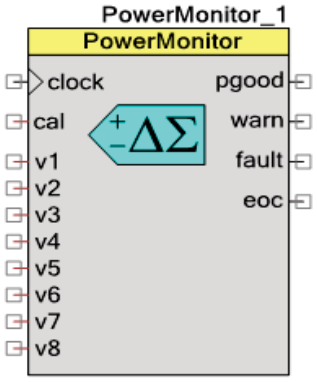
Power Monitor
1.30



Features

- Interfaces to up to 32 DC-DC power converters
- Measures power converter output voltages and load currents using a DelSig-ADC
- Monitors the health of the power converters generating warnings and faults based on user-defined thresholds
- Support for measuring other auxiliary voltages in the system
- Support 3.3V and 5V chip power supply

General Description

Power Converter Voltage Measurements:

For power converter voltage measurements, the ADC can be configured into single-ended mode (0-4.096 V range or 0-2.048 V range). The ADC can also be configurable into differential mode (±2.048 V range) to support remote sensing of voltages where the remote ground reference is returned to PSoC over a PCB trace. In cases where the analog voltage to be monitored equals or exceeds Vdda or the ADC range, external resistor dividers are recommended to scale the monitored voltages down to an appropriate range.

Power Converter Current Measurements:

For power converter load current measurements, the ADC can be configured into differential mode (+/- 64 mV or +/- 128 mV range) to support voltage measurement across a high-side series shunt resistor on the outputs of the power converters. Firmware APIs convert the measured differential voltage into the equivalent current based on the external resistor component value used. The ADC can also be configured into single-ended mode (0-4.096V range or 0-2.048 V range) to support connection to external current sense amplifiers (CSAs) that convert the differential voltage drop across the shunt resistor into a single ended voltage or to support power converters or hot-swap controllers that integrate similar functionality.

Auxiliary Voltage Measurements:

Up to 4 auxiliary input voltages can be connected to the ADC to measure other system inputs. The ADC can be configured into single ended mode (0-4.096 V / 0-2.048V) or differential mode (+/- 2.048 V or +/- 64 mV / +/-128mV ranges) to measure the auxiliary input voltages.

ADC Sequential Scanning:

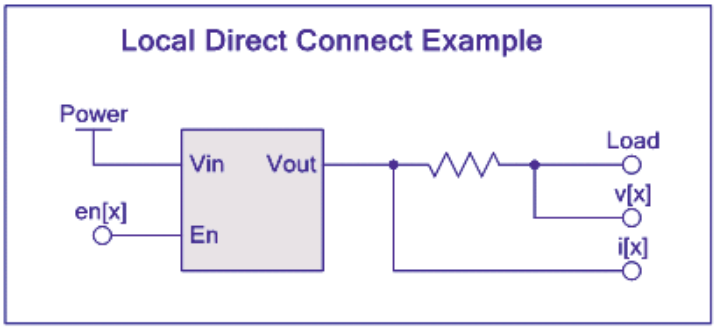
The ADC will sequence through all power converters and auxiliary inputs, if enabled, in a round-robin fashion, taking voltage measurements and load current measurements. This component will measure the voltages of all the power converters in the system, but can be configured to measure currents from a subset of the power converters – including no current measurements at all. Doing so will minimize the number of IOs required and will minimize the overall ADC scan time.

This component needs some knowledge of components external to PSoC for 2 reasons:

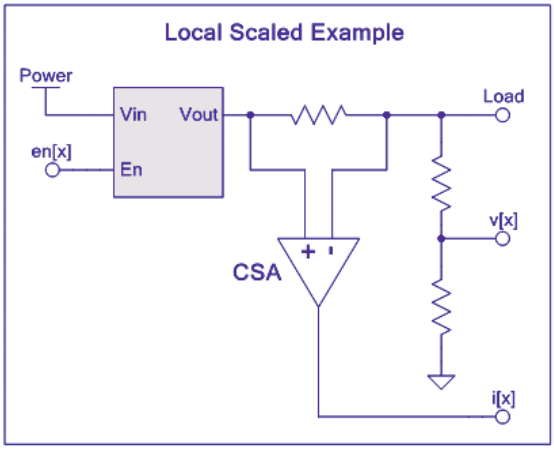
- Scaling factors for input voltages that have been attenuated to meet IO input range limits or ADC dynamic range limits where applicable
- Scaling factors for current measurements (series resistor, series inductor or CSA gain etc.)

Component Use Cases:

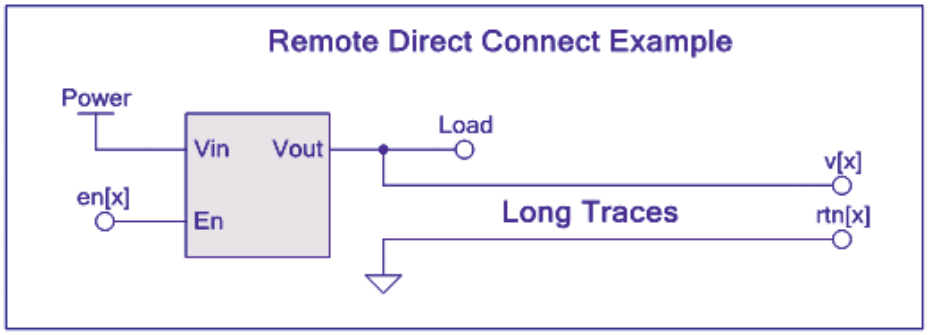
The diagram below shows the connection methodology for a power converter that has an output voltage < Vdda. The voltage sense and current sense points are taken from either side of the sense resistor and can connect directly to this component.



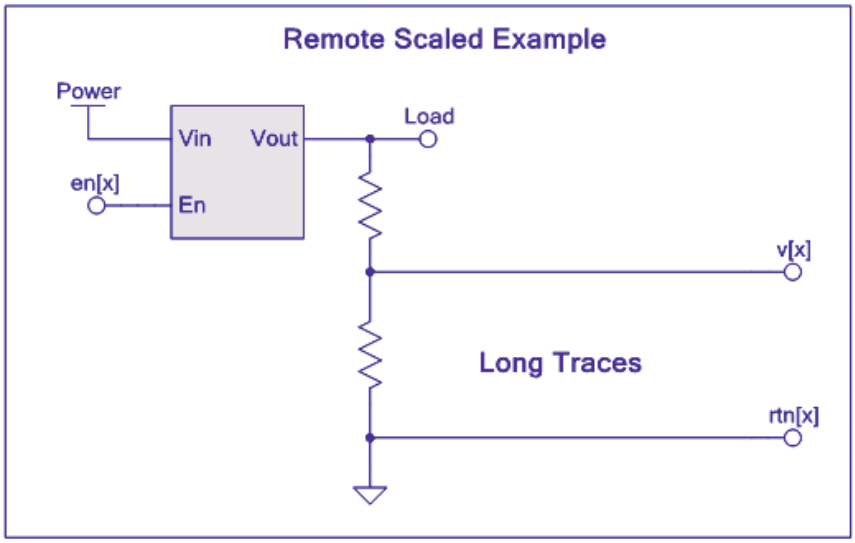
The diagram below shows the connection methodology for a power converter that has an output voltage > Vdda. An external current sense amplifier (CSA) is required to convert the differential voltage drop across the sense resistor to a single-ended voltage that connects directly to this component. The voltage sense point is scaled down to a voltage level that can directly connect to this component.



The diagram below shows the connection methodology for a remote power converter that has an output voltage < 2.048V where the remote voltage sense point and the remote ground reference are both routed back to this component.



The diagram below shows the connection methodology for a remote power converter that has an output voltage > Vdda where the remote voltage sense point is scaled using resistors and is routed back to this component along with the remote ground reference point.



Input/Output Connections

This section describes the various input and output connections for the Power Monitor. 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.

clock – Digital Input

The clock input signal is used to drive all digital output signals. The maximum frequency used for this clock is 67MHz.

cal – Analog Input *

The cal input is the calibration voltage input for calibration of the 64mV or 128mV differential voltage ADC range setting. This signal is an optional input connection. When the "cal" pin is exposed, a POR calibration occurs automatically as part of PowerMonitor_Start() API to calibrate 64mV or 128mV differential voltage ADC range. For subsequent calibrations to occur at run time, PowerMonitor_Calibrate() API should be used.

