Buck current mode with the B-G474E-DPOW1 Discovery kit

Introduction

The B-G474E-DPOW1 Discovery kit is a complete digital power starter kit controlled by the STMicroelectronics Arm® Cortex®-M4 core-based STM32G474RET6 microcontroller. The kit showcases the features of digital power including LED dimming, buck-boost with variable load, Power Delivery (USB Type-C®), and audio class-D amplification.

This application note focuses on the buck converter onboard this Discovery kit and teaches the principles of peak-current-mode control, how to design a compensator to stabilize and regulate the peak-current-mode controlled buck converter and how to implement this onboard the STM32 microcontroller.

This application note also presents the buck current mode usage with the X-CUBE-DPOWER STM32Cube Expansion Package.

Figure 1. B-G474E-DPOW1 top view

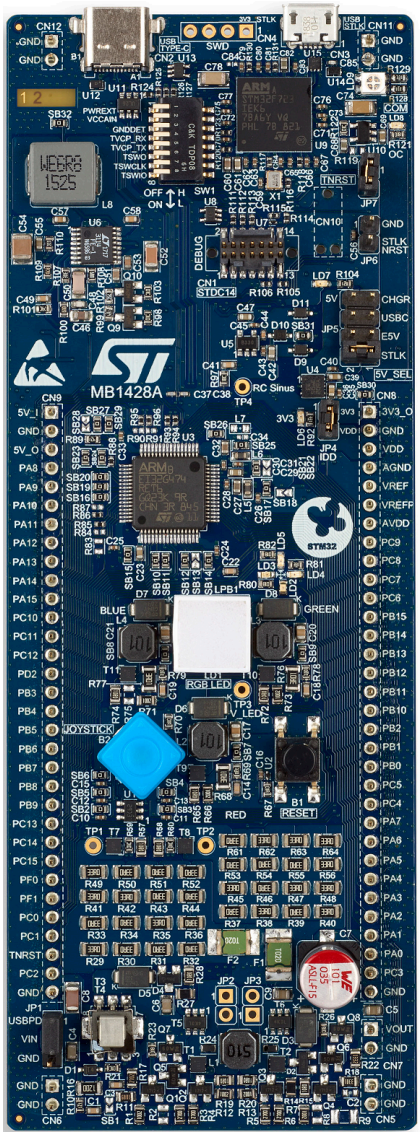
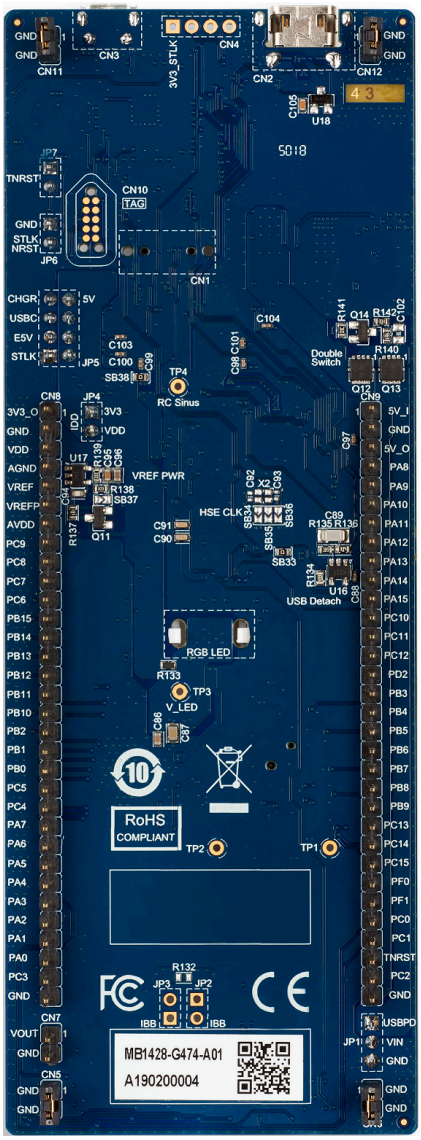


Figure 2. B-G474E-DPOW1 bottom view



Pictures are not contractual.

1 General information

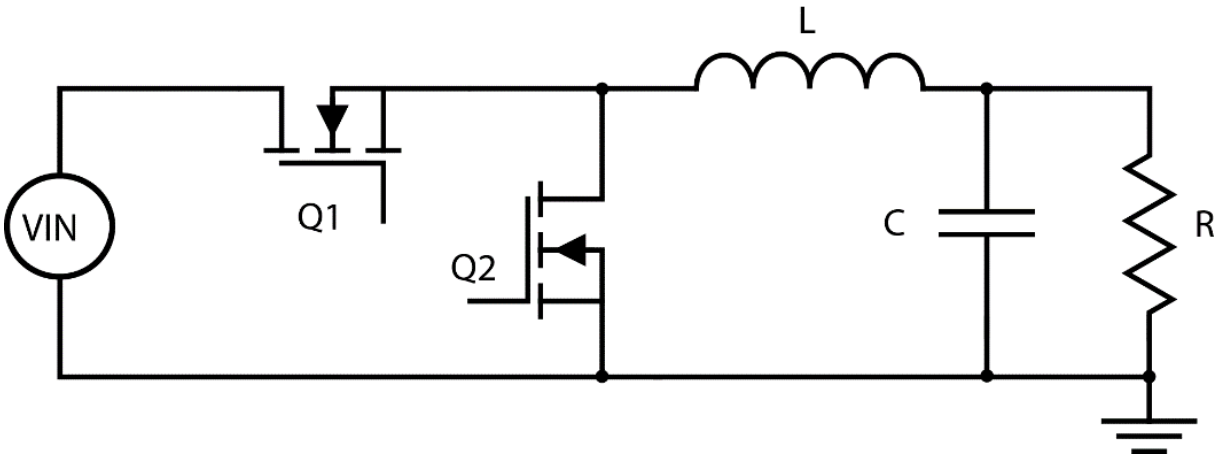
The B-G474E-DPOW1 Discovery kit runs with the STMicroelectronics Arm® Cortex®-M4 core-based STM32G474RET6 microcontroller.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2 Buck converter operation

The B-G474E-DPOW1 Discovery kit contains a synchronous buck converter power stage. The simplified schematic of the power stage for a synchronous buck converter is shown in Figure 3.

Figure 3. Simplified power stage schematic

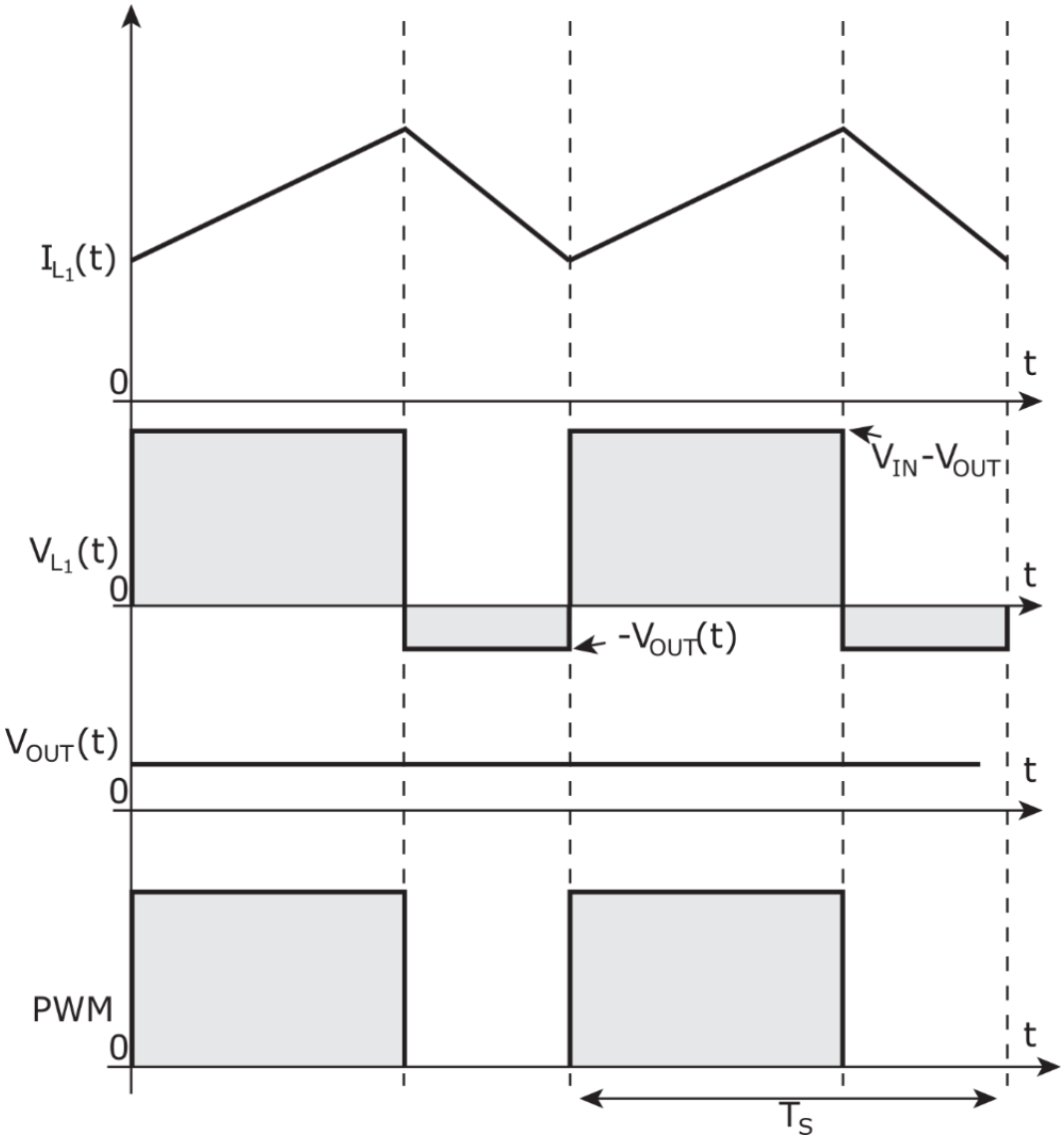


2.1 Principle of operation

The operation of the synchronous buck is as follows. At the beginning of the switching period, the PWM of the top switch (Q1) is set to HIGH and the bottom switch (Q2) is set to LOW. This turns on MOSFET Q1 and turns off MOSFET Q2. With the Q1 switch conducting, the current through the inductor L1 begins increasing linearly. At the end of the high-side duty cycle, switch Q1 is turned off.

A dead-time is inserted between the high-side and low-side PWM for switches Q1 and Q2 to prevent shoot-through, where both switches are partially on at the same time causing a large current to flow through Q1 and Q2 and can damage the MOSFETs. When this dead time has elapsed the low-side PWM for the switch Q2 goes HIGH which turns on the switch Q2. At this time the inductor acts to continue the flow of current and the current now flows through switch Q2. The current through the inductor begins decreasing linearly. This switching action is described in the buck converter waveforms of Figure 4.

Figure 4. **Buck converter operational waveforms**



The output filter capacitor Cout filters the AC component of this current while the DC component of this current is the output load current, Iout. As this is a step-down converter, the output voltage is always less than or equal to the input voltage. In continuous conduction mode, the steady-state duty cycle of the high-side switch Q1 can be calculated in (1).

$$D = \frac{V_{out}}{V_{in}} \quad (1)$$

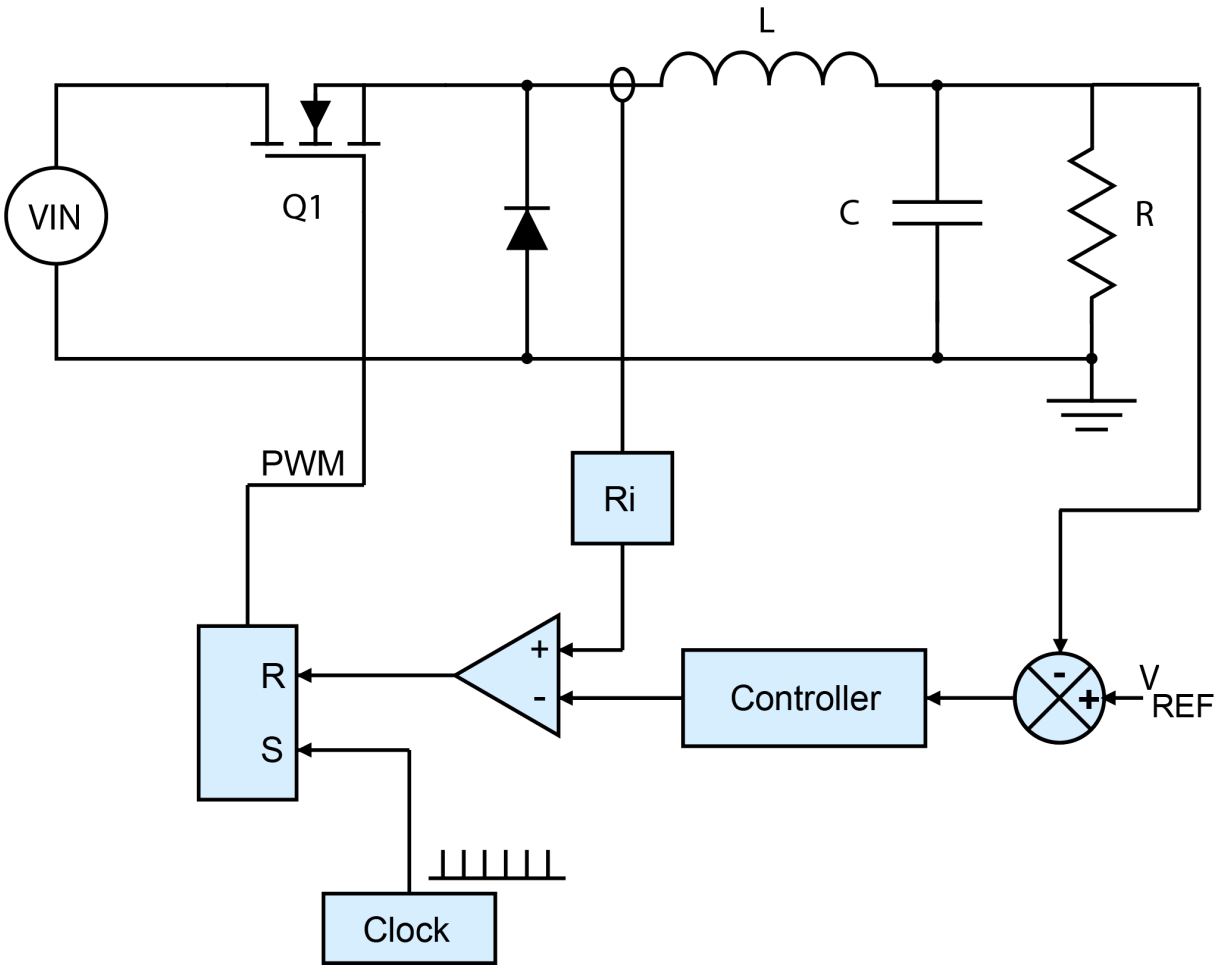
There are two main control methods for the buck converter. These are voltage mode control and peak current mode control. The software example preloaded onto the starter kit provides an example of a well-tuned digital peak-current-mode controlled buck converter.

3 Peak-current mode control explained

3.1 Peak current mode step by step

Peak-current-mode control is one of the most popular control methods for both point-of-load PSUs and offline PSUs due to its inherent current-limiting capability. Under peak current mode control, the peak of the output filter inductor current is controlled to a setpoint value. This setpoint value is determined by the output of the controller and varies on a cycle-by-cycle basis. The design of this controller is the main discussion point of this application note. However, first, the operation of a peak current mode control (PCMC) converter is explained.

Figure 5. Peak-current mode controller buck converter

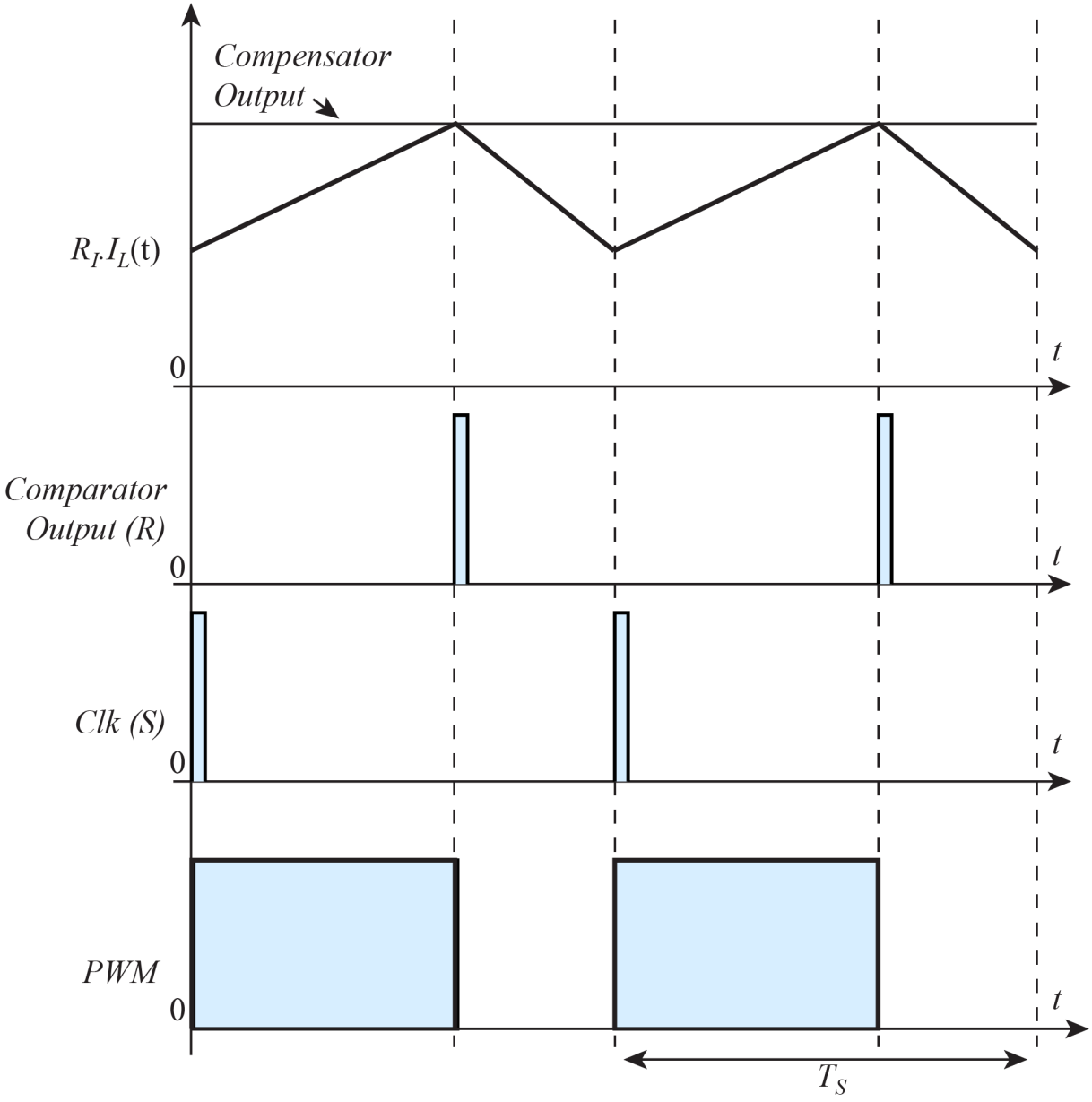


The buck converter power stage shown in Figure 5 remains the same as that discussed in the previous section. However, a current sense transducer is added to the current path of the inductor. Initially, the high-side MOSFET is turned on with no pre-determined duty cycle by the clock pulse setting the set-reset latch. The sensed inductor current is then compared with the demand peak current using a peak current comparator. When the sensed inductor current intersects with the demand peak current, the current sense comparator resets the set-reset latch, which turns off the high-side MOSFET.

The demand peak current value is determined by the output of the compensator. Under PCMC, the compensator compares the actual output voltage with the demand output voltage, Vref, and calculates the error. The error term is used as an input to the compensator, and the compensation network around the compensator determines the demand peak current value.

Therefore, the duty cycle is not set by the controller; it is determined by the rise of the inductor current during the on-time. This is shown in the peak current mode waveforms of Figure 6. Hence the modulation of the duty cycle differs significantly from that of other control schemes such as voltage-mode control, and as such, the design of the compensator differs also.

Figure 6. Peak-current mode control duty cycle modulation



3.2 Advantages and disadvantages

Peak current-mode control offers several advantages over other commonly used control methods, such as voltage-mode control. The peak of the inductor current is controlled on a cycle-by-cycle basis, and therefore this control method has inherent current limiting which can protect against short circuit or overload conditions without the need for an additional control loop.

This is particularly advantageous for topologies that contain a power stage transformer. In some of these topologies, a DC-blocking capacitor is required to prevent flux creepage, which may lead to saturation of the transformer core and catastrophic failure. PCMC limits the current by turning off the switch when the demand peak current value is reached during each switching cycle and thus prevents the core from entering saturation.

Furthermore, PCMC allows multiple power stages to be paralleled and the overall output current to be shared equally between the power stages. This is achieved by controlling all individual power stages using one outer control loop. Multi-phase converters often use this control method to achieve the highest performance and smallest size.

PCMC is commonly used to control power stages that contain a right-half plane zero in the plant transfer function, such as CCM flyback and boost converters. As discussed later, the PCMC control method results in a single plant pole at lower frequencies. This removes the crossover frequency restriction around the double pole found in voltage-mode controlled converters and therefore allows for easier stabilizing of converters with a right-half plane zero.

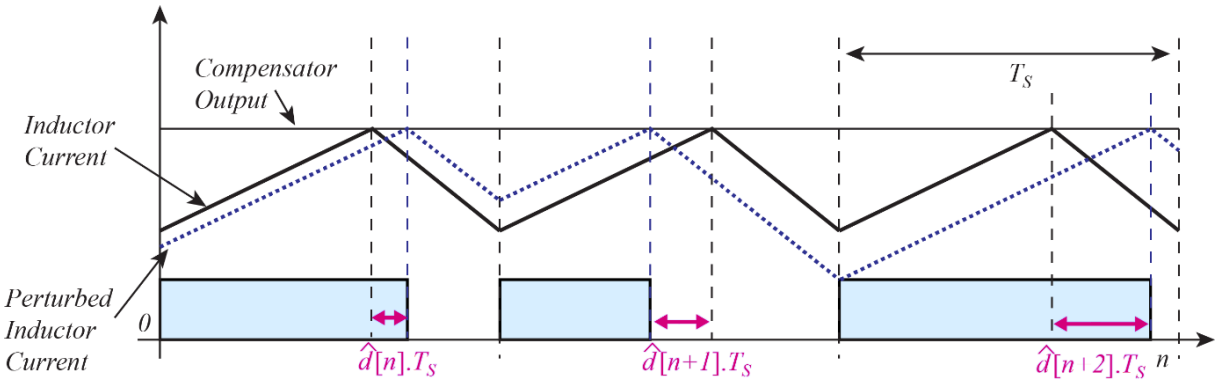
As a measure of the instantaneous inductor current compared to the demand peak current value, the PCMC converter can respond very quickly to changes in the line voltage. For example, if the line voltage decreases, the rate at which the inductor current rises may also decrease. The result may be that the on-time switch is longer to achieve the same demand peak current value. This is unlike voltage mode control, which has a much slower response to changes in the line.

However, PCMC necessitates the use of a current sense transducer, which can add circuit complexity, cost, and additional losses. The current sense signal can also be difficult to route as it is sensitive to noise pickup and may require additional filtering. Generally, the PCMC control scheme is sensitive to noise and poor PCB layout that can result in instability and oscillations of the closed-loop system.

3.3 Sub-harmonic oscillations

PCMC suffers from an inherent instability called sub-harmonic oscillations that needs to be addressed. Sub-harmonic oscillations appear on the inductor current and thus duty cycle at half the switching frequency. These oscillations are characterized by wildly changing duty cycles, from small to large and vice versa, on an alternating cycle basis.

Figure 7. Perturbation in the inductor current leads to subharmonic oscillations which increase over time

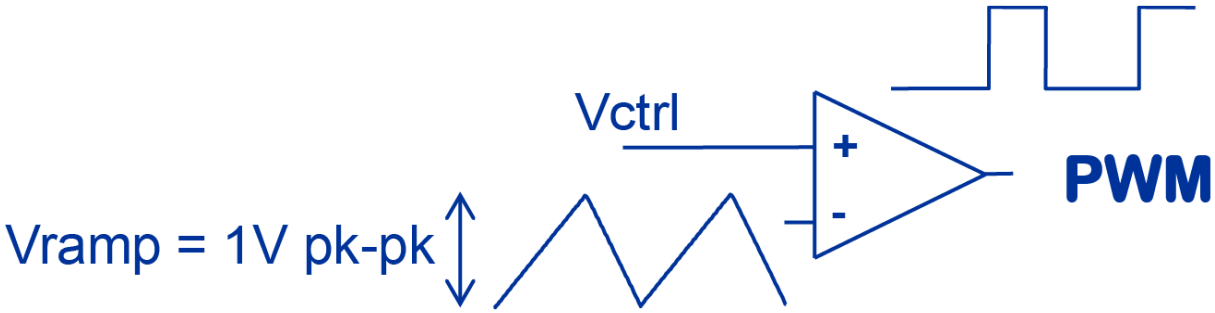


This instability is the result of the sampled nature of the PCMC system – the intersection of the sensed current with the demand current occurs once per cycle. It has been shown by Brown et al. (Brown & Middlebrook, 1981) that this intersection of sensed inductor current and demand current reference is in effect a sample and hold or a zero-order hold. This zero-order hold can be modeled using sampled data modeling techniques (Verghese, 1989). Using this, it can be shown that the zero-order hold results in a pure time delay that can be approximated by a pair of complex conjugate poles at half the switching frequency (Ridley, 1991).

3.4 Peak-current-mode modulator

Under traditional PWM control schemes, such as voltage-mode control, the output of the compensator is compared with a fixed sawtooth ramp. This ramp is usually generated by an RC network and provides duty cycle modulation, as shown in Figure 8.

Figure 8. PWM comparator comparing RC ramp to control voltage in the voltage-mode control



However, under PCMC, there is no fixed RC ramp generating a sawtooth waveform. Instead, the sensed inductor current is used as a ramp, and this is compared directly to the demand peak current value from the output of the compensator using the peak current comparator shown in Figure 9.

Figure 9. Sensed inductor current being compared to demand current when using the peak-current mode control

Subsequently, the analysis of the peak current mode modulator is complex. The full derivation of the s-domain transfer function for this converter is referenced but not included in this application note for brevity.

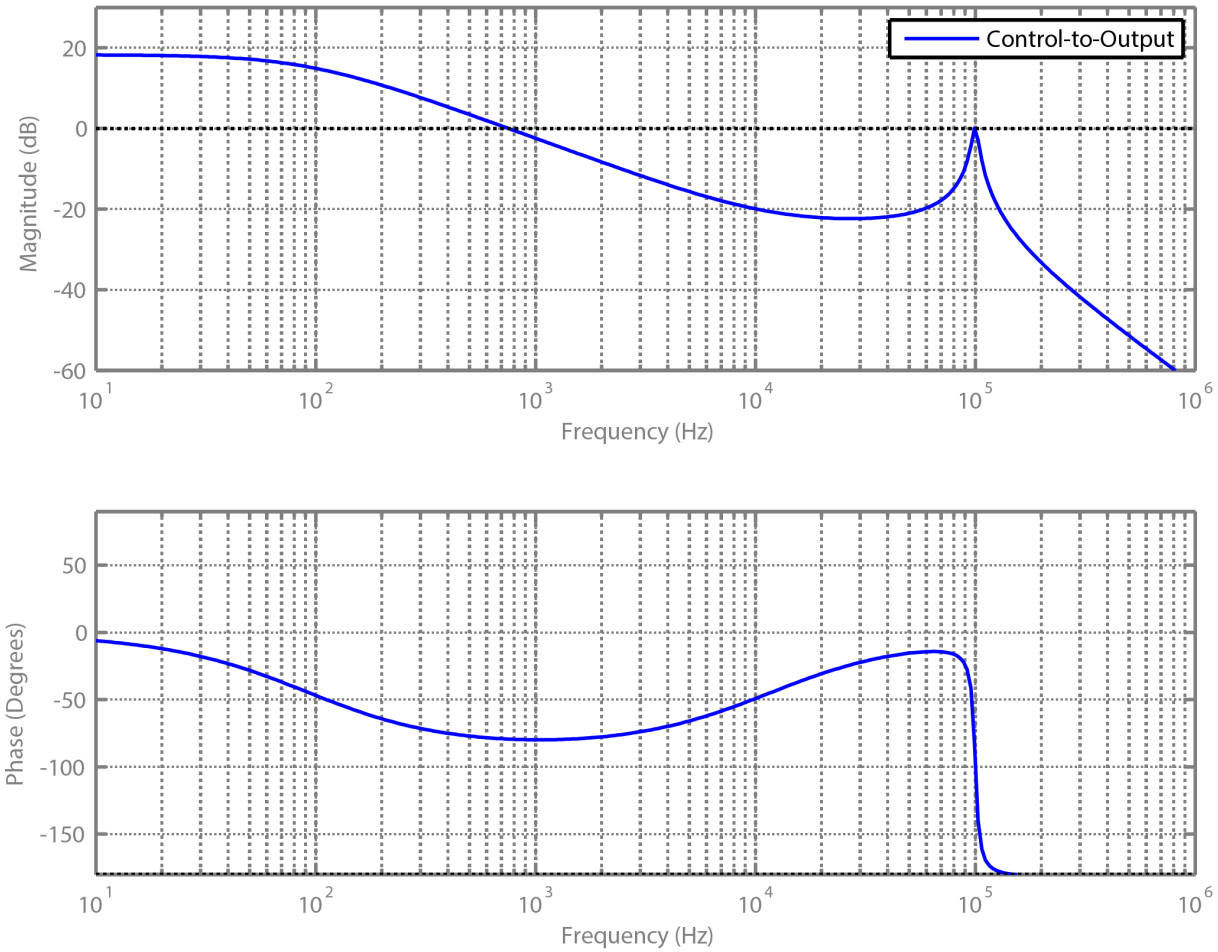
The small-signal analysis aims to describe the behavior and, therefore, the output of the system given small changes in the input. The stability of the system can then be determined using this analysis. The modeled behavior of the system is used, through means of s-domain transfer functions, to characterize the plant and then analytically design a compensator to stabilize the control loop.

3.5 Buck plant transfer function

The derivation of the plant transfer function, also referred to as the control-to-output transfer function, under peak current mode control has been the subject of many different academic papers over the years. However, in this application note, the model derived by Ridley (Ridley, 1991) is used as it compares well with the measured results. The derivation of the transfer function is complex, and only an overview is provided in this application note. Refer to the papers and references in the appendix for further detailed analysis.

Figure 10 shows the typical bode plot of the buck plant transfer function under peak current mode control.

Figure 10. Bode plot of the control-to-output transfer function



The plant transfer function consists of three terms:

- A DC gain, H_{DC}
- A low-frequency power stage transfer function, $H_P(s)$
- A high-frequency term, $H_H(s)$

3.6 DC gain

The DC gain for the peak current mode buck converter is given in (2).

$$H_{DC} = \frac{R_0}{R_1} \cdot \frac{1}{1 + \frac{R_0 T_S}{L_0}(m_c(1-D) - 0.5)} \quad (2)$$

The DC gain is a function of:

- The output load, R_0
- Current sense gain, R_1
- Switching period, T_S
- Output filter inductance, L_0
- Steady-state duty cycle, D
- Slope compensation factor, m_C, which is defined later.

Therefore, the DC gain changes depending on the load and the amount of slope compensation that is applied.

3.7 Power stage transfer function

The power stage output filter consists of an inductor and a capacitor that normally forms a double pole due to the resonance formed between the inductor and capacitor. However, under peak current mode control, the inductor's current is controlled on a cycle-by-cycle basis, and therefore the inductor acts as a constant current source.

This means that the plant power stage no longer contains the double pole present when operating under voltage-mode control. The plant power stage now only contains a single plant pole, as shown in (3).

$$H_P(s) = \frac{1}{1 + \frac{s}{\omega_{P1}}} \quad (3)$$

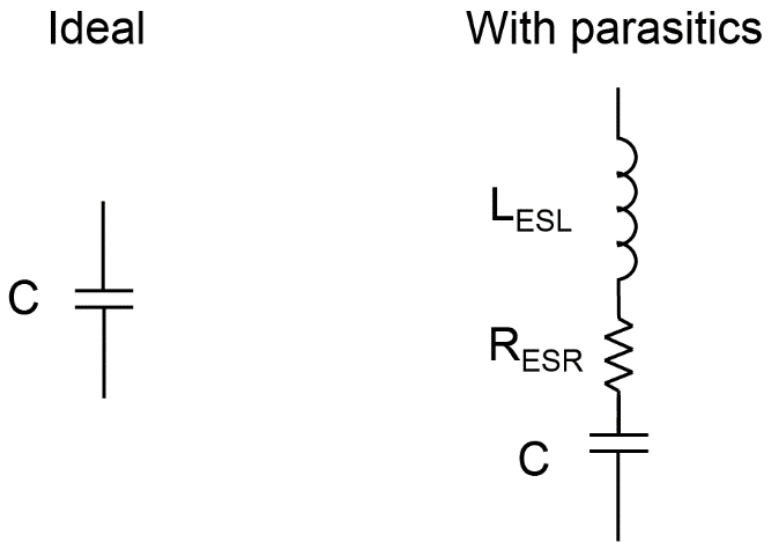
Where ω_{P1} is defined in (4).

$$\omega_{P1} = \frac{1}{R_0 C_0} + \left(\frac{T_S}{L_0 C_0} \cdot (m_C \cdot (1-D) - 0.5)\right) \quad (4)$$

3.8 Capacitor ESR zero

The power stage transfer function given in (3) does not include parasitic elements of the output filter capacitor. The parasitic elements of the capacitor are shown in Figure 11.

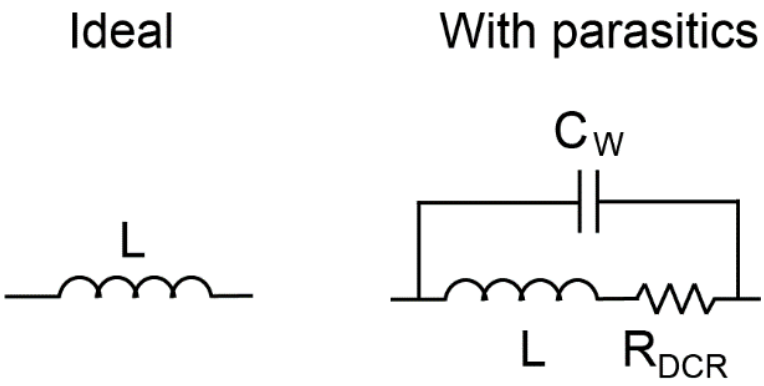
Figure 11. Ideal capacitor and capacitor with parasitic elements



The parasitic equivalent series resistance (ESR) has a significant impact on the plant transfer function for the buck converter. The parasitic equivalent series inductance of the capacitor is usually only dominant at much higher frequencies and therefore it can be ignored in this transfer function.

Likewise, the inductor has parasitic elements as shown in Figure 12. Typically, only the DC resistance (DCR) of the inductor winding is considered as the interwinding capacitance is only an issue at high frequencies which are above that of the control loop.

Figure 12. Ideal inductor and inductor with parasitic elements



The effect of the capacitor ESR is that the zero is formed in the power stage transfer function. The zero is shown in the numerator of the plant transfer function in (5).

$$H_P(s) = \frac{1 + \frac{s}{\omega_{ESR}}}{1 + \frac{s}{\omega_{P1}}} \quad (5)$$

The location of the zero is given in (6) and is dependent on the capacitance and the ESR value.

$$\omega_{ESR} = \frac{1}{C \cdot R_{ESR}} \quad (6)$$

3.9 High-frequency term

As discussed earlier, the sampled nature of the peak current mode modulation scheme results in a zero-order-hold that can be modeled as a pair of complex conjugate poles at half the switching frequency. It is the resonance of these poles that determines the prevalence of any subharmonic oscillations on the inductor current and duty cycle. The high-frequency term is given in (7).

$$H_H(s) = \frac{1}{\frac{s^2}{{\omega_N}^2} + s\frac{1}{Q\omega_N} + 1} \quad (7)$$

Where:

$$\omega_N = \pi F_S \quad (8)$$

$$Q = \frac{1}{\pi(m_C(1-D) - 0.5)} \quad (9)$$

Where m_C is the slope compensation factor, which is a ratio of S_E the external ramp slope to S_N the output inductor current slope during the on-time:

$$m_C = 1 + \frac{S_E}{S_N} \quad (10)$$

To remove the subharmonic oscillations, the damping of the double pole needs to be increased. This has the effect of reducing the Q and, therefore, the resonant peak of the double pole. The Q can be reduced by adding external slope compensation through means of the slope compensation factor m_C.

3.10 Slope compensation

There are various methods for calculating the amount of slope compensation required to remove the subharmonic oscillations. In this application note, the amount of external slope compensation required to reduce the Q of the complex conjugate pair of poles to 1 is calculated. This results in a slightly underdamped pair of poles; however, the resonant peak is greatly reduced compared to a system with no external damping.

$$m_C = \frac{1 + \frac{\pi}{2}Q}{\pi Q(1 - D)}$$

Thus:

$$m_C = \frac{1 + \frac{\pi}{2}}{\pi(1 - D)}$$

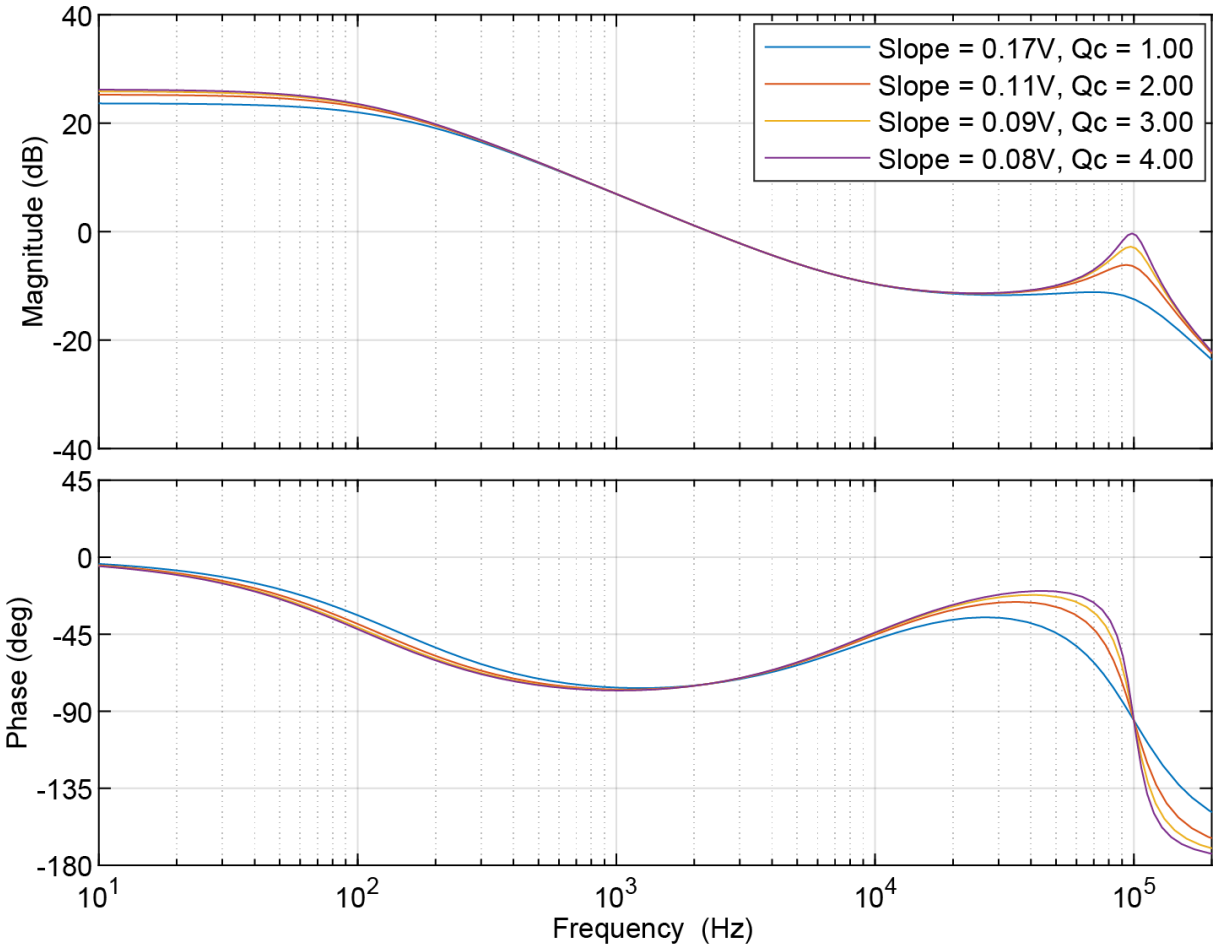
This can be solved for the required external slope compensation ramp height using (10) and the equation for the measured output inductor current slope during the on-time, which for a buck converter is given as:

$$S_N = \frac{(V_{IN} - V_{OUT}) \cdot R_I}{L_0}$$

It is important to ensure that no more damping than required is added to the system as this has the effect of limiting the maximum duty cycle, impairing the transient response, and potentially causing the converter to operate as if under voltage-mode control with the double pole of the LC output filter returning in the transfer function.

Therefore, it is always important to calculate the amount of slope compensation required, ensure that there are no subharmonic oscillations at maximum load and minimum line voltage, and then measure the loop to ensure correct current-mode operation. Figure 13 shows the plant of the power stage with varying amounts of slope compensation applied. The resonant peak of the complex conjugate pair of poles can be seen reducing as the slope compensation and thus damping is increased.

Figure 13. Damping of double poles at half the switching frequency changing with applied slope compensation



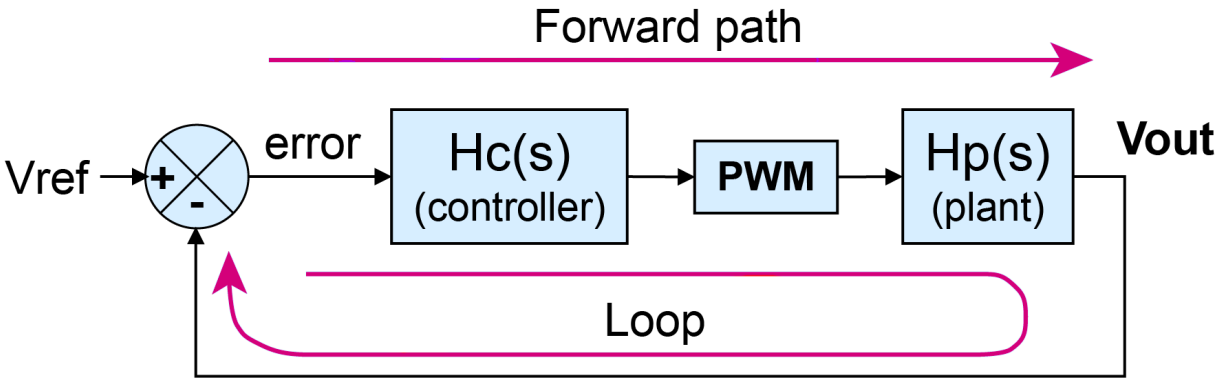
Using (11), the amount of slope compensation to achieve a Q of 1 with the complex conjugate pair of poles at half the switching frequency can be calculated.

$$V_{PP} = (m_C - 1)S_N T_S \quad (11)$$

4 Peak current mode compensator design

4.1 Loop stability criteria

Figure 14. **Closed-loop and open-loop transfer functions**



Consider the control loop of the buck converter shown in Figure 14. In this figure, the forward and loop paths of the control loop are identified. The transfer function for the closed-loop system is given in (12).

