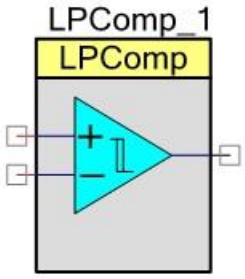
Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the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

Low Power Comparator (LPComp_PDL)

1.10



Features

- Low input offset
- Internal reference voltage
- Multiple speed modes
- Low-power mode
- Wake from low power modes
- Multiple interrupt and output modes

General Description

The Low Power (LP) Comparator PDL (LPComp_PDL) Component provides access to the low power comparators implemented using the fixed function LP comparator block that is present in PSoC 6.

The LPComp_PDL Component is a graphical configuration entity built on top of the LPComp driver available in the Peripheral Driver Library (PDL). It allows schematic-based connections and hardware configuration as defined by the Component Configure dialog.

When to Use a LPComp_PDL

The main purpose of these comparators is to offer fast analog signal comparison of internal and external analog signals in all system power modes. The connection options are described as follows:

- Compare two voltages on external pins.
- Compare a voltage from an external pin against an internally generated analog signal through AMUXBUS.
- Compare a voltage from an external pin against the local voltage reference.
- Compare two internal voltages through AMUXBUS-A/AMUXBUS-B.

The comparator output value can be inspected by the CPU. The comparator interrupt output signal is ORed together with interrupt output signal from the other low power comparator on the device as combined interrupt source. Interrupt is not cleared automatically. It is user

responsibility to do that. This combined signal is available as the output of the global signal reference Component and can be used as an interrupt/wakeup source from deep-sleep mode. Refer to the Operation in Low Power Mode section for more details.

In PSoC 6 devices, the comparator output is available directly, or after synchronous edge detection as level or pulse. It can be used to trigger an interrupt, routed to digital logic, or sent to a pin. Refer to the Functional Description section for details.

Quick Start

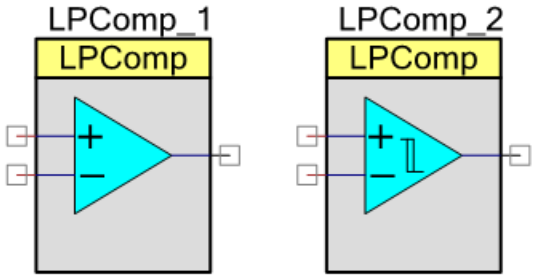
1. Drag a LPComp_PDL Component from Component Catalog Analog/Comparators folder onto your schematic (the placed instance takes the name LPComp_1).
2. Drag a Analog Pin from the Component Catalog Cypress/Ports and Pins folder onto your schematic (the placed instance takes the name Pin_1). Connect it to the positive LPComp_1 input.
3. Drag a Digital Output Pin from the Component Catalog Cypress/Ports and Pins folder onto your schematic (the placed instance takes the name Pin_2). Connect it to the LPComp_1 output.
4. Build the project in order to verify the correctness of your design, add the required PDL modules to the Workspace Explorer, and generate the configuration data for the LPComp_1 instance.
5. In the *main.c* file, initialize the peripheral and start the application:

```
#include "project.h"

int main(void)
{
    /* Configure LPComp */
    Cy_LPComp_Init(LPComp_1_HW, LPComp_1_CHANNEL, &LPComp_1_config);

    /* Set channel 0 power mode - Ultra Low Power mode */
    Cy_LPComp_SetPower(LPComp_1_HW, LPComp_1_CHANNEL,
CY_LPCOMP_MODE_ULP);

    /* Set the local reference voltage to the negative terminal   and set
a GPIO input on the positive terminal for the input signal */
    Cy_LPComp_SetInputs(LPComp_1_HW, LPComp_1_CHANNEL, CY_LPCOMP_SW_GPIO,
CY_LPCOMP_SW_LOCAL_VREF);

    for(;;)
    {
        /* Place your application code here. */
    }
}
```

6. Build and program the device.

Input/Output Connections

This section describes the various input and output connections for the LPComp_PDL. The symbol for this Component is annotated to denote the selection of hysteresis.



