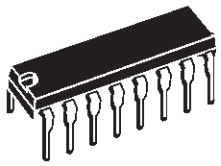
STV9936P/STV9936S

120MHz OSD for monitors
4 true independant windows concept

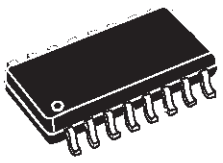
PRELIMINARY DATA

FEATURES

- **Horizontal frequency up to 150 kHz**
- **On chip pixel clock generator from 7.68 MHz to 120 MHz, without any quartz**
- **16-pin SO or DIP packages**
- **Programmable horizontal resolutions from 384 to 1524 dots per scan line**
- **4 independant windows all with characters display**
- **Windows overlapping with automatic control of display priorities and scrolling menu effect**
- **Independant and programmable display, positions and sizes for each window**
- **Transparent background or 8 background colors programmable for each window**
- **Window size up to 16 rows of 32 characters,**
- **Each window has its own bordering or shadowing with programmable color, height and width**
- **Clear bits to erase windows separately**
- **Common programmable positioning to control centered display**
- **256 standard and 16 multicolor characters or graphic fonts in ROM. Character fonts can be customized with a mask programmable ROM**
- **Characters :**
  - Commun character height and row space. Character height from 18 to 127 lines and space lines from 0 to 62 split above and below character rows,
  - 12 x 18 dots matrix per character,
  - display of up to 704 characters,
  - programmable shadowing on characters for each window, separately
  - 32 possibilities of background, foreground, blinking character colors programmable for each character: 8 possibilities per window,
  - 8 colors selectable for standard characters,
  - 8 colors + transparent selectable for background.
- **On the screen**
  - fade-in, fade-out effect.
  - possibility of full screen display with a selectable color.
- **I2C interface for microcontroller with slave address BA(H), in read and write modes.**



PDIP16 (Plastic Dual In line Package)
ORDER CODE: STV9936P



SO16narrow(Plastic Micropackage)
ORDER CODE: STV9936S
Weight 0.20 g

Table of contents

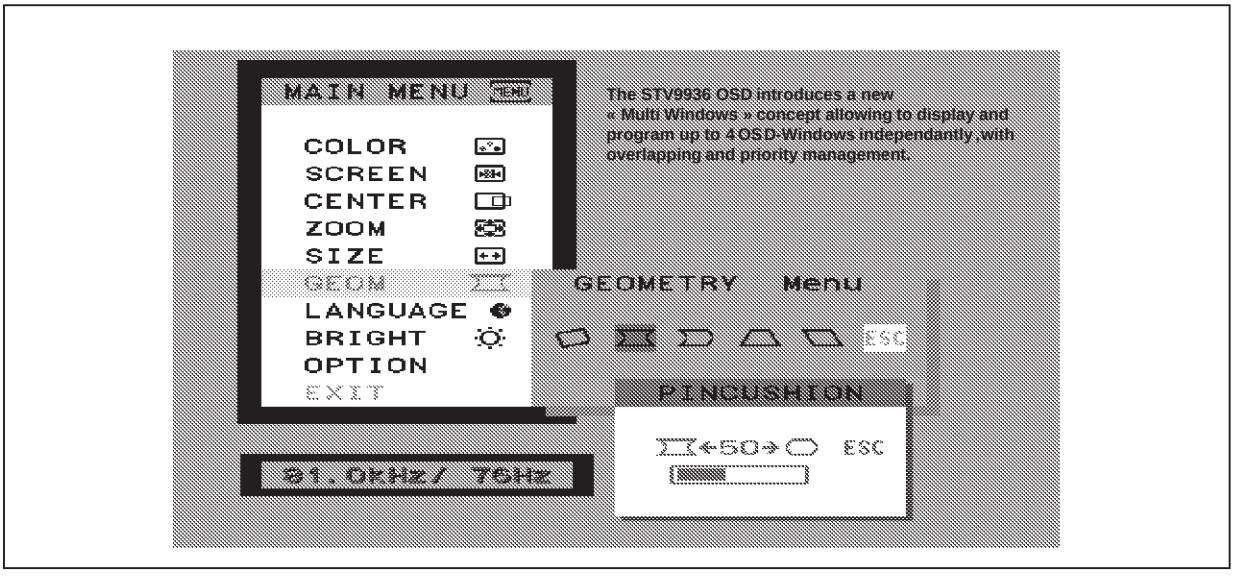
1 Introduction - general description

The STV9936 is a new On Screen Display device with multiple menus display for monitor applications.

In addition to the usual general features of an OSD, the STV9936 main characteristics are listed here below:

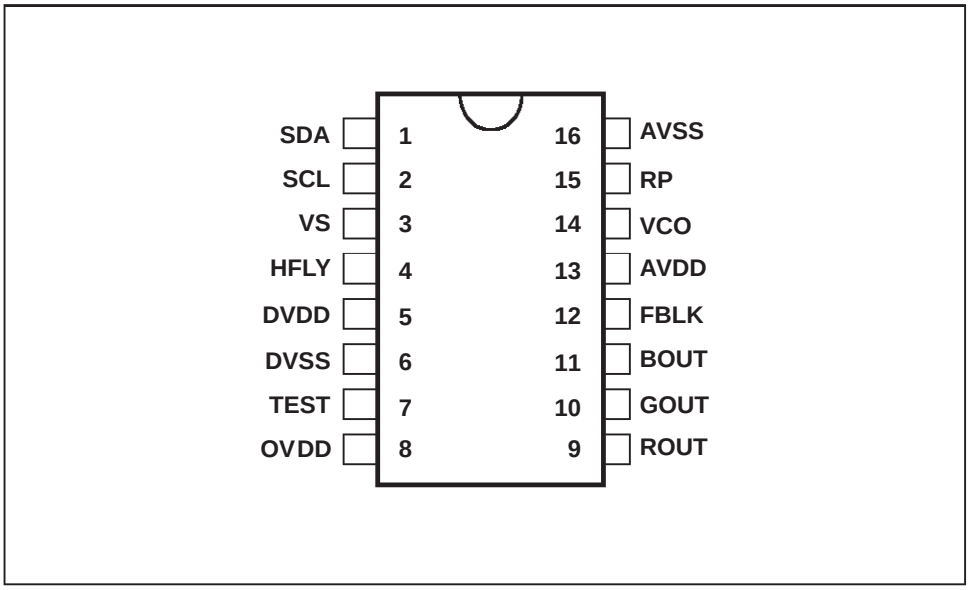
- display of up to 4 menus simultaneously, anywhere on the screen. The 4 independant windows, all displaying characters, can be overlapped. Display priorities is automatically controlled. Windows sizes and positions are independently programmable. The display as scrolling menu is easily programmable
- programming of the general OSD and of the 4 windows is controlled by an I2C bus in read and write modes, to suit the various CRT displays.
- associated with an easily programmable character height, the internal PLL, without quartz, generates the programmable pixel clock defining the character width. Thus, making the device suitable for multi-sync applications.
- a maximum of 704 characters, defined in the mask-programmable ROM, are distributed among the 4 windows and displayed simultaneously.

Figure 1: Multi windows concept with character display



2 Pin description

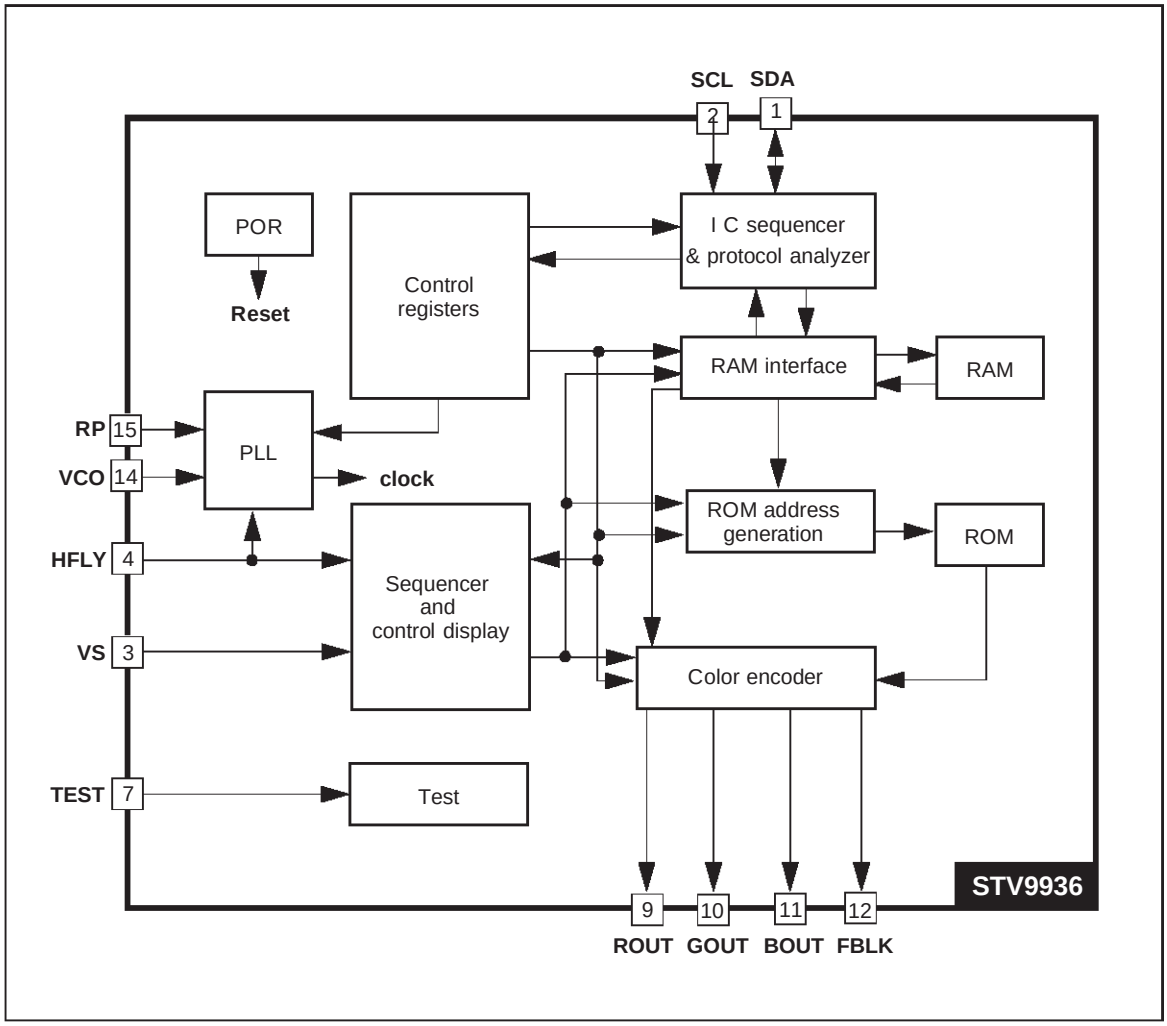
Figure 2: Pin connections



N°	Pin Name	Direction	Digital/ Analog	Function
1	**SDA**	I/O	Digital	Serial data of I2C bus
2	**SCL**	Input	Digital	Serial clock of I2C bus
3	**VS**	Input	Digital	Vertical synchro input
4	**HFLY**	Input	Digital	Horizontal synchro input
5	**DVDD**	-	supply	Digital power supply
6	**DVSS**	-	supply	Digital ground
7	**TEST**	Input	Digital	Remains at 0 (for test purpose only)
8	**OVDD**	-	supply	Digital power supply
9	**ROUT**	Output	Digital	Red color output
10	**GOUT**	Output	Digital	Green color output
11	**BOUT**	Output	Digital	Blue color output
12	**FBLK**	Output	Digital	Fast blanking output
13	**AVDD**	-	supply	Analog power supply
14	**VCO**	I/O	Analog	for VCO
15	**RP**	I/O	Analog	for VCO
16	**AVSS**	-	supply	Analog ground

