# Guidelines for the buck voltage mode on the B-G474E-DPOW1 Discovery kit

## Introduction

The B-G474E-DPOW1 discovery kit is a complete digital power starter kit controlled by the 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 document focuses on the onboard step-down converter of the discovery kit. It details: the principles of voltage-mode control, and how to design a compensator to stabilize and regulate the voltage-mode controlled step-down converter. It explains how to implement these on the STM32 microcontroller.

The document also presents the buck voltage-mode usage with the X-CUBE-DPOWER STM32Cube expansion package.

Figure 1. **B-G474E-DPOW1 top view**

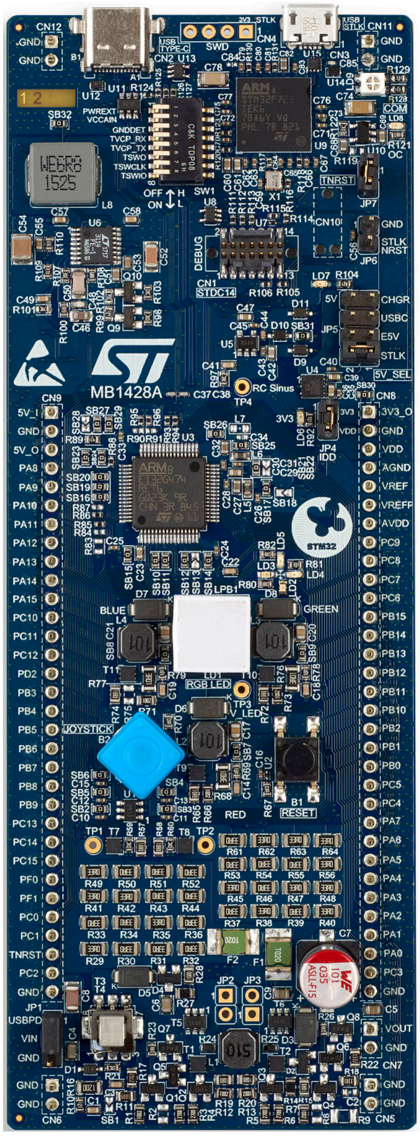
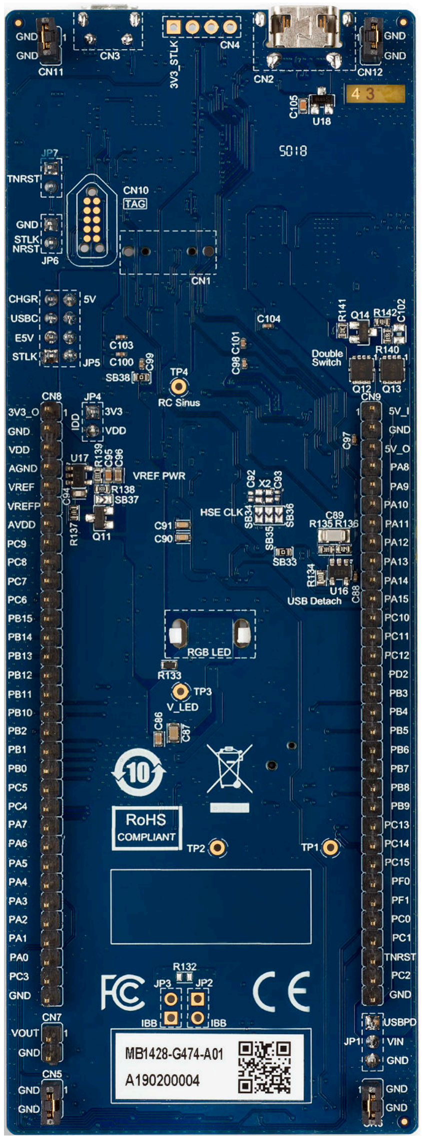


Figure 2. **B-G474E-DPOW1 bottom view**



*Pictures are not contractual.*

# 1 General information

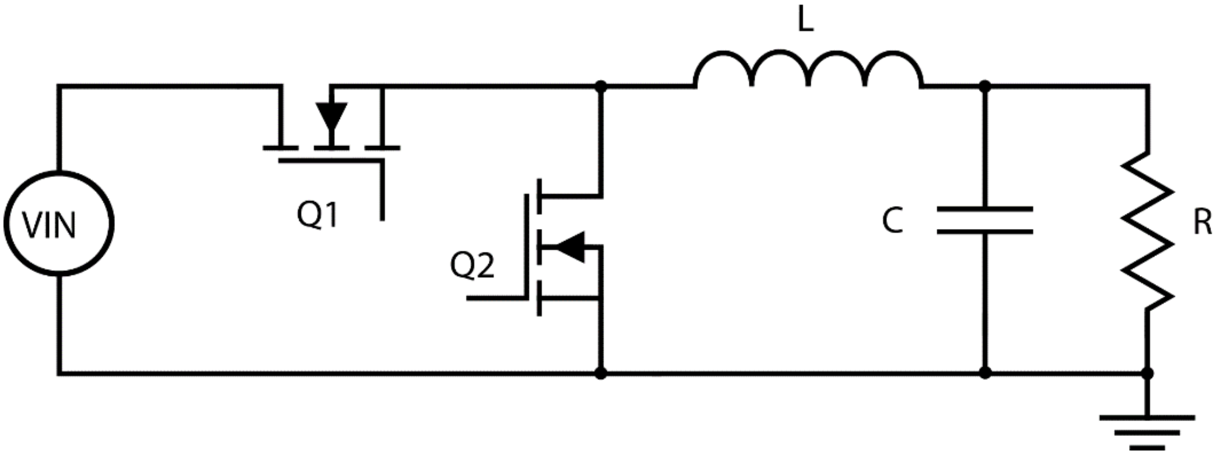
The B-G474E-DPOW1 discovery kit runs with the STMicroelectronics STM32G474RET6 microcontroller, based on an Arm® Cortex®-M4 core.

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

# 2 Step-down converter operation

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

**Figure 3. Simplified power stage schematic**

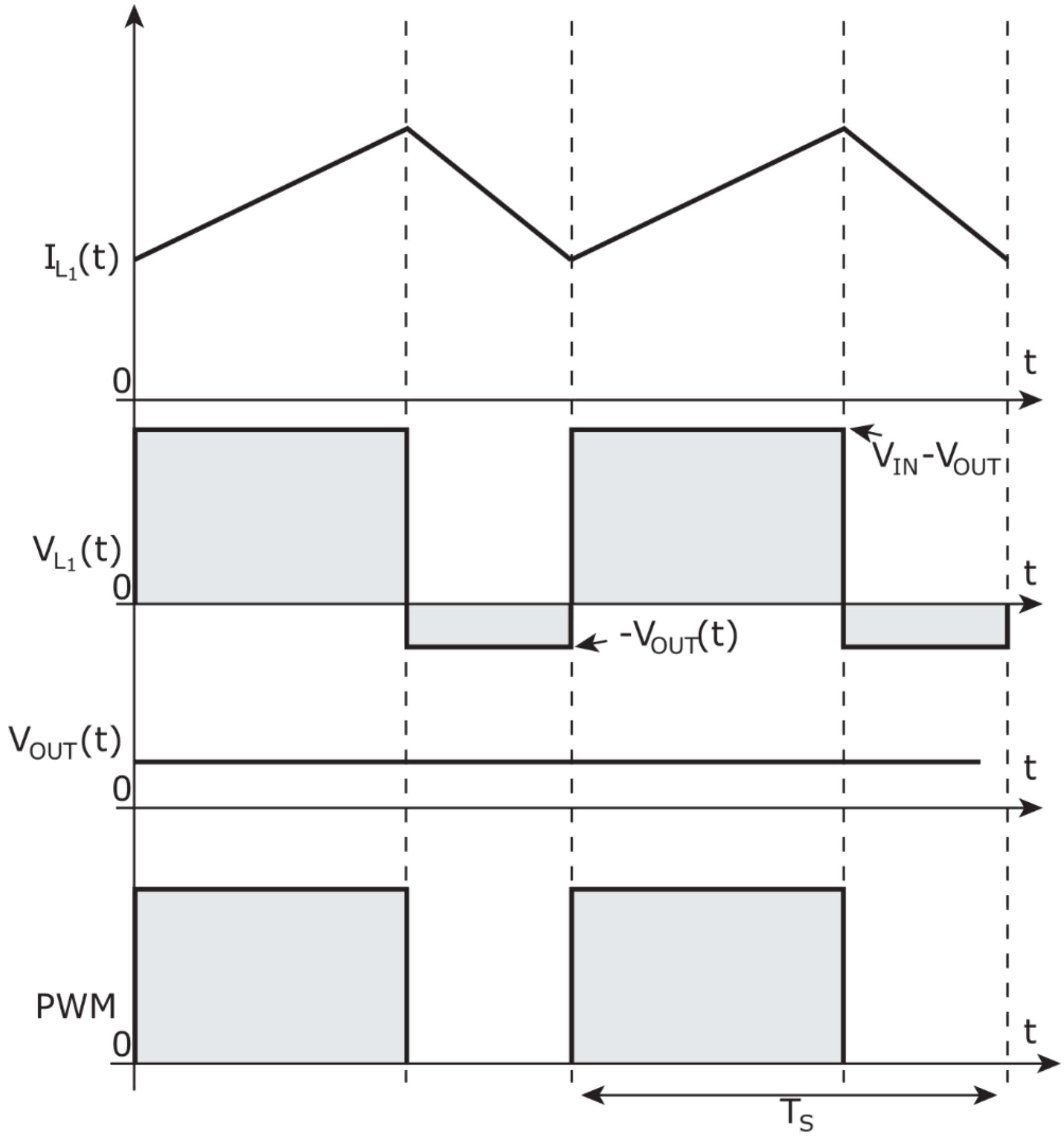



## 2.1 Principle of operation

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 waveforms of Figure 4.

Figure 4. **Step-down converter operational waveforms**




The output filter capacitor $C_{out}$ filters the AC component of this current while the DC component of this current is the output load current, $I_{out}$. 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 as:

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

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

# 3 Voltage mode control explained

## 3.1 Advantages and disadvantages

Voltage mode control is one of the most popular control methods due to its simplicity and effectiveness in regulating an output voltage given any changes in load. Under voltage mode control, the output voltage is measured and compared with the desired set point or reference. The difference between the actual output voltage and the desired output voltage is used as an input to a controller and the output of the controller determines the new value of the duty cycle to close the control loop.

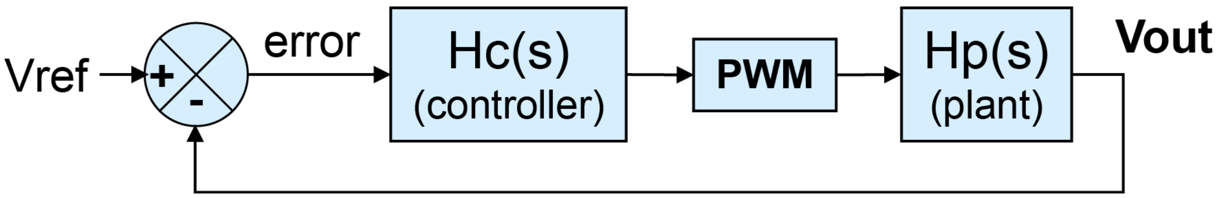
Therefore, unlike other methods of control such as peak current mode, the voltage mode controller only needs information about the output voltage to close the control loop and provide a regulated and stable output voltage. Typically, some current information is required to provide an overall power limit. Alternatively, this can be crudely achieved by limiting the maximum duty cycle.

Further advantages of voltage-mode control include the ability to step down the voltage by a significant amount and maintain proper regulation for small and no loads. The PCB layout process for a voltage-mode controlled step-down converter is typically simplified due to not requiring a switch or inductor current sense transducer, and voltage mode control is less sensitive to nonoptimal PCB layouts.

As discussed so far, voltage mode control has good load regulation. It compensates for any changes in load to maintain a set output voltage. However, in its standard implementation, voltage mode control has poor line regulation. There is a delay between the line voltage increasing or decreasing and the duty cycle adjusting to compensate for the subsequent change in output voltage. Therefore, it is common to add a `feed-forward` term to the control loop, which is a measure of the input voltage.

## 3.2 Step-by-step control loop

**Figure 5. Step-down converter voltage control loop**




The simplified control loop of a voltage mode converter is shown in Figure 5. Here, the power stage containing the switching MOSFET, output filter inductor, and output filter capacitor are all shown within the block $H_p(s)$. This is referred to as the plant transfer function. The controller, which is designed to compensate for this control loop, is shown as the block $H_C(s)$. This is referred to as the controller transfer function. The PWM block is usually included as part of the plant, $H_p(s)$ block, however, it is separated so that the effects of this block can be analyzed independently.

For this discussion, the current work is performed in the continuous-time domain, also known as the s-domain, and this is why the plant and controller are a function of the Laplace operator. The translation from the continuous-time domain to the discrete-time domain is discussed later in this application note.

At a steady state, the plant power stage provides a fixed output voltage given the fixed input voltage and duty cycle. In Figure 5 the output voltage of the power stage is fed back and compared with a reference, $V_{REF}$. This is the desired output voltage and therefore if there is any deviation from the reference there is a nonzero error term.

The error term is used as an input to the controller, $H_C(s)$. The controller manipulates the error term in some way depending on the type of controller that is designed. The output of the controller is the duty cycle, which is used as an input to the power stage and modulates the MOSFET switches.

This closed-loop control process can be easily visualized through a numerical example. For now, let us assume that the controller $H_C(s)$ is a simple proportional controller with unity gain and the reference is the desired output voltage of 3.3 V.

If the output voltage dropped down to 3.2 V due to an increase in output load, then the error term can be calculated as follows:

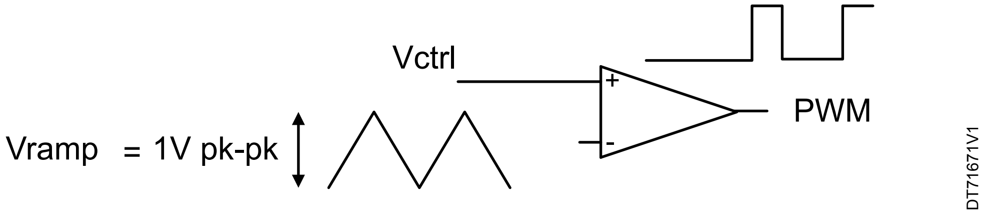
$$V_{ERROR} = V_{REF} - V_{OUT}$$
$$V_{ERROR} = 3.3\,V - 3.2\,V$$
$$V_{ERROR} = 0.1\,V$$

As the proportional controller has unity gain, the output of the controller can be calculated as follows:

$$H_C(s) = \frac{Y(s)}{X(s)}$$
$$Y(s) = H_C(s) \times X(s)$$
$$Y(s) = 1 \times 0.1\,V$$
$$Y(s) = 0.1\,V$$

The output of the controller is then used as an input for the PWM block. The PWM block takes the controller output and converts this to an effective duty cycle. There are several different methods for achieving this and, for analog voltage mode control, this is typically achieved using a PWM comparator and an RC ramp as shown in Figure 6.

**Figure 6. PWM comparator comparing RC ramp to control voltage**




Within the PWM block the control voltage, $V_{CTRL}$, which is the output of the previous controller block, is compared with an RC ramp using a comparator. The output of the comparator is high when the control voltage is larger than the RC ramp voltage. Therefore, to achieve a 100% duty cycle the control voltage needs to be equal to the maximum RC ramp height. This is typically internal to many analog ICs and the implementation differs in digital and is discussed later in this application note. If for now, it is assumed that the maximum ramp height is 1 V, then the duty cycle can be calculated as follows:

$$Duty = Y(s) \times PWM_{COMP}$$
$$Duty = 0.1\,V \times \left(\frac{100\,\%}{1\,V}\right)$$
$$Duty = 10\,\%$$

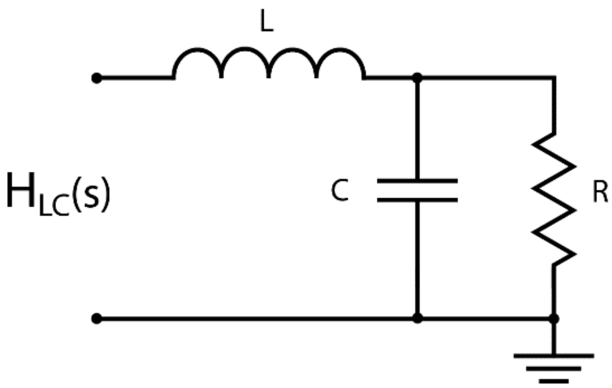
This is then used as an input to the plant block, which contains the power stage and thus subsequently determines the output voltage and the loop is now closed. This simplified analysis helps to explain the different blocks within the closed-loop system. However, in reality, the analysis performed is a small-signal analysis to determine the s-domain transfer functions for the different blocks within the system.

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.

## 3.3 Buck plant transfer function

The derivation of the buck power stage transfer function is straightforward under voltage-mode control. The output filter inductor resonates with the output filter capacitor forming a double pole in the plant transfer function. Consider the output LC filter of the step-down converter shown in Figure 7.

**Figure 7. LC output filter of the step-down converter power stage**




The transfer function of the LC filter can be determined using Laplace and by analyzing the impedances in series and parallel.

$$Z_1 = sL$$

$$Z_2 = \frac{1}{sC} \parallel R$$

$$Z_2 = \frac{R.\frac{1}{sC}}{R + \frac{1}{sC}}$$

$$Z_2 = \frac{R}{sCR + 1}$$

$$H_{LC}(s) = \frac{Z_2}{Z_1 + Z_2}$$

$$H_{LC}(s) = \frac{\frac{R}{sCR+1}}{sL + \frac{R}{sCR+1}}$$

$$H_{LC}(s) = \frac{R}{sL(sCR + 1) + R}$$

$$H_{LC}(s) = \frac{R}{s^2LCR + sL + R}$$

$$H_{LC}(s) = \frac{1}{s^2LC + s\frac{L}{R} + 1}$$

Putting this in the standard form for a second-order polynomial:

$$H_{LC}(s) = \frac{1}{\frac{s^2}{\omega_{LC}{}^2} + s\frac{1}{Q\omega_{LC}} + 1}$$

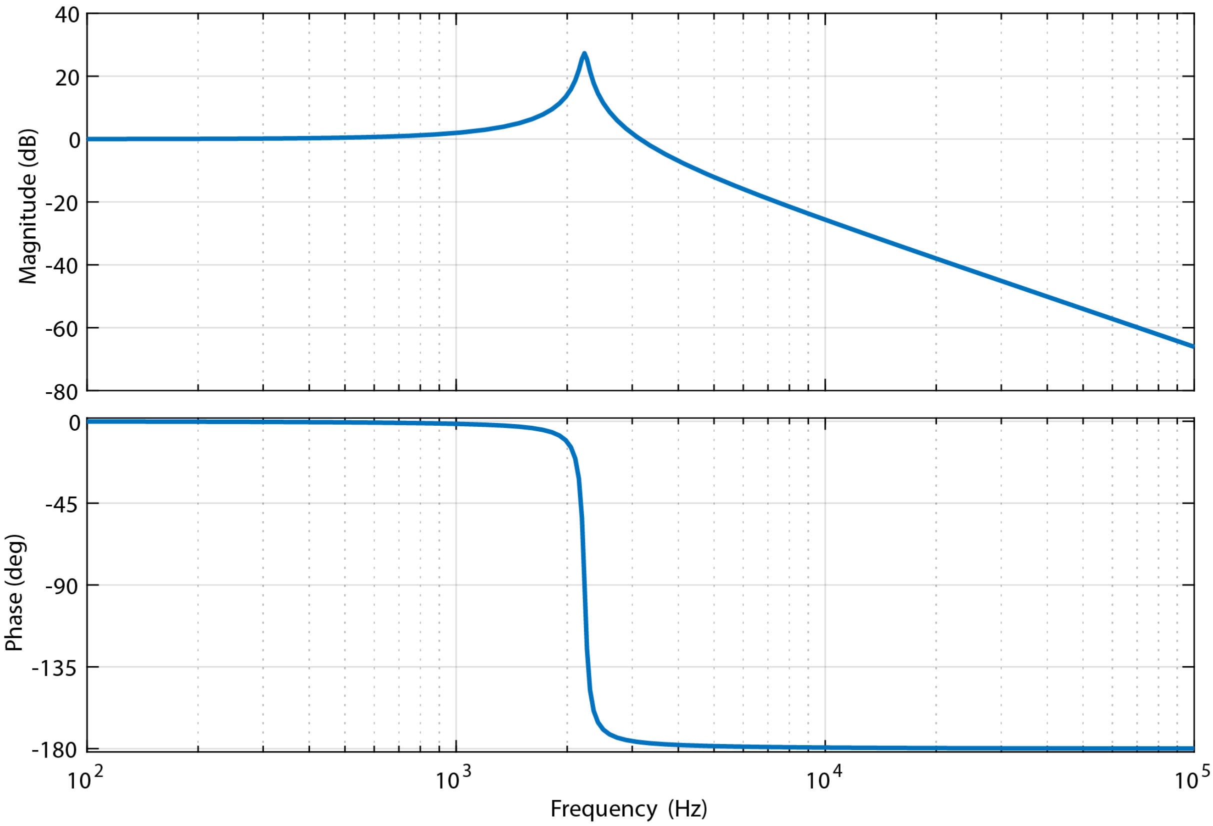
Therefore:

$$\omega_{LC} = \frac{1}{\sqrt{LC}}$$

$$Q = \frac{1}{2\zeta} = R\sqrt{\frac{C}{L}}$$

From this analysis, it can be seen that the double pole has an undamped natural frequency, $\omega_n$, and a resonant peak depending on the Q. The typical Bode plot of this double pole output filter is shown in Figure 8.

**Figure 8. Bode plot of the LC filter transfer function**




In the Bode plot shown in Figure 8, the low-frequency gain is 0 dB. This is because the LC filter has no effect at low frequencies. However, in reality, there is some low-frequency gain introduced by the PWM block. As discussed earlier, the gain of this depends upon the height of the RC ramp in analog, in digital the implementation of the PWM gain differs as there is no RC ramp or analog comparator. The effect of the PWM block is that the low-frequency gain is shifted up by some amount. The amount by which the Bode plot is shifted can be calculated as follows:

$$G_{PWM} = \frac{output}{input}$$

$$G_{PWM} = \frac{V_{OUT(MAX)}}{V_{RAMP(MAX)}}$$

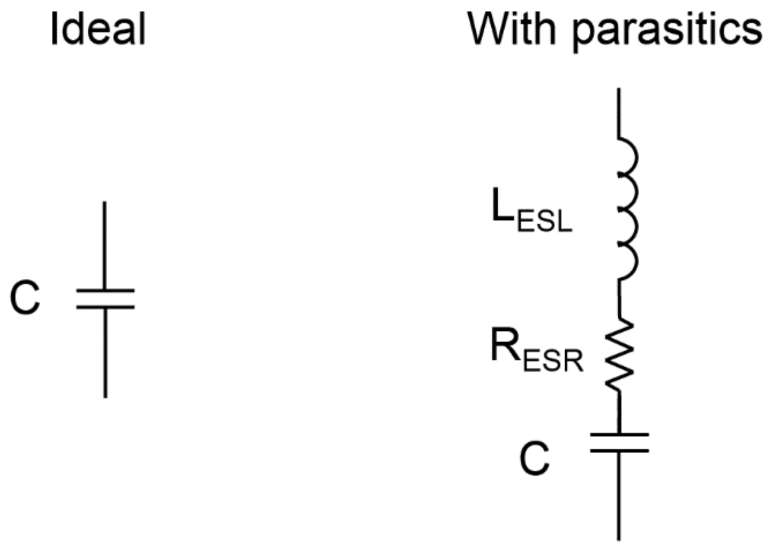
This equation is valid at other ramp heights provided that the corresponding duty cycle and output voltage are calculated. For example:

$$G_{PWM} = \frac{V_{OUT(@50\%\ duty)}}{V_{RAMP(@50\%\ duty)}}$$

## 3.4 Capacitor ESR zero

The model discussed so far does not include the parasitic elements of the output filter capacitor. The parasitic elements of the capacitor are shown in Figure 9.

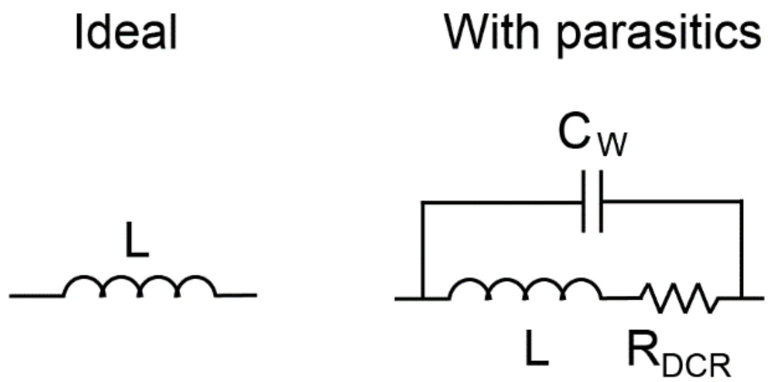
**Figure 9. Ideal capacitor and capacitor with parasitic elements**




The parasitic equivalent series resistance (ESR) has a significant impact on the plant transfer function for the step-down 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 10. 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 10. 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 (2).

$$H_P(s) = \frac{1 + \frac{s}{\omega_{ESR}}}{\frac{s^2}{{\omega_{LC}}^2} + s\frac{1}{Q\omega_{LC}} + 1} \tag{2}$$

The location of the zero is given in (3) and is dependent on the capacitance and the ESR value.

$$\omega_{ESR} = \frac{1}{C \cdot R_{ESR}} \tag{3}$$

The addition of the capacitor ESR and inductor DCR also affects the damping of the double pole and the natural frequency. The updated equation for the Q and natural frequency of the double pole is given in (4) and (5).

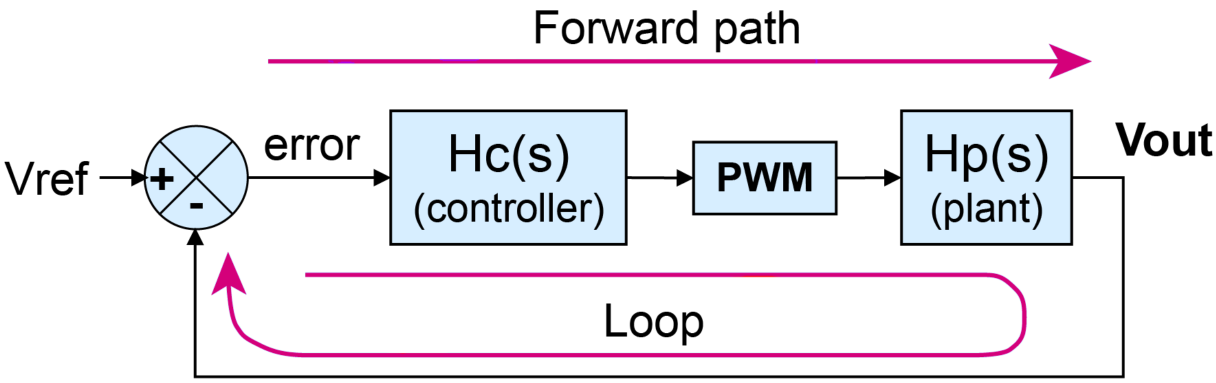
$$\omega_{LC} = \frac{1}{\sqrt{LC\left(1 + \frac{R_{ESR}}{R}\right)}} \tag{4}$$

$$Q = \frac{1}{\omega_{LC}\left(\frac{L}{R} + C\left(R_{ESR} + \left(1 + \frac{R_{ESR}}{R}\right)R_{DCR}\right)\right)} \tag{5}$$

# 4 Voltage mode compensator design

## 4.1 Loop stability criteria

**Figure 11. Closed and open-loop transfer functions**




Consider the control loop of the step-down converter shown in Figure 11, where the forward and loop paths of the control loop are identified. The transfer function for the closed-loop system is given in (6).

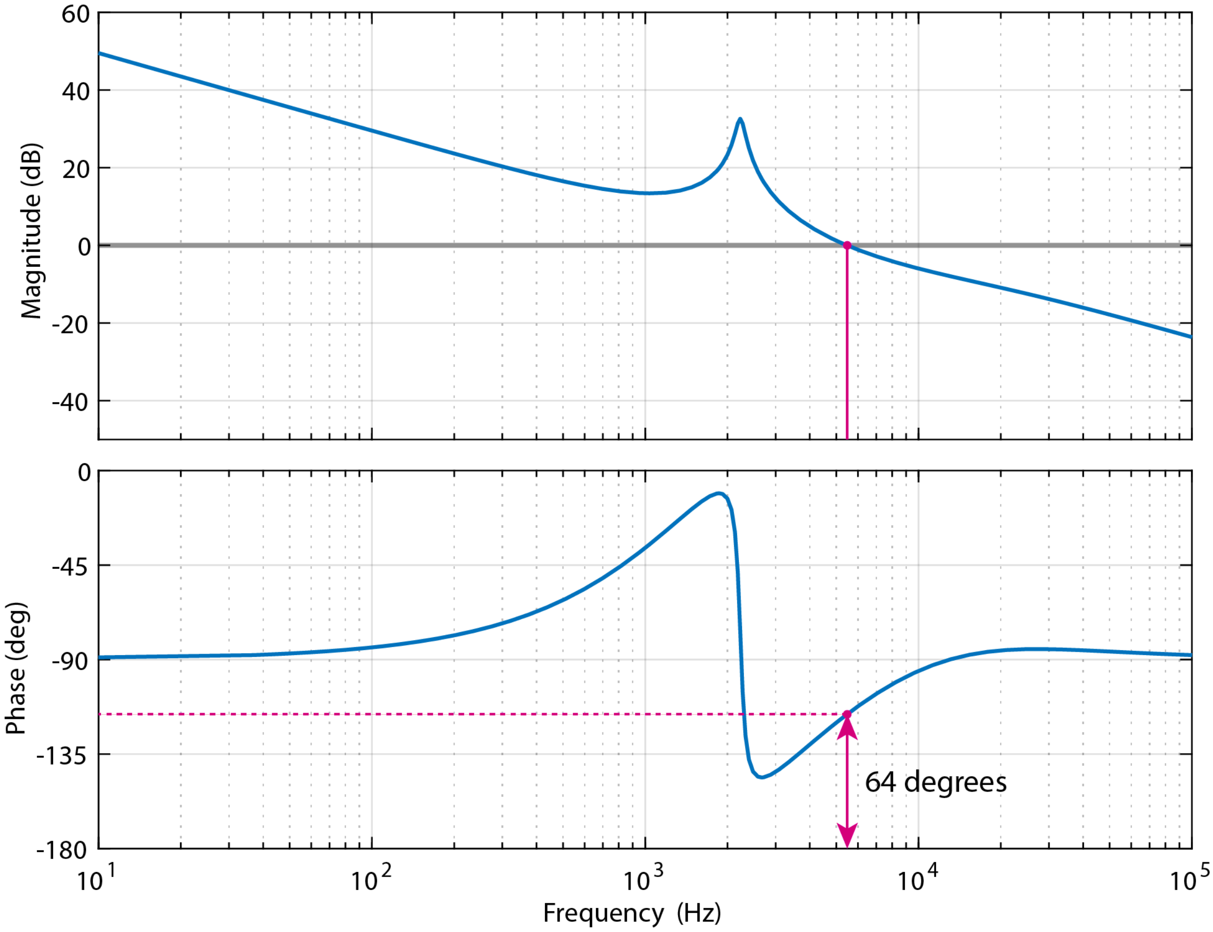
$$TF = \frac{Forward}{1 - Loop} \quad (6)$$

The closed-loop transfer function can be derived by applying this to Figure 11:

$$H_{CL}(S) = \frac{Forward}{1 - (-Loop)} = \frac{H_c(S) \,.\, G_{PWM} \,.\, H_p(S)}{1 + H_c(S) \,.\, G_{PWM} \,.\, H_p(S)} \quad (7)$$

The loop response, also called the open-loop, is the controller 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 is shown in Figure 12.

**Figure 12. Open-loop Bode plot of the step-down converter: plant, PWM, and compensator**




Several terms can be defined from this Bode plot. The first is the crossover 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. It means 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 (7).

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 12.

## 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. In a standard analog voltage-mode controlled buck, the plant phase rolls off due to the double pole. The capacitor ESR zero adds phase to the loop however it cannot be relied upon to achieve stability. Furthermore, the op-amp internal to analog control ICs introduces a phase roll-off as the frequency approaches the bandwidth.

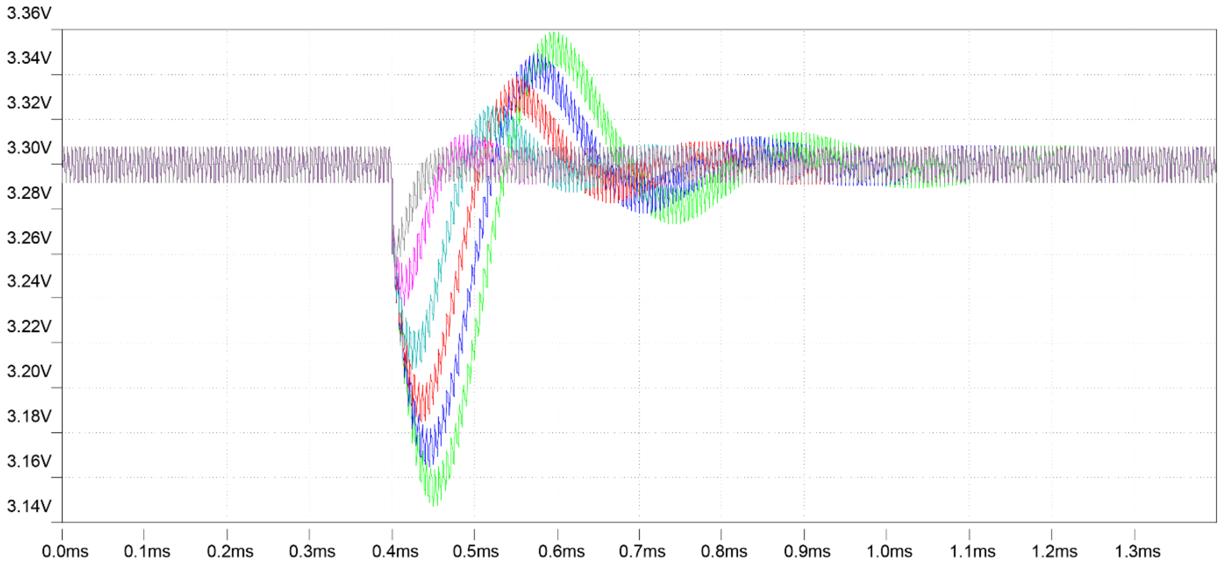
With a digital compensator, there is no analog op-amp bandwidth to consider. However, the delays within the digital system introduce a phase loss. As the frequencies approach the sampling frequency this phase loss becomes significant. A method for calculating the anticipated phase loss is covered later 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 result in a slower response to line and 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 13 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 13. Step response for a system with decreasing crossover and phase margin (grey to green)**



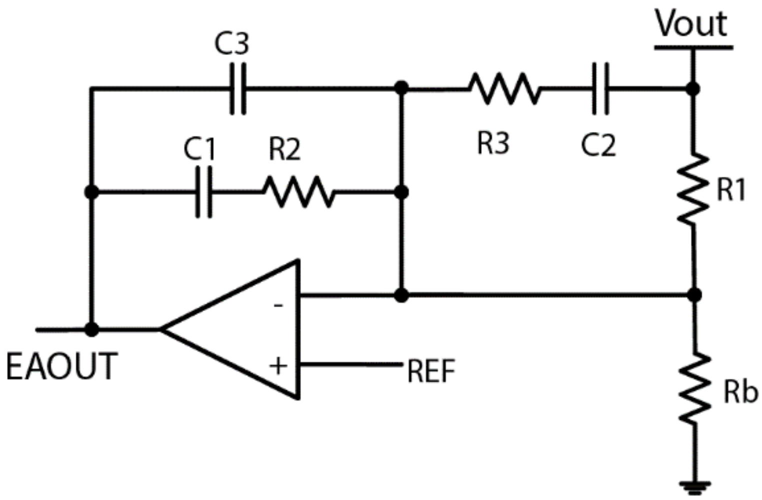

Phase margin specification: a minimum of 45° of phase margin at the crossover frequency, ideally between 50° and 60°

## 4.3 Types of compensators

A compensator needs to be designed and added into the loop. The aim is 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 combinations of capacitors and resistors determine 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 voltage mode control, the Type-III compensator is required. This is because Type-II does not provide enough zeros to compensate for the phase lag. It is due to the double pole of the voltage mode buck plant transfer function. The circuit for a Type-III compensator is shown in Figure 14.

