AN76439

PSoC® 3 and PSoC 5LP - Phase-Shift Full-Bridge Modulation and Control

Author: Ross Fosler/Srinivas NVNS
Associated Project: Yes
Associated Part Family: All PSoC 3 and PSoC 5LP parts
Software Version: PSoC Creator™ 2.2 or higher
Related Application Notes: AN76496

If you have a question, or need help with this application note, contact the author at ross@cypress.com OR snvn@cypress.com.

AN76439 introduces phase-shift full-bridge modulation for PSoC 3 and PSoC 5LP. This application note describes in detail the implementation of phase-shift modulation in UDBs with some discussion on how to control the full-bridge for Power applications.

Contents

Introduction 1
Introduction to PSFB Modulation..2
PSFB Modulator Implementation.......................................6
  Master Pair Gate Signal Generation.............................6
  Slave Pair Gate Signal Generation...............................6
  Modulation Techniques 7
  Delay Generation Techniques 8
A Simple PSFB Component 11
Peak Current Control Example..12
Included Project Examples..14
  Quick Instructions to Get Going..................................16
  Project Example 1 – Basic PSFB Modulation.............17
  Project Example 2 – Analog PSFB Modulation 18
  Project Example 3 - PSFB Modulation with Delay......19
  Project Example 4 – Interleaved PSFB Modulation....20
Conclusion 21
Appendix A: Using the PSFB Component in Your Project 22
Document History..24
Worldwide Sales and Design Support.............................25

Introduction

There are a variety of modulation approaches and control techniques that apply to power applications. With the introduction of PSoC 3 and PSoC 5LP and the significant programmable analog and digital functions that come with both product families, there is very little to limit the possibilities, especially advanced modulation and control techniques.

This application note introduces phase-shift full-bridge (PSFB) modulation. This is a modulation commonly found in zero-voltage switching (ZVS) converters, a group within the family of soft-switched converters. This application note focuses on how a PSFB modulator is implemented in PSoC. Both analog and digital design variations are explored. In addition there is some light discussion about control approaches.

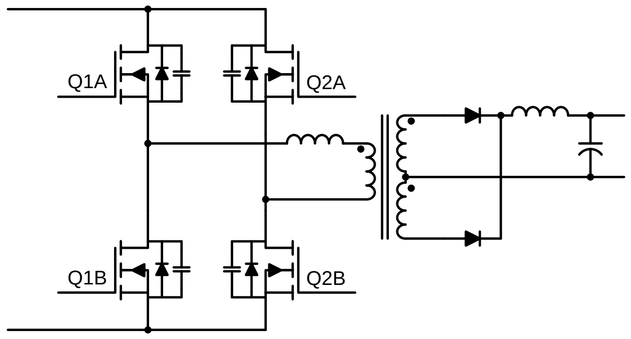
Note that although the PSFB is common in ZVS design, this application note does not discuss in any significant detail the fundamentals of ZVS and soft switching. Also it is not the intent of this application note to introduce the fundamentals of power electronics, an interesting and vast subject in its own right. These topics are left for you to explore.

Also keep in mind that this application note assumes that you are familiar with developing applications using PSoC Creator for PSoC 3 or PSoC 5LP. If you are new to PSoC 3 or PSoC 5LP, introductions can be found in AN54181, Getting Started with PSoC 3 and AN77759, Getting Started with PSoC 5LP. If you are new to PSoC Creator, see the PSoC Creator home page.

Introduction to PSFB Modulation

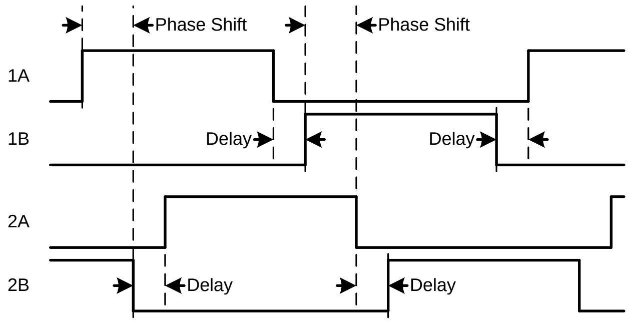
The full-bridge can be thought of as two half-bridges with the load driven between each leg. Figure 1 shows a schematic example of a basic full-bridge converter. The master leg is defined by MOSFETs Q1A and Q1B, and the slave leg is defined by Q2A and Q2B in this representation. Notice that the load in this example is a transformer with the transformer output diode rectified into an LC filter. In most modern high efficiency designs the output is likely to be synchronously rectified rather than diode rectified. Also note that the PSFB is not limited to the schematic shown in Figure 1; other variations exist depending on the application.

Figure 1. PSFB Schematic Example



For phase-shift modulation, each leg of the full-bridge (half-bridge) is modulated with a complementary square wave, as Figure 2 shows. The transfer of energy is controlled by modulating the phase relationship between the complementary signal pairs driving the master and slave legs of the converter. The amount of overlap determines transfer energy, as you would expect.

Figure 2. Phase-Shift Modulated Signals



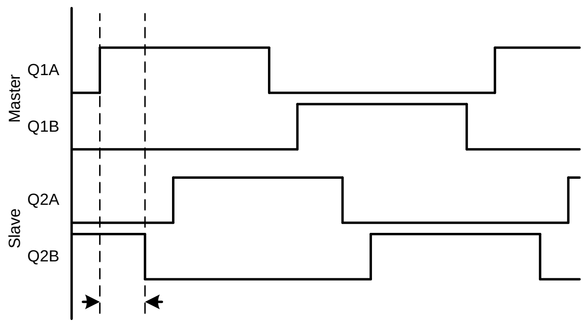
A dead time is usually inserted between each of the complementary signals to avoid shoot-through (also shown in Figure 2). In soft-switched designs, where the PSFB is commonly employed, the delay is set to achieve zero-voltage switching. ZVS is a design approach that significantly reduces the switching losses by switching when the voltage across the switched MOSFET is at or near zero.

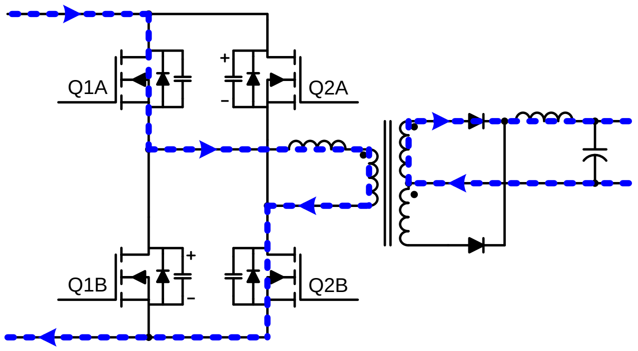
For the sake of introducing typical phase-shift full-bridge modulation, I only touch upon zero-voltage switching concepts. However, just a reminder, it is not my intent to discuss ZVS and the detailed theory behind it. Such discussion of ZVS is beyond the scope of this application note and is left for future exploration. There are numerous resources available that explore this topic (again another very interesting subject).

Let us now examine in detail each of the phases, or stages, in the modulation cycle.

Stage 1 – Forward Conduction

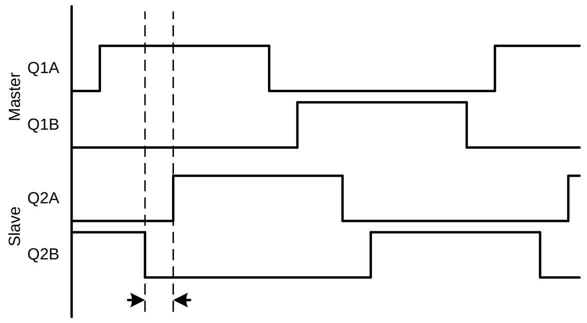
A good starting point is to look at the period where the master FET Q1A and slave FET Q2B are both conducting. During this period, current flows through the transformer and any series and/or leakage inductance. You might notice that the opposing MOSFETs that are not conducting maintain full voltage across their output capacitance (C_{oss}).

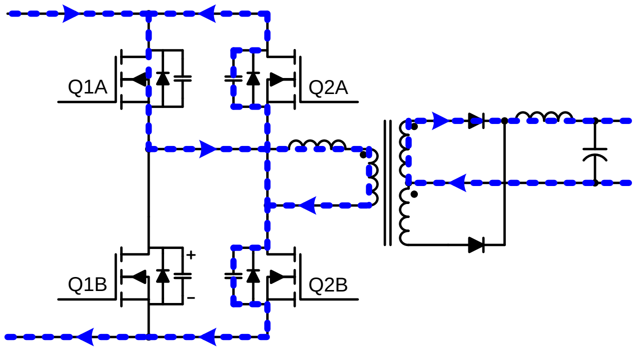




Stage 2 – Slave-Leg Transition

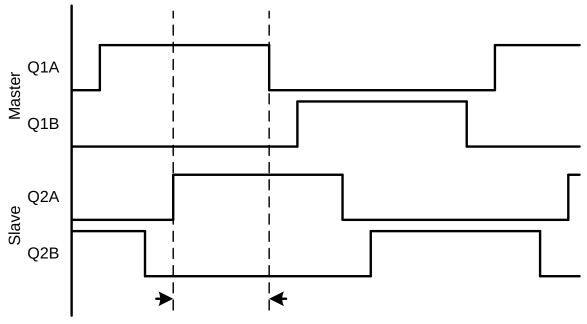
After the forward conduction period, Q2B is switched off. The PSFB enters a period where the energy stored in the series inductance (including leakage inductance) and the previously charged MOSFET, Q2A, is released into the system. Notice that the series inductance and the MOSFET's C_{oss} of the slave leg form a resonant tank. Q2B C_{oss} is charged and Q2A C_{oss} is discharged. In ZVS designs the circuit and delay are tuned so that Q2A is switched on in the next transition somewhere near the point where the voltage across Q2A is zero.

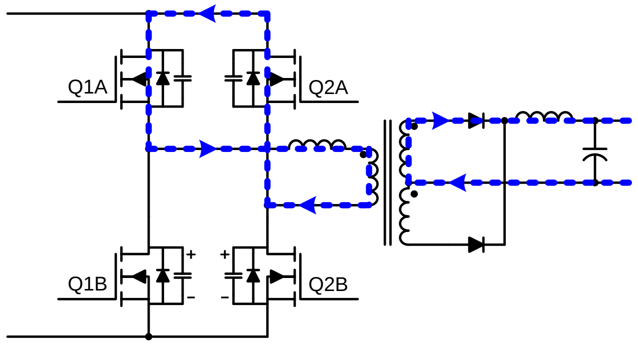




Stage 3 – Freewheeling

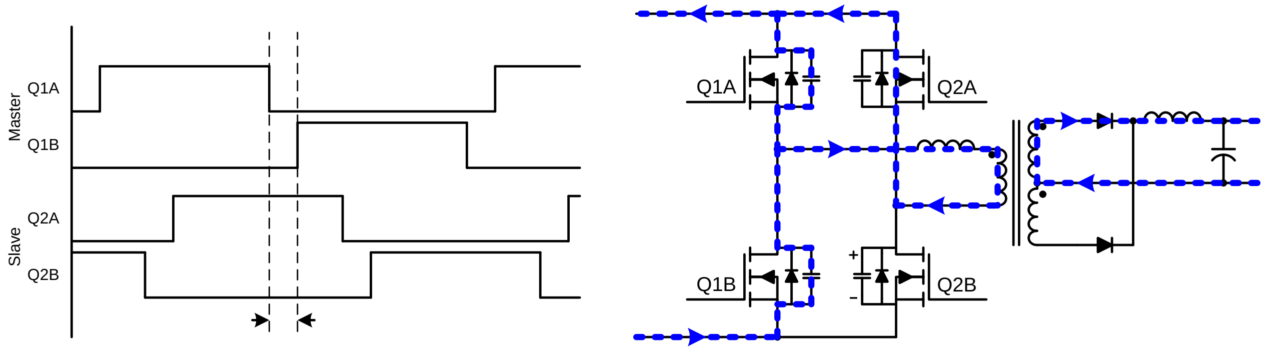
During the freewheeling period MOSFETs Q1A and Q2A are conducting; they maintain any current flow that may exist in the series inductance. Again notice that now C_{oss} of Q1B and Q2B are charged to the full voltage of the input.





