

Digital Power Supply from Scratch in 60 Minutes



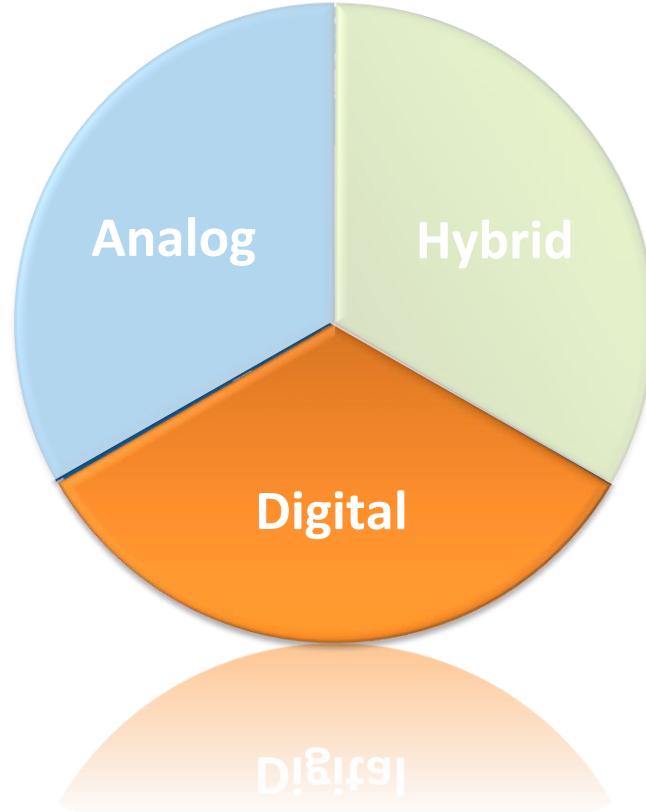
A Leading Provider of Smart, Connected and Secure Embedded Control Solutions



Presented by Andy Reiter
15th November 2022

The World of Power Control

Standard Analog Controllers
Integrated solutions for dedicated topologies, power levels and applications



Programmable Digitally Enhanced Power Analog
Integrated solutions for various topologies, power levels and applications

Fully Digitally Controlled Systems
High-end control solutions for various topologies, power levels and applications providing highest flexibility

Agenda

Digital Power Supply from Scratch in 60 Minutes*

*: most probably not true ☺

- Overview Digital Power Starter Kit 3 (DPSK3)
- Requirements and Scope
- Hands-On
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB® PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start and Error Handling State Machine to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

Agenda

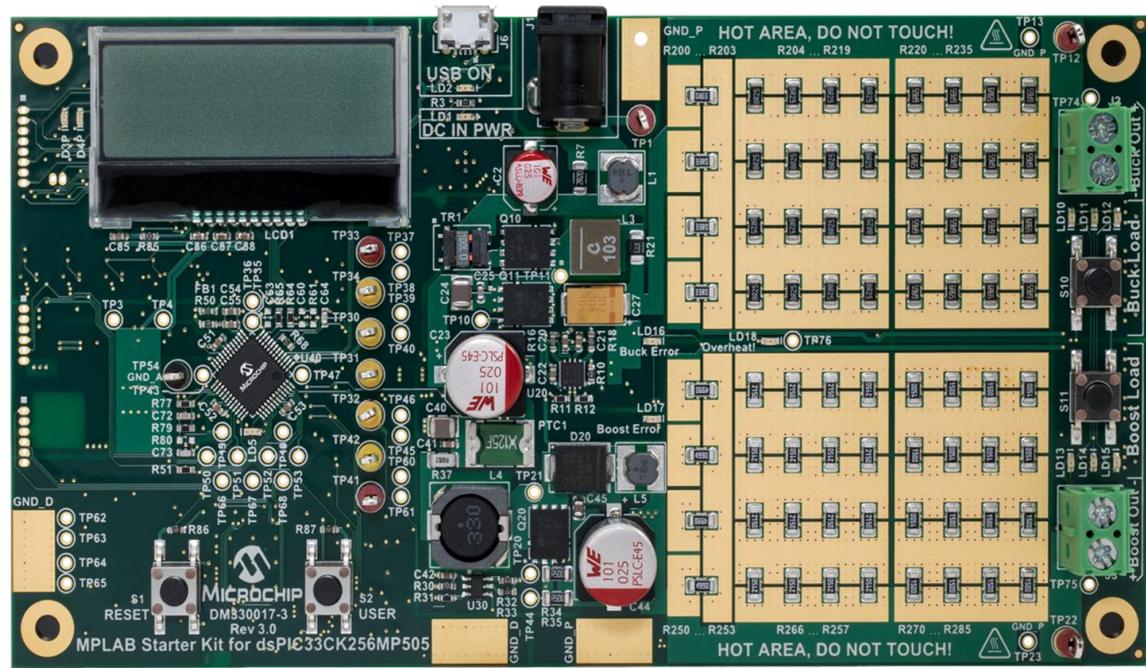
Digital Power Supply from Scratch in 60 Minutes*

*: no, honestly - it's a lie 😊

- Overview Digital Power Starter Kit 3 (DPSK3)
- Requirements and Scope
- Hands-On
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB® PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start and Error Handling State Machine to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

dsPIC33C Digital Power Starter Kit (DPSK3)

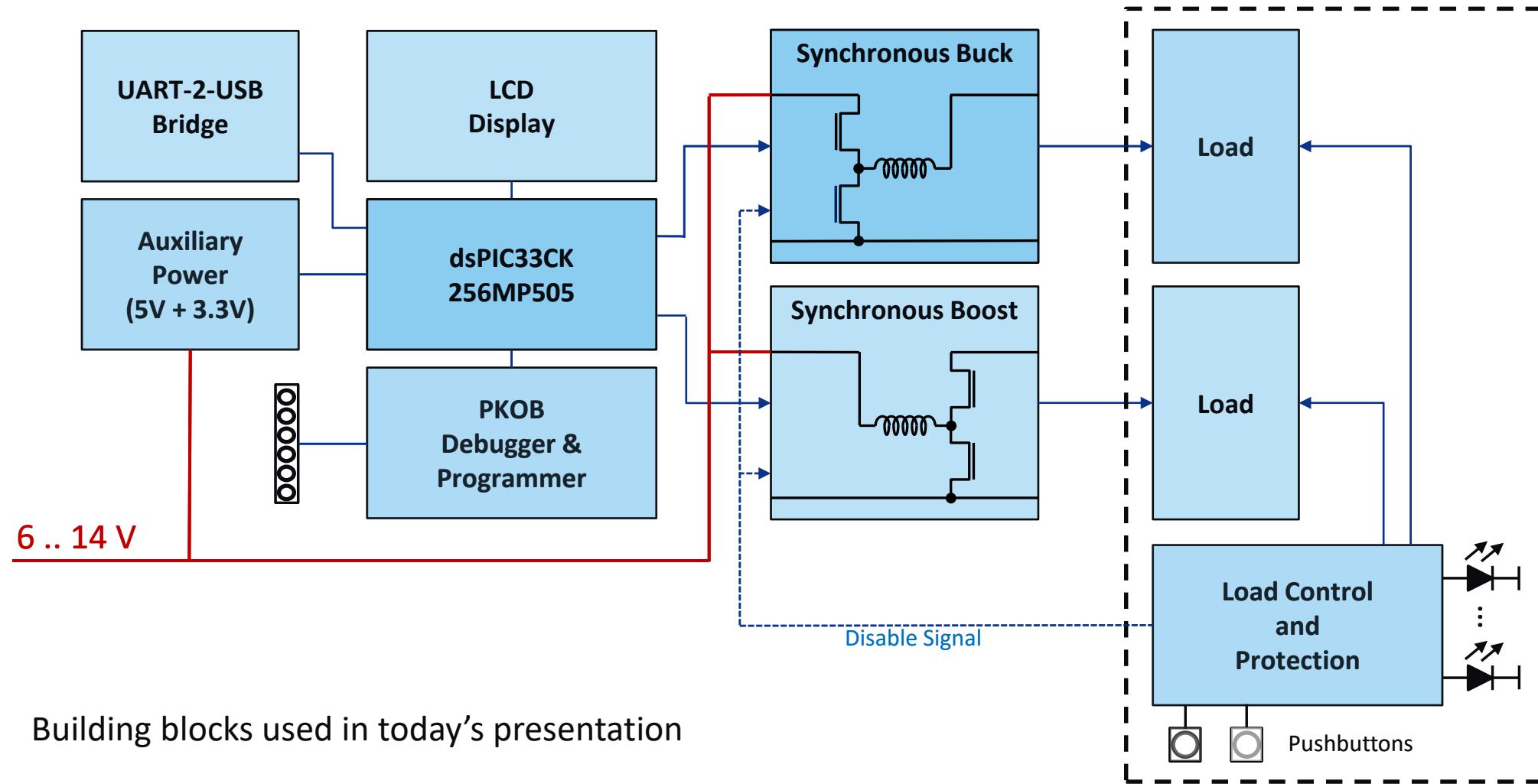
- On-board [dsPIC33CK256MP505](#) DSC
- PIC24F Auxiliary Microcontroller managing loads and protection circuit auto recovery
- Two Independent DC/DC Converter Topologies:
 - Synchronous Buck Converter
 - Asynchronous Boost Converter
- Independent resistive loads
 - Four selectable Constant Load Levels
 - Three Selectable Step Load Levels
- Protection circuitry
 - Over Current Protection (OCP)
 - Over Voltage Protection (OVP)
 - Over Temperature Protection (OTP)
- Development Features
 - [PKOB4 On-Board Programmer/Debugger](#)
 - LC Display User Interface
 - USB/UART Bridge (Standard VCP)

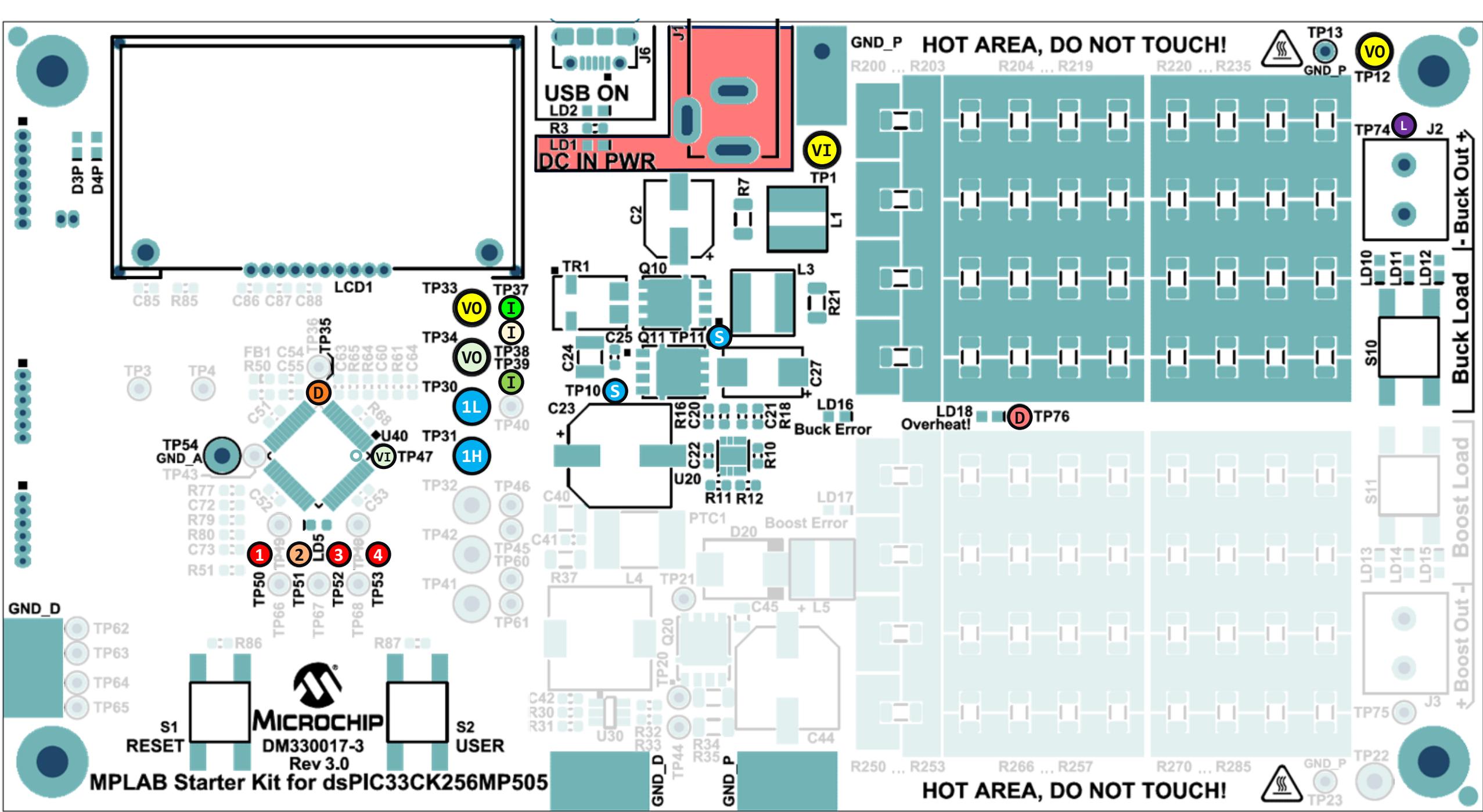


Available Now:

Part-No: DM330017-3

dsPIC33C Digital Power Starter Kit 3 (DPSK3)





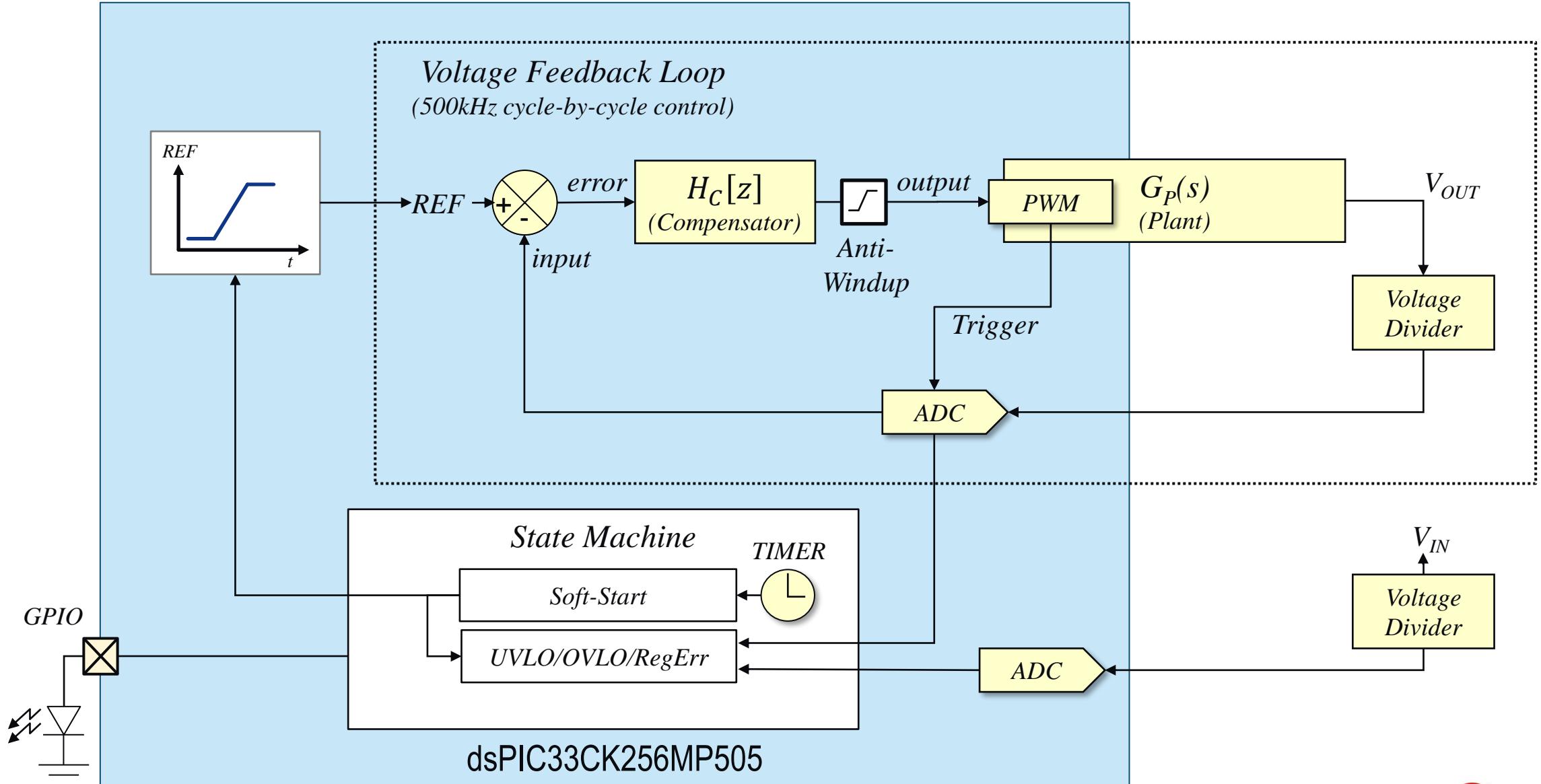
Agenda

Digital Power Supply from Scratch in 60 Minutes*

*: who had the idea to put this in the title?

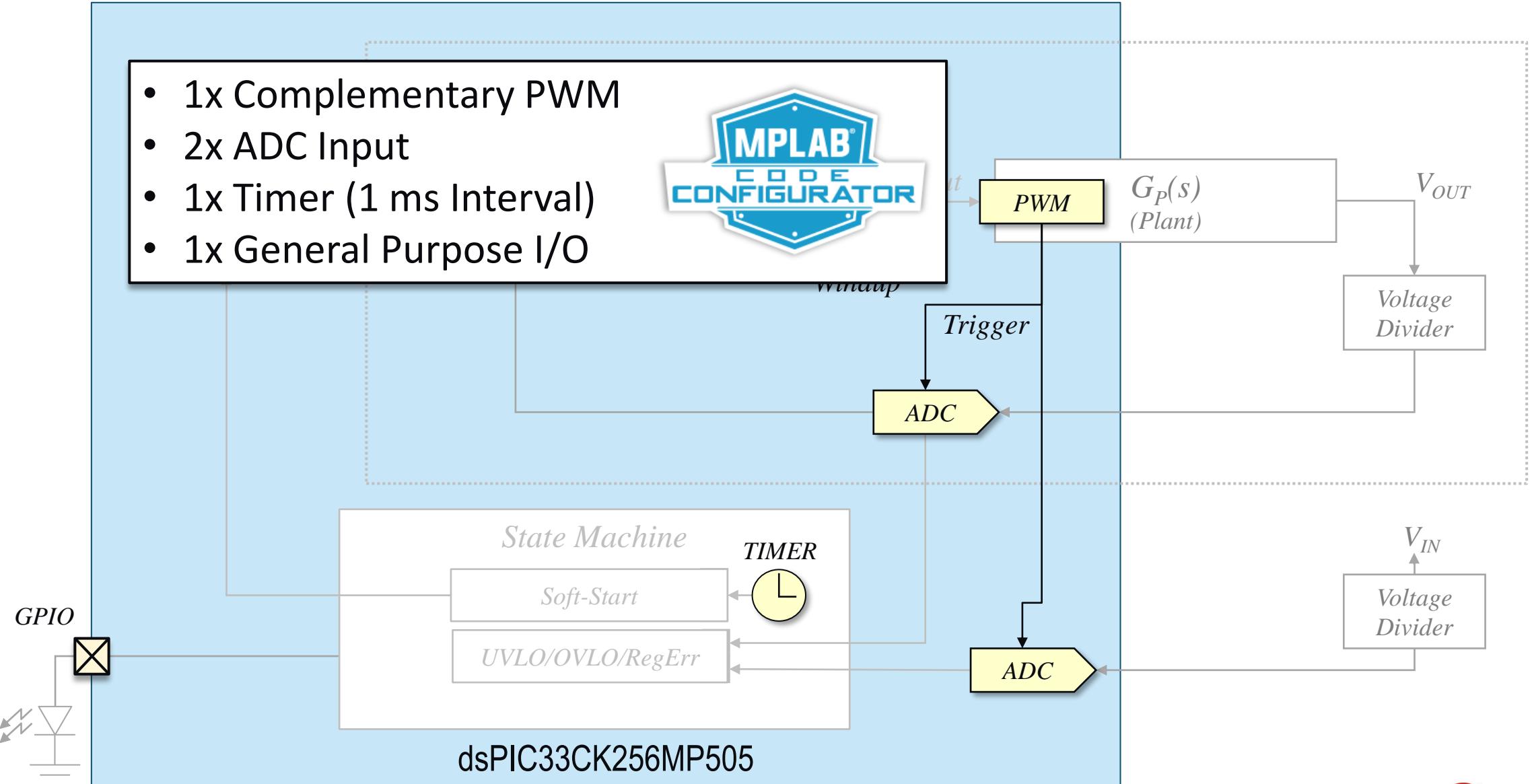
- Overview Digital Power Starter Kit 3 (DPSK3)
- Requirements and Scope
- Hands-On
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB® PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start and Error Handling State Machine to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

Requirements

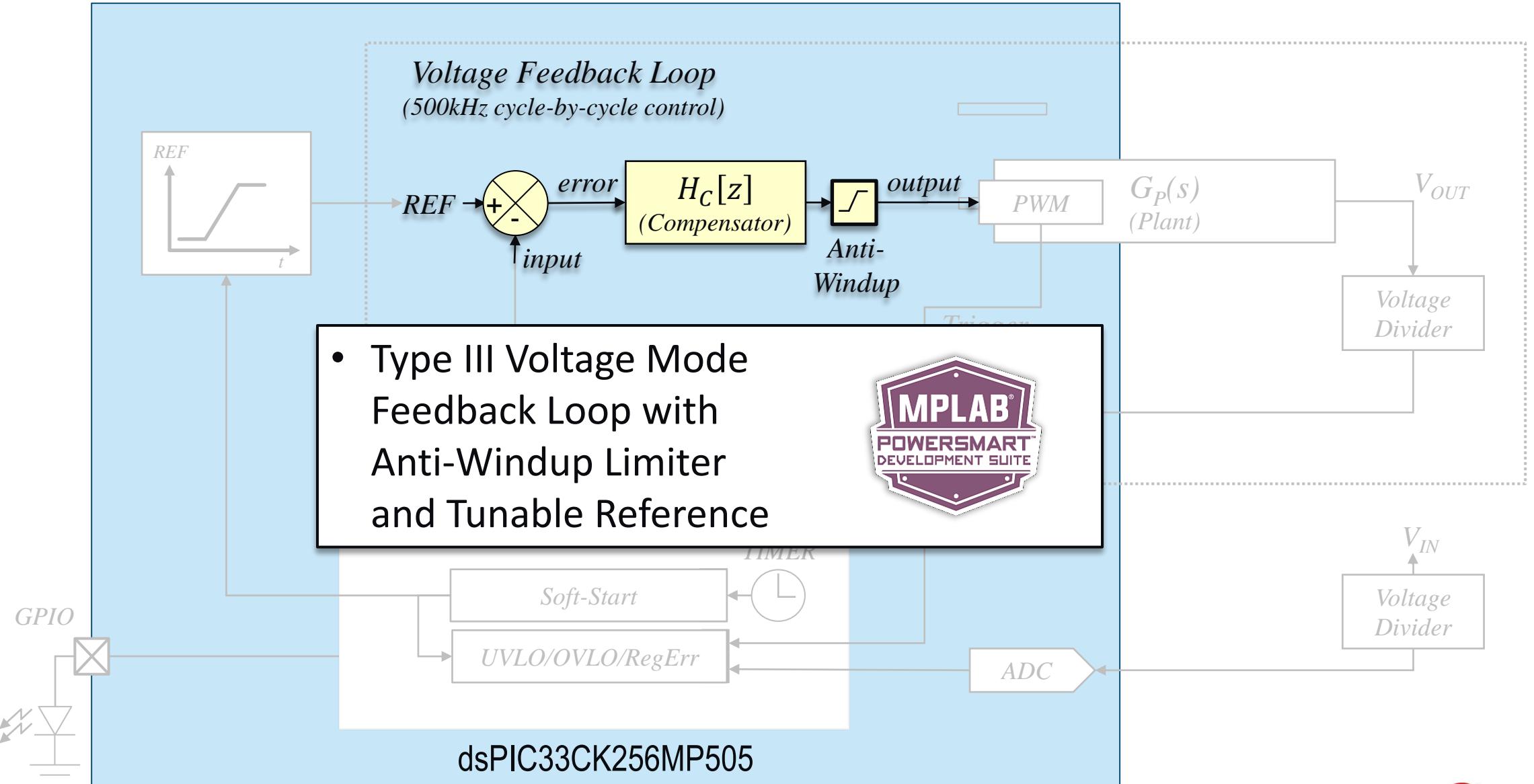


Requirements

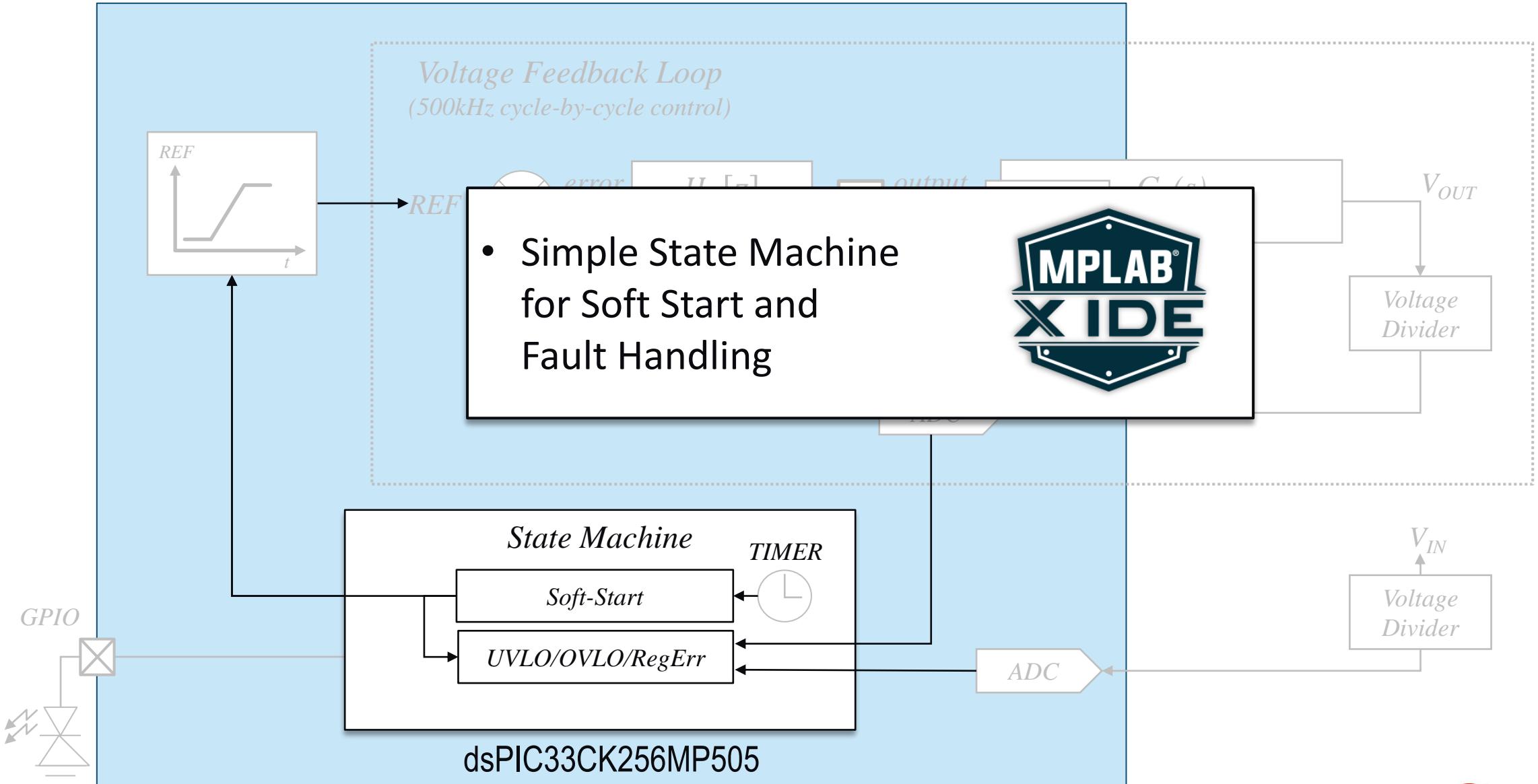
- 1x Complementary PWM
- 2x ADC Input
- 1x Timer (1 ms Interval)
- 1x General Purpose I/O



Requirements



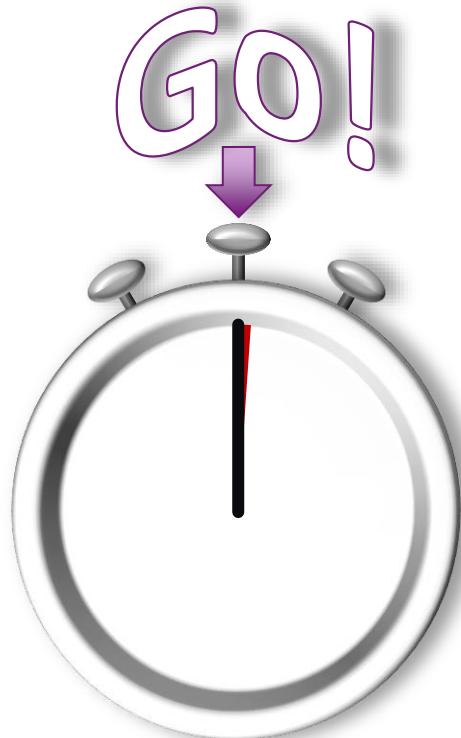
Requirements



This is from where we start...

Estimated Time to Output Voltage (ETOV): 1:00:00 *

*: Ah! Now you are starting to cheat!



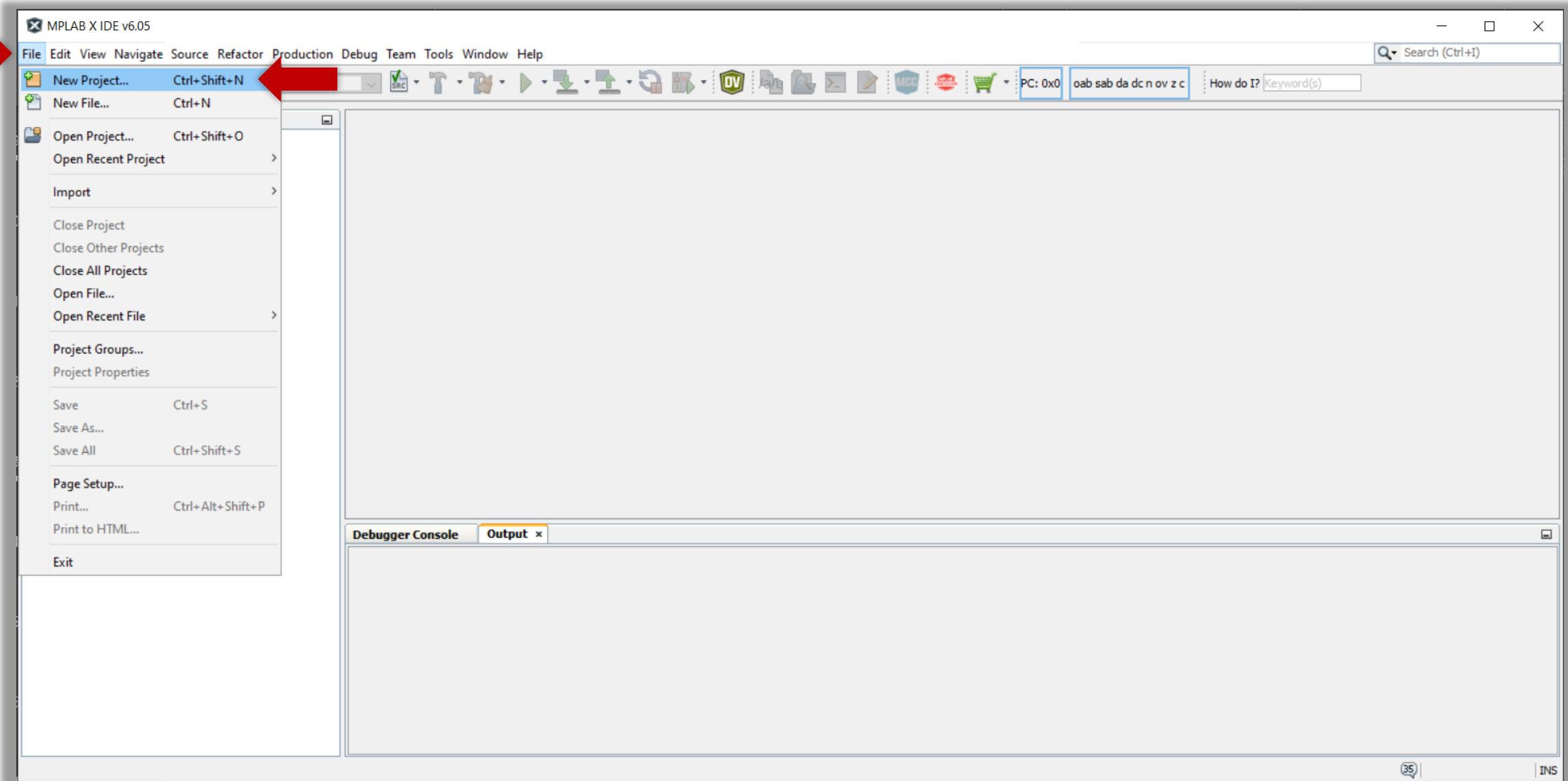
Agenda

Digital Power Supply from Scratch in 60 Minutes*

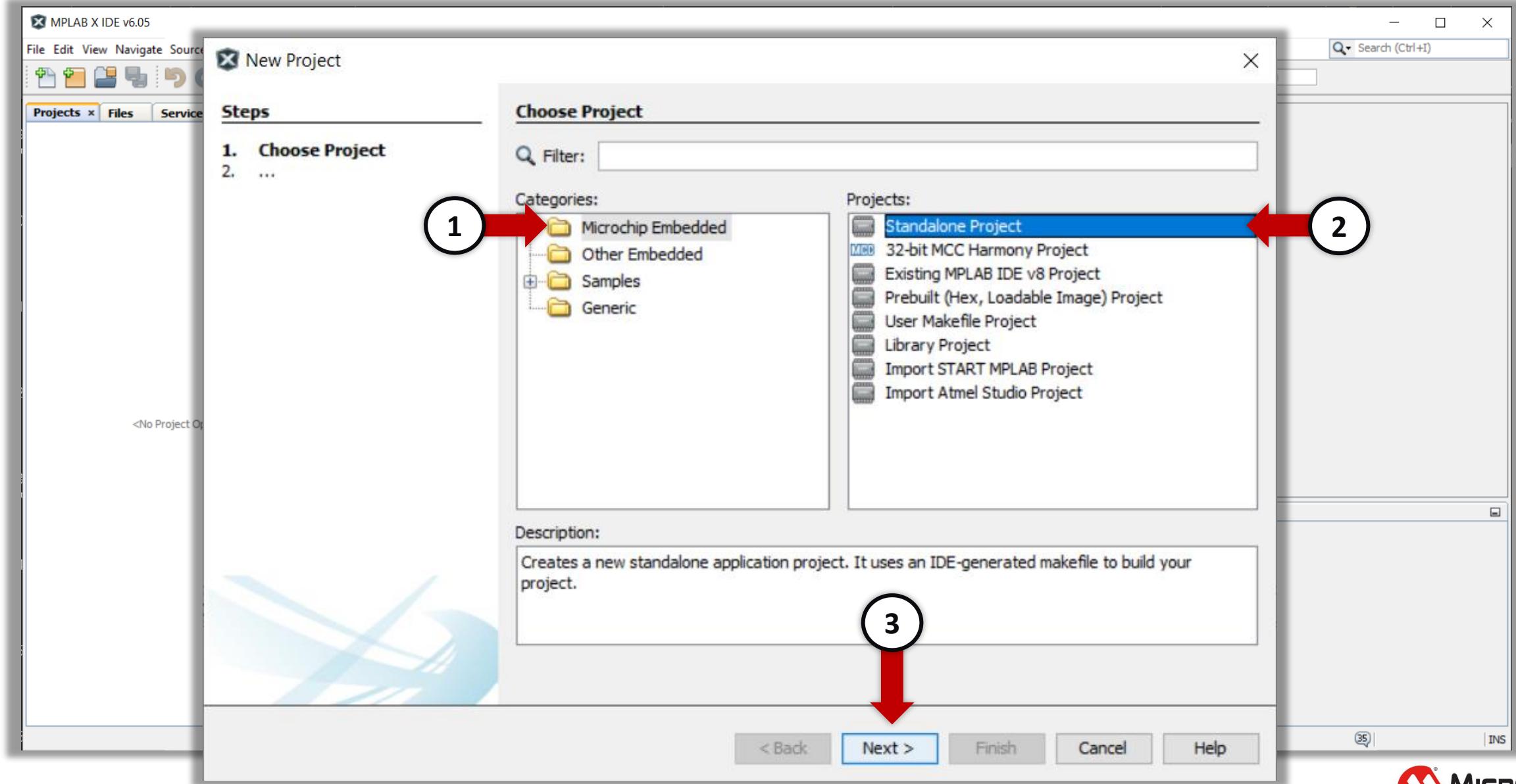
*: ok, now it's getting real....

- Overview Digital Power Starter Kit 3 (DPSK3)
- Requirements and Scope
- **Hands-On**
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB® PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start and Error Handling State Machine to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

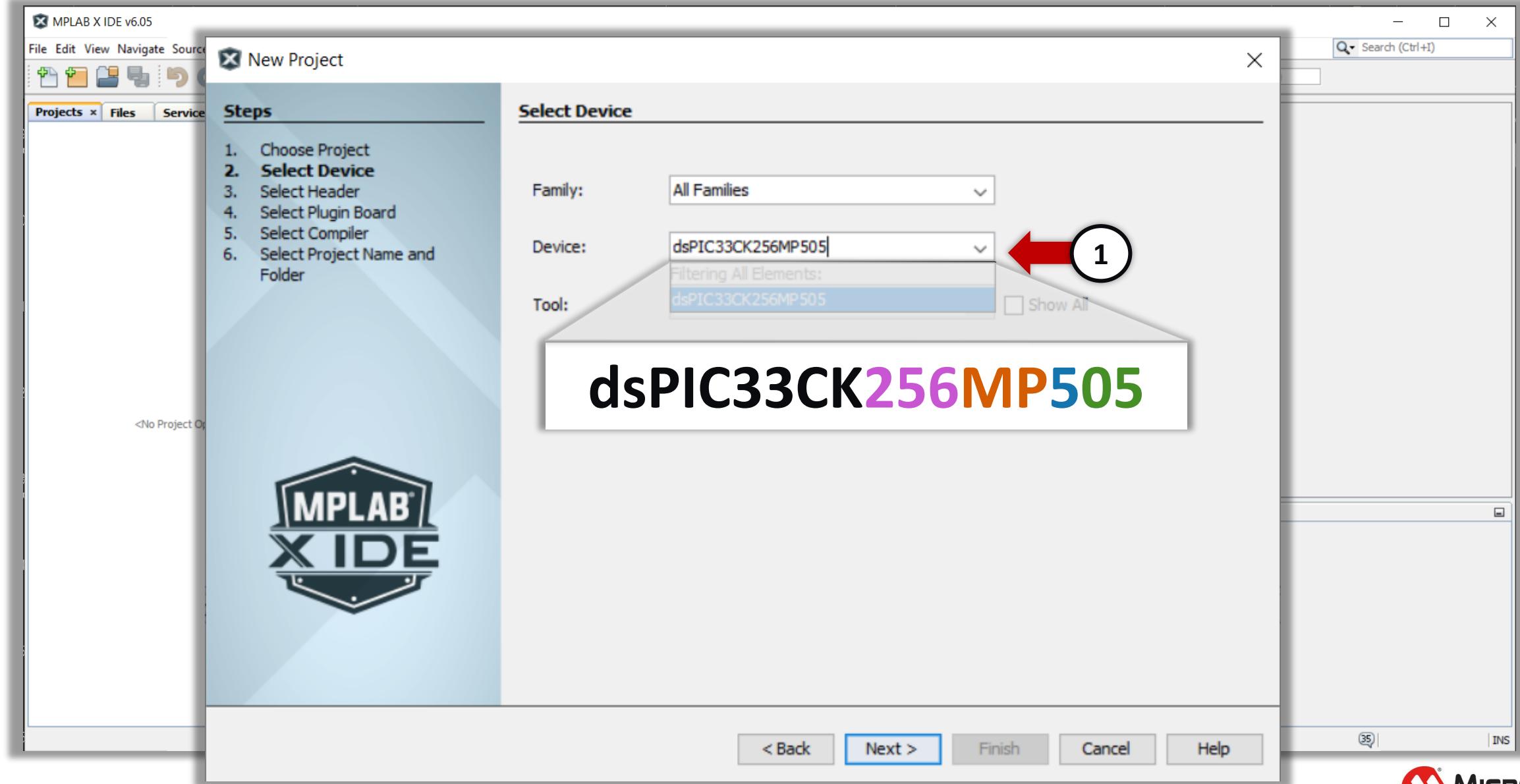
Creating a New Project



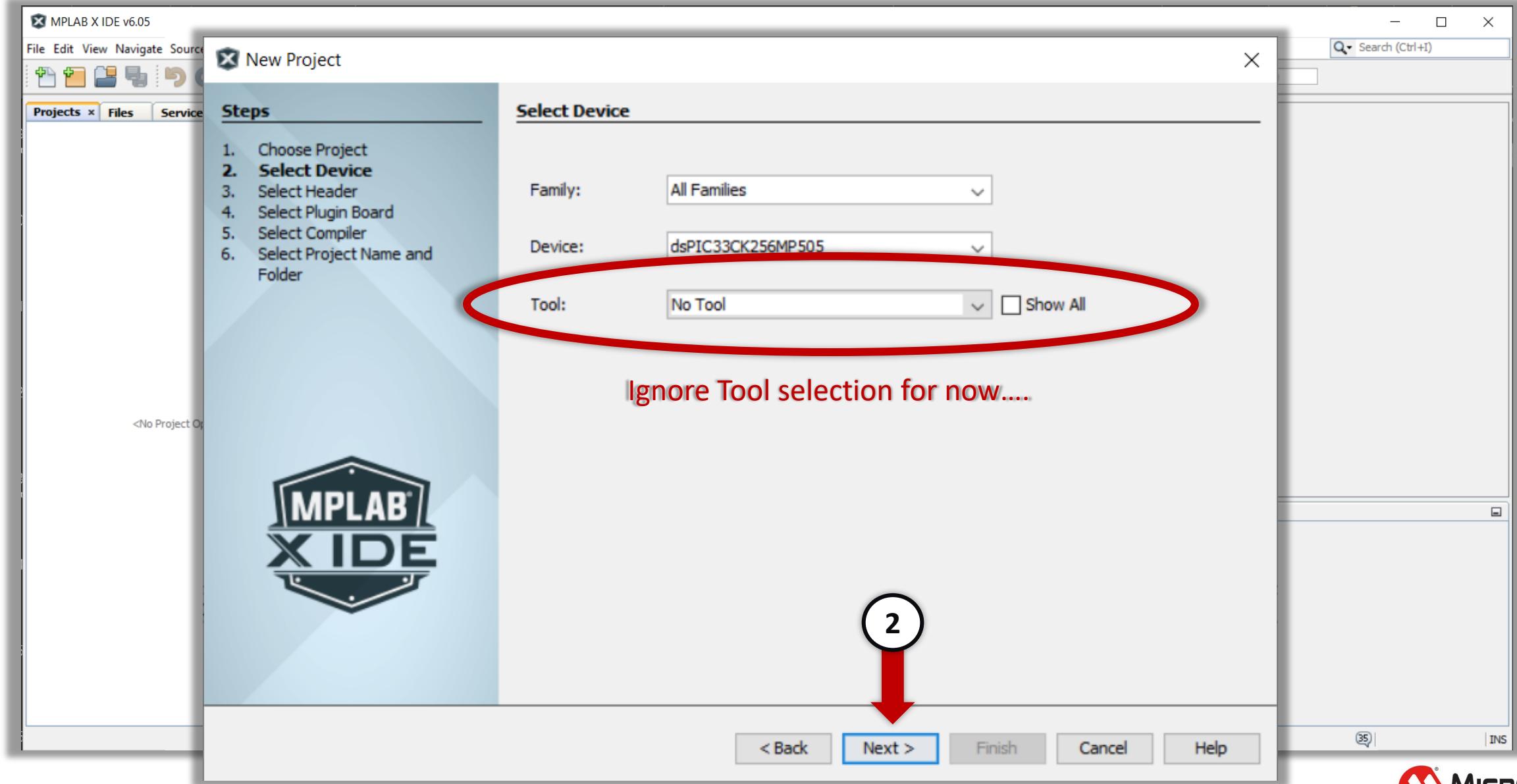
Create a Standalone Project



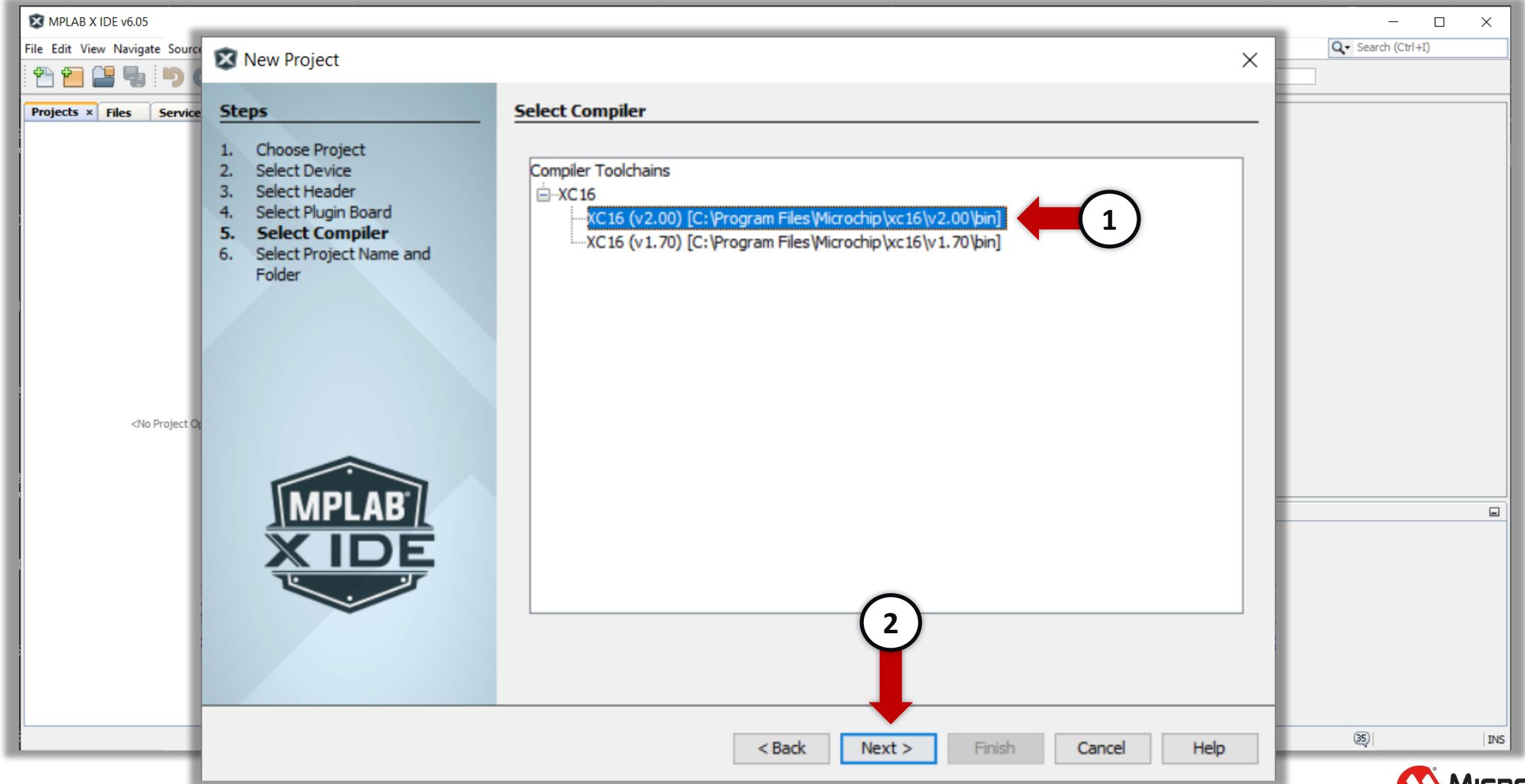
Select Target Device & Programming Tool



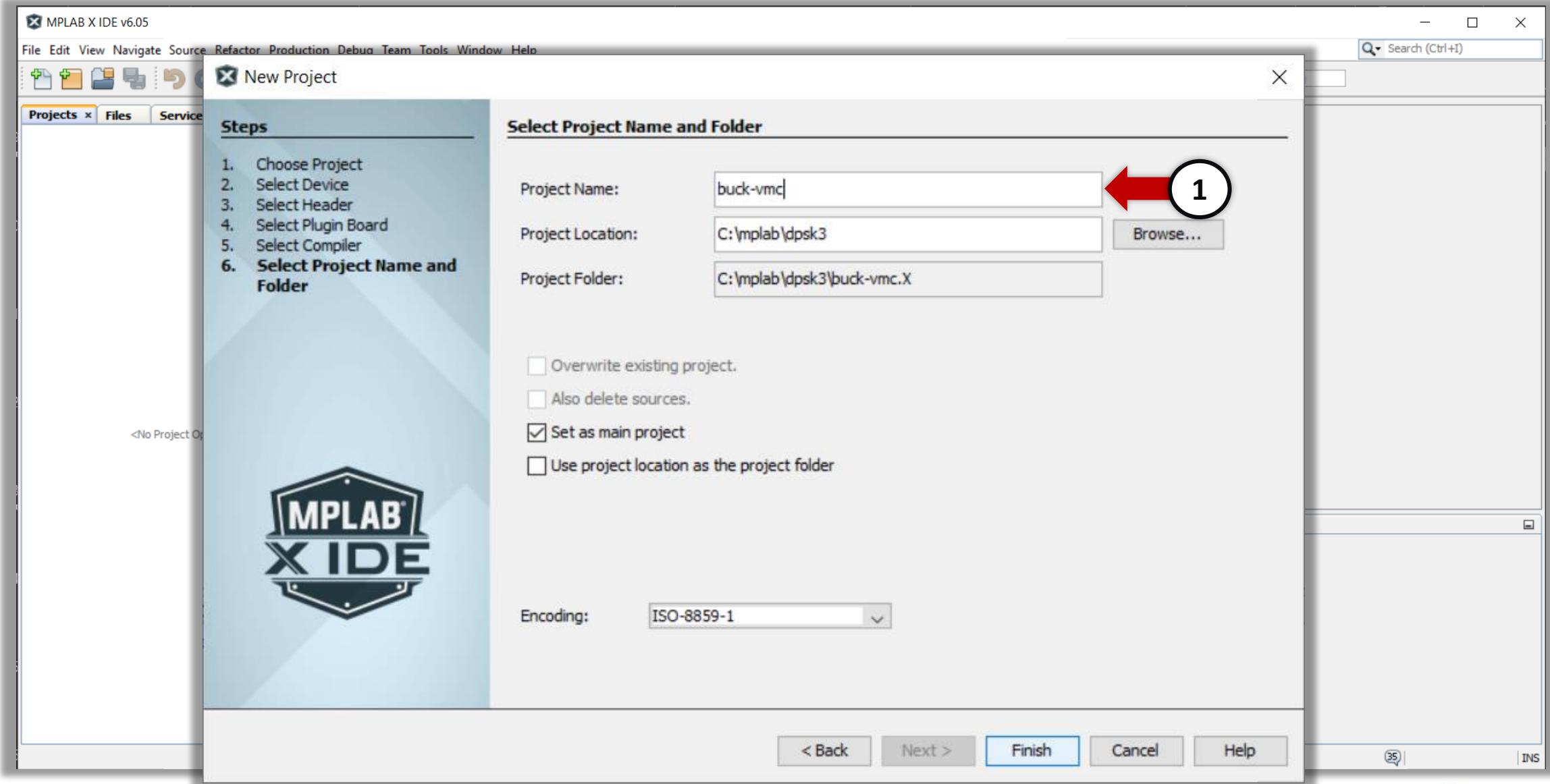
Select Target Device & Programming Tool



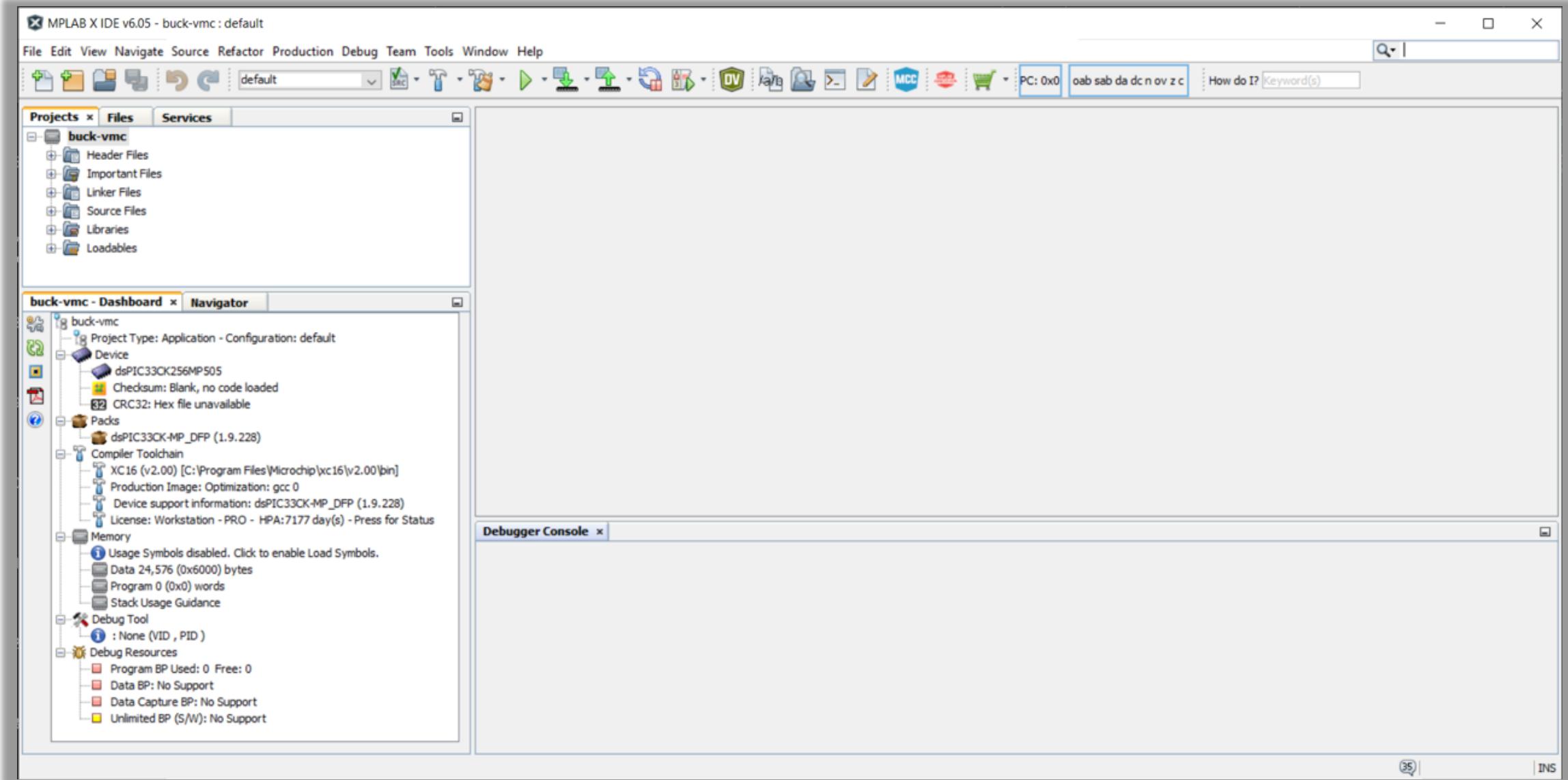
Select C Compiler



Select Project Name & Location



Empty Project Has Been Created!



