constructed from 8 bytes, with 6 high-order bits displayed and 2 low-order bits not displayed. Pattern generator table base is stored in the register R#4.

Example of pattern generator table is provided below.

MSB LSB

Offset	7	6	5	4	3	2	1	0	
0			■						Pattern number o (PN0)
1		■		■					
2	■				■				
3	■				■				
4	■	■	■	■	■				
5	■				■				
6	■				■				
7									
8	■	■	■	■					Pattern number 1 (PN1)
9	■				■				
10	■				■				
11	■	■	■	■					
12	■				■				
13	■				■				
14	■	■	■	■					
15									
	● ● ●			● ● ●			● ● ●		
2040	■		■		■				Pattern number 255 (PN255)
2041		■		■		■			
2042	■		■		■				
2043		■		■		■			
2044	■		■		■				
2045		■		■		■			
2046	■		■		■				
2047		■		■		■			

3.1.3. Pattern Layout Table settings

The pattern layout table is a map of the screen (per screen image). Every location of the screen contains code of the pattern displayed at respective location. This table has 40*24 (960) locations where defined patterns can be displayed. Pattern layout table base address is stored in register R#2, and corresponds to the cell (0, 0) with address 0 in the picture below.

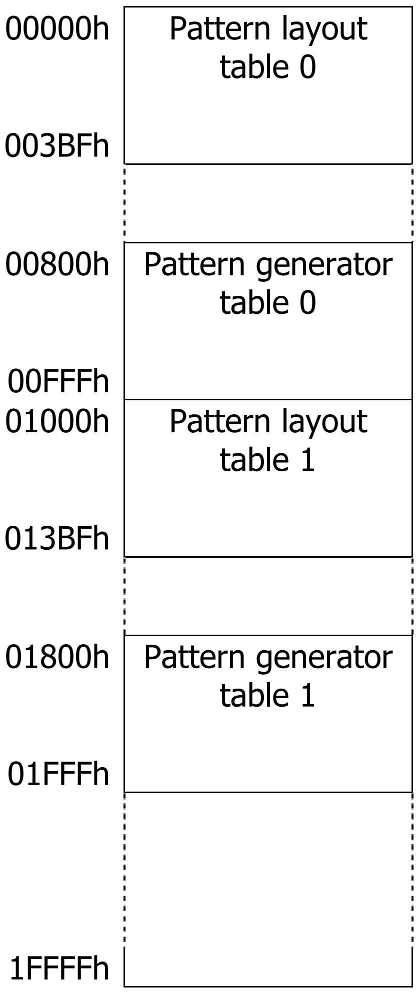
		Columns							
		0	1	2	3			39	← X
Rows	0	0	1	2	3	...	...	39	
	1	40	41	42	43	...	...	79	
		...	...	...	...	...	...	...	
		...	...	...	...	...	...	...	
	22	880	881	882	...	...	...	919	
	23	920	921	922	...	...	...	959	
	↑ Y								

3.1.4. Color register settings

Color settings are located in the register R#7. Bits TC3…TC0 specify pattern color code of the pixels identified as "1" in the bitmap values of pattern generator table, bits BD3…BD0 specify pattern color code of the pixels identified as "0" in the bitmap values of pattern generator table as well as screen border color.

Note: screen border color is the same as the pattern backdrop color in TEXT1 mode.

3.1.5. Example of VRAM allocation for TEXT1 mode



MSB	7	6	5	4	3	2	1	0	LSB
R#2	0	0	0	0	0	0	0	0	
		A16	A15	A14	A13	A12	A11	A10	

MSB	7	6	5	4	3	2	1	0	LSB
R#4	0	0	0	0	0	0	0	1	
			A16	A15	A14	A13	A12	A11	

MSB	7	6	5	4	3	2	1	0	LSB
R#2	0	0	0	0	0	1	0	0	
		A16	A15	A14	A13	A12	A11	A10	

MSB	7	6	5	4	3	2	1	0	LSB
R#4	0	0	0	0	0	0	1	1	
			A16	A15	A14	A13	A12	A11	

Maximum of 32 pages may be allocated in the same manner if VDP has 128K bytes attached to it.

3.2. TEXT2 mode

Characteristics	
Pattern size (w*h)	6 dots * 8 dots
Patterns	256 types
Screen size (w*h)	80 * 24 patterns if LN bit of R#9 set to 0 80 * 26.5 patterns if LN bit of R#9 set to 1
Pattern colors	Two colors out of 512 (per screen), four if using blinking
VRAM area per screen	8K bytes

Controls	
Pattern font	VRAM pattern generator table
Screen pattern location	VRAM pattern name table
Pattern color code 1	High-order four bits of R#7
Pattern color code 0	Low-order four bits of R#7
Background color code	Low-order four bits of R#7
Blinking pattern color code 1	High-order four bits of R#12
Blinking pattern color code 0	Low-order four bits of R#12

Mode flags					
Bit	M5 (R#0)	M4 (R#0)	M3 (R#0)	M2 (R#1)	M1 (R#1)
Value	0	1	0	0	1

Other flags	
LN bit of R#9	0: 24 lines displayed 1: 26.5 lines displayed

MSX system default values			
BASIC SCREEN number	Pattern generator	Pattern layout	Pattern color table
0, width 41 … 80	01000h … 017FFh	00000h … 0077Fh 00000h … 0086Fh	00800h … 008EFh 00800h … 0090Dh

3.2.2. Pattern Generator Table

Organization of pattern generator table is the same as in TEXT1 mode. Register R#4 defines base address of the table.

3.2.3. Pattern layout table settings

The pattern layout table is a map of the screen (per screen image). Every location of the screen contains code of the pattern displayed at respective location. This table has 40*24 (2160) locations where defined patterns can be displayed. Pattern layout table base address is stored in register R#2 (see below), and corresponds to the cell (0, 0) with address 0 in the picture below. Note that if LN bit of R#9 is set to 1, then 26 lines are displayed, plus an upped half of 27th pattern line is displayed.

	MSB	7	6	5	4	3	2	1	0	LSB
R#2		0	A16	A15	A14	A13	A12	1	1	Pattern layout table

Screen mapping of pattern layout table is provided below.

		Columns							
		0	1	2	3			79	← X
Rows	0	0	1	2	3	…	…	79	
	1	80	81	82	83	…	…	159	
		…	…	…	…	…	…	…	
		…	…	…	…	…	…	…	
	25	2000	2001	2002	…	…	…	2079	
	26	2080	2081	2082	…	…	…	2159	
	↑ Y								

3.2.4. Color table settings

Each position on the screen has separate bit for the blinking attribute, and if this bits is set to 1, blinking will be applied to the pattern placed in this area in the pattern layout table. Table start address is set in registers R#3 and R#10.

	MSB	7	6	5	4	3	2	1	0	LSB
R#3		A13	A12	A11	A10	A9	1	1	1	Color table base address registers
R#10		0	0	0	0	0	A16	A15	A14	

Screen mapping of color table is provided below.

MSB	7	6	5	4	3	2	1	0	LSB
0	(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(5, 0)	(6, 0)	(7, 0)	
1	(8, 0)	(9, 0)	(10, 0)	(11, 0)	(12, 0)	(13, 0)	(14, 0)	(15, 0)	
	…	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…		
269	(72, 26)	(73, 26)	…	…	…	…	…	(79,26)	

3.2.5. Color register settings

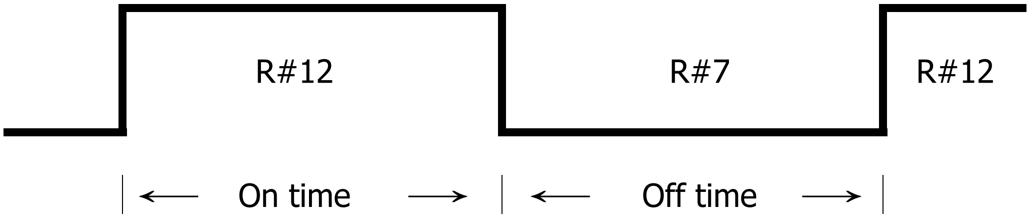
Color settings are located in the registers R#7 and R#12.

Bits TC3…TC0 of R#7 specify pattern color code of the pixels identified as "1" in the bitmap values of pattern generator table, bits BD3…BD0 specify pattern color code of the pixels identified as "0" in the bitmap values of pattern generator table as well as screen border color. Alternate blinking color is set in R#12.

Note: screen border color is the same as the pattern backdrop color in TEXT2 mode.

3.2.6. Blink register settings

Color codes set in registers R#7 and R#12 will be alternately displayed; programmer can control period of blinking (time on and time off) through register R#13.

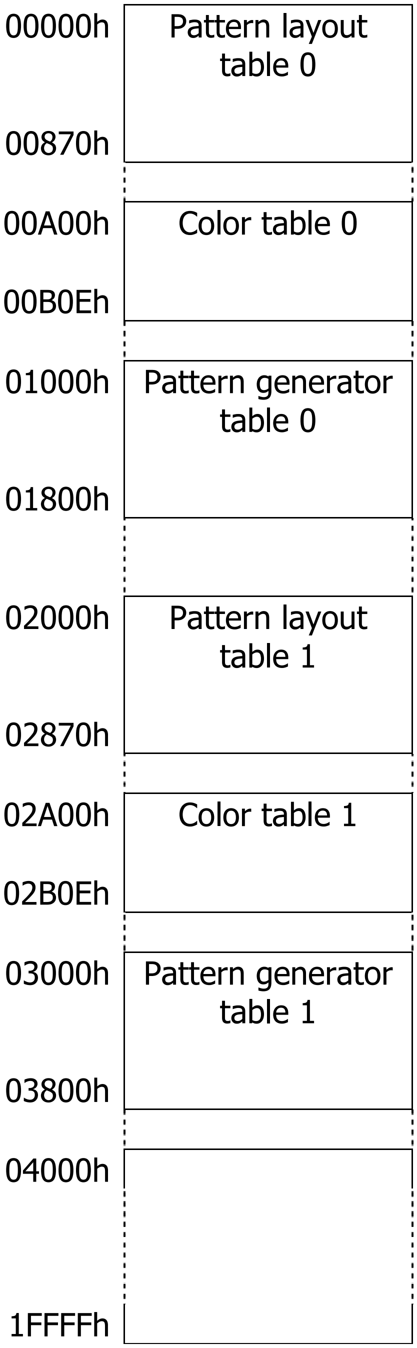


	MSB	7	6	5	4	3	2	1	0	LSB
R#13		ON3	ON2	ON1	ON0	OF3	OF2	OF1	OF0	Blinking period register
		On time				Off time				

The NTSC timing data is provided in the table below.

Delay data (binary)	Time (ms)	Delay data (binary)	Time (ms)
0 0 0 0	0	1 0 0 0	1335.1
0 0 0 1	166.9	1 0 0 1	1509.9
0 0 1 0	333.8	1 0 1 0	1668.8
0 0 1 1	500.6	1 0 1 1	1835.7
0 1 0 0	667.5	1 1 0 0	2002.6
0 1 0 1	834.4	1 1 0 1	2169.5
0 1 1 0	1001.3	1 1 1 0	2336.3
0 1 1 1	1168.2	1 1 1 1	2503.2

3.2.7. Example of VRAM allocation for TEXT2 mode



MSB	7	6	5	4	3	2	1	0	LSB
R#2	0	0	0	0	0	0	1	1	
		A16	A15	A14	A13	A12			

MSB	7	6	5	4	3	2	1	0	LSB
R#3	0	0	1	0	1	1	1	1	
	A13	A12	A11	A10	A9				

MSB	7	6	5	4	3	2	1	0	LSB
R#10	0	0	0	0	0	0	0	0	
						A16	A15	A14	

MSB	7	6	5	4	3	2	1	0	LSB
R#4	0	0	0	0	0	0	1	0	
			A16	A15	A14	A13	A12	A11	

Maximum of 16 pages may be allocated in the same manner if VDP has 128K bytes attached to it.

3.3. MULTICOLOR (MC) mode

Characteristics	
Screen composition (w*h)	64 * 48 color blocks
Color blocks	Sixteen colors out of 512 colors
Sprite mode	Sprite mode 1
VRAM area per screen	4K bytes

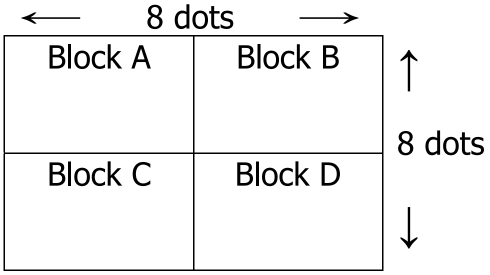
Controls	
Color block color code	VRAM pattern generator table
Color block location	VRAM pattern name table
Background color code	Low-order four bits of R#7
Sprites	VRAM sprite attribute table VRAM sprite pattern table

Mode flags					
Bit	M5 (R#0)	M4 (R#0)	M3 (R#0)	M2 (R#1)	M1 (R#1)
Value	0	0	0	1	0

MSX system default values				
BASIC SCREEN number	Pattern generator	Pattern layout	Sprite patterns	Sprite attributes
3	00000h ... 007FFh	00800h ... 00AFFh	03800h ... 03FFFh	01B00 ... 01B7Fh

3.3.1. Pattern Generator Table

Each pattern is made up of four color blocks. These patterns are of size of 8*8 for the screen display of 256*192 dots.



For each block A, B, C, and D sixteen colors can be specified. Two bytes are used for each pattern.

MSB	7	6	5	4	3	2	1	0	LSB
	Color code A				Color code B				
	Color code C				Color code D				

There're four color blocks (8 bytes) for each pattern layout location. Specific color block is used to display depending on the y-coordinate.

Pattern name #n			MSB	7	6	5	4	3	2	1	0	LSB
	A	B		Color code A				Color code B				Color of the pattern when y=0, 4, 8, 12, 16 or 20
	C	D		Color code C				Color code D				
	E	F		Color code E				Color code F				Color of the pattern when y=1, 5, 9, 13, 17 or 21
	G	H		Color code G				Color code H				
	I	J		Color code I				Color code J				Color of the pattern when y=2, 6, 10, 14, 18 or 22
	K	L		Color code K				Color code L				
	M	N		Color code M				Color code N				Color of the pattern when y=3, 7, 11, 15, 19 or 23
	O	P		Color code O				Color code P				

Start of the pattern generator table should be set in register R#4.

3.3.2. Pattern layout table settings

The pattern layout table is a map of the screen (per screen image), containing one byte for each screen location. Each byte specifies unique pattern number. Start of the pattern name table should be set in register R#2.

		Columns							
		0	1	2	3			31	← X
Rows	0	0	1	2	3	…	…	31	
	1	32	33	34	35	…	…	63	
		…	…	…	…	…	…	…	
		…	…	…	…	…	…	…	
	22	704	705	706	…	…	…	735	
	23	736	737	738	…	…	…	767	
	↑ Y								

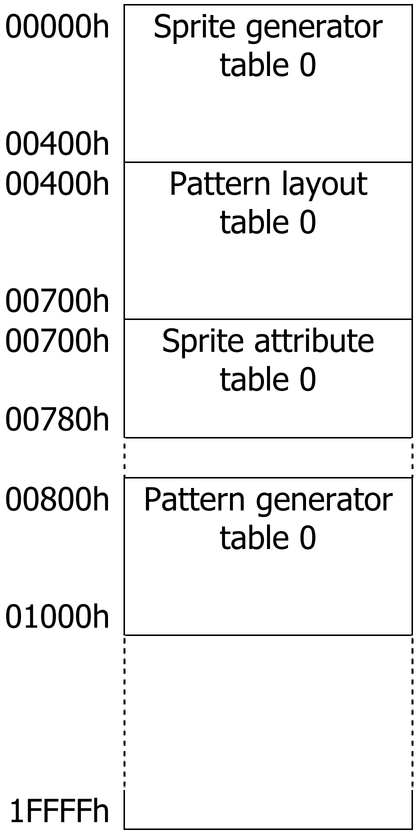
3.3.3. Color register settings

You can set color of the margin of the screen (backdrop color) specifying BD3…BD0 bits in register R#7. Note that bits TC3…TC0 of R#7 are ignored.

3.3.4. Sprite settings

Set the start address of the sprite attribute table in registers R#5 and R#11; set start address of the sprite pattern generator table in register R#6. For details about sprites please refer to section "Sprite mode 1".

3.3.5. Example of VRAM allocation for MULTICOLOR mode



Maximum of 32 pages may be allocated in the same manner if VDP has 128K bytes attached to it.

3.4. GRAPHIC1 (G1) mode

Characteristics	
Pattern size (w*h)	8 dots * 8 dots
Patterns	256 types
Screen size (w*h)	32 * 24 patterns (256 * 192 pixels)
Pattern colors	16 colors out of 512 (per screen)
Sprite mode	Sprite mode 1
VRAM area per screen	4K bytes

Controls	
Pattern font	VRAM pattern generator table
Screen pattern location	VRAM pattern name table
Pattern color codes 1 & 0	Specified as a group for each 8-bit pattern in color table
Background color code	Low-order four bits of R#7
Sprites	VRAM sprite attribute table VRAM sprite pattern table

Mode flags					
Bit	M5 (R#0)	M4 (R#0)	M3 (R#0)	M2 (R#1)	M1 (R#1)
Value	0	0	0	0	0

MSX system default values					
BASIC SCREEN number	Pattern generator	Pattern layout	Pattern colors	Sprite patterns	Sprite attributes
1	00000h … 017FFh	01800h … 01AFFh	02000h … 0201Fh	03800h … 03FFFh	01B00 … 01B7Fh

3.4.1. Pattern Generator Table

The pattern generator table is a location in VRAM that stores patterns (font). Each pattern has number from PN0 to PN255. The font displayed on the screen for each pattern is constructed from 8 bytes, with all 8 bits of each byte displayed. Pattern generator table base is stored in the register R#4. Example of pattern generator table is provided below.

	MSB							LSB	
Offset	7	6	5	4	3	2	1	0	
0			■	■	■				Pattern number o (PN0)
1		■				■			
2	■						■		
3	■						■		
4	■	■	■	■	■	■	■		
5	■						■		
6	■						■		
7									
8	■	■	■	■	■	■			Pattern number 1 (PN1)
9	■						■		
10	■						■		
11	■	■	■	■	■	■			
12	■						■		
13	■						■		
14	■	■	■	■	■	■			
15									
	• • •			• • •			• • •		
2040	■		■		■		■		Pattern number 255 (PN255)
		■		■		■		■	
2042	■		■		■		■		
2043		■		■		■		■	
2044	■		■		■		■		
2045		■		■		■		■	
2046	■		■		■		■		
2047		■		■		■		■	

3.4.2. Pattern layout table settings

The pattern layout table is a map of the screen (per screen image). Every byte location of the screen contains code of the pattern displayed at respective location. This table has 32*24 (767) locations where defined patterns can be displayed. Pattern layout table base address is stored in register R#2, and corresponds to the cell (0, 0) with address 0 in the picture below.

		Columns							
		0	1	2	3			31	← X
Rows	0	0	1	2	3	…	…	31	
	1	32	33	34	35	…	…	63	
		…	…	…	…	…	…	…	
		…	…	…	…	…	…	…	
	22	704	705	706	…	…	…	735	
	23	736	737	738	…	…	…	767	
	↑ Y								

3.4.3. Color register settings

You can set color of the margin of the screen (backdrop color) specifying BD3…BD0 bits in register R#7. Note that bits TC3…TC0 of R#7 are ignored.

3.4.4. Color table settings

Table start address is set in registers R#3 and R#10.

	MSB	7	6	5	4	3	2	1	0	LSB
R#3		A13	A12	A11	A10	A9	A8	A7	A6	Color table base address registers
R#10		0	0	0	0	0	A16	A15	A14	

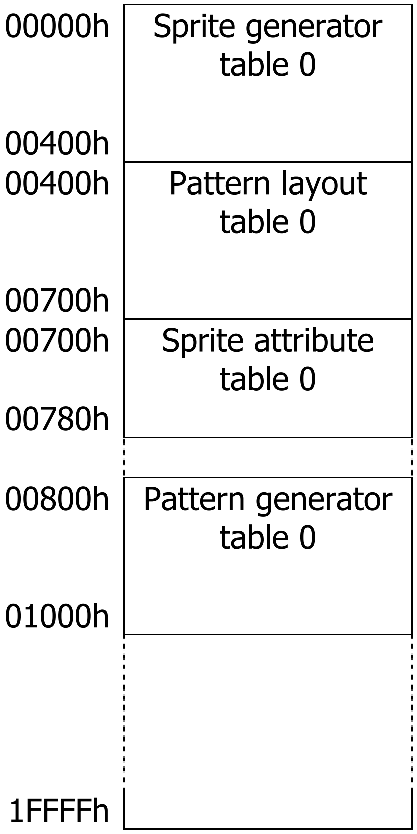
Color table size is 32 bytes, each byte organized in the same way as register R#7 (FC is foreground color, BC is background color). Patterns 0...7 are assigned the first color from color table, patterns 8…15 are assigned second color from the color table, etc, and patterns F8h…FFh are assigned 31st color from the color table. Color table map is provided below.