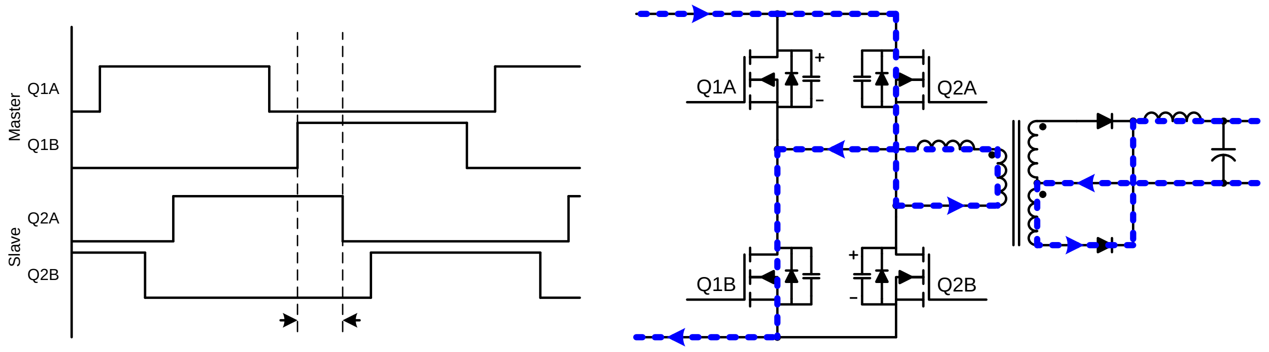
Stage 4 – Master-Leg Transition

The next phase I refer to as the master leg transition. Notice again that the series inductance and C_{oss} of the master leg's MOSFETs now form a resonant tank. Q1A's C_{oss} is charged while Q1B's C_{oss} is discharged.



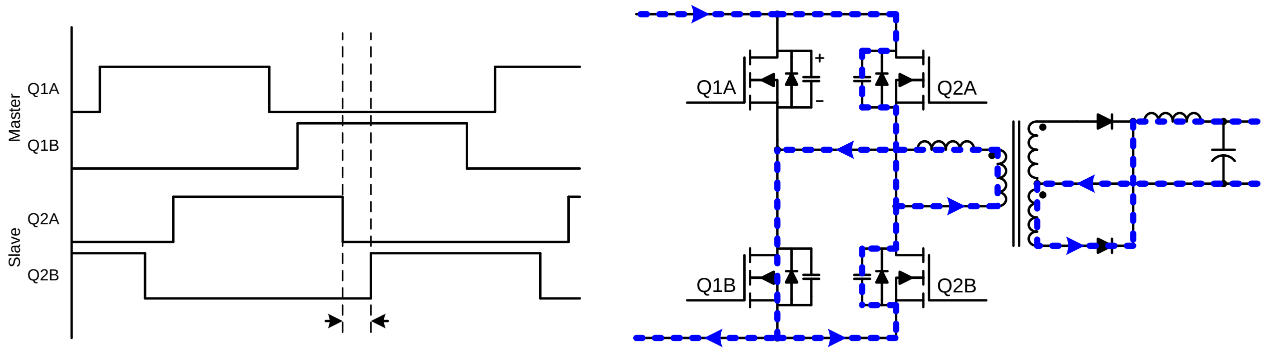
Stage 5 – Forward Conduction

Similar to Stage 1, the master FET Q1B and slave FET Q2A are both conducting. Again during this period current is flowing through the transformer and any series and/or leakage inductance. Notice that the opposing MOSFETs that are not conducting maintain full voltage across their output capacitance.



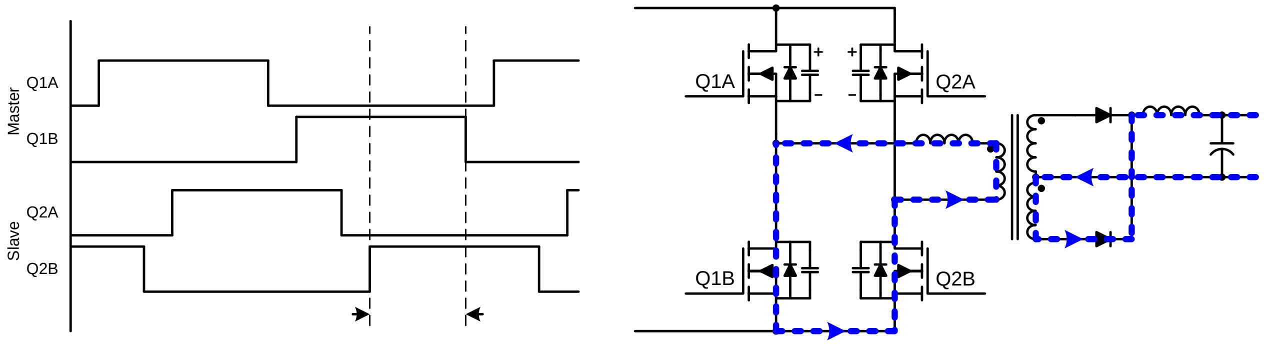
Stage 6 – Slave-Leg Transition

Q2A is switched off. The PSFB enters a period where the energy stored in the series inductance and the previously charged MOSFET, Q2B, is released into the system. As in previous stages, the series inductance and MOSFETs C_{oss} of the slave leg form a resonant tank. Q2A is charged and Q2B is discharged. In ZVS designs the circuit and delay are tuned so that Q2A is switched on in the next transition somewhere near the point where the voltage across Q2B is zero.



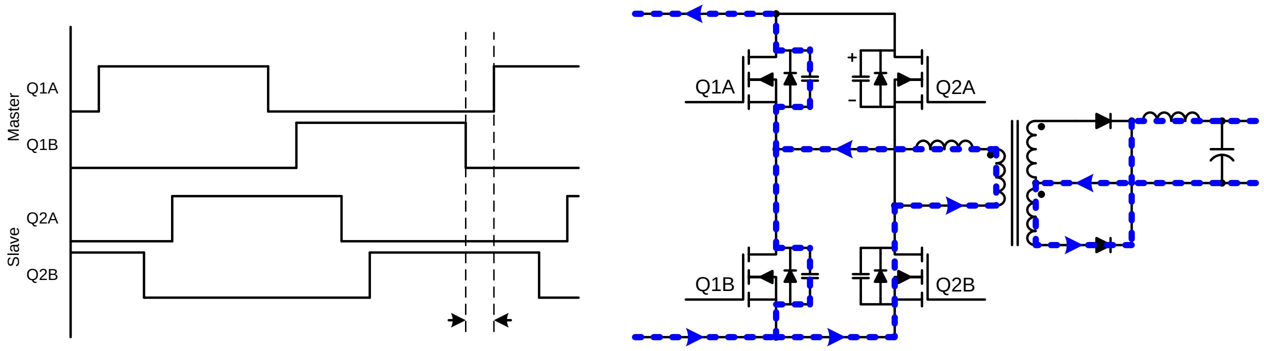
Stage 7 – Freewheeling

During the second freewheeling period MOSFETs Q1B and Q2B are conducting; they maintain any current flow that may exist in the series inductance. Again notice that now C_{oss} of Q1A and Q2A are charged to the full voltage of the input.



Stage 8 – Master-Leg Transition

The final phase is again a master leg transition. The series inductance and C_{oss} of the master leg's MOSFETs form a resonant tank. Q1A's C_{oss} is charged while Q1B's C_{oss} is discharged. After this stage the modulation cycle starts again with stage 1.



PSFB Modulator Implementation

The PSFB modulator has several major functional elements or pieces:

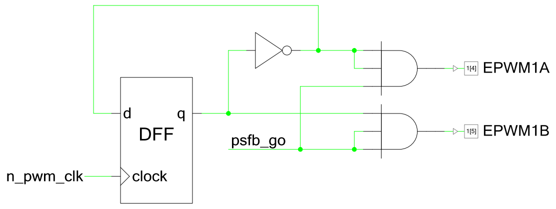
- Control logic to support the complementary master (fixed) gate drive logic.
- Complementary slave (phase shifted) gate drive logic.
- Event generation translated from pulse-width modulation, to determine the phase shift.
- And last, as with most synchronous modulators, there is delay control between edges.

The following sections describe these functional elements with variations to the design approach. This is the part of the power of PSoC - its vast flexibility which makes it easy to quickly build and test different solutions.

Master Pair Gate Signal Generation

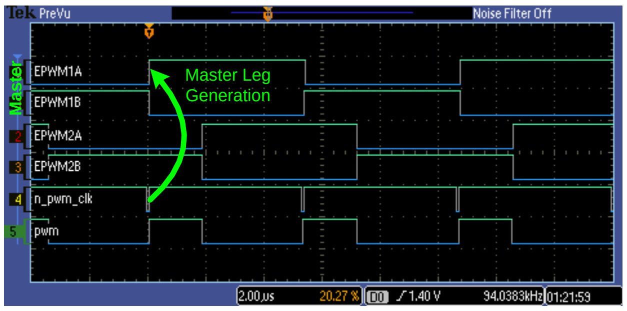
The gate signal generation for the complementary master pair is straightforward - clock the inverted output of a 'D' flip-flop back to its input as Figure 3 shows. This causes the outputs to toggle and form a square wave at half the clock frequency. The complementary pair is derived from the inverted and non-inverted outputs of the flip-flop.

Figure 3. Master Gate Signal Generation



The AND gates in Figure 3 provide an option to turn off the gate drive logic effectively allowing 'output disable' capability. You also might notice that three-terminal AND gates are used with two of the inputs shorted together. Although not shown here, the extra inputs are present for adding delay generation circuitry, which will be discussed later. Figure 4 shows the complementary nature of the signal and the link between the master leg synchronizing events.

Figure 4. Master Gate Signal Generation



Slave Pair Gate Signal Generation

The complementary slave signal pair is derived, in part, from the master signal pair. This is because the state of the slave leg signals depend on the phase of the master leg signals. A trigger signal, generated by the PWM block, is used to set the edge, and the master phase logic output sets the slave's direction. A simple SR latch holds the slave output state until the next trigger arrives. The latching event is a falling edge transition on the PWM block output or the clock to the master.

It is necessary for the latching event to come from both the master source and the slave source. This is provided to support the situation when the PWM is at zero duty cycle. If it were not for the master synchronizing event the slave would never update at the zero duty cycle singularity.

Just like the master leg design, there is a pair of AND gates, as Figure 5 shows. These gates provide an option to turn off the gate drive logic effectively allowing 'output disable' capability. And just like the master signals, three-terminal AND gates are used with two of the inputs shorted together to add delay. Again, this will be discussed later.

Figure 5. Slave Gate Signal Generation

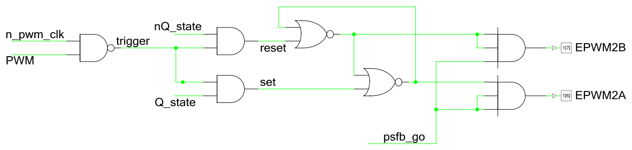
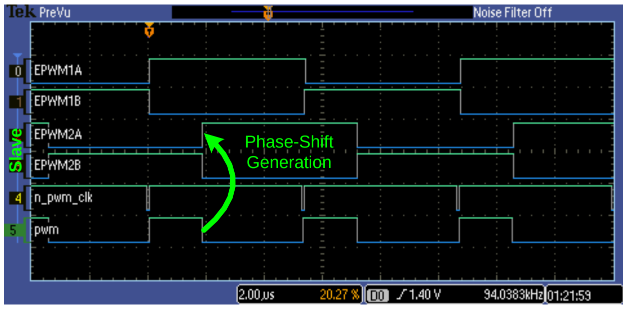


Figure 6 shows an example of what the output looks like when it is operating near 100 kHz. The master pair gate signals (the inverting and non-inverting outputs of the 'D' flip-flop) are reset and set respectively. On the falling edge of the PWM input signal, the SR latch of the slave logic output is set to the non-inverting output of the master pair that is currently logic high. This sets the slave output high driving the low side MOSFET gate signal low and the high side MOSFET gate signal high. This state remains until the next low transition of the PWM signal occurs.

Figure 6. Slave Gate Signal Generation



Note The SR latch in Figure 5 is intentionally included. PSoC Creator permits this; however, an SR latch is a combinational loop, and PSoC Creator generates a warning on combinational loops in logic. This is because PSoC Creator can only perform timing analysis on sequential logic paths. Thus it warns that it must break the loop to analyze.

Modulation Techniques

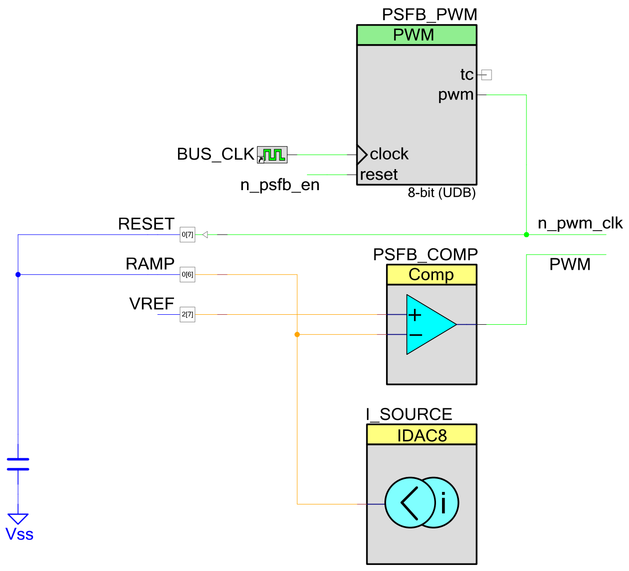
Since the modulator requires only falling edges to indicate phase transitions, these edges can be generated by an analog programmed source as well as any digital source. PSoC's flexibility allows you to easily implement either analog or digital. Let us examine each method:

Analog Modulation Source

Figure 7 shows an analog PWM generated by an external capacitor and a current source or IDAC combined with a comparator. A simple time-based function (a digital PWM in this example) is used to reset the integration in the capacitor and trigger the PSFB slave pair. .