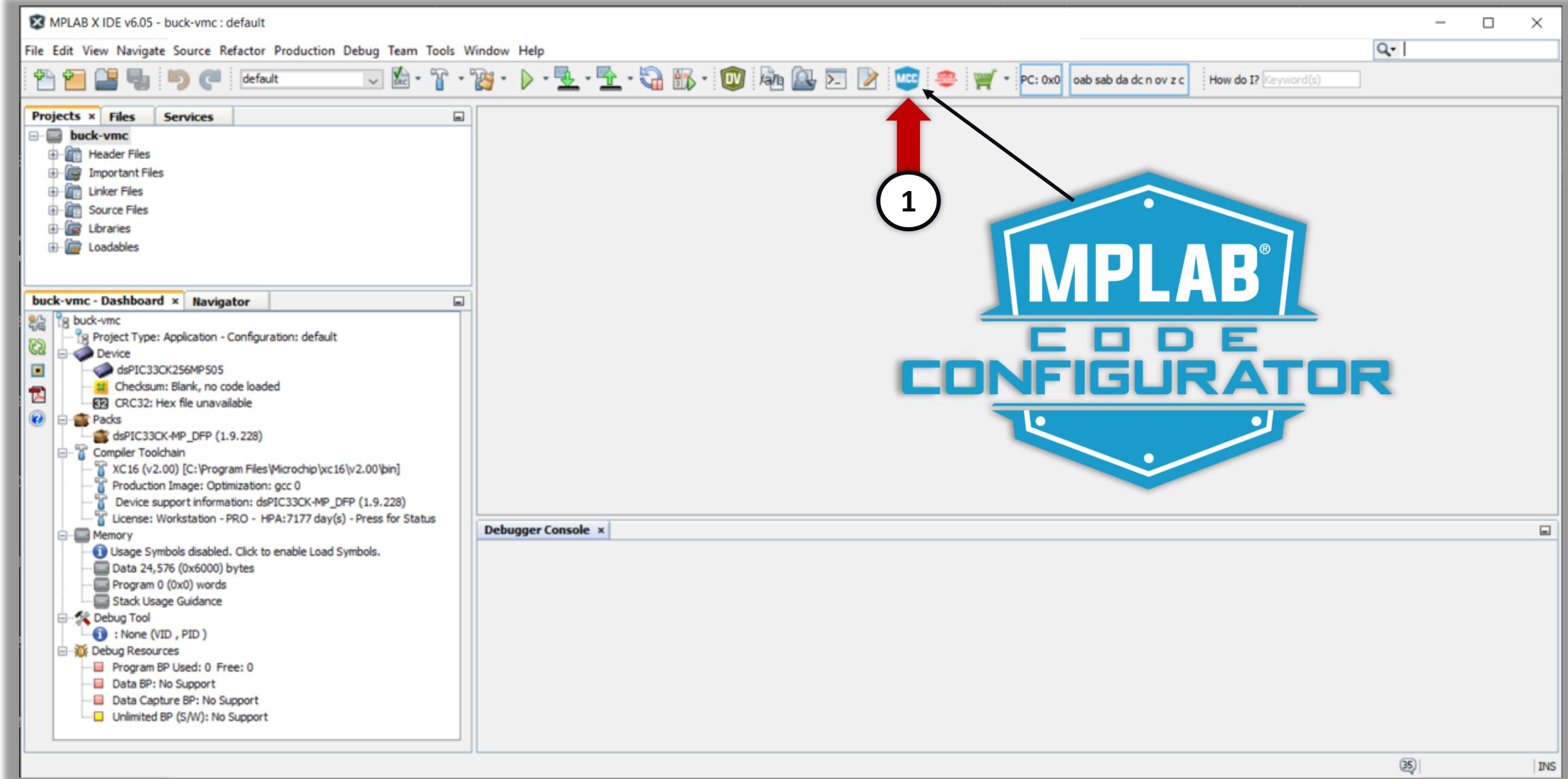
Agenda

Digital Power Supply from Scratch in 60 Minutes*

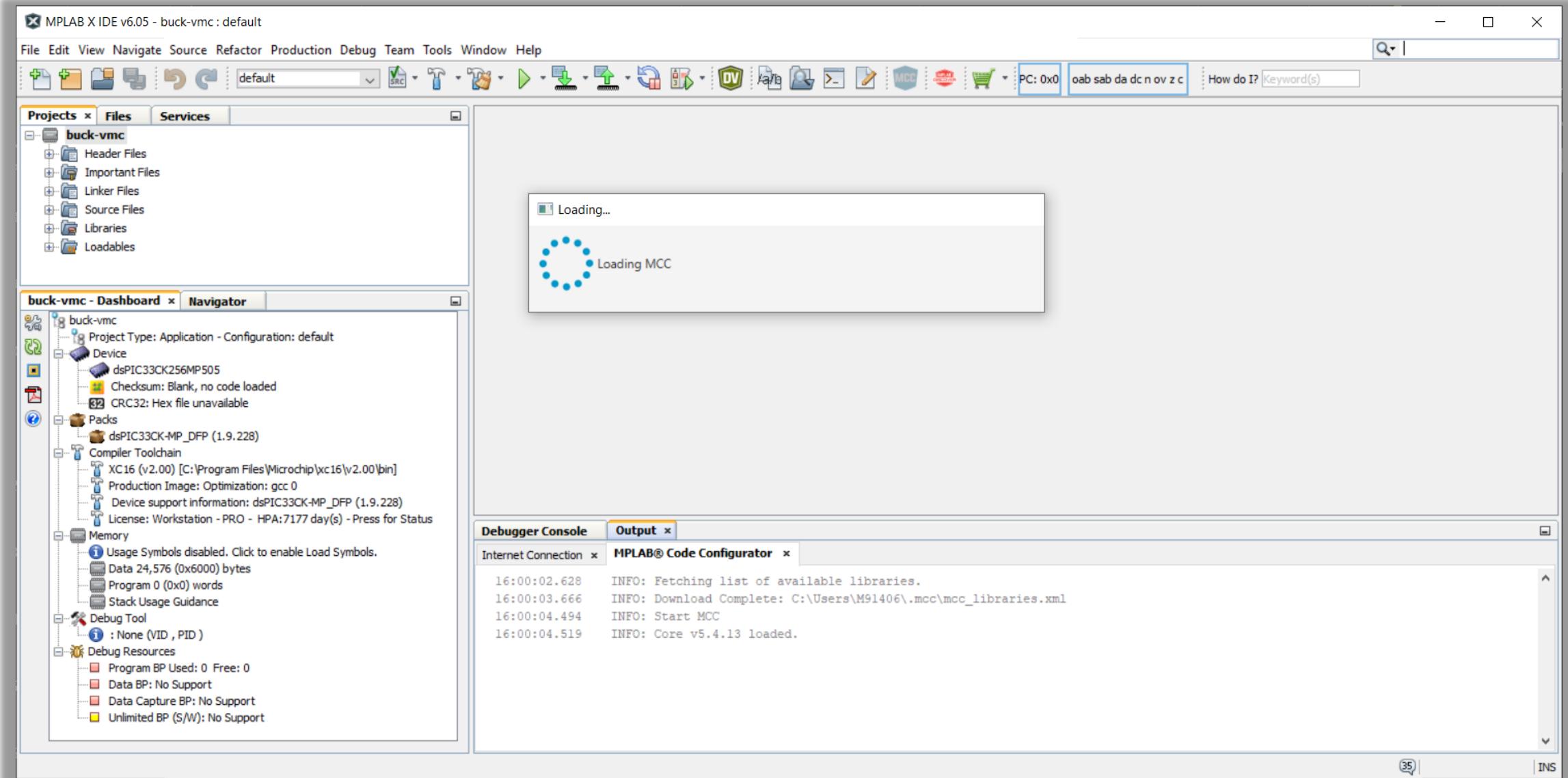
*: Curious to see how this is playing out

- Overview Digital Power Starter Kit 3 (DPSK3)
- Requirements and Scope
- **Hands-On**
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB® PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start and Error Handling State Machine to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

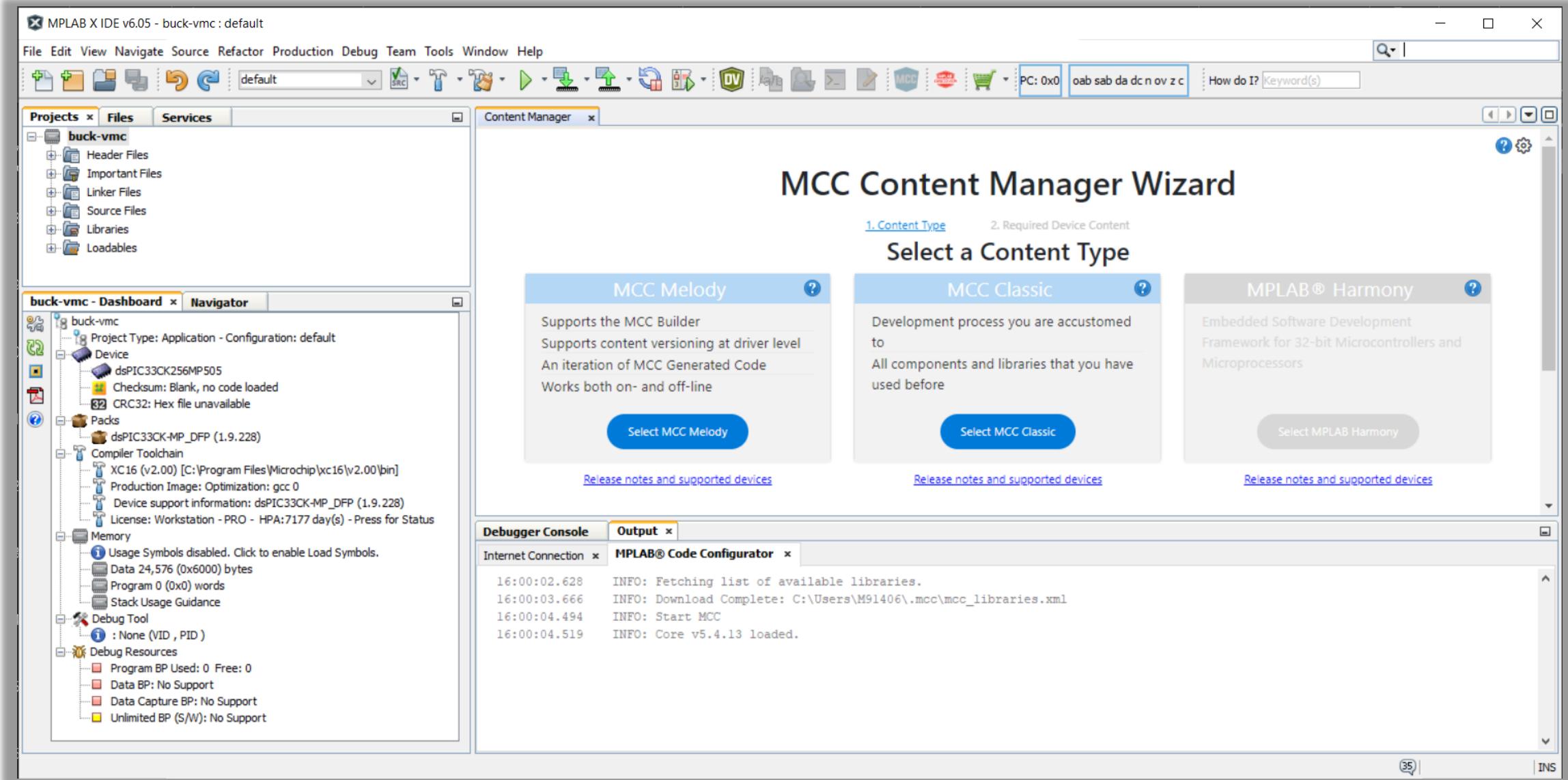
Open MPLAB® Code Configurator



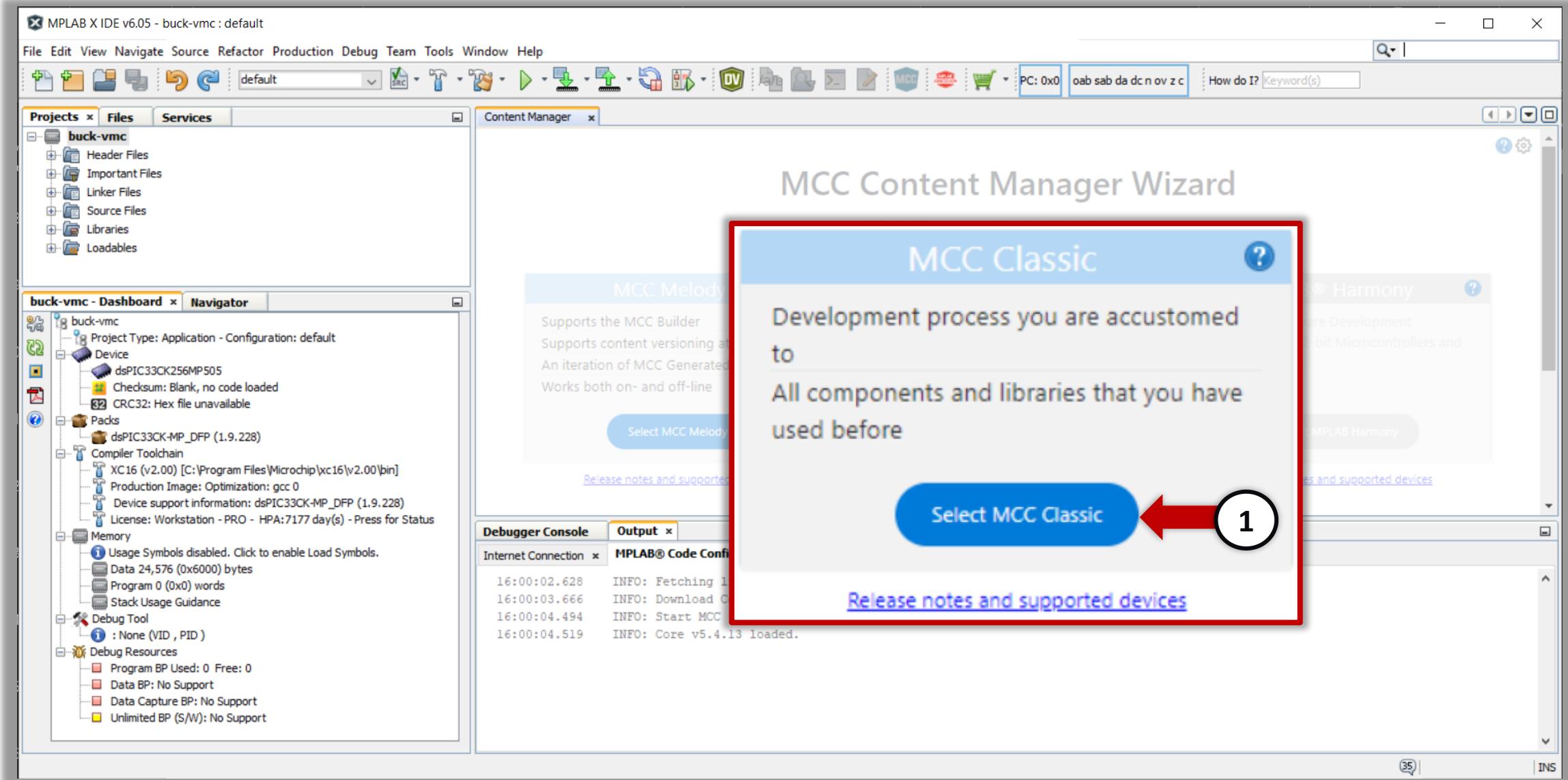
Open MPLAB® Code Configurator



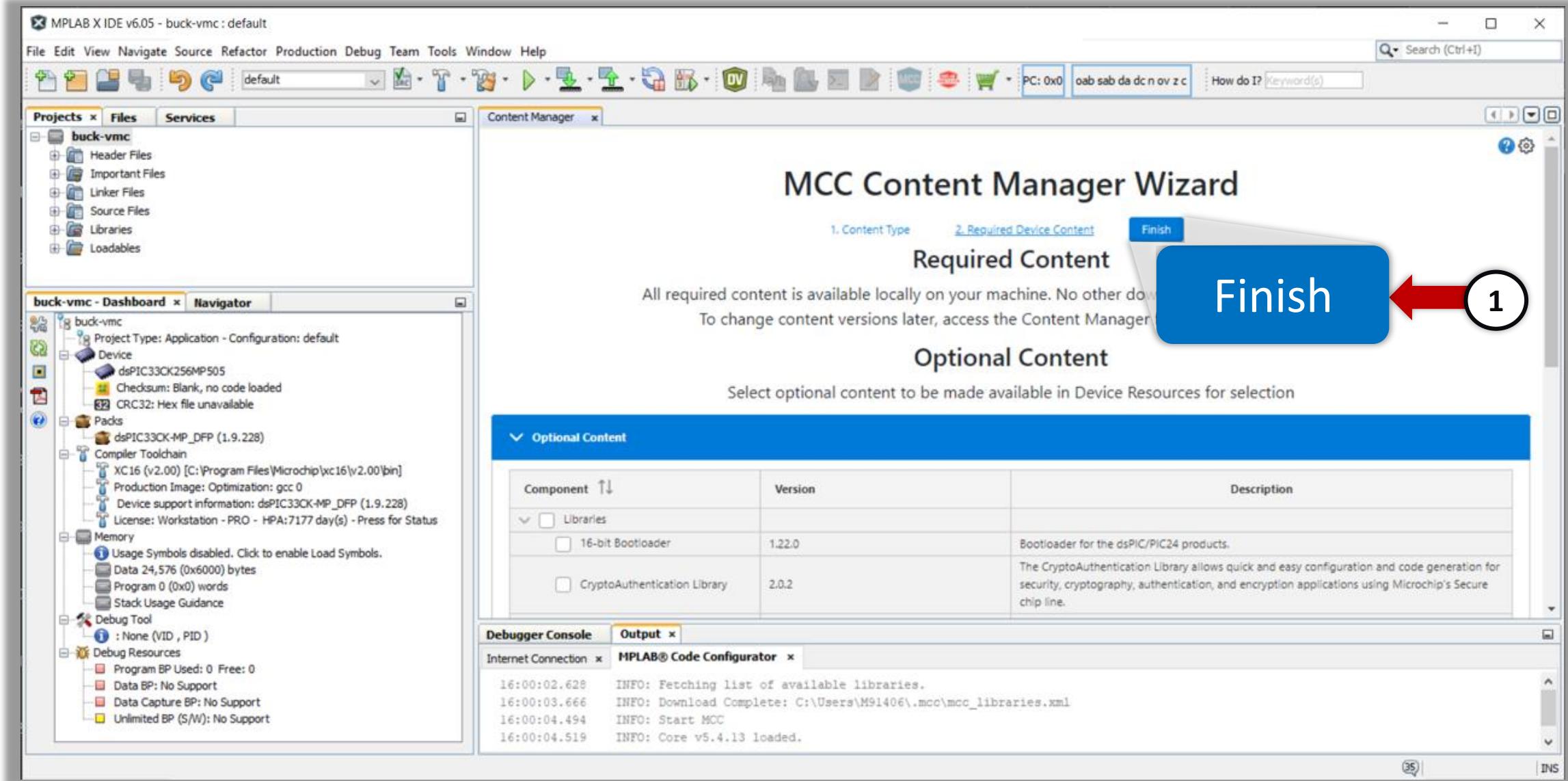
Open MPLAB® Code Configurator



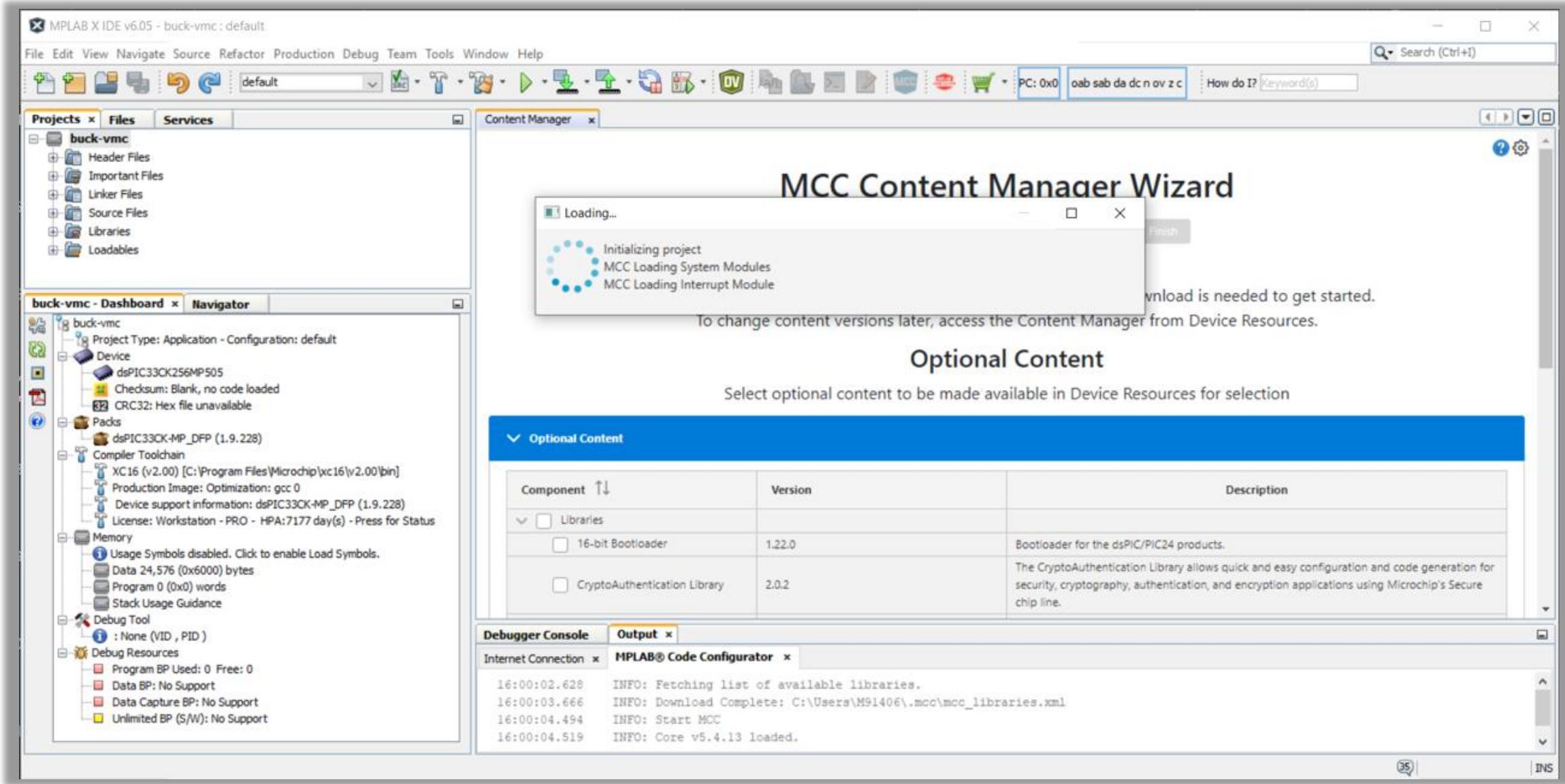
Open MPLAB® Code Configurator



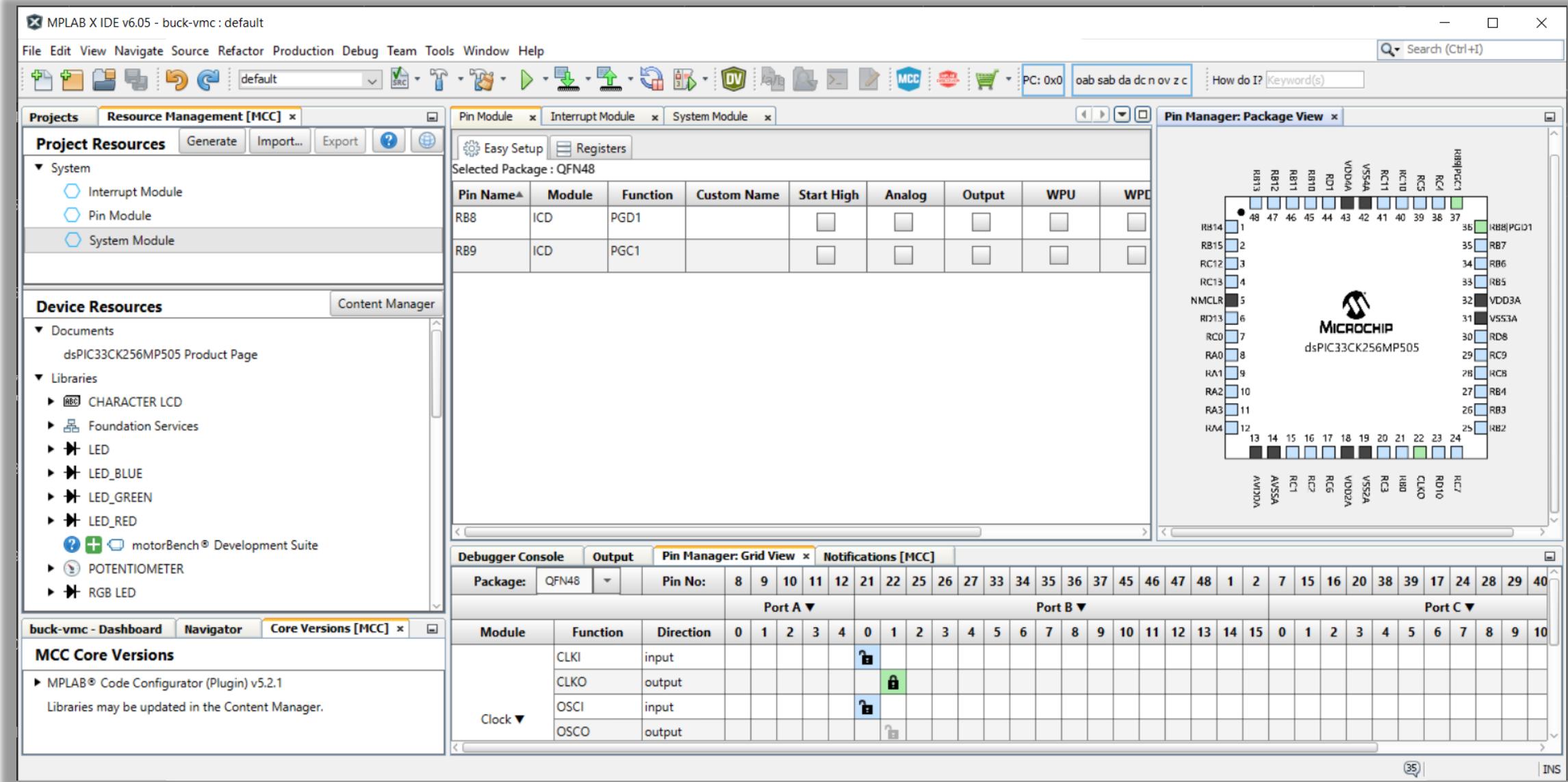
Open MPLAB® Code Configurator



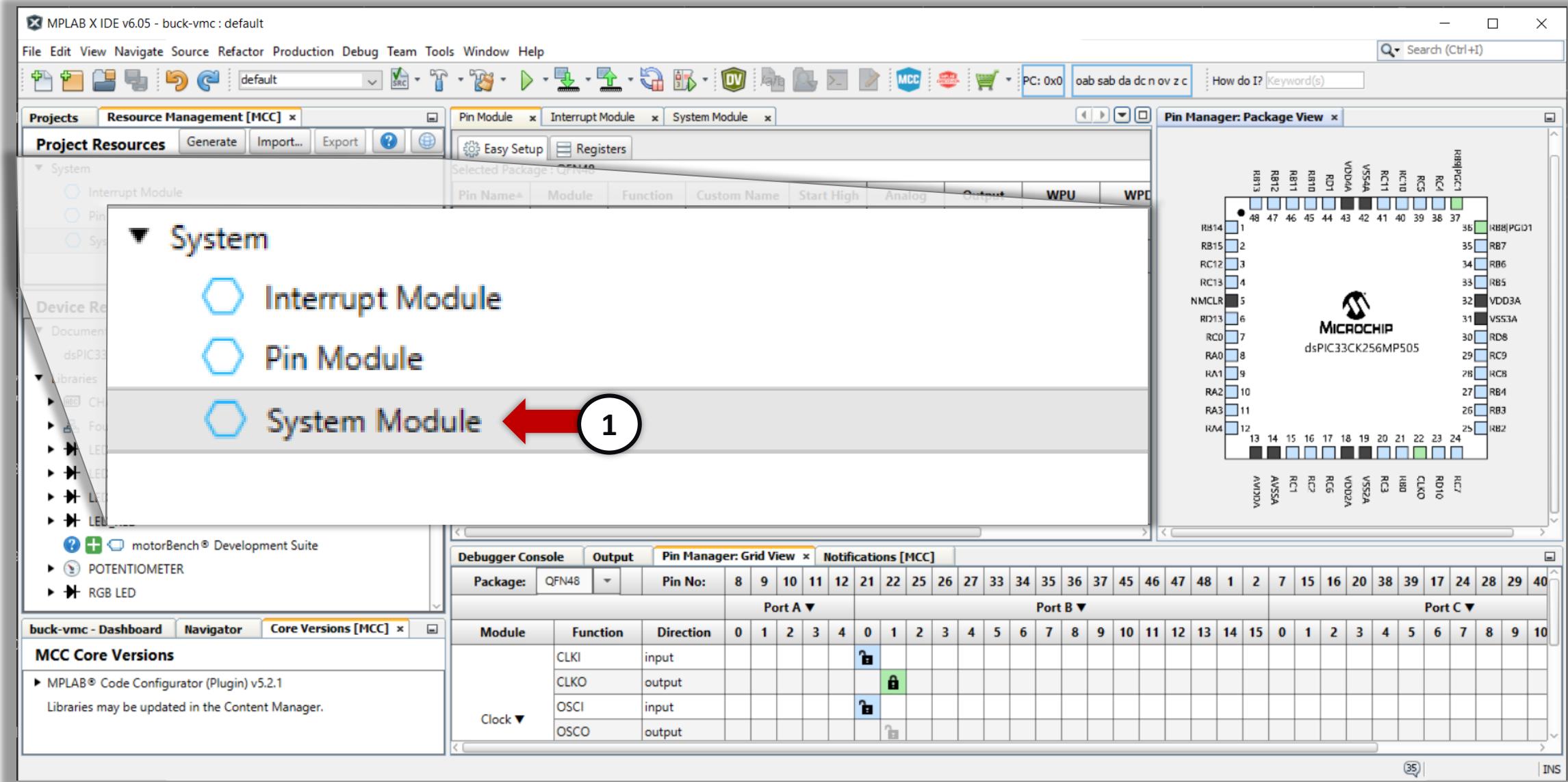
Open MPLAB® Code Configurator



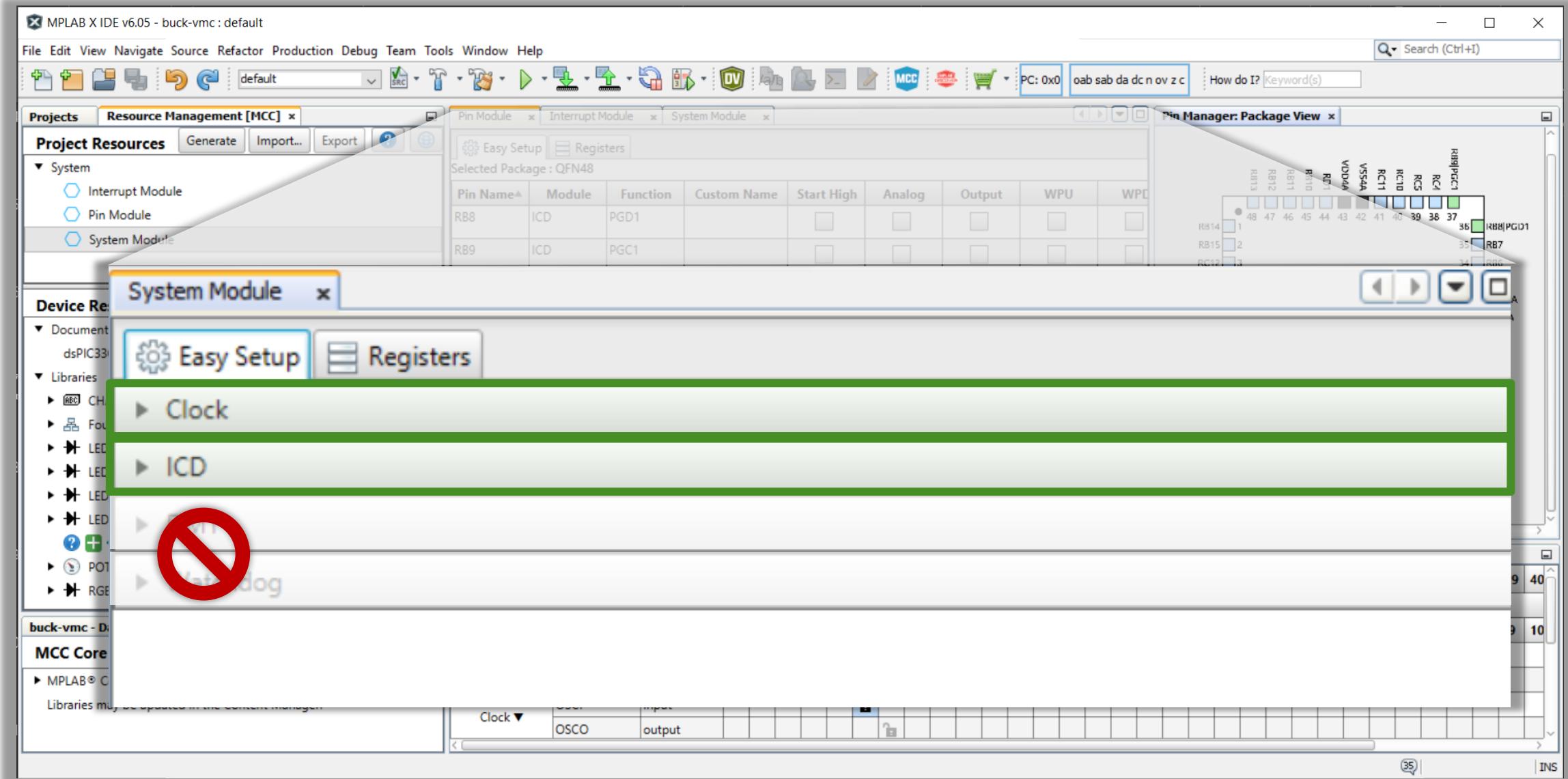
Open MPLAB® Code Configurator



Configuring the System Oscillator

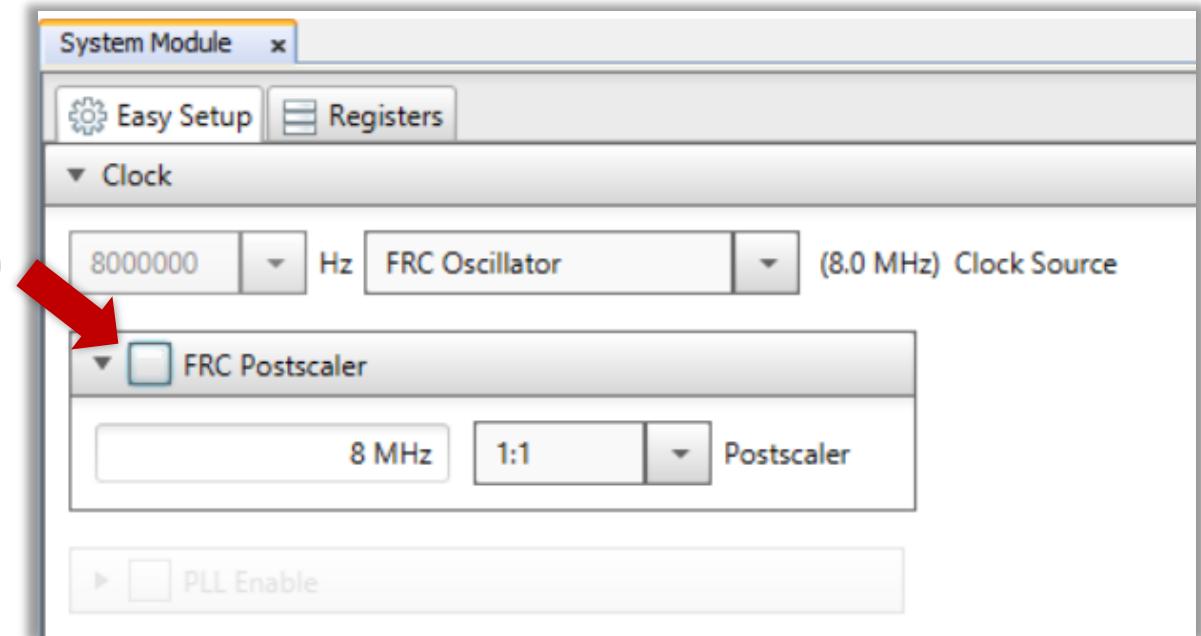


Configuring the System Oscillator



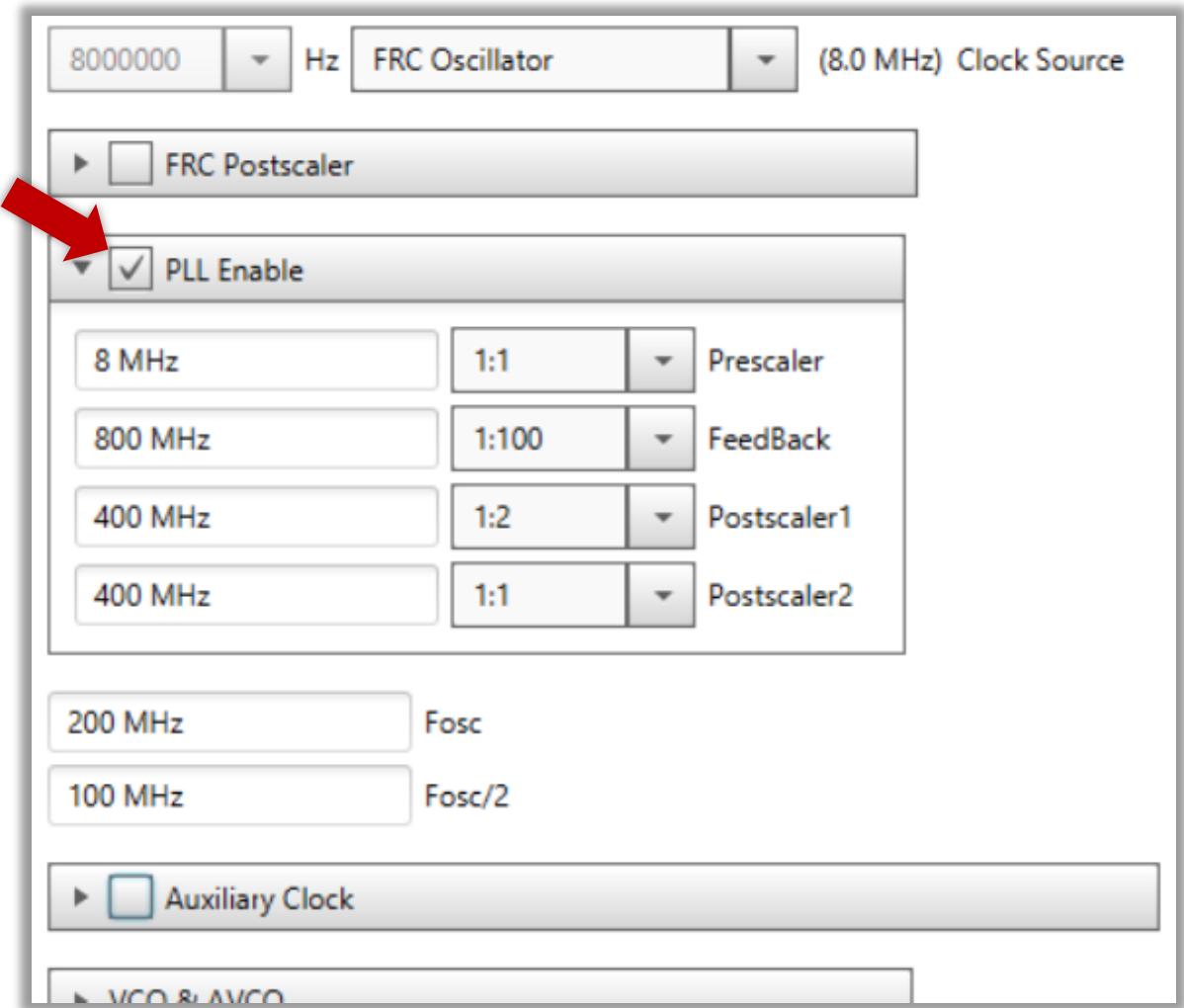
Configuring the System Oscillator

- Disable FRC Postscaler
- Enable PLL
- Enable Auxiliary Clock
- Set VCO/AVCO Dividers
- Set OSC2 as General Purpose IO
- Select ICD Port PGC1/PGD1

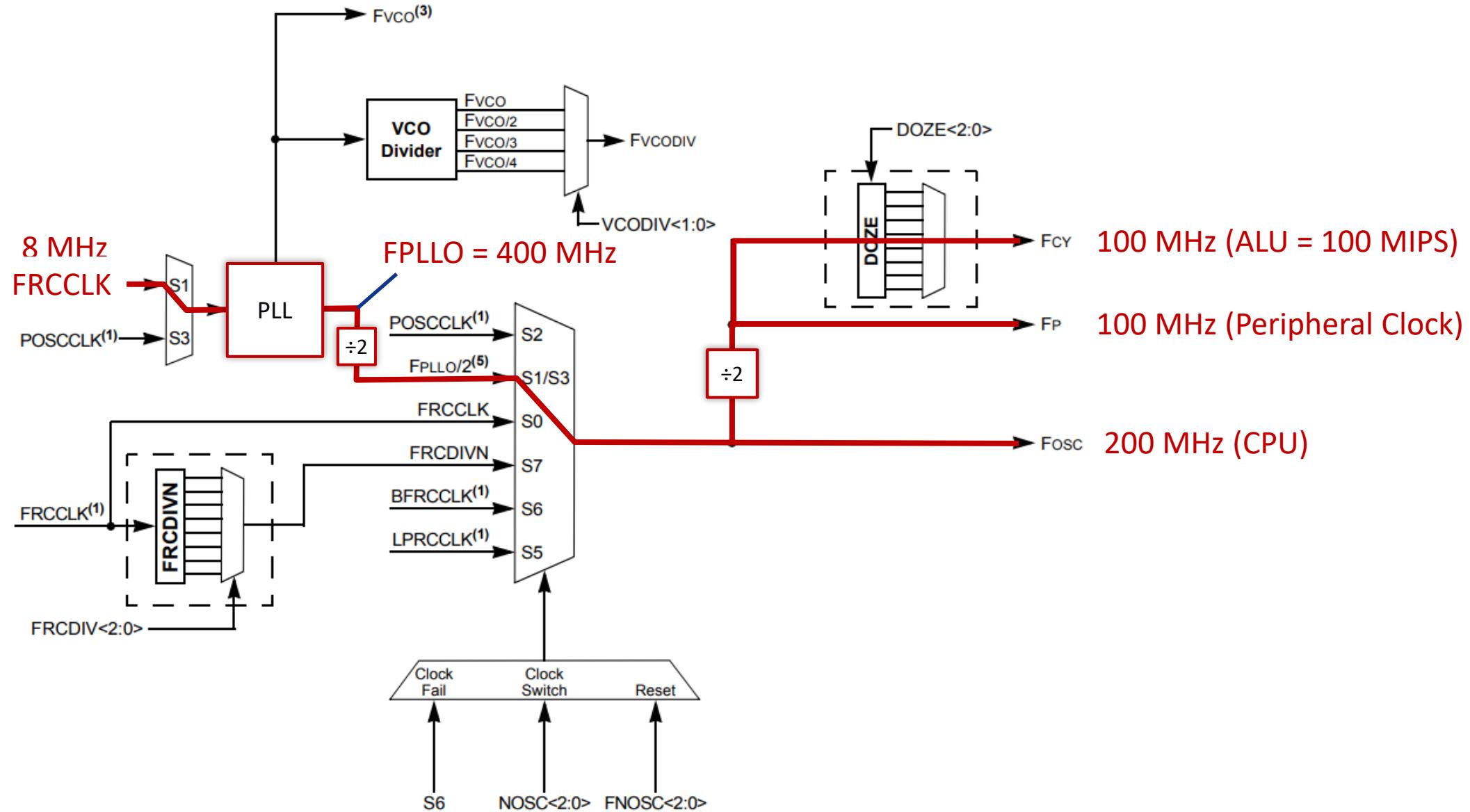


Configuring the System Oscillator

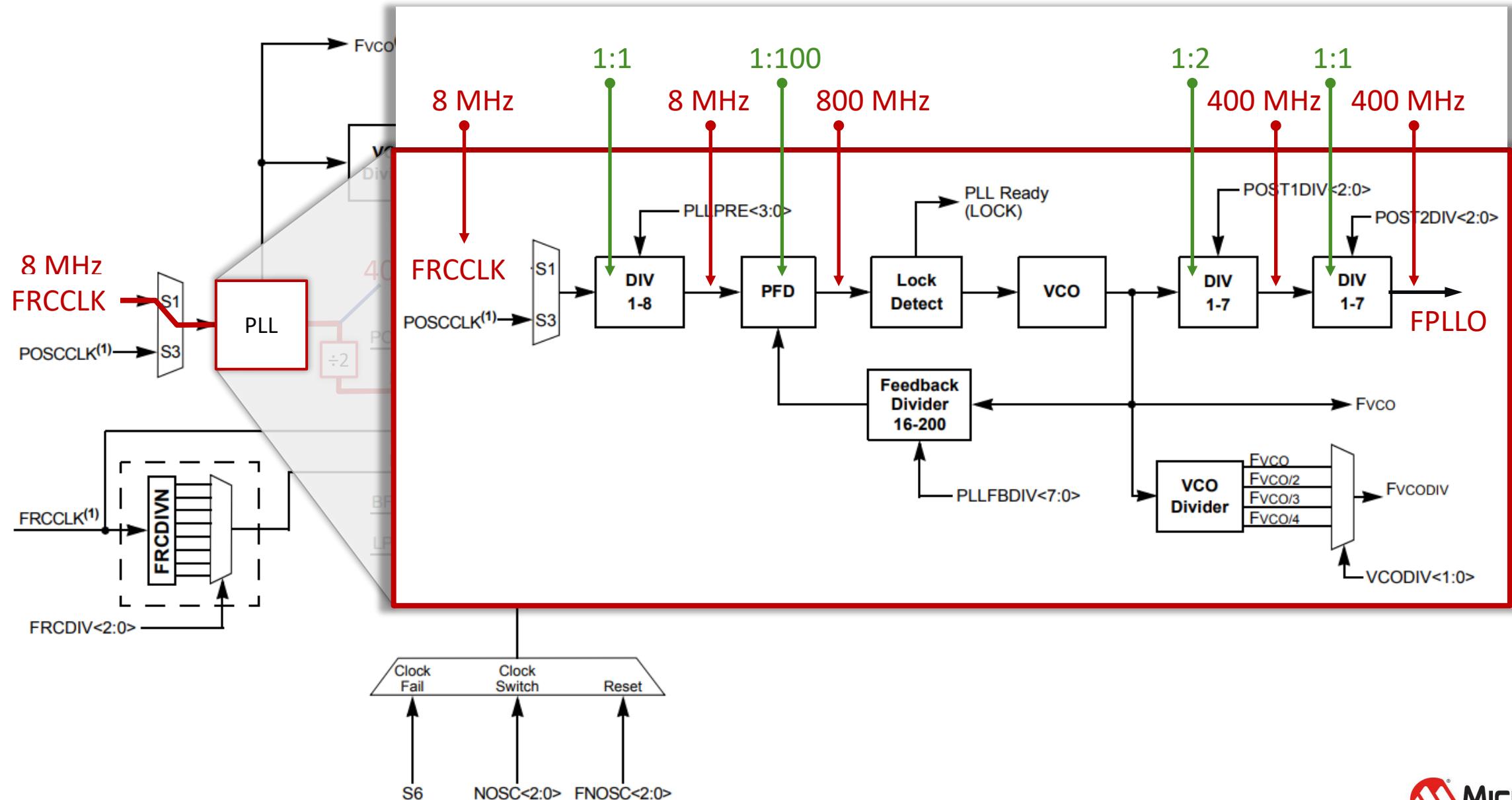
- Disable FRC Postscaler
- **Enable PLL**
- Enable Auxiliary Clock
- Set VCO/AVCO Dividers
- Set OSC2 as General Purpose IO
- Select ICD Port PGC1/PGD1



Configuring the System Oscillator

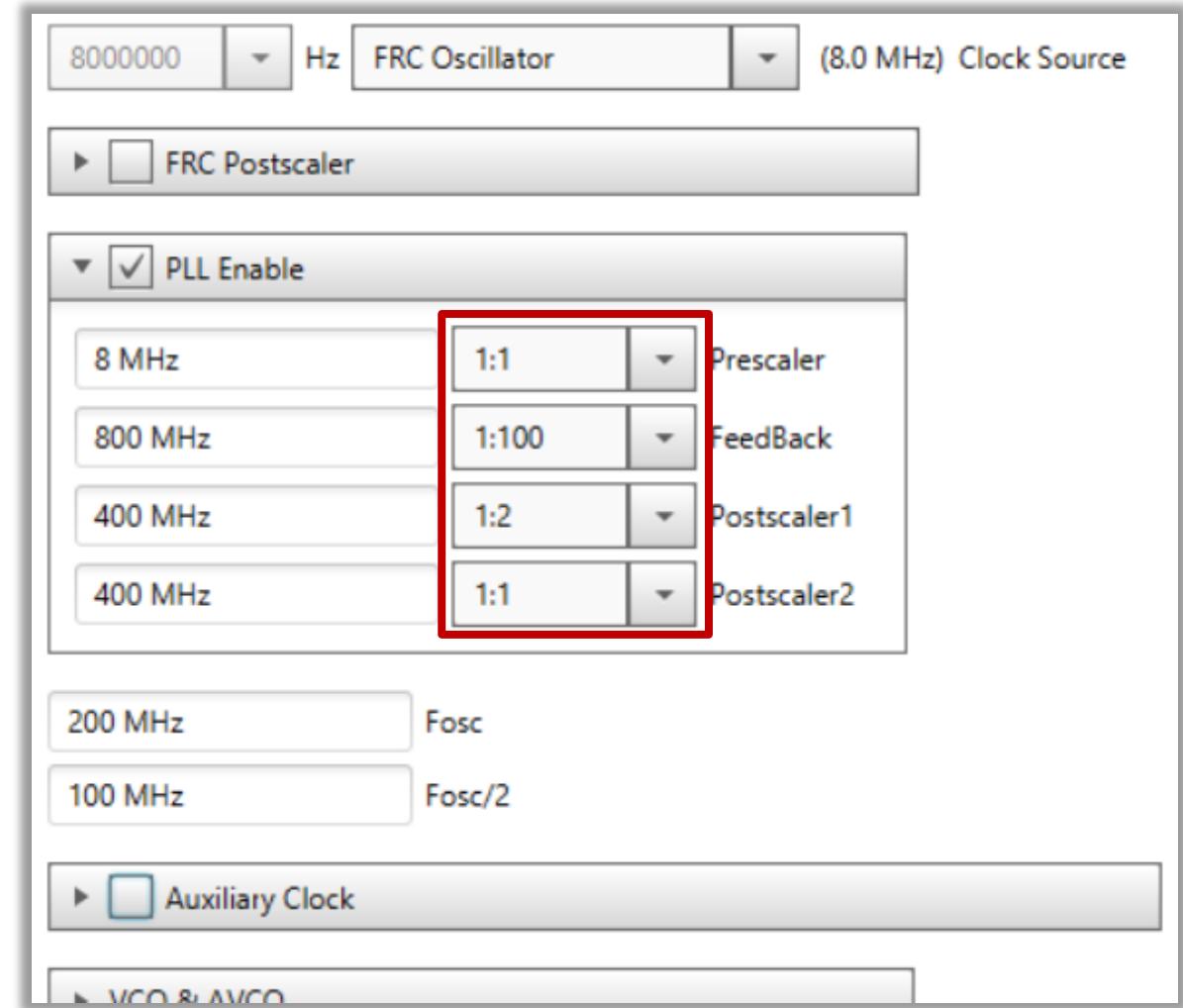


Configuring the System Oscillator



Configuring the System Oscillator

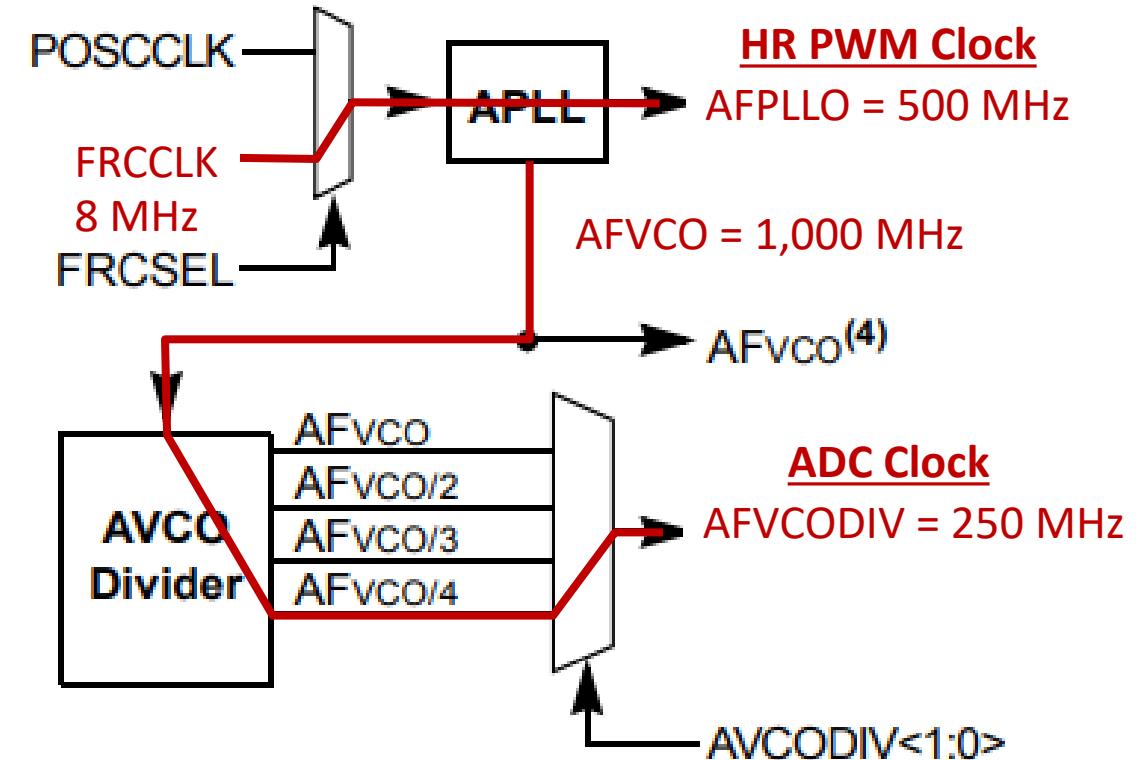
- Disable FRC Postscaler
- **Enable PLL**
 - Prescaler = 1:1
 - Feedback = 1:100
 - Postscaler 1 = 1:2
 - Postscaler 2 = 1:1
- Enable Auxiliary Clock
- Set VCO/AVCO Dividers
- Set OSC2 as General Purpose IO
- Select ICD Port PGC1/PGD1



Configuring the System Oscillator

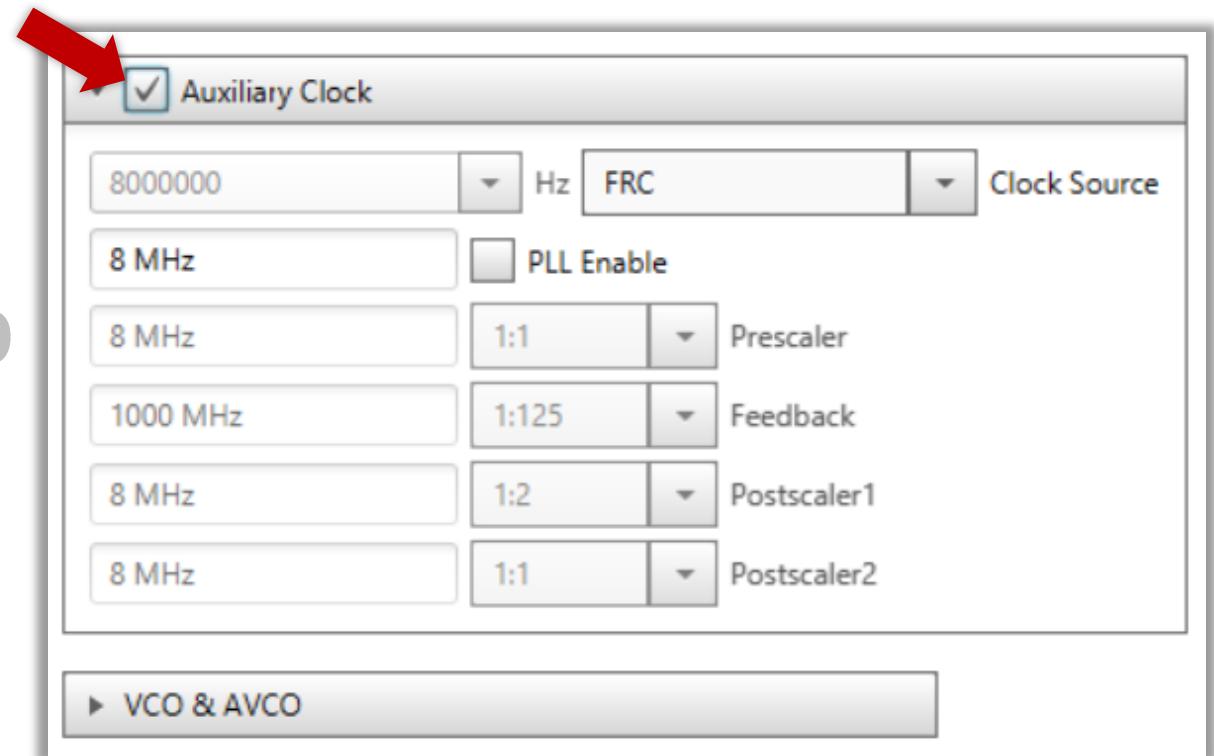
- **Auxiliary Clock Configuration**

- PWM and ADC must be set on same clock domain **to reduce clock jitter**
- Both, PWM and ADC, have their own, internal clock dividers



