Chapter 8: Supplemental Material

This chapter contains additional information about the electronics referenced in this course. It is intended for the people taking this course without a strong background in electronics.

Table of contents

Document conventions

Convention	Usage	Example
`Courier New`	Displays code and text commands	`CY_ISR_PROTO(MyISR);` `make build`
Italics	Displays file names and paths	*sourcefile.hex*
[**bracketed, bold**]	Displays keyboard commands in procedures	[**Enter**] or [**Ctrl**] [**C**]
Menu > Selection	Represents menu paths	**File > New Project > Clone**
Bold	Displays GUI commands, menu paths and selections, and icon names in procedures	Click the **Debugger** icon, and then click **Next**.

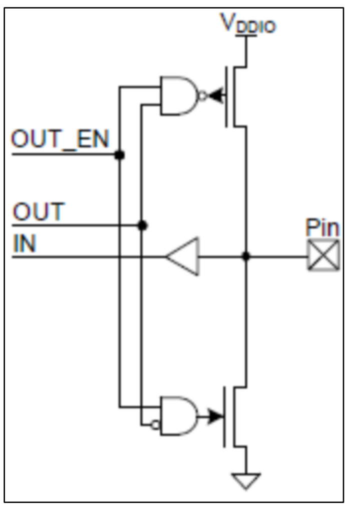
8.1 Internal Components

8.1.1 General Purpose Input Output (GPIO) (a.k.a. Pin)

GPIOs are the interface between the PSoC™ and the outside world. They allow you to connect the PSoC™ to just about any type of external component, be it analog or digital. To enable this functionality, PSoC™ GPIOs can take on any one of 14 different drive modes. Each drive mode is a combination of of 7 output configurations along with the selection to enable or disable the input buffer. When enabled, the input buffer allows the CPU to directly read a digital value from the pin. The output configurations that PSoC™ GPIOs can take on are the following:

Note: The triangles in these schematics represent input buffers. Analog connections are taken directly from the pin

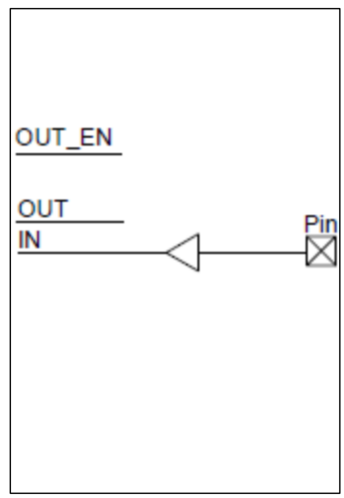
Strong Drive



- This mode allows you to perform digital output operations. Although you can technically perform digital input operations in this mode it is not recommended.
- This mode is often used to drive LEDs.

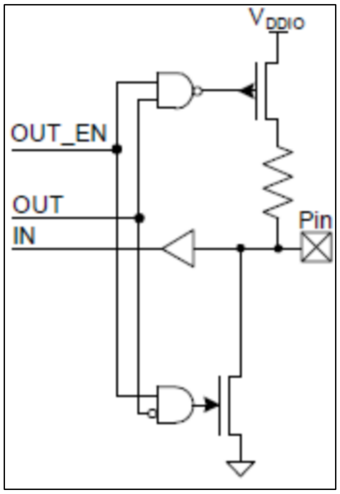
Digital High-Z, Input buffer on

Analog High-Z, Input buffer off



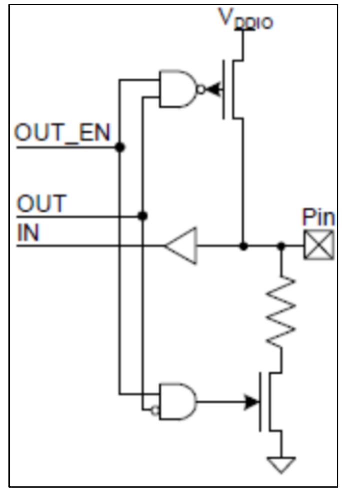
- This mode allows you to perform analog input and digital input operations. When inputting analog signals the input buffer is disabled.