**Figure 14. Type-III compensator implemented in analog using op-amp**




The transfer function for this circuit can be derived in the same manner as that of the buck power stage. For brevity, the full transfer function is included without derivation in (8).

$$H_C(s) = \left(\frac{\omega_{CP0}}{s}\right)\frac{\left(\frac{s}{\omega_{CZ1}}+1\right)\left(\frac{s}{\omega_{CZ2}}+1\right)}{\left(\frac{s}{\omega_{CP1}}+1\right)\left(\frac{s}{\omega_{CP2}}+1\right)} \quad (8)$$

The transfer function in (8) has two zeros, two poles, 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 (9) through (13).

$$\omega_{CZ1} = \frac{1}{R_2 C_1} \quad (9)$$

$$\omega_{CZ2} = \frac{1}{C_2(R_1 + R_3)} \quad (10)$$

$$\omega_{CP0} = \frac{1}{R_1(C_1 + C_3)} \quad (11)$$

$$\omega_{CP1} = \frac{(C_1 + C_3)}{R_2 C_1 C_3} \quad (12)$$

$$\omega_{CP2} = \frac{1}{R_3 C_2} \quad (13)$$

In digital, this compensator is implemented on the MCU. 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 double pole due to the resonance between the output filter inductor and capacitor.
- The plant has a single zero due to the ESR of the output filter capacitor.
- The plant has a DC gain.

The effects of the plant double pole can be partially canceled out by placing the two compensator zeros at the natural frequency of the double pole. It is of course not possible to completely cancel out the complex conjugate double pole of the plant with two real zeros in the compensator, however, it lessens the effect of the double pole and provide 180° of phase boost at around one decade above this frequency:

$$\{\omega_{CZ1}, \omega_{CZ2}\} = \omega_n = \frac{1}{\sqrt{LC}} \tag{14}$$

Now that the two zeros of the compensator are placed, the two poles of the compensator and the pole at the origin remain. One of the compensator's poles can be used to cancel out the parasitic ESR zero of 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.

$$\omega_{CP1} = \omega_{ESR} = \frac{1}{C.R_{ESR}} \tag{15}$$

The second compensator pole can be placed at half the switching frequency and in doing so the high-frequency gain is attenuated while not having a significant impact on the phase at around the crossover frequency.

$$\omega_{CP2} = \pi F_s \tag{16}$$

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 -20 dB/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 rejections and 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 in (17) must be solved for the compensator pole at the origin:

$$20\log_{10}|H_P(j\omega)| + 20\log_{10}|H_C(j\omega)| = 0\ dB \tag{17}$$

Given that the plant transfer function and compensator transfer functions are both known, this can be solved for the unknown term $\omega_{CP0}$ with the result shown in (18). Evaluating this equation determines the amount of gain which needs to be added by the compensator to achieve the specified crossover frequency.

$$\omega_{CP0} = \frac{\omega_X}{\frac{V_{IN}}{V_{RAMP}} \times \frac{\sqrt{1+\left(\frac{\omega_X}{\omega_{ESR}}\right)^2}}{\sqrt{\left(\frac{\omega_X}{\omega_{LC}\times Q}\right)^2+\left(1+\frac{-\omega_X{}^2}{\omega_{LC}{}^2}\right)^2}} \times \frac{\sqrt{1+\left(\frac{\omega_X}{\omega_{CZ1}}\right)^2}\times\sqrt{1+\left(\frac{\omega_X}{\omega_{CZ2}}\right)^2}}{\sqrt{1+\left(\frac{\omega_X}{\omega_{CP1}}\right)^2}\times\sqrt{1+\left(\frac{\omega_X}{\omega_{CP2}}\right)^2}}} \tag{18}$$

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 one of the compensator zeros to achieve the specified phase margin. This is achieved by solving the equality shown in (19) for the compensator zero $\omega_{CZ1}$.

$$\angle(H_P(j\omega).H_C(j\omega)) = -\pi + \theta_M \tag{19}$$

(19) 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 $\omega_{CZ1}$ and the result is given in (20).

$$\omega_{CZ1} = \frac{\omega_X}{\tan\left(-\frac{\pi}{2}+\Phi_M - tan^{-1}\frac{\omega_X}{\omega_{PP1}} - tan^{-1}\frac{\omega_X}{\omega_{PP2}} + tan^{-1}\frac{\omega_X}{\omega_{CP2}} - tan^{-1}\frac{\omega_X}{\omega_{CZ2}}\right)} \tag{20}$$

Where $\omega_{PP1}$ and $\omega_{PP2}$ are the complex conjugate poles of the double pole formed by the inductor and capacitor in the 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

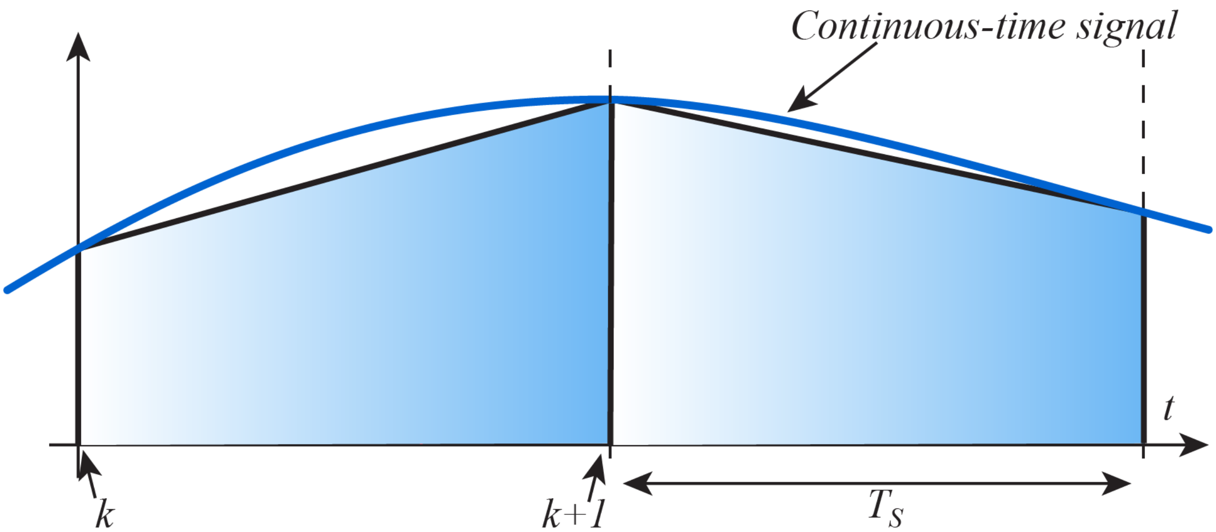
With the poles and zeros of the compensator placed in the continuous-time domain, the s-domain, they must be converted into the discrete-time domain to implement the controller on the MCU. To implement the discrete-time controller, the continuous-time Type III compensator, which is discussed in the previous 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 (21).

$$z = e^{sT} \tag{21}$$

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 15.

**Figure 15. Continuous-time signal sampled in discrete time using a trapezoidal approximation**

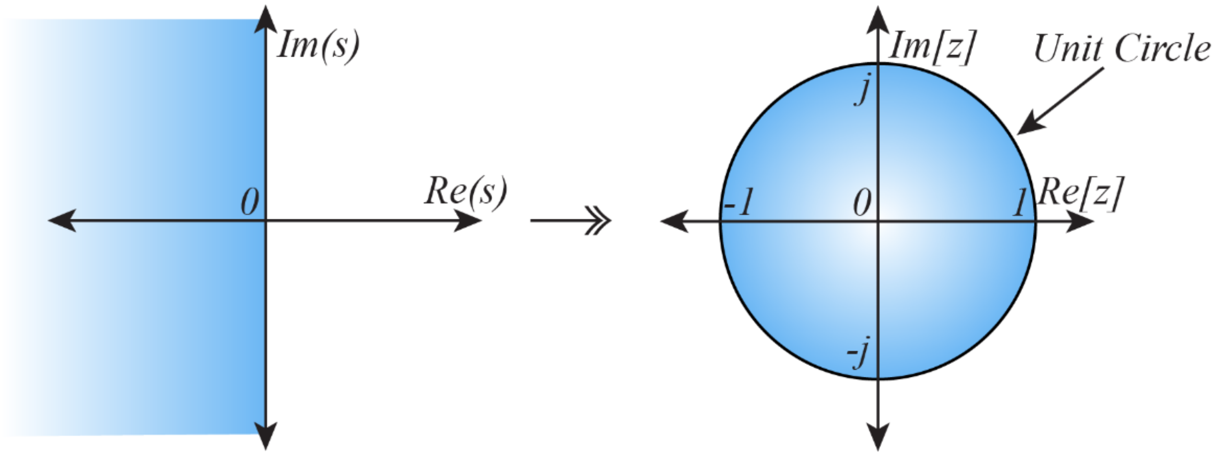



This trapezoidal approximation gives rise to the transform given in (22).

$$s \leftarrow \frac{2}{T_S}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{22}$$

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 16.

**Figure 16.** **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 3p3z controller

The bilinear transform is applied to the s-domain Type-III compensator by replacing every instant of s in the s-domain transfer function with the bilinear mapping. The initial substitution is given in (23) and the simplified result is given in (24).

$$H_C[z] = \left(\frac{\omega_{CP0}}{\frac{2}{T_S}\frac{1-z^{-1}}{1+z^{-1}}}\right)\frac{\left(\frac{\frac{2}{T_S}\frac{1-z^{-1}}{1+z^{-1}}}{\omega_{CZ1}}+1\right)\left(\frac{\frac{2}{T_S}\frac{1-z^{-1}}{1+z^{-1}}}{\omega_{CZ2}}+1\right)}{\left(\frac{\frac{2}{T_S}\frac{1-z^{-1}}{1+z^{-1}}}{\omega_{CP1}}+1\right)\left(\frac{\frac{2}{T_S}\frac{1-z^{-1}}{1+z^{-1}}}{\omega_{CP2}}+1\right)} \quad (23)$$

$$H_C[z] = \frac{B_3 z^{-3} + B_2 z^{-2} + B_1 z^{-1} + B_0}{-A_3 z^{-3} - A_2 z^{-2} - A_1 z^{-1} + 1} \quad (24)$$

In the discrete-time z-domain, the transfer function now has three z-domain poles and three z-domain zeros. Therefore, this controller is now referred to as a three-pole three-zero controller or 3p3z 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 (25) to (31).

$$B_3 = \frac{T_S \times \omega_{CP0} \times \omega_{CP1} \times \omega_{CP2} \times (-2 + T_S \times \omega_{CZ1}) \times (-2 + T_S \times \omega_{CZ2})}{(2 \times (2 + T_S \times \omega_{CP1}) \times (2 + T_S \times \omega_{CP2}) \times \omega_{CZ1} \times \omega_{CZ2})} \quad (25)$$

$$B_2 = \frac{T_S \times \omega_{CP0} \times \omega_{CP1} \times \omega_{CP2} \times \left(-4 + 3T_S{}^2 \times \omega_{CZ1} \times \omega_{CZ2} - 2T_S \times (\omega_{CZ1} + \omega_{CZ2})\right)}{(2 \times (2 + T_S \times \omega_{CP1}) \times (2 + T_S \times \omega_{CP2}) \times \omega_{CZ1} \times \omega_{CZ2})} \quad (26)$$

$$B_1 = \frac{T_S \times \omega_{CP0} \times \omega_{CP1} \times \omega_{CP2} \times \left(-4 + 3T_S{}^2 \times \omega_{CZ1} \times \omega_{CZ2} + 2T_S \times (\omega_{CZ1} + \omega_{CZ2})\right)}{(2 \times (2 + T_S \times \omega_{CP1}) \times (2 + T_S \times \omega_{CP2}) \times \omega_{CZ1} \times \omega_{CZ2})} \quad (27)$$

$$B_0 = \frac{T_S \times \omega_{CP0} \times \omega_{CP1} \times \omega_{CP2} \times (2 + T_S \times \omega_{CZ1}) \times (2 + T_S \times \omega_{CZ2})}{(2 \times (2 + T_S \times \omega_{CP1}) \times (2 + T_S \times \omega_{CP2}) \times \omega_{CZ1} \times \omega_{CZ2})} \quad (28)$$

$$A_3 = \frac{(-2 + Ts \times \omega_{CP1}) \times (-2 + Ts \times \omega_{CP2})}{(2 + Ts \times \omega_{CP1}) \times (2 + Ts \times \omega_{CP2})} \quad (29)$$

$$A_2 = \frac{\left(12 - Ts^2 \times \omega_{CP1} \times \omega_{CP2} - 2 \times Ts \times (\omega_{CP1} + \omega_{CP2})\right)}{(2 + Ts \times \omega_{CP1}) \times (2 + Ts \times \omega_{CP2})} \quad (30)$$

$$A_1 = \frac{\left(-12 + Ts^2 \times \omega_{CP1} \times \omega_{CP2} - 2 \times Ts \times (\omega_{CP1} + \omega_{CP2})\right)}{(2 + Ts \times \omega_{CP1}) \times (2 + Ts \times \omega_{CP2})} \quad (31)$$

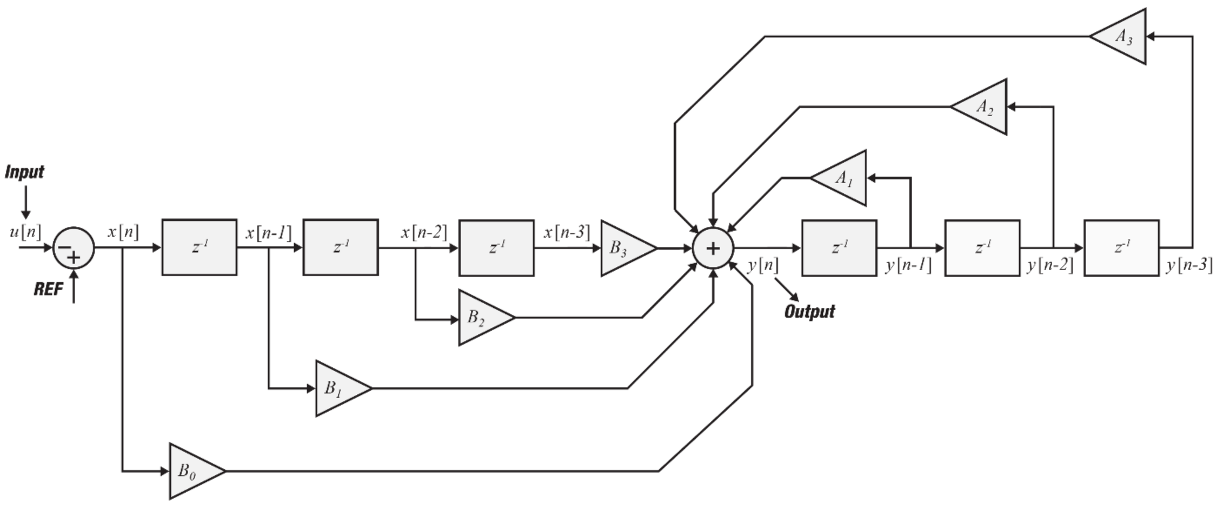
There is no need to calculate these controller coefficients by hand as the software tool Biricha ST-WDS available from *www.biricha.com/st-wds* performs these calculations.

Finally, now that the controller is in the discrete-time 3p3z 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 (32).

$$\begin{aligned} y[n] &= A_1 y[n-1] + A_2 y[n-2] + A_3 y[n-3] \\ &+ B_0 x[n] + B_1 x[n-1] + B_2 x[n-2] + B_3 x[n-3] \end{aligned} \quad (32)$$

Where y[n] is the output for the current sampling interval. The structure of this controller is depicted in Figure 17.

**Figure 17. 3p3z controller structure**



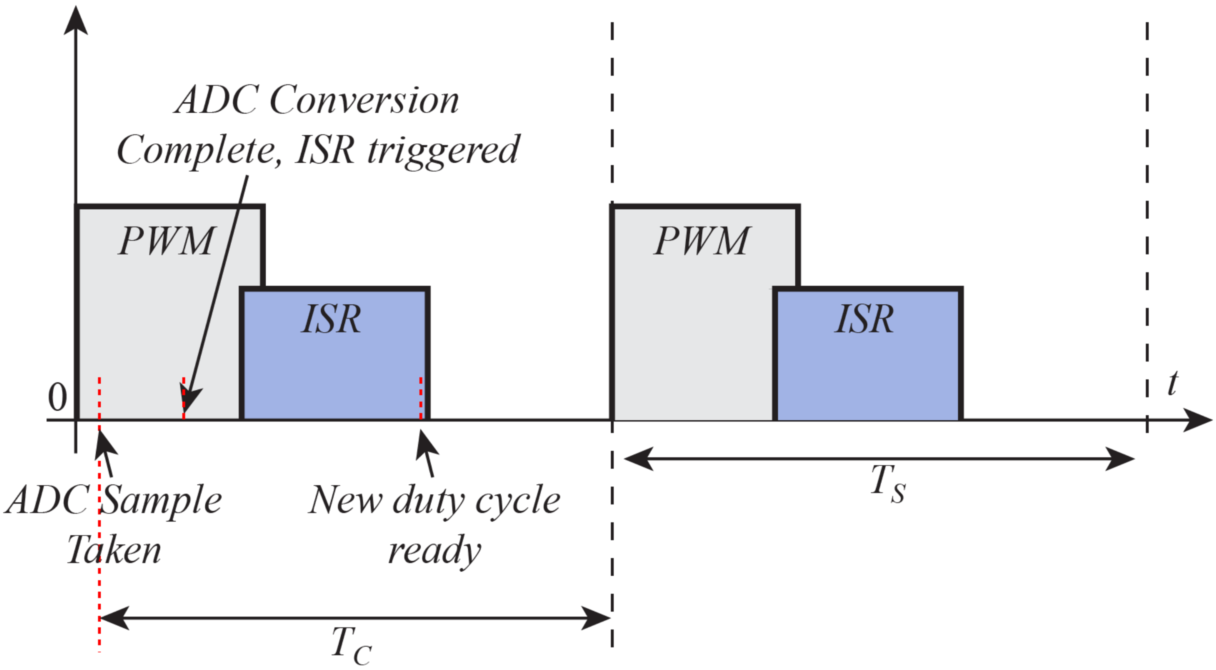

For this voltage mode step-down converter, the current output of the controller may be the new value of the duty cycle that is used in the following switching period. Therefore, there is now an equation that can implement the analytically designed controller and is calculated by the MCU at every sampling 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 of the duty cycle. This sampling and calculation time may take several hundred nanoseconds and the new value of the duty cycle is ready to use towards the end of that switching period. This situation is described in Figure 18.

Figure 18. Sources of delay within the discrete-time system




However, as the new duty cycle 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 the duty cycle, and that duty cycle is held constant for the remainder of the next switching period. Therefore, there is one zero-order hold in this system. The frequency response of a ZOH is given in (33).

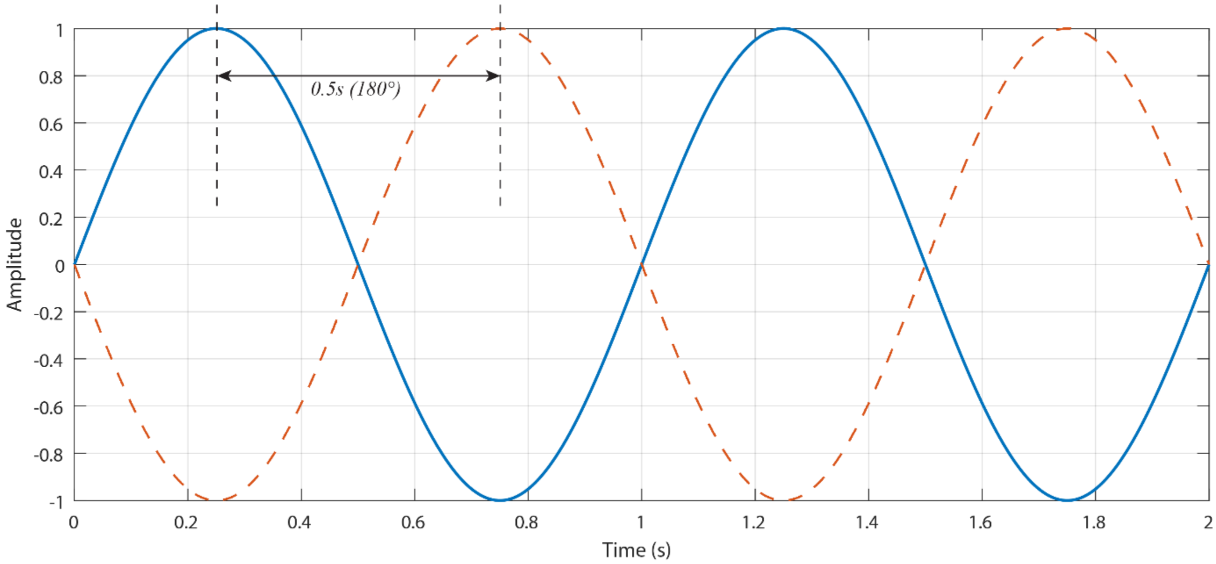
$$F_{ZOH}(j\omega) = \frac{1 - e^{-j\omega T_S}}{j\omega} \tag{33}$$

This can be shown to be a complex number expressed in polar form, and therefore the phase angle is given in (34).

$$\angle F_{ZOH}(j\omega) = -\omega\frac{T_S}{2} \tag{34}$$

To understand how this phase angle can relate to a time delay, consider as an example the 1-Hz sine wave shown in Figure 19. If this sine wave is delayed by 0.5 s, the result is the sine wave shown as a dashed line in Figure 19.

**Figure 19. 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 (35).

$$p.d. = -2\pi f T_D \tag{35}$$

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.

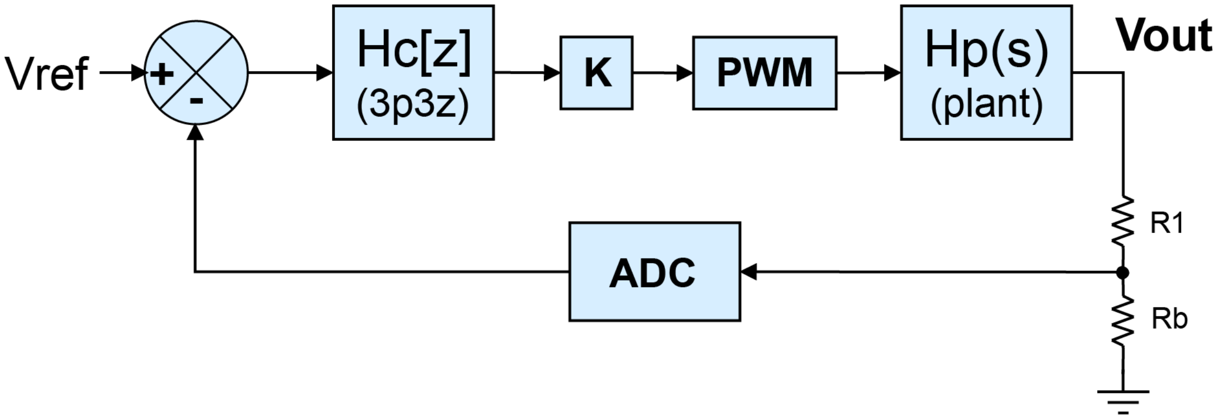
$$p.e. = -2\pi f_X T_D \tag{36}$$

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 (36). 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 3p3z controller in Figure 20 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 20. Additional gains within the digital system**




Consider the block diagram of the digital control loop shown in Figure 20. 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 (37).

$$G_{PD} = \frac{R_b}{R_1 + R_b} \tag{37}$$

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 (38).

$$G_{ADC} = \frac{2^{12} - 1}{3.3\,V} \tag{38}$$

The output of the controller is used to update the duty cycle register and therefore this must be in the correct scaling. The PWM is implemented using the HRTIM module onboard the MCU and the timer-compare register is used to control the effective duty cycle. The HRTIM module outputs a duty cycle of 100% when the timer-compare register is set to the period value. Therefore, the gain of the PWM module can be calculated using (39). This is similar to the RC ramp and analog comparator gain seen earlier in the analog voltage-mode example.

$$G_{PWM} = \frac{100\,\%}{period} \tag{39}$$

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 (40).

$$K = \frac{1}{G_{PD} G_{ADC} G_{PWM}} \tag{40}$$

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 account for the voltage divider on the input of the ADC. Therefore, the reference can be calculated using (41).

$$V_{REF} = V_{OUT} G_{PD} G_{ADC} \tag{41}$$

# 6 Software implementation

## 6.1 Targeted application

The following configuration sets up the STM32 MCU to operate a closed-loop voltage mode step-down converter using the onboard peripherals including the ADC, DMA, and HRTIM. The FMAC is used to implement the 3p3z 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 used for other tasks, or running more power supplies. The example software project to accompany this application note is called `Buck_VoltageMode_HW` as it uses as many hardware peripherals as possible. However, it is possible to use the main core to implement the 3p3z 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_VoltageMode_SW`.

## 6.2 Configuration using STM32CubeMX

The following section contains step-by-step instructions for recreating the STM32CubeMX project for the step-down converter under voltage 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.

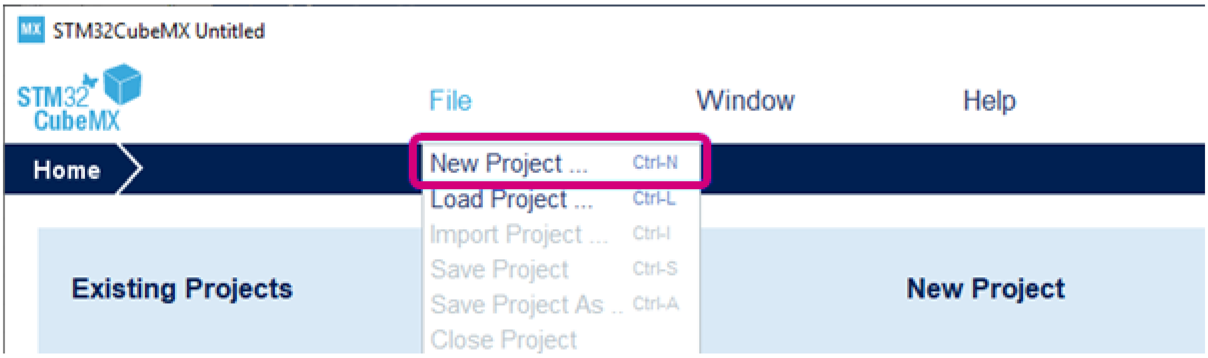
Open STM32CubeMX by clicking on the icon shown in Figure 21 (note that the icon may differ slightly).

**Figure 21. 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.

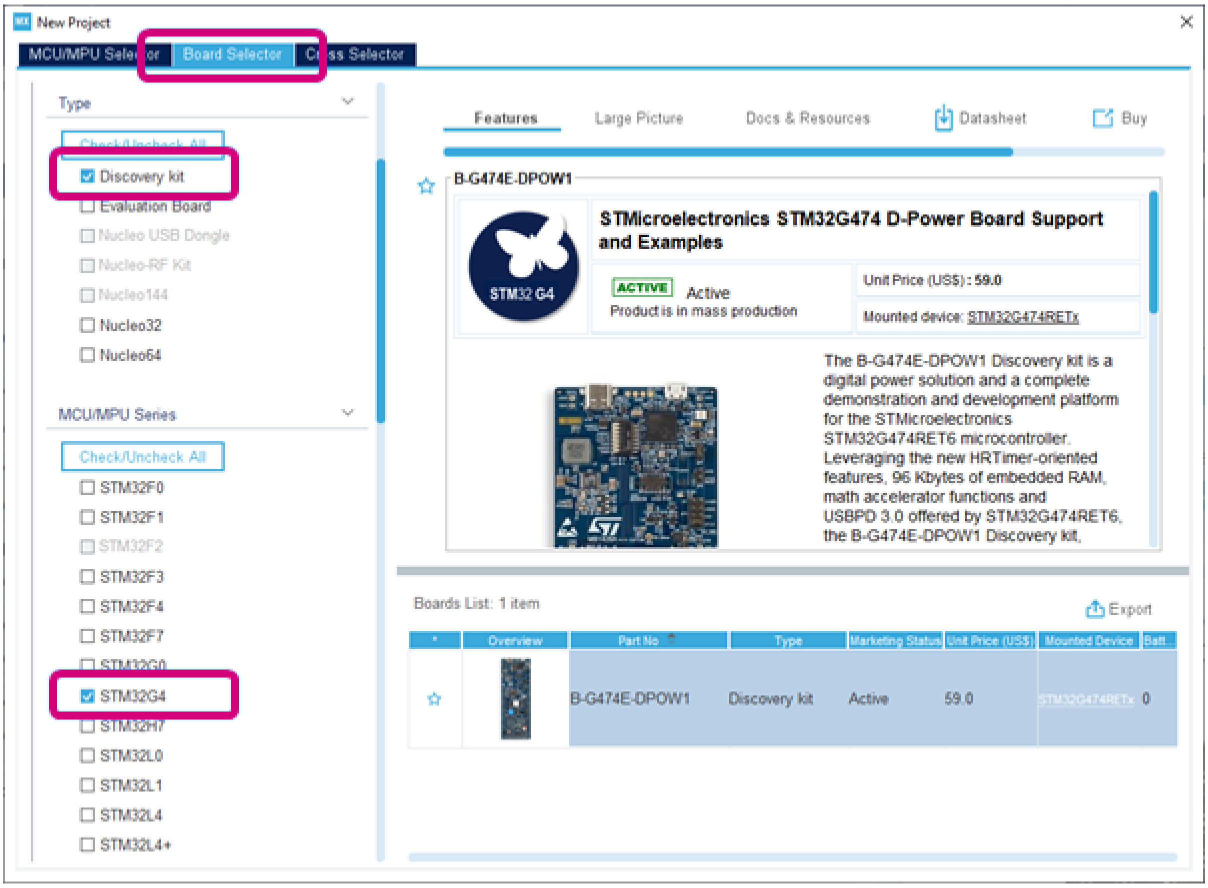
Within the STM32CubeMX window click on `File`, `New Project`.

**Figure 22. 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`.

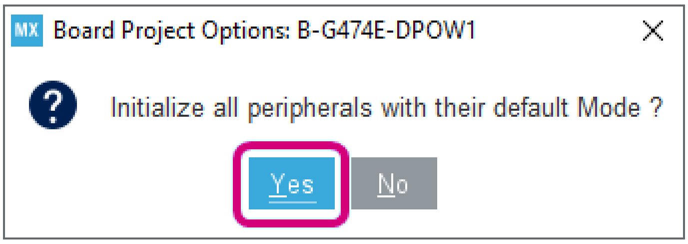
**Figure 23. Board selection**




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 24. Peripherals default mode initialization**



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

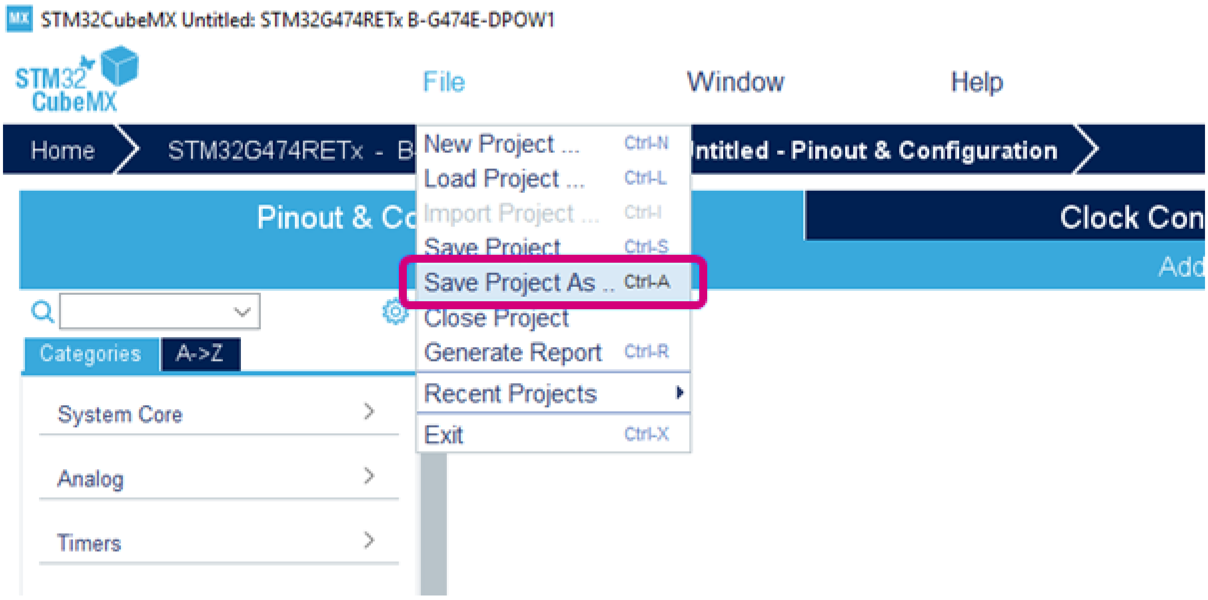



Figure 25. **Project naming**

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.