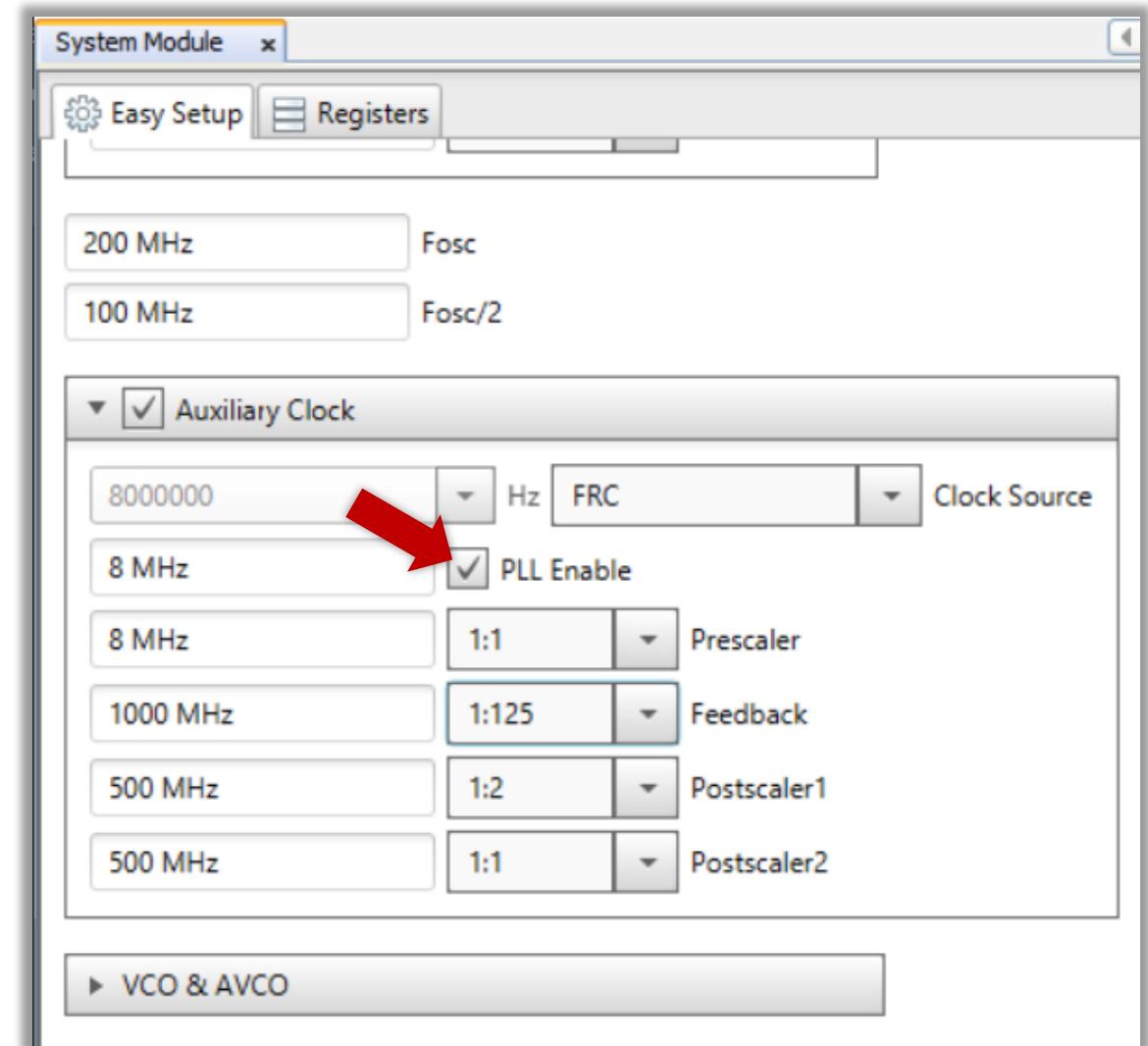
Configuring the System Oscillator

- Disable FRC Postscaler
- Enable PLL
- **Enable Auxiliary Clock**
- Set VCO/AVCO Dividers
- Set OSC2 as General Purpose IO
- Select ICD Port PGC1/PGD1



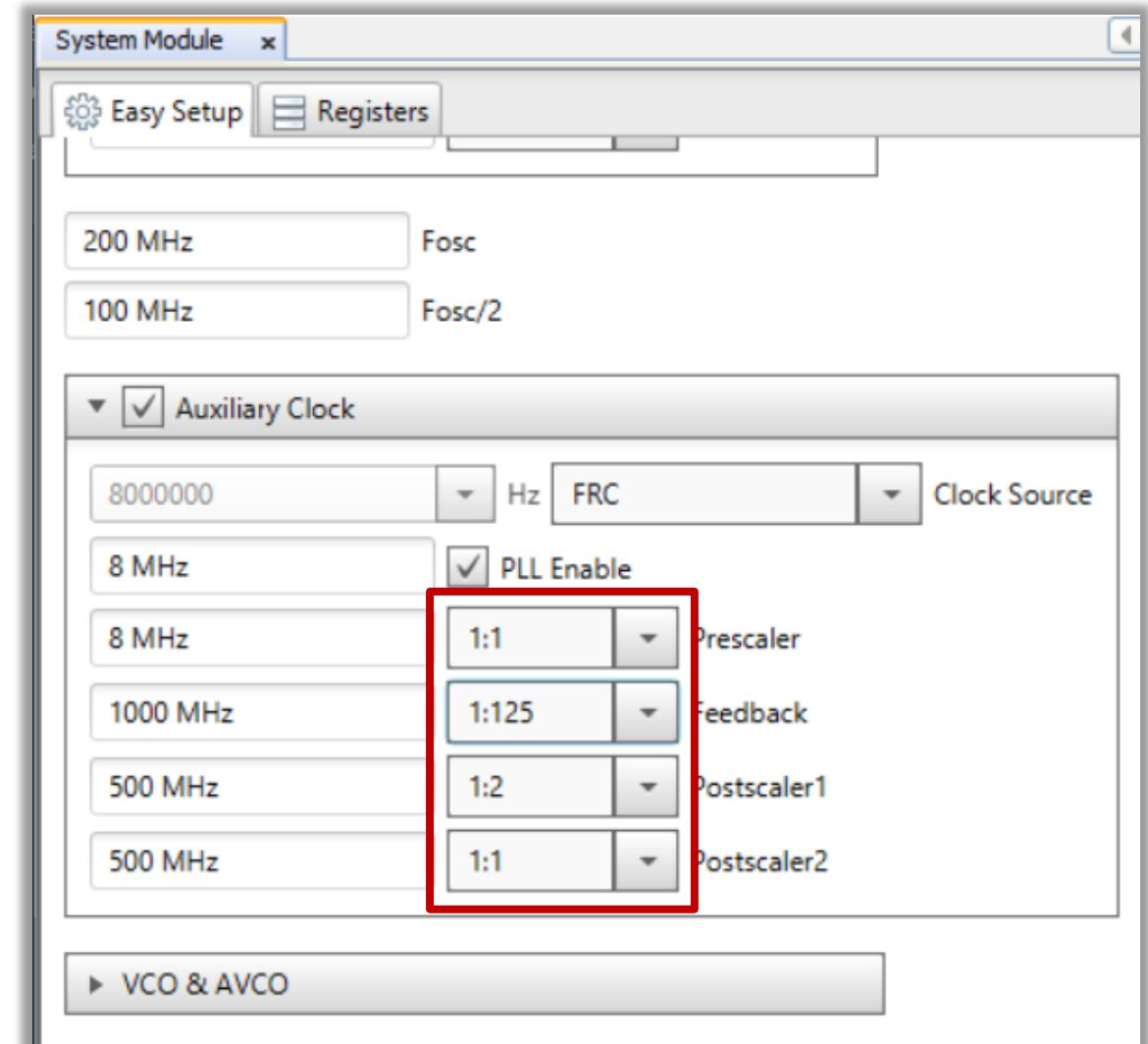
Configuring the System Oscillator

- Disable FRC Postscaler
- Enable PLL
- **Enable Auxiliary Clock**
 - Enable PLL
 - Prescaler = 1:1
 - Feedback = 1:125
 - Postscaler 1 = 1:2
 - Postscaler 2 = 1:1
- Set VCO/AVCO Dividers
- Set OSC2 as General Purpose IO
- Select ICD Port PGC1/PGD1



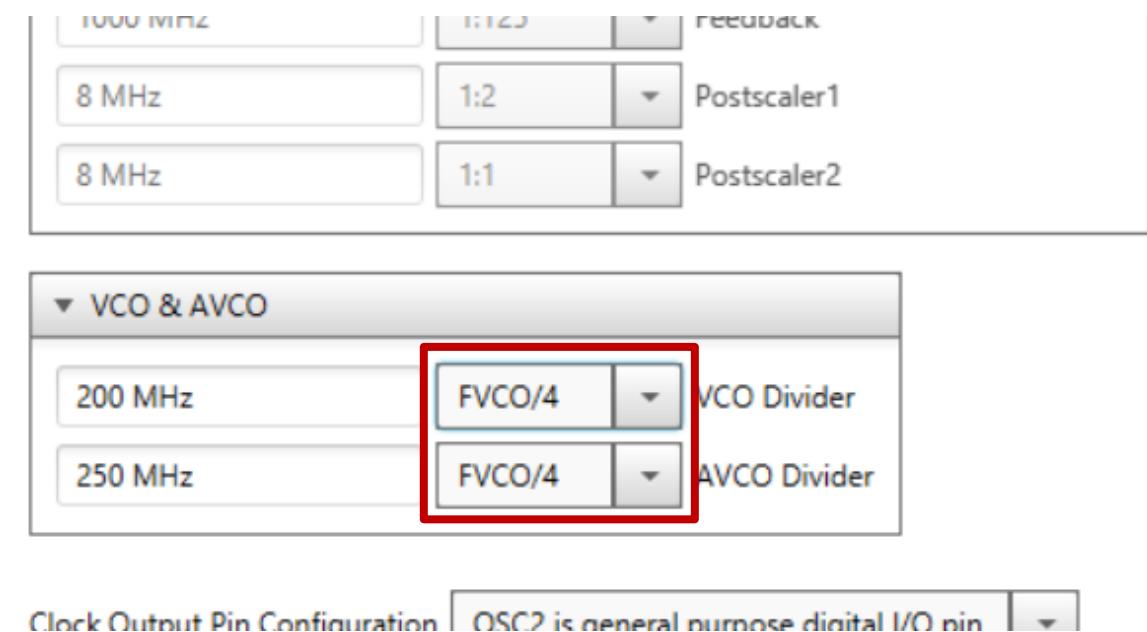
Configuring the System Oscillator

- Disable FRC Postscaler
- Enable PLL
- **Enable Auxiliary Clock**
 - Enable PLL
 - Prescaler = 1:1
 - Feedback = 1:125
 - Postscaler 1 = 1:2
 - Postscaler 2 = 1:1
- Set VCO/AVCO Dividers
- Set OSC2 as General Purpose IO
- Select ICD Port PGC1/PGD1



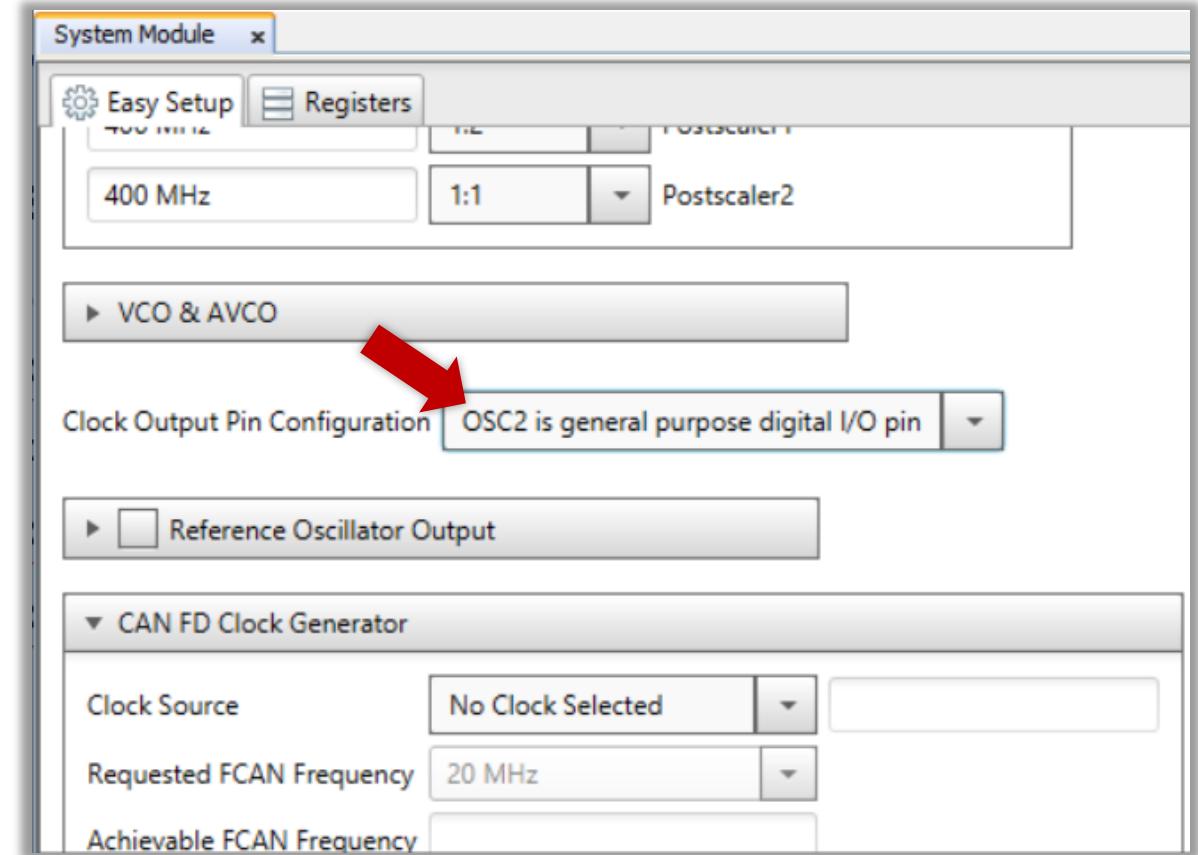
Configuring the System Oscillator

- Disable FRC Postscaler
- Enable PLL
- Enable Auxiliary Clock
- **Set VCO/AVCO Dividers**
- Set OSC2 as General Purpose IO
- Select ICD Port PGC1/PGD1



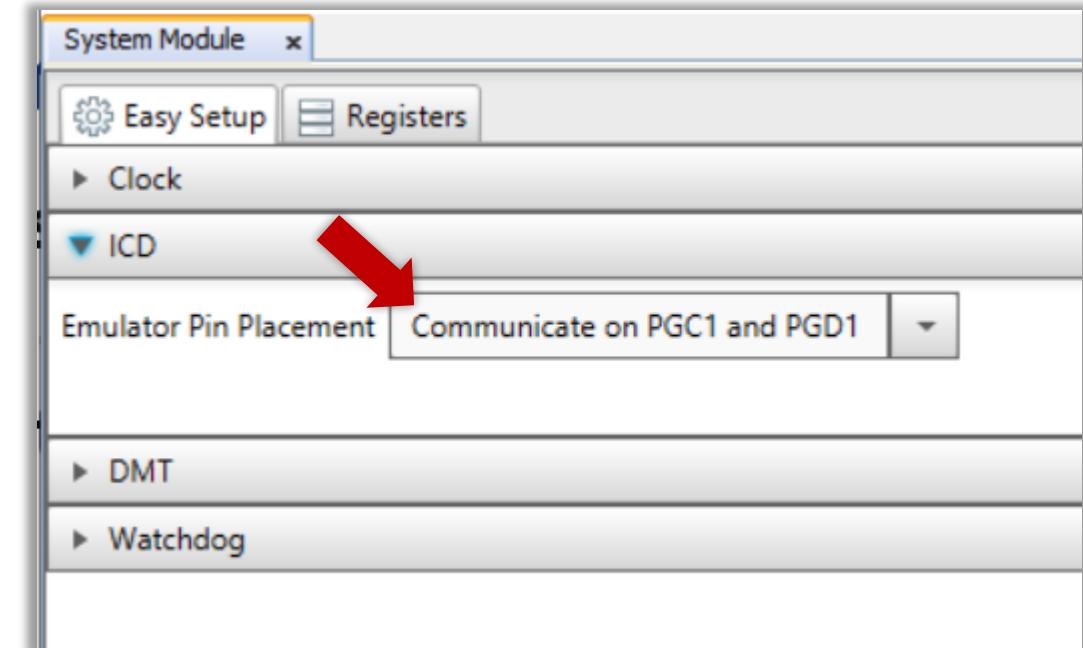
Configuring the System Oscillator

- Disable FRC Postscaler
- Enable PLL
- Enable Auxiliary Clock
- Set VCO/AVCO Dividers
- **Set OSC2 as General Purpose IO**
- Select ICD Port PGC1/PGD1

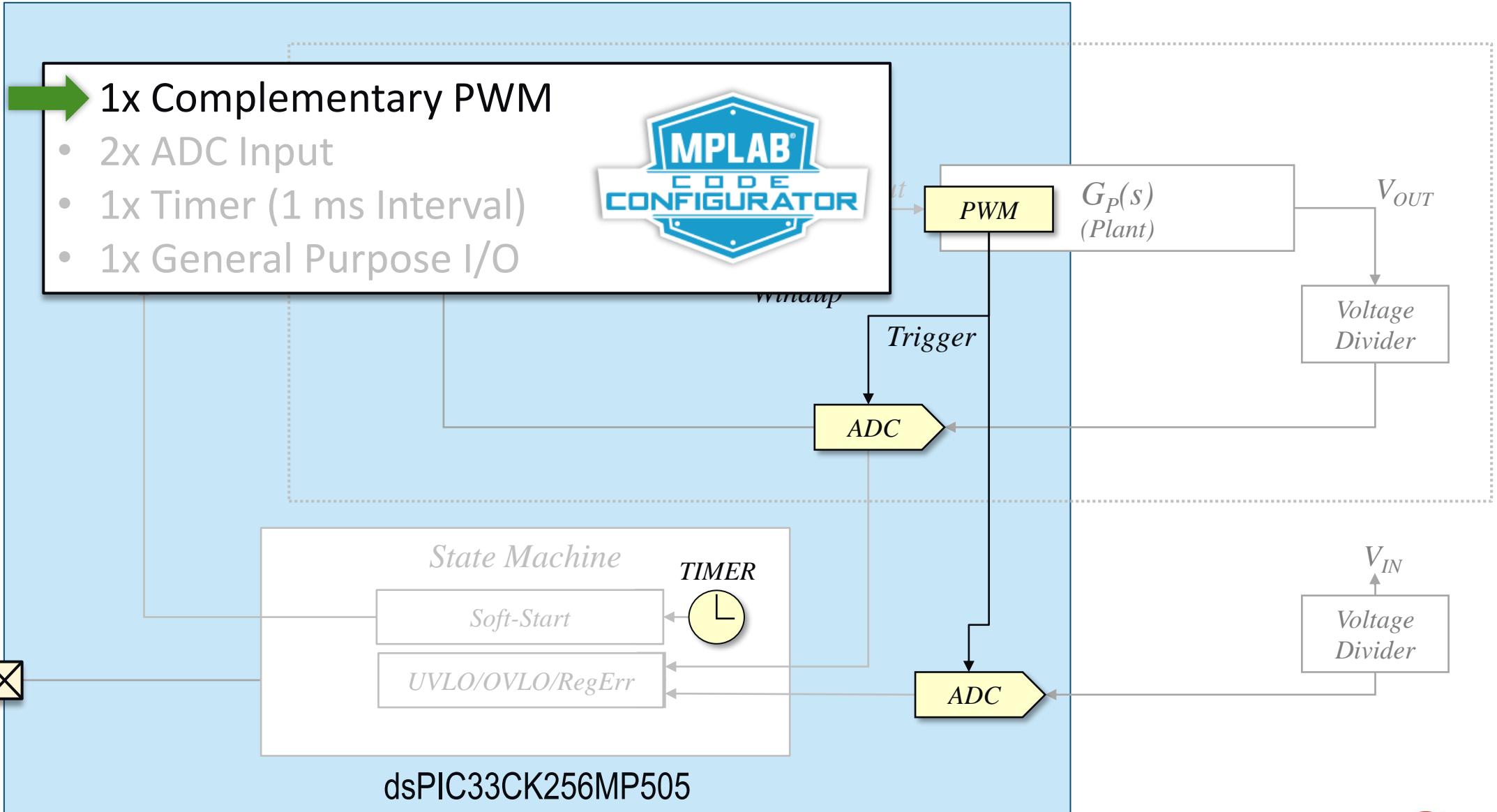


Configuring the System Oscillator

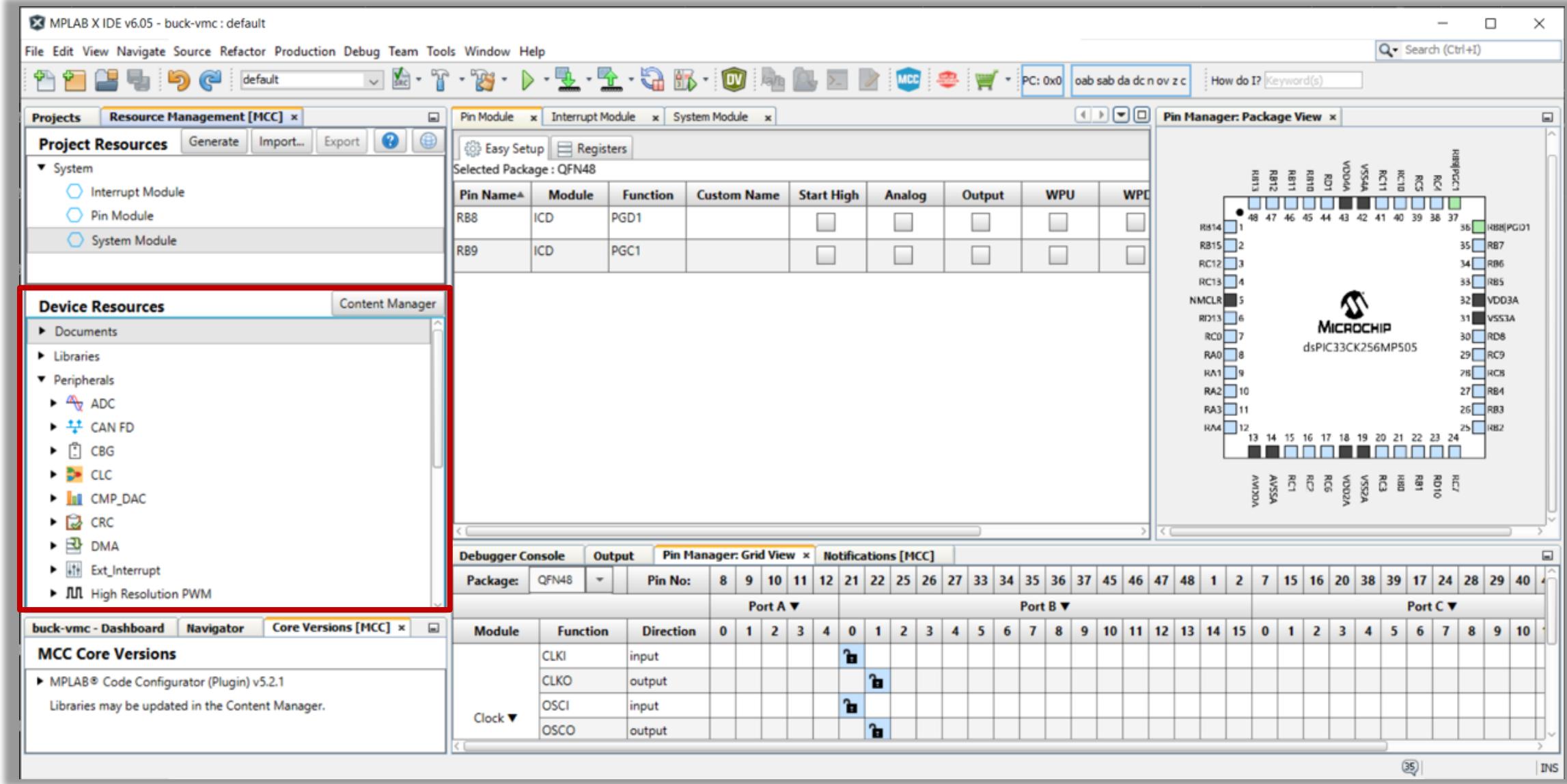
- Disable FRC Postscaler
- Enable PLL
- Enable Auxiliary Clock
- Set VCO/AVCO Dividers
- Set OSC2 as General Purpose IO
- **Select ICD Port PGC1/PGD1**



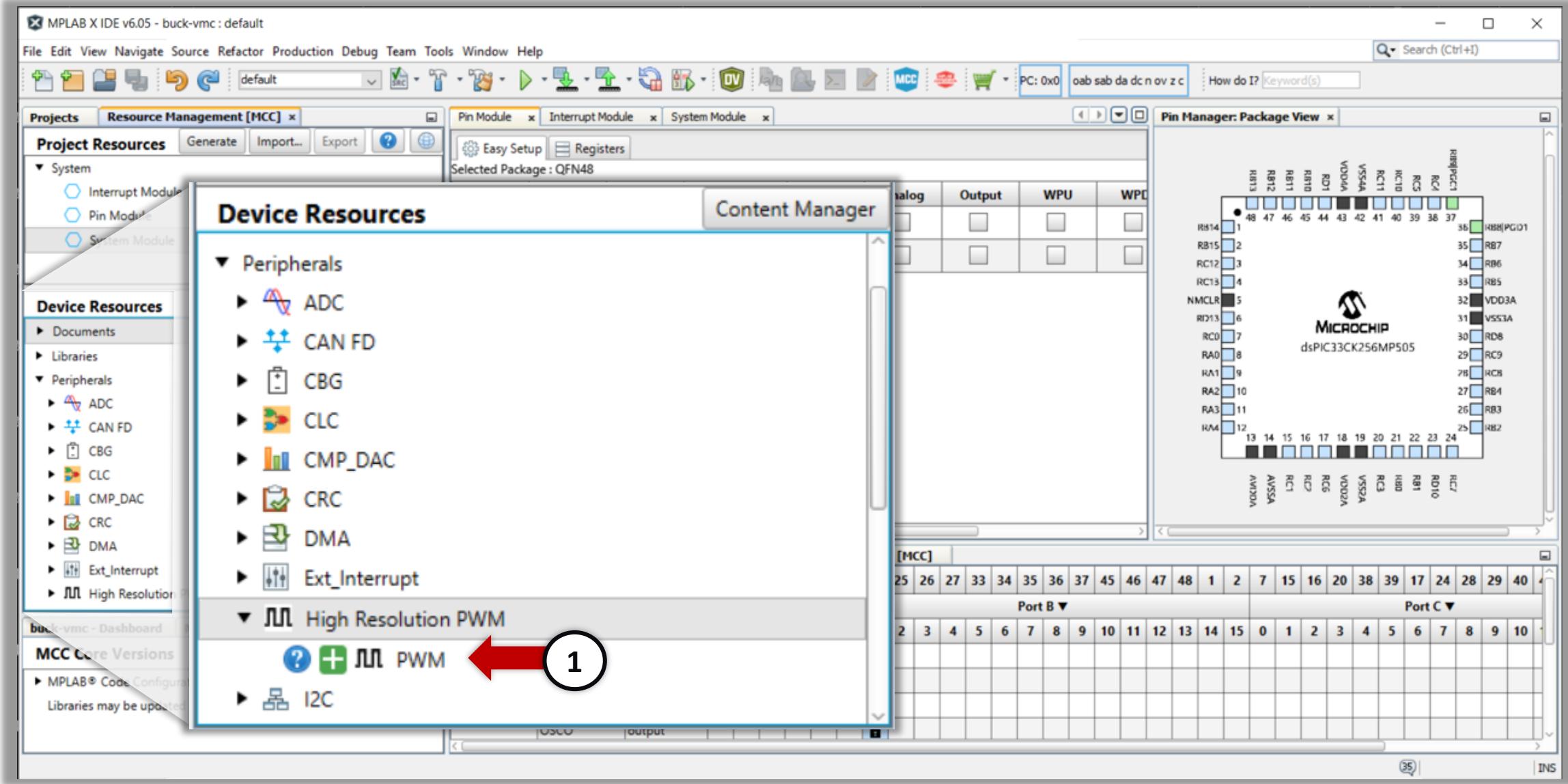
Requirements



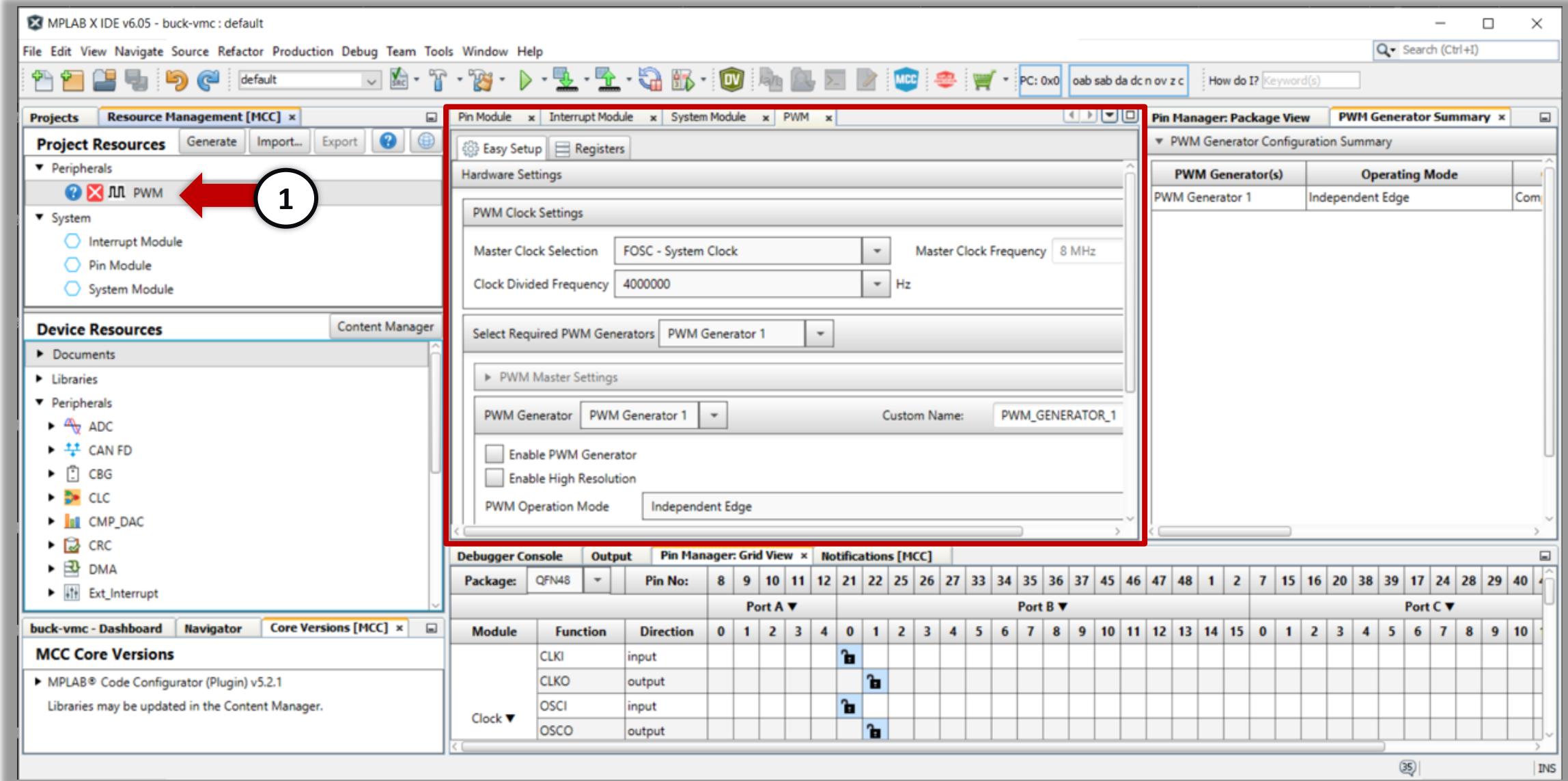
Configuring the PWM



Configuring the PWM

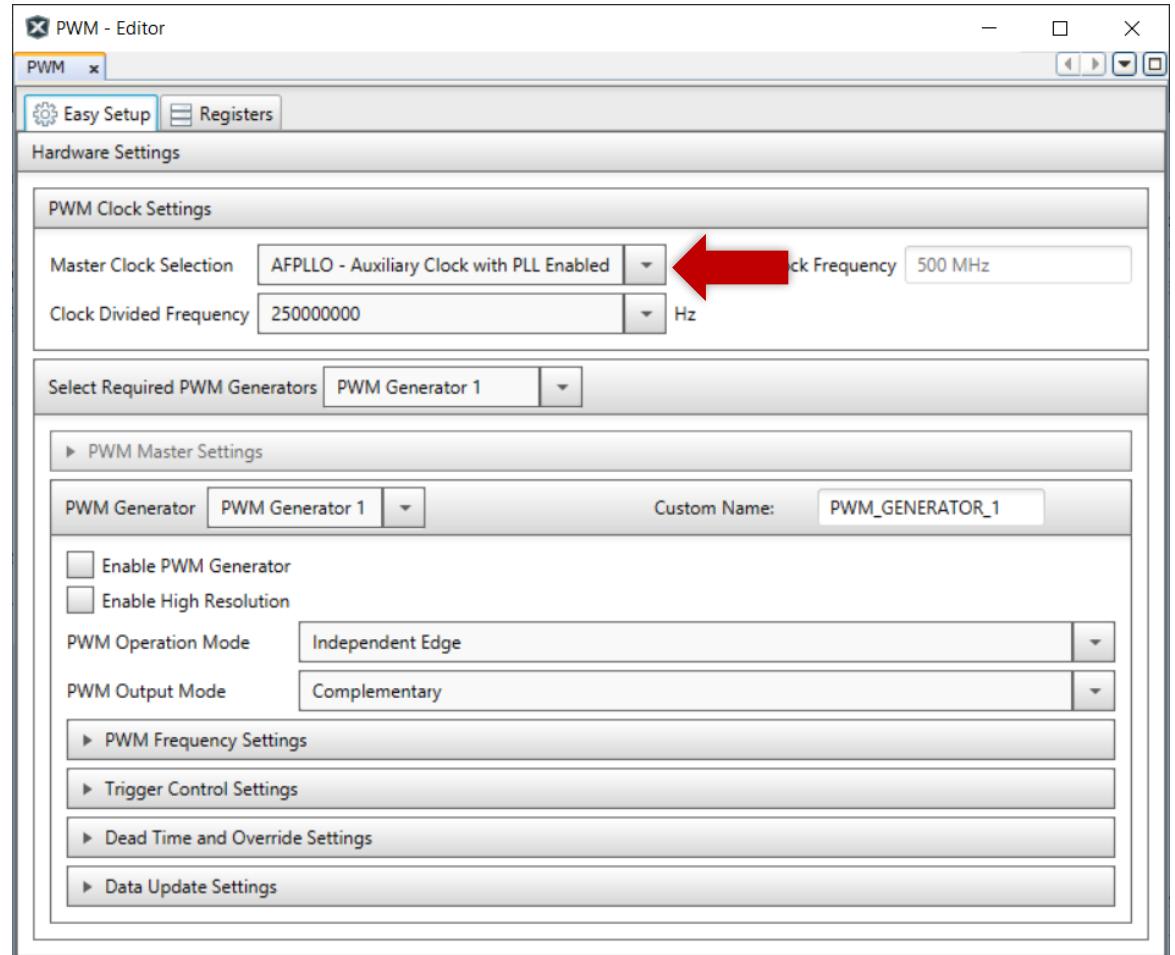


Configuring the PWM



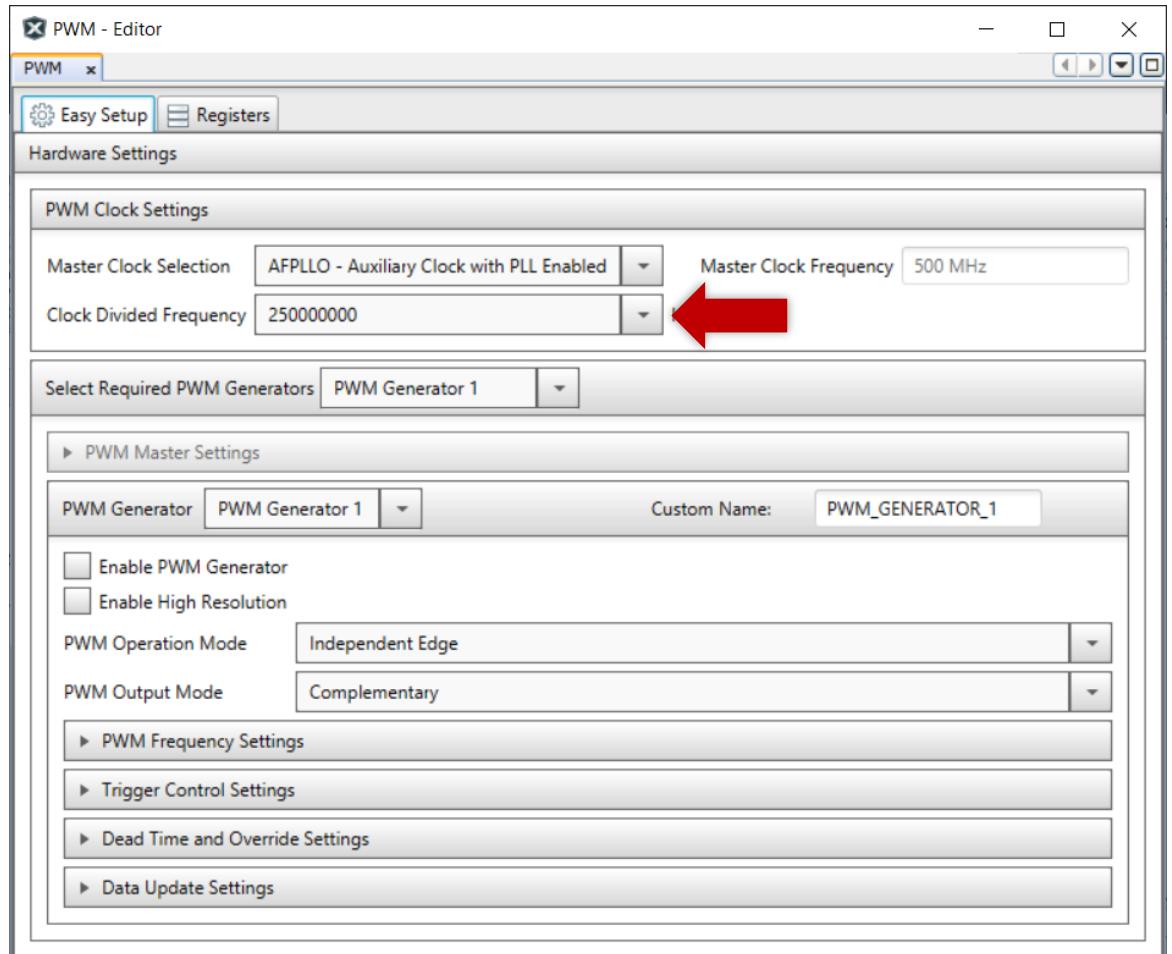
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)



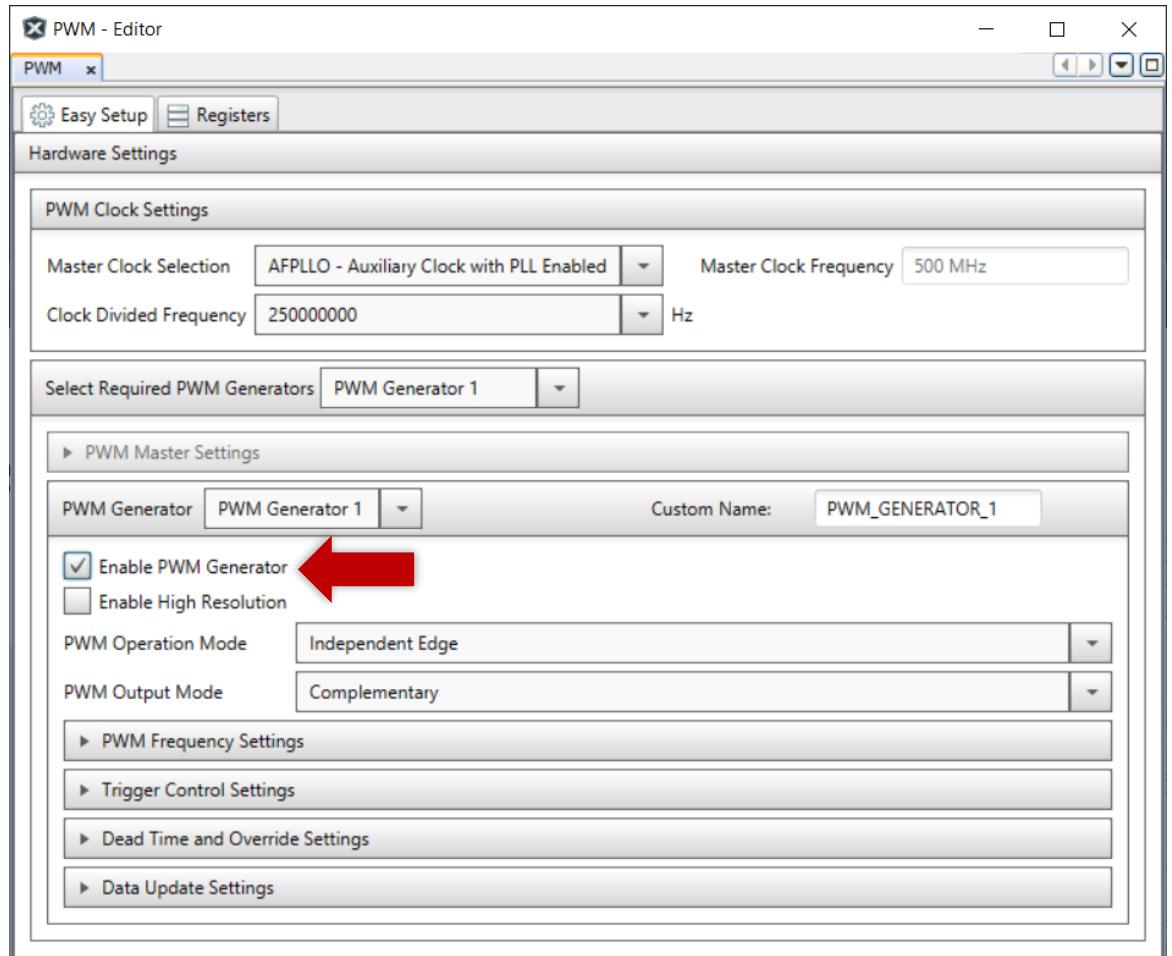
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency (=250 MHz)



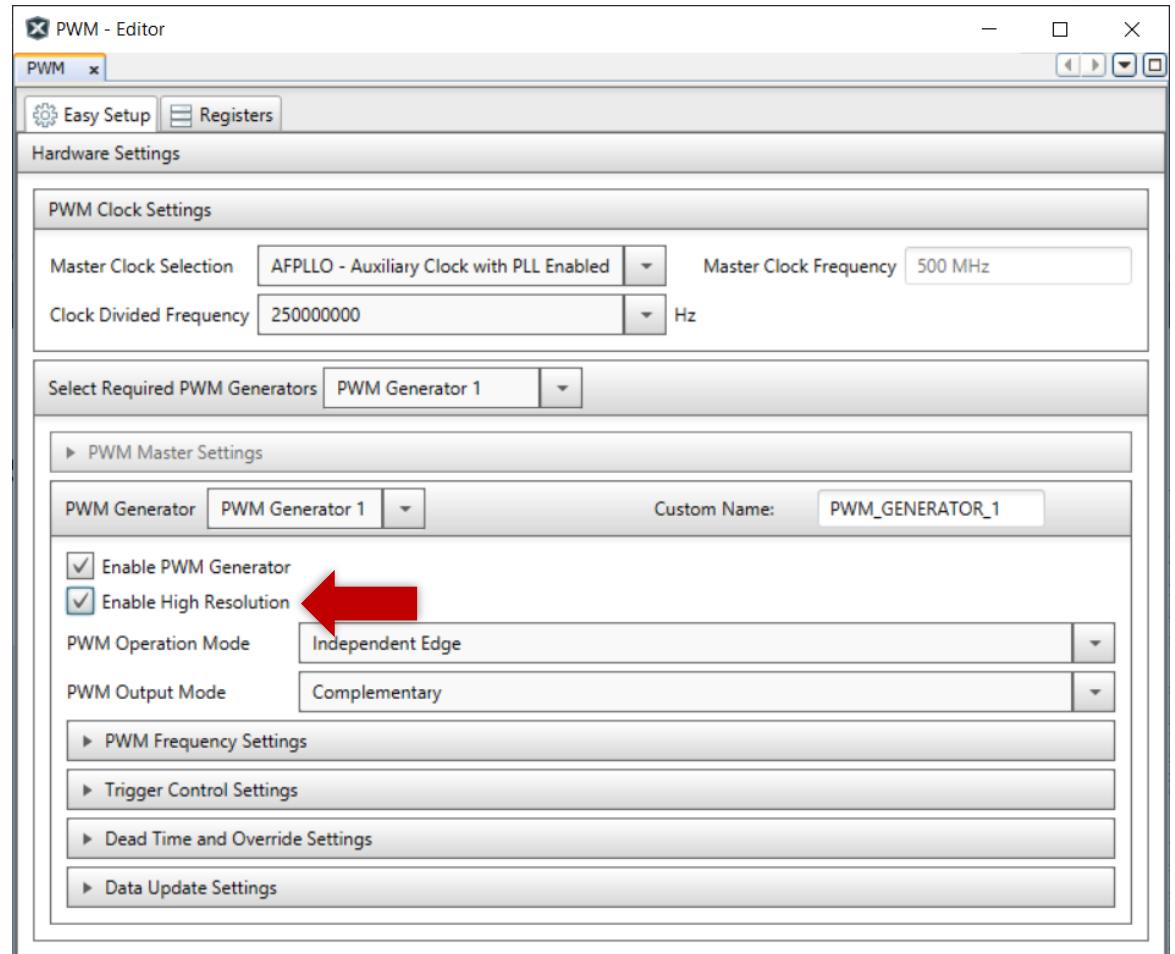
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1



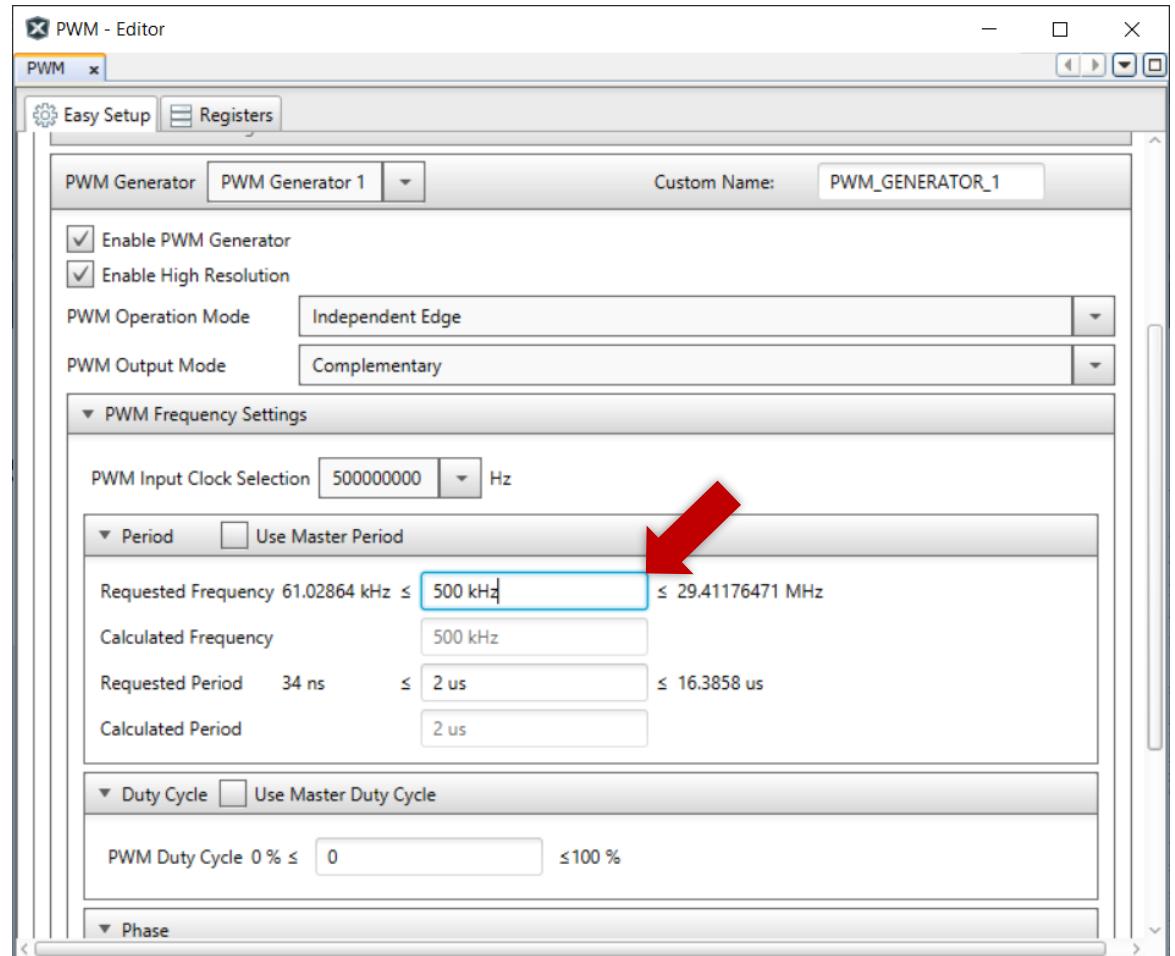
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1
- **Enable High Resolution**



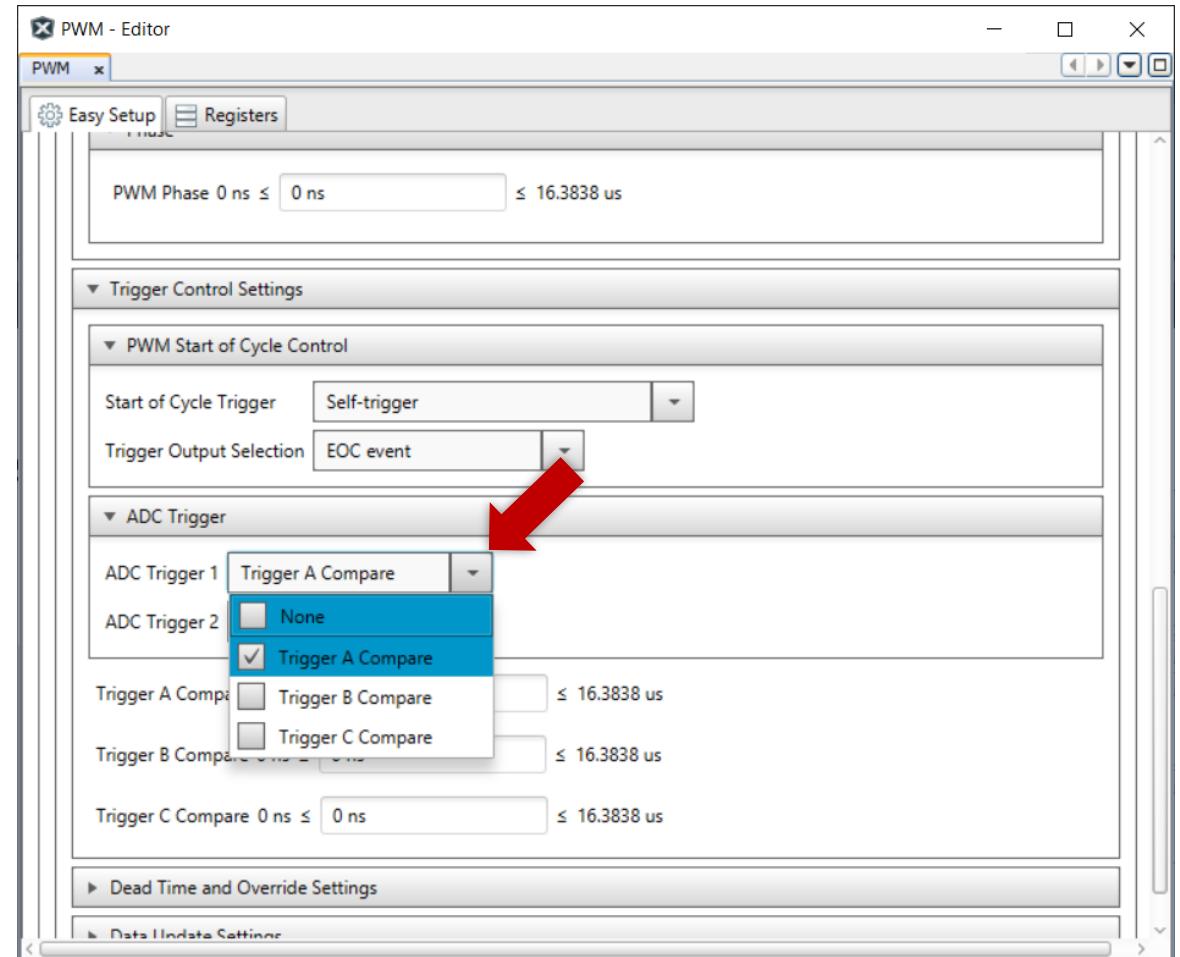
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1
- Enable High Resolution
- Set PWM Frequencies (=500 kHz)



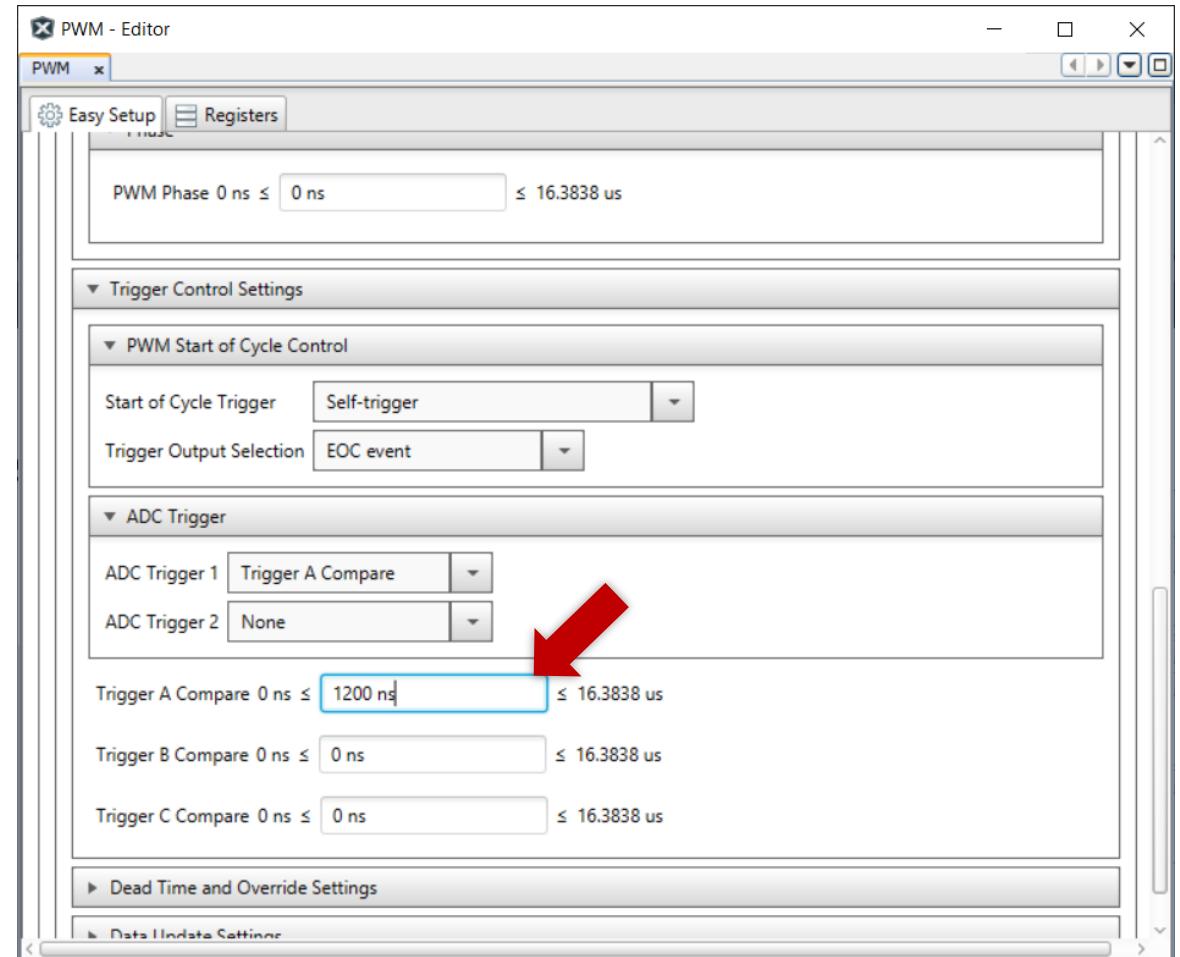
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1
- Enable High Resolution
- Set PWM Frequencies (=500 kHz)
- Set ADC Trigger 1 (=Trigger A Compare)



