OpenCores.Org

ROSETTA

Configurable Dot Matrix Display Controller

Roberto M. Cibils

Rev. 0.1 June 4, 2006

Revision History

Rev.	Date	Author	Description
0.1	06/04/06	Roberto M. Cibils	First Draft

Contents

Introduction

The **ROSETTA** **C**onfigurable **D**ot **M**atrix **D**isplay **C**ontroller core provides a modular expandable interface for any dimension displays build from LEDs dot matrix structures. This core can be configured for synthesis and P&R with any number of DM LEDs **slices**. These **slices** can be arranged in any number to form linear or plane screens.

In designs with a big number of **slices**, the number of pins required for the implementation device selected could collapse. In order to overcome this inconvenience, the output bus has a multiplexed structure, and the size of the display can be expanded using external buffer devices.

In limited size implementations, the display columns could optionally be serviced using the controller chip drivers, if a device with enough driving capability is used. On the other hand, the row lines will always need some external buffering.

Features

- **WISHBONE** interface in 8,16, 32 or 64-bit data bus modes
- **SystemC, Verilog** and **VHDL** languages
- Use of **Dot Matrix LEDs** Display structures.
- Configurable for single color, RG or RGB LEDs use
- Operation Modes: Test, Full Text, Full Graphics, Mixed Text & Graphics, Animation
- Scroll: Horizontal, Vertical and Combined
- Colors: configurable from monochromatic to any number of colors
- Variable Buffer size
- Modular expansion

Linear Array Displays Implementation

The **slices** can be arranged in a linear array to show a string of characters, forming a "one-dimensional" display. The **ROSETTA** controller can provide service up to **h** slices, each having **w** columns width and **r** rows thickness. Each slice is called a **sector**. In order to extend the size of the display, several **Latch & Drivers** devices should be used, one for each **sector**.

Both, the **rows** and the **columns** buses are multiplexed. In the first case, just one row at a time is enabled. In this case, bus driving is needed for each sector. In the case of the **columns bus**, the refreshing operation is made one sector at a time. Every sector columns pattern needs to be latched during other sectors refreshing.

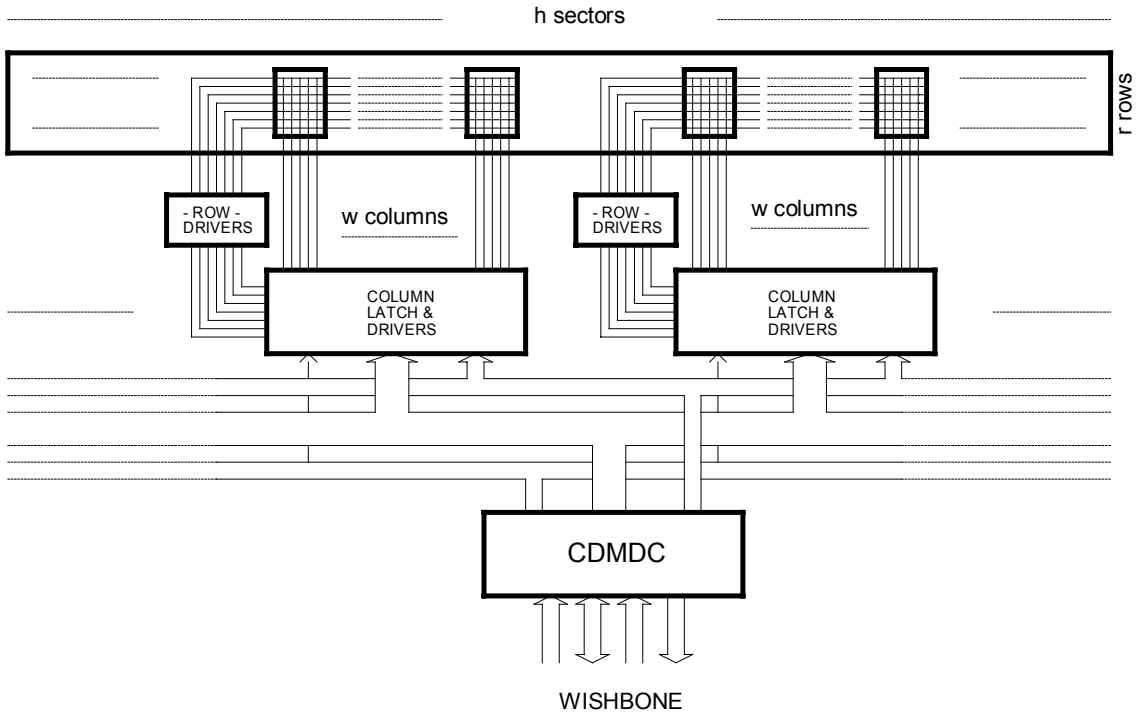

Figure 1.1

Matrix Array Displays Implementation

The sectors can also be arranged in a piled up way, forming a "two-dimensional" or matrix array display with **h** x **r** x **w** pixels size.

