Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

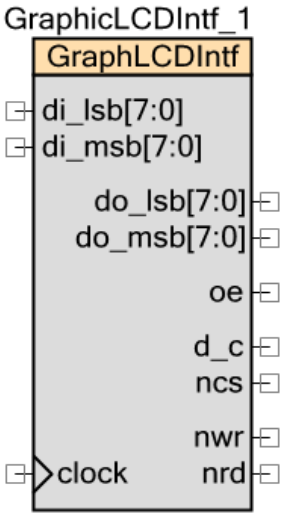
Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

Graphic LCD Interface (GraphicLCDIntf)

1.80

Features



- 8- or 16-bit interface to Graphic LCD Controller
- Compatible with many graphic controller devices
- Can be used with SEGGER emWin graphics library
- Performs read and write transactions
- 2 to 255 cycles for read low pulse width
- 1 to 255 cycles for read high pulse width
- Implements typical i8080 interface

General Description

The Graphic LCD Interface (GraphicLCDIntf) Component provides the interface to a graphic LCD controller and driver device. These devices are commonly integrated into an LCD panel. The interface to these devices is commonly referred to as an i8080 interface. This is a reference to the historic parallel bus interface protocol of the Intel 8080 microprocessor.

This Component is designed to work with the SEGGER emWin graphics library. This library provides a full-featured set of graphics functions for drawing and rendering text and images. The emWin graphics library is available for PSoC 3, PSoC 4, PSoC 5LP devices: www.cypress.com/go/comp_emWin

When to Use a GraphicLCDIntf

LCD controllers and driver devices are commonly integrated into an LCD panel. They either include or provide the interface to the frame buffer for the display and manage that buffer. The GraphicLCDIntf Component performs read and write transactions to this controller. These transactions have the following parameters:

- Read or write
- Address: A one-bit address driven on the d_c pin
- Data (8 or 16 bits): Sent on “do” for writes and read on “di” for reads

Settings for Different Controllers

The GraphicLCDIntf Component supports many controllers. Use the following parameters when you configure this Component.

- Clock frequency: The frequency for the clock driving this Component is often limited by minimum pulse width low for the write signal (this value can be found in the Graphic LCD Controller datasheet). The write pulse is low for a single clock period, so set the clock frequency to satisfy this requirement.
- Read pulse width high: This setting in the customizer is measured in clock cycles. The clock period times the number of cycles set for the pulse width high must satisfy the requirement for read pulse width high for the controller.
- Read pulse width low: This parameter is set in the same way as the read pulse width high parameter. The timing for the read pulse width low must satisfy the controller's requirement for the read pulse width and the requirement for read access time. The data is sampled one clock cycle before the end of the active low read pulse, so the pulse width must be long enough that the access time is satisfied.

The following lists the settings for the applicable LCD controller:

Solomon Systech SSD1289

- Clock frequency: 20 MHz (50 ns)
- Read pulse width high: 10 clock cycles (500 ns)
- Read pulse width low: 10 clock cycles (500 ns)

Solomon Systech SSD2119

- Clock frequency: 25 MHz (40 ns)
- Read pulse width high: 13 clock cycles (500 ns)
- Read pulse width low: 13 clock cycles (500 ns)

Himax HX8347A

- Clock frequency: 28.5 MHz (35 ns)
- Read pulse width high: 3 clock cycles (105 ns)
- Read pulse width low: 11 clock cycles (385 ns)

ILITEK ILI9325

- Clock frequency: 20 MHz (50 ns)
- Read pulse width high: 3 clock cycles (150 ns)
- Read pulse width low: 3 clock cycles (150 ns)

Epson S1D13743

- Clock frequency: 33 MHz (30 ns)
- Read pulse width high: 2 clock cycles (60 ns)
- Read pulse width low: 5 clock cycles (150 ns)

Sitronix ST7789S

- Clock frequency: 33 MHz (30ns)
- Read pulse width high: 3 clock cycles (90 ns)
- Read pulse width low: 12 clock cycles (360 ns)

Input/Output Connections

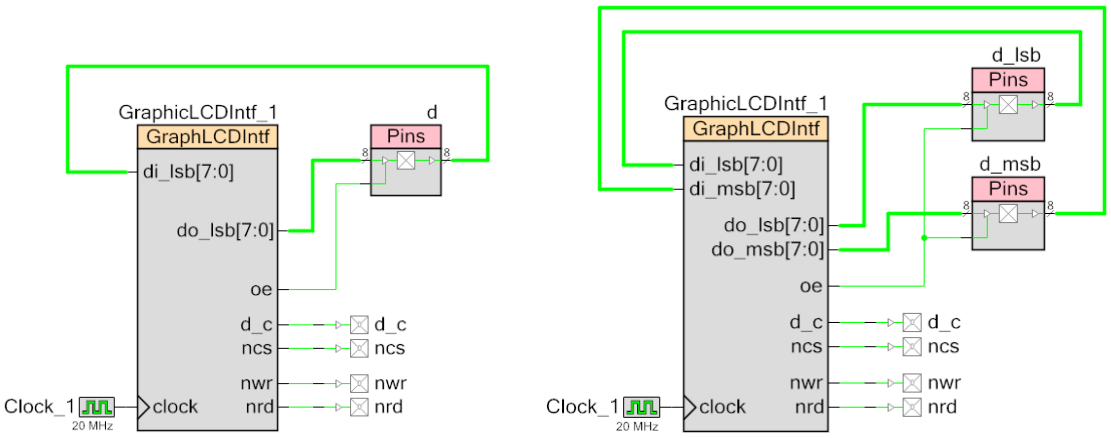
This section describes the input and output connections for the GraphicLCDIntf Component. An asterisk (*) in the list of I/Os indicates that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