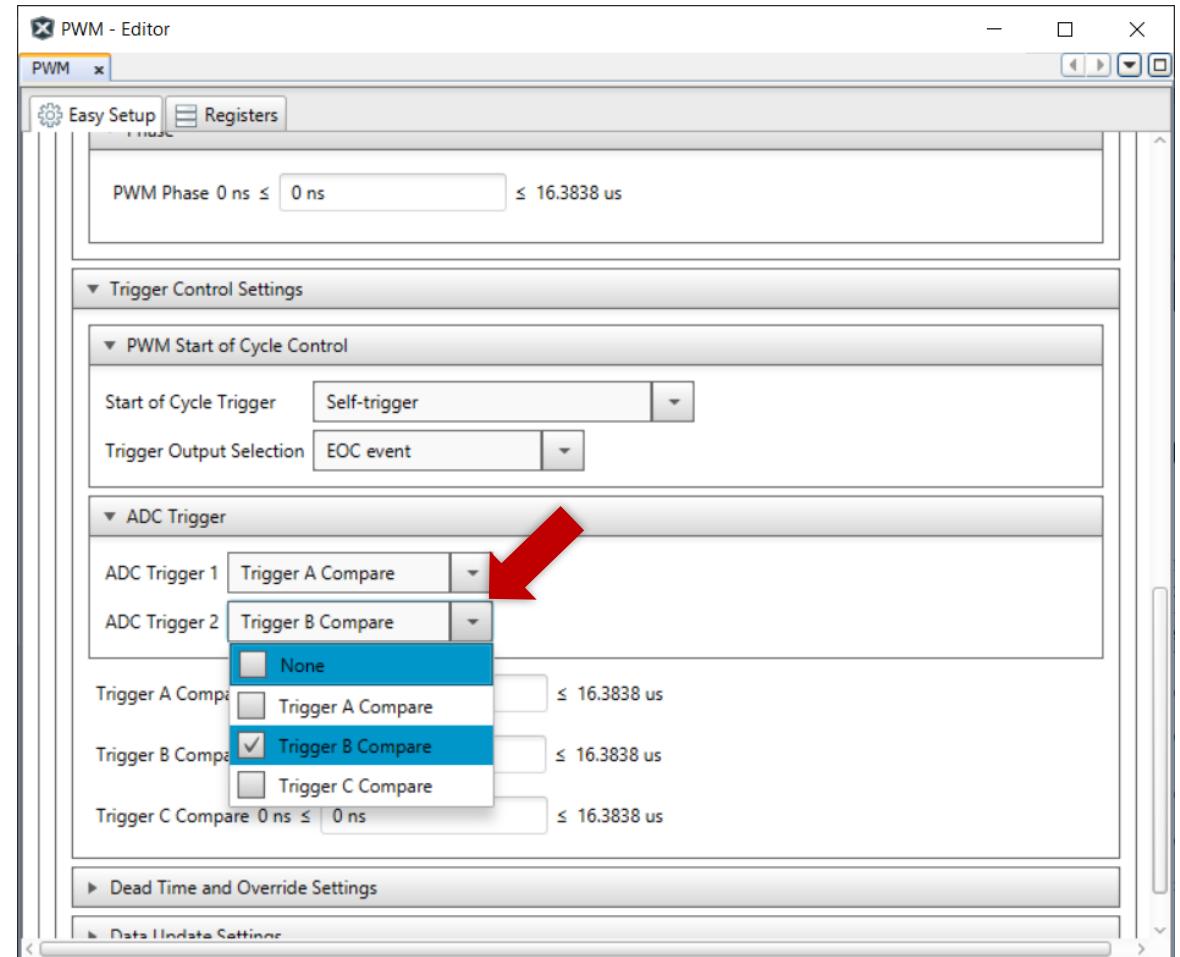
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1
- Enable High Resolution
- Set PWM Frequencies (=500 kHz)
- Set ADC Trigger 1 (=Trigger A Compare)
- **Set Trigger A Compare (=1200 ns)**



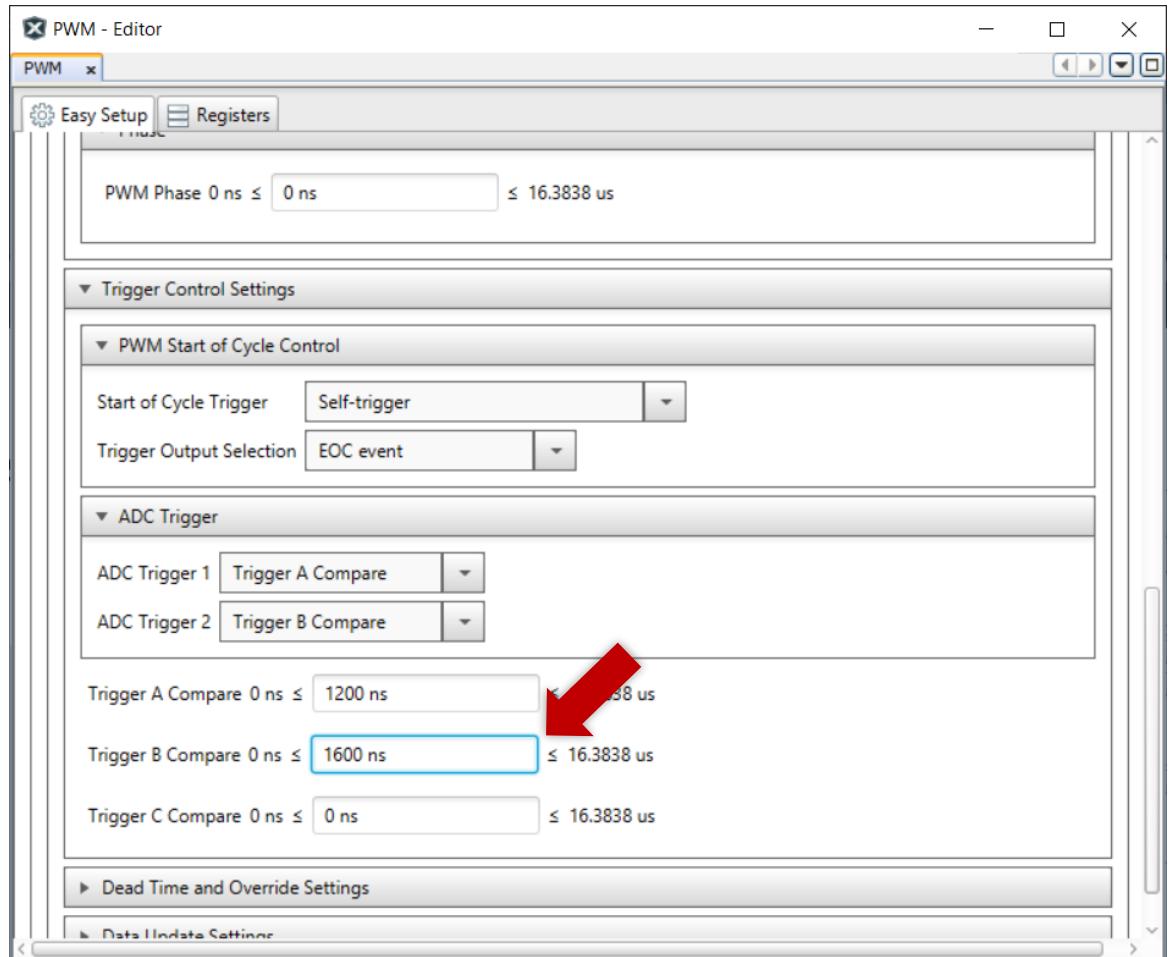
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1
- Enable High Resolution
- Set PWM Frequencies (=500 kHz)
- Set ADC Trigger 1 (=Trigger A Compare)
- Set Trigger A Compare (=1200 ns)
- **Set ADC Trigger 2 (=Trigger B Compare)**



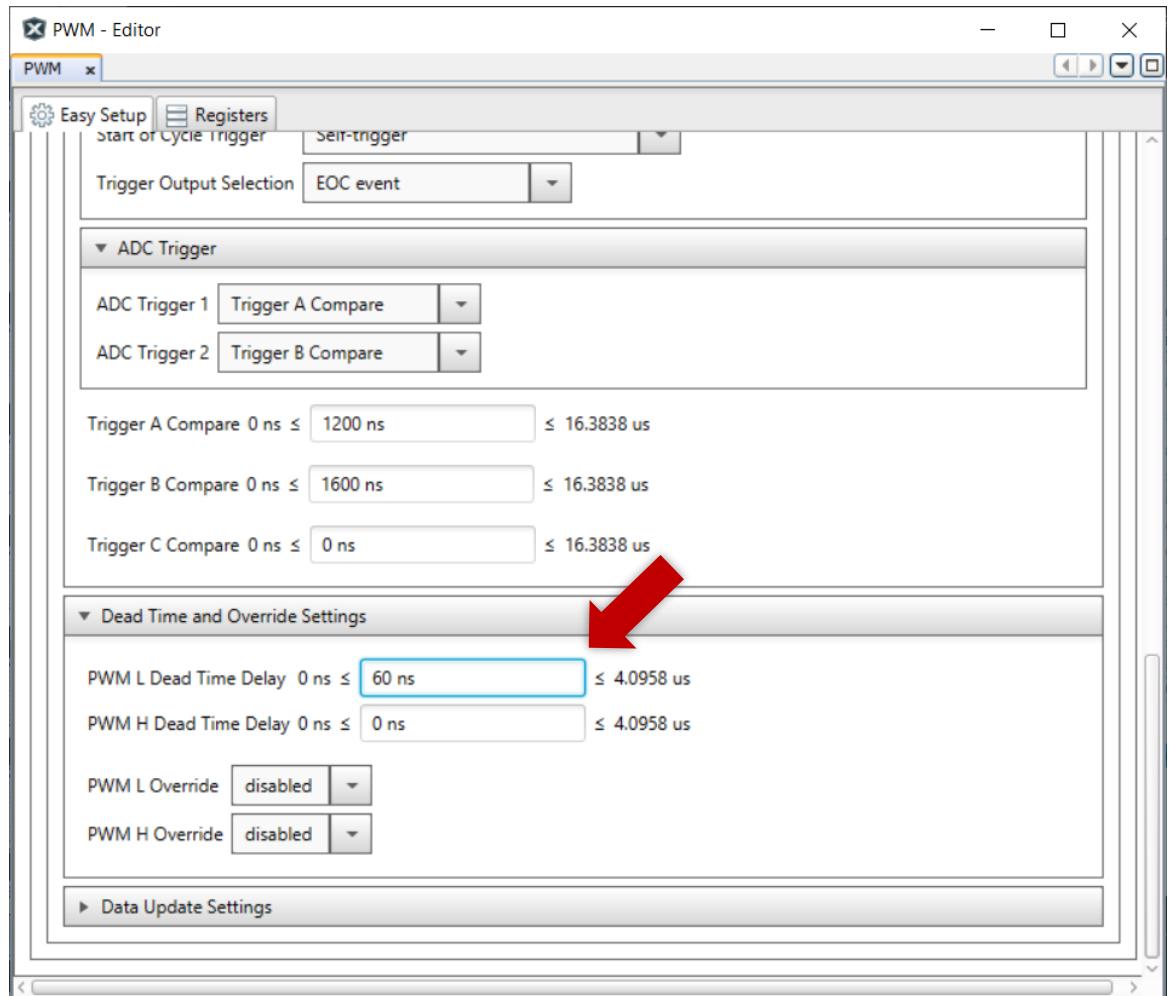
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1
- Enable High Resolution
- Set PWM Frequencies (=500 kHz)
- Set ADC Trigger 1 (=Trigger A Compare)
- Set Trigger A Compare (=1200 ns)
- Set ADC Trigger 2 (=Trigger B Compare)
- Set Trigger B Compare (=1600 ns)



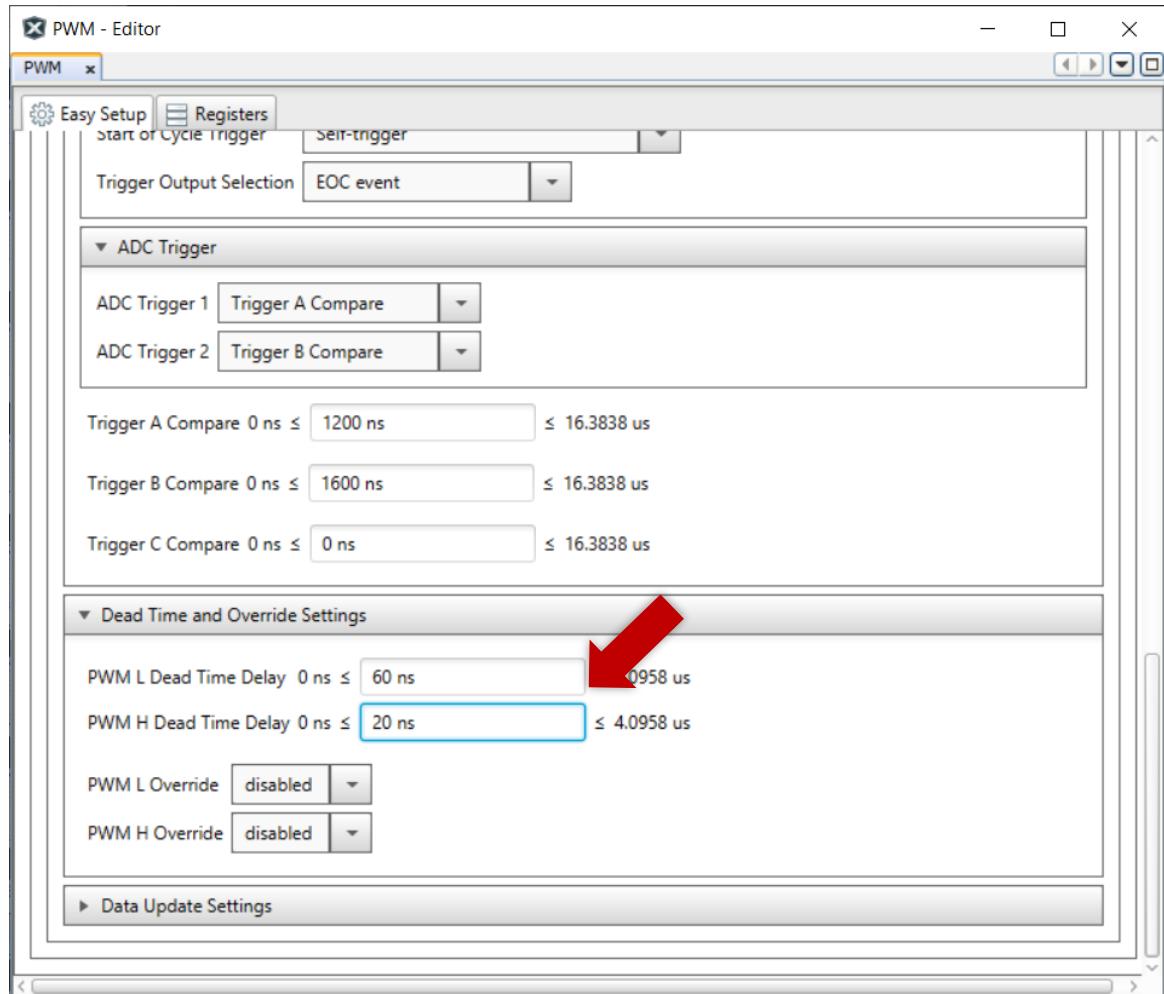
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1
- Enable High Resolution
- Set PWM Frequencies (=500 kHz)
- Set ADC Trigger 1 (=Trigger A Compare)
- Set Trigger A Compare (=1200 ns)
- Set ADC Trigger 2 (=Trigger B Compare)
- Set Trigger B Compare (=1600 ns)
- Set PWM L Dead Time (=60 ns)



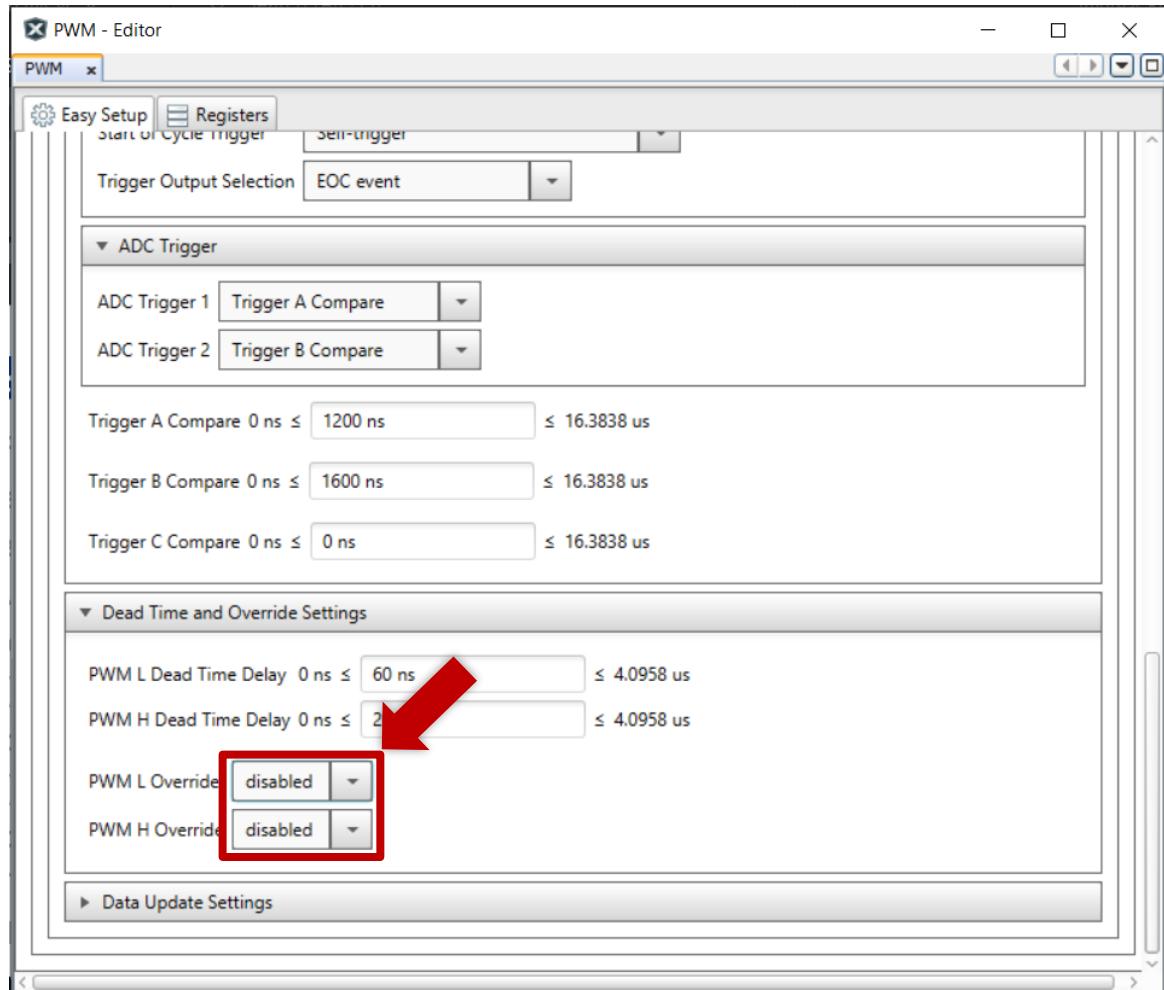
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1
- Enable High Resolution
- Set PWM Frequencies (=500 kHz)
- Set ADC Trigger 1 (=Trigger A Compare)
- Set Trigger A Compare (=1200 ns)
- Set ADC Trigger 2 (=Trigger B Compare)
- Set Trigger B Compare (=1600 ns)
- Set PWM L Dead Time (=60 ns)
- **Set PWM H Dead Time (=20 ns)**



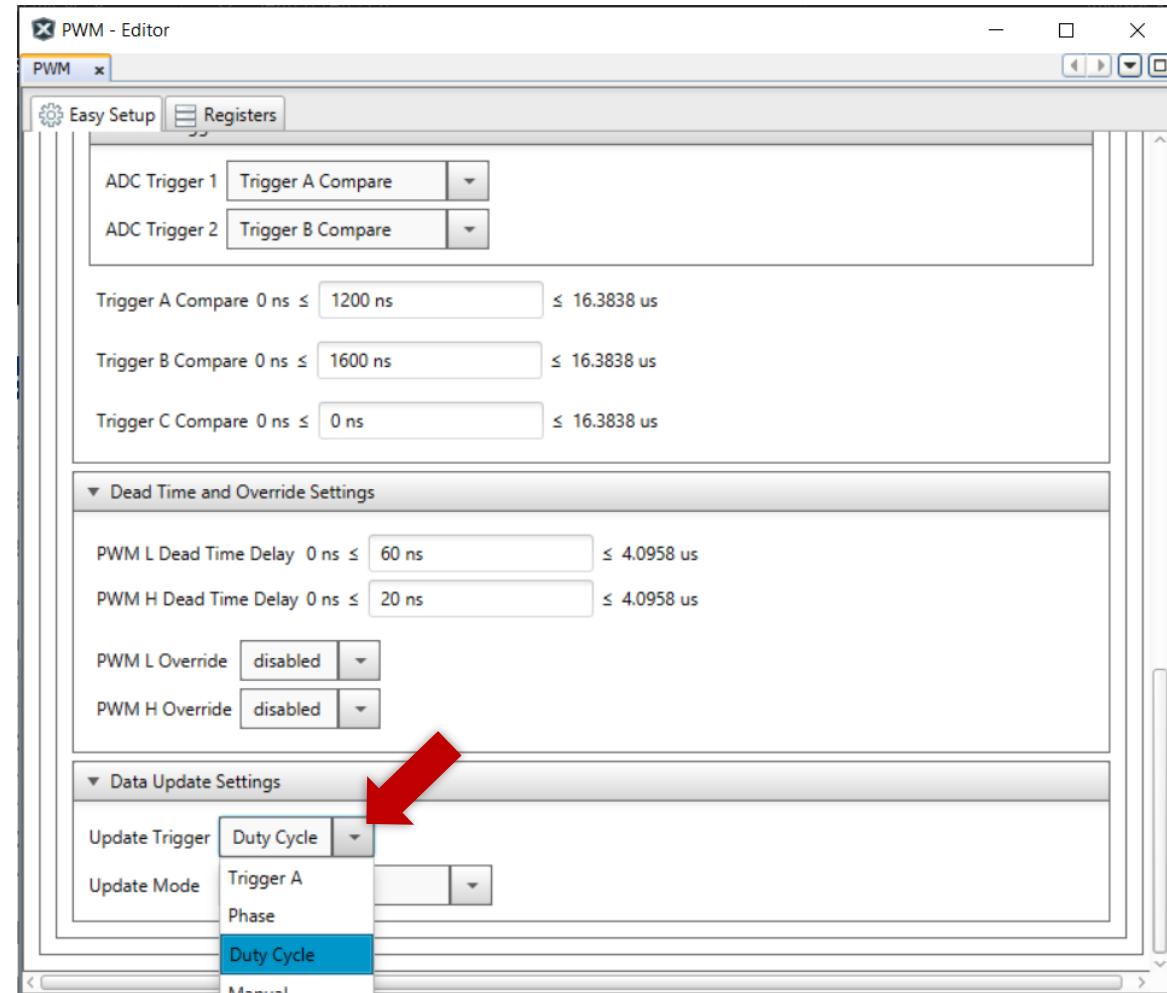
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1
- Enable High Resolution
- Set PWM Frequencies (=500 kHz)
- Set ADC Trigger 1 (=Trigger A Compare)
- Set Trigger A Compare (=1200 ns)
- Set ADC Trigger 2 (=Trigger B Compare)
- Set Trigger B Compare (=1600 ns)
- Set PWM L Dead Time (=60 ns)
- Set PWM H Dead Time (=20 ns)
- **Set PWM L/H Override (=disabled)**



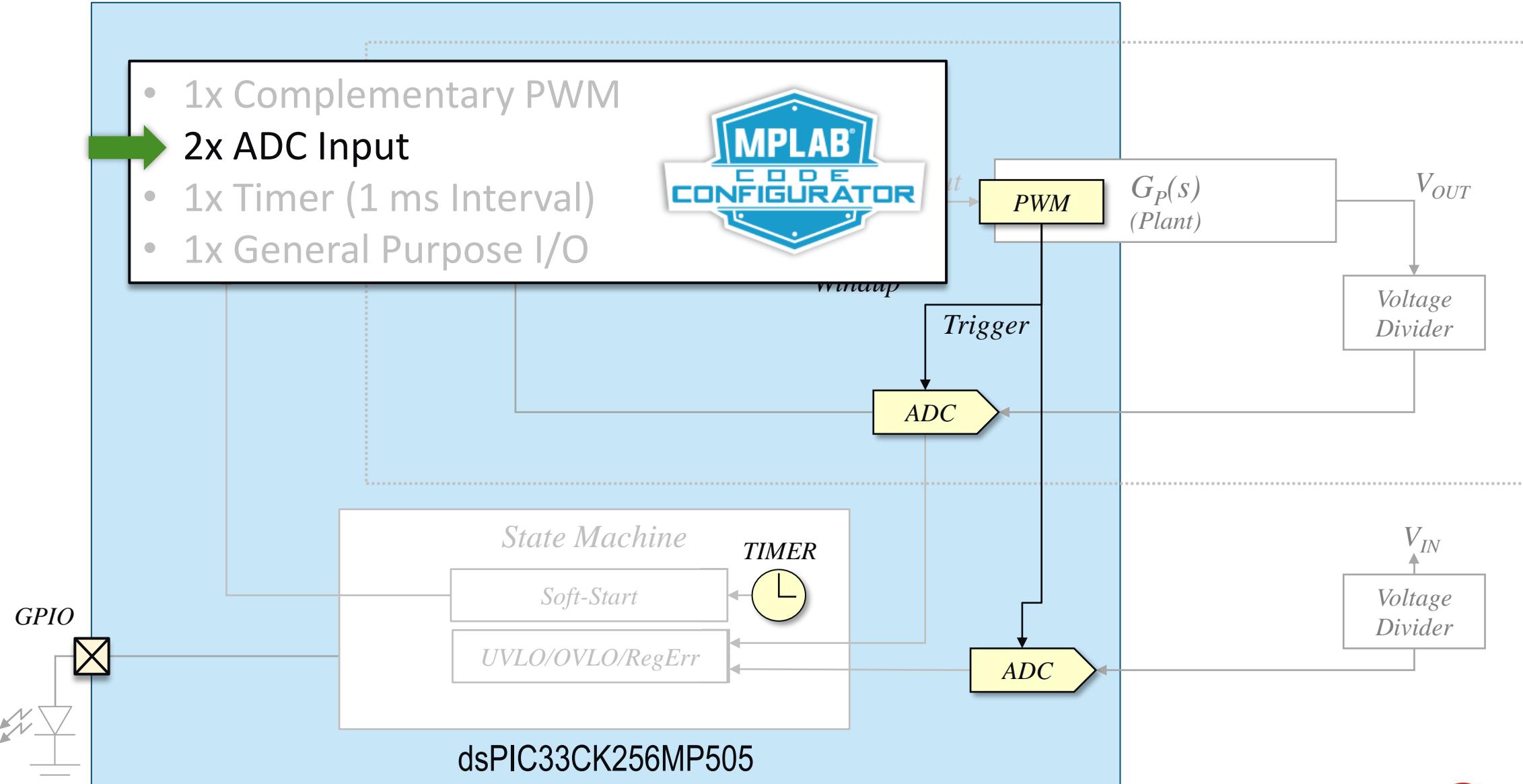
Configuring the PWM

- Set Input Clock (AFPLL0=500 MHz)
- Select Clock Divider Frequency
- Enable PWM Generator #1
- Enable High Resolution
- Set PWM Frequencies (=500 kHz)
- Set ADC Trigger 1 (=Trigger A Compare)
- Set Trigger A Compare (=1200 ns)
- Set ADC Trigger 2 (=Trigger B Compare)
- Set Trigger B Compare (=1600 ns)
- Set PWM L Dead Time (=60 ns)
- Set PWM H Dead Time (=20 ns)
- Set PWM L/H Override (=disabled)
- **Set Data Update Trigger (=Duty Cycle)**

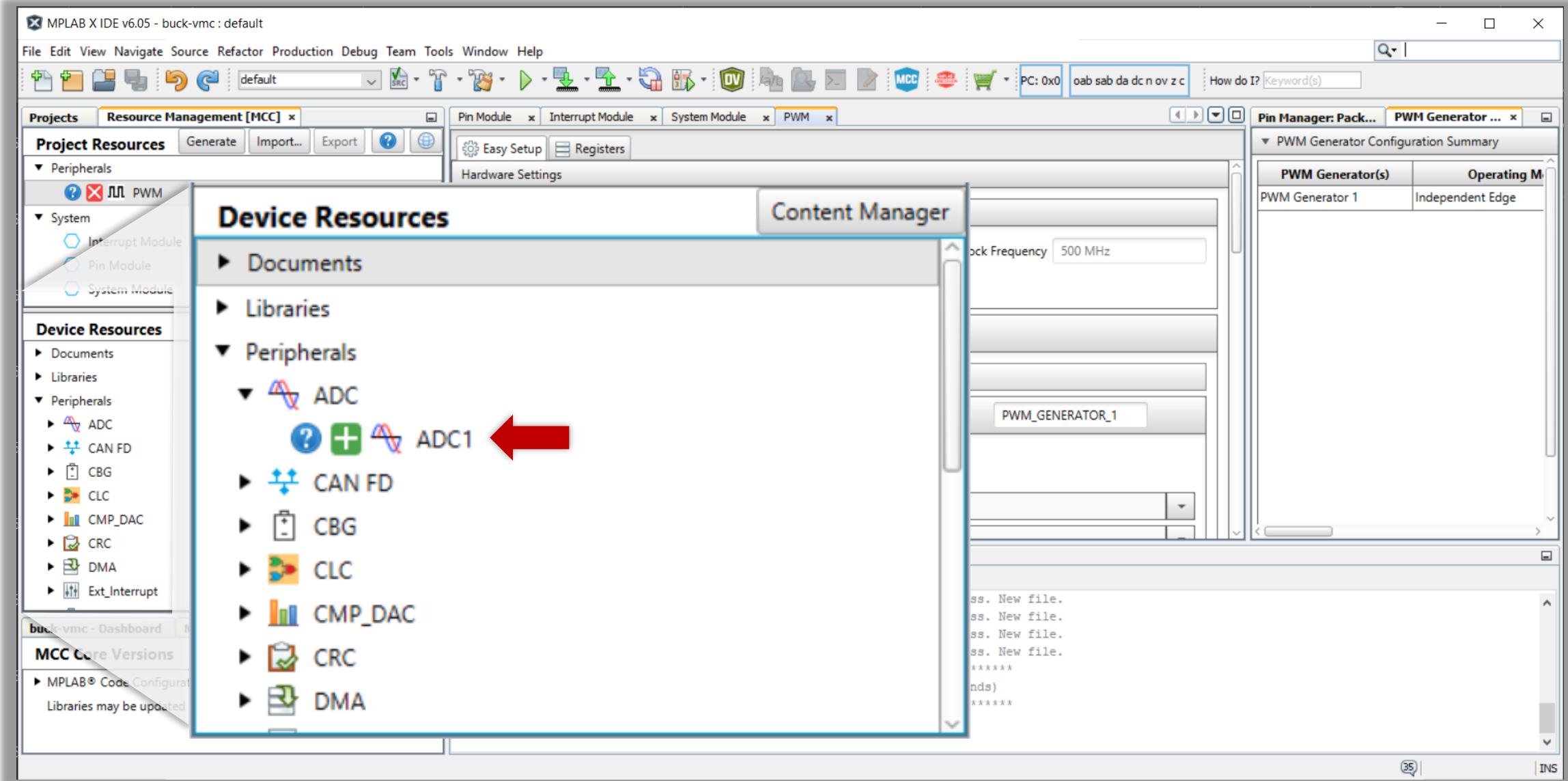


Requirements

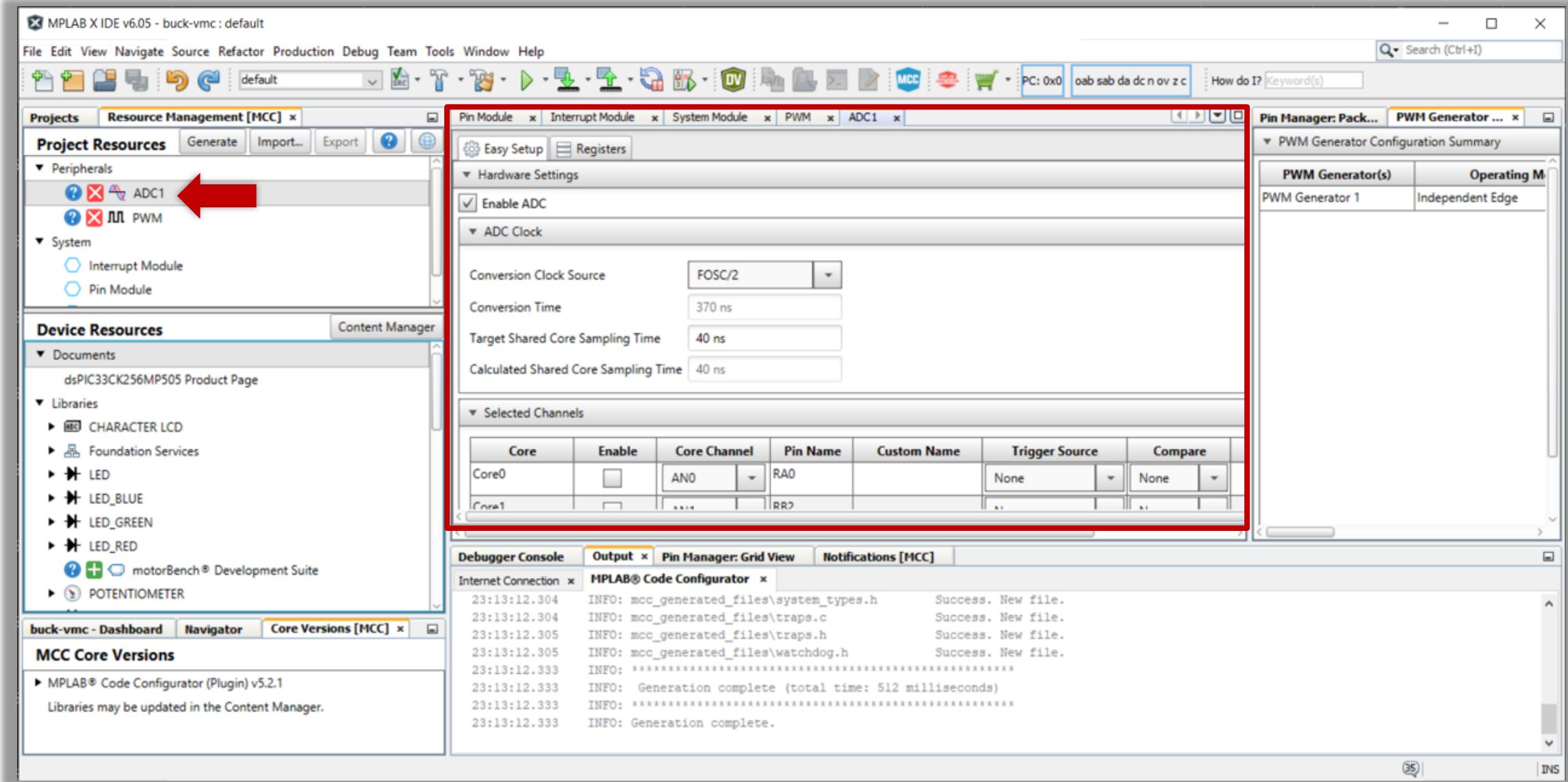
- 1x Complementary PWM
- **2x ADC Input**
- 1x Timer (1 ms Interval)
- 1x General Purpose I/O



Configuring the ADC

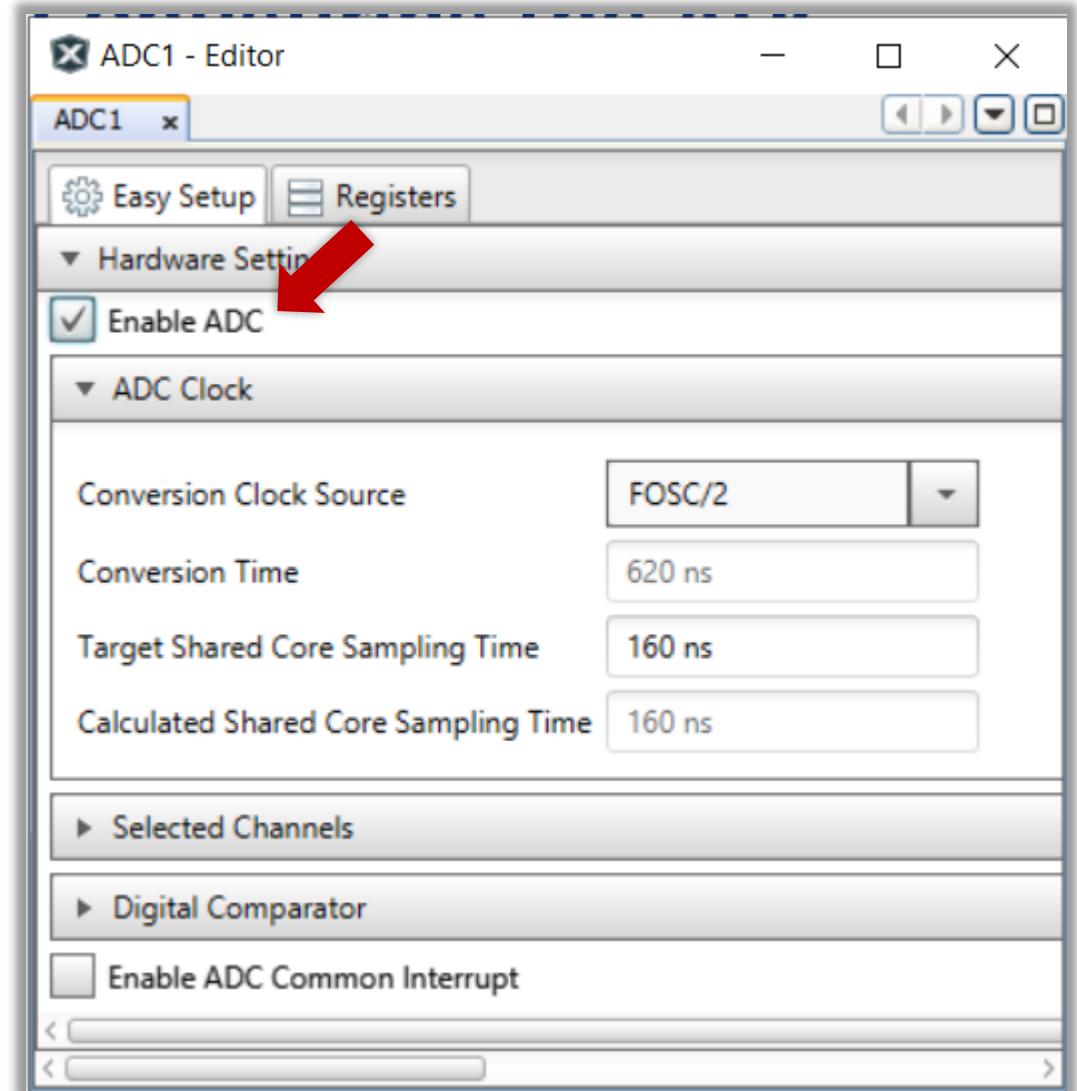


Configuring the ADC



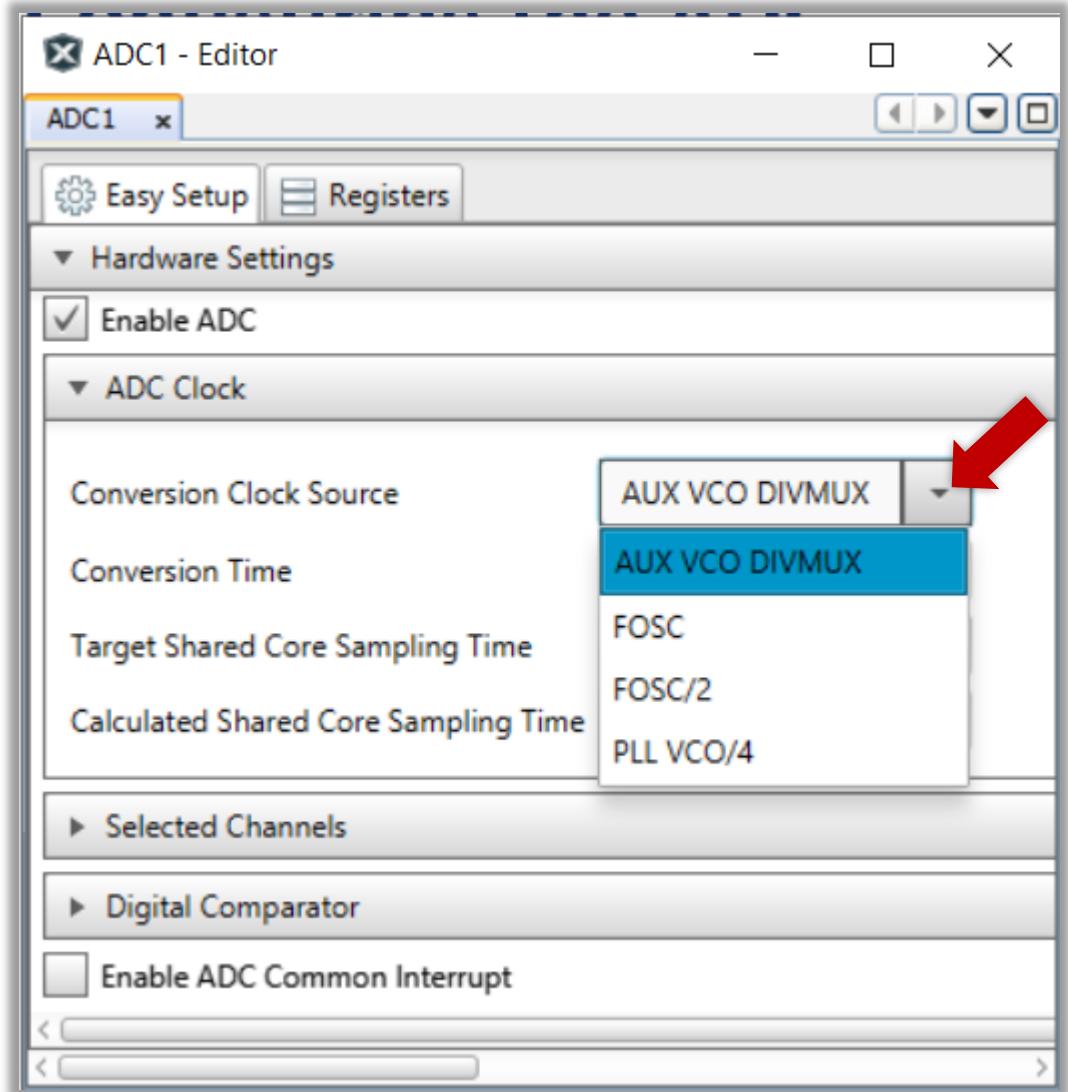
Configuring the ADC

- Enable ADC



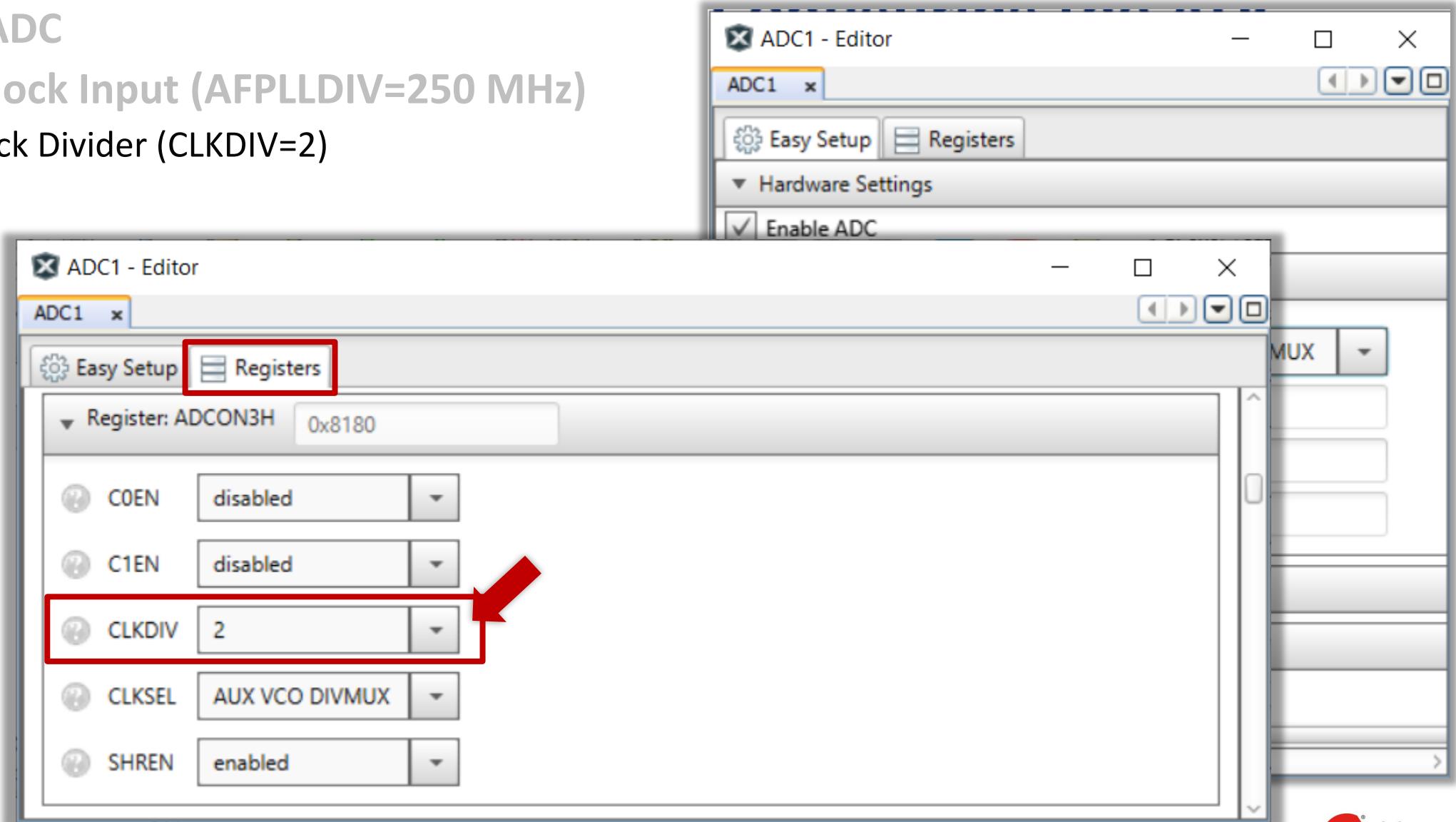
Configuring the ADC

- Enable ADC
- Select Clock Input (AFPLL DIV=250 MHz)



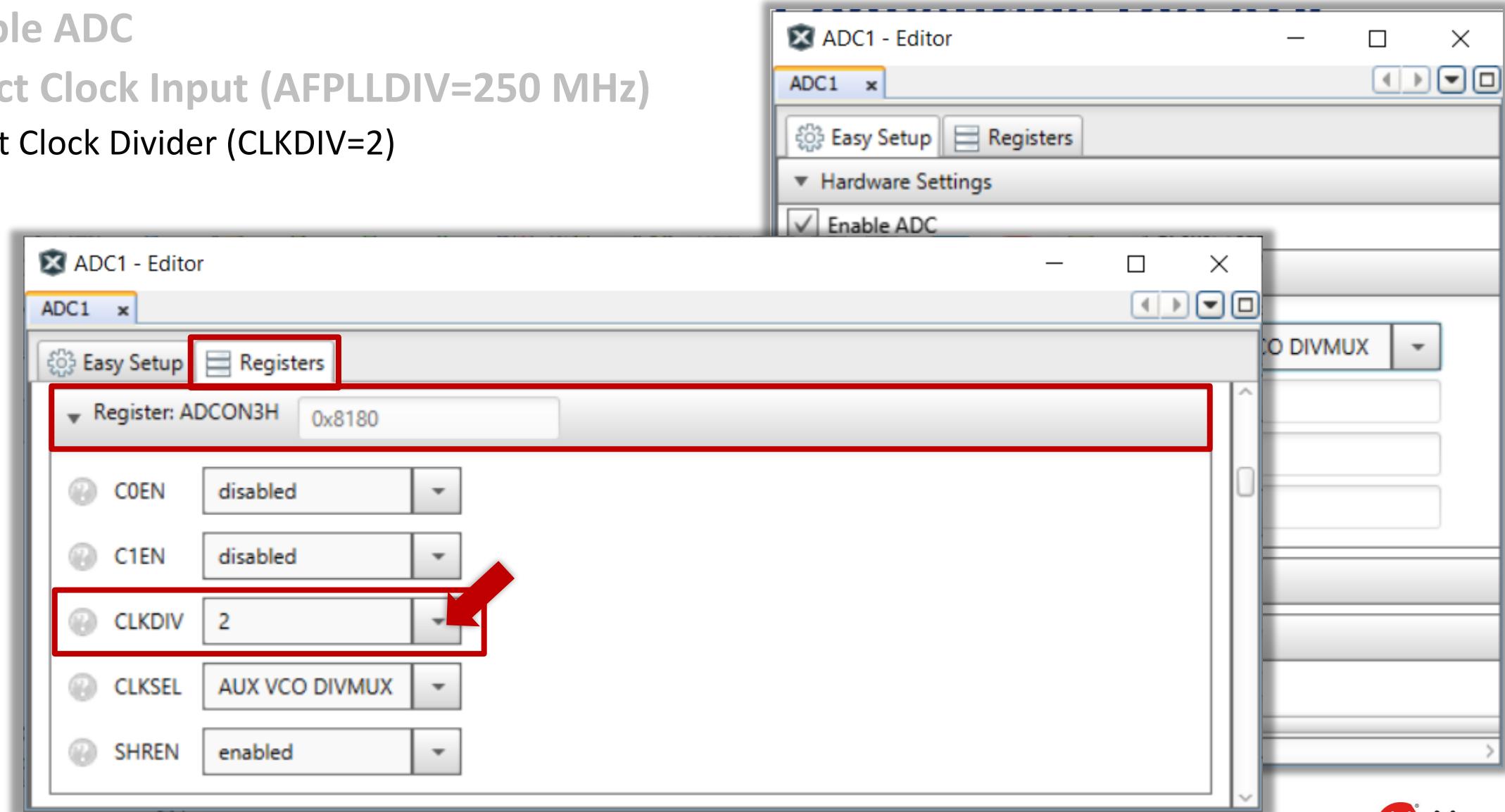
Configuring the ADC

- Enable ADC
- Select Clock Input (AFPLL DIV=250 MHz)
 - Set Clock Divider (CLKDIV=2)



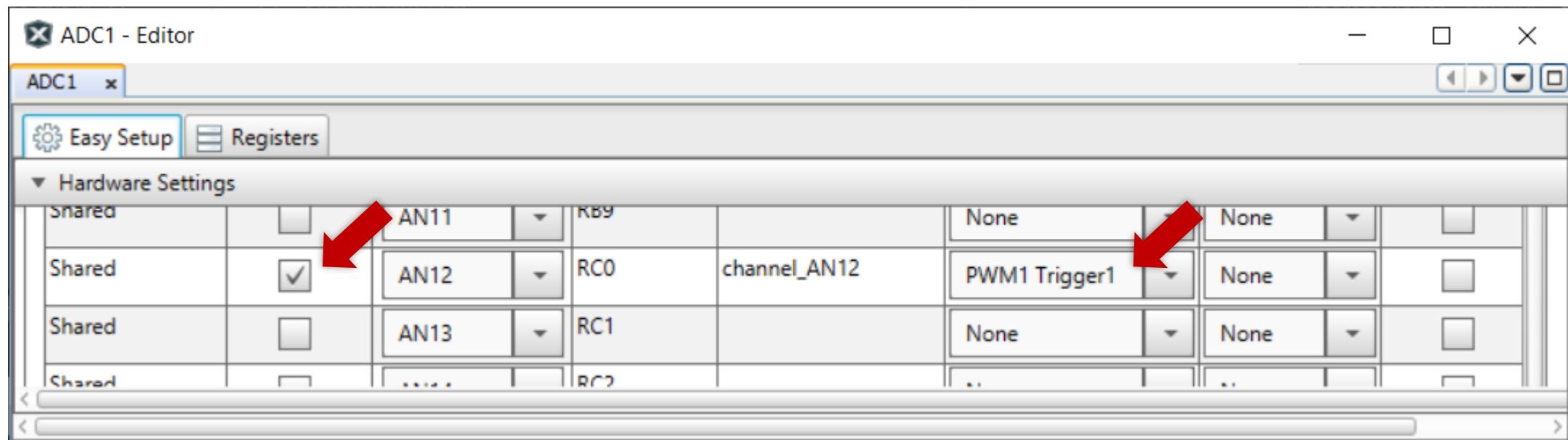
Configuring the ADC

- Enable ADC
- Select Clock Input (AFPLL DIV=250 MHz)
 - Set Clock Divider (CLKDIV=2)



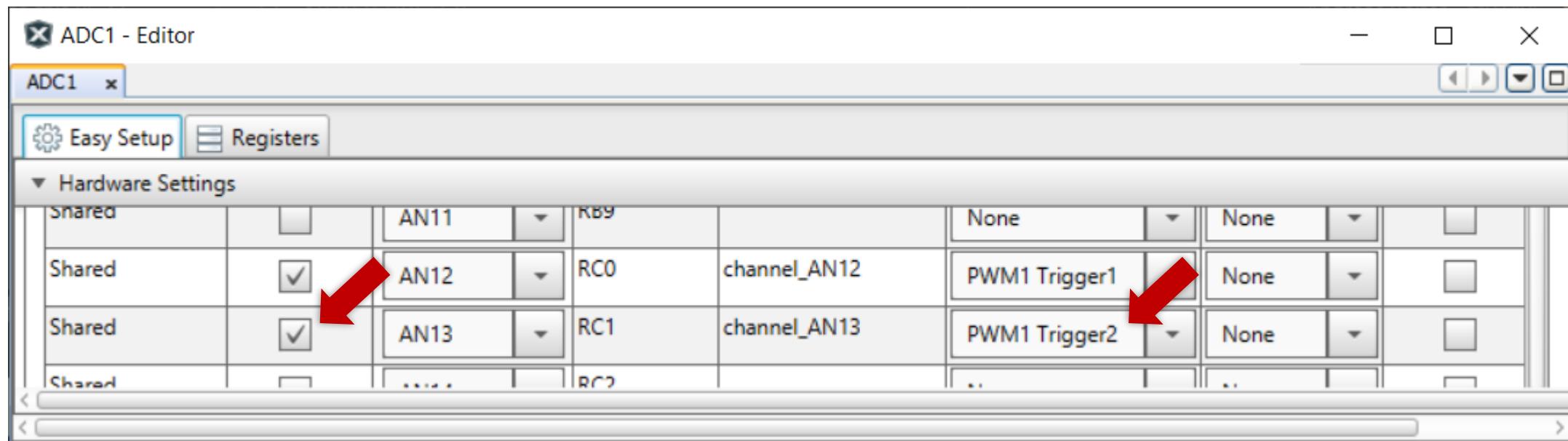
Configuring the ADC

- Enable ADC
- Select Clock Input (AFPLL DIV=250 MHz)
- Enable ADC Input AN12, Select Trigger Source (=PWM1 Trigger 1)



