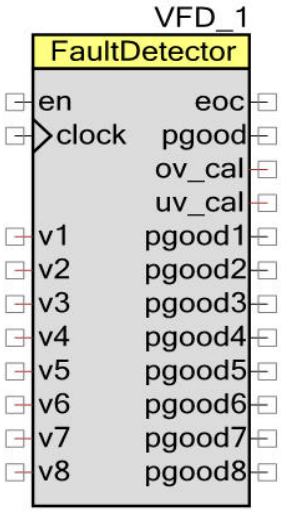
Voltage Fault Detector (VFD)

2.20



Features

- Monitor up to 32 voltage inputs
- User-defined over and under voltage limits
- Simply outputs a good/bad status result

General Description

The Voltage Fault Detector component provides a simple way to monitor up to 32 voltage inputs against user-defined over and under voltage limits without using the ADC and without having to write any firmware. The component simply outputs a good/bad status result ("power good" or pgood[x]) for each voltage being monitored.

The component operates entirely in hardware without any intervention from PSoC's CPU core resulting in known, fixed fault detection latency.

Note This component supports PSoC 3 and PSoC 5LP devices only.

When to Use a VFD

The Voltage Fault Detector component is capable of interfacing to up to 32 voltage inputs and is responsible for determining the health of those voltages by comparing them to either a user-defined under-voltage (UV) threshold or over-voltage (OV) threshold or both.

Input/Output Connections

This section describes the various input and output connections for the Voltage Fault Detector.

Clock – Input

Clock used to set the time base for the component should be set to 16x the desired multiplexing frequency. When internal OV and UV thresholds are generated by voltage DACs (VDACs), the multiplexing frequency is determined by the VDAC update rate and the maximum component clock rate. When the VDACs are configured for 0-1V range, the multiplexing frequency cannot exceed 250 kHz (limited by the maximum frequency for the component clock of 4 MHz). When the VDACs are configured for 0-4V range, the multiplexing frequency cannot exceed 200 kHz

(clock = 3.2 MHz) factoring in the VDAC update rate plus DMA time to adjust DACs and analog settling time.

When external references are selected, the user can set the timebase to a frequency that meets the system requirements. In that case, the VDAC settling time does not need to be factored in, because the VDACs are not present. In this usage case, the OV and/or UV thresholds will be common across the entire voltage set to be monitored, so the frequency is limited only by the analog voltage settling time and the maximum frequency of operation of the component's state machine. The maximum frequency supported is 4 MHz.

In either case, since DMA is involved and needs to run to completion within the time window dictated by the multiplexing frequency selected, this component inherently dictates a minimum BUS_CLK frequency. The component minimum BUS_CLK:clock ratio for this component is 2:1.

Enable – Input

This synchronous active high signal gates the clock input to the state machine controller. One purpose of this input is to support VDAC calibration.

Over Voltage Reference – Analog Input

This analog input is exposed only when the "ExternalRef" parameter is true. In this case, the user provides an over voltage threshold that replaces the internal OV VDAC. This can come from a PSoC pin or through a separate instantiation of a VDAC, for example.

Under Voltage Reference – Analog Input

This analog input is exposed only when the "ExternalRef" parameter is true. In this case, the user provides an under voltage threshold that replaces the internal UV VDAC. This can come from a PSoC pin or through a separate instantiation of a VDAC, for example.

Voltages – Analog Input

These analog inputs are the voltages that this component needs to monitor. The number of terminals displayed depends on the number of voltages selected, up to a maximum of 32.

Power Good – Output

- **Global**: A single, active high signal indicating all voltages are within range
- **Individual**: An array of active-high signals, one for each voltage input indicating v[x] is within range

End of Cycle – Output

This terminal pulses active high after every voltage input has been compared to its reference threshold(s). It indicates the end of one complete comparison cycle. For example, this signal

could be used to capture the reference voltage VDAC ouputs (ov_cal and uv_cal) for calibration purposes.

Over Voltage VDAC – Analog Output

This analog output is exposed when the “ExternalRef” parameter is true. The purpose of this is to enable calibration of the OV VDAC. To properly support the calibration activity, the component needs to be disabled through an API call or by de-asserting the en terminal.

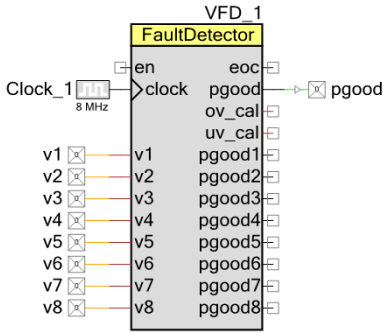
Under Voltage VDAC – Analog Output

This analog output is exposed when the “ExternalRef” parameter is true. The purpose of this is to enable calibration of the UV VDAC. To properly support the calibration activity, the component needs to be disabled through an API call or by de-asserting the en terminal.

