SpartanMC
Display Controller

Table of Contents

List of Figures

List of Tables

Display Controller

The display controller is a periphal SpartanMC device for driving several types of displays. It is possible to control either a segment based or a pixel based display. For resource optimization both parts are seperated into independent controller modules selectable from the jConfig device menu. For a segment based LCD, a special circuit is required for connecting the SpartanMC FPGA to the device (see later). The pixel based display requires a circuit including elements for controlling the backlight and contrast voltage. Below, both controller parts are described in detail, starting with the segment dispaly controller.

1. Controller for segment based displays

This module makes it possible to control a segment based display with a user defined number of digits and segments. The required memory for storing the digit's segment assignments is already included (its content depends on the defined settings). From those segment assigments the signals for the display's multiplex driving are generated. In case the display is a liquid crystal display (LCD), the differing signal sequence for driving LCDs is generated accordingly. This requires a corresponding circuit for connecting the display as shown in the figure below. There, the micro controller's output is connected in the center of a voltage divider with two equal resistors. The divider itself is connected to operating voltage and ground.
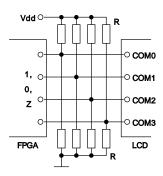

Figure 1: Circuit for connecting the LCD

If the display is not connected by using the given circuit, it will be damaged permanently! Because the dc voltage is not excluded.

1.1. Periphal registers

Table 1: Configuration registers of the segment display controller

Offset	Name	Description	Access	Initialization
0	REG_ENABLE	De-/Activates the display	read/ write	0x00000
1	REG_ADDR	Contains the adress of the digit to read from or write to.	read/ write	0x00000
2	REG_DATA	Contains the segment assignments according to the given digit address	read/ write	0x00000

Table 1: Configuration registers of the segment display controller

For accessing the segment memory the adress register must first be set to the correct digit number. Afterwards its current content can be read from or written to the data register.

1.2. Memory layout

The memory layout is slightly unconventional. This is caused by the required flexibility for configuring the number of segments/commons. Hence the data word is divided into the number of parts the display has commons. This allows the controller to compute the segment data sequentially for each common cycle instead of picking the required segment information out of the hole data word (which is quite complex to realize dynamically in hardware). The exact segment order depends on the used display. The first part of the data word (starting with bit 0) contains all assignments for the segments to be driven at common cycle 0, the next part for common cycle 1 and so on. For a 14 segment display with 4 commons the data word would look like "mPnd lcke gbjf haiS", where each letter represents one segment according to the typical segment order of such a display.

1.3. Module parameters

Table 2: Parameters of the segment display controller

Parameter	Description
SYSTEM_FREQUENCY	Clock the system is currently driven by (for example 25 MHz)

Parameter	Description
SEGMENT_FREQUENCY	Specifies the frequency for driving a single segment (see display's specification)
NUMBER_OF_COMMONS	Number of the display's common connections (number of anodes for LED displays)
NUMBER_OF_SEGMENTS	Number of segment connection for the whole display
NUMBER_OF_DIGITS	Number of the display's equal digits
BIT_PER_DIGIT	Width of a single memory word inside the segment memory (usually equal the the number of segments per digit)
IS_LCD	Set to 1 for a LCD, 0 for a LED display

Table 2: Parameters of the segment display controller

2. Controller for pixel based displays

This part of the display controller allows the driving of almost any pixel based displays. Therefore it provides the required memories. Depending on the module's configuration the operation of a graphic and a text mode is possible whereas both modes run independly. Inside the text mode a blinking cursor is displayed whose appearance can be defined by the user. The required codepage for converting the character codes to according pixel data can also be changed by the user. By default the codepage is initialized during the synthesis of the design with the "codepage 437", known from the original IBM PC. This initialization is defined in the given user constraint file (UCF). Inside the graphic mode there are several hardware accelerated functions for accessing the video memory (e.g. SetPixel or Line).