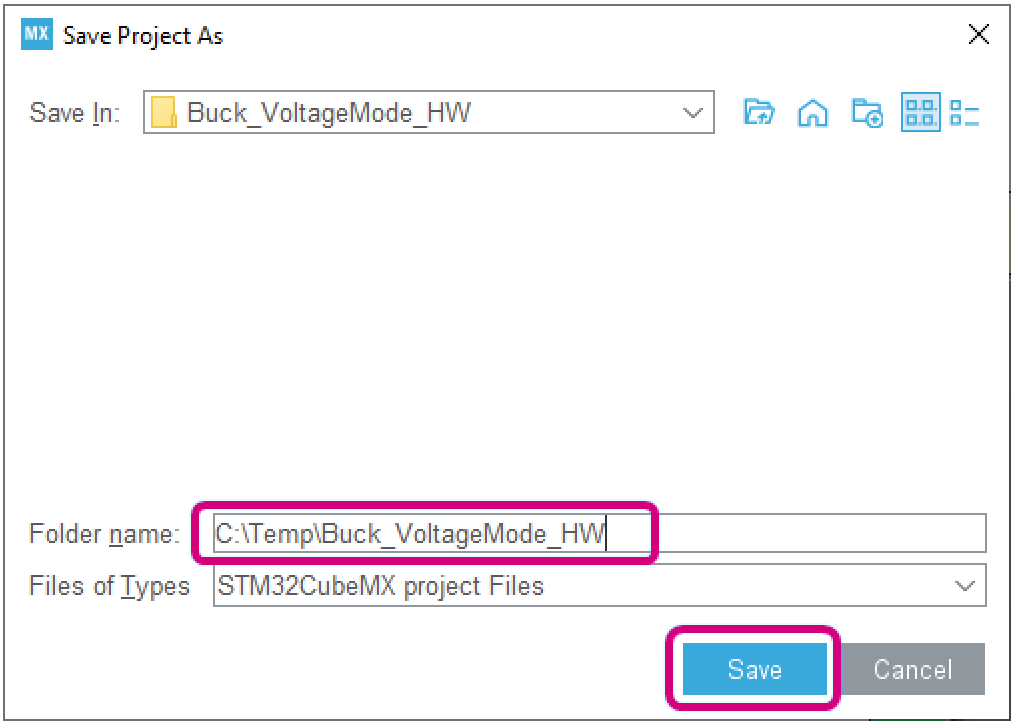
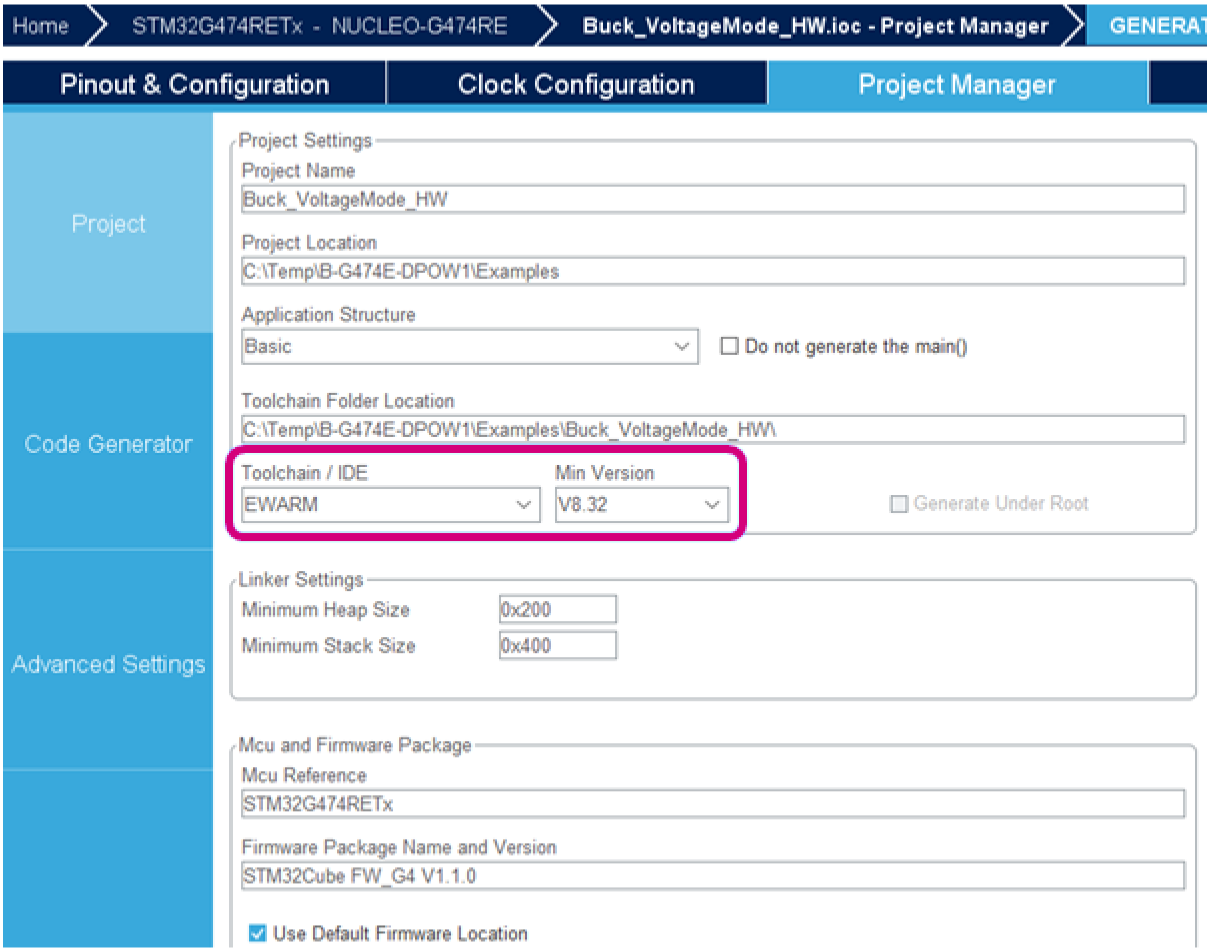



Figure 26. **Project saving**

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 27.

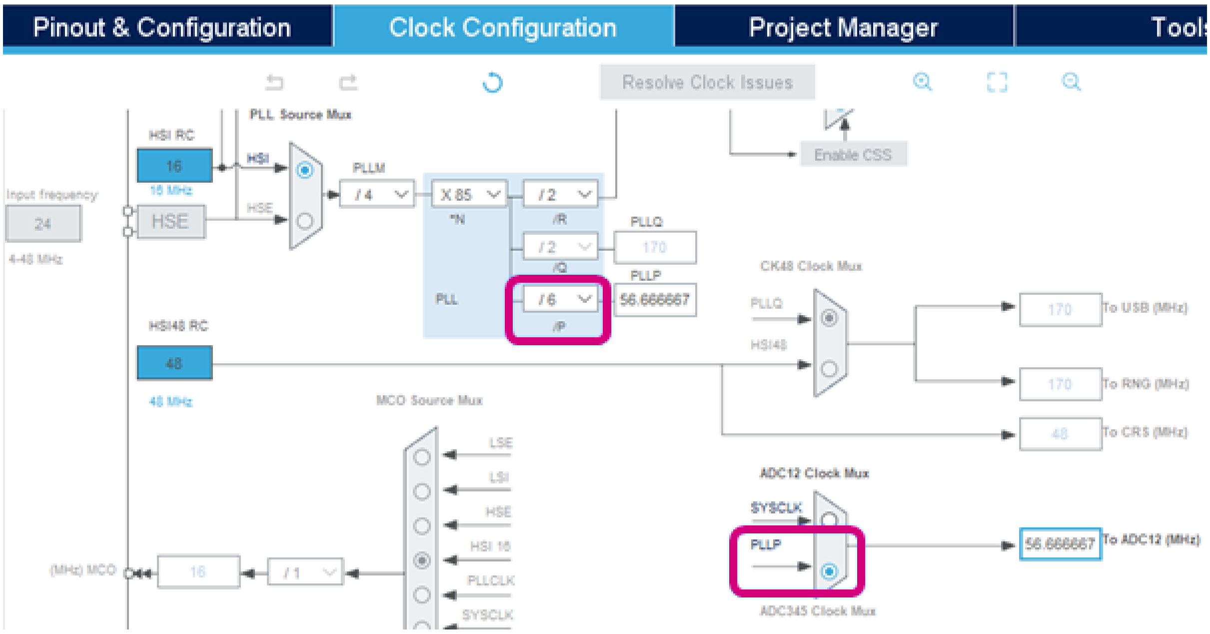
Figure 27. Project manager configuration



### 6.2.1 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 28.

**Figure 28. Clock configuration window**

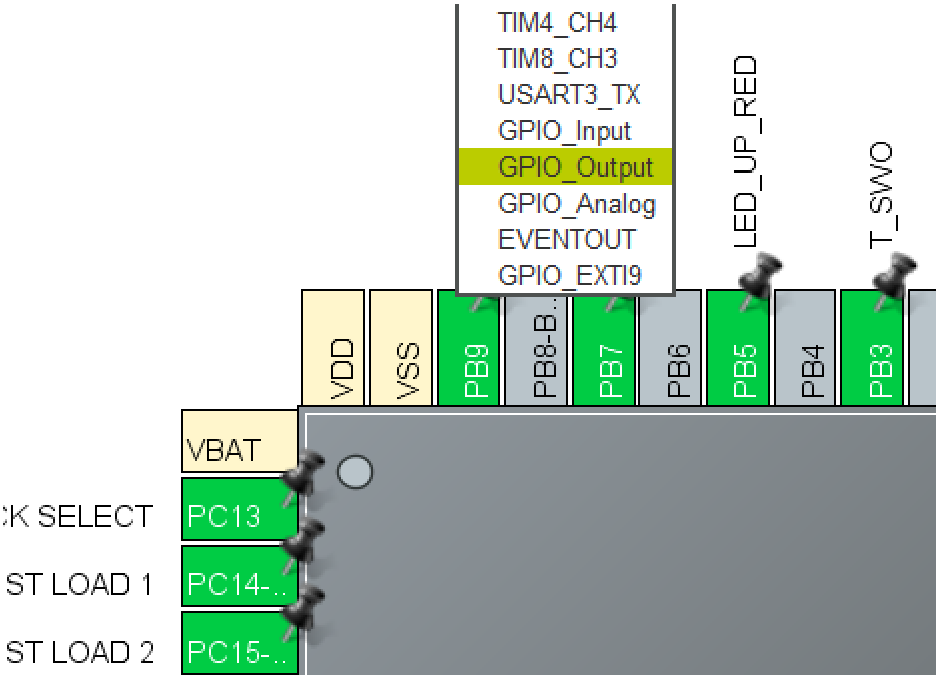



The ADC12 clock may now be 56.66 MHz.

### 6.2.2 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 29.

Figure 29. 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 30.

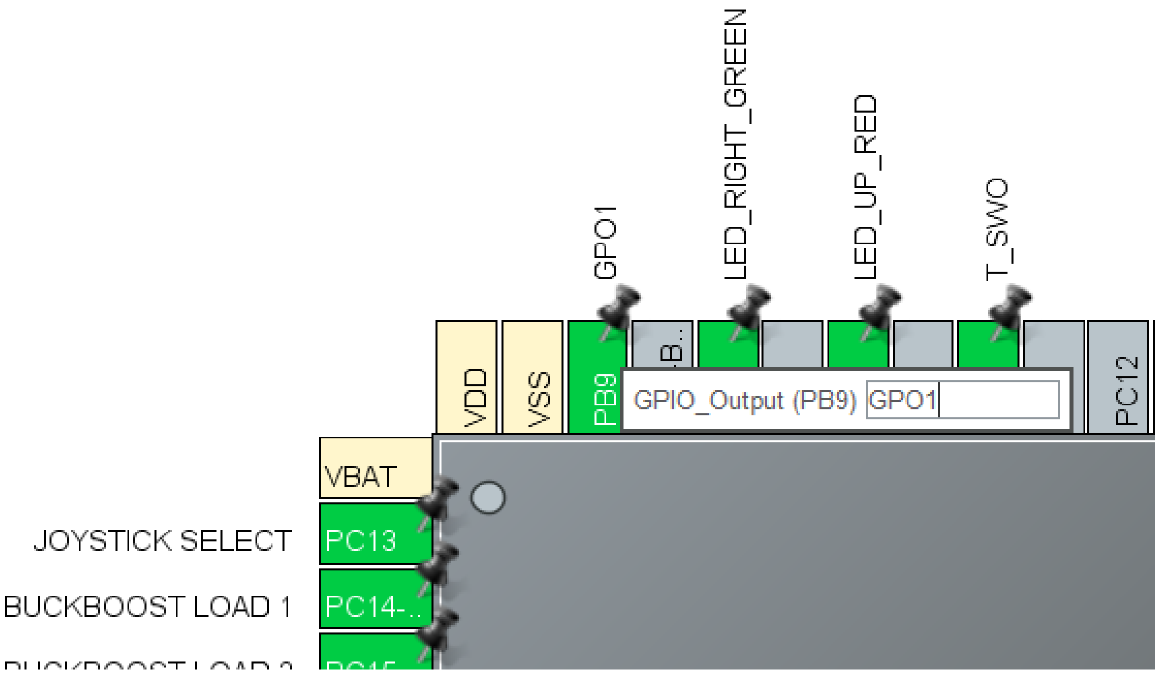
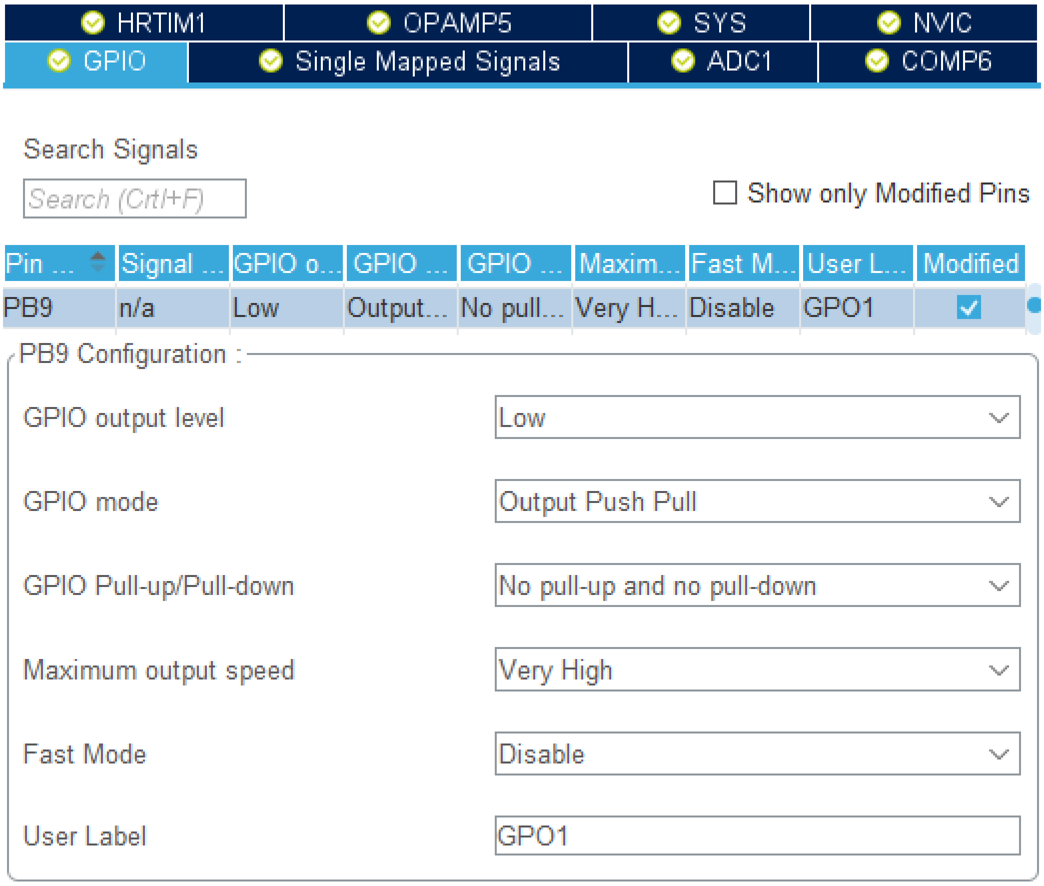



**Figure 30. GPIO renaming**

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 per the image below.

**Figure 31. GPIO maximum output speed setting**




The configuration for GPO1 is now complete.

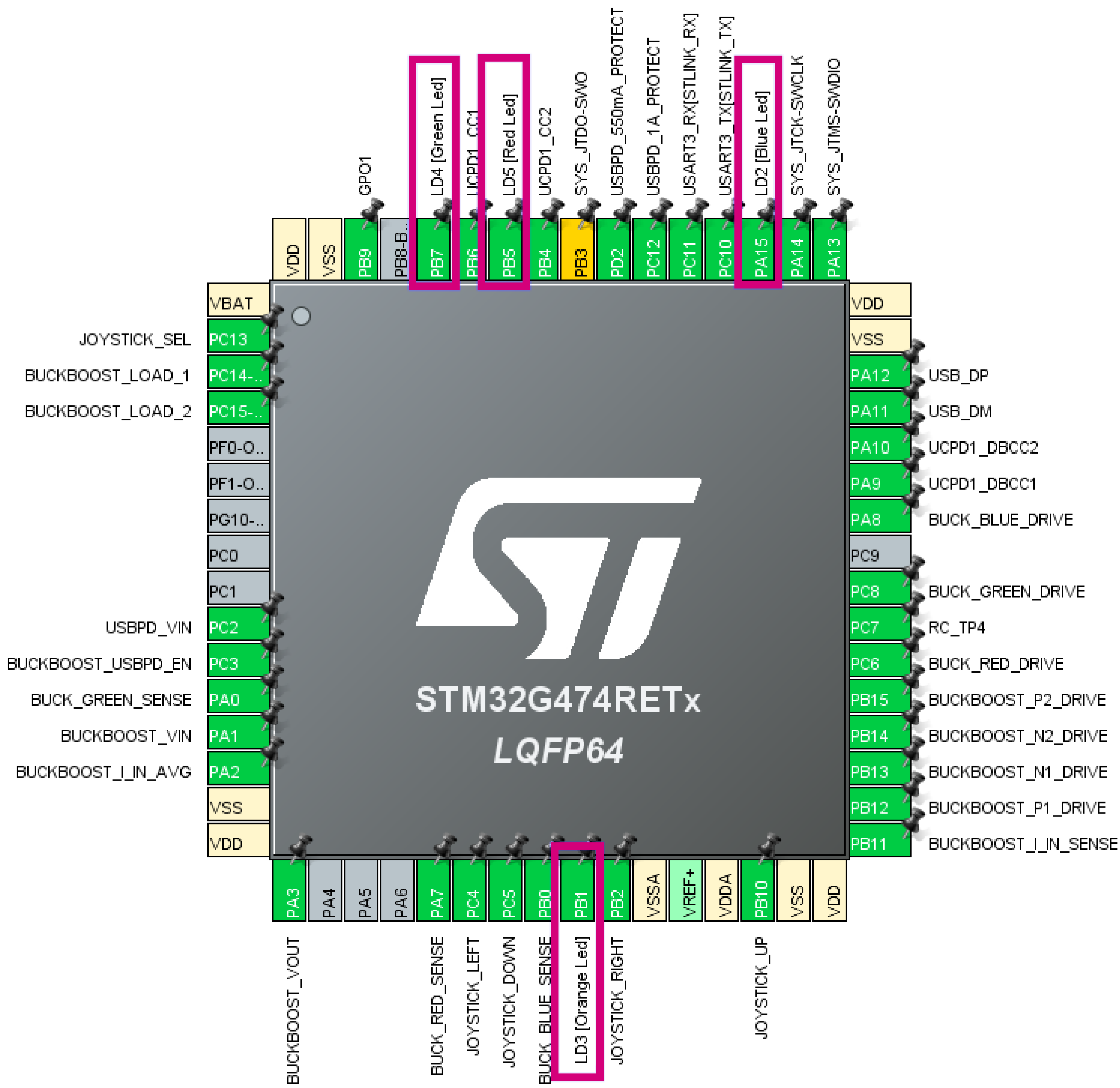
Now, some of the default pin labels must be changed to work with the example code provided. Within the pinout view of the MCU, locate the following pins, right-click on them and select `Enter User Label` and change the label to the new label listed in the table below:

**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 the image below.

**Figure 32. Pins location**

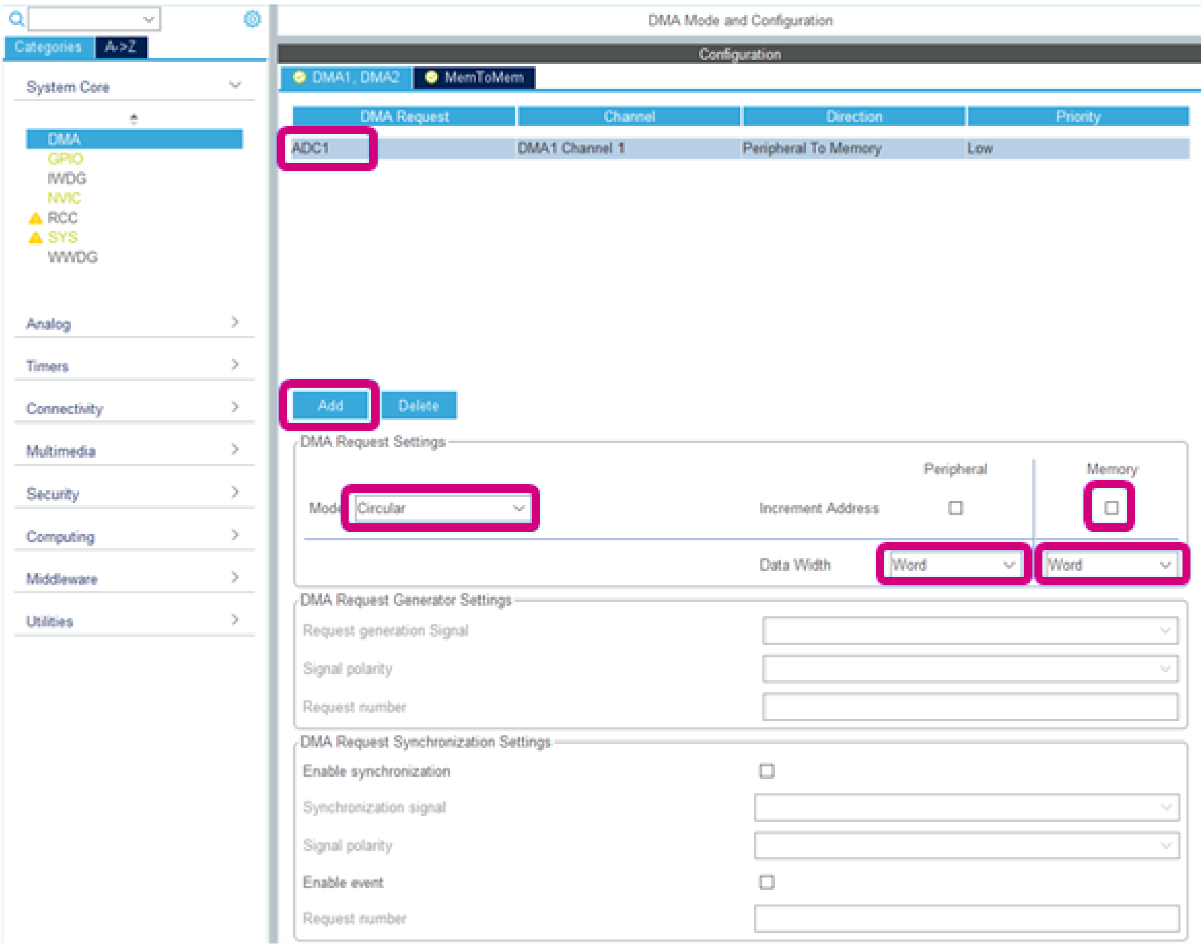



The GPIO configuration is now complete.

### 6.2.3 ADC peripheral configuration

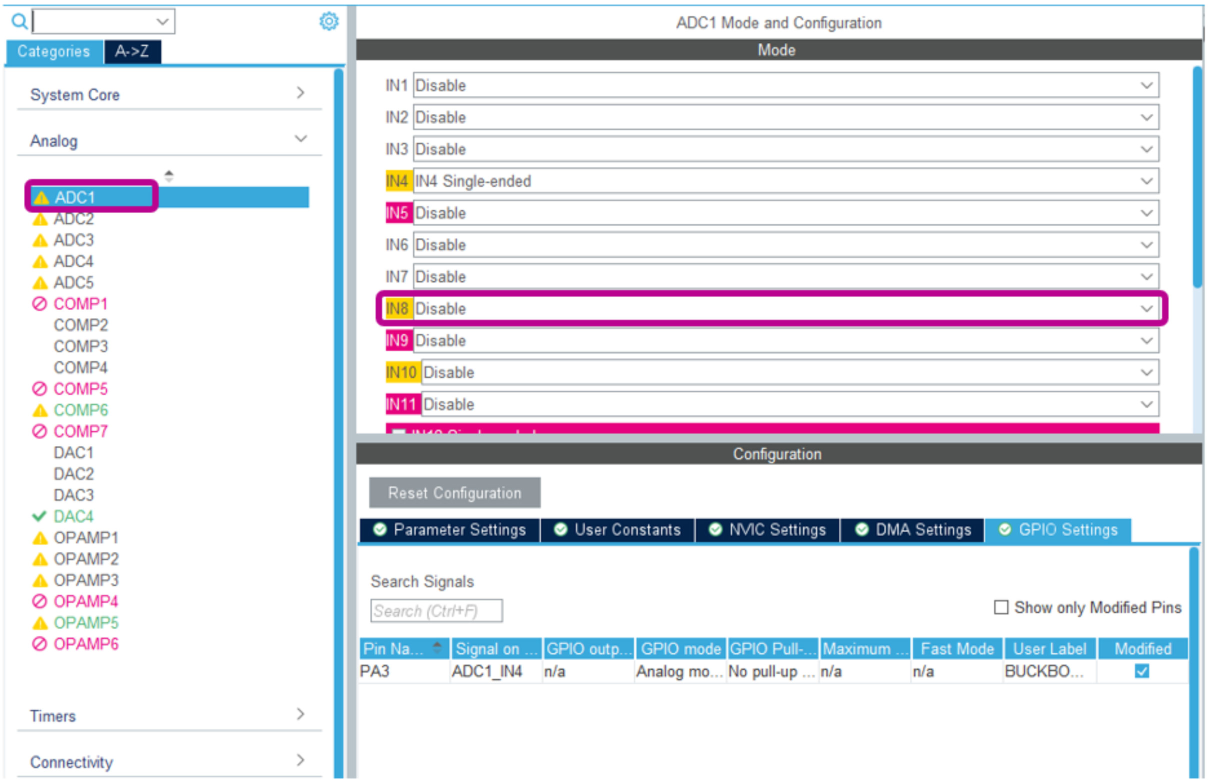
The peripherals can now be configured. On the top menu select `Pinout & Configuration` and then on the left-hand pane, expand the `System Core` list and click on `DMA`. The DMA is used to copy the result from the ADC result register to the FMAC for the execution of the controller. By default, there is 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 33.

**Figure 33. 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`.

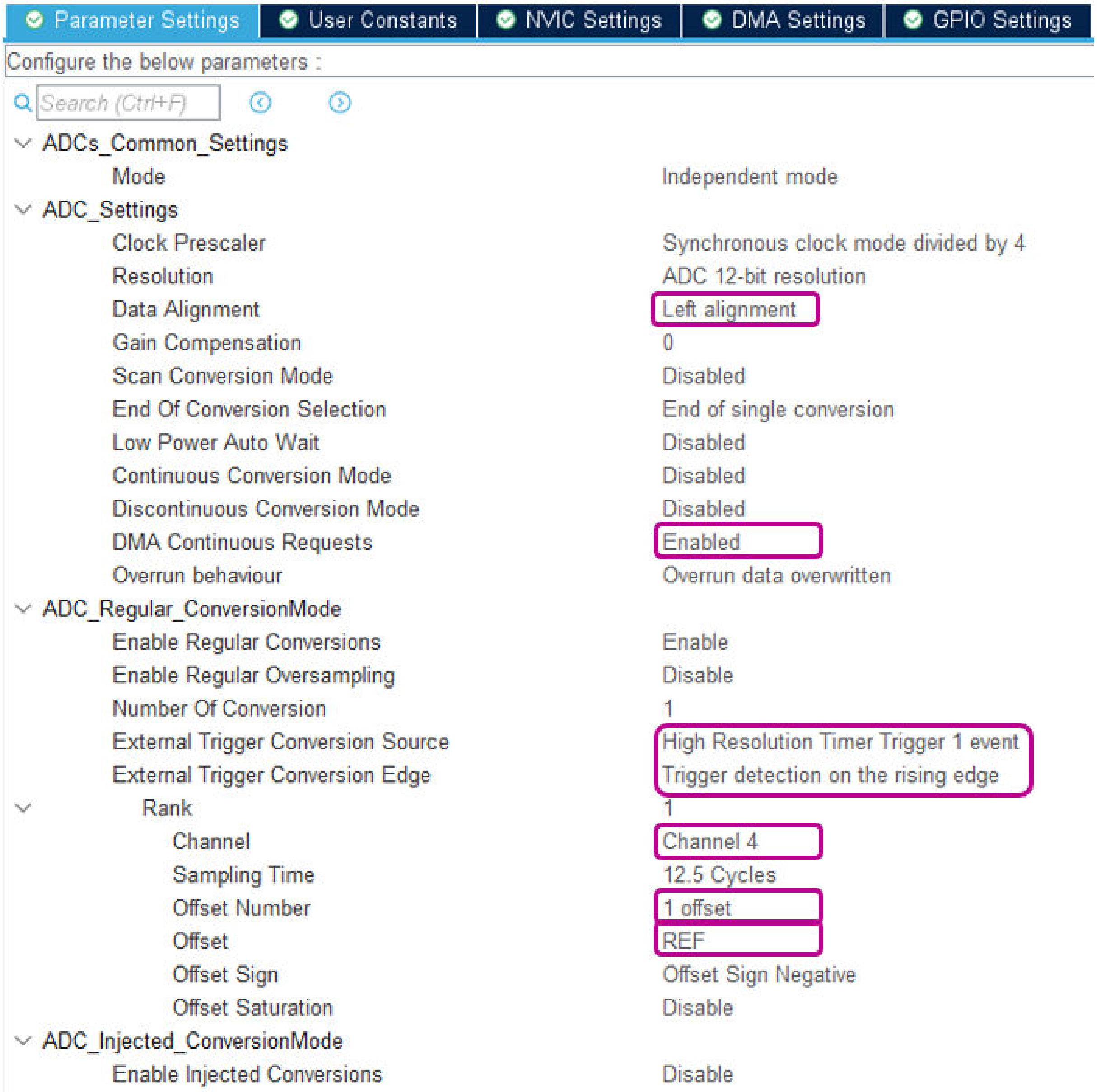
**Figure 34. Removing pin from the project**




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 step-down 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 voltage mode step-down 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 35. 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.

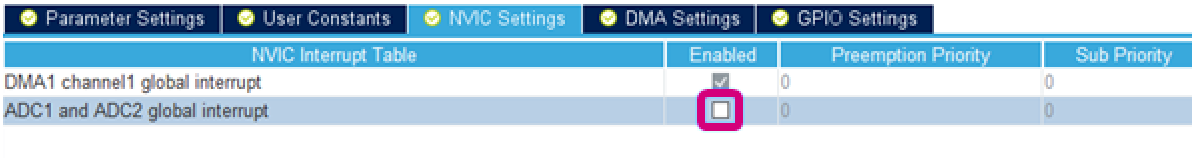



**Figure 36. NVIC settings tab**

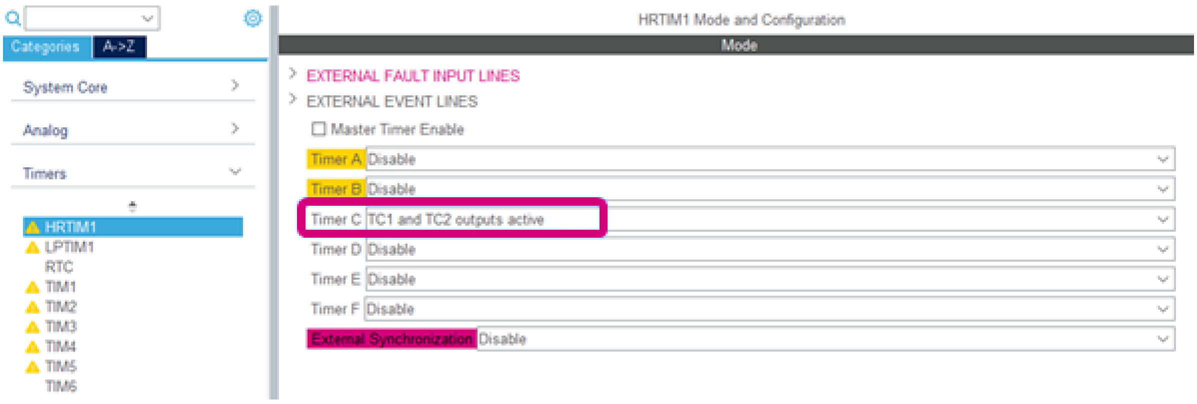
The ADC configuration is now complete.

### 6.2.4 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 voltage mode step-down converter, only `Timer C` is required. Disable all of the other timers as shown in the screenshot below.

**Figure 37. All timers except C disabled**




The power stage of the discovery kit contains boost switches that are following the buck stage and are used when operating as a noninverting 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`.

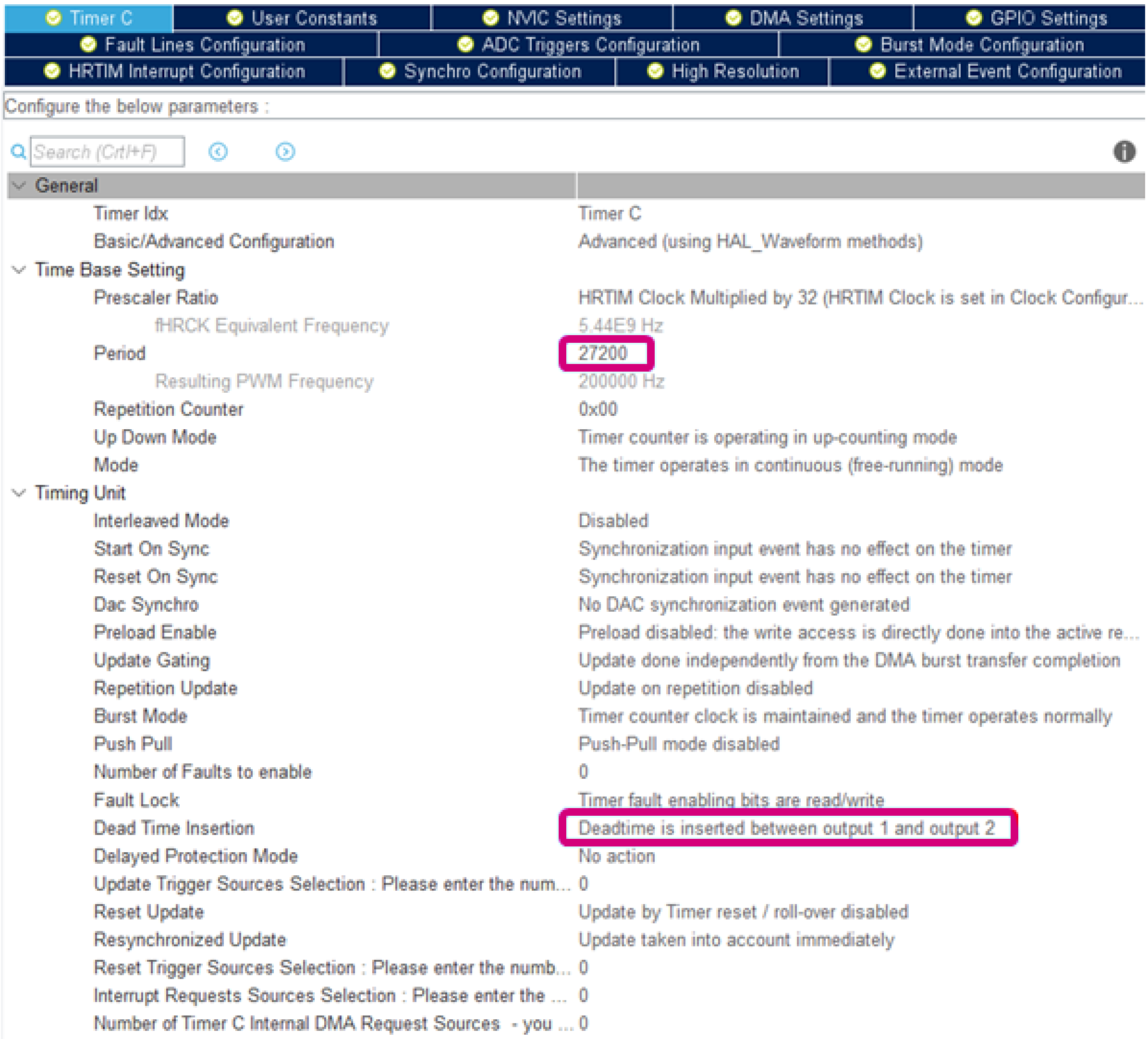
Right-click on PB14 and select `Enter User Label`. Enter the label: `BUCKBOOST_N2_DRIVE`.