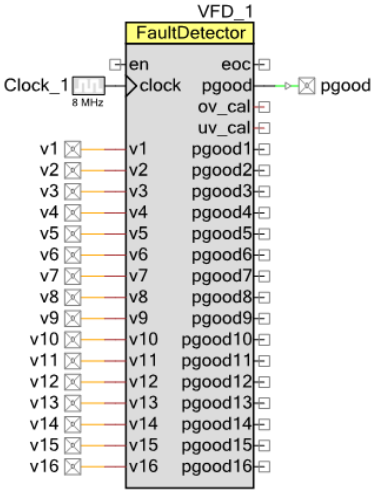
Schematic Macro Information

The Voltage Fault Detector provides the following schematic macros:

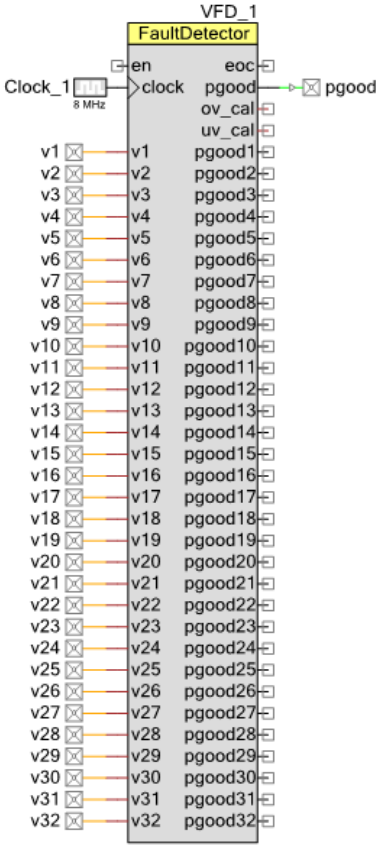
8 Rails Voltage Fault Detector



16 Rails Voltage Fault Detector



32 Rails Voltage Fault Detector

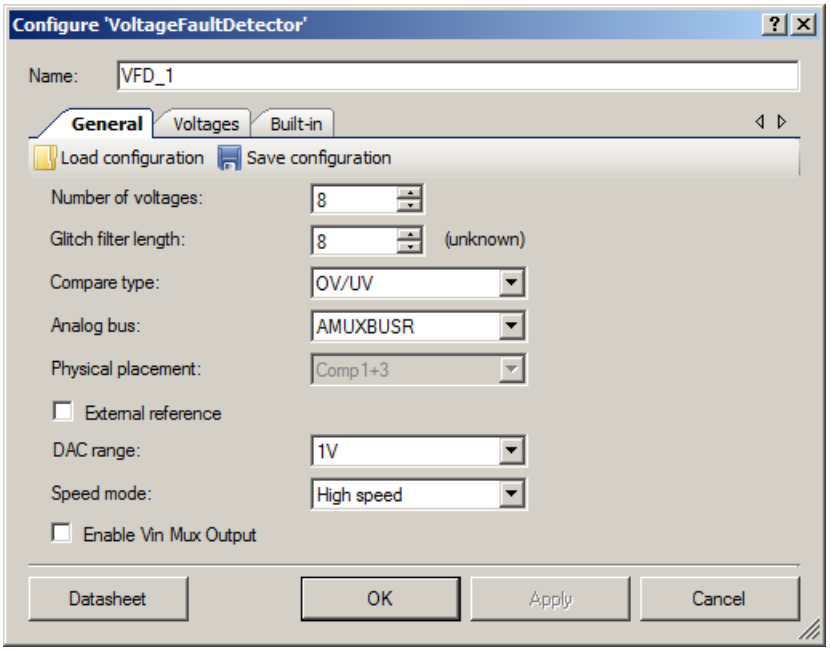


The symbol dynamically resizes depending on the number of voltages selected for fault detection in the component customizer.

Component Parameters

Drag a VFD onto your design desktop and double-click it to open the Configure dialog.

General Tab



Load configuration

Restores all customizer settings, including table, from an external file. Keyboard shortcut – [Ctrl] [L]

Save configuration

Stores all customizer settings, including table, in an external file. Keyboard shortcut – [Ctrl] [S]

Number of Voltages

Number of voltages to be monitored. Range=1-32 (default 8).

Compare Type

Pull-down list to select comparator type. Options = OV/UV, OV only, UV only (default = OV/UV).

Glitch Filter Length

Glitch filter length. Absolute units depend on the reference clock input. Options = 1..255 (default = 8)

External Reference

Check box to enable or disable external references (default unchecked).

