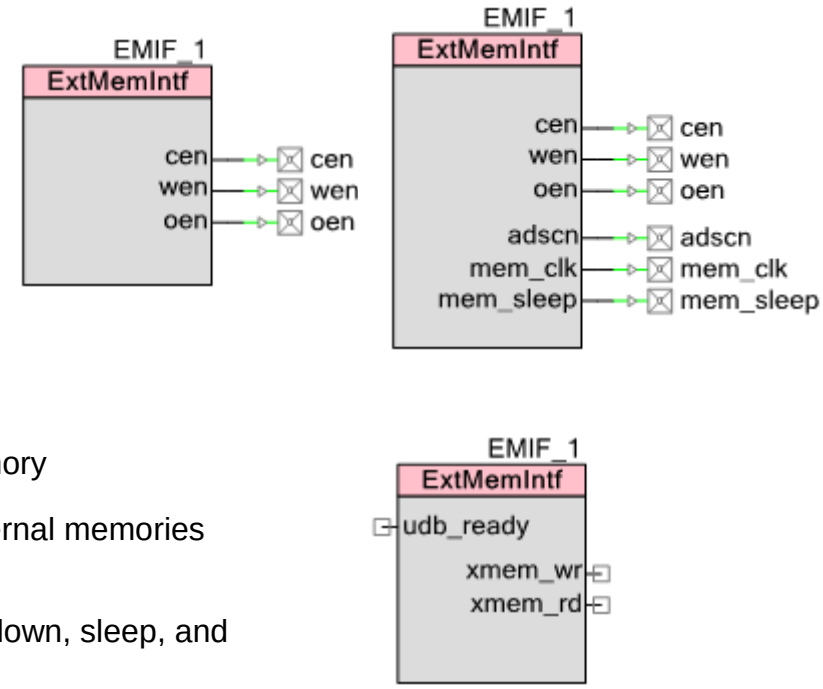
External Memory Interface (EMIF)

1.20

Features



- 8-, 16-, 24-bit address bus width
- 8-, 16-bit data bus width
- Supports external synchronous memory
- Supports external asynchronous memory
- Supports custom interface for memory
- Supports a range of speeds of external memories (from 5 to 200 ns)
- Supports external memory power-down, sleep, and wakeup modes

General Description

The EMIF component enables access by the CPU or DMA to memory ICs external to the PSoC 3/5LP. It facilitates setup of the EMIF hardware, as well as UDBs and GPIOs as required. The EMIF can control synchronous and asynchronous memories without the need to configure any UDBs in synchronous and asynchronous modes. In UDB mode, UDBs must be configured to generate external memory control signals.

When to Use an EMIF

The EMIF is used to expand available memory space. This allows for expanded data storage in off-chip memory devices. External memory ICs connected to a PSoC are typically intended to hold large arrays of data, such as text, audio, or video content. Other expected applications include data logging and use as a buffer for external LCD pixel data.

Input/Output Connections

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

udb_ready – Input*

Signal from external logic, indicating to PHUB that the AHB bus transaction is complete (active high). This input is visible when the **External Memory Type** parameter is set to **Custom**.

xmem_wr – Output*

Generic write signal to custom interface (active high). Signal from external logic, indicating to PHUB that the AHB bus transaction is complete (active high). This output is visible when the **External Memory Type** parameter is set to **Custom**.

xmem_rd – Output*

Generic read signal to custom interface (active high). This output is visible when the **External Memory Type** parameter is set to **Custom**.

cen – Output*

Chip enable to external memory (active low). This output is visible when the **External Memory Type** parameter is set to **Synchronous** or **Asynchronous**.

wen – Output*

Write enable to external memory (active low). This output is visible when the **External Memory Type** parameter is set to **Synchronous** or **Asynchronous**.

oen – Output*

Output enable to external memory (active low). This output is visible when the **External Memory Type** parameter is set to **Synchronous** or **Asynchronous**.

mem_clk – Output*

Clock to external synchronous memory. This output is visible when the **External Memory Type** parameter is set to **Synchronous**.

adscn – Output*

Address strobe to external synchronous memory (active low). This output is visible when the **External Memory Type** parameter is set to **Synchronous**.

mem_sleep – Output*

Sleep signal to external synchronous memory sleep pin (active high). This output is visible when the **External Memory Type** parameter is set to **Synchronous**.

Schematic Macro Information

By default, the PSoC Creator Component Catalog contains schematic macro implementations for the EMIF component. These macros contain already connected output pins. Schematic macros are available both for asynchronous External Memory Interface and synchronous External Memory Interface.

