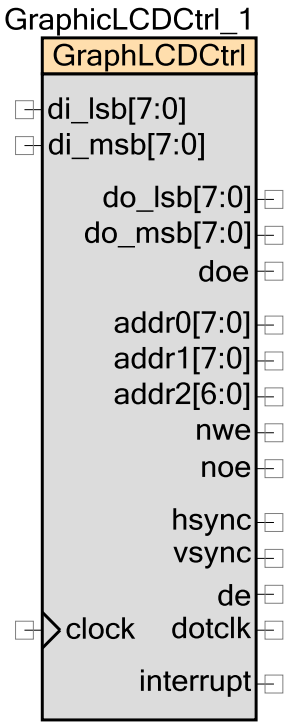
Graphic LCD Controller (GraphicLCDCtrl)

1.50



Features

- Fully programmable screen size support up to HVGA resolution including:
  - QVGA (320x240) @ 60 Hz 16 bpp
  - WQVGA (480x272) @ 60 Hz 16 bpp
  - HVGA (480x320) @ 60 Hz 16 bpp
- Virtual screen operation support
- Read and write transactions during the blanking intervals
- Generation of continuous timing signals to the panel without CPU intervention
- Supports up to a 23-bit address and a 16-bit data async SRAM device used as externally provided frame buffer
- Selectable interrupt pulse generated at the entry and exit of the horizontal and vertical blanking intervals

General Description

The Graphic LCD Controller (GraphicLCDCtrl) component provides the interface to an LCD panel that has an LCD driver, but not an LCD controller. This type of panel does not include a frame buffer. The frame buffer must be provided externally.

This component also interfaces to an externally provided frame buffer implemented using a 16-bit wide async SRAM device.

When to use a GraphicLCDCtrl

The GraphicLCDCtrl component supports many LCD panels. It directly drives the control signals and manages the frame buffer in an external SRAM. The component accesses data from the SRAM and displays it on the LCD through the control of the dotclk, hsync, vsync, and de outputs.

The frame buffer SRAM can only be accessed for reads and writes when it is not in use to refresh the LCD panel. If a read or write is requested during the refresh period, the APIs

provided will wait until the refresh gets to a blanking period. During the blanking period, the read or write will be completed.

An interrupt can be used to indicate the entry and exit from blanking periods. This is particularly useful when coupled with an RTOS, which can swap in or swap out tasks that require access to the frame buffer when a blanking period is entered and exited.

Input/Output Connections

This section describes the various input and output connections for the GraphicLCDCtrl. Some I/Os may be hidden on the symbol under the conditions listed in the description of that I/O.

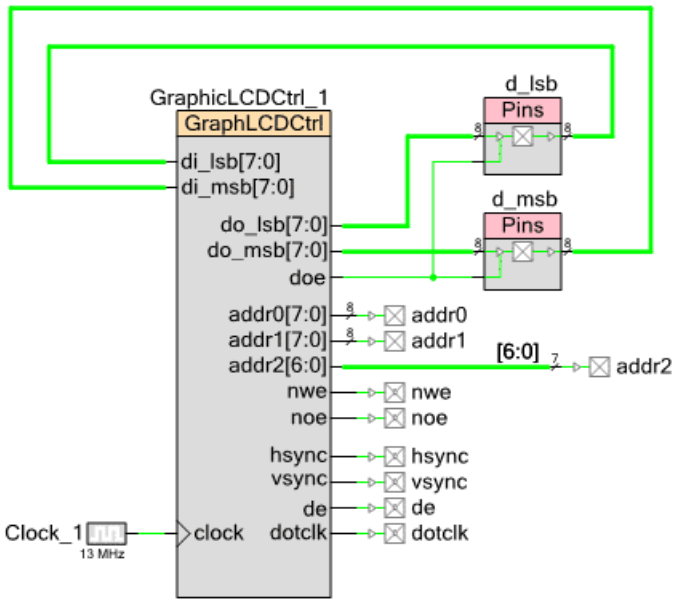
Input	May Be Hidden	Description
di_lsb[7:0]	N	Lower 8 bits of the input data bus. Used for data during a read transaction. These signals should be connected to an input pin on the device and the "Input Synchronized" selection for these pins should be disabled. The signals themselves are inherently synchronized already since they are being driven based on synchronous output signals.
di_msb[7:0]	N	Upper 8 bits of the input data bus. Used for data during a read transaction. Only present for 16-bit interface mode. These signals should be connected to an input pin on the device and the "Input Synchronized" selection for these pins should be disabled. The signals themselves are inherently synchronized already since they are being driven based on synchronous output signals.
clock	N	Clock that operates this component. It is twice the frequency of the dotclk.