DAC Range

Pull-down list to select internal VDAC range. Options = 1V, 4V (default = 1V). This pull-down list is grayed out if ExternalRef is checked.

Physical Placement

Pull-down list to select placement options. Some options will be grayed out depending on the CompareType parameter setting. DAC selection will be tied to the CMP selection. That is DAC0 goes with CMP0 etc. Options = Comp0, Comp1, Comp2, Comp3, Comp0+2, Comp1+3 (default = Comp1+3).

Analog Bus

Pull-down list to select routing options. Some options will be grayed out depending on the PhysicalPlacement parameter setting. This parameter will place an analog routing constraint on the analog net in the schematic that feeds the input to the comparator. Options = AMUXBUS, AMXUBUSR, AMXUBUSL, Unconstrained (default = AMUXBUSR).

Speed Mode

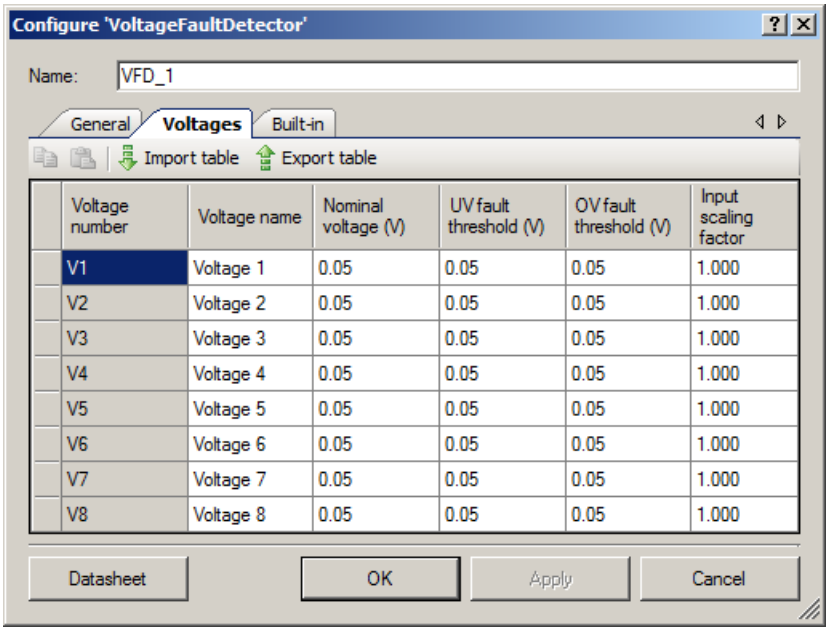
Pull-down list to select speed mode of internal VDAC(s) and Comparator(s). If External Reference option is set this setting is applied to Comparator(s) only. Options = High speed, Low speed (default = High speed).

Select High speed for fastest possible fault detection response time. Select Low speed to minimize power consumption. For additional information about speed settings of each of these components please refer to the appropriate datasheet.

Enable Vin Mux Output

Enable output from the Analog MUX. If enabled, vin_muxed output is visible and can be used to monitor current input voltage.

Voltages 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].

Label[x]

Text field, 16 characters. For annotation purposes only. By default this field is empty and no value is required.

VNom[x]

Nominal voltage. For annotation purposes only. Range=0.01–65.54 V

UVFault[x]

Under voltage fault threshold. Range=0.01–65.54 V

OVFault[x]

Over voltage fault threshold. Range=0.01–65.54 V

Scale[x]

Input voltage scaling factor. Indicates the amount of attenuation applied to the converter output voltage before connecting to PSoC. Range=0.001-1.000 (default 1.000).

Placement

The choice of analog routing channel to use is set to AMUXBUSR by default, but the user can override this to maximize routing efficiency.

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 " VFD_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 "VFD"

Functions

Function	Description
VFD_Start()	Enables the component
VFD_Stop ()	Disables the component
VFD_Init()	Initializes the component
VFD_Enable()	Enables hardware blocks
VFD_GetOVUVFaultStatus()	Returns over/under voltagefault status of each voltage input (Applicable if Compare Type is set to “OV/UV”)
VFD_GetOVFaultStatus()	Returns over voltagefault status of each voltage input (Applicable if Compare Type is set to “OV only”)
VFD_GetUVFaultStatus()	Returns under voltagefault status of each voltage input (Applicable if Compare Type is set to “UV only”)
VFD_SetUVFaultThreshold()	Sets the under voltage fault threshold for the specified voltage input
VFD_GetUVFaultThreshold()	Returns the under voltage fault threshold for the specified voltage input
VFD_SetOVFaultThreshold()	Sets the over voltage fault threshold for the specified voltage input
VFD_GetOVFaultThreshold()	Returns the under voltage fault threshold for the specified voltage input
VFD_SetUVGlitchFilterLength()	Sets the UV glitch filter length