Figure 1. Asynchronous External Memory Interface Schematic Macro

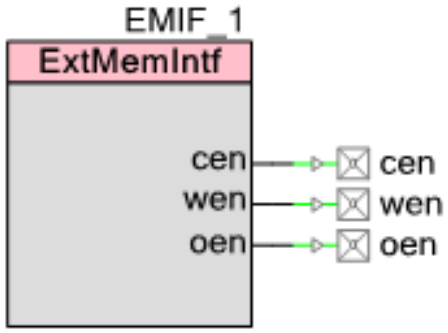
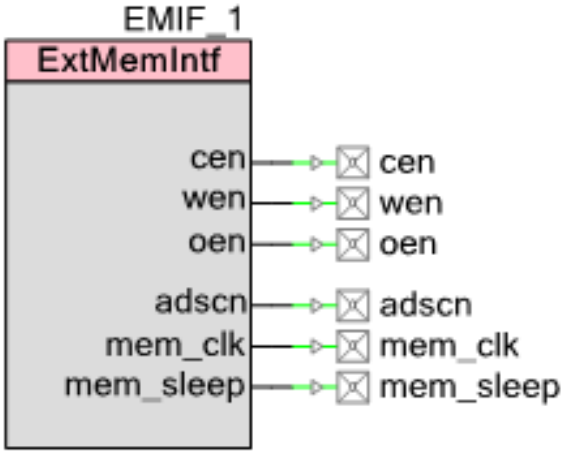


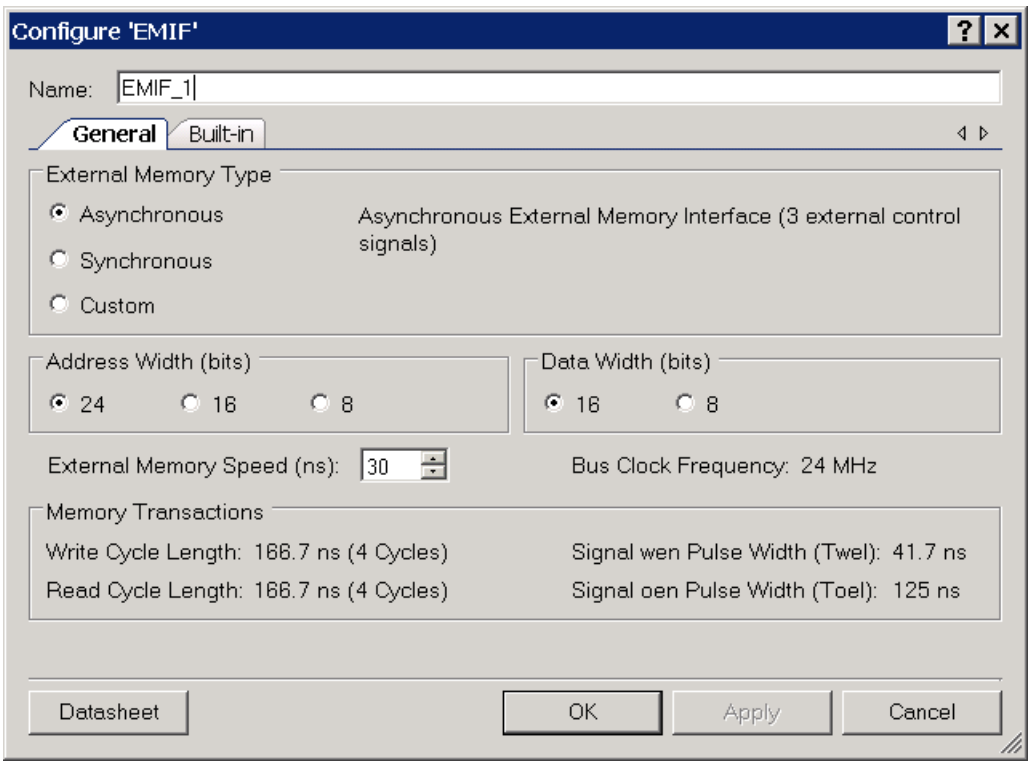
Figure 2. Synchronous External Memory Interface Schematic Macro



Component Parameters

Drag an EMIF onto your design and double click it to open the **Configure** dialog.

Figure 3. Configure EMIF Dialog



The EMIF provides the following parameters:

External Memory Type

Determines external memory type. Default external memory type is **Asynchronous**.

Address Width (bits)

Determines the number of bits in the address bus for external memory. The default address bus width is **24 bits**.