Note The input voltage given to this pin should not exceed 100% of differential ADC range (either 64mV range or 128mV range) used.

v[x] – Analog Input

The v[x] are analog inputs that connect to the power converter output voltage as seen by their loads. This could be a direct connection to the power converter output, or a scaled version using external scaling resistors. Every power converter will have voltage measurement enabled. The

component supports a maximum of 32 voltage input terminal pins and the unused terminals are hidden.

i[x] – Analog Input *

The i[x] are analog inputs that enable this component to measure power converter load currents. This could be a differential voltage measurement across a shunt resistor along with the corresponding v[x] input or could be a single-ended connection to an external CSA. Current monitoring is optional on a power converter by power converter basis. When differential v[x] voltage measurement is selected for a power converter in the component customizer, current measurement is disabled for that power converter in order to limit the number of IOs used by this component. In that case, the i[x] terminal is replaced by the rtn[x] terminal representing the differential voltage measurement return path.

This component supports a maximum of 24 current input terminals and the unused terminals are hidden. These terminals are mutually exclusive with the associated rtn[x] input terminals.

rtn[x] – Analog Input *

The rtn[x] analog inputs connect to a ground reference point that is physically close to the power converter. These terminals are only exposed when differential voltage sensing is enabled for that power converter in the component customizer. These terminals are mutually exclusive with the associated i[x] input terminals. Unused pins are hidden.

aux[x] – Analog Input *

Since this component embeds the only available DelSig ADC converter, the aux[x] analog inputs enable users to connect other auxiliary voltage inputs for measurement by the ADC. Up to 4 auxiliary input terminals are available and these terminals will be hidden if the user does not enable auxiliary input voltage monitoring in the component customizer.

aux_rtn[x] – Analog Input *

These analog inputs can connect to the auxiliary input voltage ground reference point. Up to 4 aux_rtn[x] terminals are available. These terminals will be hidden if the user does not enable auxiliary input differential voltage monitoring in the component customizer.

eoc – Output

This digital output signal is an active high pulse indicating ADC conversion complete for the current sample set. This terminal is pulsed high for one clock cycle when one ADC measurement has been taken from every analog input (voltages, currents and auxiliary). Users can use this signal to generate an application-specific interrupt to the MCU core or to drive other hardware in their schematic. One simple example might simply be to connect it to a pin to measure the ADC update rate for all the inputs. Another example might be to use the signal to run custom firmware filtering algorithms once all samples are gathered.

pgood – Output

This digital output terminal is driven active high when all power converter voltages and currents (if measured) are within a user specified operating range. The user can mask individual power converters from participating in the generation of the pgood output. An option exists in the customizer to make this terminal a bus to expose the individual pgood status outputs for each converter.

warn – Output

This digital output terminal is driven active high when one or more power converter voltages or currents (if measured) are outside the user-specified nominal range, but not by enough to be considered a fault condition. Warn pin is "sticky" (it latch HIGH) until the associated APIs are called. To clear the Warn pin, call: PowerMonitor_GetUVWarnStatus(), PowerMonitor_GetOVWarnStatus() and PowerMonitor_GetOCWarnStatus() as applicable.

fault – Output

This digital output terminal is driven active high when one or more power converter voltages or currents (if measured) are outside the user-specified nominal range to such a degree that it is considered to be a fault condition. Fault pin is "sticky" (it latch HIGH) until the associated APIs are called. To clear the Fault pin, call: PowerMonitor_GetUVFaultStatus(), PowerMonitor_GetOVFaultStatus() and PowerMonitor_GetOCFaultStatus() as applicable.

Analog Input Pin Assignment Considerations

If manual analog pin assignment is desired to simplify PCB layout, users of this component need to have some appreciation of the analog routing resources available in PSoC 3 in order to make appropriate choices. The analog routing resources are described in detail in the PSoC 3 Technical Reference Manual section 32.2. Figure 32.1 of that manual introduces the concept of “left side” vs. “right side” analog routing channels and GPIO ports. Figure 32.2 shows the detailed analog subsystem floor plan including the analog hardware blocks, most notably the DelSig ADC, and all the available analog routing channels.

Here is a summary of the routing resources as they pertain to the ADC:

1. Any GPIO input can connect to the positive terminal of the DelSig ADC
2. Only odd port pins within a given port (e.g. P0[1,3,5,7], P1[1,3,5,7] etc.) can connect to the negative terminal of the DelSig ADC

With this in mind, users of this component who wish to manually assign pins should follow this procedure to ensure a routable design:

1. Assign as many rtn[x], Direct i[x], aux_rtn[x] as you can to the left side odd port pins first: P0[1,3,5,7], P2[1,3,5,7], P4[1,3,5,7], P6[1,3,5,7], P15[5]

2. Assign any remaining rtn[x], Direct i[x], aux_rtn[x] to the right side odd port pins: P3[1,3,5,7], P5[1,3,5,7], P15[1,3], P1[1,3,5,7]
3. Assign as many v[x], aux[x], CSA i[x] as you can to the left side even port pins first: P0[0,2,4,6], P2[0,2,4,6], P4[0,2,4,6], P6[0,2,4,6], P15[4]
4. Assign any remaining v[x], aux[x], CSA i[x] to the right side even port pins: P3[0,2,4,6], P5[0,2,4,6], P15[0,2], P1[0,2,4,6]

The following notes should also be considered for optimal performance:

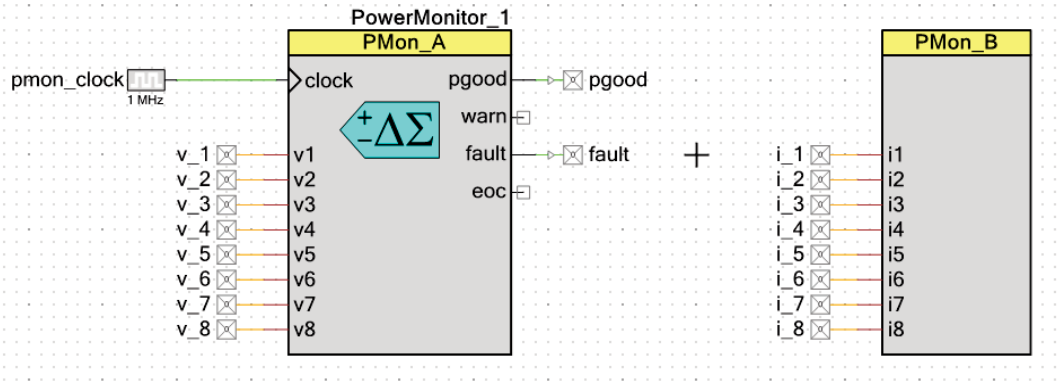
1. All P1[x] pins should be used as a last resort since that port contains the JTAG and SWD programming pins and the user will need to take that into account in their PCB design if the intention is to use those pins for digital program/test as well as for analog voltage measurements
2. When routing related differential signals, place them next to each other on adjacent pins Examples: v[x]=P0[0], rtn[x]=P0[1] or aux[x]=P4[4], aux_rtn[x]=P4[5]

Schematic Macro Information

The Power Monitor component implementation includes 3 macros shown below:

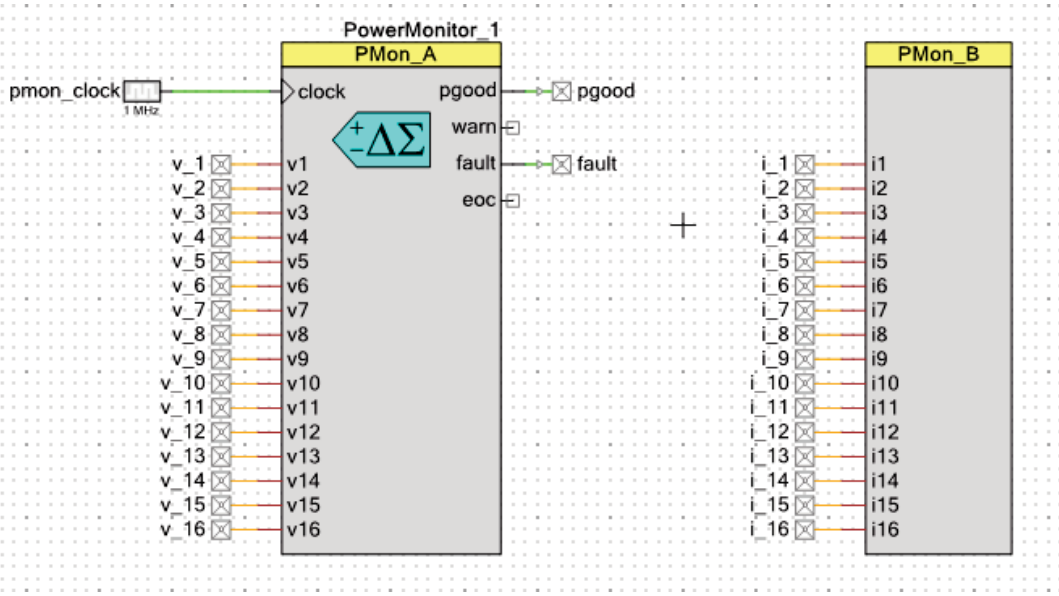
Power Monitor – 8 Rails

The macro supports 8 single-ended voltage inputs and 8 current inputs. The pgood is configured as a single bit logic level output reflecting the power good status of the system.



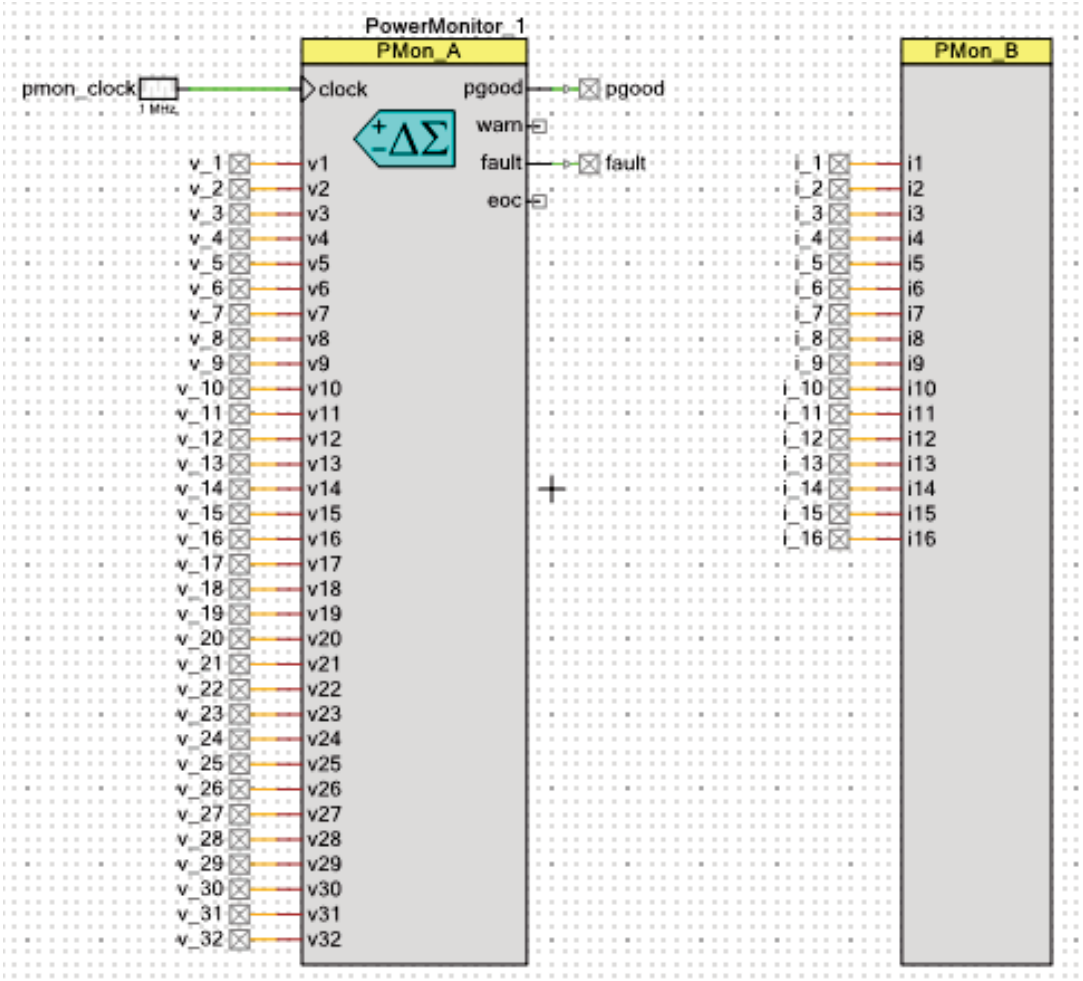
Power Monitor – 16 Rails

Many off-the-shelf Power Supervisor ASSPs support 16 secondary power converters. This macro is provided to enable users to quickly replicate that functionality. It measures 16 single-ended voltage inputs and 16 current inputs to support 16 secondary power converters. The pgood is configured as a single bit logic level output reflecting the power good status of the system.



Power Monitor – 32 Rails

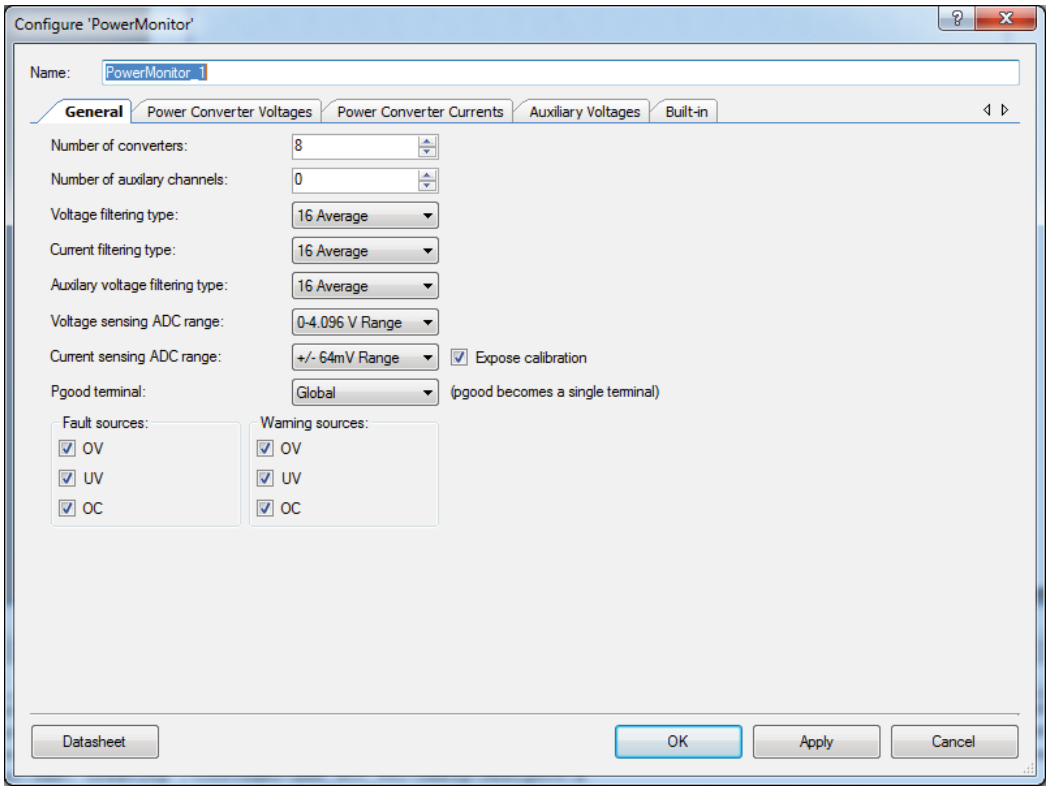
This macro is provided to enable designers to build platform solution supporting the most number of rails in the industry. The macro supports 32 single-ended voltage inputs and 16 current inputs. The pgood is configured as a single bit logic level output reflecting the power good status of the system.