Function	Description
VFD_GetUVGlitchFilterLength()	Returns the UV glitch filter length
VFD_SetOVGlitchFilterLength()	Sets the OV glitch filter length
VFD_GetOVGlitchFilterLength()	Returns the OV glitch filter length
VFD_SetUVDac()	Sets UV DAC value of each channel
VFD_GetUVDac ()	Gets UV DAC value for the specified voltage input
VFD_ VFD_SetOVDac ()	Sets OV DAC value of each channel
VFD_GetOVDac()	Gets OV DAC value for the specified voltage input
VFD_Pause()	Disables the clock to the comparator controller state machine
VFD_Resume()	Enables the clock to the comparator controller state machine
VFD_SetUVDacDirect()	Allows manual control of the UV VDAC value
VFD_GetUVDacDirect()	Returns current UV VDAC
VFD_SetOVDacDirect()	Allows manual control of the OV VDAC
VFD_GetOVDacDirect()	Returns current OV VDAC
VFD_ComparatorCal()	Runs a calibration routine
VFD_SetSpeed(speedMode);	Allows setting speed mode for the VDAC(s) (if Internal Reference option is enabled) and Comparator(s).

Global Variables

Function	Description
VFD_NUMBER_OF_VOLTAGES	Number of voltages to be monitored. Range=1-32
VFD_initOVFaultThreshold[VFD_NUMBER_OF_VOLTAGES]	Array which contains initial OV Fault Threshold values for all voltages
VFD_initUVFaultThreshold[VFD_NUMBER_OF_VOLTAGES]	Array which contains initial UV Fault Threshold values for all voltages
VFD_VoltageScale[VFD_NUMBER_OF_VOLTAGES]	Array which contains Scaling Factor values for all voltages.

void VFD_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 VFD_Stop(void)

Description: Stops the component.

Parameters: None

Return Value: None

Side Effects: pgood and pgood[x] outputs are de-asserted

void VFD_Init(void)

Description: Disables the component.

Parameters: None

Return Value: None

Side Effects: None

void VFD_Enable(void)

Description: Enables hardware blocks within the component and starts the state machine.

Parameters: None

Return Value: None

Side Effects: None

void VFD_GetOVUVFaultStatus(uint32 * ovStatus, uint32 * uvStatus)

Description: Assigns over/under voltage fault status of each voltage input to its parameters. Bits are sticky and cleared by calling this API. Applicable only if Compare Type is set to “OV/UV”.

Parameters: uint32 ovFaultStatus

Bit Field	OV Fault Status
0	1=OV fault condition on Voltage Input 1
1	1=OV fault condition on Voltage Input 2
…	…
31	1=OV fault condition on Voltage Input 32

uint32 uvFaultStatus

Bit Field	UV Fault Status
0	1=UV fault condition on Voltage Input 1
1	1=UV fault condition on Voltage Input 2
…	…
31	1=UV fault condition on Voltage Input 32

Return Value: None

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 VFD_GetOVFaultStatus(uint32 * ovStatus)

Description: Assigns over voltage fault status of each voltage input to its parameter. Bits are sticky and cleared by calling this API. Applicable only if Compare Type is set to “OV only”.

Parameters: uint32 ovFaultStatus

Bit Field	OV Fault Status
0	1=OV fault condition on Voltage Input 1
1	1=OV fault condition on Voltage Input 2
…	…
31	1=OV fault condition on Voltage Input 32

Return Value: None

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 VFD_GetUVFaultStatus(uint32 * uvStatus)

Description: Assigns under voltage fault status of each voltage input to its parameter. Bits are sticky and cleared by calling this API. Applicable only if Compare Type is set to “UV only”.

Parameters: uint32 uvFaultStatus

Bit Field	UV Fault Status
0	1=UV fault condition on Voltage Input 1
1	1=UV fault condition on Voltage Input 2
…	…
31	1=UV fault condition on Voltage Input 32

Return Value: None

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 VFD_SetUVFaultThreshold(uint8 voltageNum, uint16 uvFaultThreshold)

Description: Sets the under voltage fault threshold for the specified voltage input. The uvFaultThreshold parameter is stored in SRAM for retrieval by the GetUVFaultThreshold() API. The uvFaultThreshold parameter is converted to a VDAC value and gets written to an SRAM buffer for use by the DMA controller that drives the UV DAC. This API does not apply when ExternalRef=true.