Analog modulation has an advantage over any digital approach in that the modulation input is a continuous function; there is no definable quantization unlike a digital implementation (i.e. infinite resolution for the 'digital' thinkers like myself). However, the down-side might be that the duty cycle programming is set externally rather than a register or some easily programmable function. Another DAC could be used to program the modulation, but the effect would be similar to just implementing a digital PWM (i.e. quantization in the DAC).

Figure 7. Analog Modulation Design



Looking at Figure 7, the IDAC in this configuration is used only as a programmable current source; the current is fixed to a desired value. The current source (IDAC) into the external capacitor generates a constant voltage ramp. The digital PWM is used for nothing more than a programmable reset pulse. Thus the reset pulse width may be set based on the pin current sink capability and the capacitor tied to the pin. The RESET pin is configured as open-drain to allow the ramp to occur, which is quite apparent in Figure 8 on page 8.

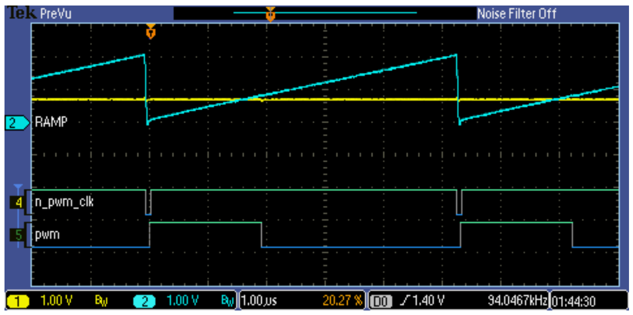
With the current programmed by a constant current source, the relation between voltage, time, and current resolves into a simple linear function:

Equation 1 $$\Delta v = i\frac{\Delta t}{C}$$

Figure 8 shows an example of the analog pulse-width modulation in operation. The ramp capacitor is approximately 470 pF with the current source set at 200 µA. The period of each half phase is around 5.3 µs yielding a ramp up to about ~2.3 V according to Equation 1.

Figure 8 also shows the analog PWM in action. The *n_pwm_clk* signal drives the master leg phase generation. The *pwm* signal drives the slave leg phase generation.

Figure 8. Analog Modulation



Digital Modulation Source

Digital modulation generation for the PSFB is nothing more than an ordinary digital counter and comparator derived pulse-width modulator (or delay-line derived if you feel challenged enough to do it). The PSoC Creator PWM Component shown in Figure 9 is sufficient for this. The clock strobe, n_pwm_clk, is derived from the terminal count strobe of the PWM's internal timer. The PWM is direct. Figure 10 shows the PWM Component configuration dialog use to set up modulation.

Figure 9. Digital Modulation Design

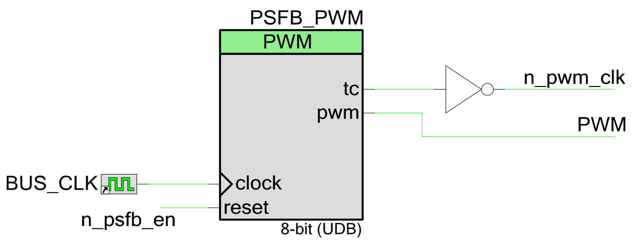
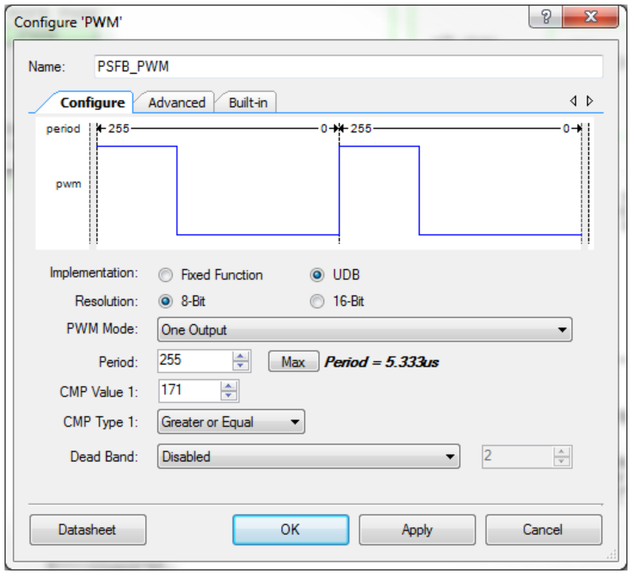


Figure 10. PWM Configuration for Digital Modulation



Name: PSFB_PWM

Implementation: Fixed Function [✓ UDB]

Resolution: [✓ 8-Bit] 16-Bit

PWM Mode: One Output

Period: 255 *Period = 5.333us*

CMP Value 1: 171

CMP Type 1: Greater or Equal

Dead Band: Disabled 2

Digital event generation is extremely simple to implement - the modulation is done entirely with register writes (no external signals). Thus, full digital control of the PSFB and the system is possible. The negative effect is quantization, which can contribute to limit-cycling. For some applications quantization may impact converter performance to the point that the method becomes unfeasible. Again, the feasibility depends enormously on the application. Note that resolution is appreciably worse as the target switching frequency goes up relative to the clock frequency.

Delay Generation Techniques

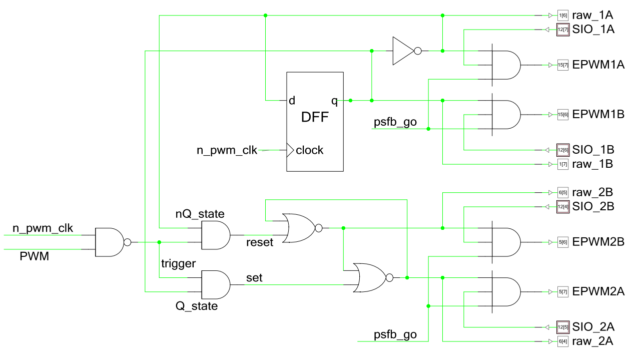
The modulation techniques described previously enable the PSFB implementations described in Introduction to PSFB Modulation on page 2. However, they cannot be directly applied to ZVS topologies because transition delays can result in hard switching and shoot-through in the half-bridges. Therefore, delays must be introduced between transitions to allow the voltage in the full-bridge to resonate down to a minimum or zero voltage before turning the FETs on.

Similar to modulation, there are analog and digital techniques for generating delays in switching, and both can be easily implemented in PSoC. Let us examine each:

Analog Programmed Delay Generation

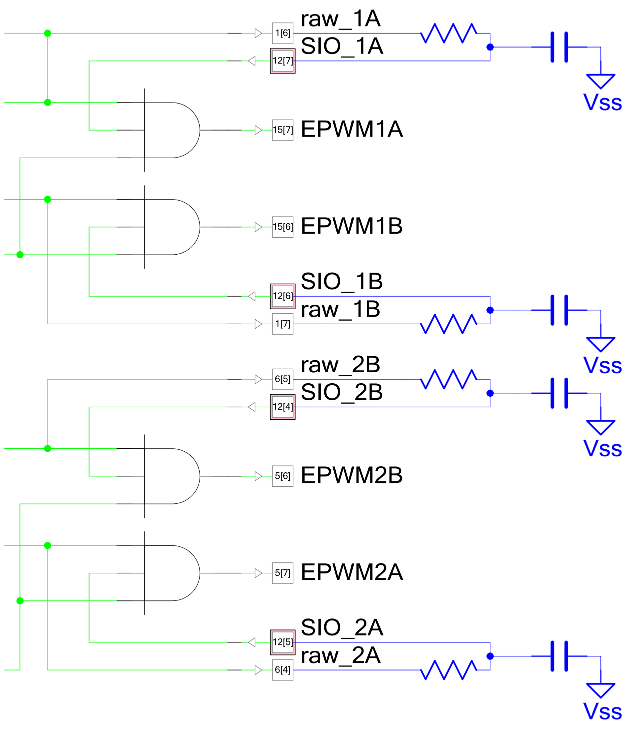
The basic idea in the analog programmed delay approach is to use the comparators in the special I/O (SIO) pins in PSoC 3 or PSoC 5LP to program delay at each transition. The SIO pins exist on PORT12. Figure 11 on page 9 shows the additional gating connection through the SIO pins.

Figure 11. Modulation with Delay Signals



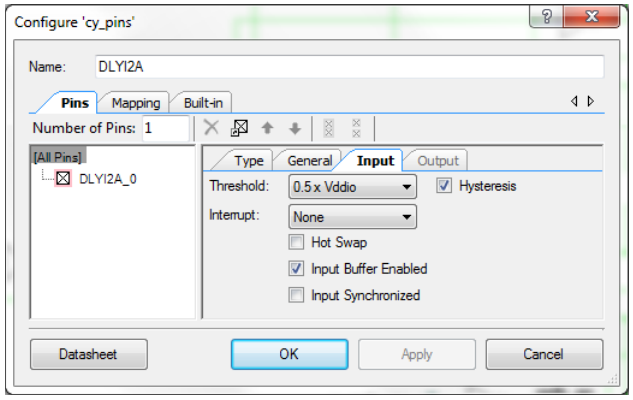
The delay in this example is set by the threshold programmed in the SIO combined with the external RC network. Figure 12 shows a zoom-in of Figure 11 around the slave leg signal generation. Additional resistors and capacitors are annotated to show the hardware programmed delay. Clearly this is an analog way of programming the delay.

Figure 12. Modulation with Delay



Note that the SIO input thresholds are programmable in the PSoC Creator Pins Component configuration dialog (Figure 13).

Figure 13. SIO Threshold Setting


Configure 'cy_pins'

Name: DLYI2A

Pins | Mapping | Built-in

Number of Pins: 1

[All Pins]
DLYI2A_0

Type | General | Input | Output

Threshold: 0.5 x Vddio
[✓] Hysteresis
Interrupt: None
[ ] Hot Swap
[✓] Input Buffer Enabled
[ ] Input Synchronized

Datasheet | OK | Apply | Cancel

Using the SIO threshold, ordinary circuit analysis of the RC network shows the delay to be:

Equation 2 $$V_{thresh} = V_{ref}\left(1 - e^{t/RC}\right)$$

For example, if the SIO comparator reference is set to 50% of V_{DDIO} then Equation 2 reduces significantly:

Equation 3 $$t = RCln(2)$$

Hence the timing of each leading transition of each gate signal can be programmed individually with a simple RC network. Individual programmability is important for ZVS applications since symmetric delay is not always desirable.

Another critical point is that the propagation delay through gate driver and buffer may not always be matched between channels. Thus the difference in propagation delay could also be accounted for by programming the delay.

Digital Programmed Delay Generation

True to PSoC character, digital delay programming is also possible. In this case the delay is programmed with synchronous logic. Note that because the logic is synchronous to the clock, the output is also synchronous; therefore, there is also quantization in the output. For some systems the finite nature of a synchronized system can yield system results that are less than ideal (certainly less ideal than infinite resolution). Figure 14 shows a clip of the logic with the delay generation in Figure 15.

Figure 14. Modulation with Delay Signals

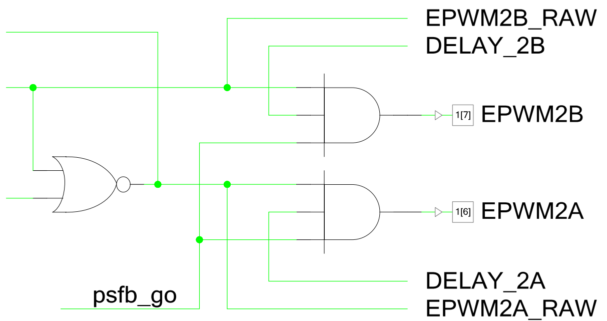
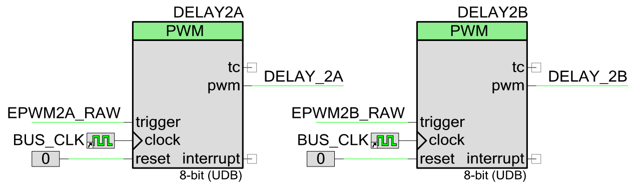


Figure 15. Digital Delay Generation

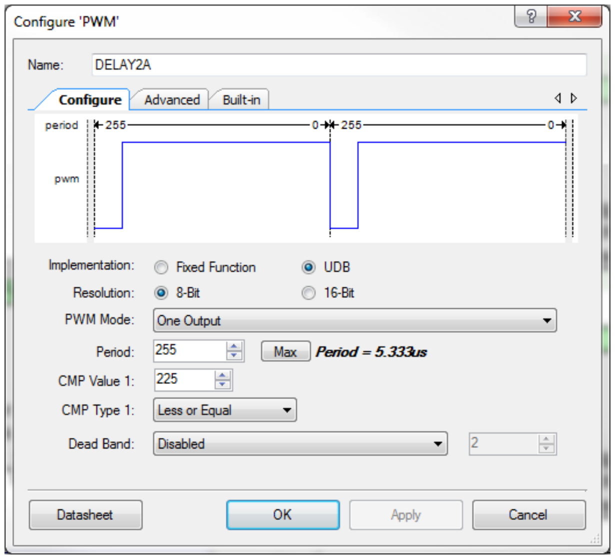


In this example the delay is programmed using one-shot triggered PWM Components. Figure 16 and Figure 17 show the PWM configurations. At the rising edge of the trigger signal the PWM module runs through its count until it hits the terminal count. Since there are four independent channels this method requires four PWM Components to generate delay for all signals that drive the PSFB (only two of them are shown here).

