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

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

AN78646

Integrated Power Manager Using PSoC® 1

Associated Project: Yes
Associated Part Family: CY8C28xxx
Software Version: PSoC Designer 5.3
Related Application Notes: AN62496, AN2249

AN78646 describes the key concepts of power management, including voltage sequencing, fault detection, voltage and current monitoring, real-time trimming, and I^2C host communication. The associated example project delivers a fully integrated power management system solution.

Contents

PSoC 1 Power Management Solution 2
Voltage Sequencing 3
UV and OV Fault Detection 4
Voltage Monitoring 7
Current Monitoring.....8
Trimming 10
Configuration Over I^2C With On-Chip EEPROM 12
Example With PM-EBK 13
Hardware Connections and Setup.....14
Executing the Project 15
Appendix A: Window Comparator Implementation.....17
Worldwide Sales and Design Support.....22

Introduction

Computing and communication systems are complex and comprise multiple subsystems. Each subsystem can have its own voltage domain, which is also known as a power rail. Each power rail needs to be powered up and monitored individually. This creates a need for power management. Because this activity is a health indicator of the system, it cannot be incorporated in any of the subsystems. For this reason, a dedicated chip is used.

Power management consists of the following parts:

- accurate and reliable voltage sequencing
- rapid power rail fault detection
- voltage and current monitoring of power rails to optimize power consumption
- real-time trimming for closed-loop control of power converters (voltage regulators)
- in-system margining
- fault/event logging in EEPROM
- communication with the host controller using I^2C, SMBus, or PMBus

Power management solutions are traditionally implemented using complex PLDs (CPLDs), FPGAs, ASSPs, and discrete analog components. Often these solutions are not cost-effective because they offer limited scalability and customizability. PSoC 1 has a unique architecture with programmable digital and analog resources that enable system designers to implement power management functions with the smallest-solution footprint and lowest BOM cost. Also, system designers can use PSoC Designer to make PSoC 1–based scalable and customizable power management system solutions.

The concepts of power management are explained through an example project that uses the CY8C28645 device:

- Time-based voltage sequencing
- Under Voltage (UV) and Over Voltage (OV) fault detection
- Voltage and current monitoring
- Real-time trimming
- System configuration over I^2C with on-chip EEPROM storage for system parameters.

In-system margining, fault/event logging, SMBus features, and PMBus features can also be incorporated with firmware modifications to the example project without any additional hardware resources.

The example project uses the CY8CKIT-001 PSoC Development Kit (DVK) with the CY8C28 Family Processor Module and the CY8CKIT-035 Power Management Expansion Board Kit (PM-EBK) to demonstrate the previously described features. The PM-EBK enables system designers to evaluate power management functions and capabilities of the PSoC architecture.

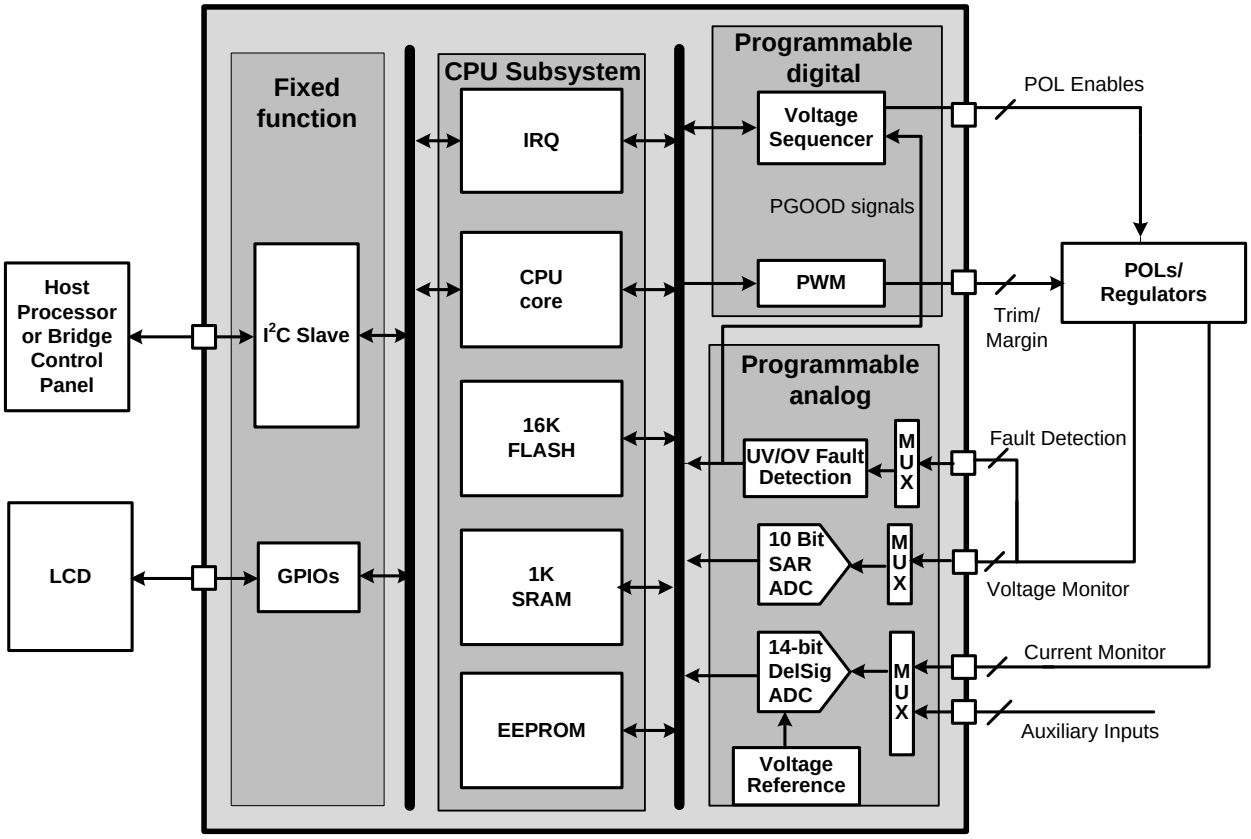
The following section explains the functionalities of power management with details on their requirements and their PSoC 1 implementation.

PSoC 1 Power Management Solution

PSoC 1 has a unique programmable architecture that consists of configurable analog and digital blocks. These blocks create peripherals, such as ADC, Timer, Counter, PWM, DAC, and Amplifier, that allow system developers to build a fully integrated, single-chip system solution.

Figure 1 shows the PSoC 1 implementation of power management.

Figure 1. PSoC 1 Power Management Solution Block Diagram



Voltage Sequencing

Voltage sequencing is a major function of any power management IC. The supply voltages to different sub-systems of a complex system are required to be sequenced, depending on operating conditions such as power up, power down, and fault conditions to ensure proper operation of all the devices. As the system becomes more complex, the number of different supply voltages in a single system increases. Therefore, an accurate and reliable power sequencing solution is required.

During powering up of the system, each of the voltage regulators needs to be powered up one after another in a sequential order with programmable delays between each of them. During sequencing, the controller enables the voltage regulator, monitors its rail for the voltage level, and proceeds to the next rail only if the voltage level is within the user-defined UV/OV range. If any power rail does not reach the expected voltage level, the controller has to indicate a sequencing fail. For this implementation, the following resources are needed:

- A 16-bit counter for generating user-defined delays. Two digital blocks of PSoC 1 are used to create a 16-bit counter with a 50-KHz internally generated clock as the time base, which gives the maximum counting time interval of 1.31 seconds before overflowing. The API provided can be used to generate a time delay from 1 msec to 1310 msec in steps of 1 msec.
- Four GPIOs to enable or disable four voltage regulators on the CY8CKIT-035 board.
- A Power Good (PGOOD) signal that indicates the correctness of the voltage on a particular rail. The PGOOD signal can be generated by either the regulator or the controller. Currently, it is generated by PSoC 1 using a window comparator. The window comparator is explained in detail in the next section.

The voltage sequencing algorithm has the following parameters:

- Sequencing Mode: This indicates the sequencing order in which the four rails have to be powered up. Six different modes are available.
- Ramp Delay: This indicates the time that the sequencer has to wait before monitoring the rail for its goodness (checking the PGOOD) after the regulator is enabled.
- Slot Delay: This indicates the time that the sequencer has to wait before sequencing the next rail after the current rail has been verified with the correct voltage (PGOOD = 1).

In this implementation, when a sequencing fail occurs, all the regulators are disabled simultaneously. An error message along with the rail number that is faulty displays on the LCD. With minor modifications to the firmware, you can easily implement power-down sequencing in different orders when a power fail occurs.

Figure 2 shows the timing diagram of a typical voltage sequencer with four power rails. Figure 3 shows the flow chart for voltage sequencing.