Data Width (bits)

Determines the number of bits in the data bus for external memory. The default data bus width is **16 bits**.

External Memory Speed (ns)

Determines external memory speed in ns. The default external memory speed is **30 ns**.

Bus Clock Frequency

Shows the selected Bus Clock frequency in megahertz.

Memory Transactions

- **Write Cycle Length** – Shows the write cycle length in nanoseconds. The Bus Clock frequency cycles are shown in parentheses.
- **Read Cycle Length** – Shows the read cycle length in the nanoseconds. The Bus Clock frequency cycles are shown in parentheses.
- **Signal wen Pulse Width (Twel)** – Shows the Twel parameter (write enable signal pulse width) in nanoseconds. See the DC and AC Electrical Characteristics section for details.
- **Signal oen Pulse Width (Toel)** – Shows the Toel parameter (write enable signal pulse width) in nanoseconds. See the DC and AC Electrical Characteristics section for details.

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 “EMIF_1” to the first instance of a component in a given design. You can rename it 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 “EMIF.”

Function	Description
EMIF_Start()	Calls EMIF_Init() and EMIF_Enable().
EMIF_Stop()	Disables the EMIF block. Returns all associated I/O ports and pins to HI-Z mode
EMIF_Init()	Initializes or restores the EMIF configuration to the current customizer state
EMIF_Enable()	Enables the EMIF hardware block, associated I/O ports and pins
EMIF_ExtMemSleep()	Sets the external memory sleep signal high; note that depending on the type of external memory IC used, the signal may need to be inverted
EMIF_ExtMemWakeup()	Sets the external memory sleep signal low; note that depending on the type of external memory IC used, the signal may need to be inverted
EMIF_SaveConfig()	Saves the user configuration of the EMIF nonretention registers. This routine is called by EMIF_Sleep() to save the component configuration before entering sleep
EMIF_Sleep()	Stops the EMIF operation and saves the user configuration along with the enable state of the EMIF

Function	Description
EMIF_RestoreConfig()	Restores the user configuration of the EMIF nonretention registers. This routine is called by EMIF_Wakeup() to restore the component configuration when exiting sleep
EMIF_Wakeup()	Restores the user configuration and restores the enable state

void EMIF_Start(void)

Description: This is the preferred method to begin component operation. EMIF_Start() calls the EMIF_Init() function, and then calls the EMIF_Enable() function.

Parameters: None

Return Value: None

Side Effects: None

void EMIF_Stop(void)

Description: Disables the EMIF block. Returns all associated I/O ports and pins to HI-Z mode

Parameters: None

Return Value: None

Side Effects: None

void EMIF_Init(void)

Description: Initializes or restores the component according to the customizer Configure dialog settings. It is not necessary to call EMIF_Init() because the EMIF_Start() routine calls this function and is the preferred method to begin component operation.

Parameters: None

Return Value: None

Side Effects: None

void EMIF_Enable(void)

Description: Activates the hardware and begins component operation. It is not necessary to call EMIF_Enable() because the EMIF_Start() routine calls this function, which is the preferred method to begin component operation.

Parameters: None

Return Value: None

Side Effects: None

void EMIF_ExtMemSleep(void)

Description: Sets the ‘mem_pd’ bit in the EMIF_PWR_DWN register. This sets the external memory sleep signal high. Depending on the type of external memory IC used, the signal may need to be inverted.

Parameters: None

Return Value: None

Side Effects: None

void EMIF_ExtMemWakeup(void)

Description: Resets the ‘mem_pd’ bit in the EMIF_PWR_DWN register. This sets the external memory sleep signal low. Depending on the type of external memory IC used, the signal may need to be inverted

Parameters: None

Return Value: None

Side Effects: None

void EMIF_SaveConfig(void)

Description: This function saves the component configuration. This will save nonretention registers. This function will also save the current component parameter values, as defined in the Configure dialog or as modified by appropriate APIs. This function is called by the EMIF_Sleep() function.

Parameters: None

Return Value: None

Side Effects: None

void EMIF_Sleep(void)

Description: This is the preferred routine to prepare the component for sleep. The EMIF_Sleep() routine saves the current component state. Then it calls the EMIF_Stop() function and calls EMIF_SaveConfig() to save the hardware configuration.

Call the EMIF_Sleep() function before calling the CyPmSleep() or the CyPmHibernate() function. Refer to the PSoC Creator *System Reference Guide* for more information about power management functions.