Resistive Pull Up



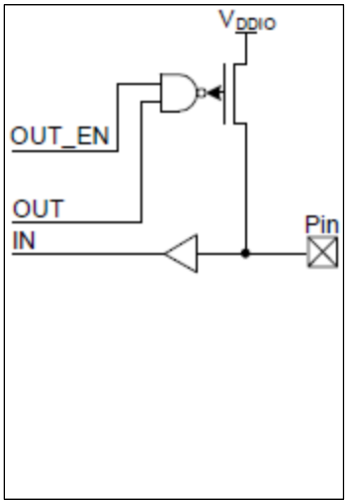
- This mode allows you to perform both digital input and digital output operations. In this state, the pin is internally driven to a "1" through a resistor, but when it is internally driven to a "0" it is directly connected to ground. The resistive "1" value can be overpowered by a stronger "0" driver from another device on the board. This allows you to drive the pin to a "1" while simultaneously allowing external components to drive the pin to a "0". This is often used for wired-or configurations such as I2C when there is no external pullup resitor on the board.
- This mode is often used for mechanical buttons that are active low. The output is set to a 1 so that it will pull the pin high when the button is not pressed.

Resistive Pull Down



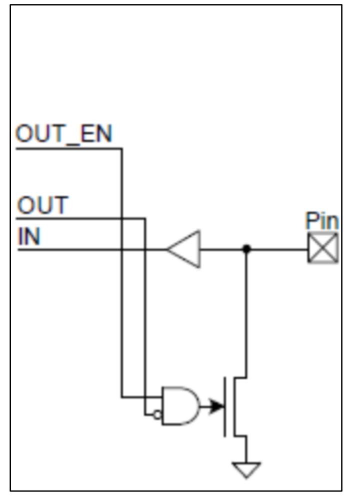
- This mode allows you to perform both digital input and digital output operations. In this state, the pin is internally driven to a "0" through a resistor, but when it is internally driven to a "1" it is directly connected to ground. The resistive "0" value can be overpowered by a stronger "1" driver from another device on the board. This allows you to drive the pin to a "0" while simultaneously allowing external components to drive the pin to a "1".

Open Drain Drives High



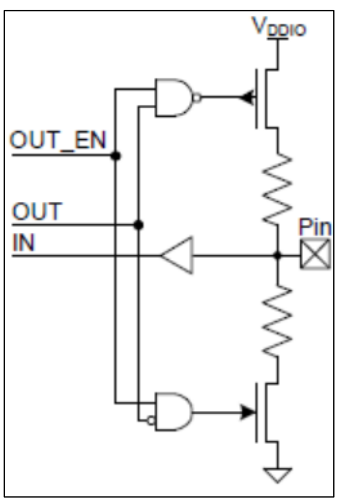
- This mode allows you to perform digital input operations, but only allows you to ouptut a "1". In this mode there is no way to drive the pin to a "0".

Open Drain Drives Low



- This mode allows you to perform digital input operations, but only allows you to ouptut a "0". In this mode there is no way to drive the pin to a "1". This is often used for wired-or configurations such as I2C when there is an exernal pullup resistor on the board.

Resistive Pull Up and Pull Down



- This mode allows you to perform both digital input, digital output, and some analog input operations. In this state, the pin is driven to both a "1" and a "0" through a resistor. Resistive drives can be overpowered by stronger drivers on the board. This allows you to drive the pin to either a "1" or a "0" while simultaneously allowing external components to drive the pin to the opposite value.

8.1.2 TCPWM

Most PSoC™ MCUs include one or more instances of Infineon's TCPWM HW block. This is a programmable digital HW block that can be configured to implement several different commonly used MCU peripherals:

- Timer/Counter
- Pulse Width Modulator
- Quadrature Decoder
- Shift Register

Note: The shift register mode is not available on all devices with TCPWM blocks.

We will broadly discuss the functionality of the timer/counter and pulse width modulator modes. The quadrature decoder and shift register modes however are out of the scope of this course. If you're interested in them you can refer to the corresponding CAT1 PDL documentation sections:

- CAT1 PDL Quadrature Decoder
- CAT1 PDL Shift Register

There are two different versions of the TCPWM block, for details and specifics about each version you can refer to the CAT1 PDL TCPWM Documentation.