Figure 2. Timing Diagram of Voltage Sequencer

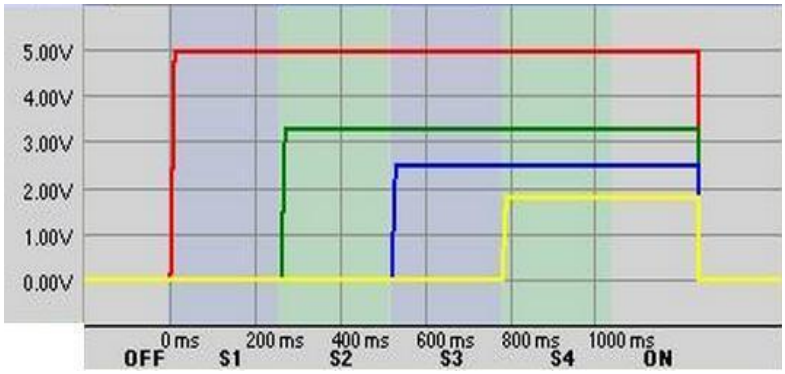
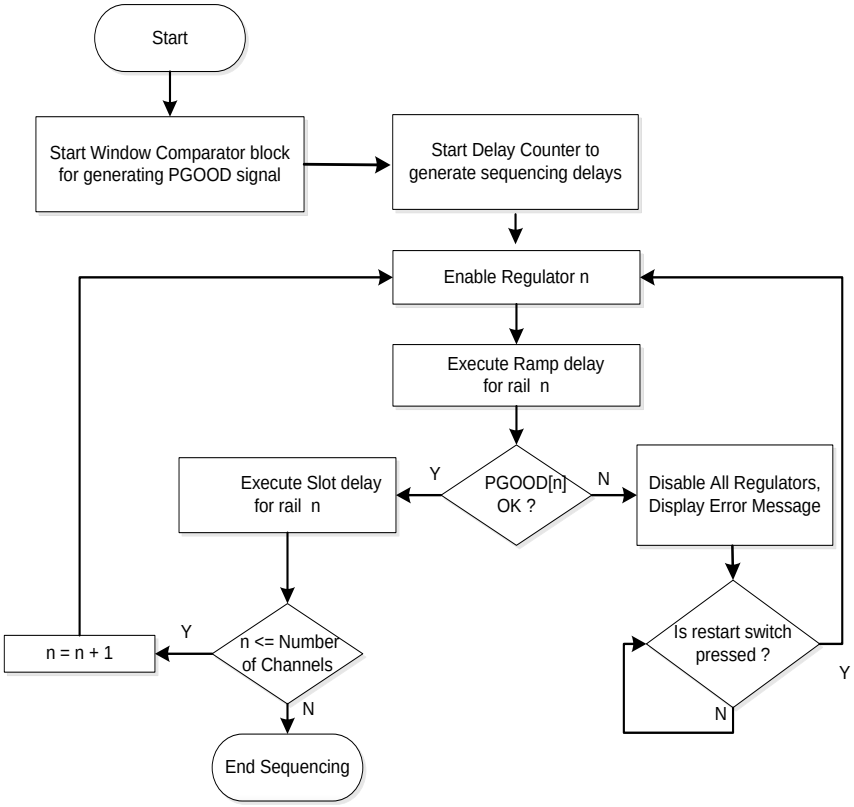


Figure 3. Flow Chart of Voltage Sequencing Firmware



UV and OV Fault Detection

Fault detection is a critical function of a power management controller. Each subsystem has a strict power supply operating range for its reliable operation. To ensure reliable operation of the subsystem, it is necessary to detect UV and OV fault conditions when any of the supply rails deviates from its safe operating range.

There are two methods for detecting a power rail fault:

Using an ADC: An on-chip ADC measures the voltage level of the power rail. The output of the ADC is compared in the firmware against UV and OV threshold voltages. When the ADC output is not within these thresholds, it is considered as a fault and the appropriate flag is set to indicate the fault condition. It is an easy implementation with fewer on-chip resource requirements. The drawbacks of this implementation are:

- CPU will be 100% used for fault detection.
- The fault detection latency depends on the ADC's sampling rate, firmware execution time (CPU clock), and number of power rails monitored. The latency goes up drastically as the number of power rails goes up.
- There is no interrupt to indicate the fault condition.

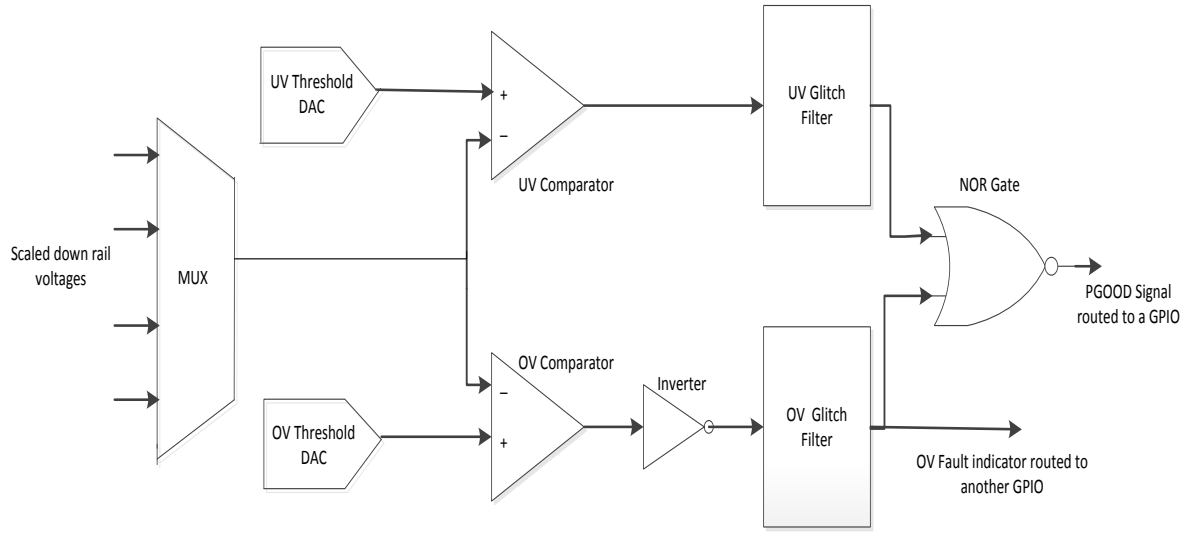
Using a hardware window comparator: The voltage level of the power rail is compared against the UV and OV thresholds using two H/W comparators. If the voltage level is not within these thresholds, the appropriate comparator generates the fault signal. The advantages of this method are:

- CPU is not used.
- The fault detection latency does not depend on any firmware execution time. It depends only on the number of power rails multiplexed for the fault detection.
- Configurable interrupt to indicate the fault condition.

Because of its superior response time, the window comparator is selected for the fault detection in the current implementation.

Figure 4 shows the block diagram of the fault detection implementation for four rails in PSoC 1.

Figure 4. Block Diagram of Window Comparator



The UV comparator checks for under voltage fault; output will be high if there is a fault. The OV comparator checks for over voltage fault; output (after the inverter) will be high if there is a fault. Eight-bit DACs are used to generate the user-defined UV and OV thresholds.

When the inputs to the comparator become nearly equal, the comparator's output may contain glitches leading to unstable fault detection output. To reliably detect the fault, you must ignore the rare glitches (transitions from 0 to 1 and 1 to 0) in the comparator's output.

To do this, glitch filters are implemented at the output of the comparators. The glitch filter will indicate a fault only when the comparator indicates a fault for a sufficient amount of time. This time required to get an accurate PGOOD signal will be determined by trial and error. This time depends on the glitch filter's clock; for this reason, it is programmable. In the current implementation, the glitch filter waits for four updates of the comparator output. If all four updates indicate a fault, the glitch filter asserts a fault.. This way, you can eliminate any glitches from the comparator whose time period is less than four updates of the comparator.

Multiplexing must be done at the input of the window comparator to find out faults on all power rails. To efficiently monitor all power rails and to decrease the fault detection time, multiplexing must be fast, and multiplexing time must be deterministic. It should not depend on any firmware cycle time.

To do this, use a counter to generate periodic interrupts. Each time inside the ISR, the CPU changes the input to the window comparator and resumes its normal operation. In the current implementation, the counter generates an interrupt every 10 µs. Each rail will be monitored for 10 µs out of a 40-µs cycle time.

The PGOOD signal is the one that indicates whether the voltage is within the UV/OV range. This is just a NOR'ed signal of the OV and UV comparator outputs. If any fault exists, PGOOD becomes 0 to indicate the fault. This is routed to a GPIO, and its falling edge interrupt is enabled. Whenever a fault occurs, the CPU will be interrupted. After the PGOOD interrupt occurs, the OV fault indicator (which is routed to another GPIO) is checked to find out whether the fault was UV or OV. Within the ISR, all rails are monitored and checked for their PGOOD status to capture all faulty rails.

Detailed User Module (UM) and register-level implementation of comparators, glitch filters, and other finer details regarding the window comparator are discussed in Appendix A: Window Comparator Implementation.

The fault detection feature works based on the following parameters:

- UV threshold percentage: This indicates the under voltage threshold for the power rails in terms of –ve percentage of deviation from nominal voltage.
- OV threshold percentage: This indicates the over voltage threshold for the power rails in terms of +ve percentage of deviation from nominal voltage.

Example:
If UV threshold = 5%, then the UV threshold voltage = Nominal voltage * 95%.

If OV threshold = 5%, then the OV threshold voltage = Nominal voltage * 105%.

These voltages will be generated by the DAC. The window comparator detects the fault when the input voltage crosses these two threshold voltages.

Figure 5 and Figure 6 show the flow charts for fault detection firmware.