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 `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 pressing enter. 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` section settings. 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 38. Timer C configuration tab - part 1**




Further down the `Timer C` configuration tab is the section 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.

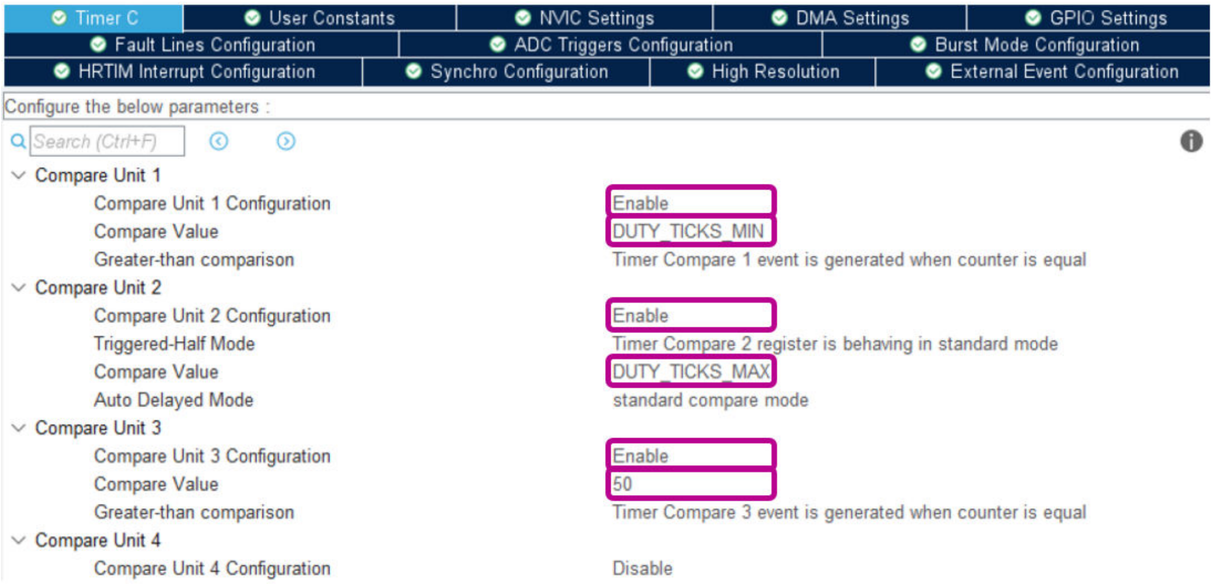
For `Compare Unit 1` section change the configuration setting to `Enable`. Set the `Compare Value` to DUTY_TICKS_MIN. This is the initial value of the duty cycle fixed at 0.35%. The compare event is used later on as a reset source to clear the output.

For `Compare Unit 2` section change the configuration setting to `Enable`. Set the `Compare Value` to DUTY_TICKS_MAX. This is the maximum value of the duty cycle fixed at 90%. The compare event is used later on as a reset source to clear the output. DUTY_TICKS_MIN and DUTY_TICKS_MAX terms that are defined on the code later on.

.

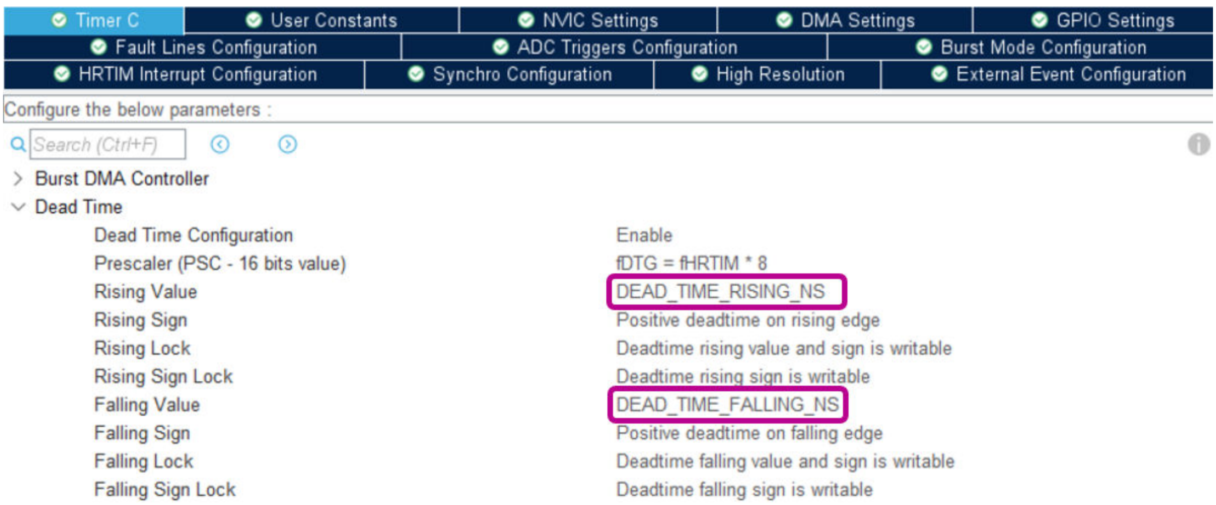
For `Compare Unit 3` section change the configuration setting to `Enable`. Enter a `Compare Value` of `50`. This is the event that is used to trigger the ADC sample. This can be moved to a clean point in the switching cycle, which means away from the turn-on or turn-off of the switch.

**Figure 39. Timer C configuration tab-part 2**




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 `DEAD_TIME_RISING_NS` and `Falling Value` to `DEAD_TIME_FALLING_NS`. `DEAD_TIME_RISING_NS` and `DEAD_TIME_FALLING_NS` defined on code to have `75` ticks `Rising value` and `200` ticks `Falling value`.

**Figure 40. Timer C configuration tab - part 3**



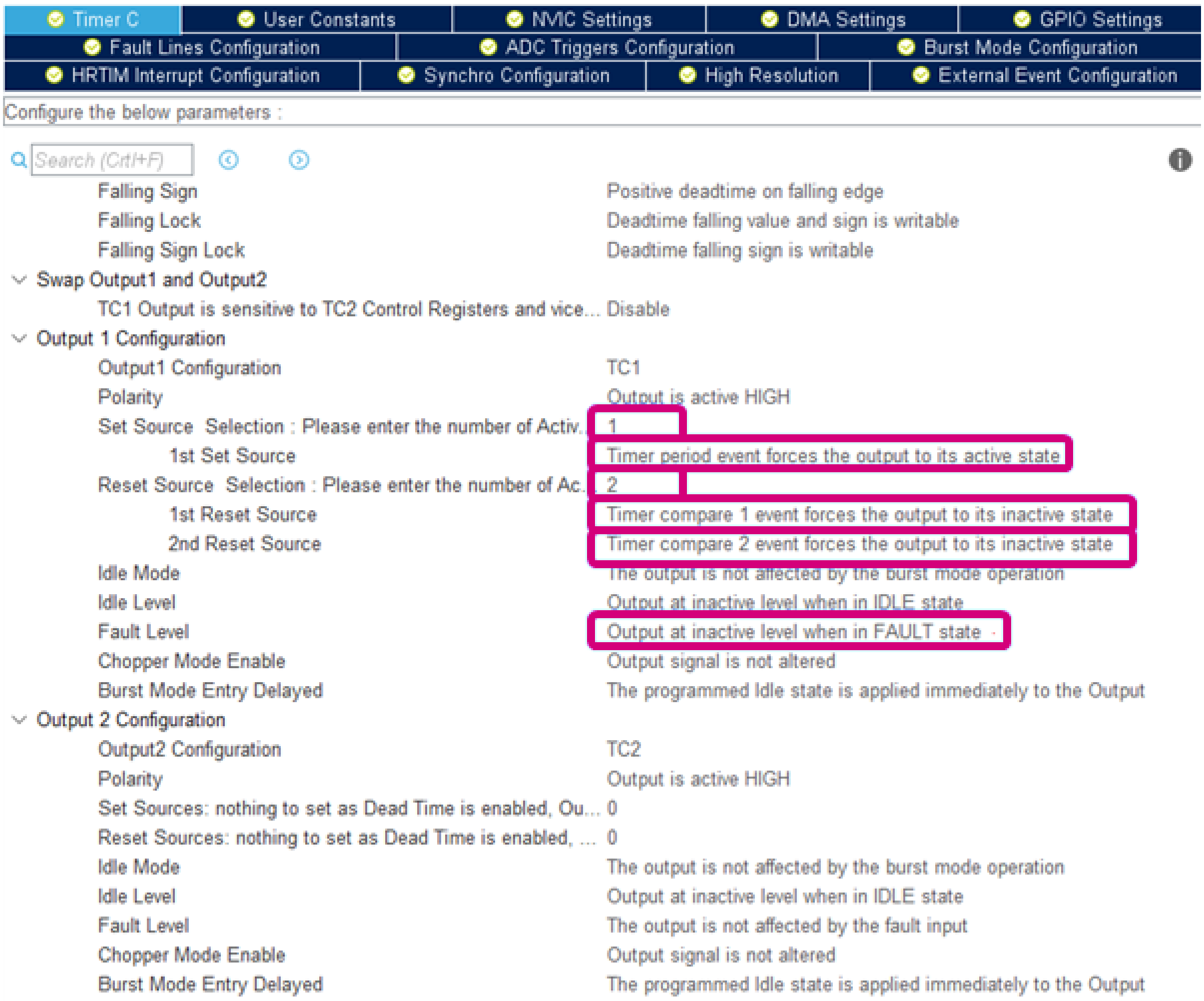

The last two sections to configure on the `Timer C` configuration tab are the `Output 1 Configuration` and `Output 2 Configuration` sections. These are located toward the bottom of the tab.

Under the `Output 1 Configuration` section, change the `Set Source Selection` number to `1`, then the `1st Set Source` event to the `Timer period 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 `Timer compare 2 event forces the output to its inactive state`.

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

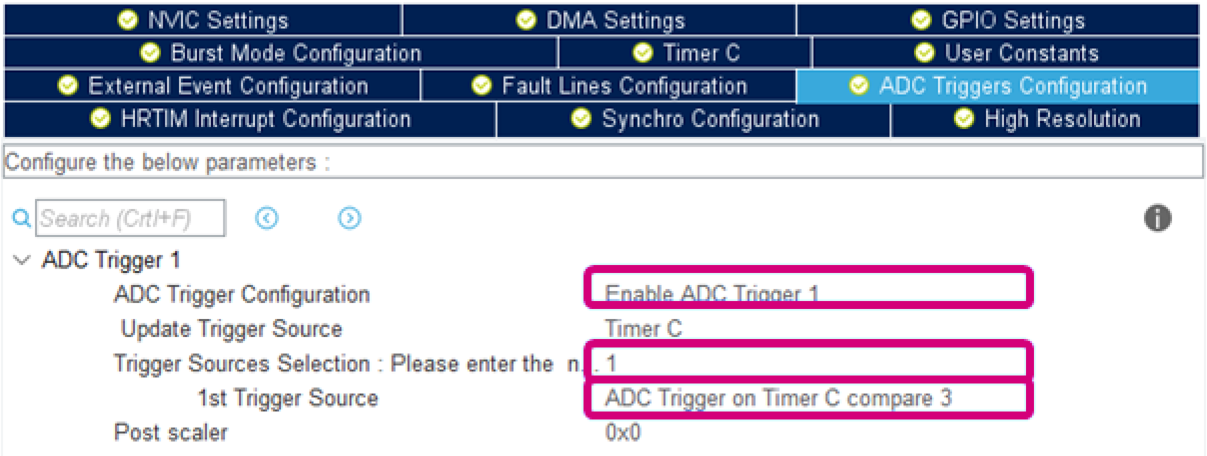
**Figure 41. Timer C configuration tab - part 4**




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

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.

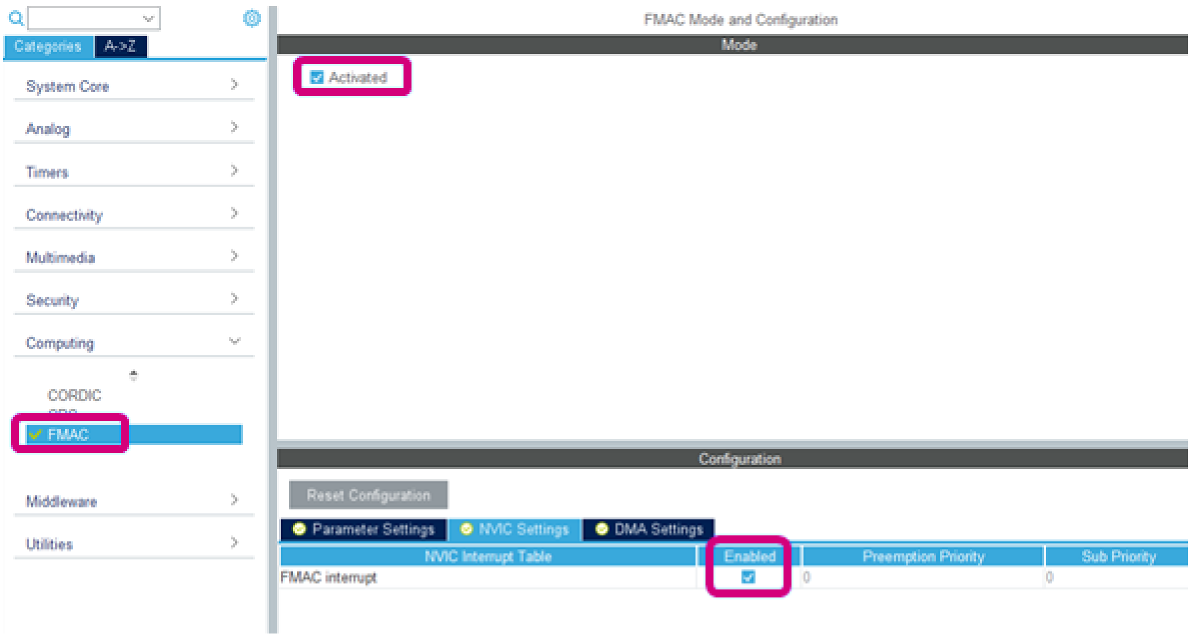
**Figure 42. ADC triggers configuration**




### 6.2.5 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 duty cycle for this voltage mode step-down 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 43. FMAC configuration window**

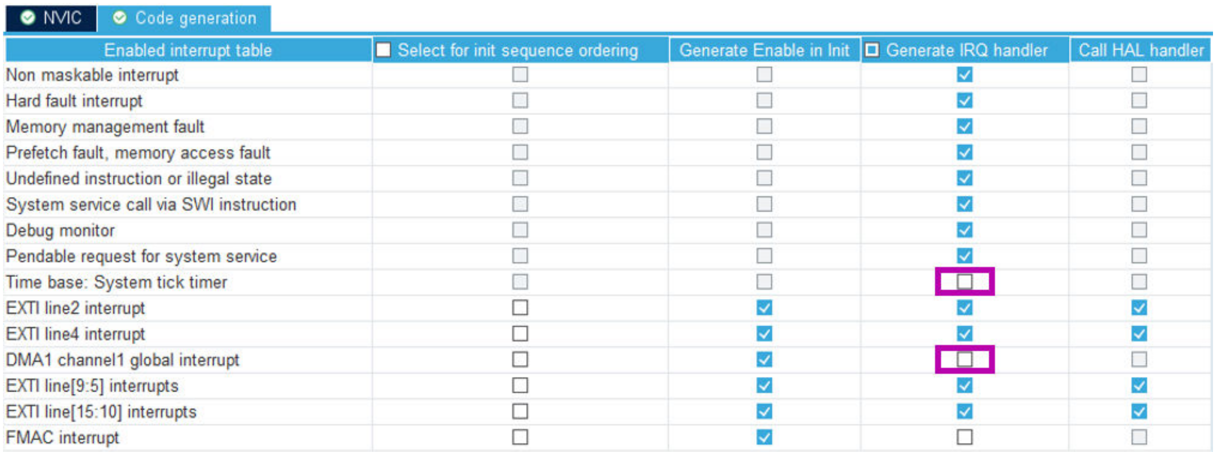


### 6.2.6 IRQ handler configuration

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 their 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` subcategory 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 per the image below.

**Figure 44. Automatic IRQ handler generation disabled**

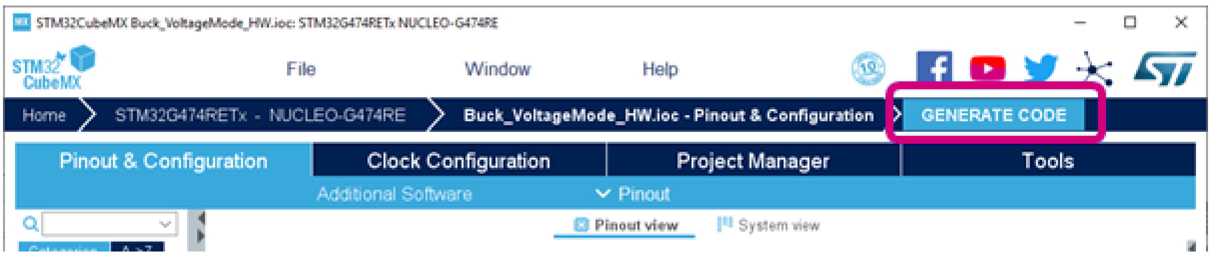

NVIC | Code generation

| Enabled interrupt table | Select for init sequence ordering | Generate Enable in Init | Generate IRQ handler | Call HAL handler |
|---|---|---|---|---|
| Non maskable interrupt | ☐ | ☐ | ☑ | ☐ |
| Hard fault interrupt | ☐ | ☐ | ☑ | ☐ |
| Memory management fault | ☐ | ☐ | ☑ | ☐ |
| Prefetch fault, memory access fault | ☐ | ☐ | ☑ | ☐ |
| Undefined instruction or illegal state | ☐ | ☐ | ☑ | ☐ |
| System service call via SWI instruction | ☐ | ☐ | ☑ | ☐ |
| Debug monitor | ☐ | ☐ | ☑ | ☐ |
| Pendable request for system service | ☐ | ☐ | ☑ | ☐ |
| Time base: System tick timer | ☐ | ☐ | ☐ | ☐ |
| EXTI line2 interrupt | ☐ | ☑ | ☑ | ☑ |
| EXTI line4 interrupt | ☐ | ☑ | ☑ | ☑ |
| DMA1 channel1 global interrupt | ☐ | ☑ | ☐ | ☐ |
| EXTI line[9:5] interrupts | ☐ | ☑ | ☑ | ☑ |
| EXTI line[15:10] interrupts | ☐ | ☑ | ☑ | ☑ |
| FMAC interrupt | ☐ | ☑ | ☐ | ☐ |

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

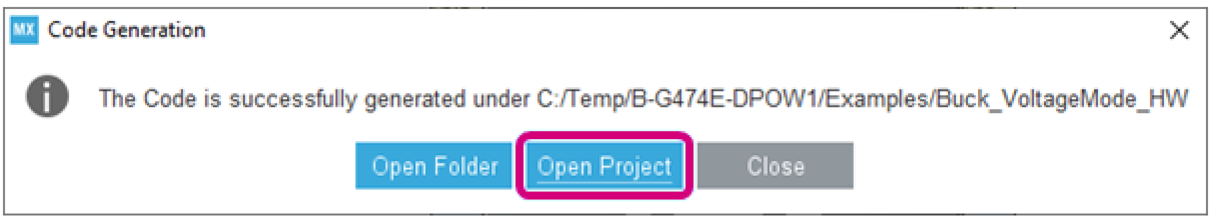
Finally, generate the project by clicking the `GENERATE CODE` button. This creates the necessary project files for the selected IDE. In this example, the IAR Embedded Workbench® from IAR Systems is used. However, there are multiple IDEs for which STM32CubeMX can generate project files, including the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

**Figure 45. 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 46. Project opening after code generation**

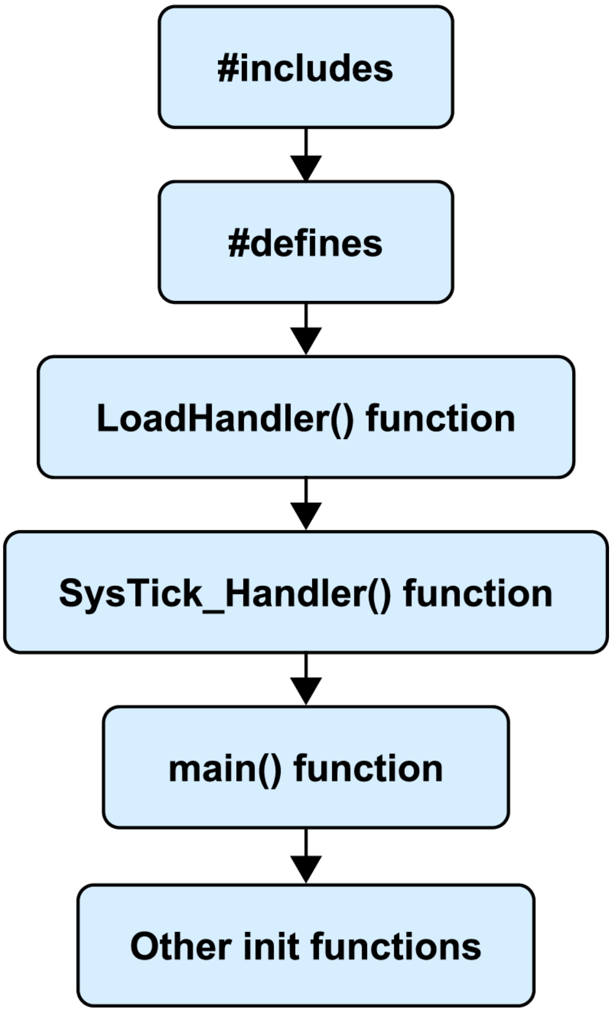


The `main.c` file contains the code for this project. Open the main.c file and read the comments.

## 6.3 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 47.

Figure 47. **Program flow within main.c**



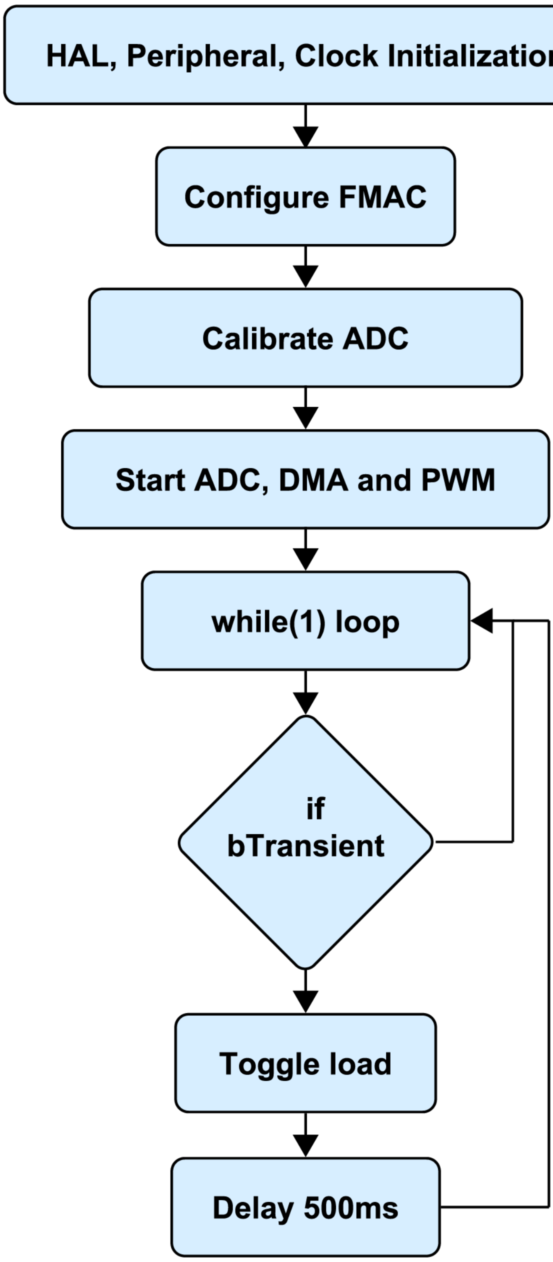
Within these code files, there are comments similar to `USER CODE BEGINS HERE` and `USER CODE ENDS HERE`. Any additional code to the project is kept between these two comments. This is because, outside of these comments, the code is automatically generated by the STM32CubeMX tool. If any changes are 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 48 shows examples of the user code locations highlighted in fuchsia.

**Figure 48. Example of user code locations within main.c**

```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;
```

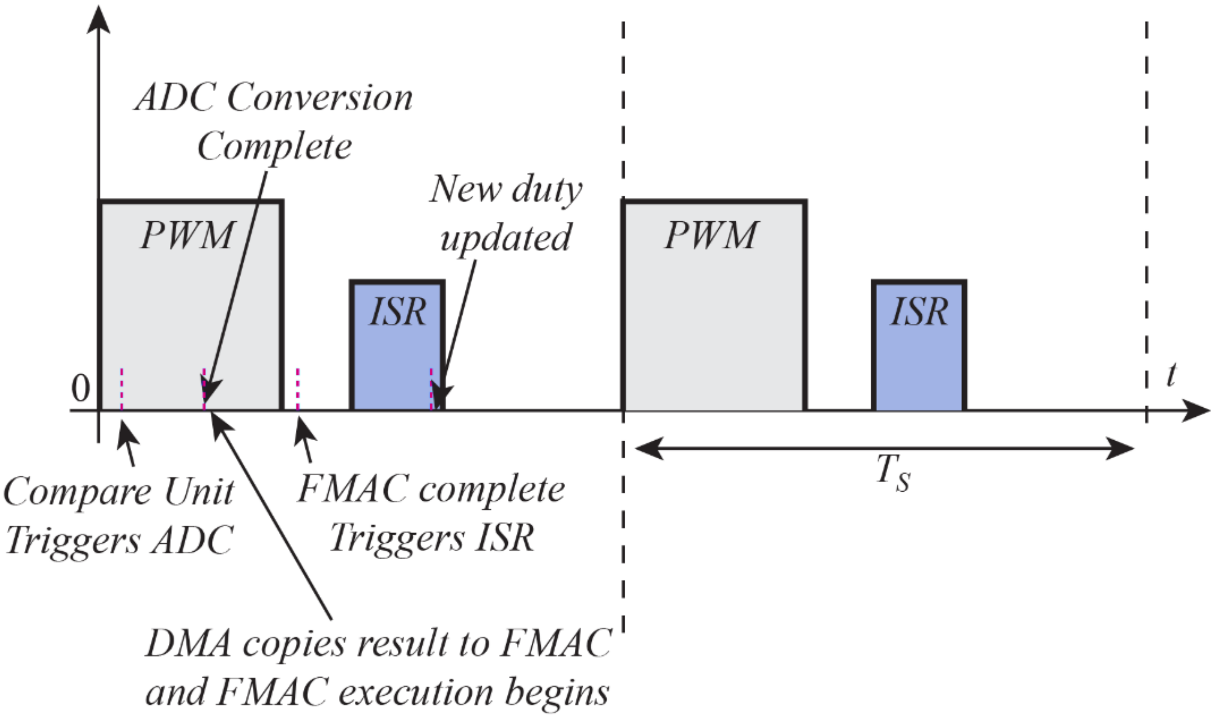
The structure of the main function within the `main.c` file is represented in Figure 49. This is the function that the MCU jumps to after power on – the entry point within the code. Therefore, this function contains calls to all of the initialization functions to set up the peripherals onboard the MCU. 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 49. 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 a certain number of 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 50.

**Figure 50. ADC, DMA, and FMAC timing diagram**

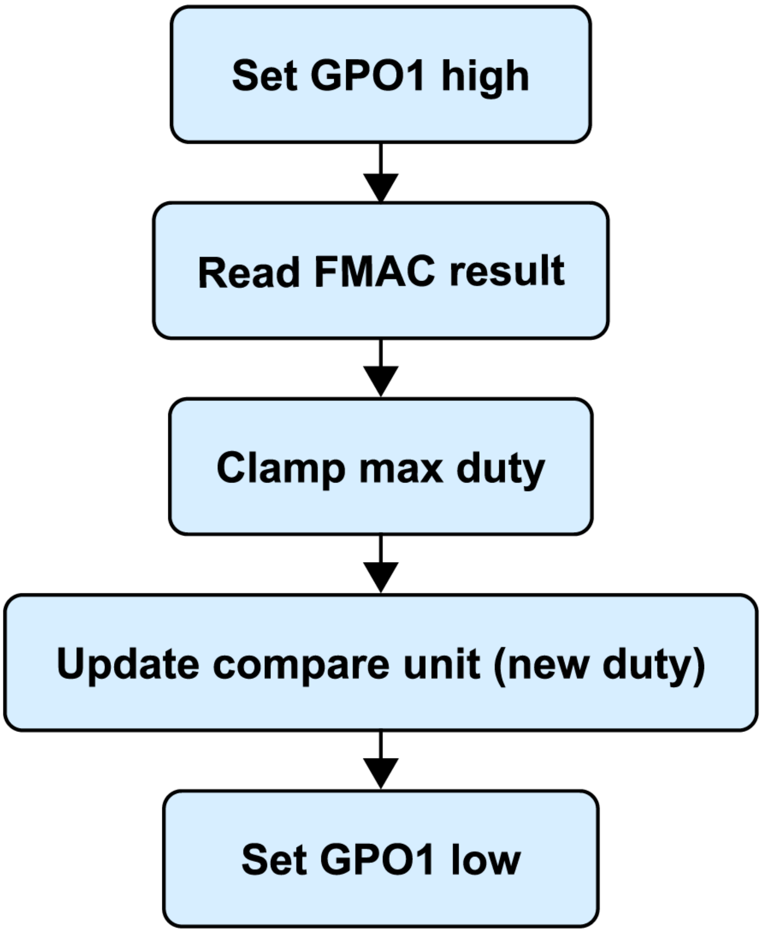


## 6.4 Interrupt service routine

After the FMAC completes the execution of the 3p3z 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 3p3z controller executed using the FMAC. The output of the controller is the new value of the duty cycle and is written to the compare unit 1 register of the HRTIM. However, the compare unit has a maximum value equal to that of the period register. Therefore, the logic depicted in Figure 51 is implemented within the user code section of the FMAC ISR.

**Figure 51. FMAC ISR flow**




## 6.5 3p3z controller coefficients

Depending on the example source, respectively either the G4 intropack example or either the D-Power pack generated example, the 3p3z controller coefficients are defined in the `main.h` (`resp. app_dpower.h`) header file. This can be easily accessed by locating `#include` "`main.h`" (`resp. #include`"`app_dpower.h`") towards the top of the `main.c` file (`resp. app_dpower.h`), then right-clicking on `main.h` (`resp. app_dpower.h`) and selecting `Open main.h` (`resp. app_dpower.h`) from the menu.

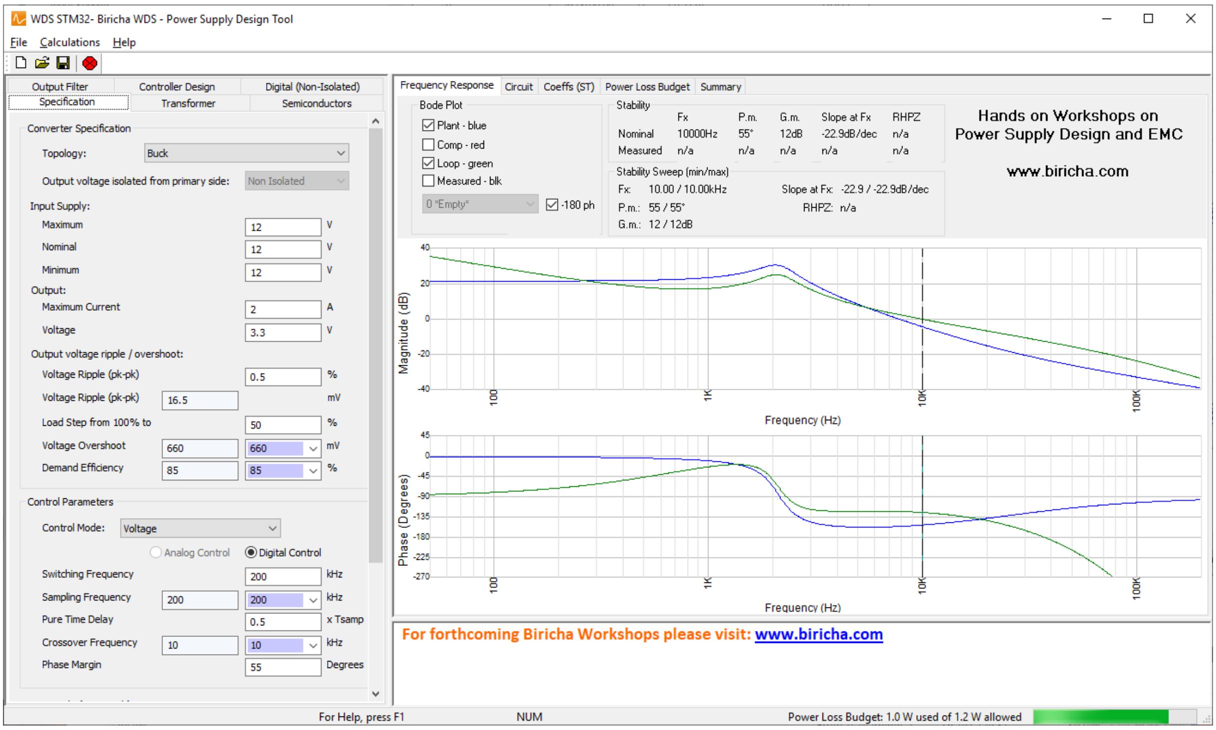
Further down this file, there is a `/* USER CODE BEGIN Private defines */` section where the definitions for the FMAC configuration begin. The controller coefficients are defined below the FMAC configuration parameters. These coefficients (B0, B1, B2, B3, A1, A2, A3) are given in a fixed-point hexadecimal 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 step required is to convert these discrete-time controller coefficients into fixed point form for use on the fixed point FMAC.

## 6.6 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.

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 recreating the WDS project settings is included here for completeness.

**Figure 52. Biricha ST-WDS**




The initial screen of ST-WDS is shown in Figure 52. 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.