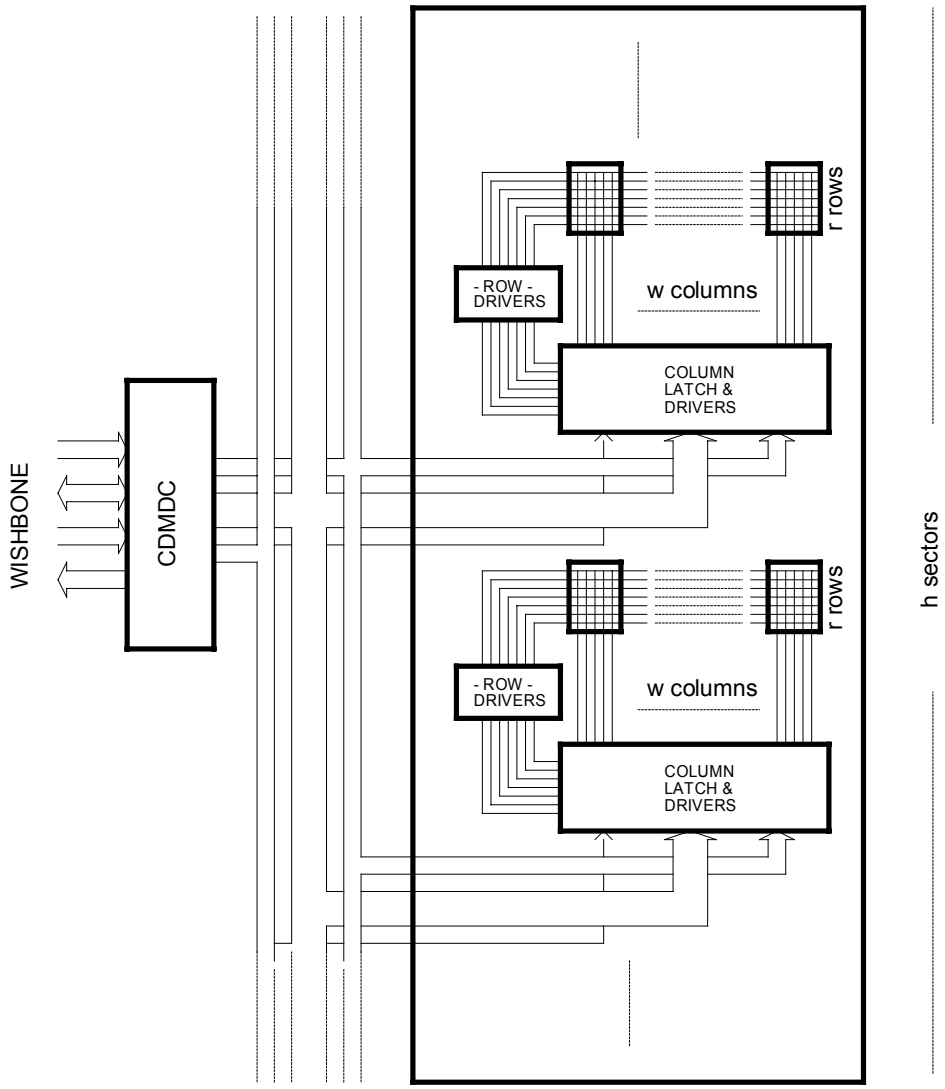

Figure 1.2

Color Management

The **ROSETTA** controller can be configured for managing monochromatic pixels (single color LEDs), RG combination pixels (RG LEDs) or RGB combination pixels (RGB LEDs) as can be seen in the following figure where the switches represents the ROSETTA outputs needed for driving each column. The parameter defining the number of primary colors used in each pixel is "**p**".

