

Voltage Monitoring and Sequencing with PSoC™

By: Ernie Buterbaugh
Associated Project: Yes
Associated Part Family: CY8C27xxx
PSoC Designer Version: 4.0

Summary

Many systems require a variety of voltages to power the ASICs, FPGAs, and standard logic found in today's designs. To further complicate the design, some of these devices require voltages to power on in a specific sequence, lest damage to the chip will occur. There are a variety of voltage monitoring and sequencing devices available today. However, they can be expensive and often don't meet the needs of the design since they offer little or no programmability. PSoC is an excellent choice for monitoring and sequencing the voltages. With its analog structure and programmability, the PSoC can be a perfect fit for the most stringent designs.

Introduction

This Application Note describes a system that uses eight different voltages. The PSoC monitors each of the eight rails and controls the voltage-enables such that they power on in sequence. Subsequent rails are activated only after the previous rail rises to a minimal voltage level. All the voltages are constantly monitored and if any voltage falls below a minimum value, the appropriate enables are turned off. The configuration can be easily adapted to satisfy other designs.

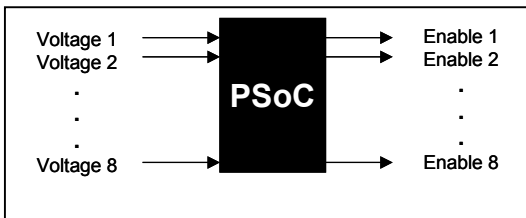


Figure 1. Voltage Monitor

Monitoring Voltages

The first step is to monitor all the voltage rails. Burdened with large capacitors, voltage rails tend to change very slowly. This allows a variety of options for capturing the value. The PSoC has resources for voltage comparators and also A/D converters. To allow the most flexibility, we'll choose the ADC method for this implementation.

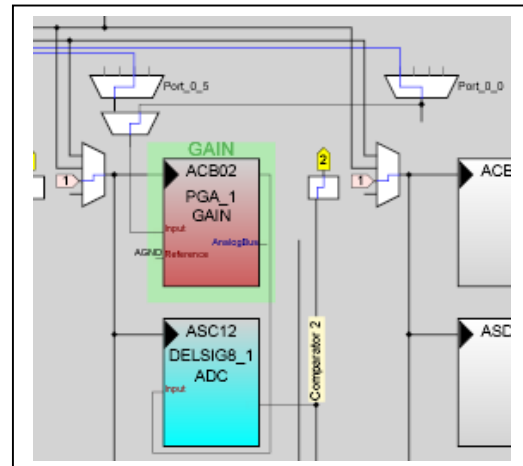


Figure 2. PGA and ADC Placement

There are several ADCs from which to choose. The simple SAR6 is an easy-to-use ADC and will suffice for many applications. This project will use the more advanced DELSIG8 8-bit delta sigma ADC for its increased resolution and accuracy. A larger 11- or 12-bit delta sigma ADC may be substituted if more accuracy is desired. With the DELSIG8, as is the case with all the delta sigma A/Ds in PSoC, an interrupt occurs whenever a conversion is complete and a digital value is ready.

Since there are eight analog inputs that we need to monitor, we'll take advantage of the 8:1 analog input mux and use only one ADC. This allows us to sequence through each rail and capture the value. The 8:1 mux feeds into a programmable gain amplifier (PGA) and then the ADC. The default position of the mux set in the Device Editor is Port0[0] – which corresponds to Voltage_1.

The gain of the PGA is set to 1X. However, depending on the value of the voltage rail, the PGA can be used to increase the gain and hence, increase the resolution of the ADC. Because of the programmable nature of the PSoC, the gain can be selected on a rail-by-rail basis. The gain of the PGA can be set in the same Interrupt Service Routine (ISR) that changes the analog mux.

Note that the input voltages have to be scaled down to 2.6V for a RefMux of Bandgap \pm Bandgap and to V_{CC} for a RefMux of $V_{CC}/2 \pm V_{CC}/2$ by external resistor dividers for proper operation. The project demonstrates the operation with voltage steps from 0.75V to 2.25V in 0.25V steps. The thresholds can be modified according to system requirements.

Referring to the ISR code in Figure 3, every time the ISR is entered, i.e., the ADC has completed an acquisition, the Trash_Can variable is checked. If the value is higher than zero, the ADC has not been sampled enough times since the analog mux had switched. After a mux change, three samples of the ADC need to be *trashed* to allow for the internal analog paths to stabilize. A trash counter is used to discard the first three samples until data is passed to the *main.c* routine. Also, if the data that was captured before hasn't been taken by the main routine, don't update with the new data yet – just return.