3 Block diagram

Figure 3: STV9936 block diagram



4 Electrical and Timing Characteristics

4.1 Absolute maximum ratings

Symbol	Parameter	Value	Unit
AV_{DD}, DV_{DD}, OV_{DD}	DC Supply Voltage	-0.5, +4.0	V
V_{IN}	Input Voltage for scl, sda, vs, hfly	-0.5, 5.5	V
	Input Voltage for test	VDD + 0.5	V
T_{oper}	Ambient Operating Temperature	0, +70	^oC
T_{stg}	Storage Temperature	-40, +125	^oC

4.2 Operating conditions

Symbol	Parameter	Min.	Typ.	Max.	Unit
VDD	DC Supply Voltage AV_{DD}, DV_{DD}, $OV_{DD.}$	3.0	3.3	3.6	V
T_{oper}	Ambient Operating Temperature	0	25	70	^oC

4.3 Electrical and timing characteristics

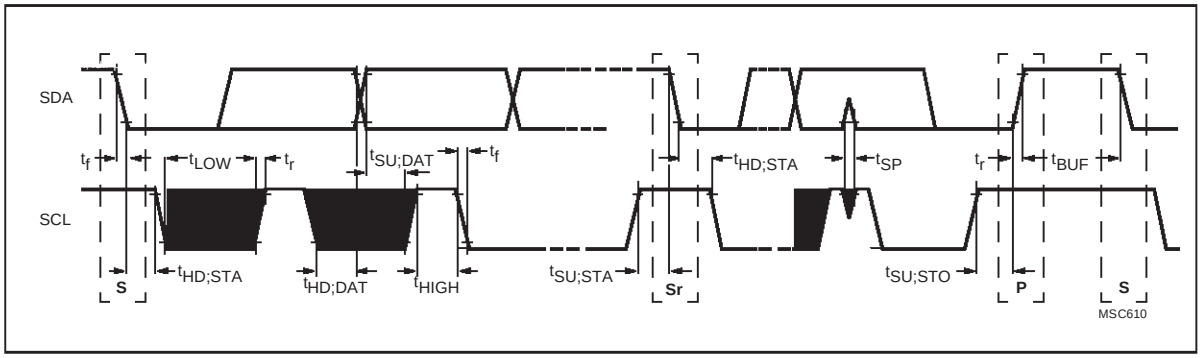
(V_{DD} = 3.3V, V_{SS} = 0V, T_A = 0 to 70^o, unless otherwise specified)

Symbol	Parameter	Min.	Typ.	Max.	Unit
ELECTRICAL CHARACTERISTICS					
IDD	Analog and Digital Supply Current AI_{DD}+ DI_{DD} + OI_{DD}	-	-	30	mA
V_{IL}	Input Low Voltage (scl, sda, vs, hfly,test)	-	-	0.8	V
V_{IH}	Input High Voltage (scl, sda, vs, hfly) test input is connected to ground	2.0	-	5.0	V
V_{OL}	Rout, Gout, Bout, Fblk Output Low Voltage (I_{OL} = 3 mA)	-	-	0.4	V
	Sda Open Drain Output Low Voltage (I_{OL} = 4 mA)	-	-	0.4	V
V_{OH}	Rout, Gout, Bout, Fblk Output High Voltage (I_{OH} = 3 mA)	2.4	-	-	V
	Sda Open Drain Output High Voltage, pulled up by external 3V to 5V power supply	-	-	5.0	V
TIMING CHARACTERISTICS					
Freq(Hline)	Horizontal Synchro Input Range	-	-	150	kHz
Trise	Rout, Gout, Bout, Fblk Output rise time (Cload = 15pF)		2		ns
Tfall	Rout, Gout, Bout, Fblk Output fall time (Cload = 15pF)		2		ns

Table 1: Characteristics of the SDA an SCL bus lines for F/S-mode I^2C-bus devices

Symbol	Parameter	Standard mode		Fast mode		Unit
		Min.	Max.	Min.	Max.	
I^2C INTERFACE: SDA AND SCL						
t_{SP}	Pulse width of spikes which must be suppressed by the input filter	n/a	n/a	0	50	ns
f_{SCL}	SCL clock frequency	0	100	0	400	kHz
$t_{HD;STA}$	Hold time (repeated) START Condition. After this period, the first clock pulse is generated	4.0	-	0.6	-	µs
t_{LOW}	LOW period of the SCL clock	4.7	-	1.3	-	µs
t_{HIGH}	HIGH period of the SCL clock	4.0	-	0.6	-	µs
$t_{SU;}\ t_{STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	µs
$t_{HD;DAT}$	Data hold time	0	3.45	0	0.9	µs
$t_{SU;DAT}$	Data set-up time	250	-	100	-	ns
t_r	Rise time of both SCL and SDA signals	-	1000	$20 + 0.1C_b$	300	ns
t_f	Fall time of both SCL and SDA signals	-	300	$20 + 0.1C_b$	300	ns
$t_{SU;}\ t_{STO}$	Set-up time for STOP condition	4.0	-	0.6	-	µs
t_{BUF}	Bus free time between a STOP and a START condition	4.7	-	1.3	-	µs
C_b	Capacitive load for each bus line	-	400	-	400	pF

Figure 4: Definition of timing for F/S-modes



5 Registers addressing

All control registers are located in window 0, row 0.

Three formats are available: a, b and c, as described in the I2C protocol (see *Section 9 on page 27*).

All addresses (FAC, and FWR bytes) are based on formats a or b, and written in hexadecimal.

6 Windows specification

Four different windows with characters display can be displayed simultaneously on screen. It is possible to have overlapping windows with automatic control of display priorities: downscale priorities from window4 to window1.

Window 1 is well adapted for the OSD general menu.

The 4 windows, each with characters display, are positioned anywhere on the screen.

The following characteristics are defined for each window:

- → window enable
- → position
- → size, adjustable with memory allocation
- → background color
- → bordering or shadowing with programmable color, height and width.

6.1 Enable display

Each window has an enable: **ENWI**

ENW1 =1: window 1 is displayed, **ENW1** = 0, not displayed

ENW2 =1: window 2 is displayed, **ENW2** = 0, not displayed

ENW3 =1: window 3 is displayed, **ENW3** = 0, not displayed

ENW4 =1: window 4 is displayed, **ENW4** = 0, not displayed

(default value = 0000)

Table 2: Enable display

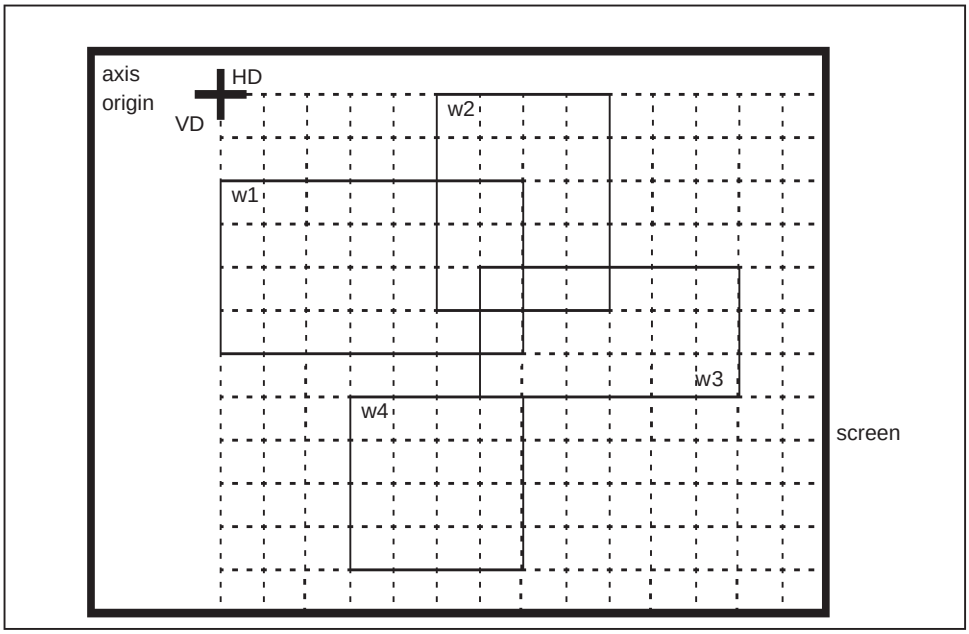
FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	07	-	-	-	-	ENW4	ENW3	ENW2	ENW1

6.2 Origin position for the 4 windows

The 4 windows are arranged in a vertical frame which origin coordinates are HD, VD (top left). When changing HD, VD values, the 4 windows within the frame position are shifted by the same value. The origin (HD, VD) can be anywhere on the screen. Adjusting the origin allows the global repositioning of the OSD.

The advantages of this system are an easier programming, the possibility to adapt the position of all windows at one time without changing each window' s relative position, the possibility for the user to program all 4 windows position.

Figure 5: Example of windows display



The axis origin is defined by (HD,VD)

Table 3: Origin of windows on horizontal axis: Horizontal Delay

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	04	-	HD(6)	HD(5)	HD(4)	HD(3)	HD(2)	HD(1)	HD(0)

The horizontal delay is:
HorDelay = HD[6:0] * 6 pixels + 50 pixels - phase error detection pulse width

The range of horizontal delay is [50 - 812] pixels by step of 6 pixels.

(default value = 0)

Table 4: Origin of windows on vertical axis: Vertical delay

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	05	VD(7)	VD(6)	VD(5)	VD(4)	VD(3)	VD(2)	VD(1)	VD(0)

The vertical delay is: VertDelay = (VD[7:0] * 4 + 2) scan lines.

The range of vertical delay is [2 - 1022] scan lines by step of 4 scan lines.

(default value = 0)

6.3 Windows position in the frame

All values are referenced to the original points (HD, VD)

Window horizontal delay

HDW1- HDW2- HDW3- HDW4 = HDWi, i from 1 to 4 = 4 registers of 7 bits

Table 5: Windows origin on horizontal axis: windows horizontal delay

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	0C, 11, 16, 1B	-	HDWI(6)	HDWI(5)	HDWI(4)	HDWI(3)	HDWI(2)	HDWI(1)	HDWI(0)

The horizontal delay of a window (i) is : HorDelayWini = **HDWI**[6:0] * 12 pixels.

The range of horizontal delay is 0 to 1524 pixels by step of 12 pixels.

The total horizontal delay of a window is:

TotHorDelay = HorDelay + **HDWI**[6:0] * 12 pixels.

TotHorDelay = HD[6:0] * 6 pixels + **HDWI**[6:0] *12 pixels + 50 pixels .

Window vertical delay

VDW1- VDW2- VDW3- VDW4 = VDWi, i from 1 to 4 = 4 registers of 6 bits

Table 6: Windows origin on vertical axis : windows vertical delay

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	0D, 12 17, 1C	-	-	VDWI(5)	VDWI(4)	VDWI(3)	VDWI(2)	VDWI(1)	VDWI(0)

The vertical delay is : **VDWI**[5:0] * (Character_Height + 2 * Space_Lines).