Positive Input – Analog Input

This input is usually connected to the voltage that is being compared. This input can be routed from a GPIO or from an internal source. When connected to an internal source the GPIO that is dedicated to this input will be consumed and not available for other uses. Also this input can be connected to AMUXBUSA/ AMUXBUSB.

Negative Input – Analog Input

This input is usually connected to the reference voltage. This input can be routed from a GPIO or from an internal source. When connected to an internal source the GPIO that is dedicated to this input will be consumed and not available for other uses. Also this input can be connected to AMUXBUSA/AMUXBUSB or Vref.

Comparator Out – Output

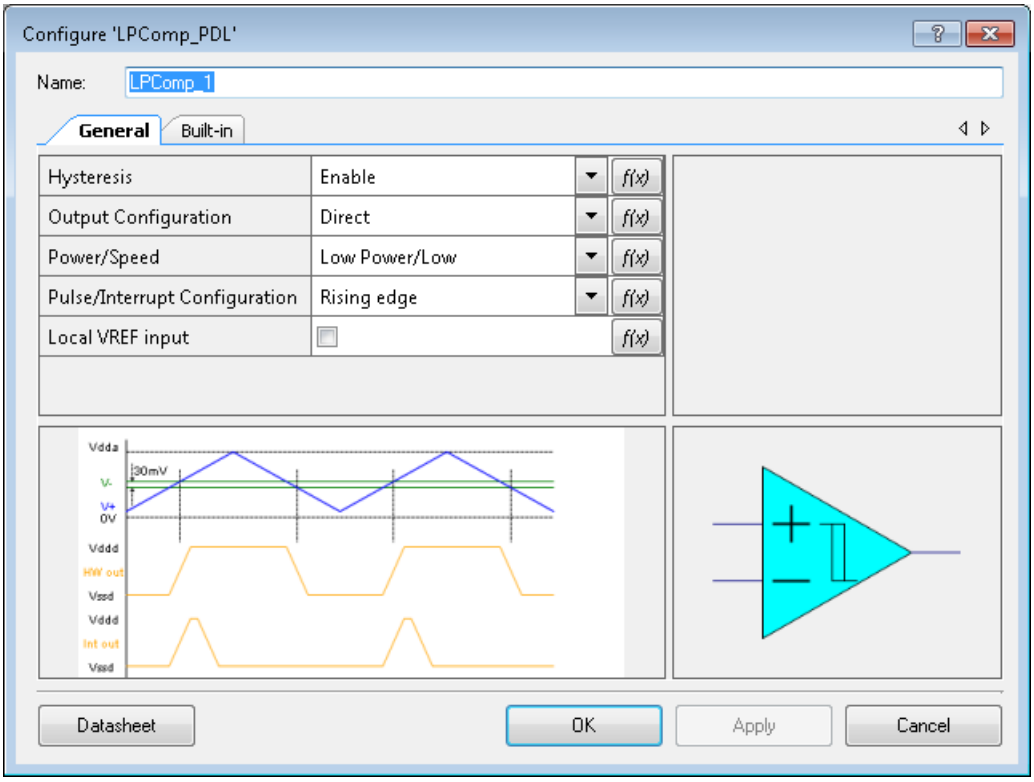
The Component output is configurable (refer to the Output Configuration section): either direct comparator out or after synchronous edge detection as level or pulse. It can be used to trigger an interrupt, routed to digital logic or sent to a pin. Refer to the Functional Description section for details.

Note This output cannot be used to wake up a device from deep-sleep mode. Global Signal Reference interrupt signal must be used instead.

Component Parameters

Drag a LPComp_PDL Component onto your design and double click it to open the Configure dialog. This dialog has the following tabs with different parameters.

General Tab



The LPComp_PDL provides the following parameters.

Output Configuration

This parameter defines mode of the LPComp_PDL output: Direct, Synchronized, or Pulse. Refer to the Functional Description section for details.

Hysteresis

This parameter allows you to add approximately 30 mV of hysteresis to the LPComp_PDL. This helps to ensure that slowly moving voltages or slightly noisy voltages will not cause the output of the LPComp_PDL to oscillate when the two input voltages are nearly equal.