Parameters: uint8 voltageNum
Specifies the voltage input number
Valid range: 1..32

uint16 uvFaultThreshold
Specifies the under voltage fault threshold in mV

Valid range: 1..65,535

Return Value: None.
The under voltage fault threshold in mV

Valid range: 1..65,535

Side Effects: uvFaultThreshold value is rounded to fit the VDAC data register format. As a result, the actual threshold value may be different from uvFaultThreshold.

void VFD_SetOVFaultThreshold(uint8 voltageNum, uint16 ovFaultThreshold)

Description: Sets the over voltage fault threshold for the specified voltage input. The ovFaultThreshold parameter is stored in SRAM for retrieval by the GetOVFaultThreshold() API. The ovFaultThreshold parameter is converted to a VDAC value and gets written to an SRAM buffer for use by the DMA controller that drives the OV DAC. This API does not apply when ExternalRef=true.

Parameters: uint8 voltageNum
Specifies the voltage input number
Valid range: 1..32

uint16 ovFaultThreshold
Specifies the over voltage fault threshold in mV

Valid range: 1..65,535

Return Value: None

Side Effects: ovFaultThreshold value is rounded to fit the VDAC data register format. As a result, the actual threshold value may be different from ovFaultThreshold.

uint16 VFD_GetUVFaultThreshold(uint8 voltageNum)

Description: Returns the under voltage fault threshold for the specified voltage input that was stored in SRAM by the SetUVFaultThreshold() API. This API does not apply when ExternalRef=true.

Parameters: uint8 voltageNum
Specifies the voltage input number
Valid range: 1..32

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

Valid range: 1..65,535

Side Effects: None

uint16 VFD_GetOVFaultThreshold(uint8 voltageNum)

Description: Returns the over voltage fault threshold for the specified voltage input that was stored in SRAM by the SetOVFaultThreshold() API. This API does not apply when ExternalRef=true.

Parameters: uint8 voltageNum
Specifies the voltage input number
Valid range: 1..32

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

Valid range: 1..65,535

Side Effects: None

void VFD_SetUVGlitchFilterLength(uint8 filterLength)

Description: Sets the UV glitch filter length

Parameters: uint8 filterLength
Absolute time units depend on the input clock frequency
Valid range: 1..255

Return Value: None

Side Effects: None

uint8 VFD_GetUVGlitchFilterLength(void)

Description: Returns the UV glitch filter length

Parameters: None

Return Value: uint8 filterLength
Absolute time units depend on the input clock frequency
Valid range: 1..255

Side Effects: None

void VFD_SetOVGlitchFilterLength(uint8 filterLength)

Description: Sets the OV glitch filter length

Parameters: uint8 filterLength
Absolute time units depend on the input clock frequency
Valid range: 1..255

Return Value: None

Side Effects: None

uint8 VFD_GetOVGlitchFilterLength(void)

Description: Returns the OV glitch filter length

Parameters: None

Return Value: uint8 filterLength
Absolute time units depend on the input clock frequency
Valid range: 1..255

Side Effects: None

void VFD_SetUVDac(uint8 voltageNum, uint8 dacValue)

Description: Calling this API does not change the UV VDAC setting immediately. Instead, the dacValue gets written to an SRAM buffer for use by the DMA controller that drives the UV DAC for the specified voltage input. This API does not apply when ExternalRef=true.

Parameters: uint8 voltageNum
Specifies the voltage input number
Valid range: 1..32

uint8 dacValue
Specifies the value to be written to the UV VDAC

Valid range: 1..255

Return Value: None

Side Effects: None

uint8 VFD_GetUVDac(uint8 voltageNum)

Description: Returns the dacValue currently being used by the DMA controller that drives the UV DAC for the specified voltage input. This API does not apply when ExternalRef=true.

Parameters: uint8 voltageNum
Specifies the voltage input number
Valid range: 1..32

Return Value: uint8 dacValue

Side Effects: None

void VFD_SetOVDac(uint8 voltageNum, uint8 dacValue)

Description: Calling this API does not change the OV VDAC setting immediately. Instead, the dacValue gets written to an SRAM buffer for use by the DMA controller that drives the OV DAC for the specified voltage input. This API does not apply when ExternalRef=true.

Parameters: uint8 voltageNum
Specifies the voltage input number
Valid range: 1..32

uint8 dacValue
Specifies the value to be written to the OV VDAC

Valid range: 1..255

Return Value: None

Side Effects: None

uint8 VFD_GetOVDac(uint8 voltageNum)

Description: Returns the dacValue currently being used by the DMA controller that drives the OV DAC for the specified voltage input. This API does not apply when ExternalRef=true.

Parameters: uint8 voltageNum
Specifies the voltage input number
Valid range: 1..32

Return Value: uint8 dacValue

Side Effects: None

void VFD_Pause(void)

Description: Disables the clock to the comparator controller state machine. Note that calling this API does not stop the DMA controller if it is in the process of executing transactions. DMA takes around 20 BUS_CLK cycles to complete assuming that no other resource is using the DMA controller at the same time. Therefore, if the purpose of calling this API is specifically to change VDAC settings (for calibration purposes for example), sufficient time should be allowed to let the DMA controller run to completion before attempting to access the VDACs directly.