We are now at the point where the data is good and is available in Register A. The value is in 2's complement and this may be the desired form. However, if 1's complement better suits the needs of the design, a simple conversion method is to XOR the MSB bit as shown in the example.

The Trash_Can variable is reset next in the ISR code since we are about to change the analog mux position. We also save the current mux position that corresponds to the captured data and then specify the next mux position. If less than eight inputs are used, the next mux position calculation is adjusted to switch to only the valid positions. The 8:1 mux is made from two 4:1 muxes;

in this case AMUX3 and AMUX2. We need to select the position in the AMUX2 or AMUX3 for the selected voltage and choose which of the two muxes is selected. Writing to the ABF_CR0 register in the PSoC selects which mux is gated to the PGA. Note that this register is in register Bank 1 and we use the M8C_SetBank1 macro to access it. If the gain of the PGA is to change, the code (a call to PGA1_Set_Gain) is placed in this section as well.

```

;;; START ISR APPNOTE MODS
push x
dec [_Trash_Can]      ; throw away after mux switch
jnz end_ret
cmp [_ADCFlag], 1    ; data not yet taken
jnz getdata
; data was ready but main loop didn't take it yet
; restore trash counter for next pass
inc [_Trash_Can]
jmp end_ret

getdata:              ;; Data in ADC is good

xor a,80H             ; Convert to 1's comp
mov [_ADCData], a    ; Save data
mov [_Trash_Can], 04H ; Reset trash counter
mov [_ADCFlag],1     ; Data available flag
mov a, [_MUXPosition] ; Save mux position
mov [_MUXData], a
inc a                 ; now set new mux position
and a, 07H            ; only 0-7 positions
mov [_MUXPosition],A ; save away

;; -----
;; This code is to select one of the 8 analog input ports
;; A has value 0-7 selecting which port
;; Value: 0: Port 0_0 Volt_1      AMUX3
;; Value: 1: Port 0_2 Volt_2      AMUX3
;; Value: 2: Port 0_4 Volt_3      AMUX3
;; Value: 3: Port 0_6 Volt_4      AMUX3
;; Value: 4: Port 0_1 Volt_5      AMUX2
;; Value: 5: Port 0_3 Volt_6      AMUX2
;; Value: 6: Port 0_5 Volt_7      AMUX2
;; Value: 7: Port 0_7 Volt_8      AMUX2
;; -----

cmp A,04H             ; If mux <4 then do amux3
jnc amux2

amux3:
call AMUX4_3_InputSelect ; set mux in column 3

M8C_SetBank1
or Reg[ABF_CR0], 0b01000000 ; Column 3
M8C_SetBank0
jmp end_ret

amux2:

call AMUX4_2_InputSelect ; set mux in column 2

M8C_SetBank1
and Reg[ABF_CR0], 0b10111111 ; Column 2
M8C_SetBank0

end_ret:
pop x
;;
;;; END ISR APPNOTE MODS

```

Figure 3. ADC ISR

Sequencing Voltages

Now that the voltages are known, they can be checked for validity and the appropriate enables turned on or off.

The *main.c* section of the code, as the extension suggests, is written in C. C allows the code to be easily maintained and modified for a variety of system needs. The approach used in this design takes advantage of the switch-case statement to build a state machine. The state machine concept provides the ability to add states as well as allows voltage enables to be sequenced in a specific order and easily modified.

Many power systems require a delay between the time a rail powers on and the subsequent power enable. The state machine design allows for this delay to be added with ease. In this example, a 10-ms delay is used between the detection of the rail and the activation of the next rail. A function called *Start_Timer()* allows the program to set the delay in 1-ms increments, simply bypassing the amount of delay time in ms to the subroutine. The timer is implemented in a digital block and causes an interrupt once the time passed to the routine expires.

As each power rail turns on and reaches the minimum value, the state machine will delay 10 ms and then turn on the next rail. This continues until all rails are on and the state machine reaches the idle state. Should any voltage fall below the minimum value, all the enables that were turned on after that power rail had come up, are turned off. The state machine then continues at that position in the sequence, waiting for the rails to become active. A small hysteresis of two counts is introduced during testing if the voltage has fallen below threshold. This eliminates any unwanted oscillations in the outputs.

The code has provisions for a minimum and a maximum voltage value. This provides the ability to window the valid range – that is, monitor for under voltage as well as over voltage. The demonstration code only uses the minimum value. Adding statements to flag and shut down when the maximum is exceeded can easily be done.

Resolution

So how accurately can the PSoC capture values? Of course that depends on a variety of factors such as the size of the ADC, the reference voltage, the gain of the PGA, and a few other factors. So let's take a couple of examples and show how the numbers compare.