8.1.2.1 Timer/Counter

A counter does exactly what its name implies, it counts. More specifically, it counts digital events (rising edges, falling edges, etc.). When counters are used to count events that occur with a fixed frequency (i.e. the edges of a clock signal), they are often called timers, as each incrementation of the counter's value corresponds to a fixed period of time. Some common use cases of counters are:

- Creating a periodic interrupt for running other system tasks
- Measuring frequency of an input signal
- Measuring pulse width of an input signal
- Measuring time between two external events
- Counting events
- Triggering other system resources after a certain number of events
- Capturing time stamps when events occur

At a minimum, this peripheral requires two digital signals to operate: a clock signal and an input signal. A counter will count digital events that occur in these two signals. These events can be several different things depending on how the counter is configured:

- Rising edges of the counter input
- Falling edges of the counter input
- Both rising/falling edges of the counter input
- Both rising/falling edges of the clock signal while the counter input is high

Note: The block is synchronous, so a rising/falling edge on the counter input are detected at the next rising clock edge.

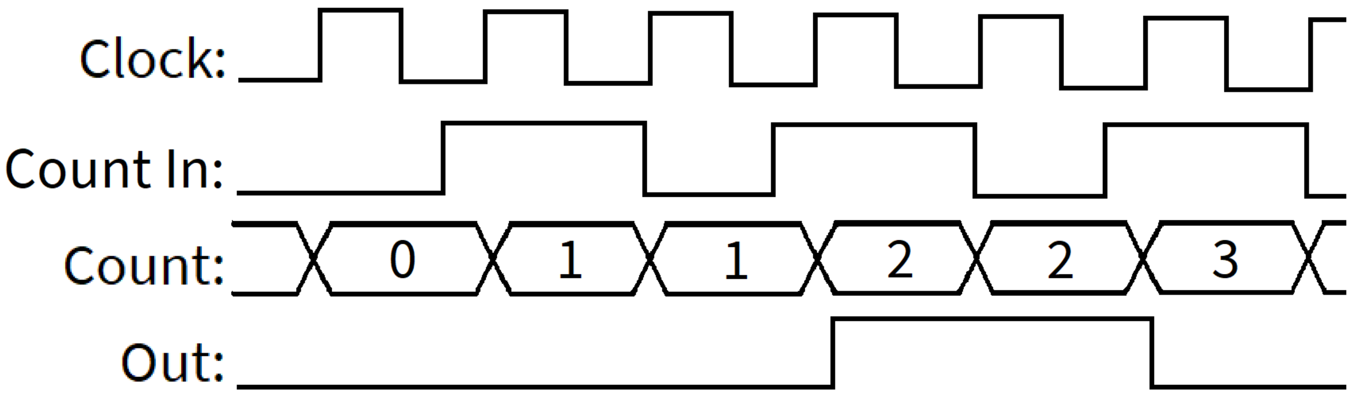
Counters can be configured to count up from zero, or down from a specified value. When they start and stop counting can be controlled either from software or via trigger signals.

Counters can be configured in one of two modes to generate useful output signals:

Compare Mode

This mode "compares" the counter's current count value against a set value. In this mode you specify a "compare value", then once per clock cycle this compare value is compared against the current count value. If the two values are the same the counter can generate a trigger signal or an interrupt.

Consider the following example in which a counter has been configured to count rising edges in its input signal with a compare value of 2:



Capture Mode

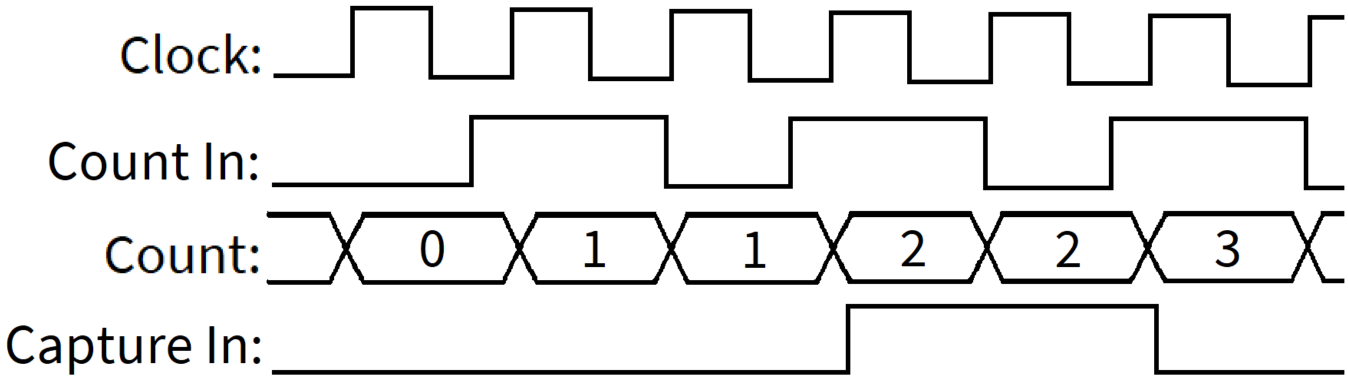
This mode "captures" the counter's current count value whenever a specified event occurs. In this mode you need to specify a capture mode and a capture input signal. The capture mode specifies when the count value will be captured and can be any of the following options:

- Rising edges of the capture input
- Falling edges of the capture input
- Both rising/falling edges of the capture input
- Both rising/falling edges of the clock signal while the capture input is high

Note: *The block is synchronous, so a rising/falling edge on the capture input are detected at the next rising clock edge.*

Whenever a capture is performed the counter can generate a trigger signal or an interrupt. The captured value can then be read by the CPU.

Consider the following example in which a counter has been configured to count rising edges in its input signal. The counter has also been configured to capture its count value whenever a rising edge is detected in the capture input signal:



In this example the counter captures the value 2.

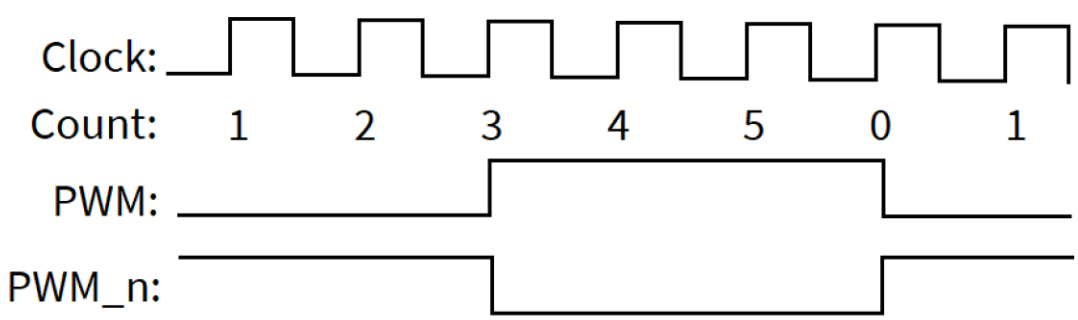
8.1.2.2 Pulse Width Modulator (PWM)

A PWM is simply a counter configured to count up from 0 to a specified period value with one or more outputs based on a compare value. A PWM can run continuously or require a trigger to begin running. Some common use cases for PWMs are:

- Creating arbitrary square wave outputs
- Driving an LED (changing the brightness)
- Driving Motors

In PSoC™ devices each PWM has two outputs, these outputs are simply the compliment of each other and are configurable based on what the period value is with respect to the compare value. A PWM outputs a "1" when the counter's current count value is >= its compare value and a "0" any other time. By changing the compare value relative to the counter's period value, you can change the percentage of time the output of the PWM is high – this is called the duty cycle. For example, a 50% duty would mean that the output is high half the time and low half the time. To accomplish this, you would set the compare value to be ½ of the period value. The speed of a PWM (the rate at which its counter runs) is based off of the PWM's clock signal.