Parameters: None

Return Value: None

Side Effects: None

void EMIF_RestoreConfig(void)

Description: This function restores the component configuration. This will restore nonretention registers. This function will also restore the component parameter values to what they were prior to calling the EMIF_Sleep() function.

Parameters: None

Return Value: None

Side Effects: Calling this function without first calling the EMIF_Sleep() or EMIF_SaveConfig() function may produce unexpected behavior.

void EMIF_Wakeup(void)

Description: This is the preferred routine to restore the component to the state when EMIF_Sleep() was called. The EMIF_Wakeup() function calls the EMIF_RestoreConfig() function to restore the configuration. If the component was enabled before the EMIF_Sleep() function was called, the EMIF_Wakeup() function will also re-enable the component.

Parameters: None

Return Value: None

Side Effects: Calling the EMIF_Wakeup() function without first calling the EMIF_Sleep() or EMIF_SaveConfig() function may produce unexpected behavior.

Defines

CYDEV_EXTMEM_BASE – Convenience macro for locating parameters in external memory.

The address space for the EMIF is outside of the lower 64K address space. In order to access the EMIF you need to use the extended range macros CY_GET_XTND_REG8() and CY_SET_XTND_REG8(). Refer to the System Reference Guide for more details on these macros.

External Memory Interface

16 Bit Memories:

DMA Transfers: For DMA transfers to/from 16 bit external memory, the BURSTCNT (Burst Count) channel configurations of odd values are not supported. This is because 8 bit transfers are not supported on 16 bit interface.

CPU Transfers: Since PSoC 3 is 8-bit processor, we cannot access 16 bit memory from CPU. PSoC 5LP CPU is 32-bit processor. It can access 16 bit memories. The constraint here is, it should not initiate 8 bit transfers to 16 bit memories.

8 Bit Memories:

DMA Transfers: For DMA transfers to/from 8 bit external memory, irrespective of XFRCNT (Transfer Count), the BURSTCNT (Burst Count) should always be 1.

CPU Transfers: Since PSoC 3 is 8 bit processor, there are no constraints here.

PSoC 5LP should only initiate 8 bit requests for 8 bit memories.

External memory interface Address Map

PSoC 3 External memory interface XDATA Data Address Map

Address Range:
0x800000 – 0xFFFFFF

PSoC 5LP External memory interface Memory Address Map

Address Range:
0x60000000 – 0x61FFFFFF

Sample Firmware Source Code

PSoC Creator provides numerous example projects that include schematics and example code in the Find Example Project 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 Example Project” topic in the PSoC Creator Help for more information.

Functional Description

This component, and PSoC Creator in general, does not provide any direct support for locating code, variables, structures, or arrays in external memory. However, there is nothing in either this component or in PSoC Creator that prevents an advanced user from creating or modifying source code, a linker script, or any other source file, for that purpose.

Initialized variables should not be placed in external memory. Do not locate code, variables, structures, and arrays autogenerated by PSoC Creator in external memory.

The PSoC EMIF hardware and associated GPIO ports support up to a 24-bit address bus, and an 8-bit or 16-bit data bus.

External memories require up to six GPIO ports to connect address, data, and control lines: one to three ports for address, one to two for data, and one for control. For address and data, the entire port must be selected. Unused pins in the control port are still available as GPIOs.

When an EMIF component is placed onto the project schematic, it is important that I/O port designations be set for address and data. Designation of address and data ports is not done on the symbol or schematic; instead, it is done in the Pins tab of the Design-Wide Resources window. Control signals are shown on the component symbol as terminals; the terminals should be routed to Pin components in the usual manner.

There are three EMIF types – asynchronous, synchronous, or (rarely used) custom. The type selected determines the size of the symbol, the number of terminals in the symbol, and some of the text in the symbol. The custom version of the component is the only one with an input terminal, udb_ready. It is expected that the xmem_wr and xmem_rd signals will be tied to UDB-based logic which will both generate appropriate control signals to the external IC, and provide a feedback signal back to the PHUB via the udb_ready terminal. It is up to you to determine appropriate timing for custom mode signals.

Resources

The External Memory Interface (EMIF) is a dedicated piece of hardware in the PSoC 3 and PSoC 5LP that is used in conjunction with UDBs to allow connection to external memory devices.