2.1. Periphal registers

Table 3: Configuration register of the matrix display controller

Offset	Name	Description	Access	Initialization
0	REG_DISPLAY_STATUS	Status register of the whole controller (see below)	read/ write	0x00100
1	REG_TEXT_COLOR	Foreground color of the displayed text. This color must come from the display's color space.	read/ write	0x0000F (white)
2	REG_TEXT_BGCOLOR	Background color of the displayed text	read/ write	0x00000

Offset	Name	Description	Access	Initialization
3	REG_TEXT_CHARPOS	Contains the coordinates of the currently drawed character. This may be important in relation to the CharLine interrupt described later.	read	0x00000
4	REG_TEXT_CURSORPOS	Position of the blinking cursor in text mode	read/ write	0x00000
5	REG_GRAPH_COORDSELECT	Register for addressing a single coordinate for the graphic functions	read/ write	0x00000
6	REG_GRAPH_COORDVALUE	Value of the selected coordinate	read/ write	0x00000
7	REG_GRAPH_COLOR	Color for a graphic function. This is a color coming from the color space of the memory (color LUT)	read/ write	0x00000

Table 3: Configuration register of the matrix display controller

2.2. Assembly of the register REG_DISPLAYSTATUS

Table 4: Register REG_DISPLAYSTATUS

Bit	Name	Description	Access	Initialization
0	STATUSBIT_DISP_BACKLIGHT	Turns the backlight on or off.	read/ write	0
1	STATUSBIT_DISP_ON	De-/activates the display.	read/ write	0
2	STATUSBIT_TEXTMODE	De-/activates the text mode, if implemented.	read/ write	0
3-5	STATUSBIT_FUNCTION_SELECT	Selects a hardware accelerated graphic function.	read/ write	000
6	STATUSBIT_FUNCTION_FLAG1	Optional flag for the graphic functions	read/ write	0
7	STATUSBIT_FUNCTION_DRAW	Starts the selected graphic function with the given parameters. After	read/ write	0

Bit	Name	Description	Access	Initialization
		setting the bit, it will be reset in the next clock cycle.		
8	STATUSBIT_FUNCTION_READY	Indicated if the currently selected graphic function is ready.	read	1
9	STATUSBIT_OVERLAY	De/activates the overlay	read/ write	0

Table 4: Register REG_DISPLAYSTATUS

2.3. Assembly of REG_TEXT_CHARPOS and REG_TEXT_CURSORPOS

Table 5: Registers REG_TEXT_CHARPOS and REG_TEXT_CURSORPOR

Bit	Name	Description
0 to 8	Y	Y coordinate in the text mode
9 to 17	X	X coordinate in the text mode

Table 5: Registers REG_TEXT_CHARPOS and REG_TEXT_CURSORPOR

2.4. Interrupts

Table 6: Interrupts of the matrix display controller

Interrupt	Description
charLine_ir	Indicates that the drawing of one charcter's pixel line in text mode has completed. This makes it possible to change the color settings for the following character lines for example.
graphFunctionReady_ir	This interrupt is triggered when a graphic function gets ready.

Table 6: Interrupts of the matrix display controller

2.5. Coding of the graphic functions

The following coding is defined in the file "display_graph_common.v" and should be changed there if required.

Number	Function	Macro name
0	invalid	-
1	SetPixel	FUNCTION_SETPIXEL
2	Line	FUNCTION_LINE
3	CopyRect	FUNCTION_COPYRECT
4	GetPixel	FUNCTION_GETPIXEL
5	FillRect	FUNCTION_FILLRECT

Table 7: Implemented graphic functions

2.6. Memory layouts

Codepage The codepage is a continous memory containing the pixel data of every displayable character. One memory word contains a single pixel line of a character. Thus one character requires a certain number of memory words depending on the configuration (by default 16). The memory offset O of one character is calculated by O = C * H, where C is the character code an H the configured number of lines per character.

Text cursor The layout of the cursor memory is equal to the one of the codepage but contains space for only one character.

Graphic memory Depending on the configuration one word of the graphic memory contains data for several pixels. Therein the MSB represents the most left and the LSB the most right pixel (almost like little endian). Usually the user has no need to access the graphic memory directly since there are functions like SetPixel and GetPixel.