Output	May Be Hidden	Description
do_lsb[7:0]	N	Lower 8 bits of the output data bus. Used for data during a write transaction.
do_msb[7:0]	N	Upper 8 bits of the output data bus. Used for data during a write transaction.
doe	N	Output enable for the data bus component within PSoC. Normally connected to the output enable of the Input/Output pin component for the data buses.
addr0[7:0]	N	Lowest 8 bits of the address bus connected to the frame buffer.
addr1[7:0]	N	Middle 8 bits of the address bus connected to the frame buffer.

Output	May Be Hidden	Description
addr2[6:0]	N	Upper 7 bits of the address bus connected to the frame buffer. The number of data bits needed by the frame buffer is dependent on the SRAM device used.
nwe	N	Active low write enable for the frame buffer SRAM.
noe	N	Active low output enable for the frame buffer.
de	N	Data enable for the panel.
hsync	N	Horizontal sync timing signal for the panel.
vsync	N	Vertical sync timing signal for the panel.
dotclk	N	Clock driven to the panel. This clock is ½ the rate of the incoming clock.
interrupt	Y	Edge triggered interrupt signal. Hidden if no interrupt generation selected.

Schematic Macro Information

The macro is configured with the default settings to interface with the Optrex T-55343GD035JU-LW-AEN panel used in the Cypress CY8CKIT-016 EBK. The clock included in the macro is set to 13 MHz, which will result in a 6.5 MHz dotclk provided to the Optrex QVGA panel.



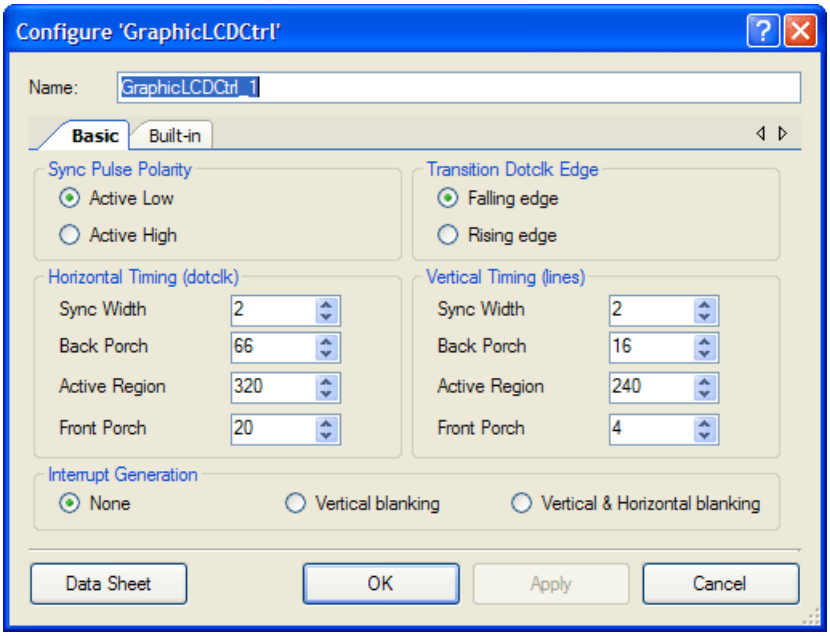
Note that typically only some of the bits of the upper address bits (addr2) will be used to connect to the frame buffer. This will depend on the size of the SRAM used. Based on the number of address bits needed, the following steps can be used to adjust the size of that bus.