Component Parameters

Drag a Power Monitor onto your design and doubleclick it to open the Configure dialog. Figure 1 shows the Configure dialog.

Figure 1.Configure Power Monitor Dialog



The component customizer UI is broken up into 4 tab pages as shown in the screenshot above.

Parameters from the General Tab

NumConverters

This parameter determines the number of converters to be monitored. The range of supported converters is 0 – 32. The default number of converters is set to **8**.

NumAuxChannels

This parameter determines the number of auxiliary voltage sources to be measured. The maximum supported auxiliary voltage sources are **4**. The default value is **0**.

AuxFilterType

The Power Monitor component supports averaging of the power converter voltage and/or load current readings. This parameter can be used to set the filter type to be applied to auxiliary voltage measurements. The average value is calculated as a running average which produces a new average with each scan that is the average of the previous N scans. The supported average filters are **None, 4 Average, 8 Average, 16 Average, 32 Average**.

CurrentFilterType

This parameter can be used to set the filter type to be applied to power converter load current measurement. The average value is calculated as a running average which produces a new average with each scan that is the average of the previous N scans. The supported average filters are **None, 4 Average, 8 Average, 16 Average, 32 Average**.

VoltageFilterType

This parameter can be used to set the filter type to be applied to power converter output voltage measurements. The average value is calculated as a running average which produces a new average with each scan that is the average of the previous N scans. The supported average filters are **None, 4 Average, 8 Average, 16 Average, 32 Average**.

Voltage sensing ADC range

This parameter can be used to select the ADC range for single-ended voltage measurements and for single-ended auxiliary voltage measurements. The available options are **0-4.096V Range** and **0-2.048V Range**.

Current sensing ADC range

This parameter can be used to select the ADC range for differential current measurements and for low range auxiliary voltage measurements. The available options are **+/-64mV Range** and **+/-128mV Range**.

ExposeCalPin

This checkbox can be used to expose the cal input analog pin for the calibration of the +/-64mV or +/-128mV ADC ranges. By default this option is **checked**.

pGoodConfig

This parameter determines whether the pgood output terminal is to be displayed as a **bus** terminal or a **single** output terminal. If this parameter is set to **individual**, then the pgood output terminal is displayed as bus. The pgood terminal becomes a single terminal if this parameter is set to **global.**

FaultSources

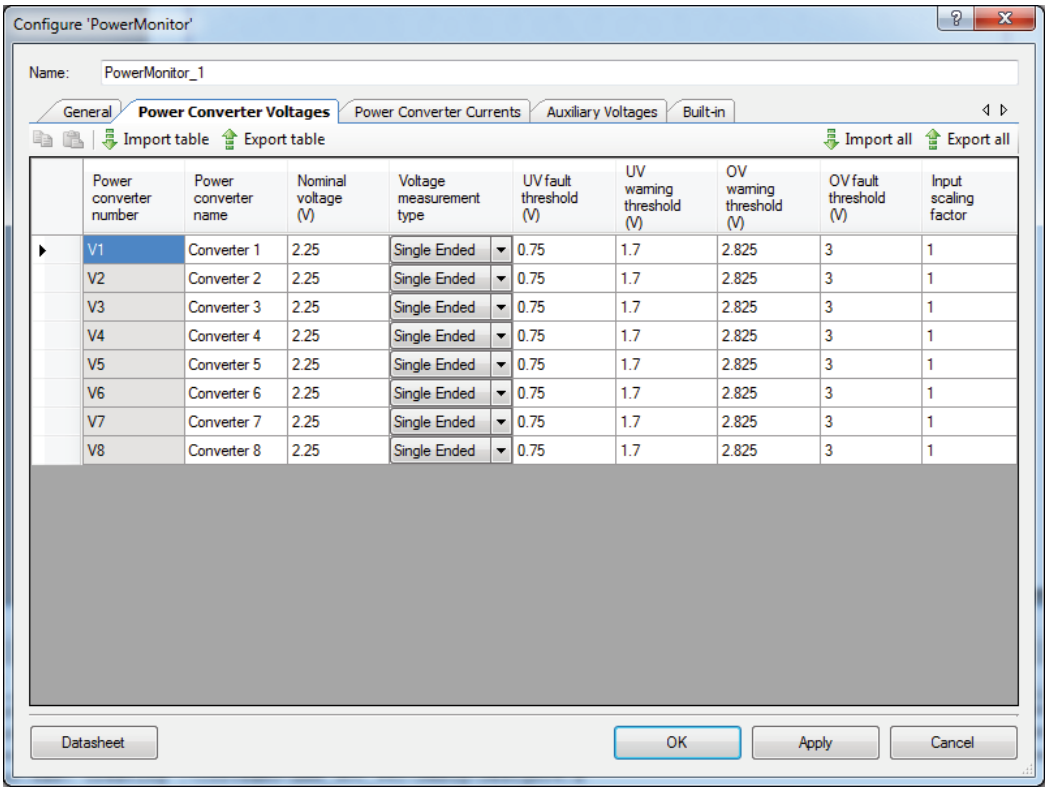
This list of check boxes can be used to set the over-current (OC), under-voltage (UV) and over-voltage (OV) fault sources. This setting applies to all configured power converters.

WarnSources

This list of check boxes can be used to set the over-current (OC), under-voltage (UV) and over-voltage (OV) warning sources. This setting applies to all configured power converters.

Power Converter Voltages Tab

This enables the user to describe the power converter voltages in the system. The figure below shows the voltage tab when Number of Converters is set to **8** in the General Tab.



Import table

Imports data from file to table cells. Supports .csv file format. Keyboard shortcut – [Ctrl] [M]

Export table

Exports data from table cells to file. Supports .csv file format. Keyboard shortcut – [Ctrl] [R].

Import all

Executes import functionality for all three tables. Keyboard shortcut – [Ctrl] [Alt] [M]

Export all

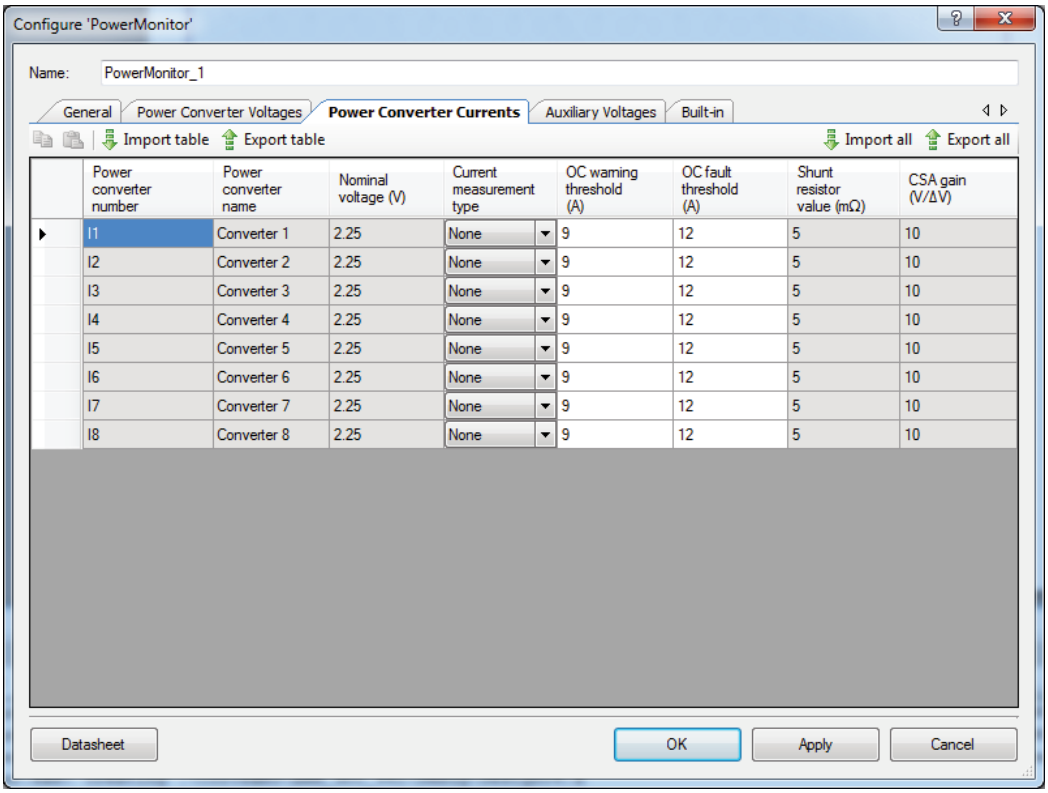
Executes export functionality for all three tables. Keyboard shortcut – [Ctrl] [Alt] [R].