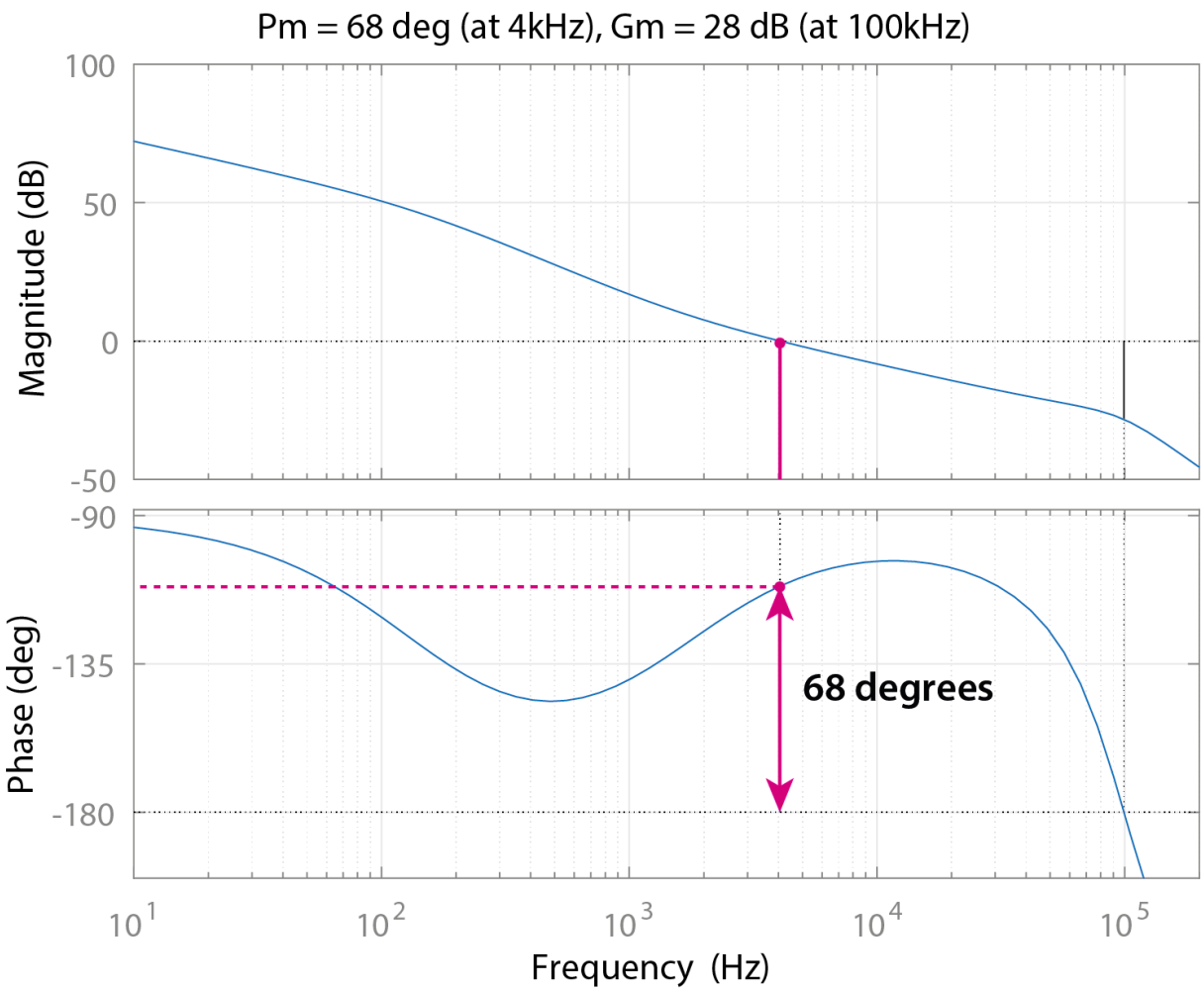
$$TF = \frac{Forward}{1 - Loop} \tag{12}$$

The closed-loop transfer function can be derived by applying this to Figure 14:

$$H_{CL}(s) = \frac{Forward}{1 - (-Loop)} \tag{13}$$

The loop response also called the open-loop, is the compensator transfer function *Hc(s)* combined with the plant or power stage transfer function Hp(s) and also includes the modulator gain – the PWM block. The Bode plot of a typical loop response for a peak current mode converter is shown in Figure 15.

Figure 15. Open-loop Bode plot of buck converter: plant and compensator



Several terms can be defined from this Bode plot. The first is the crossover frequency. This is the frequency at which the gain plot crosses the 0 dB axis. If the gain plot is falling at a rate of 20 dB/decade around the crossover frequency, then below the crossover frequency, which means at a lower frequency going left on the frequency X-axis, the gain plot has a positive gain, meaning that it has a gain greater than 1.

At the crossover frequency, the value of the phase in the open loop determines the stability of the closed-loop system. If the phase is -180° or less with a gain greater than or equal to 1, then the closed-loop system becomes unstable. This can be seen from the denominator of (13).

Therefore, to ensure stability, the phase of the open-loop system must be greater than -180° at the crossover frequency. This term is defined as the phase margin and is the amount by which the phase is above the -180° at the crossover frequency. Typically, the compensator is designed such that the phase margin is 45° or more at the crossover frequency. A phase margin of 45° equates to a loop phase of -135°. Therefore, the loop phase is 45° above the -180° point of instability. The phase margin is shown as 64° in Figure 15.

4.2 Crossover and phase margin specification

The choice of crossover frequency and phase margin determines how well the converter responds to line and load transients. Typically, the higher the crossover frequency is, the faster the response and recovery in the time domain. However, certain limitations prevent the choice of a crossover frequency that is too high. For example, in flyback and boost converters, there is a right-hand plane zero in the plant transfer function. It is typical to cross one decade below this such that it has no effect on the phase margin at the crossover frequency. Furthermore, the op-amp internal to analog control ICs introduces a phase roll-off as the frequency approaches the bandwidth.

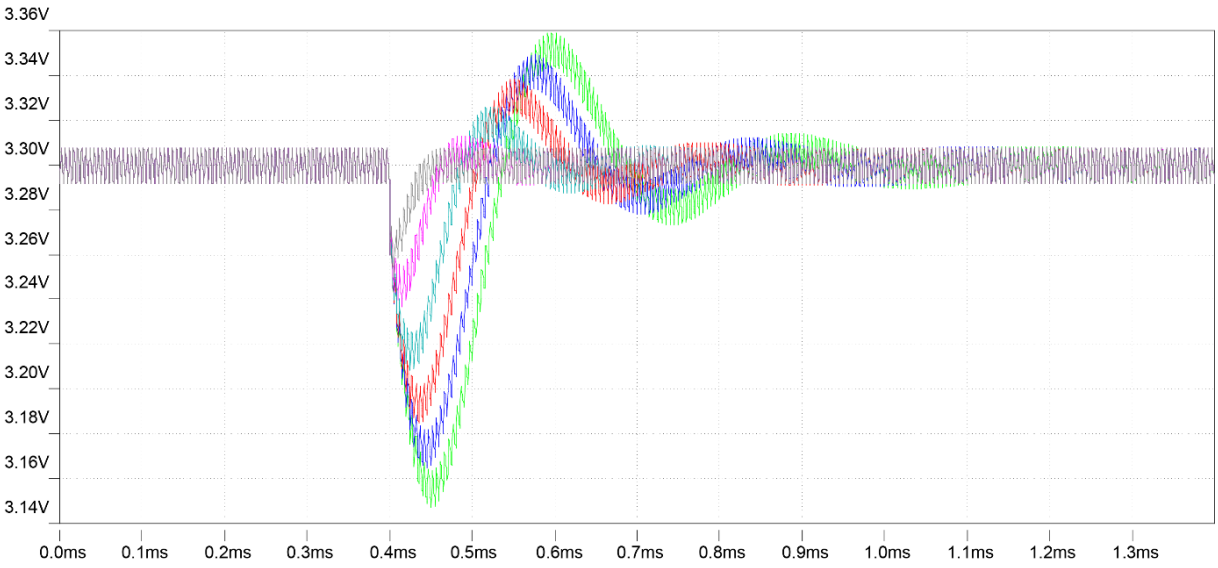
With a digital compensator, there is no analog op-amp bandwidth to consider. However, the delay within the digital system introduces a phase loss. As the frequencies approach the sampling frequency this phase loss becomes significant. A method for calculating the anticipated phase loss is covered later in this application note. Therefore, a recommended starting point for the crossover frequency is between 1/10th and 1/20th of the sampling frequency. This is assuming that the sampling frequency is the same as the switching frequency.

Crossover specification: 1/10th to 1/20th of the sampling/switching frequency

As discussed earlier, the phase margin is an indicator of the stability of the system. With 0° of phase margin, the system becomes unstable. For <30° of phase margin, the system likely has multiple oscillations in the time domain when subjected to the line and load transients. For 45° of phase margin, the system likely has one ring in the time domain after recovery from a transient. Therefore, 45° of phase margin is typically the minimum allowable.

Conversely, a larger phase margin may introduce result in a slower response to load transients. For example, a phase margin of 60° may result in zero overshoot, no ringing, and a slower recovery. Therefore, there must be a balance between the choice of crossover frequency and the amount of phase margin that the combined system has. In Figure 16 the step response for a system with decreasing crossover frequency and phase margin is shown (from grey to green). The response becomes slower and more oscillatory as the crossover and phase margin are both decreased.

Figure 16. Step response for a system with decreasing crossover and phase margin (grey to green)



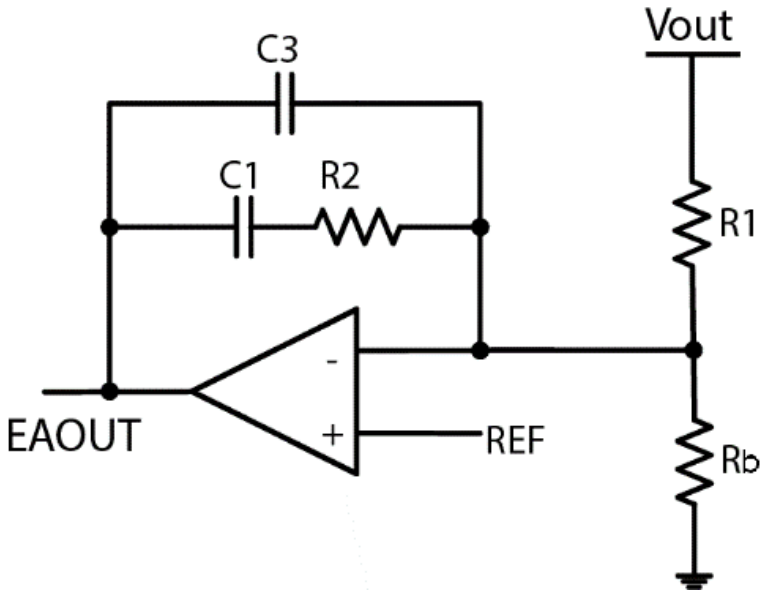
Phase margin specification: a minimum of 45° of phase margin at the crossover frequency, ideally between 50° and 60°

4.3 Types of compensators

A compensator needs to be designed and added into the loop to shape the loop response to meet the desired crossover frequency and phase margin specifications. In the analog domain, a compensator usually consists of an inverting op-amp with a compensation network of capacitors and resistors around the negative feedback path. The combination of capacitors and resistors determines the location of the poles and zeros of the compensator.

There are typically two types of compensators that are used for stabilizing power supplies. These are universally referred to as the Type-II and Type-III compensators. For peak current mode control, the Type-II compensator is typically used. The Type-III is usually used for compensation of voltage-mode controlled converters as a Type-III has an extra pole/zero pair required to compensate for the double pole of the voltage-mode plant transfer function. The circuit for a Type-II compensator is shown in Figure 17.

Figure 17. Type-II compensator implemented in analog using op-amp



The transfer function for this circuit can be derived using the standard transfer function for an inverting op-amp. For brevity, the full transfer function has been included without derivation in (14).

$$H_C(s) = \left(\frac{\omega_{CP0}}{s}\right)\frac{\left(\frac{s}{\omega_{CZ1}}+1\right)}{\left(\frac{s}{\omega_{CP1}}+1\right)} \quad (14)$$

The transfer function in (14) has one zero, one pole, and a pole at the origin. The locations of these zeros and poles are determined by the capacitors and resistors in the compensation network according to (15) through (17).

$$\omega_{CZ1} = \frac{1}{R_2 C_1} \quad (15)$$

$$\omega_{CP0} = \frac{1}{R_1(C_1 + C_3)} \quad (16)$$

$$\omega_{CP1} = \frac{(C_1 + C_3)}{R_2 C_1 C_3} \quad (17)$$

In digital, this compensator is implemented on the MCU and the translation from analog to digital is discussed in the next section.

4.4 Compensator pole/zero placement

The locations of the poles and zeros of the compensator need to be selected such that, when the compensator is combined with the plant power stage, the open-loop frequency response meets the stability criteria of:

- Desired crossover frequency
- Desired phase margin at the crossover
- A shallow slope of -20 dB/decade around the crossover

Several different methods can be applied to achieve this. In this application note, the straightforward and intuitive method of pole/zero cancellation is applied. Consider the plant transfer function discussed earlier:

- The plant has a DC gain.
- The plant has a single low-frequency plant pole.
- The plant has a single zero due to the ESR of the output filter capacitor.
- The plant has a complex pair of poles at half the switching frequency.

Firstly, one of the poles of the compensator can be used to cancel out the parasitic ESR zero in the power stage. This effectively eliminates the gain and phase contribution from the parasitic ESR zero. However, this is reliant on the accurate determination of the actual ESR value for the capacitor or capacitors used.

$$\omega_{CP1} = \omega_{ESR} = \frac{1}{C \cdot R_{ESR}} \tag{18}$$

The compensator zero can be placed at one-fifth of the crossover frequency, and in doing so there is some phase boost around the crossover frequency that helps to increase the phase margin.

$$\omega_{CZ1} = \frac{2\pi F_X}{5} \tag{19}$$

The final term of the compensator to calculate is the pole at the origin. As the name implies, this compensator pole is at the origin, and therefore setting the position of the pole at the origin is changing the gain and not moving the pole. This can be used to adjust the crossover frequency of the loop. Adding the pole at the origin to the compensator introduces the constant -20dB/decade gain roll-off, which is desirable around the crossover frequency. The pole at the origin also provides very high low-frequency gain, which removes steady-state errors and rejects low-frequency perturbations of the control loop.

The gain contributions from all of the other poles and zeros in the system must be taken into account to determine the exact gain required by the compensator to achieve the desired crossover frequency. The equality given in (20) must be solved for the compensator pole at the origin:

$$20\log_{10}\left|H_P(j\omega)\right| + 20\log_{10}\left|H_C(j\omega)\right| = 0dB \tag{20}$$

Given that the plant transfer function and compensator transfer functions are both known, this can be solved for the unknown term ω_{CP0} with the result shown in (21). Evaluating this equation determines the amount of gain which needs to be added by the compensator to achieve the specified crossover frequency.

$$\omega_{CP0} = \frac{\omega_X}{\frac{V_{IN}}{V_{RAMP}} \times \frac{\sqrt{1+\left(\frac{\omega_X}{\omega_{ESR}}\right)^2}}{\sqrt{\left(\frac{\omega_X}{\omega_{LC}\times Q}\right)^2 + \left(1+\frac{-\omega_X{}^2}{\omega_{LC}{}^2}\right)^2}} \times \frac{\sqrt{1+\left(\frac{\omega_X}{\omega_{CZ1}}\right)^2}}{\sqrt{1+\left(\frac{\omega_X}{\omega_{CP1}}\right)^2}}} \tag{21}$$

The method discussed so far gives the user control over the crossover frequency but not the phase margin. Typically, it may result in a large phase margin, however, in a digital system, the phase loss due to the digitization delays may result in a significant amount of phase margin erosion. Therefore, it is possible to derive an equation that analytically calculates the precise location of the compensator zero to achieve the specified phase margin. This is achieved by solving the equality shown in (22) for the compensator zero ω_{CZ1}.

$$\angle\left(H_P(j\omega).H_C(j\omega)\right) = -\pi + \theta_M \tag{22}$$

(22) states that the phase of the plant combined with the compensator must be equal to -180° plus the desired phase margin at the crossover frequency. Again, with some trigonometry, this equality can be solved for ω_{CZ1} and the result is given in (23).

$$\omega_{CZ1} = \frac{\omega_X}{\tan\left(-\frac{\pi}{2} + \Phi_M - tan^{-1}\frac{\omega_X}{\omega_{PP1}} - tan^{-1}\frac{\omega_X}{\omega_{PP2}} + tan^{-1}\frac{\omega_X}{\omega_{CP1}}\right)} \tag{23}$$

Where ω_{PP1} and ω_{PP2} are the complex conjugate poles of the high-frequency double pole due to the sampling effect of the PCMC plant.

This equation requires the calculation of the inverse tangent of complex numbers necessitating the use of a mathematical package to evaluate. For this application note, the software tool Biricha ST-WDS is used to perform the calculations and is available for download free of charge from the Biricha website *www.biricha.com/st-wds*. This tool is discussed in detail through means of a complete design example later in this application note.

5 Discrete-time controller

5.1 Bi-linear transform

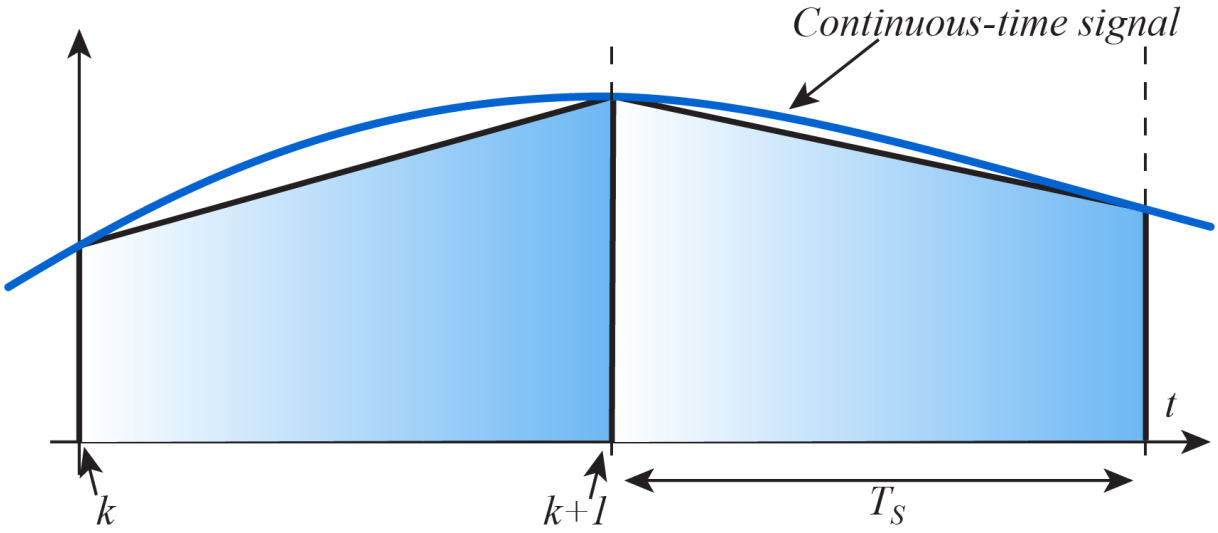
With the poles and zeros of the compensator placed in the continuous-time domain, the s-domain, they must be converted into the discrete-time domain to implement the controller on the MCU. To implement the discrete-time controller, the continuous-time Type II compensator, which is discussed in the previous section, is converted into its discrete-time equivalent.

There is a direct mapping between the continuous-time domain, the s-domain, and the discrete-time domain, the z-domain. The relationship is shown in (24).

$$z = e^{sT} \tag{24}$$

Several different methods can be used to convert an s-domain transfer function in the discrete-time z-domain. A commonly used method is the bilinear transform (also called the Tustin or trapezoidal transform). This transform approximates the continuous-time signal based on a trapezoid from k to k+1. An example of this is shown in Figure 18.

Figure 18. Continuous-time signal sampled in discrete time using a trapezoidal approximation

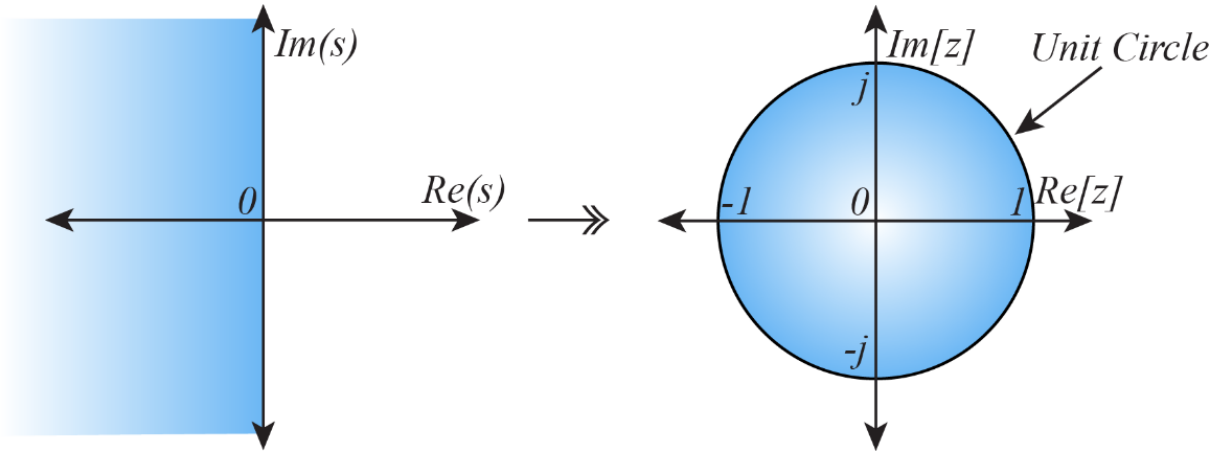


This trapezoidal approximation gives rise to the transform given in (25).

$$s \leftarrow \frac{2}{T_S}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{25}$$

This transform has the advantage that a system with stable poles and zeros, which means a left-half plane, transforms into a system with stable z-domain poles and zeros. The stable region in the z-domain is the area on or inside the unit circle. Therefore, the mapping is shown in Figure 19.

Figure 19. **Trapezoidal approximation mapping from s to z-domain**



There is inevitably some distortion with the mapping given that the entirety of the left-half s-plane is mapped to the unit circle in the z-domain. However, this distortion is only significant as the frequency approaches the sampling frequency. For this power supply application, most of the compensator poles and zeros are significantly below the sampling frequency. Furthermore, the crossover frequency is specified between 1/10th and 1/20th of the sampling frequency and there is no significant distortion from the mapping around this frequency.

However, this transform does not include the effects of the pure time delays in the discrete-time system and there is an additional phase roll-off that is not considered here. This manifests itself as a phase erosion at the crossover frequency and this must be taken into account when the compensator is being designed.

5.2 2p2z controller

The bilinear transform is applied to the s-domain Type II compensator by replacing every instant of s in the s-domain transfer function with the bilinear mapping. The initial substitution is given in (26) and the simplified result is given in (27).

$$H_C[z] = \left(\frac{\omega_{CP0}}{\frac{2}{T_S}\frac{1-z^{-1}}{1+z^{-1}}} \right) \frac{\left(\frac{\frac{2}{T_S}\frac{1-z^{-1}}{1+z^{-1}}}{\omega_{CZ1}} + 1 \right)}{\left(\frac{\frac{2}{T_S}\frac{1-z^{-1}}{1+z^{-1}}}{\omega_{CP1}} + 1 \right)} \quad (26)$$

$$H_C[z] = \frac{B_2 z^{-2} + B_1 z^{-1} + B_0}{-A_2 z^{-2} - A_1 z^{-1} + 1} \quad (27)$$

In the discrete-time z-domain, the transfer function now has two z-domain poles and two z-domain zeros. Therefore, this controller is now referred to as a two-pole two-zero controller or 2p2z for short.

The numerator of the transfer function is grouped into like terms consisting of ‘B’ coefficients and likewise, the denominator is grouped into like terms consisting of ‘A’ coefficients. These coefficients are given in (28) to (32).

$$B_0 = \frac{T_S \times \omega_{CP0} \times \omega_{CP1} \times (2 + T_S \times \omega_{CZ1})}{(2 \times (2 + T_S \times \omega_{CP1}) \times \omega_{CZ1})} \quad (28)$$

$$B_1 = \frac{{T_S}^2 \times \omega_{CP0} \times \omega_{CP1}}{(2 + T_S \times \omega_{CP1})} \quad (29)$$

$$B_2 = \frac{T_S \times \omega_{CP0} \times \omega_{CP1} \times (-2 + T_S \times \omega_{CZ1})}{(2 \times (2 + T_S \times \omega_{CP1}) \times \omega_{CZ1})} \quad (30)$$

$$A_1 = \frac{4}{(2 + Ts \times \omega_{CP1})} \quad (31)$$

$$A_2 = \frac{-(2 - Ts \times \omega_{CP1})}{(2 + Ts \times \omega_{CP1})} \quad (32)$$

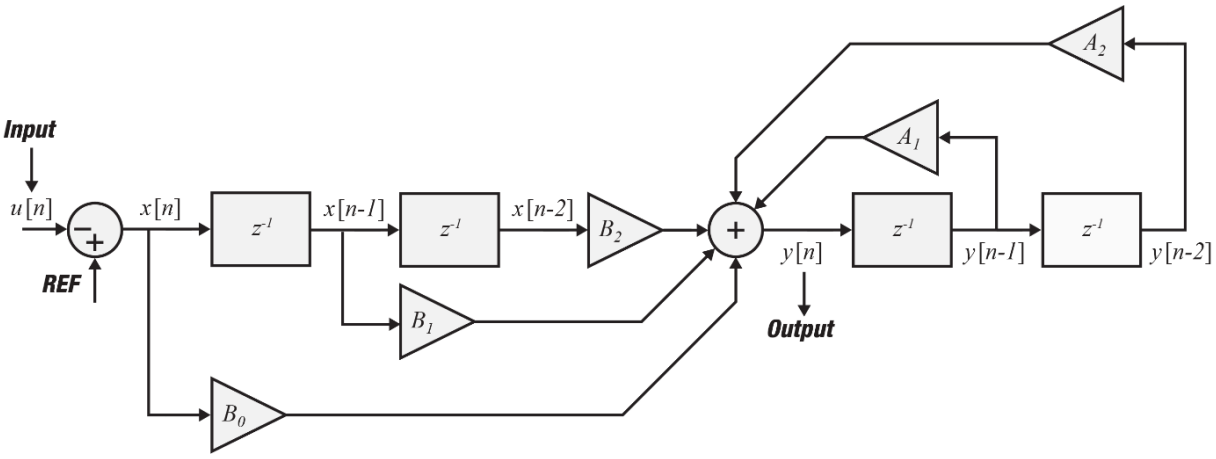
There is no need to calculate these controller coefficients by hand as the software tool Biricha ST-WDS available from *www.biricha.com/st-wds* performs these calculations.

Finally, now that the controller is in the discrete-time 2p2z form, it can be converted into a linear differential equation which can be easily implemented on the MCU. This takes advantage of the shifting property of the z-domain transfer function and converts from a z-domain to a discrete sample (33).

$$y[n] = A_1 y[n-1] + A_2 y[n-2] + B_0 x[n] + B_1 x[n-1] + B_2 x[n-2] \quad (33)$$

Where y[n] is the output for the current sampling interval. The structure of this controller is depicted in Figure 20.

Figure 20. 2p2z controller structure



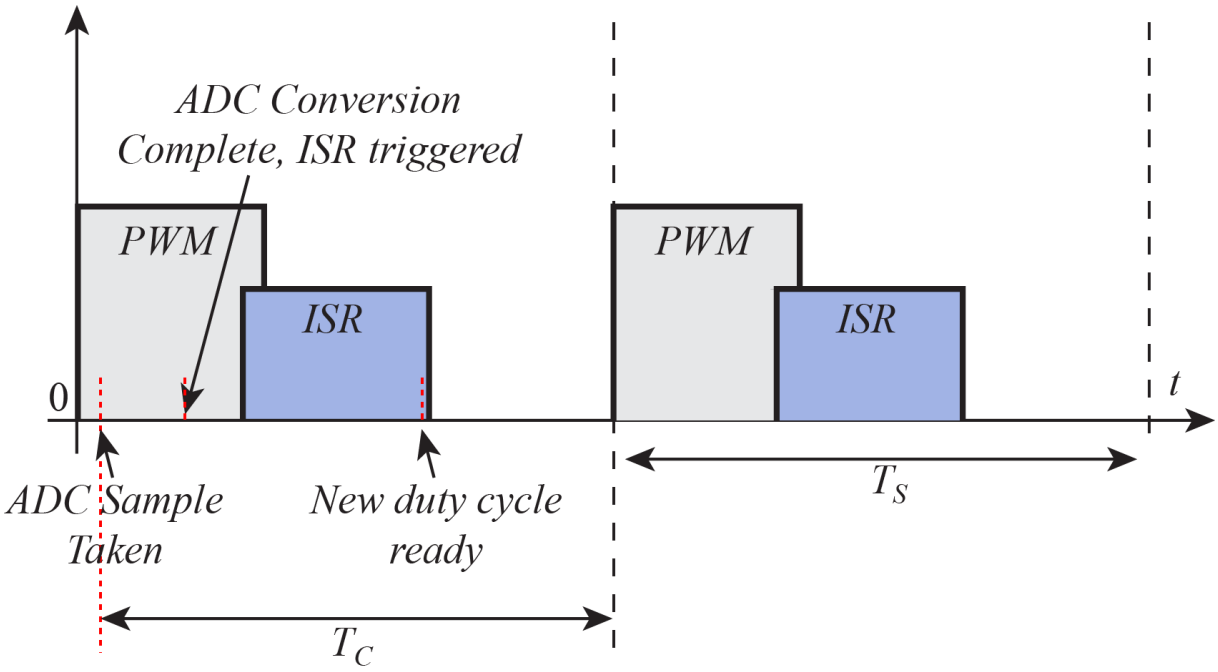
For this peak-current mode buck converter, the output of the controller during this sampling interval is the new demand peak current value which is used in the following switching period. Therefore, there is now an equation that can implement the analytically designed controller and is calculated by the MCU at every sampling interval when there is a new sample available.

5.3 Pure time delays

In this discrete-time digital system, there are additional time delays that are not present in the equivalent analog continuous-time system. A pure time delay in the discrete-time system results in a phase delay that is proportional to the frequency and time delay.

There are two sources of time delay in the digital system. The first is the time from which the output voltage sample is taken, to the time at which it is used. This is referred to as the calculation delay. In effect the older the sample the more phase delay is associated with the sample. As an ideal example, a convenient time to trigger the ADC and sample the output voltage may be at the beginning of the switching period. In this case, the output voltage is sampled, converted, and used in the controller to provide the new value demand peak current value. This sampling and calculation time may take several hundred nanoseconds and the new demand peak current value is ready to use towards the end of that switching period. This situation is described in Figure 21.

Figure 21. Sources of delay within the discrete-time system



However, as the new demanded peak current value is not used until the following switching cycle, the total time delay for this calculation time is considered to be one complete switching period. Of course, if the user can delay the trigger for the ADC until later in the switching period then it is possible to reduce this time delay.

The second contribution to the time delay is that of the zero-order hold (ZOH) introduced by this sampled data system. The ADC is triggered once per switching cycle, this sampled data is used to calculate the new value of demanded peak current value and that demanded peak current value is then held constant for the remainder of the next switching period by the DAC. Therefore, there is an additional zero-order hold in this system. The frequency response of a ZOH is given in (34).

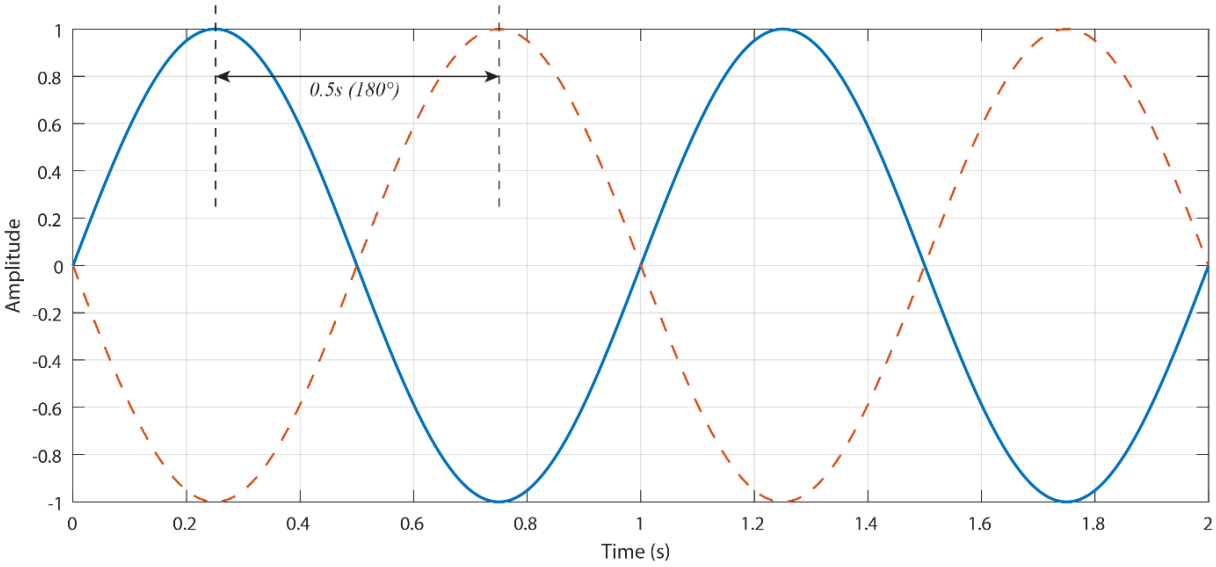
$$F_{ZOH}(j\omega) = \frac{1 - e^{-j\omega T_S}}{j\omega} \tag{34}$$

This can be shown to be a complex number expressed in polar form, and therefore the phase angle is given in (35).

$$\angle F_{ZOH}(j\omega) = -\omega\frac{T_S}{2} \tag{35}$$

To understand how this phase angle can relate to a time delay, consider as an example the 1-Hz sine wave shown in Figure 22. If this sine wave is delayed by 0.5 s, the result is the sine wave shown as a dashed line in Figure 22.

Figure 22. Time delay relationship to phase delay



It is clear from this that the time-delayed sine wave now has a phase delay of 180°. Therefore, the equation to calculate the phase delay of a sine wave given the pure time delay T_D is given in (36).

$$p.d. = -2\pi f T_D \tag{36}$$

This indicates that there is some phase delay across all frequencies, however, when $T_D << 1/f$ the phase delay is negligible. This is true for very low frequencies assuming that the phase delays in the digital system can be kept to a minimum.

The frequency at which this phase delay may become a concern is the crossover frequency of the open-loop system as this is where the phase margin is defined. Earlier in this application note, it is specified that the phase margin has a minimum value of 45° to ensure a transient response that is not oscillatory in the time domain. Therefore, the amount of phase loss at the crossover frequency can be calculated given the known time delays in the digital system.

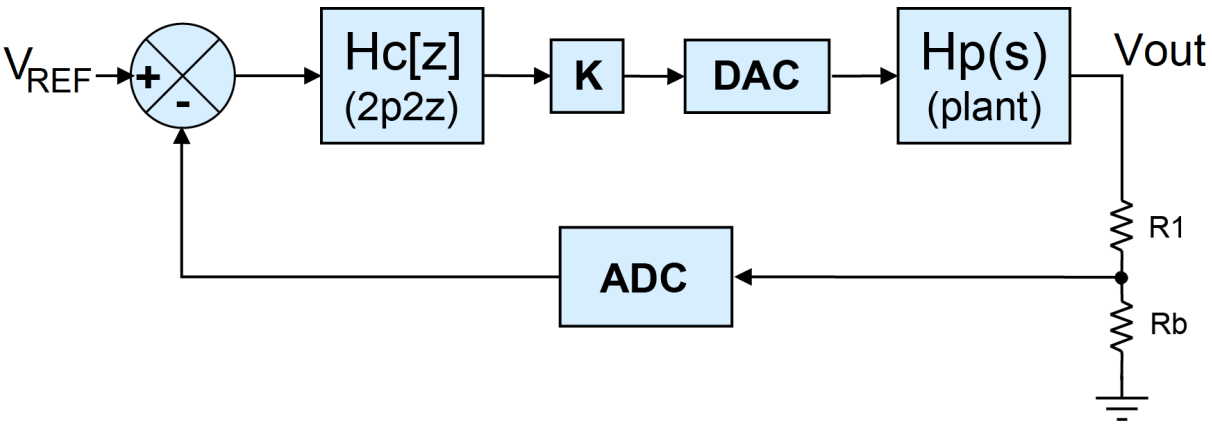
$$p.e. = -2\pi f_X T_D \tag{37}$$

The expected phase erosion at the crossover frequency can be added to the desired phase margin such that this phase loss is compensated for at design time. This can be achieved by adding the expected phase erosion to the phase margin specified in (37). This adjusts the location of the compensator zero to add the required phase boost and compensate for this phase loss such that the resulting overall phase margin meets the specification.

5.4 Digital gains

The output of the 2p2z controller in Figure 23 is scaled by a factor, K. This scaling factor is responsible for negating all of the additional gains introduced by the digital system. The compensator poles and zeros are analytically calculated to achieve the specified crossover frequency and phase margin. If the additional gains within the digital system are not accounted for then the overall loop gain, and thus the phase margin, is incorrect.

Figure 23. Additional gains within the digital system in peak current mode



Consider the block diagram of the digital control loop shown in Figure 23. The output voltage Vout needs to be scaled down to a voltage suitable for sampling by the ADC onboard the MCU. This is the purpose of the potential divider formed by R1 and Rb. This also determines the setpoint value that the controller must regulate. The gain of the pre-ADC scaling potential divider can be calculated in (38).

$$G_{PD} = \frac{R_b}{R_1 + R_b} \tag{38}$$

Then the ADC onboard the MCU converts the voltage, which is between 0 and 3.3 V, and provides a 12-bit output, meaning a number between 0 and 4095. Therefore, the gain of the ADC can be calculated using (39).

$$G_{ADC} = \frac{2^{12} - 1}{3.3\,V} \tag{39}$$

The output of the controller is used to update the DAC register, which sets the demanded peak current level for the internal comparator. The 12-bit DAC gives an output voltage of 3.3 V for a 4095 input. Therefore, the DAC gain can be calculated using (40).

$$G_{DAC} = \frac{3.3\,V}{2^{12} - 1} \tag{40}$$

To achieve the correct loop gain, the controller must negate these additional gains. That is achieved using the scaling factor K, which can be calculated in (41).

$$K = \frac{1}{G_{PD} G_{ADC} G_{DAC}} \tag{41}$$

Finally, the ADC output is compared with the digital reference set point. This digital reference must be in the same scaling as the ADC output and take into account the divider on the input of the ADC. Therefore, the reference can be calculated using (42).

$$V_{REF} = V_{OUT} G_{PD} G_{ADC} \tag{42}$$

6 Digital peak current mode implementation

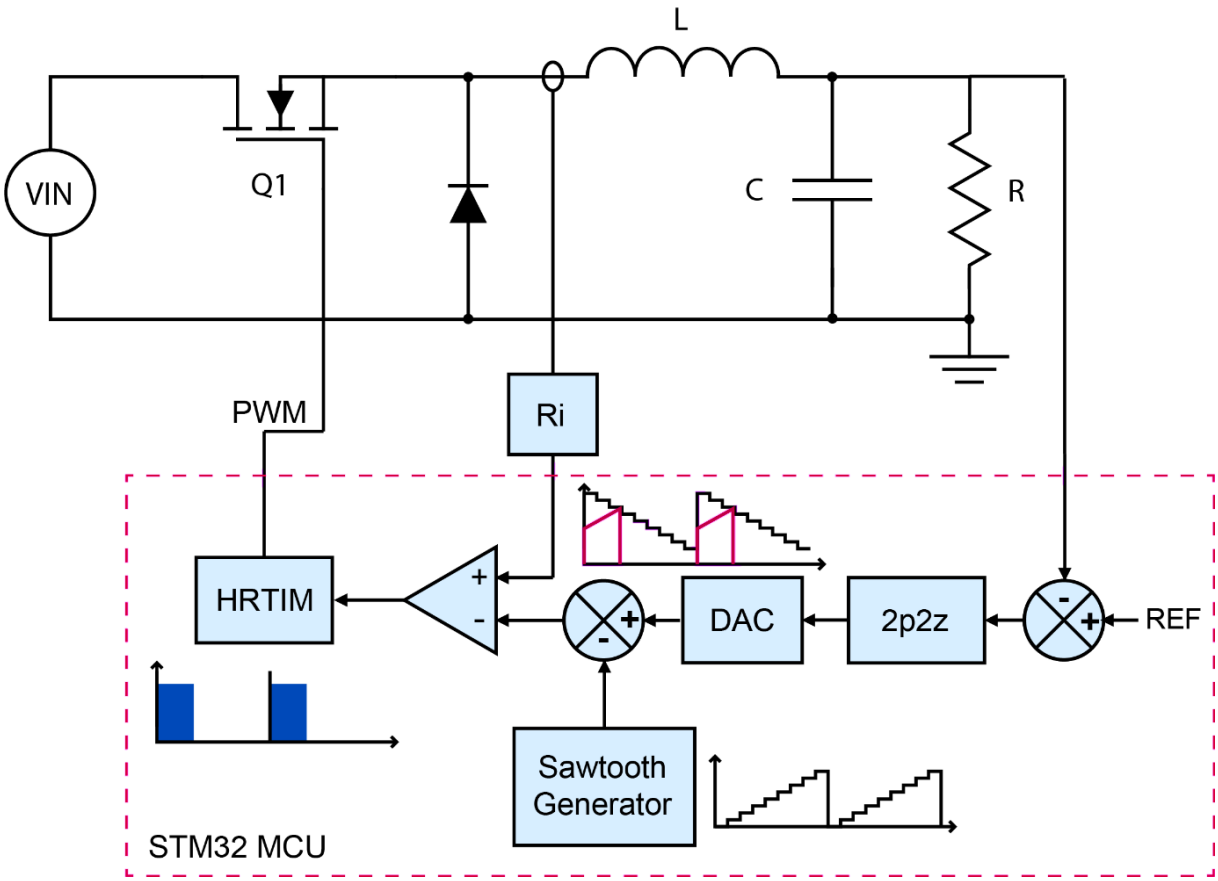
6.1 Onboard comparator and DAC

The analog peak current mode implementation discussed so far detects the intersection of the demand peak current with the actual measured current using a peak current detection comparator. For digital implementation, this peak current detection also needs to be implemented in such a way that the switch is turned off as fast as possible. Using the ADC to sample the inductor current multiple times per switching period and comparing this to the demand reference value may be too slow.

Therefore, the digital implementation uses the onboard comparator for the same purpose as the peak current detection comparator in the analog design. One input to the comparator is the measured inductor current, and the other input to the comparator is the output of the controller setting the demand peak current value. Of course, the output of the controller is a digital number and, therefore, must be converted to an analog voltage via the onboard DAC. The STM32 G4 series of MCUs contain multiple 12-bit DACs that can be used as inputs to the onboard comparators.

The output of the comparator is then used as an external event within the HRTIM module, and this is configured as a reset source for the output driving the FET. The complete digital configuration is shown in Figure 24.

Figure 24. Digital implementation of peak current mode control using STM32 G4 series of MCUs



6.2 Sawtooth waveform generator

The digital peak current mode implementation also requires the implementation of slope compensation to reduce the Q of the complex conjugate poles at half the switching frequency. This effectively damps the subharmonic oscillations and prevents the oscillations from increasing from switching period to switching period. To implement the slope compensation in the digital design, the sawtooth waveform generator that is part of the onboard DAC is used.

The sawtooth waveform generator is used to subtract a specified value from the DAC at multiple sub-intervals within the switching period. The initial value of the sawtooth waveform is the output of the 2p2z controller calculated in the previous switching period. Then, a specified value is subtracted from this at predefined intervals within the switching period. The result is a peak-to-peak ramp height equivalent to that of the slope compensation ramp applied in analog.

The sawtooth waveform generator update is triggered by one of the HRTIM compare units. There is a maximum rate of update triggers for the waveform generator, and any triggers faster than this rate is ignored. The required overall ramp height is known and therefore, given a chosen update rate, the sawtooth waveform generator decrement value can be calculated. First, the number of updates per period is selected. In this example, there are 75 updates during each 5 µs period given a DAC update rate of 15 msps.

$$Updateperiod = \frac{Switchingperiod}{Numberofupdatesperperiod} \quad (43)$$

$$Updateperiod = \frac{5\,\mu s}{75} = 66\,ns \quad (44)$$

The HRTIM compare unit 3 is configured in the next section of this application note as the trigger source for the sawtooth waveform generator update. Therefore, the 66 ns calculated in (44) is converted to the number of HRTIM timer ticks to achieve this update rate.

The value to be subtracted from the DAC at each subinterval is then calculated in (46). This needs to be converted from mV units to the correct scaling for the DAC. A function, called `SetRamp()`, is included in the example project for this board to convert from the required slope compensation ramp height to DAC decrement value.

$$\Delta V/step = \frac{V_{ramp}}{Numberofupdatesperperiod} \quad (45)$$

$$\Delta V/step = \frac{0.5\,V}{75\,steps} = 6.66\,mV/step \quad (46)$$

7 Software implementation

7.1 Targeted application

The following configuration sets up the STM32 MCU to operate a closed-loop peak-current mode buck converter using the onboard peripherals including the ADC, DMA, HRTIM, and DAC with digital slope compensation. The FMAC is used to implement the 2p2z controller. This implementation means that the main core usage is reduced to an absolute minimum and is the preferred option allowing the MCU to be utilized for other tasks, or running more power supplies. The example software project to accompany this application note is called `Buck_CurrentMode_HW` as it uses as many hardware peripherals as possible. However, it is possible to use the main core to implement the 2p2z controller instead of the FMAC. In this case, the configuration is different from that discussed below and an example of this is provided in the project titled `Buck_CurrentMode_SW`.

7.2 Configuration using STM32CubeMX

The following section contains step-by-step instructions for recreating the STM32CubeMX project for the buck converter under peak current mode control on the Discovery kit. This complete project can be downloaded by following the links provided within this application note appendix. However, the full configuration is included here for completeness.

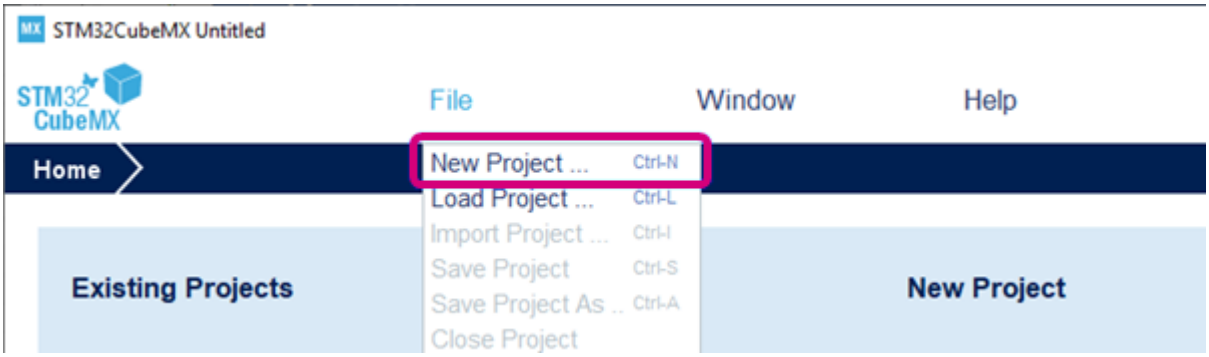
Open STM32CubeMX by clicking on the icon shown in Figure 25 (note that the icon may differ slightly).