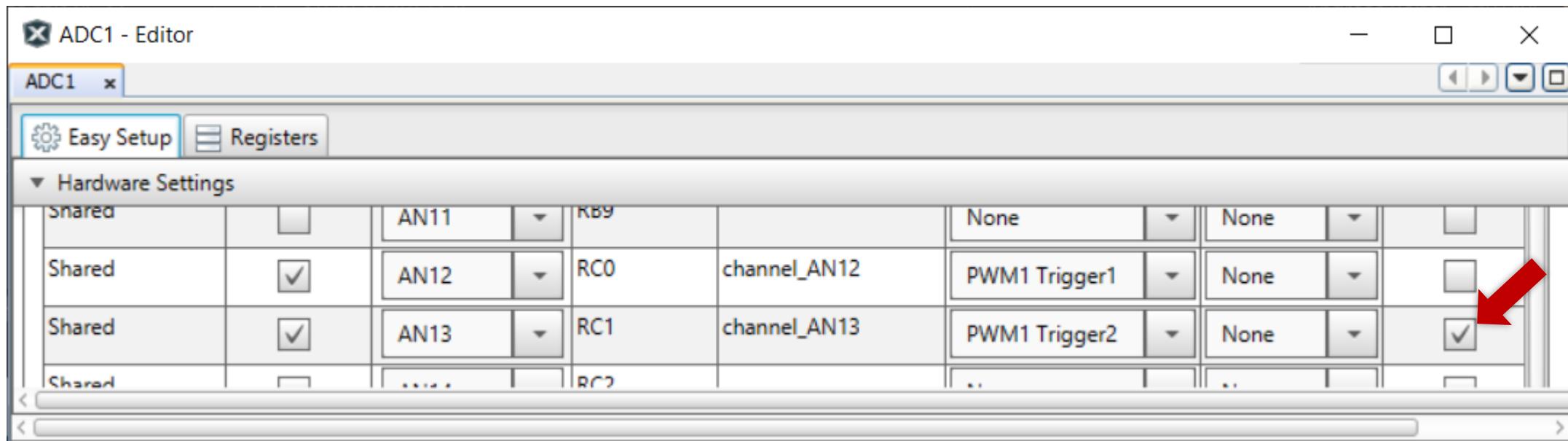
Configuring the ADC

- Enable ADC
- Select Clock Input (AFPLL DIV=250 MHz)
- Enable ADC Input AN12, Select Trigger Source (=PWM1 Trigger 1)
- **Enable ADC Input AN13, Select Trigger Source (=PWM1 Trigger 2)**



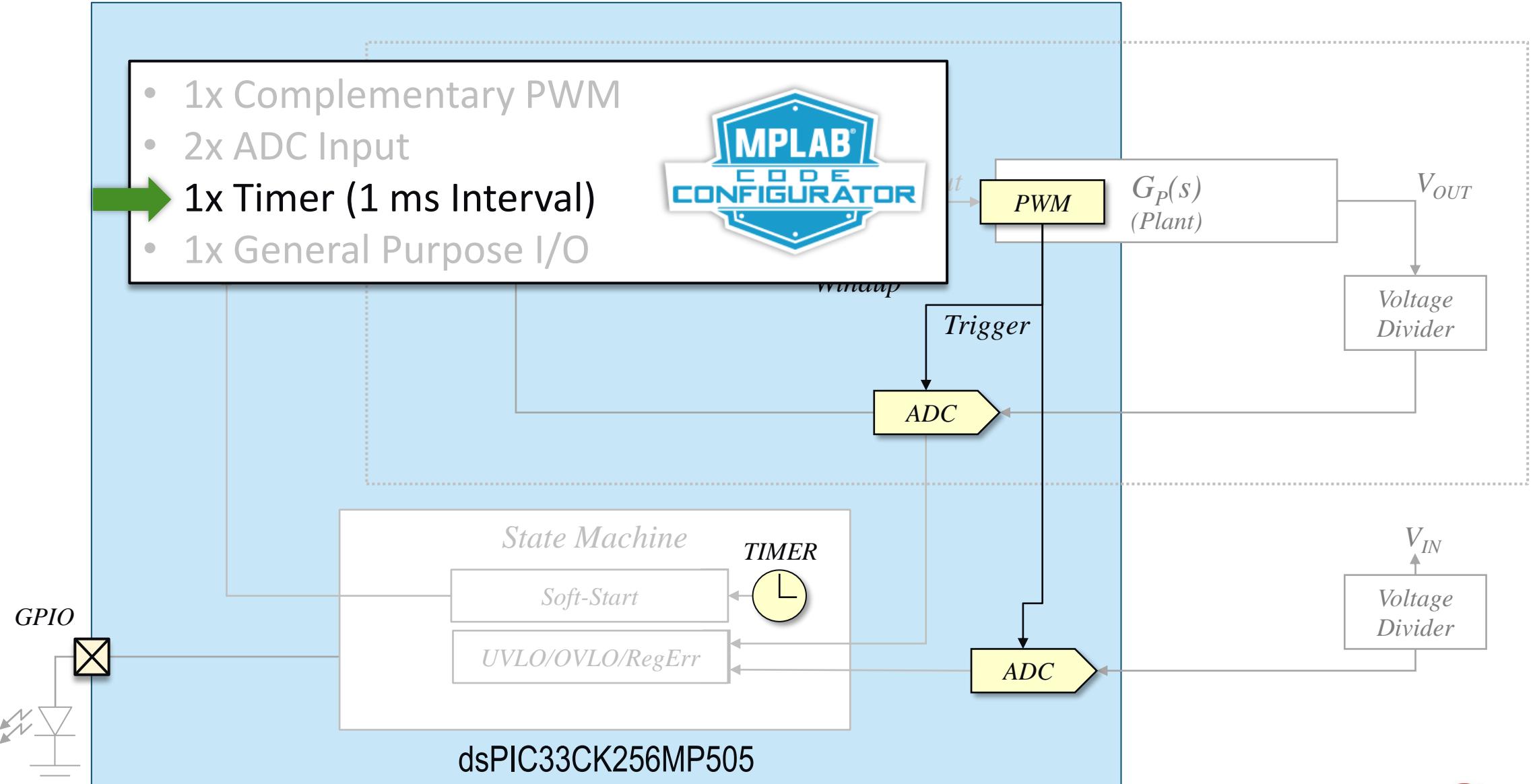
Configuring the ADC

- Enable ADC
- Select Clock Input (AFPLL DIV=250 MHz)
- Enable ADC Input AN13
- Select Trigger Source (=PWM1 Trigger 2)
- Enable ADC Input AN13 and Select Trigger Source (=PWM1 Trigger 2)
- **Enable AN13 Interrupt**

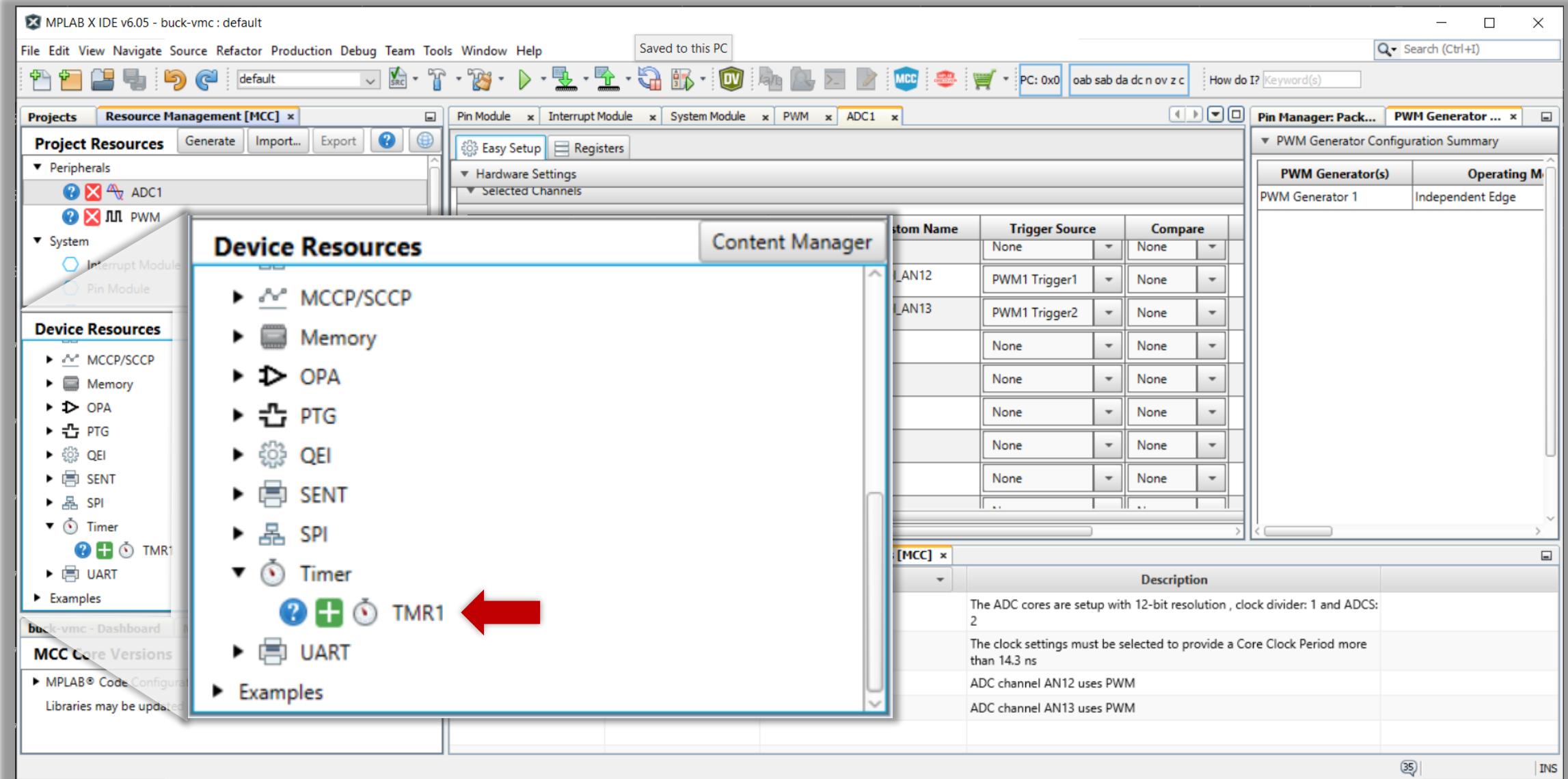


Requirements

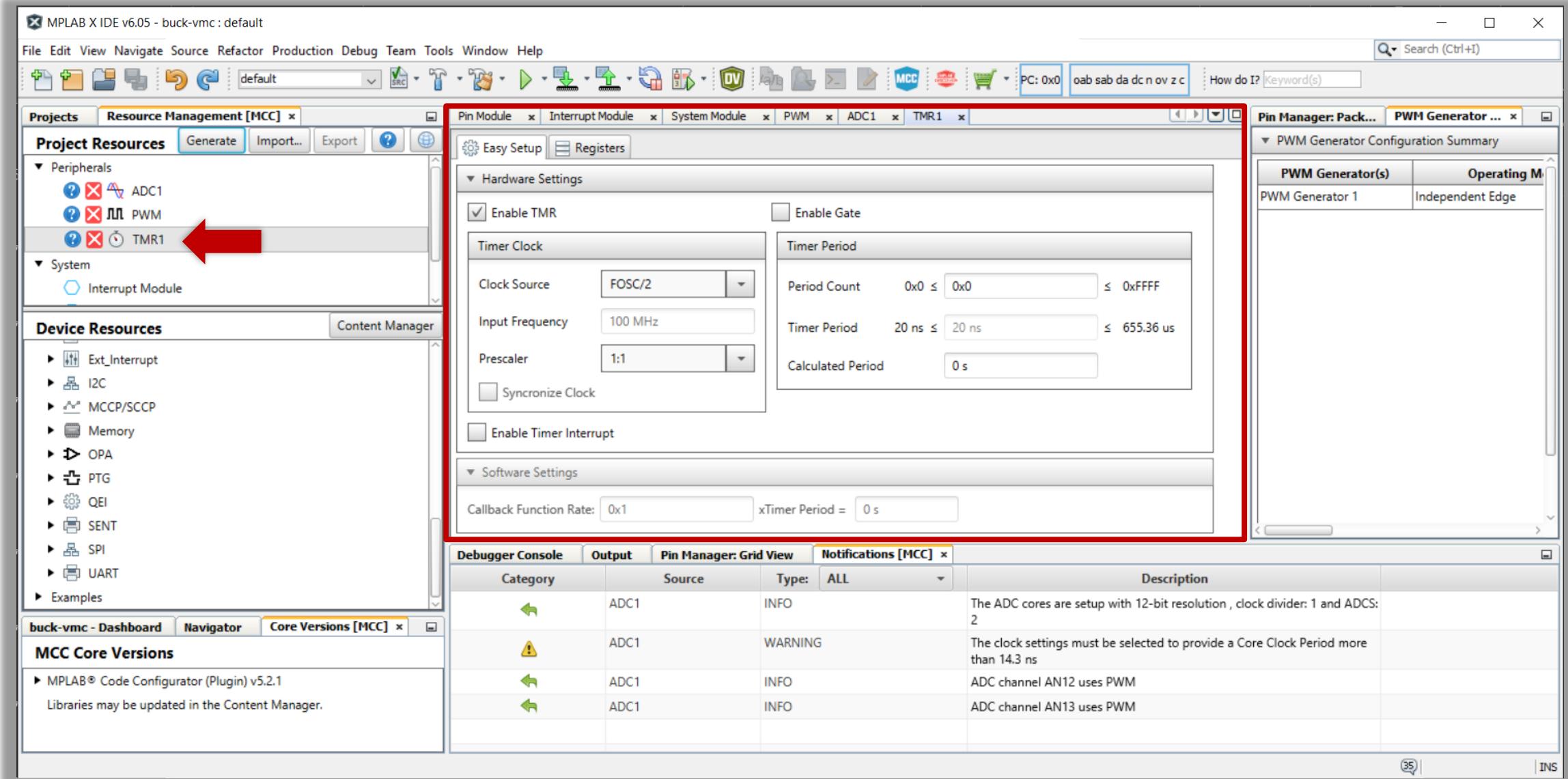
- 1x Complementary PWM
- 2x ADC Input
- 1x Timer (1 ms Interval)
- 1x General Purpose I/O



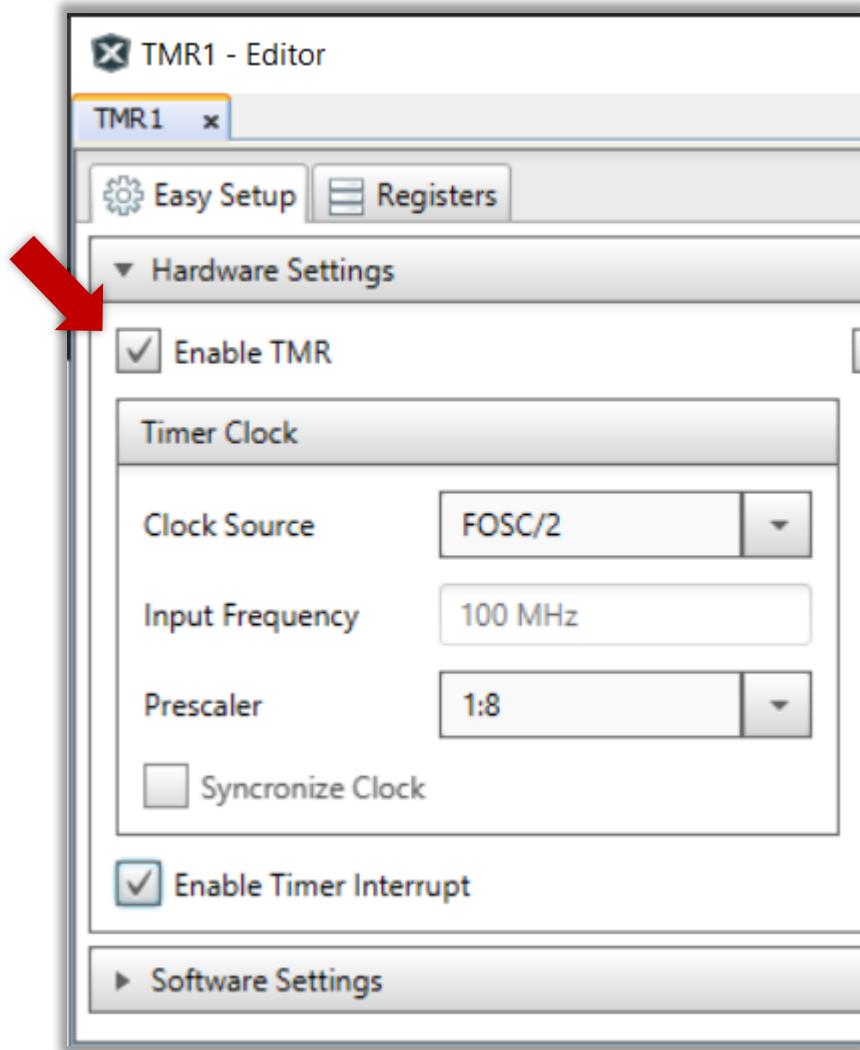
Configuring the Timer



Configuring the Timer

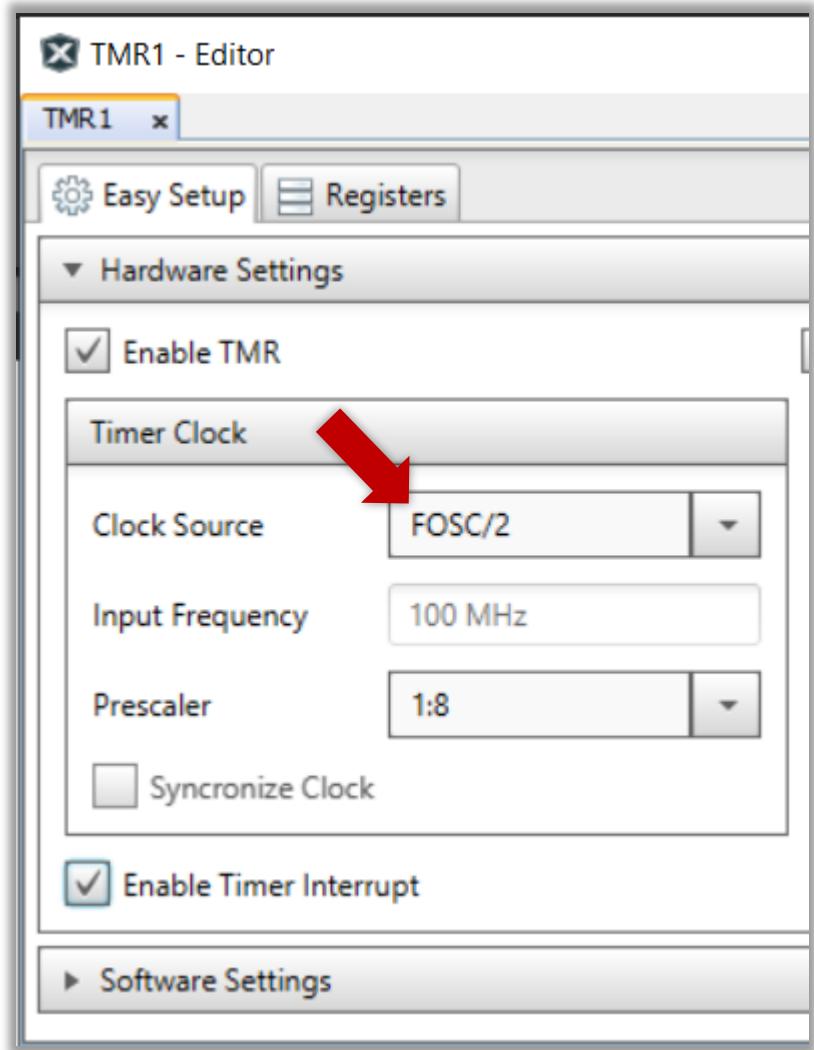


Configuring the Timer



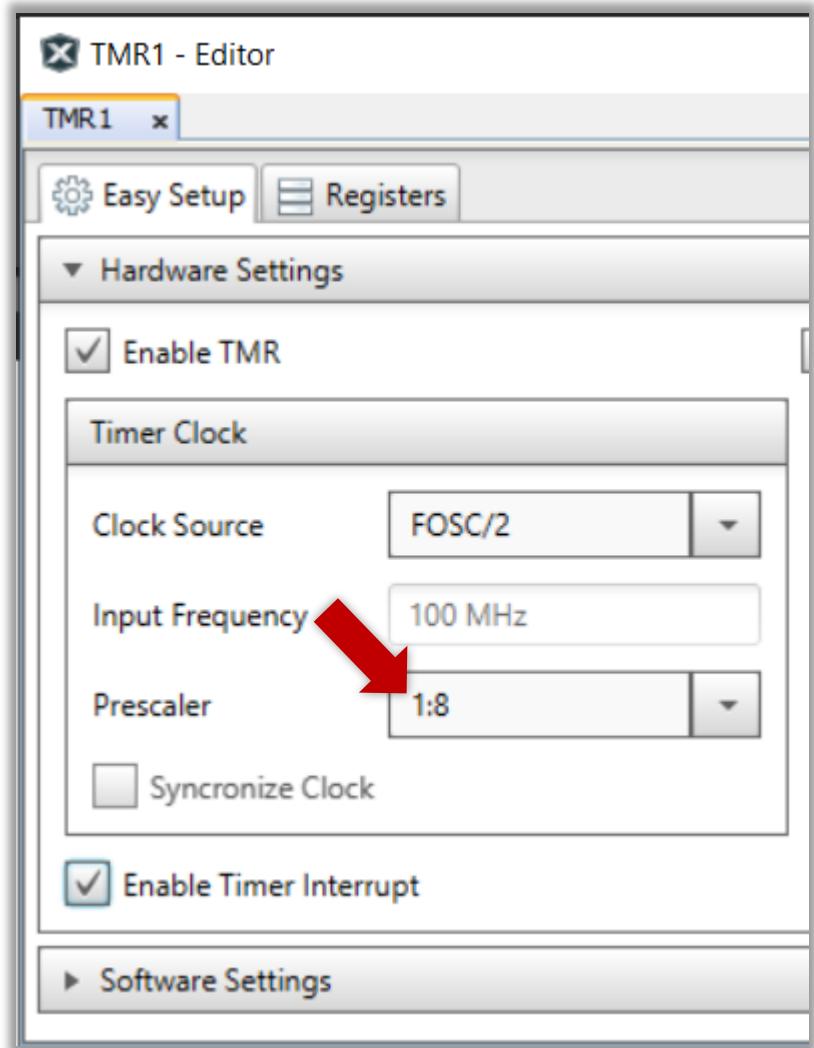
- Enable Timer 1

Configuring the Timer



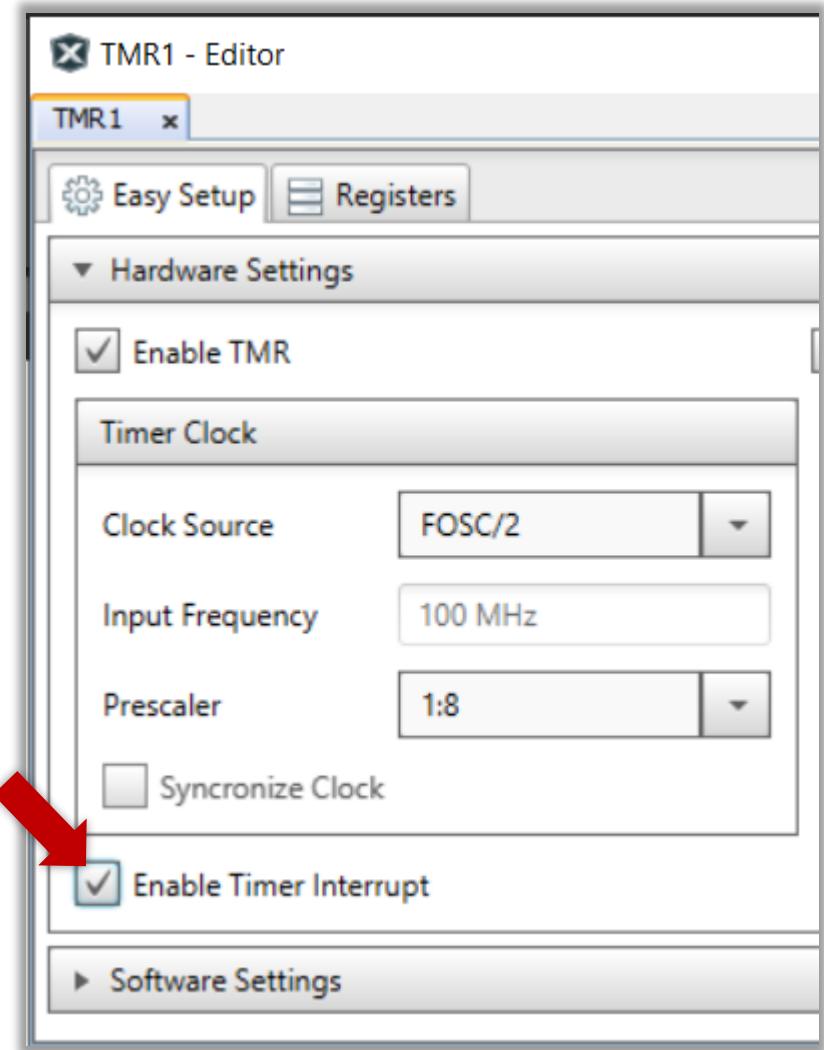
- Enable Timer 1
- Set Clock Source to FOSC/2

Configuring the Timer



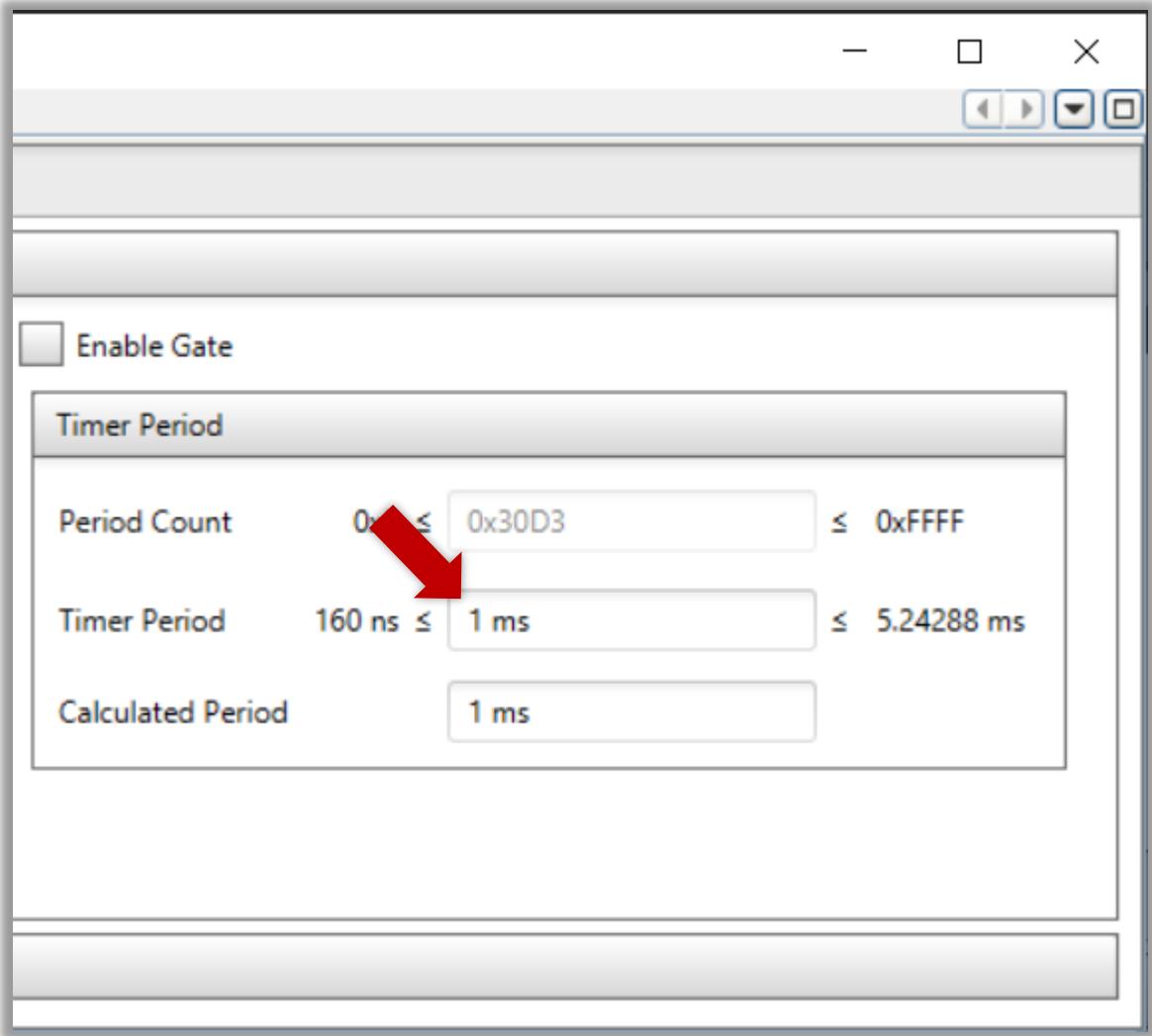
- Enable Timer 1
- Set Clock Source to FOSC/2
- Set Prescaler (=1:8)

Configuring the Timer



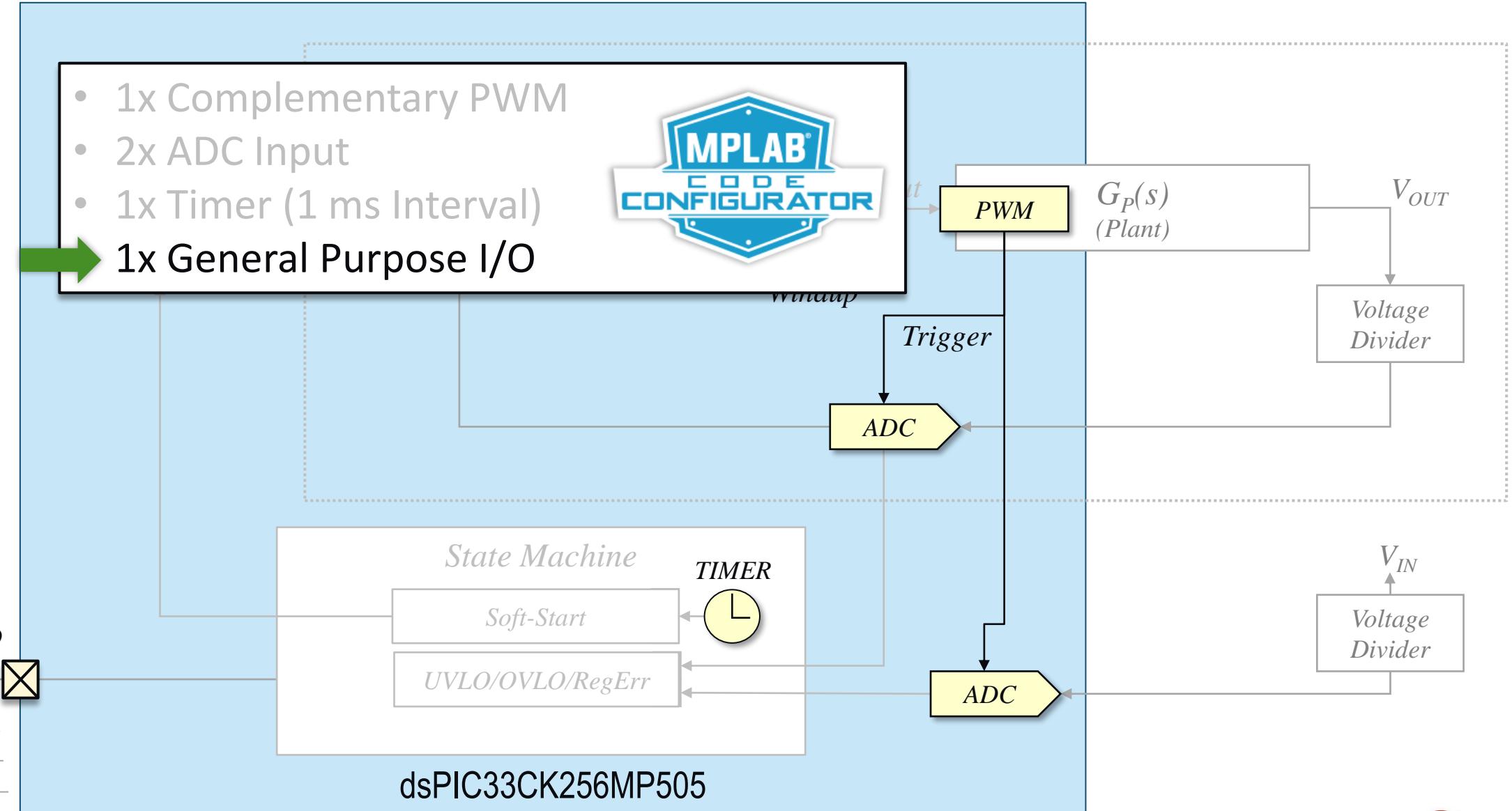
- Enable Timer 1
- Set Clock Source to FOSC/2
- Set Prescaler (=1:8)
- **Enable Timer Interrupt**

Configuring the Timer



- Enable Timer 1
- Set Clock Source to FOSC/2
- Set Prescaler (=1:8)
- Enable Timer Interrupt
- **Set Timer Period (=1 ms)**

Requirements



Configuring the General Purpose I/O

The screenshot shows the MPLAB X IDE interface with the following components:

- Top Bar:** File, Edit, View, Navigate, Source, Refactor, Production, Debug, Team, Tools, Window, Help.
- Toolbar:** Includes icons for file operations, build, simulation, and tools like MCC and PWM.
- Projects & Resource Manager:** Shows Peripherals (ADC1, PWM, TMR1) and System modules.
- Pin Manager (Grid View):** Shows pin assignments for RB6, RB8, RB9, RB14, RB15, RC0, and RC1. RB6 is highlighted with a red border.
- Pin Manager (Package View):** Shows the pin layout for the QFN48 package, highlighting pins 1 through 37.
- Grid View:** A detailed grid view of all pins across Port A, Port B, Port C, and Port D. It shows various module assignments like PWM1-H, PWM1-L, and GPIO. A red box highlights the Port B section.
- MCC Core Versions:** Shows the version of the MPLAB® Code Configurator (Plugin) v5.2.

Configuring the General Purpose I/O

Pin Module - Editor

Pin Module x

Selected Package : QFN48

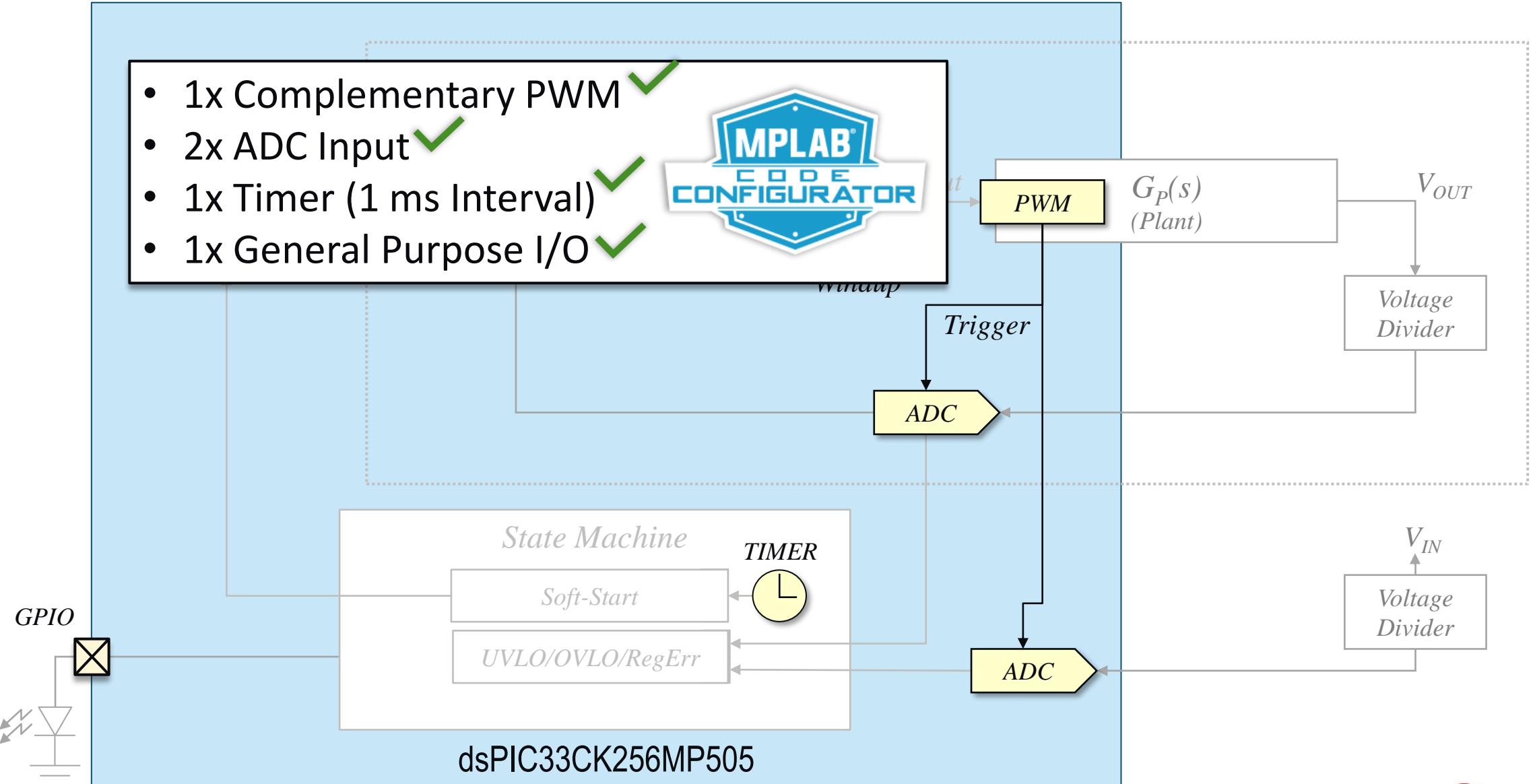
Easy Setup Registers

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	WPD	OD	IOC
RB6	Pin Module	GPIO	DGBLED	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB8	ICD	PGD1		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB9	ICD	PGC1		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB14	PWM	PWM1-H		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB15	PWM	PWM1-L		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RC0	ADC1	AN12	channel_AN	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RC1	ADC1	AN13	channel_AN	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none

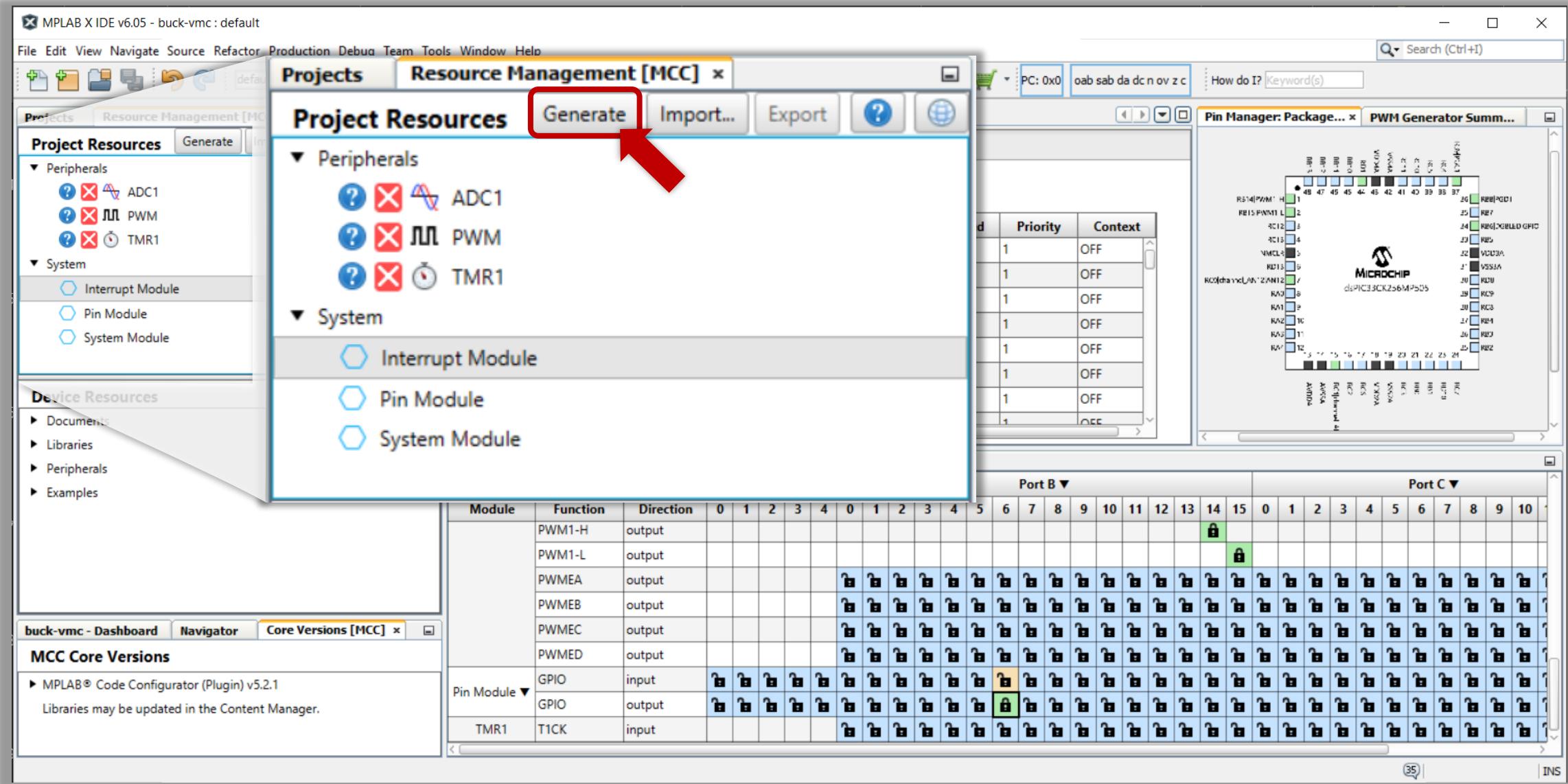
A red arrow points to the "Custom Name" column for the RB6 row, highlighting the cell containing "DGBLED".

Requirements

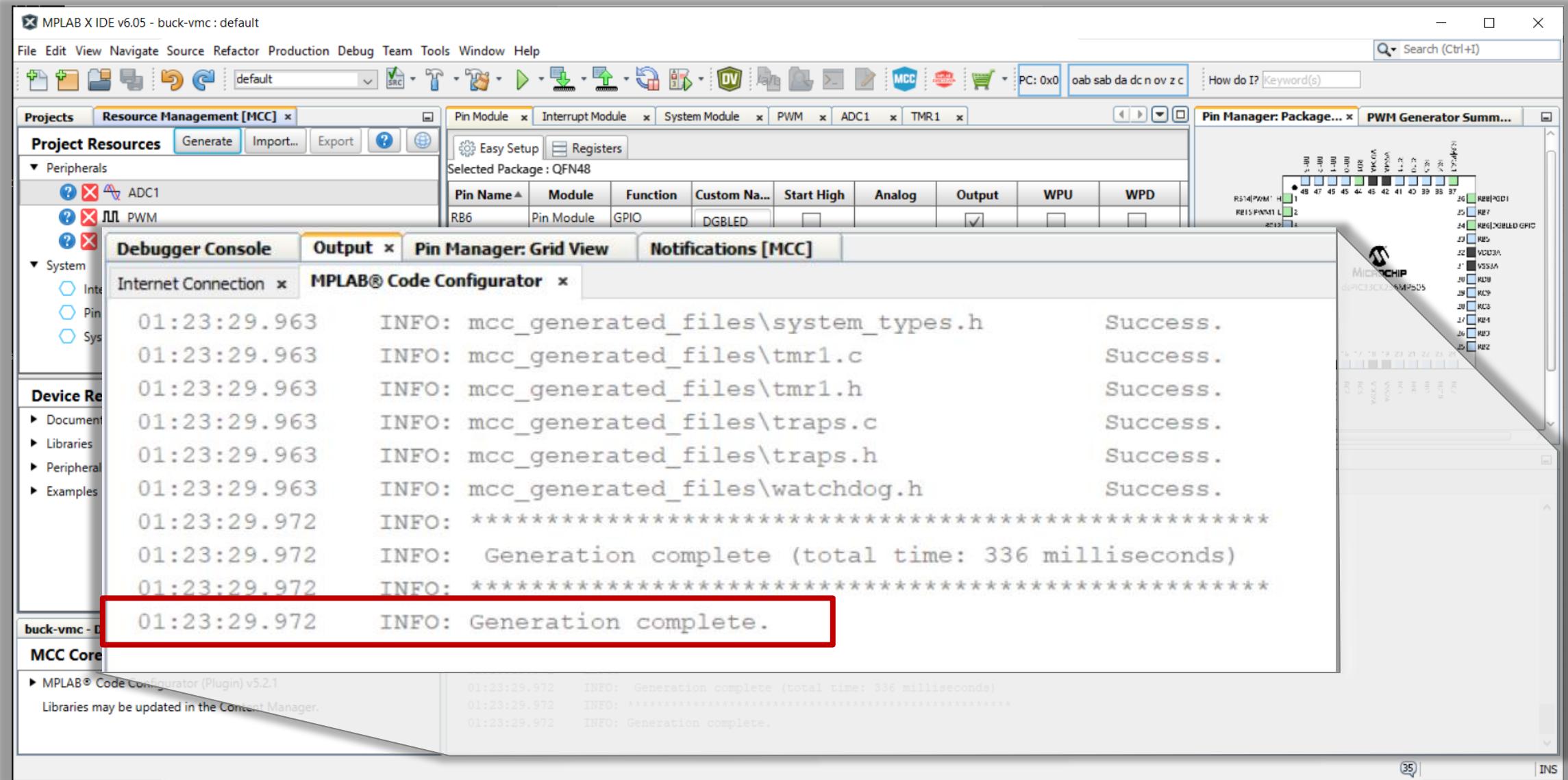
- 1x Complementary PWM ✓
- 2x ADC Input ✓
- 1x Timer (1 ms Interval) ✓
- 1x General Purpose I/O ✓



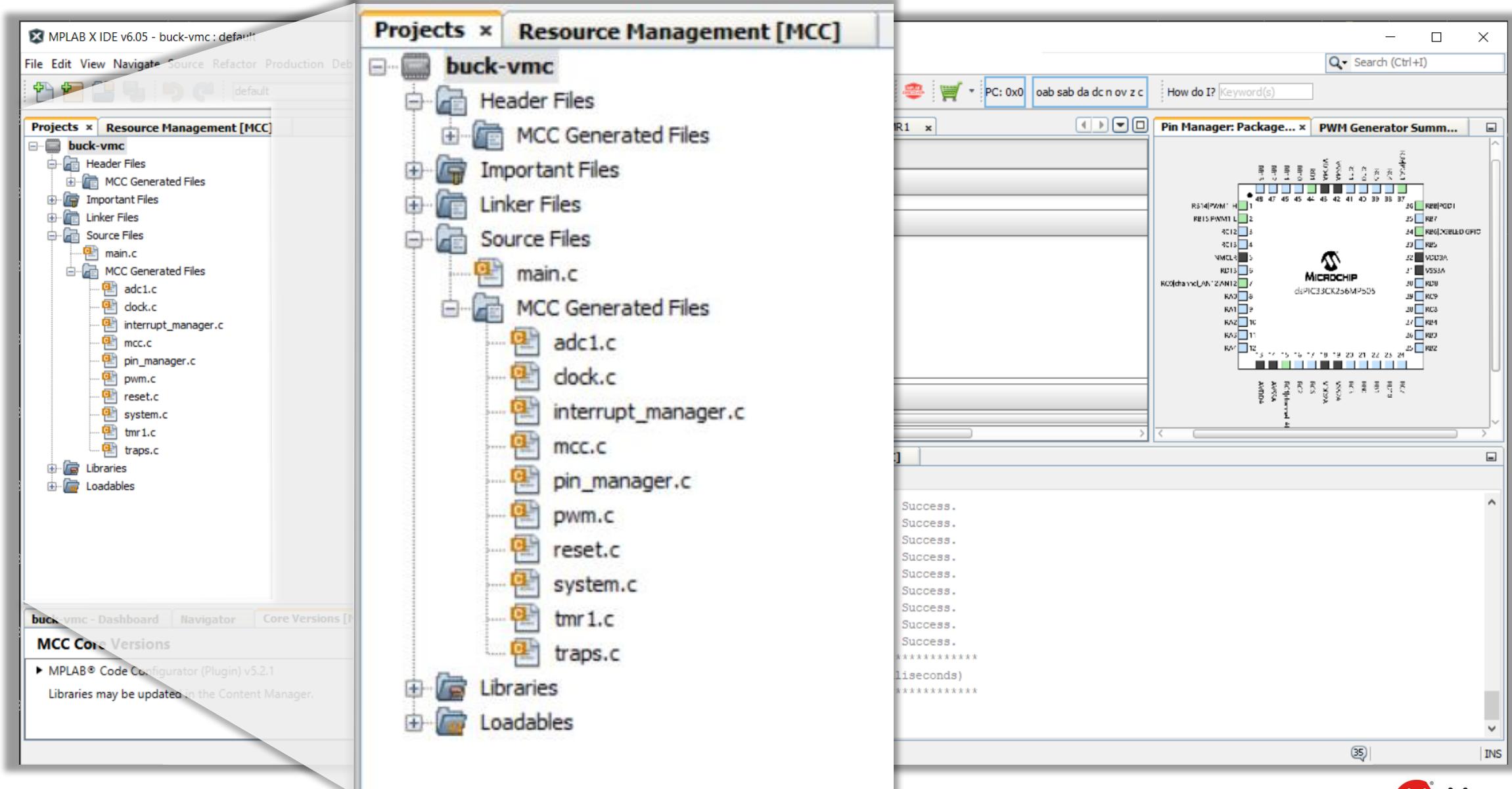
Time to Generate Code!



Time to Generate Code!



Time to Generate Code!





Coding Time !

Including MCC header files in main.c

```
/**  
 * Section: Included Files  
 */  
  
#include "mcc_generated_files/system.h"  
#include "mcc_generated_files/pin_manager.h"  
#include "mcc_generated_files/pwm.h"  
#include "mcc_generated_files/adcl.h"  
#include "mcc_generated_files/tmr1.h"  
  
[...]  
  
int main(void)  
{  
    // initialize the device  
    SYSTEM_Initialize();  
    while (1)  
    {  
        // Add your application code  
    }  
    return 1;  
}
```

Declare User Variables in main.c

```
/**  
 * Section: Included Files  
 */  
#include "mcc_generated_files/system.h"  
#include "mcc_generated_files/pin_manager.h"  
#include "mcc_generated_files/pwm.h"  
#include "mcc_generated_files/adcl.h"  
#include "mcc_generated_files/tmr1.h"  
  
// Define Global Variables:  
int adc_vin = 0;    // Scaling: 155.1 steps/v  
int adc_vbuck = 0;   // Scaling: 620.5 steps/v  
int ref_vbuck = 0;   // Scaling: 620.5 steps/v  
[...]  
int main(void)  
{  
    // initialize the device  
    SYSTEM_Initialize();  
    while (1)  
    {  
        // Add your application code  
    }  
    return 1;  
}
```

Add Timer Interrupt Service Routine to main.c

[...]

```
void TMR1_Int(void)
{
    static int led_tmr = 0;

    // blink LED @ 1Hz
    if(++led_tmr>=500) {
        led_tmr = 0;
        DGBLED_Toggle();
    }
}

int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    while (1)
    {
        // Add your application code
    }
    return 1;
}
```

Add Timer Interrupt Service Routine to main.c

```
[...]
    if(++led_tmr>=500)  {
        led_tmr = 0;
        DGBLED_Toggle();
    }
}

void AN13_Int(void)
{
    adc_vin = ADCBUF12;
    adc_vbuck = ADCBUF13;
}

int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    while (1)
    {
        // Add your application code
    }
    return 1;
}
```

Set Interrupt Handlers in `main()` in `main.c`

[...]

```
void AN13_Int(void)
{
    adc_vin = ADCBUF12;
    adc_vbuck = ADCBUF13;
}

int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    ADC1_Setchannel_AN13InterruptHandler(&AN13_Int);
    TMR1_SetInterruptHandler(&TMR1_Int);
    while (1)
    {
        // Add your application code
    }
    return 1;
}
```

Testing Time !



Give it a try...

- **Build the Project** 
 - If the Build was successful ...
 - Or check error message and fix what's broken

- **Connect the DPSK3 via USB to your PC**

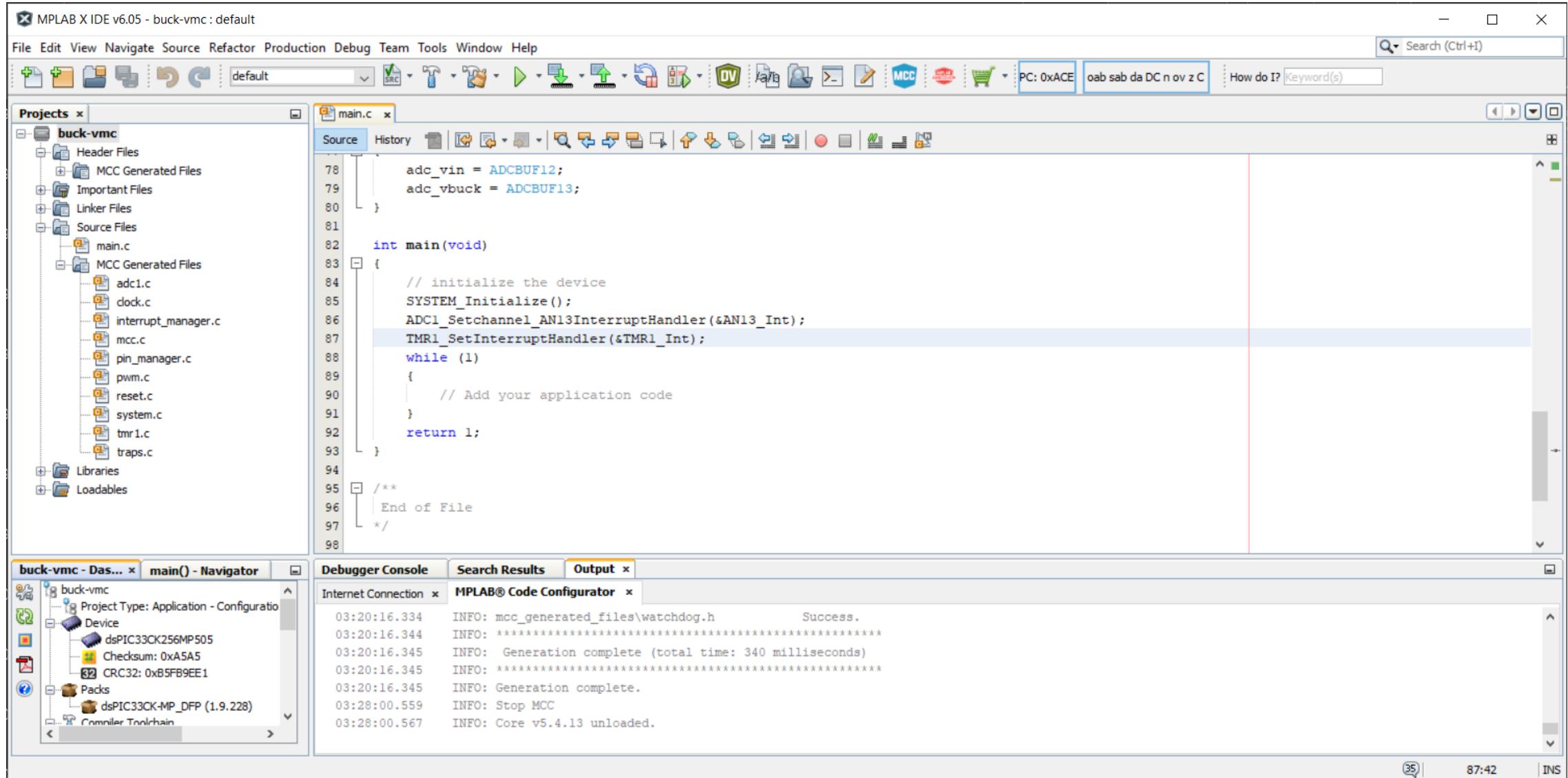
DO NOT ATTACH POWER YET !!!

- **Program the Project** 
 - When programming was successful, the red LED below dsPIC should blink

- **Close MCC**



Now we can continue coding & testing ...