Note that Figure 15 does not necessarily show the only way to inject a delay. There are many other ways to solve this problem digitally. What is shown is a convenient and highly programmable example. Simply setting the compare value in a PWM component sets the delay for the corresponding channel. There are certainly more ways to solve this problem that would likely confine the range of delay programming yet improve the resource utilization.

Figure 18 shows a running example of delay on the PSFB. In this case the delay is clearly programmed to be symmetrical for all edges.

Figure 16. Digital Delay Settings



Name: DELAY2A

Implementation: Fixed Function [✓ UDB]

Resolution: [✓ 8-Bit] 16-Bit

PWM Mode: One Output

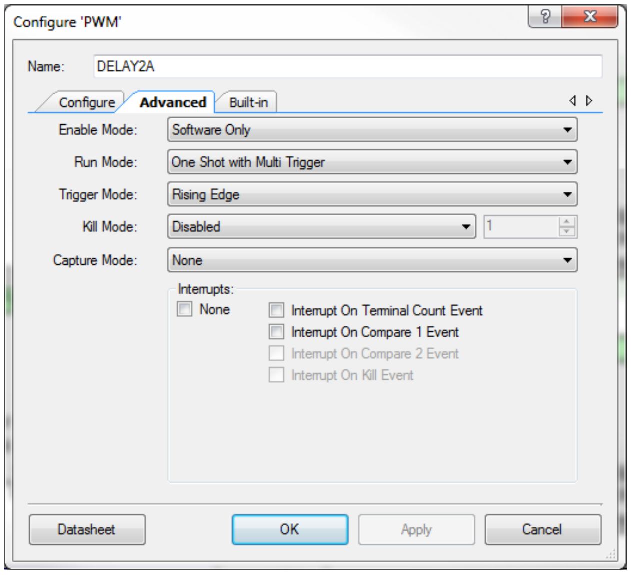
Period: 255 *Period = 5.333us*

CMP Value 1: 225

CMP Type 1: Less or Equal

Dead Band: Disabled 2

Figure 17. Digital Delay Settings (Continued)



Name: DELAY2A

Enable Mode: Software Only

Run Mode: One Shot with Multi Trigger

Trigger Mode: Rising Edge

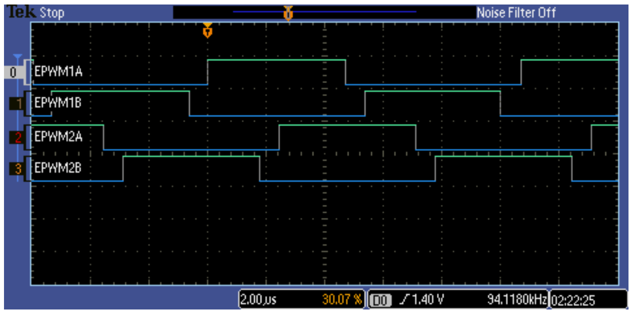
Kill Mode: Disabled 1

Capture Mode: None

Interrupts:

- ☐ None
- ☐ Interrupt On Terminal Count Event
- ☐ Interrupt On Compare 1 Event
- ☐ Interrupt On Compare 2 Event
- ☐ Interrupt On Kill Event

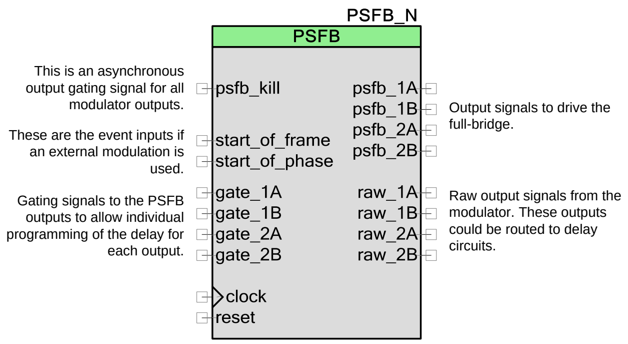
Figure 18. PSFB Example Showing Delay



A Simple PSFB Component

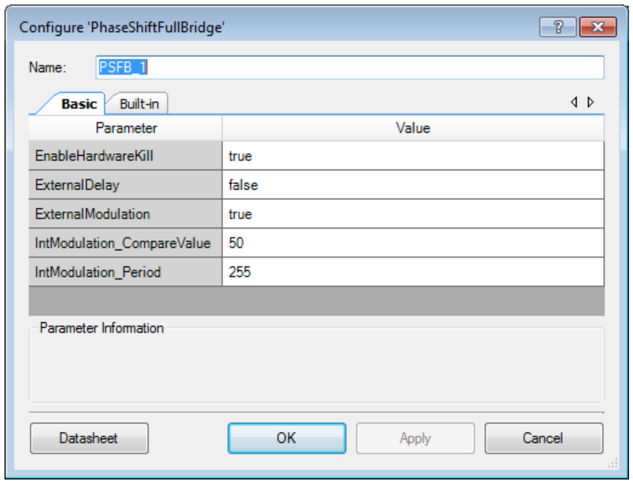
The PSFB structure and design described in PSFB Modulator Implementation on page 6 is encapsulated in a component for the projects in this application note. Figure 19 shows a view of the component with all possible inputs and outputs exposed.

Figure 19. PSFB Component


PSFB_N
PSFB
This is an asynchronous output gating signal for all modulator outputs.
psfb_kill
These are the event inputs if an external modulation is used.
start_of_frame
start_of_phase
Gating signals to the PSFB outputs to allow individual programming of the delay for each output.
gate_1A
gate_1B
gate_2A
gate_2B
clock
reset
psfb_1A
psfb_1B
psfb_2A
psfb_2B
Output signals to drive the full-bridge.
raw_1A
raw_1B
raw_2A
raw_2B
Raw output signals from the modulator. These outputs could be routed to delay circuits.

The component includes simple digital modulation for the sake of quick testing and demonstration. However, the modulation input can be exposed to allow for external operation. This allows you to use a custom analog or other digital technique for the component to translate into phase-shift modulated gate signals. A component datasheet describing the component and its parameters (Figure 20) is included with the projects. Refer to the component datasheet for additional details about the component.

Figure 20. PSFB Component Parameters



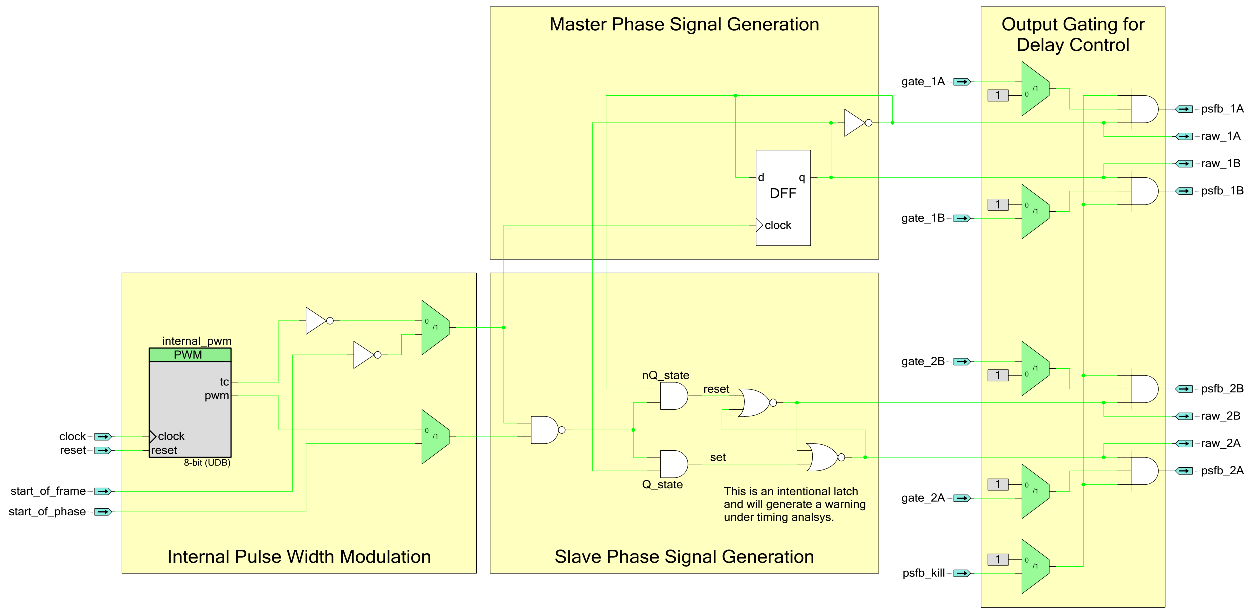
Name: PSFB_1

Parameter	Value
EnableHardwareKill	true
ExternalDelay	false
ExternalModulation	true
IntModulation_CompareValue	50
IntModulation_Period	255

Figure 21 shows an "under the hood" look at the major elements of the component:

- Master Phase Signal Generation
- Slave Phase Signal Generation
- Internal Pulse Width Modulation – A PWM is used to generate modulation. This option can be bypassed if desired by using the start_of_frame and start_of_phase inputs.
- Output Gating – Gating is for enable / disable, and signal delay programming for each rising edge.

Figure 21. The PSFB Component (Under the Hood)



Peak Current Control Example

The PSFB is mostly just a modulation technique. In most power applications the modulator is not operated without some compensation network or other control driving it. In this section I briefly present a peak current control idea that is married to the PSFB modulator. You can use this as an example to explore the vast possibilities (such as analog voltage control, digital voltage control, or even combined peak current control and digital voltage control). With PSoC the possibilities are virtually endless.

Note that this is an idea and not fully explored in hardware at the release of this application note. Exploring the finer details of peak current control is well beyond the intent of this application note (saving this for another time). This is presented as an idea to help put into perspective how complex control ideas can be built with PSoC. Figure 22 shows a current-control design in PSoC Creator.

In most real-world applications the PSFB is often employed with some form of current control strategy. Why is this?

First, pure current-control of the PSFB results in a system with a single dominant pole in its transfer function, whereas its voltage-control counterpart yields a resonant two-pole system. Thus, current-controlled systems that are derived from the 'Buck Converter' topology (such as the PSFB in Figure 1) are easier to compensate and control.

Another major benefit is flux balancing. PSFB designs (such as the one shown in Figure 1) usually have a transformer and no DC flux blocking capability in the transformer. Thus, without some management of the flux in the transformer, the transformer can 'walk' into saturation. Knowing that current is proportional to magnetic flux, such designs are naturally flux balancing with peak current-control employed.