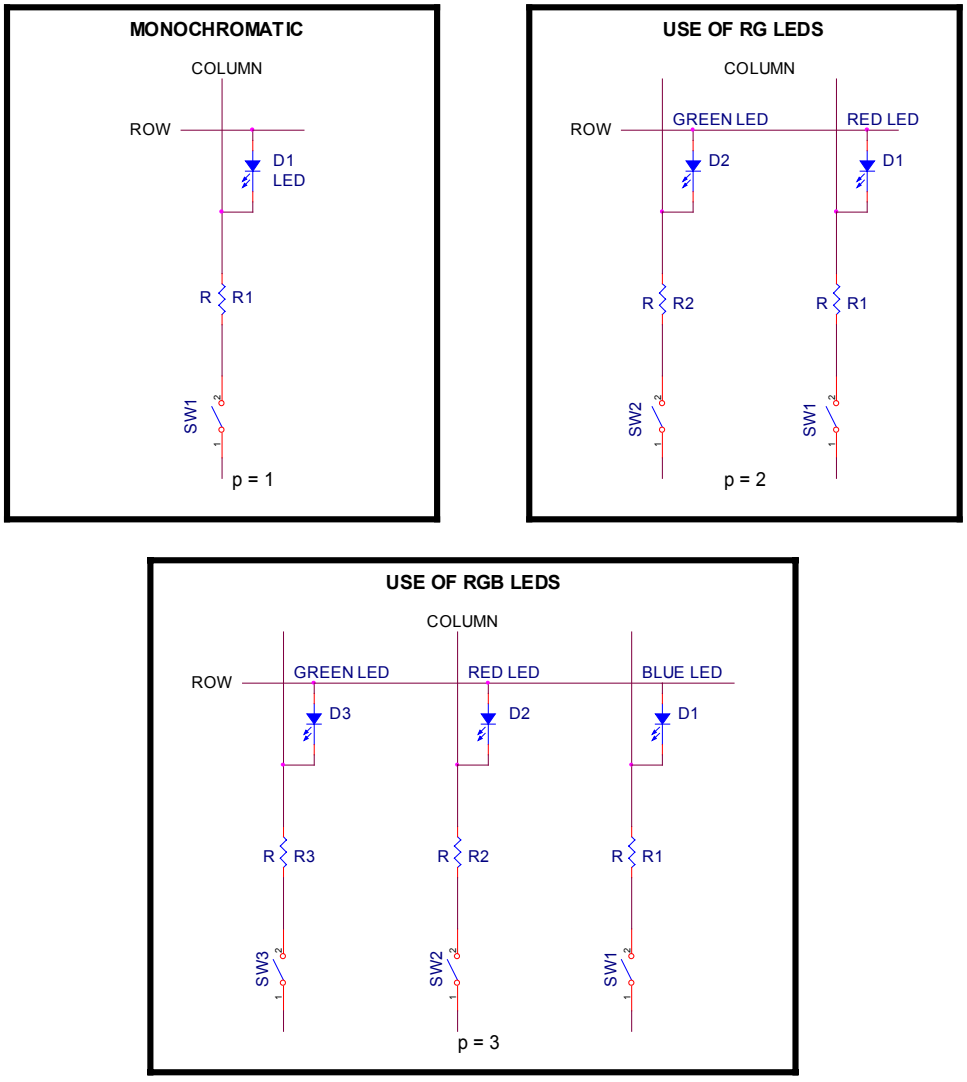

Figure 1.3

In the case of using RG LEDs or RGB LEDs as shown in the previous figure, just 4 or 8 different colors can be obtained respectively when all the different combination of "on" and "off" in the LEDs are used. If it is needed a bigger number of colors, the contribution of each LED can be modified by controlling the current through any or all of them. This control can be implemented as can be seen in the following figure:

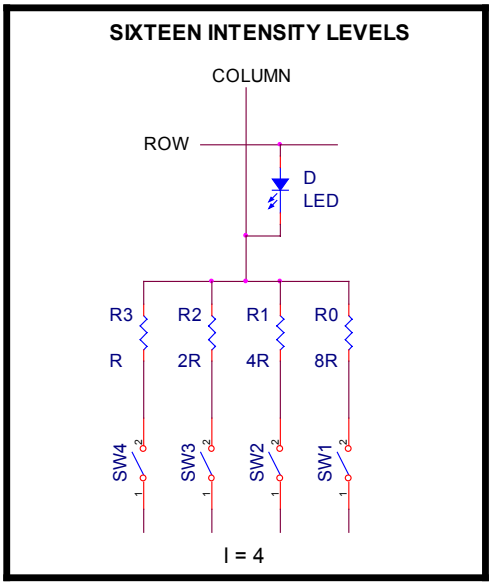

Figure 1.4

The **ROSETTA** controller can be configured for any number of different current levels in any or all the LEDs for each pixel. The parameter that defines the number of intensities selected for each primary color is "**i**", but for configuring the core we will use "**I**" where:

$$\mathbf{I = \log_2(i)}$$

Configuring the number of levels of current in any led and the number of different color LEDs used for each pixel it is possible to choose several implementation alternatives going from monochromatic toward up to any number of different colors. In the following table a group of implementations can be seen going from monochromatic up to 4096 different colors:

NUMBER OF COLORS			
i LEVELS	**MONO**	**RG**	**RGB**
2	2	4	8
4	4	16	64
8	8	64	512
16	16	256	4096

Table 1.1

In strict sense, the numbers corresponding to the column MONO don't correspond to different colors but to different intensity levels in the same color.

The price to be paid for getting more and more colors is the need of driving more lines per pixel columns as can be seen in the following table:

NUMBER OF OUTPUTS PER COLUMN			
i LEVELS	**MONO**	**RG**	**RGB**
2	1	2	3
4	2	4	6
8	3	6	9
16	4	8	12

Table 1.2

Wishbone Bus Interface Implementation

The **ROSETTA** controller can be configured for using 8, 16, 32 or 64 bits of **WISHBONE** data bus width in slave mode. It also supports single as well as block R/W cycles.

The **WISHBONE** interface uses a buffer called **TRB** (**T**emporary **R**ow **B**uffer) in order to format the data coming from the **WISHBONE** bus. This buffer is sized one row (**wI**) wide and **n** rows deep. While in Graphic mode, there is a relation of one row transferred for each row formed in the **TRB**, in Text mode it is necessary an amount of **r** x **p** transfers for each row formed due to the parallel graphic conversion of each character arrived. This condition forces the designer to chose between several trades off.

If the designer uses the **TRB** as a regular buffer (**n** = **1**), the relation between the minimum **ROSETTA** system clock frequency ($\mathbf{f_{SYS}}$) and the maximum **WISHBONE** bus clock frequency ($\mathbf{f_{WB}}$) should be:

$$\mathbf{f_{SYS}} = \mathbf{r} \times \mathbf{p} \times \mathbf{f_{WB}}$$

in order to transfer all the graphic data produced toward the **BR** in just one **WISHBONE** cycle and make the **TRB** free to store a new row. This condition could produce a requirement for a $\mathbf{f_{SYS}}$ greater than the allowed for the device selected.

To overcome this inconvenience, it is possible to use **TRB** in **FIFO** mode making **n** greater than one. The price paid for this is an increase in the amount of device's **FF**s requirement that can be estimated as **r** x **p** x (**n** – 1) **FF**s "extra" required.

The new relation between clocks frequency results:

$$\mathbf{f_{SYS}} = [(\mathbf{r} \times \mathbf{p})/\mathbf{n}] \times \mathbf{f_{WB}}$$

Another way to release the design from this constrains is to add wait states (**WSS**) on the bus operation during the period of time in which the **WISHBONE** interface is busy transferring data to the **BR**.