Next up: PWM Output Signals

```
void TMR1_Int(void)
{
    static int led_tmr = 0;

    // blink LED @ 1Hz
    if(++led_tmr>=500) {
        led_tmr = 0;
        DGBLED_Toggle();
    }

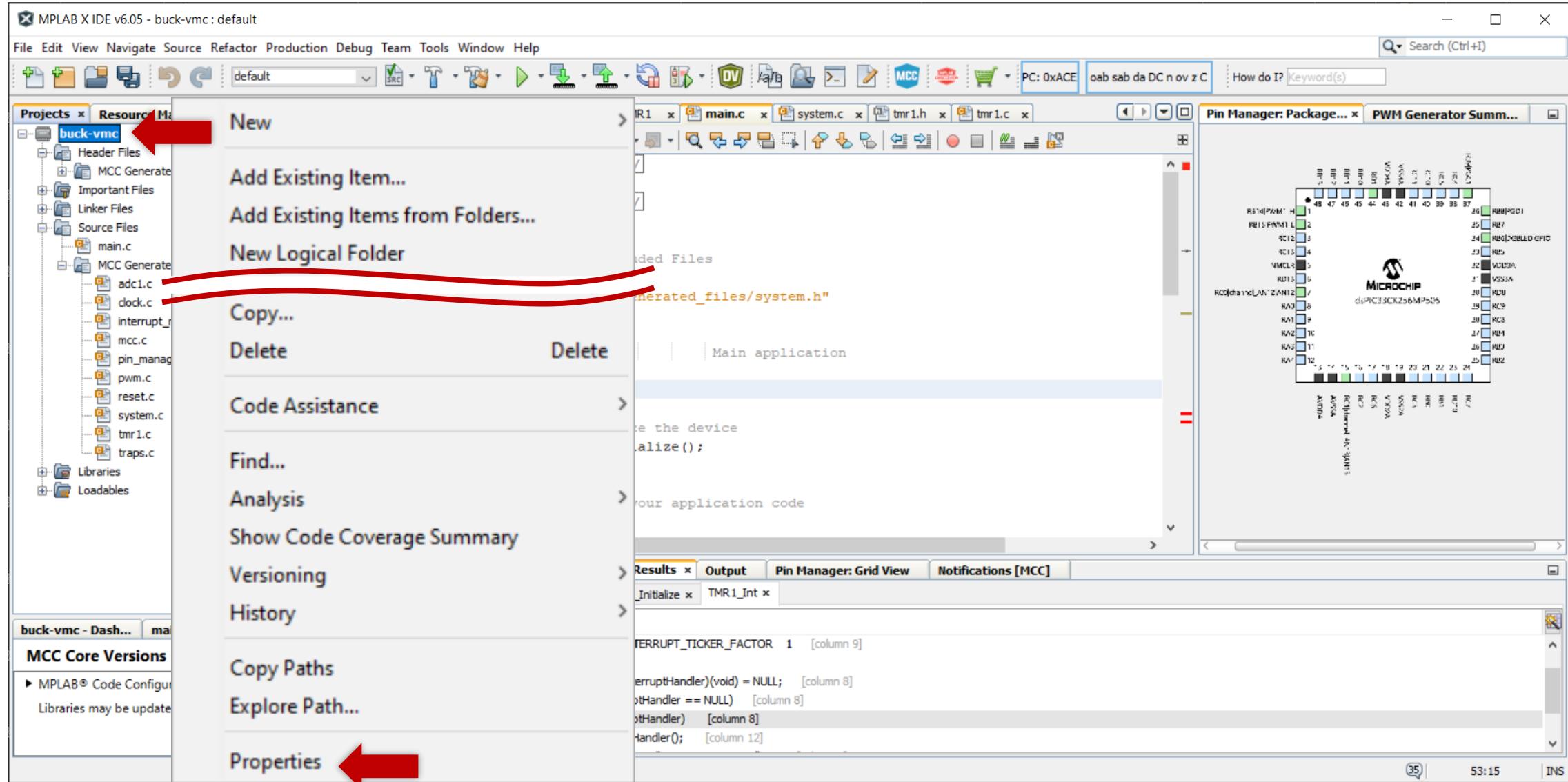
    // State Machine
    if (PG1DC < 3000) PG1DC++;
}
```

Add Open Loop Soft-Start code to prevent start of PWM from triggering protection circuit

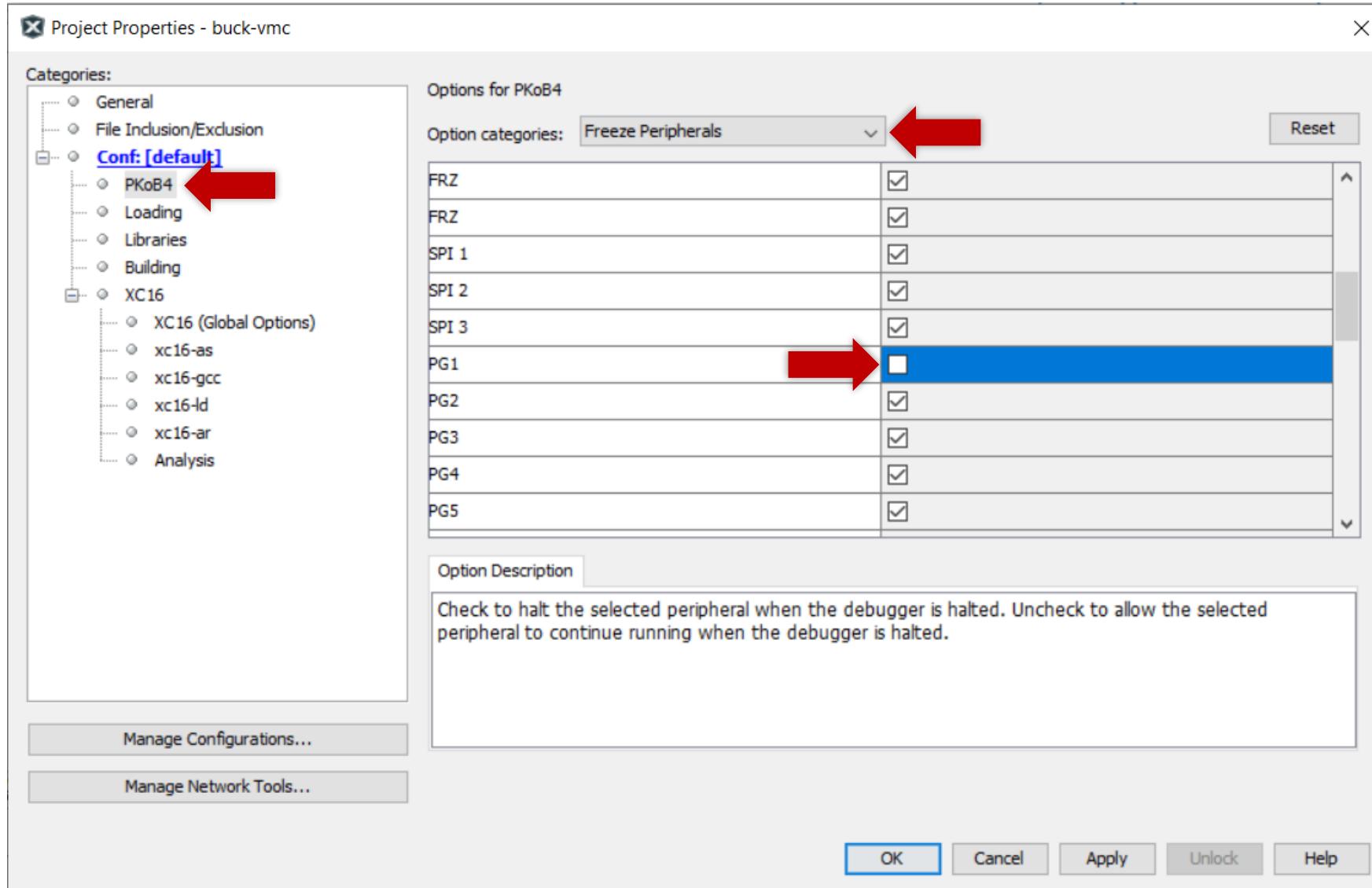
Program Target Device



“Unfreeze” PWM Generator #1



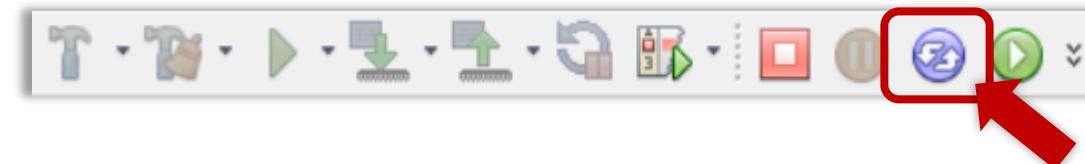
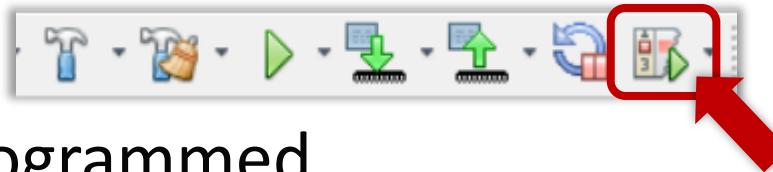
“Unfreeze” PWM Generator #1



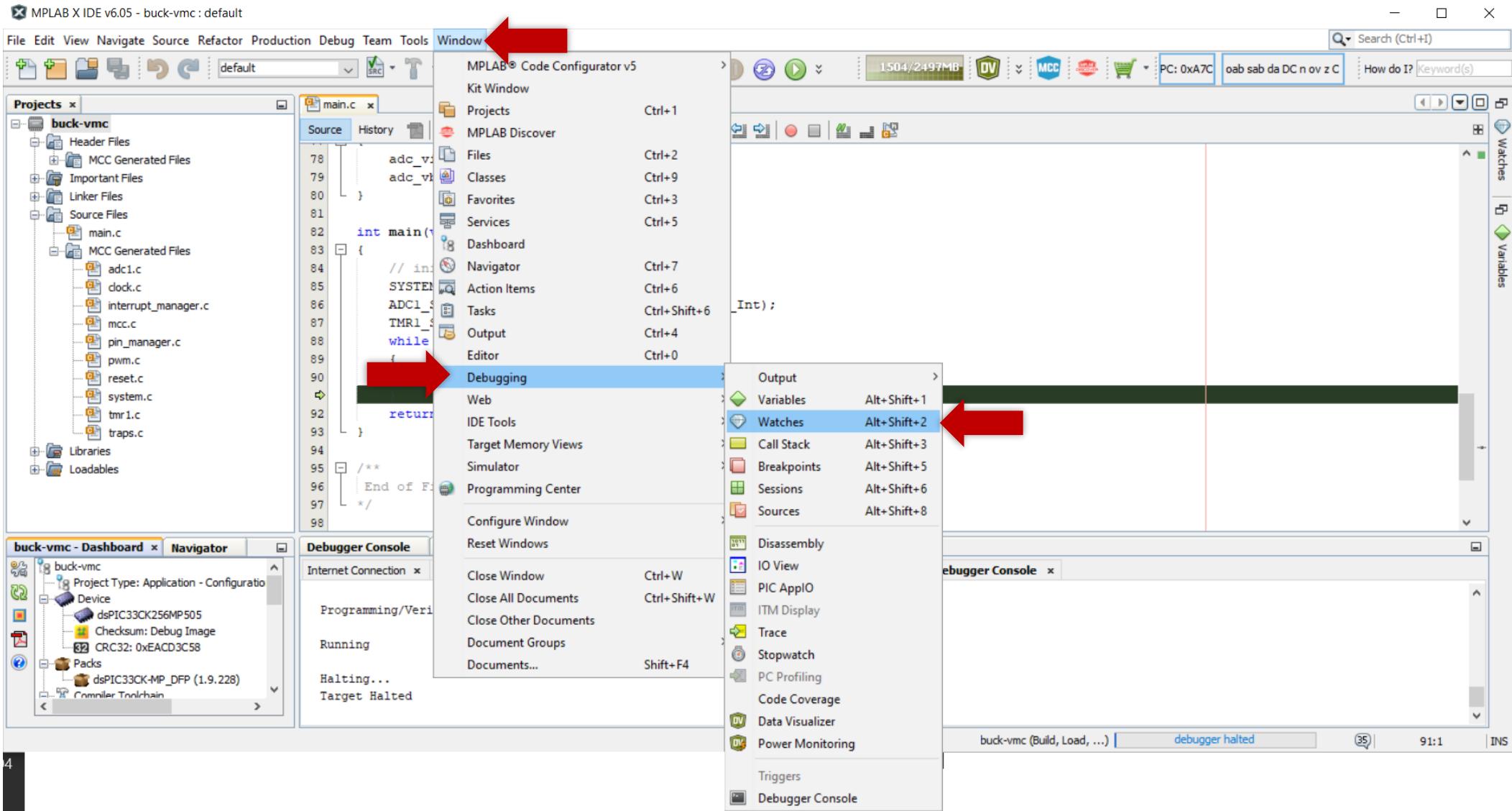
Start a Debug Session for ADC Verification

- Start a Debug Session

- Target device is being programmed
- Software stops at start of `main()`
- If not, click **Pause** and **Reset**

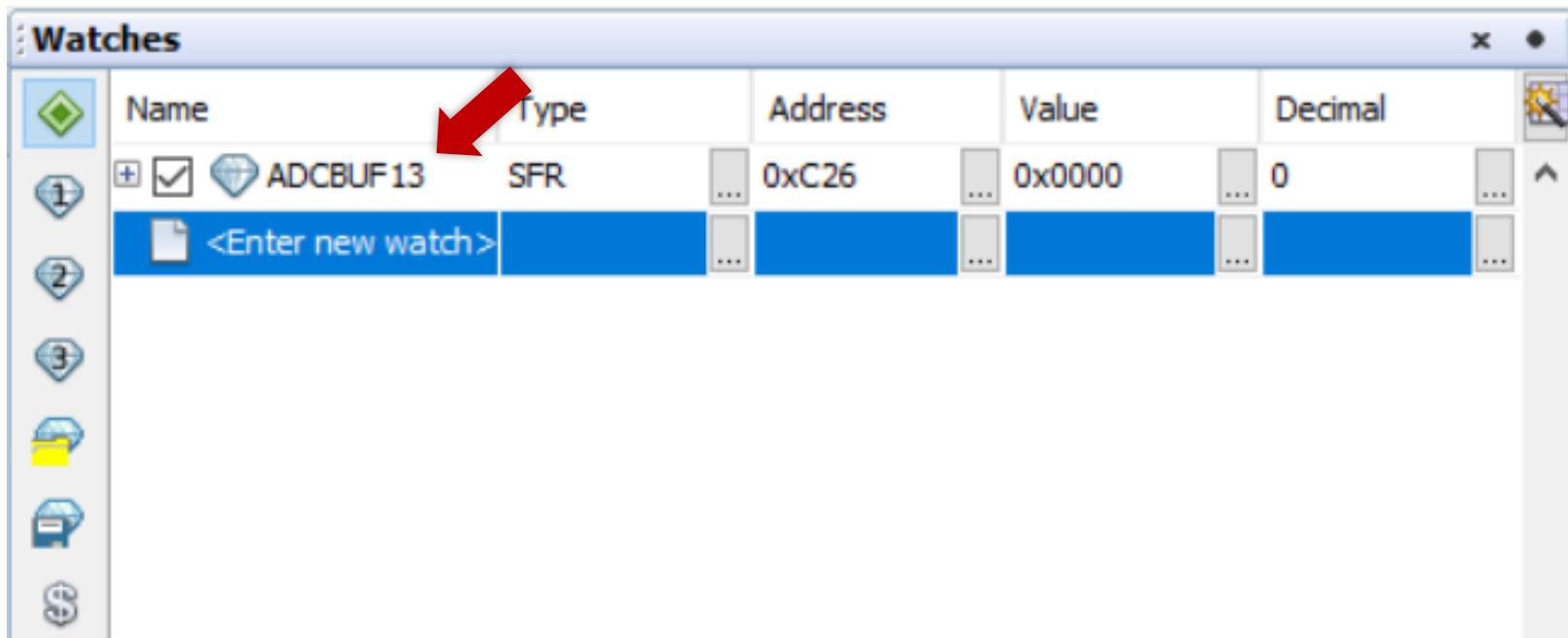


Open the Watch-Window



Analyzing Variables at Runtime

- Add register name **ADCBUF13** to Watches



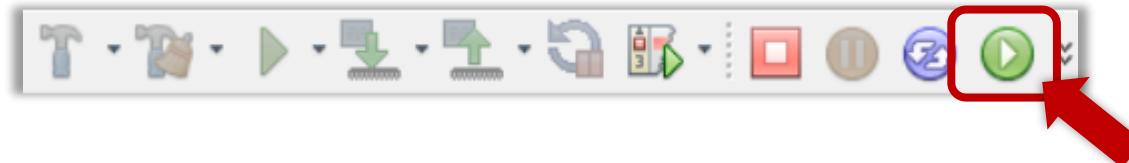
- Connect 9V Power Supply to DPSK3
- Release target device



Analyzing Variables at Runtime

- Connect 9V Power Supply to DPSK3

- Release target device



- Let it run for some seconds...

- Click Pause to halt the code execution



Analyzing Variables at Runtime

- Check **ADCBUF13** value
- Value = 2156 ticks

Name	Type	Address	Value	Decimal
ADCBUF13	SFR	0xC26	0x0876	2166
<Enter new watch>				

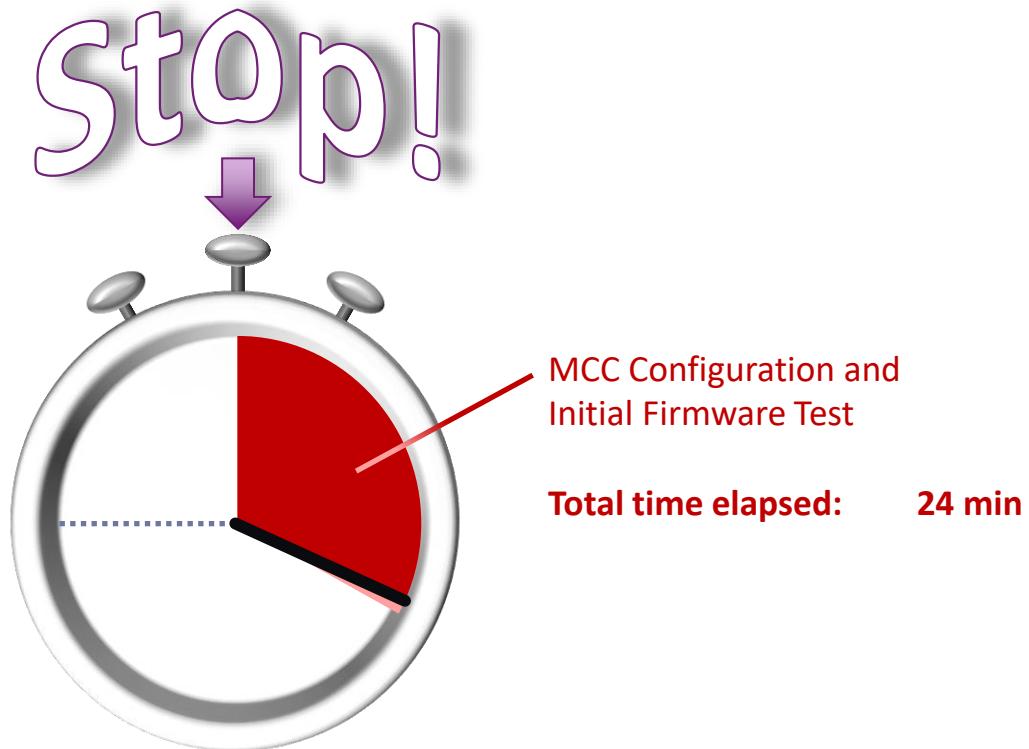
$$V_{FB} = V_{REF} \times \frac{ADCBUF13}{4095} = 3.3\text{ V} \times \frac{2156}{4095} = 1.737\text{ V}$$

$$V_{OUT} = V_{VB} \times \frac{R_1 + R_2}{R_2} = 1.737\text{ V} \times \frac{1\text{ k}\Omega + 1\text{ k}\Omega}{1\text{ k}\Omega} = 3.475\text{ V}$$

Let's take a short break...

Remaining Time until Controller Design Completion: 36 min*

*: I am speechless.....



Agenda

Digital Power Supply from Scratch in 60 Minutes*

*: still not buying it

- Overview Digital Power Starter Kit 3 (DPSK3)
- Requirements and Scope
- **Hands-On**
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB® PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start and Error Handling State Machine to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

Assign MPLAB® X Project

PowerSmart - [New PowerSmart Project.psproj]

File Edit Tools ?

Project Explorer

MPLAB X IDE Project

(no MPLAB® X IDE project associated)

Please assign an existing MPLAB® X project to this PowerSmart™ configuration by specifying its location below.

MPLAB® X Project Directories

MPLAB® X Project Location:

Active Project Configuration:

Active Target Device:

Makefile Location:

Active C Include Path:

Active Assembler Include Path:

Always reference include paths to Makefile location

Add or remove component libraries to/from your project.

Component Library

Power Supply Control

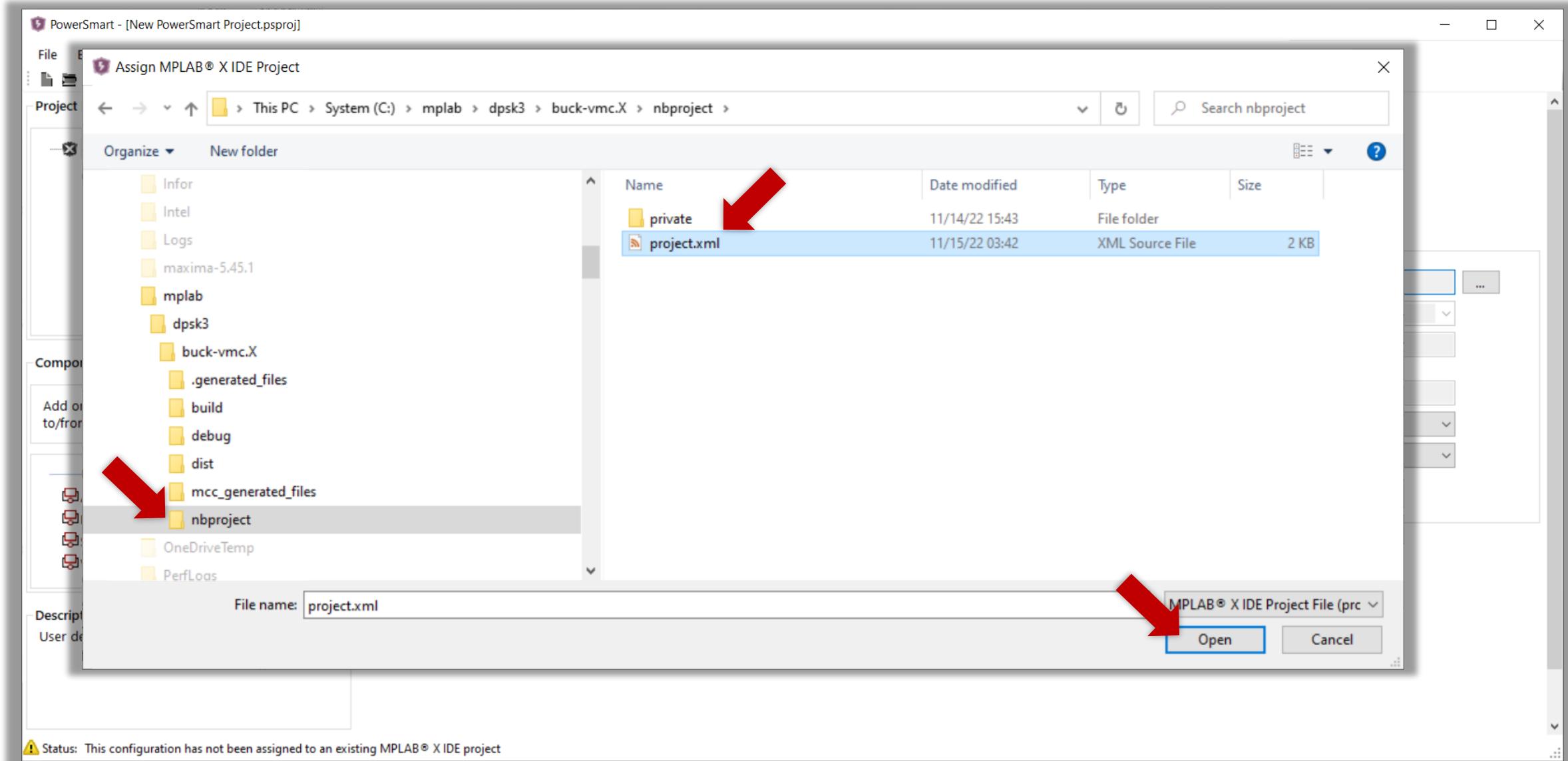
- Average Current Mode Control (ACMC)
- Peak Current Mode Control (PCMC)
- Single Control Loop (SLCOMP)
- Voltage Mode Control (VMC)

Description

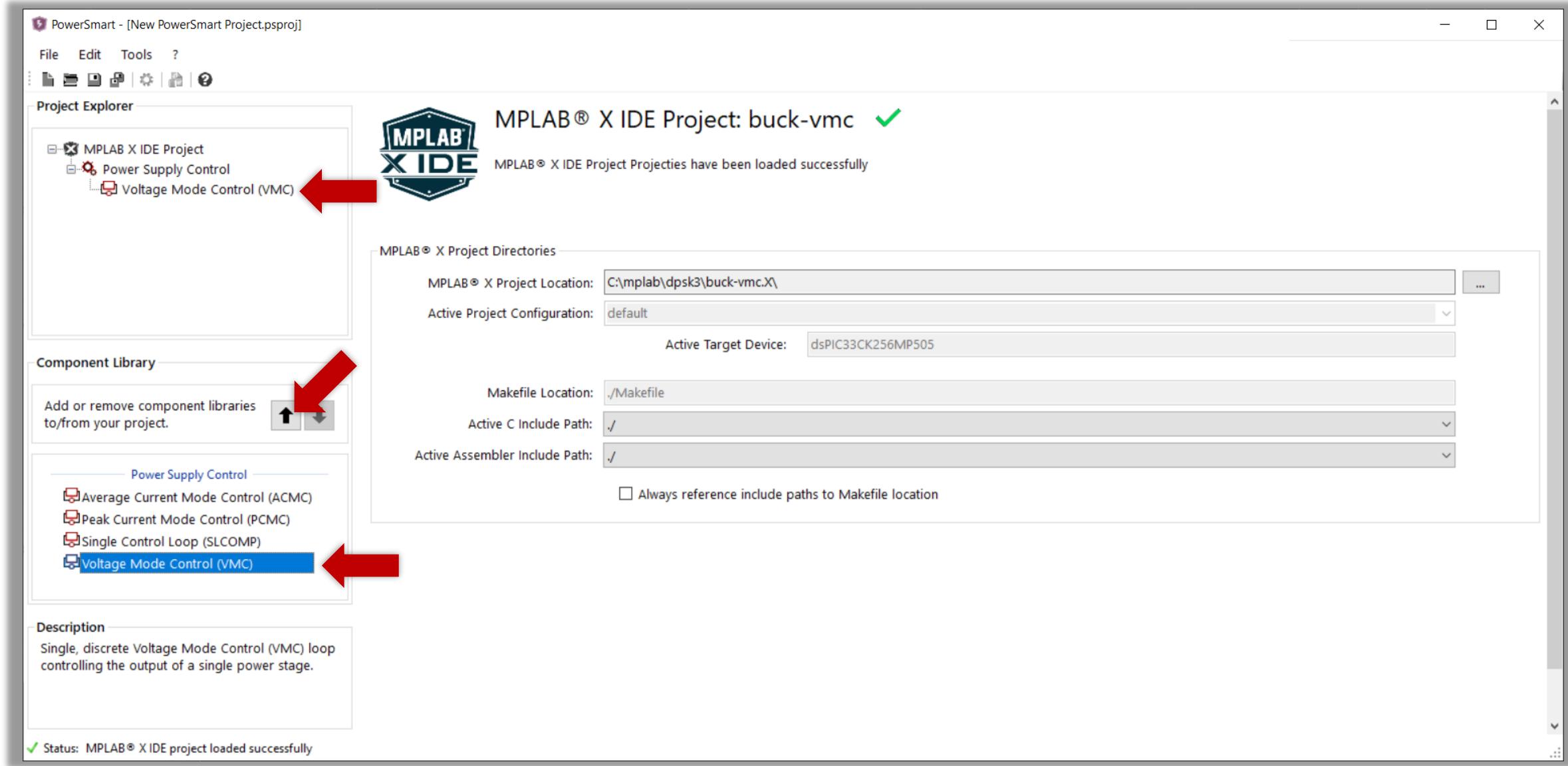
User declared MPLAB® X IDE Project Settings.

Status: This configuration has not been assigned to an existing MPLAB® X IDE project

Assign MPLAB® X Project



Add Voltage Loop Controller to PS™ Project



Open Voltage Loop Controller

PowerSmart - [New PowerSmart Project.psproj]

File Edit Tools ?

Project Explorer

- MPLAB X IDE Project
 - Power Supply Control
 - Voltage Mode Control (VMC)

Block Diagram Bode Plot

Voltage Mode Control (VMC)

Component Library

Add or remove component libraries to/from your project.

Power Supply Control

- Average Current Mode Control (ACMC)
- Peak Current Mode Control (PCMC)
- Single Control Loop (SLCOMP)
- Voltage Mode Control (VMC)**

Description

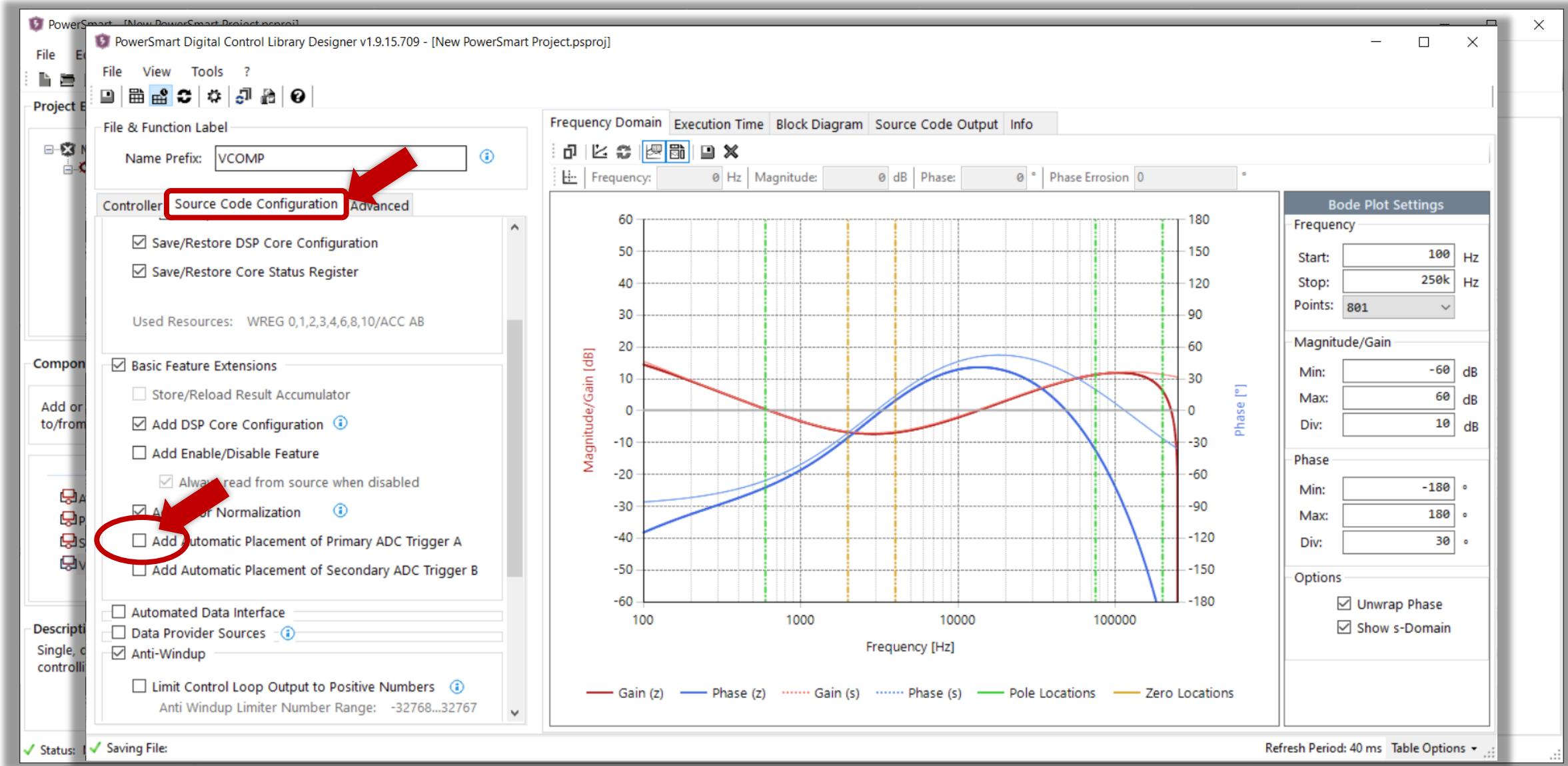
Single, discrete Voltage Mode Control (VMC) loop controlling the output of a single power stage.

✓ Status: MPLAB® X IDE project loaded successfully

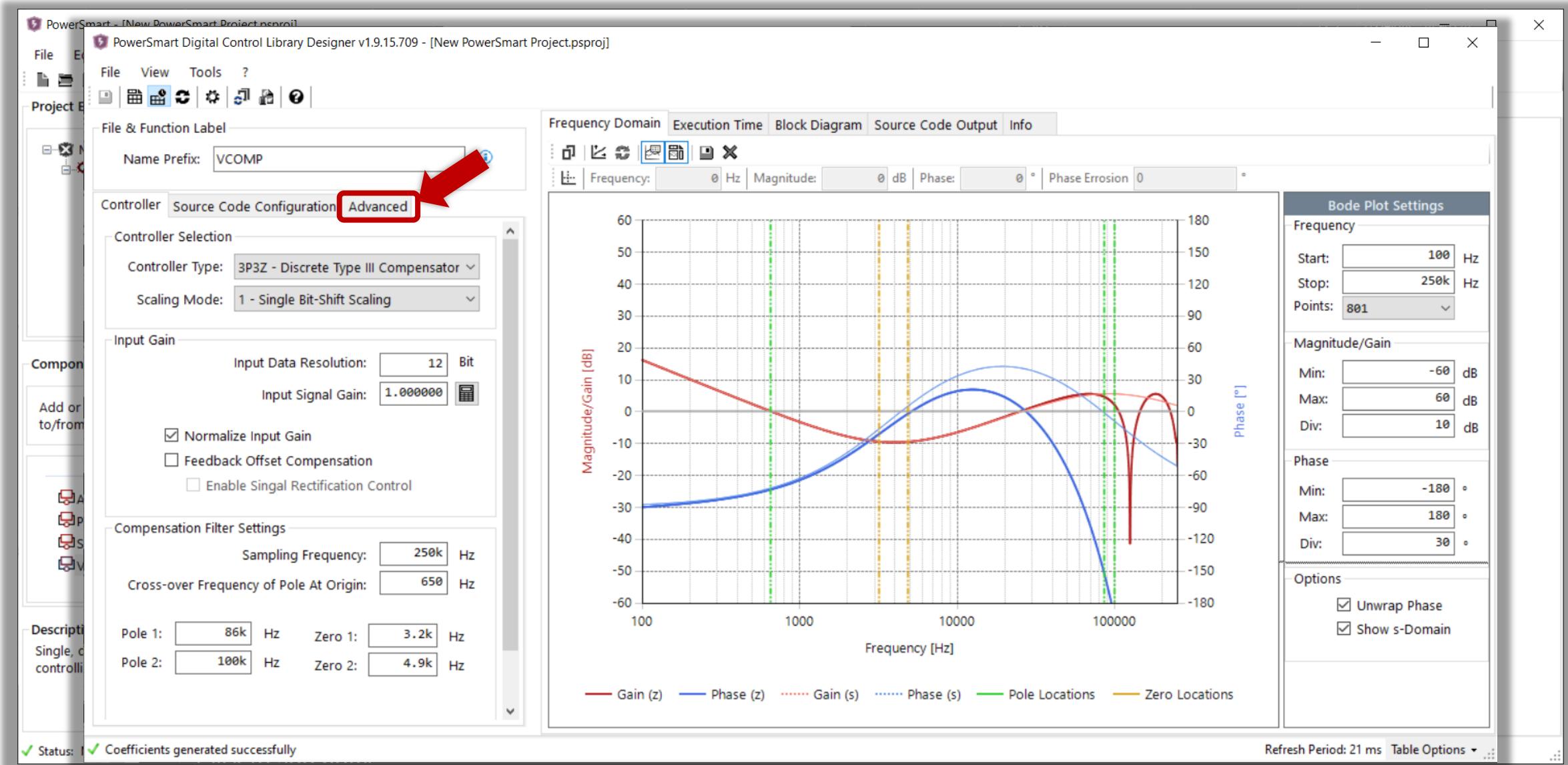
Voltage Mode Controller

The block diagram illustrates the VMC control loop. It starts with a reference voltage (REF) input, which is summed (Σ) with a feedback signal from an ADC. The summing junction also receives a VREF signal. The output of the summing junction is fed into a 'Voltage Loop Compensator H(z)' block. This block is labeled 'VCOMP (empty)'. The output of the compensator is connected to a PWM block, which generates a CLK signal. The PWM output drives a 'Converter Voltage Plant G(s)' block, labeled 'VPLANT (empty)'. The converter's output is VOUT. A feedback path from VOUT goes through a transmission function T_v(s) back to the ADC.

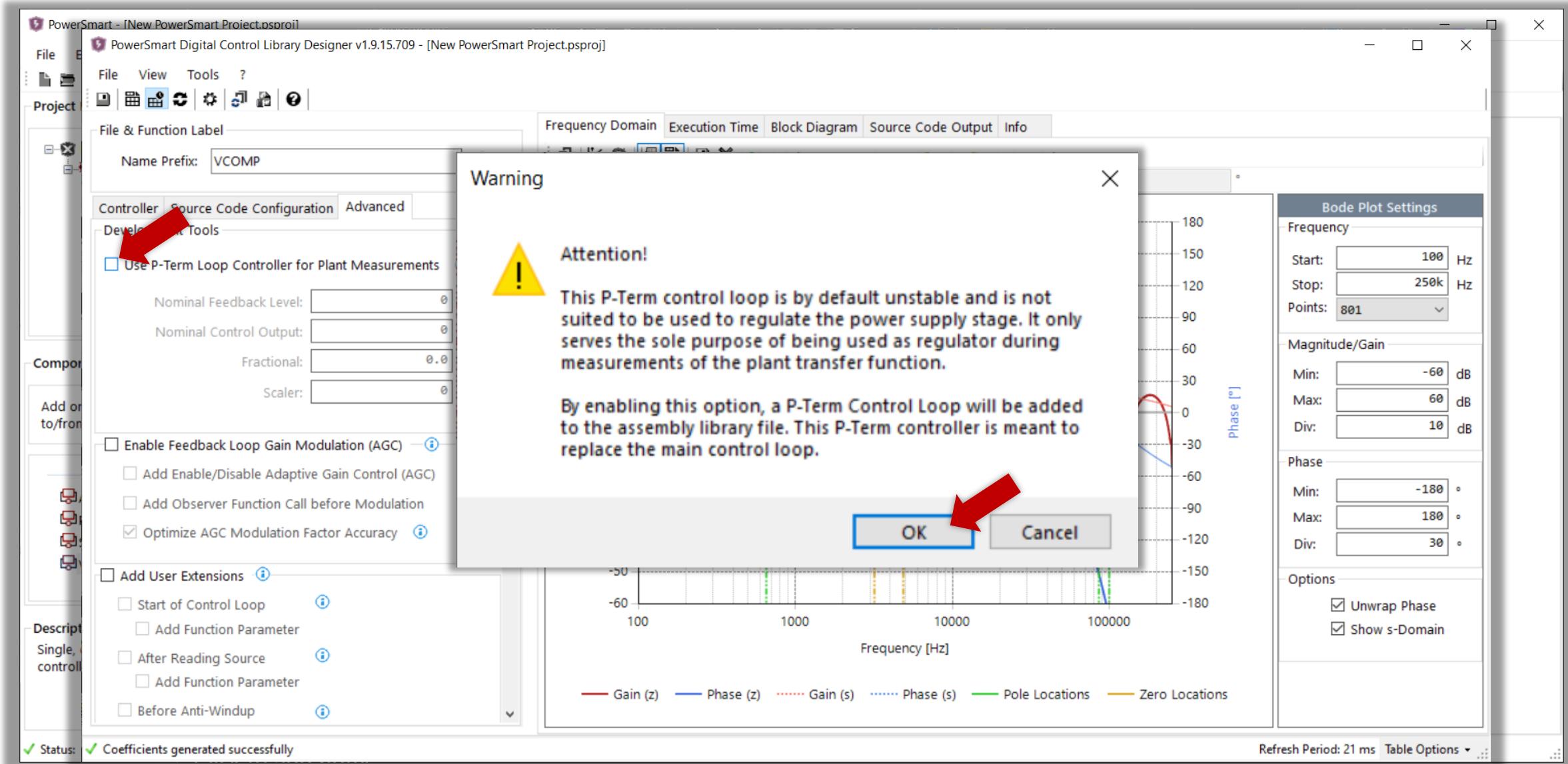
Turn Off Auto ADC Trigger Placement



Switch to Advanced Tab



Enable Pterm Controller Generation



Configure Data Input

PowerSmart - [New PowerSmart Project.psproj]

File View Tools ?

Project File & Function Label Name Prefix: VCOMP

Controller Source Code Configuration Advanced

Development Tools

- Use P-Term Loop Controller for Plant Measurements
- Nominal Feedback Level: 0
- Nominal Control Output: 0
- Fractional: 0.0
- Scaler: 0

Enable Feedback Loop Gain Modulation (AGC)

- Add Enable/Disable Adaptive Gain Control (AGC)
- Add Observer Function Call before Modulation
- Optimize AGC Modulation Factor Accuracy

Add User Extensions

- Start of Control Loop
- Add Function Parameter
- After Reading Source
- Add Function Parameter
- Before Anti-Windup

Coefficients generated successfully

Nominal Feedback Level Calculator

Voltage Feedback Shunt Amplifier Current Transformer Digital Source

Circuit

Input Scaling

ADC Reference: 3.3 V

ADC Resolution: 12 Bit

Minimum: 0

Maximum: 4095

Differential (signed)

Calculation

Nominal Sense Voltage: 3.3 V

R1: 1k Ω

R2: 1k Ω

Amplifier Gain: 1.000 V/V

Signal Gain: 0.500000 V/V

Bode Plot Settings

Frequency

Start: 100 Hz

Stop: 250k Hz

Points: 801

Magnitude/Gain

Min: -60 dB

Max: 60 dB

Div: 10 dB

Phase

Min: -180 °

Max: 180 °

Div: 30 °

Options

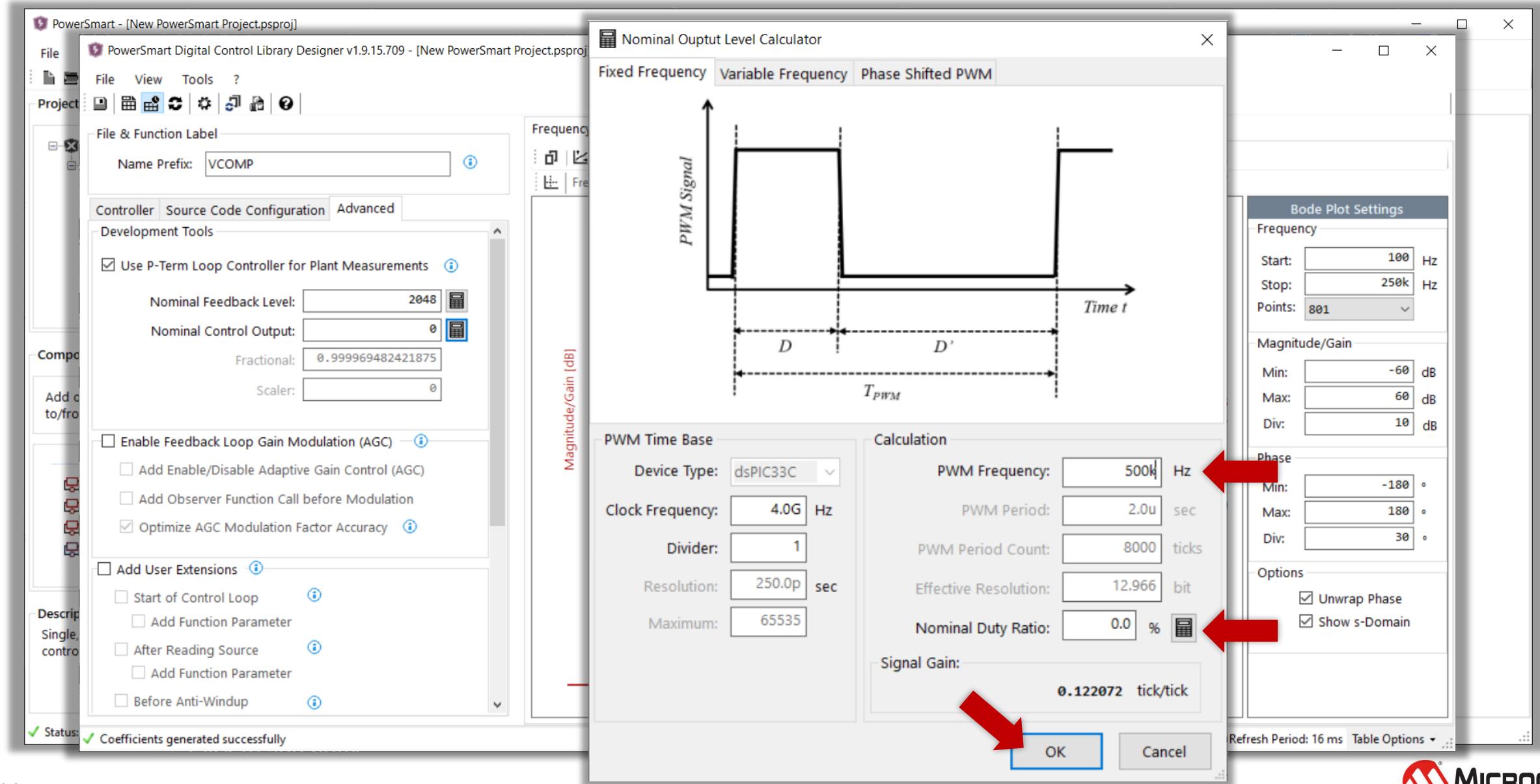
Unwrap Phase

Show s-Domain

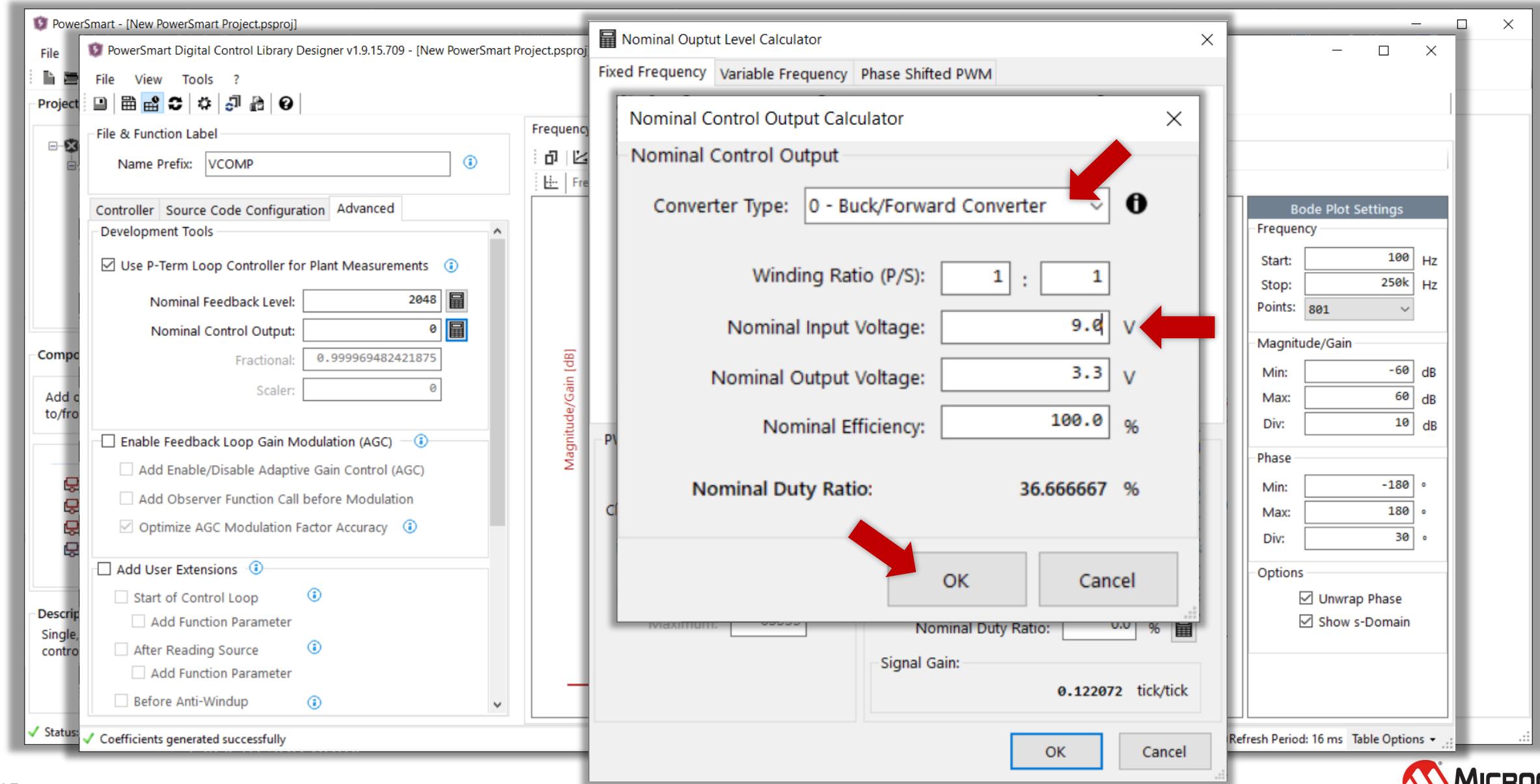
OK Cancel

Refresh Period: 21 ms Table Options

Configure Data Output Port



Get Estimated, Ideal Duty Ratio



Apply Derived Data

Screenshot of the PowerSmart Digital Control Library Designer v1.9.15.709 interface showing the Nominal Output Level Calculator and Bode Plot Settings dialog boxes.

Nominal Output Level Calculator:

- Fixed Frequency Tab:** Shows a PWM signal waveform with time axis labeled t . The period is T_{PWM} , divided into D and D' .
- PWM Time Base:**
 - Device Type: dsPIC33C
 - Clock Frequency: 4.0G Hz
 - Divider: 1
 - Resolution: 250.0p sec
 - Maximum: 65535
- Calculation:**
 - PWM Frequency: 500k Hz
 - PWM Period: 2.0u sec
 - PWM Period Count: 8000 ticks
 - Effective Resolution: 12.966 bit
 - Nominal Duty Ratio: 36.667 %
- Signal Gain:** 0.122072 tick/tick

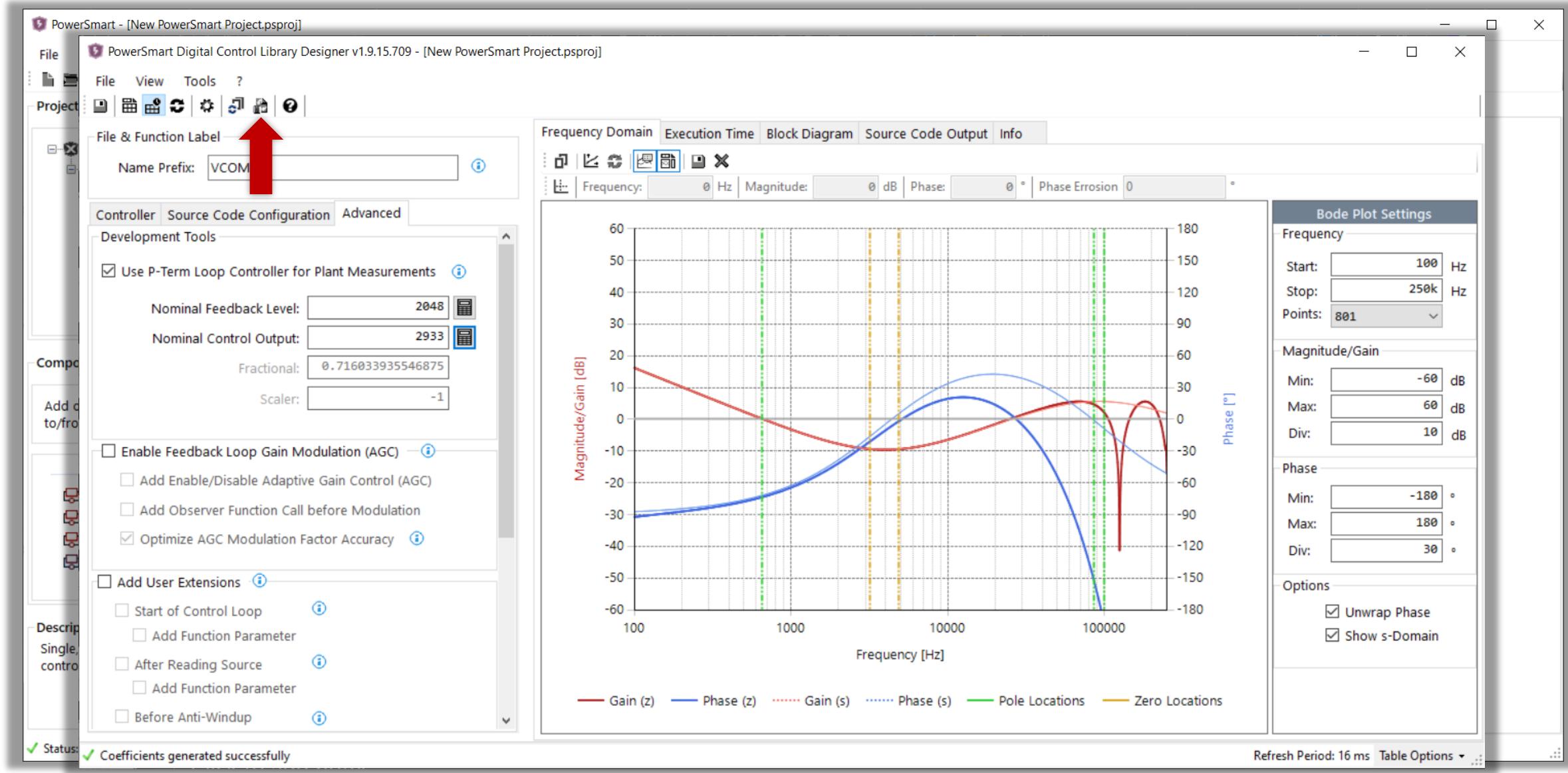
A red arrow points to the **OK** button at the bottom right of the calculator window.