Slope Compensation

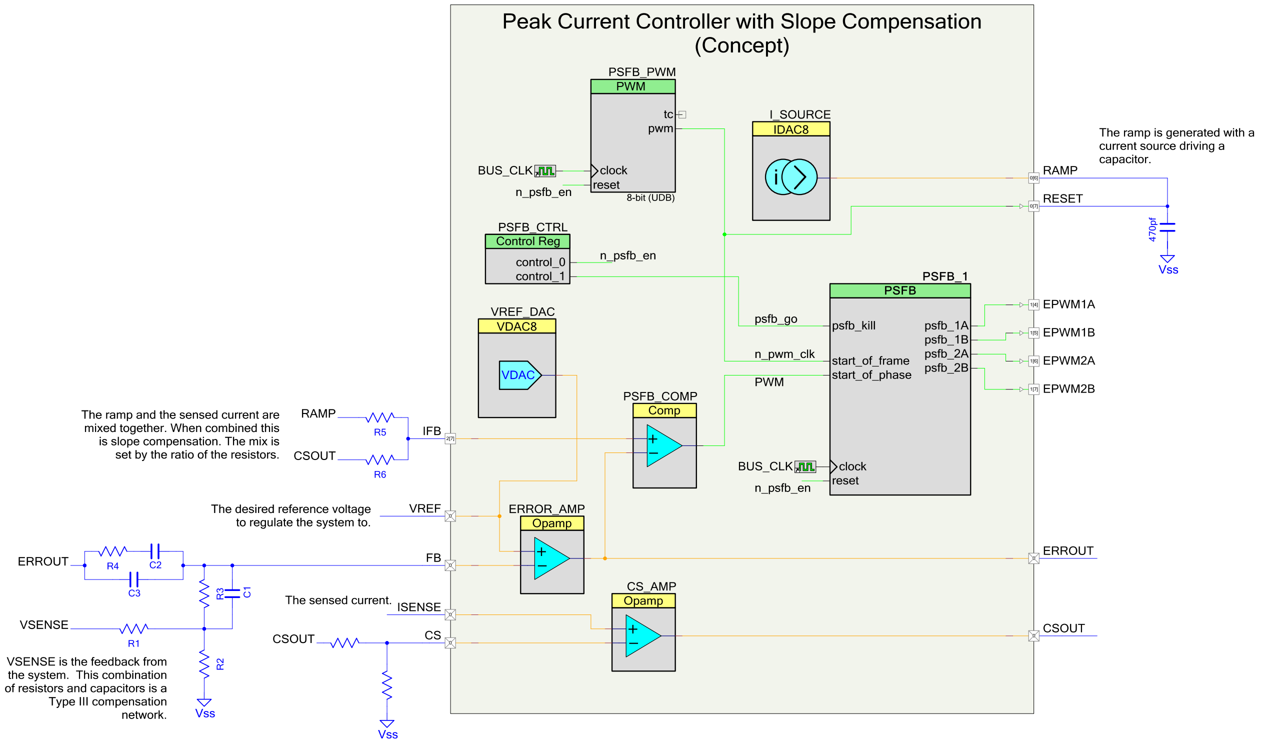
Peak current control has the well known 50% duty cycle limitation. To overcome this, an artificially generated ramp signal is mixed with the current signal to yield part voltage and part current control. The amount of ramp relative to the amount of current signal relates to how far past the 50% duty cycle point the design can be pushed.

The easiest way to think of this is to conceptualize the slope of the artificial ramp growing to infinity. If the ramp were infinite then the control would be purely based on voltage at the output of the converter. If there was no ramp then the control would be purely based on current. Anything in between is a mix of current and voltage control. Equation 4 shows the mixing ratio based on what is shown in Figure 22.

Equation 4 $$V_{IFB} \approx V_{RAMP}\frac{R_6}{R_5+R_6} + V_{CSOUT}\frac{R_5}{R_5+R_6}$$

Note that the PSFB is naturally limited to a 50% duty cycle, however slope compensation is still often employed in practice.

Figure 22. Peak Current Control with Slope Compensation


Peak Current Controller with Slope Compensation (Concept)
PSFB_PWM
PWM
tc
pwm
BUS_CLK
clock
n_psfb_en
reset
8-bit (UDB)
I_SOURCE
IDAC8
The ramp is generated with a current source driving a capacitor.
RAMP
RESET
470pf
Vss
PSFB_CTRL
Control Reg
control_0
control_1
n_psfb_en
VREF_DAC
VDAC8
VDAC
PSFB_1
PSFB
psfb_go
psfb_kill
psfb_1A
psfb_1B
psfb_2A
psfb_2B
n_pwm_clk
start_of_frame
start_of_phase
PWM
EPWM1A
EPWM1B
EPWM2A
EPWM2B
PSFB_COMP
Comp
The ramp and the sensed current are mixed together. When combined this is slope compensation. The mix is set by the ratio of the resistors.
RAMP
R5
IFB
CSOUT
R6
BUS_CLK
clock
reset
n_psfb_en
The desired reference voltage to regulate the system to.
VREF
ERROR_AMP
Opamp
ERROUT
R4
C2
C3
FB
R3
C1
ERROUT
VSENSE
R1
The sensed current.
ISENSE
CS_AMP
Opamp
CSOUT
CS
CSOUT
VSENSE is the feedback from the system. This combination of resistors and capacitors is a Type III compensation network.
R2
Vss
Vss

Ramp Generation

Ramp generation is performed by using a current source to drive into a capacitor. As noted in Analog Modulation Source on 7, the ramp voltage is as Equation 5 shows (Equation 1 repeated):

Equation 5 $$\Delta v = i\frac{\Delta t}{C}$$

When mixing with other signals, the assumption in this case is that the mixing resistors have little effect on the current. Thus Equation 5, although technically not perfect, is still a good predictor of the ramp.

Voltage Loop Compensation Network

With the addition of current control to the PSFB, the output response looks more like a single pole system rather than the two-pole resonant network. Still, although it is not theoretically necessary in the ideal situation, Type III compensation is often employed in practice to account for the loss of current information at very light load conditions. The voltage control portion of Figure 22 shows a Type III compensation network (very similar to a PID). Equation 6 shows the small-signal transfer function of the compensation network.

Equation 6. Voltage Loop Compensator Transfer Function

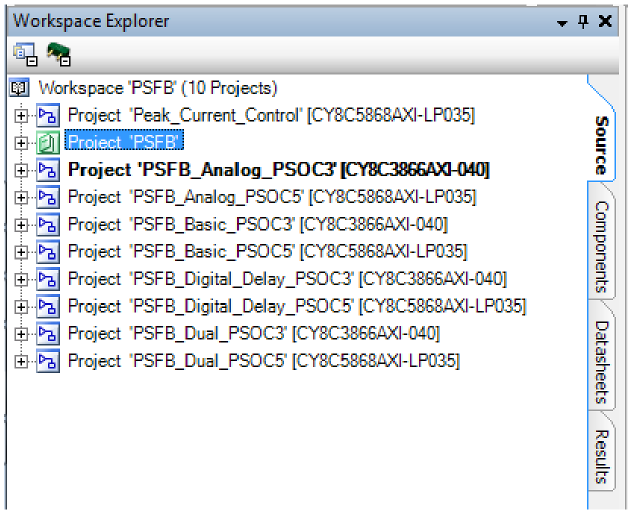
$$\frac{v(t)}{v_s(t)} = \frac{R_2R_3}{(C_3R_3 + C_2R_3)(R_1R_2 + R_2R_3 + R_1R_3)} \frac{(sC_2R_4 + 1)(sC_1R_3 + 1)}{s\left(s\frac{C_2C_3R_3R_4}{C_3R_3 + C_2R_3} + 1\right)\left(s\frac{C_1R_1R_2R_3}{R_1R_2 + R_2R_3 + R_1R_3} + 1\right)}$$

Included Project Examples

There are four projects included with this application note. I could have included several more; however, I chose to confine this document to well less than 100 pages to avoid putting readers into a deep and potentially unrecoverable hibernation.

Note that when you open the workspace you will find ten projects (Figure 23) including the PSFB component library project. The four main projects mentioned are cloned for PSoC 3 to PSoC 5LP totaling eight projects. The project "Peak_Current_Control" is not a complete project (although it does build correctly). It is provided for visual reference only. Should you decide to explore a little, this project is what is described in Figure 22.

Figure 23. Included Projects



Workspace 'PSFB' (10 Projects)
Project 'Peak_Current_Control' [CY8C5868AXI-LP035]
Project 'PSFB'
Project 'PSFB_Analog_PSOC3' [CY8C3866AXI-040]
Project 'PSFB_Analog_PSOC5' [CY8C5868AXI-LP035]
Project 'PSFB_Basic_PSOC3' [CY8C3866AXI-040]
Project 'PSFB_Basic_PSOC5' [CY8C5868AXI-LP035]
Project 'PSFB_Digital_Delay_PSOC3' [CY8C3866AXI-040]
Project 'PSFB_Digital_Delay_PSOC5' [CY8C5868AXI-LP035]
Project 'PSFB_Dual_PSOC3' [CY8C3866AXI-040]
Project 'PSFB_Dual_PSOC5' [CY8C5868AXI-LP035]

Each of the four projects demonstrates the PSFB operation. The different projects combine digital and analog resources to perform variations on the design, yet the underlying PSFB operation is still there. The following sections describe these projects and how to get them going. Here is a brief summary of the projects:

- PSFB_Basic – A basic digitally generated phase-shift modulation.
- PSFB_Analog – An analog method for generating phase-shift modulation.
- PSFB_Digital_Delay – A digitally generated phase-shift modulation scheme with digital delay.
- PSFB_Dual – A digitally generated phase-shift modulation. Two modulators are interleaved together.
- Peak_Current_Control – A project showing the peak current control concept. This project is for reference only and does not do anything.

Demonstration Hardware Setup

All of the projects are targeted for the CY8CKIT-001 with either the PSoC 3 or PSoC 5LP Family Processor module. Figure 24 on page 15 shows a picture of the setup. Here is a summary of what needs to be connected to see the demonstrations running.

- Connect VR → P2_7
- Connect P1_2 → LED4
- Connect P1_4 → Scope
- Connect P1_5 → Scope
- Connect P1_6 → Scope
- Connect P1_7 → Scope
- Install a 470 pF capacitor from GND to the proto area, and connect P0_6 and P0_7 to the other end of the capacitor in the proto area.

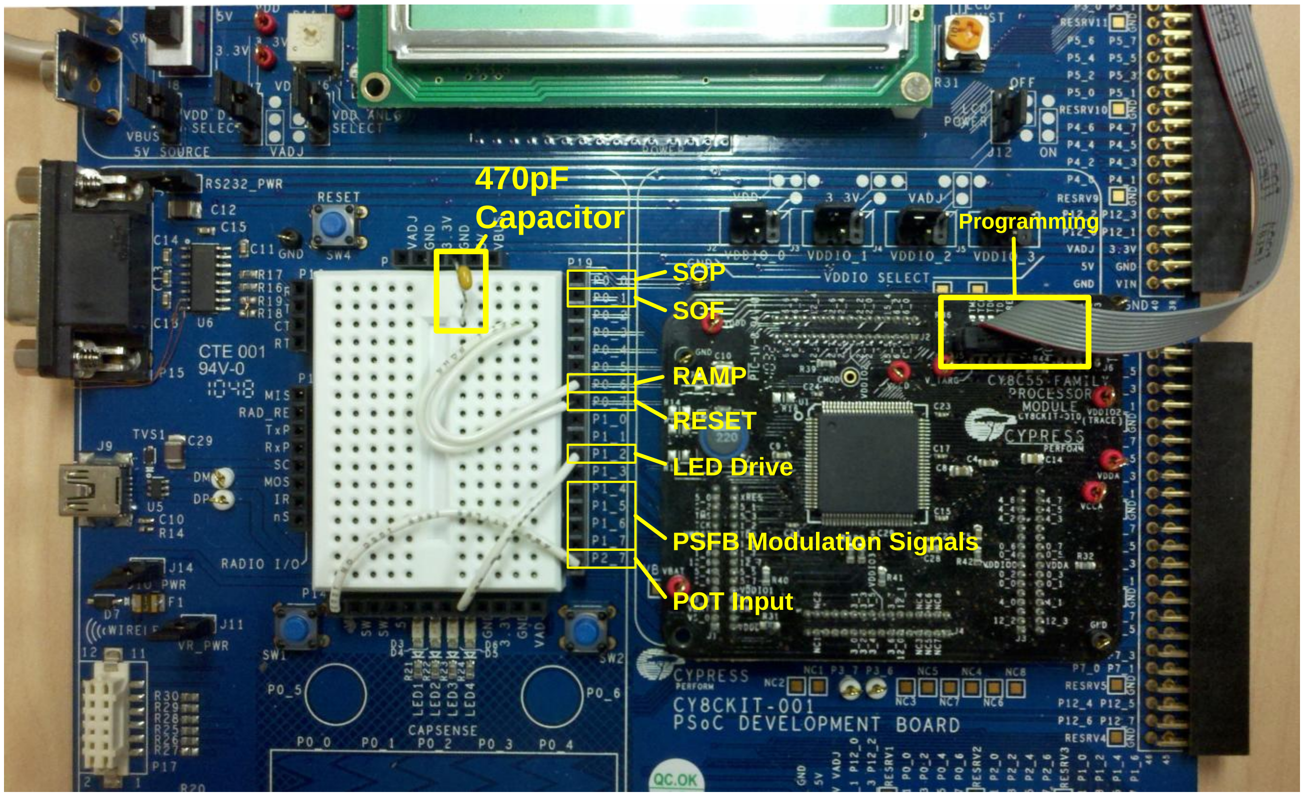
Project 2 has some interesting analog signals. These are optional; however, if you have an appropriate oscilloscope for the task, connect the following:

- Connect P0_6 → Scope (to view the ramp signal)
- Connect P2_7 → Scope (to view the POT voltage)
- Connect P0_0 → Scope (to view the start of phase)
- Connect P0_1 → Scope (to view the start of frame)

Project 4, the dual interleaved PSFB project, has additional signals that are instructive to see. If you have a capable scope that can monitor at least eight signals, then connect the following:

- Connect P2_0 → Scope (to view the second PSFB)
- Connect P2_1 → Scope (to view the second PSFB)
- Connect P2_2 → Scope (to view the second PSFB)
- Connect P2_3 → Scope (to view the second PSFB)

Figure 24. CY8CKIT-001 Hardware Setup



Quick Instructions to Get Going

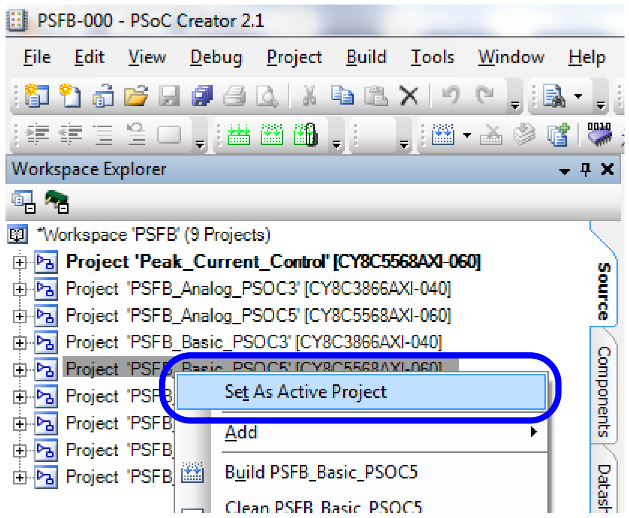
Although this application note assumes that you are familiar with PSoC Creator, this section does provide three very quick steps to get the projects running on your CY8CKIT-001.