Figure 25. STM32CubeMX icon

Now create a new STM32CubeMX project. This project configures the MCU peripherals and also generates an IAR Embedded Workbench® project. IAR Systems® is used to compile and link the code as well as for programming and debugging the MCU.

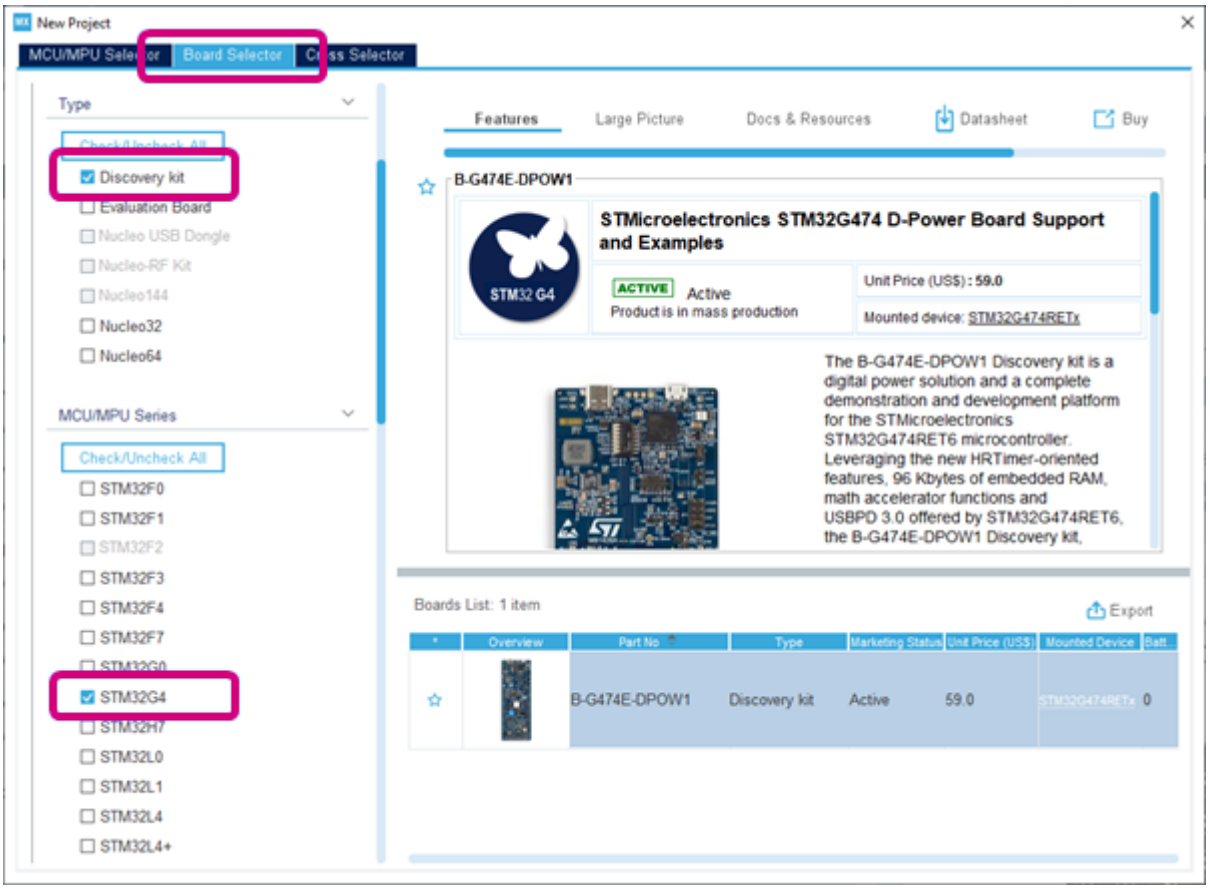
Within the STM32CubeMX window click on `File`, `New Project`.

Figure 26. New project selection



The new project device selector window now opens. Within this window click on the `Board selector` tab and filter down the boards by selecting the `Discovery kit` for the `Type` and `STM32G4` for the `MCU/MPU Series`.

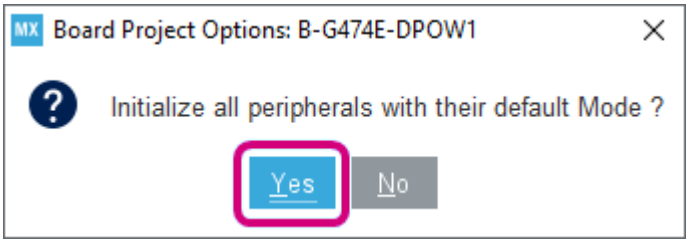
Figure 27. Board selection



This may filter down the available boards on the right-hand side of this window to include the `B-G474E-DPOW1` Discovery kit which this application note is using. Double-click on this board within the table.

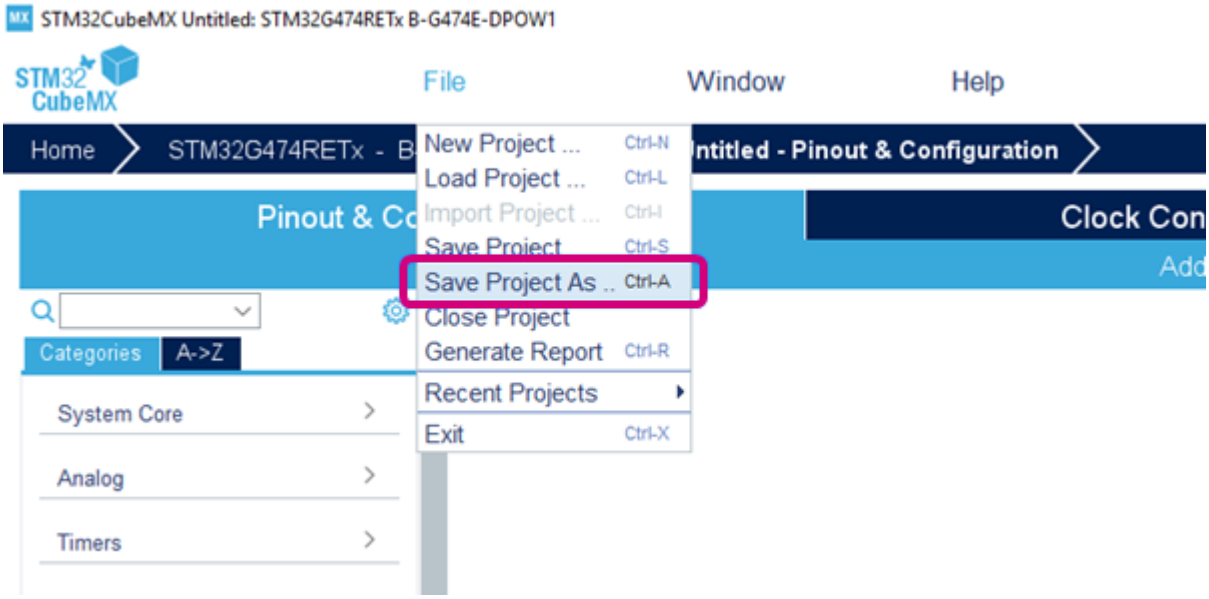
A prompt is displayed asking if the user likes to initialize all peripherals with their default mode. Click `Yes`. This sets up the pins and peripherals with their default setting for this particular evaluation board.

Figure 28. Peripherals default mode initialization



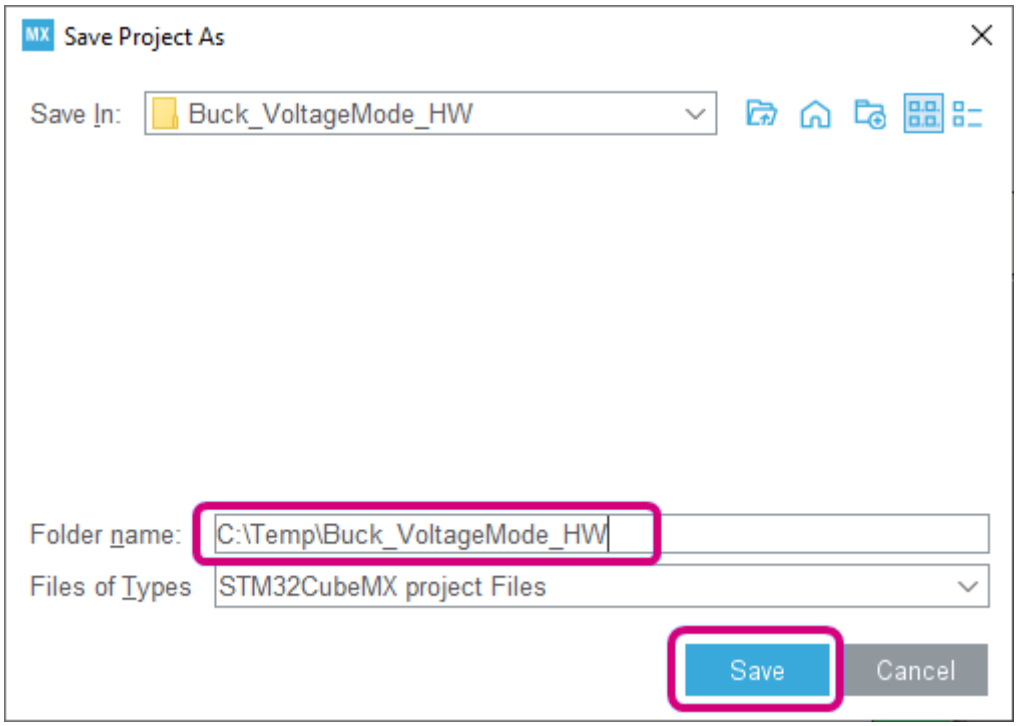
Before making any changes to the project, save the project by going to `File`, `Save Project As...`

Figure 29. Project naming



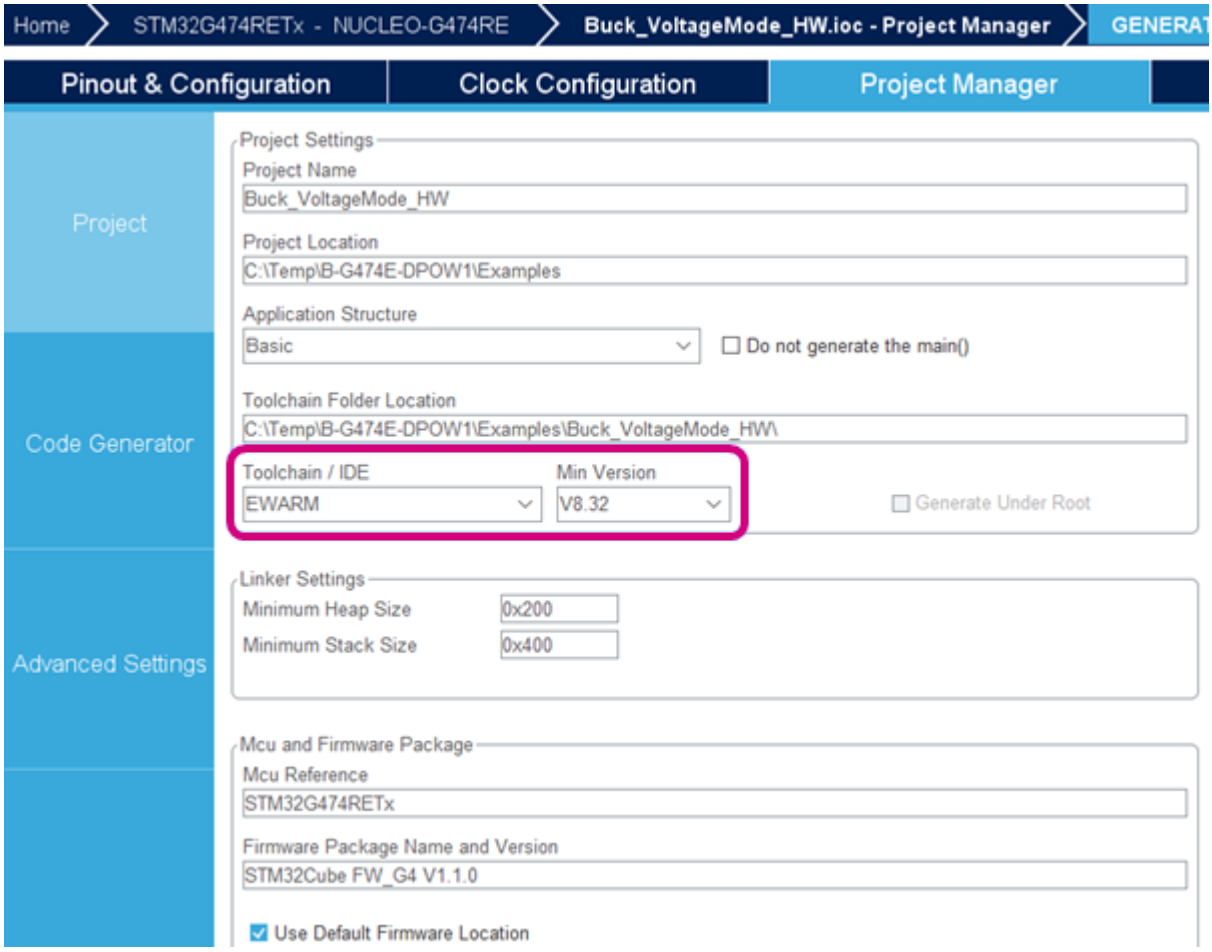
The name given to the folder, `Folder Name`, is also the name of the project. If this folder does not exist it is created. Click `Save` when done.

Figure 30. Project saving



On the right-hand side of the main window click `Project Manager` and select the preferred toolchain and version from the dropdown list as shown in Figure 31.

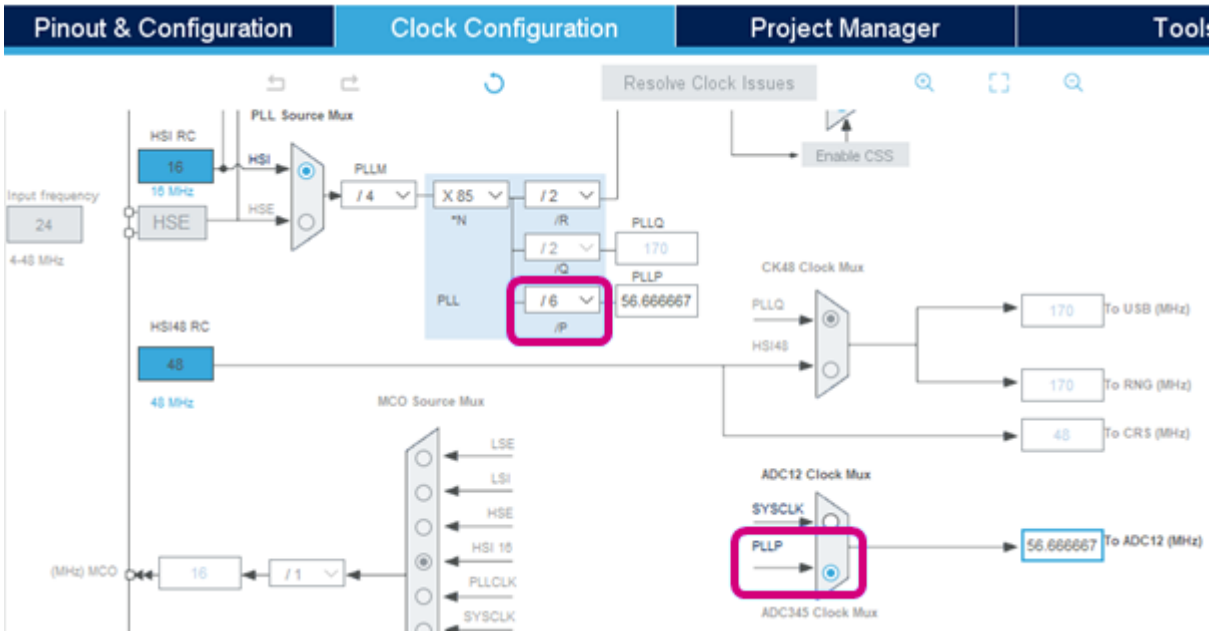
Figure 31. Project manager configuration



7.2.1 Clock configuration

Select the `Clock Configuration` tab which is along the top of the main STM32CubeMX window. Locate the PLL section and change the peripheral clock divider to `/6`. Then select the clock source for the ADC12 Clock Mux to `PLLP`. These settings are highlighted in Figure 32.

Figure 32. Clock configuration window

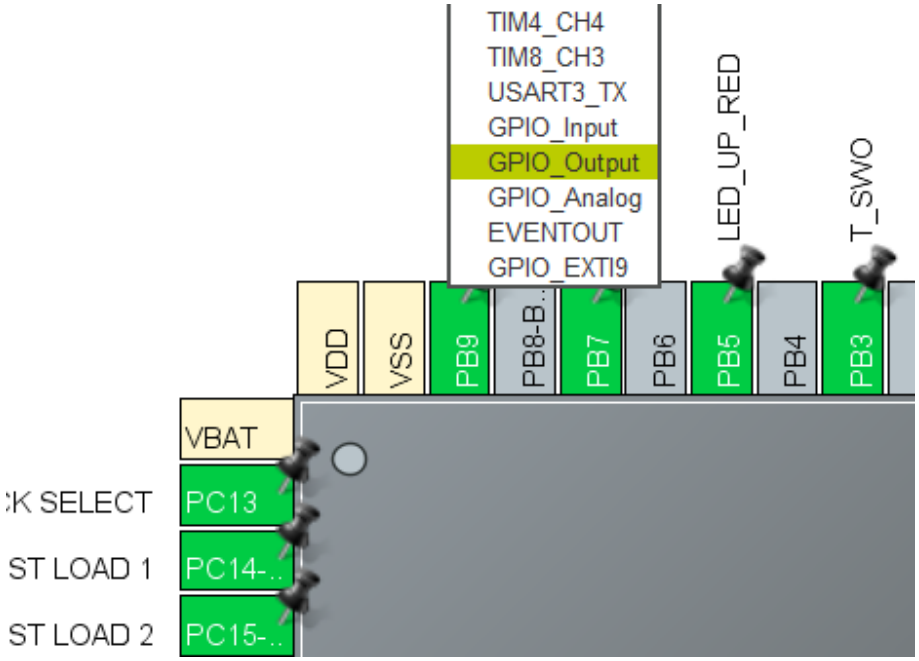


The ADC12 clock may now be 56.66 MHz.

7.2.2 GPIO peripheral configuration

For this application note, a digital pin is configured to allow timing measurements to be performed. On the `Pinout & Configuration` tab, locate the pin PB9 towards the top left of the microcontroller. Left-click on the pin PB9 and select `GPIO_Output` as in Figure 33.

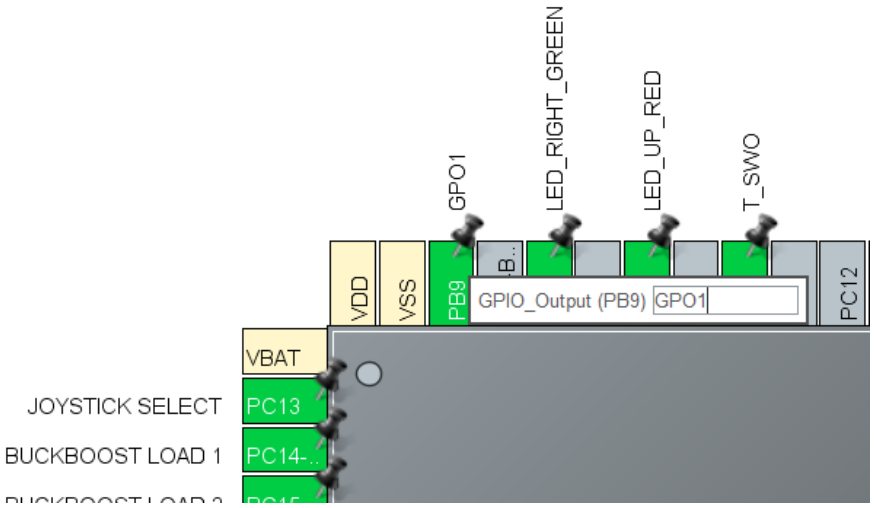
Figure 33. GPIO peripheral configuration window



Then, right-click on PB9 and select `Enter User Label`. This allows us to change the name of the pin so that it can be easily referenced within the code. This pin is called `GPO1`.

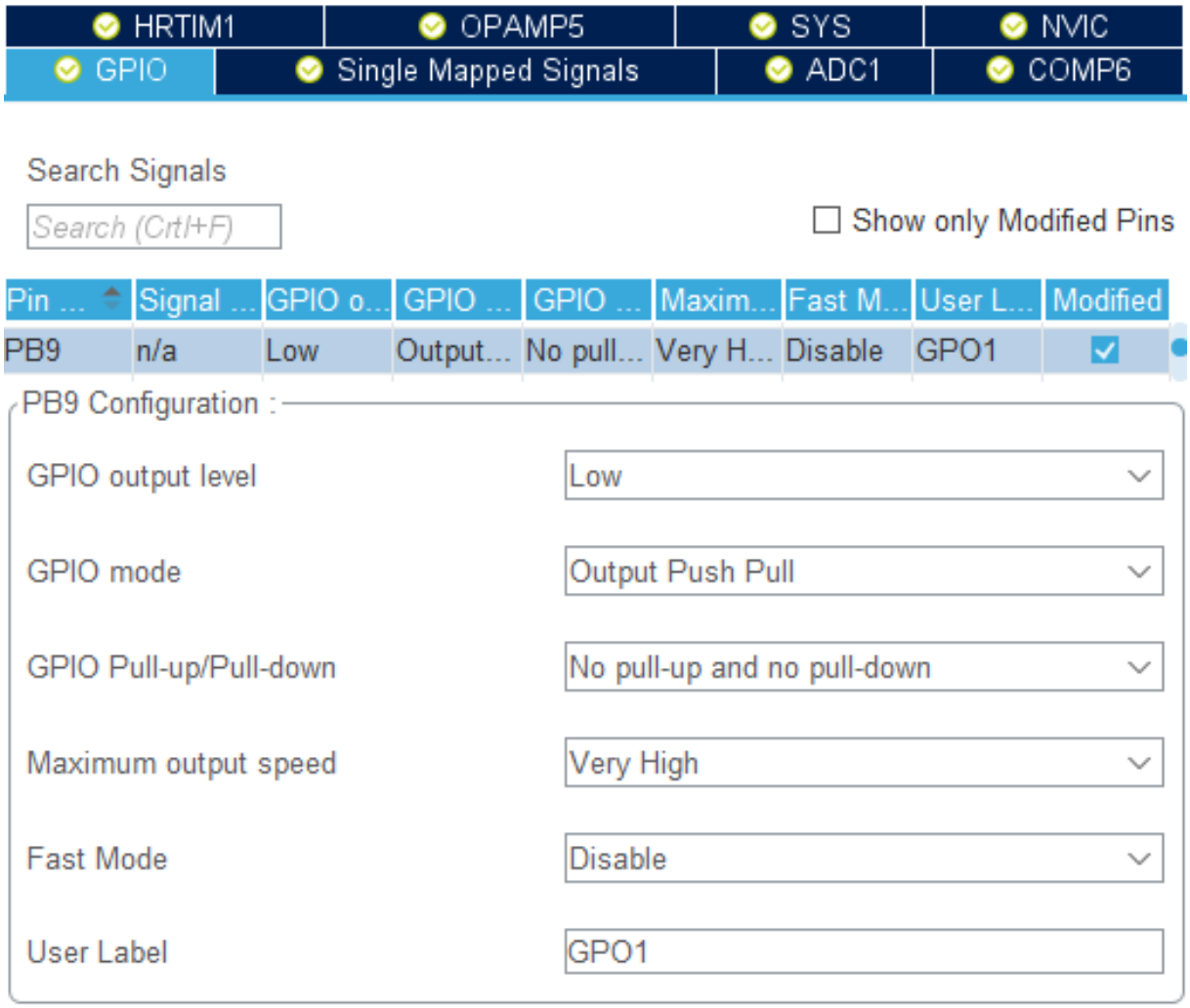
Enter this into the pop-up box as in Figure 34.

Figure 34. GPIO renaming



Now expand the `System Core` category on the left-hand side of the window. Click on `GPIO`. Under the `GPIO` tab, click on the row for PB9. Change the `Maximum output speed` setting to `Very high` as in Figure 35.

Figure 35. GPIO maximum output speed setting



The configuration for GPO1 is now complete.

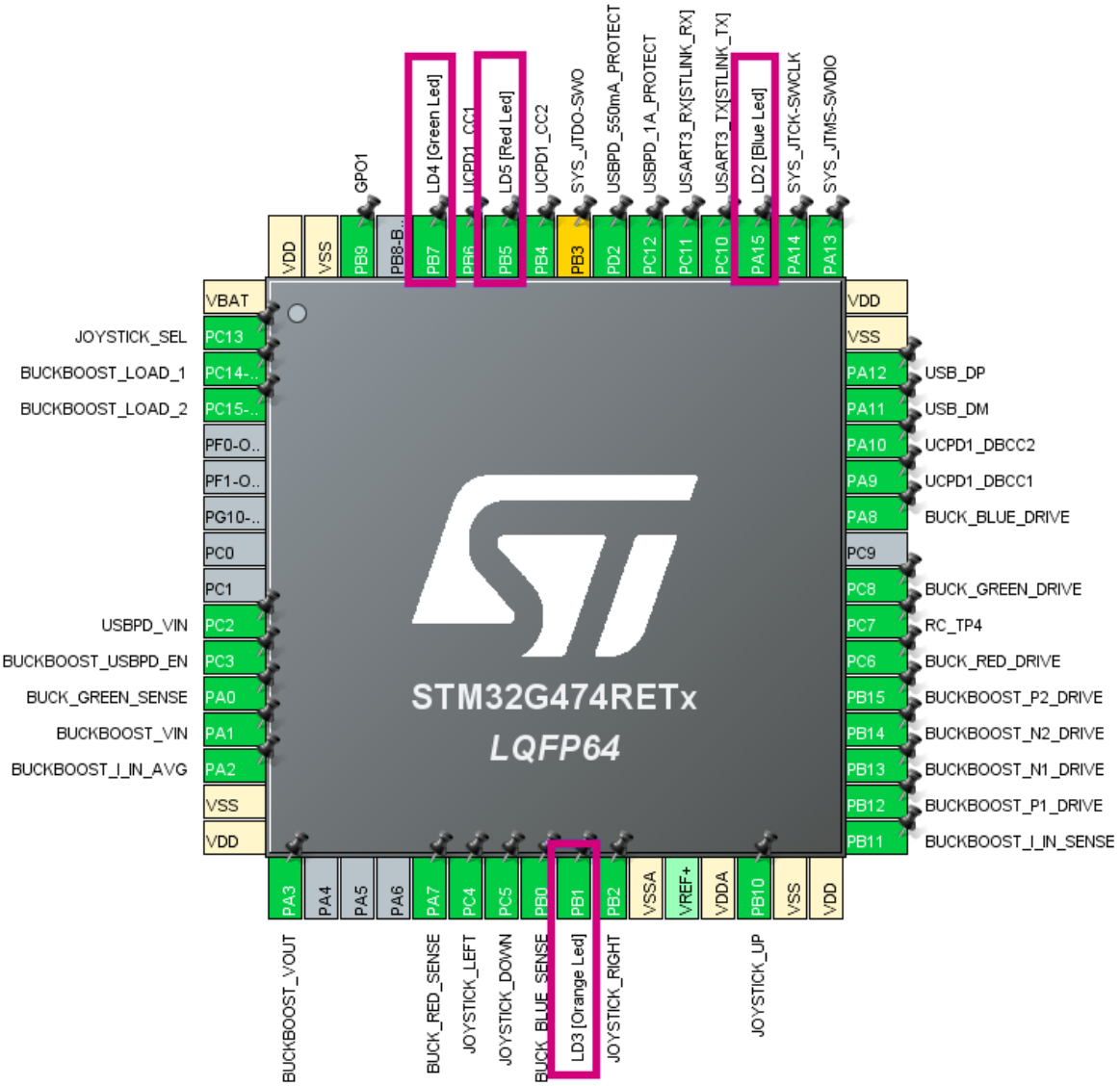
Now, some of the default pin labels must be changed to work with the example code provided. Within the pinout view of the MCU, locate the following pins, right-click on them and select `Enter User Label` and change the label to the new label listed in Table 1.

Table 1. User label setting

Pin	Existing label	New label
PB7	LD4 [Green LED]	LED_RIGHT_GREEN
PB5	LD5 [Red LED]	LED_UP_RED
PA15	LD2 [Blue LED]	LED_DOWN_BLUE
PB1	LD3 [Orange LED]	LED_LEFT_ORANGE

The pin locations are highlighted in Figure 36.

Figure 36. Pins location

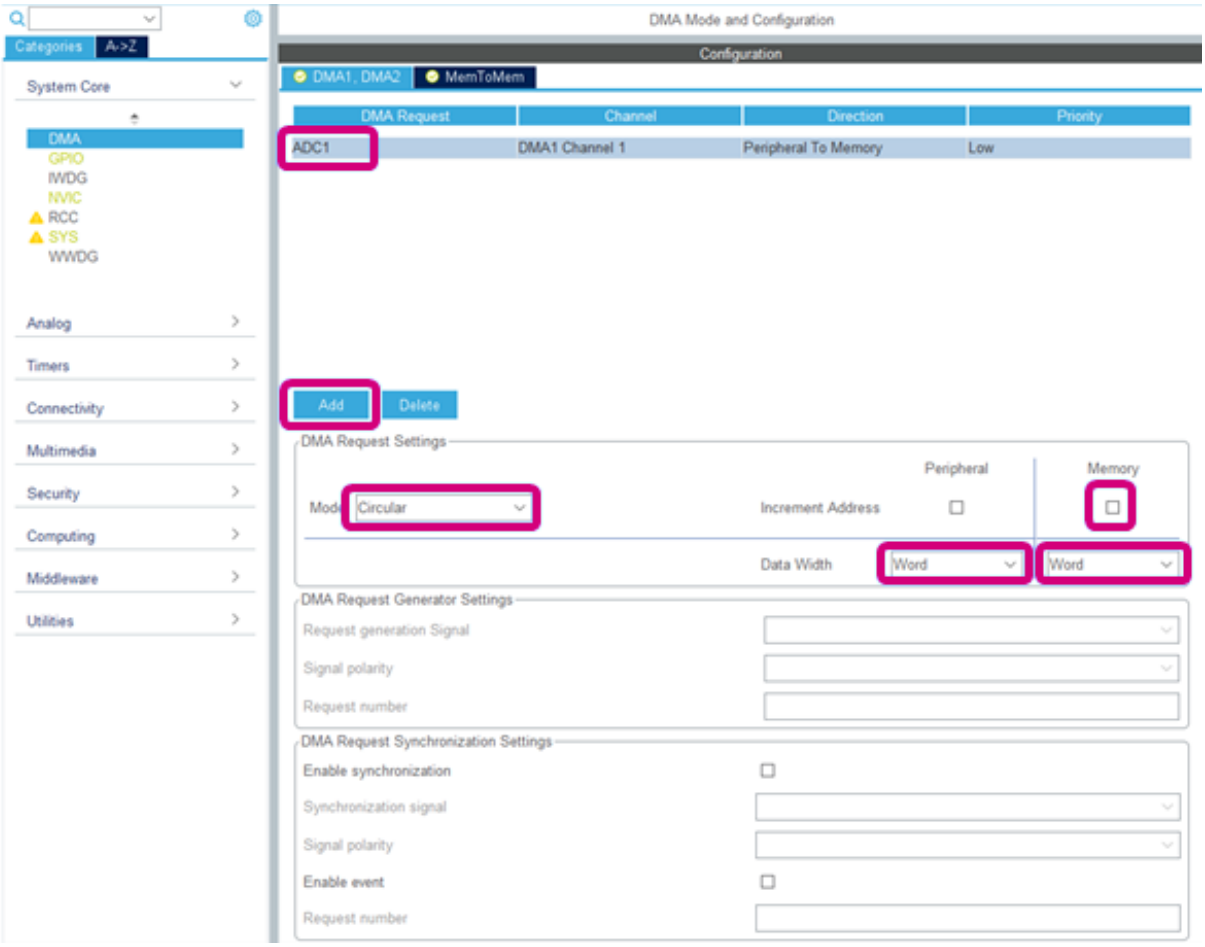


The GPIO configuration is now complete.

7.2.3 ADC/FMAC peripheral configuration

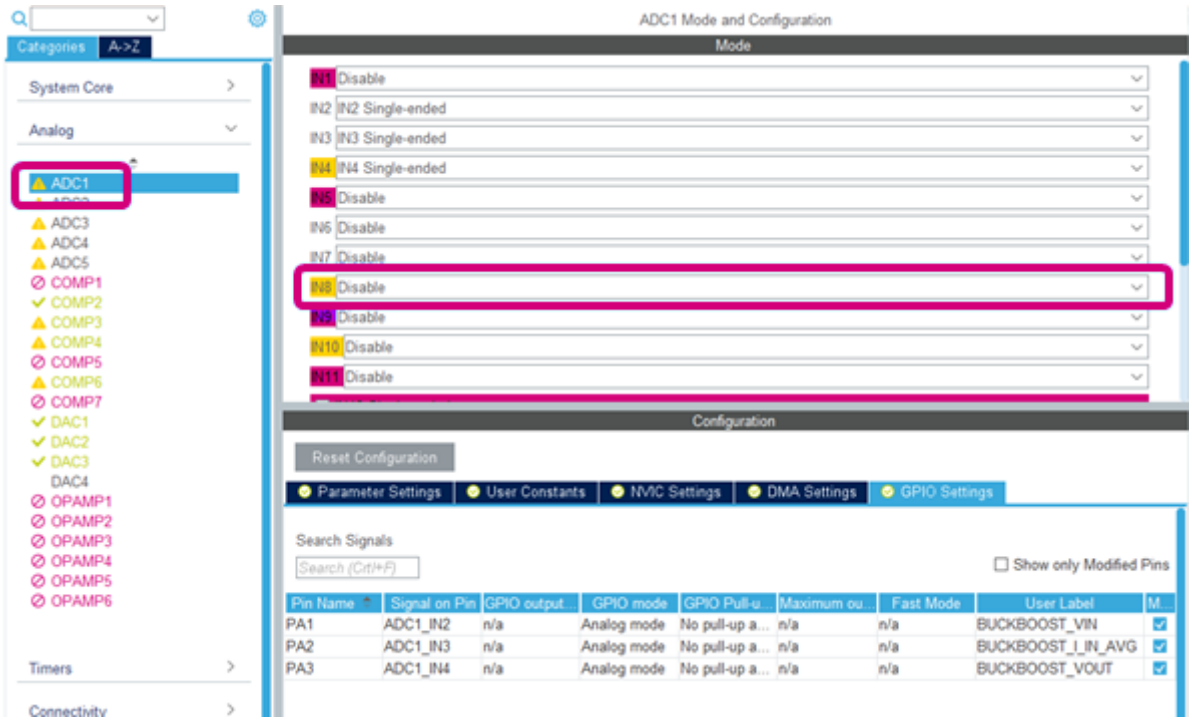
The peripherals can now be configured. On the top menu select `Pinout & Configuration` and then on the left-hand pane, expand the `System Core` list and click on `DMA`. The DMA is used to copy the result from the ADC result register to the FMAC for the execution of the controller. By default, there is no DMA request configured and therefore one must be added for the ADC. On the `DMA1, DMA2` tab, click the `Add` button. Select `ADC1` as the `DMA Request` source. Set the `Mode` to `Circular` and untick the incremented address for memory. Change the `Data Width` to `Word` for both Peripheral and Memory. The configuration window may now look like the screenshot in Figure 37.

Figure 37. DMA mode and configuration screen



The DMA configuration on this tab is now complete. Next, the ADC is configured. Expand the Analog peripheral list and click on ADC1. Click on the GPIO Settings tab and the list of pins already configured is shown. These are configured based on the EVM default which is loaded when the project is created. However, the USBPD_VIN is not required for this project and this can be removed by changing IN8 to Disable.

Figure 38. Removing pin from the project



On this screen, it can also be seen that ADC Channel 4 (ADC_IN4) has the user label BUCKBOOST_VOUT and is connected to PA3. This is the output voltage of the buck converter. This is required in the next configuration step.

Click on the Parameter Settings tab. Here, under the ADC_Settings section, the Data Alignment may be changed to Left Aligned. This stores the 12-bit result from the ADC in the 16-bit results register with left alignment. That means the upper 12-bits (plus 1 sign bit) of the 16-bit register are used.

Change the DMA Continuous Requests setting to Enabled.

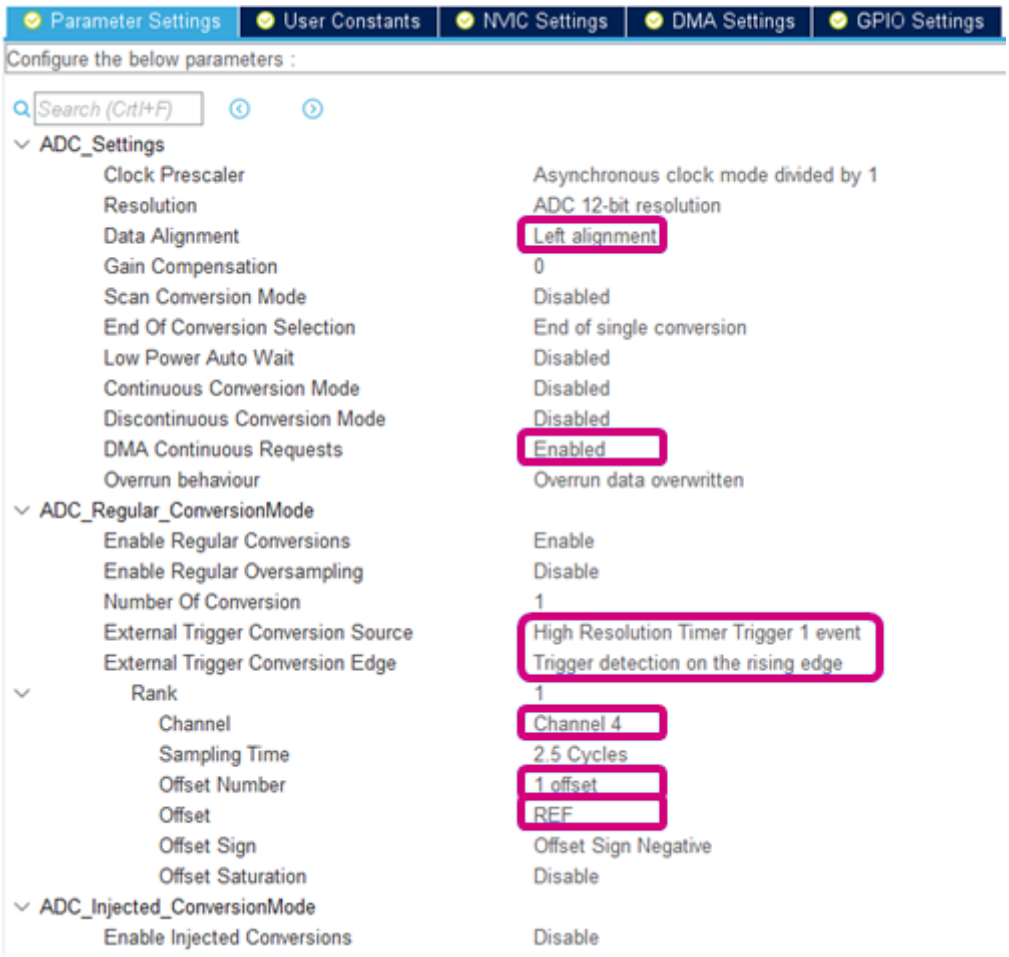
Under the ADC_Regular_ConversionMode section, set the External Trigger Conversion Source to High Resolution Trigger 1 event. Then set the External Trigger Conversion Edge to Trigger detection on the rising edge.

For this peak-current mode buck converter, only one ADC conversion is required. This is the output voltage rail that is regulated. Therefore, the number of conversions is currently set to one. This conversion can be configured by expanding the Rank subcategory.

Set the Channel to Channel 4 (this is the channel associated with the output voltage from the previous step). Then configure the Offset Number to 1 Offset. The offset feature of the ADC allows an offset to be subtracted from the ADC value before it is stored in the results register. This is used to perform the error calculation, VERR = VREF – VOUT, in hardware.

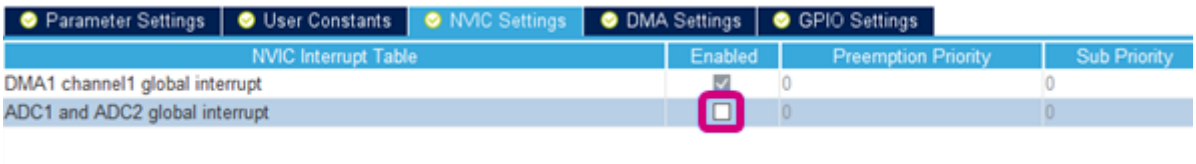
Set the `Offset` value to `REF`. REF is a term that is defined in the code later on. By default, STM32CubeMX has error-checking on the input fields. To disable this and allow the currently unknown value REF to be entered into this input field, click on the cog symbol on the right-hand side of the input value and change the check to `No check`.

Figure 39. ADC parameter settings



Still, on the ADC1 configuration pane, click on the `NVIC Settings` tab and disable the ADC interrupt by unticking the `ADC1 and ADC2 global interrupt` checkbox.

Figure 40. NVIC settings tab



The ADC configuration is now complete.

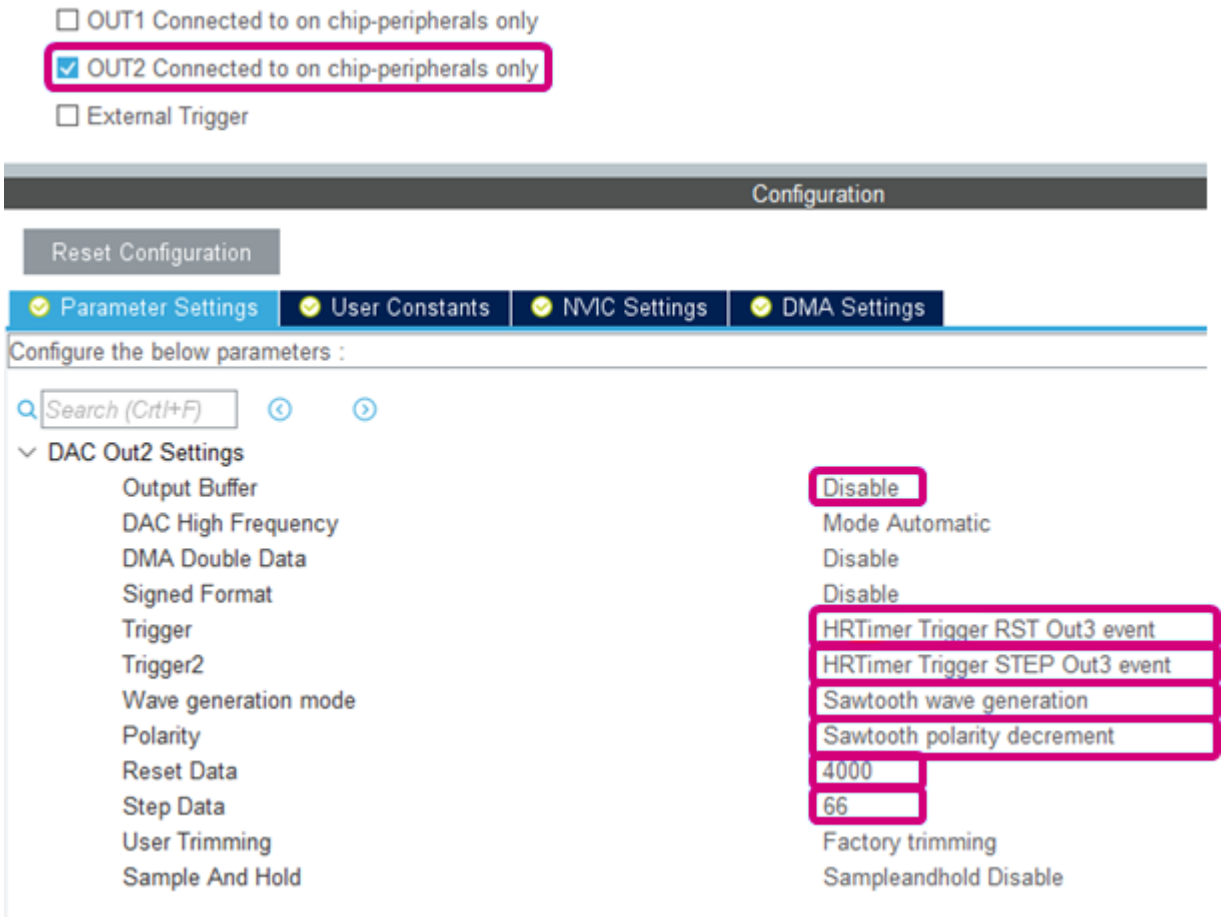
7.2.4 DAC peripheral configuration

Next, on the Analog peripheral category, disable COMP2, COMP3, COMP4, and DAC1 as these are not used in this example. To do this, uncheck the Input [+] checkbox under COMP2 and COMP3. Then change the OUT1 mode to Disable on DAC1.