- Configure the addr2 output pin component and set the "Number of Pins"
- Right click on the signal driving the output pin and select "Edit Name And Width". Then adjust the "Left Index" to reflect the width of the output pin.

Also, the "Input Synchronized" option is unchecked on all of the data pins and the generation of APIs for all pins is turned off.

Parameters and Setup

Drag a GraphicLCDCtrl component onto your design and double click it to open the Configure dialog. The default GraphicLCDCtrl settings are configured for operation with the Optrex board and the CY8CKIT-016 EBK.



The GraphicLCDCtrl dialog contains the following parameters. All of these settings are compile-time selections and there is no need to change these settings at run time. They are all characteristics of the panel and frame buffer SRAM being used.

Sync Pulse Polarity

Based on this setting the hsync and vsync signals will either be Active High (pulse generated is a high pulse) or Active Low. Both hsync and vsync polarity are controlled by this single selection. The default setting is Active Low.

Transition Dotclk Edge

Dotclk is the clock that is sent to the panel, off of which the panel operates. When the transition is set to rising edge, all of the associated signals such as hsync, vsync, and de will transition on the rising edge of dotclk. When set to falling edge they will transition on the falling edge of dotclk.

Typically if the panel specifies a setup and hold time to one edge of dotclk, then you'll want to configure this setting to the other edge of the clock. This will provide approximately one half clock cycle of both setup and hold. The default setting is Falling edge.

Horizontal Timing (dotclk)

- **Sync Width –** Defines the horizontal Sync Width in dotclks. This value can be set between 1 and 256 clock cycles. The default setting is 2.
- **Back Porch –** Defines the horizontal Back Porch width in dotclks. This value can be set between 6 and 256 clock cycles. A minimum of 6 is used to give an early enough indication to the state machine to prevent a read or write access that could not complete before the active screen area began. Some panel specifications consider the back porch as the region between the end of the sync signal to the start of the active region. Other panel specifications consider the back porch to be the region from the start of the sync pulse to the start of the active region. This component measures the back porch as the period from the end of the sync pulse to the start of the active region. The default setting is 66.
- **Active Region –** Defines the horizontal Active Region width in dotclks. The Active Region is implemented using a setting that is a multiple of 4. This allows for regions as large as 1024 x 1024 while only using 8-bit counters. All popular screen sizes are multiples of 4 in both directions. This value can be set between 4 and 1024 (must be a multiple of 4) clock cycles. The default setting is 320.
- **Front Porch –** Defines the horizontal Front Porch width in dotclks. This value can be set between 1 and 256 clock cycles. The default setting is 20.

Vertical Timing (lines)

- **Sync Width –** Defines the vertical Sync Width in lines. This value can be set between 1 and 256 clock cycles. The default setting is 2.
- **Back Porch –** Defines the vertical Back Porch width in lines. This value can be set between 1 and 256 lines. Some panel specifications consider the back porch as the region between the end of the sync signal to the start of the active region. Other panel specifications consider the back porch to be the region from the start of the sync pulse to the start of the active region. This component measures the back porch as the period from the end of the sync pulse to the start of the active region. The default setting is 16.
- **Active Region –** Defines the vertical Active Region width in lines. The Active Region is implemented using a setting that is a multiple of 4. This allows for regions as large as 1024 x

1024 while only using 8-bit counters. All popular screen sizes are multiples of 4 in both directions. This value can be set between 4 and 1024 (must be a multiple of 4) lines. The default setting is 240.

- **Front Porch –** Defines the vertical Front Porch width in lines. This value can be set between 1 and 256 lines. The default setting is 4.

Interrupt Generation

Defines the settings for interrupt generation. The default setting is None.