If you are new to PSoC 3 or PSoC 5LP, some useful starting information can be found in AN54181, Getting Started with PSoC 3 and AN77759, Getting Started with PSoC 5LP. If you are new to PSoC Creator, see the PSoC Creator home page. Any of these resources can get you started opening, building and programming PSoC Creator projects into your demonstration board.

Step 1 – Set the Active Project

The PSoC Creator workspace can hold many projects, and needs to know what project to work from. To set a project as the active project, right click the project title in the Workspace Explorer window and select ***Set As Active Project***. Figure 25 shows an example.

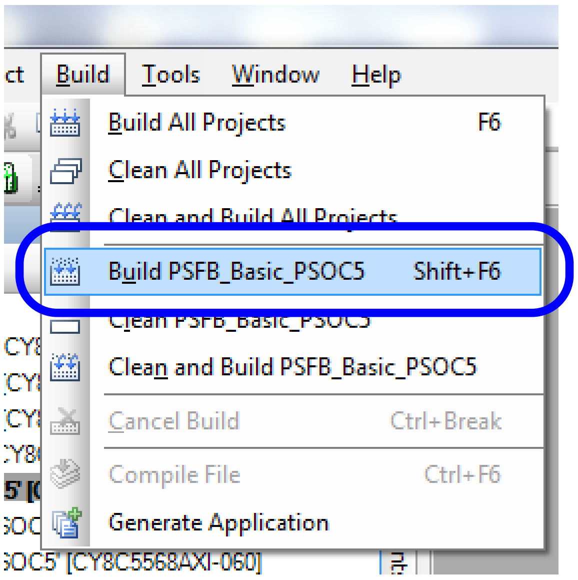
Figure 25. Set the Active Project


PSFB-000 - PSoC Creator 2.1
File Edit View Debug Project Build Tools Window Help
Workspace Explorer
*Workspace 'PSFB' (9 Projects)
Project 'Peak_Current_Control' [CY8C5568AXI-060]
Project 'PSFB_Analog_PSOC3' [CY8C3866AXI-040]
Project 'PSFB_Analog_PSOC5' [CY8C5568AXI-060]
Project 'PSFB_Basic_PSOC3' [CY8C3866AXI-040]
Set As Active Project
Add
Build PSFB_Basic_PSOC5
Clean PSFB_Basic_PSOC5
Source
Components

Step 2 – Build the Project

Select menu item ***Build → <Project_Name>***, or press Shift-F6, to build the active project. <Project_Name> is the name of the project that you want to build. Figure 26 shows an example.

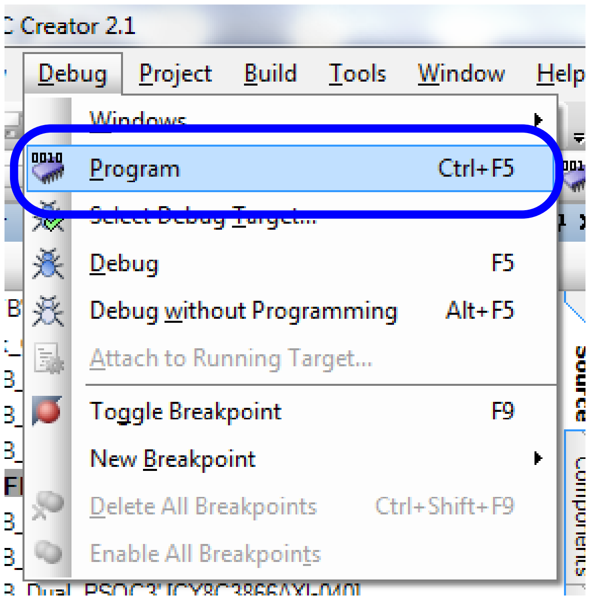
Figure 26. Build the Project



Step 3 – Program PSoC

After the project has been successfully built, you can program the device from PSoC Creator environment. Connect the programming connector of the MiniProg3 to the development board and the USB connector to your PC. Select ***Debug → Program*** (Figure 27), or press Ctrl+F5, to program the device.

Figure 27. Program PSoC


Creator 2.1
Debug Project Build Tools Window Help
Windows
Program Ctrl+F5
Select Debug Target...
Debug F5
Debug without Programming Alt+F5
Attach to Running Target...
Toggle Breakpoint F9
New Breakpoint
Delete All Breakpoints Ctrl+Shift+F9
Enable All Breakpoints

Project Example 1 – Basic PSFB Modulation

This project shows the basic operation of the PSFB modulation using the PSFB Component provided with this application note. In this case the phase modulation is digitally generated internal to the component. Figure 29 shows the design.

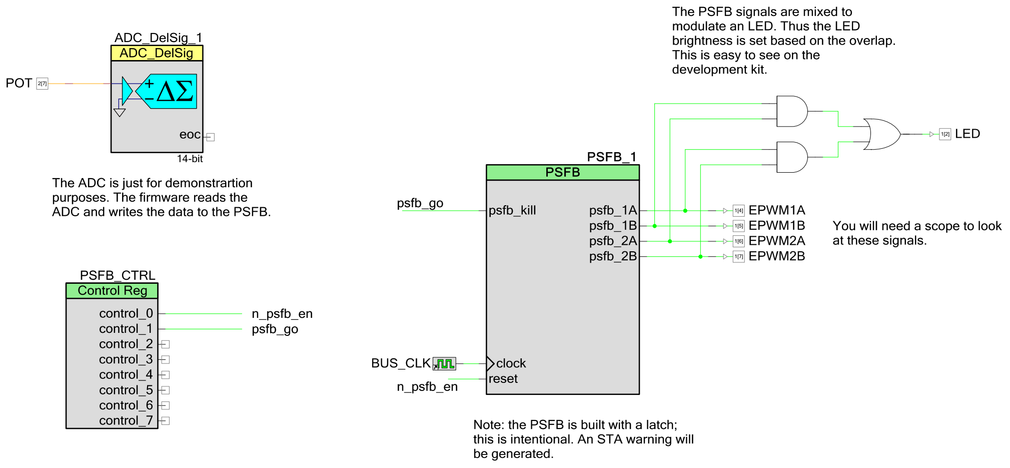
An ADC is instantiated to take an input from a potentiometer and use that information to drive the digital phase modulation. The firmware (Figure 28) is the link between the ADC and PSFB. The firmware conditions the data, limiting its range and truncates the unused bits from the ADC, before writing it to the PSFB.

A small amount of logic is used to mix the phase information back to pulse-width modulation to drive an LED. There is no practical reason to do this other than to demonstrate modulation is working on the demonstration board without the need of an oscilloscope. Thus, when wired, the LED dims and brightens when the modulation is changed (via the POT).

Figure 28. Project 1 Code

```
void main()
{

    /* Start the Phase-Shifted Full-Bridge */
    PSFB_1_Start();
    PSFB_CTRL_Write(0x02);

    /* Start the ADC */
    ADC_DelSig_1_Start();
    ADC_DelSig_1_StartConvert();

    /* Loop forever here */
    for(;;)
    {
        int16 val;

         /* Grab the analog data */
        val = ADC_DelSig_1_GetResult16();

        /* Limit the data to 0 < X < 16383 */
        if (val < 0) {val = 0;}
        else if (val > 16383) {val = 16383;}

        /* Put the upper 8-bit value on the
        phase-shifted full-bridge */
        PSFB_1_WriteCompare(val >> 6);

    }
}
```

Figure 29. PSFB Modulation with No External Modulation Source


ADC_DelSig_1
ADC_DelSig
POT
eoc
14-bit
The ADC is just for demonstrartion purposes. The firmware reads the ADC and writes the data to the PSFB.
PSFB_CTRL
Control Reg
control_0
control_1
control_2
control_3
control_4
control_5
control_6
control_7
n_psfb_en
psfb_go
psfb_go
BUS_CLK
n_psfb_en
PSFB_1
PSFB
psfb_kill
clock
reset
psfb_1A
psfb_1B
psfb_2A
psfb_2B
EPWM1A
EPWM1B
EPWM2A
EPWM2B
You will need a scope to look at these signals.
The PSFB signals are mixed to modulate an LED. Thus the LED brightness is set based on the overlap. This is easy to see on the development kit.
LED
Note: the PSFB is built with a latch; this is intentional. An STA warning will be generated.

Results

Figure 30 shows an example with relatively small shift, and Figure 31 shows an example with relatively large shift. In a typical application such as the design shown in Figure 1 the high phase shift would result in a higher output voltage.

Figure 30. Modulation Results, Less Shift

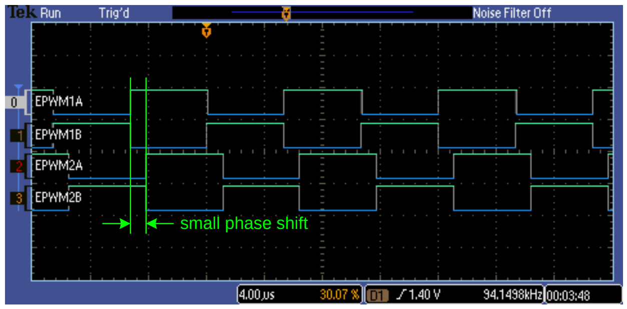
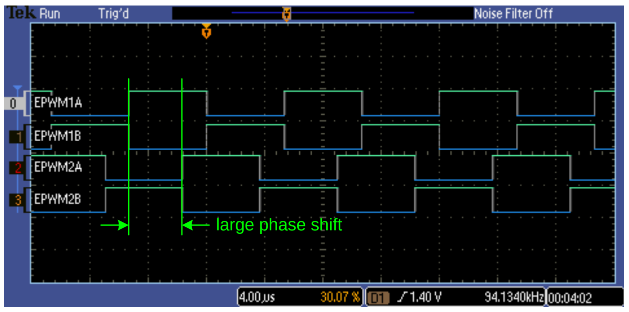


Figure 31. Modulation Results, More Shift

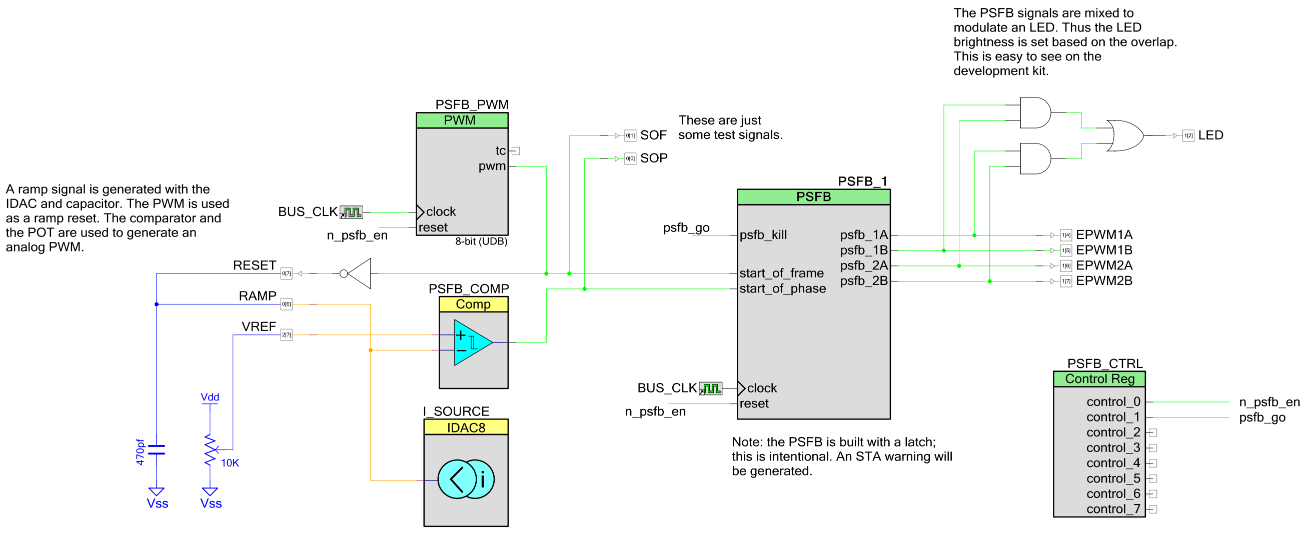


Project Example 2 – Analog PSFB Modulation

This project is almost exactly the same as Project 1; it is altered slightly to show analog modulation. The details of analog pulse-width modulation are described in Analog Modulation Source on page 7; the modulation technique is the same and is shown in Figure 32. In this case the POT is used as in Project 1 to set the phase shift. The LED is also driven to show modulation in operation without a scope. The LED varies in brightness based on the phase.