Figure 5. Flow Chart for Initialization of Window Comparator and Multiplexing ISR

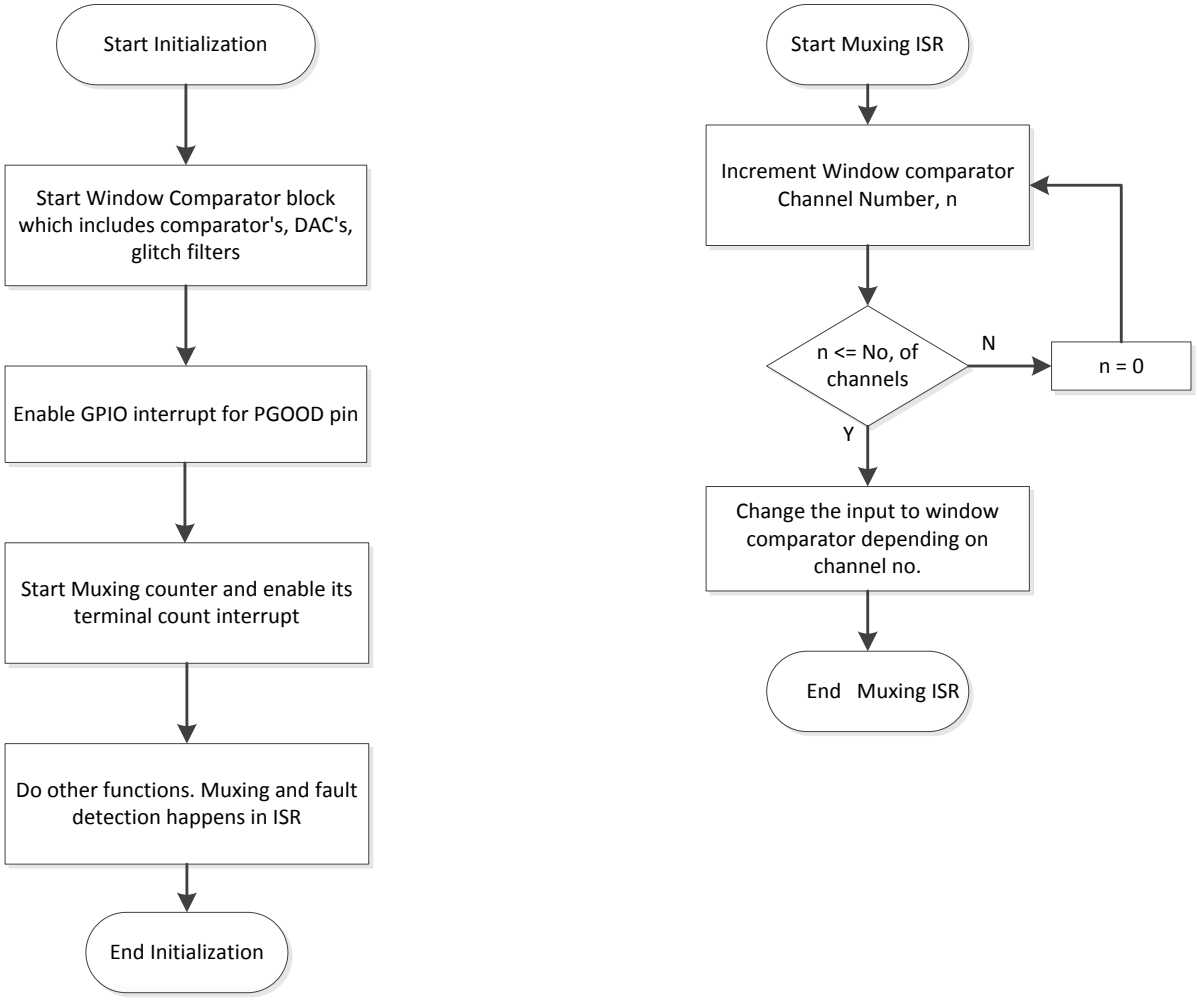
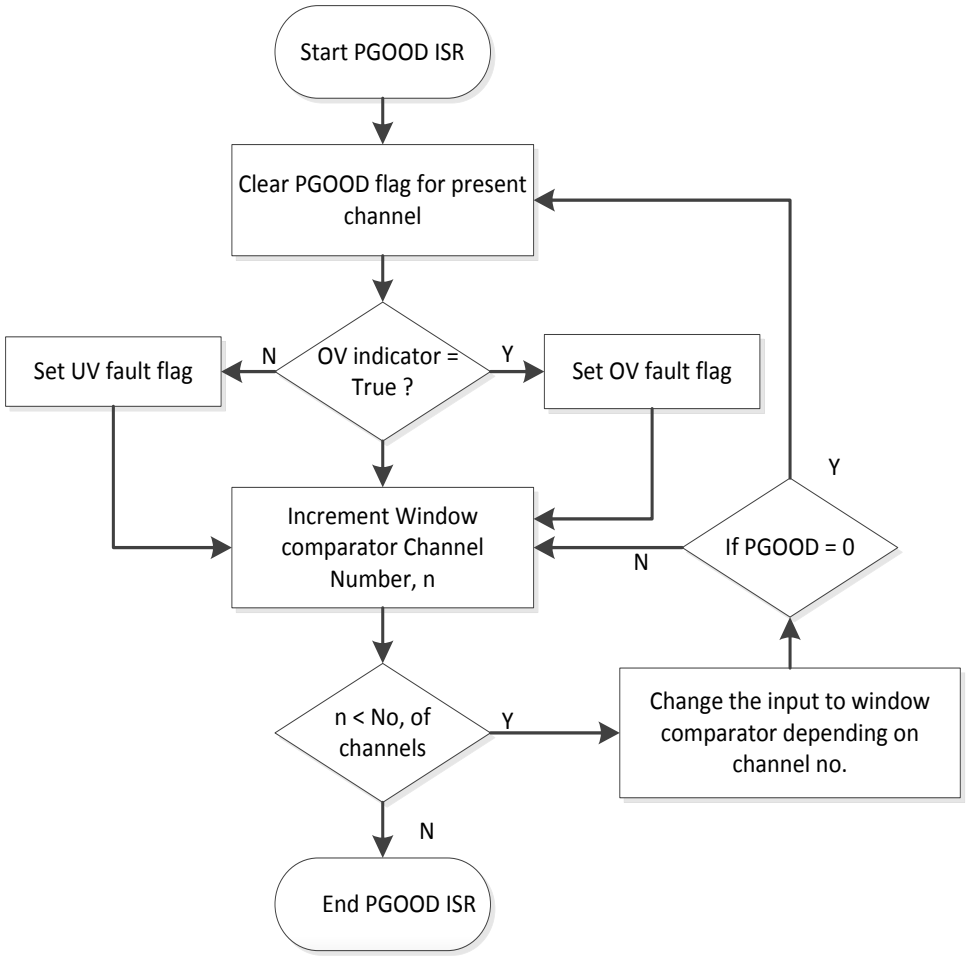


Figure 6. Flow Chart for PGOOD ISR or Fault Detection ISR



Voltage Monitoring

Voltage monitoring in a power management system consists of measuring all of the rail voltages. Use the measured rail voltages to trim and regulate the power consumption, based on the load connected to the rail. Also, the voltages can be communicated to the host so that it can data log the information.

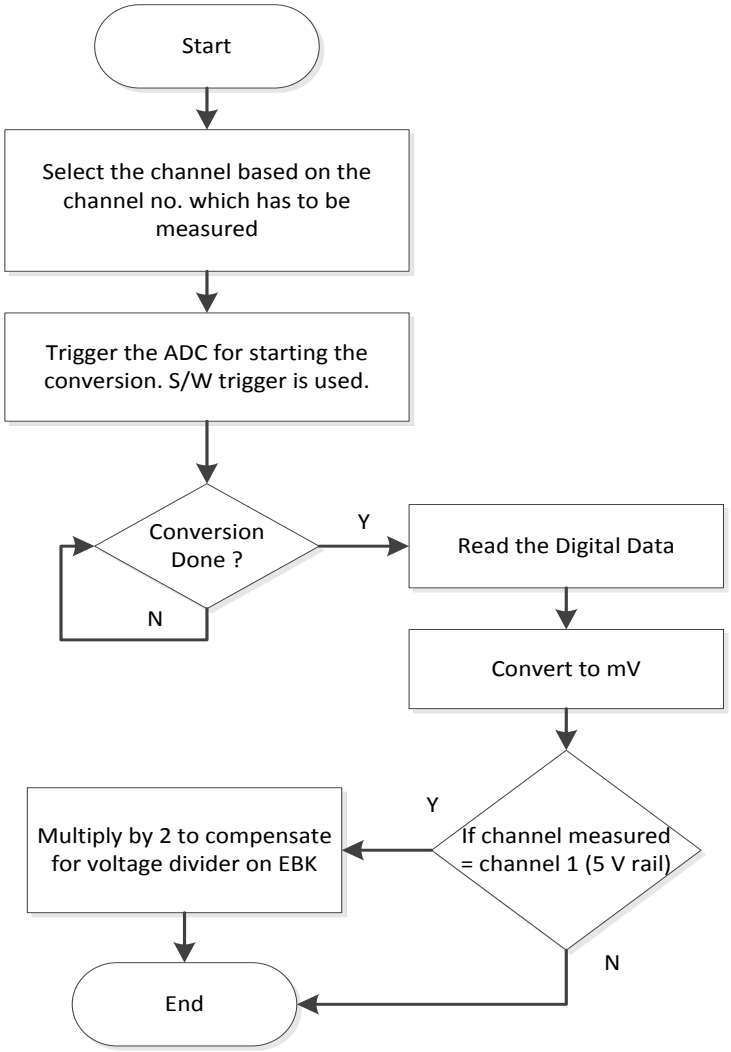
Voltage monitoring is not a time-critical feature compared to sequencing and fault detection. However, it has critical accuracy requirements. Measure the voltage with greater accuracy if the trimming is done using these voltages.