Parameters:

- **Label[x] –** This is a text field to give the name for power converter. This is used only for annotation purposes. The maximum allowed characters are 16. By default this field is populated with the name “Converter x”.
- **VNom[x] –** This is the nominal converter output voltage. This is used only for annotation purposes. The nominal voltage range is **0.001 - 65.535 V**. By default this field is populated with a value 2.25.
- **VType[x] –** This parameter determines type of voltage measurement for the power converter. The options are **Single Ended** or **Differential.** If **Differential** option is selected, then that power converter forfeits the current measurement. In this case, the symbol will display a terminal with name “rtn” which can be connected to reference ground point to measure the differential voltage.
- **UVWarn[x] –** This parameter helps to set the **Under-Voltage (UV)** warning threshold for the specified power converter. The allowed warning threshold range is **0.001-65.535 V**. By default, the component will use this threshold value. The user can change the under-voltage warning threshold at run time using the provided API. Please refer API section for more details.
- **OVWarn[x] –** This parameter helps to set the **Over-Voltage (OV)** warning threshold for the specified power converter. The allowed warning threshold range is **0.001-65.535 V**. By default, the component will use this threshold value. The user can change the over-voltage warning threshold at run time using the provided API. Please refer API section for more details.
- **UVFault[x] –** This parameter helps to set the **Under-Voltage** (UV) fault threshold for the specified power converter. The allowed fault threshold range is **0.001-65.535 V**. By default, the component will use this threshold value. The user can change the under-voltage fault threshold at run time using the provided API. Please refer API section for more details.
- **OVFault[x] –** This parameter helps to set the **Over-Voltage (OV)** fault threshold for the specified power converter. The allowed fault threshold range is **0.001-65.535 V**. By default, the component will use this threshold value. The user can change the over-voltage fault threshold at run time using the provided API. Please refer API section for more details.
- **Scale[x] –** This parameter sets the input voltage scaling factor for the specified power converter. This scaling factor indicates the amount of attenuation applied to the converter output voltage external to PSoC. The allowed range is **0.001- 1.000**. The default value is **1.000.**

Power Converter Load Currents Tab

This tab enables the user to describe the power converter load currents in the system. Below figure shows the current tab when Number of Converters parameter is set to 8 in the General Tab.



Parameters:

Many aspects of this tab inherit the features from the **Voltages** tab. Below are the parameters affected:

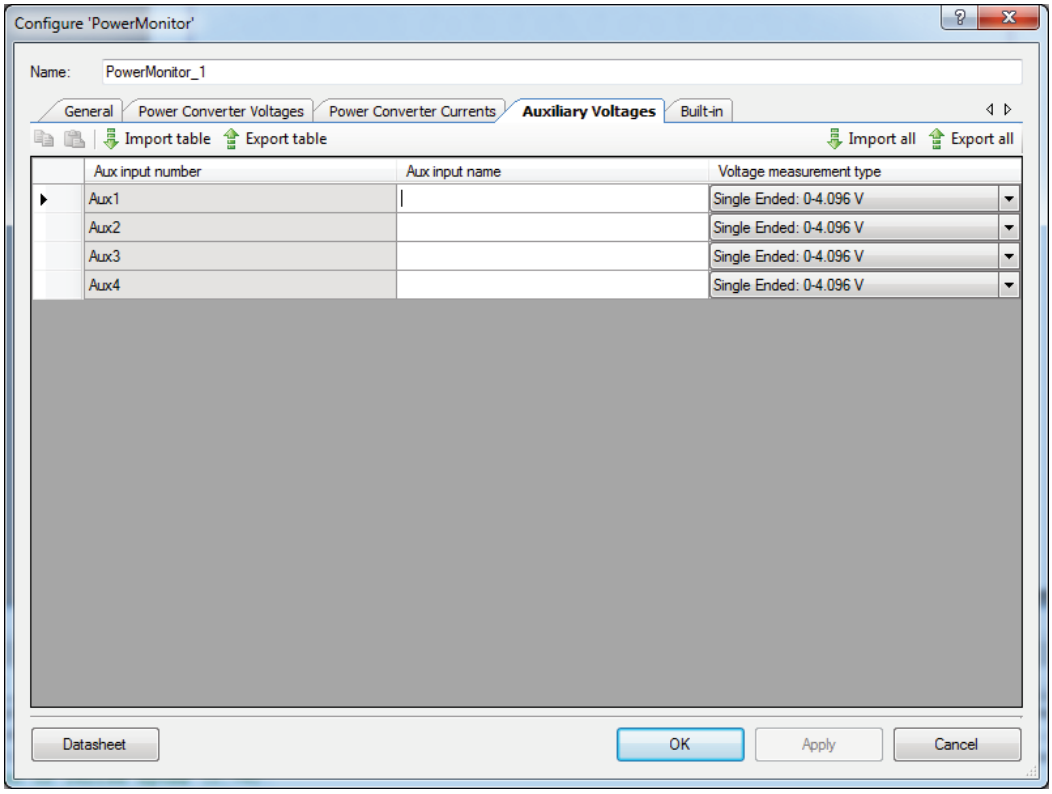
1. The converter name column is a display propagating forward the parameters entered into the Voltages Tab.
2. Nominal Output Voltage column is a display propagating forward the parameters entered into the Voltages Tab.
3. Any converter that was set to VType = Differential in the voltages tab forfeits the capability to measure current. The associated row in this table will be grayed out and current measurement type column entry will set to “None”

- **IType[x] –** This parameter sets the current measurement type for the specified power converter. The options are **None, Direct** or **CSA.**

- **OCWarn[x] –** This parameter sets the over-current (OC) warning threshold. The allowed range is **0.01 – 655.35 A**. This entry will be grayed out if the associated **IType[x]** is set to **None**.
- **OCFault[x] –** This parameter sets the over-current fault threshold. The allowed range is **0.01 – 655.35 A**. This entry is grayed out if the associated **IType[x]** is set to **None**.
- **RShunt[x] –** This parameter sets the shunt resistor value. The allowed range is **0.01 – 2500.00 mΩ**. This entry will be grayed out if the associated **IType[x]** is set to **None**.
- **CSAGainp[x] –** This parameter sets the CSA differential to single ended gain. The allowed range is **1.00 – 500.00**. This entry is grayed out if the associated **IType[x]** is set to **None** or **Direct**.

Auxiliary Voltages Tab

The Auxiliary Voltages Tab enables the user to describe the auxiliary voltage inputs in the system. The number of rows shown in this tab depends on the number of auxiliary voltages entered in the General Tab.



Parameters:

- **Voltage measurement type –** This parameter selects the type of auxiliary voltage measurement. The options are **Single Ended: 0-4.096V or Single Ended: 0-2.048V,**

Differential: +/- 2.048V and Differential: +/- 64mV or **Differential: +/- 128mV** depending on the **ADC range** parameters setting on the **General** tab.

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 "PowerMonitor_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 "PowerMonitor."

Note: When using PSoC3 silicon, the user should create a Keil .cyre reentrancy file and should add the CyIntSetVector(), CyIntSetPriority(), PowerMonitor_PM_AMux_Current_Unset() and PowerMonitor_PM_AMux_Voltage_Unset() APIs in this file to avoid reentrancy related warnings during project compilation.