MSB	7	6	5	4	3	2	1	0	LSB
0	FC3	FC2	FC1	FC0	BC3	BC2	BC1	BC0	Color for patterns 0...7
1	FC3	FC2	FC1	FC0	BC3	BC2	BC1	BC0	Color for patterns 8…15
	…	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…		
31	FC3	FC2	FC1	FC0	BC3	BC2	BC1	BC0	Color for patterns F8…FF

3.4.5. Sprite settings

Set the start address of the sprite attribute table in registers R#5 and R#11; set start address of the sprite pattern generator table in register R#6. For details about sprites please refer to section "Sprite mode 1".

3.4.6. Example of VRAM allocation for GRAPHIC1 mode



Maximum of 32 pages may be allocated in the same manner if VDP has 128K bytes attached to it.

3.5. GRAPHIC2 (G2) and GRAPHIC3 (G3) modes

Characteristics	
Pattern size (w*h)	8 dots * 8 dots
Patterns	768 types (256 per ⅓ of the screen)
Screen size (w*h)	32 * 24 patterns (256 * 192 pixels)
Pattern colors	16 colors out of 512 (per screen)
Sprite mode*	Sprite mode 1 (GRAPHIC 2) Sprite mode 2 (GRAPHIC 3)
VRAM area per screen	16K bytes

* GRAPHIC modes 2 and 3 are identical except for sprite modes

Controls	
Pattern font	VRAM pattern generator table
Screen pattern location	VRAM pattern name table
Pattern color codes 1 & 0	Specified in pattern color table for 8-pixel groups
Background color code	Low-order four bits of R#7
Sprites	VRAM sprite attribute table VRAM sprite pattern table

Mode flags: GRAPHIC2					
Bit	M5 (R#0)	M4 (R#0)	M3 (R#0)	M2 (R#1)	M1 (R#1)
Value	0	0	1	0	0

Mode flags: GRAPHIC3					
Bit	M5 (R#0)	M4 (R#0)	M3 (R#0)	M2 (R#1)	M1 (R#1)
Value	0	1	0	0	0

MSX system default values

BASIC SCREEN number	Pattern generator	Pattern layout	Pattern colors	Sprite patterns	Sprite attributes	Sprite colors
G2: 2	00000h … 017FFh	01800h … 01AFFh	02000h … 037FFh	03800h … 03FFFh	01B00 … 01B7Fh	---
G3: 4					01E00 … 01E7Fh	01C00h … 01DFFh

3.5.1. Screen map and pattern tables

Unlike in other modes, in this mode screen is divided vertically into three logical parts. Every part has its own pattern generator and pattern color tables, but all of them share the same pattern layout table, one after another. Screen map is provided below.

Rows	Columns							
	0	**1**	**2**	**3**			**31**	**← X**
0	0, 0	1, 0	2, 0	3, 0	…	…	31, 0	Upper ⅓ of the screen
1	0, 1	1, 1	2, 1	3, 1	…	…	31, 1	
	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…	
	0, 7	1, 7	2, 7	…	…	…	31, 7	
	0, 8	1, 8	2, 8	…	…	…	31, 8	Middle ⅓ of the screen
	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…	
	0, 15	1, 15	2, 15	…	…	…	31, 15	
	0, 16	1, 16	2, 16	…	…	…	31, 16	Lower ⅓ of the screen
	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…	
22	0, 22	1, 22	2, 22	…	…	…	31, 22	
23	0, 23	1, 23	2, 23	…	…	…	31, 23	
↑ Y								

	Pattern generator table	Pattern color table
(base address) 00000h 007FFh	Pattern images for upper ⅓ of the screen (256)	Pattern colors for upper ⅓ of the screen (256)
00800h 00FFFh	Pattern images for middle ⅓ of the screen (256)	Pattern colors for middle ⅓ of the screen (256)
01000h 017FFh	Pattern images for lower ⅓ of the screen (256)	Pattern colors for lower ⅓ of the screen (256)

3.5.2. Pattern Tables

The pattern generator table is a location in VRAM that stores patterns (font). Each pattern has number from PN0 to PN255, for every ⅓ of the screen. The font displayed on the screen for each pattern is constructed from 8 bytes, with all 8 bits of each byte displayed. Pattern generator table base is stored in the register R#4.

	MSB	7	6	5	4	3	2	1	0	LSB
R#4		0	0	A16	A15	A14	A13	1	1	Pattern generator table base address

The pattern layout table is a map of the screen (per screen image). Every byte location of the screen contains code of the pattern displayed at respective location. This table has three 32*8 locations (upper, middle and lower) arranged consecutively where defined patterns can be displayed. Pattern layout table base address is stored in register R#2, and corresponds to the cell (0, 0) with address 0 in the picture below.

Example of pattern generator table is provided below. Color table identifies color 1 (upper four bits) and color 0 (lower four bits) for every row of the pattern.

	MSB							LSB		MSB							LSB	
Offset	7	6	5	4	3	2	1	0		7	6	5	4	3	2	1	0	Offset
0			■						Pattern number 0 (PN0)	1	1	1	1	0	0	0	0	0
1		■		■						1	1	1	1	0	0	0	0	1
2	■				■					1	1	1	1	0	0	0	0	2
3	■				■					1	1	1	1	0	0	0	0	3
4	■	■	■	■	■					1	1	1	1	0	0	0	0	4
5	■				■					1	1	1	1	0	0	0	0	5
6	■				■					1	1	1	1	0	0	0	0	6
7										1	1	1	1	0	0	0	0	7
8	■	■	■	■					Pattern number 1 (PN1)	1	1	1	1	0	0	0	0	8
9	■				■					1	1	1	1	0	0	0	0	9
10	■				■					1	1	1	1	0	0	0	0	10
11	■	■	■	■						1	1	1	1	0	0	0	0	11
12	■				■					1	1	1	1	0	0	0	0	12
13	■				■					1	1	1	1	0	0	0	0	13
14	■	■	■	■						1	1	1	1	0	0	0	0	14
15										1	1	1	1	0	0	0	0	15
	●			●			●											

	⋮			⋮			⋮											
2040	■		■		■				Pattern number 255 (PN255)	1	1	1	1	0	0	0	0	2040
2041		■		■		■				1	1	1	1	0	0	0	0	2041
2042	■		■		■					1	1	1	1	0	0	0	0	2042
2043		■		■		■				1	1	1	1	0	0	0	0	2043
2044	■		■		■					1	1	1	1	0	0	0	0	2044
2045		■		■		■				1	1	1	1	0	0	0	0	2045
2046	■		■		■					1	1	1	1	0	0	0	0	2046
2047		■		■		■				1	1	1	1	0	0	0	0	2047
										Color 1				Color 0				

3.5.3. Color table settings

Table start address is set in registers R#3 and R#10.

	MSB	7	6	5	4	3	2	1	0	LSB
R#3		A13	1	1	1	1	1	1	1	Color table base address registers
R#10		0	0	0	0	0	A16	A15	A14	

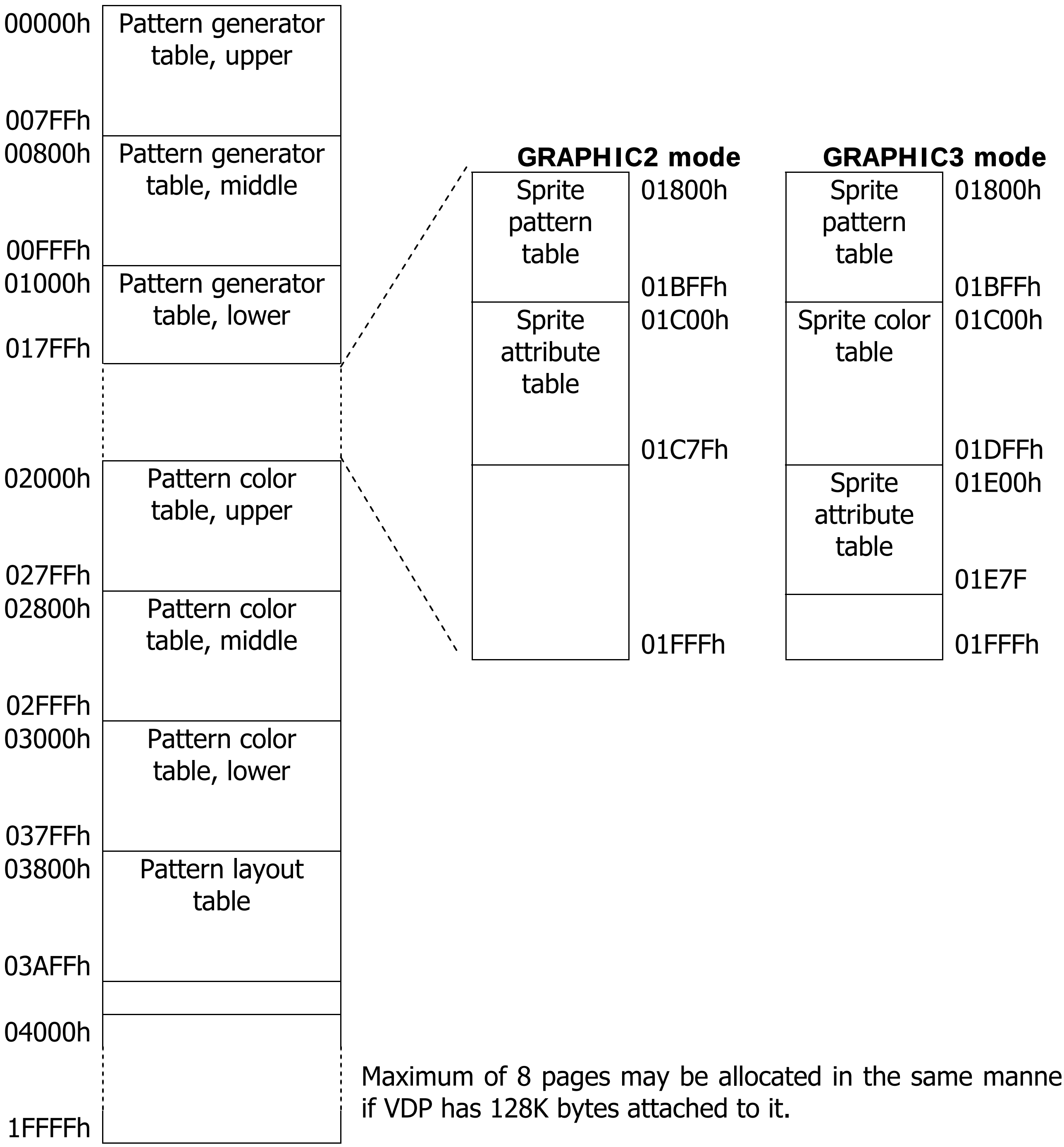
3.5.4. Color register settings

You can set color of the margin of the screen (backdrop color) specifying BD3…BD0 bits in register R#7. Note that bits TC3…TC0 of R#7 are ignored.

3.5.5. Sprite settings

Set the start address of the sprite attribute table in registers R#5 and R#11; set start address of the sprite pattern generator table in register R#6. For details about sprites in GRAPHIC2 mode, please refer to section "Sprite mode 1"; for details about sprites in GRAPHIC3 mode, please refer to section "Sprite mode 2"

3.5.6. Example of VRAM allocation for GRAPHIC2 and GRAPHIC3 modes



Maximum of 8 pages may be allocated in the same manner if VDP has 128K bytes attached to it.

3.6. GRAPHIC4 (G4) mode

Characteristics	
Bit-mapped Graphics mode	
Screen size (w*h)	256 * 192 pixels if LN bit of R#9 set to 0 256 * 212 pixels if LN bit of R#9 set to 1
Pattern colors	16 colors out of 512 (per screen)
Sprite mode	Sprite mode 2
VRAM area per screen	32K bytes

Controls	
Graphics	VRAM pattern name table
Background color code	Low-order four bits of R#7
Sprites	VRAM sprite attribute table VRAM sprite pattern table

Mode flags					
Bit	M5 (R#0)	M4 (R#0)	M3 (R#0)	M2 (R#1)	M1 (R#1)
Value	0	1	1	0	0

MSX system default values				
BASIC SCREEN number	Pattern layout (bitmap)	Sprite patterns	Sprite attributes	Sprite colors
5	00000h … 069FFh	07800h … 07FFFh	07600 … 0767Fh	07400h … 075FFh

3.6.1. Pattern layout table settings

The pattern layout table is a map of the screen (per screen image). Every byte location of the screen contains color codes for **two** dots. This is bitmap graphics mode, and there's no pattern generator table.

Rows				Columns				
	0	1	2	3			255	← X
0	0, 0	1, 0	2, 0	3, 0	…	…	255, 0	
1	0, 1	1, 1	2, 1	3, 1	…	…	255, 1	
	…	…	…	…	…	…	…	
	…	…	…	…	X, Y	…	…	
191	0, 191	1, 191	2, 191	…	…	…	255,191	LN=0
	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…	
211	0, 211	1, 211	2, 211	…	…	…	255,211	LN=1
↑ Y								

MSB	7	6	5	4	3	2	1	0	LSB
0	Color code for (0, 0)				Color code for (1, 0)				Base address
1	Color code for (2, 0)				Color code for (3, 0)				
127	Color code for (254, 0)				Color code for (255, 0)				
128	Color code for (0, 1)				Color code for (1, 1)				
27134	Color code for (252, 211)				Color code for (253, 211)				
27135	Color code for (254, 211)				Color code for (255, 211)				

Pattern layout table base address is stored in register R#2, and corresponds to the cell (0, 0) in the picture above.

	MSB	7	6	5	4	3	2	1	0	LSB
R#2		0	A16	A15	1	1	1	1	1	Pattern layout table base address

3.6.2. Color register settings

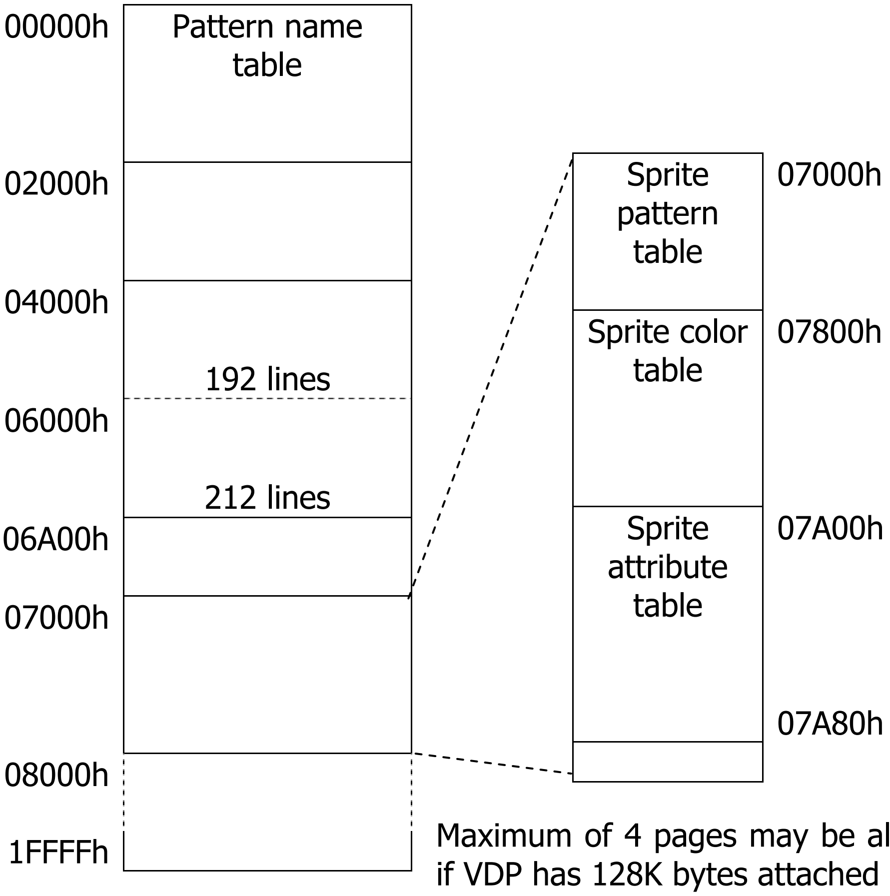
You can set color of the margin of the screen (backdrop color) specifying BD3…BD0 bits in register R#7. Note that bits TC3…TC0 of R#7 are ignored.

3.6.3. Sprite settings

Set the start address of the sprite attribute table in registers R#5 and R#11; set start address of the sprite pattern generator table in register R#6. For details about sprites please refer to section “Sprite mode 2”

	MSB 7	6	5	4	3	2	1	0 LSB	
R#5	A14	A13	A12	A11	A10	1	1	1	Sprite attribute table low
R#11	0	0	0	0	0	0	A16	A15	Sprite attribute table high
R#6	0	0	A16	A15	A14	A13	A12	A11	Sprite pattern generator table

3.6.4. Example of VRAM allocation for GRAPHIC4 mode



Maximum of 4 pages may be allocated in the same manner if VDP has 128K bytes attached to it.

3.7. GRAPHIC5 (G5) mode

Characteristics	
Bit-mapped Graphics mode	
Screen size (w*h)	512 * 192 pixels if LN bit of R#9 set to 0 512 * 212 pixels if LN bit of R#9 set to 1
Pattern colors	4 colors out of 512 (per screen)
Sprite mode	Sprite mode 2
VRAM area per screen	32K bytes

Controls	
Graphics	VRAM pattern name table
Background color code	Low-order four bits of R#7
Sprites	VRAM sprite attribute table VRAM sprite pattern table

Mode flags					
Bit	M5 (R#0)	M4 (R#0)	M3 (R#0)	M2 (R#1)	M1 (R#1)
Value	1	0	0	0	0

MSX system default values				
BASIC SCREEN number	Pattern layout (bitmap)	Sprite patterns	Sprite attributes	Sprite colors
6	00000h … 069FFh	07800h … 07FFFh	07600 … 0767Fh	07400h … 075FFh

3.7.1. Pattern layout table settings

The pattern layout table is a map of the screen (per screen image). Every byte location of the screen contains color codes for **four** dots. This is bitmap graphics mode, and there’s no pattern generator table.

		Columns						
Rows	0	1	2	3			511	← X
0	0, 0	1, 0	2, 0	3, 0	…	510, 0	511, 0	
1	0, 1	1, 1	2, 1	3, 1	…	…	511, 1	
	…	…	…	…	…	…	…	
	…	…	…	…	X, Y	…	…	
191	0, 191	1, 191	2, 191	…	…	…	511,191	LN=0
	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…	
211	0, 211	1, 211	2, 211	…	…	…	511,211	LN=1
↑ Y								

MSB	7 6	5 4	3 2	1 0	LSB
0	Color code for (0, 0)	Color code for (1, 0)	Color code for (2, 0)	Color code for (3, 0)	Base address
1	Color code for (4, 0)	Color code for (5, 0)	Color code for (6, 0)	Color code for (7, 0)	
127	Color code for (508, 0)	Color code for (509, 0)	Color code for (510, 0)	Color code for (511, 0)	
128	Color code for (0, 1)	Color code for (1, 1)	Color code for (2, 1)	Color code for (3, 1)	
27135	Color code for (508, 211)	Color code for (509, 211)	Color code for (510, 211)	Color code for (511, 211)	

Pattern layout table base address is stored in register R#2, and corresponds to the cell (0, 0) in the picture above.

	MSB	7	6	5	4	3	2	1	0	LSB
R#2		0	A16	A15	1	1	1	1	1	Pattern layout table base address

3.7.2. Color register settings

You can set color of the margin of the screen (backdrop color) specifying BD3…BD0 bits in register R#7. Note that bits TC3…TC0 of R#7 are ignored.

3.7.3. Sprite settings

Set the start address of the sprite attribute table in registers R#5 and R#11; set start address of the sprite pattern generator table in register R#6. For details about sprites please refer to section "Sprite mode 2"

	MSB	7	6	5	4	3	2	1	0	LSB
R#5		A14	A13	A12	A11	A10	1	1	1	Sprite attribute table low
R#11		0	0	0	0	0	0	A16	A15	Sprite attribute table high
R#6		0	0	A16	A15	A14	A13	A12	A11	Sprite pattern generator table

3.7.4. Hardware tiling function

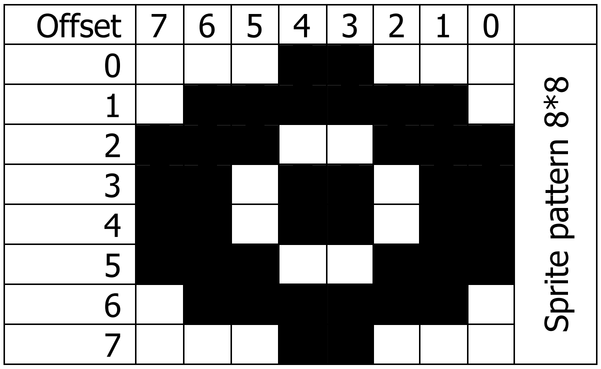
VDP can only display 4 solid colors in G5 mode however pixels are so small that combination of two pixels from the set of those 4 solid colors possible produces another visible color, a mixture of the applied two.

This feature is only available in G5 mode, and is applied to the sprites and to the screen border color. G5 has 512 pixels in its X-axis, but sprites' X-coordinate is between 0 and 255. This means that for single sprite dot there're two font pattern dots, and from forn pattern point of view sprite of 8*8 has size of 16*8.