I/O Name	Type	Description
di_lsb[7:0]	Input	The lower eight bits of the input data bus. They are used for data during a read transaction. Connect these to an input pin on the device and disable the “Input Synchronized” selection for this pin. The signals themselves are inherently synchronized because they are driven based on synchronous output signals.
di_msb[7:0] *	Input	The upper eight bits of the input data bus. They are used for data during a read transaction. They are only present for 16-bit interface mode. Connect these signals to an input pin on the device and disable the “Input Synchronized” selection for this pin. The signals themselves are inherently synchronized because they are driven based on synchronous output signals.
clock	Input	The clock that operates this Component. The GraphicLCDIntf operates entirely from a single clock connected to the Component.
do_lsb[7:0]	Output	The lower eight bits of the output data bus. They are used for data during a write transaction.
do_msb[7:0] *	Output	The upper eight bits of the output data bus. They are used for data during a write transaction. They are only present for 16-bit interface mode.

I/O Name	Type	Description
oe	Output	The output enable for the data bus. It is normally connected to the output enable of the Input/Output pin Component for the data buses. Refer to the Schematic Macro Information to see how this signal is used.
d_c	Output	Data/Command signal. This signal indicates a data transaction when high and a command transaction when low.
ncs	Output	Active-low chip select.
nwr	Output	Active-low write control signal.
nrd	Output	Active-low read control signal.

Schematic Macro Information

PSoC Creator supplies two macros in addition to the standard symbol entry in the Component catalog. One macro is for an 8-bit implementation connected to pins and a clock. The other is for a 16-bit implementation connected to pins and a clock.

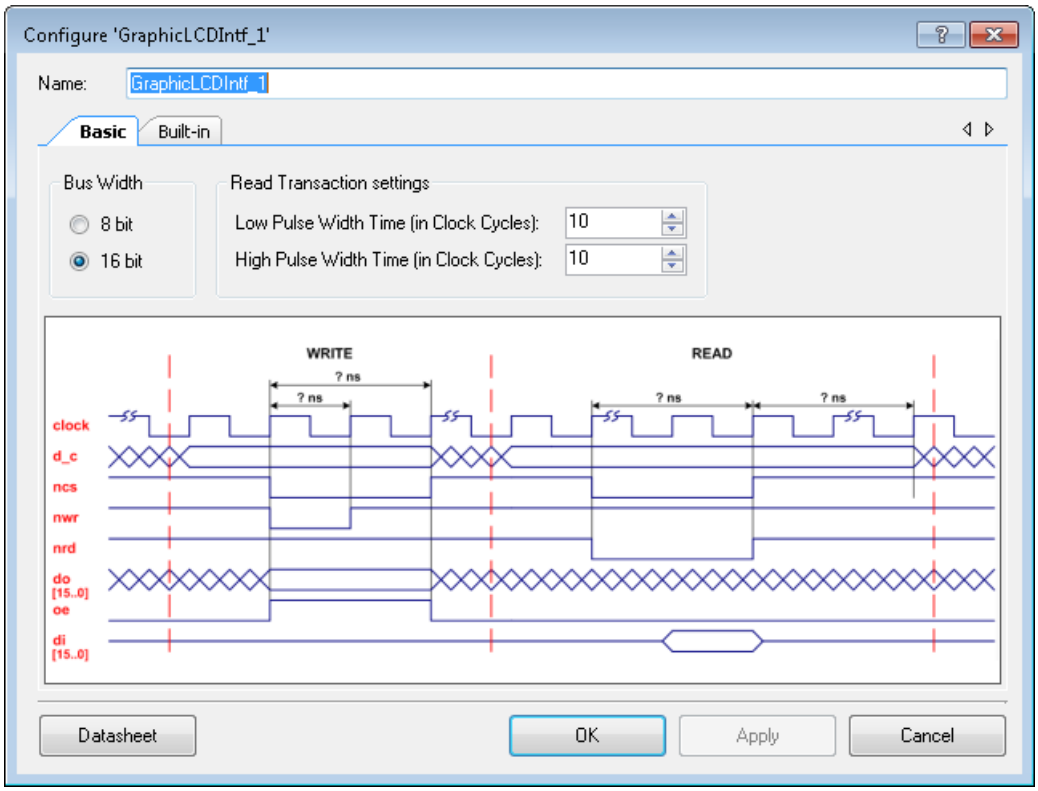


Each macro has the clock set to 20 MHz and the pulse width settings left at the default. These are the correct settings for the SSD1289 Controller.

The “Input Synchronized” option is unchecked on all of the data pins and API generation for all of the pins is turned off.

Component Parameters

Drag a GraphicLCDIntf Component onto your design and double-click it to open the Configure dialog. The default GraphicLCDIntf settings are the proper settings for operation with the Solomon Systech SSD1289 Controller.



Bus Width

Determines whether the Component supports an 8- or 16-bit parallel interface to a graphic LCD controller. The default setting is **16 bit**.

Low Pulse Width Time

Determines the number of clock cycles required for the read pulse width low for the controller. This value can be set between 2 and 255 clock cycles (the minimum is 2 because the read value must be sampled one clock before the end of the pulse). The default setting is **10**.

High Pulse Width Time

Determines the number of clock cycles required for read pulse width high for the controller. This value can be set between 1 and 255 clock cycles. The default setting is **10**.