Parameters: None

Return Value: None

Side Effects: Stops the fault detection state machine. Does not stop the DMA controller immediately.

void VFD_Resume(void)

Description: Enables the clock to the comparator controller state machine.

Parameters: None

Return Value: None

Side Effects: Restarts the fault detection logic

void VFD_SetUVDacDirect(uint8 dacValue)

Description: Allows manual control of the UV VDAC value. The dacValue is written directly to the UV VDAC component. Useful for UV VDAC calibration. Note that if the VFD component is running when this API is called, the state machine controller will override the UV VDAC value set by this API call. Call the Pause API to stop the state machine controller if manual UV VDAC control is desired. This API does not apply when ExternalRef=true.

Parameters: uint8 dacValue

Valid range: 1..255

Return Value: None

Side Effects: Calling this API may cause the comparator to trigger a fault condition. To prevent this, call the VFD_Pause() API prior to calling this API

uint8 VFD_GetUVDacDirect(void)

Description: Returns current UV VDAC. The returned dacValue is read directly from the UV VDAC component. Useful for UV VDAC calibration. Note: if this API is called while the component is running, it isn't possible to know which voltage input the returned UV VDAC value is associated with. Call the Pause API to stop the state machine controller if manual UV VDAC control is desired. This API does not apply when ExternalRef=true.

Parameters: None

Return Value: uint8 dacValue

Side Effects: None

void VFD_SetOVDacDirect(uint8 dacValue)

Description: Allows manual control of the OV VDAC value. The dacValue is written directly to the OV VDAC component. Useful for OV VDAC calibration. Note that if the VFD component is running when this API is called, the state machine controller will override the OV VDAC value set by this API call. Call the Pause API to stop the state machine controller if manual OV VDAC control is desired. This API does not apply when ExternalRef=true.

Parameters: uint8 dacValue

Valid range: 1..255

Return Value: None

Side Effects: Calling this API may cause the comparator to trigger a fault condition. To prevent this, call the VFD_Pause() API prior to calling this API

uint8 VFD_GetOVDacDirect(void)

Description: Returns current OV VDAC. The returned dacValue is read directly from the VDAC component. This is useful for OV VDAC calibration.

Note If this API is called while the component is running, it is impossible to know which voltage input the returned OV VDAC value is associated with. Call the Pause API to stop the state machine controller if manual UV VDAC control is desired. This API does not apply when ExternalRef=true.

Parameters: None

Return Value: uint8 dacValue

Side Effects: None

void VFD_ComparatorCal(enum compType)

Description: Runs a calibration routine that measures the selected comparator's offset voltage by shorting its inputs together and corrects for it by writing to the CMP block's trim register

Parameters: enum compType

Valid values: VFD_OV, VFD_UV

Return Value: None

Side Effects: Calling this API may cause the comparator to trigger a fault condition. To prevent this, call the VFD_Pause() API prior to calling this API

void VFD_SetSpeed(uint8 speedMode)

Description: Allows setting speed mode for the VDAC(s) (if Internal Reference option is enabled) and Comparator(s).

Parameters: uint8 speedMode

Valid values: VFD_HIGH_SPEED, VFD_LOW_SPEED

Return Value: None

Side Effects: None

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 Voltage Fault Detector component has not been verified for MISRA-C:2004 coding guidelines compliance.

This component has the following embedded components: Software and Hardware AMUX, DMA. Refer to the corresponding component datasheets for information on their MISRA compliance and specific deviations.