This is a choice that could be useful if the bus masters has the capability to deal with that. Anyway, **r** x **p** **WSS** bus cycles could involve too much time for the **ROSETTA** to keep using the bus. Nevertheless, a suitable combination of **TRB** depth and **WSS** could be the optimum choice. In that case the new relation between frequencies will be:

$$\mathbf{f_{SYSmín} = [(r \times p)/(n \times N_{WSS})] \times f_{WB}}$$

where:

$\mathbf{N_{WSS}}$ = number of wait states added

Parameters Configuration Constrains

Although the **ROSETTA** controller is a configurable core, the range of values that each parameter can assume is limited. Following we will enumerate several design rules that must be taken into account when defining this values.

1- The number of luminous intensities **i** should always be an integer, power of two with a minimum value of 2^1. There is not a maximum limit in the value for this parameter.

2- The color component **p** index should always be an integer with a minimum value of 1 and a maximum of 3.

3- The page number **d** and the number of slices **h** just have the limitation of being integers with a minimum value of 1. It is desirable to use numbers being power of 2 as frequently as possible in order to make optimum the use of logic.

4- Using a multiple of 6 for the slice number of columns **w** allows an optimum use of all the columns of the display when the **ROSETTA** is used in Text mode.

5- The number of rows in a slice **r** can assume any integer value, but if it is defined with values under 7, the table of patterns included in Appendix A could not be used. When this parameter is over 7, the "extra" rows will appear as blank.

6- The **TRB** (**T**emporary **R**ow **B**uffer) depth **n** should always be an integer with a minimum value equal to 1 and a maximum of **r** x **p**. In this last case the differences between the internal logic clock and the **WISHBONE** clock disappears and the clock domains become just one. Although, the amount of **FF**s needed to implement such logic grows up to one slice "extra".

7- There are no limitations in the **ROSETTA** for the number of wait states ($\mathbf{N_{WSS}}$) that can be added. The bus and system constrains are the only conditions to be taken in account to define this parameter.

Architecture

The code is composed by the following files:

a) **In SystemC:** sc_main.cpp, tru_scanner.cpp/h, scroller.cpp/h, wb_interface.cpp/h, registers.cpp/h, shifter.cpp/h, sector_decoder.cpp/h and row_decoder.cpp/h.

b) **In Verilog:** rosetta.v, tru_scanner.v, scroller.v, wb_interface.v, registers.v, shifter.v, sector_decoder.v and row_decoder.v

c) **In VHDL:** rosetta.vhd, tru_scanner.vhd, scroller.vhd, wb_interface.vhd, registers.vhd, shfter.vhd, sector_decoder.vhd and row_decoder.v