Clock Selection

There is no internal clock in this Component. You must attach a clock source. This Component operates from a single clock connected to the Component.

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 discuss each function in more detail.

By default, PSoC Creator assigns the instance name “GraphicLCDIntf_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 generated for the Component. For readability, the instance name used in the following table is “GraphicLCDIntf.”

Functions

Function	Description
GraphicLCDIntf_Start()	Starts the GraphicLCDIntf interface.
GraphicLCDIntf_Stop()	Disables the GraphicLCDIntf interface.
GraphicLCDIntf_Write8()	Initiates a write transaction on the 8-bit parallel interface.
GraphicLCDIntf_Write16()	Initiates a write transaction on the 16-bit parallel interface.
GraphicLCDIntf_WriteM8()	Initiates multiple write transactions on the 8-bit parallel interface.
GraphicLCDIntf_WriteM16()	Initiates multiple write transactions on the 16-bit parallel interface.
GraphicLCDIntf_Write8_A0()	Initiates a write transaction on the 8-bit parallel interface, d_c line low.
GraphicLCDIntf_Write16_A0()	Initiates a write transaction on the 16-bit parallel interface, d_c line low.
GraphicLCDIntf_Write8_A1()	Initiates a write transaction on the 8-bit parallel interface, d_c line high.
GraphicLCDIntf_Write16_A1()	Initiates a write transaction on the 16-bit parallel interface, d_c line high.
GraphicLCDIntf_WriteM8_A0()	Initiates multiple write transactions on the 8-bit parallel interface, d_c line low.
GraphicLCDIntf_WriteM16_A0()	Initiates multiple write transactions on the 16-bit parallel interface d_c line low.
GraphicLCDIntf_WriteM8_A1()	Initiates multiple write transactions on the 8-bit parallel interface, d_c line high.
GraphicLCDIntf_WriteM16_A1()	Initiates multiple write transactions on the 16-bit parallel interface d_c line high.
GraphicLCDIntf_Read8()	Initiates a read transaction on the 8-bit parallel interface.
GraphicLCDIntf_Read16()	Initiates a read transaction on the 16-bit parallel interface.
GraphicLCDIntf_ReadM8()	Initiates multiple read transactions on the 8-bit parallel interface.
GraphicLCDIntf_ReadM16()	Initiates multiple read transactions on the 16-bit parallel interface.
GraphicLCDIntf_Read8_A1()	Initiates a read transaction on the 8-bit parallel interface.

Function	Description
GraphicLCDIntf_Read16_A1()	Initiates a read transaction on the 16-bit parallel interface.
GraphicLCDIntf_ReadM8_A1()	Initiates multiple read transactions on the 8-bit parallel interface, d_c line high.
GraphicLCDIntf_ReadM16_A1()	Initiates multiple read transactions on the 16-bit parallel interface, d_c line high.
GraphicLCDIntf_Sleep()	Saves the configuration and disables the GraphicLCDIntf.
GraphicLCDIntf_Wakeup()	Restores the configuration and enables the GraphicLCDIntf.
GraphicLCDIntf_Init()	Initializes or restores the default GraphicLCDIntf configuration.
GraphicLCDIntf_Enable()	Enables the GraphicLCDIntf.
GraphicLCDIntf_SaveConfig()	Saves the configuration of the GraphicLCDIntf.
GraphicLCDIntf_RestoreConfig()	Restores the configuration of the GraphicLCDIntf.

void GraphicLCDIntf_Start(void)

Description: This function enables Active mode power template bits or clock gating as appropriate. Configures the Component for operation.

void GraphicLCDIntf_Stop(void)

Description: This function disables Active mode power template bits or gates clocks as appropriate.

void GraphicLCDIntf_Write8(uint8 d_c, uint8 wrData)

Description: This function initiates a write transaction on the 8-bit parallel interface. The write is a posted write, so this function returns before the write has actually completed on the interface. If the command queue is full, this function does not return until space is available to queue this write request.

Parameters: d_c: Data (1) or Command (0) indication. Passed to the d_c pin

wrData: Data sent on the do_lsb[7:0] pins

void GraphicLCDIntf_Write16(uint8 d_c, uint16 wrData)

Description: This function initiates a write transaction on the 16-bit parallel interface. The write is a posted write, so this function returns before the write has actually completed on the interface. If the command queue is full, this function does not return until space is available to queue this write request.

Parameters: d_c: Data (1) or Command (0) indication. Passed to the d_c pin

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

void GraphicLCDIntf_WriteM8(uint8 d_c, uint8 wrData[], uint16 num)

Description: This function initiates multiple write transactions on the 8-bit parallel interface. Writing of multiple bytes with one execution of GraphicLCDIntf_WriteM8, instead of multiple executions of GraphicLCDIntf_Write8 increases the write performance on the interface.

Parameters: d_c: Data (1) or Command (0) indication. Passed to the d_c pin

wrData: Pointer to a write buffer. Data from the buffer are sent on the do_lsb[7:0] pins

num: Number of bytes to write

void GraphicLCDIntf_WriteM16(uint8 d_c, uint16 wrData[], uint16 num)

Description: This function initiates multiple write transactions on the 16-bit parallel interface. Writing of multiple words with one execution of GraphicLCDIntf_WriteM16, instead of multiple executions of GraphicLCDIntf_Write16 increases the write performance on the interface.