The range of vertical delay is 0 to 63 rows of characters by step of 1 characters row.

The height of character row is: Row_Height = Character_Height + 2 * Space_Lines

(See *Figure 5 on page 11*)

Window total vertical delay

TotVertDelay = VertDelay + **VDWI**[5:0] * Row_Height.

TotVertDelay = (VD[7:0] * 4 + 2) scan lines + **VDWI**[5:0] * Row_Height.

6.4 Window size : number of character rows and character columns

Window horizontal size

HSW1- HSW2- HSW3- HSW4 = HSWi, i from 1 to 4 = 4 registers of 5 bits

Table 7: Window horizontal size

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	0E, 13, 18, 1D	-	-	-	HSWI(4)	HSWI(3)	HSWI(2)	HSWI(1)	HSWI(0)

The horizontal size of a window 'i' is : HorSizeWini = (HSWi[4:0] +1) characters.

The range of horizontal size is 1 to 32 characters.

1 character is 12 pixels long.

Window vertical size

VSW1- VSW2- VSW3- VSW4 = VSWi, i from 1 to 4 = 4 registers of 4 bits

Table 8: Window vertical size

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	0F, 14 19, 1E	-	-	-	-	VSWI(3)	VSWI(2)	VSWI(1)	VSWI(0)

The vertical size is VertSizeWini = (VSWi[3:0] + 1) * Row_Height

The range of vertical size is from 1 to 16 characters rows by step of 1 characters row.

Row_Height = Character_Height + 2 * Space_Lines.

According to the example shown in *Figure 8*,

Table 9: Example of origin and size of windows

window i	HD	VD	HSWi	VSWi
window 1	0	2	7	4
window 2	5	0	4	5
window 3	6	4	6	3
window 4	3	7	4	4

6.5 Windows background color

RGBW1- RGBW2- RGBW3- RGBW4 = RGBWI, i from 1 to 4 = 4 registers of 4 bits

Table 10: Background color of each window

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	10, 15 1A, 1F	-	-	-	-	TI	RWI	GWI	BWI

RWI- GWI-BWI: Red- Green- Blue of Window "i"

TI = 1 : transparent background, active video displayed as background

TI = 0 : **RGBWI** color background

(default value = 0111: white windows)

6.6 Windows bordering / shadowing

6.6.1 Enable bordering or shadowing

ENBS1- ENBS2- ENBS3- ENBS4 = ENBSI, i from 1 to 4 = 4 bits

Bordering or Shadowing choice:

BSW1- BSW2- BSW3- BSW4 = BSWI, i from 1 to 4 = 4 bits

ENBSI	= 0	no bordering, no shadowing (default value = 0)
	= 1	bordering or shadowing selected.
BSWI	= 0	bordering is selected (default value = 0)
	=1	shadowing is selected.

Table 11: Enable bordering or shadowing

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	07	ENBS4	ENBS3	ENBS2	ENBS1				
80	10, 15 1A, 1F	-	-	-	BSWI	-	-	-	-

6.6.2 Border or shadow color

It is programmable separately for each window: 4 registers of 3 bits

Table 12: Border or shadow color

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	10, 15 1A, 1F	WSRI	WSGI	WSBI	-	-	-	-	-

WSRGBI: WSRI-WSGI- WSBI: color of bordering or shadowing of window 'i' ; i from 1 to 4.

(default value = 000: black)

6.6.3 Bordering or shadowing sizes

- Bordering/shadowing width for each window : **BSWWI**: 4 registers of 3 bits

(default value = 000).

The width is from 0 to 14 pixels by step of 2 pixels.

width(i) = **BSWWI[2:0]** * 2 pixels.

- Bordering/shadowing height for each Window : **BSHWI** : 4 registers of 4 bits

(default value = 0000)

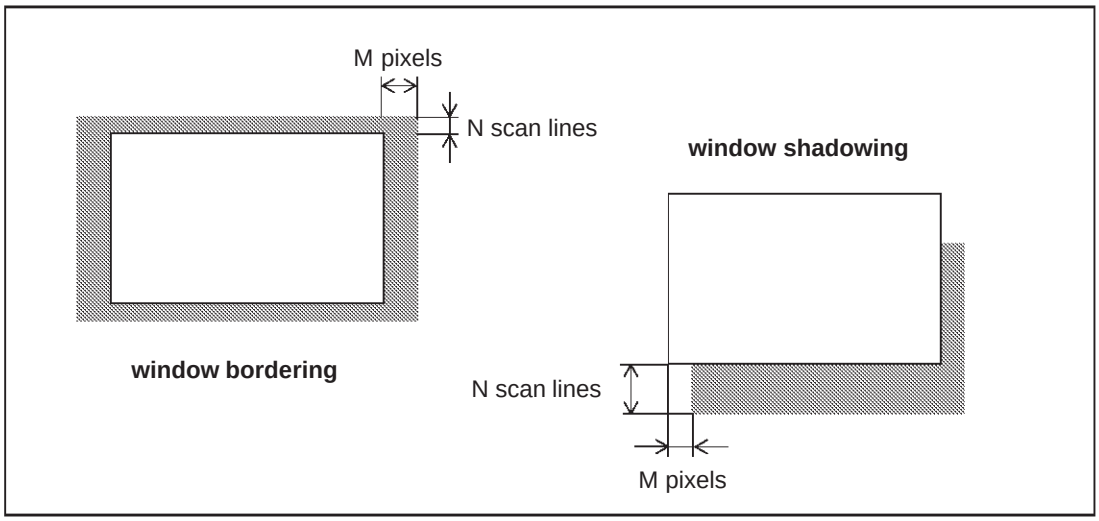
The height is from 0 to 30 lines by step of 2 scan lines.

height(i) = **BSHWI[3:0]** * 2 scan lines.

Table 13: Bordering or shadowing size

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	0E, 13 18, 1D	BSWWI(2)	BSWWI(1)	BSWWI(0)	-	-	-	-	
80	0F, 14 19, 1E	BSHWI(3)	BSHWI(2)	BSHWI(1)	BSHWI(0)	-	-	-	-

Figure 6: Illustration of window bordering and shadowing



6.7 Windows priority management

The priority of windows display is the following: window 4, 3, 2, 1.

Considering different window background colors:

window1 : cyan,

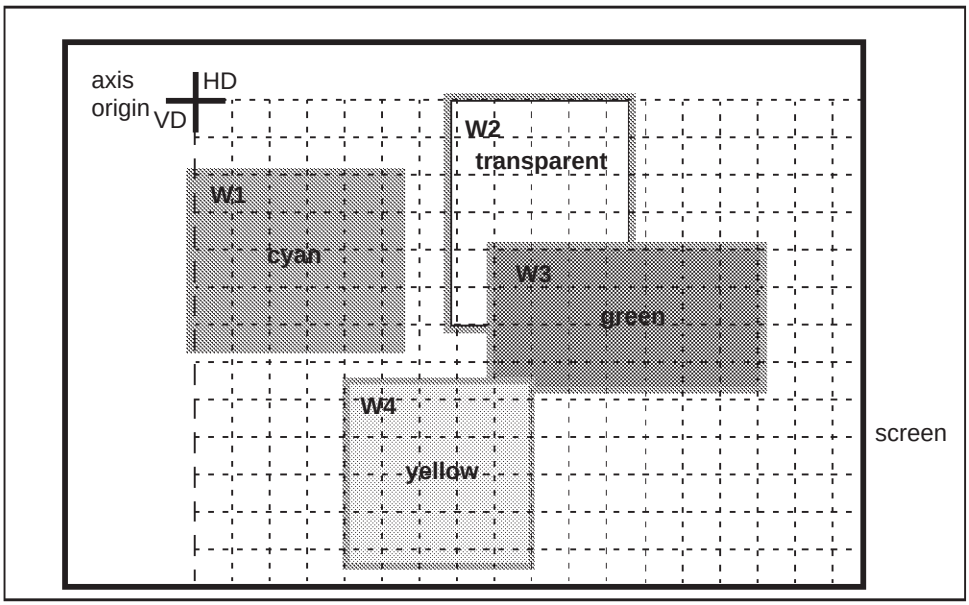
window2 : transparent,

window3: green,

window4: yellow

and also different windows bordering, the previous example from *Figure 5* becomes :

Figure 7: Example of windows display with priority



7 Characters specification

7.1 Generalities

There are:

- 256 monochrome characters and 16 multicolor characters in a ROM
- 32 to 127 characters per line
- character height varies between 18 and 127 scan lines
- 0 to 62 scan space lines between character rows, with the same number of lines above and below the rows of characters.

With the possibility to select:

- blinking for each character.
- characters shadowing in each window
- background and foreground character colors: for each character, among a Color-shop of 8 Color-boxes per window. There is a Color-shop for each window. The Color-boxes define the background colors and the foreground characters colors and blinking.

7.2 Horizontal resolution

Table 14: Horizontal resolution

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	01	-	HR(6)	HR(5)	HR(4)	HR(3)	HR(2)	HR(1)	HR(0)

HR[6:0] defines the number of pixels per line expressed in characters unit (range of 32 to 127 characters per line).

If **HR[6:0]** < 32 then HR = 32. (default value = 32 dec)

A character being 12 pixels long, the number of pixels per line varies from 384 to 1524.

Meanwhile, the maximun pixel frequency must be respected (Fpixel max = 120MHz).

7.3 Character height

Table 15: Character height

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	02	-	CH(6)	CH(5)	CH(4)	CH(3)	CH(2)	CH(1)	CH(0)

The character height varies from 18 to 127 lines (7 * 18 +1 lines). The **CH** register is on 7 bits: **CH[6:0] is the number of lines used to display the characters**. It is very easy for the user to program the character height.

The character in ROM is coded with 18 lines (default value = 18 dec).

When CH < 18, the character height is 18.

When CH = N * 18, all ROM lines are repeated N times, with N in the range of 1 to 7.

When N*18 < CH < (N+1)*18 , some ROM lines are repeated (N+1) times as shown in *Table 16*.

Table 16 shows which ROM line numbers, from 0 to 17, are repeated depending on CH[6:0] value.

Table 16: Repeated ROM lines: ( ) = no repeated lines, (r) repeated ROM lines. a = number of repeated lines

CH value	a	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18, 36, 54,72,90,108,126	0																		
19, 37, 55, 73, 91, 109,127	1									r									
20, 38, 56, 74, 92 ,110,	2						r							r					
21, 39, 57, 75, 93, 111	3					r				r					r				
22, 40, 58, 76, 94, 112	4				r				r			r				r			
23, 41, 59, 77, 95, 113	5			r			r			r				r			r		
24, 42, 60, 78, 96, 114	6		r			r			r			r			r			r	
25, 43, 61, 79, 97, 115	7	r			r			r		r			r			r		r	
26,44,62, 80, 98, 116	8		r		r		r		r			r		r		r		r	
27,45,63, 81, 99, 117	9		r		r		r		r		r		r		r		r		r
28,46,64, 82, 100, 118	10	r		r		r		r		r	r		r		r		r		r
29,47,65, 83, 101, 119	11		r	r		r	r		r		r	r		r	r		r	r	
30,48,66, 84, 102, 120	12	r		r	r		r	r		r	r		r	r		r	r		r
31,49,67, 85, 103, 121	13	r	r		r	r		r	r		r	r	r		r	r		r	r
32,50,68, 86, 104, 122	14	r	r	r		r	r	r		r	r		r	r	r		r	r	r
33,51,69, 87, 105, 123	15	r	r	r	r		r	r	r		r	r	r	r		r	r	r	r
34,52,70, 88, 106, 124	16		r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	
35,53,71, 89, 107, 125	17		r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

7.4 Row to row spacing: space lines

The row spacing indicated in the **RSPA** register (5 bits) is the number of scan lines above and below the characters. The row height is defined as follows:

Row_Height = Character_Height + 2 * Space_Lines.