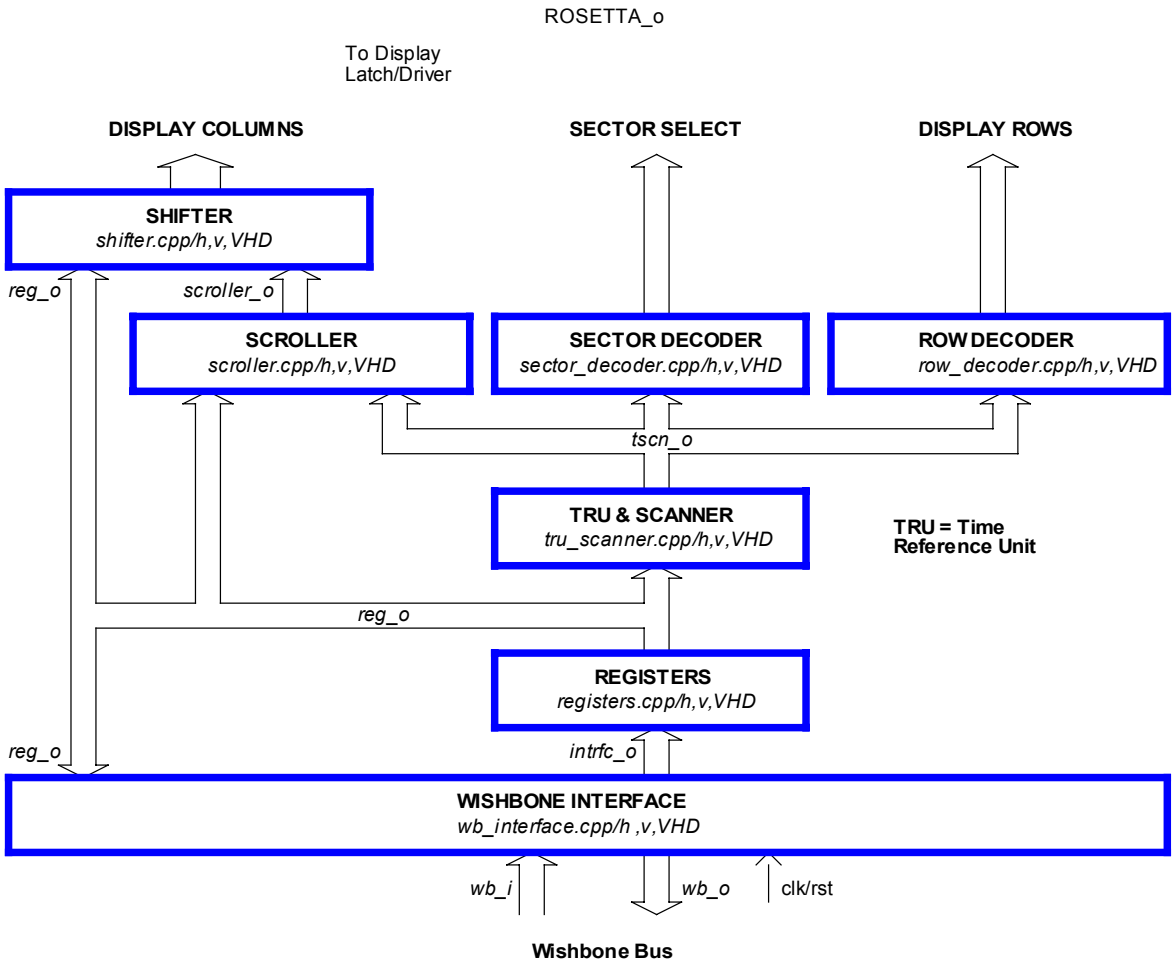

Figure 2.1

Registers

List of Registers

Name	Address	W	D	Access	Description
Buffer **R**egister (**BR**)	BASE	**wI**	**s***	W	**s*** BRSs **RAM** for display data storage
Active **P**age **R**egister (**APR**)	BASE + **1**	**D**$^{+}$	1	R/W	Contains the active page identification
Page **A**ddress **R**egister (**PAR**)	BASE + **2**	**D**$^{+}$	1	W	Register for **page** addressing in **BR** storage operations
Sector **A**ddress **R**egister (**SAR**)	BASE + **3**	**H**$^{\#}$	1	W	Register for **sector** addressing in **BR** storage operations
Mode **C**ontrol **R**egister (**MCR**)	BASE + **4**	8	1	R/W	Reset, mode, and scroll control
Color **C**ontrol **R**egister (**CCR**)	BASE + **5**	**C**$^{\wedge}$	1	R/W	**BR** and Slice clear, and Text mode color control
Configuration **R**ead **O**nly **R**egister (**CROR**)	-------	2	1	-------	Implements configuration details

Table 3.1: List of registers

******BR Size:* "**s**" = **dh** [**BRS**s] or **dhrp** [**wI** bit **words**]

$^{+}$***APR*** *&* ***PAR Size:*** "**D**" = $\log_2$(**d**)

$^{\#}$***SAR Size:*** "**H**" = $\log_2$(**h**)

^*CCR* ***Size*****: "C" = p**+1

Buffer Register

BR Register Structure

The **B**uffer **R**egister (**BR**) is composed of **s** slices ("**s**" = **dh** [**BRS**s]) named **BRS**s (**B**uffer **R**egister **S**lices). **BR** is sized **d** pages depth with **h** slices height. Each slice is **wI** width. Only one page is shown in the display at a time, the other pages are hidden and needs the use of scroll or animation modes to be shown.

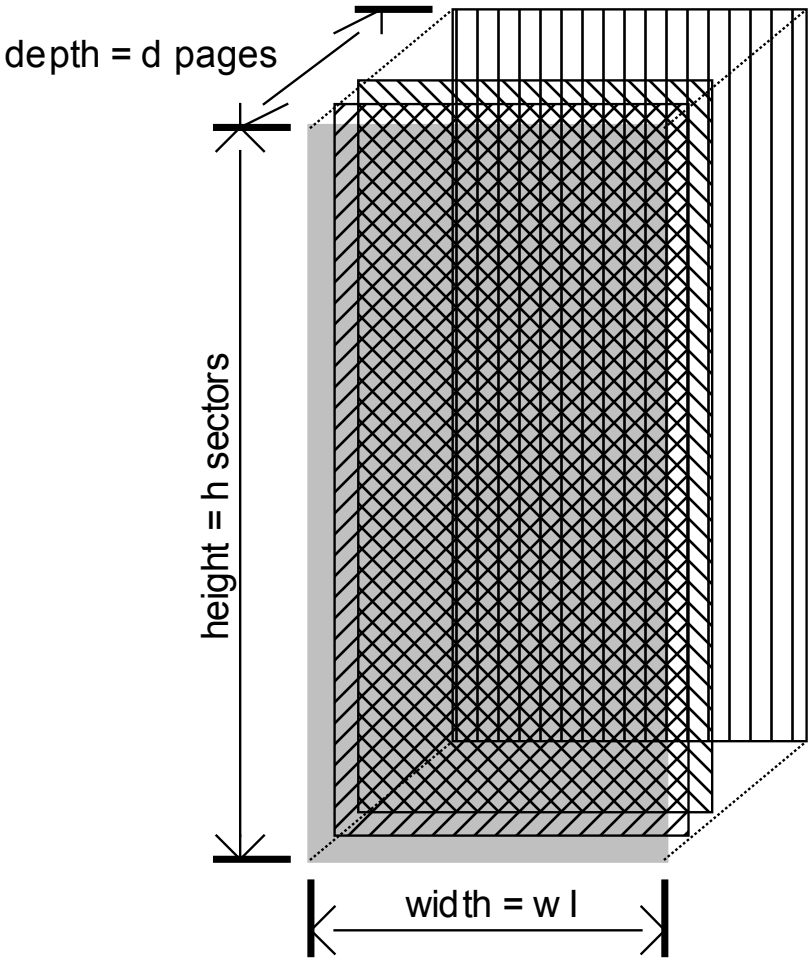

Figure 3.1

BRS Internal Structure

In the physical display any **sector** maps to a logical cell named **B**uffer **R**egister **S**lice (**BRS**). Each **BRS** is a **p** elements vector where **p** is the number of primary colors used for any pixel (**p** = 1,2 or 3).

$$BRS = \begin{bmatrix} X0 \\ \vdots \\ Xp\text{-}1 \end{bmatrix} \quad for\ p = 1,\ 2\ or\ 3$$

Each element "**X**" is an array with **r** words height, **wI** bits long each. Here "**I**" is the base two logarithm of the number of intensities selected for each primary color (**$I = \log_2(i)$**).