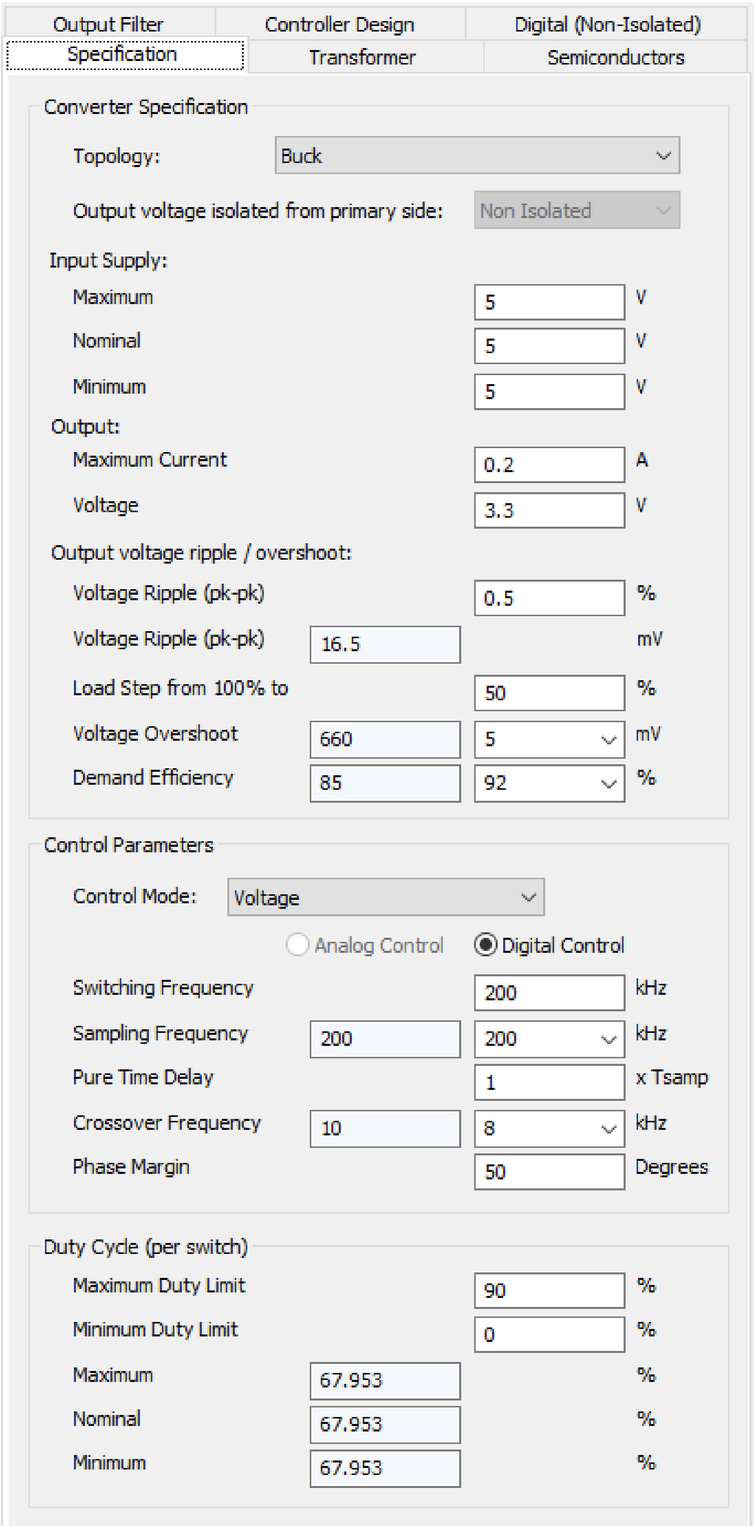
To calculate the digital controller coefficients the power supply specification must first be entered. On the `Specification` tab, enter the specification shown in Table 2.

**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 | Voltage, digital control |
| Switching frequency | 200 kHz |
| Pure time delay | 1 |
| Crossover frequency | 8 kHz |
| Phase margin | 50° |
| Maximum duty limit | 90% |
| Minimum duty limit | 0% |

The `Specification` tab may now look like the screenshot shown in Figure 53.

**Figure 53. ST-WDS specification tab**



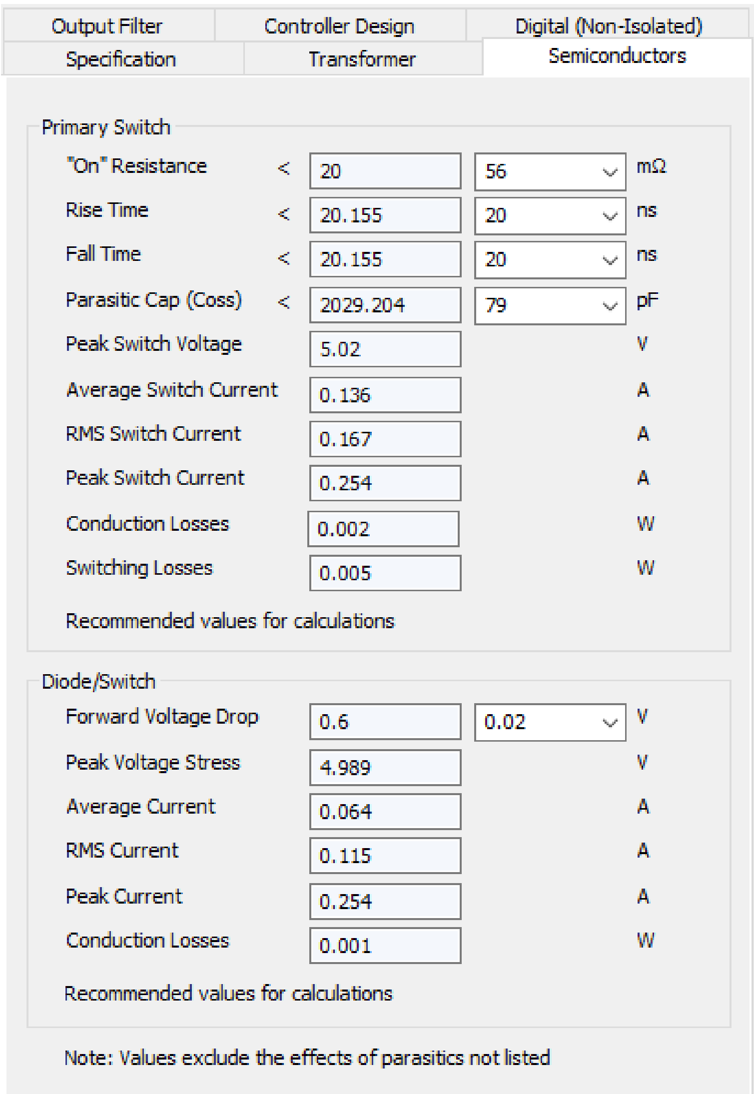
The `Transformer` tab is not used as this is a step-down converter. This tab is only applicable for topologies that include a power stage transformer. Next, click on the `Semiconductors` tab and enter the specification shown in Table 3.

**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 54.

**Figure 54. ST-WDS semiconductor tab**



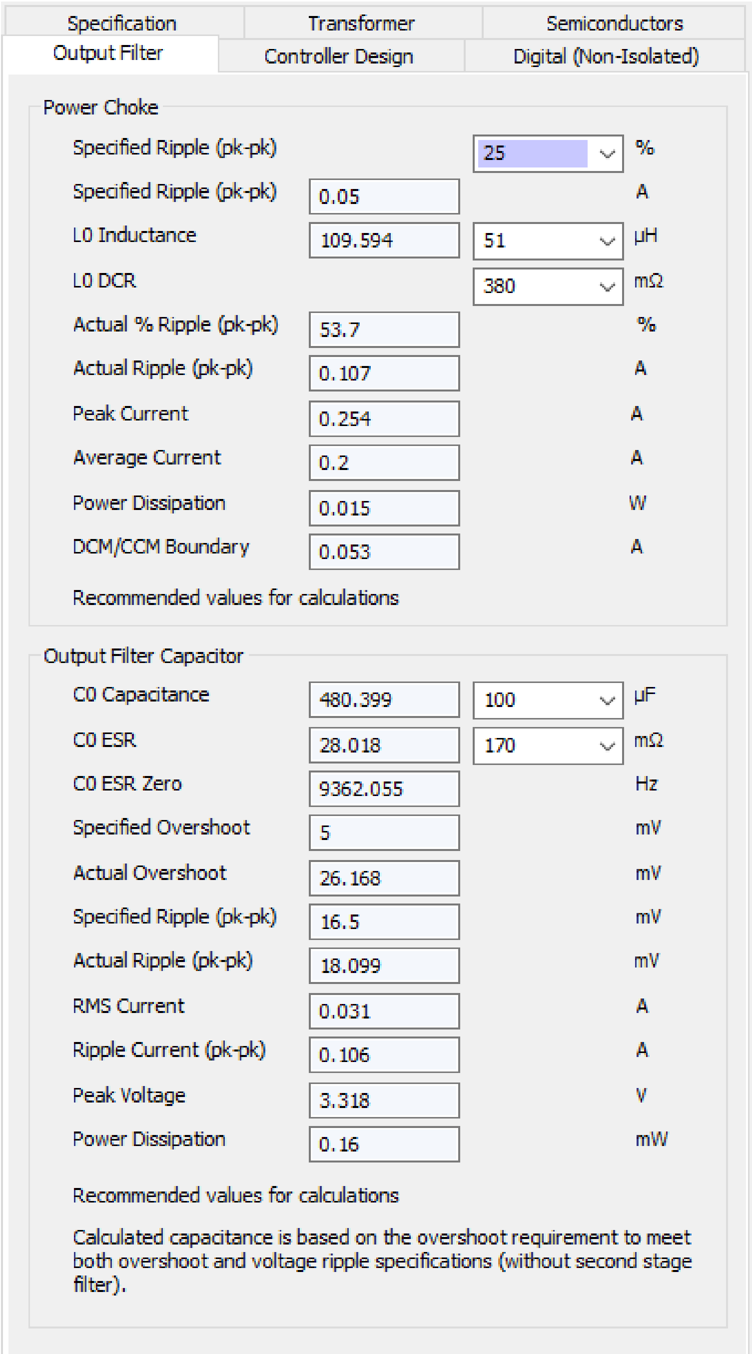
The `Output Filter` tab is next and has the specification shown in Table 4.

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 55.

**Figure 55. ST-WDS output filter tab**

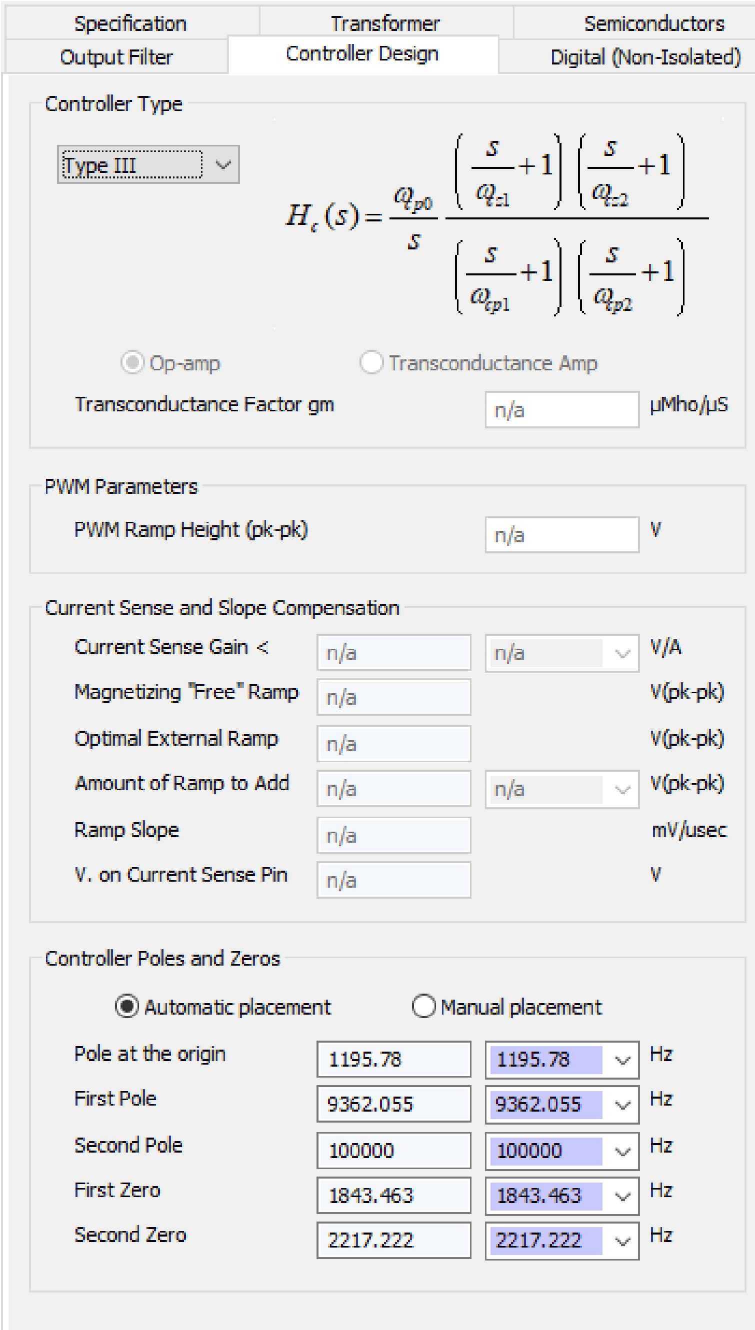


On the next tab, `Controller Design`, a Type-III compensator is automatically designed by ST-WDS by placing the compensator poles and zeros to achieve the desired crossover frequency and phase margin. There is no need to enter any parameters into this tab as the poles and zeros are already calculated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

**Table 5. Controller design parameters**

| Controller design parameter | Value |
|---|---|
| Pole at the origin | 1195.78 Hz |
| First pole | 9362.055 Hz |
| Second pole | 100000 Hz |
| First zero | 1843.463 Hz |
| Second zero | 2217.222 Hz |

**Figure 56. ST-WDS controller design tab**

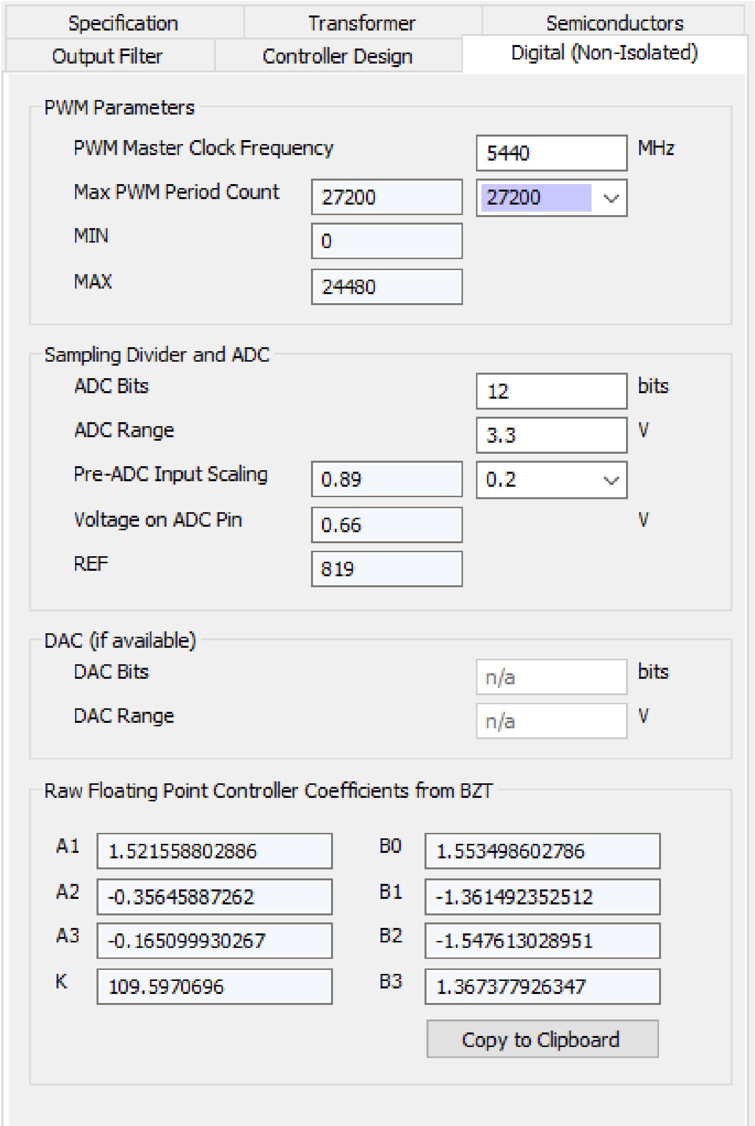



The last configuration tab is `Digital (Non-Isolated)`. Enter the specification given in the following table.

**Table 6. Digital nonisolated parameters**

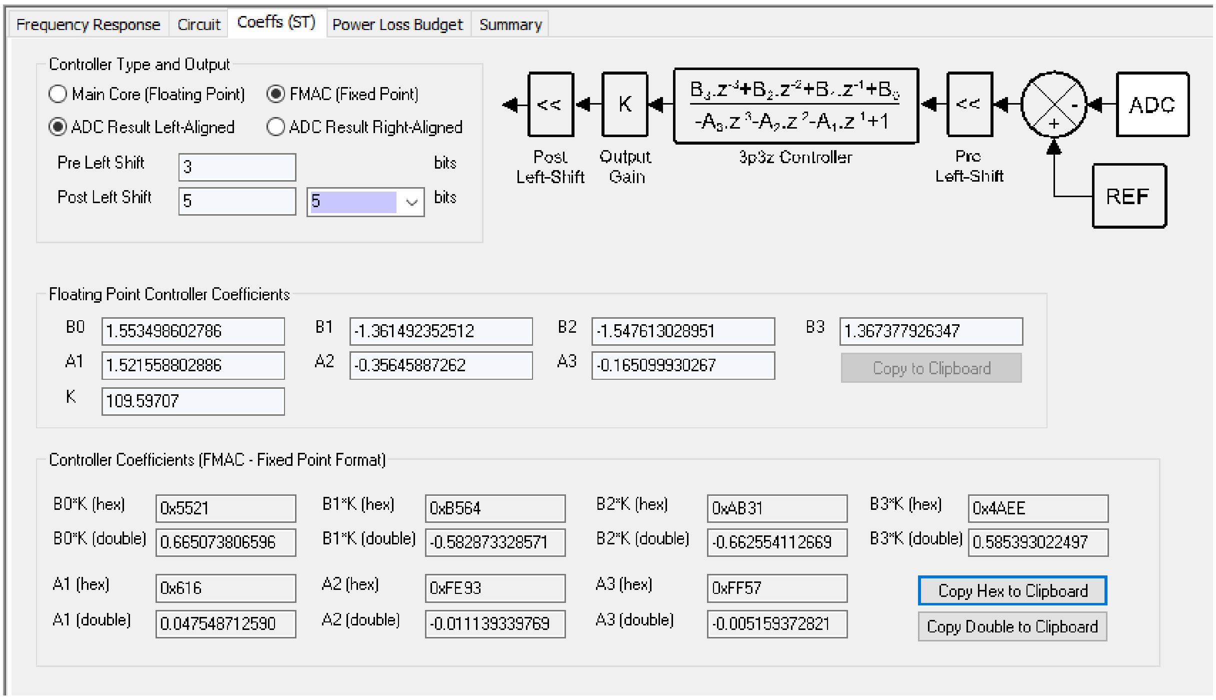
| Digital nonisolated tab parameter | Value |
|---|---|
| PWM controller clock frequency | 5440 MHz |
| Maximum PWM period count | 27200 |
| ADC bits | 12 bits |
| ADC range | 3.3 V |
| Pre-ADC input scaling | 0.2 |

Figure 57. ST-WDS digital nonisolated tab



Now that the controller coefficients are calculated, they need to be converted into the format required for a 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 58.

