Application Report
SPRABE7A–April 2012

Digital Peak Current Mode Control With Slope Compensation Using the TMS320F2803x

Richard Poley and Ali Shirsavar

ABSTRACT

This application report describes a method of implementing the digital peak current mode control of a power supply using the Texas Instruments Piccolo™ MCUs. The theory of operation and all relevant equations are given, as well as a design example and complete compile ready project files. The implementation method described is similar to that taught in Digital Power Design Workshops delivered by Biricha Digital Power Ltd.

Project collateral and source code discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/sprabe7.

Contents

List of Figures

1 Introduction

The operation of a digital peak current mode converter is similar to its analog counterpart. Figure 1 shows a non-synchronous buck converter using analog peak current mode control. In an equivalent digital converter (shown in Figure 2) the compensation network, error amplifier, slope compensation, and pulse width modulation (PWM) generator are all replaced by a microprocessor working in the discrete time domain. This application report describes the use of a Texas Instruments Piccolo MCU (TMS320F2803x) for use in a digital peak current mode power supply. Source code used is provided by Biricha Digital Power Ltd. Note that Texas Instruments also provides a separate Digital Power library for the C28x platform [10] as part of controlSUITE™. Slope compensation using controlSUITE is achieved using alternative means. Nevertheless, the theory and control design procedures presented in this document can be applied to both methods.

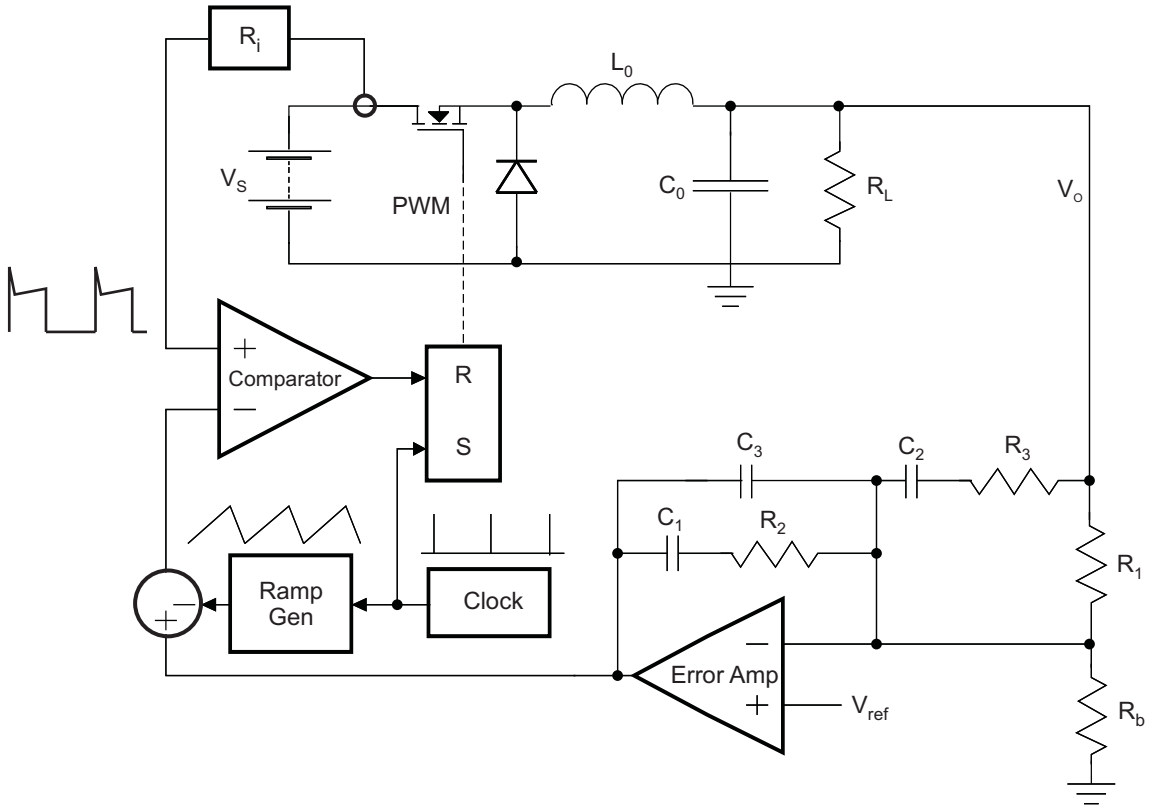

Figure 1. Analog Peak Current Mode Control

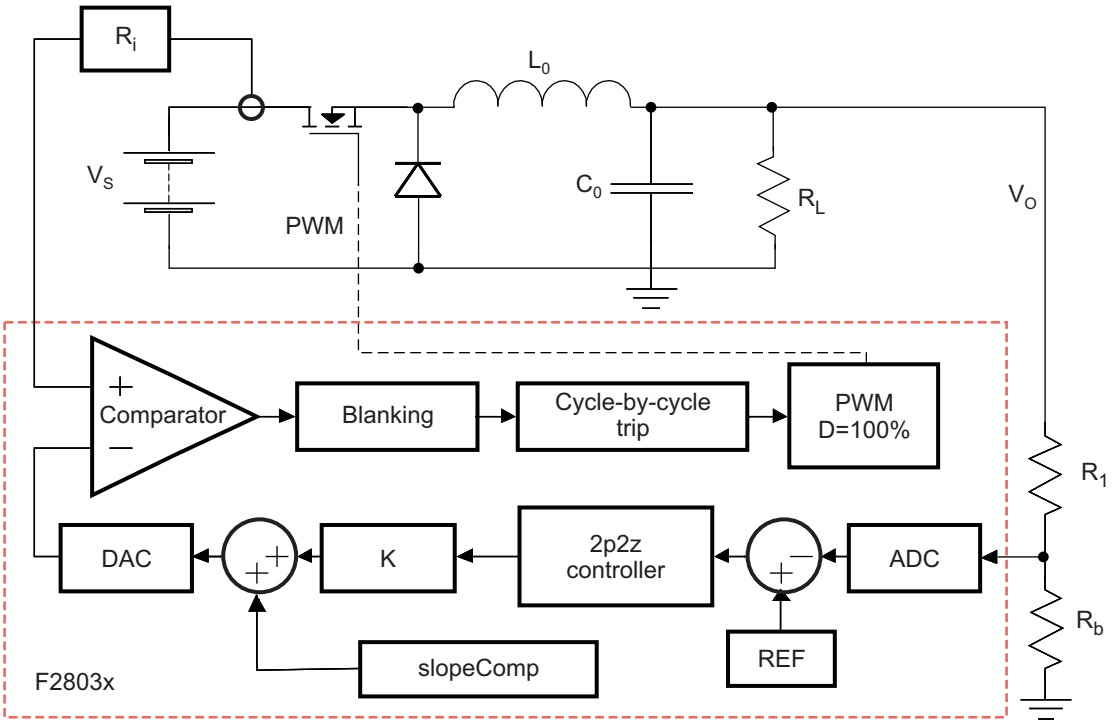

Figure 2. Digital Peak Current Mode Control

As can be seen from Figure 2, the duty is initially set to 100% but tripped using the cycle-by-cycle trip feature of the processor. The output voltage of the converter is applied to a resistive "sampling divider" network that is connected to Piccolo's analog-to-digital converter (ADC). The voltage is sampled and converted to a digital value. A digital reference (REF) is subtracted from the digital value and this error value is used as an input to the digital controller (2p2z Controller). This represents the error amplifier and compensation network of the analog equivalent.

The output of the controller is multiplied by a gain term K, which scales the output of the controller to a digital value suitable for use with the digital analog converter (DAC) of the comparator module. The value of K can be determined as follows:

- The output voltage is applied to the sampling divider. This sampling divider has a gain referred to as *SamplingGain.*
- The ADC has a voltage range of 0 V to 3.3 V (*ADCmaxV*) and a 12-bit digital output giving *ADCbits* = 4095.
- The DAC of the comparator has a 10-bit input range, *DACbits=1023*, and an analog output voltage of 0 V to 3.3V (*DACmaxV*).

These three items are combined to determine the value of K [6].

$$K = \frac{1}{SamplingGain} \times \frac{ADC\max V}{ADCbits} \times \frac{DACbits}{DAC\max V} \quad (1)$$

The output of the 2p2z controller is scaled by K and used as an input to the DAC connected to the comparator's inverting input. The non-inverting comparator input is connected to the current sense transformer, the gain of which is represented by the Ri block. Spurious tripping of the comparator by current spikes caused by turning the MOSFET switch on, can be avoided by the use of leading edge blanking within the Piccolo's comparator module. This feature is described in Section 6. When the inductor current reaches the level of the DAC output, the output of the comparator will go high. This causes a cycle-by-cycle trip event to occur within the digital compare sub-module of the PWM module, forcing the PWM signal low for the remainder of the switching period. Therefore, as with the analog equivalent, the duty is determined by the peak of the current through the power stage of the converter.