SM2 sprite color table uses 4 bits for each line of sprite. For every dot in sprite pattern table defined as "1" in sprite image bitmap, lower two bits from this 4-bit set define color of odd pixels of the sprite, and higher two bits from this set define color of even pixels of the sprite.

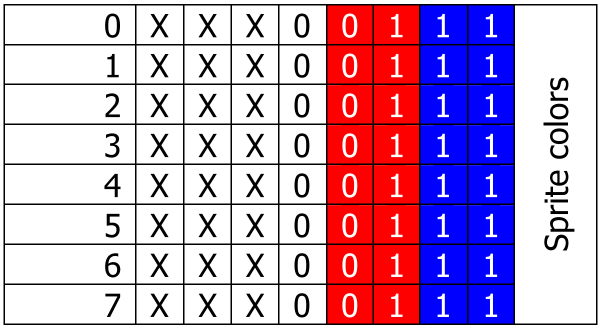
In the example above imagine that palette color 1 is set to red (7, 0, 0) and color 3 set to blue (0, 0, 7).

MSB LSB



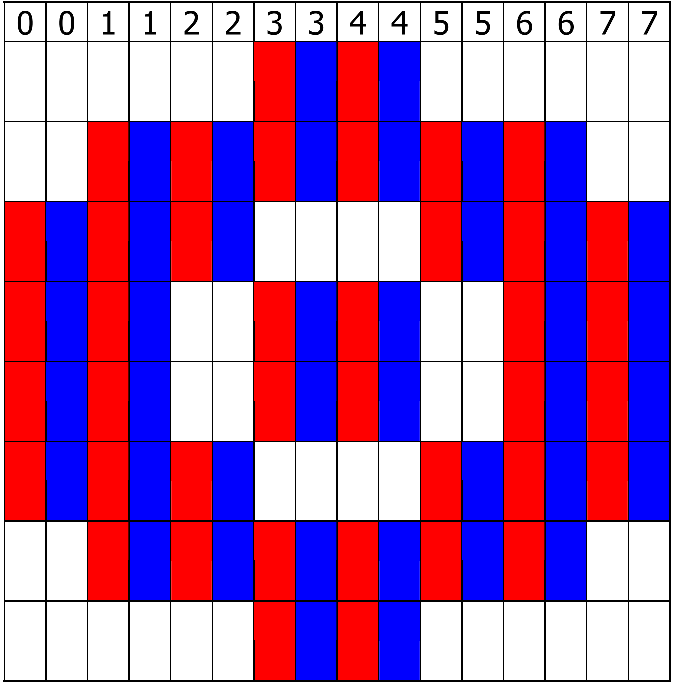
Offset	7	6	5	4	3	2	1	0	
0				■	■				Sprite pattern 8*8
1		■	■	■	■	■	■		
2	■	■	■			■	■	■	
3	■	■		■	■		■	■	
4	■	■		■	■		■	■	
5	■	■	■			■	■	■	
6		■	■	■	■	■	■		
7				■	■				

MSB LSB



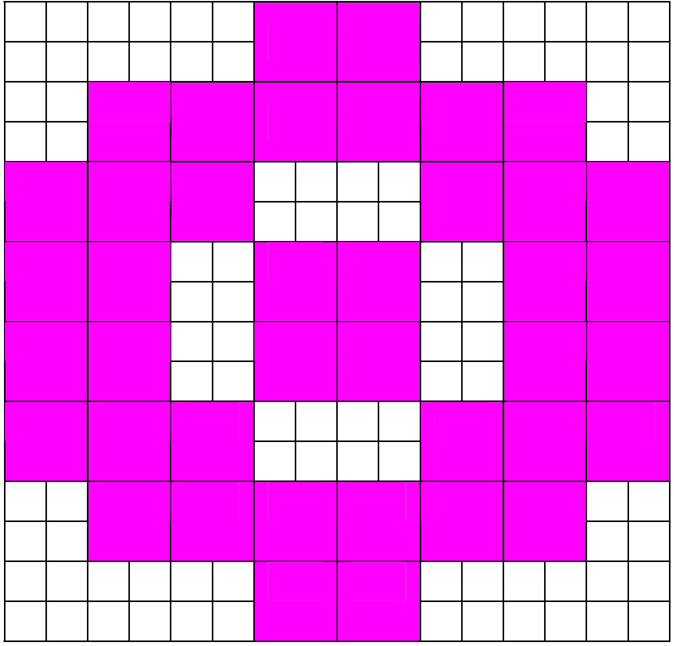
0	X	X	X	0	0	1	1	1	Sprite colors
1	X	X	X	0	0	1	1	1	
2	X	X	X	0	0	1	1	1	
3	X	X	X	0	0	1	1	1	
4	X	X	X	0	0	1	1	1	
5	X	X	X	0	0	1	1	1	
6	X	X	X	0	0	1	1	1	
7	X	X	X	0	0	1	1	1	

Actual sprite representation on the screen in G5 mode (8*8 size)



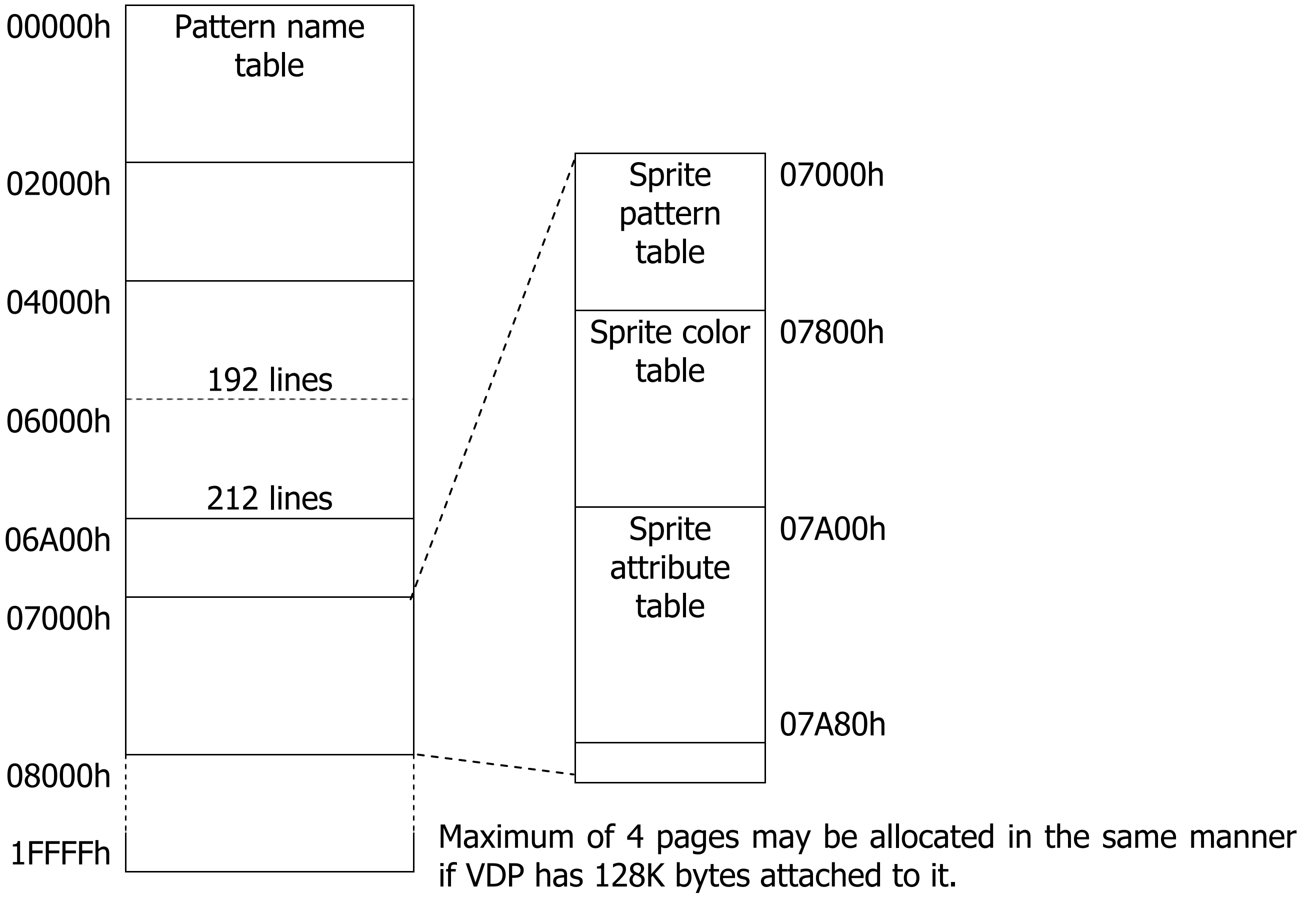
0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7
						🟥	🟦	🟥	🟦						
		🟥	🟦	🟥	🟦	🟥	🟦	🟥	🟦	🟥	🟦	🟥	🟦		
🟥	🟦	🟥	🟦	🟥	🟦					🟥	🟦	🟥	🟦	🟥	🟦
🟥	🟦	🟥	🟦			🟥	🟦	🟥	🟦			🟥	🟦	🟥	🟦
🟥	🟦	🟥	🟦			🟥	🟦	🟥	🟦			🟥	🟦	🟥	🟦
🟥	🟦	🟥	🟦	🟥	🟦					🟥	🟦	🟥	🟦	🟥	🟦
		🟥	🟦	🟥	🟦	🟥	🟦	🟥	🟦	🟥	🟦	🟥	🟦		
						🟥	🟦	🟥	🟦						

Sprite image – how sprite is seen (perceived) by the user in G5 mode



						🟪	🟪	🟪	🟪						
						🟪	🟪	🟪	🟪						
		🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪		
		🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪		
🟪	🟪	🟪	🟪	🟪	🟪					🟪	🟪	🟪	🟪	🟪	🟪
🟪	🟪	🟪	🟪	🟪	🟪					🟪	🟪	🟪	🟪	🟪	🟪
🟪	🟪	🟪	🟪			🟪	🟪	🟪	🟪			🟪	🟪	🟪	🟪
🟪	🟪	🟪	🟪			🟪	🟪	🟪	🟪			🟪	🟪	🟪	🟪
🟪	🟪	🟪	🟪			🟪	🟪	🟪	🟪			🟪	🟪	🟪	🟪
🟪	🟪	🟪	🟪			🟪	🟪	🟪	🟪			🟪	🟪	🟪	🟪
🟪	🟪	🟪	🟪	🟪	🟪					🟪	🟪	🟪	🟪	🟪	🟪
🟪	🟪	🟪	🟪	🟪	🟪					🟪	🟪	🟪	🟪	🟪	🟪
		🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪		
		🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪		
						🟪	🟪	🟪	🟪						
						🟪	🟪	🟪	🟪						

3.7.5. Example of VRAM allocation for GRAPHIC5 mode



Maximum of 4 pages may be allocated in the same manner if VDP has 128K bytes attached to it.

3.8. GRAPHIC6 (G6) mode

Characteristics	
Bit-mapped Graphics mode	
Screen size (w*h)	512 * 192 pixels if LN bit of R#9 set to 0 512 * 212 pixels if LN bit of R#9 set to 1
Pattern colors	16 colors out of 512 (per screen)
Sprite mode	Sprite mode 2
VRAM area per screen	64K bytes

Controls	
Graphics	VRAM pattern name table
Background color code	Low-order four bits of R#7
Sprites	VRAM sprite attribute table VRAM sprite pattern table

Mode flags					
Bit	M5 (R#0)	M4 (R#0)	M3 (R#0)	M2 (R#1)	M1 (R#1)
Value	1	0	1	0	0

MSX system default values				
BASIC SCREEN number	Pattern layout (bitmap)	Sprite patterns	Sprite attributes	Sprite colors
7	00000h … 0D3FFh	0F000h … 0F7FFh	0FA00 … 0FA7Fh	0F800h … 0F9FFh

3.8.1. Pattern layout table settings

The pattern layout table is a map of the screen (per screen image). Every byte location of the screen contains color codes for **two** dots. This is bitmap graphics mode, and there's no pattern generator table.

		Columns						
Rows	0	1	2	3			511	← X
0	0, 0	1, 0	2, 0	3, 0	…	510, 0	511, 0	
1	0, 1	1, 1	2, 1	3, 1	…	…	511, 1	
	…	…	…	…	…	…	…	
	…	…	…	…	X, Y	…	…	
191	0, 191	1, 191	2, 191	…	…	…	511,191	LN=0
	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…	
211	0, 211	1, 211	2, 211	…	…	…	511,211	LN=1
↑ Y								

MSB	7 6 5 4	3 2 1 0	LSB
0	Color code for (0, 0)	Color code for (1, 0)	Base address
1	Color code for (2, 0)	Color code for (3, 0)	
255	Color code for (510, 0)	Color code for (511, 0)	
256	Color code for (0, 1)	Color code for (1, 1)	
54270	Color code for (508, 211)	Color code for (509, 211)	
54271	Color code for (510, 211)	Color code for (511, 211)	

Pattern layout table base address is stored in register R#2, and corresponds to the cell (0, 0) in the picture above.

	MSB	7	6	5	4	3	2	1	0	LSB
R#2		0	A16	1	1	1	1	1	1	Pattern layout table base address

3.8.2. Color register settings

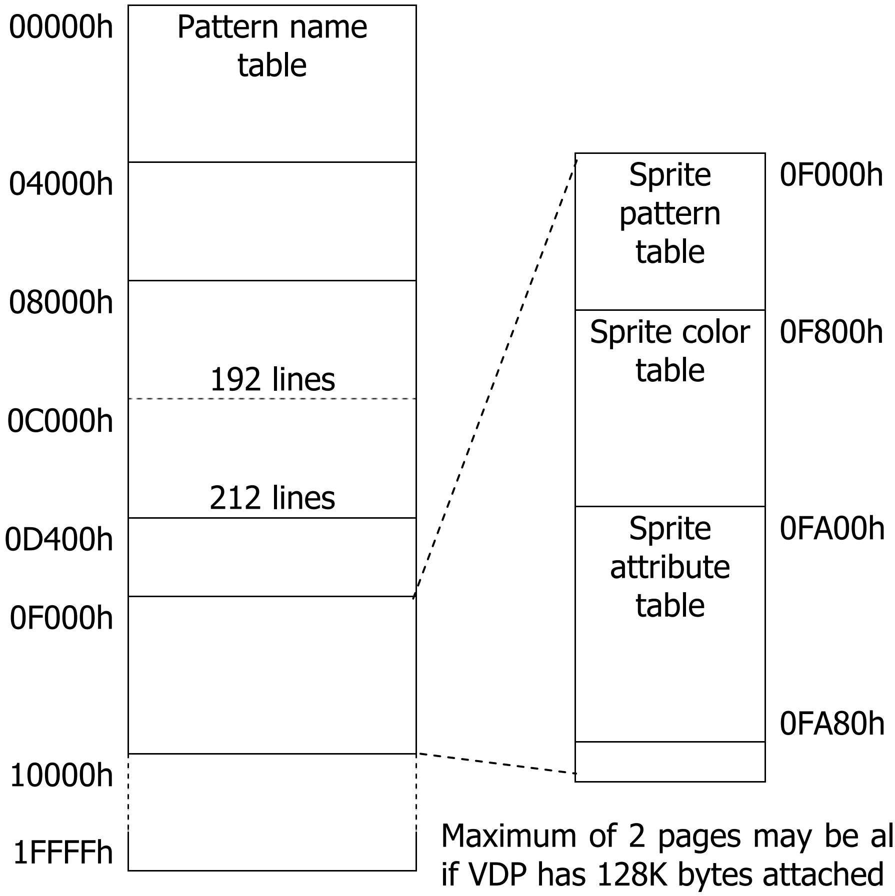
You can set color of the margin of the screen (backdrop color) specifying BD3…BD0 bits in register R#7. Note that bits TC3…TC0 of R#7 are ignored.

3.8.3. Sprite settings

Set the start address of the sprite attribute table in registers R#5 and R#11; set start address of the sprite pattern generator table in register R#6. For details about sprites please refer to section "Sprite mode 2"

MSB	7	6	5	4	3	2	1	0	LSB
R#5	A14	A13	A12	A11	A10	1	1	1	Sprite attribute table low
R#11	0	0	0	0	0	0	A16	A15	Sprite attribute table high
R#6	0	0	A16	A15	A14	A13	A12	A11	Sprite pattern generator table

3.8.4. Example of VRAM allocation for GRAPHIC4 mode



Maximum of 2 pages may be allocated in the same manner if VDP has 128K bytes attached to it.

3.9. GRAPHIC7 (G7) mode

Characteristics	
Bit-mapped Graphics mode	
Screen size (w*h)	256 * 192 pixels if LN bit of R#9 set to 0 256 * 212 pixels if LN bit of R#9 set to 1
Pattern colors	256 colors (per screen)
Sprite mode	Sprite mode 2
VRAM area per screen	64K bytes

Controls	
Graphics	VRAM pattern name table
Background color code	Low-order four bits of R#7
Sprites	VRAM sprite attribute table VRAM sprite pattern table

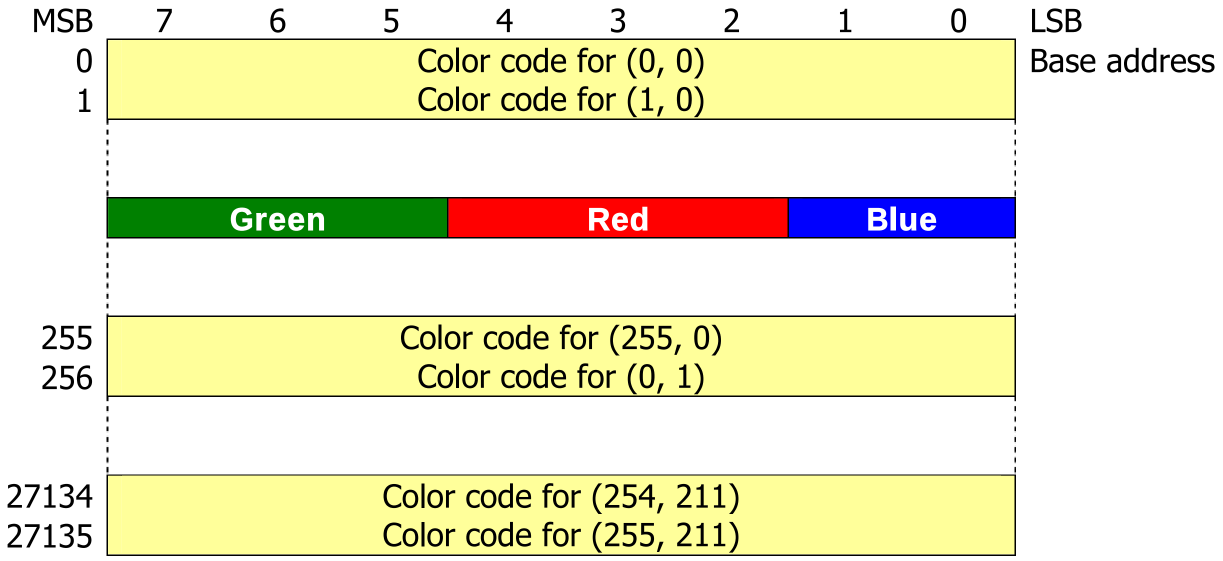
Mode flags					
Bit	M5 (R#0)	M4 (R#0)	M3 (R#0)	M2 (R#1)	M1 (R#1)
Value	1	1	1	0	0

MSX system default values				
BASIC SCREEN number	Pattern layout (bitmap)	Sprite patterns	Sprite attributes	Sprite colors
8	00000h … 0D3FFh	0F000h … 0F7FFh	0FA00 … 0FA7Fh	0F800h … 0F9FFh

3.9.1. Pattern layout table settings

The pattern layout table is a map of the screen (per screen image). Every byte location of the screen contains color code for **one** single dot. This is bitmap graphics mode, and there's no pattern generator table.

Rows	Columns							
	0	1	2	3			255	← X
0	0, 0	1, 0	2, 0	3, 0	…	254, 0	255, 0	
1	0, 1	1, 1	2, 1	3, 1	…	…	255, 1	
	…	…	…	…	…	…	…	
	…	…	…	…	X, Y	…	…	
191	0, 191	1, 191	2, 191	…	…	…	255,191	LN=0
	…	…	…	…	…	…	…	
	…	…	…	…	…	…	…	
211	0, 211	1, 211	2, 211	…	…	…	255,211	LN=1
↑ Y								



Pattern layout table base address is stored in register R#2, and corresponds to the cell (0, 0) in the picture above.

	MSB	7	6	5	4	3	2	1	0	LSB
R#2		0	0	A16	1	1	1	1	1	Pattern layout table base address
				**						

**Note that in G6 and G7 modes location of bit A16 differs

3.9.2. Color register settings

You can set color of the margin of the screen (backdrop color) specifying all the 8 bits in register R#7 (256 possible colors in total).