ROW		X STRUCTURE w COLUMNS			
0	r ROWS	pixel(w-1)0[I-1:0]	----------	pixel10[I-1:0]	pixel00[I-1:0]
1		pixel(w-1)1[I-1:0]	----------	pixel11[I-1:0]	pixel01[I-1:0]
2		pixel(w-1)2[I-1:0]	----------	pixel12[I-1:0]	pixel02[I-1:0]
3		pixel(w-1)3[I-1:0]	----------	pixel13[I-1:0]	pixel03[I-1:0]
4		pixel(w-1)4[I-1:0]	----------	pixel14[I-1:0]	pixel04[I-1:0]
5		pixel(w-1)5[I-1:0]	----------	pixel15[I-1:0]	pixel05[I-1:0]
-----		----------	----------	----------	----------
r		pixel(w-1)(r-1)[I-1:0]	----------	pixel1(r-1)[I-1:0]	pixel0(r-1)[I-1:0]

Table 3.1

Each word represents one **slice** single primary color **LED**s row. Reading X from top to bottom, the first one represents the upper row while the last one represents the lower row. In Text modes, the slice is filled from right to left with the 5 x 7 patterns shown on Apendix A. In this case, the color is defined by the color field (bits **p** downto 1) in the **C**olor **C**ontrol **R**egister (**CCR**). In Graphic modes, the color of each pixel in the **slice** is mapped by groups of **I** bits in the correspondent word.

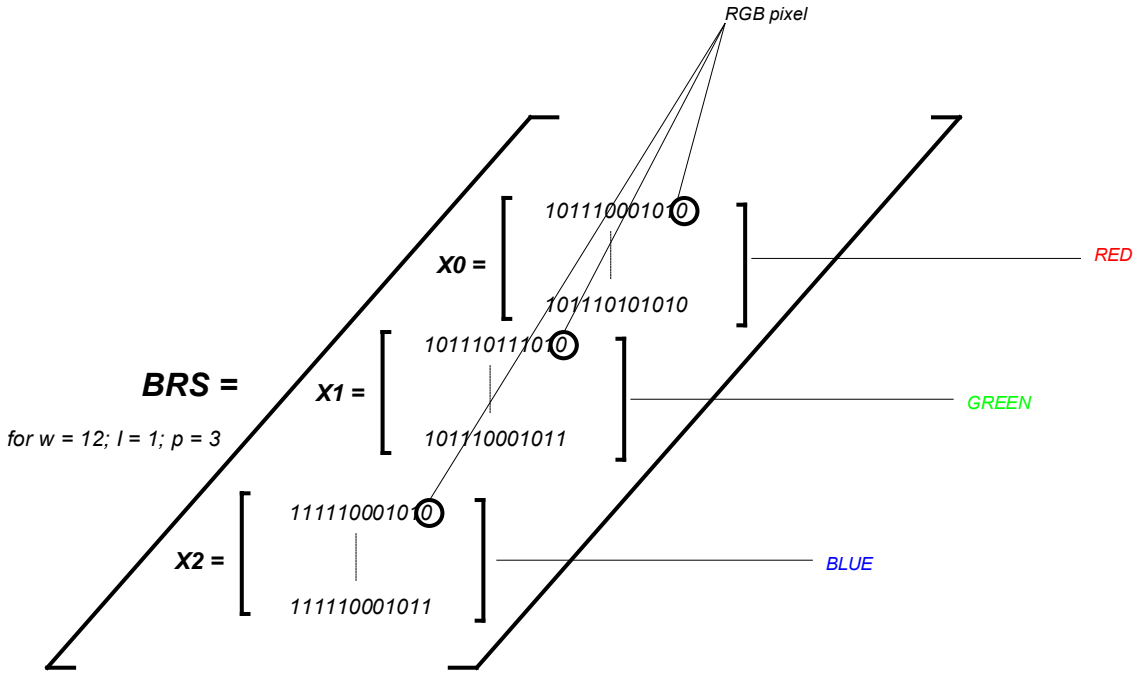

Figure 3.2

Description

Each **BRS** cell is addressed by writing its correspondent address on the **PAR** and **SAR** registers. The first one, to select the page and the second for the sector chosen.

Any address from the **BR** address map point to the beginning of each **BRS** cell (X0, row0). The access to any **BRS** internal position is sequential instead of random. That means that, using graphic mode, **w** = 32, **p** = 3, **I** = 4, **r** = 8 and data bus width = 32 bits, we need twelve successive accesses to the same address in order to write all the color components of a single **BRS** word. In the same case, we will also need 96 operations (to the same address) in order to access a complete **BRS**.

Each internal **wI** bits length word is accessed from the less significant to the more significant byte, and each **X** internal structure is accessed sequentially from its internal addresses 0 to **r**-1, which maps from the upper to the lower row direction in a **sector**. The **X** elements are accessed from 0 to (**p**-1) (first red, then green, then blue).

In the same case, if we are using the text mode and the **ROSETTA** input data bus width is 8 bits, we will need only five access operations to the same address, in order to fill one **BRS** cell.