Now configure the DAC used for setting the peak current detection threshold. On the DAC4 configuration pane, check the box next to OUT2 connected to on chip-peripherals only. Then, under the Parameter Settings tab, DAC Out2 Settings section, set Output Buffer to Disable, set Trigger to HRTimer Trigger RST Out3 event, and change Trigger2 to HRTimer Trigger STEP Out3 event.

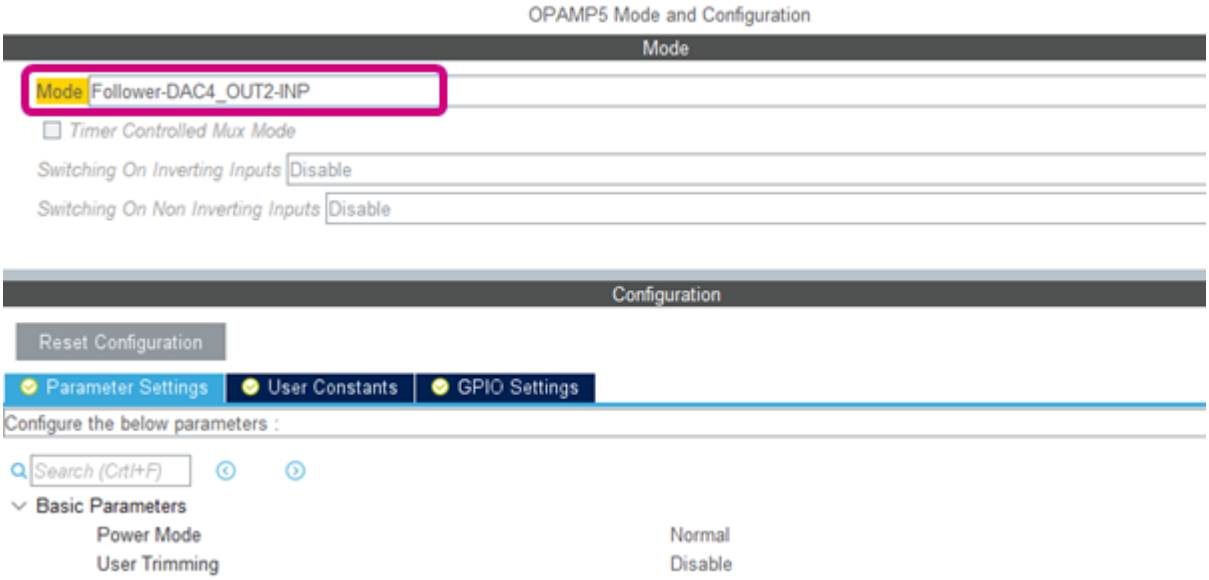
Then, to configure the slope compensation using the built-in sawtooth waveform generator, change the Wave generation mode to Sawtooth wave generation. Set the Polarity to Sawtooth polarity decrement and Reset Data to 4000 with Step Data to 66. These are initial values and the actual values for the sawtooth waveform generator are discussed later and set at run-time in the code.

Figure 41. DAC peripheral configuration 1/3



Next, on the Pinout view of the MCU locate PA8, left-click on this and change the operation of this pin to OPAMP5_VOUT. Then, on the Analog peripheral category, select OPAMP5. Change the Mode to Follower-DAC4_OUT2-INP. This connects the DAC to OPAMP5 and configures it in follower mode allowing us to view the output of the DAC on a physical pin.

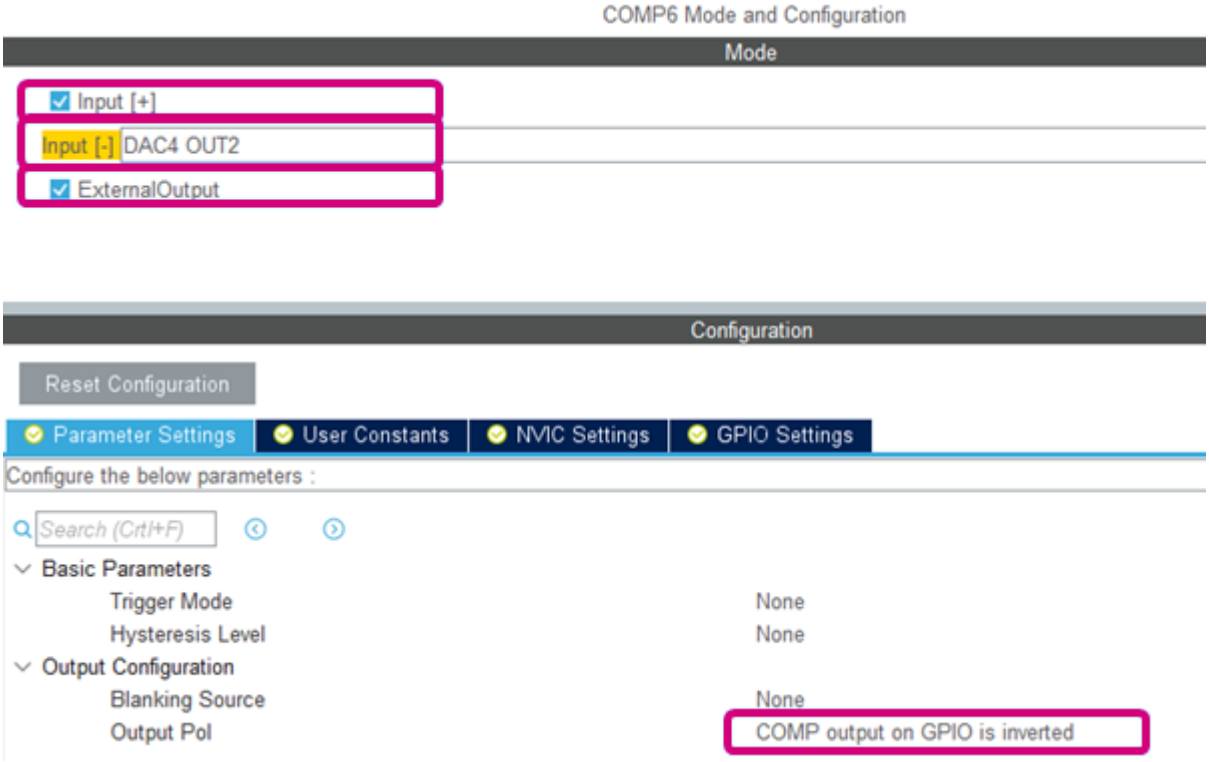
Figure 42. DAC peripheral configuration 2/3



Next, on the Pinout view of the MCU locate PA10, left-click on this and change the operation of this pin to COMP6_OUT.

Then, also on the Analog peripheral category, select COMP6. The onboard comparator is used to reset the HRTIM output when the measured current reaches the value set by the DAC. Tick the box next to Input [+] . Then select DAC4 Out2 as the Input [-]. Tick the box next to External Output to allow us to view the point at which the comparator changes on the oscilloscope. Under the Parameter Settings tab, modify the Output Pol to COMP output on GPIO is inverted as in Figure 43.

Figure 43. **DAC peripheral configuration 3/3**

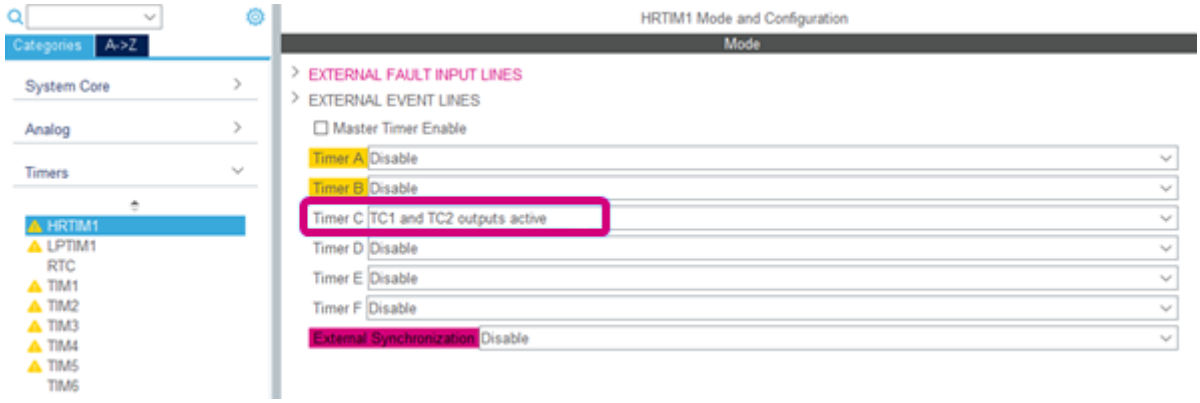


7.2.5 HRTIM configuration

Next, the hi-resolution timer module is configured to generate the PWM. On the left-hand side peripherals pane, expand the `Timers` category, and click on `HRTIM1`.

The Discovery kit has multiple switching power stages onboard and the timers for these are configured by default when the project is created. However, for this peak-current mode buck converter, only `Timer C` is required. Disable all of the other timers as shown in Figure 44.

Figure 44. **All timers except C are disabled**



The power stage of the Discovery kit contains boost switches that are following the buck stage and are used when operating as a non-inverting buck-boost converter. These boost switches are controlled by Timer D, however, in this application note, the boost switches are not driven by PWM. The boost switches form part of the buck power stage current path, and therefore they must be configured in either a HIGH or LOW state to allow the buck stage to function correctly.

Now that Timer D is disabled, go back to the Pinout view of the microcontroller. PB15 and PB14 may be highlighted in yellow. These are the pins that control the boost switches. Left-click on PB15 and PB14 and configure them both as a GPIO_Output.

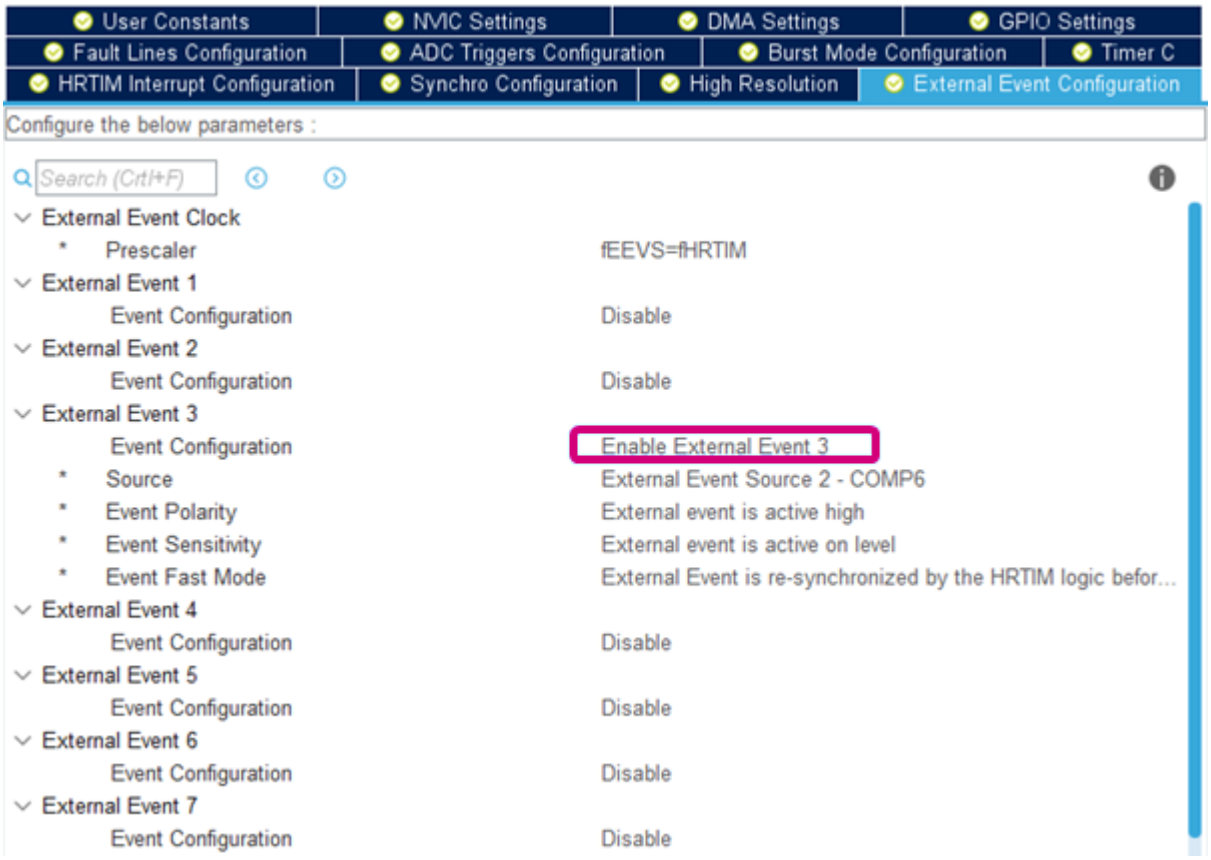
Right-click on PB15 and select Enter User Label. Enter the label: BUCKBOOST_P2_DRIVE.

Right-click on PB14 and select Enter User Label. Enter the label: BUCKBOOST_N2_DRIVE.

Within the program code, these pins are set either HIGH or LOW to enable or disable the boost switches and allow the converter to operate under buck mode.

Next, within the HRTM1 configuration pane, click on the External Event Configuration tab. Here the output of the comparator is connected to the HRTIM module. Under the External Event 3 section, change the Event Configuration to Enable External Event 3. The source of the external event is, by default, External Event Source 2 – COMP6. This is the comparator configured in the previous step.

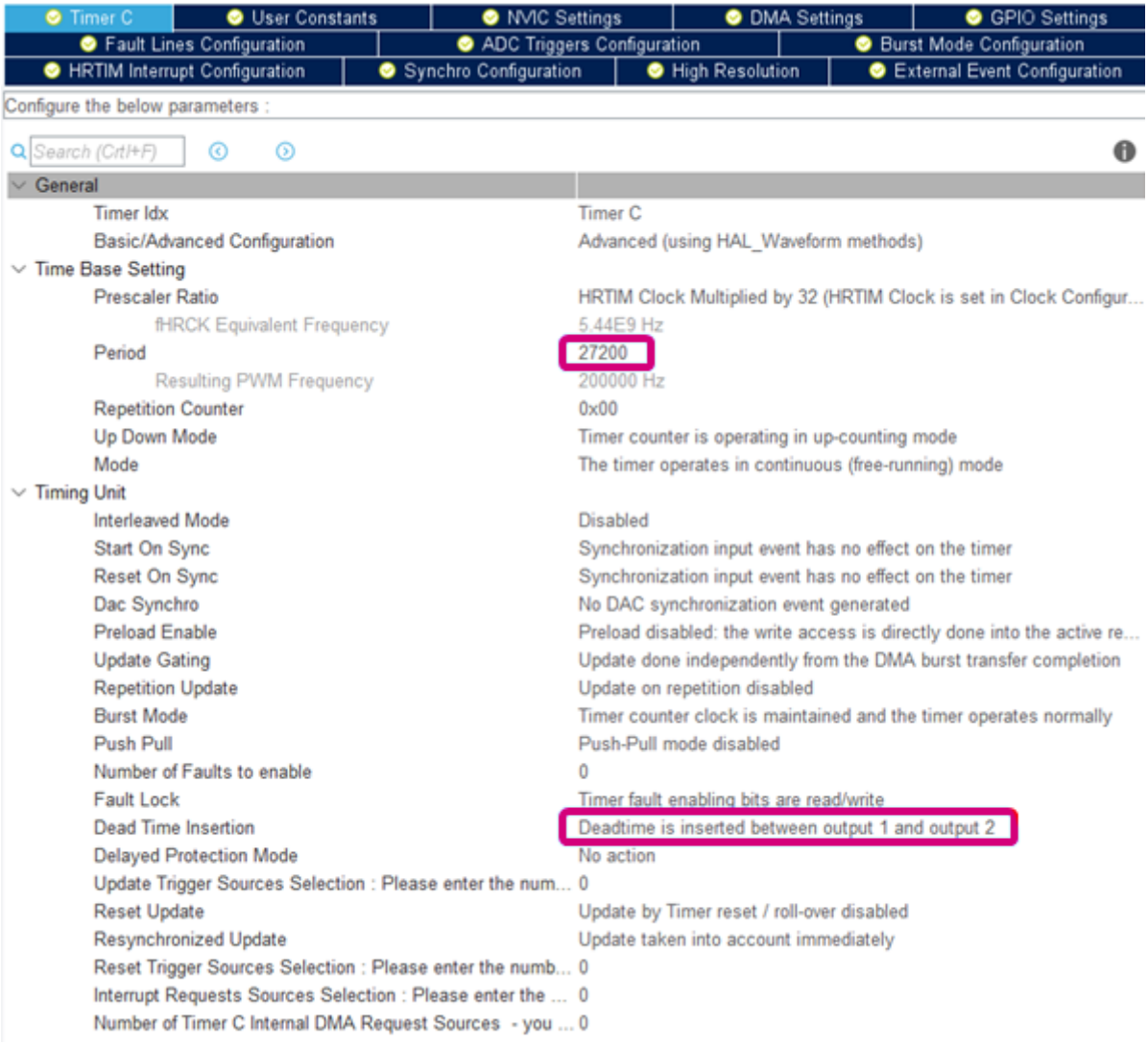
Figure 45. External event configuration window



Now click on the Timer C configuration tab. First, the Time Base Settings section must be configured. Set the period value to 27200, and the tool may automatically switch from hex to decimal when enter is pressed. The resulting PWM period may now be calculated as 200 000 Hz.

The next section on the Timer C configuration tab is the Timing Unit Setting section. Here, change the Dead Time insertion to Deadtime is inserted between output 1 and output 2. This enables the deadtime section which allows the outputs to be configured such that both high-side and low-side switches are never driven at the same time.

Figure 46. Timer C configuration tab - part 1



Further down the Timer C configuration tab are the sections for configuring the compare units. These compare units allow events to be configured based on a comparison between the individual compare unit and the timer counter. For this example, three compare units are configured as follows.

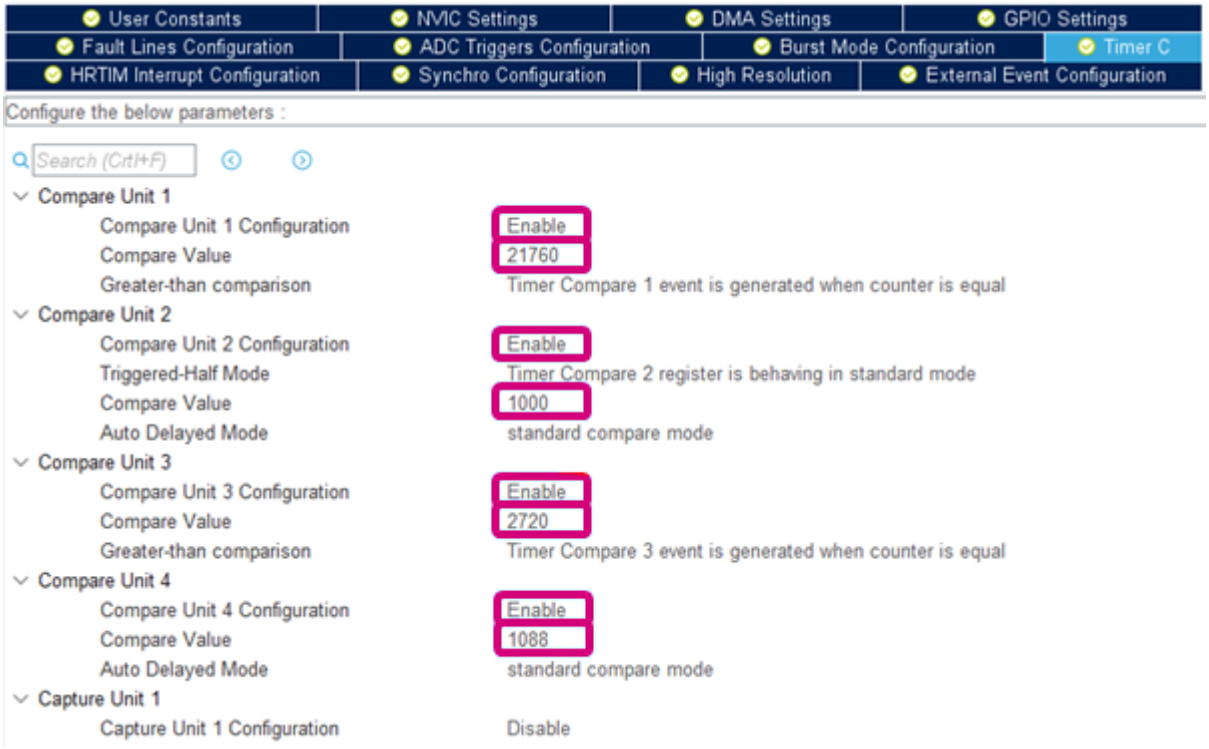
For Compare Unit 1 section change the configuration setting to Enable. Enter a Compare Value of 21760. This is the initial value of the duty cycle fixed at 80%. The compare event is used later on as a reset source to clear the output.

For Compare Unit 2 section change the configuration setting to Enable. Enter a Compare Value of 1000. This compare unit is used to trigger each step of the slope compensation sawtooth generator. The actual value is calculated during the run-time of the code.

For Compare Unit 3 section change the configuration setting to Enable. Enter a Compare Value of 2720. This is the blank window during which time any output from the peak current detection comparator is ignored. This equates to approximately 300 ns after the rising edge of the PWM, configured in the next step.

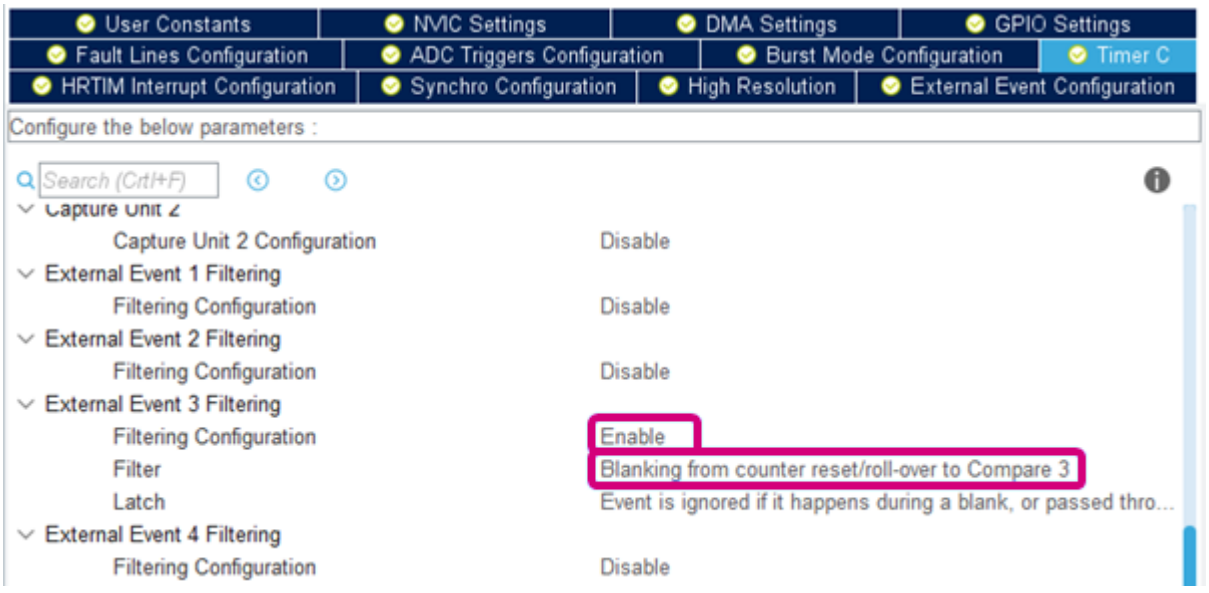
For `Compare Unit 4` section change the configuration setting to `Enable`. Enter a `Compare Value` of `1088`, approximately 200 ns after the start of the switching cycle. This is the event that is used to set the PWM output HIGH. A small delay is required from the beginning of the switching period as if the comparator trip event must be cleared before the output can be set HIGH by one of the set sources.

Figure 47. Timer C configuration tab - part 2



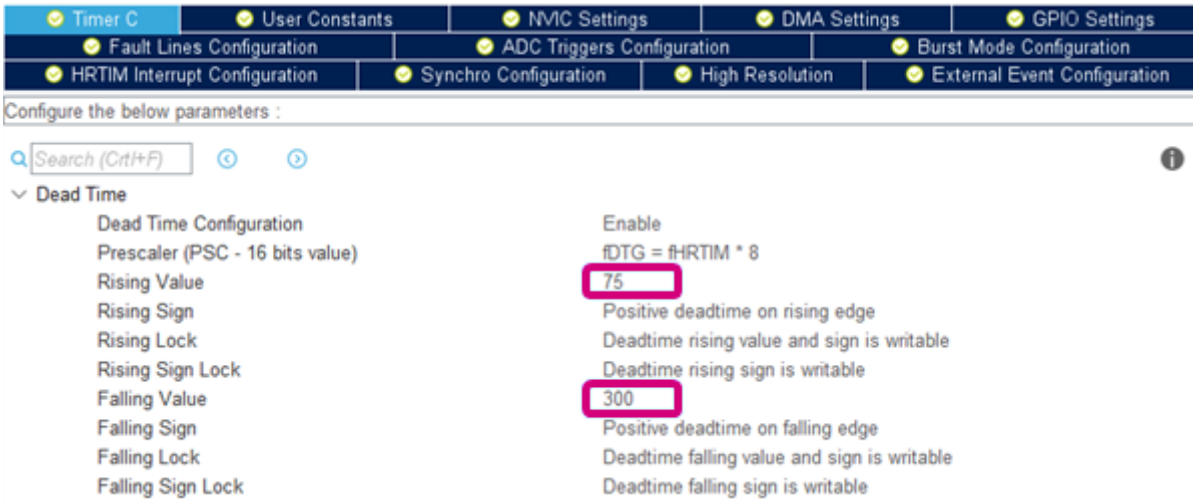
Further down the `Timer C` configuration tab is the `External Event 3 Filtering` section. Change the `Filtering Configuration` to `Enable`. Set the `Filter` to `Blanking from counter reset/roll-over to Compare 3`.

Figure 48. Timer C configuration tab - part 3



The next section to configure on the `Timer C` configuration tab is the `Dead Time` section. Locate this section and set the `Rising Value` to `75` and `Falling Value` to `300`.

Figure 49. Timer C configuration tab - part 4



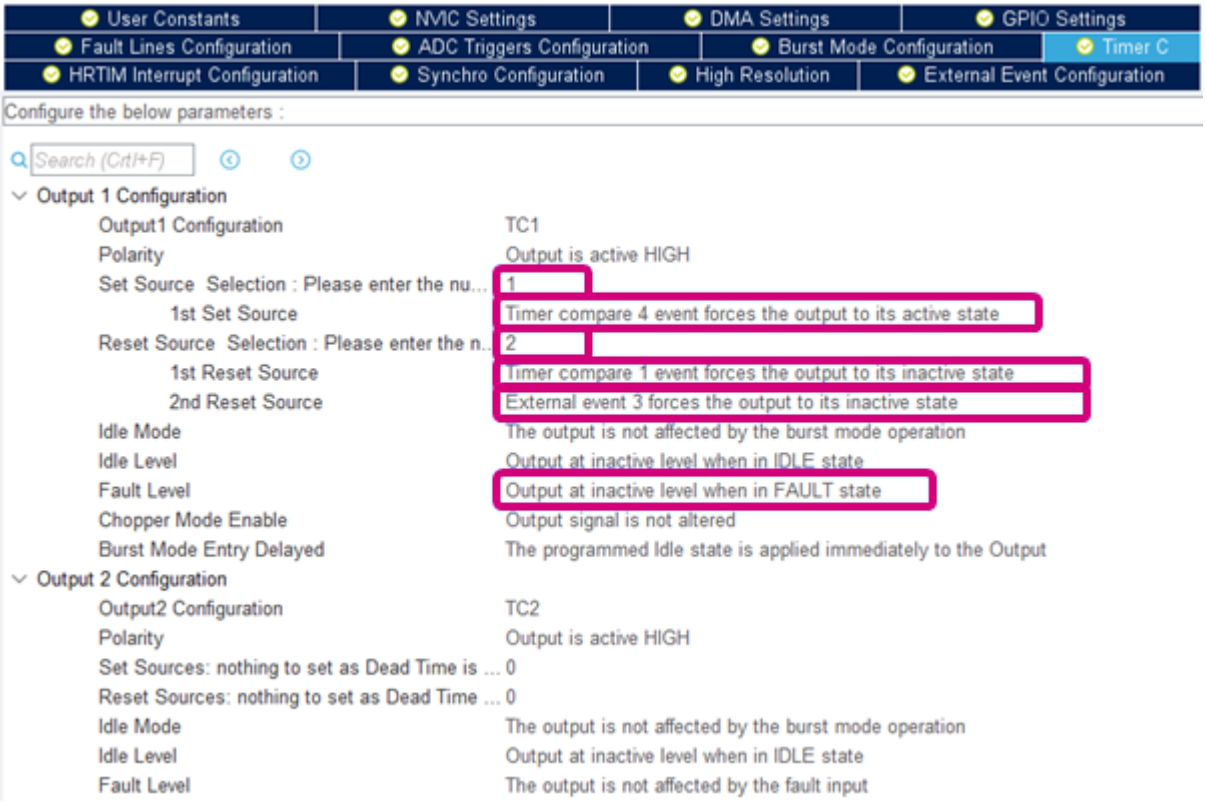
The last two sections to configure on the `Timer C` configuration tab are the `Output 1 Configuration` and `Output 2 Configuration` sections. These are located toward the bottom of the tab.

Under the `Output 1 Configuration` section, change the `Set Source Selection` number to `1`. Change the `1st Set Source` event to `Timer compare 4 event forces the output to its active state`.

Change the `Reset Source Selection` number to `2`. Change the `1st Reset Source` event to `Timer compare 1 event forces the output to its inactive state`. Then change the `2nd Reset Source` event to `External event 3 forces the output to its inactive state`.

Finally, set the `Fault Level` to `Output at inactive level when in FAULT state`.

Figure 50. Timer C configuration tab - part 5

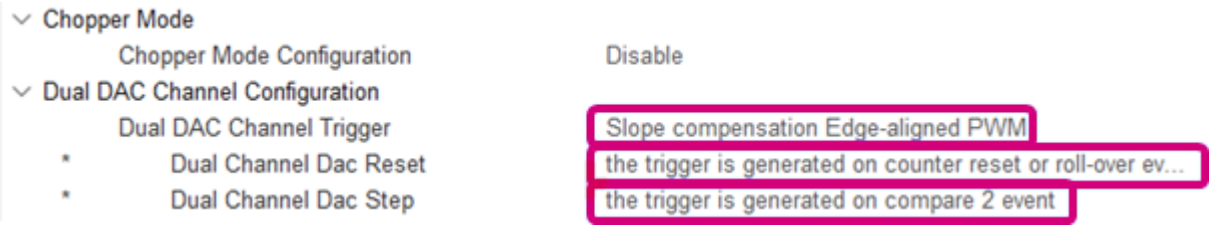


The `Output 2 Configuration` section is left unchanged as this output is driven by the dead-time configuration entered earlier.

Finally, at the very bottom of the `Timer C` configuration tab, under the `Dual DAC Channel Configuration` section set the `Dual DAC Channel Trigger` to `Slope compensation Edge-aligned PWM`.

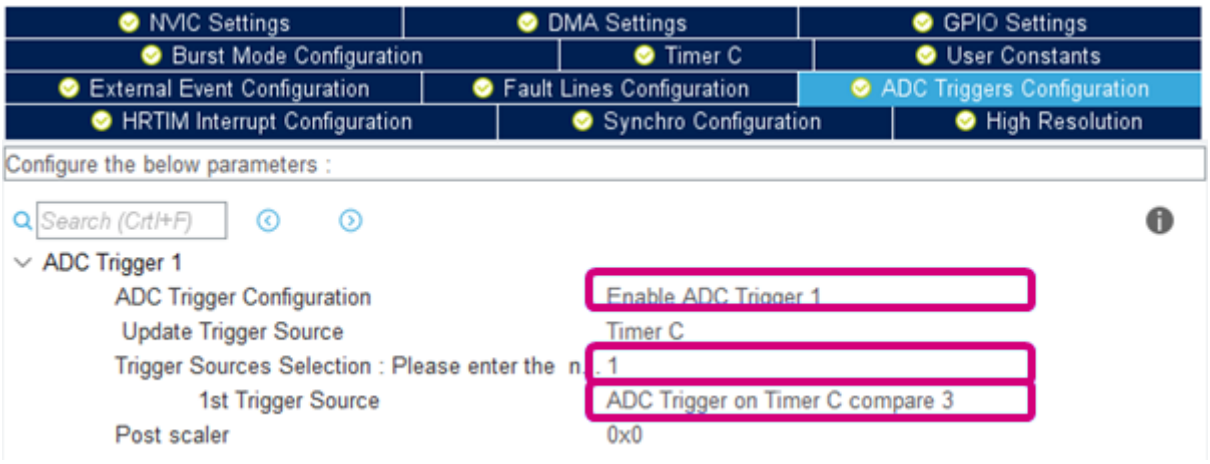
Here the event which resets the DAC value is configured, in this case, `the trigger is generated on counter reset or roll-over event`, and the event which triggers a step in the sawtooth waveform generator configured earlier is selected, in this case, `the trigger is generated on compare 2 event`.

Figure 51. Timer C configuration tab - part 6



Finally, the ADC trigger must be configured for the HRTIM1 module. Under `HRTIM1` select the `ADC Triggers Configuration` tab. Enable the `ADC Trigger 1` and set the `Update Trigger Source` to `Timer C`, and enter the number of `Trigger Sources Selection` to `1`. From the `1st Trigger Source` drop-down box, select `ADC Trigger on Timer C compare 3`. This is the event that triggers the ADC.

Figure 52. ADC triggers configuration



7.2.6 FMAC configuration

The last peripheral which requires configuration is that of the FMAC. As discussed earlier, the FMAC is used to execute the controller and compute the new value of the demanded peak-current reference for this peak-current mode buck converter. To enable the FMAC, expand the `Computing` category on the left-hand side of the window. Select `FMAC` and on the configuration pane tick the `Activated` box. Also, tick the box to enable the FMAC interruption under the `NVIC Settings` tab.

Figure 53. FMAC configuration window

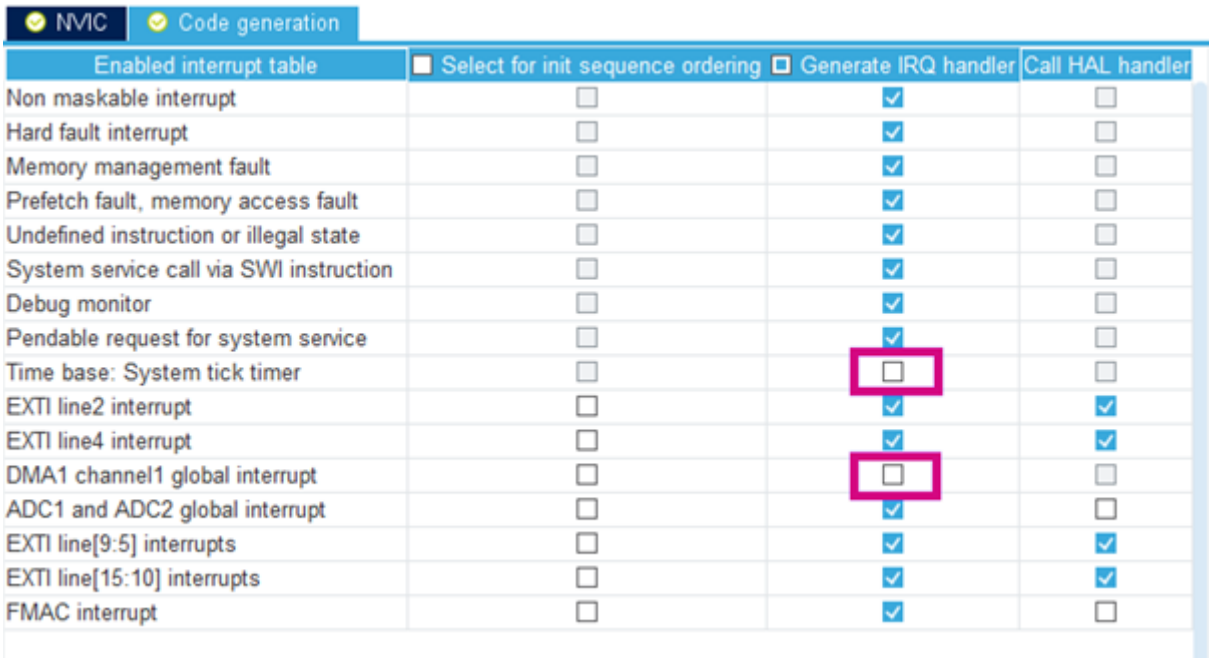
7.2.7 IRQ handler configuration

By default, STM32CubeMX creates interrupt handlers for the configured peripherals. These are created in a separate interrupt handler .c file within the project. If the user wishes to create his interrupt handler function then the automatic generation of the IRQ handler function must be disabled for that peripheral.

In this application note, the user writes his SysTick handler function, and therefore the automatic generation of this IRQ handler must be disabled. To do this click on the `NVIC` sub-category within the `System Core` category on the left-hand side of the window. Click on the `Code Generation` tab.

Untick the `Generate IRQ handler` checkbox for `Time base: System tick timer` as in Figure 54.

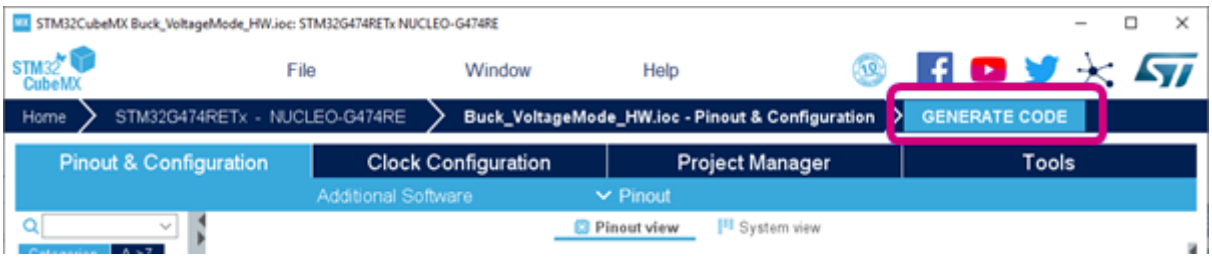
Figure 54. Automatic IRQ handler generation disabled



Within the code, the DMA interrupt is disabled, and therefore the IRQ handler for the DMA is also not needed and can be unticked.

Finally, generate the project by clicking the `GENERATE CODE` button. This creates the necessary project files for the selected IDE. In this example, IAR Embedded Workbench® from IAR Systems is used. However, there are multiple IDEs for which STM32CubeMX can generate project files, including the free use of STM32CubeIDE from STMicroelectronics.

Figure 55. Code generation launch



A prompt appears once the project files are created, which asks if the user likes to open the project. Click `Open Project` to load the project into the selected IDE.

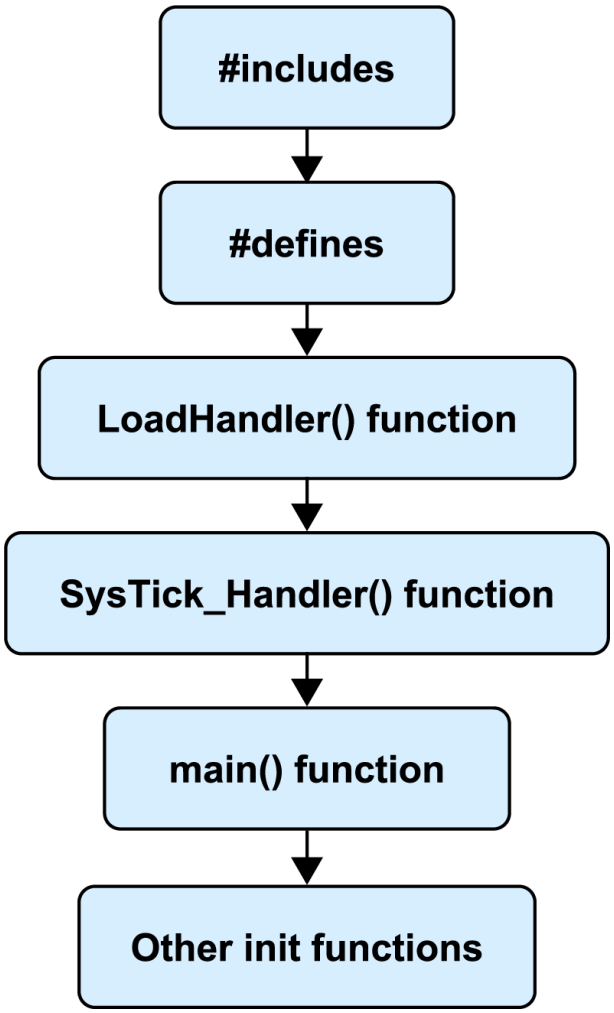
Figure 56. Project opening after code generation

The `main.c` file contains the code for this project, open the main.c file and read the comments.

7.3 Program flow description

The `main.c` file contains the majority of the code for setting up and initializing the peripherals. Some of the code within this file is automatically generated by the STM32CubeMX tool as per the configuration process in the previous step of this application note. The `main.c` file is structured as shown in Figure 57.

Figure 57. Program flow within main.c



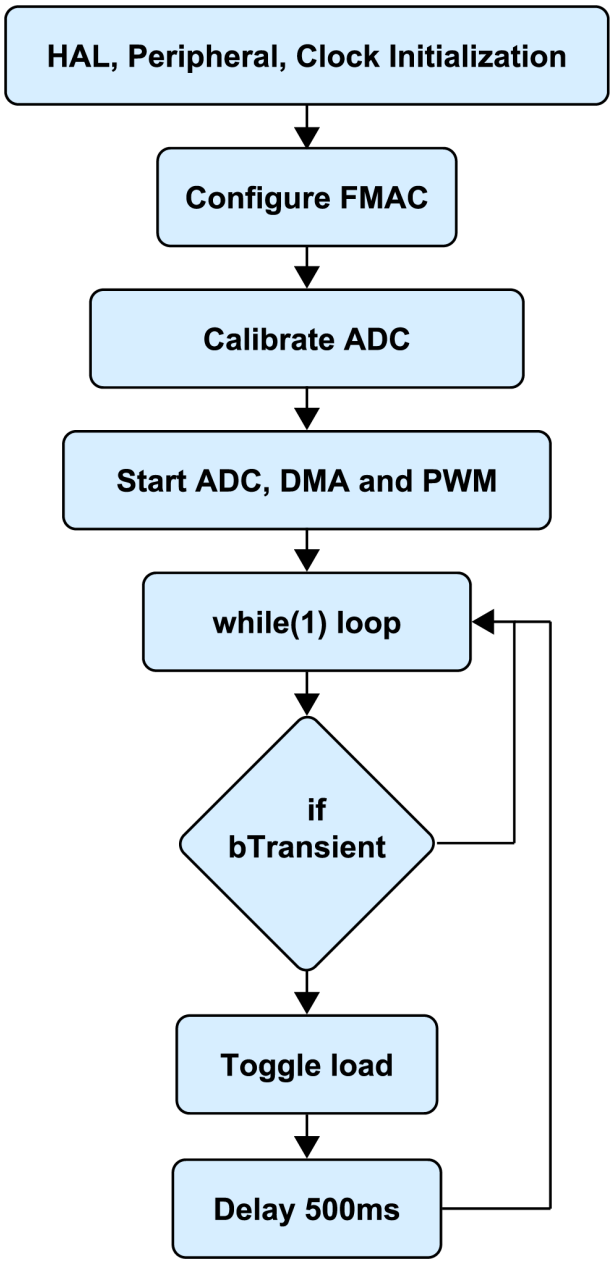
Within these code files, there are comments similar to `USER CODE BEGINS HERE` and `USER CODE ENDS HERE`. Any additional code to the project is kept between these two comments. This is because, outside of these comments, the code is automatically generated by the STM32CubeMX tool. If any changes are made to the project configuration within the tool, and the code is re-generated by clicking `GENERATE CODE`, the additional changes made outside of these comment bounds are lost. Keeping the user code within these comment bounds ensures that the code persists after the next time the code is re-generated using the STM32CubeMX tool. Figure 58 shows examples of the user code locations highlighted in fuchsia.