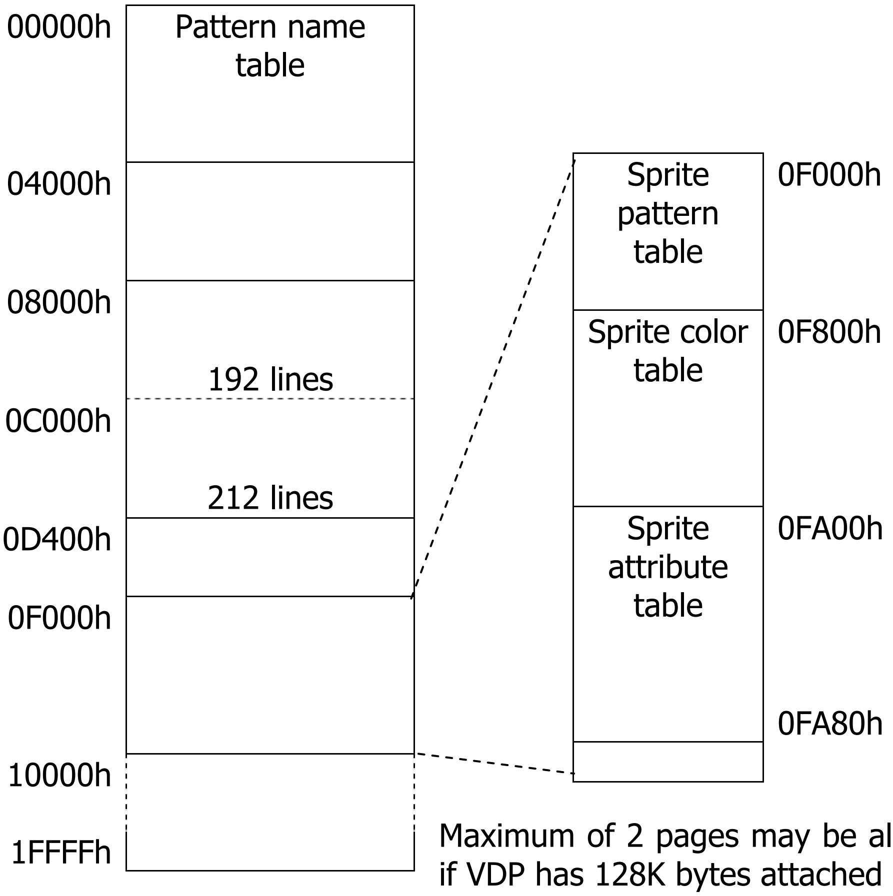
	MSB	7	6	5	4	3	2	1	0	LSB
R#7		DB7	DB6	DB5	DB4	BD3	BD2	BD1	BD0	Screen margin color
		Screen margin / backdrop color								

3.9.3. Sprite settings

Set the start address of the sprite attribute table in registers R#5 and R#11; set start address of the sprite pattern generator table in register R#6. For details about sprites please refer to section "Sprite mode 2"

	MSB	7	6	5	4	3	2	1	0	LSB
R#5		A14	A13	A12	A11	A10	1	1	1	Sprite attribute table low
R#11		0	0	0	0	0	0	A16	A15	Sprite attribute table high
R#6		0	0	A16	A15	A14	A13	A12	A11	Sprite pattern generator table

3.9.4. Example of VRAM allocation for GRAPHIC7 mode



Maximum of 2 pages may be allocated in the same manner if VDP has 128K bytes attached to it.

4. COMMANDS

4.1. Types of commands

Commands are used to perform specific complex operations on the video memory, and thus on the image displayed on the screen. See the list of available commands in the table below.

Command name	Destination	Source	Unit	Mnemonic	CM3	CM2	CM1	CM0
High-speed move	VRAM	CPU	Byte	**HMMC***	1	1	1	1
	VRAM	VRAM		**YMMM***	1	1	1	0
	VRAM	VRAM		**HMMM**	1	1	0	1
	VRAM	VDP		**HMMV**	1	1	0	0
Logical move	VRAM	CPU	Dot	**LMMC**	1	0	1	1
	CPU	VRAM		**LMCM**	1	0	1	0
	VRAM	VRAM		**LMMM**	1	0	0	1
	VRAM	VDP		**LMMV**	1	0	0	0
Line	VRAM	VDP		**LINE**	0	1	1	1
Search	VRAM	VDP		**SRCH**	0	1	1	0
Pset	VRAM	VDP		**PSET**	0	1	0	1
Point	VDP	VRAM		**POINT**	0	1	0	0
Invalid					0	0	1	1
					0	0	1	0
					0	0	0	1
Stop				**STOP**	0	0	0	0

*In G4 and G6 modes, the lower one bit, and in G5 mode, the lower two bits are lost in registers related to X-coordinate (DX, NX)

The process of execution of VDP commands consists of several steps:

- Ensure that current mode is G4 – G7. In other modes result is not guaranteed
- Check the bit 0 (CE, command execution) flag in status register S#2 to be 0. If it’s 1, then previous command is in progress and program needs to wait for completion or issue STOP command
- Set necessary parameters for command execution in registers R#32 to R#45 as necessary. It is easy to write whole the set of values to registers using indirect register addressing mode with auto-increment turned on
- Write command code to the R#46 (CMR, command register)
- Wait till command execution is completed by checking bit 0 (CE) of S#2 to be 0
- If current command needs to be aborted, execute STOP command

4.2. Page concept

As we have already seen, programmer can have several options to place tables in the video memory by altering base address registers. In some modes there're more options (like in Text 1 there're 32 options and in GRAPHICS7 there're only 2 options). This introduces a concept of the page defined by where VDP currently operates and where it takes information for picture display from.

Page concept is stricter in relation to VDP commands. VRAM access is defined by X and Y coordinates, and not by physical address within VRAM. Coordinate X is specified by 9 bits, and can be in range of 0…511, coordinate Y is specified by 10 bits, and can be in range of 0…1023. See the table below for VRAM paging in various video modes.

GRAPHIC4			Address	GRAPHIC5		
(0, 0)		(255, 0)	0000h	(0, 0)		(511, 0)
	Page 0				Page 0	
(0, 255)		(255, 255)		(0, 255)		(511, 255)
(0, 256)		(255, 256)	08000h	(0, 256)		(511, 256)
	Page 1				Page 1	
(0, 511)		(255, 511)		(0, 511)		(511, 511)
(0, 512)		(255, 512)	10000h	(0, 512)		(511, 512)
	Page 2				Page 2	
(0, 767)		(255, 767)		(0, 767)		(511, 767)
(0, 768)		(255, 768)	18000h	(0, 768)		(511, 768)
	Page 3				Page 3	
(0, 1023)		(255, 1023)	1FFFFh	(0, 1023)		(511, 1023)

GRAPHIC7				GRAPHIC6		
(0, 0)		(255, 0)	0000h	(0, 0)		(511, 0)
	Page 0				Page 0	
(0, 255)		(255, 255)		(0, 255)		(511, 255)
(0, 256)		(255, 256)	10000h	(0, 256)		(511, 256)
	Page 1				Page 1	
(0, 511)		(255, 511)	1FFFFh	(0, 511)		(511, 511)

In G4 and G5 modes there're four pages; in G6 and G7 modes there're only two pages. For example, setting Y coordinate in G5 mode to 658 will automatically choose page #2 with specific translated initial VRAM address.

Note that VDP is capable of displaying maximum of 212 lines, and programmer has an option to use register R#23 to set visible screen window position within active page.

4.3. Logical operations

When executing logical commands LINE, PSET and LOGICAL MOVE, it is possible to define logical operation to be done on the color of the pixels. The four bits identifying the logical operation should be written in lower four bits of R#46 (command register) together with the command code.

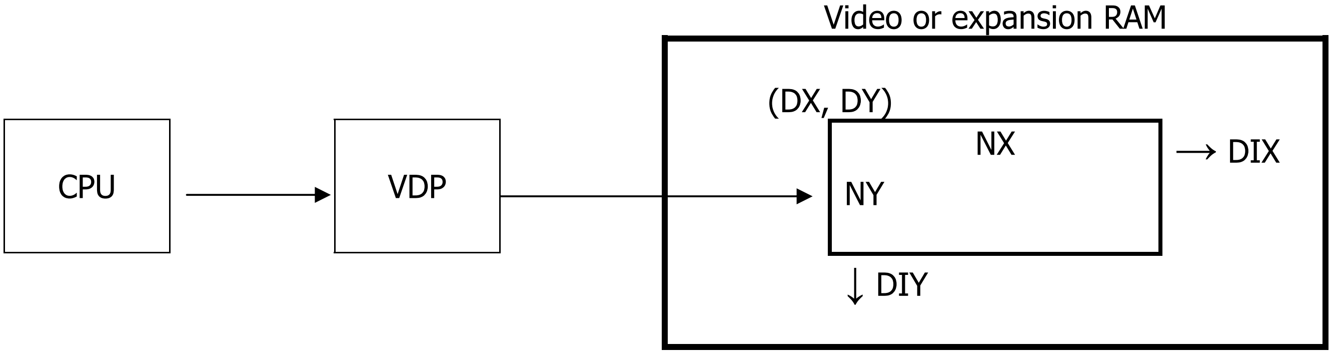
Name	Operation	LO3	LO2	LO1	LO0
IMP	DC=SC	0	0	0	0
AND	DC=SC&DC	0	0	0	1
OR	DC=SC|DC	0	0	1	0
XOR	DC=SC^DC	0	0	1	1
NOT	DC=!SC	0	1	0	0
Invalid		0	1	0	1
		0	1	1	0
		0	1	1	1
TIMP	If SC=0 then DC=DC else DC=SC	1	0	0	0
TAND	If SC=0 then DC=DC else DC=SC&DC	1	0	0	1
TOR	If SC=0 then DC=DC else DC=SC|DC	1	0	1	0
TXOR	If SC=0 then DC=DC else DC=SC^DC	1	0	1	1
TNOT	If SC=0 then DC=DC else DC=!SC	1	1	0	0
Invalid		1	1	0	1
		1	1	1	0
		1	1	1	1

*SC = source color code
*DC = destination color code

4.4. Explanation of the commands

4.4.1. HMMC (High speed move CPU to VRAM)

HMMC command is used to transfer data from the CPU to video or expansion RAM into a specified rectangular area through VDP. When using this command, note the limitation on the X coordinate in various modes (255 or 511).



HMMC execution order

Step 1: Set necessary coordinates in command registers

MSB	7	6	5	4	3	2	1	0	LSB
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX*: Destination X
R#37	0	0	0	0	0	0	0	DX8	
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY: Destination Y
R#39	0	0	0	0	0	0	DY9	DY8	
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX*: Number of dots in X-axis
R#41	0	0	0	0	0	0	0	NX8	
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY: Number of dots in Y-axis
R#43	0	0	0	0	0	0	NY9	NY8	

*In G4 and G6 modes, the lower one bit, and in G5 mode, the lower two bits are lost in registers related to X-coordinate (DX, NX)

Step 2: Set color register value

The first byte transferred from CPU after starting executing the HMMC command should be located in color register R#44 (CLR). Format of color data depends on the graphics mode.

MSB	7	6	5	4	3	2	1	0	LSB
R#44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	G4, G6
	X=2N				X=2N+1				(N=0..127)
	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	G5 (N=0…127)
	X=4N		X=4N+1		X=4N+2		X=4N+3		
	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	G7
	X=N (One dot)								

<u>Step 3:</u> Select destination memory and direction from base coordinate

MSB	7	6	5	4	3	2	1	0	LSB
R#45	0	-	MXD	-	DIY	DIX	-	-	

DIX: 0: Right, 1: Left — X transfer direction

DIY: 0: Down, 1: Up — Y transfer direction

MXD: 0: VRAM, 1: ExpRAM — Destination select

<u>Step 4:</u> Execute the command

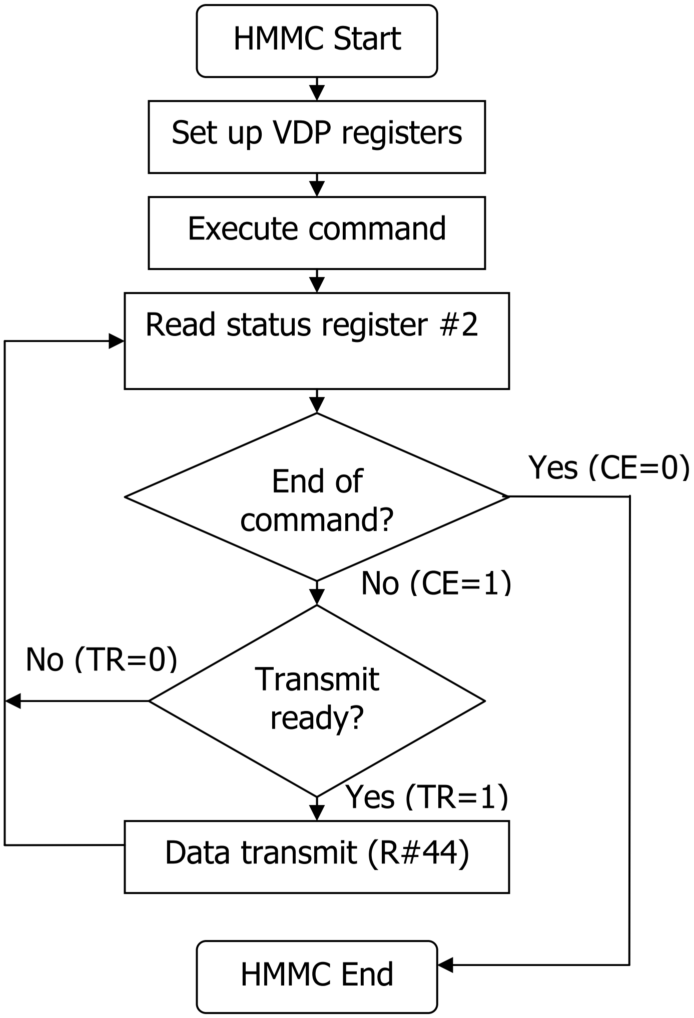
MSB	7	6	5	4	3	2	1	0	LSB
R#46	1	1	1	1	-	-	-	-	HMMC cmd

<u>Step 5:</u> Send data and wait for completion

While command is being executed by VDP, CE bit of status register S#2 will be set to 1. When command is complete, it will be reset to 0.

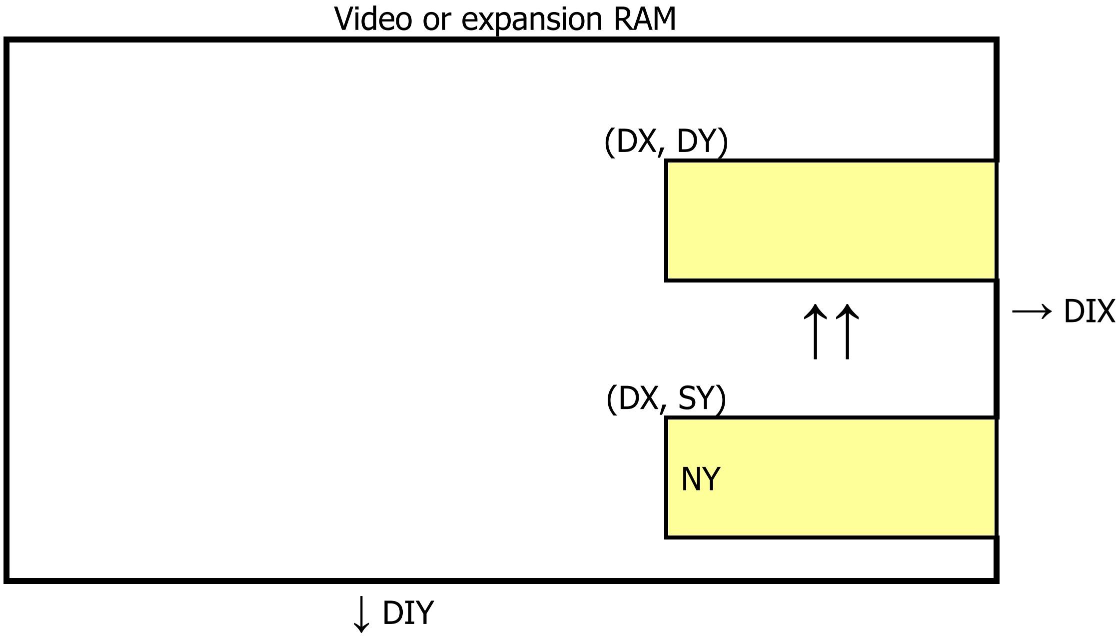
When VDP sets TR bit of status register S#2 to "1" application can send next data to the VDP color register R#44 (CLR). If TR bit is 0, then application should not send data and wait till TR bit is set to 1 or terminate the command if needed.

Flowchart of HMMC execution from CPU point of view



4.4.2. YMMM (High speed move VRAM to VRAM, Y coordinate only)

YMMM command is used to transfer data from the area specified by DX, SY, NY, DIX, DIY and the right (or left) edge of the screen, in the Y-direction determined by DY.



YMMM execution order

Step 1: Set necessary coordinates in command registers

	MSB	7	6	5	4	3	2	1	0	LSB
R#34		SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY: Source
R#35		0	0	0	0	0	0	SY9	SY8	transfer point Y
R#36		DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX*: Source
R#37		0	0	0	0	0	0	0	DX8	transfer point X
R#38		DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY: Destination
R#39		0	0	0	0	0	0	DY9	DY8	transfer point Y
R#42		NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY: Number of
R#43		0	0	0	0	0	0	NY9	NY8	dots in Y-axis

*In G4 and G6 modes, the lower one bit, and in G5 mode, the lower two bits are lost in registers related to X-coordinate (DX, NX)

Step 2: Select destination memory and direction from base coordinate

	MSB	7	6	5	4	3	2	1	0	LSB
R#45		0	-	MXD	-	DIY	DIX	-	-	

0: Right 1: Left	X transfer direction
0: Down 1: Up	Y transfer direction
0: VRAM 1: ExpRAM	Destination select

Step 3: Execute the command

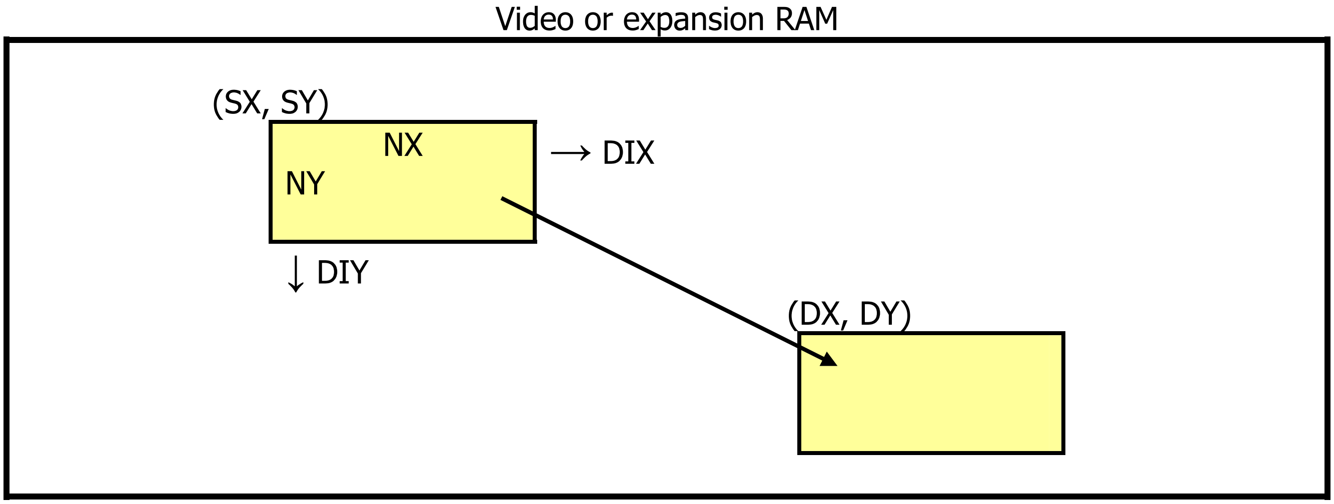
	MSB	7	6	5	4	3	2	1	0	LSB	
R#46		1	1	1	0	-	-	-	-		YMMM cmd

Step 4: Wait for command execution completion

While command is being executed by VDP, CE bit of status register S#2 will be set to 1. When command is complete, it will be reset to 0.

4.4.3. HMMM (High speed move VRAM to VRAM)

HMMM command is used to transfer data from one specific rectangular area in VRAM of expansion RAM to another area within VRAM of expansion RAM. Note the limitation on the X coordinate, which is dependent on the current graphics mode (255 or 511).