The color of the spacing lines is the color of the associated character background.

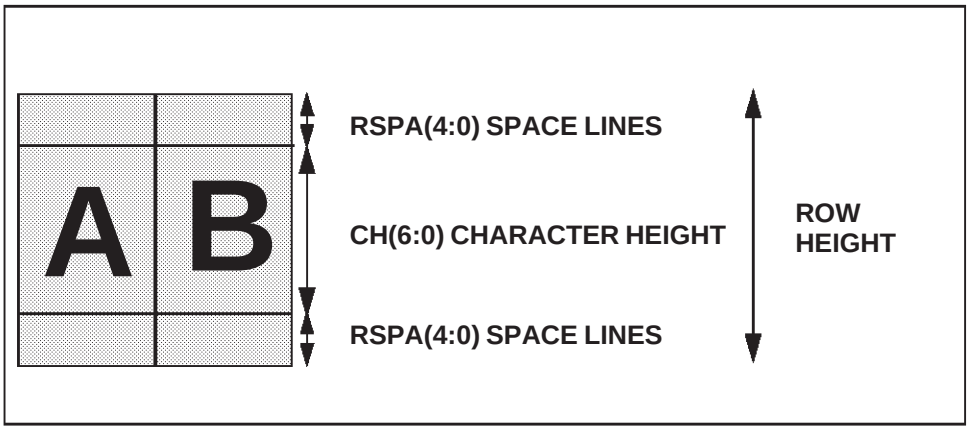
The number of spacing lines varies from 0 to 31 scan lines. Accordingly, the number of lines between the characters varies from 0 to 62 scan lines by step of 2.

(default value = 00000)

Table 17: RSPA: Row to Row Spacing

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	03	-	-	-	RSPA(4)	RSPA(3)	RSPA(2)	RSPA(1)	RSPA(0)

Figure 8: Row height definition



7.5 Color of characters, background and blinking

We define a color-shop of 8 color boxes for each window.

The user programs each color box to select the characters colors and blinking.

There are 4 color-shops, 1 per window, offering the user 32 possibilities of character coloring.

As the color-boxes are in RAM, the user must write into the color-box prior to using it.

The color-boxes are in window 0, row 2.

Table 18: One color box

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
82	from 00 to 1F	BC	BR	BG	BB	BLINK	FR	FG	FB

7.5.1 FR-FG-FB : character color

Table 19: character color

FR	FG	FB	color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

7.5.2 BLINK

BLINK =1 : blink active (character blinks),

BLINK = 0 no blink

7.5.3 BC and BR-BG-BB : background color of the character.

The character background color attribute has higher priority than the window background color.

BC: background color for the character,

BC = 0 : the character background color is the color of the window.

BC = 1 : the character background color is defined with (**BB,BG,BR**).

Table 20: Background color of the character

BC	BR	BG	BB	color
1	0	0	0	Black
1	0	0	1	Blue
1	0	1	0	Green
1	0	1	1	Cyan
1	1	0	0	Red
1	1	0	1	Magent
1	1	1	0	Yellow
1	1	1	1	White
0	x	x	x	window color (see *Table 21*)

Table 21: Background color priority

BC	TI	Background color
1	X	BR-BG-BB = the background color of the character
0	0	RGBWi = window background color
0	1	Transparent background (Video active)

7.6 Multicolor characters

16 multicolor characters are selectable by 4 bits : **MCOLOR[3:0]**, one bit per window :

MCOLOR(3) for window 4

MCOLOR(2) for window 3

MCOLOR(1) for window 2

MCOLOR(0) for window 1

The selection is as follows:

MCOLOR(I) = 0 : the 256 monochrome ROM characters from $00 to $FF.

MCOLOR(I) =1 : the 240 monochrome ROM characters from $00 to $EF and the 16 multicolor characters from $F0 to $FF.

RGB = “000” corresponds to the black color.

No shadowing on multicolor character.

No background on multicolor characters.

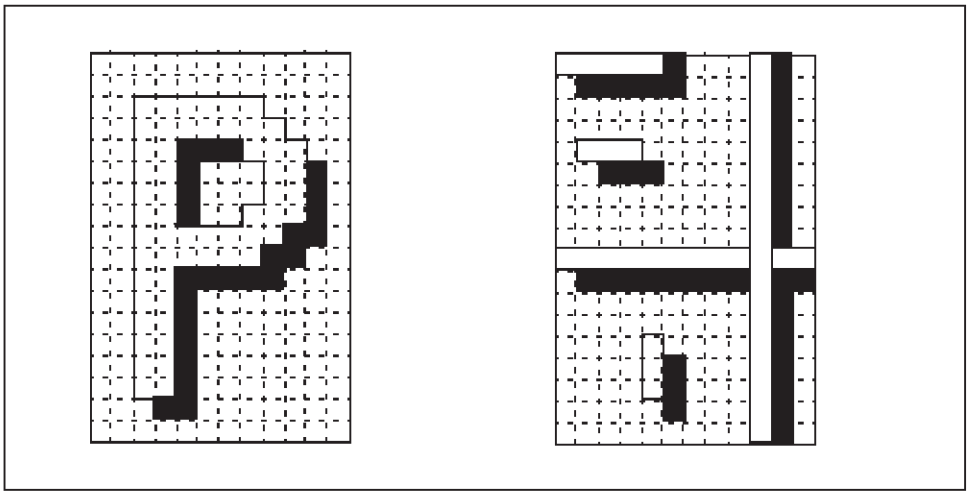
Blinking on multicolor characters: background color coded in the associated Color box.

Table 22: Multicolor character

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	08	MCOLOR4	MCOLOR3	MCOLOR2	MCOLOR1	-	-	-	-

7.7 Character shadowing

Figure 9: Character shadowing



The shadowing enable is programmable for each windows, the shadowing color is black.

One bit per window : **CSHA[3:0]**

CSHA(3) for window 4

CSHA(2) for window 3

CSHA(1) for window 2

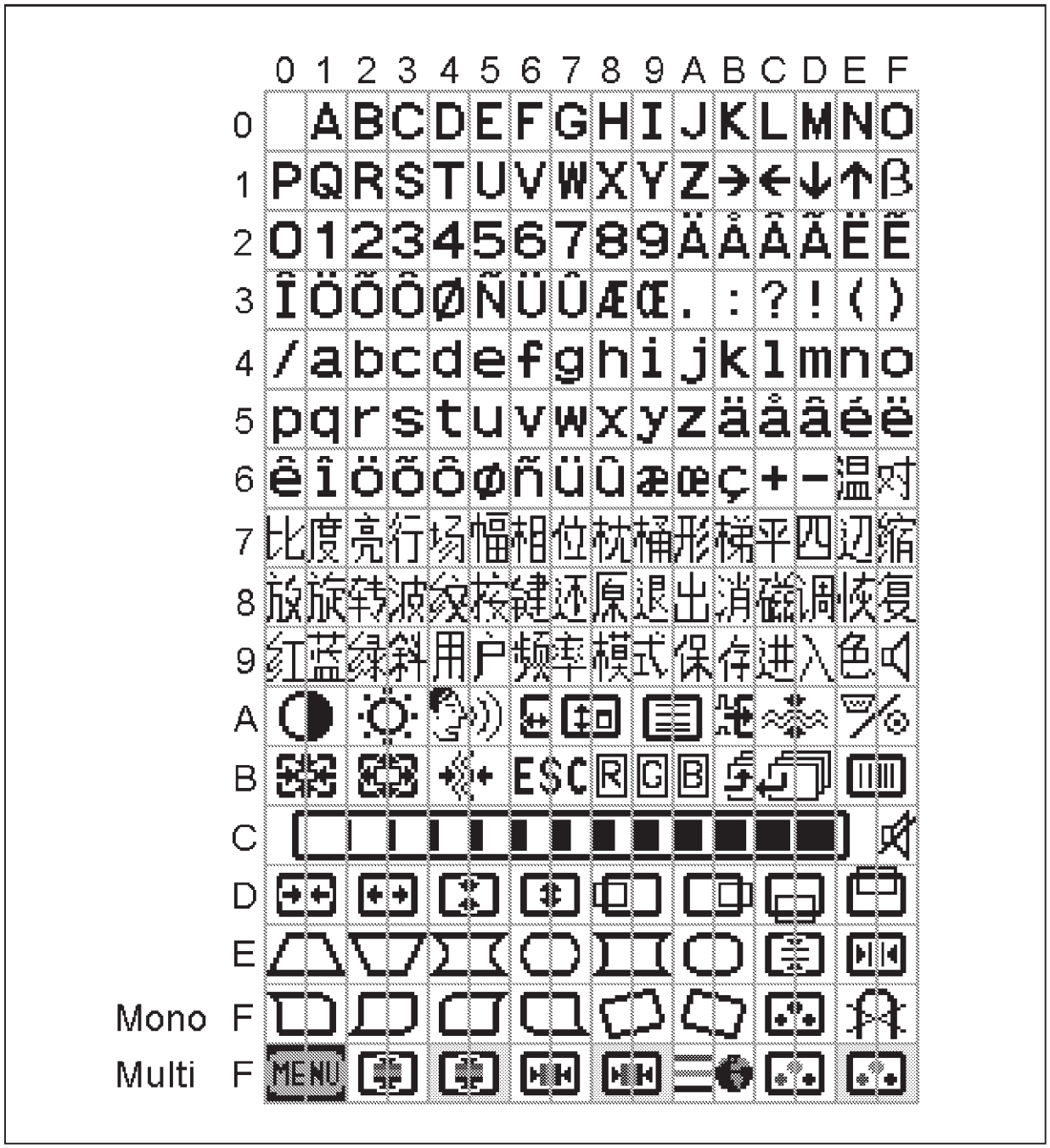
CSHA(0) for window 1

CSHA(I) = 0 : no character shadowing, **CSHA(I)** =1 : character shadowing; (default value = 0000)

Table 23: Character shadowing

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	08	-	-	-	-	CSHA4	CSHA3	CSHA2	CSHA1

Figure 10: Character fonts



8 RAM specification

8.1 Character coding

Each character to display is coded with 11 bits in the RAM with the following addressing:

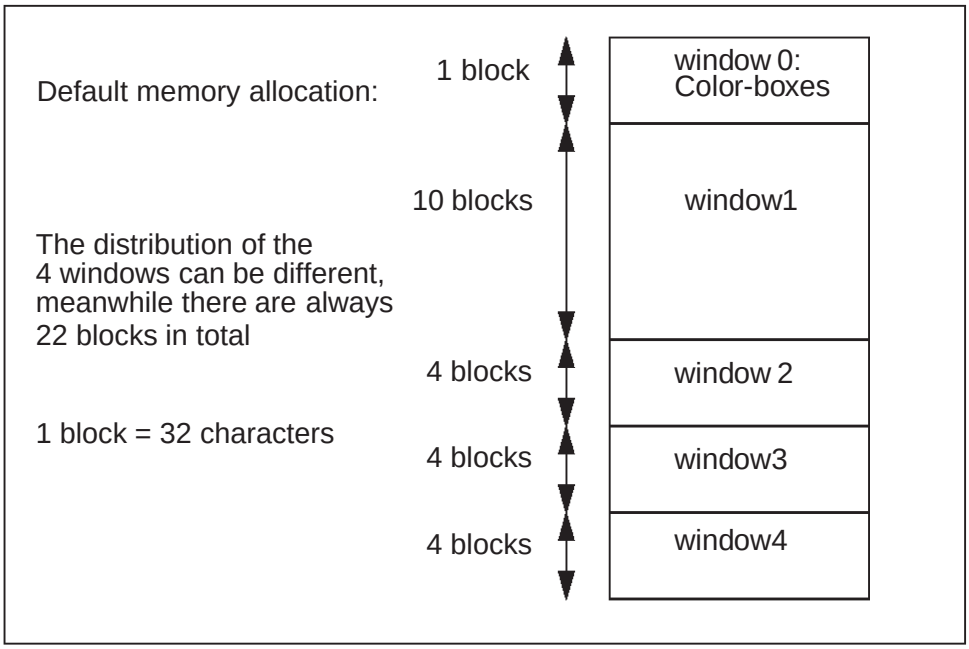
- the character code : 8 bits to address the ROM Code : **RC[7:0]**
- the color code: 3 bits to select 1 among the 8 color boxes of the related window: **CB[2:0]**

Table 24: Character coding

FWR	FAC	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
See *Table 29*		CB(2)	CB(1)	CB(0)	RC(7)	RC(6)	RC(5)	RC(4)	RC(3)	RC(2)	RC(1)	RC(0)

8.2 Window memory space

Figure 11: Window memory space



8.3 Memory size allocation

The total number of characters or spaces is up to 704 with a maximum window size of 16 rows of 32 characters.

The character codes of each window are located in a specific memory space. This memory allocation is programmable for each window. The window size must be inferior or equal to its memory allocation. Any window size can be modified within its specific memory space, the other windows are not affected by this operation.

The user must reserve a memory space for the largest window “i”.

According to the example shown in *Figure 5*, the total numbers of characters/spaces are:

Table 25: Windows sizes

window i	size
window 1	28
window 2	20
window 3	18
window 4	16
total	82

For example, to change the size of window 3 from 3 rows of 6 characters to 5 rows of 4 characters, the resulting size is 20 characters. The number of rows increases and the number of characters per row decreases. The required memory is at least 20 characters/spaces.

The memory allocation is made by block of 32 characters/spaces.

The maximum size of a window is 16 rows of 32 characters = 512 characters corresponding to 16 blocks of 32 characters/spaces.

1 block is reserved for the color-boxes (see *Chapter 7: Characters specification on page 17*).

22 blocks of 32 characters are free for characters codes (704 characters max).

The RAM allocation for each window is described into the 3 following registers of 4 bits:

ALW1- ALW2- ALW3 = ALWI, i from 1 to 3

Window 4 memory allocation uses the remaining memory space.

ALWI [3:0]: The number of memory blocks allocated for window “i” is (**ALWI** +1), the range of allocation is 1 to 16 blocks of 32 characters/spaces.

The total number of blocks is 22.

Note: If the user changes only 1 window allocation, the RAM addresses of the following windows change. Consequently we advise you to write the allocation when the windows are not displayed to avoid false images.

At general reset, the default allocations are :

Window 1 : 10 blocks of 32 words = 320 characters (ALW1 = 9).

Window 2 : 4 blocks of 32 words = 128 characters (ALW2 = 3).

Window 3 : 4 blocks of 32 words = 128 characters (ALW3 = 3).

Window 4 : the remaining RAM (4 blocks = 128 characters).

Table 26: Memory size allocation

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	09	ALW2(3)	ALW2(2)	ALW2(1)	ALW2(0)	ALW1(3)	ALW1(2)	ALW1(1)	ALW1(0)
80	0A					ALW3(3)	ALW3(2)	ALW3(1)	ALW3(0)

8.4 Windows reset

It is possible to reset all the data from1 window in the RAM.

RESETWI =1 : reset all the window data of the allocation memory space.

These bits are automatically cleared when the RAM allocation reset is finished.

(default value = 0000)

All programmable registers are in row 0 (coli, row 0)

Table 27: Allocation enable and windows reset

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	0B	-	-	-	-	RESETW4	RESETW3	RESETW2	RESETW1

9 I2C protocol

The serial interface with the microcontroller is an I2C bus with 2 wires: SCL and SDA.The OSD is a slave circuit with 2 modes: Write and Read.

9.1 Data to write

In the OSD, the I2C allows to write - read:

- the control data
- the character codes to display and their respective color codes
- the color boxes (8 color boxes per window).

Each character code is related to its own window, row and column. Consequently, the protocol of the I2C transmission includes this information (window, row and column) to define the position of the character on the screen. These 3 pieces of information about the position are transmitted in 2 bytes.

As each character on the screen has its own color code, the same protocol is used to write all the color codes and character codes. Only the bit called 'A' allows to distinguish the character codes from the color codes corresponding to 1 position on the screen.

The control data are also written with the same protocol using windows, rows and columns. Window 0 is reserved for control data and color boxes.

9.2 Transmission formats

Three different formats

There are 3 transmission formats to suit the amount of data to update. The transmission format is coded in the "window/row/column" bytes.

The 3 transmission formats are shown here below:

- format (a): S-FWR-FAC-D → FWR-FAC-D → FWR-FAC-D → FWR-FAC-D ...stop
- format (b): S-FWR-FAC-D → FAC-D → FAC-D → FAC-D ...stop
- format (c): S-FWR-FAC-D → D → D →D ...stop

Where S, FWR, FAC, D are transmitted bytes,

S = Slave address = BAh,

FWR = Window and Row address

FAC = Format, Abit and Column address

D = control data or Color-box codes (3 bits) or character codes (8 bits).

In (c) format, data D order of automatical incrementation is: column then row then window.

Table 28: The different bytes coded in the I2C transmission

b7	b6	b5	b4	b3	b2	b1	b0	byte
1	**W[2:0]**			**R[3:0]**				**FWR**
0	**F**	**A**	**C[4:0]**					**FAC**
D[7:0]								**D**: control data (in window 0 only)
0	0	0	0	0	**D[2:0]**			**D**: Color-box code
D[7:0]								**D**: character code

9.2.1 FWR

b7 = 1 to mark the “Window&Row” byte,

W[2:0] : window number

000: control data and color boxes

001: window 1

010: window 2

011: window 3

100: window 4

R[3:0]: row number from 0 to 15: a window has a maximum of 16 rows.

9.2.2 FAC

b7 = 0, to mark the “Attribute&Column” byte,

F = 0: format (a) or (b) , F = 1: format (c) ,

A: transmission of character code or color code

0: character code

1: color code

When reading or writing control data and/or color boxes, **A** = 0

C[4:0]: Column number : 32 possible columns from [00000] to[11111].

9.2.3 D[7:0]

Color codes are on 3 bits

Character codes are on 8 bits.

9.2.4 Configuration of transmission formats

Table 29: Configuration of transmission formats

		b7	b6	b5	b4	b3	b2	b1	b0	byte	format
Address bytes of characters codes	windows & rows	1	**W[2:0]**			**R[3:0]**				**FWR**	a b c
	column(a,b)	0	0	0	C[4:0]					**FAC**	a or b
	column(c)	0	1	0	C[4:0]					**FAC**	c
Address bytes of Color codes	windows & rows	1	**W[2:0]**			**R[3:0]**				**FWR**	a b c
	column(a,b)	0	0	1	**C[4:0]**					**FAC**	a or b
	column(c)	0	1	1	**C[4:0]**					**FAC**	c

All formats must start with S, FWR and FAC bytes.

9.3 Format changing

From format (a) to format (b)

S-FWR(0)- FAC(0)-D(0) → FWR(1)- FAC(1)- D(1) → FWR(2)- FAC(2)- D(2) → FAC(3)- D(3) → FAC(4)- D(4) → FAC(5)- D(5) ...

F bit from FAC byte is always 0 in this case.

From format (a) to format (c)

S - FWR(0)- FAC(0)- D(0) → FWR(1)- FAC(1)- D(1) → FWR(2)- FAC(2)- D(2) → D(3) -> D(4) → D(5) ...

The “F” bit from the FAC byte is as follows:

F(0) = F(1) = “0”

F(2) = “1”

From format (b) to format (a)

S - FWR(0)- FAC(0)-D(0)→ FAC(1)- D(1) → FAC(2)-D(2) → FWR(3)- FAC(3)- D(3) → FWR(4)- FAC(4)- D(4) ...

F bit from FAC byte is always 0 in this case.

From format (b) to format (c)

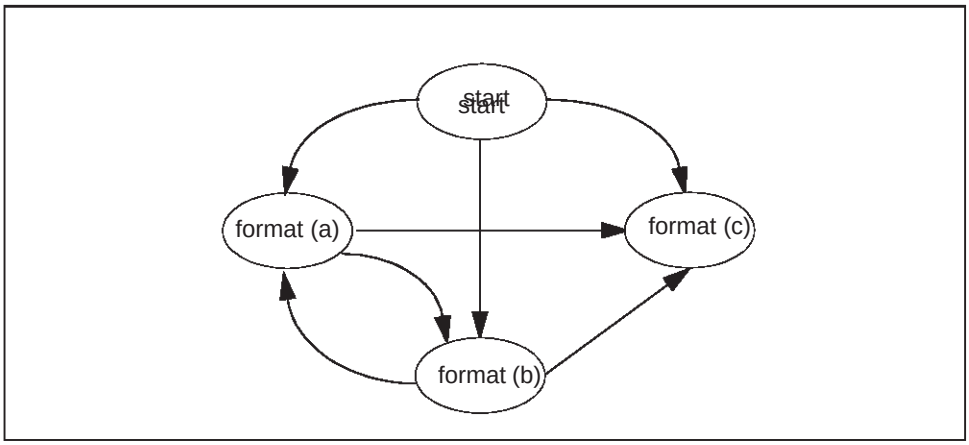
S - FWR(0)- FAC(0)- D(0) → FAC(1)- D(1) → FAC(2)- D(2) -> D(3) → D(4) ...

The “F” bit from the FAC byte is as follows:

F(0) = F(1) = “0” and F(2) = “1”

It is not possible to change from format (c) back to format (b) or (a).

Figure 12: Formats changing



9.4 Use of the different formats

Format (a) is suitable to update small amounts of data which are allocated with different window address, row address and column address.

Format (b) is recommended to update data with the same window, same row address, but a different column address and to change the character/Color-box attribute (bit A), or to write in a different I2C control register.

Format (c) is appropriate to update massive amounts of data from a full window or full screen. The windows, rows and columns addresses are incremented automatically when this format is applied. Data are written to fill all the allocation memory of the windows.

9.5 Read mode

The transmission format is shown as below:

Start - S(w) - FWR- FAC - Stop - Start - S(r) - D -> D -> D -> D ...Stop

with:
S(w) = Slave address in write mode = BAh = 10111010,
S(r) = Slave address in read mode = BBh = 10111011.

Registers and data in RAM are readable.

This mode is useful when developing the OSD application.