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 5 (GCC)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Asynchronous	168	2	N/A	N/A	204	5
Synchronous	185	2	N/A	N/A	236	5
Custom	154	2	N/A	N/A	188	5

DC and AC Electrical Characteristics

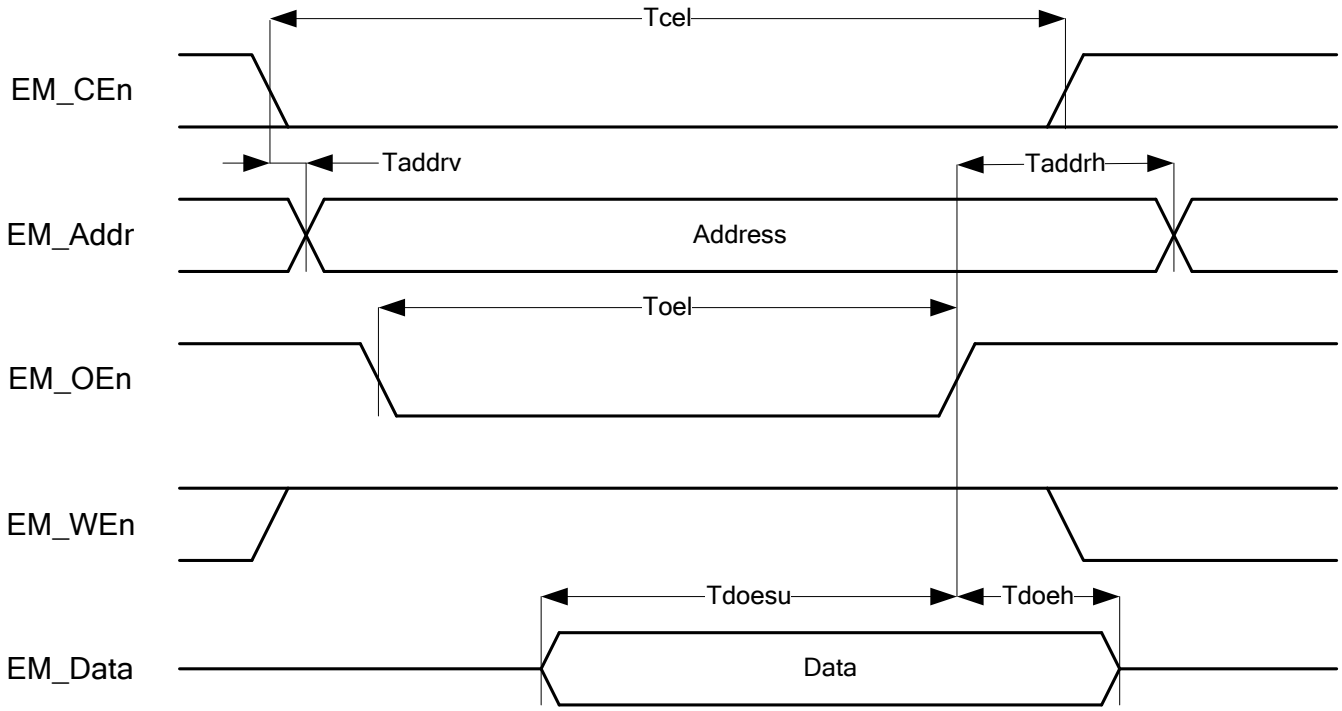
Specifications are valid for –40 °C ≤ T_A ≤ 85 °C and T_J ≤ 100 °C, except where noted.
Specifications are valid for 1.71 V to 5.5 V, except where noted.

T (EMIF clock period) = 1/f (EMIF_clock),

where:

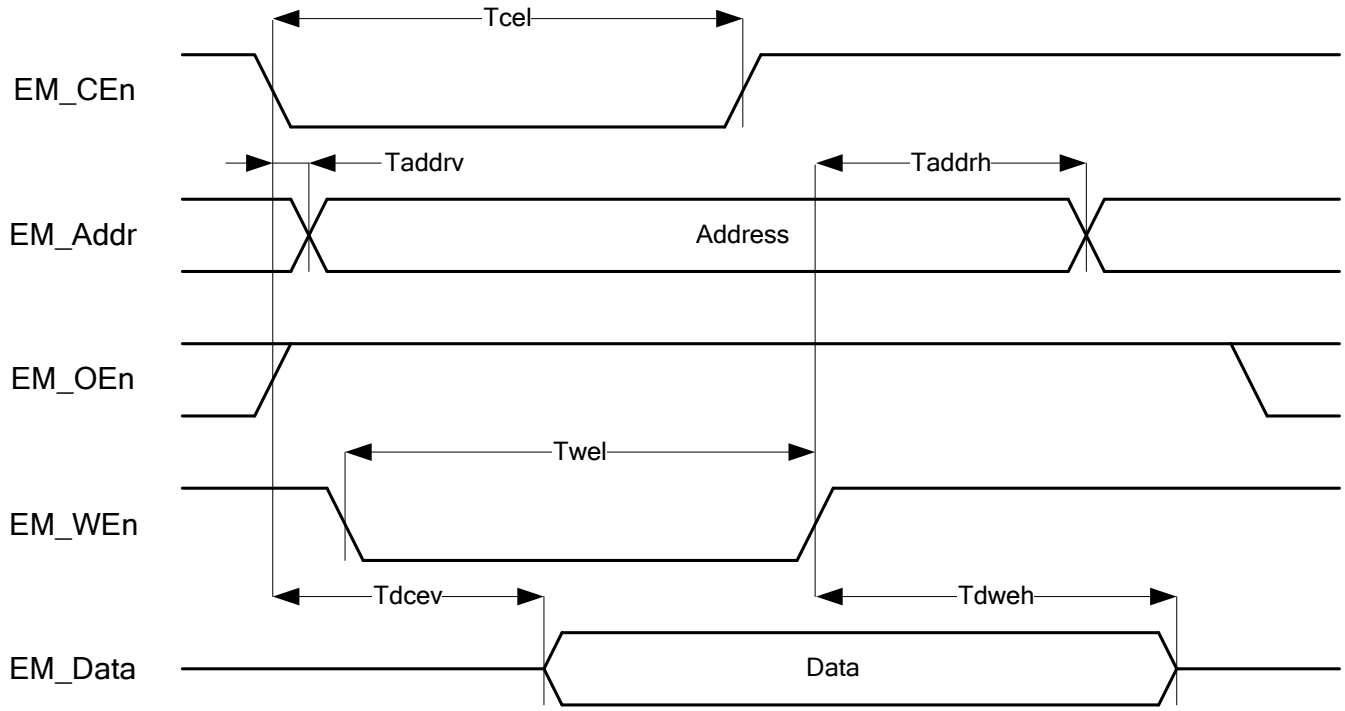
- f (EMIF_clock) = f (BUS_CLK) if BUS_CLK <= 33MHz.
- f (EMIF_clock) = 1/2*f (BUS_CLK) if 33MHz <= f(BUS_CLK) <= 66MHz.
- f (EMIF_clock) = 1/3* f (BUS_CLK) if BUS_CLK frequency is >= 67MHz (PSoC 5LP only).