Parameters: d_c: Data (1) or Command (0) indication. Passed to the d_c pin

wrData: Pointer to a write buffer. Data from the buffer are sent on the do_msb[7:0] (most significant byte) and do_lsb[7:0] (least significant byte) pins

num: Number of words to write

void GraphicLCDIntf_Write8_A0(uint8 wrData)

Description: This function initiates a command write transaction on the 8-bit parallel interface with the d_c pin set to 0. 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: wrData: Data sent on the do_lsb[7:0] pins.

void GraphicLCDIntf_Write16_A0(uint8 wrData)

Description: This function initiates a command write transaction on the 16-bit parallel interface with the d_c pin set to 0. 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: wrData: Data sent on the do_msb[7:0] (most significant byte) and do_lsb[7:0] (least significant byte) pins.

void GraphicLCDIntf_Write8_A1(uint8 wrData)

Description: This function initiates a data write transaction on the 8-bit parallel interface with the d_c pin set to 1. 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: wrData: Data sent on the do_lsb[7:0] pins.

void GraphicLCDIntf_Write16_A1(uint8 wrData)

Description: This function initiates a data write transaction on the 16-bit parallel interface with the d_c pin set to 1. 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: wrData: Data sent on the do_msb[7:0] (most significant byte) and do_lsb[7:0] (least significant byte) pins.

void GraphicLCDIntf_WriteM8_A0(uint8 wrData[], int num)

Description: This function initiates multiple data write transactions on the 8-bit parallel interface with the d_c pin set to 0. 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: wrData: Pointer to a write buffer. Data from the buffer are sent on the do_lsb[7:0] pins

num: Number of bytes to write

void GraphicLCDIntf_WriteM16_A0(uint16 wrData[], int num)

Description: This function initiates multiple data write transactions on the 16-bit parallel interface with the d_c pin set to 0. 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: wrData: Pointer to a write buffer. Data from the buffer are sent on the do_msb[7:0] (most significant byte) and do_lsb[7:0] (least significant byte) pins

num: Number of words to write

void GraphicLCDIntf_WriteM8_A1(uint8 wrData[], int num)

Description: This function initiates multiple data write transactions on the 8-bit parallel interface with the d_c pin set to 1. 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: wrData: Pointer to a write buffer. Data from the buffer are sent on the do_lsb[7:0] pins

num: Number of bytes to write

void GraphicLCDIntf_WriteM16_A1(uint16 wrData[], int num)

Description: This function initiates multiple data write transactions on the 16-bit parallel interface with the d_c pin set to 1. 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: wrData: Pointer to a write buffer. Data from the buffer are sent on the do_msb[7:0] (most significant byte) and do_lsb[7:0] (least significant byte) pins

num: Number of words to write

uint8 GraphicLCDIntf_Read8(uint8 d_c)

Description: This function initiates a read transaction on the 8-bit parallel interface. The read executes after all currently posted writes have completed. This function waits until the read completes and then returns the read value.

Parameters: d_c: Data (1) or Command (0) indication. Passed to the d_c pin.

Return Value: 8-bit read value from the di_lsb[7:0] pins

uint16 GraphicLCDIntf_Read16(uint8 d_c)

Description: This function initiates a read transaction on the 16-bit parallel interface. The read executes after all currently posted writes have completed. This function waits until the read completes and then returns the read value.

Parameters: d_c: Data (1) or Command (0) indication. Passed to the d_c pin.

Return Value: 16-bit read value from the di_msb[7:0] (most significant byte) and di_lsb[7:0] (least significant byte) pins

void GraphicLCDIntf_ReadM8 (uint8 d_c, uint8 rdData[], uint16 num)

Description: This function initiates a read transaction on the 8-bit parallel interface. The read executes after all currently posted writes have completed. This function waits until the read completes and then returns the read value.

Parameters: d_c: Data (1) or Command (0) indication. Passed to the d_c pin.

rdData: Pointer to a read buffer. Data written to the buffer from the di_lsb[7:0] pins.

num: Number of words to read.

void GraphicLCDIntf_ReadM16 (uint8 d_c, uint16 rdData[], uint16 num)

Description: This function initiates a read transaction on the 16-bit parallel interface. The read executes after all currently posted writes have completed. This function waits until the read completes and then returns the read value.

Parameters: d_c: Data (1) or Command (0) indication. Passed to the d_c pin.

rdData: Pointer to a read buffer. Data written to the buffer from the di_msb[7:0] (most significant byte) and di_lsb[7:0] (least significant byte) pins.

num: Number of words to read.

uint8 GraphicLCDIntf_Read8_A1(void)

Description: This function initiates a data read transaction on the 8-bit parallel interface with the d_c pin set to 1. 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.

Return Value: 8-bit read value from the di_lsb[7:0] pins

uint16 GraphicLCDIntf_Read16_A1(void)

Description: This function initiates a data read transaction on the 16-bit parallel interface with the d_c pin set to 1. 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.

Return Value: 16-bit read value from the di_msb[7:0] (most significant byte) and di_lsb[7:0] (least significant byte) pins

void GraphicLCDIntf_ReadM8_A1 (uint8 rdData[], uint16 num)

Description: This function initiates a read transaction on the 8-bit parallel interface with the d_c pin set to 1. The read executes after all currently posted writes have completed. This function waits until the read completes and then returns the read value.

Parameters: rdData: Pointer to a read buffer. Data written to the buffer from the di_lsb[7:0] pins.

num: Number of words to read.

void GraphicLCDIntf_ReadM16_A1 (uint16 rdData[], uint16 num)

Description: This function initiates a read transaction on the 16-bit parallel interface with the d_c pin set to 1. The read executes after all currently posted writes have completed. This function waits until the read completes and then returns the read value.