The requirement for voltage monitoring is an ADC with multiple input channels. The following applies to voltage monitoring and PSoC 1:

- A dedicated 10-bit SAR ADC has been used for measuring voltages of four rails on CY8CKIT-035. The reference for the ADC is derived from the on-chip Reference generator mux (RefMux) and the conversion range has been selected with 0 to 4.16 V. To measure the 5.0V rail, it has been scaled down to 2.5 V by using the voltage divider on the EBK.
- The measurement is done in one-shot mode. The ADC will be triggered for each measurement.
- Because the measured voltage is used for trimming, the voltage measurement and trimming are inter-related and occur sequentially.
- The four rail voltages from the EBK are multiplexed to the same SAR ADC.

Figure 7 shows the flow chart for voltage-monitoring firmware.

Figure 7. Flow Chart for Voltage-Monitoring Firmware



Current Monitoring

Current monitoring is implemented using a 14-bit ΔΣ ADC.

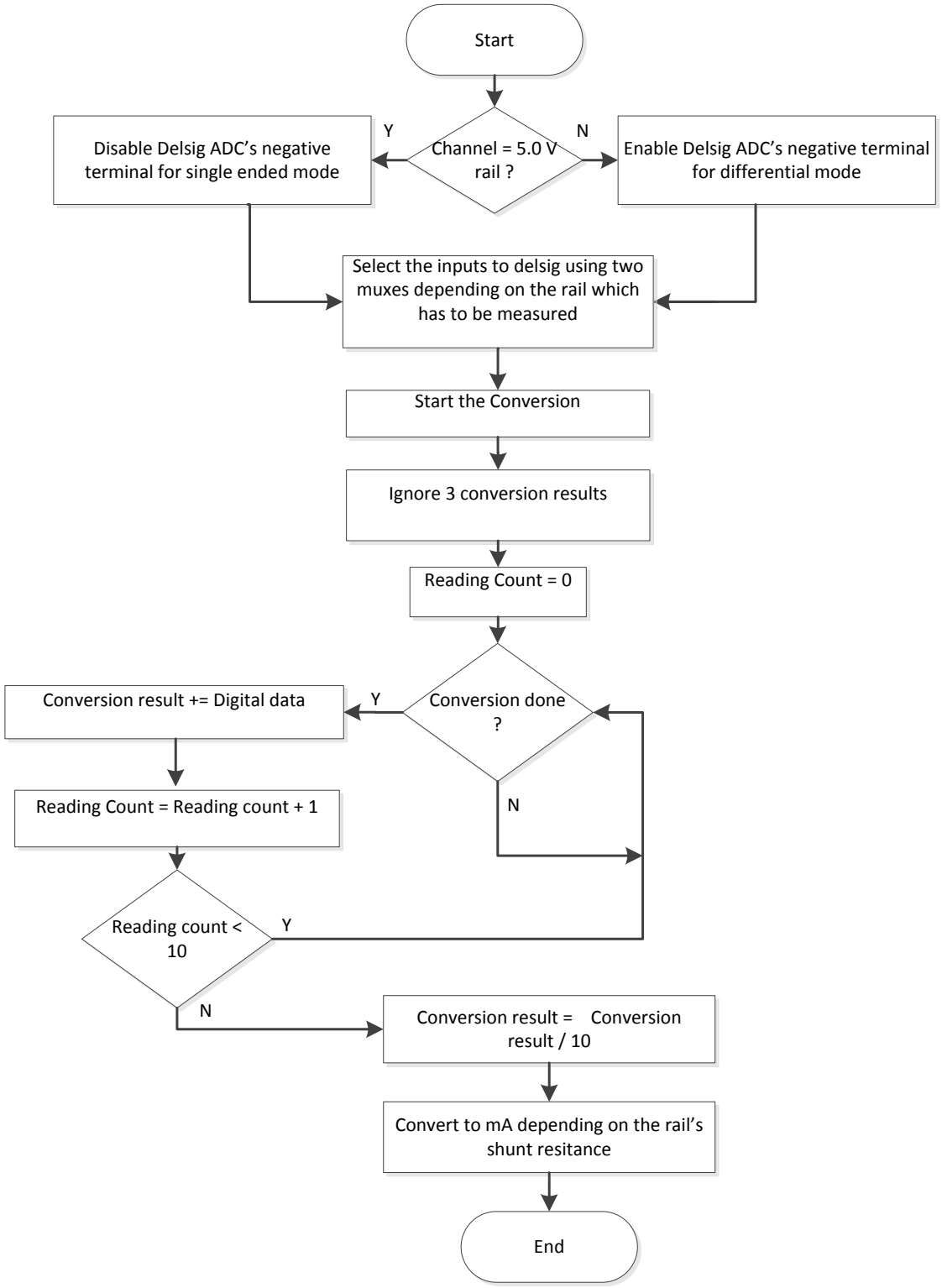
The ΔΣ ADC works with same reference voltage as that of SAR ADC; therefore, the conversion range is from 0 to 4.16 V. All four rails are multiplexed to the same ΔΣ ADC for the current measurement.

- For rail 1 (5.0 V), a high-side current sense amplifier on the EBK provides a single-ended output and is measured by the ADC in single-ended mode.
- For the remaining three rails, the differential mode of the ADC is used to sense the current from shunt resistors that are placed on high sides of the rail in the EBK. Two Programmable Gain Amplifiers (PGAs) with unity gain are used as buffers for differential signals.
- To nullify the offset and gain errors of the ADC, the zero voltage reading and the full-scale reading are measured and stored during startup. These are used to remove the effect of errors from further ADC readings.

At the time of multiplexing, after the input to the ADC is changed, three samples of digital data must be discarded. This is required because the DelSig works by oversampling the input and decimating the single-bit results. Because all output data will be dependent on analog input over a period of time, the conversions that happen just after changing the input will be improper and invalid.

Figure 8 shows the flow chart for current-monitoring firmware.

Figure 8. Flow Chart for Current-Monitoring Firmware



Trimming

Trimming is the ability to control the output voltage of the voltage regulator by providing the feedback to the regulator in a closed loop fashion. This feature is particularly attractive in two cases: (1) Systems that need to be forward-compatible with new voltage levels for loads such as next-generation memories or FPGAs, or (2) to increase accuracy of the regulator output voltage.

System designers can use either DAC or an RC-filtered PWM signal to control the regulator's feedback input so as to implement real-time trimming. The current example project uses the second method. The tolerance limit for each rail is defined by the user. When the monitored rail voltage (see Voltage Monitoring) is outside of the tolerance limit, the firmware changes the PWM duty cycle in either direction. The resultant PWM signal is then filtered by the external RC filter to generate the required feedback signal. This action brings the regulator output back within the tolerance limit.

The EBK has an RC filter for each regulator that does the filtering. The filter has a 3 dB frequency of 159.15 Hz for 3.3 V, 2.5 V, and 1.8 V regulators. For a 5.0-V regulator, it is 79.5 Hz. The PWM output frequency is selected such that at least 60 dB attenuation is achieved. The input clock for the PWM block is set at 48 MHz. For an 8-bit PWM implementation, the PWM output frequency will be 187.5 KHz, which is greater than the 60 dB attenuation frequency of 79.5K Hz, for a 5.0-V regulator, and 159 KHz for the other three regulators, thus assuring the required attenuation to generate stable analog output.

Choosing R1, R2, R3, R4 and C1:

Selecting R1 and R2 depends on the desired output voltage (Vout) and the design guidelines for the particular regulator being used. Refer datasheet of ADP3331 for more information.

Select R3 to limit Vout to a desired maximum voltage. R3 controls the amount of influence that the filtered PWM voltage has on the regulator output voltage, Vout. Choose R3 such that VoutMax is below the desired maximum. VoutMax occurs when the PWM duty cycle is 0%, essentially adding R3 in parallel with R2 (assuming for the moment that R4 is much smaller than R3 and, therefore, R4 can be temporarily ignored). Then when the PWM signal is at 0% duty cycle, Vfb is given by:

$$V_{fb} = \frac{V_{outMax}}{R1 + \frac{R2\ x\ R3}{R2 + R3}}\ x\ \frac{R2\ x\ R3}{R2 + R3}$$

Rearranging for R3,

$$R3 = \frac{V_{fb}\ x\ R1\ x\ R2}{V_{outMaz} x\ R2 - V_{fb} x\ (R1\ x\ R2)} \qquad (1)$$

R4 and C1 form a low-pass circuit to attenuation of the PWM signal and mitigate the ripple introduced at

Vfb and subsequently exposed on Vout. The corner frequency, Fc, is given below:

$$F_C = \frac{1}{2\ x\ \pi\ x\ R4\ x\ C1}$$

Rearranging, $$R4 = \frac{1}{2\ x\ \pi\ x\ F_C x\ C1} \qquad (2)$$

The amount of attenuation (in dB) of the PWM output frequency (FPWM) is given by

$$20\ x\ \log\frac{F_C}{F_{PWM}}$$

For example, to attenuate F_{PWM} of 187.5 kHz by -40 dB, then F_C should be 1.875 kHz. Using this F_C and picking a C1 value, R4 can be found out from equation (2) above.

Note that R4 should be much smaller than R3 to minimize changes to VoutMax from equation (1). A typically reasonable ratio between R3 to R4 is on the order of 10:1.

The trimming feature works based on the "Trim percentage" parameter. This configurable parameter indicates the amount to which the rail voltage has to be trimmed in terms of percentage of nominal voltage of that rail. Each of the rails can have a different trim percentage.

Figure 9 shows the feedback circuit for real-time trimming.

Figure 10 shows the flow chart for real-time trimming firmware.