Figure 4. Asynchronous Read Cycle Timing



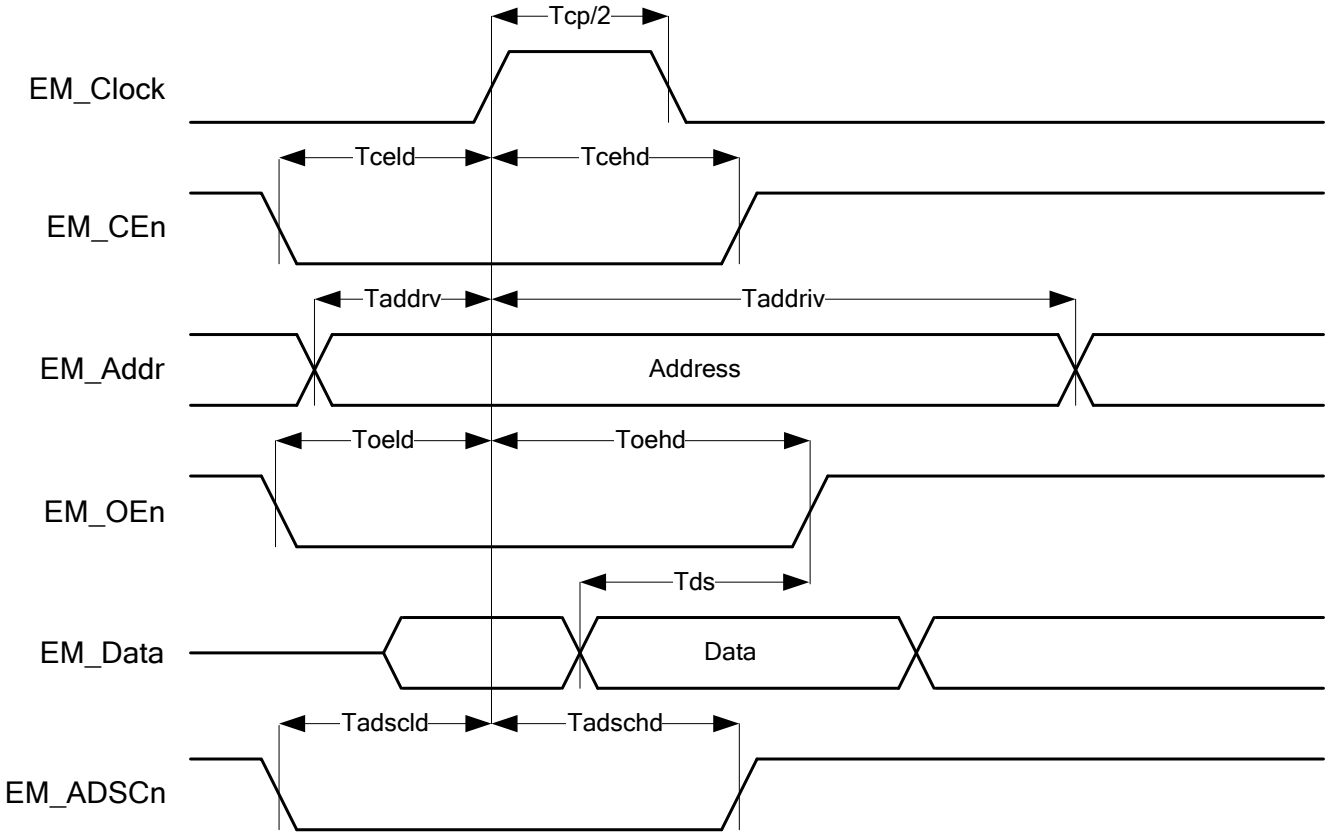
Symbol	Description	Conditions	Min	Typ	Max	Unit
T	EMIF clock period	$V_{DDA} \geq 3.3$ V	30.3	-	-	ns
Tcel	EM_CEn low time		2T – 5	-	2T + 5	ns
Taddrv	EM_CEn low to EM_Addr valid		-	-	5	ns
Taddrh	Address hold time after EM_WEn high		T	-	-	ns
Toel	EM_OEn low time		2T – 5	-	2T + 5	ns
Tdoesu	Data to EM_OEn high setup time		T + 15	-	-	ns
Tdoeh	Data hold time after EM_OEn high		3	-	-	ns

Figure 5. Asynchronous Write Cycle Timing



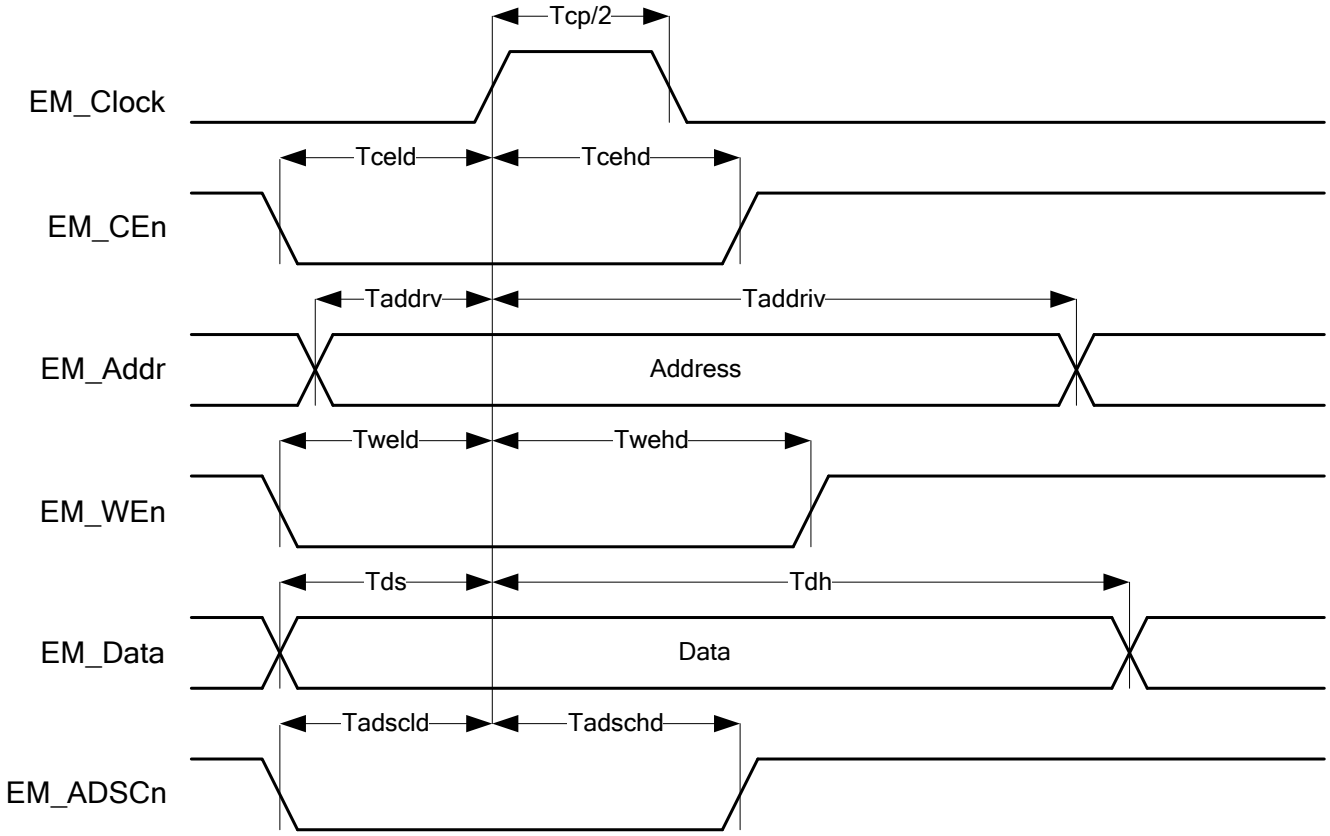
Symbol	Description	Conditions	Min	Typ	Max	Unit
T	EMIF clock period	$V_{DDA} \geq 3.3$ V	30.3	-	-	ns
Tcel	EM_CEn low time		T – 5	-	T + 5	ns
Taddrv	EM_CEn low to EM_Addr valid		-	-	5	ns
Taddrh	Address hold time after EM_WEn high		T	-	-	ns
Twel	EM_WEn low time		T – 5	-	T + 5	ns
Tdcev	EM_CEn low to data valid		-	-	7	ns
Tdweh	Data hold time after EM_WEn high		T	-	-	ns