**Figure 58. 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 (0x5521)
#define B1 (0xB564)
#define B2 (0xAB31)
#define B3 (0x4AEE)
#define A1 (0x616)
#define A2 (0xFE93)
#define A3 (0xFF57)
#define pre_shift (+3)
#define post_shift (+5)
#define REF (812)
#define DUTY_TICKS_MIN (0x60)
#define DUTY_TICKS_MAX (24480)
```

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

## 6.7 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. And this happens 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. 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.

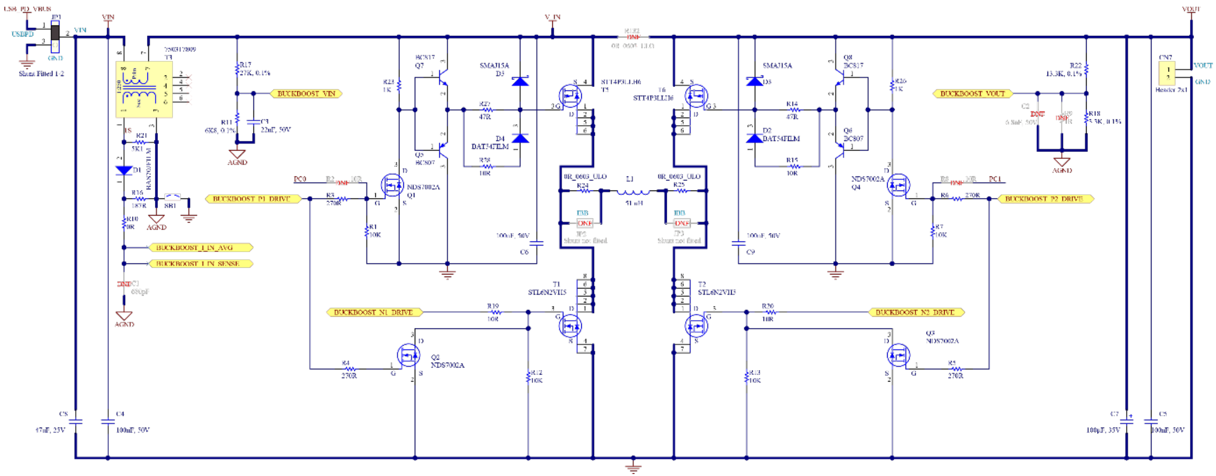
# 7 Design example

## 7.1 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 step-down converter power stage and presents the design equations used to select the appropriate components.

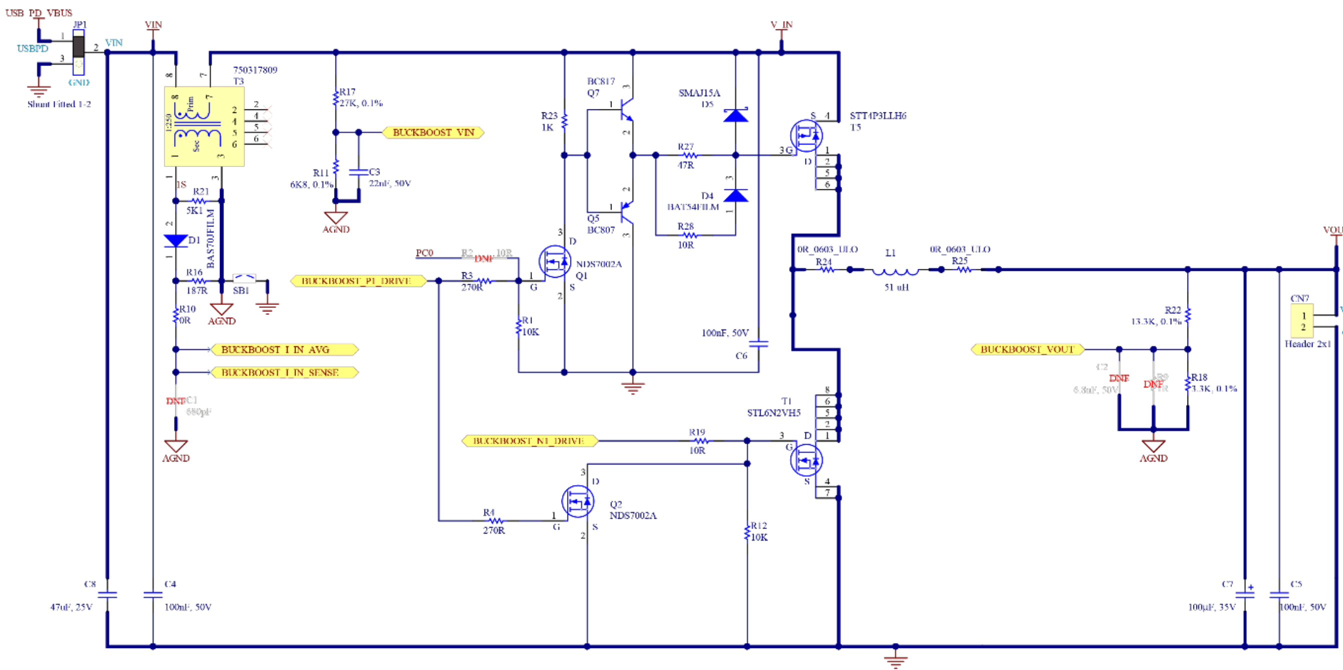
The step-down converter is implemented on this discovery kit as part of a buck-boost converter. It means that there are boost switches as well as buck-switching FETs. The extract from the schematic shown in Figure 59 identifies the relevant switches. The full schematic can be downloaded from B-G474E-DPOW1.

Figure 59. 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). The simplified circuit shown in Figure 60 can be used to describe the power stage.

Figure 60. Step-down 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 (42). This does not include any of the parasitics.

$$D = \frac{V_{OUT}}{V_{IN}} = \frac{3.3\,V}{5\,V} = 66\,\% \quad (42)$$

### 7.1.1 Power inductor

The power stage inductor is sized-based on the maximum allowable ripple current. It is according to the switching frequency given the input and output voltage specification. Using the standard equation for the voltage rise on an inductor given in (43) rearranged for inductance, the inductance required to meet the specification can be calculated using (44).

$$V = L \cdot \frac{dI}{dt} \quad (43)$$

$$L_0 = \frac{D \cdot T_S \cdot (V_{IN} - V_{OUT})}{\Delta I_{L\%} \cdot I_O} \quad (44)$$

Where $T_S$ is the switching period, therefore:

$$L_0 = \frac{66\,\% \cdot 5\,\mu s \cdot (5\,V - 3.3\,V)}{50\,\% \cdot 0.2\,A} = 56\,\mu H \quad (45)$$

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.

### 7.1.2 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 step-down converter is due to the voltage drop 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.

$$R_{ESR(\max)} = \frac{V_{RIPPLE}}{\Delta I_L} \quad (46)$$

$$R_{ESR(\max)} = \frac{V_{OUT} \cdot V_{RIPPLE\,\%} \cdot L_0}{D \cdot T_S \cdot (V_{IN} - V_{OUT})} \quad (47)$$

$$R_{ESR(\max)} = \frac{3.3\,V \cdot 1\,\% \cdot 51\,\mu H}{66\,\% \cdot 5\,\mu s. \cdot (5\,V - 3.3\,V)} = 0.33\,\Omega \quad (48)$$

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 $1.7 \times 10^{-5}$, therefore the capacitance can be calculated in (49).

$$C_{0(\min)} = \frac{1.7 \cdot 10^{-5}}{R_{ESR(\max)}} \quad (49)$$

$$C_{0(\min)} = \frac{1.7 \cdot 10^{-5}}{0.33} = 51\,\mu F \quad (50)$$

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

## 7.2 PCB layout

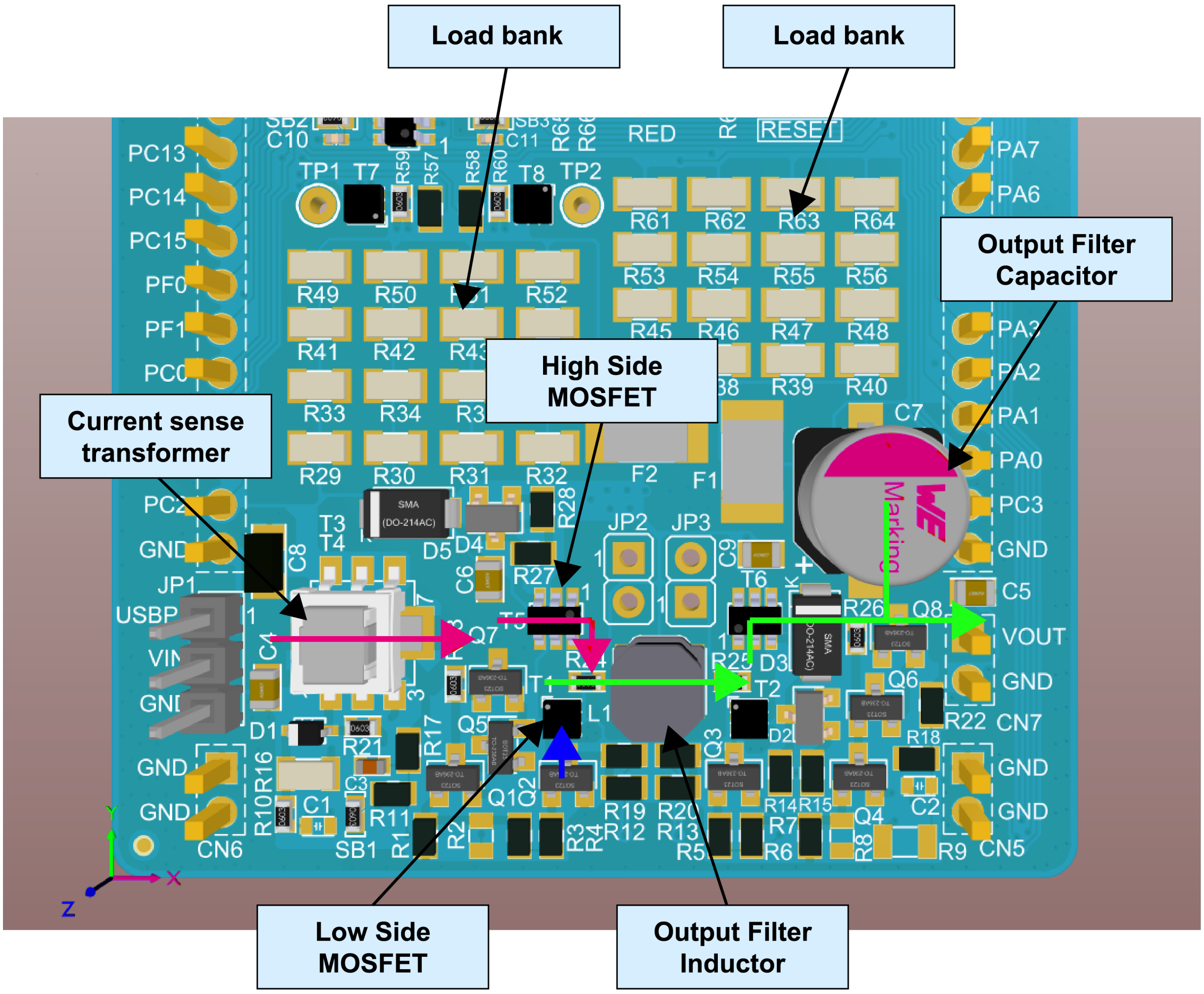
The layout of the step-down converter is shown in Figure 61. In this figure, various key components of the step-down converter are identified. When laying out the step-down converter PCB, it is important to consider good PCB design practices. For example, loops with high di/dt must be minimized. Although 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 step-down converter) make the resistive divider that scales the output voltage of the step-down converter before it is fed to the ADC. There may be an antialiasing 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 that can be picked up by this trace.

The ADC pin for this output voltage feedback is PA3 that 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 61 the current path of the step-down converter is shown as a series of colored arrows. The fuchsia arrow indicates the current through the current sense transformer (not used for voltage mode) 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 61. Hardware layout of the power stage**



# 8 Getting started

## 8.1 Overall usage

The STM32 MCU embedded in the discovery kit comes preflashed with example software. This example software exercises the other functions of the discovery kit such as control of the RGB LED. To run the step-down 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 v6.9.2) 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 Type-C® and connect this to CN2 on the discovery kit.
3. Ensure that the jumper JP1 is in the USB PD-VIN position.

## 8.2 Loading the project

This voltage 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*

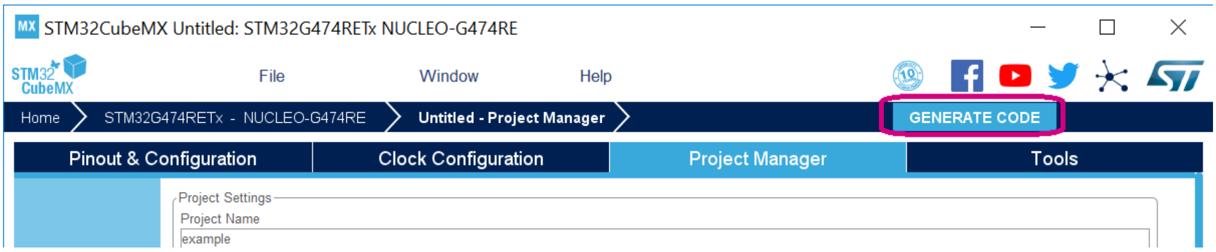
Opening the Project

1. Open STM32CubeMX by clicking on the icon shown in Figure 62 (note that the icon and the path may differ slightly).

**Figure 62. Example.ico**

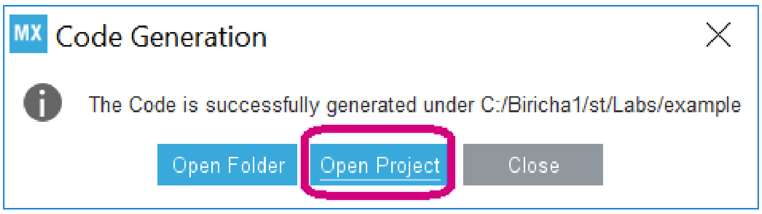

2. Generate the project by clicking the `GENERATE CODE` button within STM32CubeMX.

**Figure 63. Code generation**



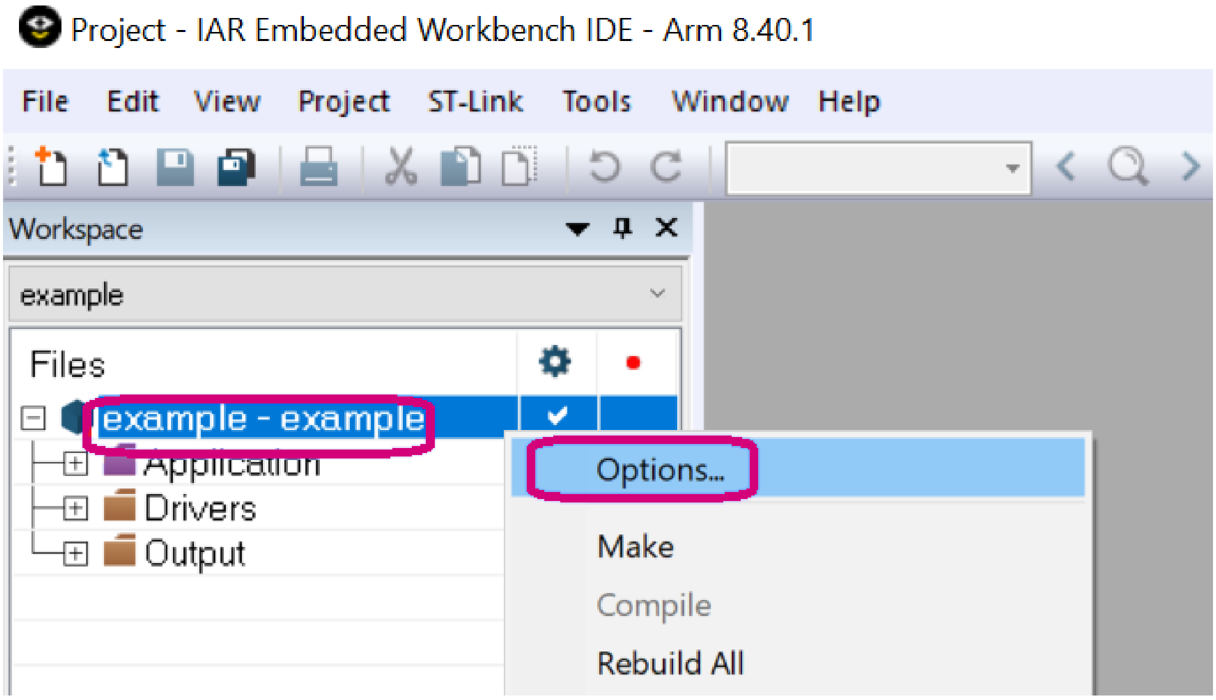

3. This generates the project for the selected IDE. In this case, it is an IAR Systems® project.
4. When prompted with the dialog box, open the project by clicking `Open Project`.

**Figure 64. Project opening**



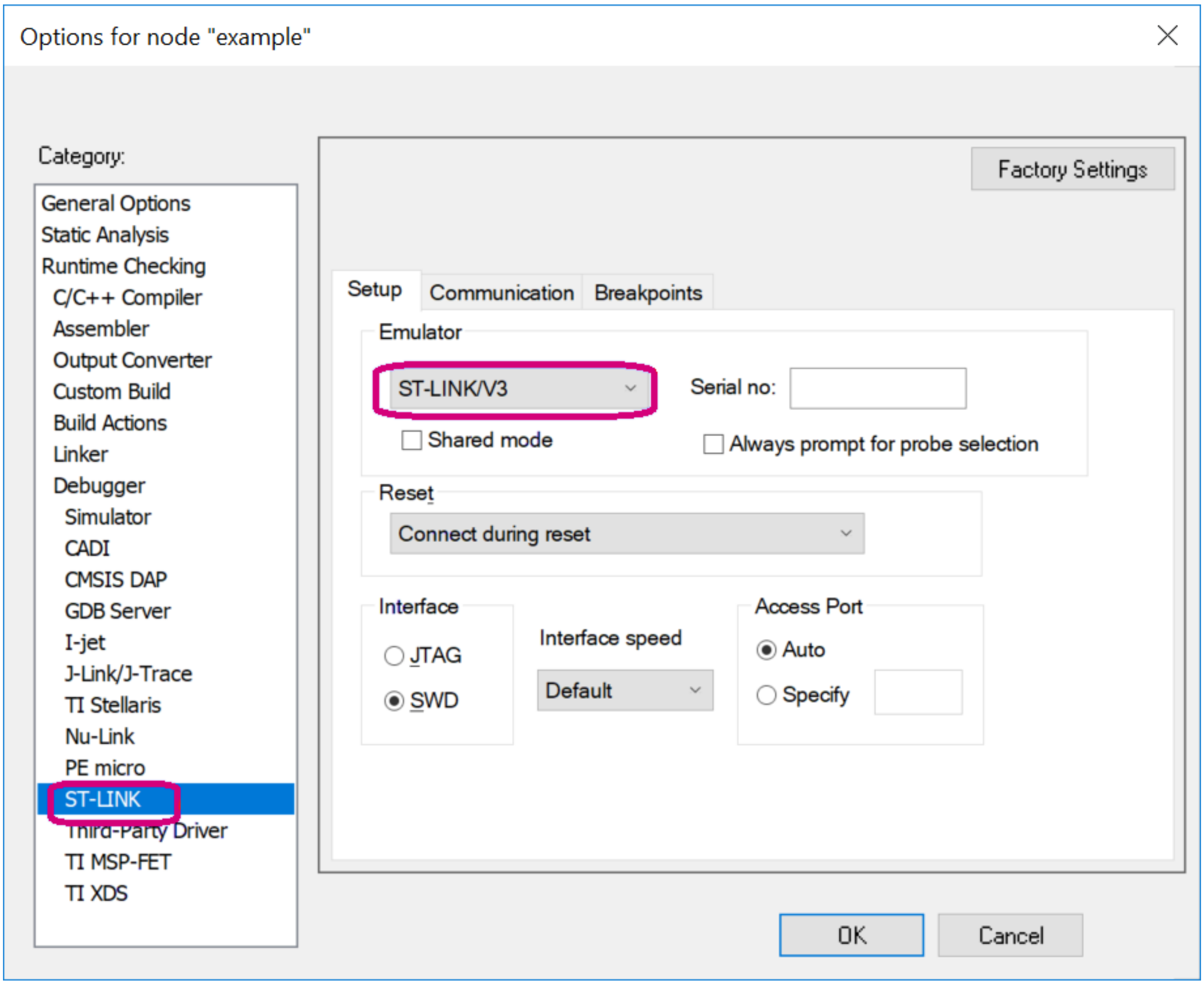

5. Depending on the IAR Embedded Workbench® used version, the default debugger may need to be changed to STLINK-V3. Right-click on the project name and select `Options`.

Figure 65. Example options



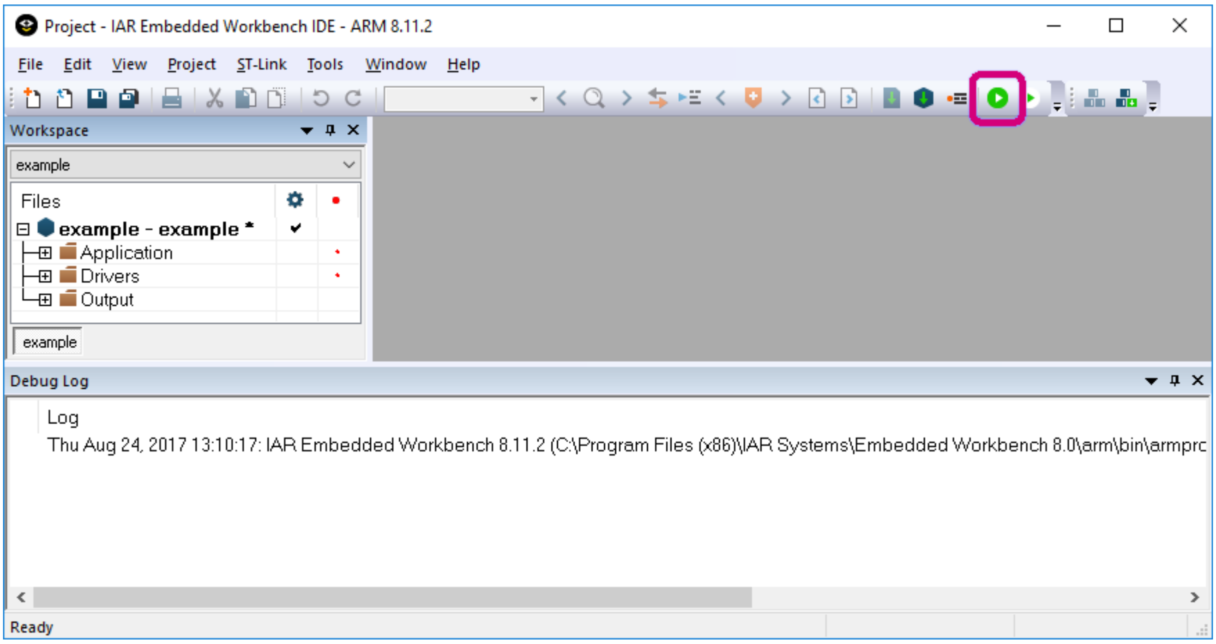

Then select `ST-LINK` and choose `ST-LINK/V3`. Click `OK` to finish.

**Figure 66.** **ST-LINK version selection**



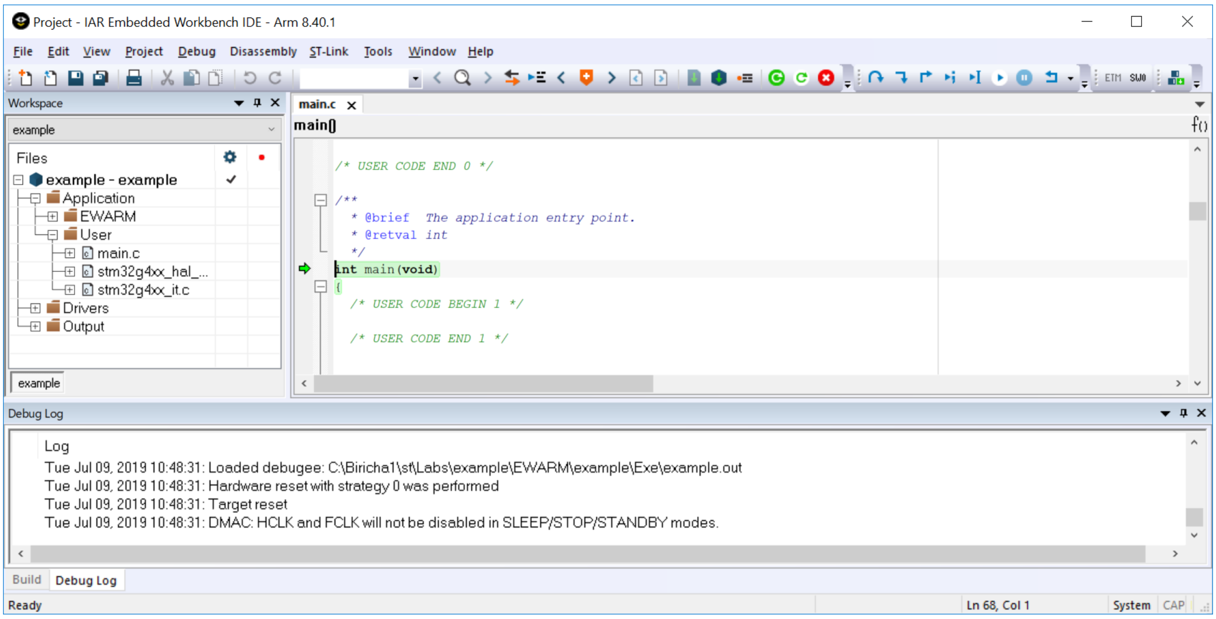

6. Inside the IAR Systems® IDE click the `Download and Debug` icon that compiles and downloads the code to the MCU.

**Figure 67. Code compilation and downloading**



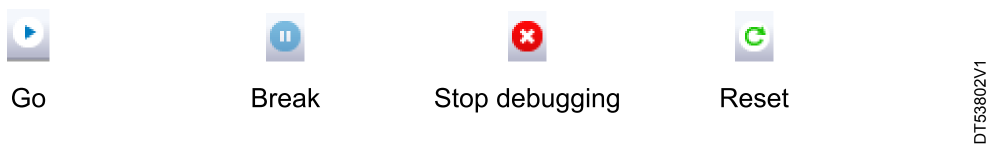

7. G4 IntroPack example, in the `application`, `User` folder, the three generated C-files are:
   a. `main.c`: Initialization code
   b. `stm32g4xx_hal_msp.c`: MCU support package initialization
   c. `stm32g4xx_it.c`: Interrupt handlers for ISRs
8. D-Power pack example, in the `application`, `User` folder, the generated C-files are:
   a. `app_X-CUBE-DPower.c`: Application program body.
   b. `DCDC_Control_Layer.c`: Hardware control function body.
   c. `DCDC_Fault_Processing.c`: DC/DC fault management.
   d. `DCDC_Globals.c`: Global variables for the application.
   e. `DCDC_IRQ_Handlers.c`: Interrupt service routines.
   f. `DCDC_PWMnCurrVoltFdbk.c`: Functions to drive PWM signals accordingly to existing features.
   g. `DPC_STM32_Interface.c`: Source file for the library wrapper.
   h. `main.c`: Main program body
   i. `stm32g4xx_hal_msp.c`: MSP initialization and deinitialization codes.
   j. `stm32g4xx_it.c`: Interrupt service routines.
9. The code is now ready to run. The IDE moves the program counter to the `int main(void)` function.

Figure 68. Program counter set for running




10. The buttons along the top menu bar have the functions described below. Click the `Go` button to run the code.

Figure 69. Top menu buttons




- 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

11. Click on `Stop Debugging` to end the debug session.
12. 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.
13. Close IAR Systems® to stop the debugger. If IAR Systems® asks to `Terminate the debug session`, click OK.

## 8.3 Onboard load

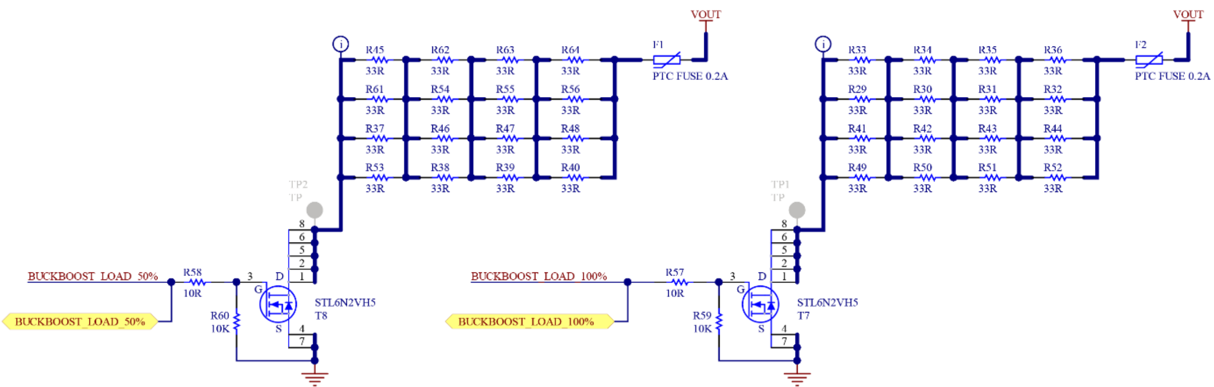
The step-down converter on the discovery kit includes two onboard load banks. Figure 61 shows the two load banks and the position of two test pins that indicate whether the load bank is enabled (test pins are not populated). When TP1 is pulled LOW, load bank 1 is ON. When TP2 is pulled LOW, load bank 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%<br>Load 1 OFF<br>Load 2 OFF | ∞ Ω | 0 A | 0 W | All OFF |
| 50%<br>Load 1 ON<br>Load 2 OFF | 33 Ω | 0.1 A | 0.33 W | Green |
| 100%<br>Load 1 ON<br>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 70.

Figure 70. **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 the 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. It is used to test the transient response of the step-down converter and the control loop regulating the output voltage. The status of the load bank during the transient is indicated by the LEDs on the discovery kit board. If load bank 1 is enabled, the green LED lights up. If load bank 2 is enabled, the orange LED also lights up.

## 8.4 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 configuration data for STM32CubeMX setting up the 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 configuration data for STM32CubeMX setting up the 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 configuration data for STM32CubeMX setting up the 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 G4 intro pack example:**

- `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`
  - It interrupts service routines.
- `system_stm32g4xx.c`
  - It provides the SystemInit() initialization functions for this MCU configuring the system clock.

**Source codes D-Power pack example:**

- `app_X-CUBE-DPower.c`
  - This file provides functions for this application to control the hardware on the board.
- `DCDC_Control_Layer.c`
  - Hardware control functions body
- `DCDC_Fault_Processing.c`
  - DC/DC fault management
- `DCDC_Globals.c`
  - Global variables for the application
- `DCDC_IRQ_Handlers.c`
  - Interrupt Service routines
- `DCDC_PWMnCurrVoltFdbk.c`
  - Functions to drive PWM signals accordingly to existing features
- `DPC_STM32_Interface.c`
  - Source file for the library wrapper
- `main.c`
  - Main program body
- `stm32g4xx_hal_msp.c`
  - MSP initialization and deinitialization codes
- `stm32g4xx_it.c`
  - Interrupt service routines

**ST-WDS from Biricha files:**

- `Buck_VoltageMode_xxx.wds`
  – To match the STM32CubeMX project file, the ST-WDS configuration file enables the user to load the step-down converter design 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_VoltageMode_xxx.bode3`
  – To match the STM32CubeMX project file, the Bode Analyzer Suite configuration file configures the tool for loop measurement of the step-down converter. The file also contains the previously measured traces stored in memory locations.

## 8.5 Open-loop operation

Before exercising the step-down 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 step-down 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 step-down 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 the 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 = 200 ticks = 147 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 71. Between the falling edge of PB12 and the rising edge of PB13, there are 147 ns of dead time as measured in Figure 72.

Figure 71. **Rising edge dead-time, Ch1: high-side FET, Ch2: low-side FET, dead-time measured as 54 ns**

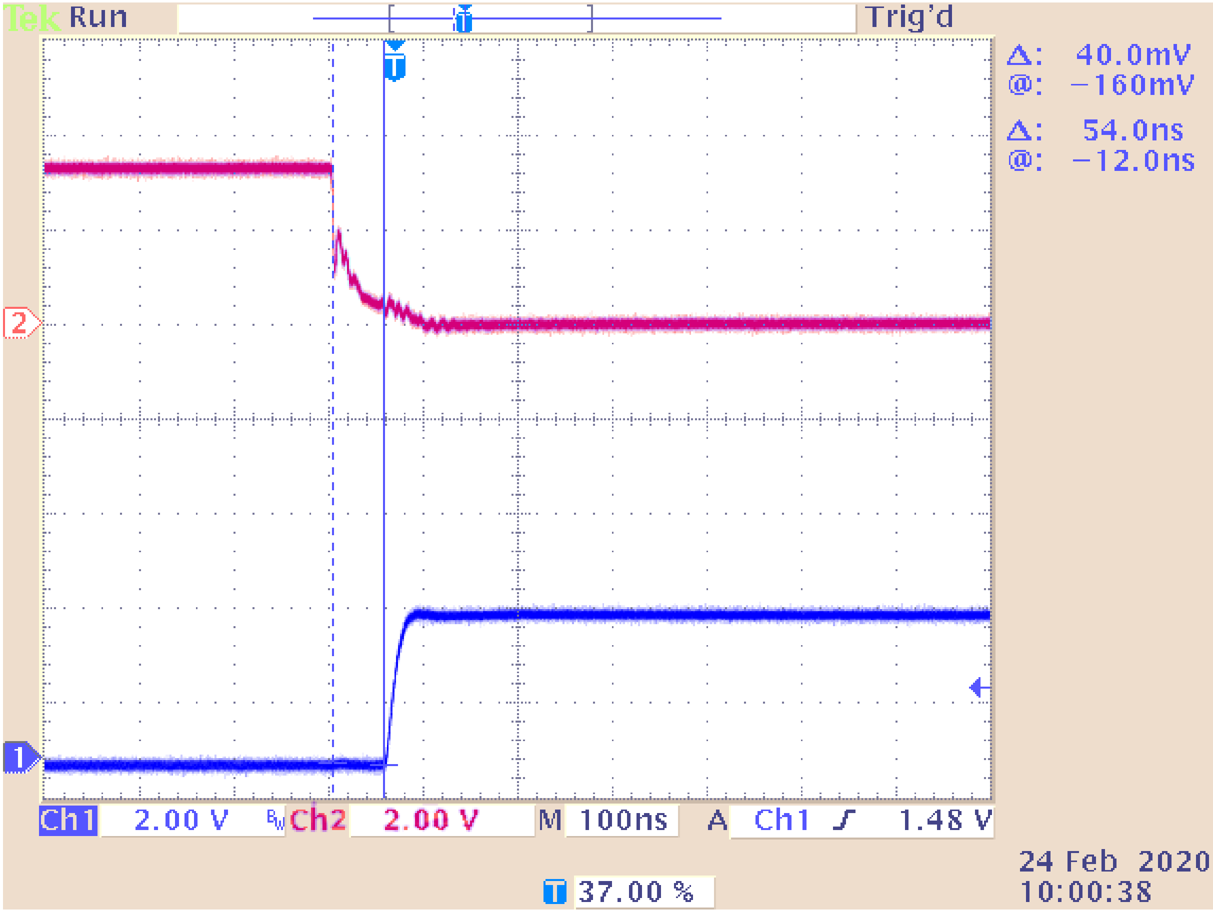
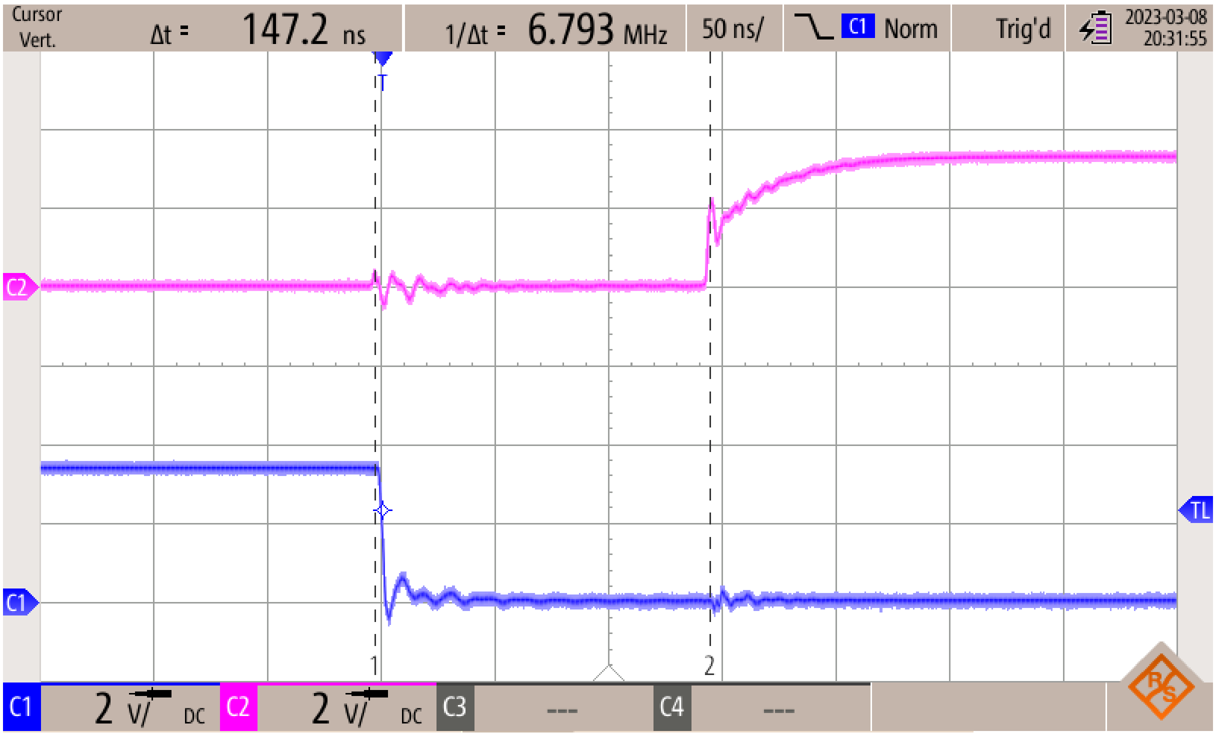


Figure 72. **Falling edge dead-time, Ch1: high-side FET, Ch2: low-side FET, dead-time measured as 147 ns**




The output voltage of the step-down 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 1.5 V. If the load is added by moving the joystick left or right, then the output voltage also changes.

## 8.6 Closed-loop control

### 8.6.1 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 step-down 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`. Rebuild 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. 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%<br>Load 1 OFF<br>Load 2 OFF | 0 A | 3.3 V |
| 50%<br>Load 1 ON<br>Load 2 OFF | 0.1 A | |
| 100%<br>Load 1 ON<br>Load 2 ON | 0.2 A | |

### 8.6.2 Transient response

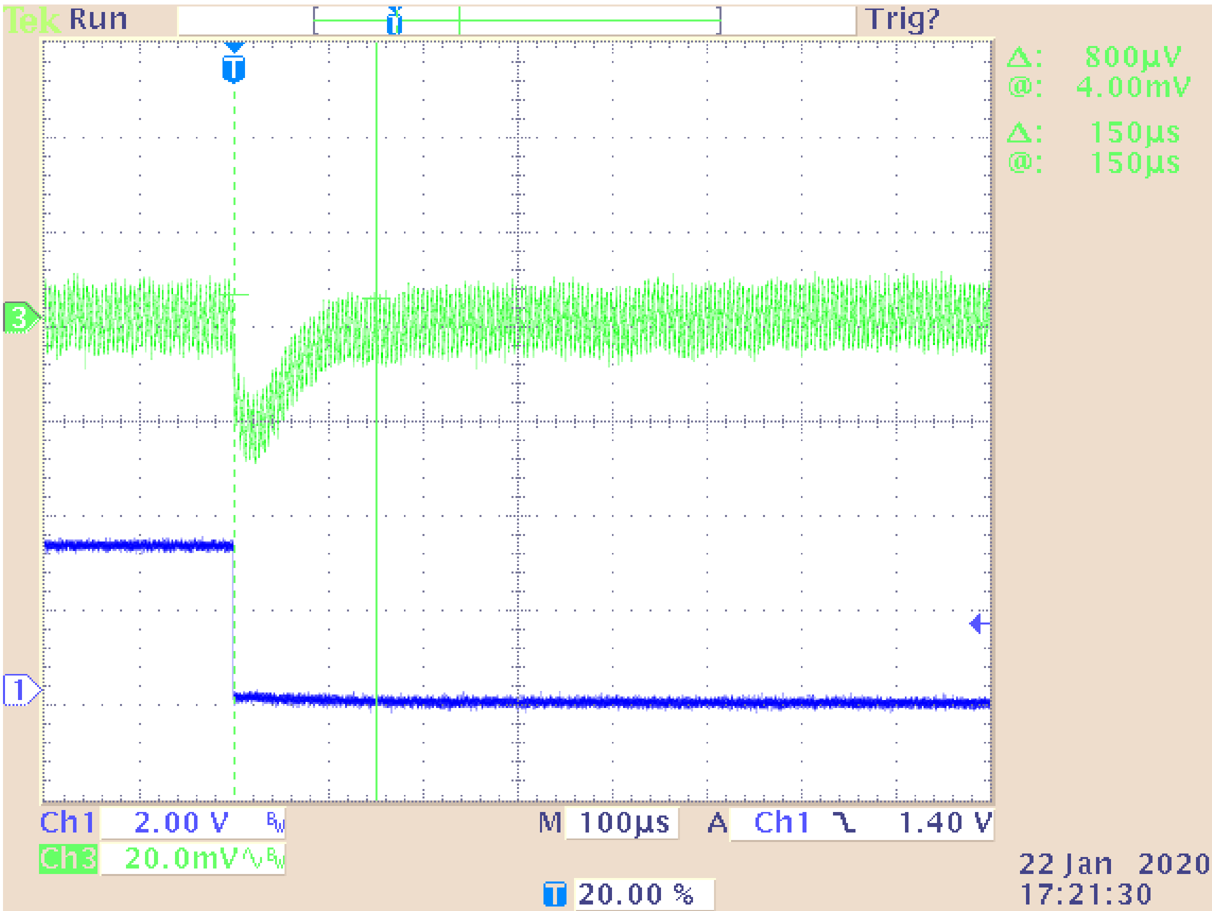
The onboard load allows the user to perform transient response tests on the closed-loop digital power supply to assess the performance of the controller. The transient response test is a useful method of determining if the implemented controller is stable, the speed of the response, and whether there 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 $V_{out}$
- Channel 2 -> Put the 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 interval. 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 73.

**Figure 73. Output voltage (Ch3) transient from load change 50% to 100% (Ch1)**




*Note: For this figure, output voltage undershoot is 30 mV and settling time is 150 µs.*

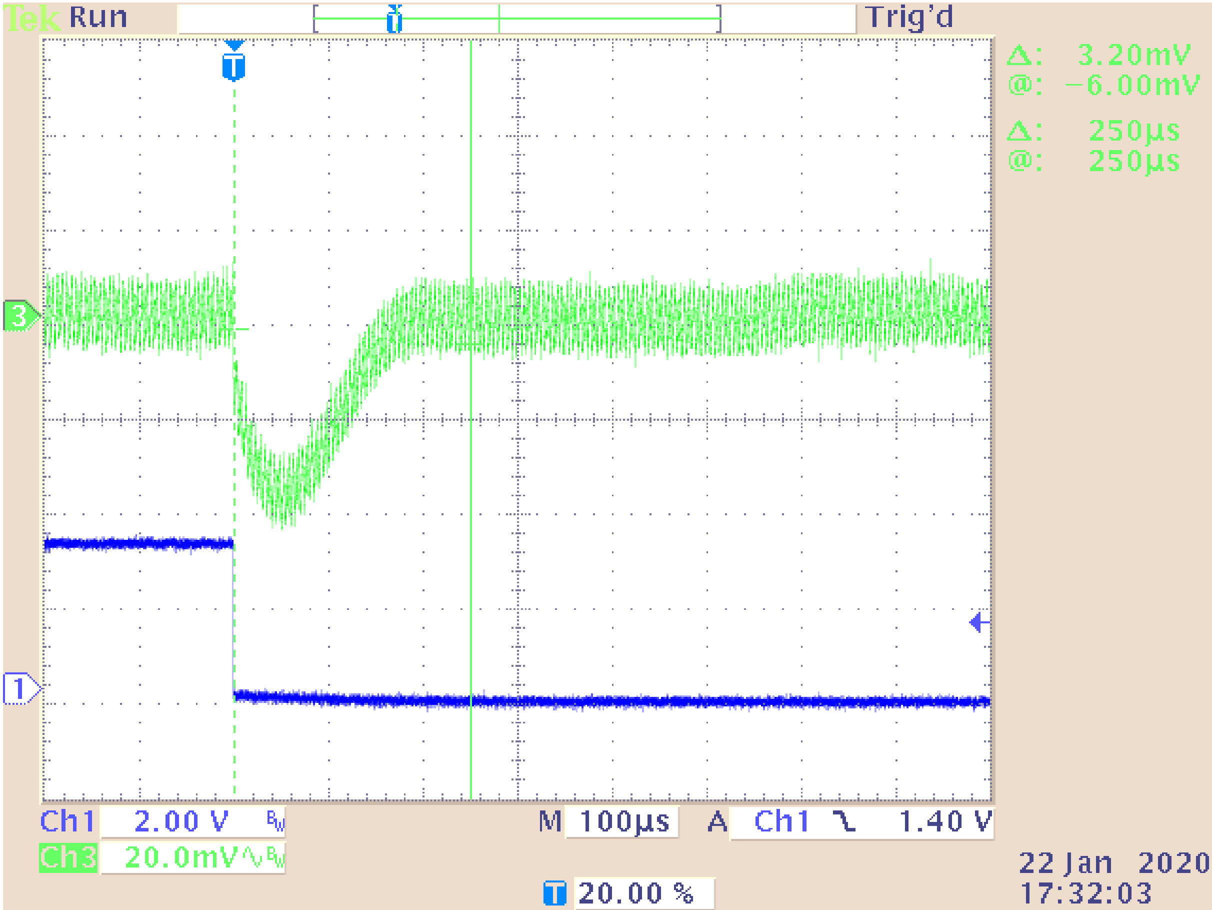
Figure 73 shows the transient response of the closed-loop digital step-down converter for a 50% to 100% step load transient with a settling time of fewer than 150 µs, and an undershoot of less than 40 mV with no ringing.

### 8.6.3 Impact of crossover frequency on the transient response

The transient response of the converter in Figure 73 indicates a stable and well-designed controller. This is using the controller which is described in detail in Section 4 of this application note. The controller is designed to have a crossover frequency of 8 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 74. Impact of crossover frequency. In this plot, the controller is redesigned with a crossover frequency of 4 kHz, half that of the previous controller.

**Figure 74. Impact of crossover frequency**



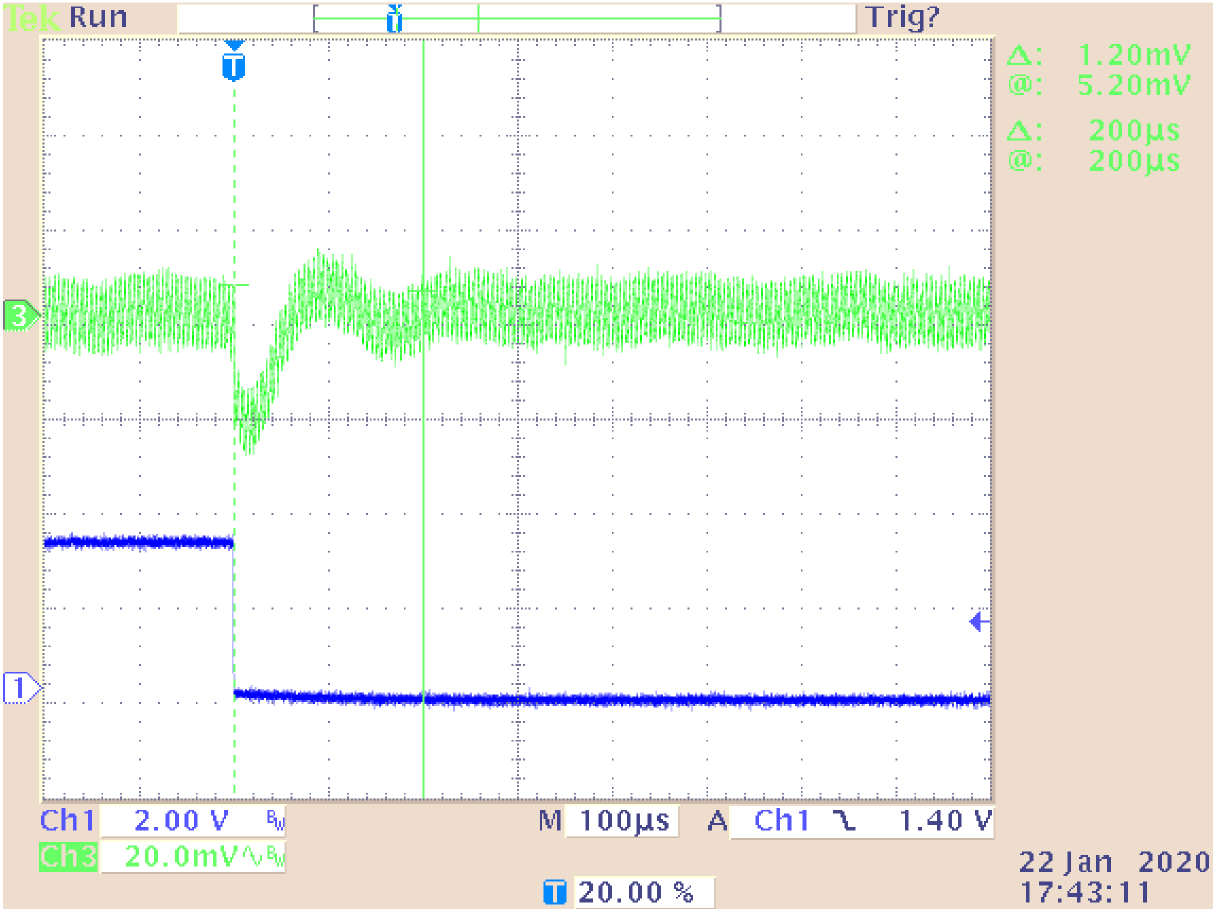

*Note: For this figure, output voltage (Ch3) transient from load change 50% to 100% (Ch1) with loop crossover of 4 kHz, output voltage undershoot is 40 mV and settling time is 250 µs.*

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 8.6.4: Impact of phase margin on the transient response.

### 8.6.4 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, 8 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 75 for the system with a phase margin of 30°.

**Figure 75. Impact of phase margin on the transient response**




*Note: For this figure, output voltage (Ch3) transient from load change 50 to 100% (Ch1) with loop crossover of 8 kHz and a designed phase margin of 30°, output voltage undershoot is 30 mV and settling time is 200 µs.*

In Figure 75 there are several oscillations in the transient response as it recovers 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 there are oscillations during the recovery. If the phase margin is reduced further, the oscillations take longer to decay. With a phase margin of 0°, the system may oscillate indefinitely.

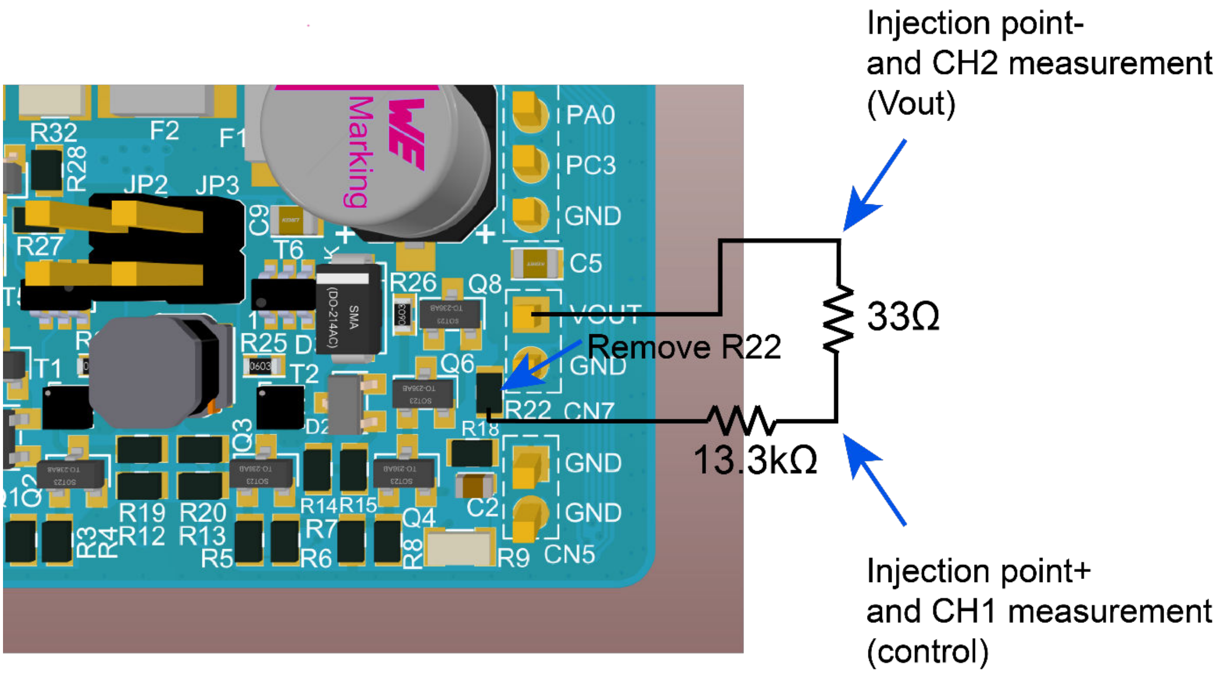
### 8.6.5 Bode plot measurement under closed-loop control

The real-life loop response of the step-down 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.

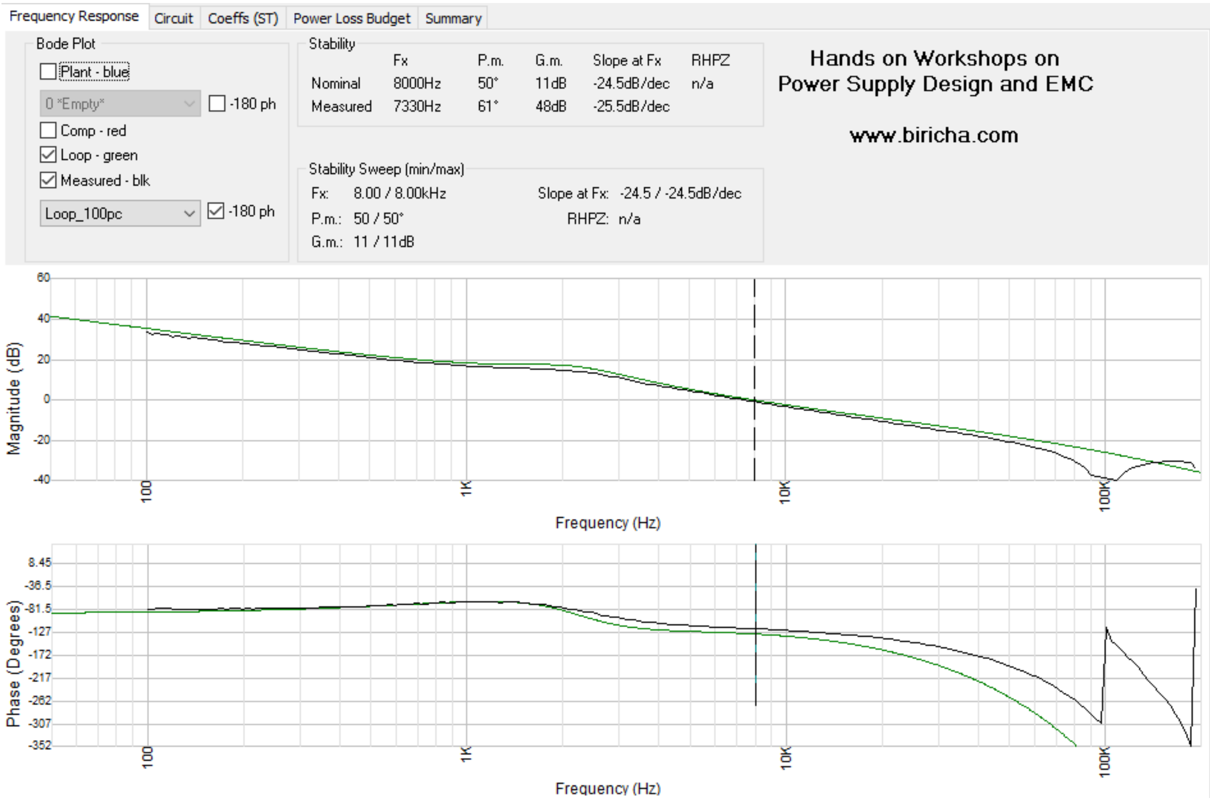
The schematic for this modification is shown in Figure 76. The measurement probes are then connected as follows: CH1 (or the control measurement point) is connected to the end of the injection resistor, which is not connected to the output voltage. Then CH2 (or the output measurement point) is connected 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 76. 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 77.

**Figure 77. Loop measurement at 100% load imported into ST-WDS**




*Note:* *Black means the measured loop, and green means the simulated loop.*

In Figure 77 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 8 kHz in simulated and 7.33 kHz as measured.

The phase response is also a good match at the lower frequencies. Around the double-pole frequency of the plant (2 kHz), there is some discrepancy that is likely due to the AC resistance of the inductor that is not included in the plant model. This has some effect on the Q factor of the complex conjugate pole, and therefore the rate at which the phase transitions. The result is a higher phase margin than anticipated. If the phase margin is too high, it is possible to redesign the control loop to take into account the additional phase. However, it is also prudent to check the crossover and phase margin at different loads. This is discussed in the next section.

## 8.7 Waveform display in IAR Systems®

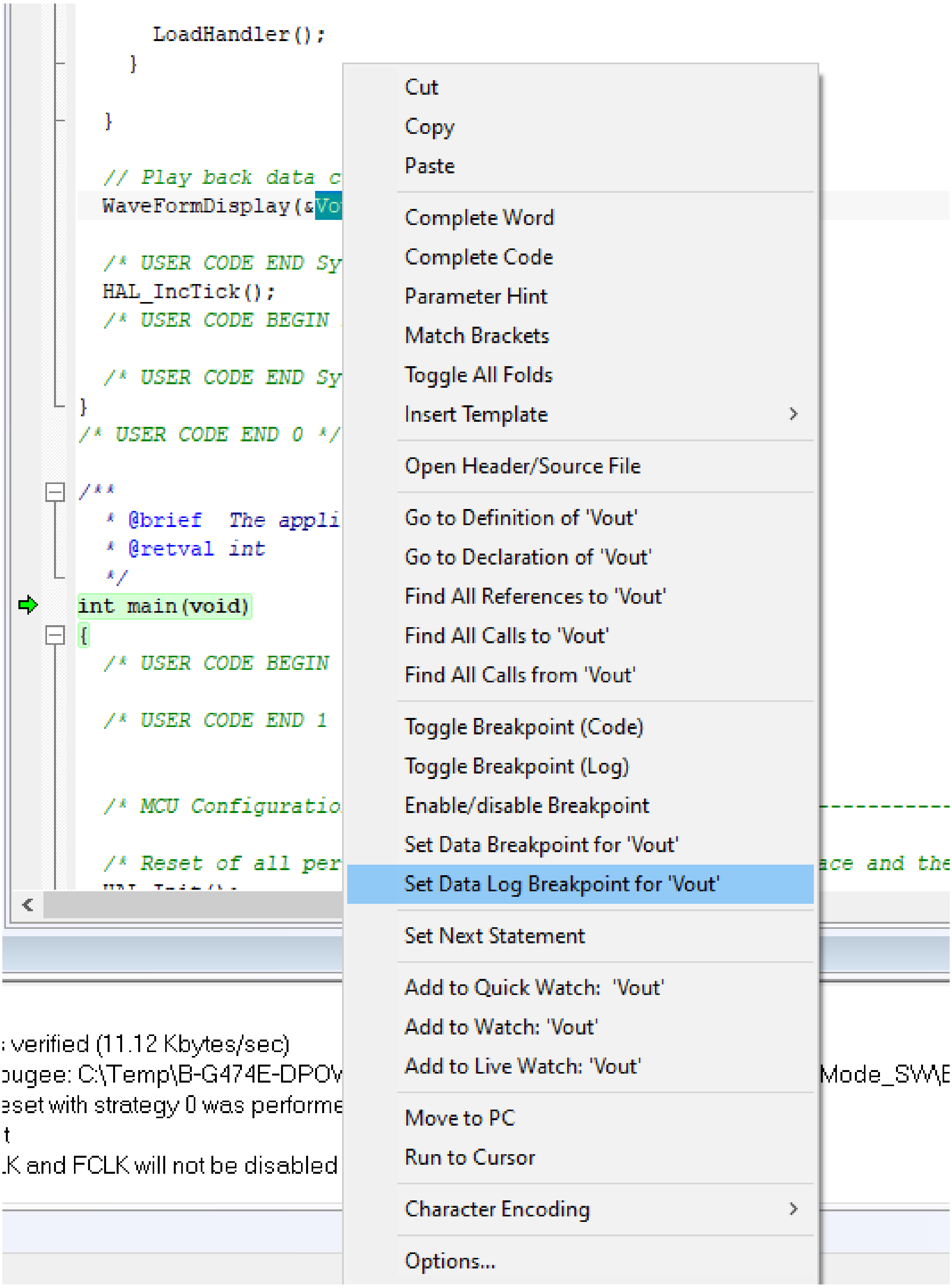
If the user has access to IAR Systems® then it is possible to use the timeline feature within this to plot variables captured using data log breakpoints. This is suitable for slow-changing variables in real-time, or fast-changing variables that are stored in a buffer and then transferred via the debugger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

An example of this functionality is included in the project provided `Buck_VoltageMode_SW`. Note that this project executes the 3p3z 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_VoltageMode_SW.ioc` file within the folder and click `GENERATE CODE`. Open IAR Systems®, locate, and uncomment the line of code `#define PLOT_WAVEFORM`, 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 78. **Add the 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.

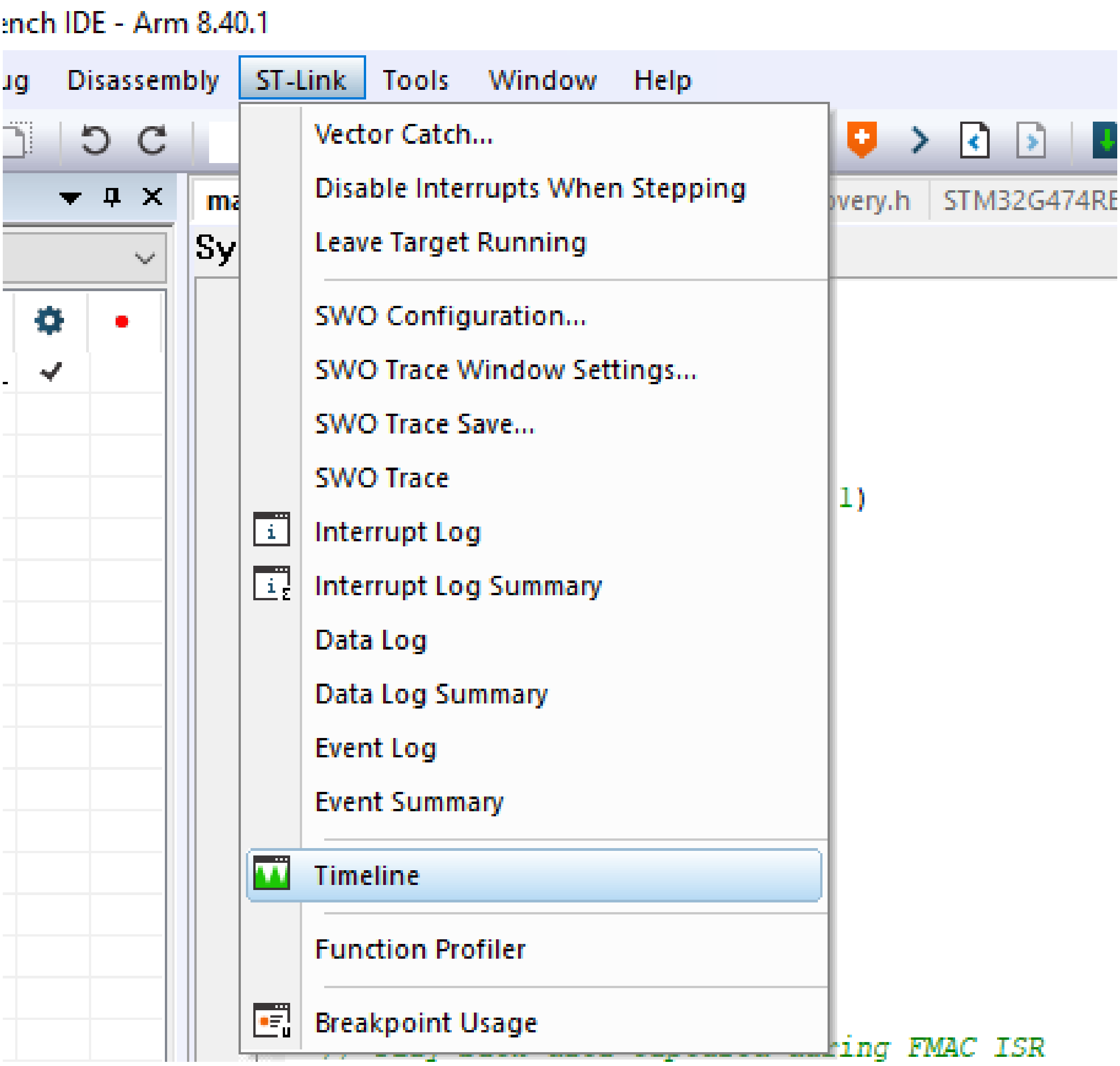
The data is captured within the ADC ISR located in the `stm32g4xx_it.c` file. The code snippet below shows the data being passed to the `WaveFormRecord()` function each time the ISR is called.