Function	Description
PowerMonitor_Start()	Initializes the Power Monitor with default customizer values.
PowerMonitor_Stop ()	Disables the component. ADC sampling stops.
PowerMonitor_Init()	Initializes the component. Includes running self-calibration.
PowerMonitor_Enable()	Enables hardware blocks within the component and starts scanning.
PowerMonitor_EnableFault()	Enables generation of the fault signal.
PowerMonitor_DisableFault()	Disables generation of the fault signal.
PowerMonitor_SetFaultMode()	Configures fault sources from the component.
PowerMonitor_GetFaultMode()	Returns enabled fault sources from the component.
PowerMonitor_SetFaultMask()	Enables or disables faults from each power converter through a mask.
PowerMonitor_GetFaultMask()	Returns fault mask status of each power converter.
PowerMonitor_GetFaultSource()	Returns pending fault sources from the component.
PowerMonitor_GetOVFaultStatus()	Returns over voltage fault status of each power converter. The status is reported regardless of the Fault Mask.
PowerMonitor_GetUVFaultStatus()	Returns under voltage fault status of each power converter. The status is reported regardless of the Fault Mask.
PowerMonitor_GetOCFaultStatus()	Returns over current fault status of each power converter. The status is reported regardless of the Fault Mask.
PowerMonitor_EnableWarn()	Enables generation of the warning signal.

Function	Description
PowerMonitor_DisableWarn()	Disables generation of the warning signal.
PowerMonitor_SetWarnMode()	Configures warning sources from the component.
PowerMonitor_GetWarnMode()	Returns enabled warning sources from the component.
PowerMonitor_SetWarnMask()	Enables or disables warnings from each power converter through a mask.
PowerMonitor_GetWarnMask()	Returns warning mask status of each power converter.
PowerMonitor_GetWarnSource()	Returns pending warning sources from the component.
PowerMonitor_GetOVWarnStatus()	Returns over voltage warning status of each power converter. The status is reported regardless of the Warning Mask.
PowerMonitor_GetUVWarnStatus()	Returns under voltage warning status of each power converter. The status is reported regardless of the Warning Mask.
PowerMonitor_GetOCWarnStatus()	Returns over current warning status of each power converter. The status is reported regardless of the Warning Mask.
PowerMonitor_SetUVWarnThreshold()	Sets the power converter under voltage warning threshold for the specified power converter.
PowerMonitor_GetUVWarnThreshold()	Returns the power converter under voltage warning threshold for the specified power converter.
PowerMonitor_SetOVWarnThreshold()	Sets the power converter over voltage warning threshold for the specified power converter.
PowerMonitor_GetOVWarnThreshold()	Returns the power converter over voltage warning threshold for the specified power converter.
PowerMonitor_SetUVFaultThreshold()	Sets the power converter under voltage fault threshold for the specified power converter.
PowerMonitor_GetUVFaultThreshold()	Returns the power converter under voltage fault threshold for the specified power converter.
PowerMonitor_SetOVFaultThreshold()	Sets the power converter over voltage fault threshold for the specified power converter.
PowerMonitor_GetOVFaultThreshold()	Returns the power converter over voltage fault threshold for the specified power converter.
PowerMonitor_SetOCWarnThreshold()	Sets the power converter over current warning threshold for the specified power converter.
PowerMonitor_GetOCWarnThreshold()	Returns the power converter over current warning threshold for the specified power converter.
PowerMonitor_SetOCFaultThreshold()	Sets the power converter over current fault threshold for the specified power converter.
PowerMonitor_GetOCFaultThreshold()	Returns the power converter over current fault threshold for the specified power converter.

Function	Description
PowerMonitor_GetConverterVoltage()	Returns the power converter output voltage for the specified power converter.
PowerMonitor_GetConverterCurrent()	Returns the power converter load current for the specified power converter.
PowerMonitor_GetAuxiliaryVoltage()	Returns the voltage for the auxiliary input.
PowerMonitor_Calibrate()	Calibrates the ADC across the various range settings.

Global Variables

Variable	Description
PowerMonitor_initVar	This global variable is used to indicate whether the PowerMonitor has been initialized.
PowerMonitor_initThreshold	This global variable is used to indicate whether the PowerMonitor threshold levels have been initialized. Please refer the PowerMonitor component datasheet for detailed description.
PowerMonitor_iirInit	This global variable is used to indicate whether the PowerMonitor calibration filters have been initialized. Please refer the PowerMonitor component datasheet for detailed description.
PowerMonitor_warnWin	This structure variable is used to hold the user provided over voltage, under voltage and over current warning threshold values for each of the power converters.
PowerMonitor_faultWin	This structure variable is used to hold the user provided over voltage, under voltage and over current fault threshold values for each of the power converters.
PowerMonitor_adcConvNow	This global variable indicates the power converter for which conversion is in progress.
PowerMonitor_adcConvNext	This global variable indicates the power converter which is scheduled for next conversion.
PowerMonitor_adcConvNextPreCal	This global variable holds the next converter number before switching to calibration process if requested.
PowerMonitor_adcConvCallType	This indicates the calibration type is in progress.
PowerMonitor_faultMask	Holds the fault mask value for each of the power converters.
PowerMonitor_warnMask	Holds the warning mask value for each of the power converter
PowerMonitor_faultEnable	Holds the fault enable/disable state for the component.
PowerMonitor_warnEnable	Holds the warning enable/disable state for the component.
PowerMonitor_warnSources	Holds the warning sources set for the component.
PowerMonitor_faultSources	Holds the fault sources set for the component.

Variable	Description
PowerMonitor_OVWarnStatus	Holds the over voltage warning status for each of the power converter.
PowerMonitor_UVWarnStatus	Holds the under voltage warning status for each of the power converter.
PowerMonitor_OCWarnStatus	Holds the over current warning status for each of the power converter.
PowerMonitor_OVFaultStatus	Holds the over voltage fault status for each of the power converter.
PowerMonitor_UVFaultStatus	Holds the under voltage fault status for each of the power converter.
PowerMonitor_OCFaultStatus	Holds the over current fault status for each of the power converter.

void PowerMonitor_Start(void)

Description: Enables the component. Calls the Init() API if the component has not been initialized before. Calls Enable() API.

Parameters: None

Return Value: None

Side Effects: None

void PowerMonitor_Stop (void)

Description: Disables the component. ADC sampling stops.

Parameters: None

Return Value: None

Side Effects: pgood, warn, fault and eoc outputs are de-asserted

void PowerMonitor_Init(void)

Description: Initializes the component. Includes running self-calibration

Parameters: None

Return Value: None

Side Effects: None

void PowerMonitor_Enable(void)

Description: Enables hardware blocks within the component and starts scanning.

Parameters: None

Return Value: None

Side Effects: None

void PowerMonitor_EnableFault(void)

Description: Enables generation of the fault signal. Specifically which fault sources are enabled is configured using the PowerMonitor_SetFaultMode() and the PowerMonitor_SetFaultMask() APIs. Fault signal generation is automatically enabled by Init().

Parameters: None

Return Value: None

Side Effects: None

void PowerMonitor_DisableFault(void)

Description: Disables generation of the fault signal.

Parameters: None

Return Value: None

Side Effects: Fault output is de-asserted

void PowerMonitor_SetFaultMode(uint8 faultMode)

Description: Configures fault sources from the component. Three fault sources are available: OV, UV and OC. This is set to the customizer setting by Init().

Parameters: uint8 faultMode

Bit Field	Enabled Fault Source
0: OV_FAULT	1=Enable OV fault
1: UV_FAULT	1=Enable UV fault
2: OC_FAULT	1=Enable OC fault
7:3	Reserved. Write with all zeroes

Return Value: None

Side Effects: None

uint8 PowerMonitor_GetFaultMode(void)

Description: Returns enabled fault sources from the component

Parameters: None

Return Value:

Bit Field	Information
0: OV_FAULT	1=OV faults are enabled
1: UV_FAULT	1=UV faults are enabled
2: OC_FAULT	1=OC faults are enabled
7:3	Reserved. Returns all zeroes

Side Effects: None

void PowerMonitor_SetFaultMask(uint32 faultMask)

Description: Enables or disables faults from each power converter through a mask. Masking applies to all fault sources. Masking applies for Fault generation and Power Good generation. By default all power converters have their fault masks enabled.