9.6 Addressing map

window	row	column	data
window 0	row0	column 0 to 31	control data (8 bits)
window 0	row1	column 0 to 31	reserved (do not write)
window 0	row2	column 0 to 31	color boxes (8 bits)
window 1,2,3,4	row 0 to n (n = 15 max)	column 0 to m (m = 31 max)	characters coding (11bits)

10 Pixel clock generator

The clock generator is used to synchronize the display clock with the horizontal HFLY signal. This block is based on a PLL function to perform a good jitter. The pixel frequency is defined with the line frequency :**FREQ(HLINE)**, the horizontal resolution (**HR[6:0]** register)

Freq($H_{pix)}$ = 12 x HR x Freq(Hline)

VCO[1:0] select the apropriate curve partition of the VCO :

VCO [1:0]	= 0,0	⇒	7.68	MHz	< H_{pix} <	15	MHz (default value = 00)
VCO[1:0]	= 0,1	⇒	15	MHz	< H_{pix} <	30	MHz
VCO[1:0]	= 1,0	⇒	30	MHz	< H_{pix} <	60	MHz
VCO[1:0]	= 1,1	⇒	60	MHz	< H_{pix} <	120	MHz

Table 30: VCO range

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	00	-	-	-	-	-	-	VCO1	VCO0

11 General OSD programmation

11.1 Enable OSD

ENOSD = 1: OSD active,

ENOSD = 0: **FBLK** = 0 (if fbkpol =0) and ROUT, GOUT, BOUT pins =000 (if **RGBPOL** =0) (default value = 0)

Table 31: Enable OSD

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	00	-	-	-	ENOSD	-	-	-	-

11.2 FADE-in, FADE-out

FADE = 1: active, **FADE** = 0: inactive (default value = 0)

Table 32: Fade

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	00	-	-	FADE	-	-	-	-	-

11.3 Fast blanking control: FBK bit and FSRGB register

FBK = 1: **FBLK** always at “1”, the video area is replaced by the color coded in the **FSRGB** register (Full screen register).

FBK = 0: **FBLK** at “1” when an OSD window is displayed (default value = 0)

Full screen register:

FSRGB = FSR (red), FSG (green), FSB(blue) (default value = 000 : black)

Table 33: Full screen registers

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	01	FBK	-	-	-	-	-	-	-
80	03	FSR	FSG	FSB	-	-	-	-	-

11.4 Signals polarity and triggering

Table 34: Signal polarity

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	00	FBKPOL	RGBPOL	-	-	VSP	HSP	-	-

Vertical synchro triggering (VS input)

VSP = 0: falling edge active, **VSP** =1: rising edge active. (default value = 0)

Horizontal synchro triggering (HFLY input)

HSP = 0: falling edge active, **HSP** =1: rising edge active.(default value = 0)**RGBPOL**

ROUT, GOUT, BOUT outputs polarity:

RGBPOL = 0 : RGB active at 1 (default value = 0)

RGBPOL = 1 : RGB active at 0

FBKPOL

FBLK output polarity: (default value = 0)

FBLKPOL = 0 when OSD display, **FBLK** = 1
when active video, **FBLK** = 0

FBLKPOL = 1 when OSD display, **FBLK** = 0
when active video, **FBLK** = 1

Table 35: RGB outputs control

ENOSD bit	RGBPOL bit	RGB outputs	display
1	0	active at 1	OSD
1	1	active at 0	OSD
0	0	000	video
0	1	111	video

Table 36: FBLK output control

ENOSD bit	FBLKPOL bit	FBK bit	FBLK ouput	display	
1	0	0	0	video	default value
1	0	0	1	OSD	
1	0	1	1	OSD	Full Screen
1	1	0	0	OSD	FBLK inverted
1	1	0	1	video	
1	1	1	0	OSD	Full Screen
0	0	x	0	video	No OSD
0	1	x	1	video	No OSD

11.5 Reset

Power On Reset

The digital core and the PLL are asynchronously reset at Power On.

Soft reset

RST = 1 : the digital core is reset. All control registers, except PLL registers, are reset at the same value as at power on reset.

RST =0 at power-up.

It is not necessary to write **RST**=0 to stop the reset. This bit is automatically cleared.

PLL registers reset

RST_PLL = 1 : **HR[6:0], VCO[1:0]** registers are reset at the same value as at power-on reset.

RST_PLL =0 at power-up.

It is not necessary to write **RST_PLL** =0 to stop the reset. This bit is automatically cleared.

Table 37: Reset

FWR	FAC	b7	b6	b5	b4	b3	b2	b1	b0
80	06	-	-	-	-	-	-	RST_PLL	RST

12 Registers

12.1 Registers specification

Table 38: Global parameters

name	section and page	address		description	default value
		col	bits		
ENOSD	*Section 11.1 on page 32*	0	4	enable OSD	0
HR[6:0]	*Section 7.2 on page 17*	1	[6:0]	Horizontal resolution; unit = characters per line	32
VCO[1:0]	*Section 10 on page 31*	0	[1:0]	VCO range	00
CH[6:0]	*Section 7.3 on page 17*	2	[6:0]	Character height; unit = line	18
RSPA[4:0]	*Section 7.4 on page 18*	3	[4:0]	Row spacing (n lines above + n lines below char)	0
HSP	*Section 11.4 on page 33*	0	2	Horizontal sync polarity (0=falling edge)	0
VSP		0	3	Vertical sync polarity (0=falling edge)	0
RGBPOL		0	6	RGB polarity (0=positive)	0
FBKPOL		0	7	Fast blank polarity (0 =positive)	0
HD[6:0]	*Section 6.2 on page 10*	4	[6:0]	Horizontal delay reference (step = 6 pixels)	0
VD[7:0]		5	[7:0]	Vertical delay reference (step = 4 lines)	0
RST	*Section 11.5 on page 34*	6	0	Soft reset (1= reset)	0
RST_PLL		6	1	PLL registers reset (1= reset)	0
FADE	*Section 11.2 on page 32*	0	5	Enable fade	0
FBK	*Section 11.3 on page 32*	1	7	Fast blanking control (0 = normal)	0
FSRGB		3	[7:5]	Full screen colour	black

Table 39: Windows parameters

name	name/ window	section and page	address (hex)		description	default value
			col	bits		
ENWI	**ENW4,3,2,1**	*Section 6.1 on page 10*	7	3,2,1,0	Window enables	0000
ENBSI	**ENBS4,3,2,1**	*Section 6.6.1 on page 14*	7	7,6,5,4	Border/shadow enables	0000
CSHA[3:0]		*Section 7.7 on page 22*	8	[3:0]	Character shadowing enables	0000
MCOLOR[3:0]		*Section 7.6 on page 21*	8	[7:4]	Multicolour enables	0000
ALWI[3:0]	**ALW1[3:0]**	*Section 6.5 on page 14*	9	[3:0]	Windows allocations : 32 * (ALWi + 1) characters 1 block = 32 characters (ALWi +1) blocks	9 = 320 char.
	ALW2[3:0]		9	[7:4]		3 = 128 char.
	ALW3[3:0]		A	[3:0]		3 = 128 char.
RESETWI	**RESETW4,3,2,1**	*Section 8.4 on page 26*	B	3,2,1,0	Windows resets (1 = reset)	0000
HDWI[6:0]	**HDW1[6:0]**	*Section 6.3 on page 12*	C	[6:0]	Horizontal delay: (12 * HDWi) pixels	0
	HDW2[6:0]		11	[6:0]		32
	HDW3[6:0]		16	[6:0]		0
	HDW4[6:0]		1B	[6:0]		16
VDWI[5:0]	**VDW1[5:0]**		D	[5:0]	Vertical delay: VDWi rows	0
	VDW2[5:0]		12	[5:0]		0
	VDW3[5:0]		17	[5:0]		12
	VDW4[5:0]		1C	[5:0]		12
HSWI[4:0]	**HSW1[4:0]**	*Section 6.4 on page 13*	E	[4:0]	Horizontal size: (HSWi + 1) characters	25 = 26 char.
	HSW2[4:0]		13	[4:0]		9 = 10 char.
	HSW3[4:0]		18	[4:0]		15 = 16char.
	HSW4[4:0]		1D	[4:0]		15 = 16char.
VSWI[3:0]	**VSW1[3:0]**		F	[3:0]	Vertical size: (VSWi + 1) rows	11 = 12 rows
	VSW2[3:0]		14	[3:0]		4 = 5 rows
	VSW3[3:0]		19	[3:0]		7 = 8 rows
	VSW4[3:0]		1E	[3:0]		7 = 8 rows

Table 39: Windows parameters

name	name/ window	section and page	address (hex)		description	default value
			col	bits		
RGBWI	**RGB W1**	*Section 6.5 on page 14*	10	2,1,0	window background colour: 000=black, 001=blue, 010=green, 011=cyan, 100=red, 101=magenta, 110=yellow, 111=white	white
	RGB W2		15	2,1,0		white
	RGB W3		1A	2,1,0		white
	RGB W4		1F	2,1,0		white
TI	**T1**		10	3	1 = transparent background (video active), 0 = RGBWi (if BC bit of colorbox = 0)	0
	T2		15	3		0
	T3		1A	3		0
	T4		1F	3		0
BSWI	**BSW1**	*Section 6.6.1 on page 14*	10	4	0 = border, 1 = shadow	0
	BSW2		15	4		0
	BSW3		1A	4		0
	BSW4		1F	4		0
WSRGBI	**WS RGB 1**	*Section 6.6.2 on page 15*	10	7,6,5	Border/shadow colour	black
	WS RGB 2		15	7,6,5		black
	WS RGB 3		1A	7,6,5		black
	WS RGB 4		1F	7,6,5		black
BSWWI[2:0]	**BSWW1[2:0]**	*Section 6.6.3 on page 15*	E	[7:5]	Border/shadow width: (BSWWi * 2) pixels	0
	BSWW2[2:0]		13	[7:5]		0
	BSWW3[2:0]		18	[7:5]		0
	BSWW4[2:0]		1D	[7:5]		0
BSHWI[3:0]	**BSHW1[3:0]**		F	[7:4]	Border/shadow height: (BSHWi * 2) lines	0
	BSHW2[3:0]		14	[7:4]		0
	BSHW3[3:0]		19	[7:4]		0
	BSHW4[3:0]		1E	[7:4]		0

The color boxes are located at addresses: window=0, row=2. See *Section 7.5 on page 19*.

The character codes are located at addresses from window1 to window4, as described in *Section 8.1 on page 24*.

It is recommended to write “0” for the data indicated with “-” in *Table 40* and *Table 42*.