Figure 9. Feedback Generation Circuit With Voltage Regulator

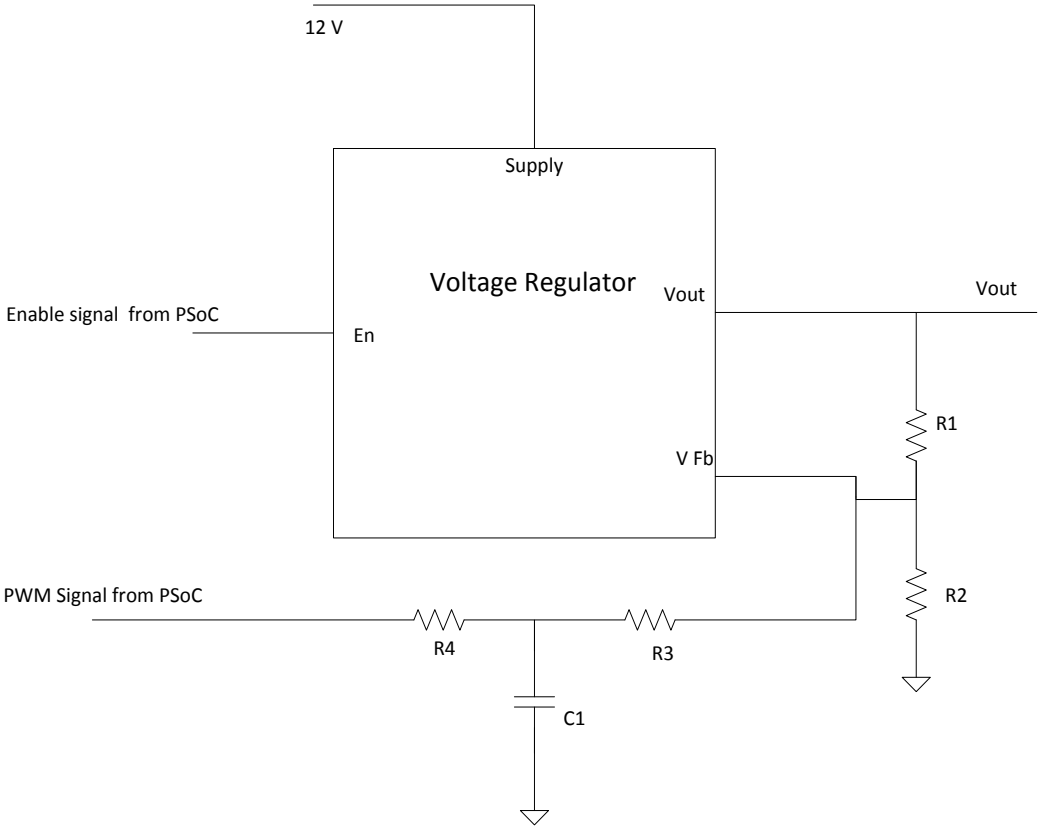
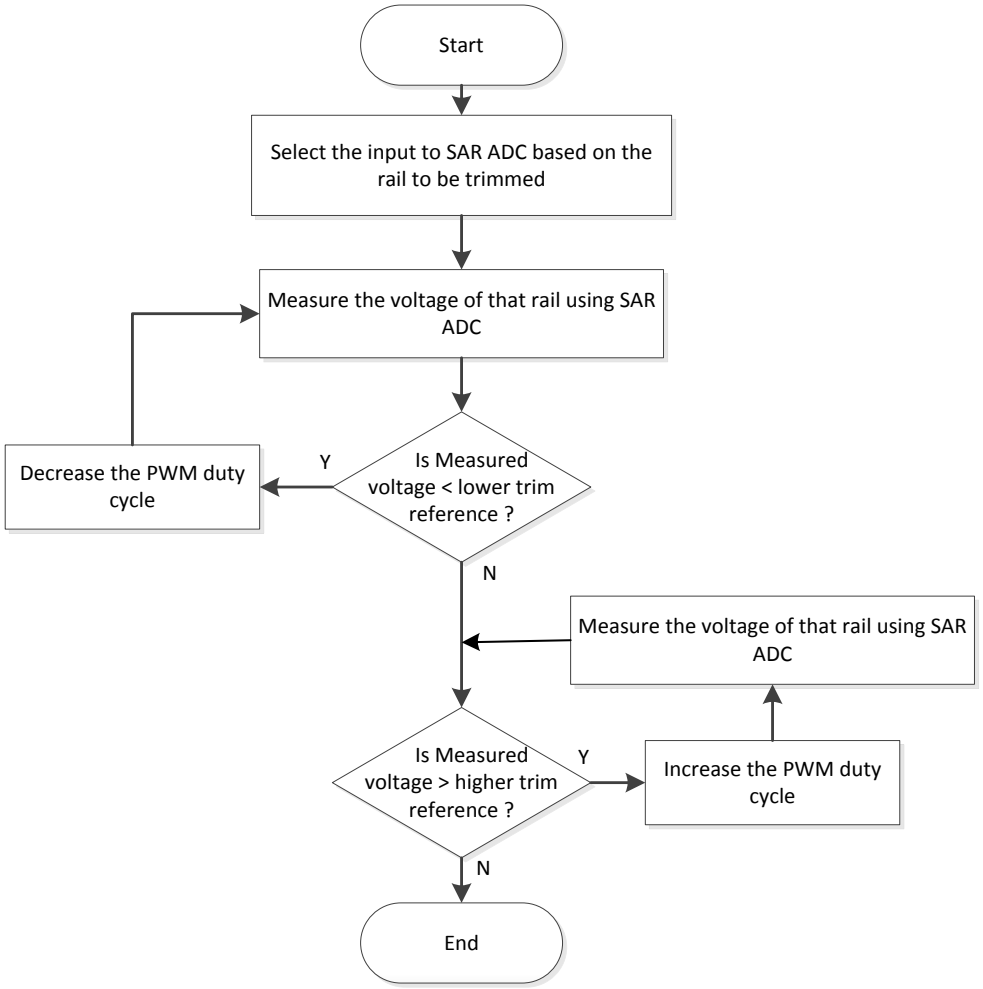


Figure 10. Flow Chart for Real-Time Trimming Firmware



Configuration Over I²C With On-Chip EEPROM

In some power management systems, the host processor communicates to the power management controller over I²C, SMBus, or PMBus for configuration, control, and diagnostics.

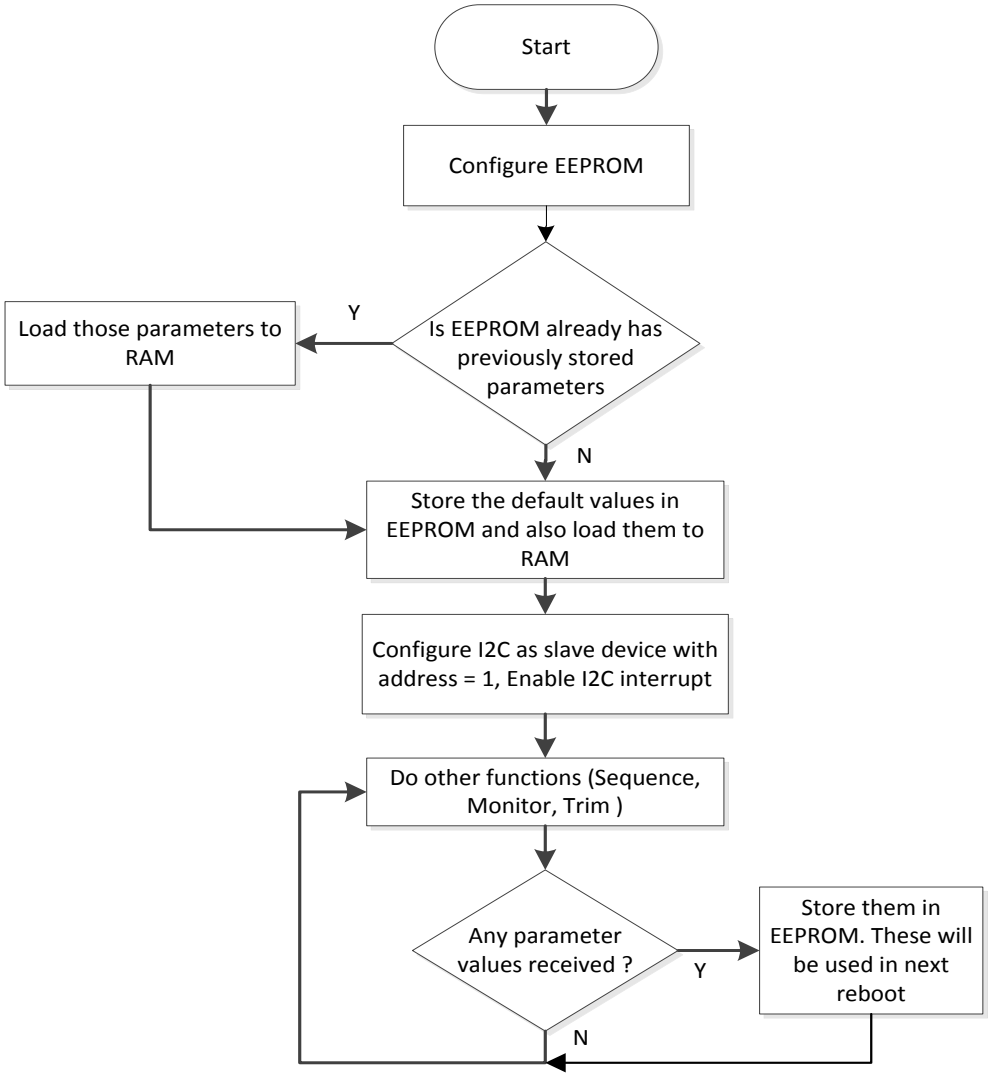
In this implementation, a PSoC is configured as an I²C slave and can receive the system parameters required for the operation from a host. PSoC 1 uses EEPROM to store these system parameters and to retrieve them during the next startup. These parameters are as follows:

- Sequencing mode
- Ramp delay for each rail
- Slot delay for each rail
- UV threshold percentage
- OV threshold percentage
- Trim percentage for each rail

The default values for these parameters will be used only when the controller is booted for the first time after programming; they will be stored in EEPROM until new values are received. The user can change the values of these parameters by sending the I²C commands from a master (Bridge Control Panel or I²C-USB bridge). These commands are explained in the next section. When the controller gets the new values, it stores them in EEPROM. These new values will be retrieved and used for the operation after the controller is restarted.