Sample Firmware Source Code

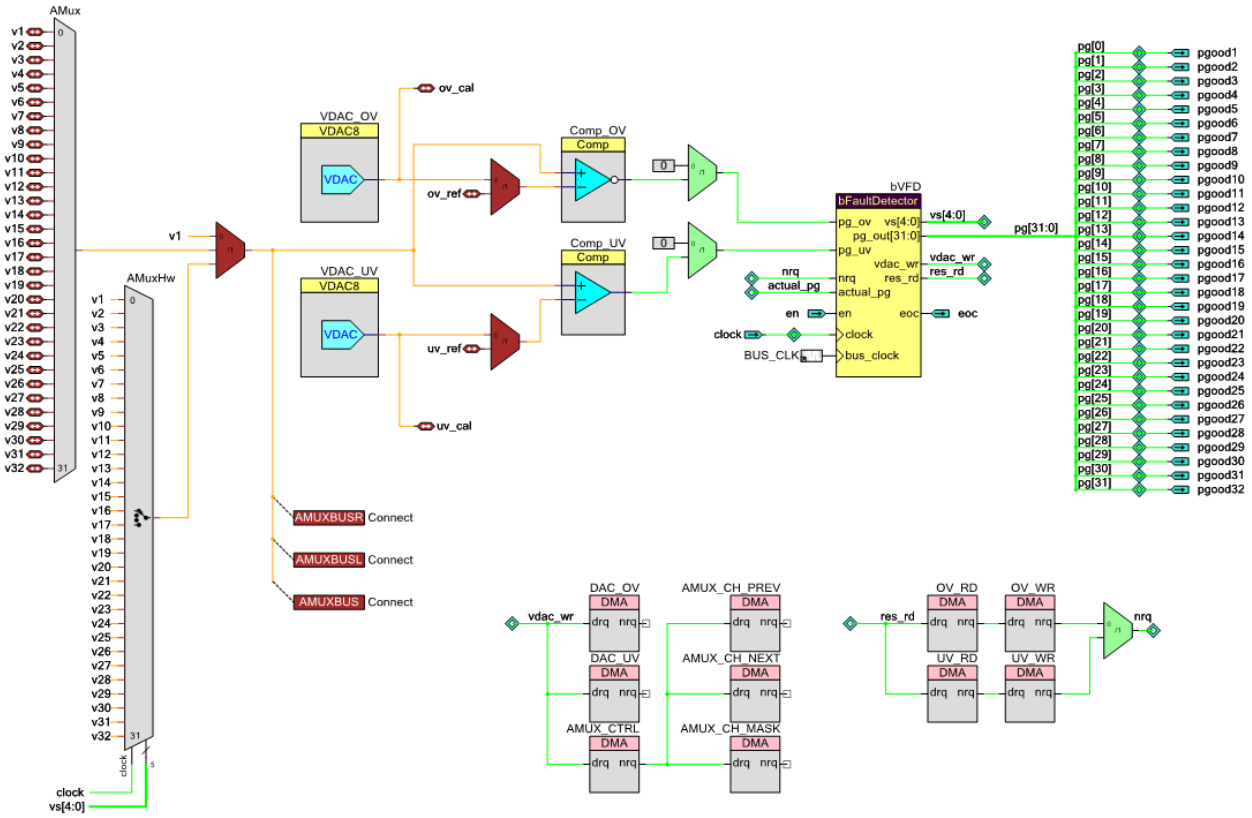
PSoC Creator provides numerous example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start 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 diagram below shows the schematic representation of the Voltage Fault Detector.

Figure 1. Schematic Representation of Voltage Fault Detector



The vs[4:0] signals are responsible for connecting the voltage to be tested to the analog multiplexer. They are derived from a down counter block, so the voltages are multiplexed in reverse order. The signals have a dead time between them ensuring a break-before-make connection

The first event is to latch the most recent comparator output into the Glitch Filter after the next external voltage input is selected for monitoring and new OV/UV threshold data has been written to the VDACs (if External Reference option is disabled). Latching of the current comparator value(s) occurs 9 component clock cycles after the VDACs are updated (if External Reference option is disabled) or from the moment when scanning of the current voltage has begun (if External Reference option is enabled).

The vdac_wr control signal is generated next and is used to DMA the OV and UV thresholds for the next voltage input into the VDACs (if External Reference option is disabled).

The res_rd control signal is generated last and is used to switch the context of the Glitch Filter in and out of SRAM. The DMA controllers with the “_RD” suffix read the Glitch Filter count for the current voltage and write it to SRAM. The DMA controllers with the “_WR” suffix write the glitch

filter result from the last measurement of the next voltage back from SRAM back to the Glitch Filter

The “Glitch Filter” is a counter with a programmable period (the Glitch Filter delay). It gets reset to zero anytime the associated comparator “power good” output is good. Its main purpose is to prevent temporary input voltage glitches that are outside the specified OV/UV limits from generating a fault condition. Any time the comparator output is bad, the counter inside the Glitch Filter starts to increment. Once it hits the user-specified Glitch Filter delay value, indicating a persistent voltage-out-of-range condition, the Glitch Filter output reflects the power bad condition. Thus the Glitch Filter propagates good values immediately, but filters bad values only once the user-specified Glitch Filter delay expires. The Glitch Filter outputs feed back into the bVoltageFaultDetector component where they are latched into status registers for the CPU to access and are also used to generate the “power good” outputs for the component “pgood[x]”.

When internal VDACs are used to set the OV/UV thresholds, the user can select between the 0-1V range and the 0-4V range. The advantage of the 0-1V range is that the VDAC update rate is higher (1 MHz) compared to the 250 kHz of the 0-4V range. This translates to a faster fault detection time, critical in many applications. In either case, all voltages to be monitored need to be scaled such that they fall within the selected VDAC range in the extreme case. That is, the OV threshold for any given voltage must fall within the VDAC max limits. The general guideline is that all voltages should be scaled such that the nominal voltage is within 85% of the selected VDAC upper limit. This is assuming that the OV threshold is no more than 10% above nominal. The customizer and the component datasheet need to provide recommendations for scaling and enforce checks to make sure that the user’s settings match hardware capabilities.