2 Modeling the Converter

The locations of the poles and zeros of the compensation network are found through a knowledge of the control-to-output transfer function (for example, the transfer function of the power stage). They must be placed in order to achieve a stable system with suitable phase and gain margins at the desired crossover frequency. The process of compensator pole/zero design are illustrated using a Buck converter example.

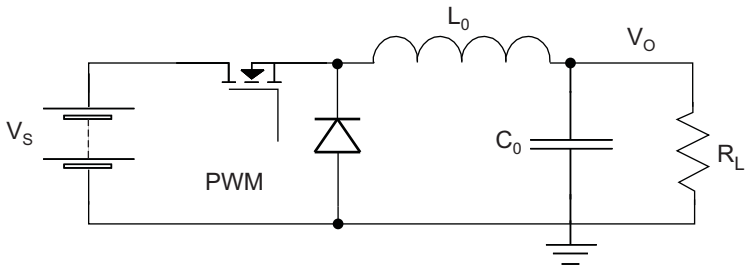

Figure 3. Buck Converter Topology

With current mode control, the inductor shown in the Buck converter in Figure 3 essentially becomes a current controlled source. In [3] the small-signal model of the Buck power stage is given as:

$$H_b(s) = \frac{R_L}{R_i} \times \frac{1 + \frac{s}{\omega_{esr}}}{1 + \frac{s}{\omega_{op}}} \tag{2}$$

This has a pole, ω_{op}, from the output capacitance and load resistance and a zero, ω_{esr}, from the output capacitance and its equivalent series resistance. However, this is not the complete control-to-output transfer function for the converter being controlled under peak current mode. A double pole is also introduced at half the switching frequency and the low-frequency gain is affected by the magnitude of the compensation ramp.

The complete control-to-output transfer function for a Buck converter, as described in [1], is:

$$H_p(s) = \frac{V_{OUT}(s)}{V_{ERR}(s)} = \frac{R_0}{R_i} \times \frac{1}{\left[1 + \frac{R_0 T}{L_0} \times \left(m_c(1-D) - 0.5\right)\right]} \times \frac{1 + \frac{s}{\omega_{esr}}}{1 + \frac{s}{\omega_{op}}} \times \frac{1}{1 + \frac{s}{\omega_n Q_c} + \frac{s^2}{\omega_n^2}} \tag{3}$$

Where:

$$D = \frac{V_O}{V_{IN}} \quad \omega_n = \frac{\pi}{T} \tag{4}$$

The plant's pole from the capacitor and load resistance:

$$\omega_{op} = \frac{1}{R_0 C_0} + \frac{T}{L_0 C_0} \times \left(m_c(1-D) - 0.5\right) \tag{5}$$

The plant's zero from the capacitor and its ESR:

$$\omega_{esr} = \frac{1}{R_{esr} C_0} \tag{6}$$

Switching period: T

Current-sense transformer gain: R_i

Load resistance: R_0

Output voltage: V_o

Input voltage: V_{IN}

Slope compensation factor: m_c

In [1], the sampling effect quality factor is given as:

$$Q_C = \frac{1}{\pi\left(m_C\left(1-D\right)-0.5\right)} \tag{7}$$

Therefore, the choice of quality factor (Q_c), and thus the prevalence of the resonant peak at half the switching frequency determines the size of the external ramp that needs to be added to the sensed inductor current. Typically, (Q_c) is used in order to avoid sub-harmonic oscillations. This allows m_c and the size of the slope compensation ramp to be easily calculated (described in Section 5).

3 Compensation Design

The poles and zeros of the compensation network should be placed according to the analysis of the control-to-output transfer function. A typical example could be to use a Type II compensator for the control of a Buck converter under peak current mode. The transfer function of a Type II compensation network is:

$$H_C\left(s\right) = \frac{\omega_{cp0}}{s} \times \frac{1+\frac{s}{\omega_{cz1}}}{1+\frac{s}{\omega_{cp1}}} \tag{8}$$

The pole, ω_{cp1}, of the compensator is set to the frequency of the ESR zero in the control-to-output transfer function in order to approximately cancel out its effects.

$$\omega_{cp1} = \frac{1}{R_{esr}C_0} \tag{9}$$

The zero, ω_{cz1} , is set to achieve a suitable phase margin and ω_{cp0} is set to achieve the desired crossover frequency. Both the crossover due to the pole at the origin and the zero of the compensator can be calculated analytically. In this application report an approximate solution is given; for the exact solution see the *Biricha Digital Power's Workshop Handbook*.

The frequency of the compensator zero should be set to 20% of the required crossover frequency. Under most circumstances this results in satisfactory phase margin.

$$\omega_{cz1} = \frac{1}{5} \times 2\pi\ f_x \tag{10}$$

Finally, the crossover due to the pole at the origin (or gain of the compensator) is calculated to achieve the desired crossover frequency, f_x. After analyzing the Buck converter's control-to-output transfer function, Equation 9 has been derived for directly calculating of the compensator [1] through [3].

$$\omega_{cp0} = \frac{1.23 f_x R_i \left(L_0 + 0.32 R_0 T\right)\sqrt{1 - 4 f_x^2 T^2 + 16 f_x^4 T^4}\sqrt{1 + \frac{39.48 C_0^2 f_x^2 L_0^2 R_0^2}{\left(L_0 + 0.32 R_0 T\right)^2}}}{L_0 R_0} \tag{11}$$

4 Discrete Time Conversion

In the previous section, the poles and zeros of the analog compensation network were calculated based on a mathematical model of the Buck converter. In order to implement a discrete time compensator, these poles and zeros must be converted into the digital domain. This involves converting from the continuous time s-domain to the discrete time z-domain. Of the various methods to achieve this, the Bilinear (or Tustin) transform represents a relatively simple and effective method.

The transfer function of an analog compensation network can be converted in to the z-domain by replacing the 's' terms with:

$$s \leftarrow \frac{2}{T}\frac{z-1}{z+1} \tag{12}$$

…where T is the sampling period. Equation 12 may then be substituted into the Type II controller transfer function given in Equation 8 to yield the equivalent discrete time compensator transfer function (Equation 13):

$$H_c[z] = \frac{\omega_{cp0}}{\frac{2}{T}\frac{z-1}{z+1}} \times \frac{1 + \frac{\frac{2}{T}\frac{z-1}{z+1}}{\omega_{cz1}}}{1 + \frac{\frac{2}{T}\frac{z-1}{z+1}}{\omega_{cp1}}} \tag{13}$$

After some algebraic manipulation, this can be represented in terms of a standard discrete two-pole two-zero discrete controller transfer function [7].

$$H_c[z] = \frac{B_2 + B_1 z + B_0 z^2}{-A_2 - A_1 z + z^2} \tag{14}$$

Where:

$$B_0 = \frac{\left(T\omega_{cp0}\omega_{cp1}\left(2 + T\omega_{cz1}\right)\right)}{\left(2\left(2 + T\omega_{cp1}\right)\omega_{cz1}\right)}$$

$$B_1 = \frac{\left(T^2\omega_{cp0}\omega_{cp1}\right)}{\left(2 + T\omega_{cp1}\right)}$$

$$B_2 = \frac{\left(T\omega_{cp0}\omega_{cp1}\left(-2 + T\omega_{cz1}\right)\right)}{\left(2\left(2 + T\omega_{cp1}\right)\omega_{cz1}\right)}$$

$$A_1 = \frac{4}{\left(2 + T\omega_{cp1}\right)}$$

$$A_2 = \frac{\left(-2 + T\omega_{cp1}\right)}{\left(2 + T\omega_{cp1}\right)} \tag{15}$$

Note that all the variables are now known and therefore the coefficients can be calculated.

To obtain the linear difference equation (LDE) from the two-pole two-zero discrete transfer function, first multiply both top and bottom through byz^{-2}.

$$H_c[z] = \frac{Y[z]}{X[z]} = \frac{B_2 z^{-2} + B_1 z^{-1} + B_0}{-A_2 z^{-2} - A_1 z^{-1} + 1} \tag{16}$$

Equation 16 can now be rearranged to obtain the linear difference equation that can be calculated using a microprocessor:

$$\begin{aligned} &Y[z]\left(-A_2 z^{-2} - A_1 z^{-1} + 1\right) = X[z]\left(B_2 z^{-2} + B_1 z^{-1} + B_0\right) \\ &-A_2 y[n-2] - A_1 y[n-1] + y[n] = B_2 x[n-2] + B_1 x[n-1] + B_0 x[n] \\ &y[n] = B_2 x[n-2] + B_1 x[n-1] + B_0 x[n] + A_2 y[n-2] + A_1 y[n-1] \end{aligned} \tag{17}$$

Where x[n] is the error input to the controller for this sampling period and y[n] is the controller output for this sampling period. x[n-1] denotes the pervious sampling period and x[n-2] is two sampling periods in the past.