Consider the following example in which a PWM has a period value of 5, a compare value of 3 and is configured to increment its count on the rising edge of its clock signal:



Note: *A period of 5 means the full period is 6 clock cycles because the count starts at 0.*

Note: *"PWM" is the PWM's main output while "PWM_n" is the PWM's complimented output.*

8.1.3 Analog to Digital Converter (ADC)

An ADC is a hardware component that can read a voltage from an analog pin (single ended mode) or pair of analog pins (differential mode) and convert it into a digital value which the PSoC CPU can interpret. In single ended mode, the ADC reads the voltage on a specified analog pin relative to ground or some other reference value, while in differential mode the ADC reads the voltage between two specified analog pins. The number of bits in the result an ADC produces determines the resolution of the ADC. For example, the result a 12-bit ADC can produce will be in the range:

- $0 \rightarrow 2^{12} = 0 \rightarrow 4096$ for single ended measurements
- $-2^{11} \rightarrow 2^{11} = -2048 \rightarrow +2048$ for differential measurements

The input voltage range of a PSoC™ ADC is configurable and varies between devices. Generally, the range of voltages you can measure is 0 → 2*Vref for single ended measurements and Vx ± Vref for differential measurements, where Vref is a reference voltage produced on chip and Vx is the negative input to the differential measurement.

The result produced by an ADC has the unit "counts". Each "count" represents a fraction of the total input voltage range. For example: If the input range is ±5V (10V total) and the ADC has a 12-bit resolution, then each count $= \frac{10\,V}{(2^{12}-1)} \approx 0.00242$ V/count, which is also = 409.5 counts/V.

Example: Given input voltage = 2 V, how many counts does the ADC return?

Answer: counts = 409.5 * 2 = 819 counts

PSoC™ devices have two different kinds of ADCs

- Successive Approximation Register (SAR)
- Sigma-Delta or Delta-Sigma (Both names are often used interchangeably)

The specifics of how each of these ADC designs works is beyond the scope of this course, but if you are interested some good resources to start with are:

- PSoC™ 4 SAR ADC Datasheet
- PSoC™ 6 SAR ADC Datasheet
- Cypress Delta-Sigma ADC Datasheet
- SAR ADC Architecture
- Sigma-Delta ADC Interactive Illustration

8.1.4 Serial communication

Serial communication is often used to send data from one device to another. In serial communication, data is sent through one or more physical wires directly connecting devices, continuously, one bit at a time (hence the name serial). Data can be transmitted one direction at a time (half-Duplex), or in both directions simultaneously (full-Duplex).

There are a very large number of serial communication standards and protocols, but some of the most popular in the world of embedded systems are: UART, I^2C, and SPI. The specifics of how each of these protocols works is outside of the scope of this course, but if you are interested some good resources to start with are:

- UART Description
- I2C Description
- SPI Description

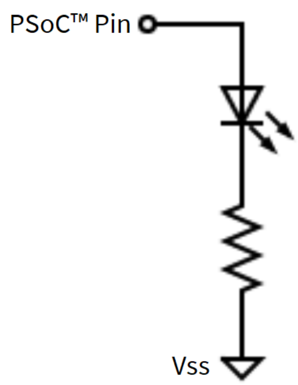
8.2 External Components

8.2.1 Light Emitting Diode (LED)

An LED is a device that glows when you pass electrical current through it. The brightness of the LED depends on the amount of current that passes through it. If you pass too much current through the LED it will blow up (think fire and smoke). In general, LEDs are connected in series with a resistor that limits the amount of current (remember Ohms law? V=IR… look at the schematics below). You can vary the brightness of an LED by either controlling the input voltage (which limits the current) or by "blinking" the LED faster than the human eye can see and varying the duty-cycle (see PWM). An LED is connected to a PSoC™ in one of two ways:

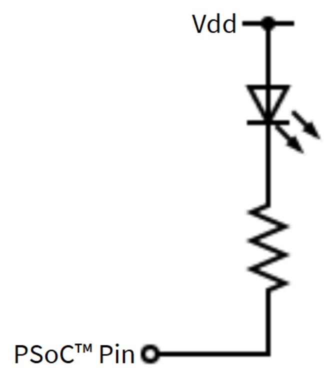
- Active High (driving the LED with a "1" lights it up)
- Active Low (driving the LED with a "0" lights it up)

Active High



- When the PSoC™ drives a "1", current flows out of the PSoC™, through the LED and into Vss (ground).
- When the PSoC™ drives a "0", no current flows.

Active Low



- When the PSoC™ drives a "0" the current flows out of Vdd, through the LED, and into the PSoC™.
- When the PSoC™ drives a "1", no current flows.