Figure 58. Example of user code locations within main.c

```
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */


  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_RTC_Init();
  MX_ADC1_Init();
  MX_HRTIM1_Init();
  MX_FMAC_Init();
  /* USER CODE BEGIN 2 */

    /*## Configure the FMAC peripheral ##########################################*/
  sFmacConfig.InputBaseAddress  = INPUT_BUFFER_BASE;
  sFmacConfig.InputBufferSize   = INPUT_BUFFER_SIZE;
  sFmacConfig.InputThreshold    = INPUT_THRESHOLD;
  sFmacConfig.CoeffBaseAddress  = COEFFICIENT_BUFFER_BASE;
```

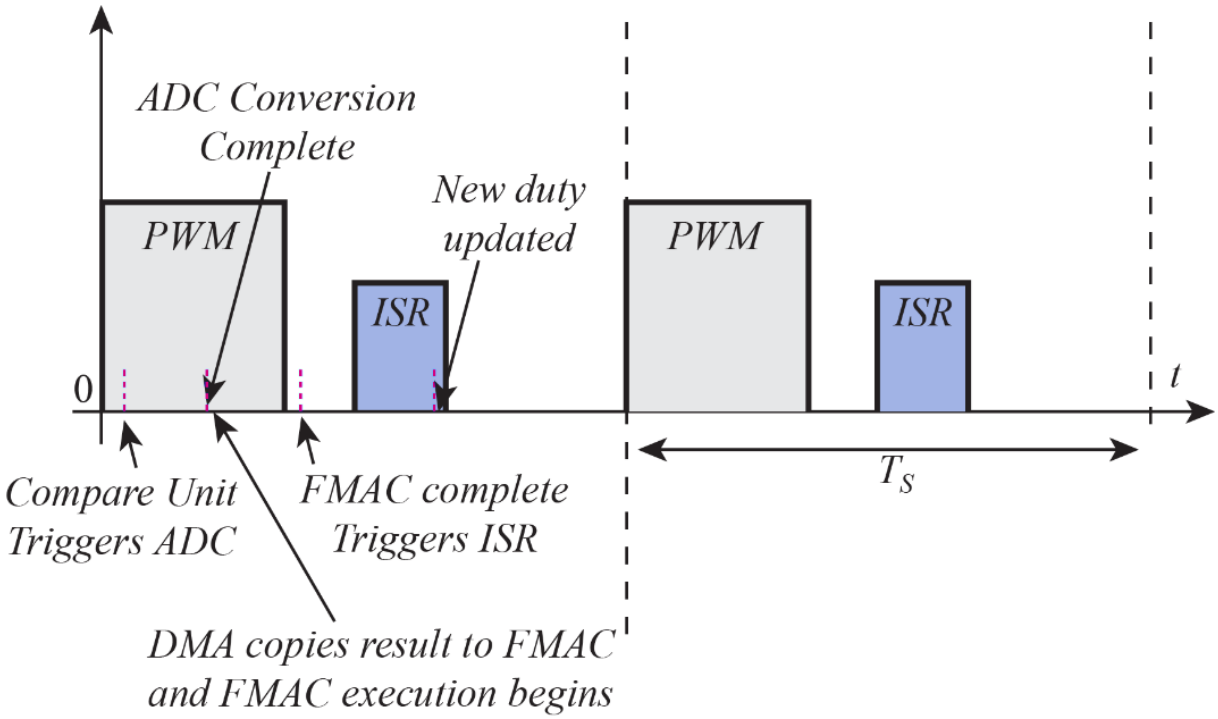
The structure of the main function within the `main.c` program is represented in Figure 59. This is the function that the MCU jumps to after power on – the entry point within the code. Therefore, this function contains calls to all of the initialization functions to set up the peripherals onboard the MCU. After this, the controller for the buck converter is set up and initialized with the calculated coefficients. Finally, the ADC is started and begins sampling the output voltage and the PWM outputs are enabled to drive the buck switches.

Figure 59. Function flow of main() within main.c



The sampling of the ADC module is triggered by the HRTIM module, in this case by comparing unit 3, which is set to several HRTIM ticks after the beginning of the switching period to avoid switching noise corrupting the sample. Once the sampling and conversion are complete, the ADC triggers the DMA to copy the result directly from the `ADC results` register to the FMAC input register. This process is depicted in Figure 60.

Figure 60. ADC, DMA, and FMAC timing diagram

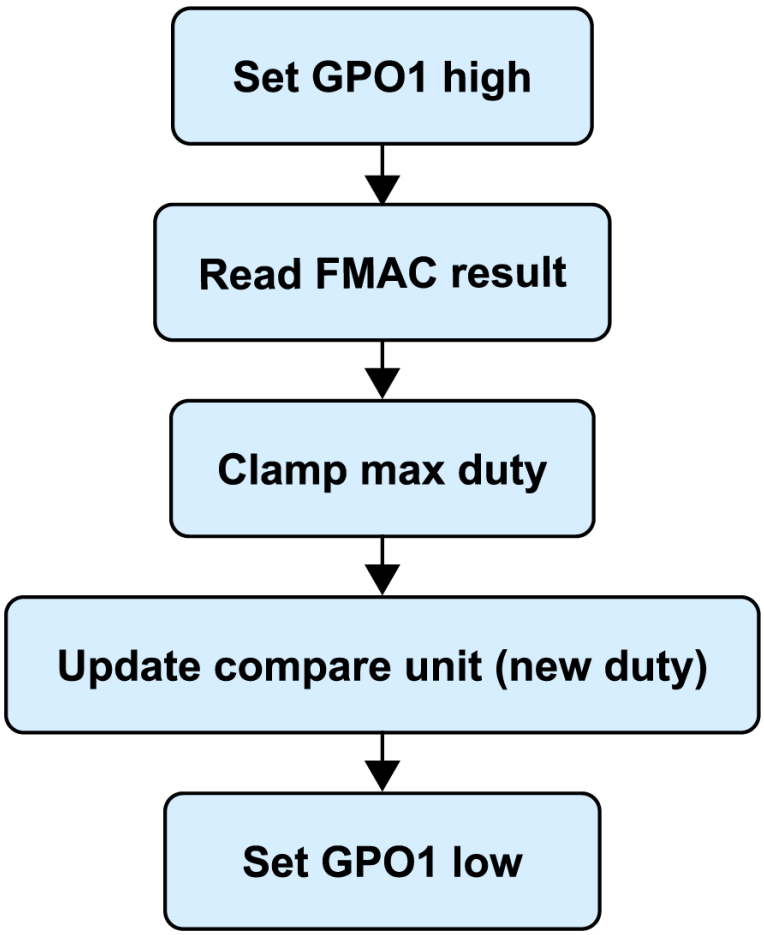


7.4 Interrupt service routine

After the FMAC completes the execution of the 2p2z controller, it triggers an interrupt. The interruption causes the MCU to jump from wherever it is, most likely sitting in the `while(1)` loop within `main()`, and execute the code within the FMAC ISR. The FMAC ISR is a separate function that is located towards the end of the `stm32g4xx_it.c` file included with this project. It has the function name `FMAC_IRQHandler(void)`.

The purpose of this FMAC ISR is to perform bounds checking on the output of the 2p2z controller executed using the FMAC. The output of the controller is the new demanded peak-current value and is written to the DAC register of the comparator module. However, the DAC register has a maximum value equal to the number of bits of the DAC. Therefore, the logic depicted in Figure 61 is implemented within the user code section of the FMAC ISR.

Figure 61. FMAC ISR flow



7.5 2p2z controller coefficients

The 2p2z controller coefficients are defined in the `main.h` header file. This can be easily accessed by locating the `#include "main.h"` towards the top of the `main.c` file, right-clicking on `main.h` and selecting `Open main.h` from the menu. Within this header file, there are several definitions for the pin names which are automatically generated by STM32CubeMX.

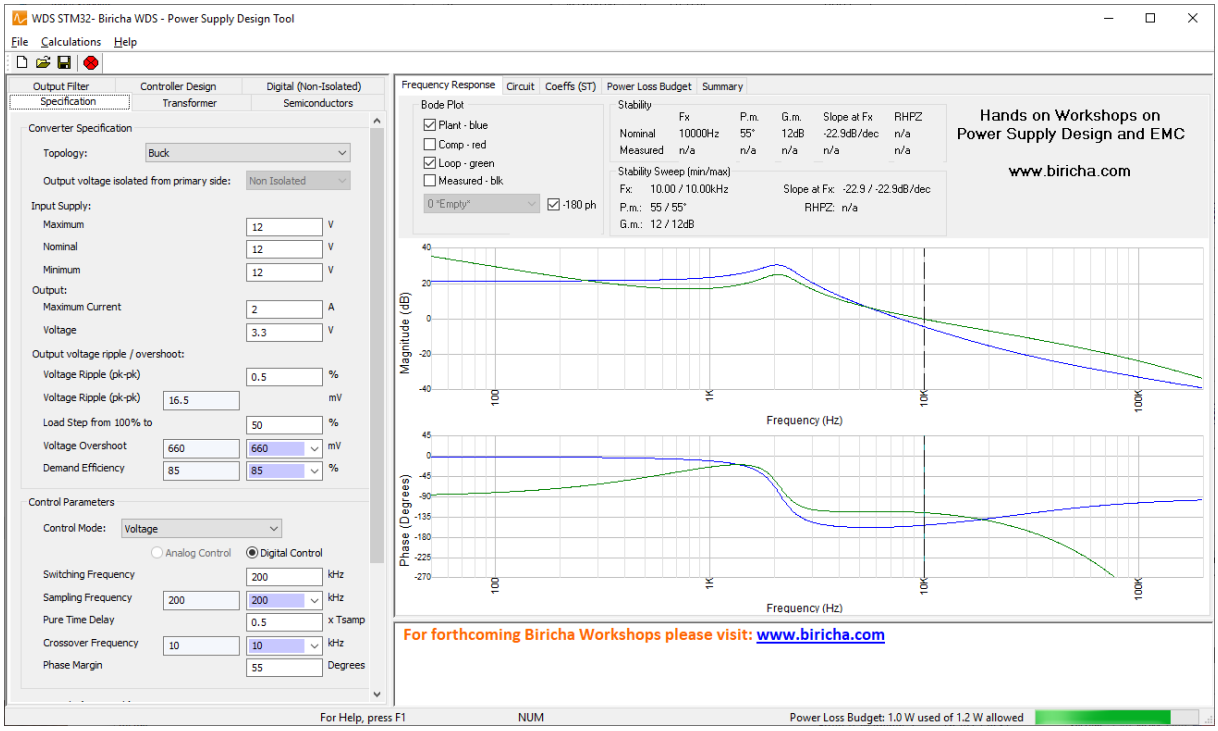
Further down this file, there is a `/* USER CODE BEGIN Private defines */` section where the definitions for the FMAC configuration begin. The controller coefficients are defined below the FMAC configuration parameters. These coefficients (B0, B1, B2, A1, A2) are given in a fixed-point hexadecimal form. Earlier in this application note, it is shown how the compensator poles and zeros, in the continuous-time domain, are converted into discrete-time controller coefficients. The last required step is to convert these discrete-time controller coefficients into fixed-point form for use on the fixed-point FMAC.

7.6 ST-WDS configuration

The power supply design tool ST-WDS from Biricha is used to generate the fixed-point controller coefficients for this controller implemented on the FMAC. ST-WDS is free-to-use and can be downloaded for free from the Biricha website. Visit *www.biricha.com/st-wds* for more information.

The `.wds` file associated with this application note is included in the project downloads in the appendix of this application note. However, the process for re-creating the WDS project settings is included here for completeness.

Figure 62. Biricha ST-WDS



The initial screen of ST-WDS is shown in Figure 62. The left-hand pane is used for entering the specification of the power supply along with the other pertinent design parameters. The right-hand pane displays the control loop Bode plots, schematic and outputs the required digital controller coefficients.

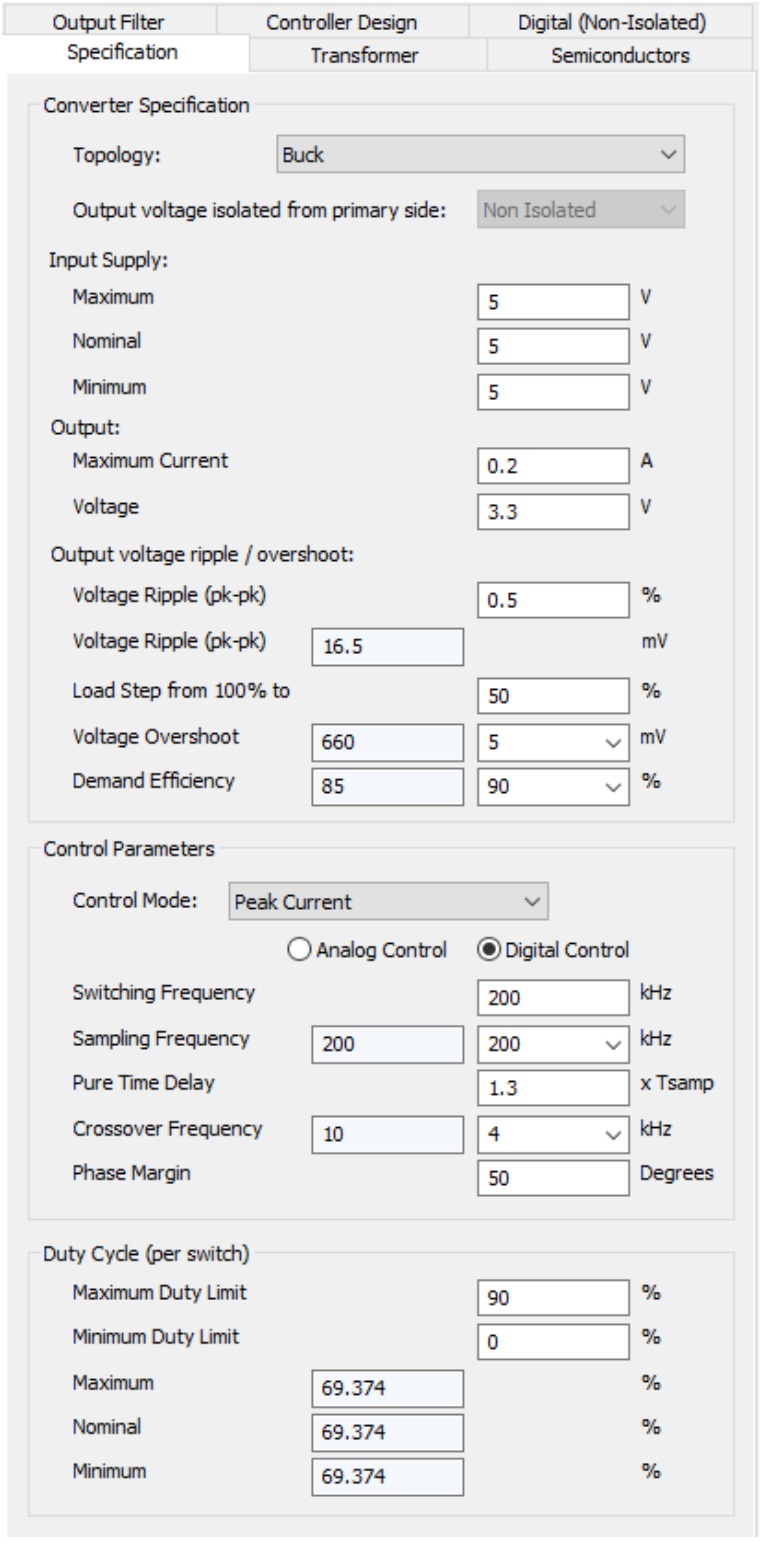
To calculate the digital controller coefficients the power supply specification must first be entered. On the `Specification` tab, enter the specification shown in Table 2.

Table 2. Discovery kit specification

Specification tab parameter	Value
Topology	Buck
Input supply max	5 V
Input supply nom	5 V
Input supply min	5 V
Output maximum current	0.2 A
Output voltage	3.3 V
Voltage ripple (peak-to-peak)	0.5%
Load step 100% to…	50%
Voltage overshoot	5 mV
Demand efficiency	92%
Control mode	Peak current, Digital control
Switching frequency	200 kHz
Pure time delay	1.3
Crossover frequency	4 kHz
Phase margin	50°
Maximum duty limit	90%
Minimum duty limit	0%

The `Specification` tab may now look like the screenshot shown in Figure 63.

Figure 63. ST-WDS specification tab



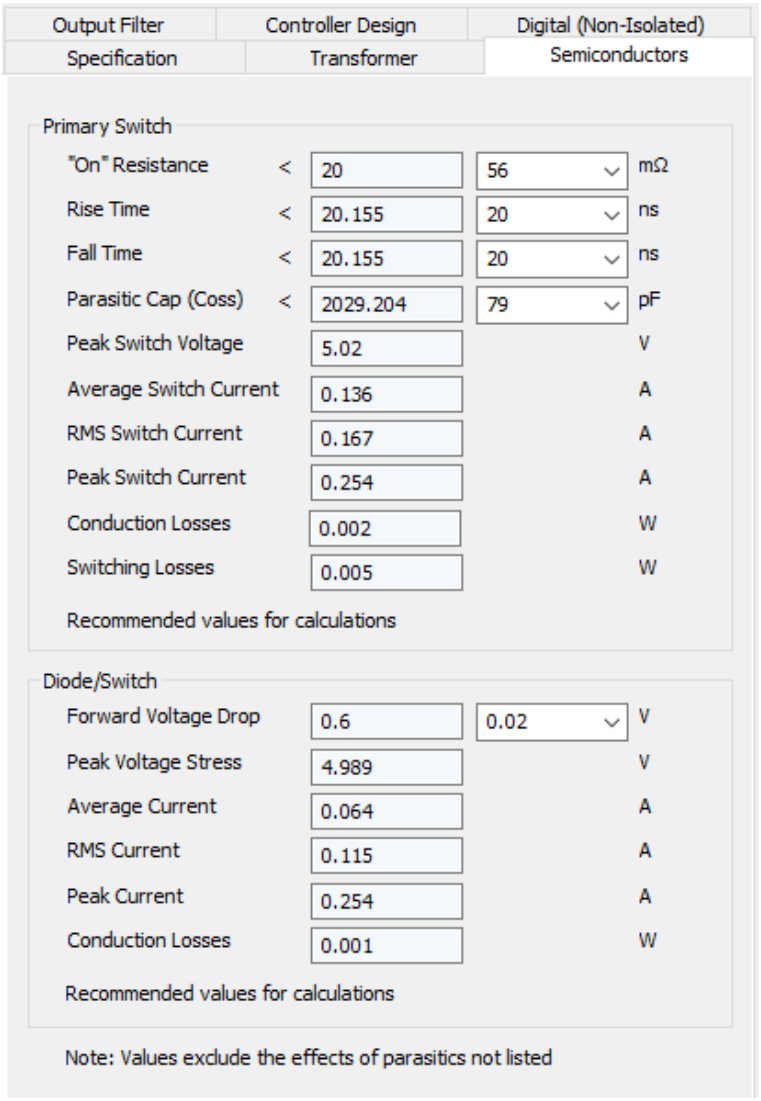
The Transformer tab is not used as this is a buck converter and this tab is only applicable for topologies which include a power stage transformer. Next, click on the Semiconductors tab and enter the specification shown in Table 3.

Table 3. Semiconductors tab parameter

Semiconductors tab parameter	Value
ON resistance	56 mΩ
Rise time	20 ns
Fall time	20 ns
Parasitic capacitance (Coss)	79 pF
Forward voltage drop	0.02 V

The Semiconductors tab may now look like the screenshot shown in Figure 64.

Figure 64. ST-WDS semiconductor tab



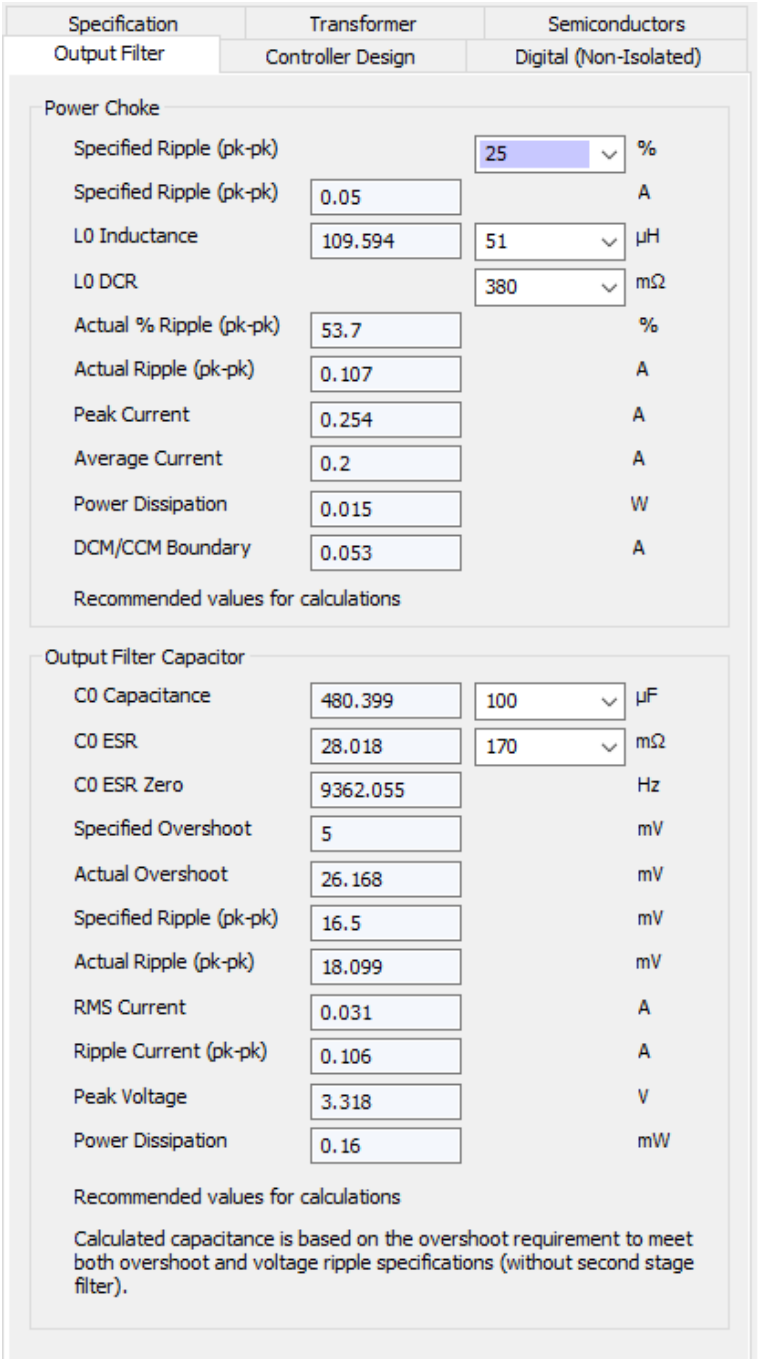
The `Output Filter` tab is next and has the specification shown in Table 4.

Table 4. Output filter parameters

Output filter tab parameter	Value
Specified peak-to-peak ripple	25%
L0 inductance	51 µH
L0 DCR	380 mΩ
C0 capacitance	100 µF
C0 ESR	170 mΩ

The `Output Filter` tab may now look like the screenshot shown in Figure 65.

Figure 65. ST-WDS output filter tab

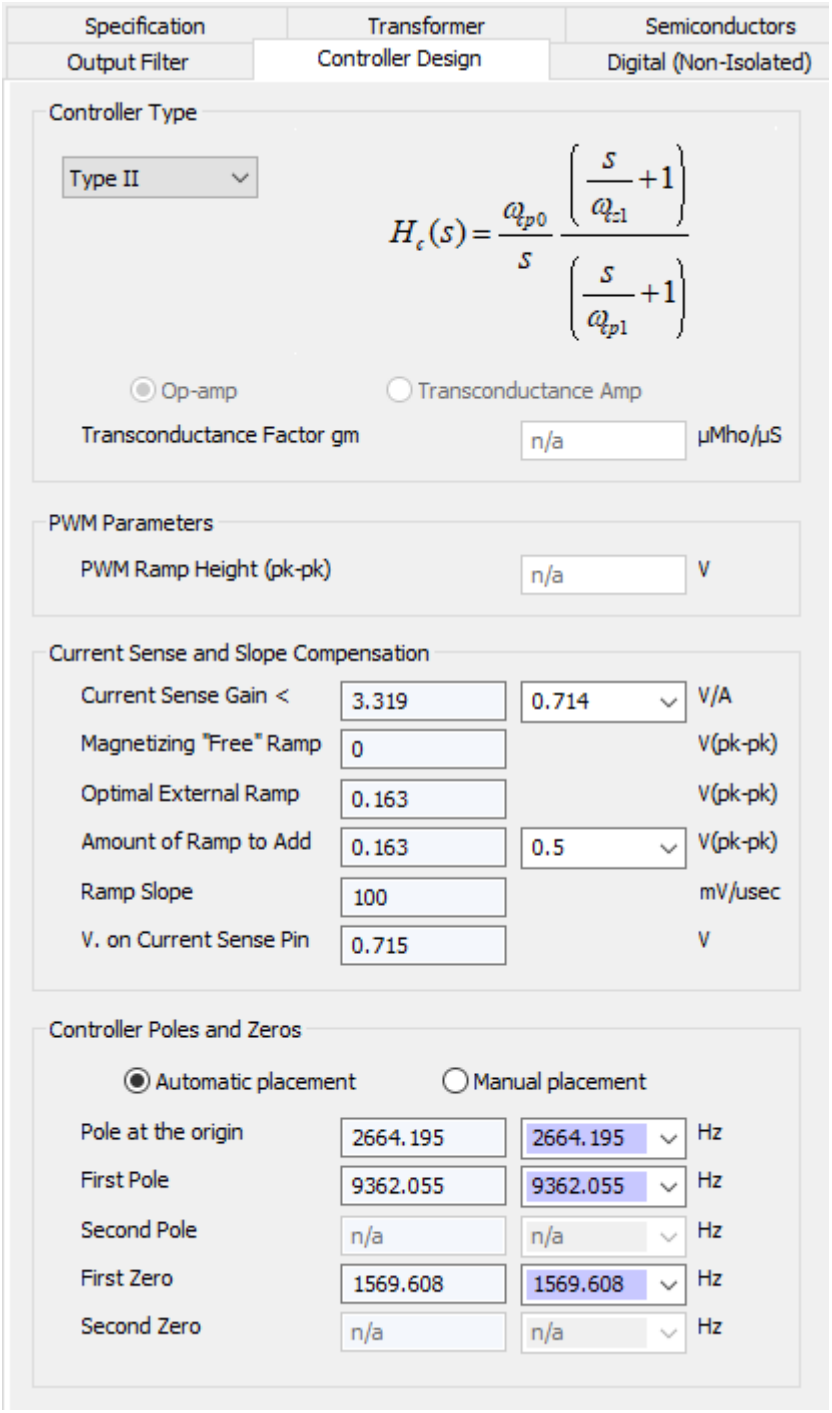


On the next tab, `Controller Design`, a Type II compensator is automatically designed by ST-WDS by placing the compensator poles and zeros to achieve the desired crossover frequency and phase margin. There is no need to enter any parameters into this tab as the poles and zeros are already calculated and may match those given in Table 5.

Table 5. Controller design parameters

Controller design parameter	Value
Current Sense Gain	0.714 V/A
Amount of Ramp to Add	0.5 V
Pole at the origin	2664.195 Hz (automatically calculated)
First Pole	9362.055 Hz (automatically calculated)
First Zero	1569.608 Hz (automatically calculated)

Figure 66. **ST-WDS controller design tab**

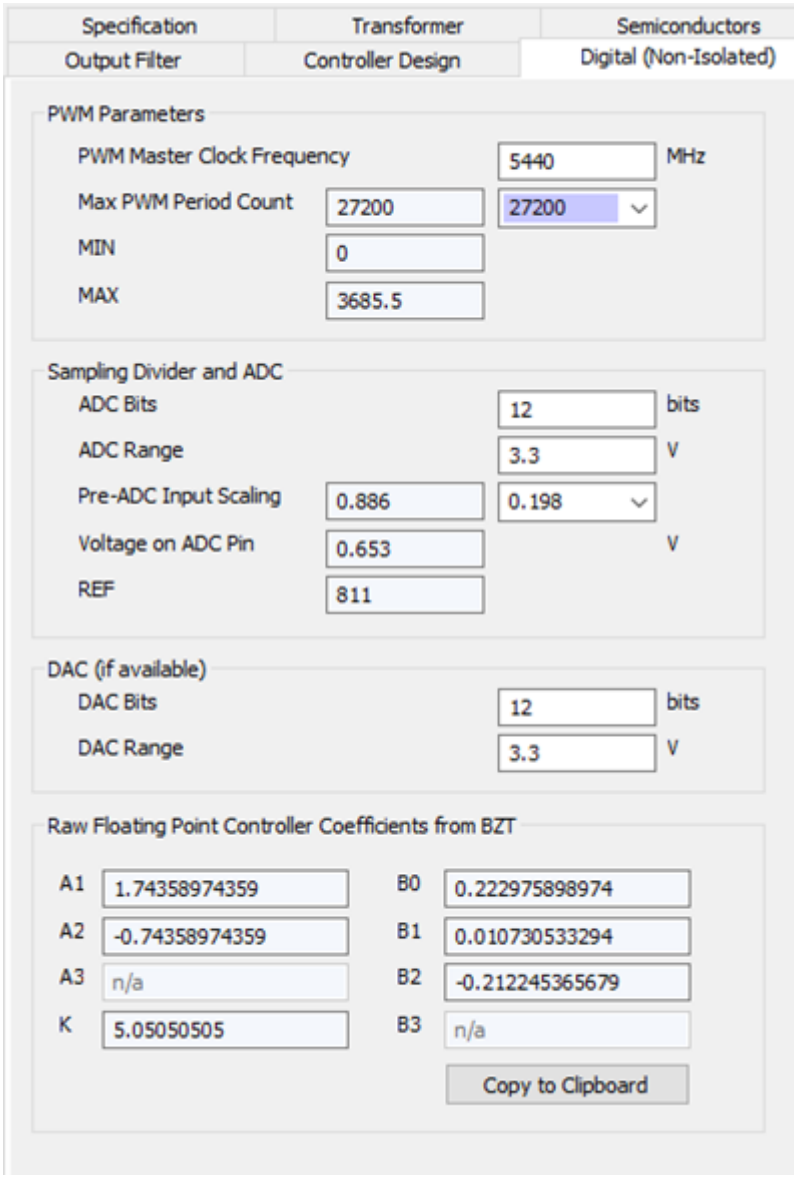


The last configuration tab is `Digital (Non-Isolated)`. Enter the specification given in Table 6 into this tab.

Table 6. Digital non-isolated parameters

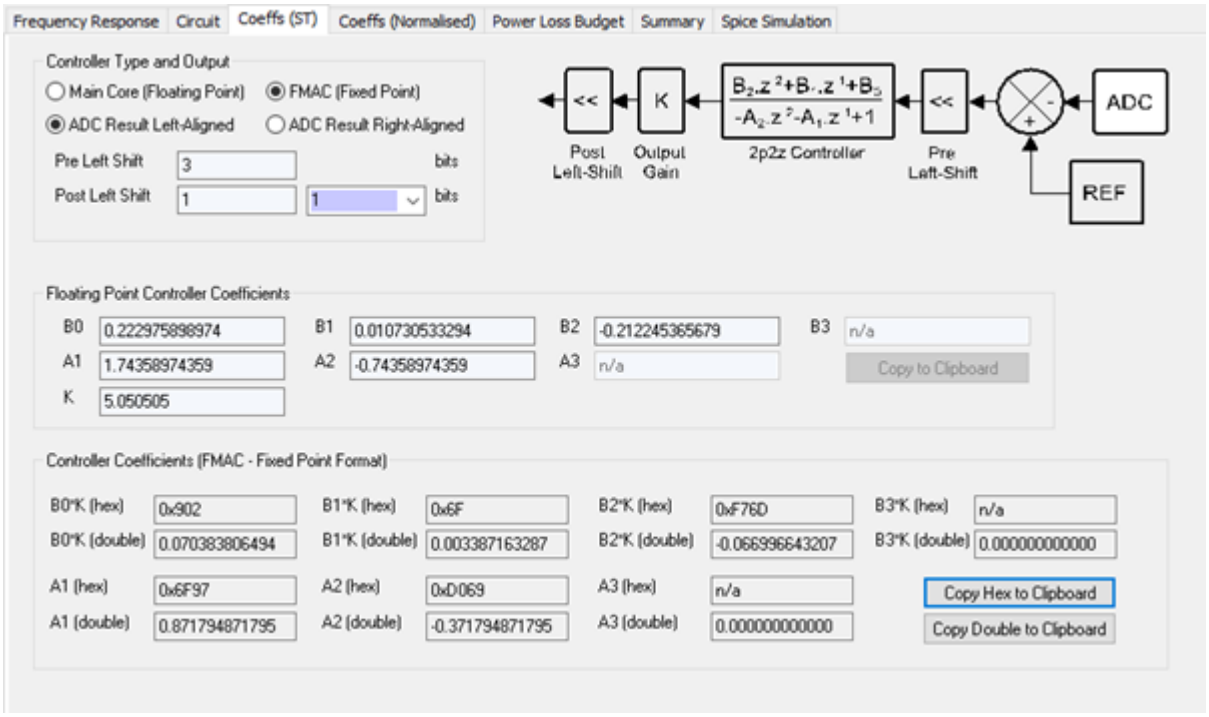
Digital non-isolated tab parameter	Value
PWM master clock frequency	5440 MHz
Maximum PWM period count	27200
ADC bits	12 bits
ADC range	3.3 V
Pre-ADC input scaling	0.198
DAC bits	12
DAC range	3.3 V

Figure 67. ST-WDS digital non-isolated tab



Now that the controller coefficients are calculated, they need to be converted into the format required for use with the fixed point FMAC. To do this within ST-WDS, click on the `Coeffs (ST)` tab on the right-hand pane. Here, set the `Controller Type and Output` setting to `FMAC (Fixed Point)`. Earlier within the STM32CubeMX configuration, the ADC result is configured to be left-aligned. Therefore, click on the `ADC Result Left-Aligned` radio button within ST-WDS. The pane may now look like Figure 68.

Figure 68. Fixed point controller coefficient calculation in ST-WDS



The fixed-point coefficients are now displayed at the bottom of the window. It is now possible to copy these to the clipboard for use within the code. Click the `Copy Hex to Clipboard` button. The following coefficients may now be on the clipboard:

```
#define B0 (0x902)
#define B1 (0x6F)
#define B2 (0xF76D)
#define A1 (0x6F97)
#define A2 (0xD069)
#define pre_shift (+3)
#define post_shift (+1)
#define REF (811)
#define DUTY_TICKS_MIN (0)
#define DUTY_TICKS_MAX (3686)
#define SLOPE_VPP (0.5000)
#define DECVAL (0.3650)
```

These are the coefficients that are used within the example application.

7.7 CCM-SRAM usage

The CCM-SRAM is an area of memory that is tightly coupled to the Arm® Cortex® core. This allows the core to execute the code at the maximum clock rate without any wait-states as typically found when execution from the Flash memory.

This functionality is ideal for routines that are time-critical such as the control loop implementation discussed in this application note. The STM32G474RE device featured on the Discovery kit contains 32 Kbytes of CCM-SRAM that can also be accessed via the DMA.

To use the CCM-SRAM area of memory, the memory areas must be defined in the linker files and the code must be copied from the Flash memory to the CCM-SRAM area at program startup. Therefore, there are several steps to follow to achieve this. The application note *Use STM32F3/STM32G4 CCM SRAM with IAR™ EWARM, Keil® MDK-ARM and GNU-based toolchains* (AN4296) contains step-by-step instructions for implementing this functionality using the different IDEs.

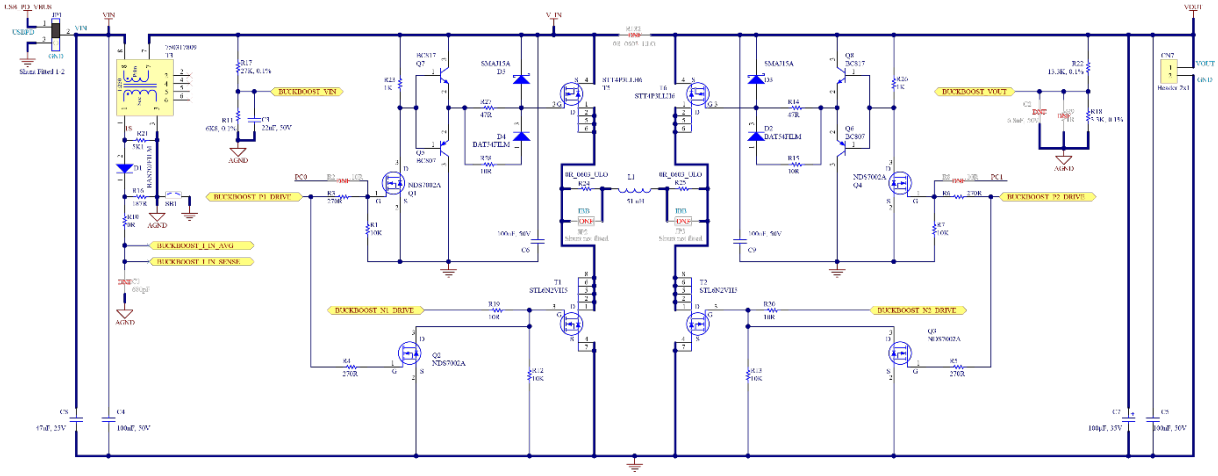
8 Design example

8.1 Power stage component selection

The Discovery kit is designed to allow the user to explore the many features that the STM32G4 Series has to offer. This section focuses on the design of the buck-converter power stage and presents the design equations used to select the appropriate components.

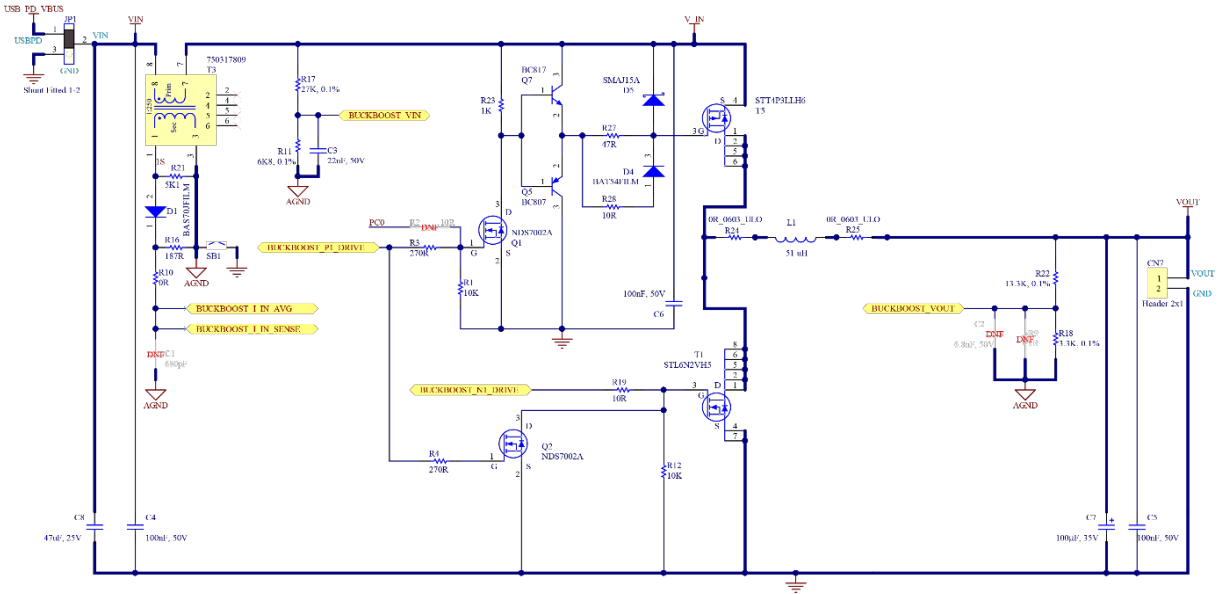
The buck converter is implemented on this Discovery kit as part of a buck-boost converter. It means that there are boost switches as well as buck-switching FETs. The extract from the schematic shown in Figure 69 identifies the relevant switches. The full schematic can be downloaded from B-G474E-DPOW1

Figure 69. Discovery kit schematic: buck-boost power stage



When using the converter in buck mode, the boost switches are not driven (the high-side switch is ON and the low-side switch is OFF) and the simplified circuit shown in Figure 70 can be used to describe the power stage.

Figure 70. Buck-converter simplified power stage



There are some constraints on the choice of the power stage components as this power stage is used for both buck and boost applications. In this design example, the buck specification given in Table 7 is considered. The power stage can also be designed using the ST-WDS Power Supply Design Tool from Biricha, however, the equations for selecting the power inductor and filter capacitor are presented here for completeness.

Table 7. Power stage parameters

Specification	Value
Input voltage	5 V
Output voltage	3.3 V
Output current (I_O)	0.2 A
Ripple current ($\Delta I_{L\%}$)	50%
Output voltage ripple	1% peak-to-peak
Switching frequency	200 kHz

First, the steady-state duty cycle is calculated using (47). This does not take into account any of the parasitics.

$$D = \frac{V_{OUT}}{V_{IN}} = \frac{3.3\,V}{5\,V} = 66\,\% \tag{47}$$

8.1.1 Power inductor

The power stage inductor is sized based on the maximum allowable ripple current at the switching frequency given the input and output voltage specification. Using the standard equation for the voltage rise on an inductor given in (48) rearranged for inductance, the inductance required to meet the specification can be calculated using (49).

$$V = L \cdot \frac{dI}{dt} \tag{48}$$

$$L_0 = \frac{D \cdot T_S \cdot (V_{IN} - V_{OUT})}{\Delta I_{L\,\%} \cdot I_O} \tag{49}$$

Where T_S is the switching period. Therefore:

$$L_0 = \frac{66\,\% \cdot 5\,\mu s \cdot (5\,V - 3.3\,V)}{50\,\% \cdot 0.2\,A} = 56\,\mu H \tag{50}$$

The REDEXPERT online tool from Würth Elektronik can be used to identify a suitable inductor. For this power stage, a 51 µH inductor is selected.

8.1.2 Output filter capacitor

The output filter capacitor is sized based on either the output voltage ripple requirement or the transient load step requirement, whichever requires the larger value of capacitance. The majority of the voltage ripple at the switching frequency on the output voltage of the buck converter is due to the voltage generated across the parasitic equivalent series resistance (ESR) of the electrolytic capacitor used in the output filter. The maximum value of the ESR can be calculated given the voltage ripple requirement and the known current ripple.

$$R_{ESR(\max)} = \frac{V_{RIPPLE}}{\Delta I_L} \tag{51}$$

$$R_{ESR(\max)} = \frac{V_{OUT} \cdot \cdot V_{RIPPLE\,\%} \cdot L_0}{D \cdot T_S \cdot (V_{IN} - V_{OUT})} \tag{52}$$

$$R_{ESR(\max)} = \frac{3.3\,V \cdot 1\,\% \cdot 51\,\mu H}{66\,\% \cdot 5\,\mu s. \cdot (5\,V - 3.3\,V)} = 0.33\,\Omega \tag{53}$$

For a given series of electrolytic capacitors, the product of the capacitance and ESR is relatively constant. Therefore, given the maximum ESR value calculated previously, the required capacitance can be calculated. For the WCAP-ASLL aluminum electrolytic capacitors from Würth Elektronik, this constant is around 1.7x10^{-5}, therefore the capacitance can be calculated in (54).

$$C_{0(\min)} = \frac{1.7 \cdot 10^{-5}}{R_{ESR(\max)}} \tag{54}$$

$$C_{0(\min)} = \frac{1.7 \cdot 10^{-5}}{0.33} = 51\,\mu F \tag{55}$$

As this is the minimum value of output capacitance, a capacitor with 100 µF is selected.

8.2 PCB layout

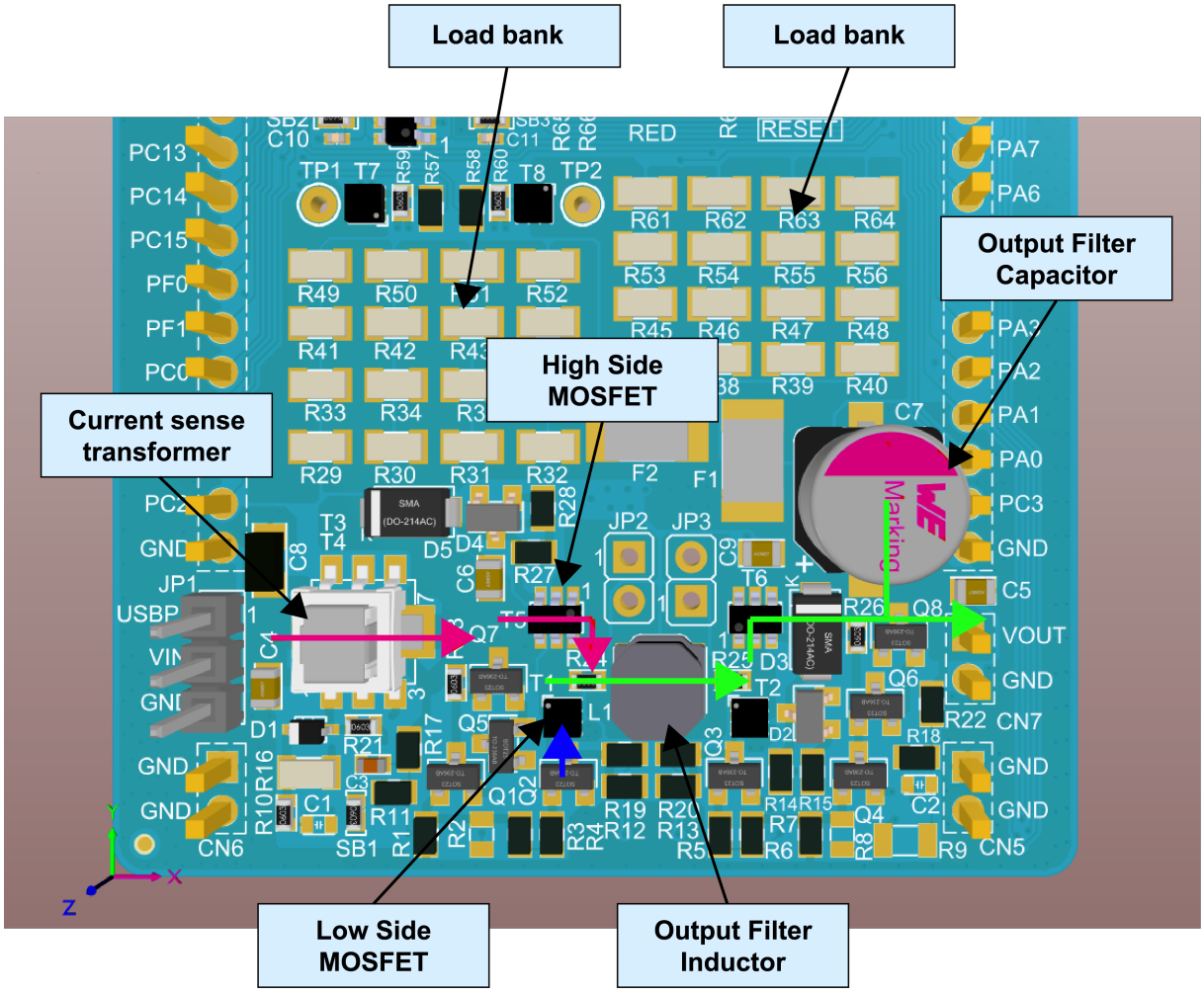
The layout of the buck converter is shown in Figure 71. In this figure, various key components of the buck converter are identified. When laying out the buck converter PCB, it is important to consider good PCB design practices. For example, loops with high di/dt must be minimized. Although this is a very low-power design, switching noise can still couple onto critical traces such as those routed to the ADC.