HMMM execution order

<u>Step 1:</u> Set necessary coordinates in command registers

	MSB	7	6	5	4	3	2	1	0	LSB
R#32		SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	SX*: Source
R#33		0	0	0	0	0	0	0	SX8	transfer point X
R#34		SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY: Source
R#35		0	0	0	0	0	0	SY9	SY8	transfer point Y
R#36		DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX*:Destination
R#37		0	0	0	0	0	0	0	DX8	transfer point X
R#38		DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY: Destination
R#39		0	0	0	0	0	0	DY9	DY8	transfer point Y
R#40		NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX*: Number of
R#41		0	0	0	0	0	0	0	NX8	dots in x-axis
R#42		NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY: Number of
R#43		0	0	0	0	0	0	NY9	NY8	dots in Y-axis

*In G4 and G6 modes, the lower one bit, and in G5 mode, the lower two bits are lost in registers related to X-coordinate (DX, NX)

<u>Step 2:</u> Select destination memory and direction from base coordinate

	MSB	7	6	5	4	3	2	1	0	LSB
R#45		0	-	MXD	MSX	DIY	DIX	-	-	

0: Right 1: Left	X transfer direction
0: Down 1: Up	Y transfer direction
0: VRAM 1: ExpRAM	Source location select
0: VRAM 1: ExpRAM	Destination location select

<u>Step 3:</u> Execute the command

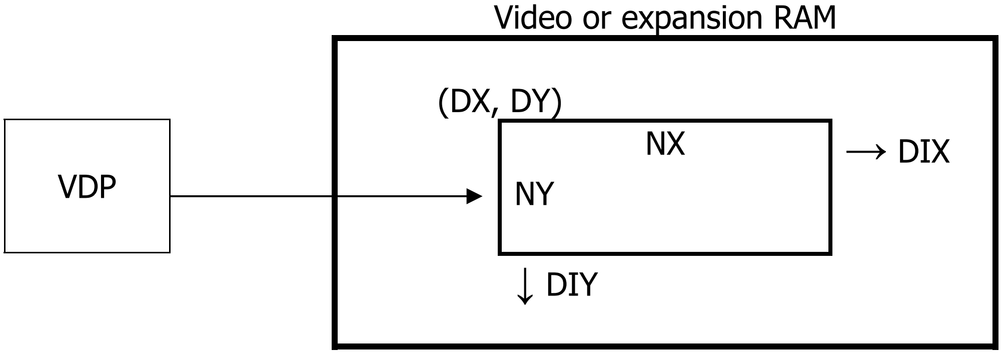
	MSB	7	6	5	4	3	2	1	0	LSB	
R#46		1	1	0	1	-	-	-	-		HMMM cmd

<u>Step 4:</u> Wait for command execution completion

While command is being executed by VDP, CE bit of status register S#2 will be set to 1. When command is complete, it will be reset to 0.

4.4.4. HMMV (High speed move VDP to VRAM)

HMMV command is used to paint in a specific rectangular area in the VRAM or expansion RAM. When using this command, note the limitation on the X coordinate in various modes (255 or 511).



HMMV execution order

Step 1: Set necessary coordinates in command registers

	MSB 7	6	5	4	3	2	1	0 LSB	
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX*:
R#37	0	0	0	0	0	0	0	DX8	Destination X
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY: Destination
R#39	0	0	0	0	0	0	DY9	DY8	Y
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX*: Number of
R#41	0	0	0	0	0	0	0	NX8	dots in X-axis
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY: Number of
R#43	0	0	0	0	0	0	NY9	NY8	dots in Y-axis

*In G4 and G6 modes, the lower one bit, and in G5 mode, the lower two bits are lost in registers related to X-coordinate (DX, NX)

Step 2: Set color register value

The first byte transferred from CPU after starting executing the HMMC command should be located in color register R#44 (CLR). Format of color data depends on the graphics mode.

	MSB	7	6	5	4	3	2	1	0	LSB	
R#44		CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0		G4, G6
		X=2N				X=2N+1					(N=0..127)
		CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0		G5 (N=0…127)
		X=4N		X=4N+1		X=4N+2		X=4N+3			
		CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0		G7
		X=N (One dot)									

Step 3: Select destination memory and direction from base coordinate

	MSB	7	6	5	4	3	2	1	0	LSB
R#45		0	-	MXD	-	DIY	DIX	-	-	

0: Right 1: Left	X transfer direction
0: Down 1: Up	Y transfer direction
0: VRAM 1: ExpRAM	Destination select

Step 4: Execute the command

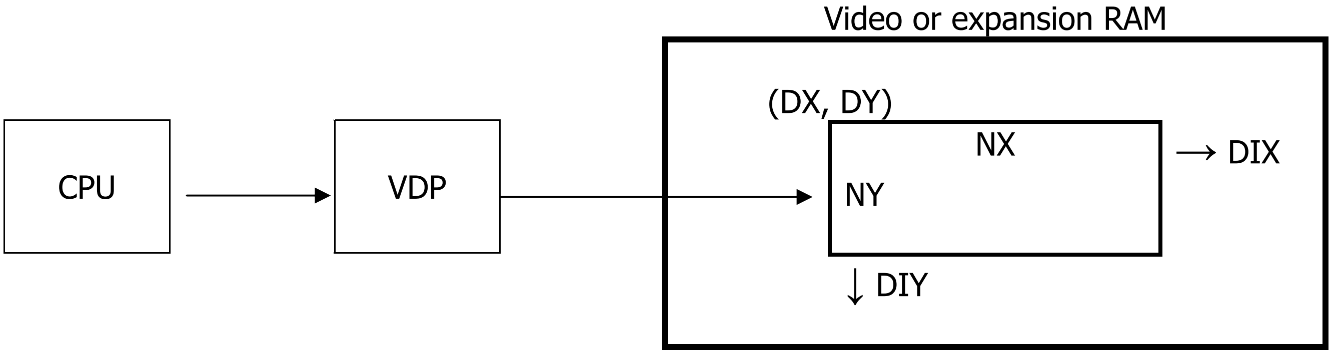
	MSB	7	6	5	4	3	2	1	0	LSB	
R#46		1	1	0	0	-	-	-	-		HMMV cmd

Step 5: Wait for command execution completion

While command is being executed by VDP, CE bit of status register S#2 will be set to 1. When command is complete, it will be reset to 0.

4.4.5. LMMC (Logical move CPU to VRAM)

LMMC command is used to transfer data from the CPU to video or expansion RAM into a specified rectangular area through VDP. The units used are dots.



LMMC execution order

Step 1: Set necessary coordinates in command registers

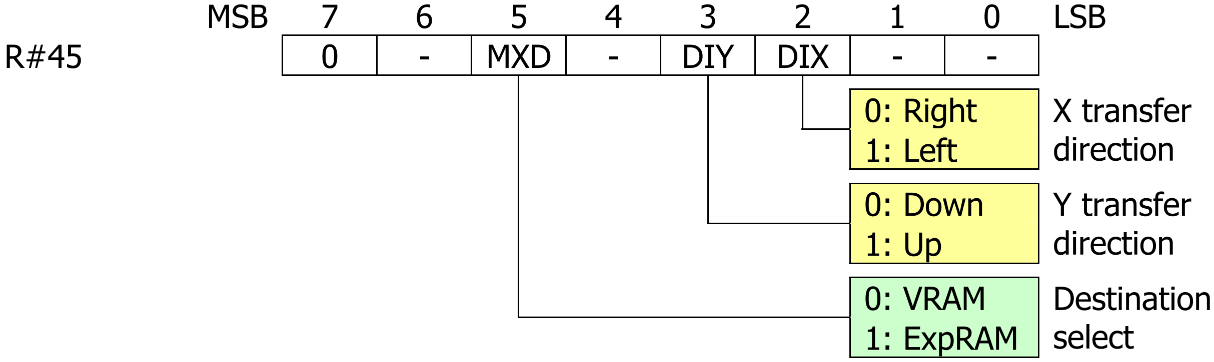
MSB	7	6	5	4	3	2	1	0	LSB
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX: Destination
R#37	0	0	0	0	0	0	0	DX8	X (0…511)
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY: Destination
R#39	0	0	0	0	0	0	DY9	DY8	Y (0…1023)
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX: Number of
R#41	0	0	0	0	0	0	0	NX8	dots in X-axis
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY: Number of
R#43	0	0	0	0	0	0	NY9	NY8	dots in Y-axis

Step 2: Set color register value

The first byte transferred from CPU after starting executing the LMMC command should be located in color register R#44 (CLR). Format of color data depends on the graphics mode.

MSB	7	6	5	4	3	2	1	0	LSB
R#44	-	-	-	-	CR3	CR2	CR1	CR0	G4, G6
	-	-	-	-	-	-	CR1	CR0	G5
	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	G7

Step 3: Select destination memory and direction from base coordinate



	MSB	7	6	5	4	3	2	1	0	LSB
R#45		0	-	MXD	-	DIY	DIX	-	-	

DIX: 0: Right 1: Left — X transfer direction

DIY: 0: Down 1: Up — Y transfer direction

MXD: 0: VRAM 1: ExpRAM — Destination select

Step 4: Execute the LMMC command

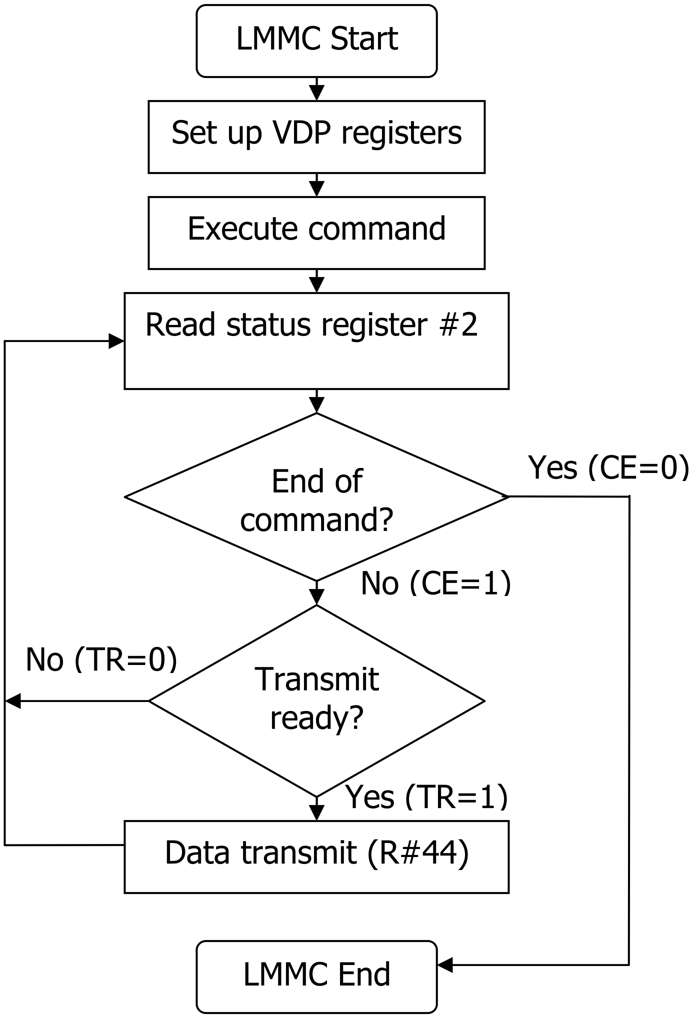
	MSB	7	6	5	4	3	2	1	0	LSB
R#46		1	0	1	1	LO3	LO2	LO1	LO0	LMMC cmd
						Logical operation				

Step 5: Send data and wait for completion

While command is being executed by VDP, CE bit of status register S#2 will be set to 1. When command is complete, it will be reset to 0.

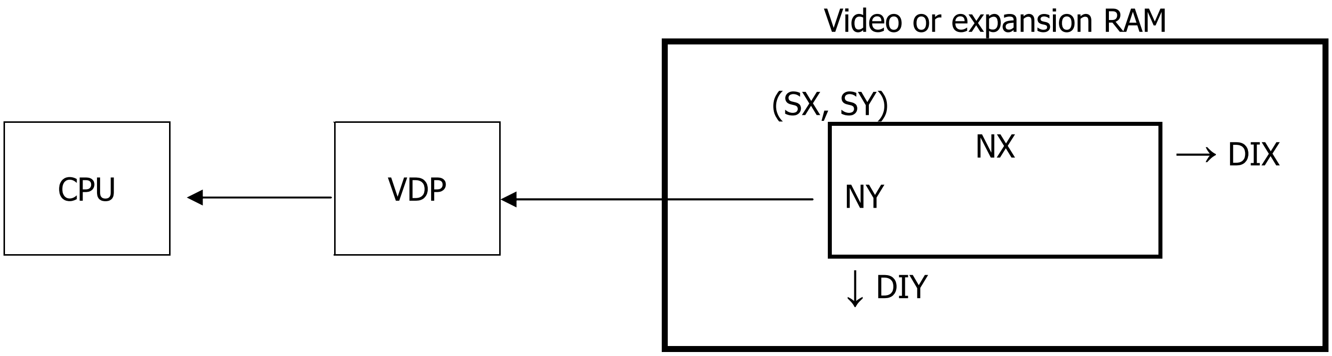
When VDP sets TR bit of status register S#2 to "1" application can send next data to the VDP color register R#44 (CLR). If TR bit is 0, then application should not send data and wait till TR bit is set to 1 or terminate the command if needed.

Flowchart of LMMC execution from CPU point of view



4.4.6. LMCM (Logical move VRAM to CPU)

LMCM command is used to transfer data from the video or expansion RAM in a specified rectangular area to the CPU through VDP. The units used are dots.



LMCM execution order

Step 1: Set necessary coordinates in command registers

MSB	7	6	5	4	3	2	1	0	LSB
R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	SX: Source X
R#33	0	0	0	0	0	0	0	SX8	(0…511)
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY: Source Y
R#35	0	0	0	0	0	0	SY9	SY8	(0…1023)
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX: Number of
R#41	0	0	0	0	0	0	0	NX8	dots in X-axis
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY: Number of
R#43	0	0	0	0	0	0	NY9	NY8	dots in Y-axis

Step 2: Select source memory and direction from base coordinate

MSB	7	6	5	4	3	2	1	0	LSB
R#45	0	-	-	MXS	DIY	DIX	-	-	

DIX	0: Right 1: Left	X transfer direction
DIY	0: Down 1: Up	Y transfer direction
MXS	0: VRAM 1: ExpRAM	Source select

Step 3: Clear the TR flag

Read the value from status register S#7. This step is required in order to clear the TR flag. Please refer to flowchart in this section for the LMCM command.

Step 4: Execute the LMCM command

	MSB	7	6	5	4	3	2	1	0	LSB
R#46		1	0	1	0	-	-	-	-	LMCM cmd

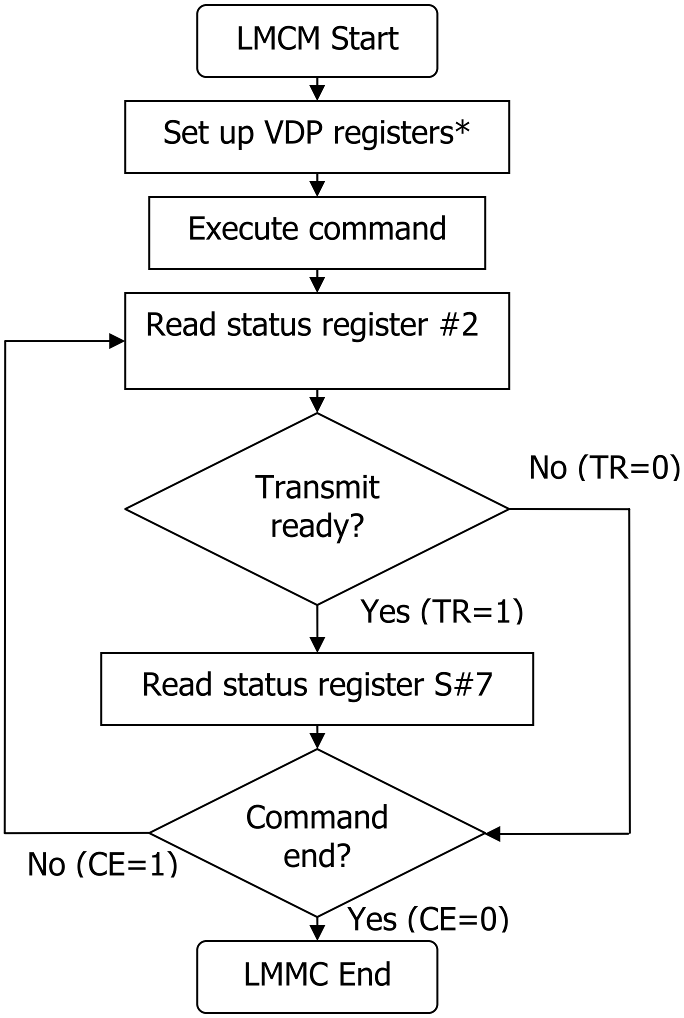
Step 5: Read dot color code and check fir command end

Use status register S#7 to get the color code of the dot. Format of color data depends on the graphics mode.

	MSB	7	6	5	4	3	2	1	0	LSB
S#7		-	-	-	-	C3	C2	C1	C0	G4, G6
		-	-	-	-	-	-	C1	C0	G5
		C7	C6	C5	C4	C3	C2	C1	C0	G7

Check CE bit of status register S#2 for command completion. CE set to 0 is the only sign of the completed LMCM command.

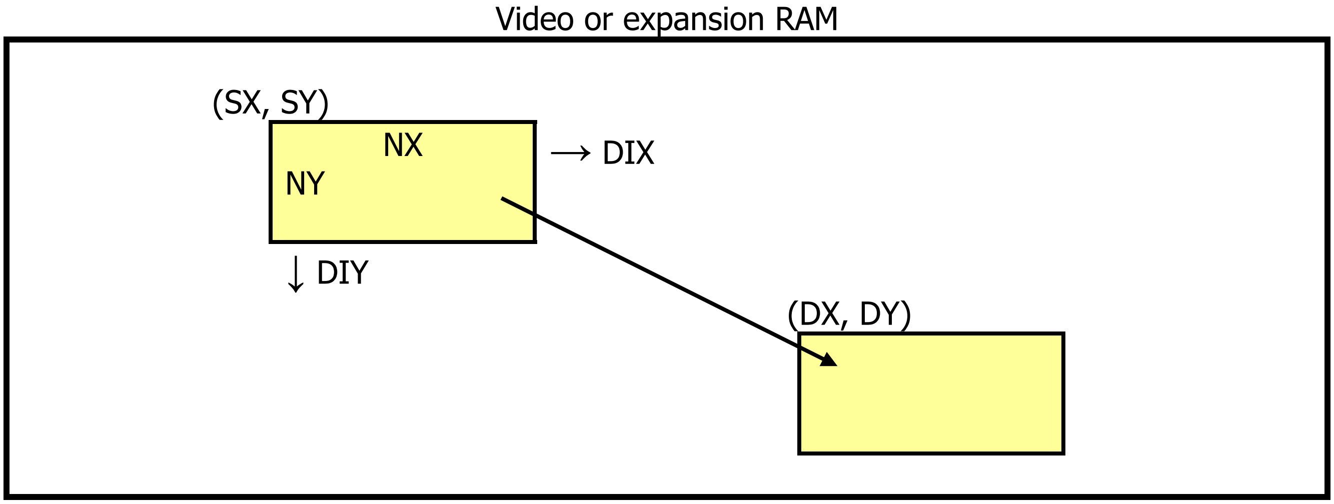
Flowchart of LMCM execution from CPU point of view



*TR must be cleared before command execution

4.4.7. LMMM (Logical move VRAM to VRAM)

LMMM command is used to transfer data from one specific rectangular area in VRAM of expansion RAM to another area within VRAM of expansion RAM. The units used are dots.



LMMM execution order

Step 1: Set necessary coordinates in command registers

	MSB	7	6	5	4	3	2	1	0	LSB	
R#32		SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0		SX: Source
R#33		0	0	0	0	0	0	0	SX8		transfer point X
R#34		SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0		SY: Source
R#35		0	0	0	0	0	0	SY9	SY8		transfer point Y
R#36		DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0		DX: Destination
R#37		0	0	0	0	0	0	0	DX8		transfer point X
R#38		DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0		DY: Destination
R#39		0	0	0	0	0	0	DY9	DY8		transfer point Y
R#40		NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0		NX: Number of
R#41		0	0	0	0	0	0	0	NX8		dots in x-axis
R#42		NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0		NY: Number of
R#43		0	0	0	0	0	0	NY9	NY8		dots in Y-axis

<u>Step 2:</u> Select destination memory and direction from base coordinate

	MSB	7	6	5	4	3	2	1	0	LSB
R#45		0	-	MXD	MSX	DIY	DIX	-	-	

DIX	0: Right 1: Left	X transfer direction
DIY	0: Down 1: Up	Y transfer direction
MSX	0: VRAM 1: ExpRAM	Source location select
MXD	0: VRAM 1: ExpRAM	Destination location select

<u>Step 3:</u> Define logical operation and execute the command

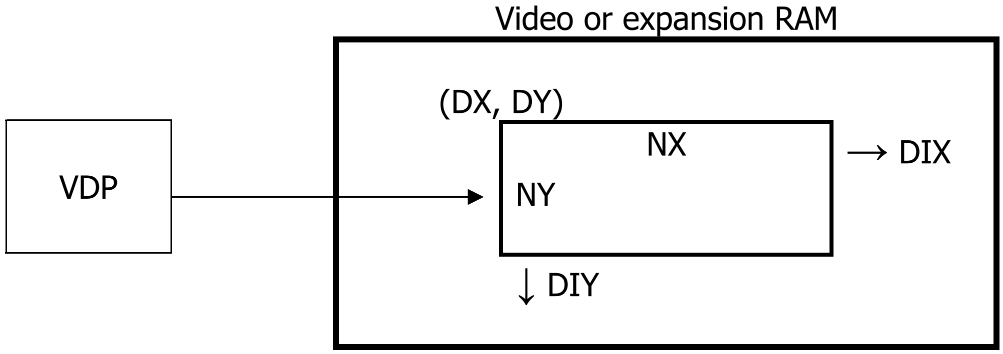
	MSB	7	6	5	4	3	2	1	0	LSB	
R#46		1	0	0	1	LO3	LO2	LO1	LO0		LMMM cmd
						Logical operation					

<u>Step 4:</u> Check for command completion

CPU should check CE bit of status register S#2 to identify if VDP has completed execution of the command. When command is being executed, CE bit is set to 1; when command is complete, CE bit will be reset to 0.