Parameters: rdData: Pointer to a read buffer. Data written to the buffer from the di_msb[7:0] (most significant byte) and di_lsb[7:0] (least significant byte) pins.

num: Number of words to read.

void GraphicLCDIntf_Sleep(void)

Description: Stops the Component operation and saves the user configuration.

void GraphicLCDIntf_Wakeup(void)

Description: Restores the user configuration and restores Component state.

Side Effects: Calling the GraphicLCDIntf_Wakeup() function without first calling the GraphicLCDIntf_Sleep() function may produce unexpected behavior.

void GraphicLCDIntf_Init(void)

Description: This function initializes or restores the Component according to the customizer Configure dialog settings. It is not necessary to call GraphicLCDIntf_Init() because the GraphicLCDIntf_Start() routine calls this function and is the preferred method to begin Component operation. Only the static Component configuration that defines Read Low and High Pulse Widths will be restored to its initial values.

Side Effects: This reinitializes the Component but it does not clear data from the FIFOs, and it does not reset the Component hardware state machine. The current transaction is performed on the bus.

void GraphicLCDIntf_Enable(void)

Description: This function activates the hardware and begins Component operation. It is not necessary to call GraphicLCDIntf_Enable() because the GraphicLCDIntf_Start() routine calls this function, which is the preferred method to begin Component operation.

void GraphicLCDIntf_SaveConfig(void)

Description: This function saves the Component configuration and non-retention registers. This function is called by the GraphicLCDIntf_Sleep() function.

void GraphicLCDIntf_RestoreConfig(void)

Description: This function restores the configuration of GraphicLCDIntf non-retention registers. The API is called by GraphicLCDIntf_Wakeup function.

Global Variables

Variable	Description
GraphicLCDIntf_initVar	Indicates whether the Graphic LCD Interface has been initialized. The variable is initialized to 0 and set to 1 the first time GraphicLCDIntf_Start() is called. This allows the Component to restart without reinitialization after the first call to the GraphicLCDIntf_Start() routine. If reinitialization of the Component is required then the GraphicLCDIntf_Init() function can be called before the GraphicLCDIntf_Start() or GraphicLCDIntf_Enable() function.

API Memory Usage

The Component memory usage varies significantly, depending on the compiler, device, number of APIs used and Component configuration. The following table provides the memory usage for all APIs available in the given Component configuration.

The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design the map file generated by the compiler can be analyzed to determine the memory usage.

Configuration	PSoC 3 (Keil_PK51)		PSoC 4 (GCC)		PSoC 5LP (GCC)		PSoC 6 MCU (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
8-bit interface	303	1	352	1	292	1	300	1
16-bit interface	355	1	360	1	300	1	308	1

Sample Firmware Source Code

PSoC Creator provides many code examples that include schematics and example code in the Find Code Example dialog. For Component-specific examples, open the dialog from the Component Catalog or an instance of the Component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

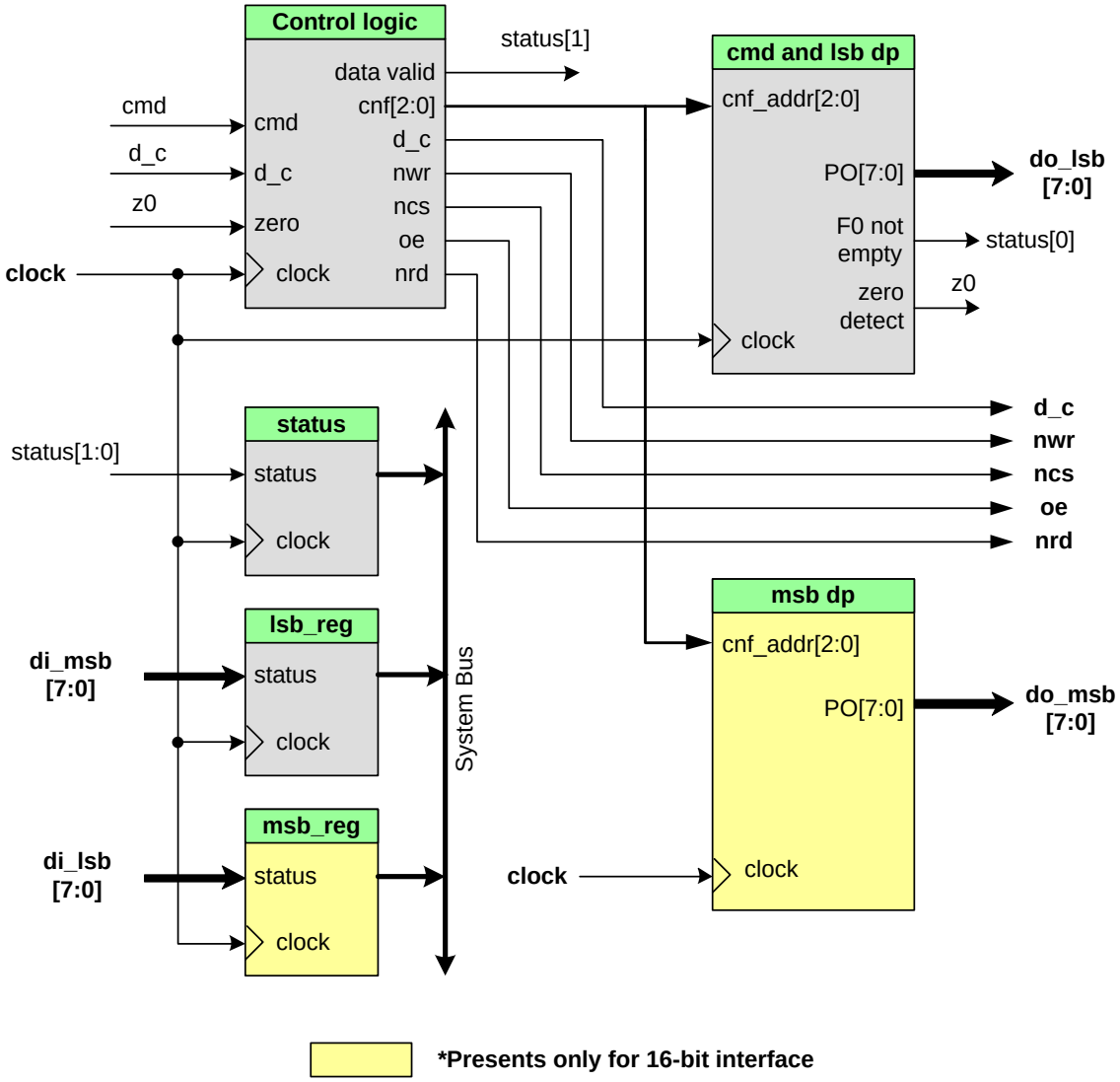
Refer to the “Find Code Example” topic in the PSoC Creator Help for more information.