Figure 11 shows the flow chart for I²C communication and EEPROM storage.

Figure 11. Flow Chart for I²C Communication and EEPROM Storage Firmware



Example With PM-EBK

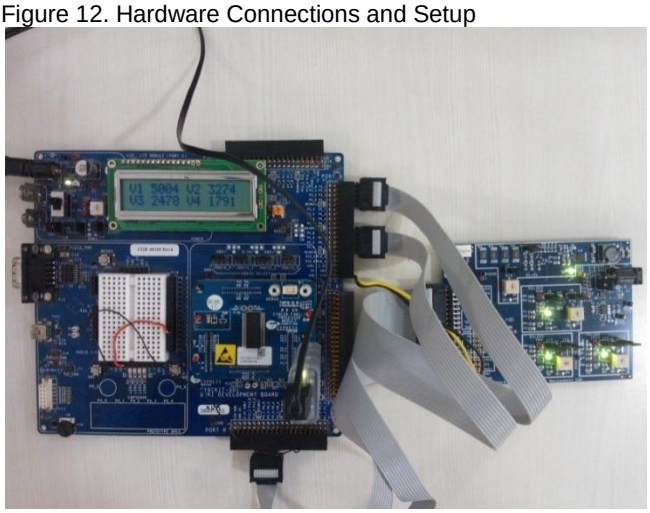
The example project is developed for the CY8C28645 PSoC 1 device using PSoC Designer. To demonstrate the power management functions, the following tools are required:

- CY8CKIT – 001, PSoC DVK
- CY8CKIT – 020, processor module (included in CY8CKIT – 001)
- CY8CKIT – 035, Power management EBK
- CY8CKIT – 002, PSoC MiniProg3 Program and Debug kit (comes with CY8CKIT – 001) or CY3217 – MiniProg1 and CY3240-I2USB Bridge
- PSoC Designer 5.3 IDE
- Bridge Control Panel software (automatically installed with PSoC Programmer).

Hardware Connections and Setup

Figure 12 shows the hardware connections and setup. A custom cable was used for testing and development.

Figure 12. Hardware Connections and Setup



The steps to set up the hardware are as follows:

1. Consult Table 1 to connect the CY8CKIT-035 to CY8CKIT-001

Table 1. Hardware Connections

PME EBK (CY8CKIT - 035) Pins	Ports on CY8CKIT - 001	I/O with regard to PSoC
V4EN - Pin 1	P3[6]	Digital Output, Enable signal
V3EN - Pin 2	P3[7]	Digital Output, Enable signal
V2EN - Pin 3	P3[4]	Digital Output, Enable signal
V1EN- Pin 4	P3[5]	Digital Output, Enable signal
AGND – Pin 9	AGND	----
V4S – Pin 11	P4[6]	Analog Input for window comparator
V3S – Pin 12	P4[7]	Analog Input for window comparator
V2S – Pin 13	P4[4]	Analog Input for window comparator
V1S – Pin 14	P4[5]	Analog Input for window comparator
TR4 – Pin 15	P4[2]	Digital Output, Driven by PWM
TR3 – Pin 16	P4[3]	Digital Output, Driven by PWM
TR2 – Pin 17	P4[0]	Digital Output, Driven by PWM
TR1 – Pin 18	P4[1]	Digital Output, Driven by PWM
AGND – Pin 19	AGND	----
V4P – Pin 21	P0[6]	Analog Input for Delsig ADC +ve pin
V4N – Pin 22	P0[7]	Analog Input for SAR ADC and Delsig –ve pin
V3P – Pin 23	P0[4]	Analog Input for Delsig ADC +ve pin
V3N – Pin 24	P0[5]	Analog Input for SAR ADC and DelSig –ve pin
V2P – Pin 25	P0[2]	Analog Input for Delsig ADC +ve pin
V2N – Pin 26	P0[3]	Analog Input for SAR ADC and Delsig –ve pin
I1 – Pin 27	P0[0]	Analog Input for Delsig ADC +ve pin
V1 – Pin 28	P0[1]	Analog Input for SAR ADC
AGND – Pin 29	AGND	----
DGND – Pin 37	GND	Digital Ground
DVIN – Pin 39	VIN , 12V supply	12 V supply from DVK to EBK
DGND – Pin 40	GND	Digital Ground

2. Make sure no other pins on the EBK are connected anywhere (NC). Pins P3[2] and P3[3] on the DVK should not be connected anywhere (NC), because P3[2] carries a PGOOD signal and P3[3] carries the OV fault indicator. Connect SW1 switch input on the DVK to P2 [7] on the DVK.
3. Connect LED1 to P1[3] on the DVK.
4. Jumper settings for CY8CKIT – 001 are shown in Table 2.

Table 2. CY8CKIT – 001 Jumper Settings

Jumper	Setting
J6	VDD_ANALOG to VDD
J7	VDD_DIG to VDD
J8	VDD to VREG
J12	LCD POWER to ON
SW3	5 V Position

5. Set CY8CKIT-035 to run from a 12-V DVK supply using J5 (Set J5 to DVK position).
6. Build the given example project and program the CY8C28xxx module using MiniProg3 or MiniProg1.
7. Connect the I^2C-USB bridge or MiniProg3 between the PC and the I^2C header on the processor module. Open the Bridge Control Panel and connect to the I^2C-USB/MiniProg3 in Bridge Control Panel.
8. Power the CY8CKIT-001 using a 12-V adapter.

Executing the Project

The following conventions are used for explanation:
Channel 1 -> 5.0 V rail
Channel 2 -> 3.3 V rail
Channel 3 -> 2.5 V rail
Channel 4 -> 1.8 V rail

The following events will occur after the DVK is powered up with the 12-V adapter:

1. The regulators are turned on in a sequential order with configured delays. This is indicated by the LEDs on PM-EBK associated with each regulator. The default sequencing order is Channel 1 -> 2 -> 3 -> 4. The default delays are:

   Ramp Delays of all channels = 10 ms

   Slot Delays of all channels = 100 ms
2. The measured voltages of each rail will be displayed on the LCD as shown in Figure 13. The trimming feature will be enabled for 1% threshold.

Figure 13. Voltages Displayed on LCD



3. If SW1 is pressed, the current delivered by each regulator will be measured and displayed as shown in Figure 14. Pressing SW1 again returns to the voltage display mode.

Figure 14. Currents Displayed on LCD



   Change the load resistances for each rail on PM-EBK by varying the potentiometer connected to each rail. This varies the current flowing through these loads. The varying current will be displayed on the LCD.
4. Simulate fault detection by removing jumper J6 on PM-EBK. This blocks the supply to the other three regulators and generates a UV condition on Rails 2, 3, and 4. After the fault is detected, the LCD displays the “Power Fail” Message and the rails that have the fault and the type of fault (OV or UV) as Figure 15 shows. Press SW1 to restart the sequencing.

Figure 15. Faults Displayed on LCD



   To simulate individual rail faults, the voltage on that particular rail has to exceed the UV/OV threshold. This can be done by disabling the trimming and forcing the output of the regulator to cross the thresholds with suitable PWM duty cycle. Increasing the PWM duty cycle decreases the output, and decreasing it increases the output. To disable trimming for channel 1, comment out the line "Trim_Regulator(CHANNEL_1);" in the State_Machine.c file.
5. To configure the parameters, use the Bridge Control Panel, which acts as an I^2C master, communicates to PSoC (slave), and sends the parameters. Use this command format in the Bridge Control Panel:

   w <7-bit Slave Address> <Write Address> <Sequencing Mode> <Slot Delay_1> <Ramp Delay_1> <Slot Delay_2> <Ramp Delay_2> <Slot Delay_3> <Ramp Delay_3> <Slot Delay_4> <Ramp Delay_4> <UV Threshold percentage> <OV Threshold percentage> <Trim Percentage_1> <Trim Percentage_2> <Trim Percentage_3> <Trim Percentage_4> <Sync Flag>

   Example: `w 01 0 02 03 E8 01 F4 03 E8 01 F4 01 F4 01 F4 01 F4 01 F4 4 4 2 2 2 2 1`

Permitted values for each field:

Slave Address: Use the list option in Bridge Control Panel to find out the 7-bit slave address. Use this for the slave address.