The A_n and B_n coefficients in the LDE could be calculated using Equation 14, however there is no need for this transform to be calculated by hand. An on-line automated tool exists on the Biricha Digital website [12] to convert from s-domain poles and zeros to the coefficients required by the discrete time digital controller [9].

5 Slope Compensation

As with analog peak current mode control, ramp slope compensation needs to be applied such that no sub-harmonic oscillations occur. These oscillations are caused by the current feedback loop that has a high frequency term in the control-to-output transfer function. The required ramp can be calculated analytically [1]. For all power converters using peak current mode control, the high frequency term is given by:

$$F_h(s) = \frac{1}{1 + \frac{s}{\omega_n Q_c} + \frac{s^2}{\omega_n^2}} \tag{18}$$

where the sampling effect quality factor, Q_c, is given by Equation 7.

The aim of adding slope compensation is to remove the sub-harmonic oscillations. This is achieved by adding sufficient ramp to reduce the $Q_c$of the double pole to a value of 1 or less [3]. The slope compensation factor is made the subject of Equation 7 and the result is simplified by setting Q_c=1, then:

$$m_c = \frac{1 + \pi 0.5}{\pi (1 - D)} \tag{19}$$

Using the method described in [3], the peak-to-peak value of the external compensation ramp has been calculated in [2] and is given in Equation 20.

$$V_{PP} = -\frac{(0.18 - D) R_i T_s V_{IN}}{L_0} \tag{20}$$

This value can now be used in a digital slope compensation algorithm [11]. This capability is implemented on the TMS320F2803x using the functions provided by the Biricha Digital's Chip Support Library (CSL) [8]. The CSL is a comprehensive set of library functions written specifically for easy programming of the peripherals of the C2000™ family of MCUs.

The Biricha code uses the Control Law Accelerator (CLA) of the TMS320F2803x to implement the slope compensation. The CLA is an integrated floating-point core that allows high speed, real-time control algorithms to be executed in parallel with the main CPU.

The compensation slope is subtracted from the DAC register at discrete intervals within the switching period. Therefore, the height of the slope must be converted to a digital value and divided up into a number of steps that will be subtracted over each switching period.

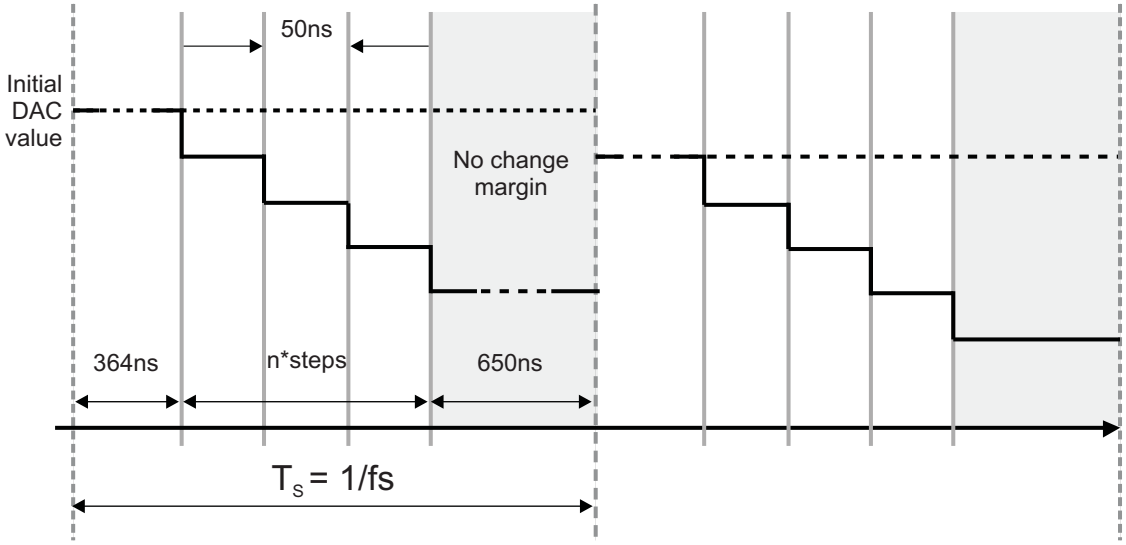

Figure 4. Digital Slope Compensation

The comparator DAC has 10-bits of resolution and an output voltage range of 0 V to 3.3 V. Therefore, the ramp height can be converted into a discrete number as follows:

$$Ramp = V_{PP} \times \frac{1023}{3.3} \tag{21}$$

Each step of the discrete ramp requires three instructions to complete. Each instruction takes 16.666 ns to execute assuming a 60 MHz system clock. Therefore, each step lasts for a fixed duration of 50 ns. The designer must calculate how many 50 ns steps are required within each period and what the *ΔRamp* should be for each step (a negative value added to the DAC during each step).

The DAC value must be set before the CLA slope code begins executing. This is because the CLA code reads the DAC value, adds the *ΔRamp*, and then writes this back to the DAC register. Therefore, the CLA must update the DAC value with the final step well before the end of the switching period. When the new DAC value is written, near the beginning of the next period, it must not be overwritten with the previous value (with *ΔRamp* added by the CLA).

Therefore, allowing a safety margin, the time period of the slope must be less than the switching period. The DAC value is first adjusted by the CLA 364 ns after the PWM interrupt. As a safety margin, allow the slope code to finish executing thirteen steps (650 ns) before the end of the switching period. The number of steps can be calculated as follows:

$$Steps = \left[\frac{1}{f_s} - 364\ ns - (13 \times 50\ ns)\right] \times \frac{1}{50\ ns} \tag{22}$$

The *ΔRamp* value can be calculated given that the number of steps is now known:

$$\Delta Ramp = -\frac{Ramp}{Steps} \tag{23}$$

The CLA slope compensation code is generated using a macro that must be placed at the top of the C file, before the main function begins. A full description of the code structure will be given shortly by the way of a design example. For now it is important to become familiar with how the calculated parameters are passed to the slope compensation code:

```
CLA_slopeCode( Name, Comp, Pwm, Delta, Steps )
```

The values passed to this function must be literals. Constants, variables or other macros cannot be used. The parameters are defined as follows:

Name – The name of the CLA task, in this case “SlopeTask".

Comp – The module number of the comparator module being used.

Pwm – The module number of the PWM module being used.

Delta – The *ΔRamp* value to be added to the DAC register each step.

Steps – The number of steps (each lasting 50 ns for a 60 MHz clock) to execute during each task call.

6 Leading Edge Blanking

The sensed inductor current is a noisy signal. During switch turn-on there is a large current spike that could potentially cause the comparator to trigger and a spurious PWM trip event to occur. To prevent this, leading edge blanking may be applied to the comparator module. This hardware feature allows you to specify a short “blanking window” following a PWM switch event, during which the comparator’s output is ignored by the digital compare sub-module. An example of the blanking window is shown in the shaded periods in Figure 5.

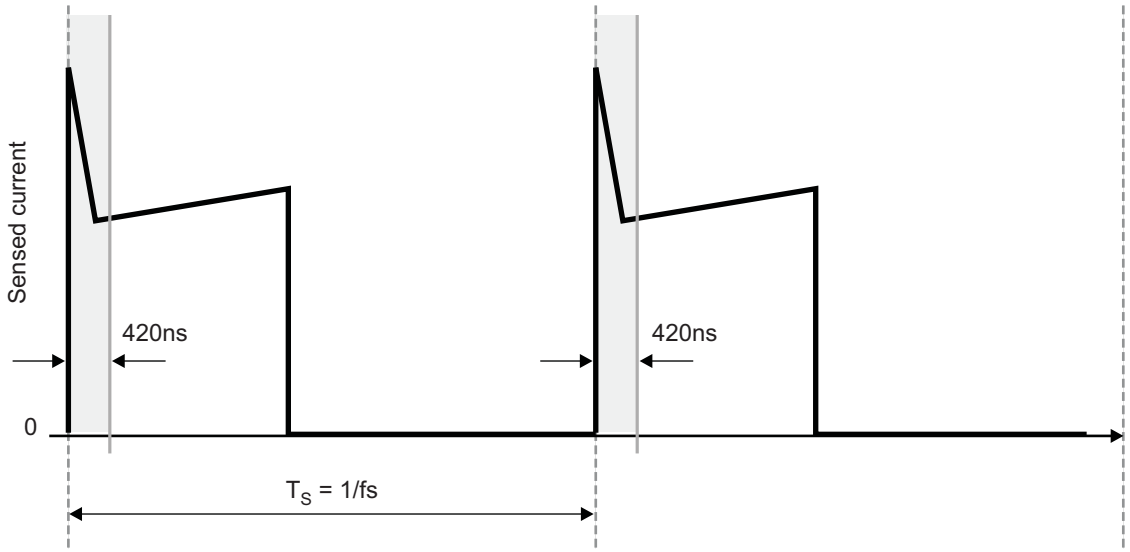

Figure 5. Measure the Required Amount of Leading Edge Blanking

The designer must specify the number of nanoseconds required for the duration of the blanking window. This is achieved using another Biricha CSL function PWM_setBlankingWindow(). An initial value should be set and its suitability can be confirmed using an oscilloscope. In the example shown in Figure 5, a window of 420 ns is required. This can be set up using the functions described below.

```
PWM_configBlanking( PWM_MOD_1, PWM_CMP_COMP2, GPIO_NON_INVERT,
 true );

PWM_setBlankingWindow( PWM_MOD_1, PWM_nsToTicks(420) );
```

The PWM_configBlanking() function configures the digital compare sub-module of the PWM module to use the output of the comparator as an event trigger.