Pulse/Interrupt Configuration

This parameter defines the event that will cause a pulse to be generated on the interrupt terminal or direct comparator output. The parameter allows you to select interrupt/output mode: Disabled, Rising edge, Falling edge, or Both edges.

Power / Speed

This parameter provides a way to optimize speed verses power consumption. The Speed/Power parameter allows you to select the speed/power level: Ultra low Power/Slow, Low Power Power/Low, Normal Power/Fast.

Local VREF input

This parameter enables the internal reference voltage. Checked value hides the negative terminal (inverting input) to prevent any external connections.

Application Programming Interface

The Application Programming Interface (API) is provided by the LPComp driver module from the PDL. The driver is copied into the “pdl\drivers\peripheral\lpcomp\” directory of the application project after a successful build.

Refer to the PDL documentation for a detailed description of the complete API. To access this document, right-click on the Component symbol on the schematic and choose the “**Open PDL Documentation…**” option in the drop-down menu.

The Component generates the configuration structures and base address described in the Global variables section. Pass the generated data structure and the base address to the associated LPComp driver function in the application initialization code to configure the peripheral. Once the peripheral is initialized, the application code can perform run-time changes by referencing the provided base address in the driver API functions.

By default, PSoC Creator assigns the instance name LPComp_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.

Global Variables

The LPComp_PDL Component populates the following peripheral initialization data structure(s). The generated code is placed in C source and header files that are named after the instance of the Component (e.g. LPComp_1.c). Each variable is also prefixed with the instance name of the Component.

cy_stc_lpcomp_config_t LPComp_1_config

The instance-specific configuration structure. This should be used in the Init() function.

Data in RAM

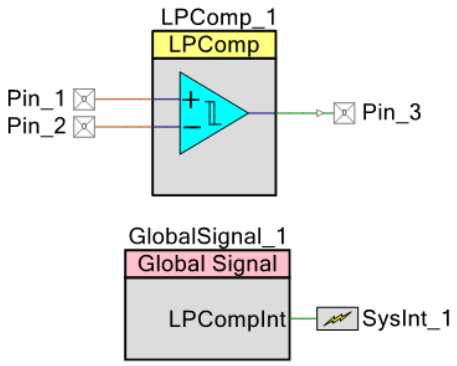
The generated data may be placed in flash memory (const) or RAM. The former is the more memory-efficient choice if you do not wish to modify the configuration data at run-time. Under the Built-In tab of the Configure dialog set the parameter CONST_CONFIG to make your selection. The default option is to place the data in flash.

Interrupt Service Routine

The LPComp_PDL supports interrupts on the various events, depends on *Pulse/Interrupt configuration* settings: Rising edge, Falling edge or Both edges. Interrupt signal goes high when any of the enabled interrupt configurations are true.

Note Interrupt is not cleared automatically. It is user responsibility to do that. Interrupt is cleared by writing a '1' in corresponding interrupt register bit position. The preferred way to clear interrupt sources is usage Cy_LPComp_ClearInterrupt() API.

The Global Signal Component should be used to access to the LPComp_PDL interrupt:



The following code is suggested:

```
#include "project.h"

void GlobalSignal_ISR_Interrupt_InterruptCallback(void)
{
    /* Interrupt does not clear automatically.
    * It is user responsibility to do that.
    */
    Cy_LPComp_ClearInterrupt(LPComp_1_HW, LPComp_1_INTR);

    /* Clear pending IRQ interrupt */
    NVIC_ClearPendingIRQ(SysInt_1_cfg.intrSrc);

    /*
    * Add user interrupt code to manage interrupt.
    */
}
```

```
int main(void)
{
    /* Interrupt handler initialization */
    Cy_SysInt_Init(&SysInt_1_cfg, GlobalSignal_ISR_Interrupt_InterruptCallback);
    NVIC_EnableIRQ(SysInt_1_cfg.intrSrc);

    /* Configure LPComp */
    Cy_LPComp_Init(LPComp_1_HW, LPComp_1_CHANNEL, &LPComp_1_config);

    /* Set channel 0 power mode - Ultra Low Power mode */
    Cy_LPComp_SetPower(LPComp_1_HW, LPComp_1_CHANNEL, CY_LPCOMP_MODE_ULP);

    /* Configure the LPComp interrupt*/
    Cy_LPComp_SetInterruptMask(LPComp_1_HW, LPComp_1_INTR_MASK);
    Cy_LPComp_SetInterruptTriggerMode(LPComp_1_HW, LPComp_1_CHANNEL,
CY_LPCOMP_INTR_RISING);

    /* Enable global interrupts. */
    __enable_irq();

    for(;;)
    {
        /* Place your application code here. */

    }
}
```