Functional Description

Block Diagram and Configuration

The GraphicLCDIntf Component is implemented as a set of configured UDBs. Figure 1 shows this implementation.

Figure 1. Block Diagram



Bus Transactions

This interface can perform either a read or a write transaction. These transactions have the following parameters:

- Read or write
- Address: In this case it is a one bit address driven on the d_c pin
- Data (8 or 16 bits): Sent on “do” for writes and read on “di” for reads.

The implementation assumes that the CPU sends a command byte to the Component using a command FIFO. That command byte indicates read or write and provides the d_c bit.

Idle Condition

When neither a read nor a write is occurring on the interface, the interface is in the idle state. The values for the output pins in that condition are:

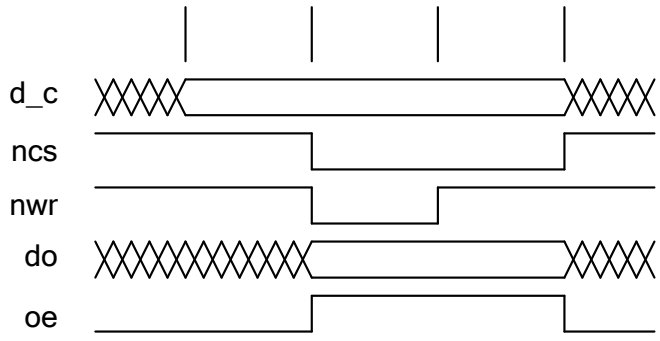
- d_c: Don’t care (may be left at its last state)
- ncs: 1
- nwr: 1
- nrd: 1
- do: Don’t care (may be left at its last state)
- oe: 0

In the description of the read and write transactions, any signal not listed is idle.

Write Transaction

Figure 2 shows the timing diagram for a write transaction on the parallel interface.

Figure 2. Write Transaction Timing Diagram



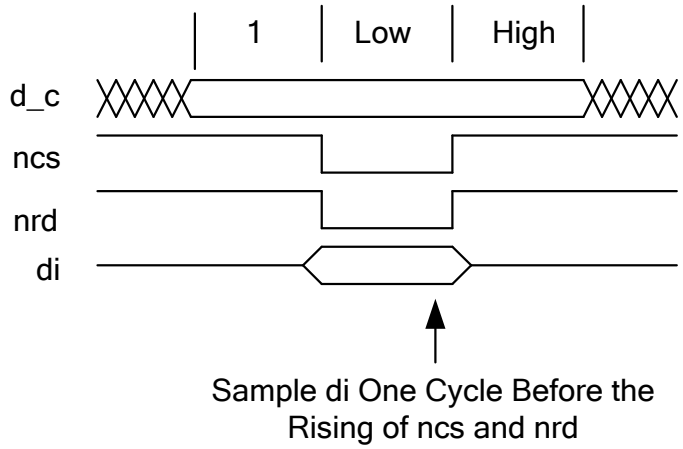
This diagram shows that the write transaction requires three clock cycles. The timing diagram is the same regardless of the bit width. This transaction can be immediately preceded or followed by another read or write transaction or may be in the idle state before or after a write transaction.

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 on parallel bus). The implementation allows the CPU to have four write requests outstanding without stalling.

Read Transaction

Figure 3 shows the timing diagram for a read transaction on the parallel interface.

Figure 3. Read Transaction Timing Diagram



This diagram shows that the read transaction requires a variable number of clock cycles depending on the setting for the high and low read pulse widths. The timing diagram is the same regardless of the bit width. Note that the data input is sampled one clock cycle before the end of the ncs and nrd low pulses. This transaction can be immediately preceded or followed by another read or write transaction or may be in the idle state before or after a read transaction.

The ordering of reads and writes is maintained (reads occur before posted writes have completed). Reads require the CPU to wait for the completion of the read transaction before proceeding.