Color LUT The color look up table (color LUT) converts the reduced color space inside the graphic memory to the display's color space. Thus the offset inside the color LUT represents a color of the graphic memory. From this offset the color LUT offers the corresponding color for the display.

2.7. Module parameters

Table 8: Parameters of the matrix display controller

Name	Description
BASE_ADDR	Base address on which the controller communicates with the SpartanMC
GRAPHMEM_BASE_ADDR	Base address on which the graphic memory is connected. The memory should be placed behind the space for I/O devices in any case. This is caused through its size, where otherwise some I/O device may be activated accidentally.

Name	Description
TEXTMEM_BASE_ADDR	Base address of the text memory
CODEPAGE_BASE_ADDR	Base address of the codepage
COLOR_LUT_BASE_ADDR	Base address of the color LUT
CURSORMEM_BASE_ADDR	Base address of the text cursor's memory
SCREEN_WIDTH	Display's width in pixels
SCREEN_HEIGHT	Display's height in pixels
DATA_WORD_WIDTH	Width of one data word transmitted to the display
BIT_PER_PIXEL	Color depth of one pixel on the display
BIT_PER_MEMORY_PIXEL	Color depth of one pixel in the graphic memory
TIMING_INIT_CLOCKS_TO_VCON_ON	Number of clocks to wait after a reset until the display's contrast voltage is activated.
TIMING_INIT_CLOCKS_TO_DISPOFF	Number of clocks to wait after a reset until the display itself is activated.
TIMING_CL1_CLOCKS_AFTER_RESET	Number of clocks to wait after a reset until a row cycle begins.
TIMING_CL1_CLOCKS_HIGH	Number of clocks the row clock is high.
TIMING_CL1_CLOCKS_LOW	Number of clocks the row clock is low.
TIMING_CL1_BEGIN_HIGH_LOW	Indicates whether a row on the display starts with the falling (1) or the rising edge (0) of the row clock.
TIMING_ROW_BREAK_DELAY	Number of clocks to wait after transmitting one row before the transmission of the next row starts.
TIMING_SCREEN_FINISH_DELAY	Number of clocks to wait after transmitting one complete screen before the transmission of the next screen continues.
TIMING_CL2_CLOCKS_AFTER_RESET	Number of clocks to wait after a reset until a data clock cycle begins.
TIMING_CL2_CLOCKS_HIGH	Number of clocks the data clock is high.
TIMING_CL2_CLOCKS_LOW	Number of clocks the data clock is low.
TIMING_FRAMESTART_CLOCKS_AFTER_RESET	Number of clocks to wait after a reset until the framestart cycle begins.
TIMING_FRAMESTART_CLOCKS_HIGH	Number of clocks the framestart signal is high.
TIMING_FRAMESTART_CLOCKS_LOW	Number of clocks the framestart signal is low.
TIMING_INVERT_CLOCKS_AFTER_RESET	Number of clocks to wait after a reset until the invert signal cycle begins.
TIMING_INVERT_CLOCKS_TOGGLE	Number of clocks after the invertsignal is inverted.
CODEPAGE_CHAR_WIDTH	Width of one character in pixles
CODEPAGE_CHAR_HEIGHT	Height of one character in pixels

Name	Description
CODEPAGE_SIZE	Number of characters in the codepage
FPGA_BRAM_SIZE	Number of words the used Block RAM can save
USE_TEXT_MODE	Defines whether the text mode is included in synthesis.
USE_GRAPH_MODE	Defines whether the graphic mode is included in synthesis.
USE_GRAPHFUNCTION_SETPIXEL	Defines whether the graphic function "SetPixel" is included in synthesis.
USE_GRAPHFUNCTION_LINE	Defines whether the graphic function "Line" is included in synthesis.
USE_GRAPHFUNCTION_COPYRECT	Defines whether the graphic function "CopyRect" is included in synthesis.
USE_GRAPHFUNCTION_GETPIXEL	Defines whether the graphic function "GetPixel" is included in synthesis.
USE_GRAPHFUNCTION_FILLRECT	Defines whether the graphic function "FillRect" is included in synthesis.

Table 8: Parameters of the matrix display controller