4.4.8. LMMV (Logical move VDP to VRAM)

LMMV command is used to paint in a specific rectangular area in the VRAM or expansion RAM. The units used are dots.



LMMV execution order

Step 1: Set necessary coordinates in command registers

MSB	7	6	5	4	3	2	1	0	LSB
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX: Destination
R#37	0	0	0	0	0	0	0	DX8	X (dots)
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY: Destination
R#39	0	0	0	0	0	0	DY9	DY8	Y (dots)
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX: Number of
R#41	0	0	0	0	0	0	0	NX8	dots in X-axis
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY: Number of
R#43	0	0	0	0	0	0	NY9	NY8	dots in Y-axis

Step 2: Set color register value

The color code to use when painting an area should be located in color register R#44 (CLR). Format of color data depends on the graphics mode.

MSB	7	6	5	4	3	2	1	0	LSB
R#44	-	-	-	-	C3	C2	C1	C0	G4, G6
	-	-	-	-	-	-	C1	C0	G5
	C7	C6	C5	C4	C3	C2	C1	C0	G7

Step 3: Select destination memory and direction from base coordinate

	MSB	7	6	5	4	3	2	1	0	LSB
R#45		0	-	MXD	-	DIY	DIX	-	-	

0: Right 1: Left	X transfer direction
0: Down 1: Up	Y transfer direction
0: VRAM 1: ExpRAM	Destination select

Step 4: Execute the command

	MSB	7	6	5	4	3	2	1	0	LSB
R#46		1	0	0	0	-	-	-	-	LMMV cmd

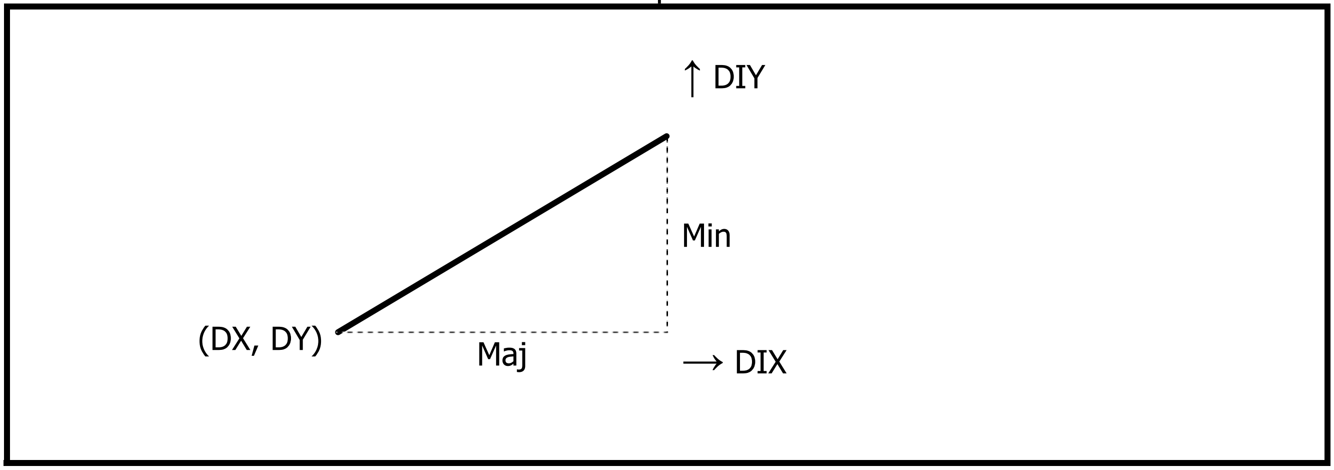
Step 5: Check for command completion

CPU should check CE bit of status register S#2 to identify if VDP has completed execution of the command. When command is being executed, CE bit is set to 1; when command is complete, CE bit will be reset to 0.

4.4.9. LINE

LINE command is used to draw straight line in VRAM of expansion RAM. The line drawn is the hypotenuse of the triangle defined by the "long" and "short" sides. The distances are defined from the single starting point. Words "long" and "short" are used to identify respective sets of registers to use to define triangle's sides: long side is defined in registers R#40 and R#41 (by 10 bits MJ9…MJ0 with value in the range 0…1023) and short side is defined in registers R#42 and R#43 (by 9 bits MI8…MI0 with value in the range 0…511). The units used are dots.

Video or expansion RAM



LINE execution order

<u>Step 1:</u> Set necessary coordinates in command registers

	MSB 7	6	5	4	3	2	1	0 LSB	
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX: Starting point X
R#37	0	0	0	0	0	0	0	DX8	
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY: Starting point Y
R#39	0	0	0	0	0	0	DY9	DY8	
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	Maj (NX): long side dots num
R#41	0	0	0	0	0	0	0	NX8	
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	Min (NY): short side dots num
R#43	0	0	0	0	0	0	NY9	NY8	

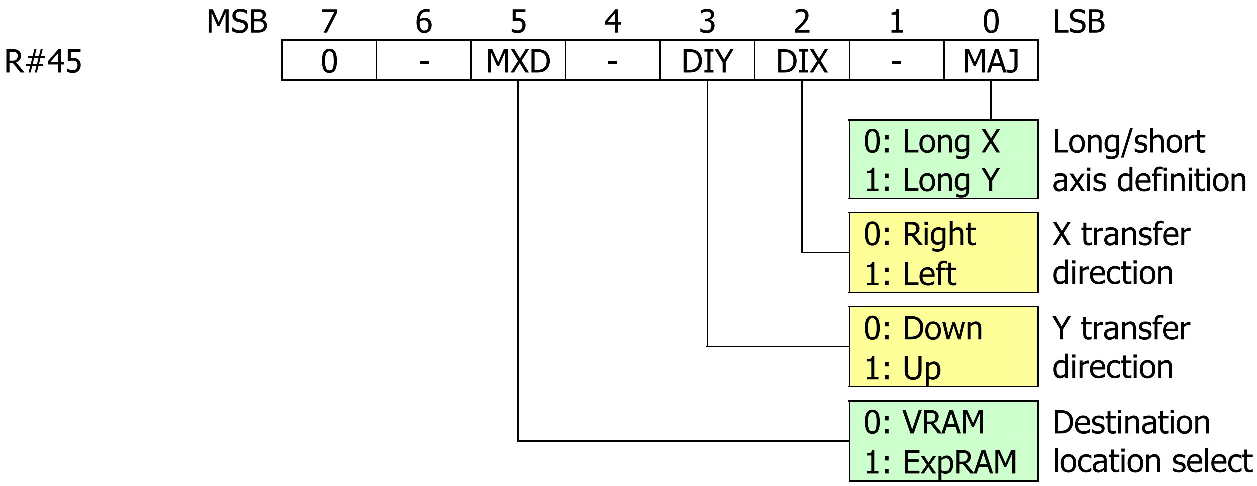
Step 2: Set color register value

Color of the resulting line mask is coded in color register R#44 (CLR). Format of color data depends on the graphics mode.

	MSB 7	6	5	4	3	2	1	0 LSB	
R#44	-	-	-	-	C3	C2	C1	C0	G4, G6
	-	-	-	-	-	-	C1	C0	G5
	C7	C6	C5	C4	C3	C2	C1	C0	G7

Step 3: Select destination memory, direction from base coordinate and orientation

Bit MAJ of the register R#45 controls which coordinate is defined in registers R#40-R#43. If this bit is 0, then long side (NY) defines triangle's side by X-axis and short side (NX) defines triangle's side by Y-axis; if this bit is 1, then long side (NY) defines triangle's side by Y-axis and short side (NX) defines triangle's side by X-axis.



	MSB 7	6	5	4	3	2	1	0 LSB
R#45	0	-	MXD	-	DIY	DIX	-	MAJ

MAJ: 0: Long X, 1: Long Y — Long/short axis definition

DIX: 0: Right, 1: Left — X transfer direction

DIY: 0: Down, 1: Up — Y transfer direction

MXD: 0: VRAM, 1: ExpRAM — Destination location select

Step 4: Define logical operation and execute the command

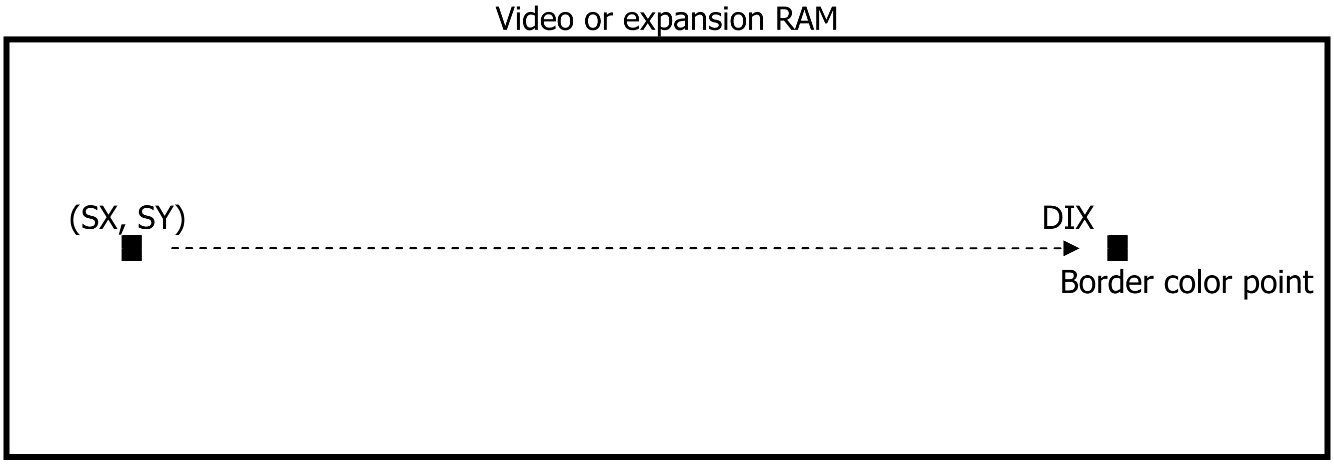
	MSB 7	6	5	4	3	2	1	0 LSB	
R#46	0	1	1	1	LO3	LO2	LO1	LO0	LINE cmd
					Logical operation				

Step 5: Check for command completion

CPU should check CE bit of status register S#2 to identify if VDP has completed execution of the command. When command is being executed, CE bit is set to 1; when command is complete, CE bit will be reset to 0.

4.4.10. SRCH

SRCH command is used to search for the specific color in VRAM of expansion RAM to the right or left of the starting point. The units used are dots.



SRCH execution order

Step 1: Set necessary coordinates in command registers

MSB	7	6	5	4	3	2	1	0	LSB
R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	SX: Starting point X
R#33	0	0	0	0	0	0	0	SX8	
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY: Starting point Y
R#35	0	0	0	0	0	0	SY9	SY8	

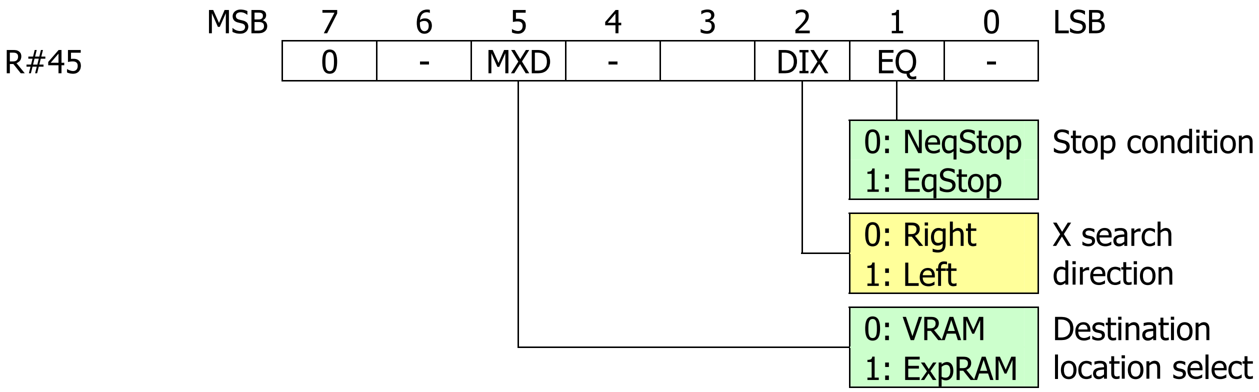
Step 2: Set color register value

Color to search for should be coded in color register R#44 (CLR). Format of color data depends on the graphics mode.

MSB	7	6	5	4	3	2	1	0	LSB
R#44	-	-	-	-	C3	C2	C1	C0	G4, G6
	-	-	-	-	-	-	C1	C0	G5
	C7	C6	C5	C4	C3	C2	C1	C0	G7

Step 3: Select destination memory, direction from base coordinate and orientation

Bit EQ of the register R#45 controls search stop condition. If it is set to 1, then SRCH command terminates when color, equal to R#44 coded color, is found; if it is set to 0, then SRCH command terminates when any other than R#44 coded color is found.



	MSB	7	6	5	4	3	2	1	0	LSB
R#45		0	-	MXD	-		DIX	EQ	-	

EQ: 0: NeqStop
1: EqStop — Stop condition

DIX: 0: Right
1: Left — X search direction

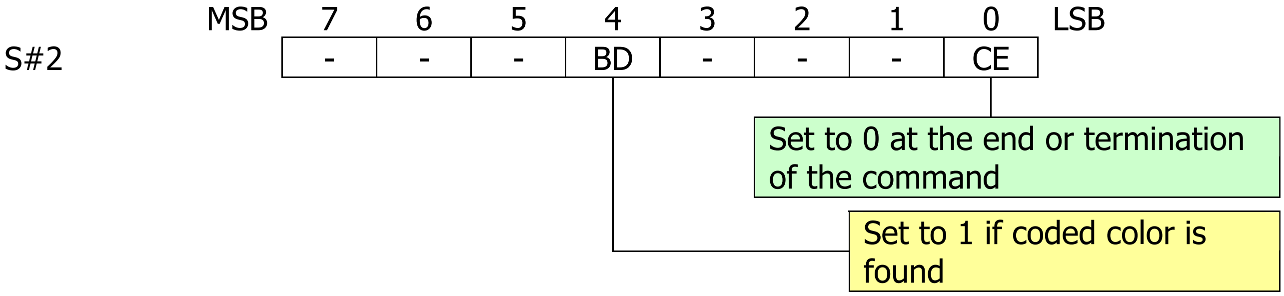
MXD: 0: VRAM
1: ExpRAM — Destination location select

<u>Step 4:</u> Execute the command

	MSB	7	6	5	4	3	2	1	0	LSB	
R#46		0	1	1	0	-	-	-	-		SRCH cmd

<u>Step 5:</u> Check for command termination or completion, and X-coordinate

Flag BD of the status register S#2 is set if coded color in register R#44 was found, otherwise this bit is reset. Current value of the X-coordinate can be read from the status registers S#8 and S#9.



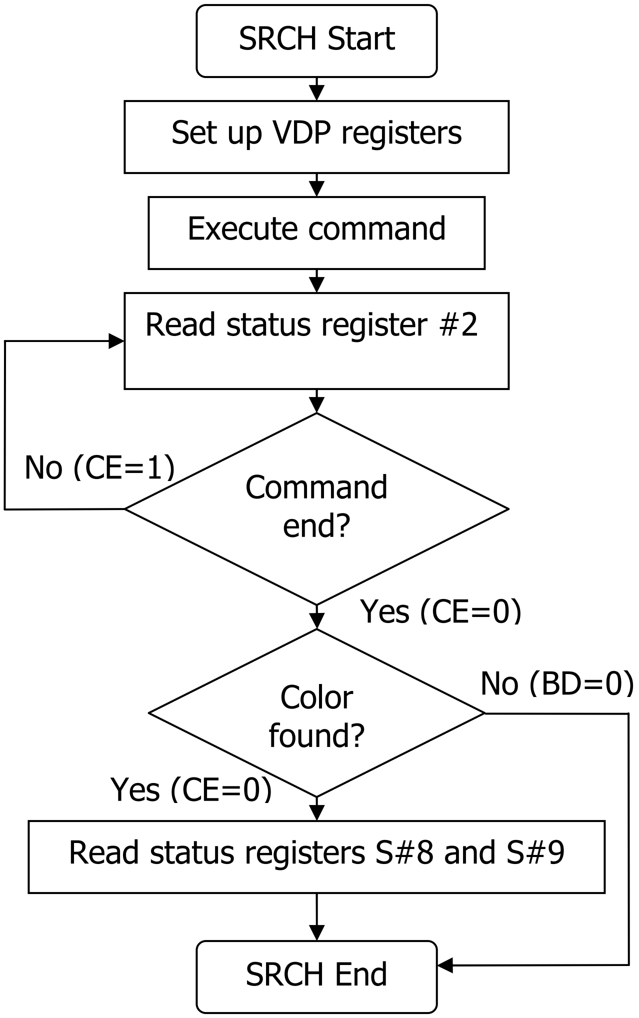
	MSB	7	6	5	4	3	2	1	0	LSB
S#2		-	-	-	BD	-	-	-	CE	

CE: Set to 0 at the end or termination of the command

BD: Set to 1 if coded color is found

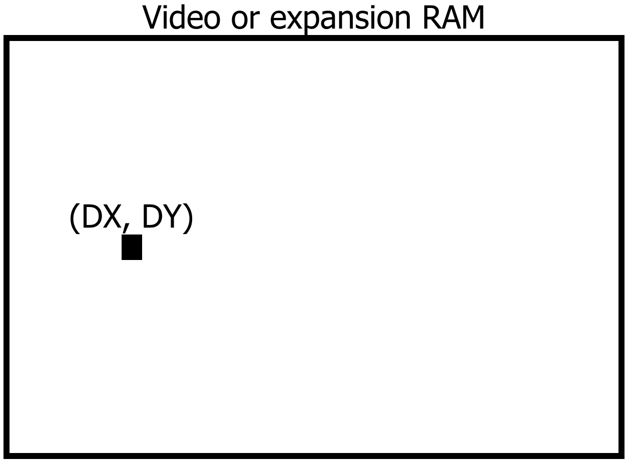
	MSB	7	6	5	4	3	2	1	0	LSB	
S#8		BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0		Current X-coordinate
S#9		1	1	1	1	1	1	1	BX8		

Flowchart of SRCH execution from CPU point of view



4.4.11. PSET

PSET command is used to draw a dot in VRAM of expansion RAM. Programmer can select logical operation on the existing color of the dot in the specified location. The units used are dots.



PSET execution order

Step 1: Set necessary coordinates in command registers

	MSB	7	6	5	4	3	2	1	0	LSB
R#36		DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX: Target
R#37		0	0	0	0	0	0	0	DX8	point X
R#38		DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY: Target
R#39		0	0	0	0	0	0	DY9	DY8	point Y

Step 2: Set color register value

Color of the dot should be coded in color register R#44 (CLR). Format of color data depends on the graphics mode. It is possible to do logical operation on the existing and new color of the defined dot through coding the code of logical operation in the CMR.

	MSB	7	6	5	4	3	2	1	0	LSB
R#44		-	-	-	-	C3	C2	C1	C0	G4, G6
		-	-	-	-	-	-	C1	C0	G5
		C7	C6	C5	C4	C3	C2	C1	C0	G7

Step 3: Select destination memory

	MSB	7	6	5	4	3	2	1	0	LSB
R#45		0	-	MXD	-	-	-	-	-	

0: VRAM
1: ExpRAM

Destination location select

Step 4: Define logical operation and execute the command

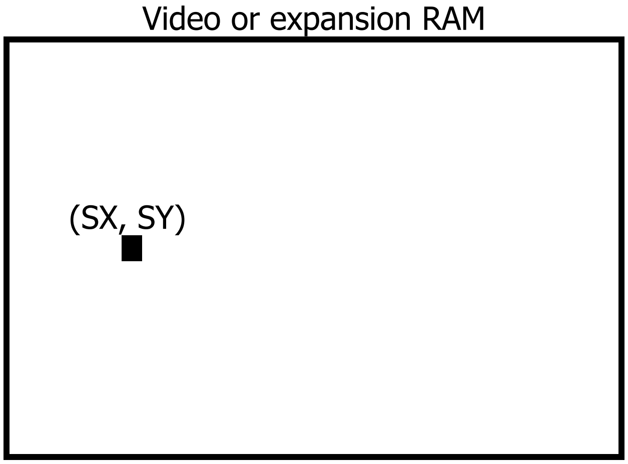
	MSB	7	6	5	4	3	2	1	0	LSB	
R#46		0	1	1	0	LO3	LO2	LO1	LO0		PSET cmd
						Logical operation					

Step 5: Check for command completion

CPU should check CE bit of status register S#2 to identify if VDP has completed execution of the command. When command is being executed, CE bit is set to 1; when command is complete, CE bit will be reset to 0.