- If the setting is for Vertical Blanking, then an interrupt pulse is generated at the start and end of the vertical blanking interval.
- If the setting is for Vertical and Horizontal Blanking, then an interrupt pulse is generated at the start and end of every active region.

  During the active vertical region, this will be an interrupt at the start and end of the active region for each line. For the vertical blanking region, this will be a single interrupt at the end of the last active line and another interrupt at the start of the first active line.

Clock Selection

There is no internal clock in this component. You must attach a clock source. The clock rate provided must be two times the desired clock rate for the output dotclk clock to the panel.

Placement

The GraphicLCDCtrl is placed throughout the UDB array and all placement information is provided to the API through the *cyfitter.h* file.

Resources

Digital Blocks					API Memory (Bytes)		Pins (per External I/O)
Datapaths	Macro cells	Status Registers	Control Registers	Counter 7	Flash	RAM	
7	35	1	1	0	TBD	TBD	45

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "GraphicLCDCtrl_1" to the first instance of a component in a given design. You can rename the instance to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "GraphicLCDCtrl".

Function	Description
void GraphicLCDCtrl_Init(void)	Initializes or restore the component parameters to the settings provided with the component customizer
void GraphicLCDCtrl_Enable(void)	Enables the GraphicLCDCtrl
void GraphicLCDCtrl_Start(void)	Starts the GraphicLCDCtrl interface.
void GraphicLCDCtrl_Stop(void)	Disables the GraphicLCDCtrl interface.
void GraphicLCDCtrl_Write(uint32 addr, uint16 data)	Initiates a write transaction to the frame buffer.
uint16 GraphicLCDCtrl_Read(uint32 addr)	Initiates a read transaction from the frame buffer.
void GraphicLCDCtrl_WriteFrameAddr(uint32 addr)	Sets the starting frame buffer address used when refreshing the screen.
uint32 GraphicLCDCtrl_ReadFrameAddr(void)	Reads the starting frame buffer address used when refreshing the screen.
void GraphicLCDCtrl_WriteLineIncr(uint32 incr)	Sets the address spacing between adjacent lines.
uint32 GraphicLCDCtr_ReadLineIncr(void)	Reads the address increment between lines.
void GraphicLCDCtrl_Sleep(void)	Saves configuration and disables the GraphicLCDCtrl
void GraphicLCDCtrl_WakeUp(void)	Restores configuration and enables the GraphicLCDCtrl
void GraphicLCDCtrl_SaveConfig(void)	Saves configuration of GraphicLCDCtrl
void GraphicLCDCtrl_RestoreConfig(void)	Restores configuration of GraphicLCDCtrl

Global Variables

Variable	Description
GraphicLCDCtrl_initVar	Indicates whether the GraphicLCDCtrl has been initialized. The variable is initialized to 0 and set to 1 the first time GraphicLCDCtrl_Start() is called. This allows the component to restart without reinitialization after the first call to the GraphicLCDCtrl_Start() routine. If reinitialization of the component is required, then the GraphicLCDCtrl_Init() function can be called before the GraphicLCDCtrl_Start() or GraphicsLCDCtrl_Enable() function.

void GraphicLCDCtrl_Init(void)

Description: This API initializes or restores the component parameters to the settings provided with the component customizer. The compile time configuration that defines timing generation is restored to the settings provided with the customizer. The run time configuration for the frame buffer address is set to 0; for the line increment it is set to the display line size.

Parameters: None

Return Value: None

Side Effects: The component must be disabled by GraphicLCDCtrl_Stop API before this API call, otherwise the component behavior can be unexpected. This will re-initialize the component with the following exceptions. It will not clear data from the FIFOs and will not reset component hardware state machines.

void GraphicLCDCtrl_Enable(void)

Description: Enables the GraphicLCDCtrl interface.