Note that there is no active code in this project other than initialization. The operation is entirely hardware based.

Figure 32. Analog PSFB Modulation


The PSFB signals are mixed to modulate an LED. Thus the LED brightness is set based on the overlap. This is easy to see on the development kit.
PSFB_PWM
PWM
tc
pwm
clock
reset
8-bit (UDB)
BUS_CLK
n_psfb_en
A ramp signal is generated with the IDAC and capacitor. The PWM is used as a ramp reset. The comparator and the POT are used to generate an analog PWM.
RESET
RAMP
VREF
PSFB_COMP
Comp
I_SOURCE
IDAC8
Vdd
470pf
10K
Vss
Vss
SOF
SOP
These are just some test signals.
LED
PSFB_1
PSFB
psfb_go
psfb_kill
start_of_frame
start_of_phase
psfb_1A
psfb_1B
psfb_2A
psfb_2B
EPWM1A
EPWM1B
EPWM2A
EPWM2B
BUS_CLK
clock
reset
n_psfb_en
Note: the PSFB is built with a latch; this is intentional. An STA warning will be generated.
PSFB_CTRL
Control Reg
control_0
control_1
control_2
control_3
control_4
control_5
control_6
control_7
n_psfb_en
psfb_go

Results

Increasing the potentiometer output voltage increases the phase and moves the slave pair gate drive signals accordingly. The LED connected to P1_2 changes its brightness accordingly. Figure 33 and Figure 34 show the outputs of the modulator as well as the RAMP, VREF, SOF, and SOP signals noted in Figure 32.

Figure 33. Modulation Results, Less Shift

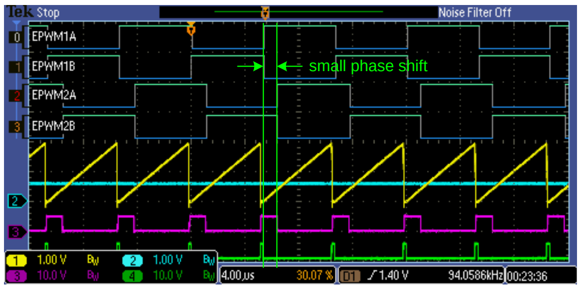
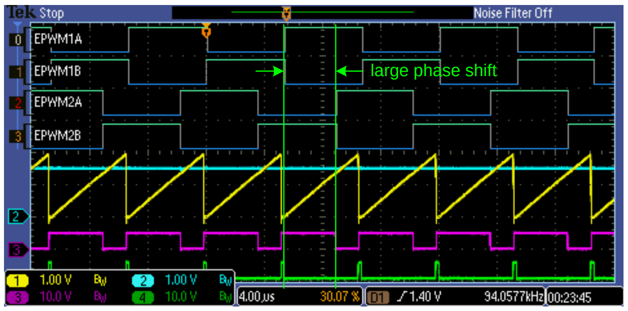


Figure 34. Modulation Results, More Shift

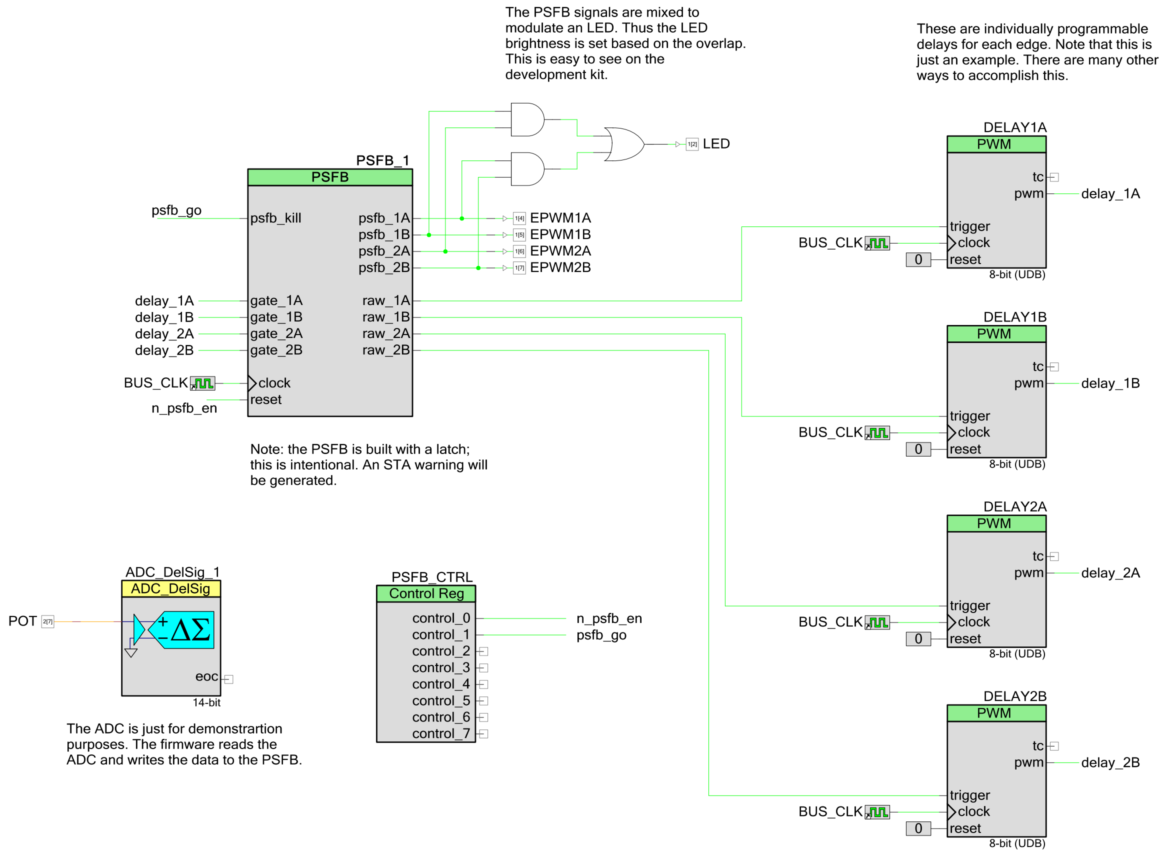


Project Example 3 - PSFB Modulation with Delay

This project is the same as Project 1; however, the design is expanded to include digital delay for each drive signal out of the PSFB, as Figure 35 shows. In this case the delay is introduced using one-shot triggered PWM Components as described in Digital Programmed Delay Generation on page 9.

The code functionality is the same as in Project 1 (except for some initialization of the delay logic). The POT voltage is sampled by the ADC and is used to set the phase shift. The LED is also driven to show modulation in operation without a scope. The LED varies in brightness based on the phase.

Figure 35. Digital Modulation with Programmed Delay



The PSFB signals are mixed to modulate an LED. Thus the LED brightness is set based on the overlap. This is easy to see on the development kit.
These are individually programmable delays for each edge. Note that this is just an example. There are many other ways to accomplish this.
LED
PSFB_1
PSFB
psfb_go
psfb_kill
psfb_1A
psfb_1B
psfb_2A
psfb_2B
EPWM1A
EPWM1B
EPWM2A
EPWM2B
delay_1A
delay_1B
delay_2A
delay_2B
gate_1A
gate_1B
gate_2A
gate_2B
raw_1A
raw_1B
raw_2A
raw_2B
BUS_CLK
clock
reset
n_psfb_en
Note: the PSFB is built with a latch; this is intentional. An STA warning will be generated.
DELAY1A
DELAY1B
DELAY2A
DELAY2B
PWM
tc
pwm
trigger
8-bit (UDB)
ADC_DelSig_1
ADC_DelSig
POT
eoc
14-bit
The ADC is just for demonstrartion purposes. The firmware reads the ADC and writes the data to the PSFB.
PSFB_CTRL
Control Reg
control_0
control_1
control_2
control_3
control_4
control_5
control_6
control_7
n_psfb_en
psfb_go

Results

Figure 36 shows an example with relatively small shift, and Figure 37 shows an example with relatively large shift. Notice that the programmed delay is quite apparent.

Figure 36. Modulation Results, Less Shift

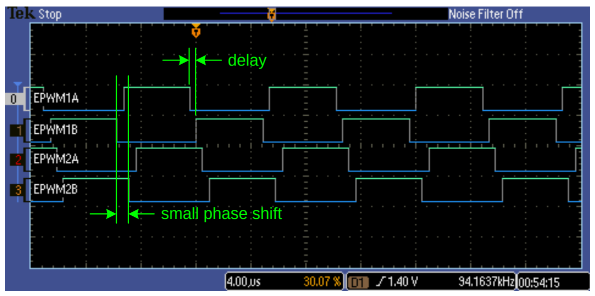
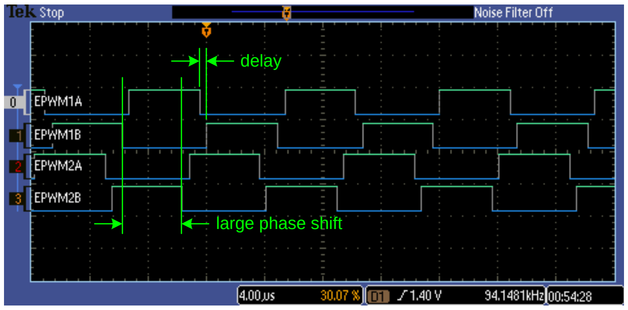


Figure 37. Modulation Results, More Shift

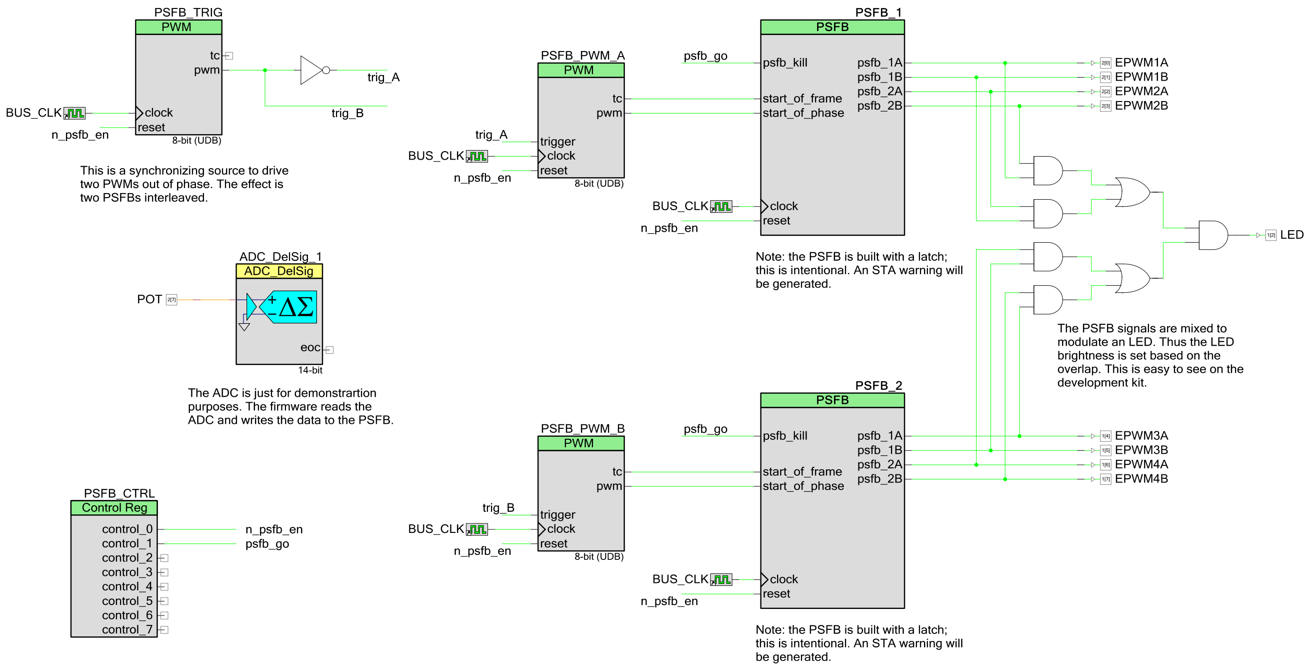


Project Example 4 – Interleaved PSFB Modulation

This project is also similar to project 1; however, the design is expanded to show a dual PSFB design with the bridges running interleaved, as Figure 38 shows.