Table 40: Control registers : window 0, W[2:0] = “000”, row = 0 R[3:0] =”0000”

address (in hexa)			b7	b6	b5	b4	b3	b2	b1	b0
FWR	FAC	Col								
80	00	0	**FBKPOL**	**RGBPOL**	**FADE**	**ENOSD**	**VSP**	**HSP**	**VCO1**	**VCO0**
80	01	1	**FBK**	**HR[6:0]** = horizontal resolution						
80	02	2	-	**CH[6:0]** = character height						
80	03	3	**FSR**	**FSG**	**FSB**	**RSPA[4:0]** = row spacing				
80	04	4	-	**HD[6:0]** = horizontal delay reference						
80	05	5	**VD[7:0]** = vertical delay reference							
80	06	6	-	-	-	-	-	-	**RST_PLL**	**RST**
80	07	7	**ENBS4**	**ENBS3**	**ENBS2**	**ENBS1**	**ENW4**	**ENW3**	**ENW2**	**ENW1**
80	08	8	**MCOLOR4**	**MCOLOR3**	**MCOLOR2**	**MCOLOR1**	**CSHA4**	**CSHA3**	**CSHA2**	**CSHA1**
80	09	9	**ALW2(3)**	**ALW2(2)**	**ALW2(1)**	**ALW2(0)**	**ALW1(3)**	**ALW1(2)**	**ALW1(1)**	**ALW1(0)**
80	0A	A	**-**	**-**	**-**	**-**	**ALW3(3)**	**ALW3(2)**	**ALW3(1)**	**ALW3(0)**
80	0B	B	**-**	**-**	**-**	**-**	**RESETW4**	**RESETW3**	**RESETW2**	**RESETW1**
80	0C	C	**-**	**HDW1(6)**	**HDW1(5)**	**HDW1(4)**	**HDW1(3)**	**HDW1(2)**	**HDW1(1)**	**HDW1(0)**
80	0D	D	**-**	**-**	**VDW1(5)**	**VDW1(4)**	**VDW1(3)**	**VDW1(2)**	**VDWI(1)**	**VDW1(0)**
80	0E	E	**BSWW1(2)**	**BSWW1(1)**	**BSWW1(0)**	**HSW1(4)**	**HSW1(3)**	**HSW1(2)**	**HSW1(1)**	**HSW1(0)**
80	0F	F	**BSHW1(3)**	**BSHW1(2)**	**BSHW1(1)**	**BSHW1(0)**	**VSW1(3)**	**VSW1(2)**	**VSW1(1)**	**VSW1(0)**
80	10	10	**WSR1**	**WSG1**	**WSB1**	**BSW1**	**T1**	**RW1**	**GW1**	**BW1**
80	11	11	**-**	**HDW2(6)**	**HDW2(5)**	**HDW2(4)**	**HDW2(3)**	**HDW2(2)**	**HDW2(1)**	**HDW2(0)**
80	12	12	**-**	**-**	**VDW2(5)**	**VDW2(4)**	**VDW2(3)**	**VDW2(2)**	**VDW2(1)**	**VDW2(0)**
80	13	13	**BSWW2(2)**	**BSWW2(1)**	**BSWW2(0)**	**HSW2(4)**	**HSW2(3)**	**HSW2(2)**	**HSW2(1)**	**HSW2(0)**
80	14	14	**BSHW2(3)**	**BSHW2(2)**	**BSHW2(1)**	**BSHW2(0)**	**VSW2(3)**	**VSW2(2)**	**VSW2(1)**	**VSW2(0)**
80	15	15	**WSR2**	**WSG2**	**WSB2**	**BSW2**	**T2**	**RW2**	**GW2**	**BW2**
80	16	16	**-**	**HDW3(6)**	**HDW3(5)**	**HDW3(4)**	**HDW3(3)**	**HDW3(2)**	**HDW3(1)**	**HDW3(0)**
80	17	17	**-**	**-**	**VDW3(5)**	**VDW3(4)**	**VDW3(3)**	**VDW3(2)**	**VDW3(1)**	**VDW3(0)**
80	18	18	**BSWW3(2)**	**BSWW3(1)**	**BSWW3(0)**	**HSW3(4)**	**HSW3(3)**	**HSW3(2)**	**HSW3(1)**	**HSW3(0)**
80	19	19	**BSHW3(3)**	**BSHW3(2)**	**BSHW3(1)**	**BSHW3(0)**	**VSW3(3)**	**VSW3(2)**	**VSW3(1)**	**VSW3(0)**
80	1A	1A	**WSR3**	**WSG3**	**WSB3**	**BSW3**	**T3**	**RW3**	**GW3**	**BW3**
80	1B	1B	**-**	**HDW4(6)**	**HDW4(5)**	**HDW4(4)**	**HDW4(3)**	**HDW4(2)**	**HDW4(1)**	**HDW4(0)**
80	1C	1C	**-**	**-**	**VDW4(5)**	**VDW4(4)**	**VDW4(3)**	**VDW4(2)**	**VDW4(1)**	**VDW4(0)**
80	1D	1D	**BSWW4(2)**	**BSWW4(1)**	**BSWW4(0)**	**HSW4(4)**	**HSW4(3)**	**HSW4(2)**	**HSW4(1)**	**HSW4(0)**
80	1E	1E	**BSHW4(3)**	**BSHW4(2)**	**BSHW4(1)**	**BSHW4(0)**	**VSW4(3)**	**VSW4(2)**	**VSW4(1)**	**VSW4(0)**
80	1F	1F	**WSR4**	**WSG4**	**WSB4**	**BSW4**	**T4**	**RW4**	**GW4**	**BW4**

Table 41: Color registers : window 0, W[2:0] = “000”, row = 2 R[3:0] =”0010”

address			b7	b6	b5	b4	b3	b2	b1	b0
FWR	FAC	Col								
82	00	0	WINDOW1	COLOR BOX 1 : BC- BR-BG-BB-BLINK-FR-FG-FB						
82	01	1	WINDOW1	Color Box 2 : BC- BR-BG-BB-blink-FR-FG-FB						
82	02	2	WINDOW1	Color Box 3 : BC- BR-BG-BB-blink-FR-FG-FB						
82	03	3	WINDOW1	Color Box 4 : BC- BR-BG-BB-blink-FR-FG-FB						
82	04	4	WINDOW1	Color Box 5 : BC- BR-BG-BB-blink-FR-FG-FB						
82	05	5	WINDOW1	Color Box 6 : BC- BR-BG-BB-blink-FR-FG-FB						
82	06	6	WINDOW1	Color Box 7 : BC- BR-BG-BB-blink-FR-FG-FB						
82	07	7	WINDOW1	Color Box 8 : BC- BR-BG-BB-blink-FR-FG-FB						
82	08	8	WINDOW2	Color Box 1 : BC- BR-BG-BB-blink-FR-FG-FB						
82	09	9	WINDOW2	Color Box 2 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1A	A	WINDOW2	Color Box 3 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1B	B	WINDOW2	Color Box 4 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1C	C	WINDOW2	Color Box 5 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1D	D	WINDOW2	Color Box 6 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1E	E	WINDOW2	Color Box 7 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1F	F	WINDOW2	Color Box 8 : BC- BR-BG-BB-blink-FR-FG-FB						
82	10	10	WINDOW3	Color Box 1 : BC- BR-BG-BB-blink-FR-FG-FB						
82	11	11	WINDOW3	Color Box 2 : BC- BR-BG-BB-blink-FR-FG-FB						
82	12	12	WINDOW3	Color Box 3 : BC- BR-BG-BB-blink-FR-FG-FB						
82	13	13	WINDOW3	Color Box 4 : BC- BR-BG-BB-blink-FR-FG-FB						
82	14	14	WINDOW3	Color Box 5 : BC- BR-BG-BB-blink-FR-FG-FB						
82	15	15	WINDOW3	Color Box 6 : BC- BR-BG-BB-blink-FR-FG-FB						
82	16	16	WINDOW3	Color Box 7 : BC- BR-BG-BB-blink-FR-FG-FB						
82	17	17	WINDOW3	Color Box 8 : BC- BR-BG-BB-blink-FR-FG-FB						
82	18	18	WINDOW4	Color Box 1 : BC- BR-BG-BB-blink-FR-FG-FB						
82	19	19	WINDOW4	Color Box 2 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1A	1A	WINDOW4	Color Box 3 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1B	1B	WINDOW4	Color Box 4 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1C	1C	WINDOW4	Color Box 5 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1D	1D	WINDOW4	Color Box 6 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1E	1E	WINDOW4	Color Box 7 : BC- BR-BG-BB-blink-FR-FG-FB						
82	1F	1F	WINDOW4	Color Box 8 : BC- BR-BG-BB-blink-FR-FG-FB						

12.2 Registers at reset

Table 42: Control registers : window 0 W[2:0] = "000"

address (in hexa)			b7	b6	b5	b4	b3	b2	b1	b0
FWR	FAC	Col								
80	00	0	**FBKPOL**=0	**RGBPOL**=0	**FADE** = 0	**ENOSD**=0	**VSP** = 0	**HSP** = 0	**VCO[1:0]** = 00	
80	01	1	**FBK** = 0	**HR[6:0]** : horizontal resolution = 32 characters						
80	02	2	-	**CH[6:0]** = character height = 18						
80	03	3	full sreen **RGB** = FS **RGB** = 000			**RSPA[4:0] =** row spacing = 0				
80	04	4	-	**HD[6:0]** = horizontal delay reference = 0 (50 pixels)						
80	05	5	**VD[7:0] =** vertical delay reference = 0 (2 lines)							
80	06	6	-	-	-	-	-	-	**RST_PLL**=0	**RST** = 0
80	07	7	**ENBS4/3/2/1** = 0000				**ENW4/3/2/1** = 0000			
80	08	8	**MCOLOR4/3/2/1** = 0000				**CSHA4/3/2/1** = 0000			
80	09	9	**ALW2[3:0]**= 3 (4 blocks = 128 characters)				**ALW1[3:0]** = 9 (10 blocks = 320 characters)			
80	0A	A	-	-	-	-	**ALW3[3:0]** = 3 (4 blocks = 128 characters)			
80	0B	B	-	-	-	-	**RESETW4/3/2/1** = 0000			
80	0C	C	-	**HDW1[6:0]** = 0						
80	0D	D	-	-	**VDW1[5:0]** = 0					
80	0E	E	**BSWW1[2:0]** = 000			**HSW1[4:0]** =25 (26 characters)				
80	0F	F	**BSHW1[3:0]** = 0000				**VSW1[3:0]** = 11 (12 row of characters)			
80	10	10	**WS RGB 1** = 000 : black			**BSW1**=0	**T1** = 0	**RGB W1** = 111 :white		
80	11	11	-	**HDW2[6:0]** = 32						
80	12	12	-	-	**VDW2[5:0]** = 0					
80	13	13	**BSWW2[2:0]** = 000			**HSW2[4:0]** = 9 (10 characters)				
80	14	14	**BSHW2[3:0]** = 0000				**VSW2[3:0]**= 4 (5 row of characters)			
80	15	15	**WS RGB 2** = 000: black			**BSW2** =0	**T2** =0	**RGB W2** = 111:white		
80	16	16	-	**HDW3[6:0]** = 0						
80	17	17	-	-	**VDW3[5:0]** = 12					
80	18	18	**BSWW3[2:0]** = 000			**HSW3[4:0]** = 15 (16 characters)				
80	19	19	**BSHW3[3:0]** = 000				**VSW3[3:0]**= 7 (8 row of characters)			
80	1A	1A	**WS RGB 3** = 000: black			**BSW3** = 0	**T3** =0	**RGB W3** = 111 :white		
80	1B	1B	-	**HDW4[6:0]** = 16						
80	1C	1C	-	-	**VDW4[5:0]** = 12					
80	1D	1D	**BSWW4[2:0]** = 000			**HSW4[4:0]** =15 (16 characters)				
80	1E	1E	**BSHW4[3:0]** = 0000				**VSW4[3:0]**= 7 (8 rows of characters)			
80	1F	1F	**WS RGB 4** = 000: black			**BSW4**=0	**T4** =0	**RGB W4** = 111:white		

13 Application hints

13.1 Software hints

13.1.1 Programming recommendations

1 Write a new allocation just before the RAM reset.

2 Write a new allocation at any time but take care of the window display.

3 When resetting the RAM and writting in it just after, write in the RAM respecting the same order as the reset does: from the first to the last reset window, from the first window address (row 0, col 0) to the last, incrementing columns, then rows, then windows.