Industry Standards

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the Component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator Components
- specific deviations – deviations that are applicable only for this Component

This section provides information on Component-specific deviations. This section provides information on Component-specific deviations. For non-PSoC 6 devices, project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment. For PSoC 6 devices, refer to the Generated Files (PSoC 6) topic in the PSoC Creator Help for information on MISRA compliance and deviations for files generated by PSoC Creator.

The GraphicLCDIntf Component has the following specific deviations:

Rule	Rule Class	Rule Description	Description of Deviation(s)
16.7	A	A pointer parameter in a function prototype should be declared as pointer to const if the pointer is not used to modify the addressed object.	Application interface defined by emWin FlexColor driver.
6.3	A	Typedefs that indicate size and signedness should be used in place of the basic numerical types.	Application interface defined by emWin FlexColor driver.
19.7	A	A function should be used in preference to a function-like macro.	Function-like macro needed to provide instance non-specific API names/backward compatibility

Registers

GraphicLCDIntf_STATUS_REG

Bits	7	6	5	4	3	2	1	0
Value	reserved						data_valid	F0_half_ empty

- F0_half_empty: If set, there is at least two bytes of room in the command/data FIFO.
- data_valid: Set if read data is valid for the CPU. This bit is cleared when the CPU reads the register.

GraphicLCDIntf_DIN_LSB_DATA_REG

Bits	7	6	5	4	3	2	1	0
Value	di_lsb[7:0]							

- The lower eight bits of the input data bus for read transaction

You can read the register value with the GraphicLCDIntf_Read8() API function for an 8-bit interface. The value is the least significant byte of returned value from the GraphicLCDIntf_Read16() API function for a 16-bit interface.

GraphicLCDIntf_DIN_MSB_DATA_REG

Bits	7	6	5	4	3	2	1	0
Value	di_msb[7:0]							

- The upper eight bits of the input data bus for read transaction

The register value is the most significant byte of returned value from the GraphicLCDIntf_Read16() API function for a 16-bit interface.

Note The DIN_LSB_DATA_REG and DIN_MSB_DATA_REG bits are cleared when CPU firmware reads these registers.

Resources

The Graphic LCD Interface Component is placed throughout the UDB array. The Component utilizes the following resources.

Configuration	Resource Type					
	Datapath Cells	Macrocells	Status Cells	Control Cells	DMA Channels	Interrupts
8-bit interface	1	11	2	–	–	–
16-bit interface	2	11	3	–	–	–