There are several choices for the RefMux setting but let's take a look at two: BG +/- BG and $V_{cc}/2 +/- V_{cc}/2$. The bandgap (BG) reference voltage of the PSoC is 1.3V with a tolerance of 1%.

The BG +/- BG setting translates into a range of 0 to 2.6V. If you use the 8-bit ADC, this yields $2.6V / 255 = 10.196$ mV per tick. If the $V_{cc}/2 +/- V_{cc}/2$ is used and V_{cc} is 3.3V, then the range to the ADC is 0 to 3.3V. This yields $3.3V / 255 = 12.941$ mV per tick. In the V_{cc} case, the accuracy of the reference is subject to the variation in the V_{cc} supply. This is often greater than the 1% of the BG reference voltage.

Offset Voltages

Like all amplifiers and ADCs, there is a DC offset voltage that is added to the incoming signal. For 3.3V operation, the PGA has a maximum offset voltage of 3.5 mV and the ADC is 5 mV. For many voltage-monitoring systems, this amount of error is inconsequential and can be ignored.

However, for those systems where the DC offset needs to be minimized, the PSoC can be *calibrated* to zero-out the DC offset. This is accomplished by bringing in a reference voltage through the same path as the voltages we are monitoring.

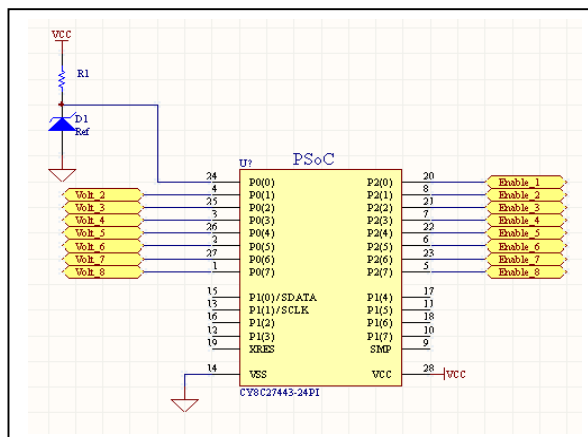


Figure 4. Voltage Monitor with Reference

The value captured from the ADC when sampling the reference voltage is now known to be exactly the reference voltage – i.e., the ADC is *calibrated*. If using the BG +/- BG as the reference, each tick from the calibrated value is a 10.156 mV delta, as discussed previously. Note that the BG still has a 1% tolerance.

Reading and calibrating to the reference voltage every pass through the mux eliminates any drift. The voltage can be from a precision external reference such as an LM4040.

Other Options

This design example has illustrated a system where the voltage values control only the output enables. However, this system can easily be expanded to provide a more capable device. One popular addition is an external serial bus that allows the reading of voltage levels. Since the PSoC supports many serial interfaces including I²C, SPI, and UART, the user can select which one best fits the system. As the voltages levels are captured, they are stored in locations for later retrieval. The serial interface can access the data and send it back to a service processor. Another approach is to create alarms that trip when the voltage is low or high and send back the alarm information to the service processor. Either way, the data is there and easily put into a format that best suits the system's needs.

Another function that can be added is a voltage-margining control. Some systems require voltages to be increased and decreased though a range to test how well the design operates in over voltage and under voltage situations.

Since the PSoC is already monitoring voltages, a function can be added to control the output of the voltage regulators. Depending on the power supply, the interface to control the output voltage may be an analog control such as a DAC or it may be a serial bus such as I²C or SMBus. Either way, the PSoC has the capability to control and adjust to meet the necessary output levels.

Conclusion

The PSoC can be easily configured to monitor up to eight different voltages. It can also supply a variety of enables and status lines that allow tailoring to any system need. Expensive linear devices typically used for voltage monitoring can be replaced with a low-cost PSoC (and you get an MCU for free).

About the Author

Name: Ernie Buterbaugh
Title: Principal Field Applications Engineer

Contact: ewb@cypress.com

Cypress MicroSystems, Inc.
2700 162nd Street SW, Building D
Lynnwood, WA 98037
Phone: 800.669.0557
Fax: 425.787.4641

<http://www.cypress.com/> / <http://www.cypress.com/support/mysupport.cfm>

Copyright © 2003-2004 Cypress MicroSystems, Inc. All rights reserved.

PSoC™ (Programmable System-on-Chip) and PSoC Designer are trademarks of Cypress MicroSystems, Inc.

All other trademarks or registered trademarks referenced herein are property of the respective corporations.

The information contained herein is subject to change without notice.