Figure 6. Synchronous Read Cycle Timing



Symbol	Description	Conditions	Min	Typ	Max	Unit
T	EMIF clock period	$V_{DDA} \geq 3.3$ V	30.3	-	-	ns
Tcp/2	EM_Clock pulse high		T/2	-	-	ns
Tceld	EM_CEn low to EM_Clock high		5	-	-	ns
Tcehd	EM_Clock high to EM_CEn high		T/2 – 5	-	-	ns
Taddrv	EM_Addr valid to EM_Clock high		5	-	-	ns
Taddriv	EM_Clock high to EM_Addr invalid		T/2 – 5	-	-	ns
Toeld	EM_OEn low to EM_Clock high		5	-	-	ns
Toehd	EM_Clock high to EM_OEn high		T	-	-	ns
Tds	Data valid before EM_OEn high		T + 15	-	-	ns
Tadscld	EM_ADSCn low to EM_Clock high		5	-	-	ns
Tadschd	EM_Clock high to EM_ADSCn high		T/2 – 5	-	-	ns

Figure 7. Synchronous Write Cycle Timing



Symbol	Description	Conditions	Min	Typ	Max	Unit
T	EMIF clock period	$V_{DDA} \geq 3.3$ V	30.3	-	-	ns
Tcp/2	EM_Clock pulse high		T/2	-	-	ns
Tceld	EM_CEn low to EM_Clock high		5	-	-	ns
Tcehd	EM_Clock high to EM_CEn high		T/2 – 5	-	-	ns
Taddrv	EM_Addr valid to EM_Clock high		5	-	-	ns
Taddriv	EM_Clock high to EM_Addr invalid		T/2 – 5	-	-	ns
Tweld	EM_WEn low to EM_Clock high		5	-	-	ns
Twehd	EM_Clock high to EM_WEn high		T/2 – 5	-	-	ns
Tds	Data valid before EM_Clock high		5	-	-	ns
Tdh	Data invalid after EM_Clock high		T	-	-	ns
Tadscld	EM_ADSCn low to EM_Clock high		5	-	-	ns
Tadschd	EM_Clock high to EM_ADSCn high		T/2 – 5	-	-	ns

Component Changes

Version	Description of Changes	Reason for Changes / Impact
1.20.a	Minor datasheet edit.	
1.20	Added update for PSoC 3/5LP	
1.10 b	Updated EMIF data sheet with EMIF memory map info	
1.10	Added PSoC 5LP support	

© Cypress Semiconductor Corporation, 2012-2015. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

CyDesigner™, Programmable System-on-Chip™, and PSoC Express™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.