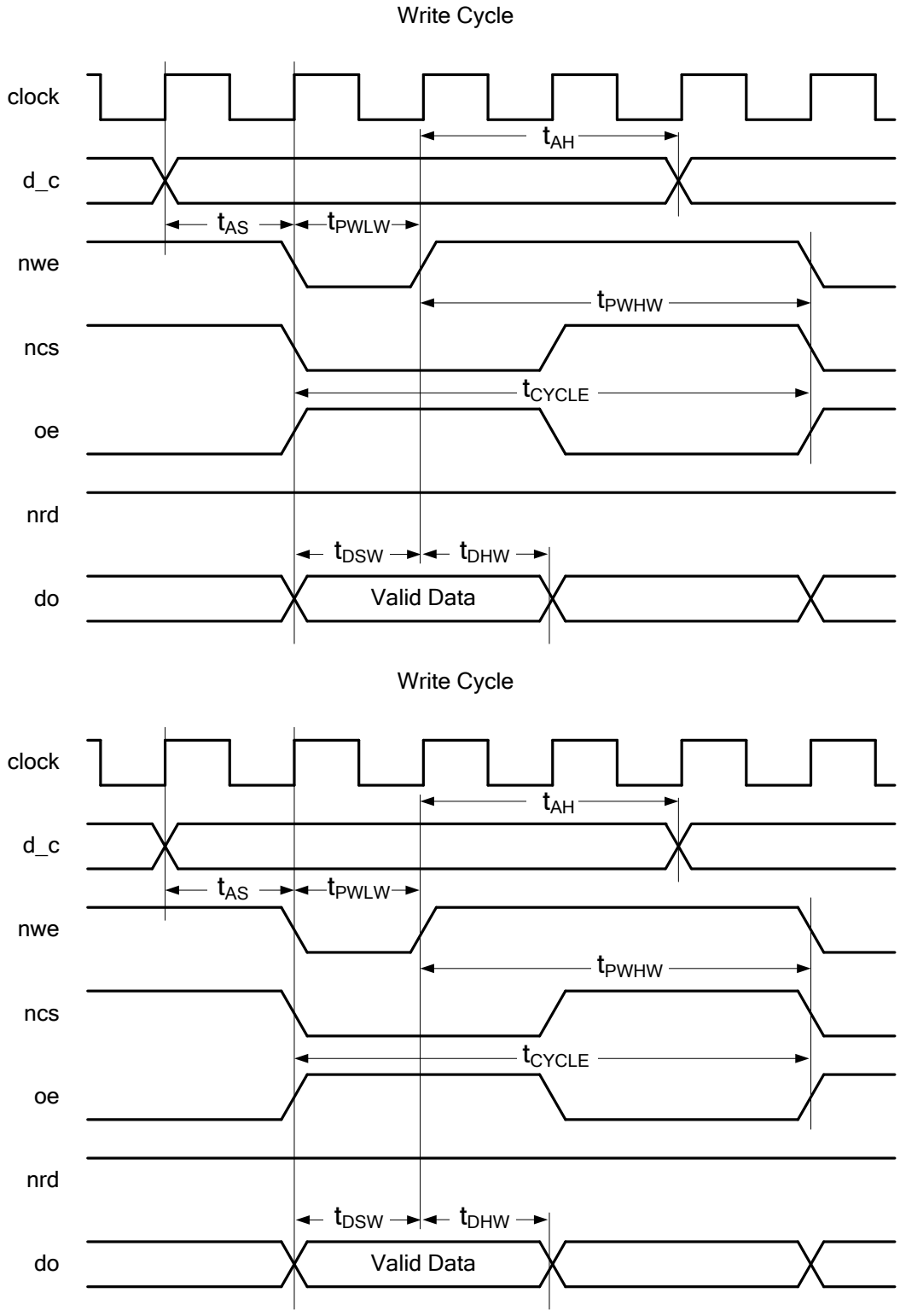
AC Characteristics

Specifications are valid for –40 °C ≤ T_A ≤ 85 °C and T_J ≤ 100 °C, except where noted.
Specifications are valid for 1.71 V to 5.5 V, except where noted.

Parameter	Description	Min	Typ	Max	Unit
f_{CLOCK}	Component clock frequency	–	–	33	MHz
t_{AS}	Address setup time	1	–	–	t_{CY_clock}[1]
t_{PWLW}	Pulse width low write	–	1	–	t_{CY_clock}
t_{PWHW}	Pulse width high write	3	–	–	t_{CY_clock}
t_{PWLR}	Pulse width low read	2	–	255	t_{CY_clock}
t_{PWHR}	Pulse width high read	1	–	255	t_{CY_clock}
t_{AH}	Address hold time				
	Write	2	–	–	t_{CY_clock}
	Read	t_{PWHR}	–	–	t_{CY_clock}
t_{CYCLE}	Clock cycle time				
	Write cycle	4	–	–	t_{CY_clock}
	Read cycle	t_{PWLR} + t_{PWRH} + 1	–	–	t_{CY_clock}
t_{DSW}	Data setup time	–	1	–	t_{CY_clock}
t_{DHW}	Data hold time	–	1	–	t_{CY_clock}
t_{ACC}	Data access time	–	t_{PWHR} – 1	–	t_{CY_clock}
t_{DHR}	Output hold time	–	0	–	t_{CY_clock}

1. $t_{CY_clock} = 1/f_{CLOCK}$. This is the cycle time of one clock period

Figure 4. Data Transition Timing Diagram



How to Use STA Results for Characteristics Data

You can calculate the maximums for your designs with the Static Timing Analysis (STA) results using the following methods:

f_{CLOCK} Maximum Component clock frequency appears in Timing results in the clock summary as the named Component clock (CLK in this case). The following graphic shows an example of the clock limitations.

- Clock Summary Section

Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	60.000 MHz	60.000 MHz	N/A	
CLK	CyMASTER_CLK	20.000 MHz	20.000 MHz	59.492 MHz	
CyBUS_CLK	CyMASTER_CLK	60.000 MHz	60.000 MHz	N/A	
CyPLL_OUT	CyPLL_OUT	60.000 MHz	60.000 MHz	N/A	

The remaining parameters are implementation-specific and are measured in clock cycles. They can be divided into two categories.

- The parameters that are used to configure the Component:

t_{PWLW} The minimum pulse width low time for the write signal

t_{PWLR} The minimum pulse width low time for the read signal

t_{PWHR} The minimum pulse width high time for the read signal

You can find the specific description of how to use these parameters when configuring the Component in the When to Use a GraphicLCDIntf section on page 1.

- The parameters that are fixed based on the Component implementation:

t_{PWHW} The minimum pulse width high time for the write signal

t_{AS} The minimum amount of time the address signal is valid before the falling edge of the nwr/nrd signal

t_{AH} The minimum amount of time the address signal is valid after the rising edge of the nwr/nrd signal

t_{CYCLE} The period of time during which a single transaction (write/read) is performed on the interface

t_{DSW} The minimum amount of time the data is valid before the rising edge of the write signal

t_{DHW} The minimum amount of time the data is valid after the rising edge of the write signal

t_{ACC} The minimum amount of time the data is sampled after the negative edge of the read signal

t_{DHR} The minimum amount of time the data should be valid after rising edge of the nrd signal

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.80.a	Updated the datasheet.	Changed link location to the library.
1.80	Added PSoC 4 and PSoC 6 MCU support. Added additional API functions. Updated the datasheet.	 Updated the MISRA Compliance, Characteristics, and API Memory sections.
1.70.c	Minor datasheet edits.	
1.70.b	Minor datasheet edits.	
1.70.a	Added MISRA Compliance section.	The Component was not verified for MISRA compliance.
1.70	Added GraphicLCDIntf_WriteM8/16 APIs to the Component.	Increase the performance of image drawing operations of the emWin graphics library.
	Added DC characteristics section to datasheet.	
1.61	Added all Component APIs with the CYREENTRANT keyword when they are included in the .cyre file.	Not all APIs are truly reentrant. Comments in the Component API source files indicate which functions are candidates. This change is required to eliminate compiler warnings for functions that are not reentrant used in a safe way: protected from concurrent calls by flags or Critical Sections.
	Added timing constraints to mark false timing paths in the Component.	Removes paths that are not used from timing analysis. This avoids false timing violation messages.
1.60.a	Removed references to the associated kits from the datasheet.	
1.60	Resampled the FIFO block status signals to the DP clock.	Allows the Component to function with the same timing results for all PSoC 3 and PSoC 5 silicons.
	Added characterization data to the datasheet	
	Minor datasheet edits and updates	

© Cypress Semiconductor Corporation, 2018. 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, WICED, 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.