Parameters: uint32 faultMask

Bit Field	Enabled Fault Source
0	1=Enable faults from Power Converter 1
1	1=Enable faults from Power Converter 2
…	…
31	1=Enable faults from Power Converter 32

Return Value: None

Side Effects: None

uint32 PowerMonitor_GetFaultMask(void)

Description: Returns fault mask status of each power converter. Masking applies to all fault sources

Parameters: None

Return Value: uint32 alertMask

Bit Field	Enabled Fault Source
0	1=Faults from Power Converter 1 are enabled
1	1=Faults from Power Converter 2 are enabled
…	…
31	1=Faults from Power Converter 32 are enabled

Side Effects: None

uint8 PowerMonitor_GetFaultSource(void)

Description: Returns pending fault sources from the component. This API can be used to poll the fault status of the component. Alternatively, if the fault pin is used to generate interrupts to PSoC's CPU core, the interrupt service routine can use this API to determine the source of the fault. In either case, when this API returns a non-zero value, the GetOVFaultStatus(), GetUVFaultStatus() and GetOCFaultStatus() APIs can provide further information on which power converter(s) caused the fault. The fault source bits are sticky and are only cleared by calling the relevant Get Status APIs.

Parameters: None

Return Value:

Bit Field	Fault Source
0: OV_FAULT	1=OV fault occurred
1: UV_FAULT	1=UV fault occurred
2: OC_FAULT	1=OC fault occurred
7:3	Reserved. Returns all zeroes

Side Effects: None

uint32 PowerMonitor_GetOVFaultStatus(void)

Description: Returns over voltage fault status of each power converter. The status is reported regardless of the Fault Mask.

Parameters: None

Return Value: uint32 ovFaultStatus

Bit Field	OV Fault Status
0	1=OV fault condition on Power Converter 1
1	1=OV fault condition on Power Converter 2
…	…
31	1=OV fault condition on Power Converter 32

Side Effects: Calling this API clears the fault condition source sticky bits. If the condition still persists then the bit will be set again after the next scan.

uint32 PowerMonitor_GetUVFaultStatus(void)

Description: Returns under voltage fault status of each power converter. The status is reported regardless of the Fault Mask.

Parameters: None

Return Value: uint32 uvFaultStatus

Bit Field	UV Fault Status
0	1=UV fault condition on Power Converter 1
1	1=UV fault condition on Power Converter 2
…	…
31	1=UV fault condition on Power Converter 32

Side Effects: Calling this API clears the fault condition source sticky bits. If the condition still persists then the bit will be set again after the next scan.

uint32 PowerMonitor_GetOCFaultStatus(void)

Description: Returns over current fault status of each power converter. The status is reported regardless of the Fault Mask.

Parameters: None

Return Value: uint32 ocFaultStatus

Bit Field	OC Fault Status
0	1=OC fault condition on Power Converter 1
1	1=OC fault condition on Power Converter 2
…	…
31	1=OC fault condition on Power Converter 32

Side Effects: Calling this API clears the fault condition source sticky bits. If the condition still persists then the bit will be set again after the next scan.

void PowerMonitor_EnableWarn(void)

Description: Enables generation of the warning signal. Specifically which warning sources are enabled is configured using the PowerMonitor_SetWarnMode() and the PowerMonitor_SetWarnMask() APIs. Warning signal generation is automatically enabled by Init().

Parameters: None

Return Value: None

Side Effects: None

void PowerMonitor_DisableWarn(void)

Description: Disables generation of the warning signal

Parameters: None

Return Value: None

Side Effects: Warning output is de-asserted

void PowerMonitor_SetWarnMode(uint8 warnMode)

Description: Configures warning sources from the component. Three warning sources are available: OV, UV and OC. This is set to the customizer setting by Init().

Parameters: uint8 warnMode

Bit Field	Enabled Warning Source
0: OV_WARN	1=Enable OV warnings
1: UV_WARN	1=Enable UV warnings
2: OC_WARN	1=Enable OC warnings
7:3	Reserved. Write with all zeroes

Return Value: None

Side Effects: None

uint8 PowerMonitor_GetWarnMode(void)

Description: Returns enabled warning sources from the component

Parameters: None

Return Value:

Bit Field	Information
0: OV_WARN	1=OV warnings are enabled
1: UV_WARN	1=UV warnings are enabled
2: OC_WARN	1=OC warningsare enabled
7:3	Reserved. Returns all zeroes

Side Effects: None

void PowerMonitor_SetWarnMask(uint32 warnMask)

Description: Enables or disables warnings from each power converter through a mask. Masking applies to all warning sources. By default all power converters have their warning masks enabled.

Parameters: uint32 warnMask

Bit Field	Enabled Warning Source
0	1=Enable warnings from Power Converter 1
1	1=Enable warnings from Power Converter 2
…	…
31	1=Enable warnings from Power Converter 32

Return Value: None

Side Effects: None

uint32 PowerMonitor_GetWarnMask(void)

Description: Returns warning mask status of each power converter. Masking applies to all warning sources

Parameters: None

Return Value: uint32 warnMask

Bit Field	Enabled Warning Source
0	1=Warnings from Power Converter 1 are enabled
1	1= Warnings from Power Converter 2 are enabled
…	…
31	1= Warnings from Power Converter 32 are enabled

Side Effects: None

uint8 PowerMonitor_GetWarnSource(void)

Description: Returns pending warning sources from the component. This API can be used to poll the warning status of the component. Alternatively, if the warning pin is used to generate interrupts to PSoC's CPU core, the interrupt service routine can use this API to determine the source of the warning. In either case, when this API returns a non-zero value, the GetOVWarnStatus(), GetUVWarnStatus() and GetOCWarnStatus() APIs can provide further information on which power converter(s) caused the warning.

Parameters: None

Return Value:

Bit Field	Warning Source
0: OV_WARN	1=OV warning occurred
1: UV_ WARN	1=UV warning occurred
2: OC_ WARN	1=OC warning occurred
7:3	Reserved. Returns all zeroes

Side Effects: None

uint32 PowerMonitor_GetOVWarnStatus(void)

Description: Returns over voltage warning status of each power converter. The status is reported regardless of the Warning Mask.

Parameters: None

Return Value: uint32 ovWarnStatus

Bit Field	OV Warning Status
0	1=OV warning condition on Power Converter 1
1	1=OV warning condition on Power Converter 2
…	…
31	1=OV warning condition on Power Converter 32

Side Effects: Calling this API clears the warning condition source sticky bits. If the condition still persists then the bit will be set again after the next scan.

uint32 PowerMonitor_GetUVWarnStatus(void)

Description: Returns under voltage warning status of each power converter. The status is reported regardless of the Warning Mask.

Parameters: None

Return Value: uint32 uvWarnStatus

Bit Field	UV fault status
0	1=UV warning condition on Power Converter 1
1	1=UV warning condition on Power Converter 2
…	…
31	1=UV warning condition on Power Converter 32

Side Effects: Calling this API clears the warning condition source sticky bits. If the condition still persists then the bit will be set again after the next scan.

uint32 PowerMonitor_GetOCWarnStatus(void)

Description: Returns over current warning status of each power converter. The status is reported regardless of the Warning Mask.

Parameters: None

Return Value: uint32 ocWarnStatus

Bit Field	OC Warning Status
0	1=OC warning condition on Power Converter 1
1	1=OC warning condition on Power Converter 2
…	…
31	1=OC warning condition on Power Converter 32

Side Effects: Calling this API clears the warning condition source sticky bits. If the condition still persists then the bit will be set again after the next scan.

void PowerMonitor_SetUVWarnThreshold(uint8 converterNum, uint16 uvWarnThreshold)

Description: Sets the power converter under voltage warning threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

uint16 uvWarnThreshold
Specifies the under voltage warning threshold in mV

The range of this value is runtime checked if this value exceeds maximum range API does nothing. Use API PowerMonitor_GetUVWarnThreshold for checking valid range.