Parameters: None

Return Value: None

Side Effects: None

void GraphicLCDCtrl_Start(void)

Description: Enables Active mode power template bits or clock gating as appropriate. Configures the component for operation.

Parameters: None

Return Value: None

Side Effects: None

void GraphicLCDCtrl_Stop(void)

Description: Disables Active mode power template bits or gates clocks as appropriate.

Parameters: None

Return Value: None

Side Effects: None

void GraphicLCDCtrl_Write(uint32 addr, uint16 data)

Description: Initiates a write transaction to the frame buffer using the address and data provided. The write is a posted write, so this function will return before the write has actually completed on the interface. If the command queue is full, this function will not return until space is available to queue this write request

Parameters: addr: Address to be sent on the address lines of the component (addr2[6:0], addr1[7:0], addr0[7:0]).

data: Data sent on the do_msb[7:0] (most significant byte) and do_lsb[7:0] (least significant byte) pins

Return Value: None

Side Effects: None

uint16 GraphicLCDCtrl_Read(uint32 addr)

Description: Initiates a read transaction from the frame buffer. The read will execute after all currently posted writes have completed. This function will wait until the read completes and then returns the read value.

Parameters: addr: Address to be sent on the address lines of the component (addr2[6:0], addr1[7:0], addr0[7:0])

Return Value: Read value from the di_msb[7:0] (most significant byte) and di_lsb[7:0] (least significant byte) pins

Side Effects: None

void GraphicLCDCtrl_WriteFrameAddr(uint32 addr)

Description: Sets the starting frame buffer address used when refreshing the screen. This register is read during each vertical blanking interval. To implement an atomic update of this register it should be written during the active refresh region.

Parameters: addr: Address of the start of the frame buffer

Return Value: None

Side Effects: None

uint32 GraphicLCDCtrl_ReadFrameAddr(void)

Description: Reads the starting frame buffer address used when refreshing the screen.

Parameters: None

Return Value: Address of the start of the frame buffer

Side Effects: None

void GraphicLCDCtrl_WriteLineIncr(uint32 incr)

Description: Sets the address spacing between adjacent lines. By default this is the display size of a line. This setting can be used to align lines to a different word boundary or to implement a virtual line length that is larger than the display region.

Parameters: incr: Address increment between lines. Must be at least the display size of a line.

Return Value: None

Side Effects: None

uint32 GraphicLCDCtrl_ReadLineIncr(void)

Description: Reads the address increment between lines.

Parameters: None

Return Value: Address increment between lines

Side Effects: None

void GraphicLCDCtrl_Sleep(void)

Description: Stops the GraphicLCDCtrl operation and saves the user configuration. Should be called just prior to entering sleep.

Parameters: None

Return Value: None

Side Effects: None

void GraphicLCDCtrl_Wakeup(void)

Description: Restores and enables the user configuration. Should be called just after awaking from sleep.

Parameters: None

Return Value: None

Side Effects: None

void GraphicLCDCtrl_SaveConfig(void)

Description: Saves the current user configuration.

Parameters: None

Return Value: None

Side Effects: None

void GraphicLCDCtrl_RestoreConfig(void)

Description: Restores the current user configuration.

Parameters: None

Return Value: None

Side Effects: If this API will be called before GraphicLCDCtrl_SaveConfig the component configuration will be restored to their default settings. The run time configuration for the frame buffer address is set to 0; for the line increment it is set to the display line size.

Sample Firmware Source Code

See the example provided with the CY8CKIT-016 Graphic LCD Controller Kit. Besides initialization of the controller, this component is typically used exclusively by the Segger emWin Graphics component.