8.2.2 Mechanical switch (push-button)

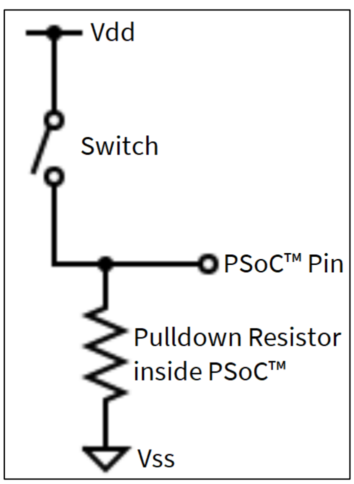
A mechanical switch is an electromechanical device that electrically connects two terminals when it is in the closed position. A very common type of switch is a push-button. In this case, the terminals are connected when the button is pressed. When connecting a button to a PSoC™, the pin is typically configured using a resistor that pulls the PSoC™ input to either Vdd or Vss. This is done so a separate resistor is not required on the board. On Infineon development kits, buttons are typically active low and do not have a separate resistor so the pin should be configured as an input with resistive pullup and the pin's initial state should be set to a 1 so that the pin will be pulled high when the button is not being pressed.

The circuit can be configured as:

- Active High (when the button is pressed the PSoC™ reads "1")
- Active Low (when the button is pressed the PSoC™ reads "0")

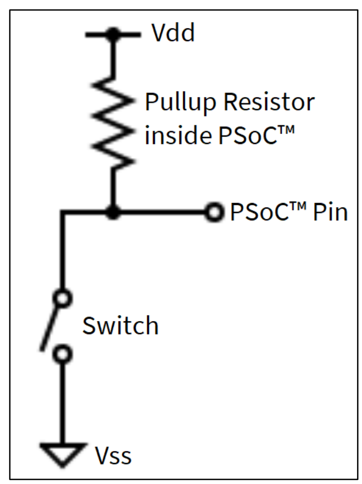
The active low case is much more common.

Active High



- When the switch is open (not pressed) the PSoC™ will see Vss on the input and will read a "0".
- When the switch is closed (pressed) the PSoC™ will see Vdd on the input and will read a "1".

Active Low

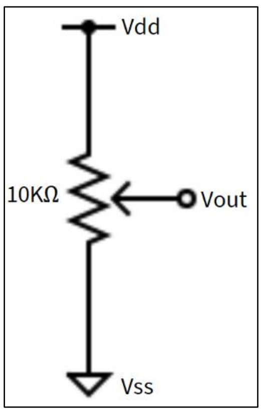


- When the switch is open (not pressed) the PSoC™ will see Vdd on the input and will read a "1".
- When the switch is closed (pressed) the PSoC™ will see Vss on the input and will read a "0".

8.2.3 Potentiometer (a.k.a. pot)

A pot is a 3-terminal electromechanical (meaning that mechanical movements cause electrical actions) analog device. Two of its terminals normally connect to power and ground while the third terminal is used for output. The output terminal produces a voltage that varies between power and ground based on the position of the dial. Mechanically, a pot uses a sliding contact along a resistor to form an adjustable resistor voltage divider. A pot may be thought of as simply an analog voltage reference source.

The arrow in the pot schematic symbol represents the variable contact, controlled by turning the dial. As the contact slides toward the power rail (Vdd), the output voltage (Vout) rises higher. As the contact slides toward ground (Vss), Vout drops.

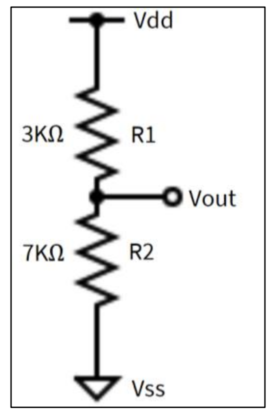


For example, suppose the dial is set so that the sliding contact is closer to the top power rail such that the top resistor is 3000 ohms and the bottom resistor is 7000 ohms.

In that case, the resistance between Vout and Vdd will be smaller than the resistance between Vout and Vss, as shown in the figure to the right. Remember the voltage divider equation?

$$Vout = Vdd\left(\frac{R2}{R1 + R2}\right) = 3.3\left(\frac{7000}{3000 + 7000}\right) = 2.31\ V$$

The output is 7/10ths of the power rail in this example.



Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Published by
Infineon Technologies AG
81726 Munich, Germany