Bode Plot Settings Dialog:

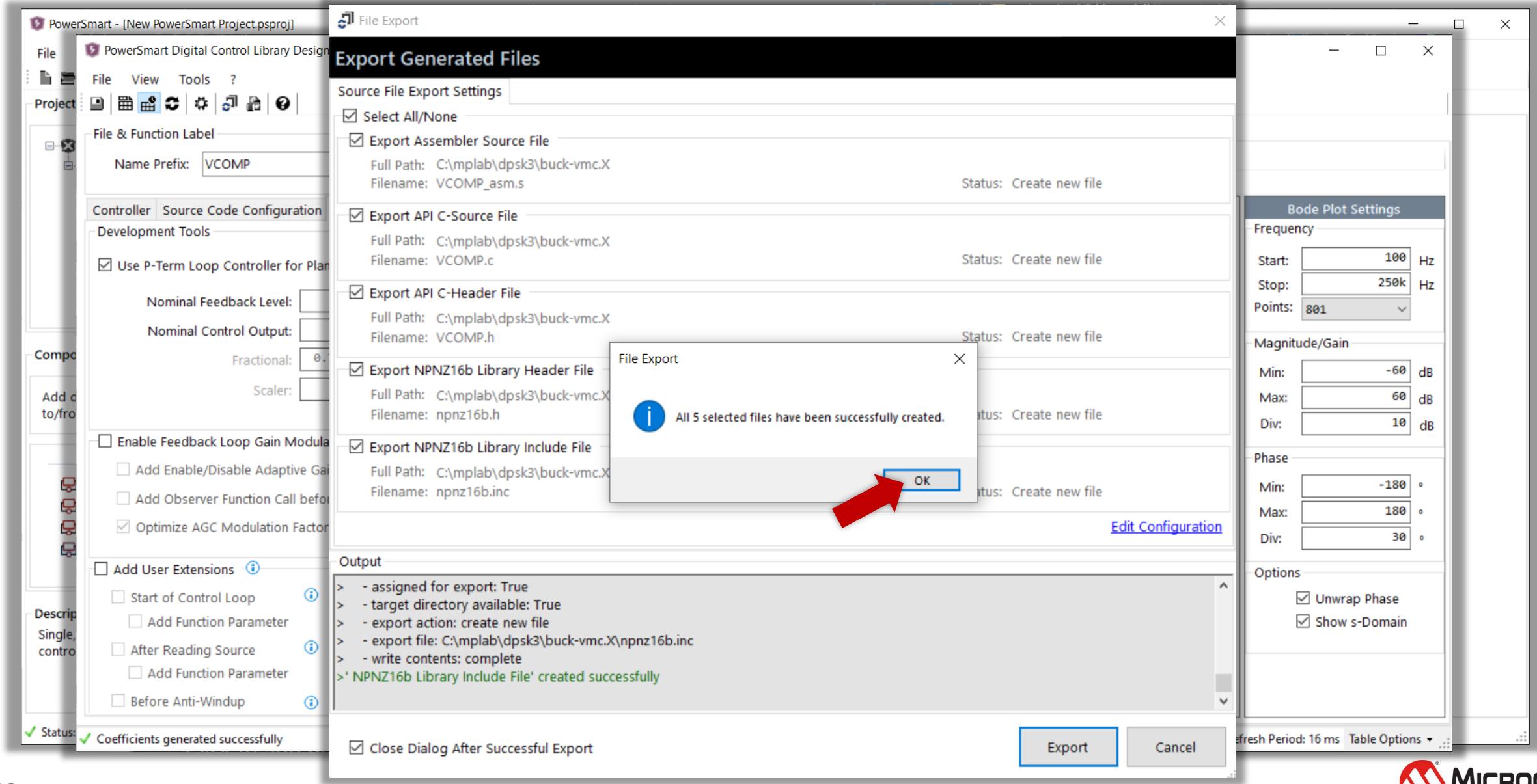
- Frequency:
 - Start: 100 Hz
 - Stop: 250k Hz
 - Points: 801
- Magnitude/Gain [dB]:
 - Min: -60 dB
 - Max: 60 dB
 - Div: 10 dB
- Phase:
 - Min: -180 °
 - Max: 180 °
 - Div: 30 °
- Options:
 - Unwrap Phase
 - Show s-Domain

Refresh Period: 16 ms Table Options

Open Code Generator

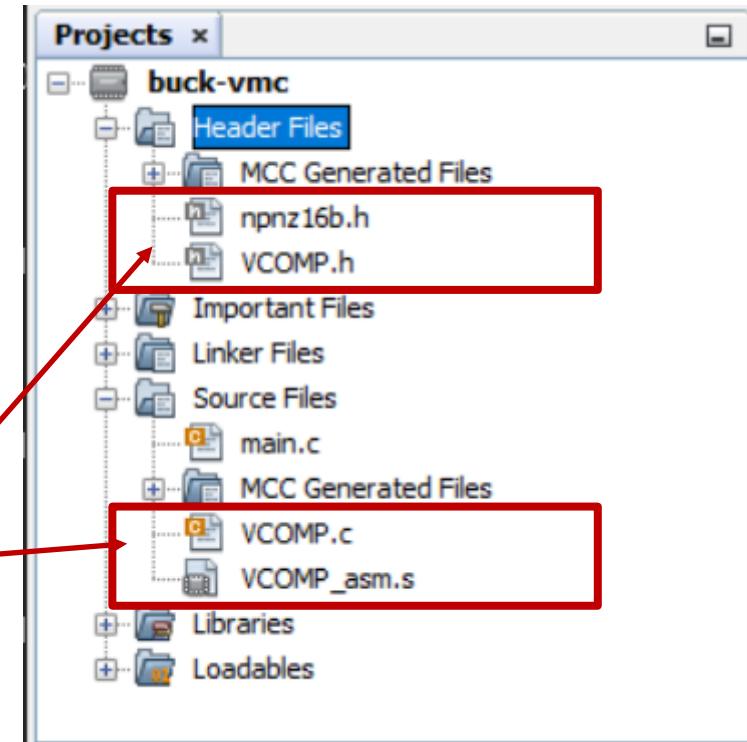


Generate Code



Import Generated Files into MPLAB® X Project

Name	Date modified	Type	Size
.generated_files	11/14/22 15:43	File folder	
build	11/15/22 02:43	File folder	
debug	11/15/22 01:35	File folder	
dist	11/15/22 02:44	File folder	
mcc_generated_files	11/15/22 01:23	File folder	
nbproject	11/14/22 15:43	File folder	
.gitignore	11/14/22 15:43	Git Ignore Source ...	1 KB
buck-vmc.mc3	11/15/22 03:38	MC3 File	2,442 KB
main.c	11/15/22 03:19	C Source File	3 KB
Makefile	11/14/22 15:43	File	4 KB
npnz16b.h	11/15/22 05:19	C Header Source F...	39 KB
npnz16b.inc	11/15/22 05:19	Include File	14 KB
VCOMP.c	11/15/22 05:19	C Source File	9 KB
VCOMP.h	11/15/22 05:19	C Header Source F...	13 KB
VCOMP_asm.s	11/15/22 05:19	Assembler Source	37 KB





Coding Time !

Removing Test Code from main.c

```
void TMR1_Int(void)
{
    static int led_tmr = 0;

    // blink LED 1Hz
    if(++led_tmr>=500) {
        led_tmr = 0;
        DGBLED_Toggle();
    }

    // State Machine
    if (PG1DC < 3000) PG1DC++;
}
```

Please remove this line

Including VCOMP header files in main.c

```
/**  
 * Section: Included Files  
 */  
  
#include "mcc_generated_files/system.h"  
#include "mcc_generated_files/pin_manager.h"  
#include "mcc_generated_files/pwm.h"  
#include "mcc_generated_files/adc1.h"  
#include "mcc_generated_files/tmr1.h"  
#include "VCOMP.h"  
[...]  
  
int main(void)  
{  
    // initialize the device  
    SYSTEM_Initialize();  
    ADC1_Setchannel_AN13InterruptHandler(&AN13_Int);  
    TMR1_SetInterruptHandler(&TMR1_Int);        while (1)  
    {  
        // Add your application code  
    }
```

Add Controller Initialization Code to main.c

```
void AN13_Int(void)
{
    adc_vin = ADCBUF12;
    adc_vbuck = ADCBUF13;
}

void CONTROL_Initialize(void)
{
    // initialize compensator
    VCOMP_Initialize(& VCOMP);
    VCOMP.Ports.Source.ptrAddress = (unsigned*) &adc_vbuck;
    VCOMP.Ports.Target.ptrAddress = (unsigned*) &PG1DC;
    VCOMP.Ports.ptrControlReference = (unsigned*) &ref_vbuck;
    VCOMP.Limits.MinOutput = 0;
    VCOMP.Limits.MaxOutput = (int) (0.8*PG1PER); // 80% DC
}

int main(void)
{
```

Call Controller Initialization from main() in main.c

```
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    CONTROL_Initialize(); 
    ADC1_Setchannel_AN13InterruptHandler(&AN13_Int);
    TMR1_SetInterruptHandler(&TMR1_Int);

    while (1)
    {
        // Add your application code
    }
    return 1;
}
```

Must be called before ADC interrupt is set

Call Controller from ADC Interrupt in main.c

```
void AN13_Int(void)
{
    adc_vin = ADCBUF12;
    adc_vbuck = ADCBUF13;
    VCOMP_PTermUpdate(&VCOMP);
}

void CONTROL_Initialize(void)
{
    // initialize compensator
    VCOMP_Initialize(& VCOMP);
    VCOMP.Ports.Source.ptrAddress = (unsigned*)&adc_vbuck;
    VCOMP.Ports.Target.ptrAddress = (unsigned*)&PG1DC;
    VCOMP.Ports.ptrControlReference = (unsigned*)&ref_vbuck;
    VCOMP.Limits.MinOutput = 0;
    VCOMP.Limits.MaxOutput = (int)(0.8*PG1PER); // 80% DC
}

int main(void)
```

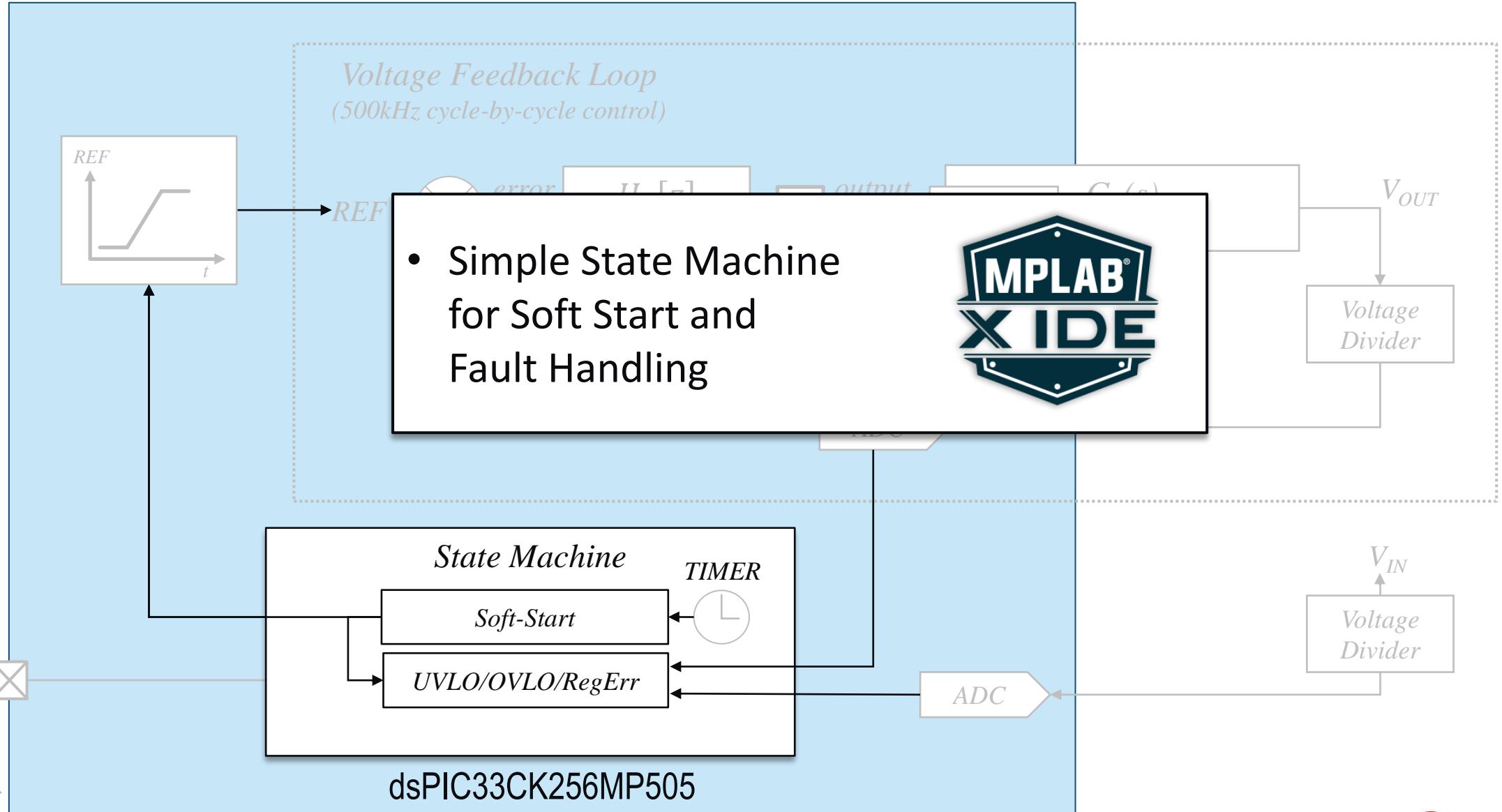
Agenda

Digital Power Supply from Scratch in 60 Minutes*

*: Are we even still within the 60 min?

- Overview Digital Power Starter Kit 3 (DPSK3)
- Requirements and Scope
- **Hands-On**
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB® PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start and Error Handling State Machine to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

Requirements



Add Scaling Macros to main.c

```
// Define Global Variables:  
int adc_vin = 0;      // Scaling: 155.1 steps/V  
int adc_vbuck = 0;    // Scaling: 620.5 steps/V  
int ref_vbuck = 0;    // Scaling: 620.5 steps/V  
  
// Add Defines for Hardware Properties  
#define AN12_VIN_SCALE   (float)( 1.0/8.0 * 4095.0/3.3 ) // Input voltage: 155 digits/V  
#define AN13_VBUCK_SCALE (float)( 1.0/2.0 * 4095.0/3.3 ) // Buck output voltage: 620 ticks/V  
  
#define VIN_TH_ON  (int16_t)( 6.5      * AN12_VIN_SCALE )    // Switch-on threshold: 6.5 V  
#define VIN_TH_OFF (int16_t)( 5.5      * AN12_VIN_SCALE )    // Switch-off (undervoltage)  
                           // threshold: 5.5 V  
#define VBUCK_REF   (int16_t)( 3.3      * AN13_VBUCK_SCALE ) // Output voltage set value: 3.3 V  
#define VBUCK_DREF  (int16_t)( 0.2      * AN13_VBUCK_SCALE ) // max. output voltage error: 200 mV  
#define VBUCK_RAMP  (int16_t)( 10e-3 * AN13_VBUCK_SCALE ) // soft start ramp: 10 mV/ms  
  
[...]
```

Add Local SM-Variables to Timer ISR in main.c

[...]

```
// Initial TMR1_Int() function, do only LED blinking
void TMR1_Int(void)
{
    static int led_tmr = 0;
    static int fsm_state = 0;
    static int start_tmr = 0;
    static int err_tmr = 0;

    // blink LED 1Hz
    if(++led_tmr>=500) {
        led_tmr = 0;
        USER_LED_Toggle();
    }
}
```

Add State Machine to Timer ISR in main.c

```
// Initial TMR1_Int() function, do only LED blinking
void TMR1_Int(void)
{
    [...]
    // output voltage supervision, increment error timer if voltage is too low
    err_tmr = (ref_vbuck-adc_vbuck)>VBUCK_DREF ? err_tmr+1 : 0;

    switch(fsm_state) {
        case 0: // State 0: wait for Vin>threshold (delay time)
            ref_vbuck = 0;
            err_tmr = 0;
            if(adc_vin<VIN_TH_ON) {
                start_tmr = 0;
            } else if(++start_tmr>2000) {
                fsm_state = 1;
            }
        break;
    }
}
```

Add State Machine to Timer ISR in main.c

```
// Initial TMR1_Int() function, do only LED blinking
void TMR1_Int(void)
{
    [...]
    switch(fsm_state) {
        case 0: // State 0: wait for Vin>threshold (delay time)
            [...]
            break;
        case 1: // State 1: ramp up buck reference value (soft start)
            if(ref_vbuck < VBUCK_REF) {
                ref_vbuck += VBUCK_RAMP;
            }
            if((adc_vin<VIN_TH_OFF) || (err_tmr>200)) { // low input voltage OR error for >200ms
                start_tmr = 0;
                fsm_state = 0;
            }
            break;
    }
}
```

Testing Time !



Give it a try...



- **Build the Project**



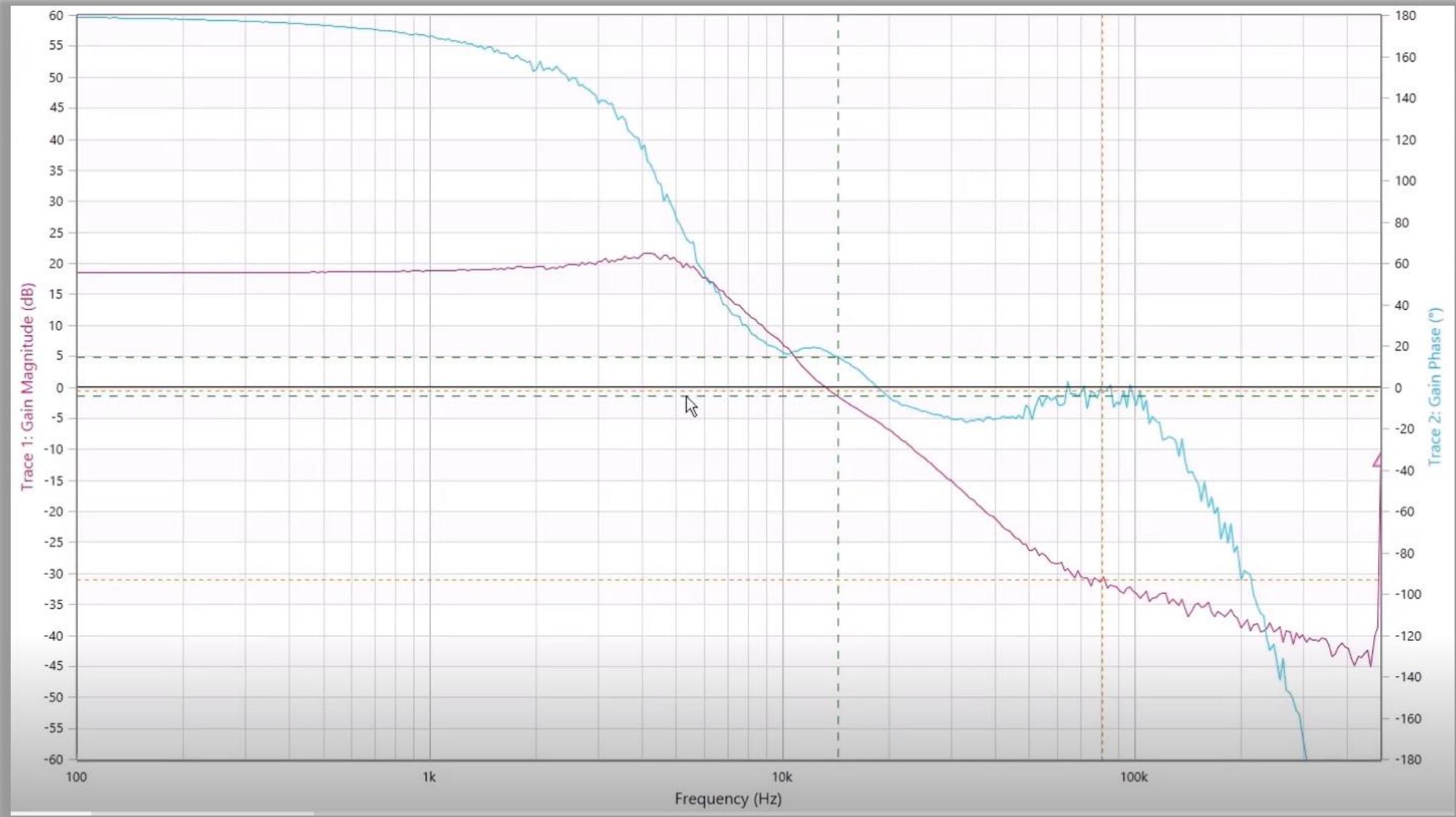
- If the Build was successful ...
- Or check error message and fix what's broken

- **Program the Project**



- When programming was successful, the red LED below dsPIC should blink

Bode Plot Results

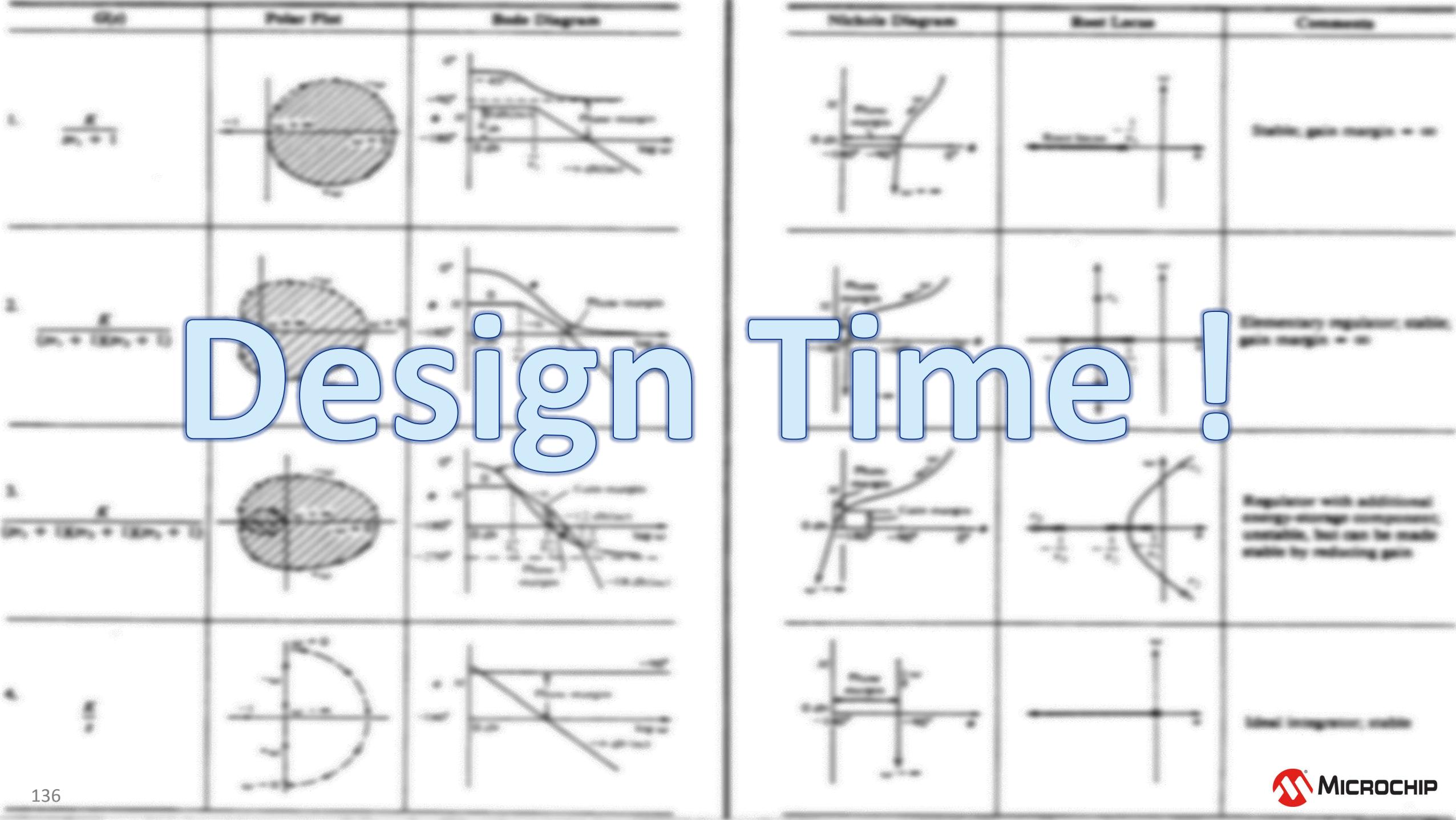


Agenda

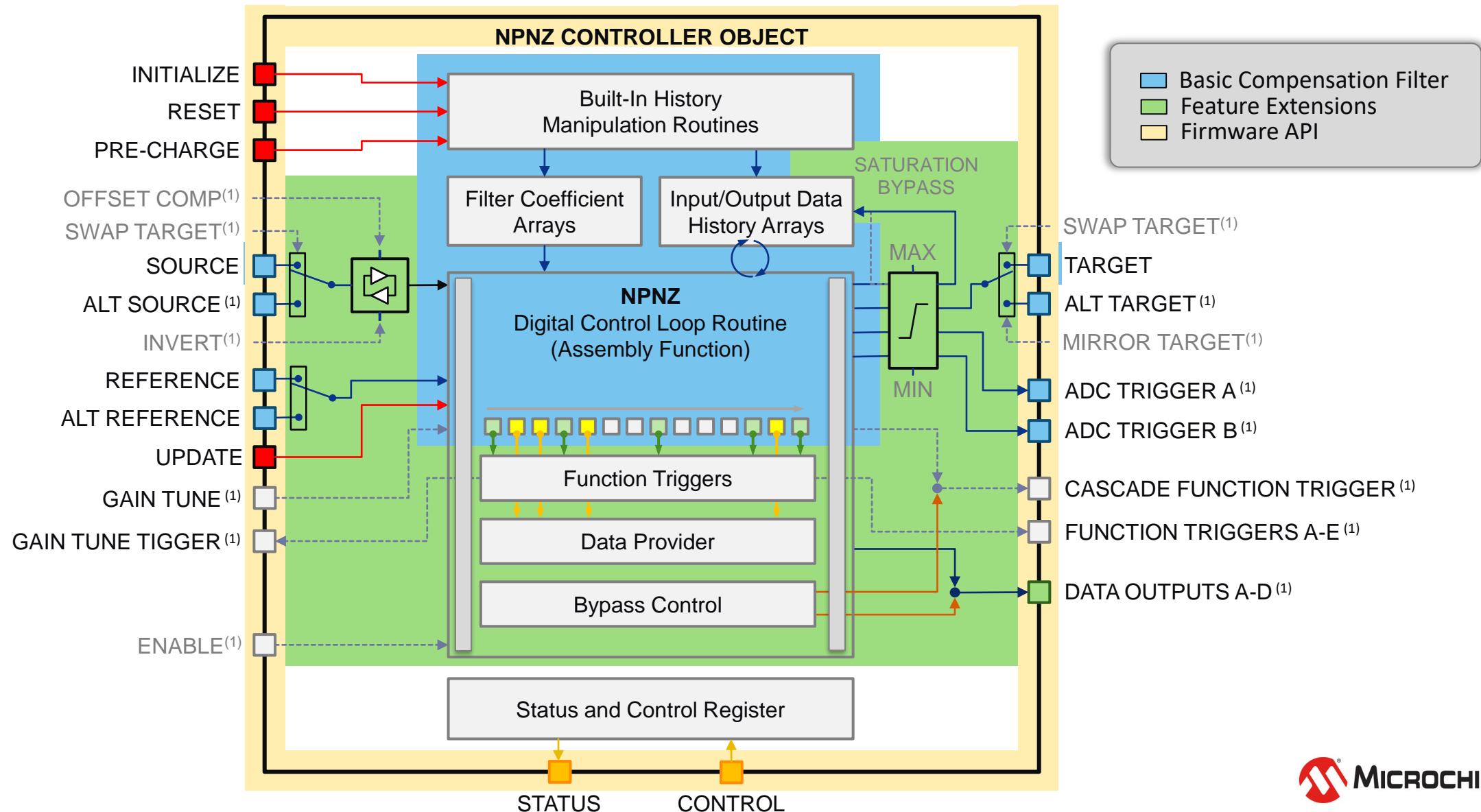
Digital Power Supply from Scratch in 60 Minutes*

*: Is anyone wearing a watch?

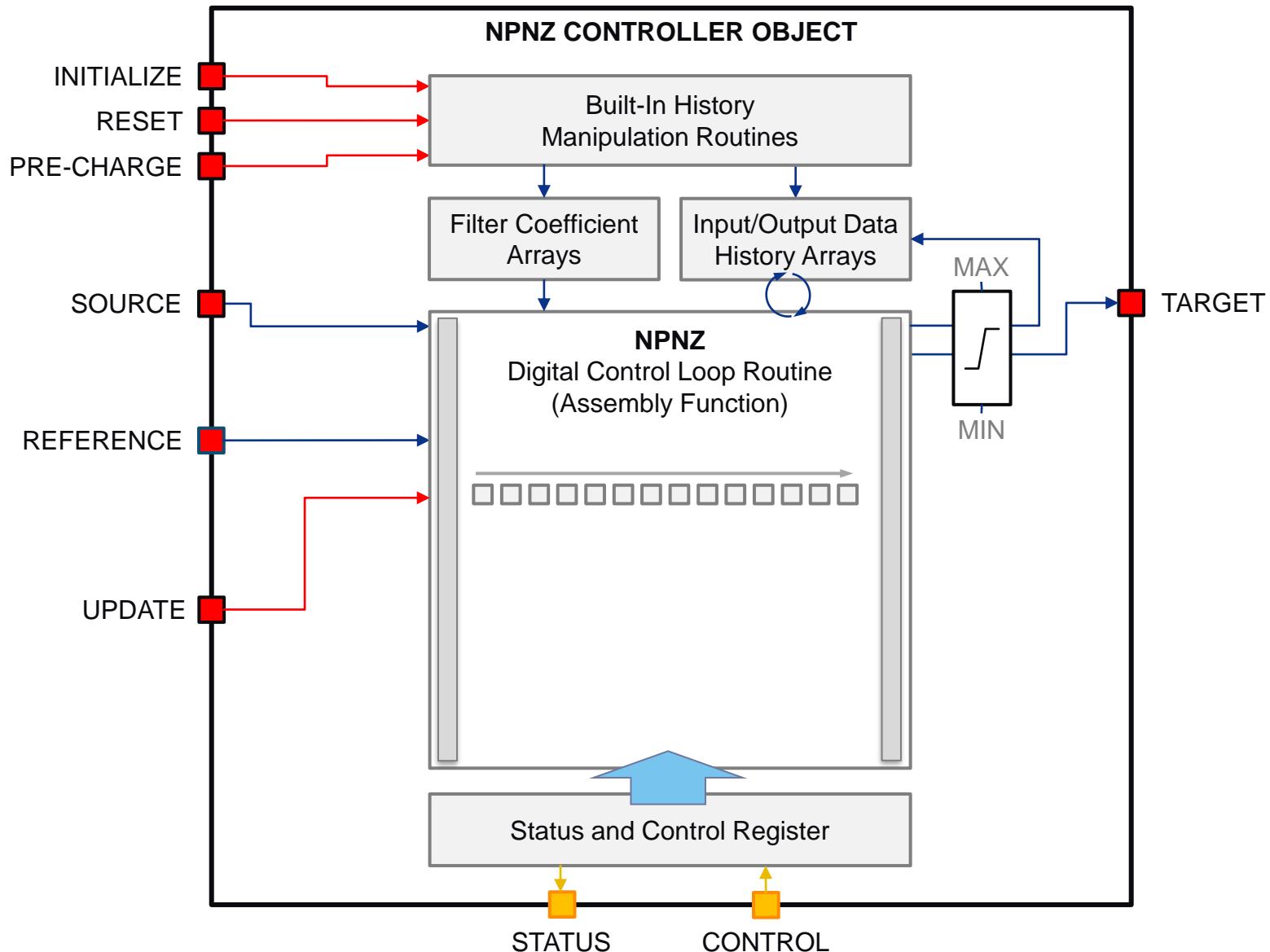
- Overview Digital Power Starter Kit 3 (DPSK3)
- Requirements and Scope
- **Hands-On**
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB® PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start and Error Handling State Machine to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A



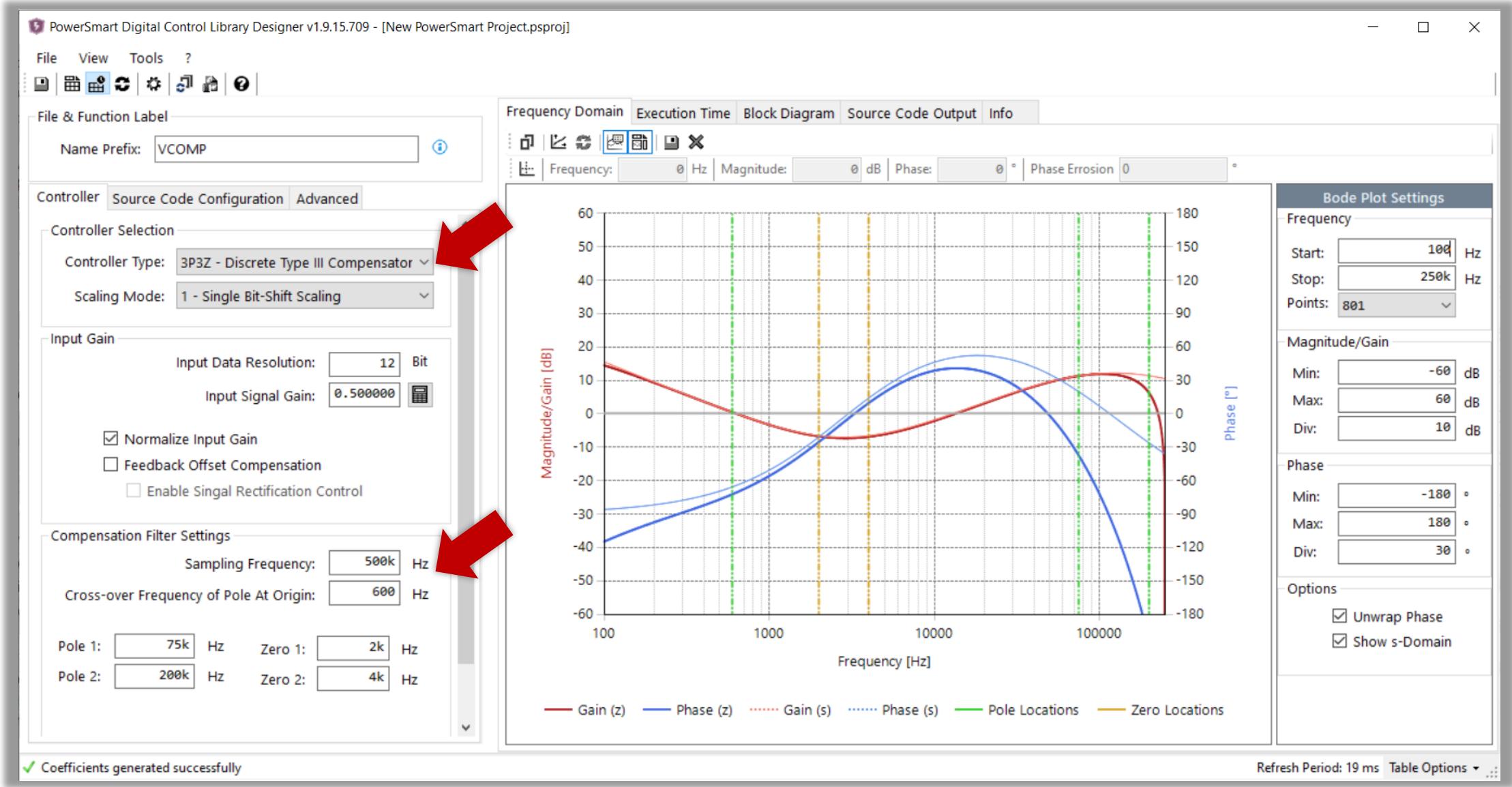
PS-DCLD Feedback Loop Block Diagram



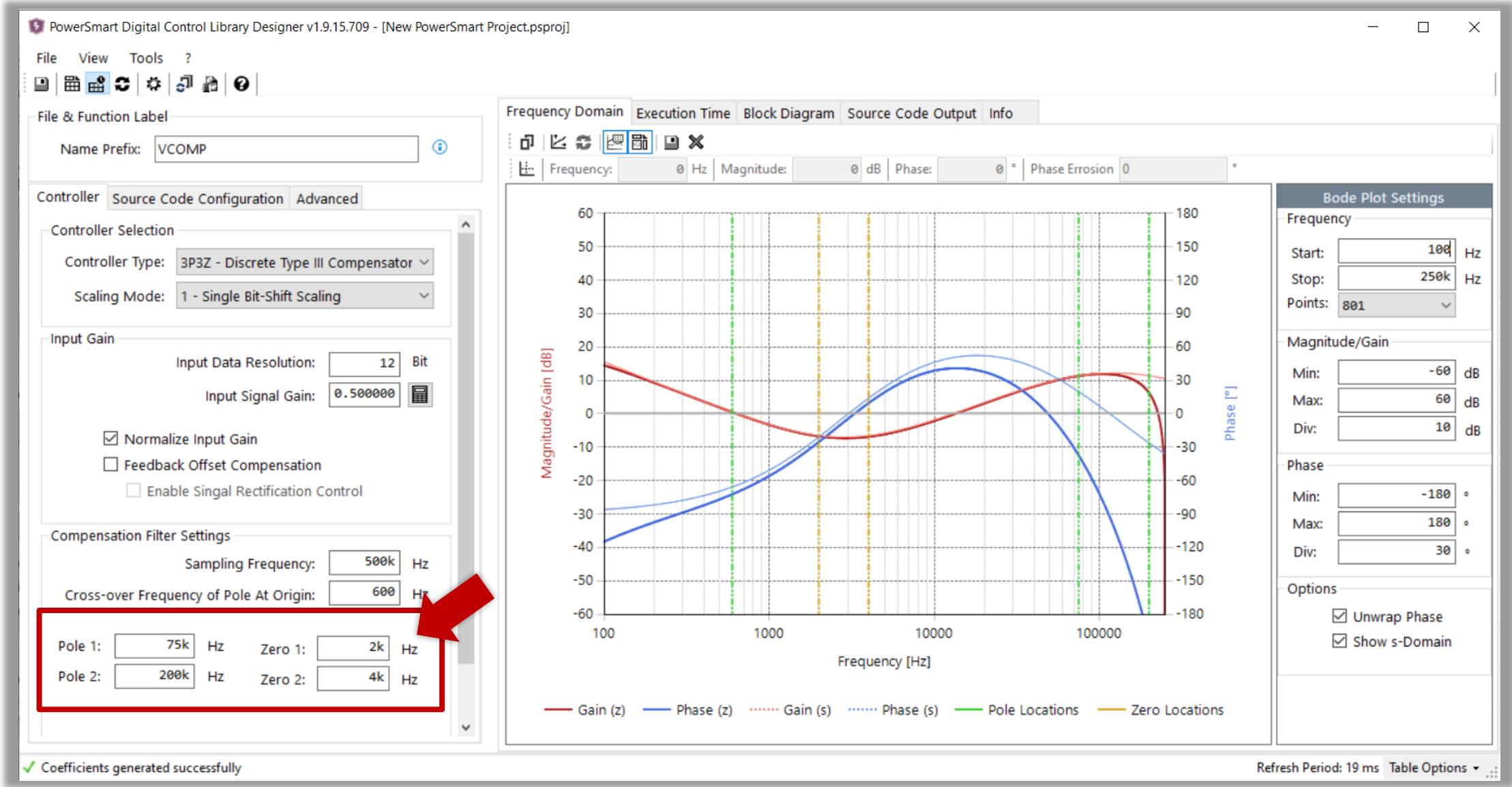
PS-DCLD VMC Feedback Loop Block Diagram



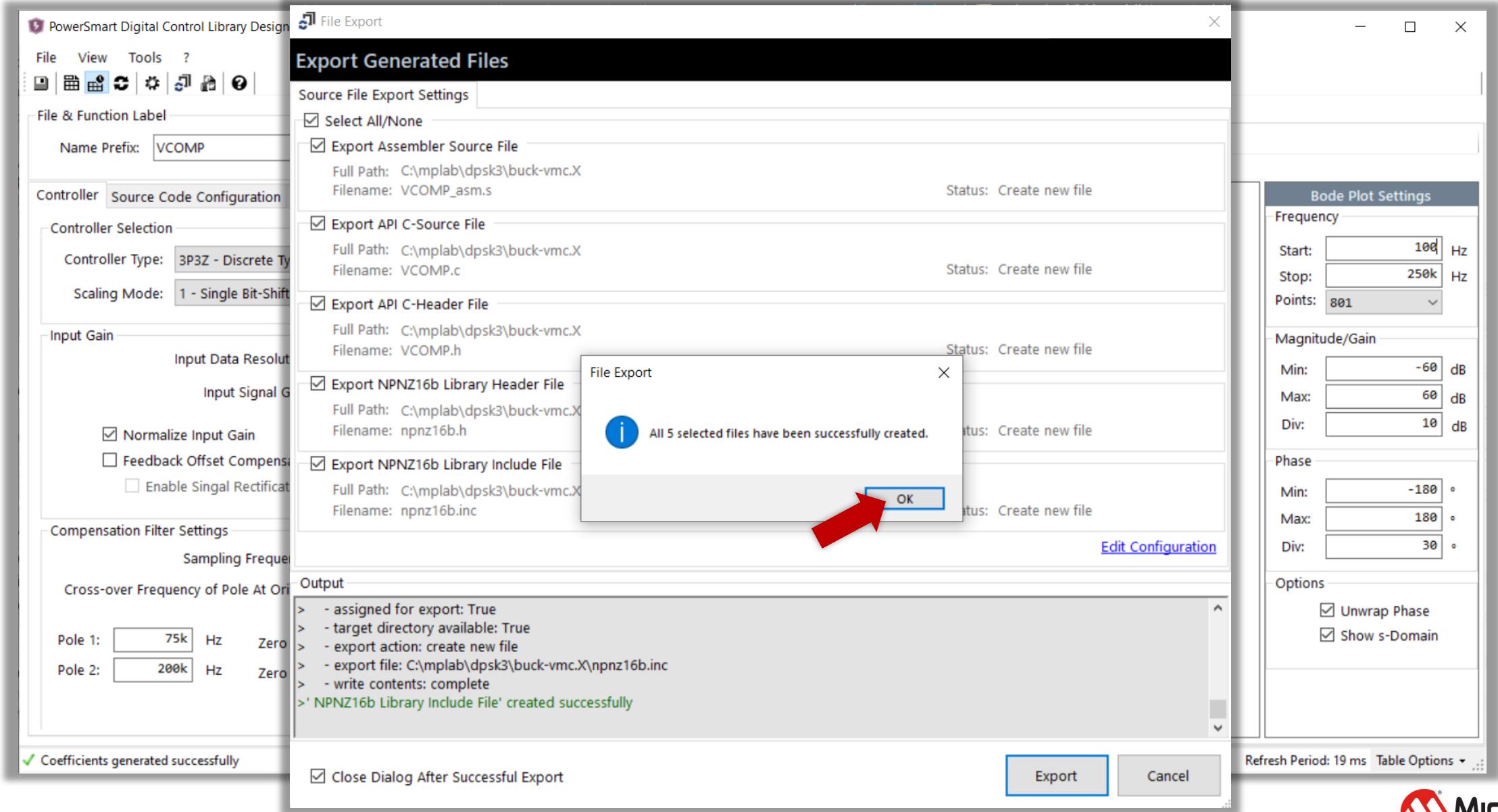
Control Loop Adjustment



Control Loop Adjustment



Control Loop Adjustment



Control Loop Implementation

```
void AN13_Int(void)
{
    adc_vin = ADCBUF12;
    adc_vbuck = ADCBUF13;
    // VCOMP_PTermUpdate (&VCOMP);
    VCOMP_Update (&VCOMP);
}
```



Replace P-Controller by new 3P3Z Controller

Testing Time !



Give it a try...



- **Build the Project**



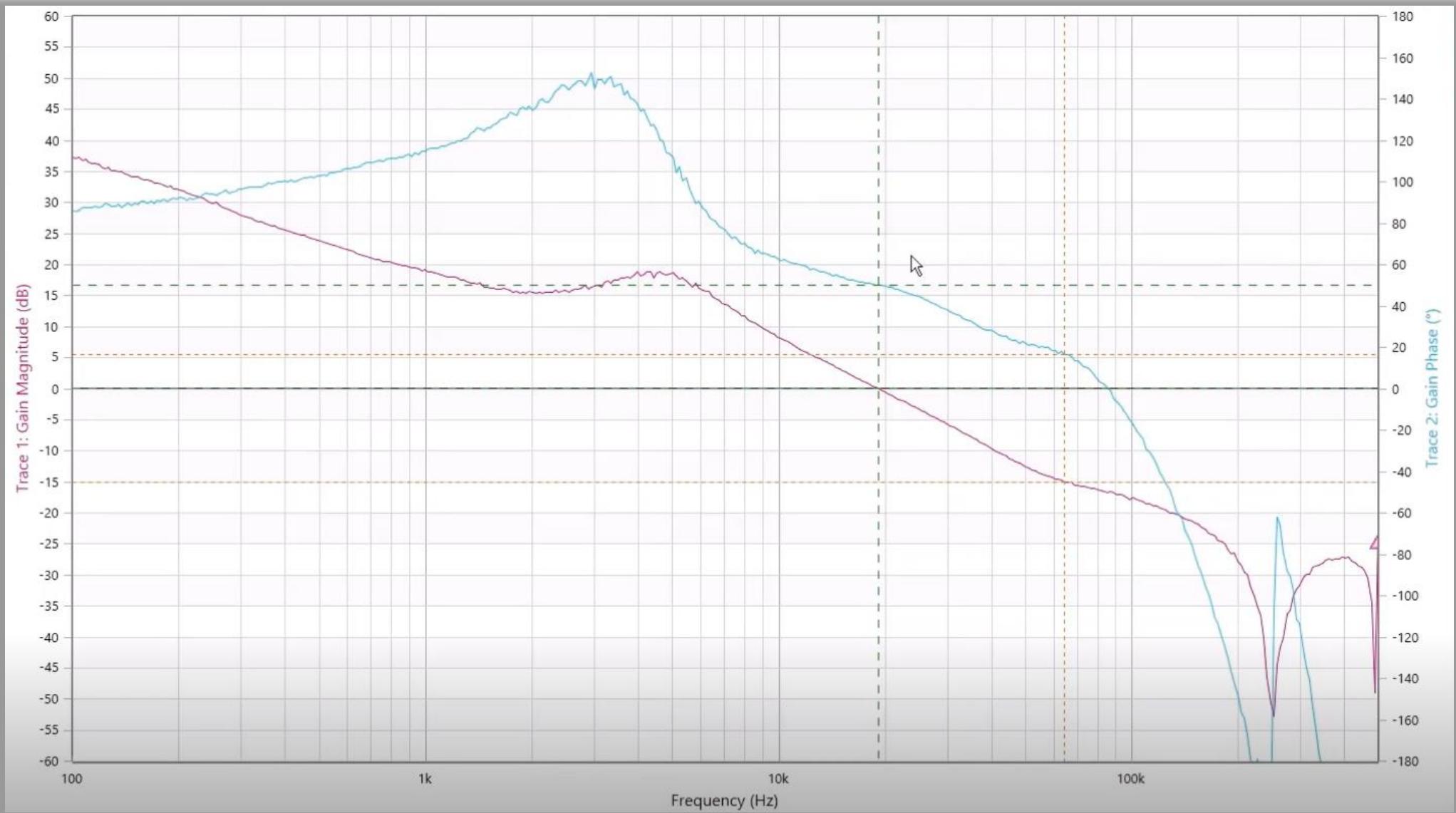
- If the Build was successful ...
- Or check error message and fix what's broken

- **Program the Project**



- When programming was successful, the red LED below dsPIC should blink

Bode Plot Results



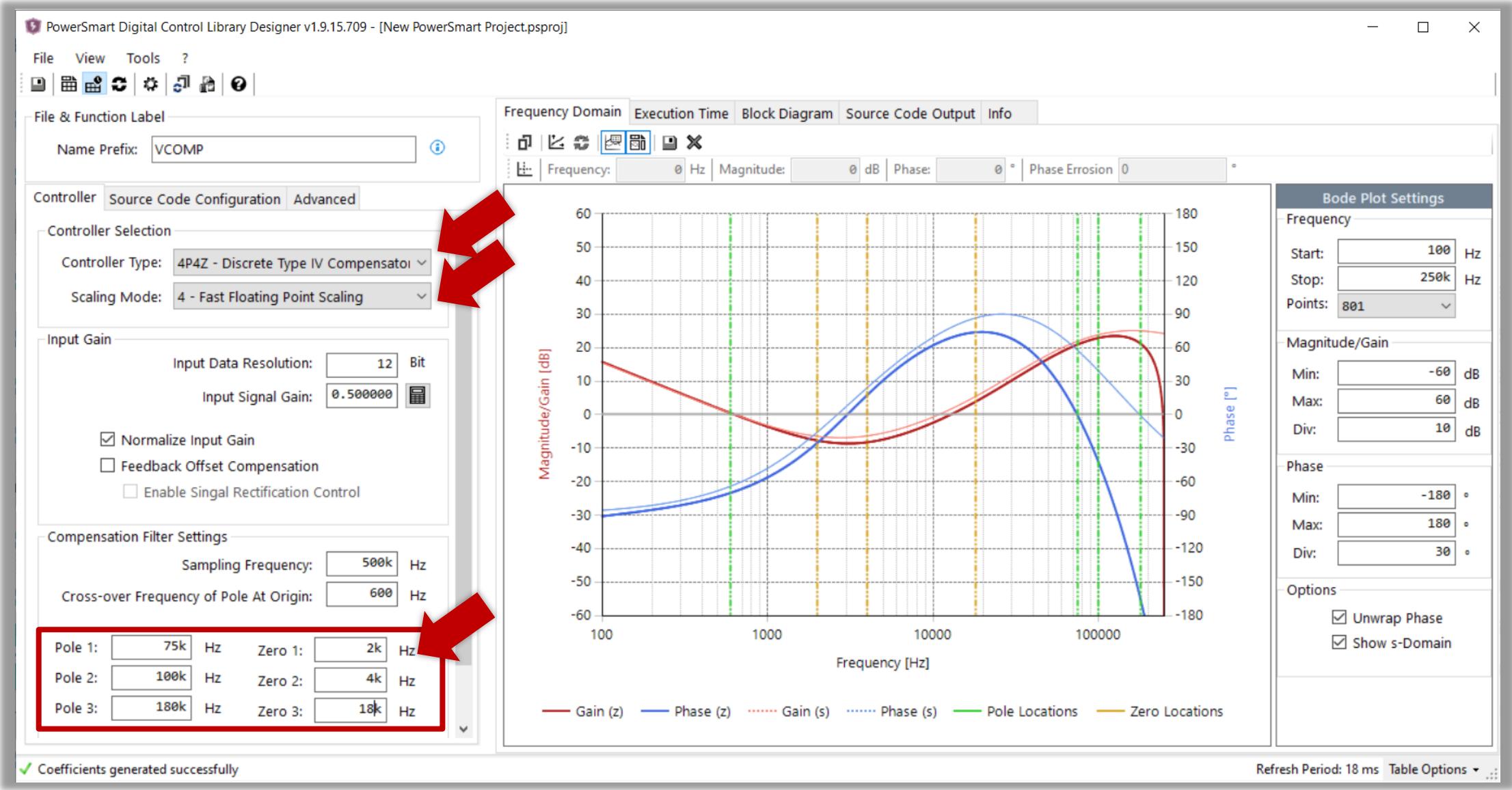
Agenda

Digital Power Supply from Scratch in 60 Minutes*

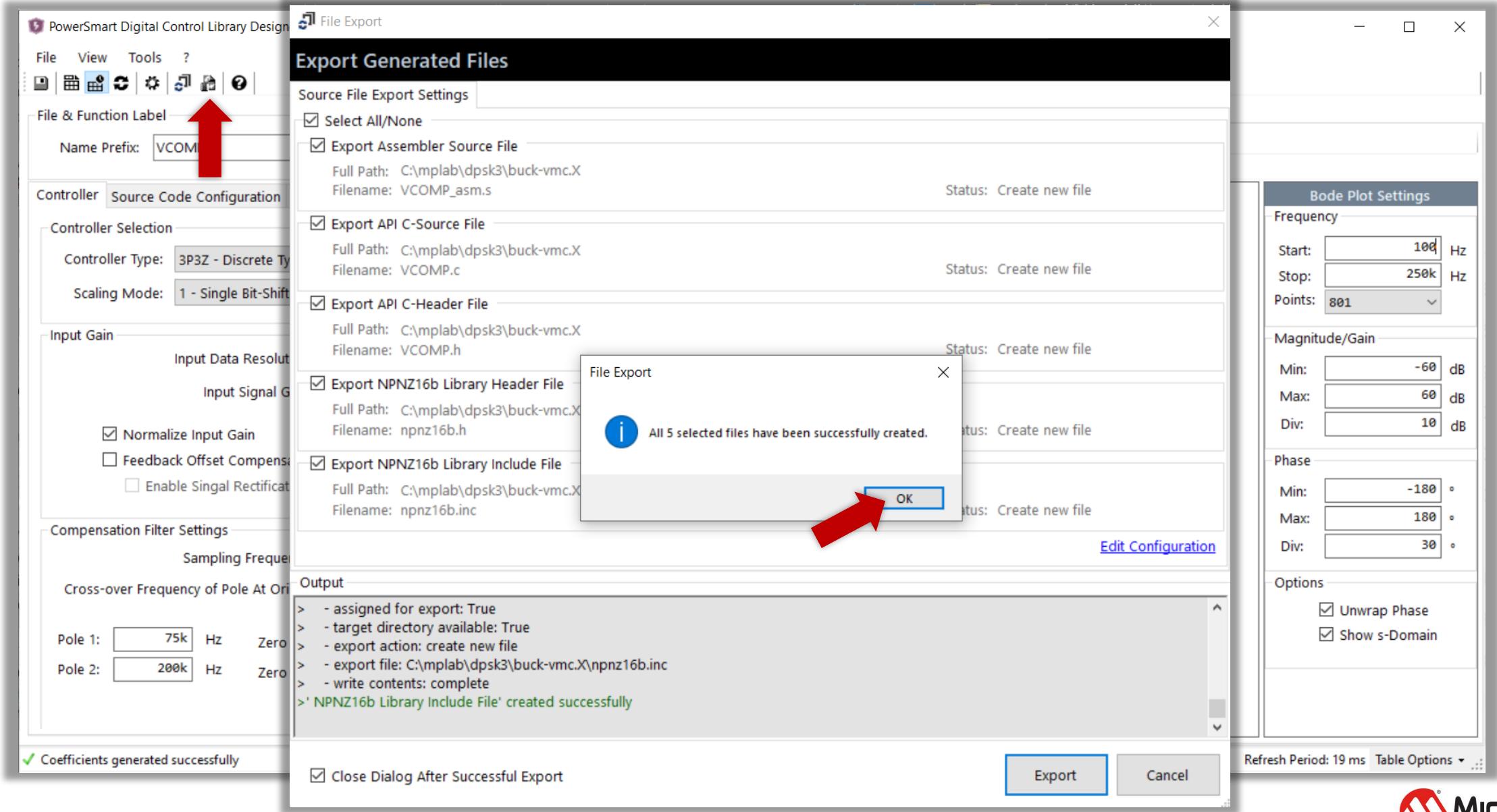
*: This will take forever!

- Overview Digital Power Starter Kit 3 (DPSK3)
- Requirements and Scope
- **Hands-On**
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB® PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start and Error Handling State Machine to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

Control Loop Adjustment



Control Loop Adjustment



Testing Time !



Give it a try...