7 Design Example

While the principles described in this application report can be applied to a converter of any type, a design example is presented for digital peak current mode Buck converter with the following specification:

V_{IN} = 12 V

V_O = 3.3 V

I_O = 2A

R_L = 1.65 Ω

L_0 = 22 µH

C_0 = 440 µF

R_{ESR} = 31 mΩ

R_i = 24 Ω/50 = 0.48 Ω

D (nominal, for a Buck)=3.3/12 = 0.275

f_s = 200 kHz

f_x = 15 kHz

The switching frequency is chosen as $_s$ = 200 kHz and the desired crossover frequency is f_x=15 kHz.

Step 1: Calculation of Slope Compensation

Step 1. Calculation of slope compensation

- Calculate the peak-to-peak value of the external compensation ramp required to achieve using Equation 20 that is specific to the Buck converter:

$$V_{PP} = -\frac{(0.18 - D)R_i T_S V_{IN}}{L_0}$$

$$V_{PP} = -\frac{(0.18 - 0.275)\times 0.48 \times 5\ \mu s \times 12}{22\ \mu H}$$

$$V_{PP} = 0.124\ V \tag{24}$$

- Convert this to a discrete number:

$$Ramp = V_{PP} \times \frac{1023}{3.3}$$

$$Ramp = 0.124 \times \frac{1023}{3.3}$$

$$Ramp = 38 \tag{25}$$

- Calculate the number of steps that the slope compensation task can execute during one switching period:

$$Steps = \left[\frac{1}{f_S} - 364\ ns - (13 \times 50\ ns)\right] \times \frac{1}{50\ ns}$$

$$Steps = \left[\frac{1}{200\ kHz} - 364\ ns - (13 \times 50\ ns)\right] \times \frac{1}{50\ ns}$$

$$Steps = 80 \tag{26}$$

- Find the Δ*Ramp* value. This is calculated from the discrete number of steps:

$$\Delta Ramp = -\frac{Ramp}{Steps}$$

$$\Delta Ramp = -\frac{38}{83}$$

$$\Delta Ramp = -0.45 \quad (27)$$

Step 2: Compensator Poles and Zeros

This step refers to when a Buck converter with a Type II compensator is used. The transfer function is given in Equation 8. The pole, ω_{cp1} , is used to cancel out the ESR zero of the output capacitor and equivalent series resistance:

$$\omega_{cp1} = \omega_{esr}$$

$$\omega_{cp1} = 73314 \ \ rads^{-1} \quad (28)$$

The zero of the compensator is used to set the phase margin of the open loop system at the crossover frequency. An approximate solution that gives reasonable results is to set the zero to one fifth of the crossover frequency.

$$\omega_{cz1} = 18850 \ \ rads^{-1} \quad (29)$$

Finally, ω_{cp0} is calculated to achieve the desired crossover frequency using Equation 9.

$$\omega_{cp0} = 363245 \ \ rads^{-1} \quad (30)$$

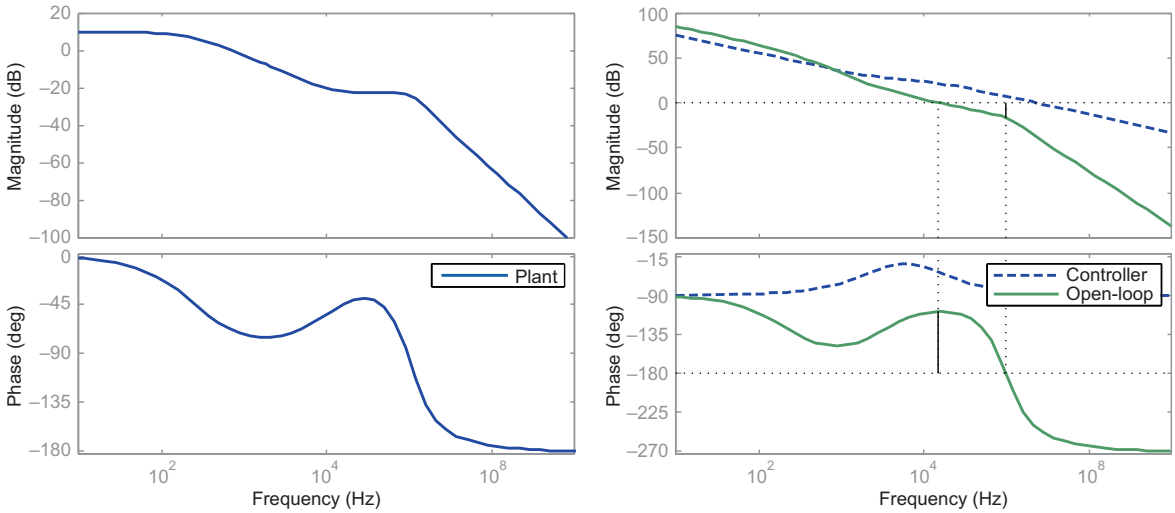

Figure 6. Bode Plots of Plant, Controller and Open-Loop System (GM = 16.6dB, PM = 70.9°)

Figure 6 shows the frequency response of the plant, controller and open loop systems. The left-hand plot of the plant transfer function clearly shows the double-pole at half the switching frequency as suggested in [3]. The choice of compensation ramp has effectively damped the resonant peak of this double pole.

The dashed line of the right-hand plot shows the pole at the origin, pole and zero of the compensator. The solid line on this plot represents the combined plant and controller transfer function. This is the open loop system response. The gain and phase margins are determined from this open loop system.

The controller has been designed to achieve large phase margin at the crossover frequency of 15kHz. The Bode diagram of the open loop system indicates that the poles and zeros of the controller give a phase margin of 70.9° at the crossover frequency.

Step 3: Bilinear Transform

The bilinear transform is performed using the automated tool for converting Type II controllers from the continuous time domain into the discrete time domain. This is found on the Biricha website: http://www.biricha.com/resources/converter.php?type=2.

The following information is entered into the online form:

Crossover frequency of analog pole at zero $f_{cp0} = \frac{\omega_{cp0}}{2\pi} = 57812\ Hz$

Frequency of second pole: $f_{cp1} = \frac{\omega_{cp1}}{2\pi} = 11668\ Hz$

Frequency of first zero: $f_{cz1} = \frac{\omega_{cz1}}{2\pi} = 3000\ Hz$

Switching frequency: $f_s = 200\ kHz$

The result is calculated as: $H[z] = \frac{U[z]}{E[z]} = \frac{B_2 z^{-2} + B_1 z^{-1} + B_0}{-A_2 z^{-2} - A_1 z^{-1} + 1}$

Where:

A1 = 1.69021629
A2 = -0.69021629
B0 = 3.12552798
B1 = 0.28131731
B2 = -2.84421068

These are the coefficients that the designer should use with the two-pole two-zero controller equation such as the one provided by Biricha Digital. If using the Biricha Digital code, all you need to do is to enter these coefficients at the top of you main C file using #define statements:

```
#define A1  +1.69021629
#define A2  -0.69021629
#define B0  +3.12552798
#define B1  +0.28131731
#define B2  -2.84421068
```

Step 4: Calculating K, the Reference and the CLA Ramp

The gain term, K, must be calculated to scale the output of the controller to a digital value that is suitable for use with the DAC of the comparator module. Equation 1 is used below. In this design example, two equal resistors are used for the sampling divider giving the divider a gain of 0.5.

$$K = \frac{1}{0.5} \times \frac{3.3}{4095} \times \frac{1023}{3.3}$$

$$K = 0.4996 \approx 0.5 \tag{31}$$

The reference value has the same purpose as that of the non-inverting input reference voltage on the error amplifier in the analog current mode power supply. It is used to calculate the digital error value by subtracting the reference value from the input value. The digital error value is then used as an input to the controller. Therefore, the reference value must be equal to the output voltage multiplied by the sampling divider gain and then converted to its digital equivalent.

$$REF = V_O \times SamplingGain \times \frac{ADCbits}{ADC\max V}$$

$$REF = 3.3 \times 0.5 \times \frac{4095}{3.3}$$

$$REF = 2047.5 \tag{32}$$

This must be rounded to an integer for use in the conversion function _IQ15toF(). Therefore, REF = 2048. In the `#define` statement this is then converted to the floating point equivalent of the IQ number.

