System Interrupt (SysInt)
1.0

Features



- Generating interrupts from hardware signals
- Assigning interrupts to a CPU core
- Configuring interrupt priority
- Interrupt vectoring and control

General Description

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

The SysInt Component provides an interface to connect hardware signals to a CPU interrupt request line. The Design-Wide Resources Interrupt Editor allows you to configure the interrupt priority, CPU core, and interrupt request line to which the interrupt is routed.

When to Use a SysInt Component

The Component should be used when a CPU interrupt needs to be triggered based on a signal from an interrupt source, for example a GPIO signal or an ADC end of conversion.

Input/Output Connections

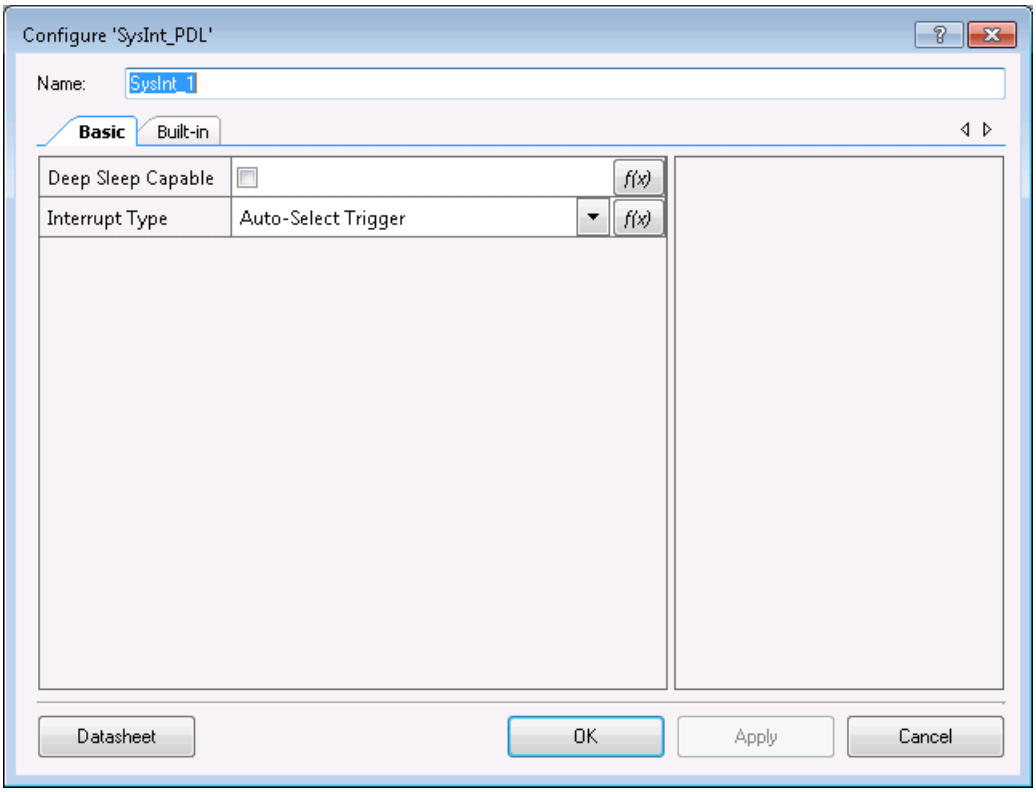
This section describes the various input and output connections for the SysInt Component.

int_signal – Input

Connect the signal that generates the interrupt to this input. When the signal value becomes logic high, the interrupt is triggered. For a Level type interrupt, the interrupt will continue to trigger as long as the signal remains logic high.

Component Parameters

Drag a SysInt Component onto your design and double-click it to open the **Configure** dialog.



The SysInt Component provides the following parameters:

Deep Sleep Capable

Determines whether the interrupt will be assigned to a device resource that is capable of operating in chip Deep Sleep power domain. Certain interrupt lines (and hence the resources associated with the interrupt lines) are Deep Sleep capable, whereas others are not. For resources that can be both (e.g., SCB8 may be Deep Sleep capable, whereas others are not), the **Deep Sleep Capable** parameter can be used to allow PSoC Creator to automatically assign based on the intended power domain of operation.

- **False** – Default
- **True**

Interrupt Type

This parameter configures which type of waveform the Component will process to trigger the interrupt. There are three possible values for this parameter:

- **Auto-Select Trigger** – This is the default setting. It inspects the driver of the **int_signal** and infers the interrupt type based on the signal source. For most fixed-function blocks, the interrupt type is LEVEL. For UDB signal sources, the interrupt type is RISING_EDGE.
- **Rising-Edge Triggered** – Triggers the interrupt on the rising edge of the source signal. This type of connection uses UDB resources.
- **Level Triggered** – Selects the source connected to the interrupt as a level-sensitive connection. All fixed function peripherals are level triggered.

As a guideline, use RISING_EDGE when capturing a signal change (for example, periodic clock), and use LEVEL when capturing a state change of a peripheral (for example, FIFO fill levels).

For Component-specific interrupt usage information, refer to the specific Component datasheet.

Application Programming Interface

Application Programming Interface (API) is provided by the the cy_sysint driver module from the PDL. The driver is copied into the “pdl\drivers\peripheral\sysint\” 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.