Again note that the code functionality is the same as in Project 1, except for initialization for the extra logic modulating two bridges. The POT voltage is sampled by the ADC and is used to set the phase shift for both bridges. The LED is also driven to show modulation in operation without a scope. The LED varies in brightness based on the programmed phase with the POT.

Figure 38. Schematic Layout for Interleaved PSFB


PSFB_TRIG
PWM
tc
pwm
clock
reset
8-bit (UDB)
BUS_CLK
n_psfb_en
trig_A
trig_B
This is a synchronizing source to drive two PWMs out of phase. The effect is two PSFBs interleaved.
PSFB_PWM_A
trigger
BUS_CLK
psfb_go
PSFB_1
PSFB
psfb_kill
start_of_frame
start_of_phase
psfb_1A
psfb_1B
psfb_2A
psfb_2B
EPWM1A
EPWM1B
EPWM2A
EPWM2B
LED
Note: the PSFB is built with a latch; this is intentional. An STA warning will be generated.
ADC_DelSig_1
ADC_DelSig
POT
eoc
14-bit
The ADC is just for demonstrartion purposes. The firmware reads the ADC and writes the data to the PSFB.
The PSFB signals are mixed to modulate an LED. Thus the LED brightness is set based on the overlap. This is easy to see on the development kit.
PSFB_2
PSFB_PWM_B
EPWM3A
EPWM3B
EPWM4A
EPWM4B
PSFB_CTRL
Control Reg
control_0
control_1
control_2
control_3
control_4
control_5
control_6
control_7
n_psfb_en
psfb_go

Results

Figure 39 and Figure 40 show the phase-shift modulation for this design. The distinction here is that two bridges are being driven. Notice that the phase relationships between the bridges; they are interleaved.

Figure 39. Modulation Results, Less Shift

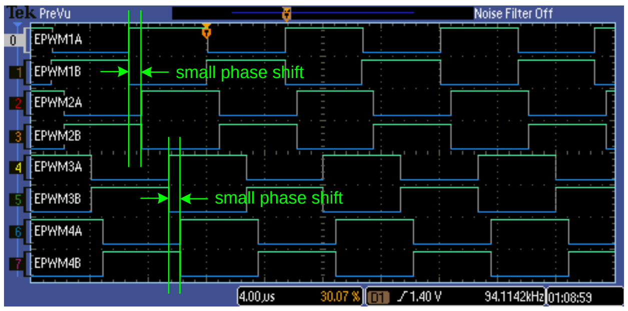
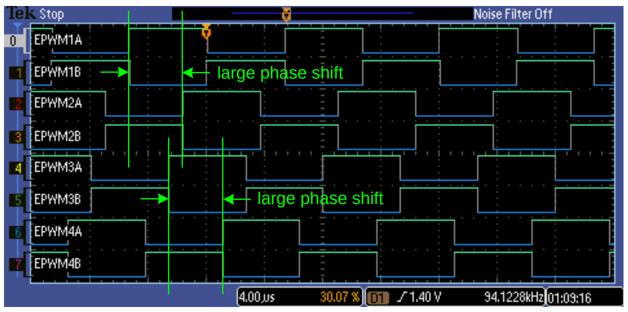


Figure 40. Modulation Results, More Shift



Conclusion

PSoC 3 and PSoC 5LP are amazingly capable devices with a wide degree of flexibility. The flexibility crosses both digital and analog. It is this flexibility that gives PSoC leverage in some Power Applications.

This application note showed how to build a phase-shift full-bridge modulator using the resources available in PSoC. The resource usage spans both the digital and analog capability of PSoC 3 and PSoC 5LP. In addition the idea of controlling the PSFB is approached.

About the Authors

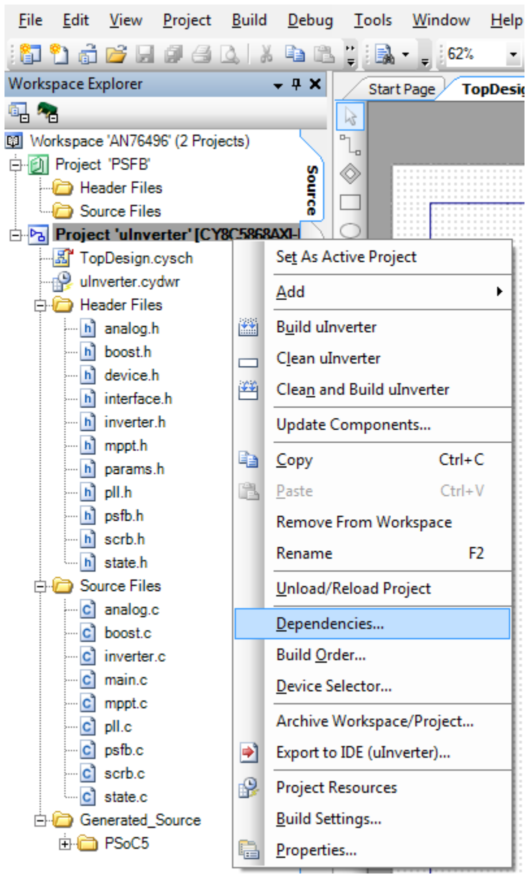
Name:	Ross Fosler
Title:	Member of Technical Staff Applications Engineer
Background:	Ross is an Electrical Engineer with several years experience designing digital controls and embedded firmware for numerous applications. His technical interests are Real-Time Embedded Processing, Control Theory, and Power Electronics.
Contact:	ross@cypress.com

Name:	Srinivas NVNS
Title:	Staff Systems Engineer
Background:	Srinivas is an electrical engineer with background in power electronics, control systems and embedded firmware. He is currently working on power applications using PSoC.
Contact:	snvn@cypress.com

Appendix A: Using the PSFB Component in Your Project

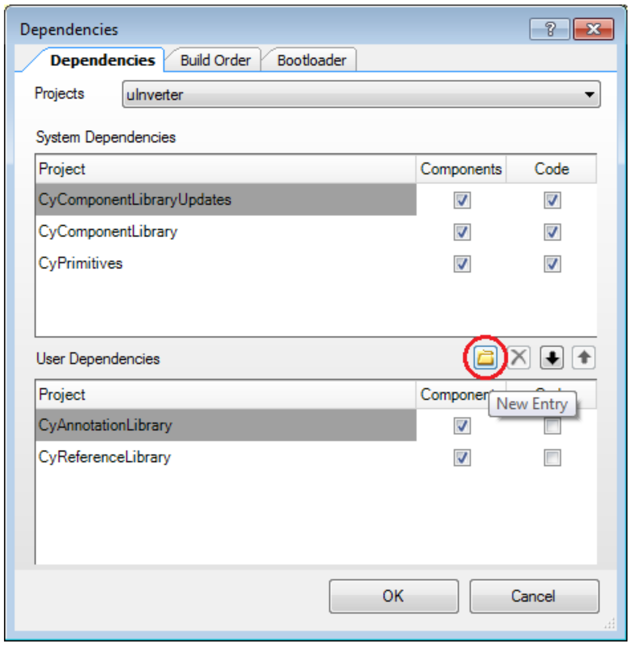
To add the PSFB component to your project, you must add it as a dependency. To do this, right-click on the project's name in Workspace Explorer of the PSoC Creator window. Select the Dependencies option in the pop-up menu, as shown in Figure 41.

Figure 41. Select Dependencies Option



When the Dependencies dialog opens, click the folder icon for User Dependencies, as shown in Figure 42.

Figure 42. Adding a User Dependency



Projects: uInverter

System Dependencies

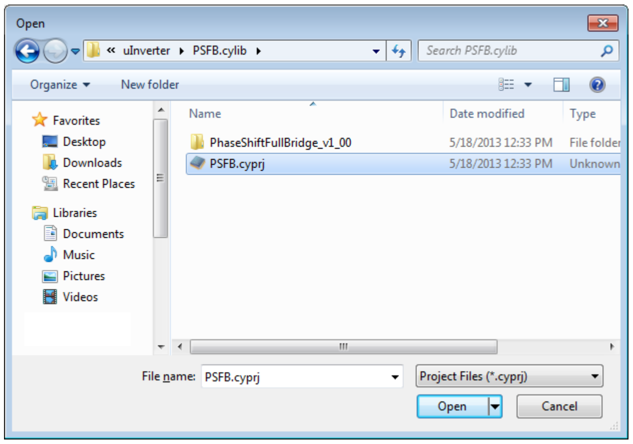
Project	Components	Code
CyComponentLibraryUpdates	✓	✓
CyComponentLibrary	✓	✓
CyPrimitives	✓	✓

User Dependencies

Project	Componen	C
CyAnnotationLibrary	✓	
CyReferenceLibrary	✓	

Navigate to the folder containing the library where the PSFB component resides. In this case, it is in the folder PSFB.cylib. Select the file PSFB.cyprj, as Figure 43 shows.

Figure 43. Select the PSFB Component

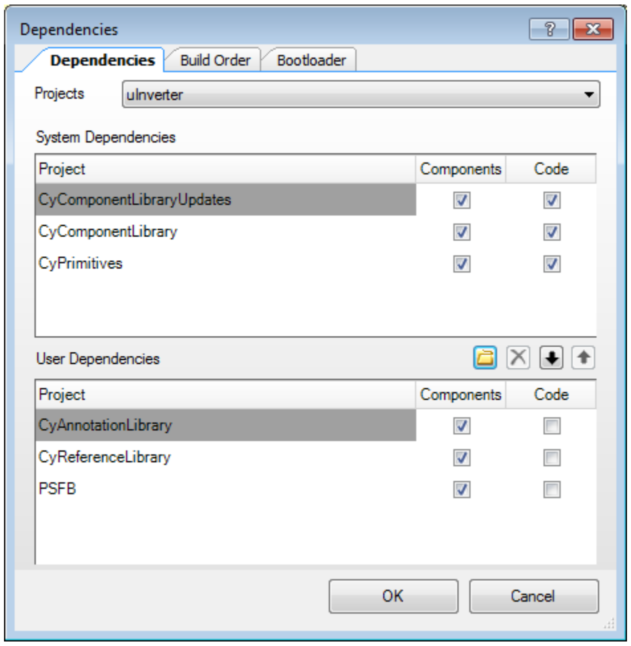


Name	Date modified	Type
PhaseShiftFullBridge_v1_00	5/18/2013 12:33 PM	File folder
PSFB.cyprj	5/18/2013 12:33 PM	Unknown

File name: PSFB.cyprj
Project Files (*.cyprj)

After the PSFB component is added to the project, you will see it in the Dependencies tab, as Figure 44 shows.

Figure 44. PSFB Component Added to Project



Projects: ulnverter

System Dependencies

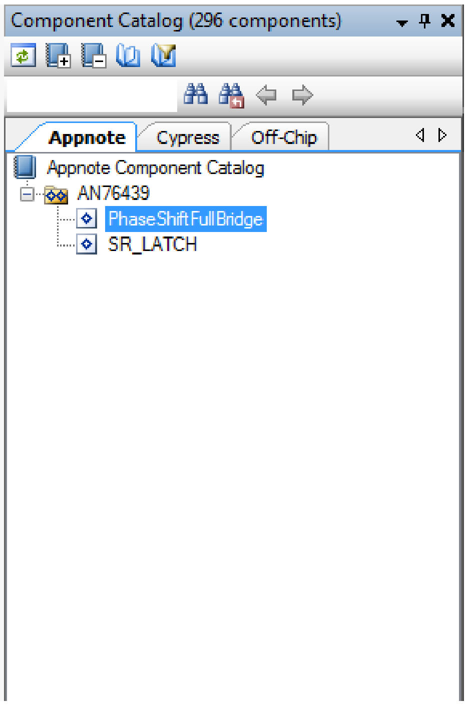
Project	Components	Code
CyComponentLibraryUpdates	✓	✓
CyComponentLibrary	✓	✓
CyPrimitives	✓	✓

User Dependencies

Project	Components	Code
CyAnnotationLibrary	✓	☐
CyReferenceLibrary	✓	☐
PSFB	✓	☐

After the component is added to the project, it will appear in the Component Catalog of PSoC Creator, as Figure 45 shows. It will be listed in the Appnote tab under Appnote Component Catalog/AN76439 entry.

Figure 45. PSFB Component in the Catalog



Document History

Document Title: AN76439 - PSoC® 3 and PSoC 5LP - Phase-Shift Full-Bridge Modulation and Control

Document Number: 001-76439

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4009565	SNVN	05/28/2013	New application note
*A	5704113	AESATMP9	04/20/2017	Updated logo and copyright.

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

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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