The Biricha 2p2z controller can now be configured by defining the following constants at the top of the main C file.

```
/* Set up the coefficients for the 2p2z controller
 */

#define K   (0.5)
#define REF (_IQ15toF(2048))
#define MIN_DUTY 0
#define MAX_DUTY 1023
#define A1  +1.69021629
#define A2  -0.69021629
#define B0  +3.12552798
#define B1  +0.28131731
#define B2  -2.84421068

#define PERIOD_NS  5000 /* Period in ns for fs = 200kHz */
```

The CLA macro must be placed before the main function beings as this generates the necessary assembly code to be executed by the CLA.

```
CLA_slopeCode( SlopeTask, 2, 1, -1.0, 80 );
```

Finally, the remaining set up and initialization code within the main C file is entered. This process is described in Section 8.

8 Setting Up the Biricha Code

The Biricha Digital Chip Support Library provides a fast and simple method of configuring the Texas Instruments F2803x DSPs for use in digital power applications. Knowledge of the DSP's internal registers and associated configuration bits is not required. In place of this, simple function calls are used. The code included with this application report provides an example of a complete digital peak current mode 2p2z controller for the TMS320F2803x device.

The Biricha Digital CSL documentation contains full descriptions and examples of all of the functions used in code and can be accessed from the following URL:https://www.ti.com/lit/zip/sprabe7. It is suggested that you refer to the Biricha documentation for more information.

Figure 7 through Figure 9 show flow diagrams of the software that accompanies this application report.

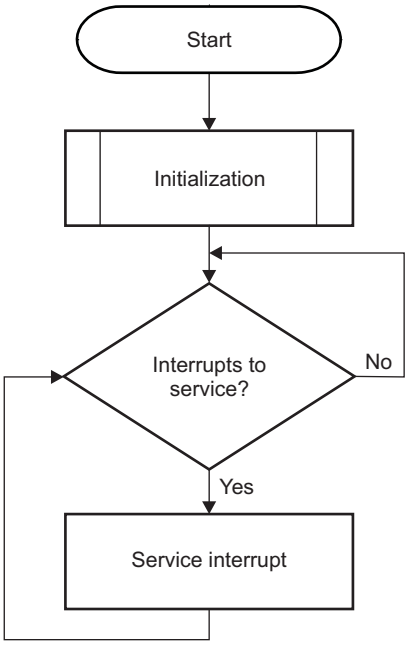

Figure 7. Main Function Program Flow

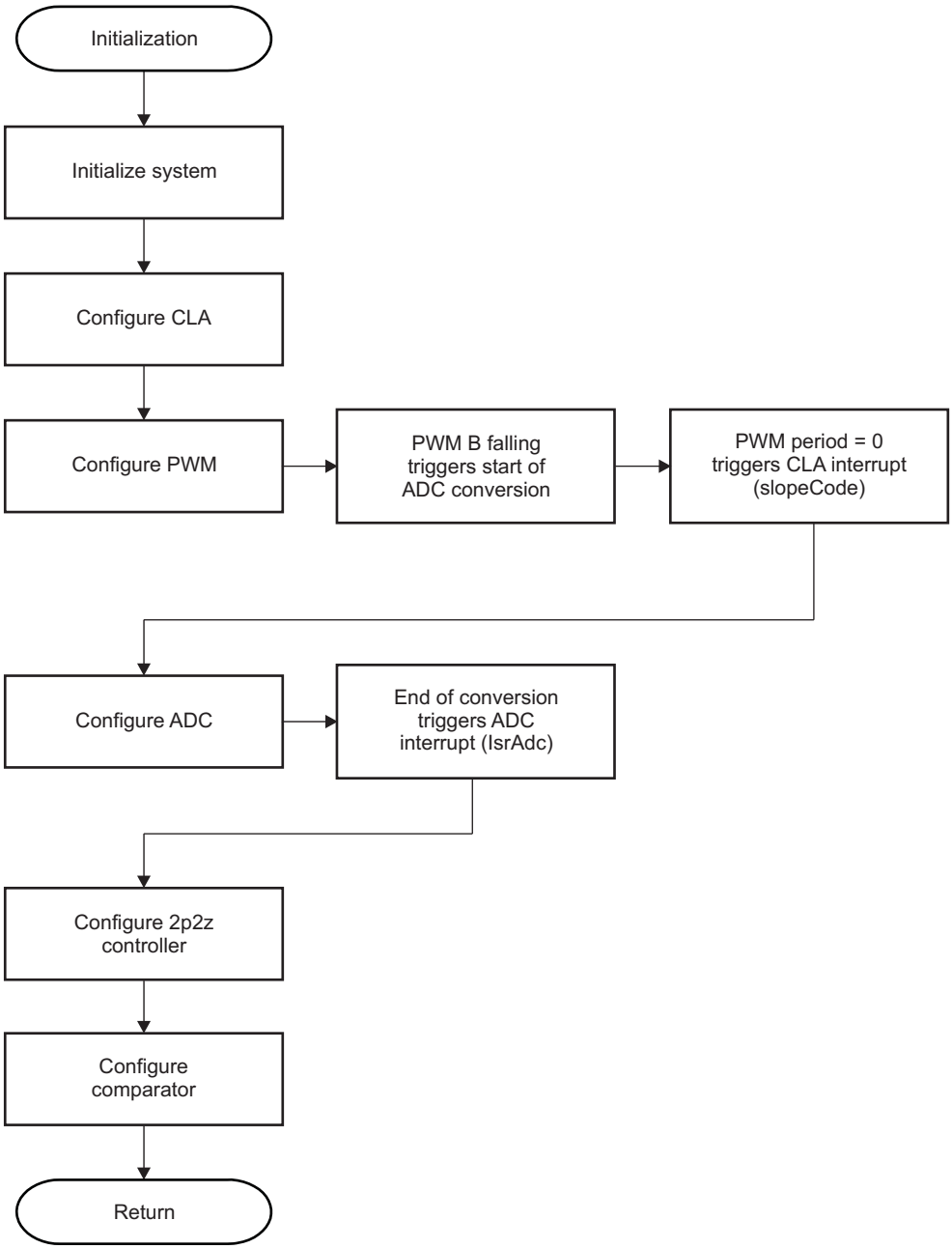

Figure 8. Main Function Initialization Routine

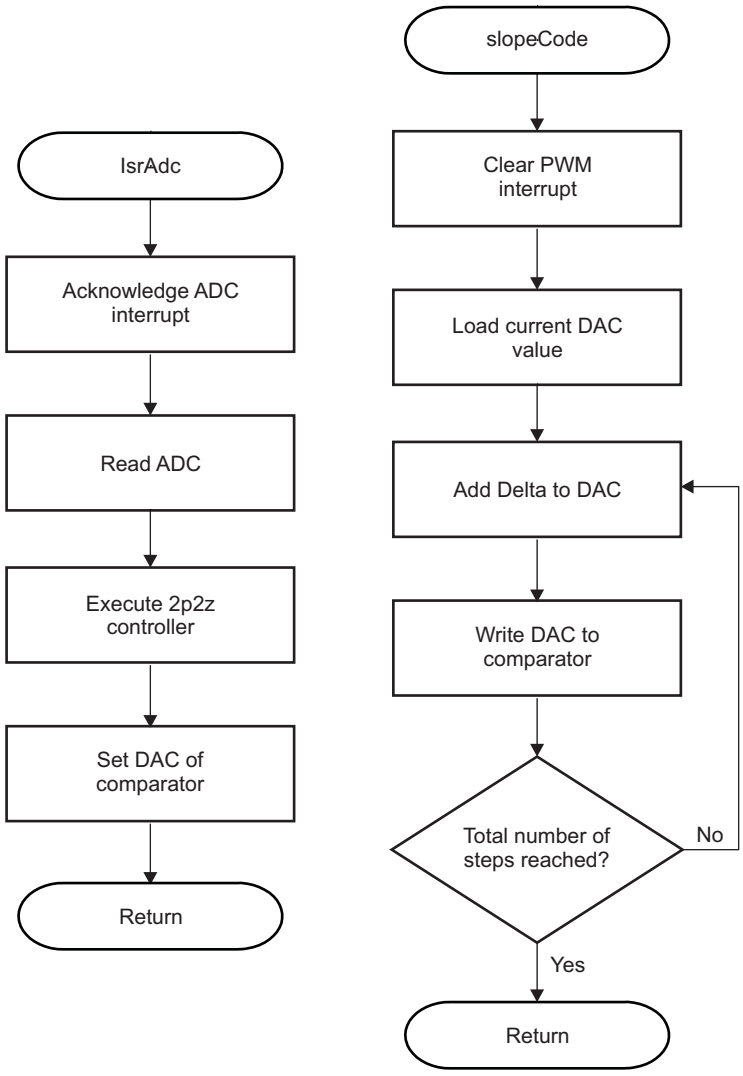

Figure 9. ADC and CLA Interrupts

An analysis of the code within the main() function follows. First of all, the system and peripherals must be initialized before they can be used.

```
/* Initialize the MCU and ADC */
SYS_init();
ADC_init();
```