4.4.12. POINT

POINT command is used to read the color of the specified dot located in VRAM of expansion RAM. The units used are dots.



POINT execution order

Step 1: Set necessary coordinates in command registers

MSB	7	6	5	4	3	2	1	0	LSB
R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	SX: Source
R#33	0	0	0	0	0	0	0	SX8	point X
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY: Source
R#35	0	0	0	0	0	0	SY9	SY8	point Y

Step 2: Select destination memory

MSB	7	6	5	4	3	2	1	0	LSB
R#45	0	-	-	-	MXS	-	-	-	ARG

MXS:
0: VRAM
1: ExpRAM
Destination location select

Step 3: Execute the command

MSB	7	6	5	4	3	2	1	0	LSB
R#46	0	1	0	0	-	-	-	-	POINT cmd

Step 4: Check for command completion

CPU should check CE bit of status register S#2 to identify if VDP has completed execution of the command. When command is being executed, CE bit is set to 1; when command is complete, CE bit will be reset to 0. After completion of the command, source dot color code can be read from status register S#7. Note that format of the data in various graphics mode differs.

MSB	7	6	5	4	3	2	1	0	LSB
S#7	-	-	-	-	C3	C2	C1	C0	G4, G6
	-	-	-	-	-	-	C1	C0	G5
	C7	C6	C5	C4	C3	C2	C1	C0	G7

4.5. Speeding up the processing of VDP commands

Programmer can use two methods to speed up VDP hardware acceleration commands. Use of both methods is related to decreasing VDP access to the VRAM and freeing available VRAM time slots for command execution.

- Disabling sprites: setting register R#8 bit SPD to 1 disables sprite processing
- Disabling screen display: setting register R#1 bit BL to 0 will completely disable screen display freeing significant memory access time for command execution

4.6. States of the registers after command execution

Setting up VDP registers for command execution is a significant task for programmer and CPU, and it will be wise to use already existing values in the registers for the next command execution or for further work with video memory. For this purpose programmer should know the resulting states of most important VDP registers. Please see the table below. Note that if program is going to use sequence of the VDP commands which use resulting values of registers from previous commands, it is important to disable interrupts so that interrupt handler routing would not accidentally change values in VDP registers and thus break the whole command sequence.

CMR H is a higher nibble of the command register R#46; CMR L is a lower nibble of the command register r#46. ARG is an argument register R#45.

The resulting values of SY*, DY* and NY* are in dots and can be calculated using equations below:

- If DIY=0 then SY*=SY+N; DY*=DY+N
- If DIY=1 then SY*=SY-N; DY*=DY-N
- NY*=NY-N

LINE command =➔ if MAJ=0 then N=N-1 ????

	SX	SY	DX	DY	NX	NY	CLR	CMR H	CMR L	ARG
HMMC	-	-	-	*	-	#	-	0	-	-
YMMM	-	*	-	*	-	#	-	0	-	-
HMMM	-	*	-	*	-	#	-	0	-	-
HMMV	-	-	-	*	-	#	-	0	-	-
LMMC	-	-	-	*	-	#	-	0	-	-
LMCM	-	*	-	-	-	#	-	0	-	-
LMMM	-	*	-	*	-	#	-	0	-	-
LMMV	-	-	-	*	-	#	-	0	-	-
LINE	-	-	-	*	-	-	-	0	-	-
SRCH	-	-	-	-	-	-	-	0	-	-
PSET	-	-	-	-	-	-	-	0	-	-
POINT	-	-	-	-	-	-	*	0	-	-

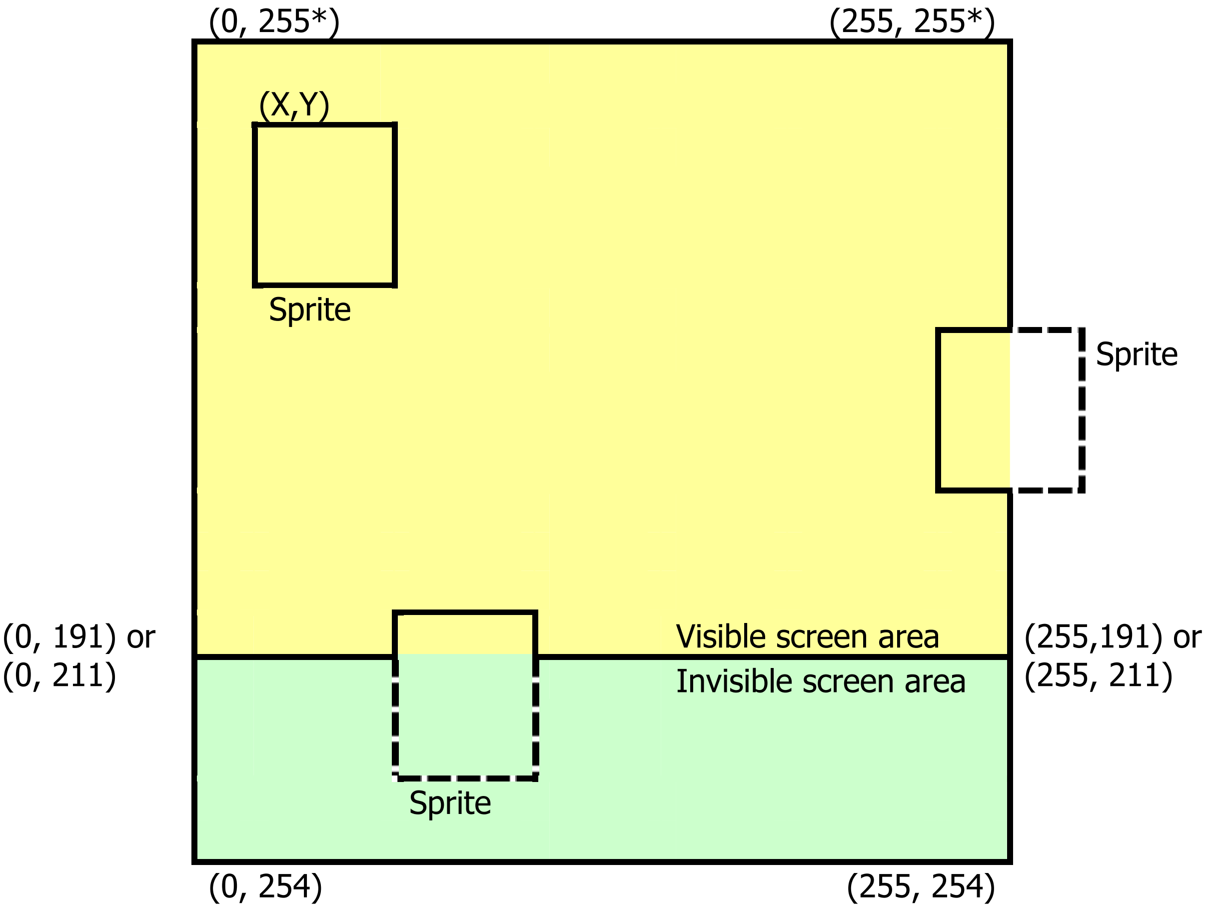
Legend:
- Unchanged
* Coordinate at the command end (SY*, DY*) or color code
# Count (NY*) when the end of the screen was detected

5. SPRITES

MSX-VIDEO can display up to 32 sprites on the screen. Depending on the sprite mode, sprite sizes can be 8*8 or 16*16 dots. X-axis coordinate of sprite location is always between 0…255, and this means that in graphics modes with 512 dots on the X-axis single sprite's dot occupy two horizontal dots of the screen.

Sprites can be placed anywhere on the screen. Sprites are processed independently of and do not affect other graphics. They can be independently switched on or off.

The following diagram conceptually shows how sprites are displayed. Note that visible screen area may have 192 or 212 pixels depending on the setting of LN bit of register R#9, and visible screen area can be vertically scrolled using vertical offset register R#23.



* They Y-coordinate of the upper edge of the sprite screen is 255

The MSX-VIDEO can work in two sprite display modes. Respective mode is selected during initialization of the screen mode.

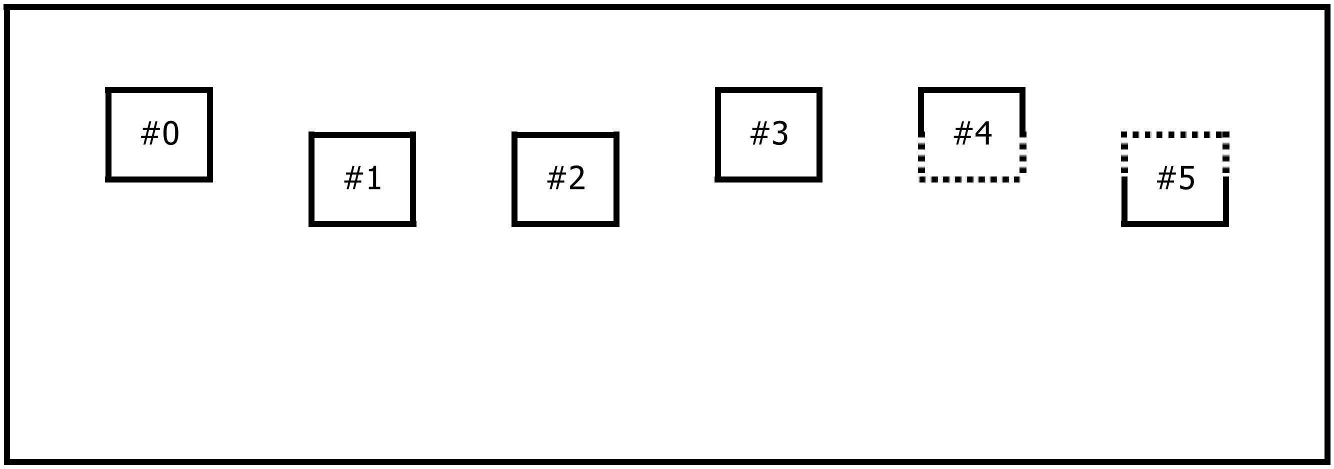
- Sprite mode 1: GRAPHIC1, GRAPHIC2, MULTICOLOR
- Sprite mode 2: GRAPHIC3 … GRAPHIC7

Sprites are not available in TEXT1 and TEXT2 modes.

5.1. SPRITE MODE 1 (G1, G2, MC)

While there's a room for 256 sprite patterns in sprite pattern generator table, VDP is only capable of displaying 32 sprites (#0…#31) – limiting factor is sprite attribute table which can hold attributes for 32 *active* sprites only.

Up to 4 sprites can be displayed on a single horizontal line, and from the set of displayed sprites on this line, the sprites with higher priority will be displayed, and the overlapping portions of lower priority sprites will not be displayed. Lower sprite numbers are assigned higher priority, with #0 is of highest priority and #31 is of lowest priority.

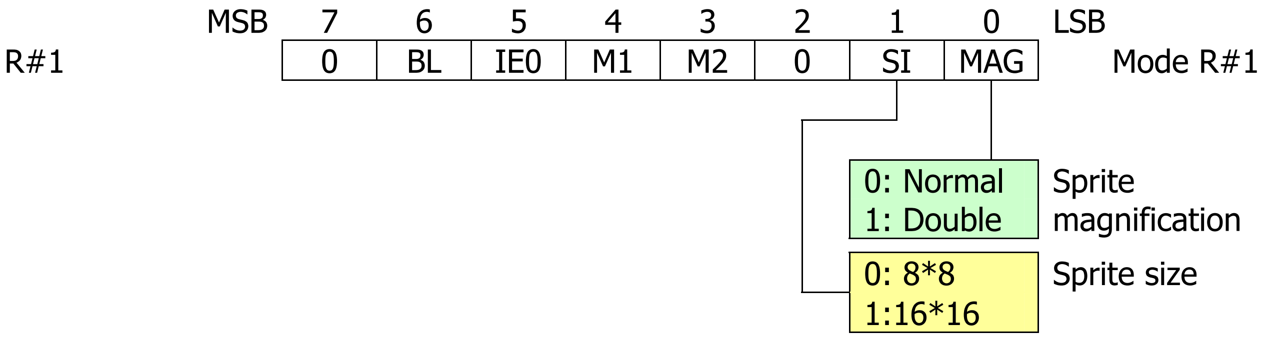


When two sprites collide (their pattern color 1 portions have overlapped), status register's S#0 bit 5 "C" is set to 1.

If there're more than 4 sprites on the single horizontal line, bit 6 of the status register S#0 "5S" is set to 1, and the lower order 5 bits of this status register S#0 will contain the number of the fifth sprite.

5.1.1. Global sprite attributes and tables (SM1)

Register R#1 contains two controls for sprites, allowing magnification and quadruple sprite pattern size.



	MSB	7	6	5	4	3	2	1	0	LSB	
R#1		0	BL	IE0	M1	M2	0	SI	MAG		Mode R#1

MAG: 0: Normal, 1: Double — Sprite magnification

SI: 0: 8*8, 1:16*16 — Sprite size

Sprites are defined by two tables: sprite pattern generator table, which controls the appearance of the dots within the sprite (being on "1" or off "0") and sprite attribute table, which controls positioning, used pattern number and the sprite color.

5.1.2. Sprite attribute table (SM1)

The sprite attribute table is an area in the VRAM that defines display coordinates for all the possible 32 sprites, their colors, pattern numbers used for display and some other flags. Each sprite has four bytes of attribute data, making up 128 bytes (80h) of the memory.

MSB	7	6	5	4	3	2	1	0	LSB
(B) 0	Y-coordinate (0…255)								
1	X-coordinate (0…255)								Attribute area for sprite #0
2	Pattern number (0…255)								
3	EC	-	-	-	Color code				
124	Y-coordinate (0…255)								
125	X-coordinate (0…255)								Attribute area for sprite #31
126	Pattern number (0…255)								
127	EC	-	-	-	Color code				

Y-coordinate defines the vertical position of the sprite. Note that if Y is equal to 208 (D0h), all lower priority sprites will not be displayed. It is important to know that when using vertical offset register R#23 to scroll the visible area of the screen, sprite with the Y-coordinate of 208 will not be displayed and in order to display sprite in this area on the scrolled screen programmer should use Y-coordinate equal to 207 or 209.

Pattern number specifies which pattern from sprite pattern generator table to use to display the sprite bitmap image. If 8*8 dots sprite size is selected, there're 256 patterns available, but if 16*16 dots sprite size is selected, there there're only 64 patterns available. In 16*16 mode two lower-order bits of the pattern number are not used (and they can hold any value).

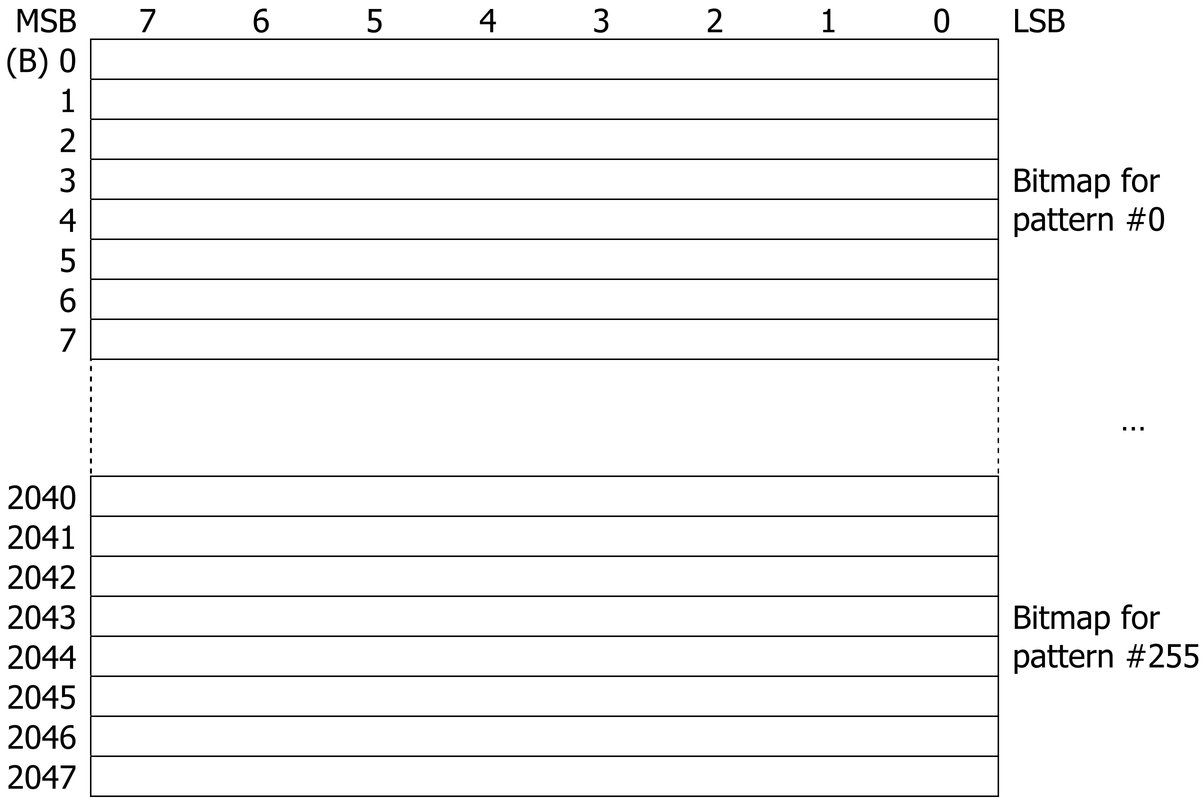
Color code specifies color for pattern color 1 (of the dots which are set to "1" in the sprite bitmap image). The dots which are set to "0" in the sprite bitmap image will appear transparent.

EC (Early clock) is used to offset sprite by 32 dots to the left. This feature is useful when programmer needs to put sprite to the left off the screen.

5.1.3. Sprite pattern generator table (SM1)

Sprite pattern generator table is an area in the VRAM specifying the sprite patterns (appearance). Base address of this table should be set in register R#6.

Eight bytes are used for each pattern, and whole table occupies 2048 bytes (800h) for all the 256 patterns possible. Each pattern is assigned the number, #0 to #255. If sprite size of 8*8 is selected (SI=0) then each displayed sprite has one pattern; if sprite size of 16*16 is selected (SI=1) then each sprite is comprised of four patterns.



5.1.4. Example of the sprite pattern generator table

8*8 sprite representation

MSB LSB

Offset	7	6	5	4	3	2	1	0	
0								■	Pattern #0
1						■	■	■	
2				■	■	■	■	■	
3					■	■	■		
4					■	■		■	
5					■	■		■	
6					■	■	■		
7				■	■	■	■	■	
8			■	■	■	■	■	■	Pattern #1
9		■	■	■	■	■	■	■	
10	■	■	■	■	■	■	■	■	
11	■	■	■	■	■	■	■	■	
12			■	■	■	■			
13				■	■				
14									
15									
16	■								Pattern #2
17	■	■	■						
18	■	■	■	■	■				
19		■	■	■					
20	■		■	■					
21	■		■	■					
22		■	■	■					
23	■	■	■	■	■				
24	■	■	■	■	■	■			Pattern #3
25	■	■	■	■	■	■	■		
26	■	■	■	■	■	■	■	■	
27	■	■	■	■	■	■	■	■	
28			■	■	■	■			
29				■	■				
30									
31									

16*16 sprite representation

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
							■	■							
					■	■	■	■	■	■					
			■	■	■	■	■	■	■	■	■	■			
				■	■	■			■	■	■				
				■	■		■	■		■	■				
				■	■		■	■		■	■				
				■	■	■			■	■	■				
			■	■	■	■	■	■	■	■	■	■			
		■	■	■	■	■	■	■	■	■	■	■	■		
	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
		■	■	■	■					■	■	■	■		
			■	■							■	■			

16*16 sprite size mode pattern layout

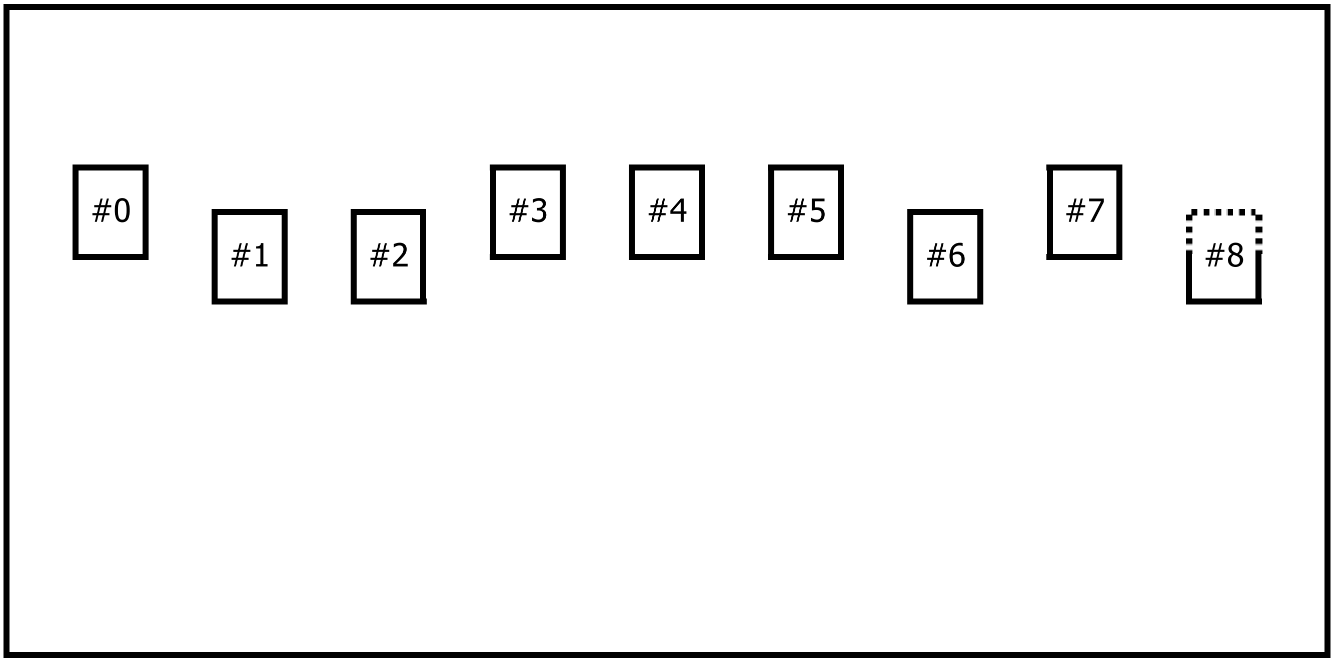
#0	#2
#1	#3

Remember that in 16*16 sprite size mode two least significant bits of pattern number are not used, and setting sprite pattern number 3 will display the same sprite image as setting sprite pattern numbers 0, 1 and 2

5.2. SPRITE MODE 2 (G3 … G7)

While there's a room for 256 sprite patterns in sprite pattern generator table, VDP is only capable of displaying 32 sprites (#0…#31) – limiting factor is sprite attribute table which can hold attributes for 32 *active* sprites only.