Code Examples and Application Notes

PSoC Creator provides access to code examples in the 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 "Code Example" topic in the PSoC Creator Help for more information.

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. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

Refer to **PSoC Creator Help > Building a PSoC Creator Project > Generated Files (PSoC 6)** for information on MISRA compliance and deviations of the files generated by PSoC Creator.

The LPComp_PDL Component has the following specific deviations:

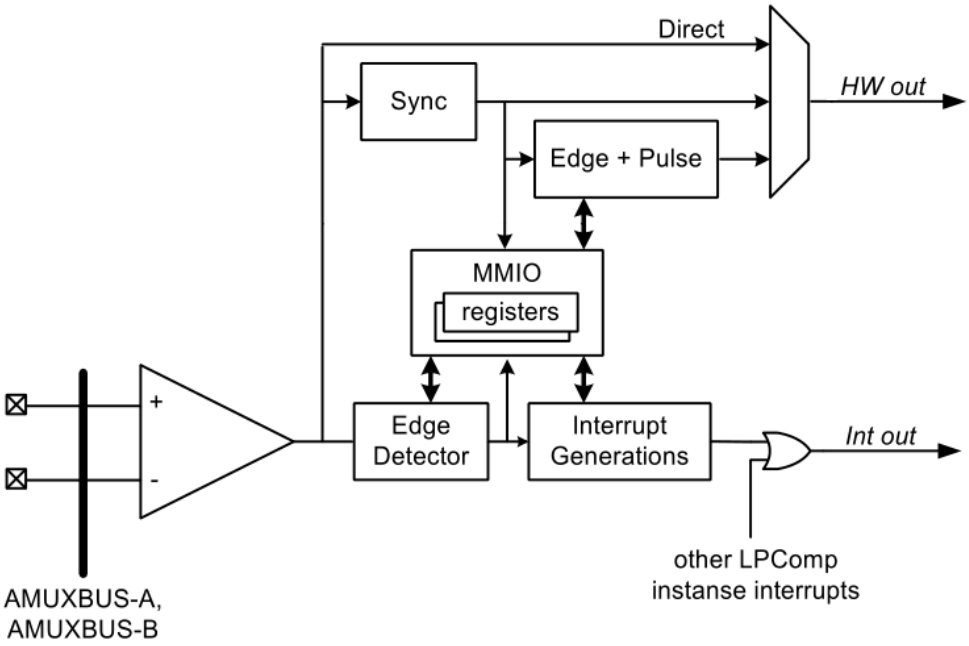
MISRA-C: 2004 Rule	Rule Class (Required/Advisory)	Rule Description	Description of Deviation(s)
1.1	R	The keyword 'inline' has been used.	Deviated since INLINE functions are used to allow more efficient code.
3.1, 11.3	A	Cast between a pointer and an integral type.	The cast from unsigned int to pointer does not have any unintended effect, as it is a consequence of the definition of a structure based on hardware registers.

Functional Description

The LPComp_PDL Component is intended to be operational in all power modes, including deep-sleep/hibernate mode. The main purpose of this Component is to offer fast detection capability in normal operating modes and ultra-low power operation in hibernate mode.

Block Diagram and Configuration

The following is a high-level block diagram.