WARNINGS:

- It is important to set up the operation mode through the **MCR** register before beginning any access operation to the **BR** register.
- When mixed operation modes (combined text and graphics) were used, special attention must be paid to the fact that the number of sequential accesses to the **BRS**s

reserved for text storage must be different to those carried out over **BRS**s reserved for graphic pattern storage.

Active Page Register – Description

Bit #	Access	Description
(**D**-1)-0	R/W	Defines: - In Fix mode the page to be shown - In Scroll modes the last page to be scanned

Reset Value:

Mode Control Register: 00h

Page Address Register – Description

Bit #	Access	Description
(**D**-1)-0	W	Defines the address of the current page being stored

Reset Value:

Mode Control Register: 00h

Sector Address Register – Description

Bit #	Access	Description
(**H**-1)-0	W	Defines the address of the current sector being stored

Reset Value:

Mode Control Register: 00h

Mode Control Register – Description

Bit #	Access	Description
0	R/W	Current slice Pointer Reset
3-1	R/W	Defines the operation mode '000' – Test '001' – Full Text '010' – Full Graphics '011' – Mixed Text & Graphics Type I '100' – Mixed Text & Graphics Type II '101' – Mixed Text & Graphics Type III '110' – Mixed Text & Graphics Type IV '111' – Animation
5-4	R/W	Defines the scroll mode '00' – Fix '01' – Horizontal Scroll '10' – Vertical Scroll '11' – Combined Scroll
6	R/W	Defines the Horizontal Scroll direction '0' – Left '1' – Right
7	R/W	Defines the Vertical Scroll direction '0' – Up '1' – Down

Reset Value:

Mode Control Register: 00h

Color Control Register – Description

Bit #	Access	Description			
0	R/W	BR Contents Clear			
p downto 1	R/W	Defines the text mode color			
			p = 1	**p** = 2	**p** = 3
		‘000’	Default color	Black	Black
		‘001’	Default color	Color 1	Red
		‘010’	Default color	Color 2	Green
		‘011’	Default color	Color1-Color2	Red-Green
		‘100’	Default color	Black	Blue
		‘101’	Default color	Color 1	Blue-Red
		‘110’	Default color	Color 2	Blue-Green
		‘111’	Default color	Color1-Color2	Red-Green-Blue

Reset Value:

Color Control Register: 00h

Configuration Read Only Register - Description

Bit #	Description
0	Display Arrangement ‘0’ – Linear array ‘1’ – Matrix array
1	Display completeness ‘0’ – Complete implementation ‘1’ – Incomplete implementation

Operation

This core can be used in eight different modes: Test, Full Text, Full Graphics, four Mixed Text and Graphics modes (**T&G I to IV**) and Animation.

Test mode turns on all the LEDs in the display with the color chosen in the color field of de **CCR**. This operation, do not depend on the contents of the **BR** register. This mode is useful for operational test of all the LEDs in the display.

In **Full Text mode**, each byte from the input data bus will be interpreted as can be seen in the table shown in Appendix A. That is: all bytes from '0' to '127' are considered as plain ASCII code, while those from 128 to 255 are seen as the custom codes shown in that table. Those bytes are grouped on a **T**emporary **R**ow **B**uffer register (**TRB**) to complete a slice, converted from ASCII code to their correspondent graphics patterns, and finally stored on the **BR**.

In the complete implementation mode a page is the maximum number of characters that can be seen in the implemented display without scrolling.

When this core is operated in text mode, a default color should be configured through the color field (bits **p** downto 1) of the **C**olor **C**ontrol **R**egister (**CCR**).

Full Graphic mode maps every composed **BRS** word to a **sector** row. Every group of **I** bits in each composed **BRS** word is mapped to one **sector** pixel. X0 bits maps to red LEDs, X1 bits maps to green LEDs and X2 bits maps to blue LEDs as can be seen in figure 4.1.

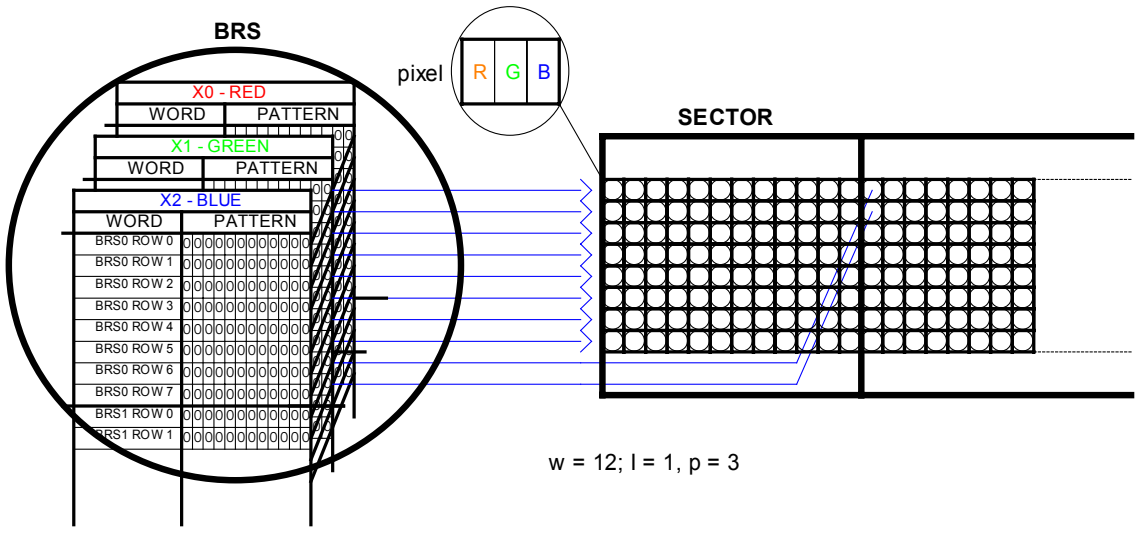

Figure 4.1. BRS mapping

Color in Graphic mode is determined by the group of bits defining a pixel in the **BRS** as:

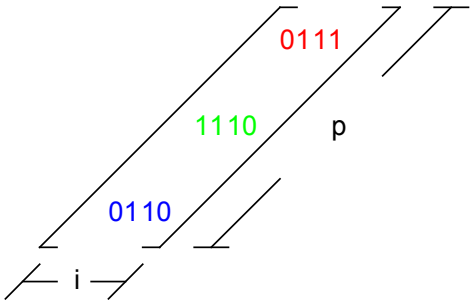

Figure 4.2

Mixed Text & Graphic mode has four different types. **Type I** interleaves sectors in text mode with sectors in graphic mode. **Type II** interleaves ¼ pages in each mode. **Type III** interleaves ½ pages and **Type IV** interleaves full pages in both modes.

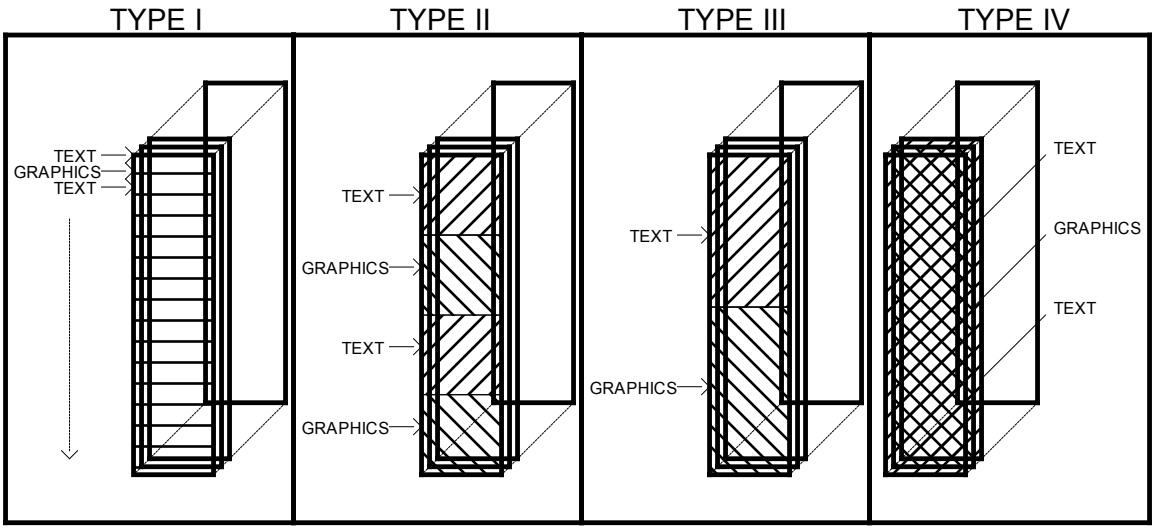

Figure 4.3. Mixed Text & Graphics Mode Types

Animation mode displays each page in sequence, from the first to the last one, repeating the cycle until the configuration is changed.

This core allows several **scrolling** types for the displayed information, it be text, graphics or mixed.

Fix mode should be used only when the string or graphic to be displayed fits into the display size chosen for the application selected. This mode maps the text/graphic data stored in the **BR** lowest address to the up/left corner of the display.

In applications where the amount of information to be shown is greater than the size of the display, one of the scrolling types should be selected. **Horizontal scroll** moves the display contents from right to left, while **vertical scroll** do that from bottom to top.

Combined scroll produced the effect of movement from the down/right corner towards the up/left corner.

The scrolling range can be limited in order to reduce the **BR** empty area exhibition through the set up of the **A**ctive **P**age **R**egister (**APR**). This range is defined in terms of the number of pages to be scrolled. The page shape and size depends on the display implementation shape and size.

Clocks

Name	Source	Rates (MHz)			Remarks	Description
		Max	Min	Resolution		
wb_clk_i	Syscon	-	-	-	Duty cycle 50/50	Wishbone clock.
clk	System clock	-	-	-	Duty cycle 50/50	Internal logic clock

Table 1: List of clocks

IO Ports

Wishbone interface signals

Port	Width	Dir	Description
wb_rst_i	1	Input	Block's WISHBONE Asynchronous Reset Input
wb_clk_i	1	Input	Block's WISHBONE Clock Input
wb_stb_i	1	Input	Block's WISHBONE Strobe Input
wb_we_i	1	Input	Block's WISHBONE Write Enable Input
wb_addr_i	3	Input	Block's WISHBONE Address Bus
wb_dat_i	8, 16, 32 or 64	Input	Block's WISHBONE Input Data Bus
wb_ack_o	1	Output	Block's WISHBONE Ack Output
wb_dat_o	8, 16, 32 or 64	Output	Block's WISHBONE Output Data Bus

Table 2: List of controller Wishbone IO ports

External (off chip) connections

Port	Width	Dir	Description
ROSETTA_o.sector_sel	h	Output	Columns Latch/Drivers enable
ROSETTA_o.dsply_col	pwI	Output	Columns Data
ROSETTA_o.row_sel	r	Output	Rows enable

Table 3: List of controller off chip IO ports

Appendix A

Correspondence between character codes and character patterns