Return Value: None

Side Effects: None

uint16 PowerMonitor_GetUVWarnThreshold(uint8 converterNum)

Description: Returns the power converter under voltage warning threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

Return Value: uint16 uvWarnThreshold
The under voltage warning threshold in mV

Side Effects: None

void PowerMonitor_SetOVWarnThreshold(uint8 converterNum, uint16 ovWarnThreshold)

Description: Sets the power converter over voltage warning threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

uint16 ovWarnThreshold
Specifies the over voltage warning threshold in mV

The range of this value is runtime checked if this value exceeds maximum range API does nothing. Use API PowerMonitor_GetOVWarnThreshold for checking valid range.

Return Value: None

Side Effects: None

uint16 PowerMonitor_GetOVWarnThreshold(uint8 converterNum)

Description: Returns the power converter under voltage warning threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

Return Value: uint16 ovWarnThreshold
The over voltage warning threshold in mV

Side Effects: None

void PowerMonitor_SetUVFaultThreshold(uint8 converterNum, uint16 uvFaultThreshold)

Description: Sets the power converter under voltage fault threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

uint16 uvFaultThreshold
Specifies the under voltage fault threshold in mV

The range of this value is runtime checked if this value exceeds maximum range API does nothing. Use API PowerMonitor_GetUVFaultThreshold for checking valid range.

Return Value: None

Side Effects: None

uint16 PowerMonitor_GetUVFaultThreshold(uint8 converterNum)

Description: Returns the power converter under voltage fault threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

Return Value: uint16 uvFaultThreshold
The under voltage fault threshold in mV

Side Effects: None

void PowerMonitor_SetOVFaultThreshold(uint8 converterNum, uint16 ovFaultThreshold)

Description: Sets the power converter over voltage fault threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

uint16 ovFaultThreshold
Specifies the over voltage fault threshold in mV

The range of this value is runtime checked if this value exceeds maximum range API does nothing. Use API PowerMonitor_GetOVFaultThreshold for checking valid range.

Return Value: None

Side Effects: None

uint16 PowerMonitor_GetOVFaultThreshold(uint8 converterNum)

Description: Returns the power converter under voltage fault threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

Return Value: uint16 ovFaultThreshold
The over voltage fault threshold in mV

Side Effects: None

void PowerMonitor_SetOCWarnThreshold(uint8 converterNum, float ocWarnThreshold)

Description: Sets the power converter over current warning threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

float ocWarnThreshold
Specifies the over current warning threshold in Amperes.

The range of this value is runtime checked if this value exceeds maximum range API does nothing. Use API PowerMonitor_GetOCWarnThreshold for checking valid range.

Return Value: None

Side Effects: None

float PowerMonitor_GetOCWarnThreshold(uint8 converterNum)

Description: Returns the power converter over current warning threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

Return Value: float ocWarnThreshold
The over current warning threshold in Amperes

Side Effects: None

void PowerMonitor_SetOCFaultThreshold(uint8 converterNum, float ocFaultThreshold)

Description: Sets the power converter over current fault threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

float ocFaultThreshold
Specifies the over current fault threshold in Amperes.

The range of this value is runtime checked if this value exceeds maximum range API does nothing. Use API PowerMonitor_GetOCFaultThreshold for checking valid range.

Return Value: None

Side Effects: None

float PowerMonitor_GetOCFaultThreshold(uint8 converterNum)

Description: Returns the power converter over current fault threshold for the specified power converter

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

Return Value: float ocFaultThreshold
The over current fault threshold in Amperes.

Side Effects: None

uint16 PowerMonitor_GetConverterVoltage(uint8 converterNum)

Description: Returns the power converter output voltage for the specified power converter. If averaging is enabled the value returned is the average value.

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

Return Value: uint16 converterVoltage
The converter output voltage in mV

Side Effects: None

float PowerMonitor_GetConverterCurrent(uint8 converterNum)

Description: Returns the power converter load current for the specified power converter. If averaging is enabled the value returned is the average value.

Parameters: uint8 converterNum
Specifies the converter number
Valid range: 1..32

Return Value: float converterCurrent
The converter output current floating point value in Amperes.

Side Effects: None

float PowerMonitor_GetAuxiliaryVoltage(uint8 auxNum)

Description: Returns the voltage for auxiliary input in units of Volts (V) independent of the ADC range setting for aux inputs.

Parameters: uint8 auxNum
Specifies the converter number
Valid range: 1..4

Return Value: float auxVoltage
Auxiliary voltage in the units of Volts(V).

Side Effects: None

void PowerMonitor_Calibrate(void)

Description: Calibrates the ADC across the various range settings. If “cal” input pin is exposed, then a valid voltage should be provided to this input pin. The cal voltage should not exceed 100% of ADC range (+/-64mV or +/-128mV) as specified in the General tab window. This voltage will be used to calibrate the low range (either +/-64mV or +/-128mV) ADC configurations.

Parameters: None

Return Value: None

Side Effects: ADC measurement of voltages and currents is temporarily suspended during this operation.

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.

The PowerMonitor component has not been verified for MISRA-C:2004 coding guidelines compliance.

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

The Power Monitor component is intended for designers of Power Supervisors who want to quickly and easily design a full-featured power monitor without having to learn the low-level details of PSoC's analog subsystem, manually setting up and configuring the ADC, configuring analog input multiplexers or worrying about calibration issues. Users can configure exactly the functionality they need for their application graphically using the component customizer. The component will take care of the implementation details for you automatically.

The component uses the 1.024 V internal precision voltage reference and multiplies it by 2 using a PGA to generate a 2.048 V offset resulting in a single ended voltage measurement range of 0 - 4.096 V using the DelSig ADC block. Differential voltage measurement range is +/- 64 mV or 128 mV.

The component supports self-calibration. Calibration will be done during initialization and then at any time when requested by firmware. The calibration is designed such that it can be done with minimal interference to the power converter sampling process.

For both voltage and current measurements, averaging of the measurements is supported. The average value is calculated as a running average, which produces a new average with each scan that is the average of the previous N scans. When averaging is enabled, the average value is used in all cases where the value of the measurement is needed (faults, warning, power good and the reading of the measurement with APIs).

Resources

This component is largely implemented in firmware. The only hardware blocks consumed are the DelSig ADC, control register and the PGA for generating the internal reference for single ended measurements.

This component operates as a background task through a repetitively called, mid-priority interrupt service routine. Designers using this component should be aware that non-interrupt driven tasks such as APIs or functions called from main() or elsewhere in the firmware, will run slower than might be expected as a result. It is therefore suggested that the CPU clock be set to at least 24 MHz to ensure adequate execution times. If other time-critical interrupt sources are required in the same design, they can be set to a higher priority to meet system performance goals.

Configuration	Resource Type			
	PGA	ADC_DelSig	Macrocells	Control Cells
Default	1	1	4	2

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
Default	12029	1085	12440	1240	12088	1224

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.

Parameter	Description	Conditions	Min	Typ	Max	Units
ADC_{Acc}	ADC measurement accuracy	Error sources internal to PSoC over the entire operating temperature range when the calibration API is called periodically. Accuracy of external components must	-	0.26	-	%

Parameter	Description	Conditions	Min	Typ	Max	Units
		be factored in to obtain system level accuracy				
T_{CONV}	ADC conversion time per measurement	Average conversion time per measurement including ADC reconfiguration time (e.g. single ended-differential)	-	150	200	µs

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.30	Added MISRA Compliance section.	The component was not verified for MISRA compliance.
1.20	Added NUMBER_OF_CONVERTERS definition	
	Added global value iirInit and initThreshold	
	Changed component macroses to 8, 16, 32 channels	
	Added macroses for interrupt management	
	Added hiding for fault/warn pin when sources are unchecked. Added Greying-out column when fault/warn sources are unchecked.	
	Corrected definitions for virelog registers	
	Fixed component output signals startup conditions	
	Added 0-2.048 single-ended voltage measurement range	
	Added URL to online datasheet	
1.10	Added support for PSoC5 LP silicon	
	Updates to Resources section in the datasheet.	

© Cypress Semiconductor Corporation, 2012. 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.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks 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.