4 Define the window horizontal size prior to writing character and color codes in RAM. HSWI is used to compute the RAM address.

13.1.2 Examples of programming

Hard reset at power-up (following a power-up)

1 Write Window 0 registers to set the OSD parameters: write

→ VCO[1:0], horizontal resolution and vertical height of characters,

→ the position of reference,

→ the allocations if they are wrong
(by default : 320 characters for window 1, 128 characters for each of the others windows)

→ the windows position and size,

→ the color-boxes that will be used.

2 Write the character codes for each window to display.

3 Write the color-box codes for each window to display.

4 Write the enable of windows : EnWi = 1 then EnOSD=1.

Change of position & size of 1 window (ex. window 3) without disable of window

1 Write new position and sizes.

2 Write new characters in the RAM.

Re-allocation, reset, and writing new characters in windows

1 Disable windows.

2 Write new allocations.

3 Reset the windows.

4 Write new positions and sizes in control registers.

5 Write new color-boxes.

6 Write new characters and color-box code.

7 Enable windows.

13.2 Hardware hints

- The serial resistors on the R, G, B and FBLK outputs must be as close as possible to the device.
- Both decoupling capacitors (100nF and 100 µF) must be as close as possible to the analog (pin 13) and digital (pin5) power supplies (see *Figure 13*).
- PLL network must be close to the device but far from the R, G, B, FBLK outputs. PLL network and R, G, B, FBLK outputs should be separated by the AVDD 3.3 V power trace (see *Figure 14* and *Figure 15*).
- PLL ground (AGND) **should not** be connected either to DVSS or to other grounds of the videoboard, as the ground is already connected internally (see *Figure 14* and *Figure 15*).

14 Application diagrams

Figure 13: STV9936 - application diagram

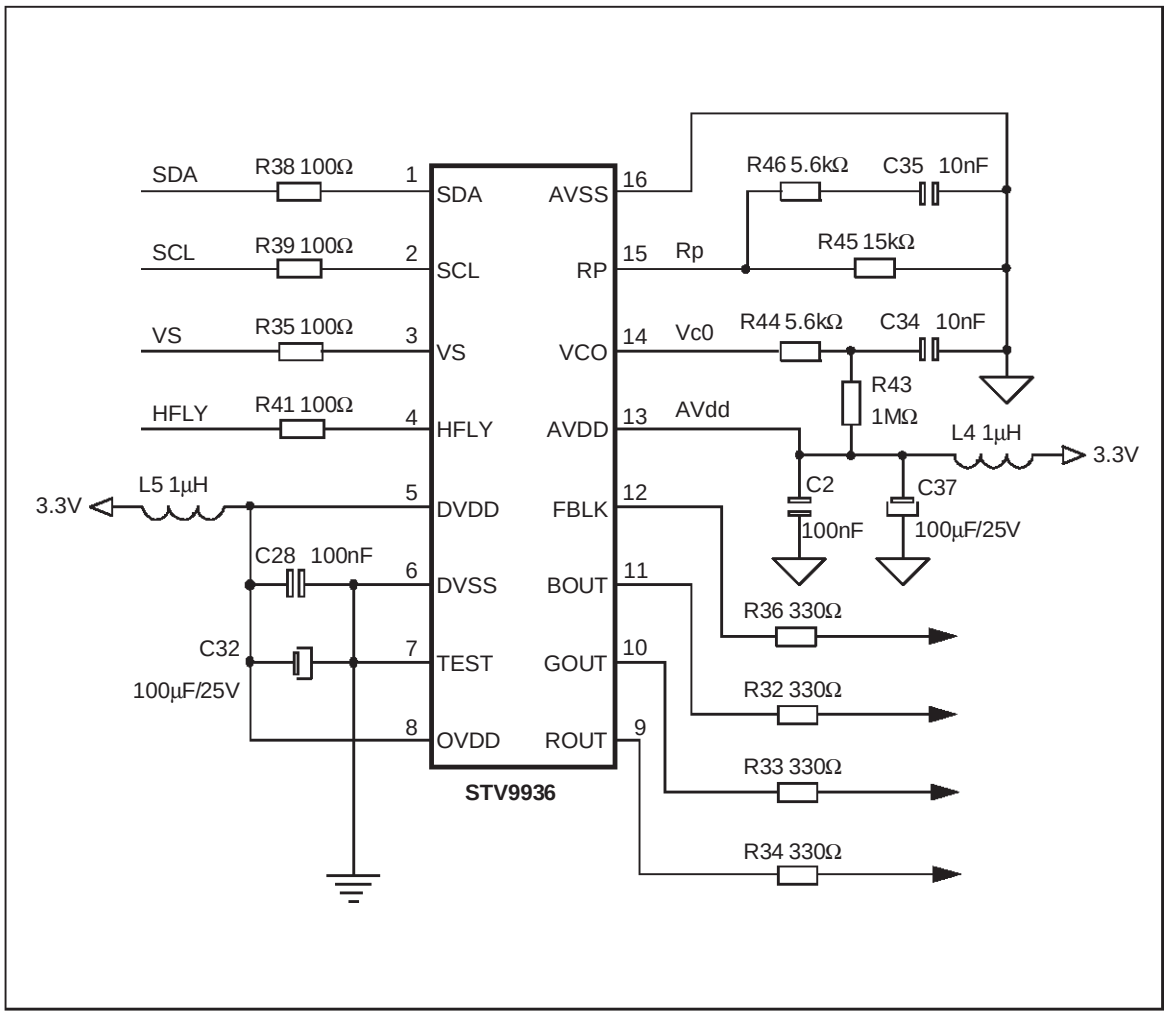


Figure 14: STV9936 demonstration board

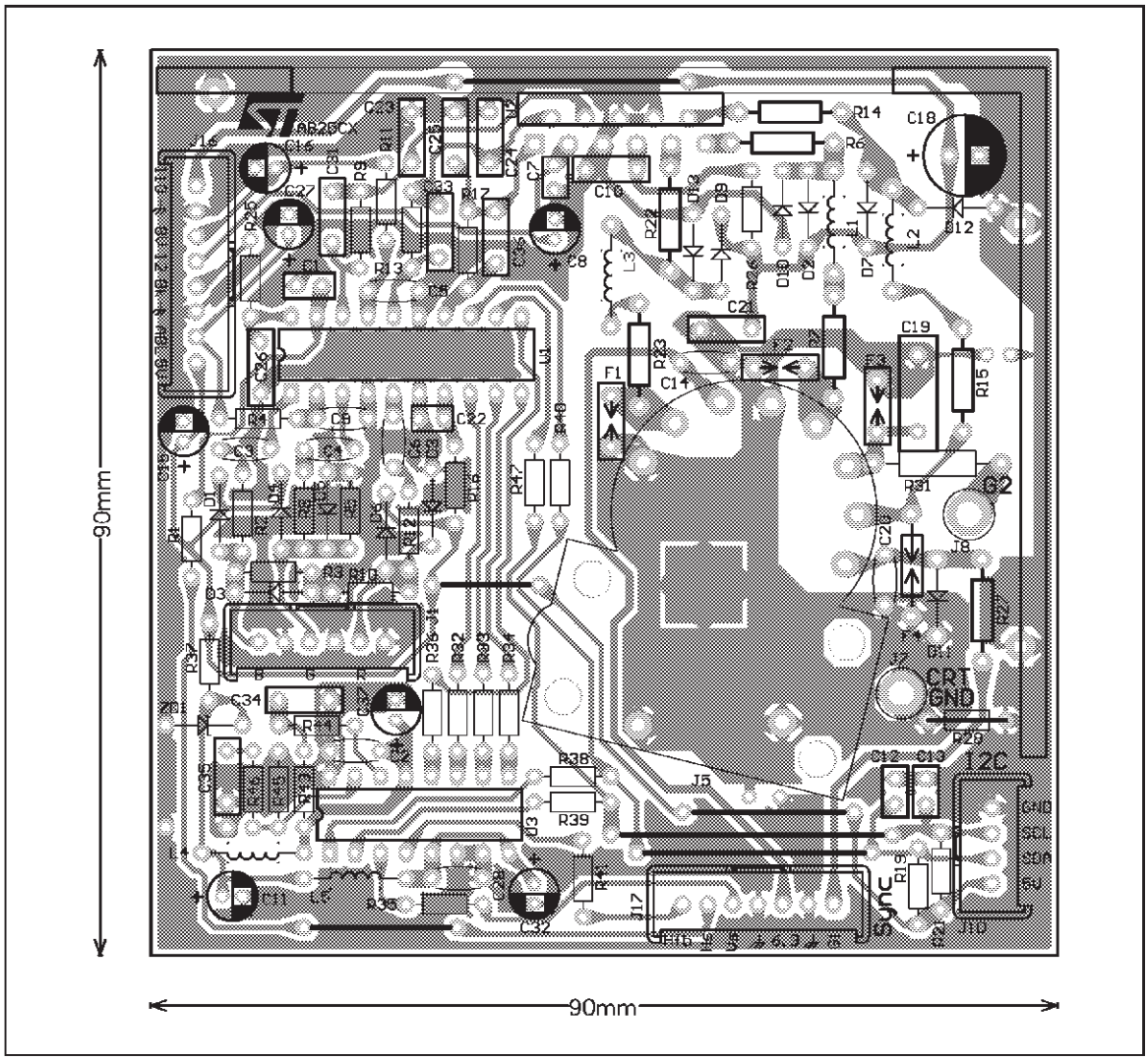


Figure 15: STV9936 - zoom

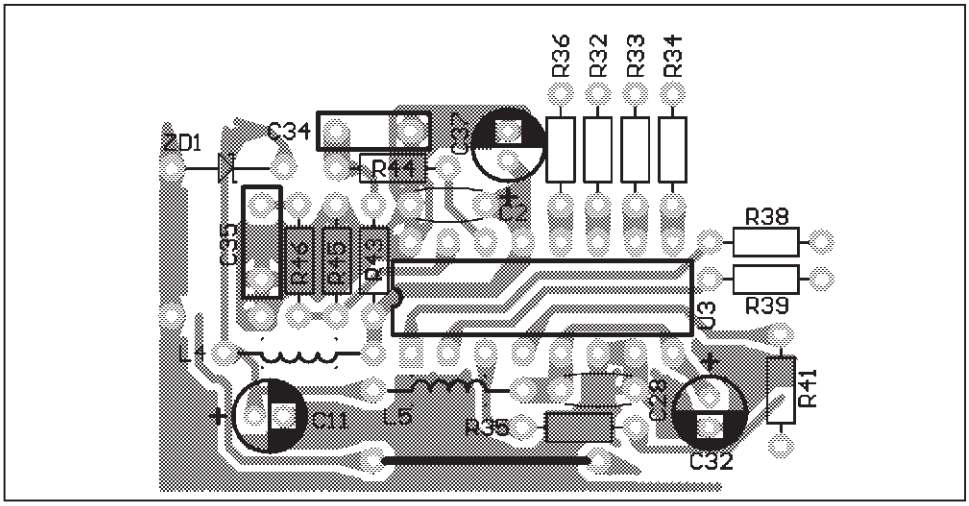


Figure 16: Demonstration board schematics of the STV955X - TDA9210 - STV9936