The CLA module is triggered by a PWM interrupt and should be configured before the PWM module is configured.

```
/* Configures CLA_MOD_1 to run CLA code "SlopeTask" whenever
 * PWM trigger occurs
 */
 CLA_config (CLA_MOD_1, &SlopeTask, CLA_INT_PWM);
```

The PWM module 1 is configured to operate at the switching frequency of 200 kHz. Channel A is connected to the MOSFET Driver IC and controls the switching of the MOSFET in the Buck power stage. Channel B is used for timing purposes:

- The duty of each PWM channel is set individually. Channel A is set to 100% duty as with peak current mode control the effective duty is determined when the current through the switch reaches the output of the controller. At this point the PWM signal goes low and the MOSFET turns off.

- The duty of PWM Channel B should be set such that the output goes low and triggers the sampling, conversion, ADC interrupt entry, 2p2z algorithm, scaling and DAC set up just before the start of the next PWM period – when PWM Channel A goes high.

Therefore, the purpose of PWM Channel B is to start this process. The falling edge of PWM Channel B is used to start the sampling process followed by all the relevant calculations. Therefore, the duty of Channel B should be set such that all sampling, conversion and subsequent calculations are completed just as PWM Channel A goes high. This time has been measured using an oscilloscope as 2.45 µs for the code included in this application report. This time can be measured by toggling a general-purpose input/output (GPIO) pin at the end of the interrupt service routine and observing the output using an oscilloscope.

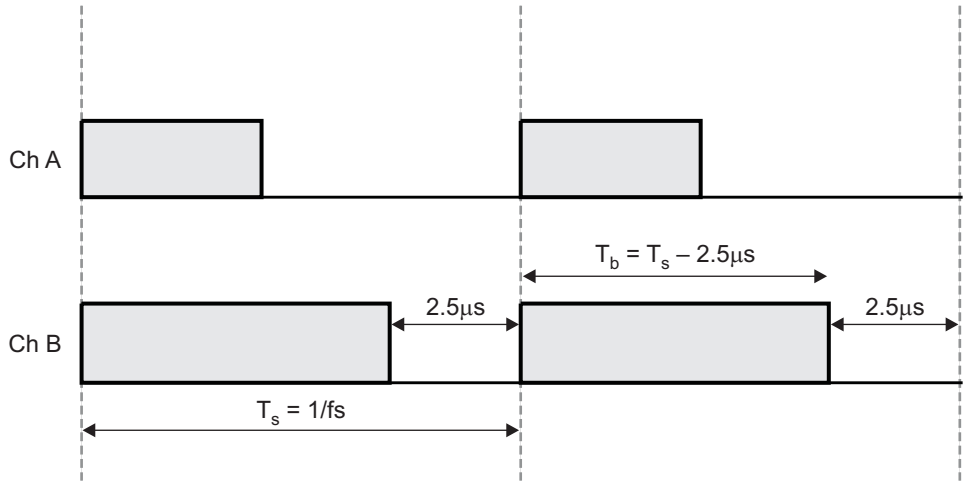

Figure 10. Channel B Duty With Respect to Channel A Duty

PWM Channel B can then be used as the trigger to start the ADC conversion. The ADC trigger is set to occur when the counter is equal to the duty value of PWM channel B; this is when the output of Channel B goes low.

```
/* Setup PWM_MOD_1 for fs = 200kHz. PWM1 Ch A is used for switching
 * the MOSFET.
 */
PWM_config( PWM_MOD_1, PWM_nsToTicks(PERIOD_NS), PWM_COUNT_UP );
PWM_pin( PWM_MOD_1, PWM_CH_A, GPIO_NON_INVERT );
PWM_pin( PWM_MOD_1, PWM_CH_B, GPIO_NON_INVERT );

/* Set the maximum duty to 100%. The trip zones (configured later)
 * will end the high output of the PWM when the current reaches
 * the slope level.
 */
PWM_setDutyA( PWM_MOD_1, PWM_nsToTicks(PERIOD_NS) );

/* Sets the PWM1 Ch B such that the calculations are complete
 * just before the rising edge PWM A.
 */
PWM_setDutyB( PWM_MOD_1, PWM_nsToTicks( PERIOD_NS-2450+0 ) );

/* This sets up the PWM Mod1 to start the ADC conversion whenever
 * PWM1 Channel B timebase counter matches Ch B's duty.
 */
PWM_setAdcSoc( PWM_MOD_1, PWM_CH_B, PWM_INT_CMPB_UP );


/* This sets up PWM Mod1 to generate an interrupt every PWM
 * cycle whenever timebase counter = 0.
 */
PWM_setCallback( PWM_MOD_1, 0, PWM_INT_ZERO, PWM_INT_PRD_1 );
```

The following functions set up the one shot trip of the PWM output triggered by the comparator output. The PWM_configBlanking() function effectively connects the comparator output to the PWM module using the digital compare sub-module. The blanking window size is set within the digital compare sub-module and the digital compare event is used for the trip zones configured within PWM module.

```
/* This effectively feeds the output of comparator Mod2 into
 * PWM Mod1 and activates the blanking by setting the digital
 * compare event PWM_DCEVT at the correct time.
 */
PWM_configBlanking( PWM_MOD_1, PWM_CMP_COMP2, GPIO_NON_INVERT,
 true );

/* Sets the size of the blanking window to 420ns */
PWM_setBlankingWindow( PWM_MOD_1, PWM_nsToTicks(420) );

/* Sets the relevant trip zones: i.e. when PWM_DCEVT occurs clear
 * PWM1 Ch A on a cycle by cycle basis but takes no action on
 * PWM1 Ch B
 */
PWM_setTripZone(  PWM_MOD_1, PWM_DCEVT, PWM_TPZ_CYCLE_BY_CYCLE );
PWM_setTripState( PWM_MOD_1, PWM_CH_A,  GPIO_CLR );
PWM_setTripState( PWM_MOD_1, PWM_CH_B,  GPIO_NO_ACTION );
```

The ADC Module 1 is configured to read and convert the output voltage from Channel B2. The conversion is triggered from the start of conversion event of PWM module 1. When the conversion is complete an interrupt is called and the interrupt service routine IsrAdc() is entered. This interrupt service routine is included in Section 12.

```
/* Configures ADC to sample Vo when triggered by PWM1 Ch B's
 * falling edge
 */
ADC_config( ADC_MOD_1, ADC_SH_WIDTH_7, ADC_CH_B2, ADC_TRIG_EPWM1_SOCB );

/* When conversion is finished, cause interrupt and jump to IsrAdc
 */
ADC_setCallback( ADC_MOD_1, IsrAdc, ADC_INT_1 );
```

The control structure is initialized with the values determined from the bilinear transform of the compensator transfer function. A soft start can also be configured.

```
/* Initialize the 2p2z control structure */
CNTRL_2p2zInit(&MyCntrl
               ,_IQ15(REF)
               ,_IQ26(A1),_IQ26(A2)
               ,_IQ26(B0),_IQ26(B1),_IQ26(B2)
               ,_IQ23(K),MIN_DUTY,MAX_DUTY
               );

/* Set up a 500ms soft-start */
CNTRL_2p2zSoftStartConfig(&MyCntrl, 500, PERIOD_NS );
```

The comparator is configured in asynchronous mode with a non-inverted output. The inverting input of the comparator is tied to the internal DAC. The DAC value is set by the control algorithm.

```
/* Configures the comparator Mod2 */
CMP_config( CMP_MOD_2, CMP_ASYNC, GPIO_NON_INVERT, CMP_DAC );
```

Finally, global interrupts must be enabled before any of the interrupts can be serviced. After this, the execution waits in an idle loop as all of the events will now occur using interrupts.

```
/* Enables global interrupts and wait in idle loop */
INT_enableGlobal(true);

while(1)
{

}
```

The complete code listing, including the code for the interrupt functions, can be found in Section 12.

9 Measurement Results

The working converter is connected to a Bode 100 frequency response analyzer from Omicron Lab in order to measure its small signal frequency response. The output voltage feedback loop is broken and the frequency analyzer is inserted in to the feedback path.