In this design, resistors R18 and R22 (towards the lower right-hand side of the buck converter) form the resistive divider which scales the output voltage of the buck converter before it is fed to the ADC. There may be an anti-aliasing filter on the ADC input to remove frequencies above the Nyquist frequency. This can be formed by placing a capacitor in parallel with R18. On this PCB there is a footprint for this, C2, however, no capacitor is placed. A potential improvement may be to add this capacitor. This may also help reduce any high-frequency switching noise which can be picked up by this trace.

The ADC pin for this output voltage feedback is PA3 which is a short distance away from this divider with no high dv/dt traces crossing this trace and a full unbroken ground plane for the return current, therefore, the signal may already be clean.

In Figure 71 the current path of the buck converter is shown as a series of colored arrows. The fushia arrow indicates the current through the current sense transformer and high-side MOSFET. During the off-time of the high-side switch, the current flows through the low-side MOSFET, which is indicated by a blue arrow. The output filter inductor and capacitor are in the output current path highlighted by the green arrows.

Figure 71. Hardware layout of the power stage



9 Getting started

9.1 Overall usage

The STM32 MCU onboard the Discovery kit comes pre-flashed with example software. This example software exercises the other functions of the Discovery kit such as control of the RGB LED. To run the buck converter, the user must first compile the project associated with this application note and flash the MCU.

The following dependencies are required to do this:

- PC with Windows® 7 or later
- STM32 compatible IDE, such as STM32CubeIDE, IAR Embedded Workbench®, or Keil® µVision
- STM32CubeMX (from v5.6.0) together with STM32Cube firmware library for G4 (from v1.2.0) installed

To get started simply:

1. Connect the micro-USB cable from the PC to CN3 on the Discovery kit.
2. Apply power via the USB-C and connect this to CN2 on the Discovery kit.
3. Ensure that the jumper JP1 is in the USB PD-VIN position.

9.2 Loading the project

This peak-current-mode buck example project, as well as the associated ST-WDS design files, are available from the Biricha website:

www.biricha.com/ST-Discovery-Kit

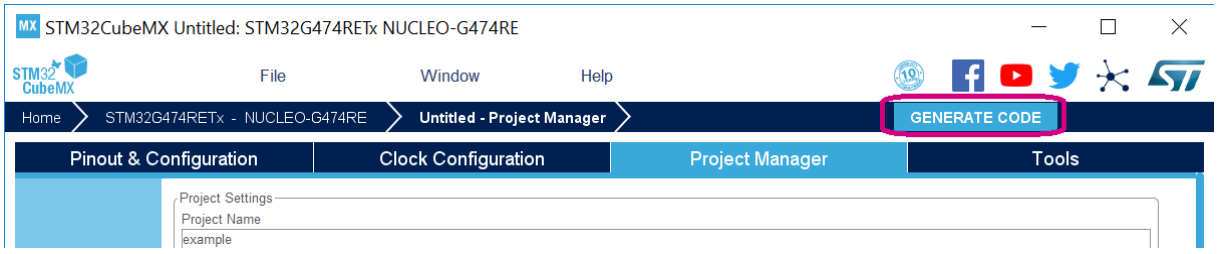
Opening the Project

1. Open STM32CubeMX by clicking on the icon shown in Figure 72 (note that the icon and the path may differ slightly).

Figure 72. Example.ico

2. Generate the project by clicking the `GENERATE CODE` button within STM32CubeMX.

Figure 73. Code generation



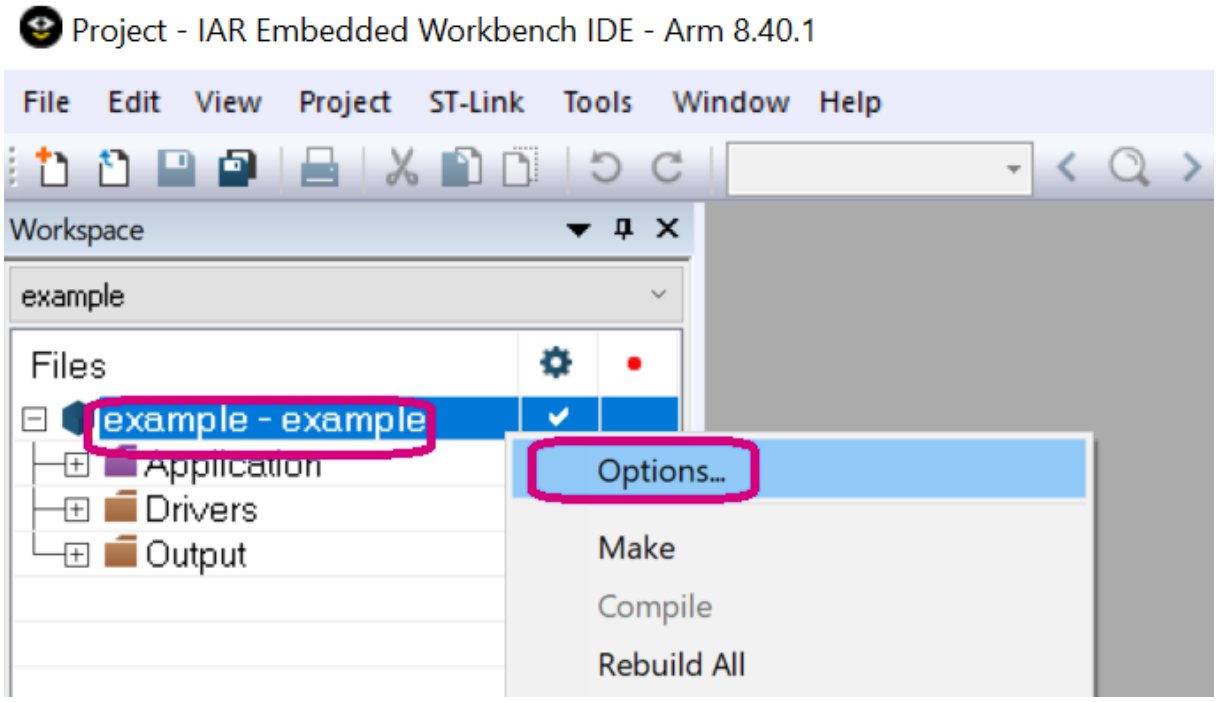
3. This generates the project for the selected IDE. In this case, it is an IAR Systems® project.

4. When prompted with the dialog box, open the project by clicking `Open Project`.

Figure 74. Project opening

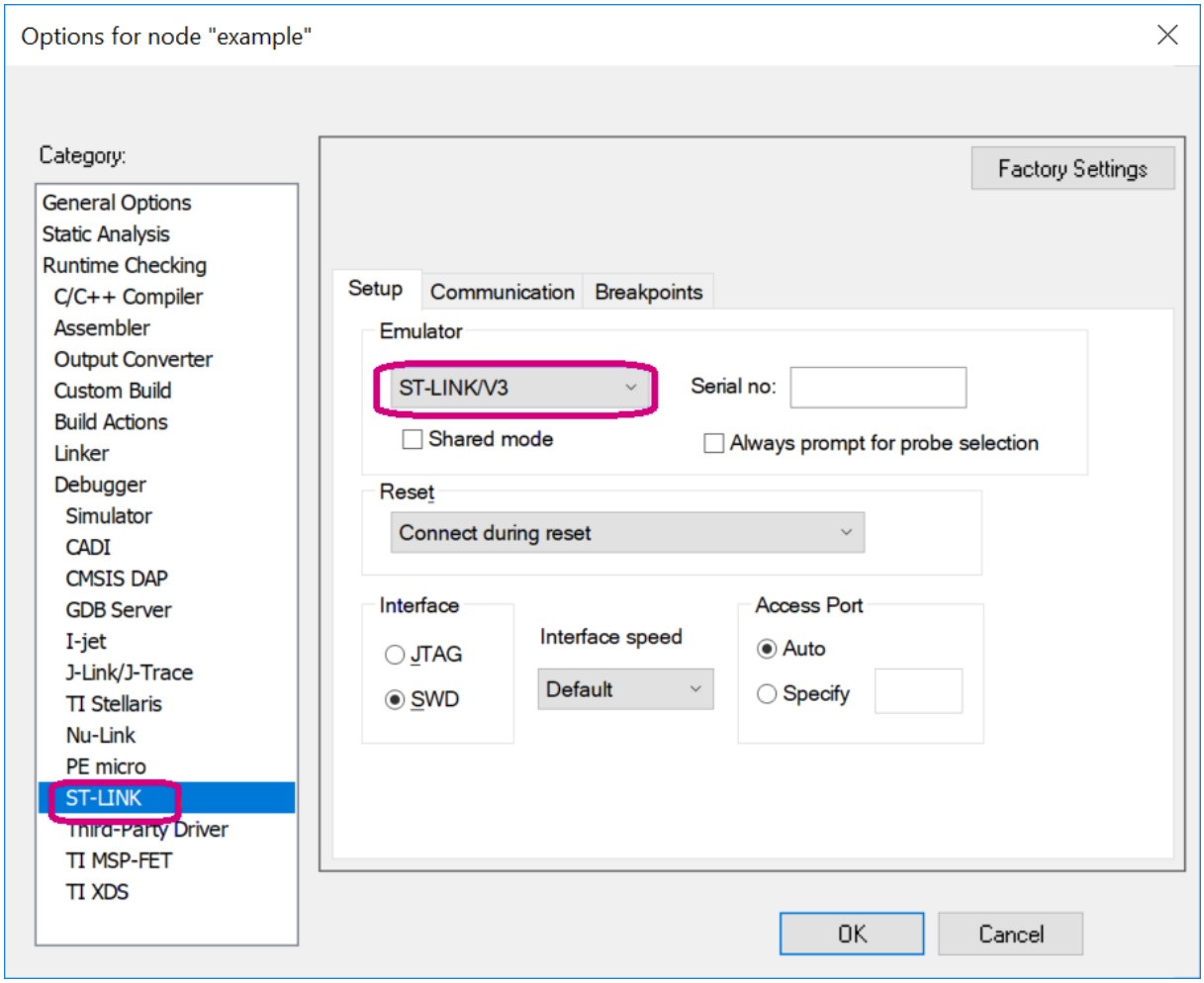
5. Depending on the IAR Embedded Workbench® used version, the default debugger may need to be changed to ST-LINK/V3. Right-click on the project name and select `Options`.

Figure 75. Example options



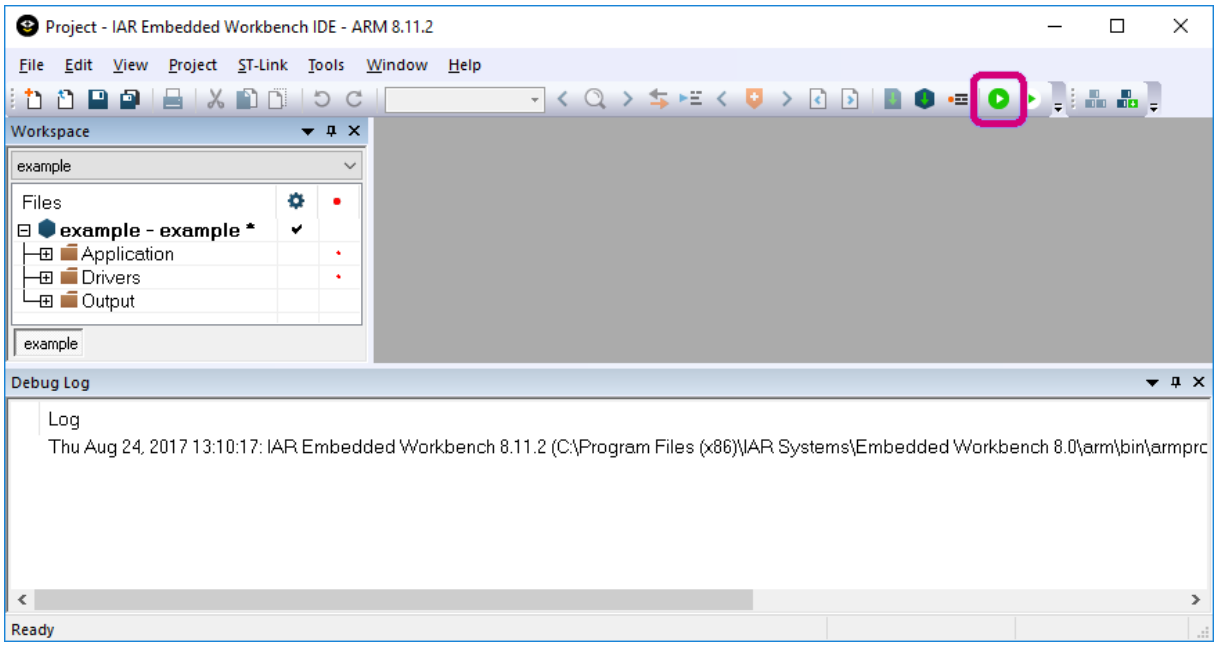
Then select `ST-LINK` and choose `ST-LINK/V3`. Click `OK` to finish.

Figure 76. **ST-LINK version selection**



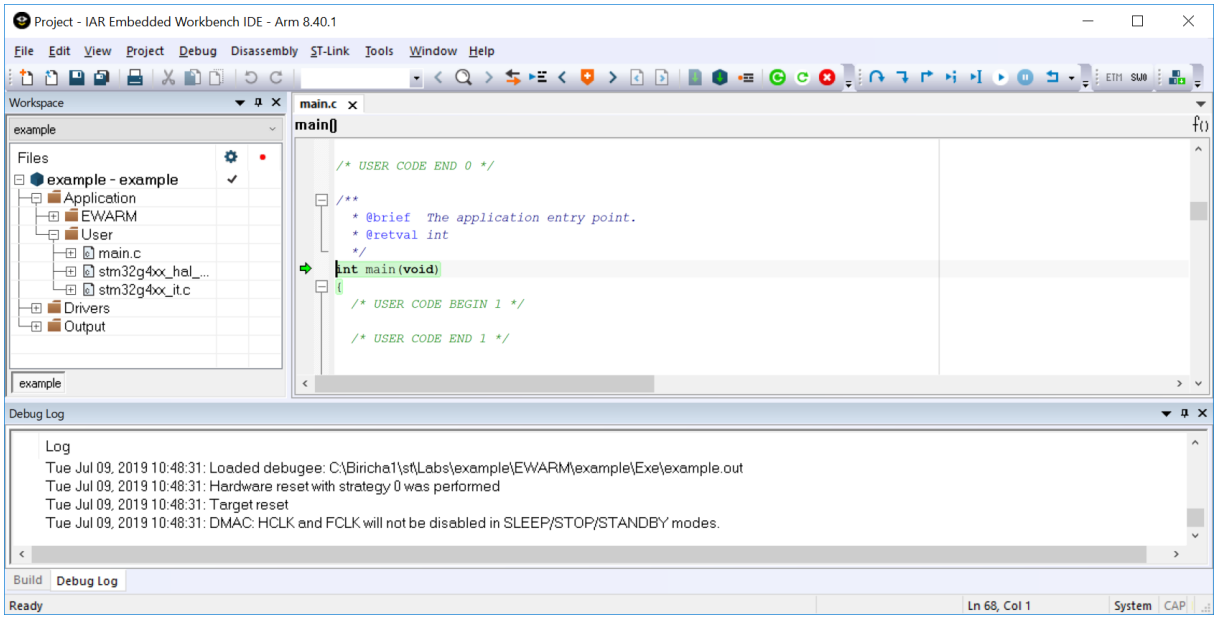
6. Inside the IAR Systems® IDE click the `Download and Debug` icon which compiles and downloads the code to the MCU.

Figure 77. Code compilation and downloading



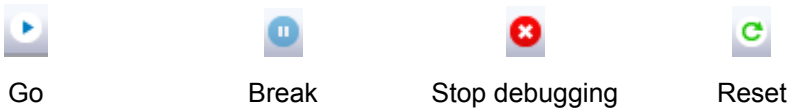
7. In the `Application, User` folder, the three generated C-files are:
   a. `main.c`: Initialization code
   b. `stm32g4xx_hal_msp.c`: MCU support package initialization
   c. `stm32g4xx_it.c`: Interrupt handlers for ISRs
8. The code is now ready to run. The IDE moves the program counter to the `int main(void)` function.

Figure 78. Program counter set for running



9. The buttons along the top menu bar have the functions described in Figure 79. Click the `Go` button to run the code.

Figure 79. Top menu buttons



DT53802V1

   - Go – This runs the code
   - Break – This halts the code
   - Stop Debugging – This terminates the debug session
   - Reset - This resets the code to the beginning and restarts
10. Click on `Stop Debugging` to end the debug session.
11. The firmware is downloaded into the microcontroller Flash memory and as such, IAR Systems® can now be closed if the debugging features are not being closed. The same program restarts each time power is applied to the Discovery kit as it is running from the MCU Flash memory.
12. Close IAR Systems® to stop the debugger. If IAR Systems® asks to `Terminate the debug session`, click OK.

9.3 Onboard load

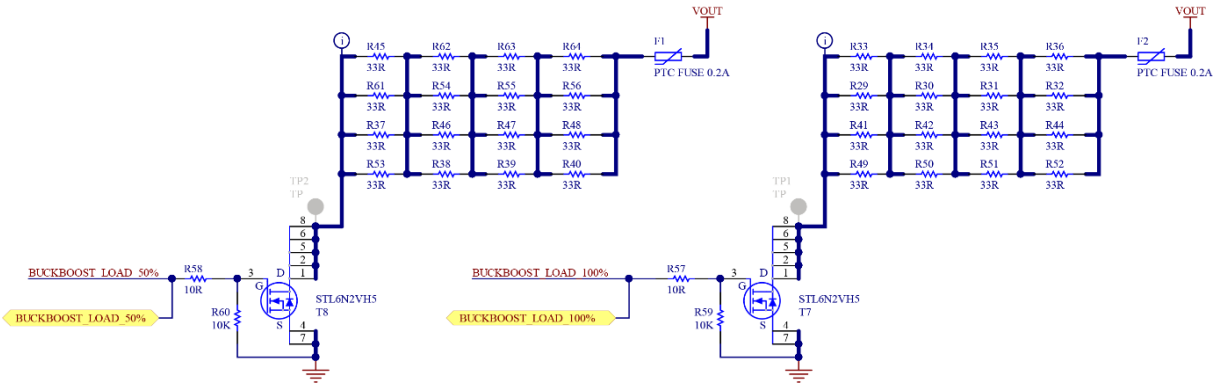
The buck converter on the Discovery kit includes two onboard load banks. Figure 71 shows the two load banks and the position of two test pins which indicate whether the load bank is enabled (test pins are not populated). When TP1 is pulled low, load bank 1 is on. When TP2 is pulled low, load bank 2 is on. The load banks have total resistance and, given the 3.3 V output voltage, the power consumption is shown in Table 8.

Table 8. Onboard load steps

Load (%)	Load (Ω)	Iout (A)	Pout (W)	LED status
0% Load 1 OFF Load 2 OFF	∞ Ω	0 A	0 W	All OFF
50% Load 1 ON Load 2 OFF	33 Ω	0.1 A	0.33 W	Green
100% Load 1 ON Load 2 ON	16.5 Ω	0.2 A	0.66 W	Green and orange

Note that the PTC (F1, F2) adds approximately 1 Ω of resistance in series with each load bank. The load banks can be controlled by the MCU through means of toggling an output pin which is connected to a MOSFET. The MOSFET switches the resistive load bank in and out of the circuit as shown in Figure 80.

Figure 80. Onboard load banks controlled via MOSFETs



Furthermore, the user can control the switching of the load using the joystick onboard the Discovery kit. The operation of the onboard load banks can be controlled as follows:

- Left: Manual adjustment of load
  - Up: Increase the load
  - Down: Decrease the load
- Right: Automatic load switching (transient mode)

The automatic load switching, or transient mode, is used later in the application note to test the transient response of the buck converter and the control loop regulating the output voltage. The status of the load bank during the transient is indicated by the LEDs on board the Discovery kit. If load bank 1 is enabled, the green LED is lit. If load bank 2 is enabled, the orange LED is also lit.

9.4 Source files

The downloadable package for this application note consists of the following files:

STM32CubeMX project files:

- `Buck_VoltageMode_HW.ioc`
  - This file contains all of the configuration data for STM32CubeMX setting up pins and peripherals used. The closed-loop is mostly hardware, thanks to the FMAC usage for the controller computation.
- `Buck_VoltageMode_SW.ioc`
  - This file contains all of the configuration data for STM32CubeMX setting up pins and peripherals used. The closed-loop is using the MCU for the controller computation.
- `Buck_VoltageMode_SW_CCM_SRAM.ioc`
  - This file contains all of the configuration data for STM32CubeMX setting up pins and peripherals used. The closed-loop is using the MCU for the controller computation, which is located in CCM-RAM for the best efficiency.

Source codes:

- `main.c`
  - This file provides the `main.c` function and associates support functions for this application to control the hardware on the board.
- `stm32g4xx_hal_msp.c`
  - Microcontroller support package initialization functions
- `stm32g4xx_it.c`
  - Interrupt service routines

- `system_stm32g4xx.c`
  - Provides the SystemInit() initialization functions for this MCU configuring the system clock

ST-WDS from Biricha files:

- `Buck_CurrentMode_xxx.wds`
  - Matching the STM32CubeMX project file, the ST-WDS configuration file allows the user to load up the design of the buck converter in the PSU design tool from Biricha. The user can then modify the control loop parameters and obtain the updated controller coefficients.

Omicron Lab Bode Analyzer Suite files:

- `Buck_CurrentMode_xxx.bode3`
  - Matching the STM32CubeMX project file, the Bode Analyzer Suite configuration file sets up the tool for loop measurement of the buck converter. The file contains the previously measured traces stored in the memory locations.

9.5 Open-loop operation

Before exercising the buck converter under closed-loop control, it is prudent to check the switching waveforms and dead time are correctly functioning. This check can be performed under the open-loop operation of the buck converter. This means that the controller is taken out of the loop and the buck switches are driven with a fixed duty cycle.

The example software is written such that when a compiler directive is defined, the FMAC interrupt is not enabled and the HRTIM module is set up with a fixed duty cycle.

To run the buck converter under open-loop, locate and uncomment the line of code `#define RUN_OPEN_LOOP`. Within the `main()` function, the function call to set up the compare unit with a fixed duty cycle is now called rather than the function, to enable the FMAC interrupt.

To check the HRTIM output waveforms, connect oscilloscope probes to the following pins:

- PB12 - BUCKBOOST_P1_DRIVE – High-side buck MOSFET
- PB13 - BUCKBOOST_N1_DRIVE – Low-side buck MOSFET

These signals must not be HIGH at the same time as this may lead to a potential shoot-through event. During the configuration of STM32CubeMX, dead time is inserted between the two channels to ensure that the switches are never ON at the same time. The amount of dead time can be measured using the scope and compared to the value set earlier in this application note.

The dead-time ticks f_{DTG} are generated from f_{HRTIM} and a prescaler. For the configuration discussed earlier in this application note:

- $f_{DTG} = f_{HRTIM} * 8$, and because $f_{HRTIM} = 170$ MHz:
- $f_{DTG} = 170$ MHz $* 8 = 1360$ MHz, therefore:
  - 1 dead-time tick = 0.735 ns

Earlier, the dead time is configured to have:

- Rising value = 75 ticks = 55 ns
- Falling value = 300 ticks = 220 ns

Therefore, between the falling edge of PB13 and the rising edge of PB12, there are 55 ns of dead time as measured in Figure 81. Between the falling edge of PB12 and the riding edge of PB13, there are 220 ns of dead time as measured in Figure 82.

Figure 81. **Rising edge dead-time. Ch1: high-side FET. Ch2: low-side FET. Dead-time measured as 54 ns.**

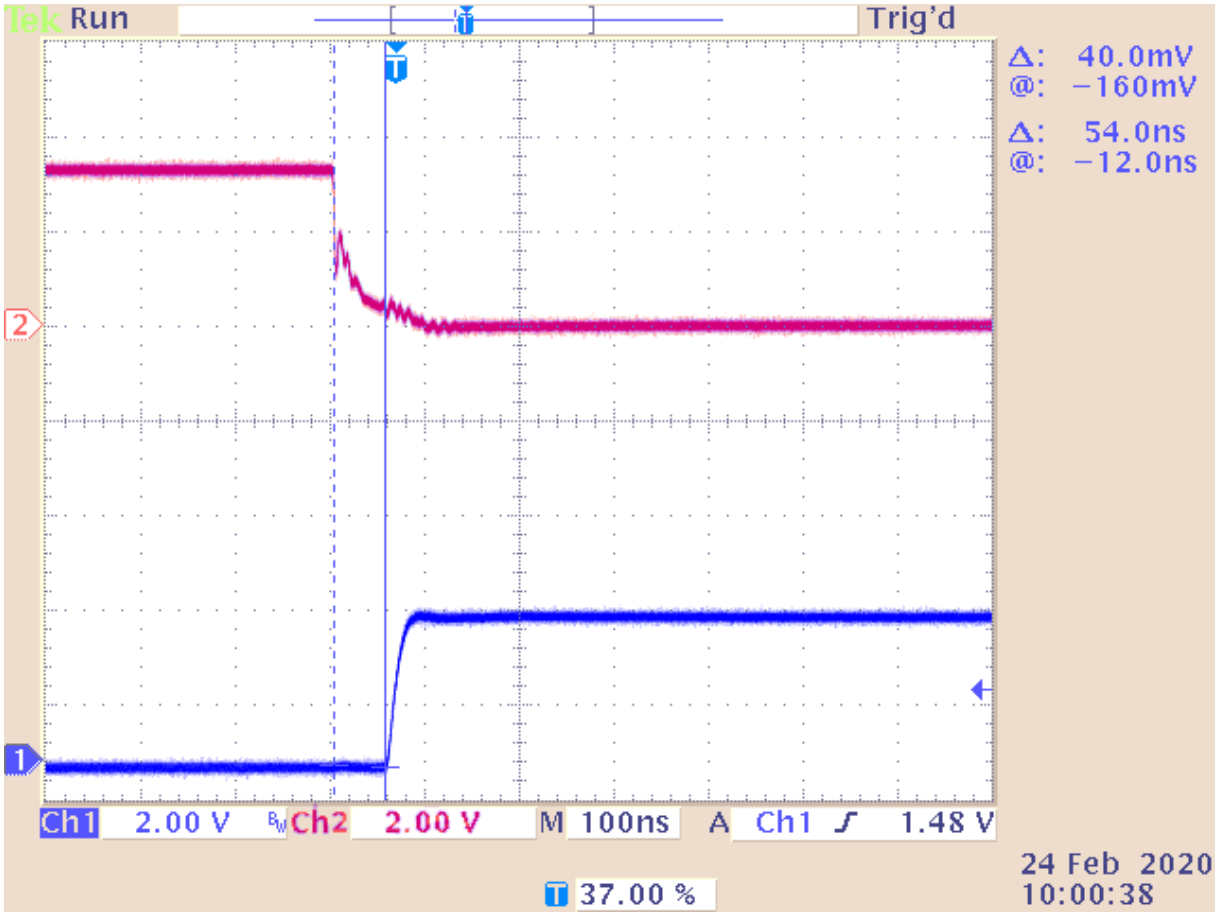
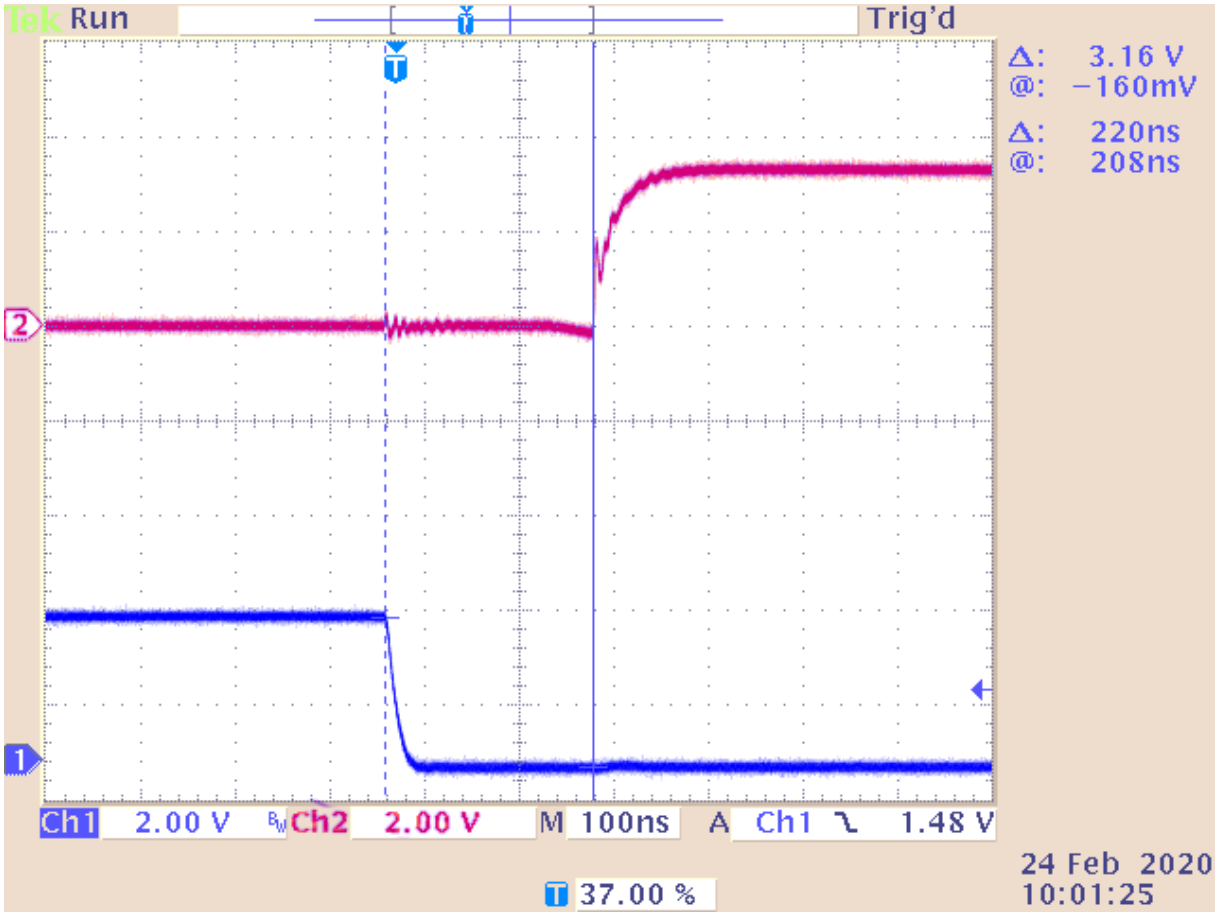


Figure 82. **Falling edge dead-time. Ch1: high-side FET. Ch2: low-side FET. Dead-time measured as 220 ns.**



The output voltage of the buck converter is currently not regulated as the FMAC ISR is not running. Instead, a fixed duty cycle is providing some output voltage by driving the switches in a complementary manner. With none of the loads enabled the output voltage may be around 2.2 V. If the load is added by moving the joystick left or right then the output voltage also changes.

9.6 Closed-loop control

9.6.1 Load regulation

The converter is initially tested under open-loop conditions with no control over the output voltage. The next step is to close the control loop to regulate the output voltage.

To run the buck converter under closed-loop control, locate and comment on the line of code `#define RUN_OPEN_LOOP` by changing it to `// #define RUN_OPEN_LOOP`. Re-build the project then download and debug the code.

With the converter running, the digital FMAC compensator now regulates any changes in the load on the converter by updating the DAC register value. The load regulation can be tested by varying the onboard load using the joystick and measuring any change in the output voltage.

Table 9. Closed-loop load regulation

Load	Iout	Vout
0% Load 1 OFF Load 2 OFF	0 A	3.3 V
50% Load 1 ON Load 2 OFF	0.1 A	
100% Load 1 ON Load 2 ON	0.2 A	

9.6.2 Transient response

The onboard load allows the user to perform transient response tests on the closed-loop digital power supply to assess the performance of the controller. The transient response test is a useful method of determining if the implemented controller is stable, the speed of the response, and whether there is a sufficient phase margin in the system.

The converter's transient response can be measured using an oscilloscope. To measure the transient response, set up the oscilloscope as follows:

- Channel 1 -> Connect to header marked Vout
- Channel 2 -> Put probe tip in the hole marked TP1
- Set coupling on Channel 1 to AC and set the volts per division to 20 mV
- Set the horizontal scale, in seconds per division time base, to 100 µs
- From the Trigger Menu: Set the trigger to the falling edge of Channel 2 and set the `Mode` to `Normal`.

Set the load to the transient mode by pressing the `Up` arrow on the blue joystick. The orange load LEDs must now flash with half a second intervals. When the green LED is on, the load is set to 50%. When both LEDs are ON, the load is set to 100%. The undershoot and the settling time can then be observed on the oscilloscope during the 50% to 100% load transient. Using the example project provided with this Discovery kit, the converter may have a well-tuned controlled loop as shown in Figure 83.

Figure 83. Output voltage (Ch3) transient from load change 50% to 100% (Ch1), output voltage undershoot = 40 mV, settling time = 300 µs

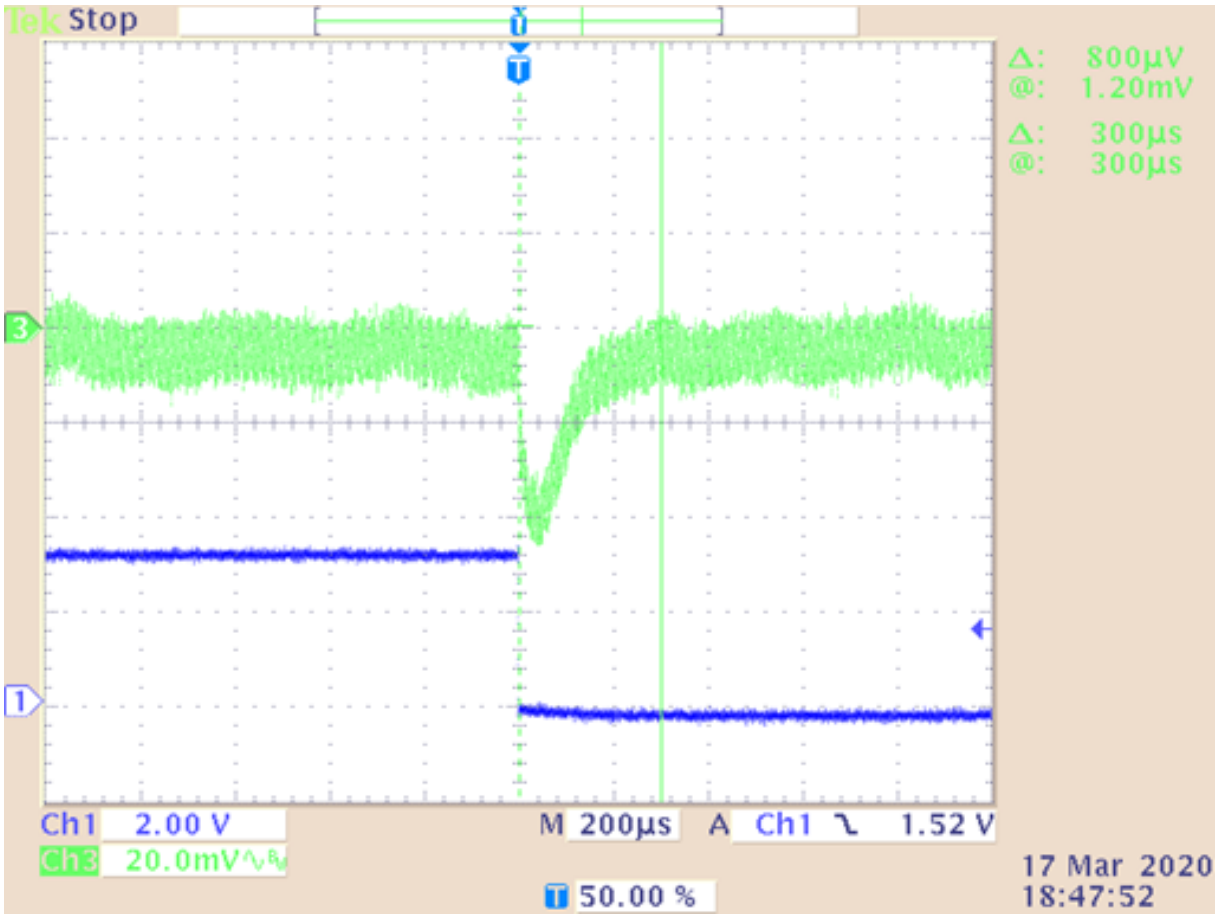


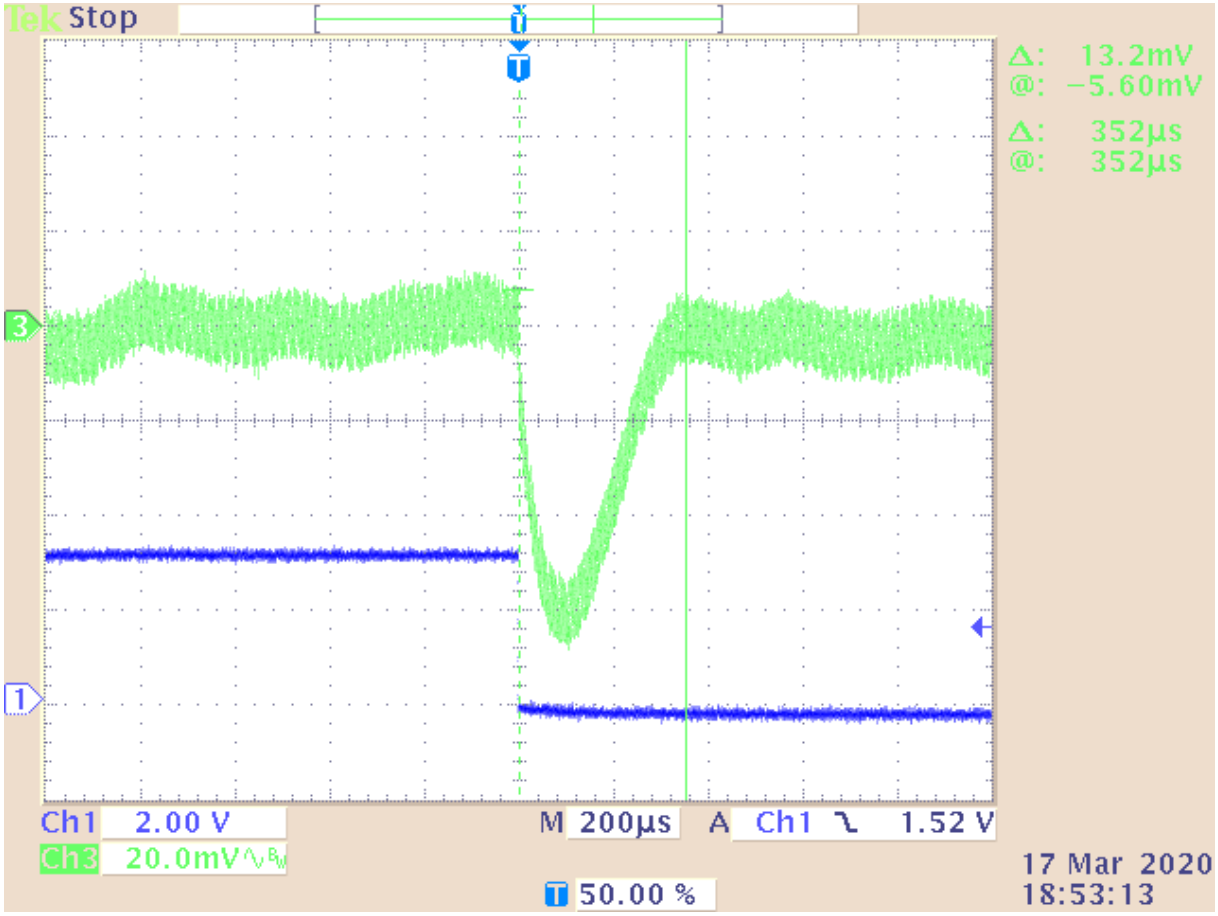
Figure 83 shows the transient response of the closed-loop digital buck converter for a 50% to 100% step load transient with a settling time of fewer than 300 µs and an undershoot of less than 50 mV with no ringing.

9.6.3 Impact of crossover frequency on the transient response

The transient response of the converter in Figure 83 indicates a stable and well-designed controller. This is using the controller which is described in detail in Section 6 of this application note. The controller is designed to have a crossover frequency of 4 kHz and a phase margin of 50°. The choice of crossover frequency and phase margin in the frequency domain affects the transient response as seen in the time domain.

For example, if ST-WDS is used to redesign the controller using a lower crossover frequency, then it may result in a slower transient response and as shown in Figure 84. In this plot, the controller is redesigned with a crossover frequency of 2 kHz, half that of the previous controller.

Figure 84. Output voltage (Ch3) transient from load change 50% to 100% (Ch1) with loop crossover of 2 kHz, output voltage undershoot = 60 mV, settling time = 350 µs

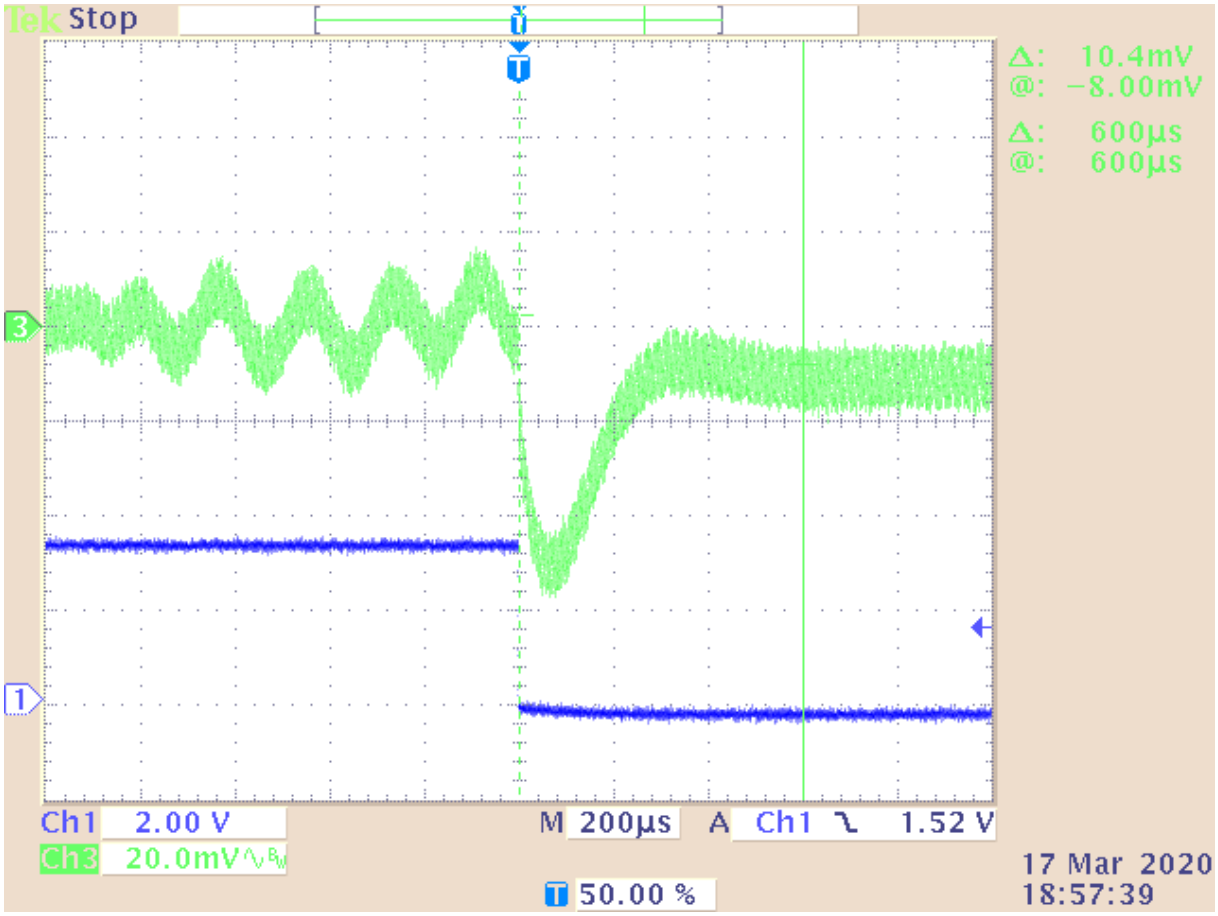


It follows therefore that if the crossover frequency is increased the transient response speeds up also and undershoot reduces. However, there is an upper limit to the crossover frequency as at higher frequencies the phase erosion becomes more significant and reduces the loop phase margin. The issue of phase erosion in the discrete-time system is discussed in detail in Section 5 Discrete-time controller.