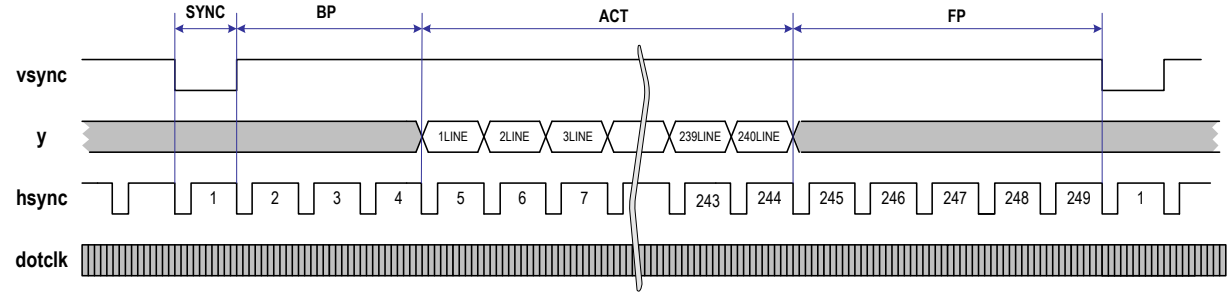
Functional Description

This component generates continuous timing signals to the panel without CPU intervention. During the refresh period, the component also generates read requests to the frame buffer scanning through a frame of 16-bit pixel data. During the blanking intervals (horizontal and vertical) the component can generate read or write transactions on the frame buffer interface.

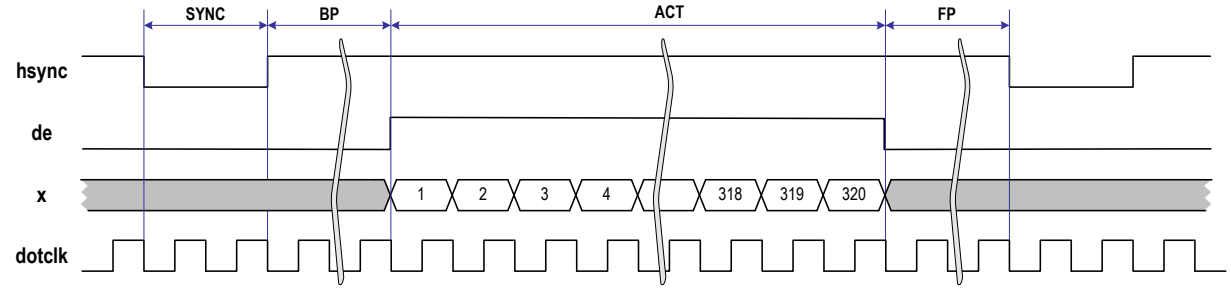
Screen Refresh and Timing

Throughout a frame time the component generates the configured vertical timing pattern on vsync and throughout each line of the frame the component generates the configured horizontal pattern on hsync. In addition to hsync and vsync, some panels require a de (data enable) signal that is active high during the active portion of the screen refresh. All panels operate in the same way, although the timing of each of the segments of the refresh period differs. The following is the timing diagram for a typical panel.

Vertical timing:



Horizontal timing:



The sequence for each frame for the vertical signal and the sequence for each line for the horizontal signal follow the following pattern:

- Sync pulse: Period where the sync pulse is active
- Back Porch: Period from the end of the sync pulse until the active display area
- Active: Display area on the screen
- Front Porch: Period from the end of the active display until the sync pulse starts

Address generation

As the screen is refreshed the component must scan through the frame buffer generating the addresses for the pixels on the screen. Each pixel requires one 16-bit read from the frame buffer. For the beginning of each line the index into the frame buffer is reset to the designated starting point for the frame buffer. The value is set to 0 initially and can then be modified using API functions. The frame buffer address does not impact the read and write API functions, it only impacts the refresh operation.

Frame Buffer Transactions

The controller component can perform either read or write transactions. These transactions have the following parameters:

- Read or write
- Address. Up to a 23 bit address
- Data. 16 bit value. Sent on "do" (data out) for writes and read on "di" (data in) for reads.

The implementation used for this component combines the 23 bits of address with a 1-bit read/write indicator. This allows the address and transaction type to be transferred to the component in 3 bytes. It also allows the transaction type and address to stay together in the datapath FIFO.