In PSoC 6 devices, each comparator has one HW output. It comes from three sources:

- Direct comparator output

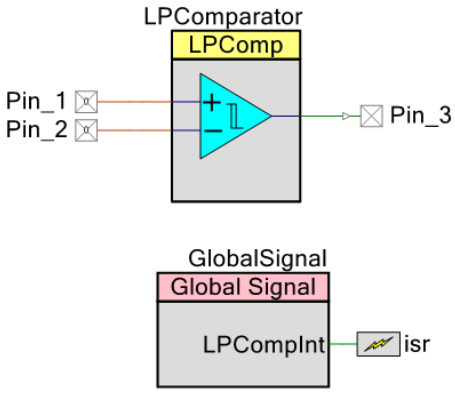
- Synchronized to SYSCLK using two flip-flops
- Edge detected pulse which have period 2*SYSCLK.

Individual comparator interrupt outputs are ORed together as single asynchronous interrupt source before sent out and used to wakeup system in low power mode. For PSoC 6 devices, the individual comparator interrupt is masked by INTR_MASK. The masked result is captured in INTR_MASKED register. Writing 1 to INTR register bit will clear the interrupt. Refer to the appropriate device *Technical Reference Manual (TRM)* for a detailed description of the registers.

Low-Power Modes

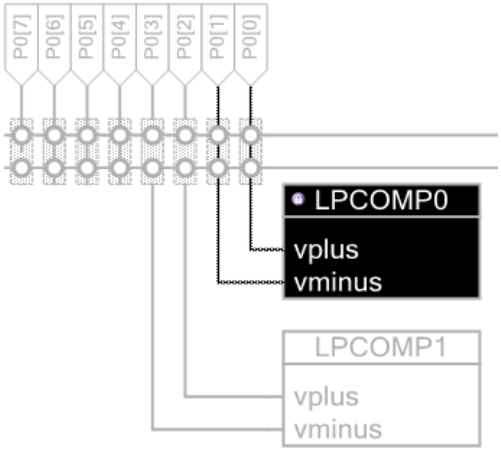
Deep Sleep and Hibernate modes are possible for the IP block instances that support Low Power. Refer to the device *Technical Reference Manual (TRM)* for details.

The LPComp_PDL Component operates in Deep Sleep/Hibernate mode **only if the Speed/Power option is set to Slow/Ultra low**. It can be used along with the Global Signal Component as a wake-up source from Deep Sleep mode. The following schematics can be used.



Select the **Combined low power comparator interrupt (LPCompInt)** option as a source for an interrupt in the Global Signal Reference Component. This Component triggers the output each time any of the enabled low power comparators generates an interrupt.

The LPComp_PDL Component input pins can be assigned to dedicated inputs or to AMUXBUSA/ AMUXBUSB. Refer to the device datasheet for the part being used for the specific physical pin connections.



Also the negative input can be connected to the internal Vref (0.45V-0.75V).

The wakeup event is when the voltage connected to the positive input (Pin_1) is greater than the negative input voltage (Pin_2).

The LPComp driver provides callback functions to handle power mode transition. These functions must be registered using the cy_syspm driver before entering Deep Sleep or Hibernate mode appropriately. Refer to the Low Power section of the LPComp driver, or the System Power Management section of the PDL Documentation for more details about callback registration.

Placement

Each comparator is directly connected to specific GPIOs for its inputs. The output connection is routed to the digital fabric. Refer to the device datasheet for the part being used for the specific physical pin connections.

Registers

See the chip Technical Reference Manual (TRM) for more information about the registers.

Resources

This Component uses one of the LP Comparators from the pair of comparators in the LP Comparator hardware block.

DC and AC Electrical Characteristics

Note Final characterization data for PSoC 6 devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site.

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.10.b	Update the datasheet.	Added text about callback functions in the Low-Power Modes section.
1.10.a	Update the datasheet.	Updated Quick Start section and code example.
1.10	New Component version.	Added power and intType members to cy_stc_lpcomp_config_t structure in the PDL library. Added Cy_LPComp_Enable() and Cy_LPComp_Disable() functions to the LPComp driver in the PDL library.
1.0.b	Updated MISRA section.	PSoC 6 generated files have a MISRA section in the PSoC Creator Help.
1.0.a	Update the datasheet.	Updated Quick Start section and code example.
1.0	New Component	

© Cypress Semiconductor Corporation, 2017. 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.