Figure 11 compares the predicted “analog” open loop model (dashed line) with the measured results (solid line). The low and high frequency measurement inaccuracies are expected.

The gain plot at low frequencies is less than the modeled result. This is due to the combination of the way in which the frequency response is measured and the discrete time nature of the controller. When the frequency of the injected signal is low, the input to the discrete time controller does not significantly change from cycle to cycle and the response cannot be accurately measured. Nevertheless, the controller is functioning correctly; the pole at origin is present within the system and there is zero steady state error.

The phase plot shows good agreement with the predicted model. As expected, phase roll-off becomes apparent as the perturbation frequency approaches the switching frequency. This is due to the sampling and reconstruction process and the pure time delays added during the calculations.

Overall the measured results show a good correlation to the results of the model. The measured open loop gain crosses the 0dB axis at the desired crossover frequency with approximately 70° of phase margin and a 12dB gain margin.

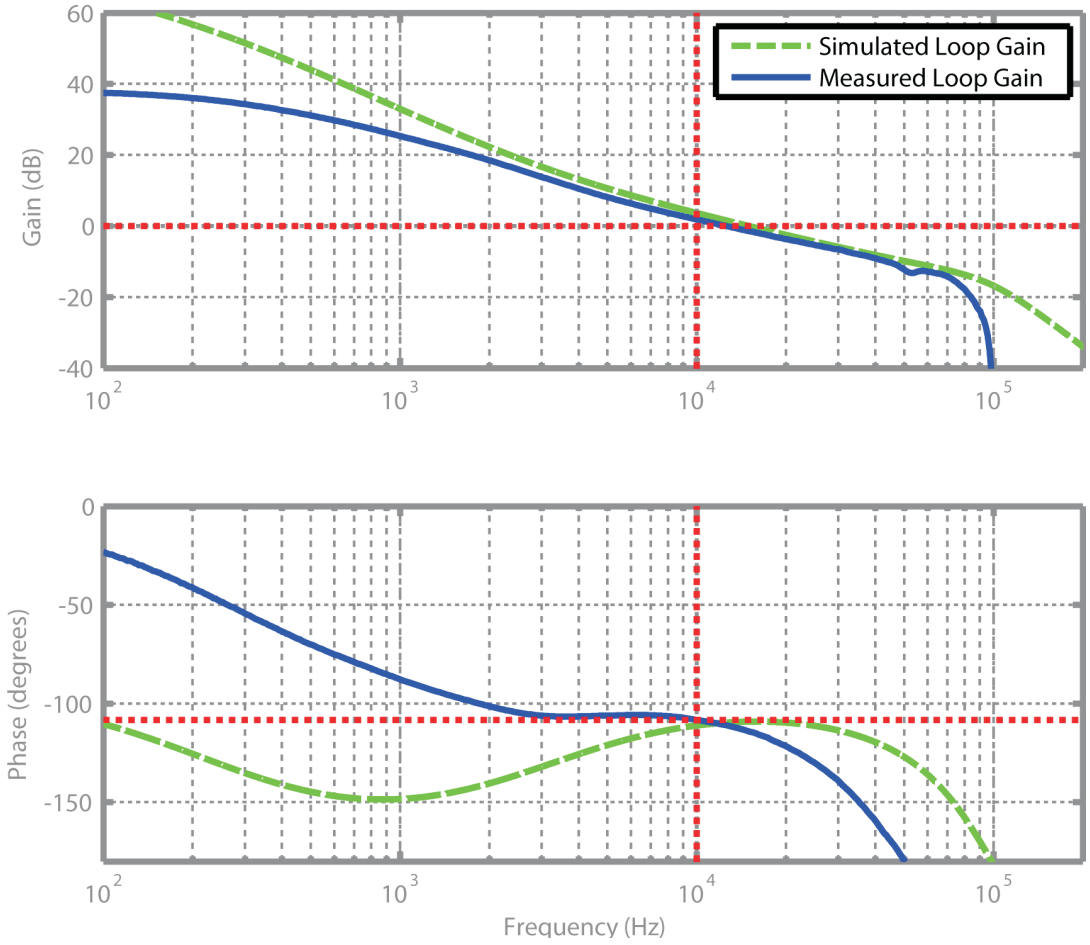

Figure 11. Gain and Phase Plots for Measured and Modeled Small-Signal Frequency Response of the Open Loop System (Measured Using Bode 100 From Omicron Lab)

10 Summary

This application report has explained one possible implementation of a digital power supply. Further applications of digital power are explored from a hardware designer's perspective in the Digital Power Workshops available from Biricha Digital [9].

11 References

[1] Ridley, R.B., A new, continuous-time model for current-mode control [power convertors], Power Electronics, IEEE Transactions on (1991), pp. 271-280.

[2] Biricha Digital Power Ltd. Digital Power Multi-Day Workshop Manual (2010), pp. 168.

[3] Ridley, R.B., A New Small-Signal Model for Current-Mode Control, Virginia Polytechnic Institute and State University, PhD. Thesis (1990).

[4] Cooke, P., Modeling average current mode control [of power convertors] (2000), pp. 256-262 vol.1.

[5] Tang, W. and Lee, F.C. and Ridley, R.B., Small-signal modeling of average current-mode control (1992), pp. 747-755.

[6] Biricha Digital Power Ltd. Digital Power Multi-Day Workshop Manual (2010), pp. 129.

[7] Biricha Digital Power Ltd. Digital Power Multi-Day Workshop Manual (2010), pp. 122.

[8] http://www.biricha.com/resources/download_resource.php?id=21

[9] In-Depth Digital Power Supply Workshop sponsored by Biricha Digital Power Ltd.: http://www.ti.com/biricha

[10] http://www.ti.com/controlsuite

[11] Hallworth, M., Shirsavar, S., Microcontroller Based Peak Current Mode Control Using Digital Slope Compensation, Power Electronics, IEEE Transactions on (2011).