To allow using the same analog pin per voltage rail for voltage measuring and fault detection the Software Analog MUX component is used along with DMA to provide the channel selection functionality. It is valid for all “Analog Bus” options except “Unconstrained”, when this solution is not applicable and the Hardware Analog MUX component is used for that purpose along with the UDB Counter.

Resources

The Voltage Fault Detector component is placed throughout the UDB array. The component utilizes the following resources (for the Compare Type set to OV/UV).

Configuration (number of voltages)	Resource Type					
	Datapath Cells	Macrocells	Status Cells	Control Cells	DMA Channels	Interrupts
8	1	58	2	3	10	–
16	1	58	4	3	10	–
24	1	75	6	3	10	–
32	1	93	8	3	10	–

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 (number of voltages)	PSoC 3 (Keil_PK51)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
8	4293	92	3578	94
16	4395	148	3658	150
24	4605	204	3738	206
32	4725	260	3798	262

DC and AC Electrical Characteristics

The following values indicate expected performance and are based on initial characterization data.

VFD DC Specifications

Component's DC characteristics are related to VDAC and Comparator components DC characteristics. Please refer to the appropriate datasheet for more information.

VFD AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
CLKFreq	Operating frequency	Internal DAC, 1V range	-	-	4	MHz
		Internal DAC, 4V range	-	-	3.2	MHz
		External reference	-	-	4	MHz
TFaultDet	Voltage fault detection time (Per rail)	1V DAC range	1.8	2	2.2	µs
		4V DAC range	4.5	5	5.5	µs
CMPTResp	Comparator response time		30	75	110	ns

Component Errata

This section lists known problems with the component.

Cypress ID	Component Version	Problem	Workaround
152757	All	The control logic responsible for selecting input voltages and configuring the OV and UV thresholds can become out of synchronization when the component is paused and then resumed. This applies both for the usage of the VFD_Pause() and VFD_Resume() functions as well as changing the value provided on the EN terminal of the component.	Once the component is started do not disable with the EN pin or pause with the VFD_Pause() function.
159310	All	Fault status of the Voltage Fault Detector is determined by reading a PGOOD and Over Voltage (OV) status register. The bits in the OV status register are sticky and the APIs responsible for returning fault status incorrectly clear the sticky bits. As a result, this can cause an issue to manifest in one of two scenarios: 1. If a UV fault occurs on a rail that was preceded by an OV fault, the GetOVUVFaultStatus() function may incorrectly report an OV fault and fail to report the UV fault. 2. When using the GetOVUVFaultStatus() or GetOVFaultStatus() APIs, the fault data returned may contain stale fault data, indicating an OV fault, for a rail that did not experience that fault condition.	When checking the fault status using the GetOVUVFaultStatus() or GetOVFaultStatus() APIs, call the respective function twice and use the result returned by the second function call.
159749	All	The General tab in the Voltage Fault Detector customizer displays a fault response time next to the Glitch Filter Length selection. The time calculated only includes the delay associated with the glitch filter and doesn't include the delay for the final scan that will detect the fault.	To more accurately determine the Fault Response Time. Take the number provided in the customizer and add one additional cycle period to that value.

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
2.20.a	Added Component Errata section.	Document known problems in the component.
	Reduced maximum clock speed from 12 MHz to 4 MHz.	Usage in systems has shown that 4 MHz is the maximum reliable operating speed for the component clock.
2.20	Added vin_muxed output. Added VFD_SetSpeed() API.	Updated requirements for the component. Fixed number of DMA channels used in the implementation.
2.10	Component schematic is updated with the latest version of all components.	
	Added MISRA Compliance section.	The component was not verified for MISRA compliance.
2.0	Added support for PSoC 5LP devices. Added GetOVUVFaultStatus() API. Changed GetOVFaultStatus() and GetOVFaultStatus() API description. Updated screenshot of the General Tab of the component's customizer. Added description of the Speed Mode Parameter. Added 'AMUXBUS' option for "Analog Bus". Added the following global arrays to the 'Global Variables' section: VFD_initOVFaultThreshold[], VFD_initUVFaultThreshold[], VFD_VoltageScale[]. Added description of the Software Analog MUX functionality. Resource usage table is updated. Minor datasheet edits.	GetOVFaultStatus() and GetOVFaultStatus() API are incompatible with v1_0.
1.0	First release	

© Cypress Semiconductor Corporation, 2014. 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.