Write Address: 0

Sequencing Mode: 1 to 6

Slot Delay: Integer value (hexadecimal, two 8-bit fields in the command) from 1 to 1310 ms

Ramp Delay: Integer value (hexadecimal, two 8-bit fields in the command) from 1 to 1310 ms

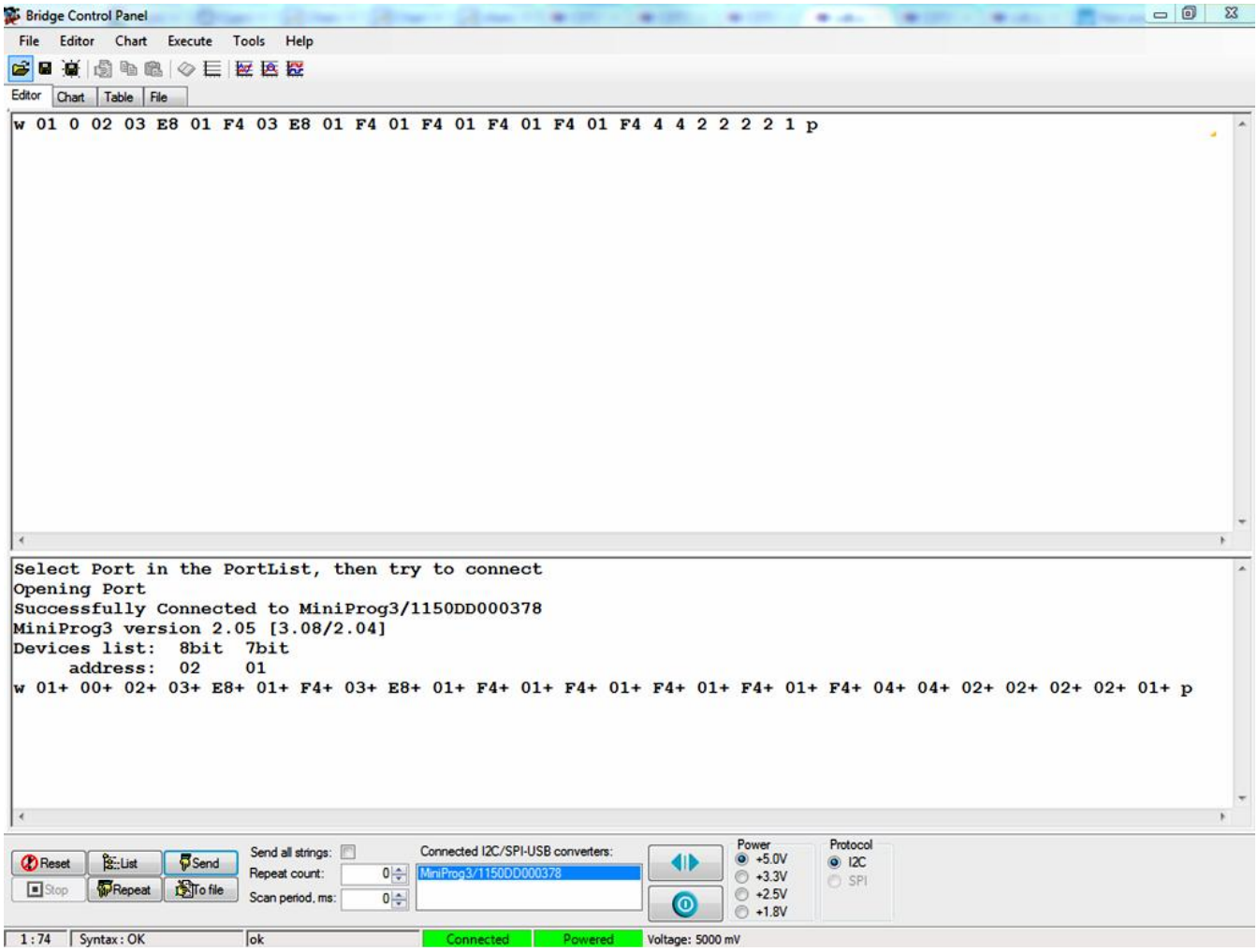
UV/OV Threshold percentage: BYTE value (hexadecimal); normally between 4% and 10%

Trim Percentage: BYTE value (hexadecimal); normally 1% to 3 %.

Sync Flag: 1

Figure 16 shows the screenshot of the Bridge Control Panel software with the example command to be used.

Figure 16. Screenshot of Bridge Control Panel Software



When this command is sent, PSoC receives the parameters and updates them to flash. While this happens, an LED connected to P1[3] will blink. If there is an error in writing to EEPROM, an error message "*Error in Writing to EEPROM, retry*" will be displayed on the LCD for a 1-second duration. The parameters must be resent from Bridge Control Panel. However, this error does not affect any other functionality.

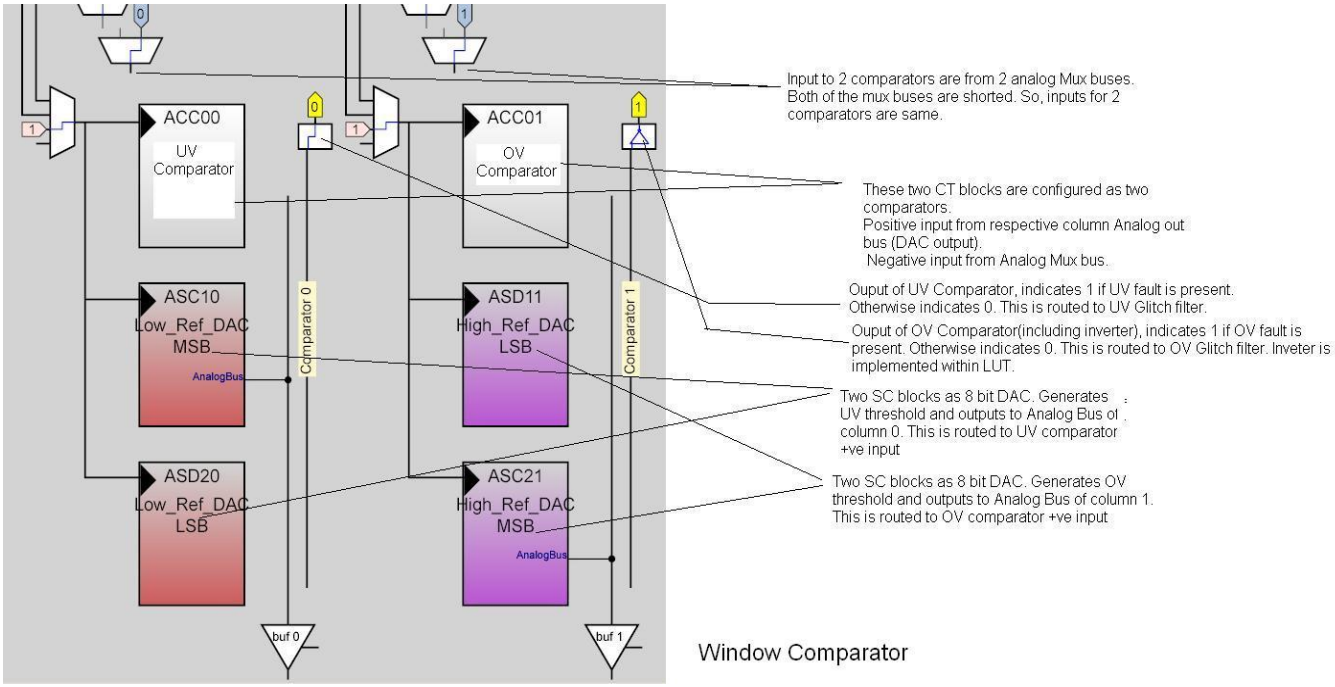
The next reboot of PSoC uses these parameters. However, when a power failure occurs, and if SW1 is pressed to restart the sequencing, the latest sequencing delays will be used. Apart from sequencing delays, no other new parameters will take effect immediately.

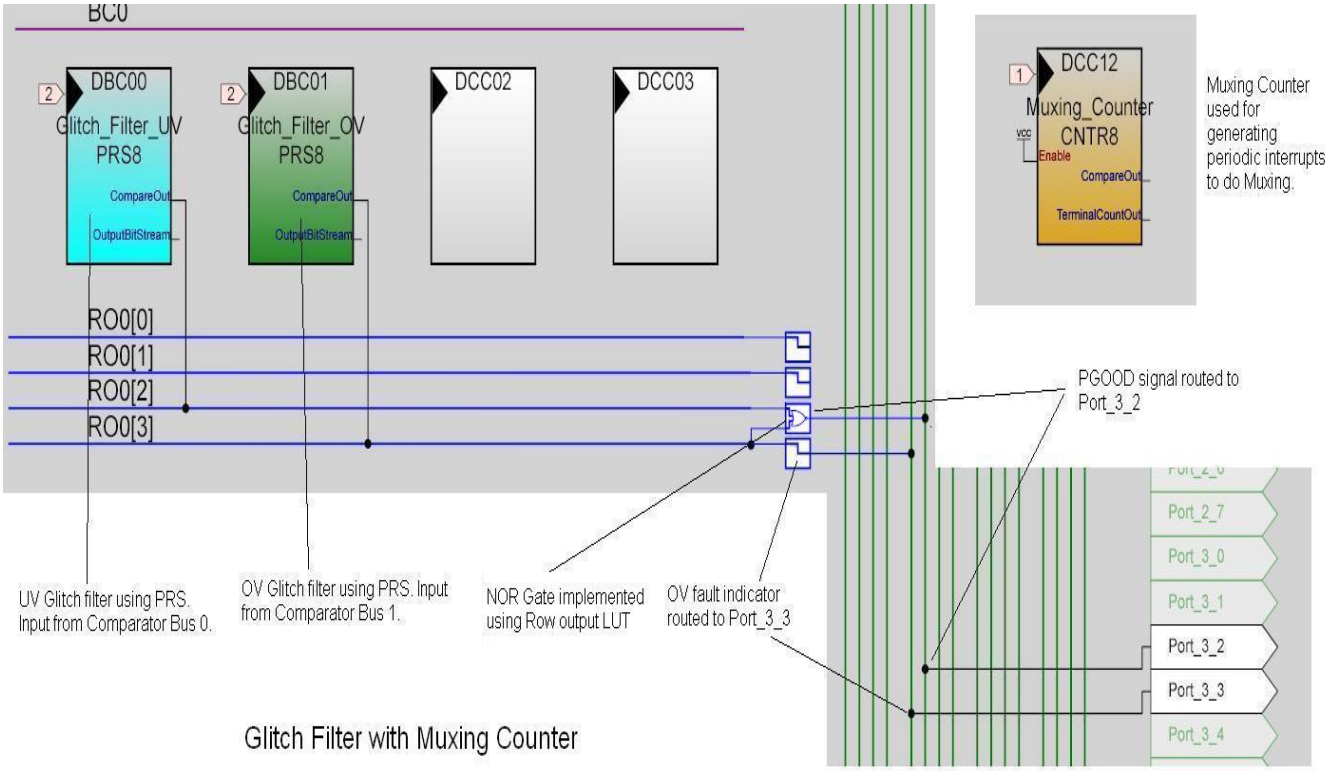
Summary

This application note introduced power management concepts and their implementation in PSoC 1, which enables designers to quickly start developing their own power management systems.