9.6.4 Impact of phase margin on the transient response

The phase margin is a measure of the relative stability of the control loop. The lower the phase margin the more oscillatory the transient response in the time domain. ST-WDS is used again to redesign the controller using a lower phase margin. The new controller is designed with the same crossover as initially, 4 kHz, however a phase margin of 30°. The resulting controller coefficients can be pasted into the code which is then recompiled and downloaded onto the MCU. The transient response is given in Figure 85 for the system with a phase margin of 30°.

Figure 85. Output voltage (Ch3) transient from load change 50 to 100% (Ch1) with loop crossover of 4 kHz and a designed phase margin of 30°, output voltage undershoot = 50 mV, settling time = 600 µs



In Figure 85 there are several oscillations on the output voltage waveform before the transient and a small overshoot during the recovery from the transient back to its steady state. The initial recovery time is shorter than when the phase margin is 50°, however, the output voltage overshoots from the desired setpoint, and the net result is that the overall recovery time is longer. If the phase margin is reduced further, the oscillations take longer to decay. With a phase margin of 0°, the system may oscillate indefinitely.

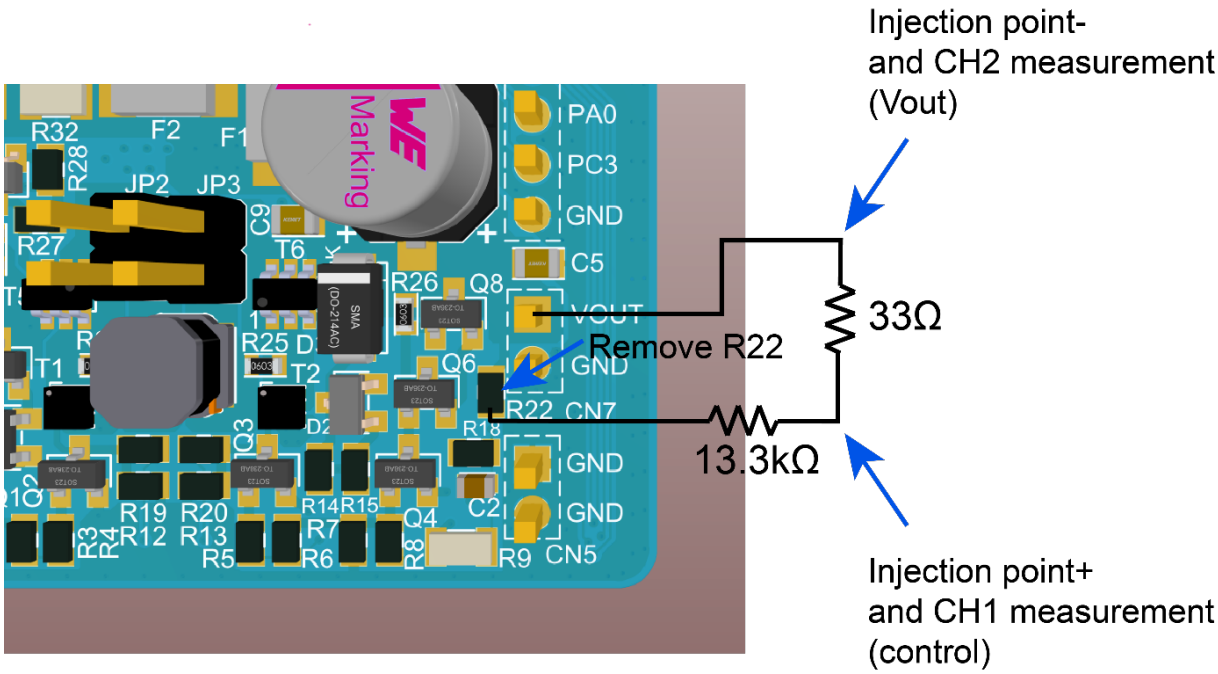
9.6.5 Bode plot measurement under closed-loop control

The real-life loop response of the buck converter can be verified through measurement using a frequency response analyzer. In this application note, the Bode 100 vector network analyzer from Omicron Lab is used to perform the measurement. The Bode 100 injects a sinusoidal signal into the feedback loop of the power supply and measures how that signal changes as it passes through the controller and the plant power stage.

A small modification is required to the Discovery kit to measure the loop. The feedback path of the output voltage must be broken and an injection resistor needs to be inserted. The injection transformer is then connected across this resistor. The injection transformer superimposes the sinusoidal signal from the Bode 100 onto the feedback voltage which is being used to close the loop.

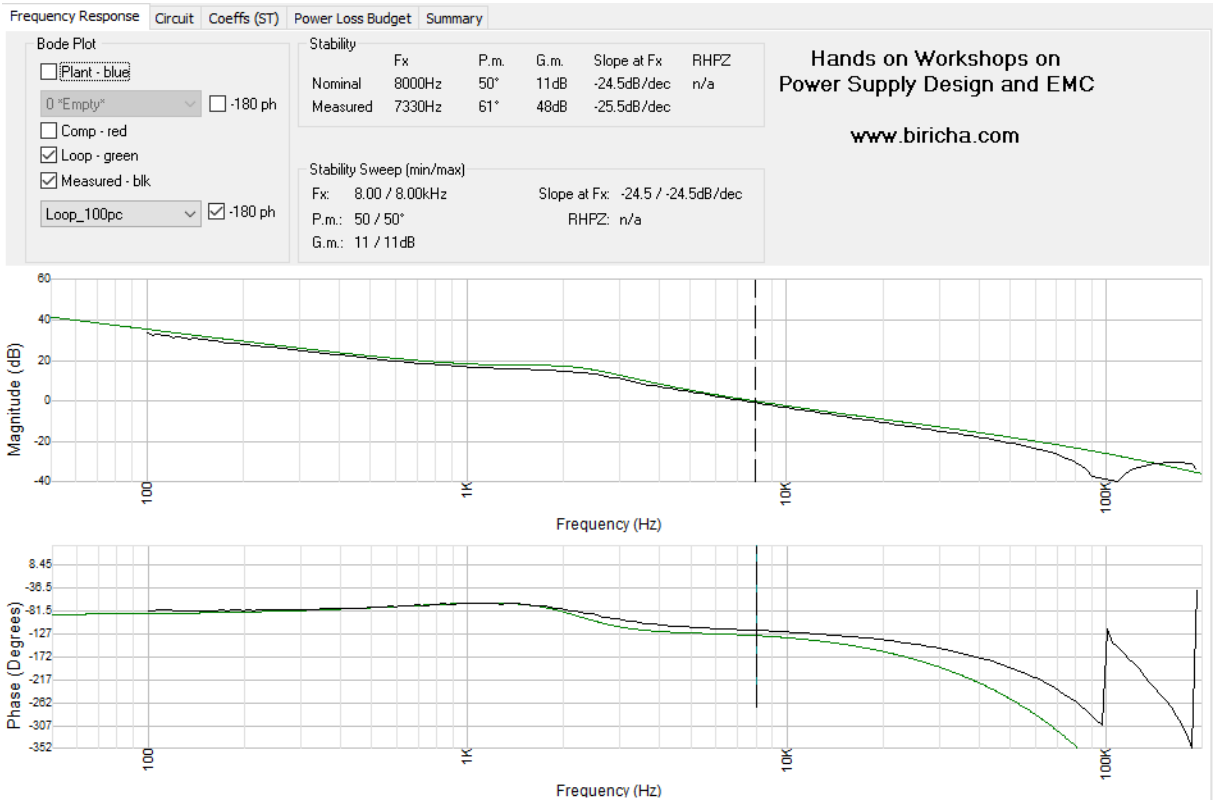
The schematic for this modification is shown in Figure 86. The measurement probes are then connected as follows: CH1 (or the control measurement point) is connected to the end of the injection resistor which is not connected to the output voltage. Then CH2 (or the output measurement point) is connected to the end of the injection resistor which is directly connected to the output voltage. The grounds of the probes may be connected to the header marked GND closest to this. Note that JP2 and JP3 are not fitted by default on this board.

Figure 86. Connection setup for control-to-output transfer function measurement



A frequency sweep from 100 Hz to 100 kHz is recommended (up to half the switching frequency) with the signal injection level adjusted to give a continuous smooth measurement result without affecting the steady-state response of the loop. Once a clean continuous measurement is obtained, the `.bode3` file can be saved from within Omicron Lab's Bode Analyzer Suite and then imported into ST-WDS as shown in Figure 87.

Figure 87. Loop measurement at 100% load imported into ST-WDS (black: measured loop, green: simulated loop)



In Figure 87 the actual measurement of the loop (control-to-output) transfer function is plotted as the black trace. This is compared with the simulated loop plotted in green. The magnitude plot shows a good correlation between simulated and measured loop responses with a crossover frequency of 4 kHz in simulated and 3.56 kHz as measured.

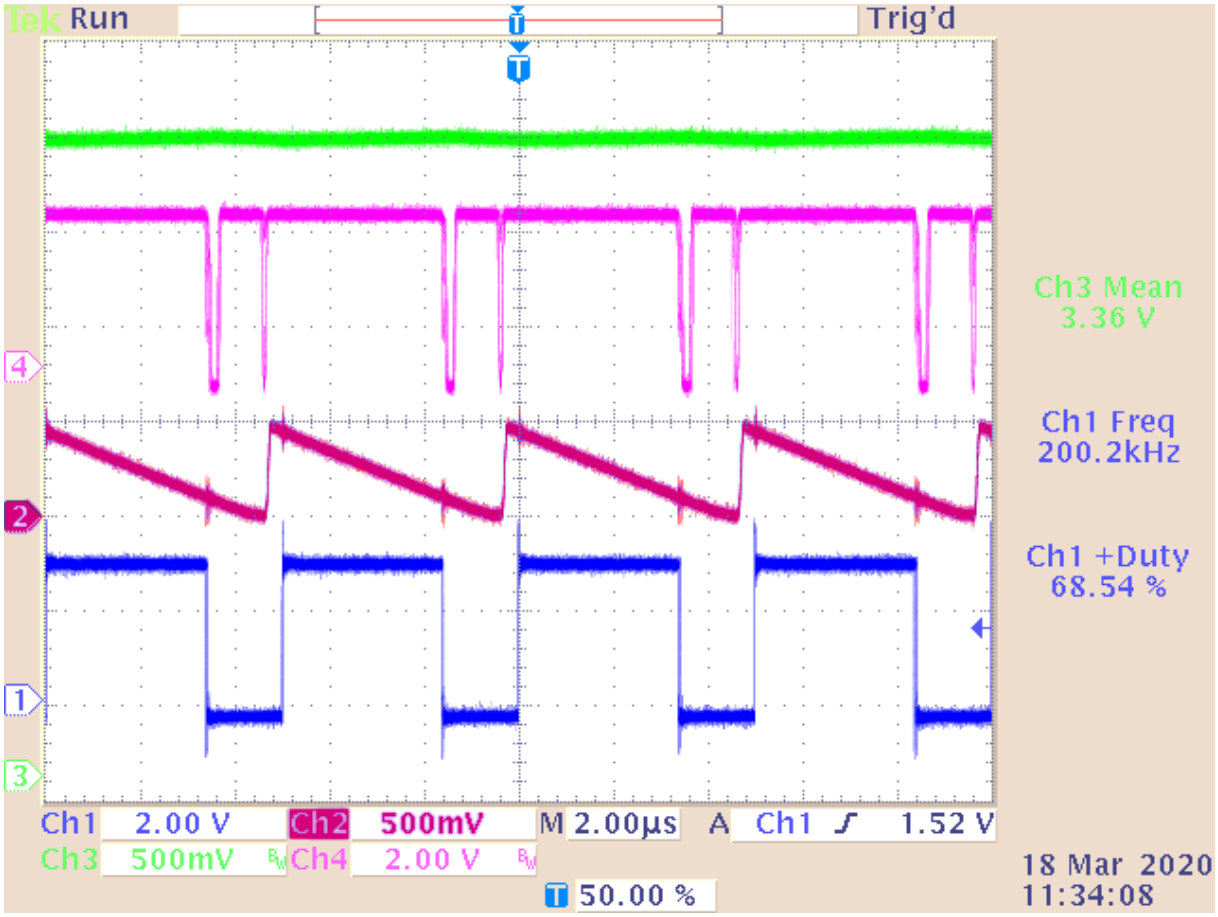
The phase response is also a good match across all frequencies until the injected frequency approaches half the sampling frequency. As we approach this Nyquist frequency the phase rolls off and beings to wrap around. It is also prudent to check the crossover and phase margin at different loads. This is discussed in the next section.

9.7 Digital slope compensation

This PCMC converter implements digital slope compensation using the sawtooth waveform generator built-in into the onboard DAC. During the configuration process, the output of the DAC is connected to OPAMP5 in follower mode to allow the user to view the output of the DAC on a physical pin. In Figure 90 the DAC output can be seen along with the PWM and comparator output. The initial value of the DAC output is set by the controller, in this case via the FMAC ISR. The sawtooth waveform generator then subtracts a specified value from the DAC at pre-defined subintervals within the switching period. The result is the sawtooth shape seen in Figure 90. The peak-to-peak value of the sawtooth waveform is equal to the slope compensation ramp height calculated in the previous steps. This can be read as 0.5 V from the oscilloscope plot and this corresponds to the value set in the example project. From this figure, the steady PWM trace indicates that there are no sub-harmonic oscillations and the complex conjugate pole at half the switching frequency has been sufficiently damped.

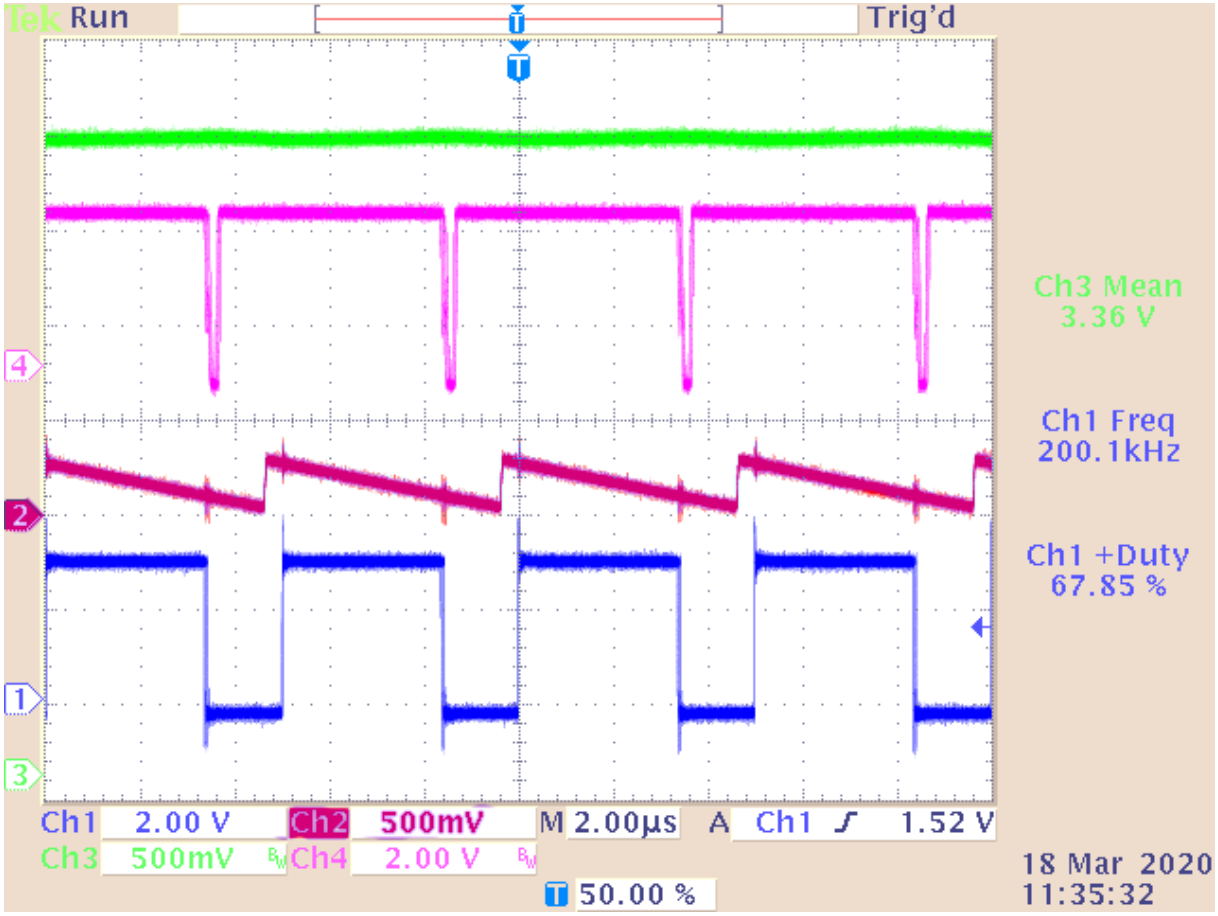
The comparator output can be seen transitioning high-to-low which occurs when the sensed current intersects with the demand current - determined by the output of the DAC. This triggers the end of the PWM high pulse. A further high-to-low transition occurs on the comparator output at the end of the switching cycle. This is when the DAC output has dropped to zero due to the slope compensation. At this moment the DAC output is lower than the voltage on the non-inverting pin and the output of the comparator transitions low. The PWM output is already low therefore this has no effect. At the beginning of the next switching period, the rise of the PWM output is delayed to allow time for the DAC to rise to its starting value and for the comparator output to return HIGH.

Figure 88. PWM (Ch1), DAC4 output via OPAMP5 (Ch2), output voltage (Ch3), comparator 6 output (Ch4), Vramp = 0.5 V



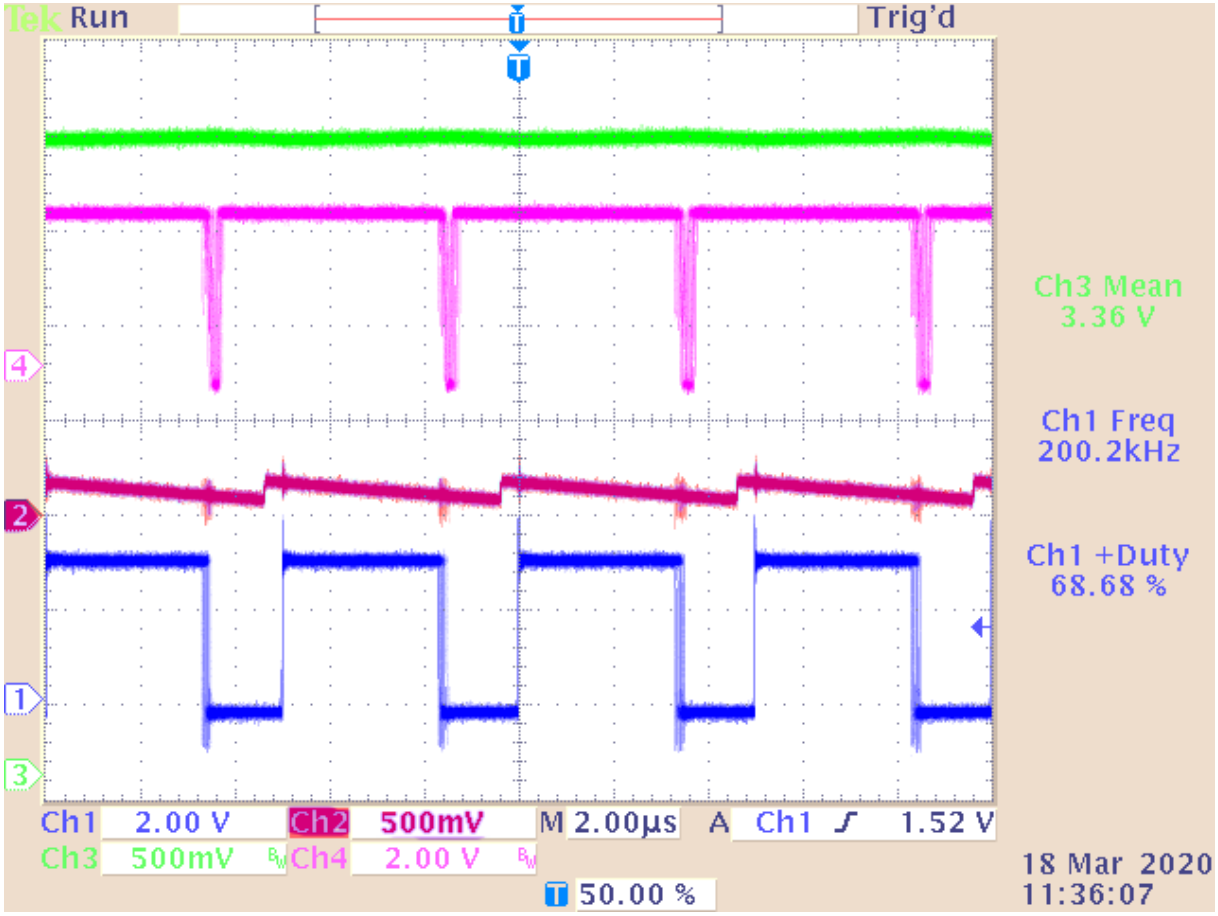
The peak-to-peak value of slope compensation can be changed at runtime by updating the variable “Vramp” in the example project using the live watch window. In Figure 89 the ramp height is reduced to 0.25 V over the switching period. The PWM remains relatively stable with no indication of sub-harmonic oscillations. The DAC output also does not reach zero at the end of the switching period, and therefore, the comparator output does not show a second transition from high to low as seen when a ramp height of 0.5 V is used.

Figure 89. PWM (Ch1), DAC4 output via OPAMP5 (Ch2), output voltage (Ch3), comparator 6 output (Ch4), Vramp = 0.25 V



Finally, in Figure 90, the ramp height is reduced to 0.1 V over the switching period. Sub-harmonic oscillations can be seen on the PWM trace in the form of jitter on the falling edge of the PWM. Decreasing the ramp height further leads to further instability due to the increased sub-harmonic oscillations.

Figure 90. **PWM (Ch1), DAC4 output via OPAMP5 (Ch2), output voltage (Ch3), comparator 6 output (Ch4), Vramp = 0.1 V**



9.8 Waveform display in IAR Systems®

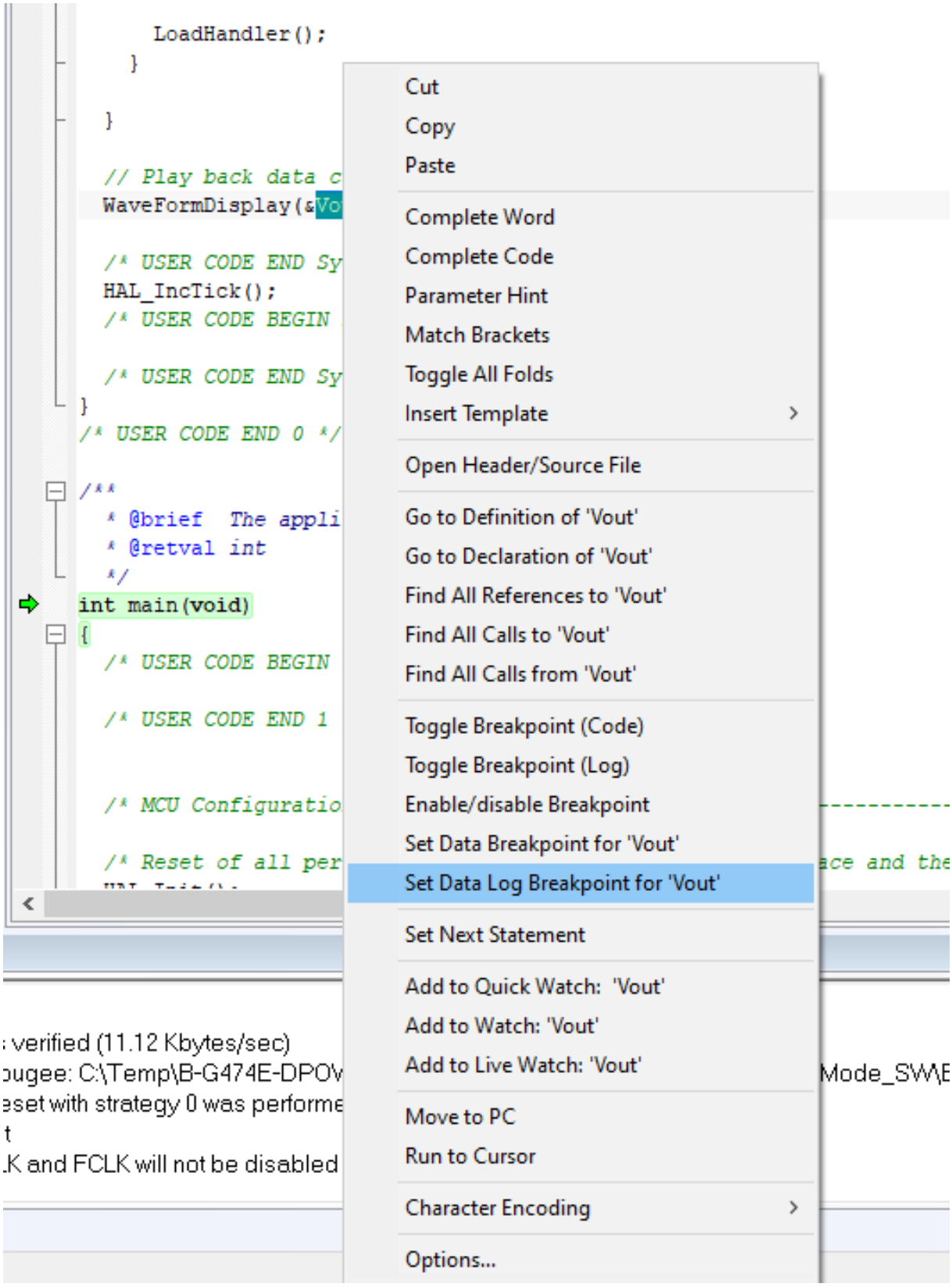
If the user has access to IAR Systems® then it is possible to use the timeline feature within this to plot variables captured using data log breakpoints. This is suitable for slow-changing variables in real-time, or fast-changing variables which are stored in a buffer and then transferred via the debugger to the IDE at a slower rate.

An example of this functionality is included in the project provided `Buck_CurrentMode_SW`. Note that this project executes the 2p2z controller called within the ADC ISR implementing a controller running on the main core rather than the FMAC. To exercise this example, open the `Buck_CurrentMode_SW.ioc` file within the folder and click `GENERATE CODE`. Open IAR Systems®, build and run the code. Note that, there is now an additional function within the `SysTick_Handler` ISR located towards the top of the `main.c` file:

```
WaveFormDisplay(&Vout, &CompareReg);
```

Double-click on `Vout` and then right-click on the highlighted text and select `Set Data Log Breakpoint for 'Vout'`. Repeat this process for `CompareReg`.

Figure 91. **Add data log breakpoint to capture the variable each time it is updated**



This forces the debugger to record the value of these variables each time they are updated. For this reason, the update of these variables is performed in a slower function within the `SysTick_Handler` so as not to overload the debugger.

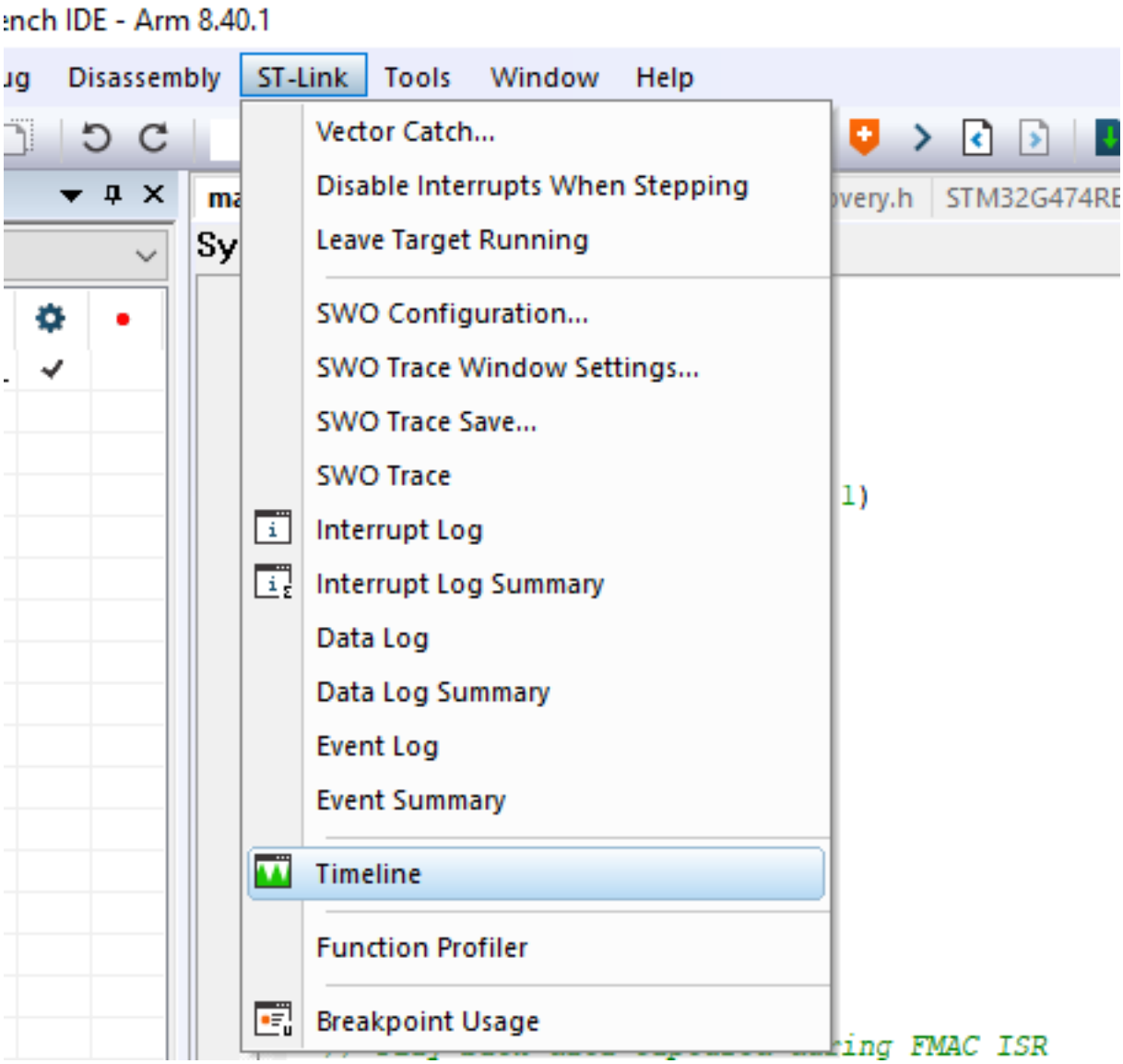
The data is captured within the ADC ISR located in the `stm32g4xx_it.c` file. The code snippet below shows the data being passed to the `WaveFormRecord()` function each time the ISR is called.

```
if (Waveform.m_State == WAVEFORM_RECORD)
  {
    WaveFormRecord(VoltageSensing, Demo.CtrlFloat.m_Out);
  }
```

The waveform structure state is changed to `WAVEFORM_RECORD` every time the user presses the load increment/ decrement button on the Discovery kit. This can be seen in the `HAL_GPIO_EXTI_Callback()` GPIO interrupt function which is included in `STM32G474RE_Discovery.c`. This function calls `WaveFormTrigger()` which in turn changes the state to begin recording the waveform during the ADC ISR.

From the `ST-Link` menu on the toolbar click on `Timeline` as shown in Figure 92.

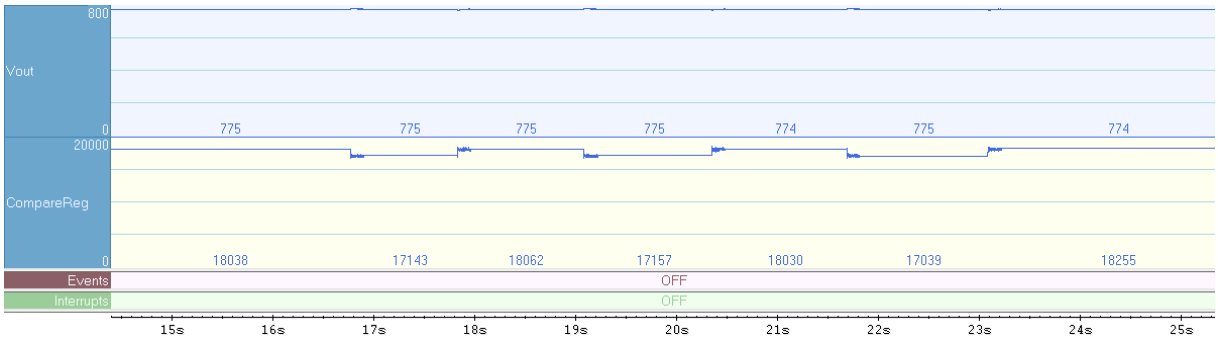
Figure 92. Enable timeline view to monitor real-time updates of variables on a graph



This shows the timeline pane in the IAR Systems® workspace. If the graph pane is not visible, the user may need to right-click in the blue area and click `Enable`.

Right-click on the `Vout` blue graph window and untick `Hexadecimal`, set `Size -> Large`, and `Zoom -> 100 ms`. Repeat this process for `CompareReg`. The graph windows must look like in Figure 93.

Figure 93. The timeline view requires some manual adjustment of the scale to view the information presented.



If the waveforms are invisible, try to right-click on the graph window, go to `Navigate -> End`, and then use the horizontal scroll bar to scroll the view to the left until the window is filled.

As mentioned above, the data captured by the ADC is re-played by the function `WaveFormDisplay` at a much slower rate to allow the debugger to capture the data via the data log breakpoint. Therefore, the horizontal time axis is now much slower than the sampled signal.

It is now possible to zoom in on the output voltage to view the transients in more detail by changing the scale. Right-click on the `Vout` graph pane and click on `Viewing Range`. In the dialog box, select `Custom Limits` and enter limit values to fill the entire vertical axis with data from the output voltage. This may take several attempts to find the appropriate values. In Figure 94, the values 765 and 785 are used for the output voltage and 15000 and 20000 are used for the compare register. The graph window must now look like this:

Figure 94. Timeline view with an adjusted scale showing the voltage transients and change in the duty cycle

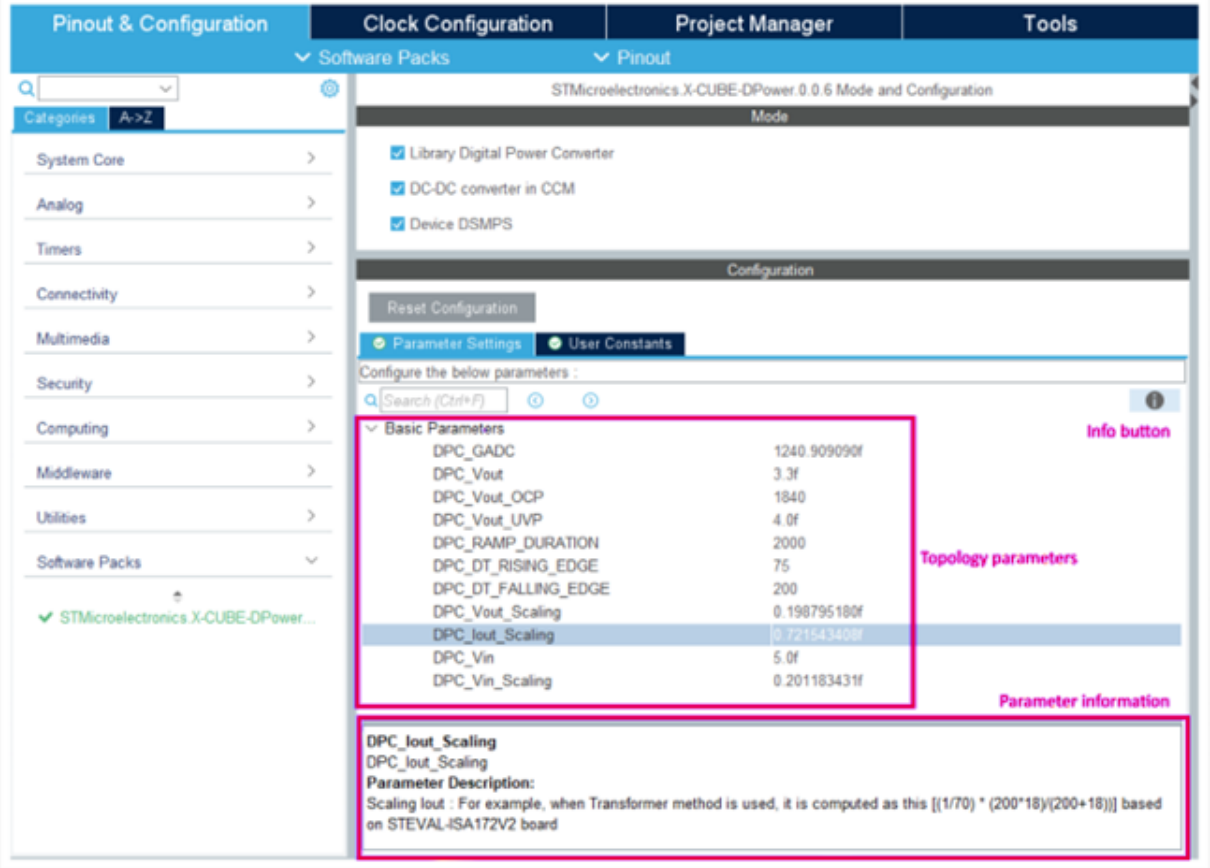
9.9 Buck converter usage with the X-CUBE-DPower pack

Refer to the X-CUBE-DPower user manual (UM3102) available at *www.st.com* to install the pack and start up with the wanted converter.

9.9.1 Configure the converter parameters using GUI

As shown in Figure 95, they are many user parameters that are now available when using X-CUBE-DPower.

Figure 95. X-CUBE-DPower – GUI parameters for the buck converter



The parameters are described using either the information button then selecting the parameter, or clicking directly on the parameter.

9.9.2 Command the converter using board UI

To interact with the user, the B-G474E-DPOW1 board implements one joystick selection and four colored LED indicators.

The buck converter, generated from X-CUBE-DPower, redefines the user interaction as presented in this paragraph. Note that it behaves differently from the standard G4 IntroPack example.

The joystick is useful to command the converter as follows:

- Up button: Activate automatic load transients toggling
- Down button: Deactivate automatic load transients toggling
- Right button: Increase the total of activated load resistors
- Left button: Decrease the total of activated load resistors
- Center button: Unused

LEDs inform about the converter status as follows:

- Green: The system is running. Highlighting details are given in the state machine diagram Figure 96.
- Red: An error or a fault is detected. Highlighting details are given in the state machine diagram Figure 96.

- Orange: While not meaningful during automatic mode activation, it reflects only the total of activated load resistors during manual mode activation as follows:
  - OFF: No load resistors
  - 1 flash/s: 50% of load resistors
  - 2 flash/s: 100% of load resistors
- Blue: Reflect the mode activation as follow:
  - ON: Automatic mode
  - OFF: Manual mode

9.9.3 Topology dedicated files

Refer to the X-CUBE-DPower user manual (UM3102) available at *www.st.com* to clarify the project files architecture and usage.

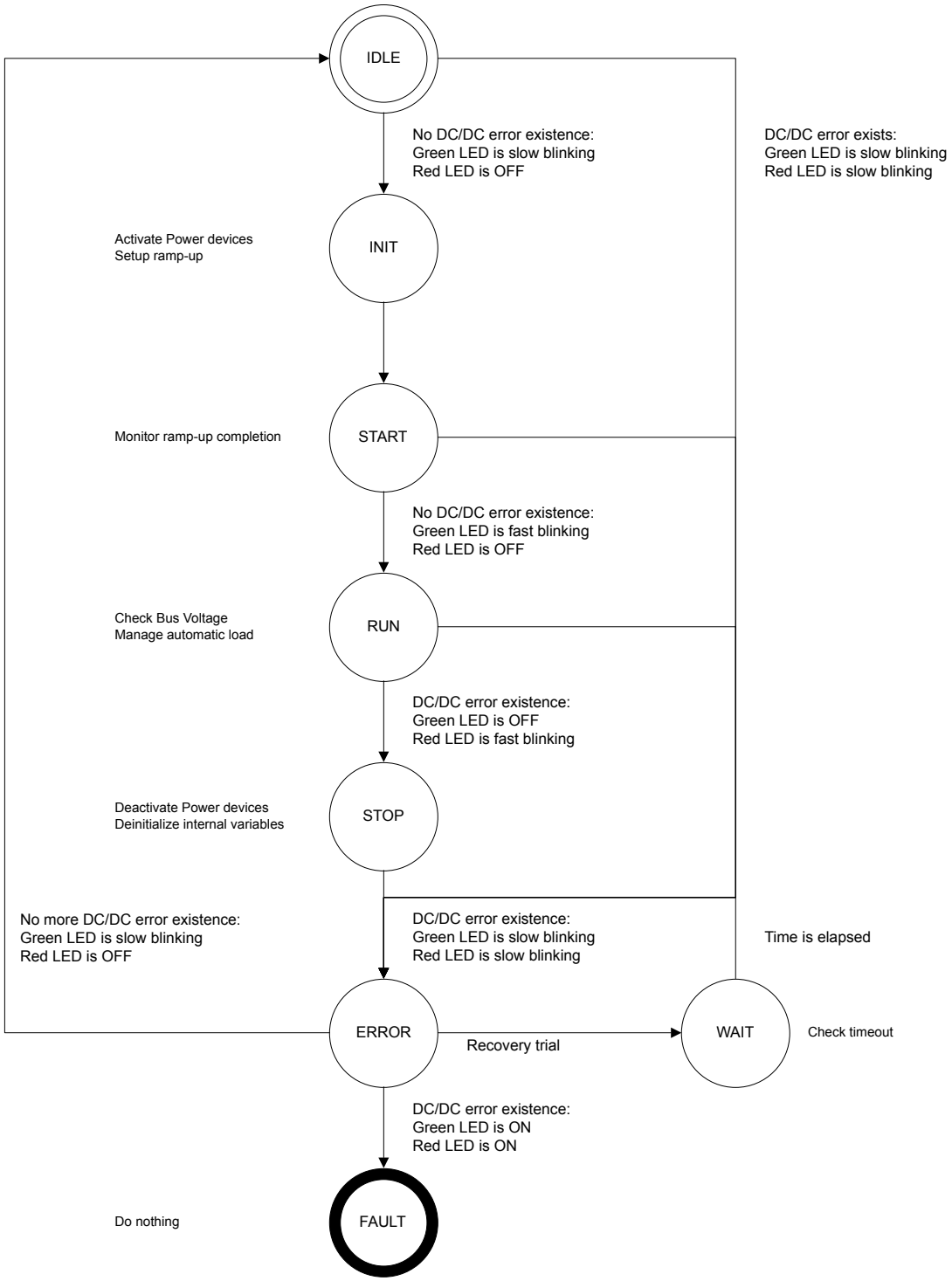
Regarding the buck converter, `app_dpower.h` implements only the following user modalities (refer to the header file for a detailed description):

- #define CHANGE_CTRL_TO_PID
- #define OVERCURRENT_PROTECTION
- #define OVERVOLTAGE_PROTECTION
- #define SHORT_CIRCUIT_PROTECTION
- #define OVERTEMPERATURE_PROTECTION
- #define DEBUG_MODE
- #define DEBUG_COMP_OUT
- #define RUN_OPEN_LOOP
- #define PLOT_WAVEFORM

9.9.4 State machine

The State Machine implementation is given in Figure 96.

Figure 96. X-CUBE-DPower – Buck converter state machine



10 Measured results

In this section the buck converter onboard the Discovery kit is exercised using the peak current mode control example provided and the measurement results are captured and discussed.

10.1 Load regulation

The converter is now operating under closed-loop control and, as there is an integrator within the control loop, there is good load regulation. This means that the converter responds to any changes in load and regulates such that the output voltage remains constant.

This can be tested by observing the output voltage on the oscilloscope and varying the load using the joystick. The duty cycle is also monitored and this changes somewhat between the different load steps due to losses within the power stage.

Figure 97 shows the output voltage and PWM for 0% load, with the onboard load banks disabled. The oscilloscope measures the output voltage with a mean value of 3.35 V with a duty cycle of 62%. The load is then increased to 50% of the rated output current in Figure 98 and the output voltage remains constant at 3.35 V, and the duty cycle increases to 65%. Finally, the load is increased to 100% of the rated output in Figure 99, and again the output voltage remains at 3.35 V, and the duty cycle increases to 68%. This indicates that the controller is regulating the output voltage of the buck converter given the changes in load.