Read and write transactions are performed during the horizontal and vertical blanking intervals.

Idle Condition

When neither a read nor a write is occurring on the frame buffer interface the interface will be in the idle state. The idle state control signals are the same as the values for reading. The values for the output pins in the idle condition are as follows:

- do: don’t care (may be left at its last state)
- doe: 0
- addr: don’t care (may be left at its last state)
- nwe: 1
- noe: 0

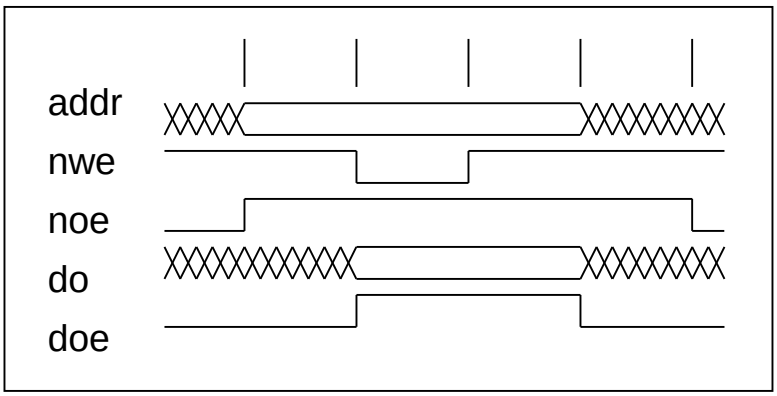
Any signal not listed n the description of the read and write transactions will be in the idle state.

Write Transaction

The component implements the timing diagram shown below for a write transaction. This diagram shows that the write transaction requires 4 dotclk cycles (all diagrams are in dotclk cycles). This transaction can be immediately proceeded or followed by another read or write transaction or may be in the idle state before or after a write transaction.

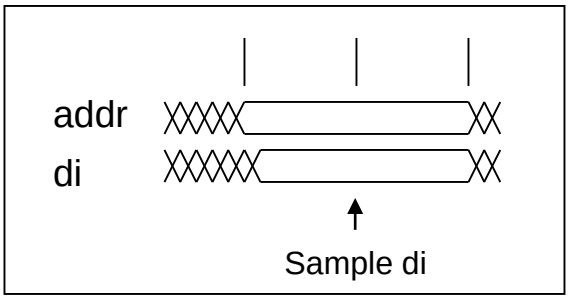
The interface to the CPU allows the CPU to make posted write requests (request a write providing the address and data and then proceed before the transaction is actually completed to the frame buffer). The implementation allows the CPU to have four write requests outstanding without stalling the CPU.

Note the pattern of the noe and doe signals which prevents the data bus from being driven by both the component and the frame buffer regardless of the skew of the signals.



Read Transaction

This component implements the timing diagram shown below for a read transaction. This transaction can be immediately proceeded or followed by another read or write transaction or may be in the idle state before or after a write transaction.



Block Diagram and Configuration

The GraphicLCDCtrl component is implemented as a set of configured UDBs. The implementation is shown in the following block diagram.

Error! Objects cannot be created from editing field codes.

Registers

GraphicLCDCtrl_STATUS_REG

Bits	7	6	5	4	3	2	1	0
Value	reserved				v_blanking	h_blanking	avail	full

- full: set if command/data FIFO is full
- avail: set if read data is valid for the CPU

- h_blanking: set during the horizontal blanking interval
- v_blanking: set during the vertical blanking interval

References

Not applicable

DC and AC Electrical Characteristics

5.0V/3.3V DC and AC Electrical Characteristics

Parameter	Typical	Min	Max	Units	Conditions and Notes
dotclk	---		---		
hsync	---		---		
vsync	---		---		

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.0.b	Minor datasheet edit.	
1.0.a	Minor datasheet edit.	
1.0	The first release of this component.	

© Cypress Semiconductor Corporation, 2010-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.