```
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 79.

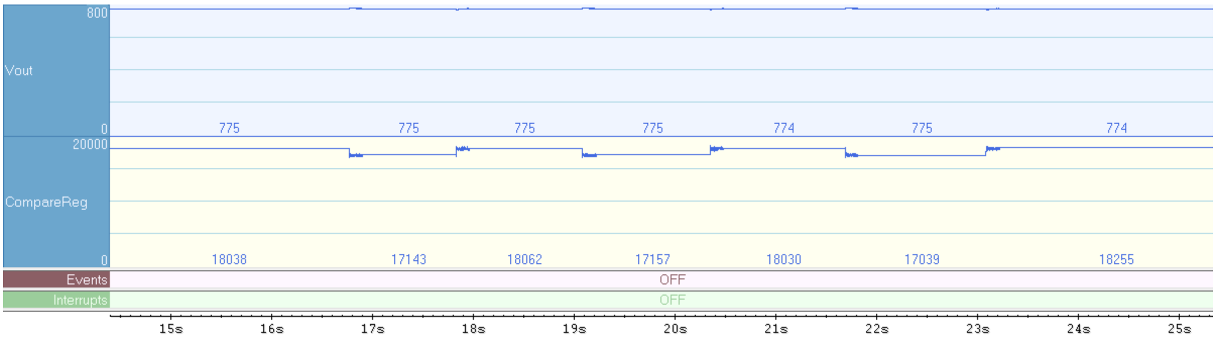
Figure 79. **Enabling timeline view**




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 Figure 80.

**Figure 80. Timeline pane**




*Note: 2For this figure, 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 replayed 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 81 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 Figure 81.

**Figure 81. Timeline view with an adjusted scale**

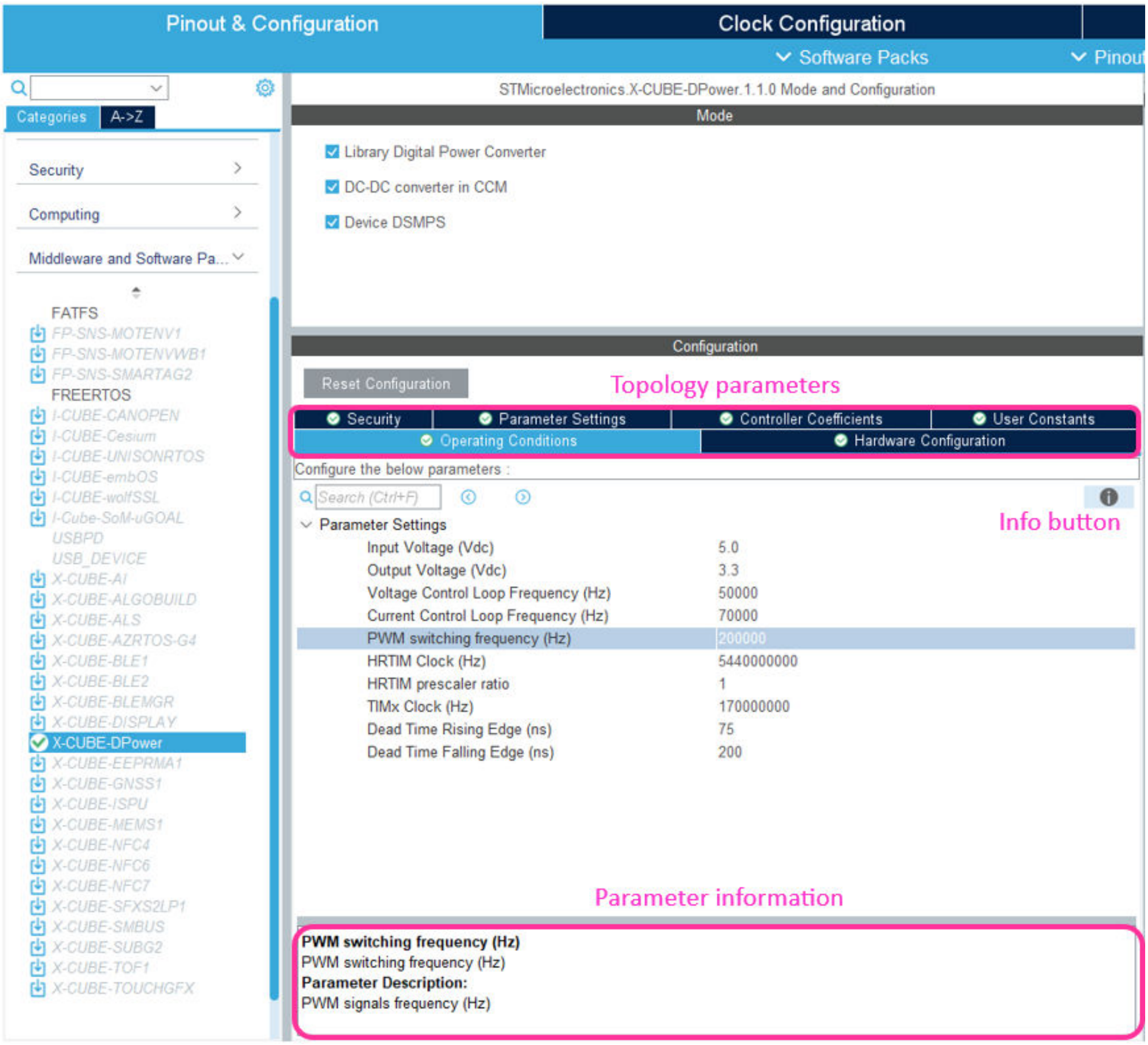
## 8.8 Step-down converter usage with the X-CUBE-DPower pack

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.

### 8.8.1 Configure the converter parameters using the GUI

As shown in Figure 82, they are many user parameters that are now available when using X-CUBE-DPower.

Figure 82. **X-CUBE-DPower – GUI parameters for the step-down converter**




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

### 8.8.2 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 step-down 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 83.
- Red: An error or a fault is detected. Highlighting details are given in the state machine diagram Figure 83.
- 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

### 8.8.3 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 step-down converter, `app_dpower.h` implements only the following user modalities (refer to the header file for a detailed description):

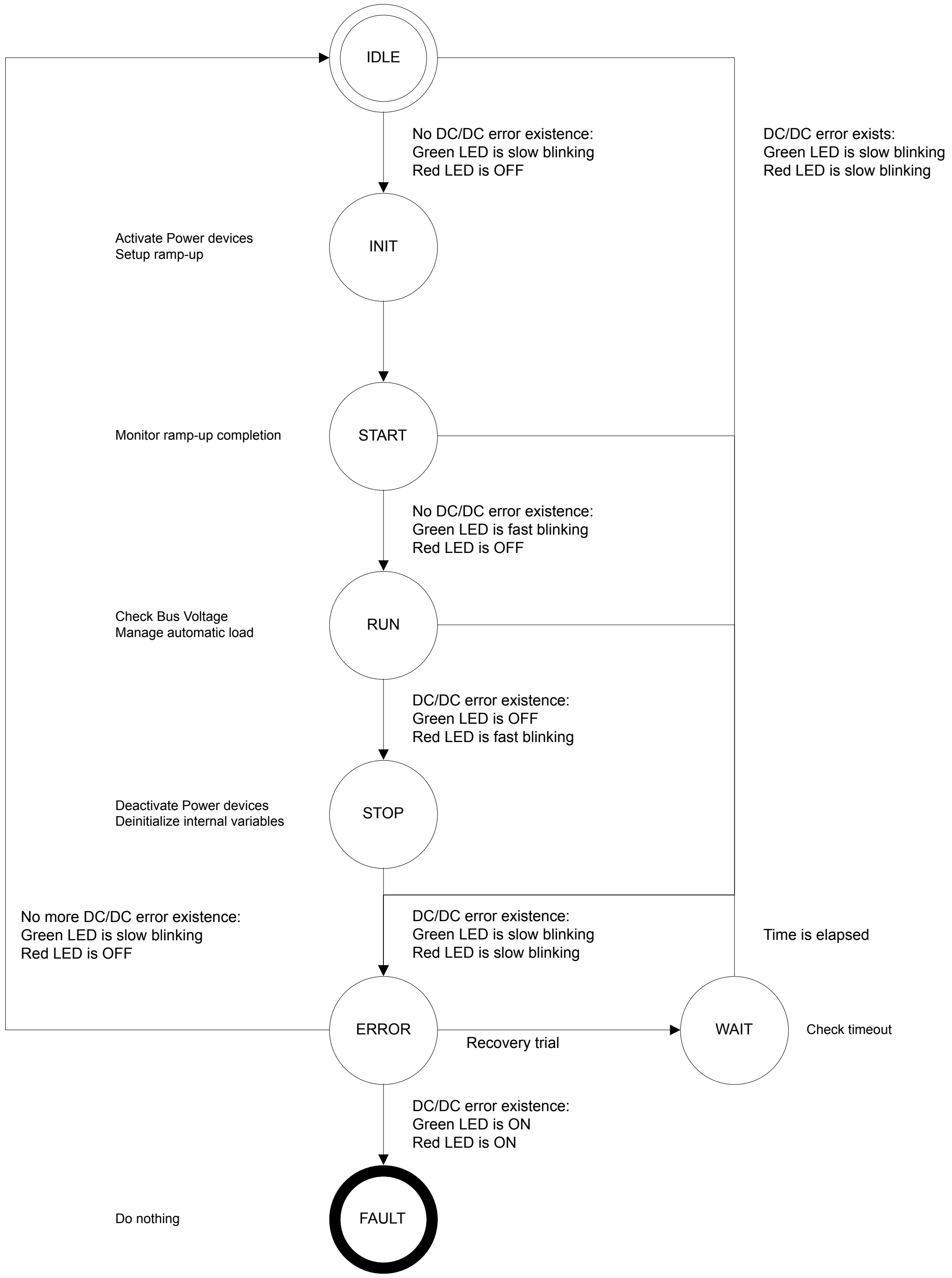
- #define CHANGE_CTRL_TO_PID
- #define OVERCURRENT_PROTECTION
- #define OVERVOLTAGE_PROTECTION
- #define SHORT_CIRCUIT_PROTECTION
- #define OVERTEMPERATURE_PROTECTION
- #define DEBUG_MODE
- #define DEBUG_COMP_OUT
- #define RUN_OPEN_LOOP
- #define PLOT_WAVEFORM

*Note: The protections that were mentioned are not currently in effect.*

### 8.8.4 State machine

The state machine implementation is provided in Figure 83.

**Figure 83. X-CUBE-DPower – step-down converter state machine**



# 9 Measurements

This section exercises the step-down converter embedded in the discovery kit using the provided voltage mode control example, and discusses the captured measurement results.

## 9.1 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 84 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 3.32 V with a duty cycle of 61.2%. The load is then increased to 50% of the rated output current in Figure 85 and the output voltage remains constant at 3.32 V and the duty cycle increases to 65.55%. Finally, the load is increased to 100% of the rated output in Figure 86 and again the output voltage remains at 3.32 V and the duty cycle increases to 66.28%. This indicates that the controller is regulating the output voltage of the step-down converter given the changes in load.

**Figure 84. Output voltage (Ch3) and PWM (Ch1) at 0% load**

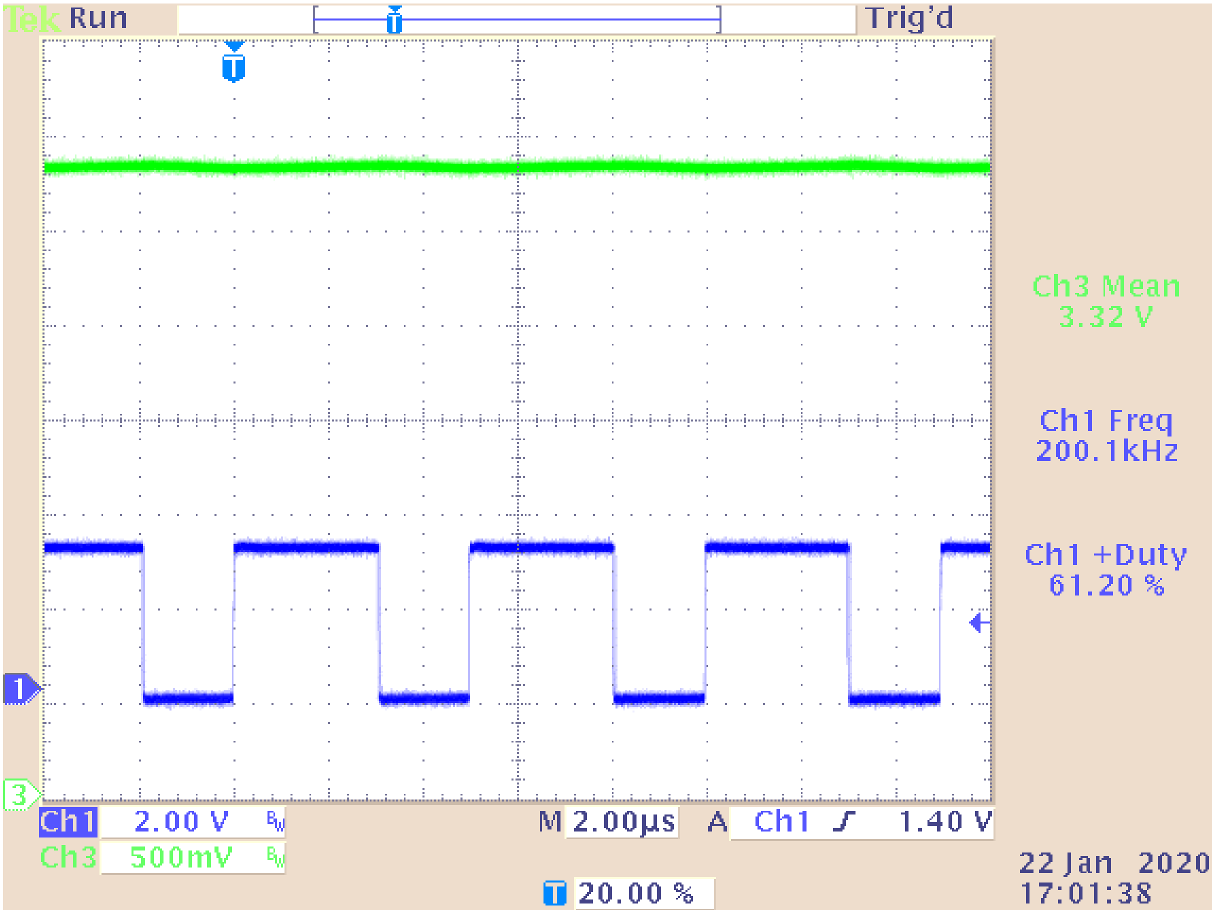


Figure 85. Output voltage (Ch3) and PWM (Ch1) at 50% load

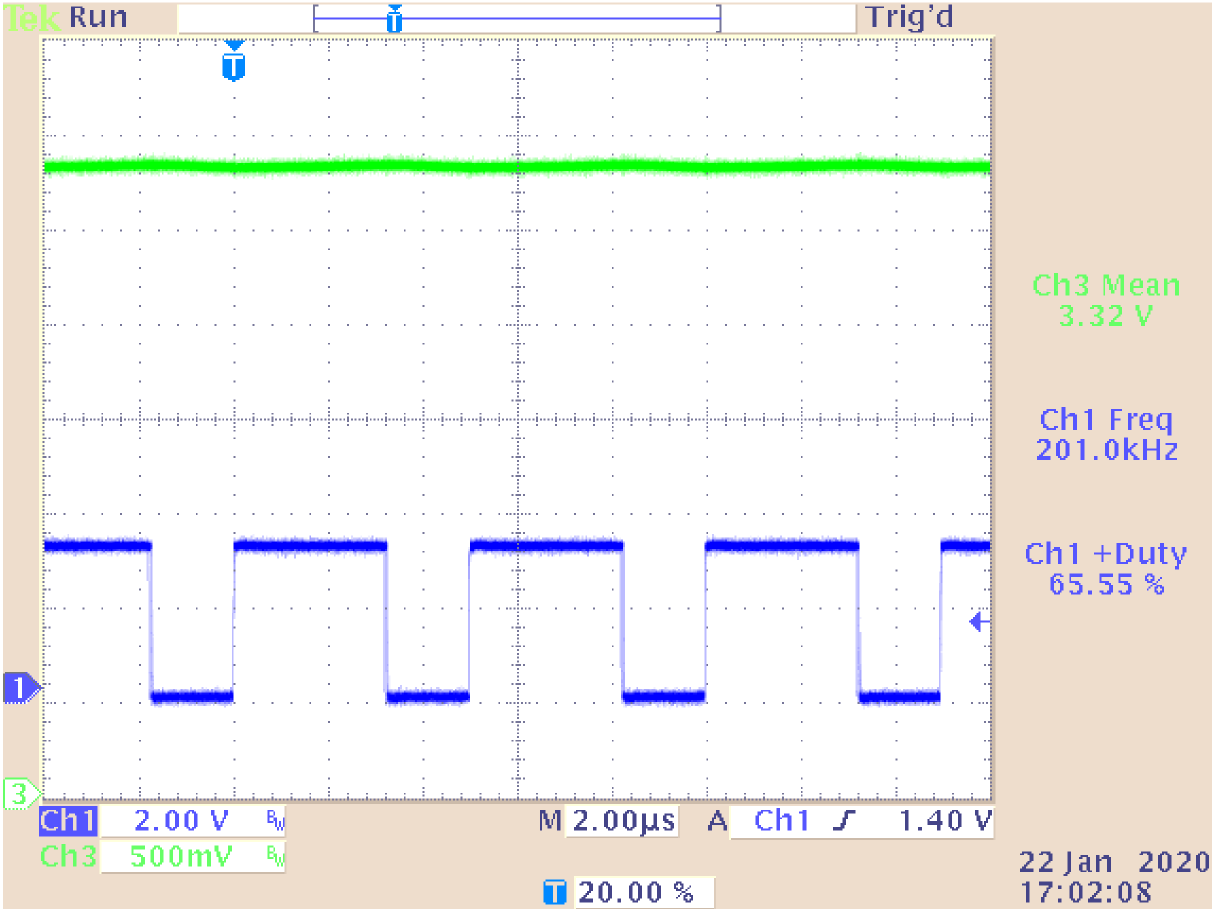
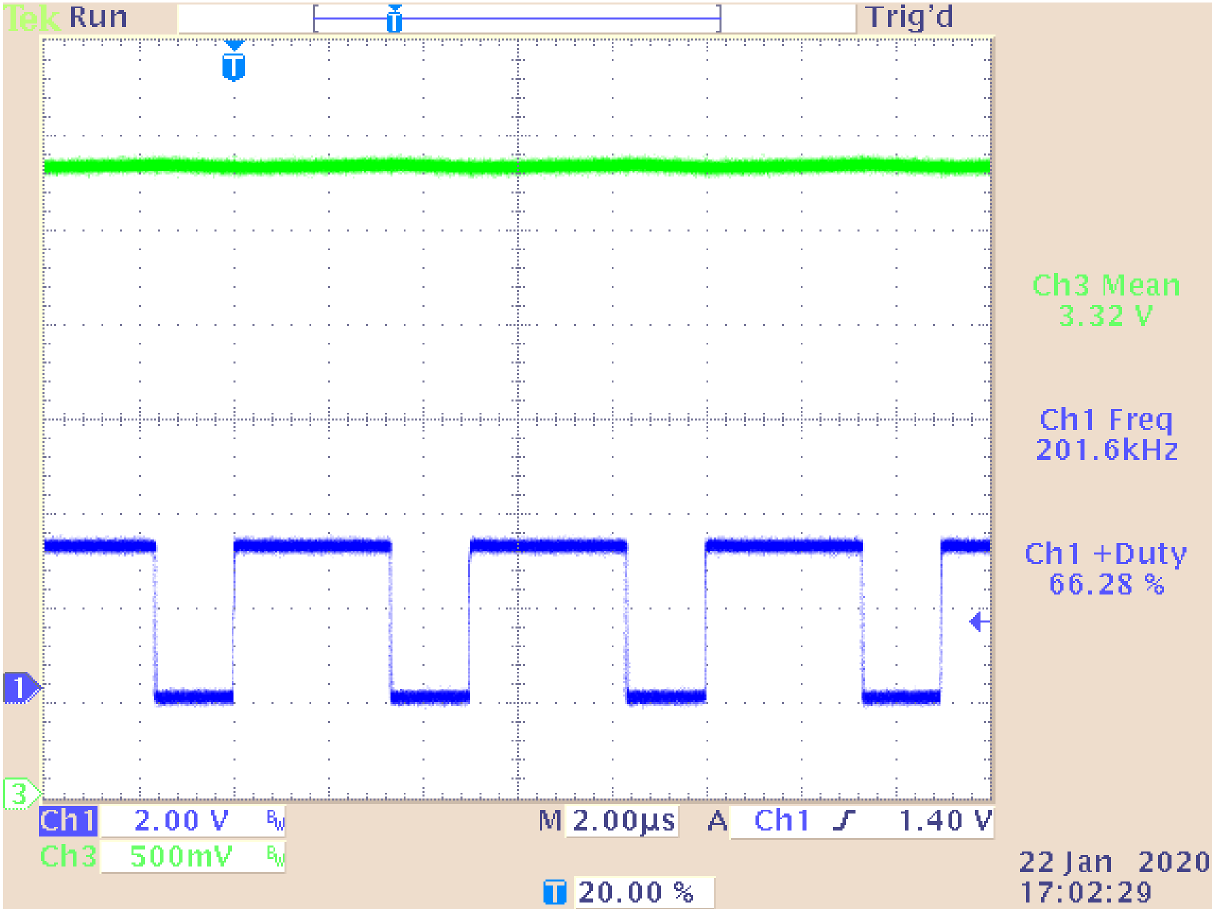


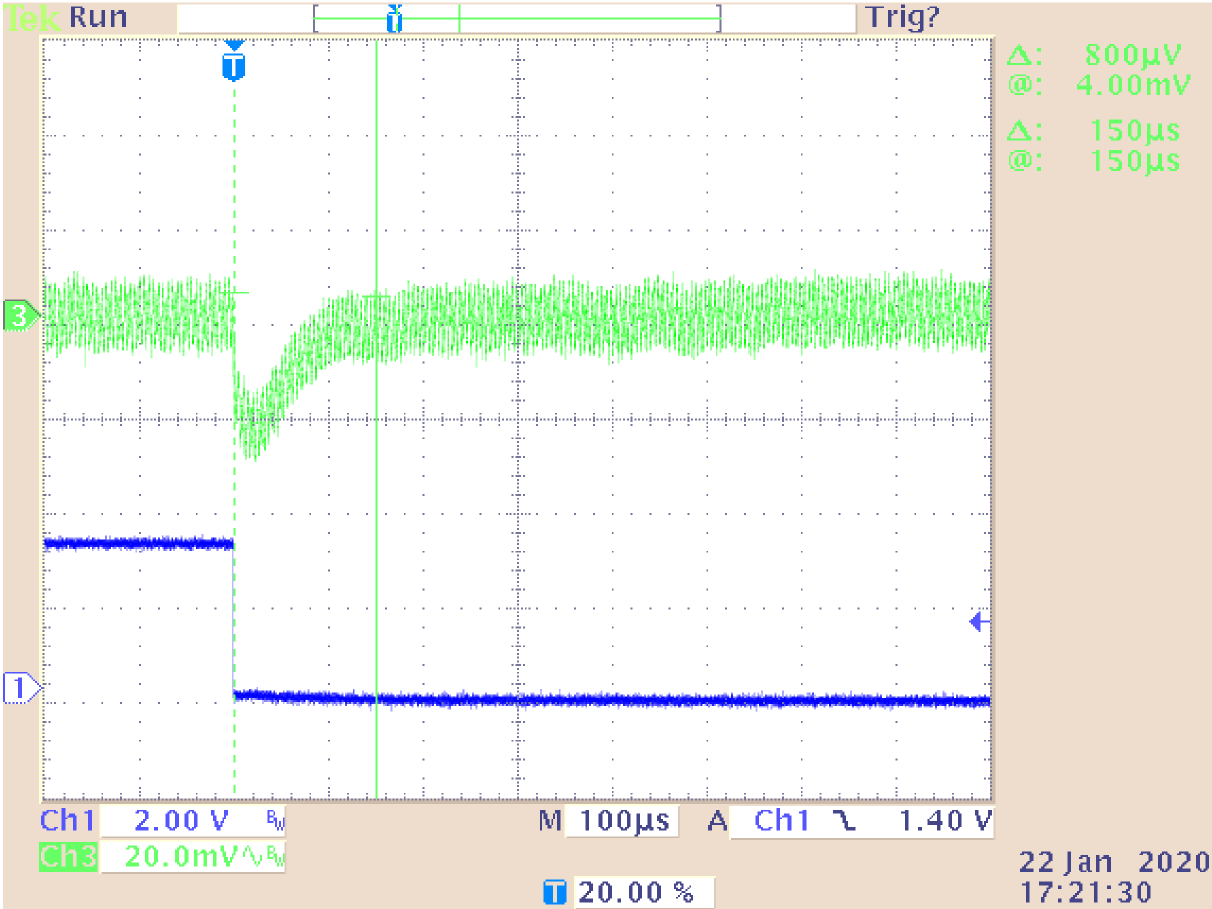
Figure 86. Output voltage (Ch3) and PWM (Ch1) at 100% load



## 9.2 Transient response tests

As discussed in Section 8: Getting started, the transient response can provide useful information about the stability of the closed-loop system. The transient response of the step-down 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 8: Getting started.