The unique ability of the PSoC architecture to combine digital logic, analog signal chain processing, and a CPU in a single device lets system designers integrate many external fixed-function ASSPs. This powerful integration capability not only reduces BOM cost, but it also results in less congested and more reliable PCB layouts.

Appendix A: Window Comparator Implementation





Why do you need an inverter at OV comparator output?

The UV comparator checks for under voltage fault and output will be high if there is a fault. The OV comparator checks for over voltage fault and output will be low if there is a fault. To make both of them assert a fault in a similar way (Logic High), an inverter is used to negate the OV comparator's output and make its output go high if there is a fault.

Inverting the inputs to the OV Comparator could have eliminated the inverter at the output. However, DAC output (OV threshold) can't be routed to the inverting pin of the comparator; therefore, you cannot invert the inputs. Also, implementing an inverter doesn't cost anything. It needs an LUT (Look Up Table) at the comparator bus, which has to be used in any case for routing the comparator output.

Why there is no UM used for the comparators?

There is no UM in PSoC Designer for the window comparator that can take reference input from a DAC. There is a "COMP" UM, which has window comparator version, but it allows only references that can be derived from Vss and Vdd. To generate finer threshold voltages, you need a DAC and you must route it to the input of the comparator. So, ACC00 and ACC01 CT blocks are configured as individual comparators with the following register writes:

```
/* UV Comp register settings */
ACC00CR0 |= 0x08; /* GAIN =1 */
ACC00CR1 &= ~0x80; /* Diables analog bus output */
ACC00CR1 |= 0x40; /* Enables comparator bus output */
ACC00CR1 |= 0x38; /* Neg input = Port pin = Analog Mux Bus*/
ACC00CR1 |= 0x06; /* Pos input = Analog bus = DAC output
ACC00CR3 |= 0x20; /* Disables AGND power */
ACC00CR2 |= 0x03; /* Turn On CT block */
/* OV Comp register settings */
ACC01CR0 |= 0x08; /* GAIN =1 */
```

```
ACC01CR1 &= ~0x80; /* Diables analog bus output */
ACC01CR1 |= 0x40; /* Enables comparator bus output */
ACC01CR1 |= 0x38; /* Neg input = Port pin = Analog Mux Bus*/
ACC01CR1 |= 0x06; /* Pos input = Analog bus = DAC output
ACC01CR3 |= 0x20; /* Disables AGND power */
ACC01CR2 |= 0x03; /* Turn on CT block */
```

DAC threshold generation example:
Assume that the UV fault is to be detected when the rail voltage is 5% below its nominal voltage level.

Nominal power rail input to the comparator $= 0.8705$ V

(scaled down level in the EBK)

$$\text{DAC reference} = 4.16 \text{ V}$$

$\text{DAC full scale value} = 256 \text{ count}$
$\text{UV threshold} = (1 - 0.05) \times (0.8705) = 0.8269 \text{ V}$

$$\text{Digital code} = \frac{\text{UV threshold}}{\text{DAC reference}} \times \text{DAC full scale value}$$
$$= \frac{0.8269}{4.16} \times 256 = 50.88 \cong 51 \text{ count}$$

How is the glitch filter implemented?
Our aim with the glitch filter is to eliminate the spurious glitches that appear in the comparator's output. For example, suppose the UV threshold is 0.8269 V and the nominal power rail input is 0.8705 V. For this input, the UV comparator outputs a 0 and doesn't indicate any fault that is expected. Now, suppose the power rail voltage is dropped to 0.83 V. Even though this is within the expected range, the comparator output won't be at 0; it fluctuates between 0 and 1. This can happen even with the OV comparator when input becomes nearly equal to the OV threshold. So, to fairly detect the fault, you must ignore these rare glitches (transitions from 0 to 1 and 1 to 0).

In this present implementation, the clock input given to the comparator block is 3 MHz output of the comparator updates at 3MHz/4 = 750 KHz (for updates at PHI 2 phase,. See the CT block section in the CY8C28xxx PSoC Technical Reference Manual for more details on clock phases). This means that the comparator bus updates its output every 1.333 µs. To fairly say that there is a fault, you must monitor this comparator output for, say, four updates. If it indicates fault for all four updates, you can be sure that there is a fault. By trial and error, you can find out how many updates are required for a fair response by neglecting glitches. In the present implementation, four updates are monitored, which corresponds to 4 * 1.33 = 5.333 µs.

The PRS user module (UM datasheet) can be used as a glitch filter. The PRS normally works by receiving an input data stream and producing an output data stream after encrypting it based on the LFSR Polynomial given. But here, the encryption feature is not needed and therefore the polynomial is set to 0. One more function of PRS is that it always compares the data in its shift register with the data in its seed register and outputs an auxiliary "compare out" depending on the equality between these register contents. For a glitch filter, this functionality of PRS is used. One 8-bit PRS is needed for one glitch filter.

The input data stream to PRS is the UV/OV comparator's output. This is sampled and shifted to shift register on each clock of PRS. At each clock, the contents are compared with seed register. If they are equal, the comparator output becomes logic high, otherwise low. If you set the seed register to 11111111, PRS compares the eight samples of comparator bus output which are shifted into shift register with 11111111b. Only if all 8 samples are equal to 1, then a fault is indicated through "Compare Out" output of PRS (compare out indicates logic high).

To make sure that four updates of the comparator and eight samples of PRS are equal, the clock of the PRS should be set accordingly. For this, the time required for PRS to sample 8 times must be equal to the time required for the comparator to update its output 4 times. Because the comparator bus updates at a 750 KHz rate, the PRS should sample at 1.5 MHz clock (750K x 2).

By using this glitch filter, you can eliminate any glitches in the comparator bus that are less than 5.33 µs (4 updates of the comparator). The glitch filter has been configured to operate this way using both register settings and PRS APIs.
For a UV glitch filter, the following are the configuration operations:

```
DBC00IN &= ~0xF0;   //Clear Higher 4 bits (Data input selection bits)
```

```
DBC00IN |= 0x40;    //Input to UV glitch filter is comparator bus 0
DBC00IN |= 0x80;     //Inverts the data input
Glitch_Filter_UV_WritePolynomial(0x00);    //Sets LFSR function polynomial to 0x00
//Sets seed value to b11111111, this ignores any signal with glitches less
than 8 clock pulses()
Glitch_Filter_UV_WriteSeed(0xFF);
Glitch_Filter_UV_Start(); //Starts PRS block
```

A similar configuration is used for the OV glitch filter.

What is the maximum (worst case) fault detection time?
The maximum time to detect a fault is defined as the time delay between the moment that the actual fault on the rail happens and the moment PSoC detects it by generating a PGOOD interrupt.

Each rail has a 10-µs slot out of a 40-µs cycle time of the window comparator where the fault can be detected for that particular rail. To detect a fault on a rail that is currently being monitored, the fault has to occur within the first 4.66 µs (10–5.33 µs) out of a 10-µs slot, because of glitch filter latency. If a fault occurs after 4.66 µs, then that fault cannot be detected in the current muxing cycle. Also, in the next cycle, the glitch filter needs at least 5.33 µs to detect a fault.

Therefore, the worst case time would be (10 µs * 3 other channels + 2* 5.33 µs) = 40.66 µs. The minimum time would be glitch filter latency = 5.33 µs.

Why is the output of the OV glitch filter routed to a pin?
The OV fault indicator is routed to a GPIO. Interrupt is not enabled for this pin. When the PGOOD interrupt occurs, it can be due to either UV or OV fault. To detect whether it is OV fault or UV fault, this OV indicator is checked for its status inside the ISR. If is indicates OV fault, then you can consider the fault as OV. Otherwise, you can consider the fault as UV.

Document History

Document Title: Integrated Power Manager using PSoC® 1 – AN78646

Document Number: 001-78646

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3612131	KERI	05/22/2012	New Spec
*A	3882793	KERI	02/20/2013	Updated the software version to PSoC Designer 5.3. Added formula and explanation about choosing resistors and capacitors for trimming circuitry. Updated the abstract. Minor text edits.
*B	4387265	SNVN	05/22/2014	Sunset Review.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

PSoC is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2012-2014. 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.
This 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.