Figure 97. Output voltage (Ch3) and PWM (Ch1) at 0% load

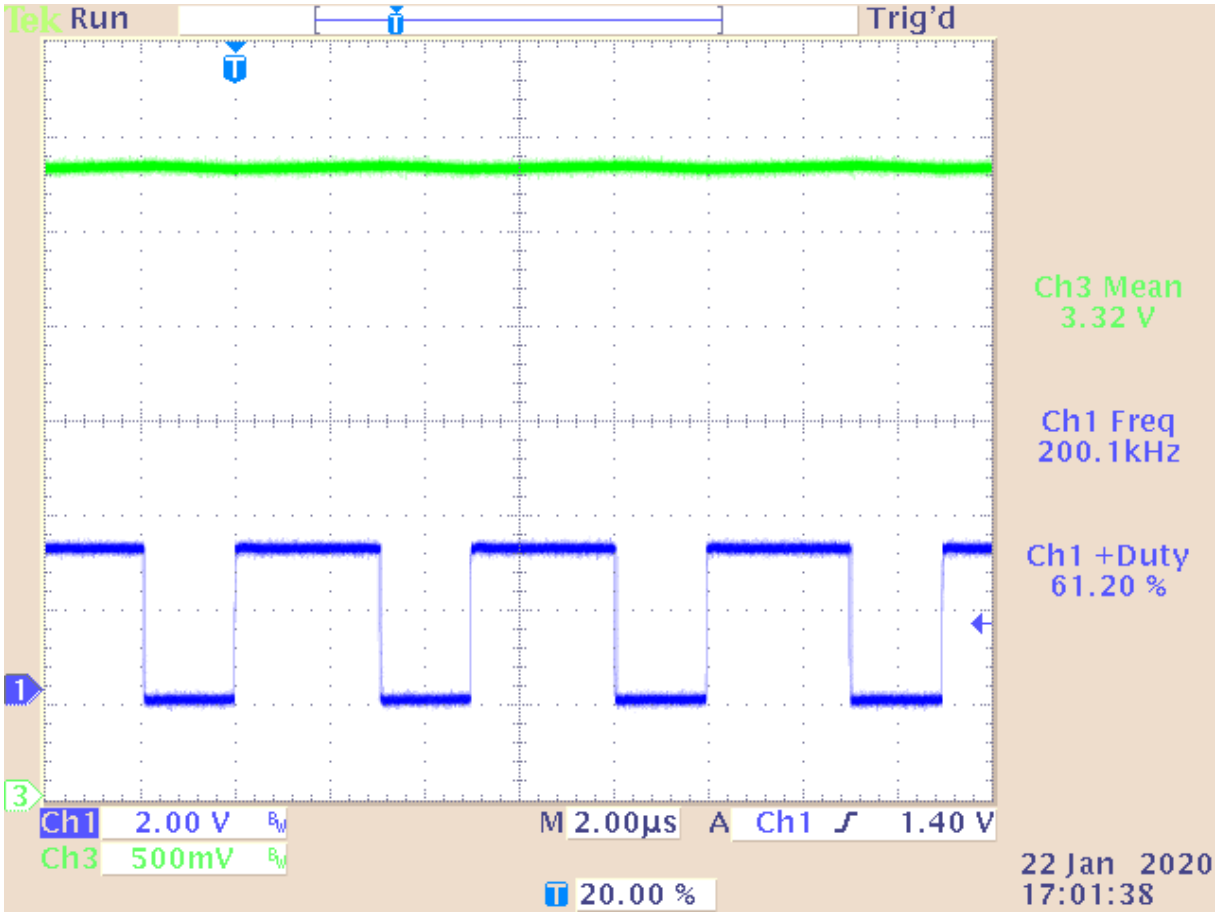


Figure 98. **Output voltage (Ch3) and PWM (Ch1) at 50% load**

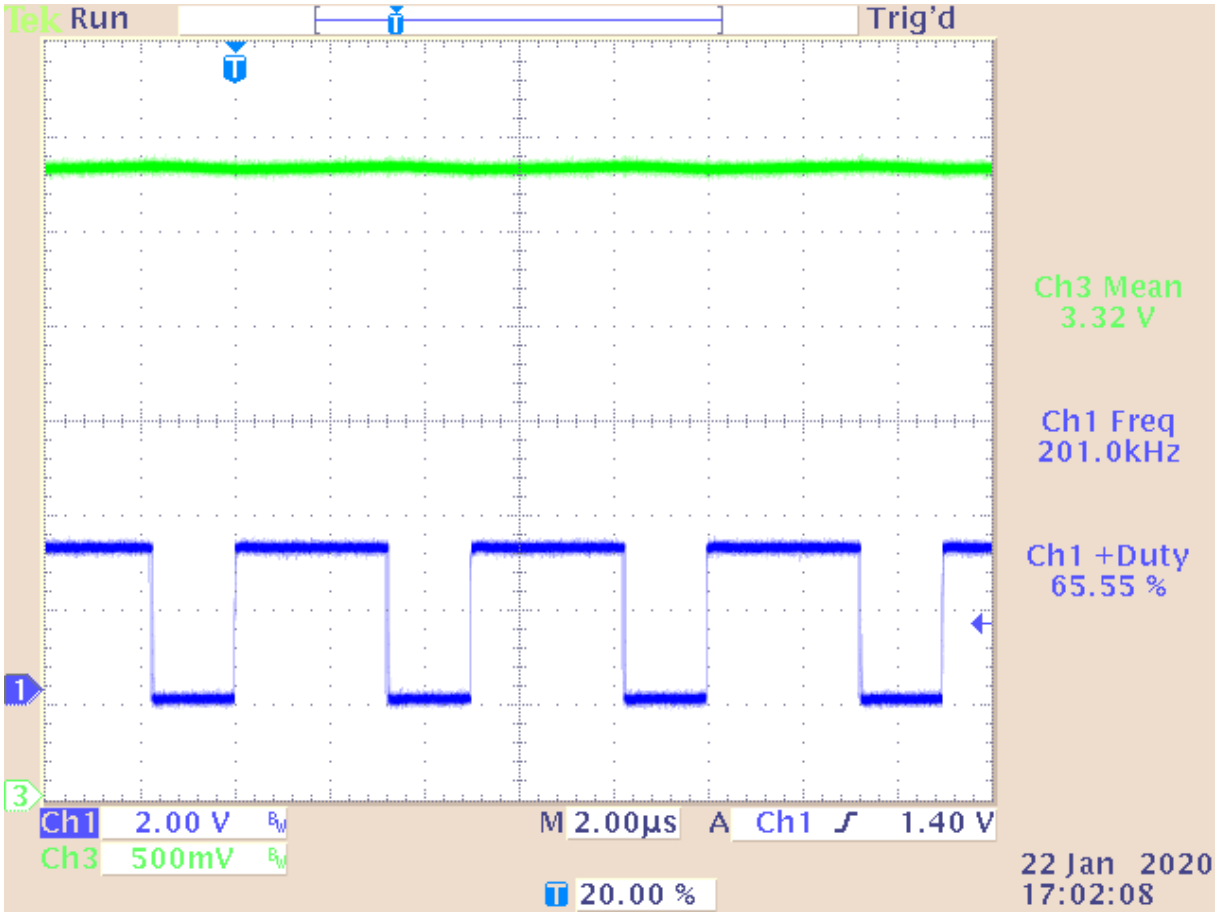
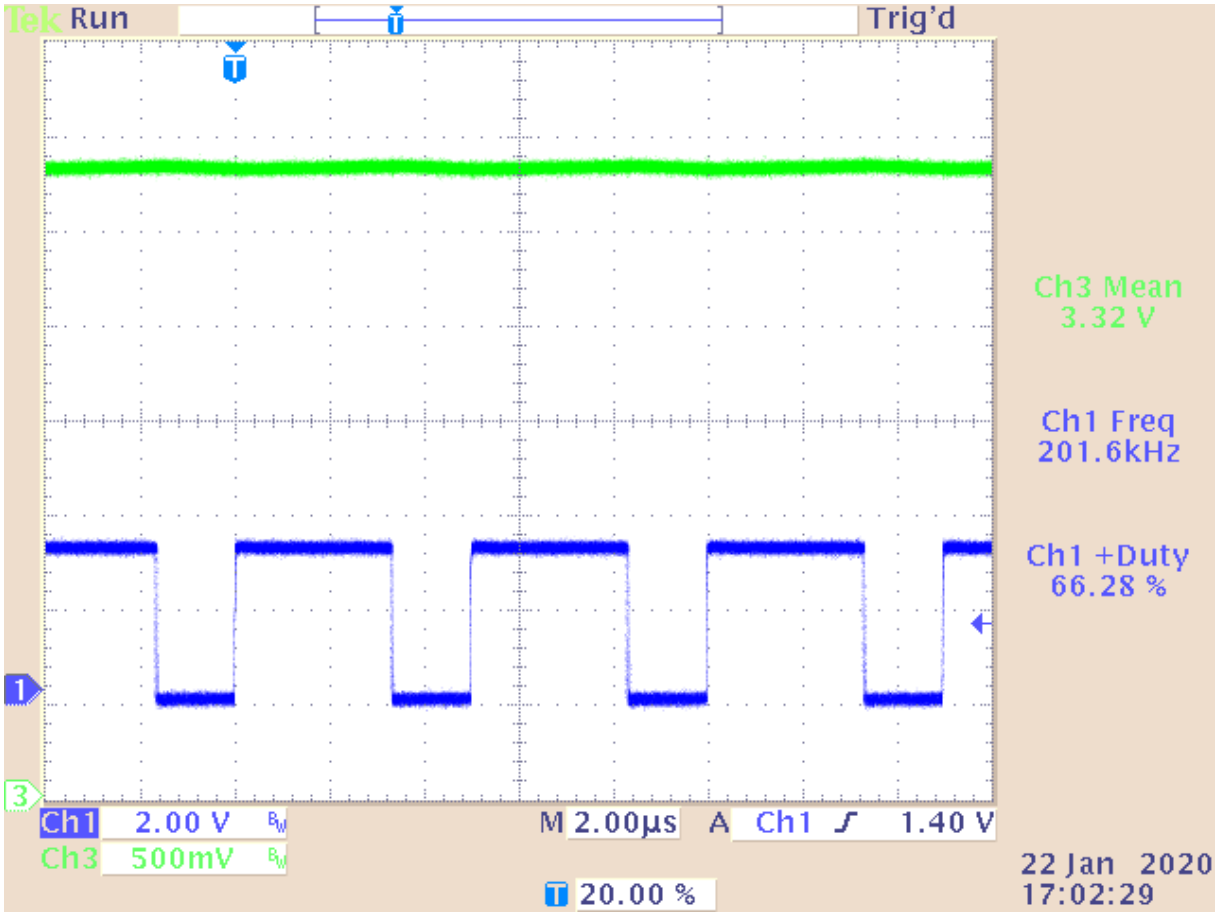


Figure 99. **Output voltage (Ch3) and PWM (Ch1) at 100% load**



10.2 Transient response tests

As discussed in Section 9 Getting started, the transient response can provide useful information about the stability of the closed-loop system. The transient response of the buck converter can be measured by placing one oscilloscope channel on the output voltage and another oscilloscope channel on the test point associated with the onboard load being switched. The output voltage channel is AC coupled to see the deviation from the setpoint at the moment of the load transient. More details on how to measure the transient response are provided in Section 9 Getting started.

Figure 100. Output voltage (Ch3) transient from load change 50% to 100% (Ch1), output voltage undershoot = 40 mV, settling time = 300 µs

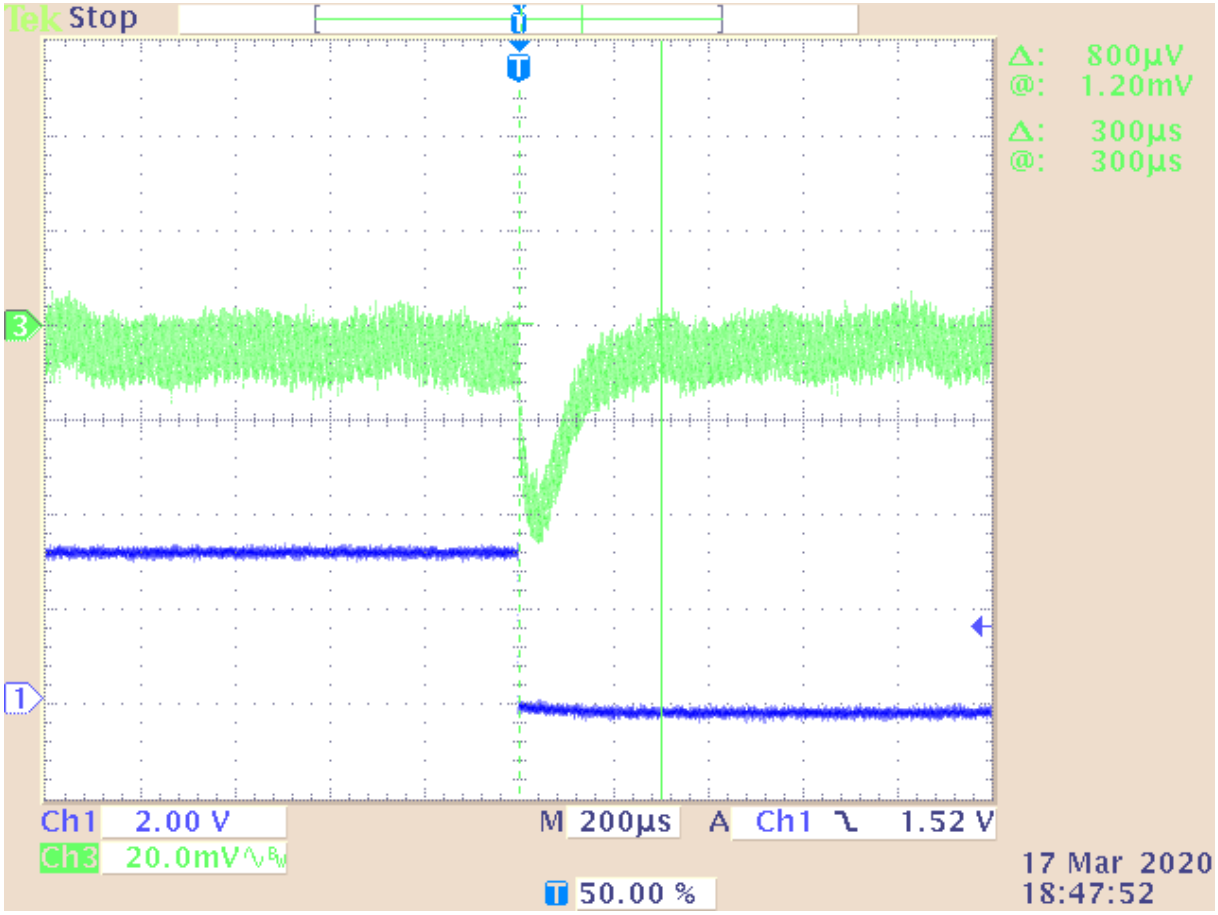
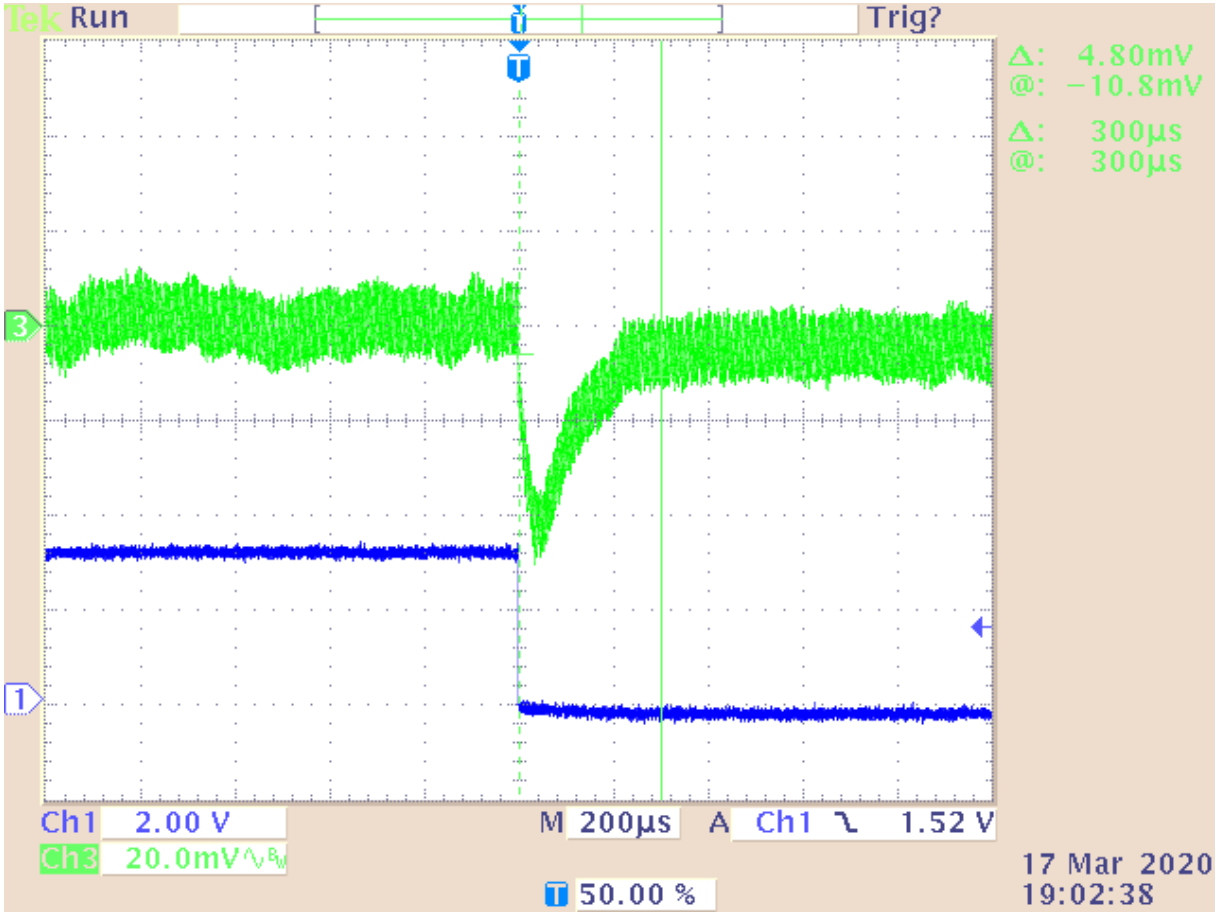


Figure 100 shows the transient response for a step-change in load from 50% to 100%. The output voltage on Channel 3 deviates from the steady state by 40 mV and recovers back to a steady state within 300 µs. Importantly, there is no ringing in the recovery. This is an indication that there is a sufficient phase margin in the loop to ensure stability. A transient response with ringing is shown in Section 9 Getting started.

It is prudent to check the transient response across a range of line and load conditions. In Figure 101 the transient response is shown for a step-change in load from 0% to 50%. As this is a synchronous buck converter under peak-current mode control, the dynamics of the system do not change from light load to full load. The undershoot is 40 mV and the recovery takes 300 µs. The recovery shows no sign of oscillation and therefore indicates that the system is stable.

Figure 101. **Output voltage (Ch3) transient from load change 0 to 50% (Ch1), output voltage undershoot = 40 mV, settling time = 300 µs**



10.3 Frequency response measurement results

The transient response can provide some information about the stability of the system and how the system responds to a large signal disturbance. However, it does not provide all of the information necessary to quantify the stability of the system. This information can be obtained by measuring the frequency response of the converter. A detailed procedure describing the measurement of the frequency response of the buck converter is given in Section 9 Getting started.

Figure 102. Measured loop response of the digital peak current mode loop at 100% load

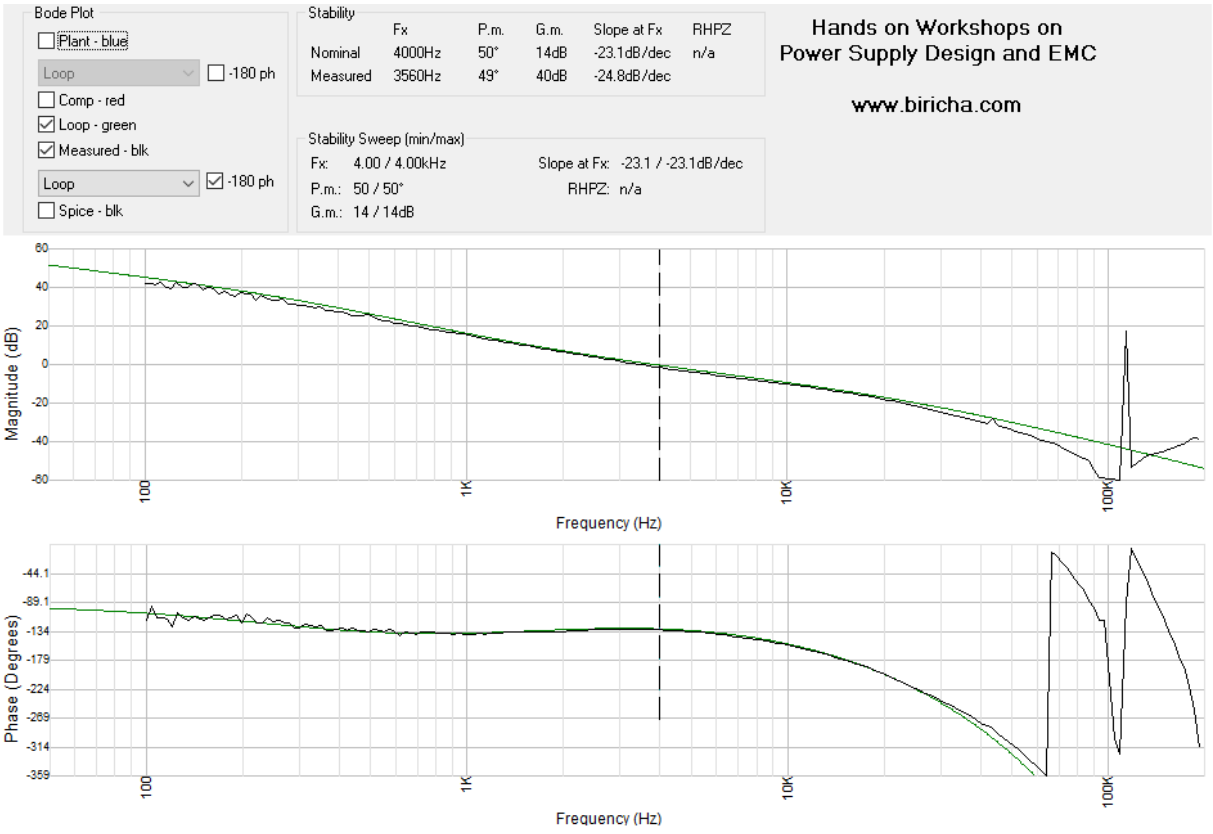
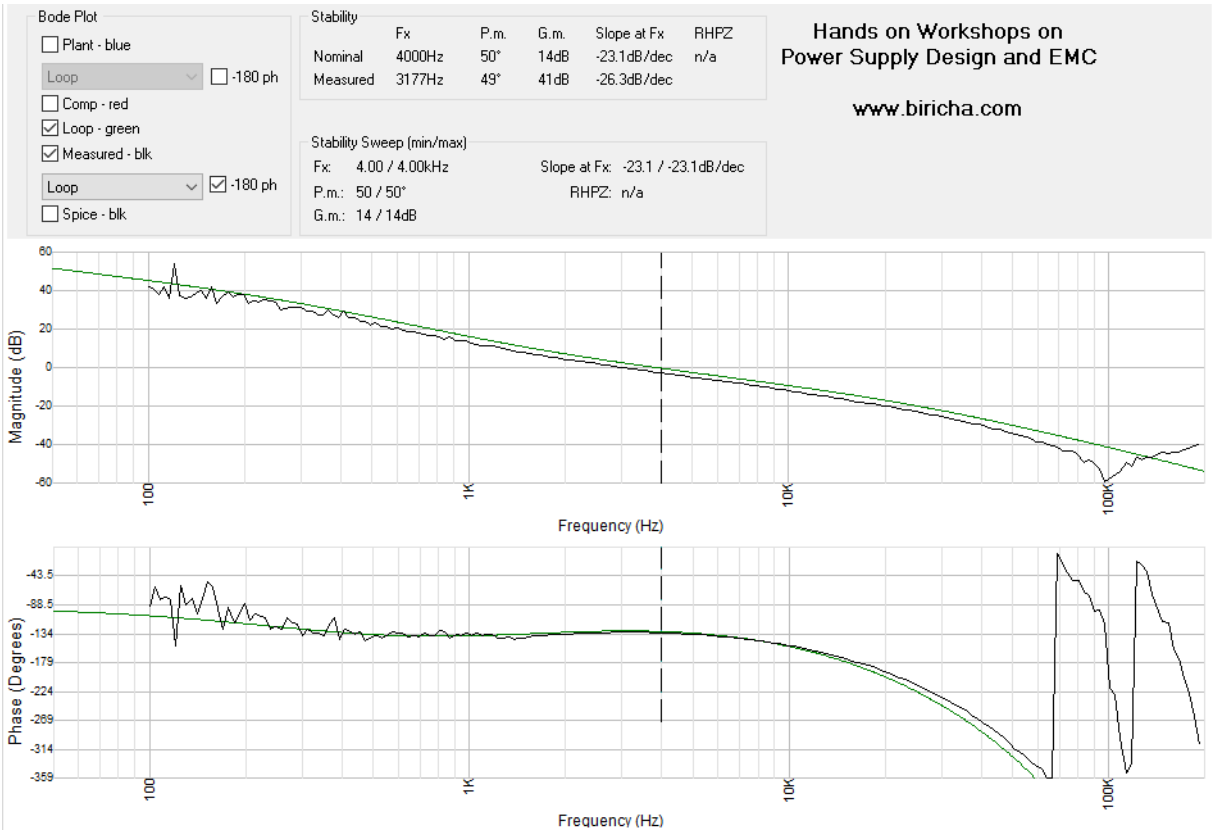


Figure 102 shows the measured control-to-output loop response of the buck converter at 100% load. The measurement of the loop is plotted as the black trace and compared with the simulated loop in green. A crossover frequency of 3.56 kHz is measured which is close to the desired crossover frequency of 4 kHz. A phase margin of 49° is achieved in the real measurement compared to 50° in the simulation. The discrepancy between the measured and simulated results is predominately due to the AC resistance of the inductor which is not included in the plant model.

In Figure 103 the loop response of the buck converter at 50% load is measured. As expected, the overall loop response is very similar to that of the system at 100% load. However, most notably, the DC gain changes. This is because the plant's DC gain is a function of the output load in a peak current mode-controlled converter. Therefore, the crossover frequency and phase margin change slightly, and the system is still stable.

Figure 103. Measured loop response of the digital voltage mode loop at 50% load

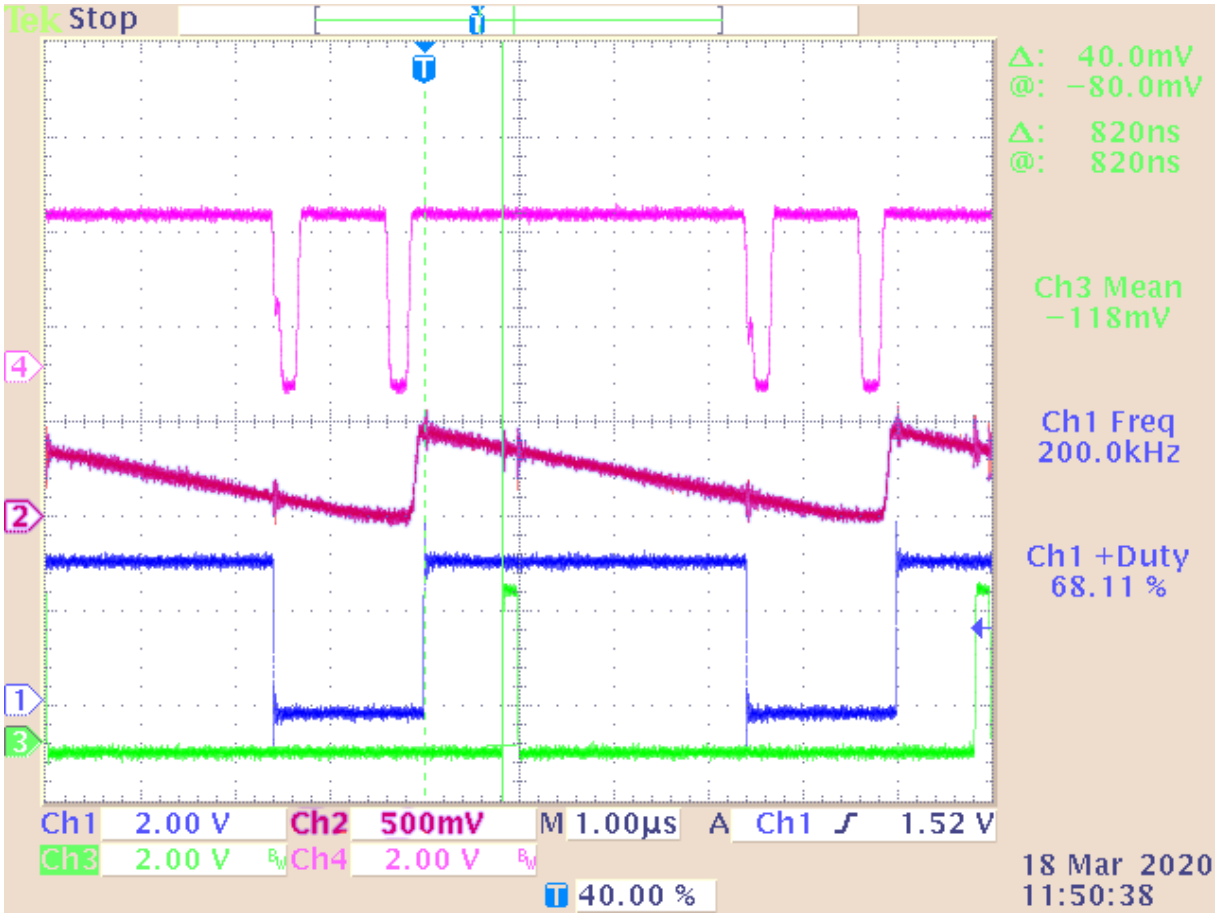


10.4 ISR plots (featuring FMAC and CPU load benefits)

The controller for this buck converter is implemented using the FMAC module onboard the STM32G4 device. The FMAC module is a hardware module that can execute the controller in a few system clock cycles and is not using up any of the main core bandwidth. An ISR is called when the FMAC has finished the computation of the controller. To measure the timings, a GPIO pin is set HIGH upon entering the FMAC ISR.

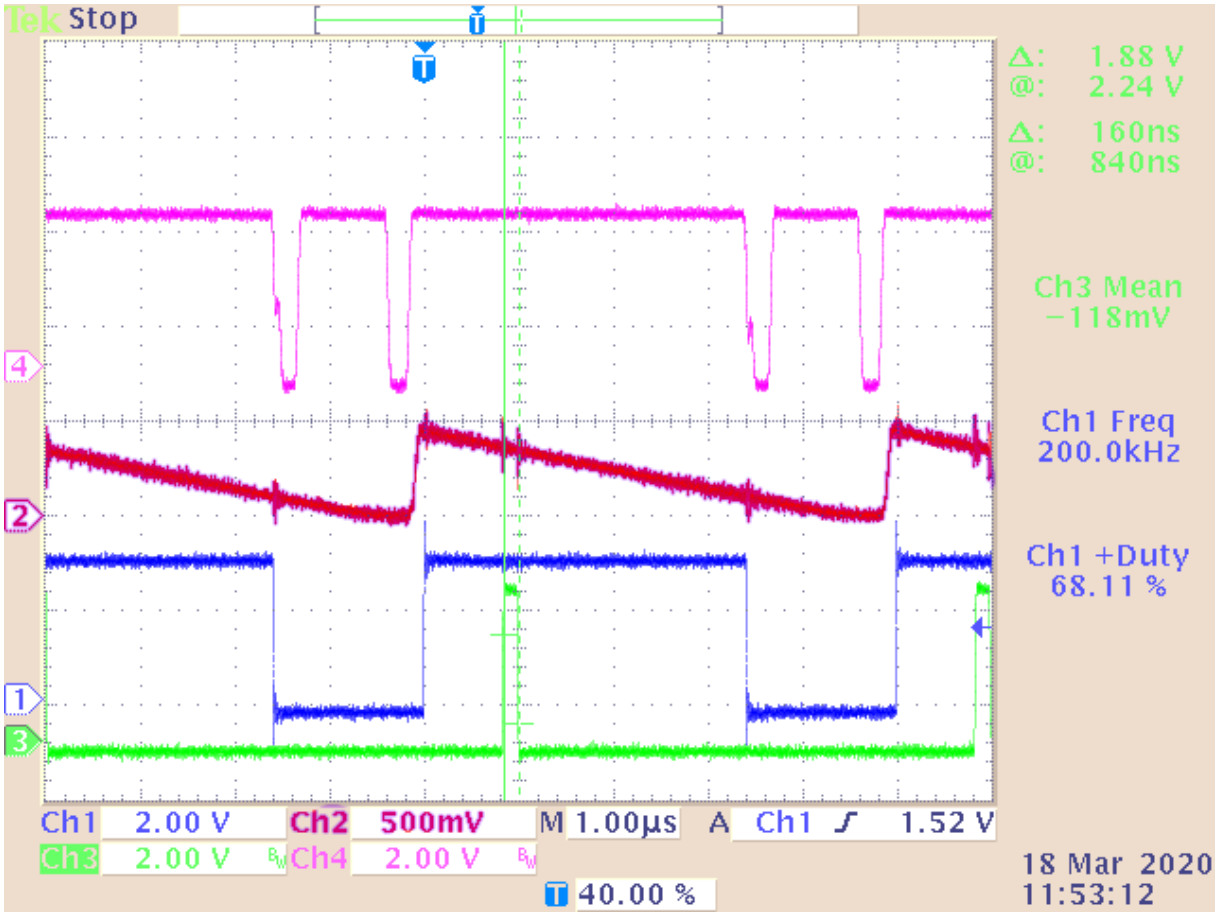
In Figure 104 the time from the ADC trigger to the FMAC ISR interrupt is measured. The ADC is triggered when the compare unit 3 match occurs 300 ns after the rising edge of the PWM signal, Channel 1, in Figure 104. When the ADC is triggered, the sample of the output voltage is taken, converted, and then copied to the FMAC using the DMA controller. After the FMAC has finished execution the ISR is triggered. The GPIO pin is then set HIGH as the MCU enters the ISR as shown on Channel 4 in Figure 104. The total time for this sample, conversion, and calculation process is measured as 820 ns - 300 ns = 520 ns.

Figure 104. PWM (Ch1) and FMAC ISR duration (Ch3). ADC trigger to ISR pin HIGH = 820 ns - 300 ns = 520 ns



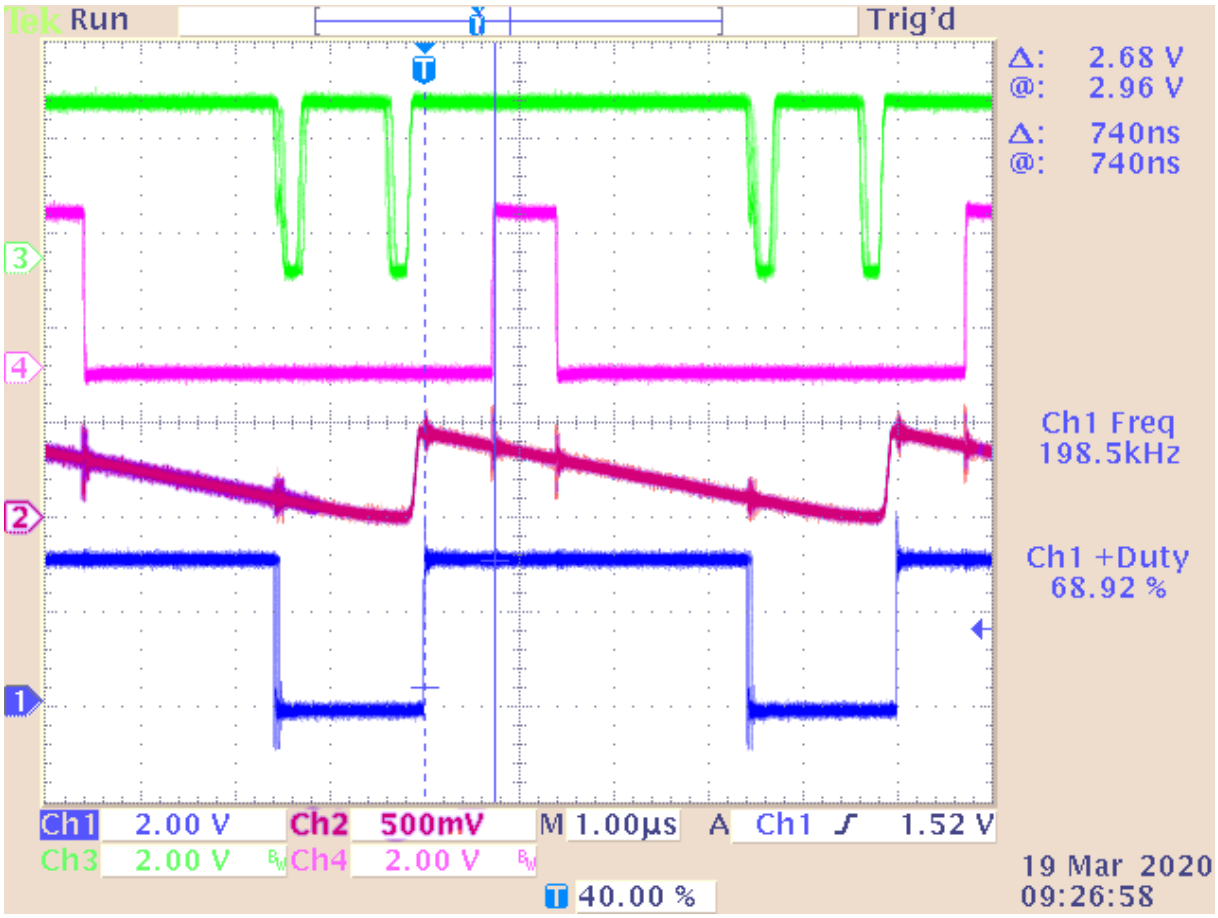
The FMAC ISR is solely used for bounds checking of the controller output to a minimum and maximum value and the update of the counter compare module register to set the new demand peak current reference value. Therefore, this is a very brief ISR containing only a few lines of code. The duration of the ISR is measured in Figure 105 as 160 ns. Although this timing measurement does not include the time required to push and pop the stack before and after the ISR, there is still clearly ample bandwidth remaining on the MCU to run other controllers or implement other functions.

Figure 105. **PWM (Ch1) and FMAC ISR duration (Ch3), ISR duration = 160 ns**



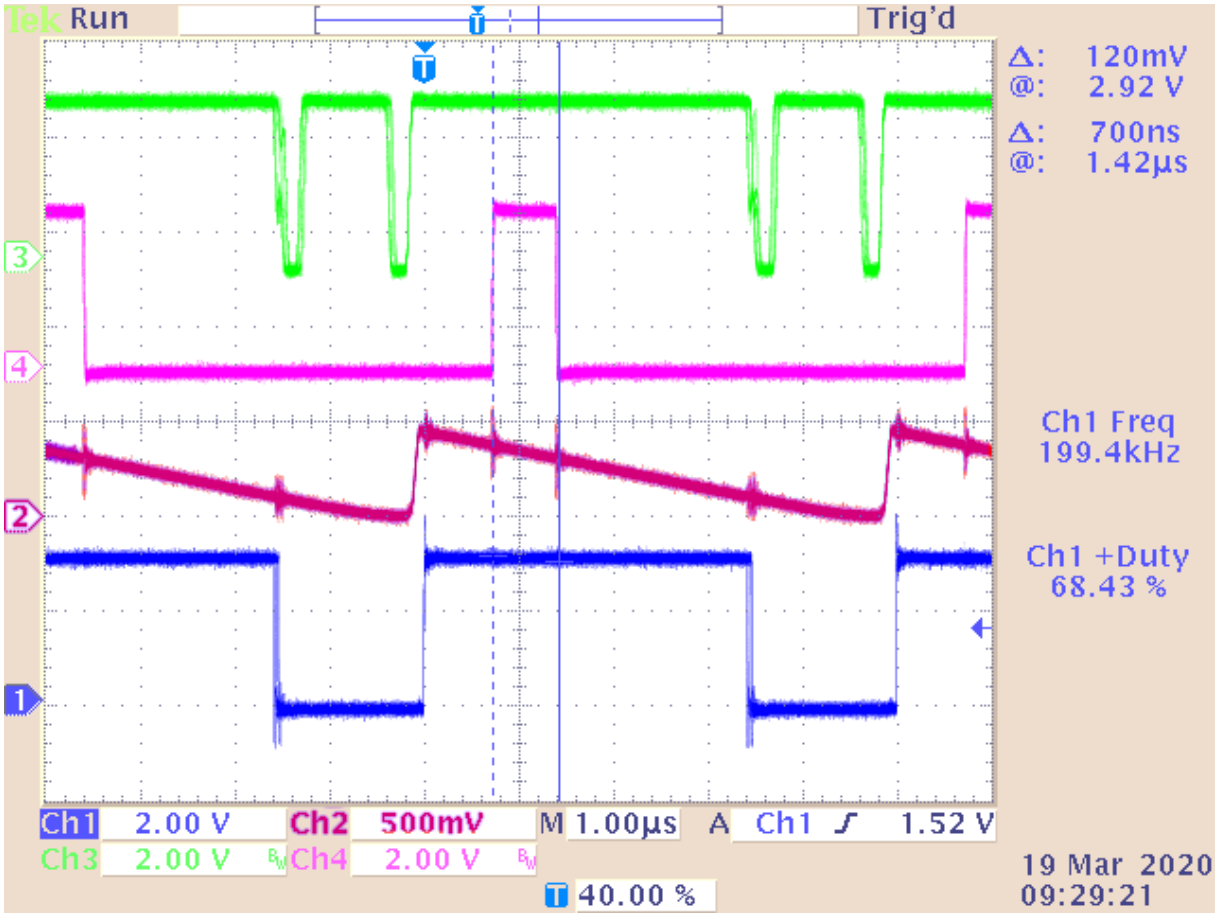
For STM32 MCUs without the FMAC module, the controller can be executed on the main core within the ADC ISR. The process for execution is then as follows. The ADC is triggered at the same point in time, the ADC samples the output voltage and completes the conversion. Now an interrupt is triggered which is serviced by the main core. The time from the ADC trigger to entering the interrupt can be measured from Figure 106 as a GPIO is again set HIGH as the ISR is entered. This time is measured as 440 ns as the PWM rising edge occurs 300 ns before the ADC is triggered.

Figure 106. PWM (Ch1) and ADC ISR duration (Ch4), ADC trigger to ISR pin HIGH = 740 ns - 300 ns = 440 ns



Within the ADC ISR, the 2p2z controller is executed. This means that the duration of the ISR is significantly longer than the FMAC ISR. The ADC ISR containing the 2p2z controller executes within 700 ns as measured in Figure 107. Therefore, the total time from the ADC trigger to the PWM update is 1.15 µs. This compares to 940 ns when the controller is implemented using the FMAC. However, the main core is now occupied for an additional 600 ns versus running the controller on the FMAC. Nevertheless, there is ample bandwidth to run other controllers or perform other functions when using either the main core or FMAC to implement the controller.

Figure 107. **PWM (Ch1) and ADC ISR duration (Ch4), ISR duration = 700 ns**



11 Summary

This application note details the design and operation of the digitally controlled peak-current-mode buck converter onboard the Discovery kit. The controller is implemented using the FMAC onboard the STM32G4 MCU from STMicroelectronics.

It is shown that, by applying control theory, the buck current mode buck converter can be stabilized to achieve a high crossover frequency and phase margin with an ideal transient response. All of the equations required to calculate the digital controller coefficients are provided in this application note. The software tool ST-WDS from Biricha can also be used to perform the same calculations with ease and can be used to obtain the required controller coefficients.

Several webinars are created to accompany this application note. Visit *www.biricha.com/st* to access the webinars related to this Discovery kit.

Biricha Digital Power also runs regular hands-on training workshops that cover the principles of power supply design, the fundamentals of control theory, the transition to the discrete-time domain, and step-by-step embedded programming. These workshops are beneficial to both analog power supply design engineers who need to get familiar with digital power and embedded systems engineers who need to understand how to design and stabilize digital power supplies.

For more information on these workshops, visit *www.biricha.com/st*

12 References

Biricha. (2020). Biricha Digital Power Workshops. *Digital Power Supply Design Using STM32*.

Brown, A. R., & Middlebrook, R. D. (1981). Sampled-Data Modeling of Switched Regulators. *proc. IEEE PESC'81*, 349-369.

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

Verghese, G. a. (1989). Averaged and sampled-data models for current-mode control: a re-examination. *Power Electronics Specialists Conference, 1989. PESC '89 Record., 20th Annual IEEE*, 484 -491 vol.1.

13 Download links

- Discovery kit schematic and PCB design files:
  www.st.com/stm32g4-dpower-disco
- Workshops and training:
  www.biricha.com/st
- Project, ST WDS, Bode Analyzer Suite downloads:
  www.biricha.com/ST-Discovery-Kit

Revision history

Table 10. **Document revision history**

Date	Version	Changes
19-Jun-2020	1	Initial release.
9-Dec-2022	2	Added *Section 9.9 Buck converter usage using the X-CUBE-DPower pack*.
6-Jan-2023	3	Updated Introduction about the buck current mode usage with X-CUBE-DPOWER.

Contents

List of tables

List of figures