PSoC Creator generates a list of # defines and initial configuration structures in the *cyfitter_sysint_cfg.c* file. This is located in the “**Pins and Interrupts**” directory in the workspace explorer. Pass the generated data structure to the associated cy_sysint 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 SysInt_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.

Global Variables

The SysInt Component populates the following peripheral initialization data structure in the *cyfitter_sysint_cfg.c* file.

cy_stc_sysint _t SysInt_1_cfg

The instance-specific configuration structure. This should be passed as a parameter to the Cy_SysInt_Init() function.

Code Examples and Application Notes

This section lists the projects that demonstrate the use of the Component.

Code Examples

PSoC Creator provides access to code examples 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.

There are also numerous code examples that include schematics and example code available online at the Cypress Code Examples web page.

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

The SysInt Component does not have any specific deviations.

Functional Description

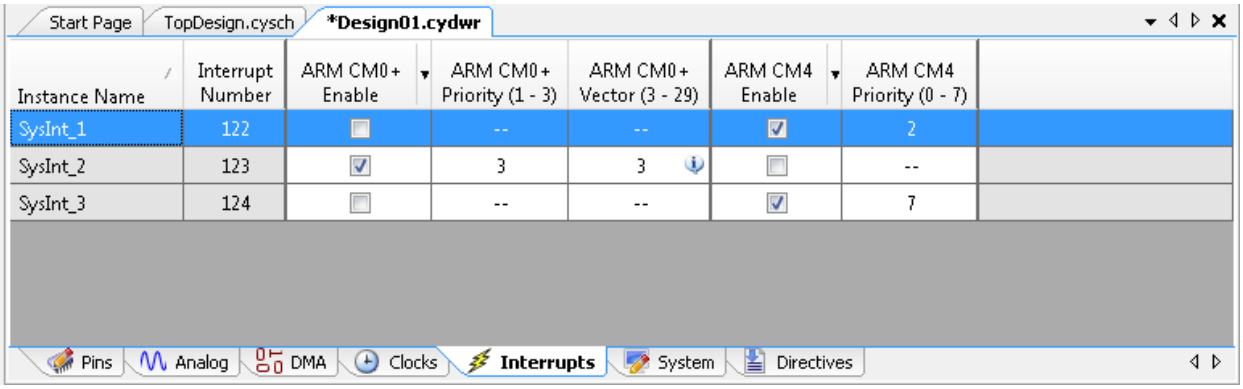
In addition to the fixed-function peripherals, any data signal in the UDB array routing can be used to generate an interrupt.

Depending on your choice of Interrupt type, several scenarios are possible.

Source	Interrupt Type	Routing	Description
Fixed-function block interrupt	Level-Triggered	Direct	Interrupt is a dedicated connection to the interrupt controller.
	Auto-Select Trigger	Direct	Interpreted as Level-Triggered.
UDB block interrupt	Level-Triggered	UDB	UDB based Components frequently use Level-Triggered. These are routed directly from the UDB array to the interrupt controller.
	Rising-Edge Triggered	UDB	The signal can route through an edge-detect circuit to the interrupt controller, if the interrupt is set as Rising-Edge Triggered.
	Auto-Select Trigger	UDB	Interpreted as Rising-Edge Triggered.
Signal on Schematic	Level-Triggered	UDB	This mode is not often used. The signal is routed through the UDB array to the interrupt controller.
	Rising-Edge Triggered	UDB	Used to detect a signal transition. Routes through edge-detect to the interrupt controller.
	Auto-Select Trigger	UDB	Interpreted as Rising-Edge Triggered

Design-Wide Resources

The use of a SysInt Component in a design results in an entry in the Design-Wide Resources Interrupt Editor, which contains the following parameters:



- **Instance Name** – Shows the Component instance names in your design. This is a read-only parameter.
- **Interrupt Number** – Indicates the interrupt vector numbers connected to the NVIC. This is read-only parameter for all cores except for CM0+. For CM0+ cores, this parameter indicates the mux feeding into the CM0+ NVIC.

- **ARM CMx Enable** – Allows allocation of the interrupt source to the core's NVIC.
- **ARM CMx Priority** – Shows and allows you to set the instance's priority, where 0 is the highest priority. The range depends on the core and priority bits used.
- **ARM CM0+ Vector** – (Specific for CM0+ cores). Allows assignment of the device interrupts to the 27 available input muxes feeding into the CM0+ NVIC.

  Note The CM0+ core has 32 muxes; 4 of which are reserved for internal use.

Resources

Each SysInt Component consumes one entry in the device interrupt vector table.

More Information

Refer to the device Technical Reference Manual (TRM) for more information on the interrupt architecture.

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.c	Minor datasheet edits.	
1.0.b	Updated note about PSoC Creator 4.1 limitations.	Not relevant in PSoC Creator 4.2 or later
	Updated information regarding available interrupts on the CM0+ core.	PSoC Creator reserves NVIC slots 0, 1, 2, 30 and 31 on the CM0+ core.
1.0.a	Updated Interrupt DWR information	Implementation change
	Expanded API section with usage detail	Better usability
1.0	New Component	

© Cypress Semiconductor Corporation, 2016-2017. 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.