**Figure 87. Output voltage (Ch3) transient from load change 50% to 100% (Ch1)**

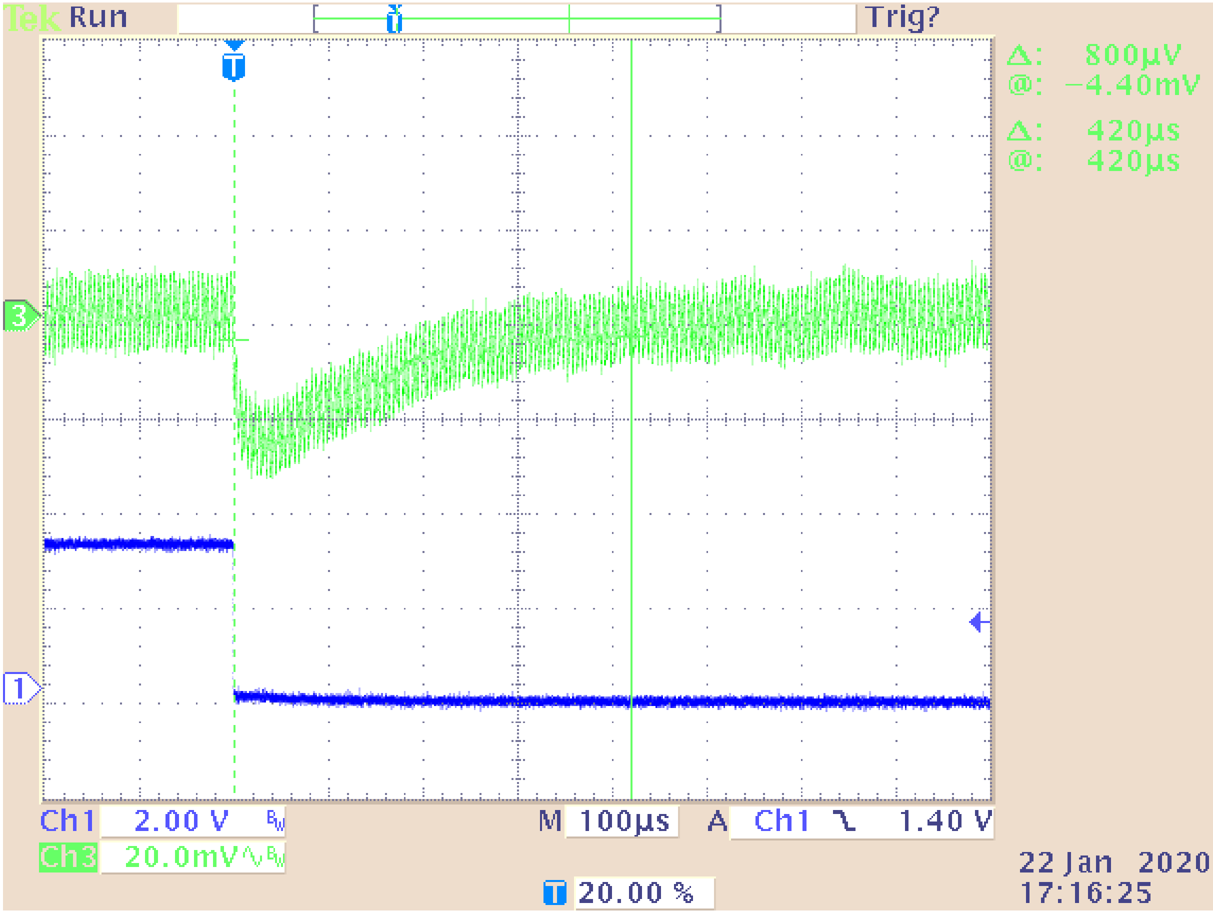



*Note: For this figure, output voltage undershoot is 30 mV and settling time is 150 µs.*

Figure 87 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 30 mV and recovers back to the steady state within 150 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 8: Getting started.

It is prudent to check the transient response across a range of line and load conditions. In Figure 88 the transient response is shown for a step-change in load from 0% to 50%. The dynamics of the system are different at light load and therefore the response is significantly slower. In this instance, the recovery takes 420 µs. However, the recovery still shows no sign of oscillation and therefore indicates that the system is stable.

**Figure 88. Output voltage (Ch3) transient from load change 0% to 50% (Ch1)**




*Note: For this figure, output voltage undershoot = 30 mV, settling time = 420 µs*

## 9.3 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 step-down converter is given in Section 8: Getting started.

**Figure 89. Measured loop response of the digital voltage mode loop at 100% load**

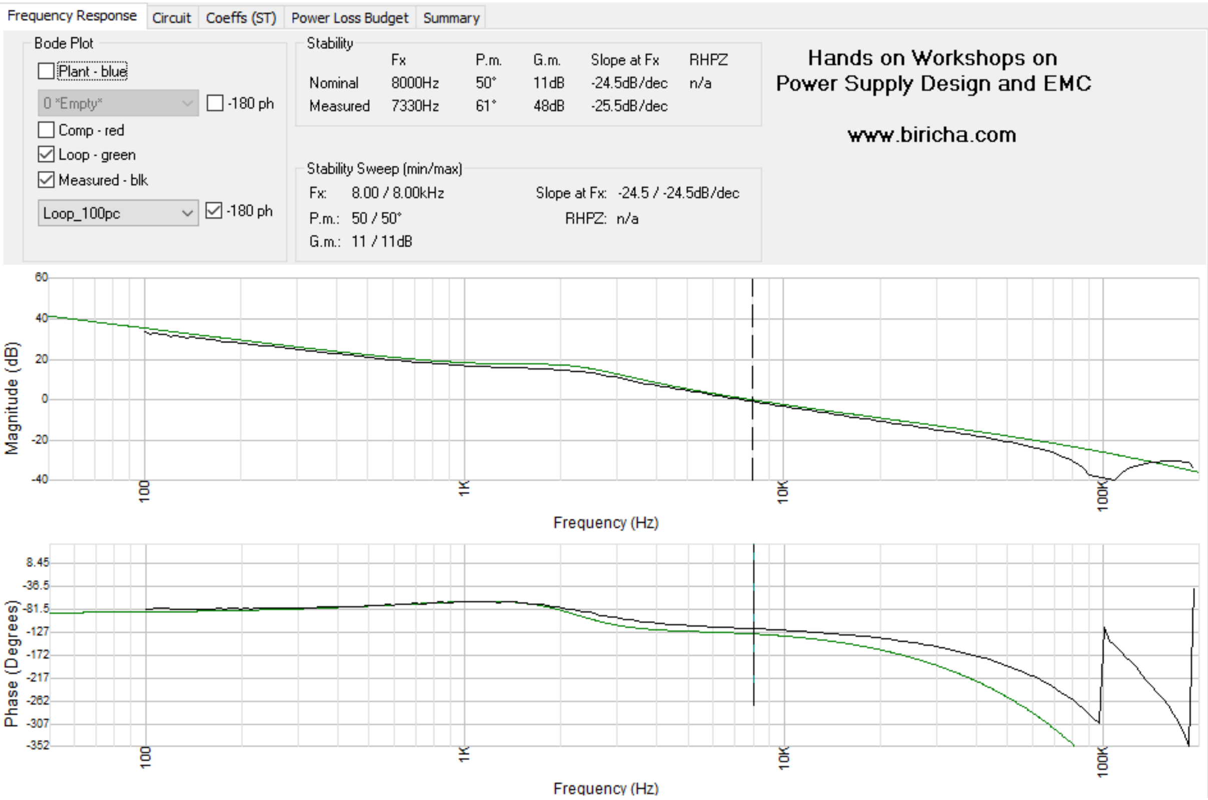



Figure 89 shows the measured control-to-output loop response of the step-down converter at 100% load. The measurement of the loop is plotted as the black trace and compared with the simulated loop in green. A crossover frequency of 7.33 kHz is measured which is close to the desired crossover frequency of 8 kHz. A phase margin of 61° 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 that is not included in the plant model. This has some effect on the Q factor of the complex conjugate pole and therefore the rate at which the phase transitions.

In Figure 90 the loop response of the step-down converter at 50% load is measured. As expected, the overall loop response is very similar to that of the system at 100% load however there is some change in the damping of the double pole as the load changes. Therefore, the crossover frequency and phase margin change slightly. The system is still stable.

With no load, the system dynamics change considerably. The system takes longer to respond to any changes and this is observed in the transient load step test. The frequency response at 0% load is shown in Figure 91 and exhibits a much lower crossover frequency of 5.4 kHz however a higher phase margin of 92°. The system is therefore still stable as observed during the transient response test.

**Figure 90. Measured loop response of the digital voltage mode loop at 50% load**

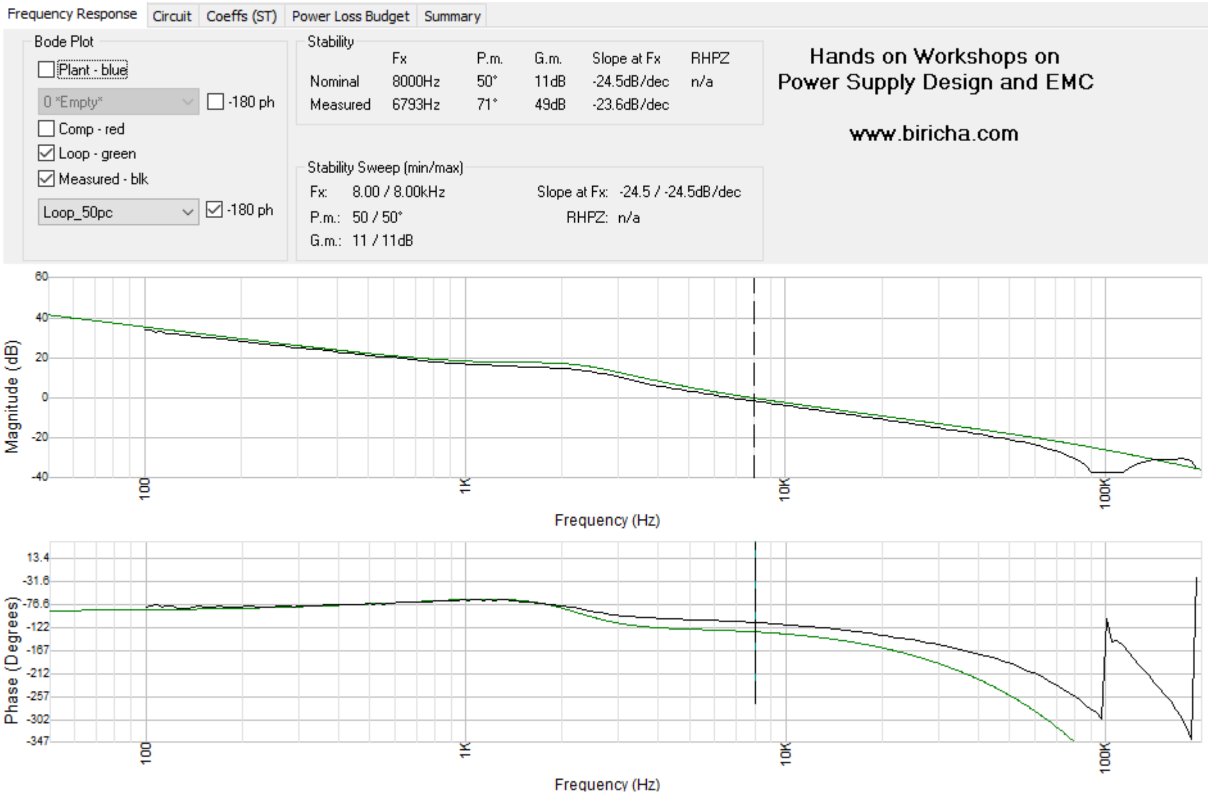
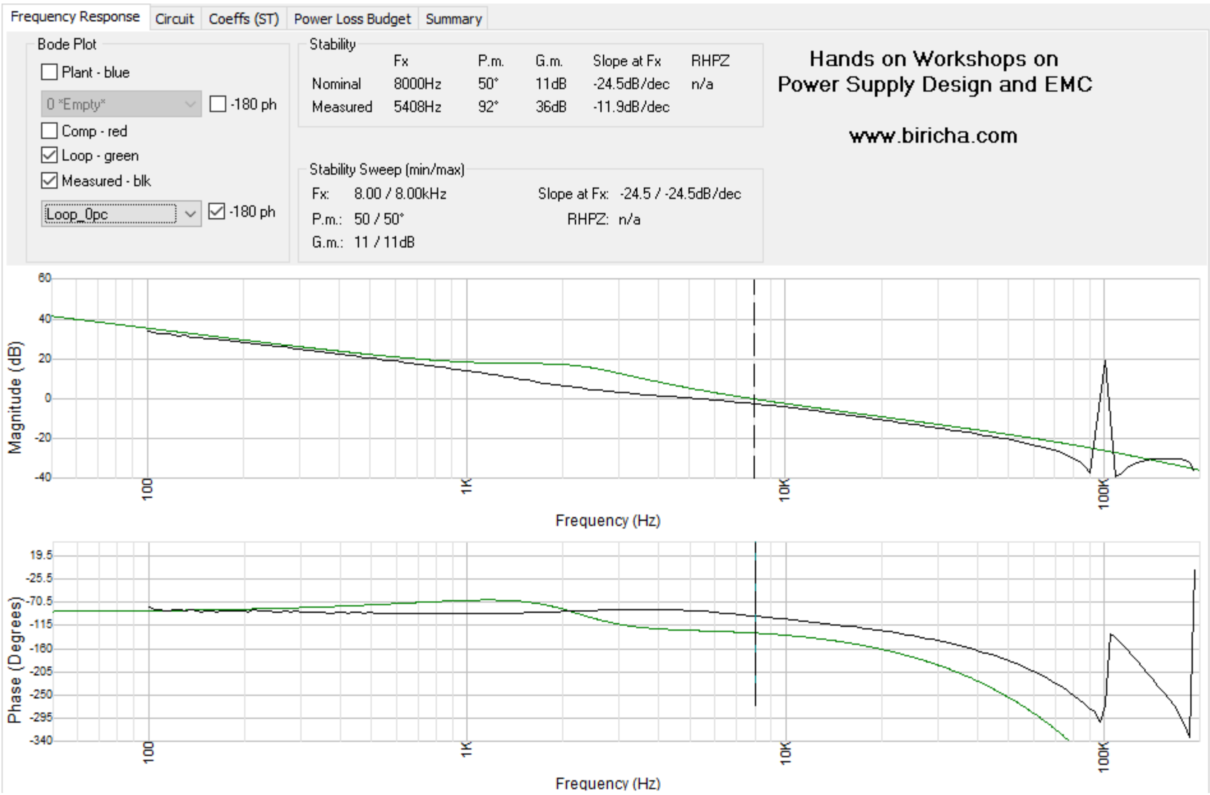



**Figure 91. Measured loop response of the digital voltage mode loop at 0% load**



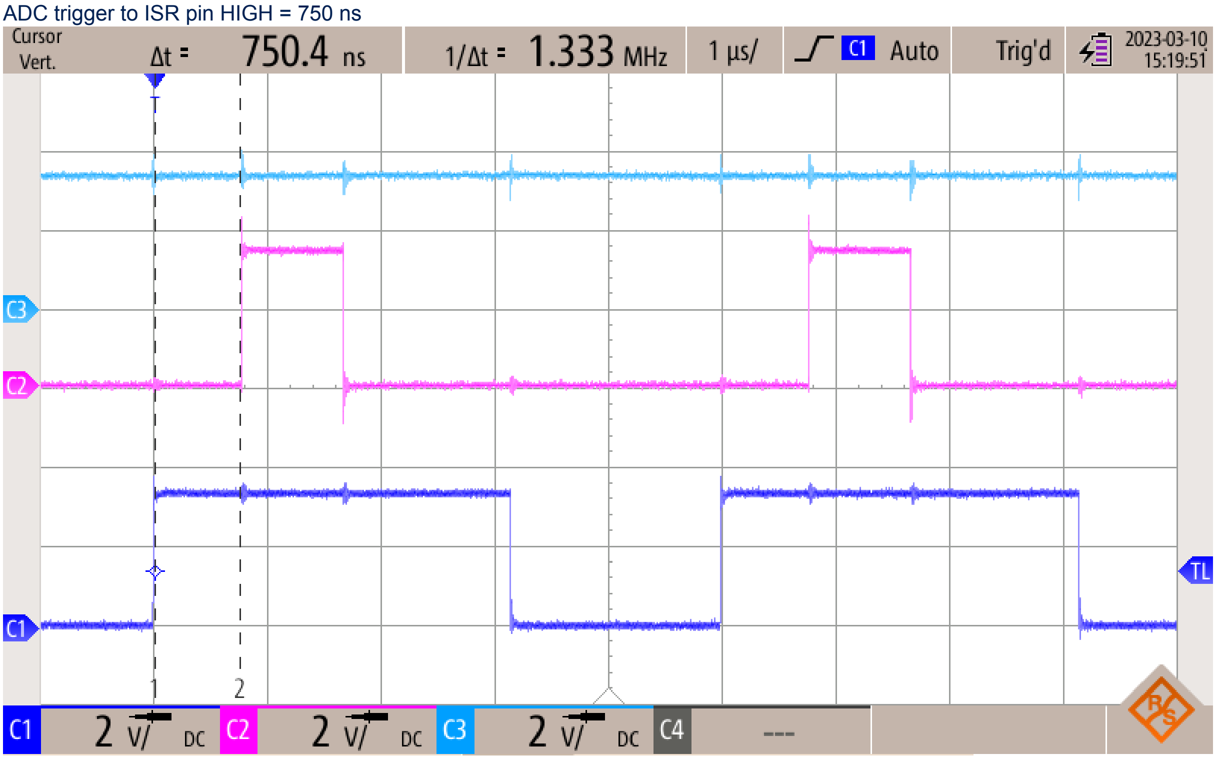
## 9.4 ISR plots (featuring FMAC and CPU load benefits)

The controller for this step-down 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 bandwidths. 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 92 the time from the ADC trigger to the FMAC ISR interrupt is measured. The ADC is triggered when the compare unit 3 match occurs shortly after the rising edge of the PWM signal, Channel 1, in Figure 92. 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 92. The total time for this sample, conversion, and calculation process is measured as 750 ns.

**Figure 92. PWM (Ch1) and FMAC ISR duration (Ch2)**

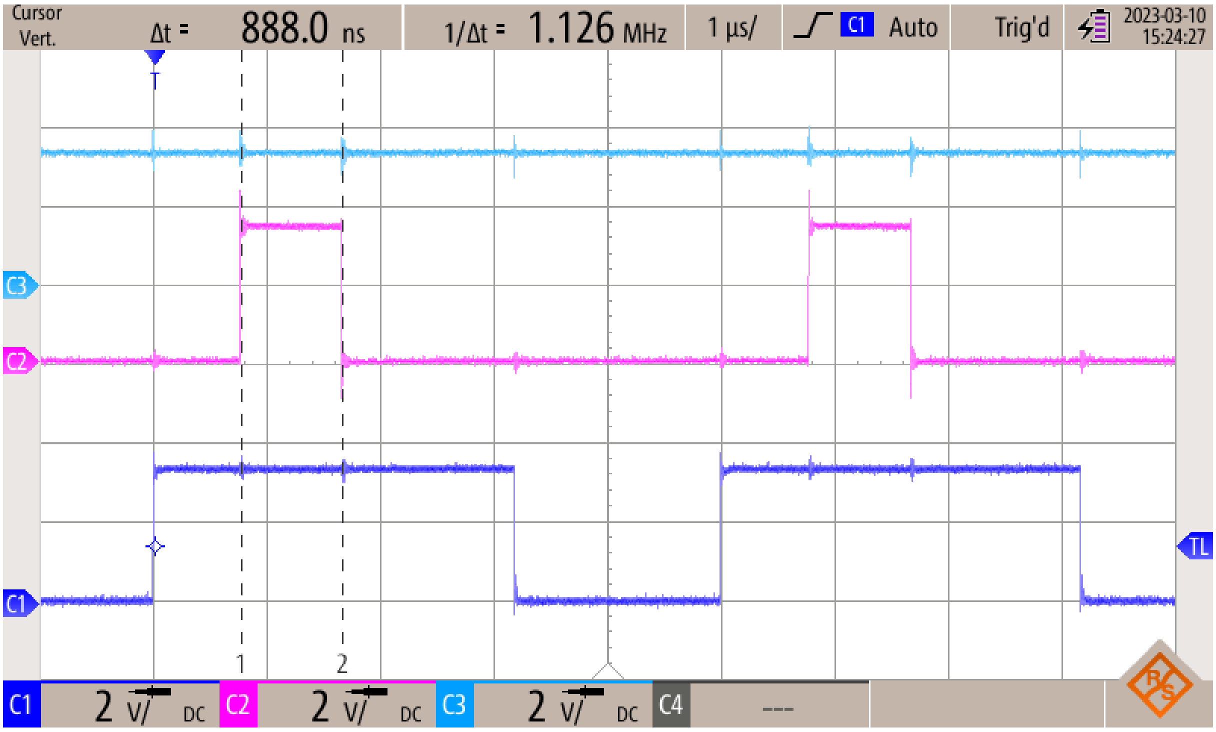
ADC trigger to ISR pin HIGH = 750 ns



The FMAC ISR is solely used for bounds checking of the controller output to a minimum and maximum value and then the update of the counter compare module register to set the new value of the duty cycle. Therefore, this is a very brief ISR containing only a few lines of code. The duration of the ISR is measured in Figure 93 as 888 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 93. PWM (Ch1) and FMAC ISR duration (Ch2)**

ISR duration = 888 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 94 as a GPIO is again set HIGH as the ISR is entered. This time is measured as 600 ns.

Within the ADC ISR, the 3p3z controller is executed. This means that the duration of the ISR is significantly longer than the FMAC ISR. The ADC ISR containing the 3p3z controller executes within 1.850 ns as measured in Figure 95. Therefore, the total time from the ADC trigger to the PWM update is 2.450 µs. This compares to 1.638 ns when the controller is implemented using the FMAC. However, the main core is now occupied for an additional 812 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 94. PWM (Ch1) and ADC ISR duration (Ch2)**

ADC trigger to ISR pin HIGH = 600 ns

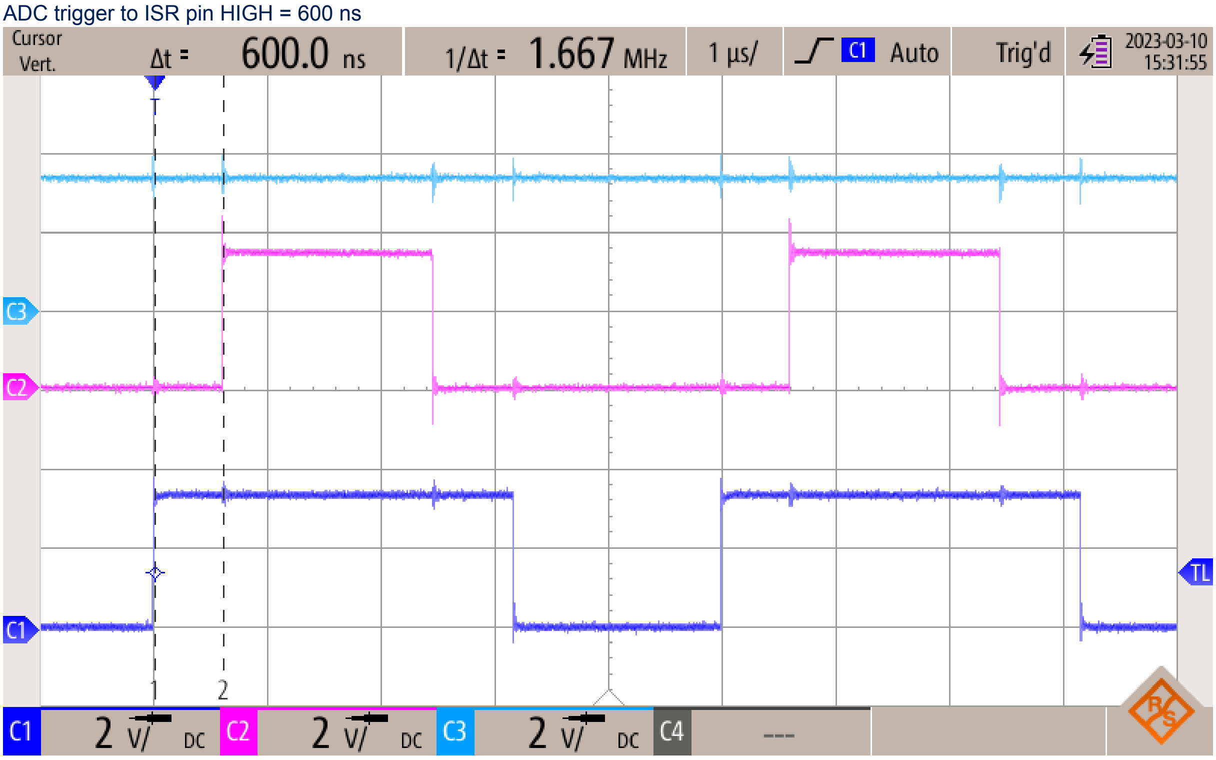
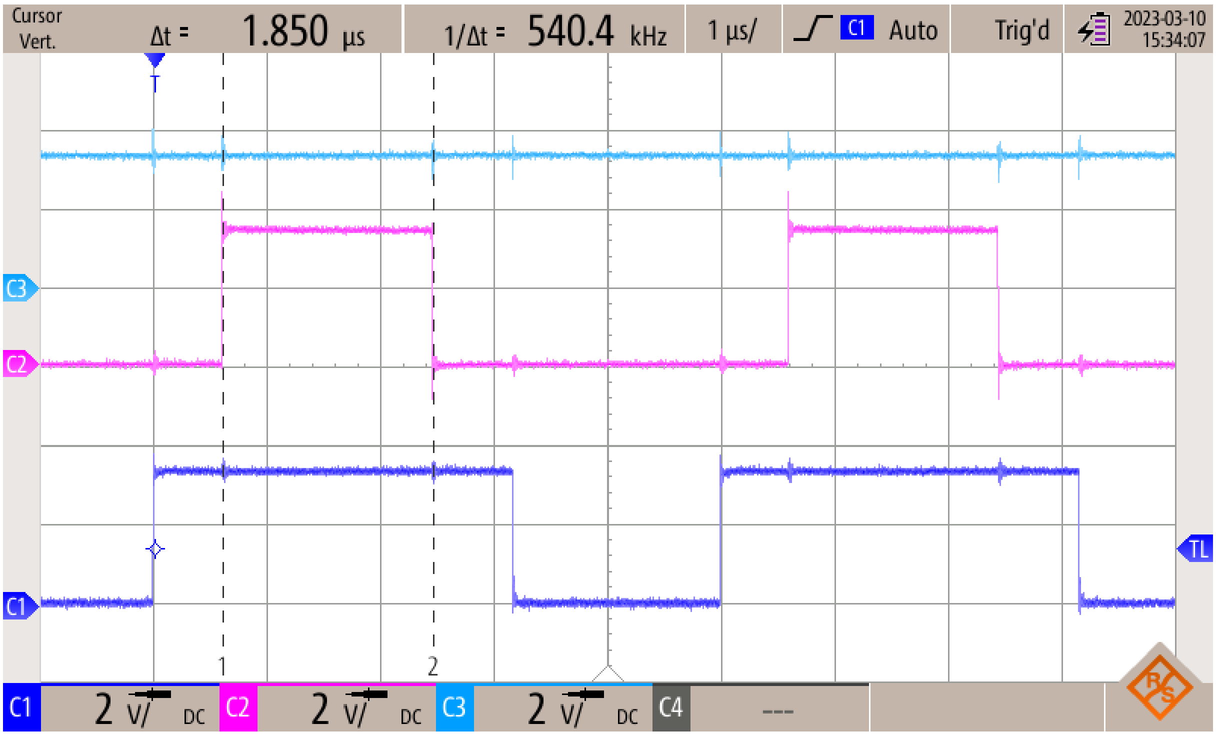


**Figure 95. PWM (Ch1) and ADC ISR duration (Ch2)**

ISR duration = 1.850 ns



# 10 Summary

This application note provides a detailed explanation of the design and operation of the digitally controlled voltage mode step-down converter on the discovery kit. The controller is implemented using the FMAC on the STM32G4 MCU from STMicroelectronics.

This application note shows that the voltage mode step-down converter can be stabilized by applying control theory. The goal is to achieve a high crossover frequency and phase margin with an ideal transient response. The document provides all the equations necessary to calculate the digital controller coefficients. Additionally, the ST-WDS software tool from Biricha can be used to perform the same calculations with ease and 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*

# 11 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. |
| 7-Dec-2022 | 2 | Added *Section 8.8 Buck converter usage using the X-CUBE-DPower pack*. |
| 6-Jan-2023 | 3 | Updated Introduction about the buck voltage mode usage with X-CUBE-DPOWER. |
| 29-Feb-2024 | 4 | Updated:<br>• Section 3.2: Step-by-step control loop<br>• Section 6.2.4: HRTIM configuration<br>• Section 6.5: 3p3z controller coefficients<br>• Section 6.6: ST-WDS configuration<br>• Section 8.1: Overall usage<br>• Section 8.2: Loading the project<br>• Section 8.4: Source files<br>• Section 8.5: Open-loop operation<br>• Section 8.7: Waveform display in IAR Systems®<br>• Section 8.8.1: Configure the converter parameters using the GUI<br>• Section 8.8.3: Topology dedicated files<br>• Section 9.4: ISR plots (featuring FMAC and CPU load benefits)<br>• Figure 34. Removing pin from the project<br>• Figure 35. ADC parameter settings<br>• Figure 39. Timer C configuration tab-part 2<br>• Figure 40. Timer C configuration tab - part 3<br>• Figure 72. Falling edge dead-time, Ch1: high-side FET, Ch2: low-side FET, dead-time measured as 147 ns<br>• Figure 82. X-CUBE-DPower – GUI parameters for the step-down converter<br>• Figure 92. PWM (Ch1) and FMAC ISR duration (Ch2)<br>• Figure 93. PWM (Ch1) and FMAC ISR duration (Ch2)<br>• Figure 94. PWM (Ch1) and ADC ISR duration (Ch2)<br>• Figure 95. PWM (Ch1) and ADC ISR duration (Ch2) |

# Contents

# List of tables

# List of figures