Up to 8 sprites can be displayed on a single horizontal line, and from the set of displayed sprites on this line, the sprites with higher priority will be displayed, and the overlapping portions of lower priority sprites will not be displayed. Lower sprite numbers are assigned higher priority, with #0 is of highest priority and #31 is of lowest priority.



When two sprites collide (their solid "1" portions have overlapped), status register's S#0 bit 5 "C" is set to 1, and the coordinates of the collision can be read from status registers S#3 … S#5.

If there're more than 8 sprites on the single horizontal line, bit 6 of the status register S#0 "5S" is set to 1, and the lower order 5 bits of this status register S#0 will contain the number of the ninth sprite.

The colors of the sprite are specified for each horizontal line.

The sprite priority may be cancelled by setting bit "CC" in the attribute table for specific line of the sprite, and if sprites overlap, logical "OR" will be done on the colors of the sprites. Therefore, while in sprite mode 1 sprites may only have two colors (color code 1 and transparent), in sprite mode 2 four colors may be displayed.

5.2.1. Global sprite attributes and tables (SM2)

Register R#1 contains two controls for sprites, allowing magnification and quadruple sprite pattern size. Register R#8 contains one control bit, "SPD", which allows disabling and enabling sprite display.

	MSB 7	6	5	4	3	2	1	0 LSB	
R#1	0	BL	IE0	M1	M2	0	SI	MAG	R#1

MAG: 0: Normal, 1: Double — Sprite magnification

SI: 0: 8*8, 1:16*16 — Sprite size

R#8	MS	LP	TP	CB	VR	0	SPD	BW	R#8

SPD: 0: Enable, 1:Disable — Sprite display control

Sprites are defined by three tables: sprite pattern generator table, which controls the appearance of the dots within the sprite (being on "1" or off "0"), sprite color table, which control colors of the sprite lines and other attributes, and sprite attribute table, which controls positioning and used pattern number.

5.2.2. Sprite attribute table (SM2)

The sprite attribute table is an area in the VRAM that defines display coordinates for all the possible 32 sprites, pattern numbers used for display and some other flags. Each sprite has four bytes of attribute data, making up 128 bytes (80h) of the memory.

MSB	7 6 5 4 3 2 1 0	LSB
(B) 0	Y-coordinate (0…255)	
1	X-coordinate (0…255)	Attribute area for sprite #0
2	Pattern number (0…255)	
3	Reserved	
⋮	⋮	
124	Y-coordinate (0…255)	
125	X-coordinate (0…255)	Attribute area for sprite #31
126	Pattern number (0…255)	
127	Reserved	

Y-coordinate defines the vertical position of the sprite. Note that if Y is equal to 216 (D8h), all lower priority sprites will not be displayed. It is important to know that when using vertical offset register R#23 to scroll the visible area of the screen, sprite with the Y-coordinate of 216 will not be displayed and in order to display sprite in this area on the scrolled screen programmer should use Y=coordinate equal to 215 or 217.

Pattern number specifies which pattern from sprite pattern generator table to use to display the sprite bitmap image. If 8*8 dots sprite size is selected, there're 256 patterns available, but if 16*16 dots sprite size is selected, there there're only 64 patterns available. In 16*16 mode two lower-order bits of the pattern number are not used (and they can hold any value).

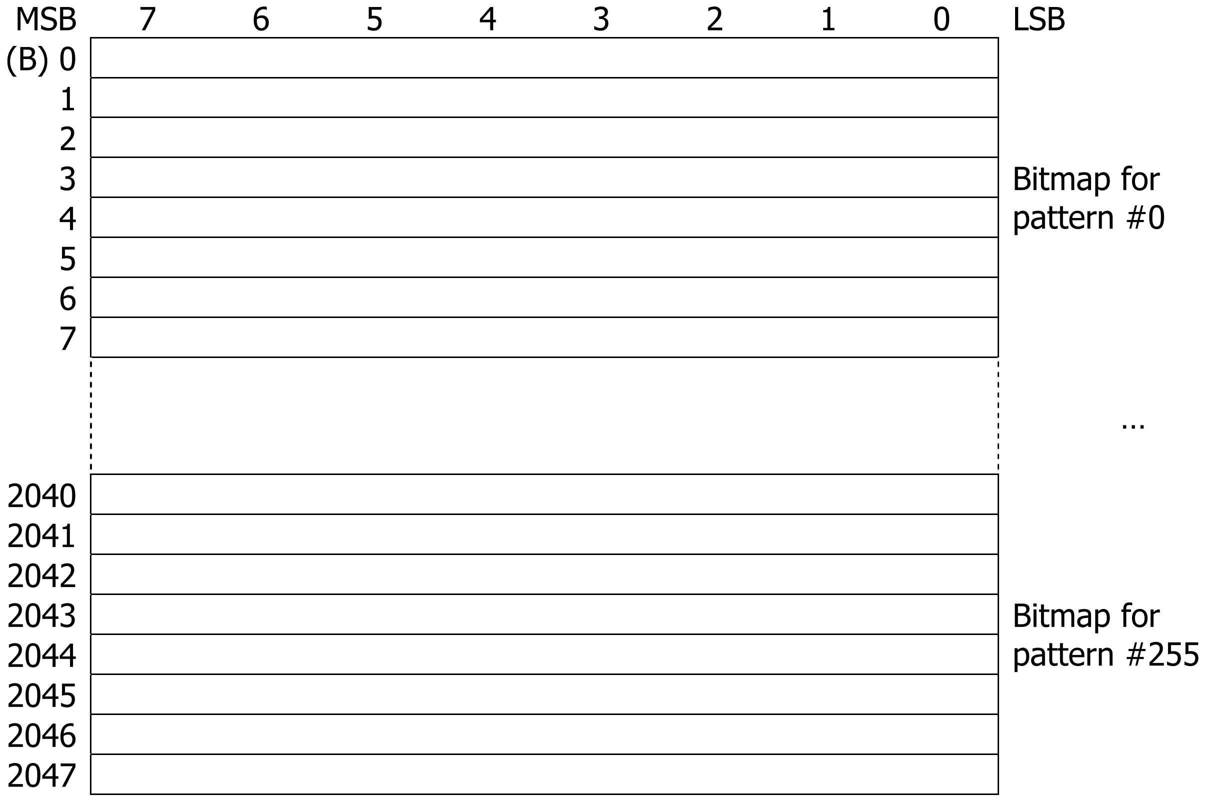
Color code is not used in sprite mode 2, and the respective area in the table is reserved.

	MSB	7	6	5	4	3	2	1	0	LSB
R#5		A14	A13	A12	A11	A10	1	X	X	Sprite attribute table low (SM2)
R#11		0	0	0	0	0	0	A16	A15	Sprite attribute table high (SM2)

5.2.3. Sprite pattern generator table (SM2)

Sprite pattern generator table is an area in the VRAM specifying the sprite patterns (appearance). Base address of this table should be set in register R#6.

Sprite pattern generator table is designed the same way as in SPRITE MODE 1 (SM1); please refer to respective section for the example. Eight bytes are used for each pattern, and whole table occupies 2048 bytes (800h) for all the 256 patterns possible. Each pattern is assigned the number, #0 to #255. If sprite size of 8*8 is selected (SI=0) then each displayed sprite has one pattern; if sprite size of 16*16 is selected (SI=1) then each sprite is comprised of four patterns.

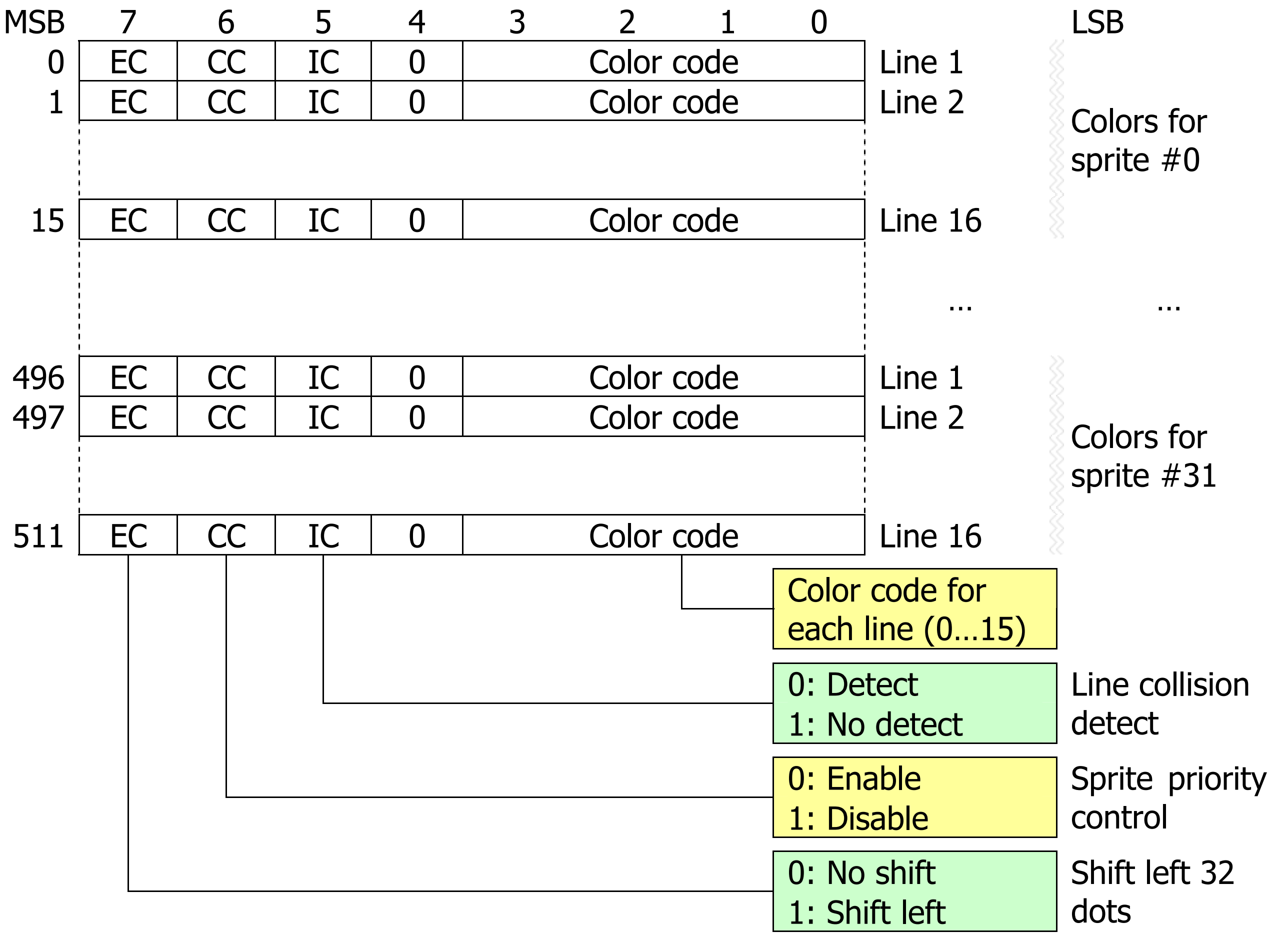


5.2.4. Sprite color table (SM2)

In SPRITE MODE 2 (SM2) the sprite color table specifies sprite colors for each sprite line. Note that sprite color 0 (i.e. the dots having "0" for them in sprite pattern generator table) is transparent unless TP bit of register R#8 is set. Each entry also defines sprite priority, collision detection, and early clock sprite display options.

The base address of the sprite color table is calculated using base address of the sprite attribute table: sprite color table is located "strictly above" the sprite attribute table, having base address of sprite attribute table minus 512 (200h).

Every sprite is assigned 16 color entries in the sprite color table. If 8*8 sprite size mode is selected, only first 8 color bytes are used for every sprite, with another 8 being not used. If 16*16 sprite size mode is selected, then all 16 color bytes are used.



5.2.5. Priority mechanism and multicolored sprite combinations (SM2)

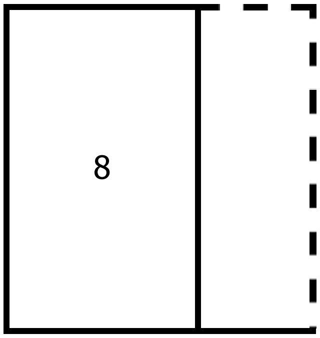
In sprite mode 2 (SM2), if the "CC" bit of the sprite color table entry is set to 1, the sprite priority mechanism is cancelled for the specified line. Collision mechanism does not work, mixing color codes of the lines of affected sprites using "OR" logical operation.

This logical color mixing will work for sprite N if its affected line has CC bit set, and this line will collide with the line of another sprite which has CC bit reset to "0" and has lower number (N-1, N-2, …, 0 – i.e. higher priority).

Moreover, the line of the sprite having CC bit set to "1" will not be displayed at all if there will be no lower number sprite with the line with CC bit set to "0" at the same screen line.

Please see below for the example for the sprite combination displayed in seven colors.

Sprite #N

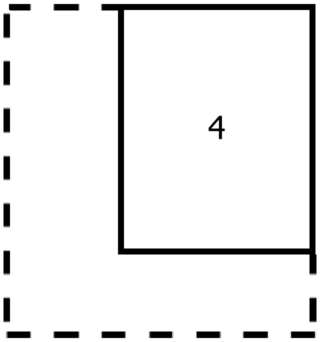


CC = all set to 0

MSB			LSB
1	0	0	0

Color code 8

Sprite #N+1

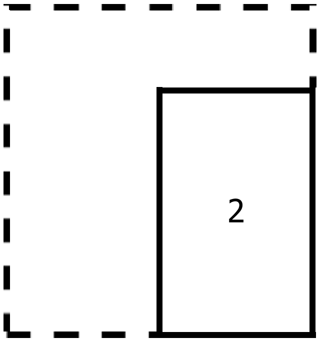


CC = all set to 1

MSB			LSB
0	1	0	0

Color code 4

Sprite #N+2

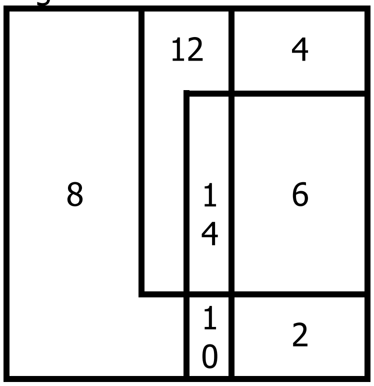


CC = all set to 1

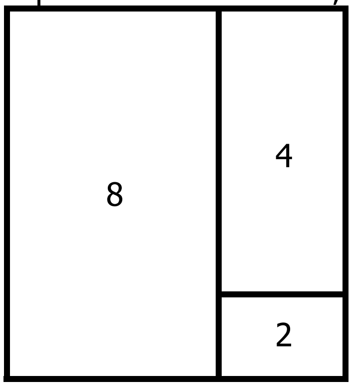
MSB			LSB
0	0	1	0

Color code 2

Resulting images in SM1 and SM2 modes (all sprites are placed at the same X, Y).



Sprite mode 2: CC flag is used, no collision detection



Sprite mode 1: collision detection turned on

5.2.6. Sprite collision

If CC bit is set to 0 and two sprites overlap with their color code 1 (the dots of two sprites defined by "1" in their bitmap), a sprite collision event occur, and bit 5 ("C") of status register S#0 will be set to "1". This bit will be reset when S#0 is read.

When collision occurs and neither mouse flag (MO) nor the light pen flag (LP) of the register R#8 are set, status registers S#3 to S#6 will contain coordinates of the collision.

When status register S#5 is read, all the contents of registers S#3 to S#6 are reset. This means that program should read status registers S#3, S#4 and S#6 before reading status register S#5. Collision coordinates can be calculated using the following formulas:

X (S#4, S#3),	Y (S#6, S#5)
XC=X-12	YC=Y-8

5.3. Special rules for sprite color settings

In all graphic modes except GRAPHIC7 mode, sprite display color is determined by the values in palette registers.

In GRAPHIC7 mode sprite colors are fixed and are not related to contents of palette registers. Table below show these effective values.

Color code				Green			Red			Blue		
C3	**C2**	**C1**	**C0**	**G2**	**G1**	**G0**	**R2**	**R1**	**R0**	**B2**	**B1**	**B0**
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0	1	1	0	1	0
0	1	0	0	0	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0	0	1	0
0	1	1	0	0	1	1	0	1	1	0	0	0
0	1	1	1	0	1	1	0	1	1	0	1	0
1	0	0	0	1	0	0	1	1	1	0	1	0
1	0	0	1	0	0	0	0	0	0	1	1	1
1	0	1	0	0	0	0	1	1	1	0	0	0
1	0	1	1	0	0	0	1	1	1	1	1	1
1	1	0	0	1	1	1	0	0	0	0	0	0
1	1	0	1	1	1	1	0	0	0	1	1	1
1	1	1	0	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1

The state of bit 5 “TP” of register R#8 affects how sprites are displayed, and affects color code 0.

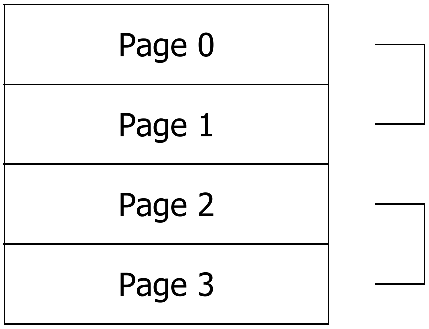
TP bit value	Behavior
TP=0	Color code 0 will be treated as transparent and invisible. All the dots corresponding to “0” bit value in sprite bitmap will be transparent. These dots are considered as non-existent and will not cause collision event
TP=1	Color code 0 will be the code corresponding to color #0 in the palette register. Note that in GRAPHIC7 mode, value of color #0 is predefined and equal to G=0, R=0 and B=0. If sprites will overlap with the dots corresponding to “0” bit value in their sprite bitmap, collision event will be generated

6. SPECIAL FUNCTIONS

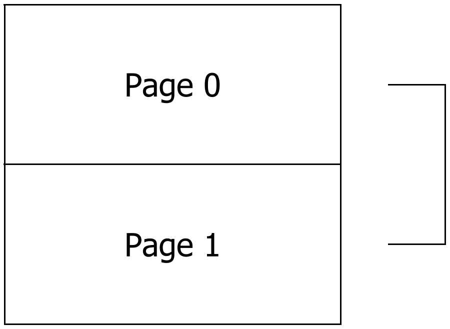
6.1. Alternate display of two graphic screen pages

Two graphics screen pages may be alternatively displayed in G4 to G7 modes automatically. In G4 and G5 modes, the following pages can be displayed alternatively: page 0 and page 1, and page 2 and page3. In G6 and G7 modes, pages 0 and 1 will be alternatively displayed. Please refer to chapter 6.2 for description of page concept.

G4 and G5 modes



G6 and G7 modes



Display time period of between 166 ms and 2053 ms may be specified for each page. See section 3.2.6 for information about available time periods.

- Specify the odd page address in the pattern layout table register R#2
- Specify ON (even page) and OFF (odd page display) intervals in register R#13

6.2. Displaying two graphics screens at 60Hz

Bit 2 “EO” of register R#9 can be used for displaying two graphics screens alternately at 60Hz. Set the odd page pattern layout table in register R#2. Then set bit 2 of register R#9 to 1.

6.3. Interlaced display

This function displays the first and the second fields on the same page: set the bit 3 “IL” of the register R#9, to 1. To display even page in the first field and odd page in second field, set odd page in pattern layout table register R#2, and set both bits 2 “EO” and 3 “IL” in register R#9, to 1.