[12] Biricha Digital website, located at http://www.biricha.com/

12 Appendix

This appendix includes the interrupt service routine.

```
/*****************************************************************************
* (c) Copyright 2012 Biricha Digital Power Limited
* FILE          : main.c
* AUTHOR        : Dr C.J.Hossack
* PROJECT       : Piccolo B Exercise3b
* Target System : DSP C280x
* CREATION DATE : 10/03/2012
* COPYRIGHT     : Copyright Biricha Digital Power Limited 2009
*                 All rights reserved. Reproduction in whole or part is
*                 prohibited without written consent of the copyright
*                 owner.
* DESCRIPTION   :
*
* This project demonstrates Peak Current Mode Control of BDP-105 Buck board
* using the Piccolo B and its CLA.
* The main core is being used to run a 2p2z controller for the current mode
* Buck Converter. Piccolo's comparator 2 is being used to detect when the peak
* current reaches its demand value.
*
* Piccolo B's CLA is being used to create the negative slope ramp needed slope
* compensation.
*
* Phase and gain margins of the digital PSU were then measured using a
* frequency response analyser:
*
*     phase margin         = 42 degrees
*     gain  margin         = 15 db
*     cross over frequency = 15 kHz
*     switching frequency  = 200kHz
*
* IMPORTANT: BDP-105 should be connected to Port 1 of the daughter card. switch
* current IL needs to be connected to the non-inverting pin of Piccolo's
* comparator. This connection is not implemented on the daughter card.
* Therefore a link wire needs to be connected from IL pin of the daughter card
* (i.e., ADC pin B0) to the Comparator 2's non-inverting pin (i.e., ADC Pin A4)
* Please ask an instructor for a link wire and make this connection.
*
* LINKS
* file:///C:/tidcs/c28/CSL_C280x/v100/doc/CSL_C280x.pdf
*****************************************************************************/
/***************************** INCLUDES SECTION ****************************/

#include "csl.h"


/************************** DECLARATIONS SECTION *************************/


/* These set up the coefficients for our 2p2z controller for BDP-105 Buck
*  Converter with a 200 kHz switching frequency and a cross over of 15kHz
*/

#define K   (0.5)
#define REF (_IQ15toF(2048))
#define MIN_DUTY 0
#define MAX_DUTY 1023
#define A1  +1.69020338
#define A2  -0.69020338
#define B0  +3.22868006
#define B1  +0.29060216
#define B2  -2.93807791
```

```
#define PERIOD_NS  5000 /*Our period in ns for fs = 200kHz */

/************************* POST DECLARATIONS SECTION ***********************/

/* Data align memory before instantiating a 2p2z controller. */
#pragma DATA_ALIGN ( MyCntrl , 64 );


/* This effectively declares a 2p2z controller called MyCntrl */
CNTRL_2p2zData MyCntrl;


/* This macro generates CLA assembly code called SlopeTask, which implements
* slope compensation by subtracting a slope, of user defined gradient, from
* the demand value of the current before it is fed to the comparator.
*
* SlopeTask -> name of the CLA task
*
* 2 -> use comparator Mod2's DAC value
*
* 1 -> Clear PWM1's interrupt flag.
*      we specified in main() that PWM1's period
*      interrupt would trigger this CLA task
*
* -1.0 -> decrement the initial value on the DAC
*         by 1 every iteration. The initial value
*         was set by the 2p2z controller before
*         slope compensation.
*
* 80 ->  total number of decrements during one
*        sampling period
*
* Each decrement takes 50ns. Therefore 80 decrements will take 4us. This will
* give us a 1 us safety margin before the next switching interval.
*/
CLA_slopeCode( SlopeTask, 2,1, -1.0, 80 );

/***************************** FUNCTIONS SECTION ***************************/

/*****************************************************************************
* FUNCTION      : IsrAdc
* DESCRIPTION   :
* This interrupt is called when the ADC sequencer has finished sampling.
*****************************************************************************/
interrupt void IsrAdc( void )
{

    /* Sets GPIO pin 12 tied to TZ test pin on hardware */
    GPIO_set( GPIO_12);


    /* Ack group and ADC SEQ interrupt. Re-enable the ADC interrupts -Int1 */
    ADC_ackInt( ADC_INT_1 );


    /* These three lines read the ADC, call the 2p2z control loop and then update
    *  the duty cycle respectively.
    */
    MyCntrl.Fdbk.m_Int = ADC_getValue(ADC_MOD_1);
    CNTRL_2p2z(andMyCntrl);


    /* This inputs the "initial" value of the demand current (from the 2p2z)
    * controller to the DAC of the comparator. i.e., the demand current before
    * slope compensation is fed to the inverting pin of the on board
    * comparator 2. This initial DAC value will later get updated by
```

```
    * the CLA's slope compensation algorithm
    */
    CMP_setDac( CMP_MOD_2, MyCntrl.Out.m_Int );


    /* Clears GPIO12 pin */
    GPIO_clr( GPIO_12);


    /* Sets up soft-start*/
    CNTRL_2p2zSoftStartUpdate(andMyCntrl);
}


/*****************************************************************************
* FUNCTION      : main
* DESCRIPTION   :
*
*****************************************************************************/
void main( void )
{

    /* Initialize the MCU, ADC and GPIO12 */
    SYS_init();
    ADC_init();
    GPIO_config( GPIO_12, GPIO_DIR_OUT, false );


    /* Configures the CLA Mod1 to run CLA code "SlopeTask" whenever PWM trigger
    * occurs - The PWM event that causes the trigger is defined later.
    */
    CLA_config( CLA_MOD_1, andSlopeTask, CLA_INT_PWM );


    /* Setup PWM Mod1 for fs = 200 kHz. PWM1 Ch A is being used for switching
    * the converter. PWM1 Ch B is being used for timing purposes - more on this
    * later.
    */
    PWM_config( PWM_MOD_1, PWM_nsToTicks(PERIOD_NS), PWM_COUNT_UP );
    PWM_pin( PWM_MOD_1, PWM_CH_A, GPIO_NON_INVERT );
    PWM_pin( PWM_MOD_1, PWM_CH_B, GPIO_NON_INVERT );


    /* Typically for digital current mode we set the PWM Ch A duty 100%; then
    * use the cycle by cycle trip function to pull the PWM pin low when the
    * current reaches our demand peak value. However for safty we have set the
    * maximum duty to 60%, i.e., if your control algorithm fails, the PWM will
    * reset after 60% rather than staying at 100%.
    */
    PWM_setDutyA(PWM_MOD_1, PWM_nsToTicks(PERIOD_NS)*0.6 );



/* PWM1 Channel A is being used for the PWM drive of the MOSFET. Hence, the
    * sampling, conversion, ADC interrupt entry, 2p2z, scaling and then DAC and
    * comparator set up must happen just before PWM1 Ch A goes high. For this
    * reason we use PWM1 Channel B to start the sampling process. The falling
    * edge of PWM1 Ch B is used to start the sampling process followed by all
    * relevant calculations. Therefore the duty of Ch B should be set such that
    * all sampling and calculations are completed just as PWM1 Ch A goes high.
    * This time has been measured on the scope as 2.45us.
```

```
                                        <---PERIOD_NS-->
                    *                    ___            ___
                    *        PWM A: ____|   |__________|   |__________|
                    *
                    *                             PERIOD_NS-2450
                    *                           _______<-----> _______
                    *        PWM B: ____________|       |______|       |______|
*

*   PWM B triggers ADC SOC here^      ^ here PWM B starts CLA slope
    *   This falling edge needs to be       compensation function for the next
    *   adjusted such that all              cycle
    *   calculations are completed
    *   before the next cycle
    *   (i.e., before the next rising edge)
    */



    /* This function sets the PWM1 Ch B such that the calculations are complete
    * just before the rising edge PWM A. PERIOD_NS is our period and set to
    * 5000 ns. Therefore we are setting our pulse width to (5000 - 2450) ns
    */
    PWM_setDutyB(PWM_MOD_1, PWM_nsToTicks( PERIOD_NS-2450+0 ) );



    /* This sets up the PWM Mod1 to start the ADC conversion whenever PWM1
    * Channel B timebase counter matches Ch B's duty. i.e., falling edge of PWM
    * Ch B triggers ADC SoC, as discussed above.
    */
    PWM_setAdcSoc( PWM_MOD_1, PWM_CH_B, PWM_INT_CMPB_UP );



    /* This sets up PWM Mod1 to generate an interrupt every PWM cycle whenever
    * timebase counter = 0. The "0" instead of an ISR function name means that
    * an interrupt is generated but no jump to an ISR function is carried out.
    * The CLA will detect this interrupt and run the CLA code instead. Finally
    * PWM_INT_PRD_1 indicates that an interrupt should be generated every cycle
    * as opposed to every other cycle
    */
    PWM_setCallback(PWM_MOD_1, 0, PWM_INT_ZERO, PWM_INT_PRD_1 );



    /*-------------------------------------------------------------**
    **     The next 5 functions set up the one-shot trip zone from   **
    **     comparator output and the leading edge blanking             **
    **-------------------------------------------------------------*/

    /* This effectively feeds the output of the comparator Mod2 into PWM Mod1
    * and activates the blanking by setting the digital compare event
    * PWM_DCEVT at the correct time. We will use PWM_DCEVT later to trip our
    * PWM.
    * The input to the blanking block is not inverted and "true" ensures that
    * the output is not synchronized with the PWM's time-base clock.
    */
    PWM_configBlanking( PWM_MOD_1, PWM_CMP_COMP2, GPIO_NON_INVERT, true );


    /* Sets the size of the blanking window to 420ns */
    PWM_setBlankingWindow( PWM_MOD_1, PWM_nsToTicks(420) );
```

```
        /* Sets up the relevant trip zones: i.e., when PWM_DCEVT occurs, clear
        * PWM1 Ch A on a cycle by cycle basis but take no action on PWM1 Ch B
        * PWM_DCEVT was set up in PWM_configBlanking().
        */
        PWM_setTripZone(  PWM_MOD_1, PWM_DCEVT, PWM_TPZ_CYCLE_BY_CYCLE );
        PWM_setTripState( PWM_MOD_1, PWM_CH_A, GPIO_CLR );
        PWM_setTripState( PWM_MOD_1, PWM_CH_B, GPIO_NO_ACTION );


        /* Configures ADC to sample Vo when triggered by PWM1 Ch B's falling edge
         */
        ADC_config( ADC_MOD_1, ADC_SH_WIDTH_7, ADC_CH_B2, ADC_TRIG_EPWM1_SOCB );

        /* When conversion is finished, cause interrupt and jump to IsrAdc */
        ADC_setCallback( ADC_MOD_1, IsrAdc, ADC_INT_1 );


        /* Initalise the 2p2z control structure */
        CNTRL_2p2zInit(andMyCntrl
        ,_IQ15(REF)
        ,_IQ26(A1),_IQ26(A2)
        ,_IQ26(B0),_IQ26(B1),_IQ26(B2)
        ,_IQ23(K),MIN_DUTY,MAX_DUTY
        );


        /* Configures the comparator Mod2 with 0 qualification window
        * i.e.. asynchronous. The comparator output is not inverted and the inverting
        * input of the comparator is tied to the on board DAC.
        */
        CMP_config( CMP_MOD_2, CMP_ASYNC, GPIO_NON_INVERT, CMP_DAC );


        /* Set up a 500ms soft-start */
        CNTRL_2p2zSoftStartConfig(andMyCntrl, 500, PERIOD_NS );

        /* Enables global interrupts and waits in idle loop */
        INT_enableGlobal(true);

        while(1)
        {

        }
    }
```

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265