- **Build the Project**

- If the Build was successful ...
- Or check error message and fix what's broken

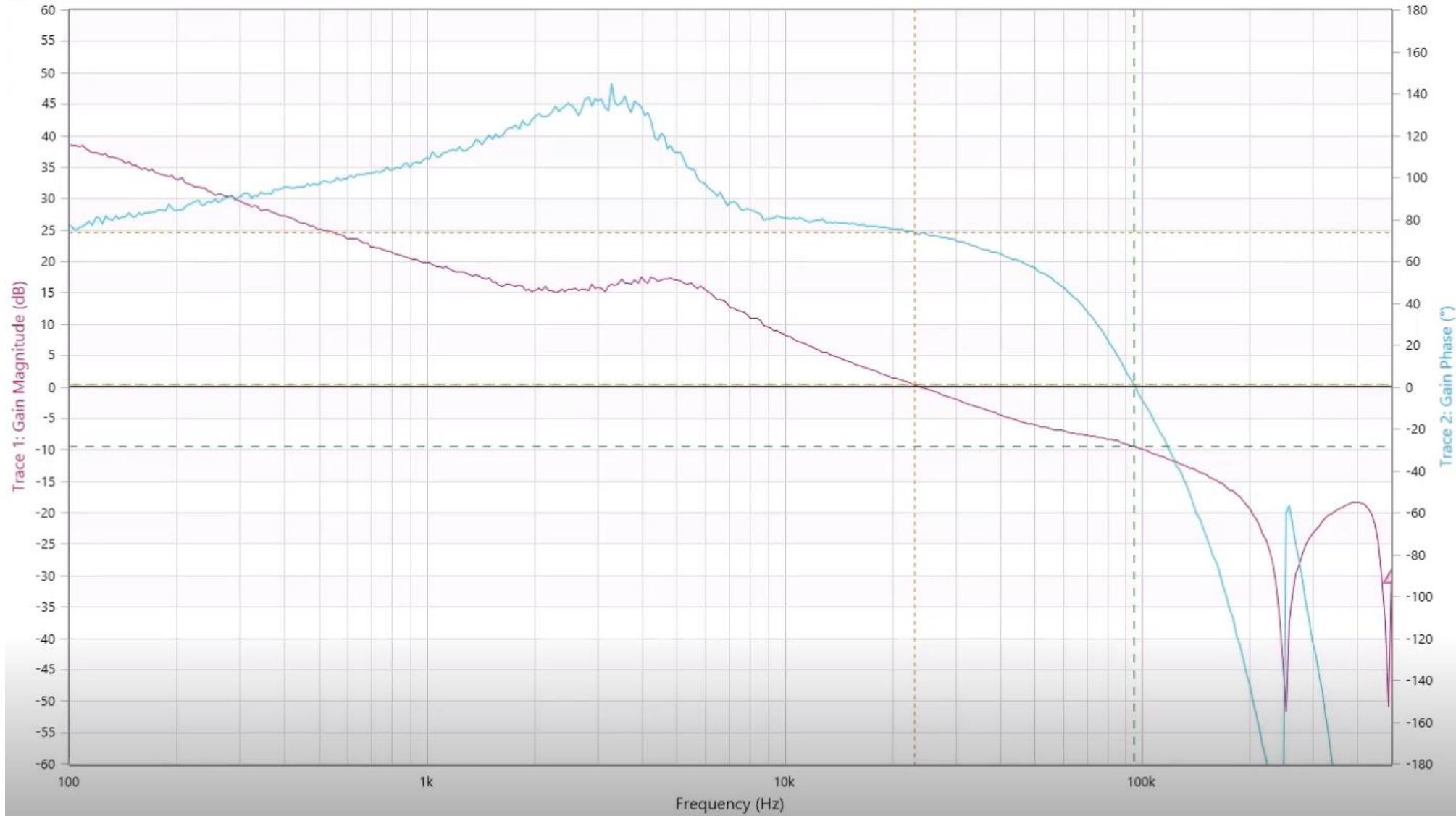


- **Program the Project**

- When programming was successful, the red LED below dsPIC should blink



Bode Plot Results



Agenda

Digital Power Supply from Scratch in 60 Minutes*

*: Curious to see how this is playing out

- Overview Digital Power Starter Kit 3 (DPSK3)
- Requirements and Scope
- Hands-On
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB® PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start and Error Handling State Machine to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

Appendix

MPLAB® PowerSmart™ SDK Beta Pre-Release Version

Digital Control Library Designer (PS-DCLD)

<https://microchip-pic-avr-tools.github.io/powersmart-dcld/>



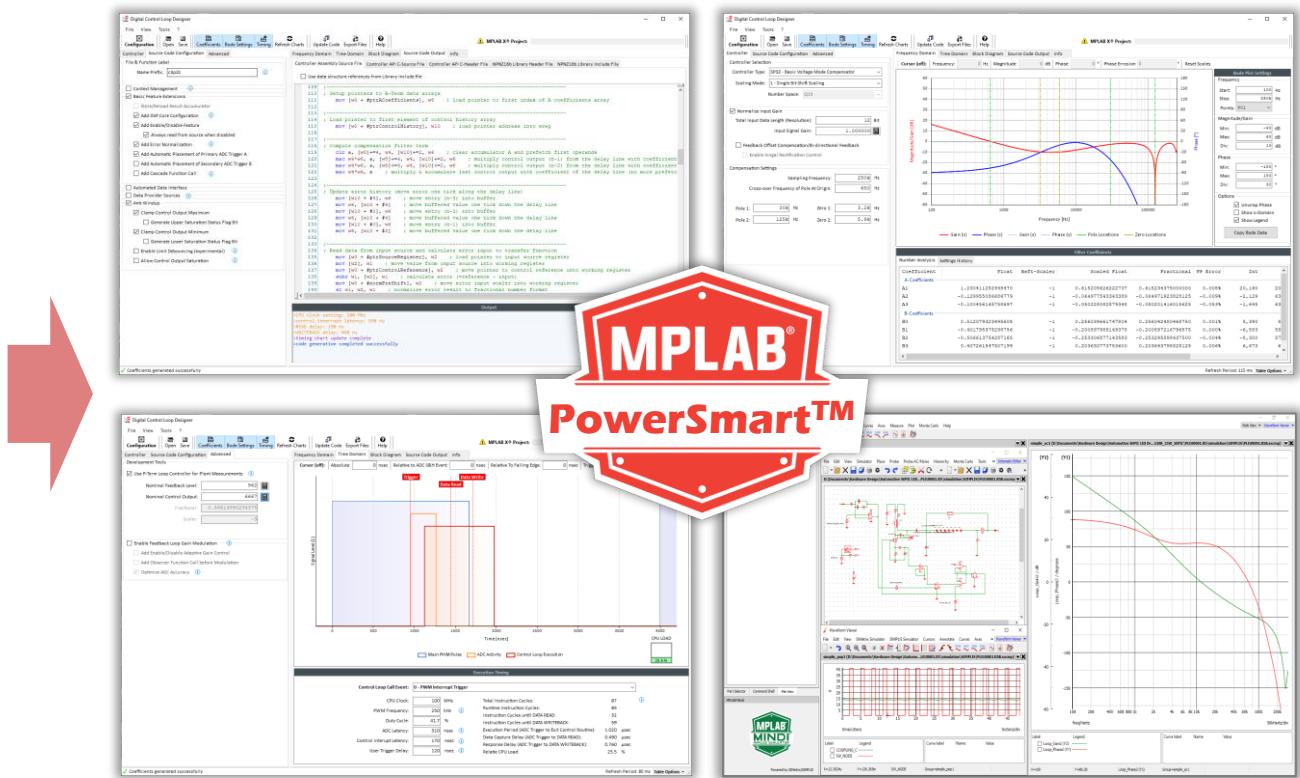
MPLAB® PowerSmart™ SDK
Digital Control Library Designer
Configuration Tool & Code Generator

MPLAB® PowerSmart™ SDK for Microchip dsPIC33® Digital Signal Controllers

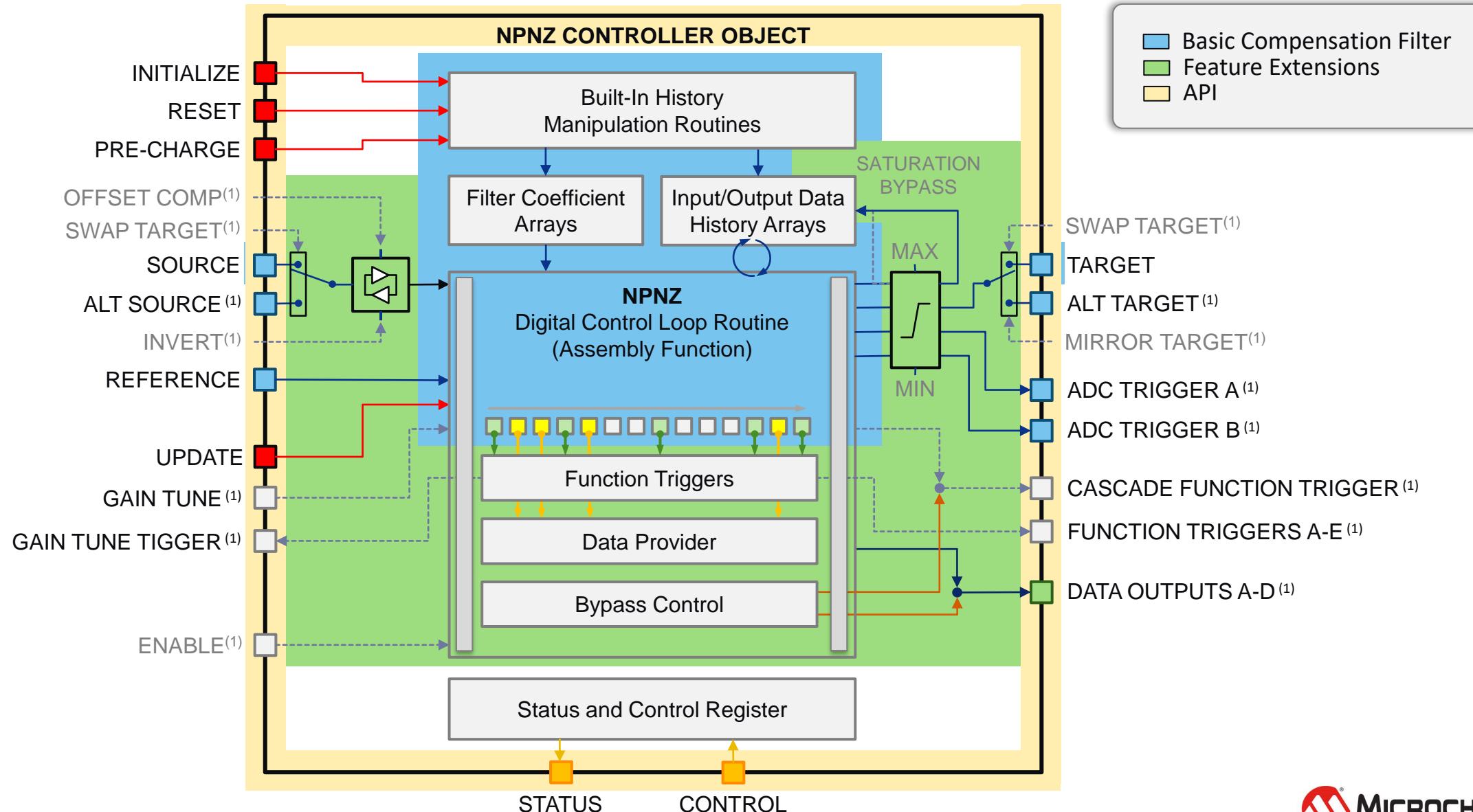
The MPLAB® PowerSmart™ SDK is a Software Development Kit (SDK) comprised of multiple, individual stand-alone tools for system definition, system modeling, code generation, control system fine tuning and real-time debugging of fully digital control systems for Switched-Mode Power Supplies (SMPS) supporting Microchip Technology's dsPIC® Digital Signal Controllers (DSC).

The major scope of this tool set is the rapid design of a digital power supply control stage rather than the power supply itself. This allows to simplify the design process to models based on interconnected transfer functions. These transfer functions are defined and configured in individual configuration windows. A transfer function can be based on generic Laplace-domain functions, being calculated at runtime or being defined by external data coming from network analyzer measurements or other third-party simulation tools such as MATLAB, SciLab, Simplis/SimMetrix, LTSpice, etc.

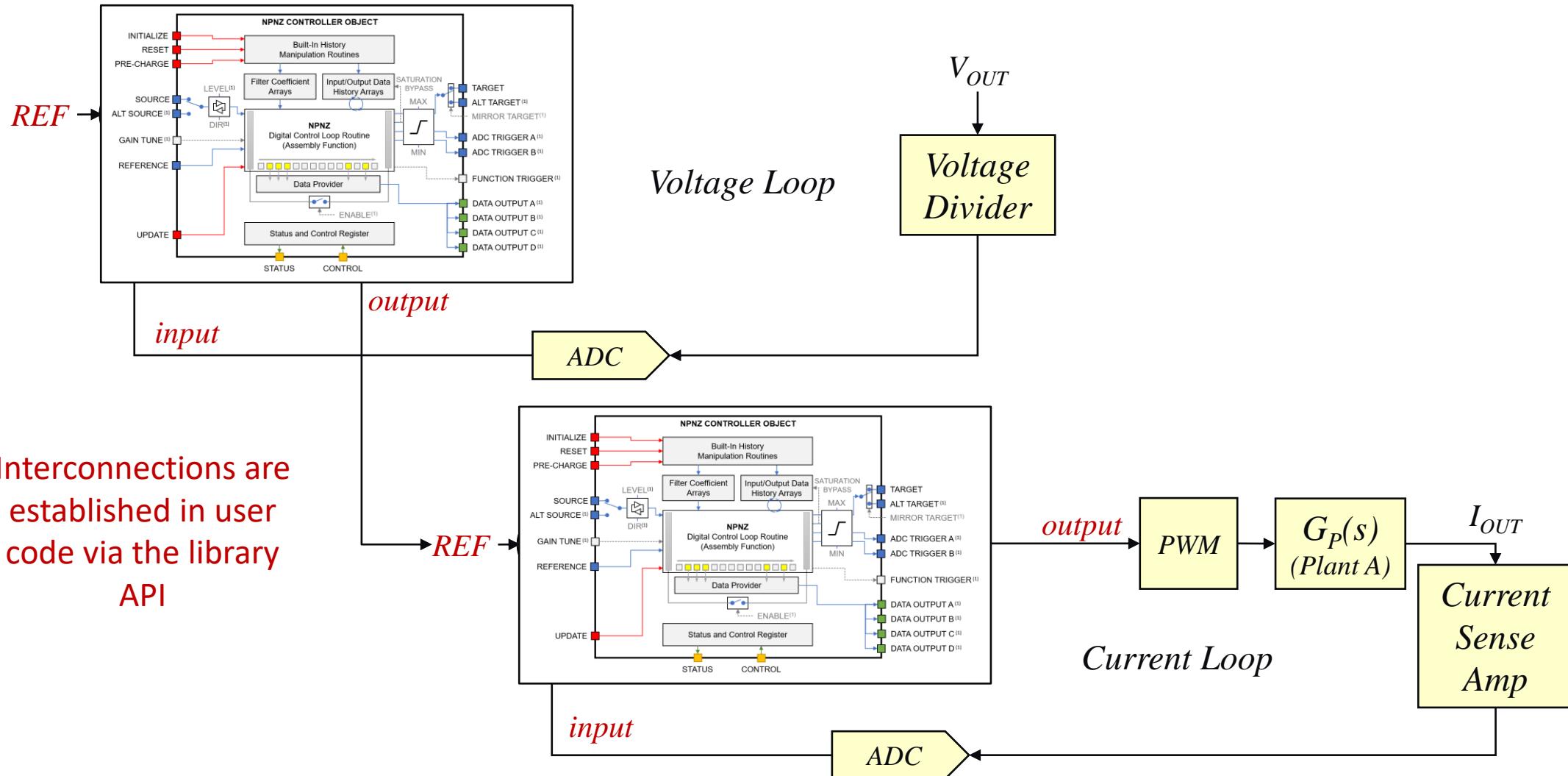
Digital Control Library Designer (PS-DCLD)



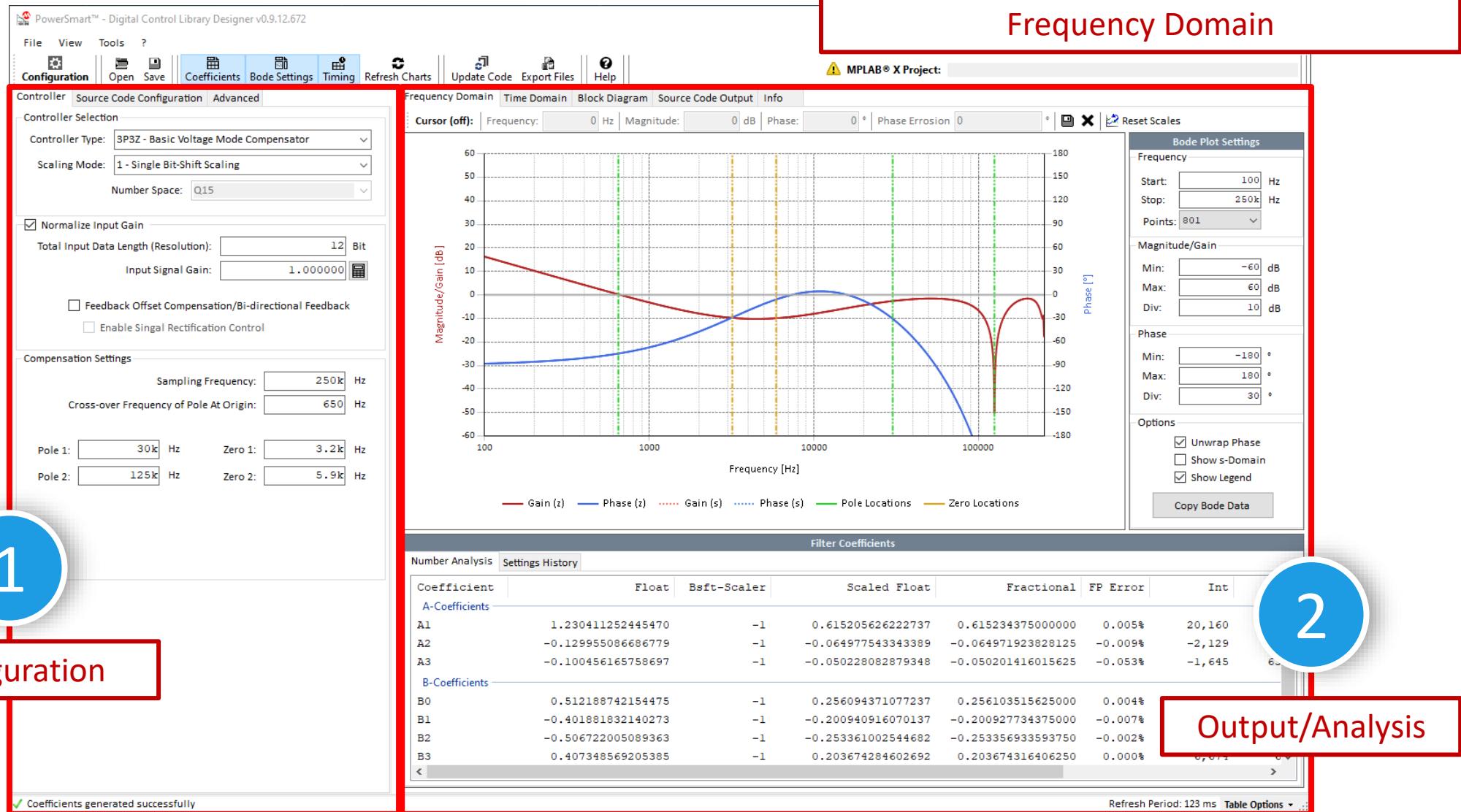
PS-DCLD Feedback Loop Block Diagram



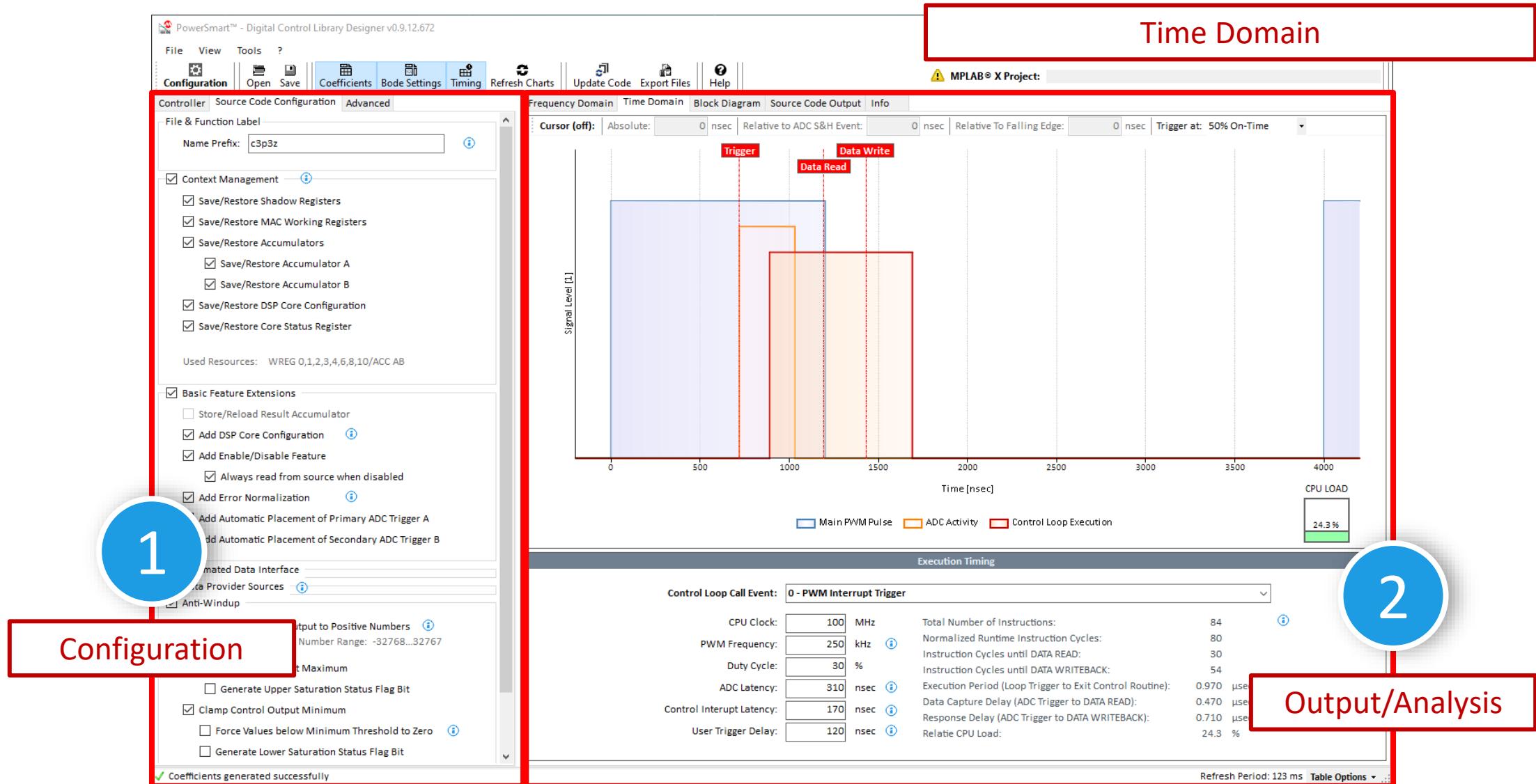
Using DCLD in Multi-Loop Systems



PowerSmart™ DCLD Main Window Overview



PowerSmart™ DCLD Main Window Overview



PowerSmart™ DCLD Main Window Overview

Block Diagram

1 Configuration

2 Output/Analysis

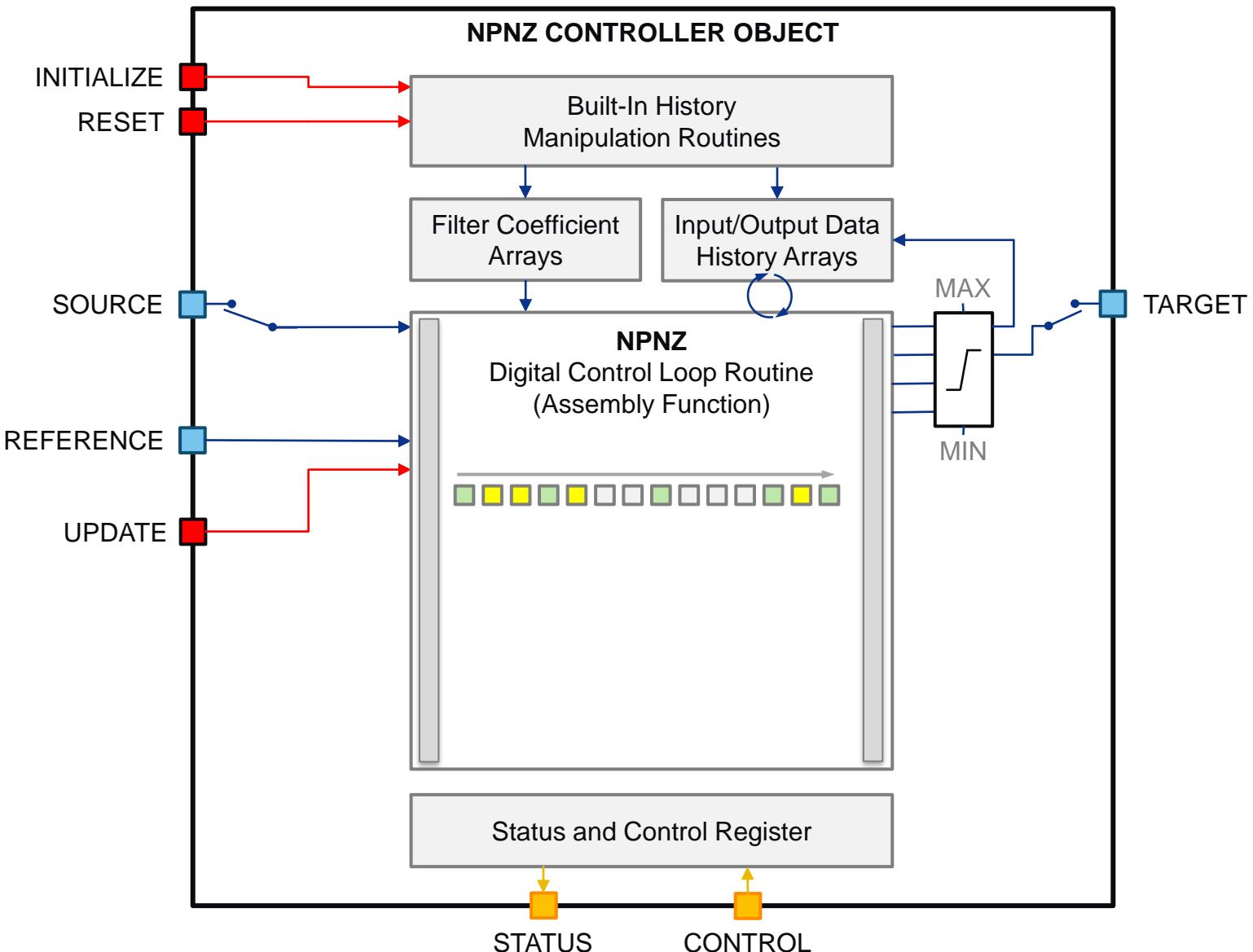
Coefficients generated successfully

PowerSmart™ DCLD Main Window Overview

The screenshot shows the PowerSmart DCLD Main Window interface. The window is divided into several sections:

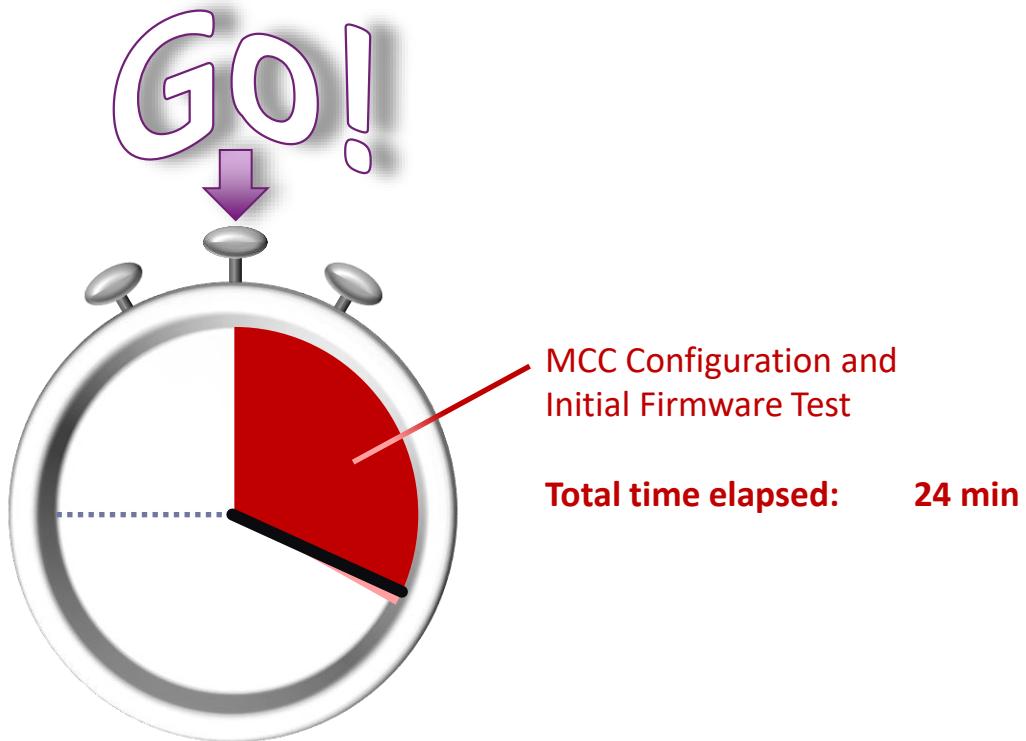
- Configuration (Section 1):** Located on the left, this section contains settings for controller configuration. It includes fields for "Name Prefix" (c3p3z), "Context Management" (checkboxes for Save/Restore Shadow Registers, MAC Working Registers, Accumulators, DSP Core Configuration, and Core Status Register), and "Basic Feature Extensions" (checkboxes for Store/Reload Result Accumulator, Add DSP Core Configuration, Add Enable/Disable Feature, Always read from source when disabled, Add Error Normalization, and Add Automatic Placement of Primary ADC Trigger A). A red box highlights the "Input to Positive Numbers" field, which has a number range of -32768...32767.
- Output/Analysis (Section 2):** Located on the right, this section displays generated assembly code in a large text area. The code is annotated with comments explaining its purpose, such as configuring the DSP for fractional operation and setting up pointers for A-term data arrays. Below the code, the "Output" pane shows logs of the generation process, including data read and write times, timing chart update completion, and code generation success. A red box highlights the message "Coefficients generated successfully".
- Code Generation (Section 3):** Located at the top right, this section is indicated by a red box around the "MPLAB® X Project:" status bar.

Buck Example Configuration



Let's continue...

Remaining Time until Controller Design Completion: 21 min

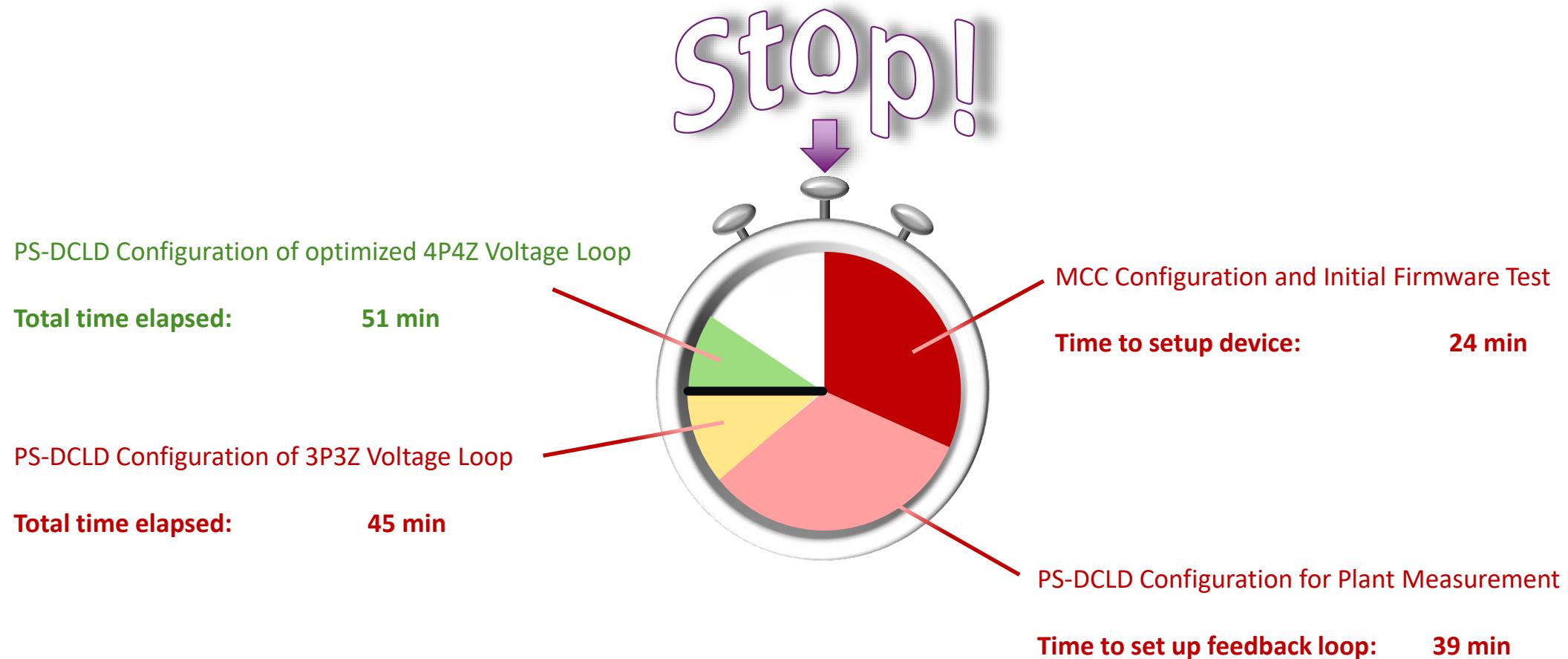


Control Configuration using MPLAB® PowerSmart™

Building Up a Digital Power Supply
from Scratch in 45 Minutes

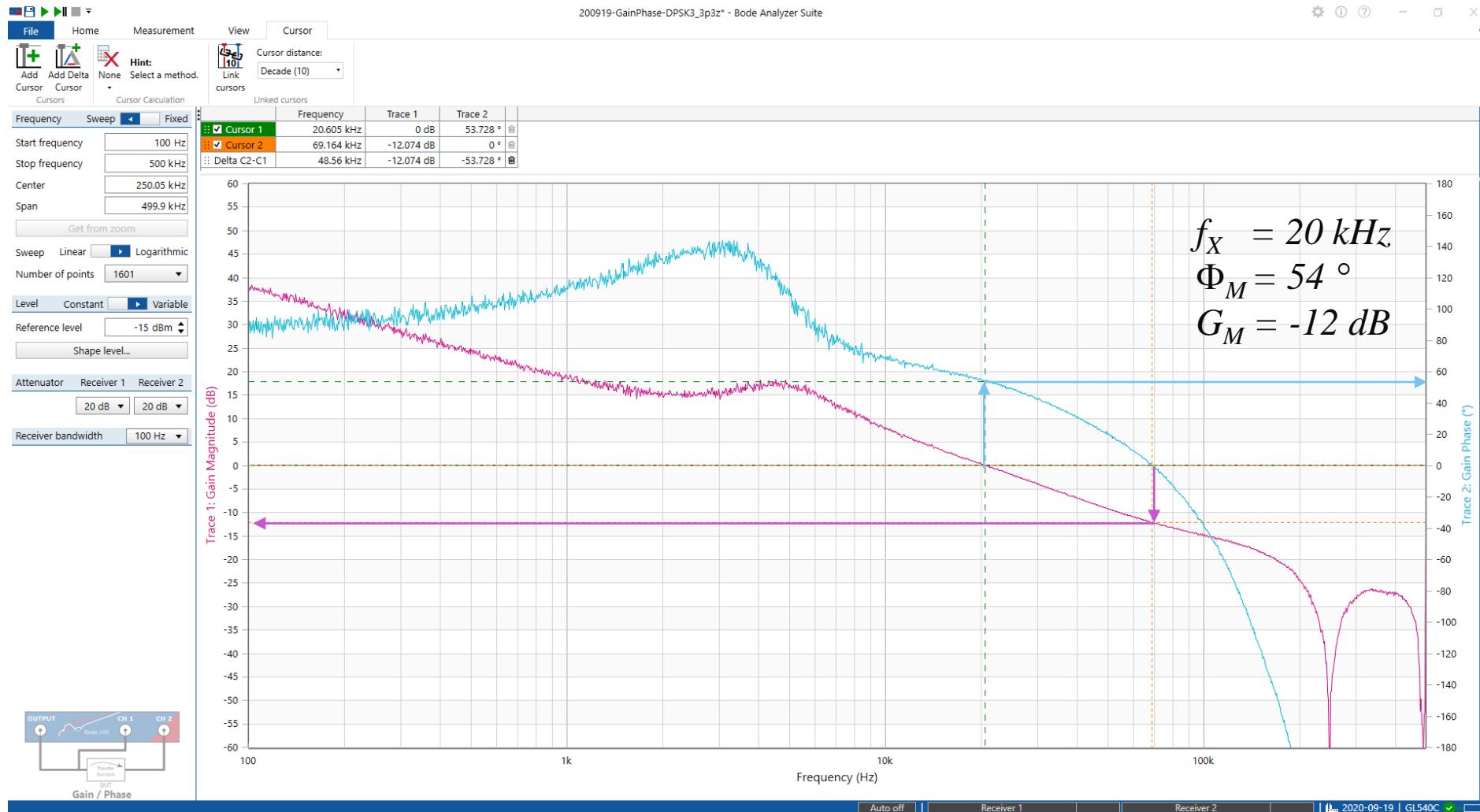
Design Time Review

Remaining Time until Controller Design Completion: 0 min



Final Test Result

$V_{IN}=9\text{ V}$, $V_{OUT}=3.3\text{ V}$, $I_{LOAD}=50\%$



Digital Power

- **Getting Started in Digital Power**

- Intelligent Power Design Center:

- <https://www.microchip.com/power>

- **How-2 Starter Kits**

- Digital Power Starter Kit 3 (Part-No. DM330017-3):

- <https://www.microchip.com/dm330017-3>

- **Training:**

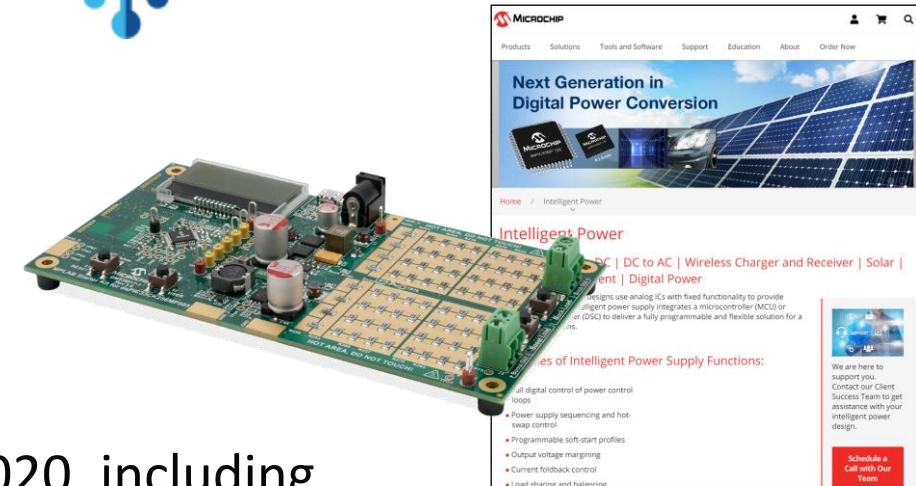
- Microchip University (Virtual Training Platform):

- <https://secure.microchip.com/mu>

Please note:

All Face-2-Face workshops have been suspended in early 2020, including the well-known MASTERs conferences usually conducted in 9 nations around the globe. For the time being all available trainings have been moved to our new virtual training platform

Microchip University.



Digital Power Design Resources



- Find hundreds of code examples and design tools on
<https://discover.microchip.com>

The screenshot shows the MPLAB® PowerSmart™ interface. It includes several windows: a main window showing a waveform graph with red and blue traces; a 'Digital Controller Designer' window with code editor and configuration tabs; a 'Block Diagram' window showing a complex circuit diagram; and a 'Source Code Output' window displaying assembly language code. A large red shield-shaped overlay with the text 'MPLAB® PowerSmart™' is positioned in the center.

The screenshot shows the MPLAB® DISCOVER web application interface. At the top, there's a navigation bar with a search bar and links for 'Open with' and 'Download'. Below it is a section titled 'Offerings' with a dropdown menu set to 'v1.0.0'. Under 'IDE Examples', there's a list of offerings including 'EPC9143 300W 16th Brick Non-Isolated Step Down Converter Advanced Voltage Mode Control Firmware'. To the right of this list is a 'About' button and a 'Details' button. Further down, there's a 'PIC-IoT AWS Sensor Node' and a 'PIC24F Hello World UART' section. At the bottom, there are two images of a green printed circuit board (PCB) labeled 'Top View' and 'Bottom View', with various component labels like 'EPC9143', 'EPC242', and '2014-003 PCB# B5194'.

Agenda

Building Up a Digital Power Supply from Scratch in 45 Minutes

- Introduction EU PFG
- Overview Digital Power Starter Kit 3 (DPSK3)
- Live Demo:
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

Questions and Answers

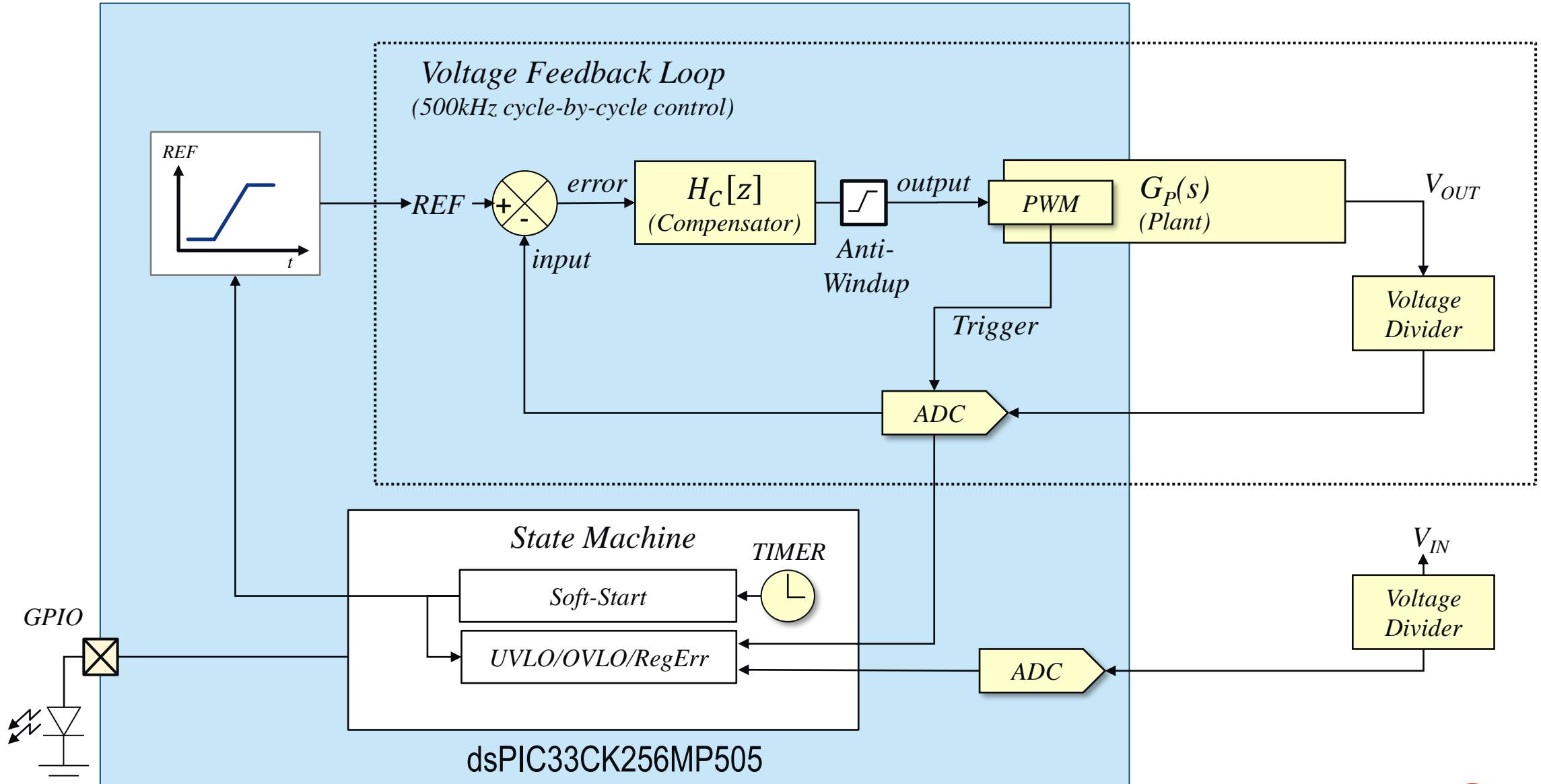
Appendix

Agenda

Building Up a Digital Power Supply from Scratch in 45 Minutes

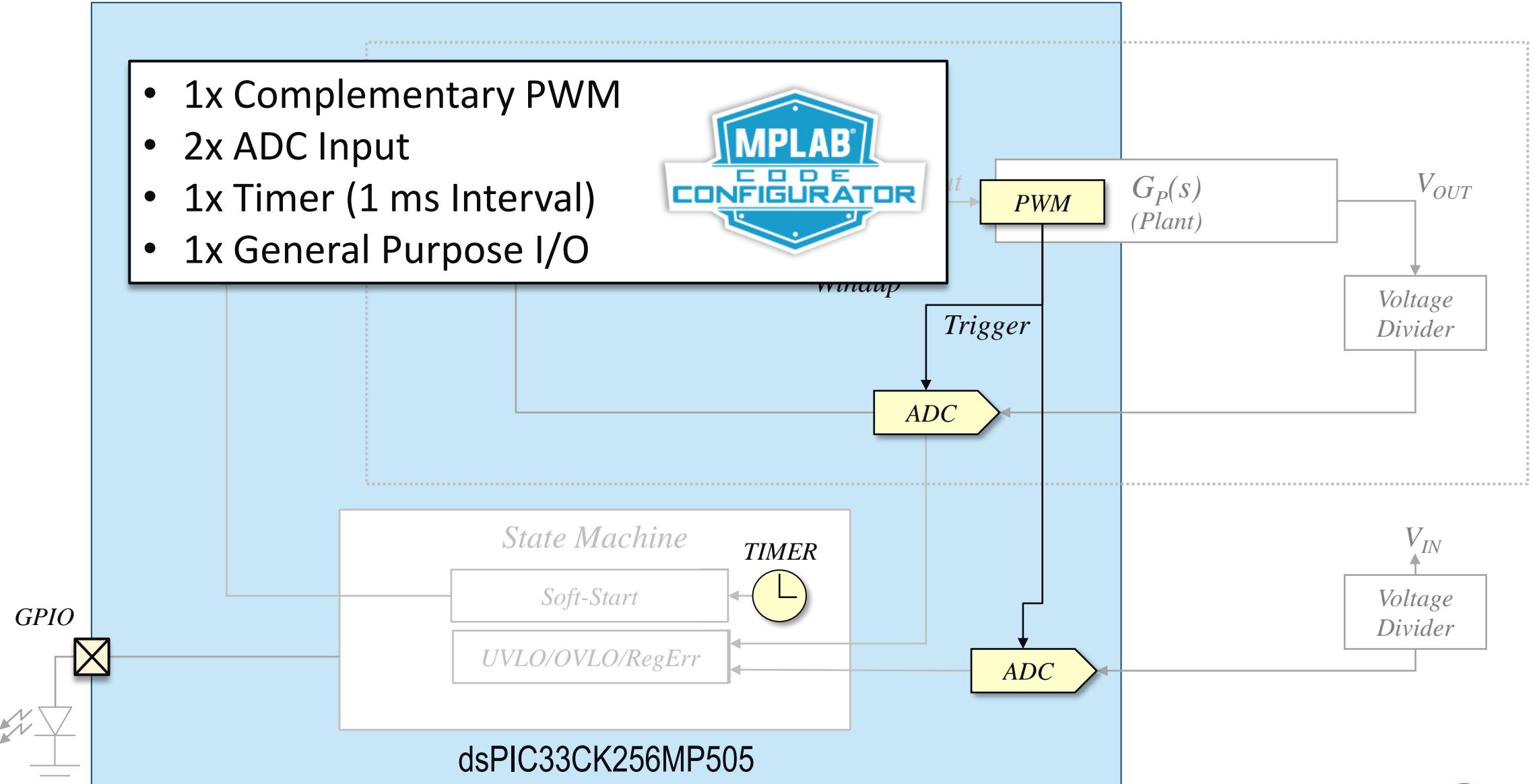
- Introduction EU PFG
- Overview Digital Power Starter Kit 3 (DPSK3)
- **Live Demo:**
 - Setting up the Project
 - Configuring Device and Peripherals
 - Using MPLAB PowerSmart™ to Build an Initial Loop
 - Adding Simple Soft Start to Project
 - Final Controller Selection and Adjustment
 - Loop Measurement & Optimization
- Q&A

Requirements



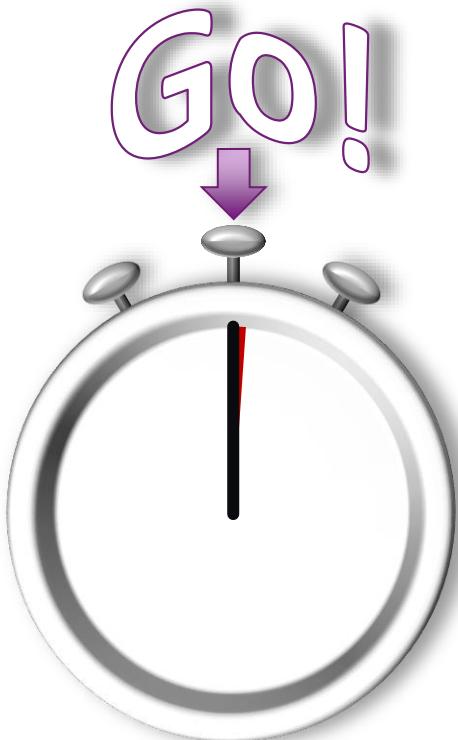
Requirements

- 1x Complementary PWM
- 2x ADC Input
- 1x Timer (1 ms Interval)
- 1x General Purpose I/O



This is from where we start...

Remaining Time until Controller Design Completion: 45 min



Open MCC in MPLAB® X IDE

The screenshot shows the MPLAB X IDE interface with the Microcontroller Configuration Center (MCC) plugin open. The main window is titled "System Module".

System Module Configuration:

- Clock:** FRC Oscillator (8.0 MHz) is selected.
- PLL Enable:** Enabled with Prescaler 1:1, FeedBack 1:100, Postscaler1 1:2, and Postscaler2 1:1.
- Auxiliary Clock:** FRC is selected as the clock source. PLL Enable is checked. Prescaler 1:1, Feedback 1:125, Postscaler1 1:2, and Postscaler2 1:1 are configured.
- VCO & AVCO:** This section is currently empty.

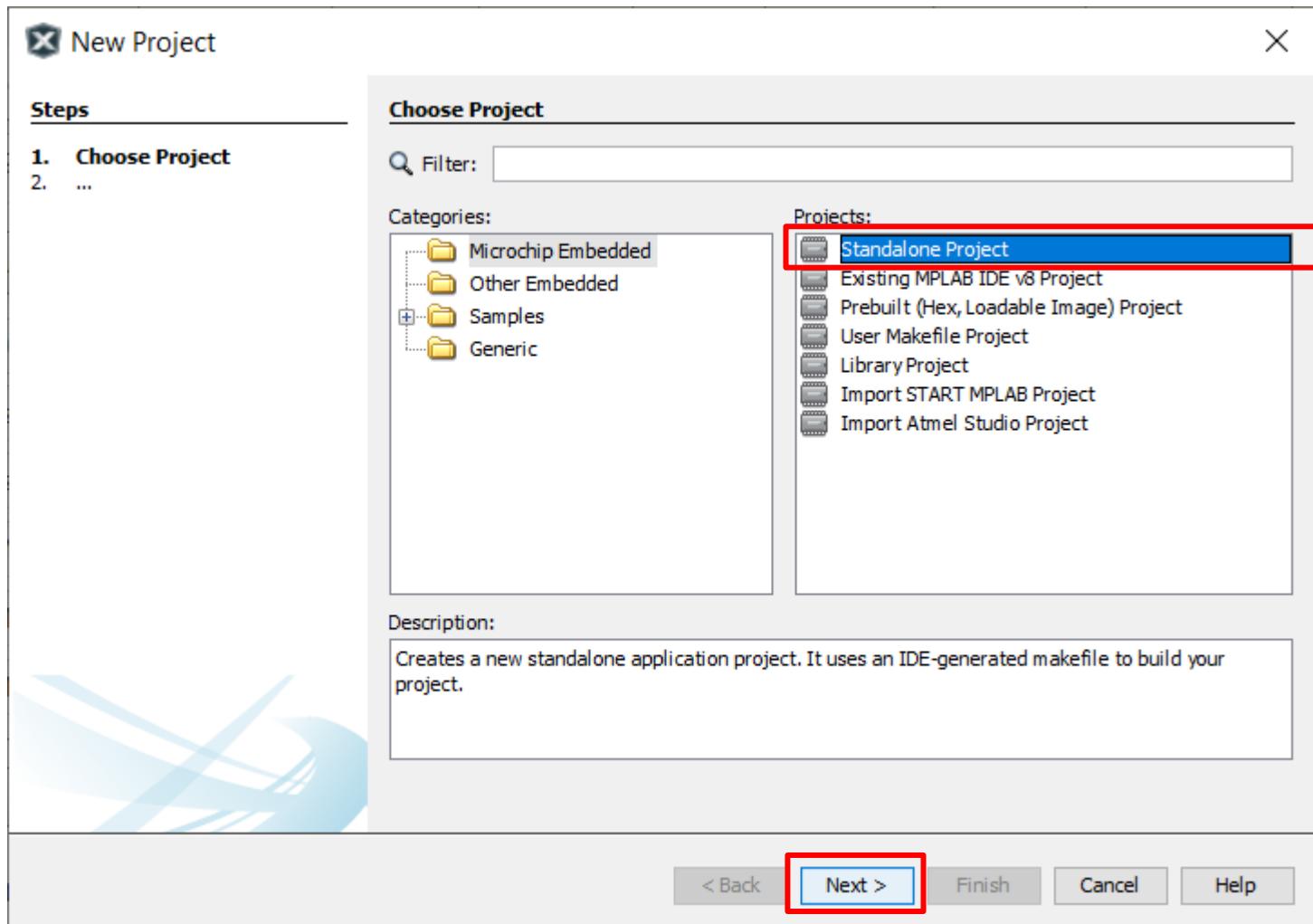
Pin Manager: Package View:

The pin map for the dsPIC33CK256MP505 is shown, displaying pin numbers 1 through 37. The pins are color-coded according to their function:

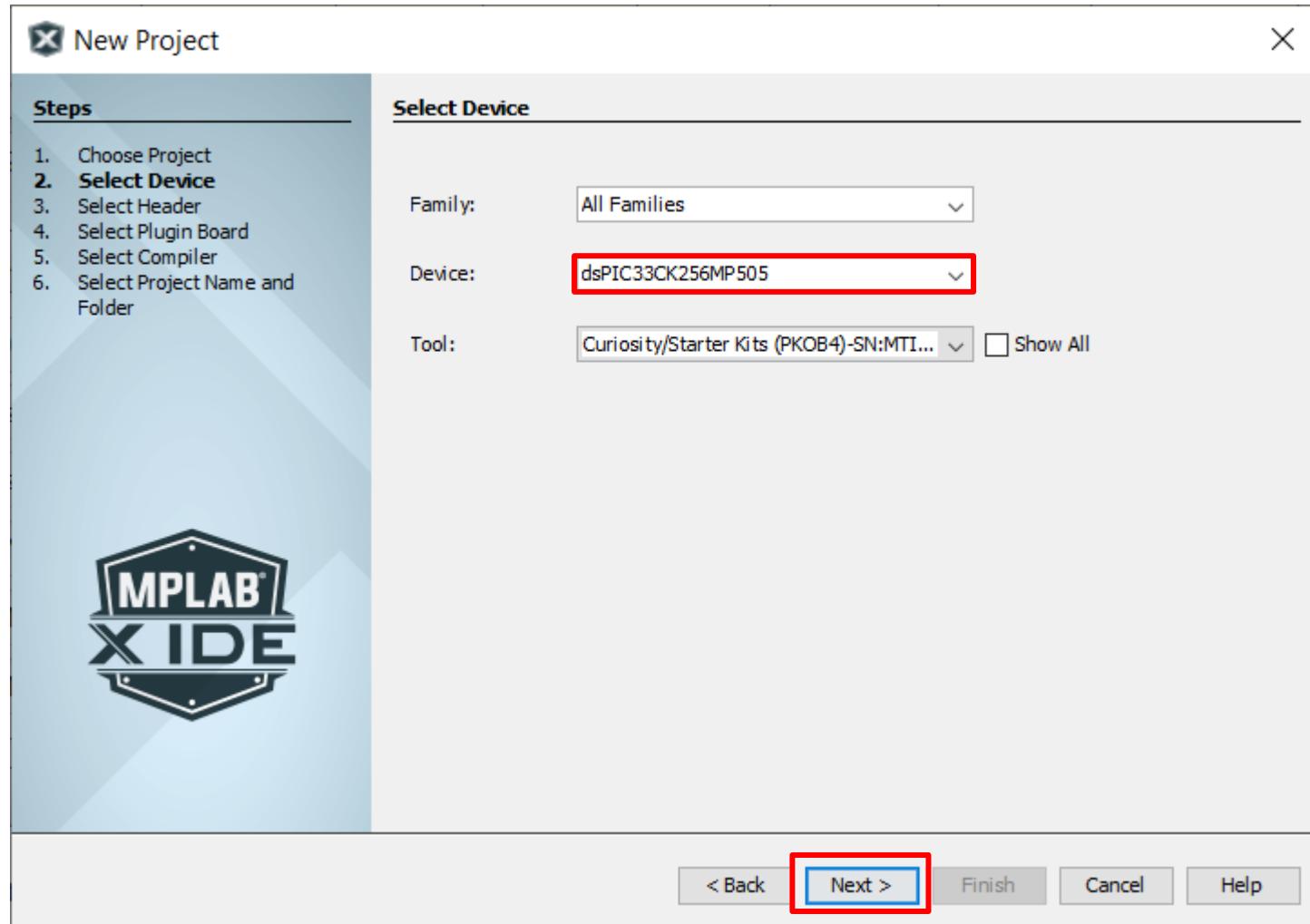
- RB Port:** RB14|PWM1-H (pin 1), RB15|PWM1-L (pin 2), RB12 (pin 3), RB11 (pin 4), RB10 (pin 5), RB9|PGC1 (pin 6), RB8|PGD1 (pin 7), RB7 (pin 8), RB6|USER_LED|GPIO (pin 9), RB5 (pin 10), RB4 (pin 11), RB3 (pin 12), RB2 (pin 13), RB1 (pin 14), RB0 (pin 15), RB13 (pin 16), RB12 (pin 17), RB11 (pin 18), RB10 (pin 19), RB9 (pin 20), RB8 (pin 21), RB7 (pin 22), RB6 (pin 23), RB5 (pin 24), RB4 (pin 25), RB3 (pin 26), RB2 (pin 27), RB1 (pin 28), RB0 (pin 29), RC10 (pin 30), RC9 (pin 31), RC8 (pin 32), RC7 (pin 33), RC6 (pin 34), RC5 (pin 35), RC4 (pin 36), and RC3 (pin 37).
- RA Port:** RA0 (pin 1), RA1 (pin 2), RA2 (pin 3), RA3 (pin 4), RA4 (pin 5), RA13 (pin 6), RA12 (pin 7), RA11 (pin 8), RA10 (pin 9), RA9 (pin 10), RA8 (pin 11), RA7 (pin 12), RA6 (pin 13), RA5 (pin 14), RA4 (pin 15), RA3 (pin 16), RA2 (pin 17), RA1 (pin 18), RA0 (pin 19), RA14 (pin 20), RA13 (pin 21), RA12 (pin 22), RA11 (pin 23), RA10 (pin 24), RA9 (pin 25), RA8 (pin 26), RA7 (pin 27), RA6 (pin 28), RA5 (pin 29), RA4 (pin 30), RA3 (pin 31), RA2 (pin 32), RA1 (pin 33), and RA0 (pin 34).
- RC Port:** RC13 (pin 1), RC12 (pin 2), RC11 (pin 3), RC10 (pin 4), RC9 (pin 5), RC8 (pin 6), RC7 (pin 7), RC6 (pin 8), RC5 (pin 9), RC4 (pin 10), RC3 (pin 11), RC2 (pin 12), RC1 (pin 13), and RC0 (pin 14).
- Other:** VSSA (pin 15), VDD2A (pin 16), and VSS2A (pin 17).

MPLAB CODE CONFIGURATOR

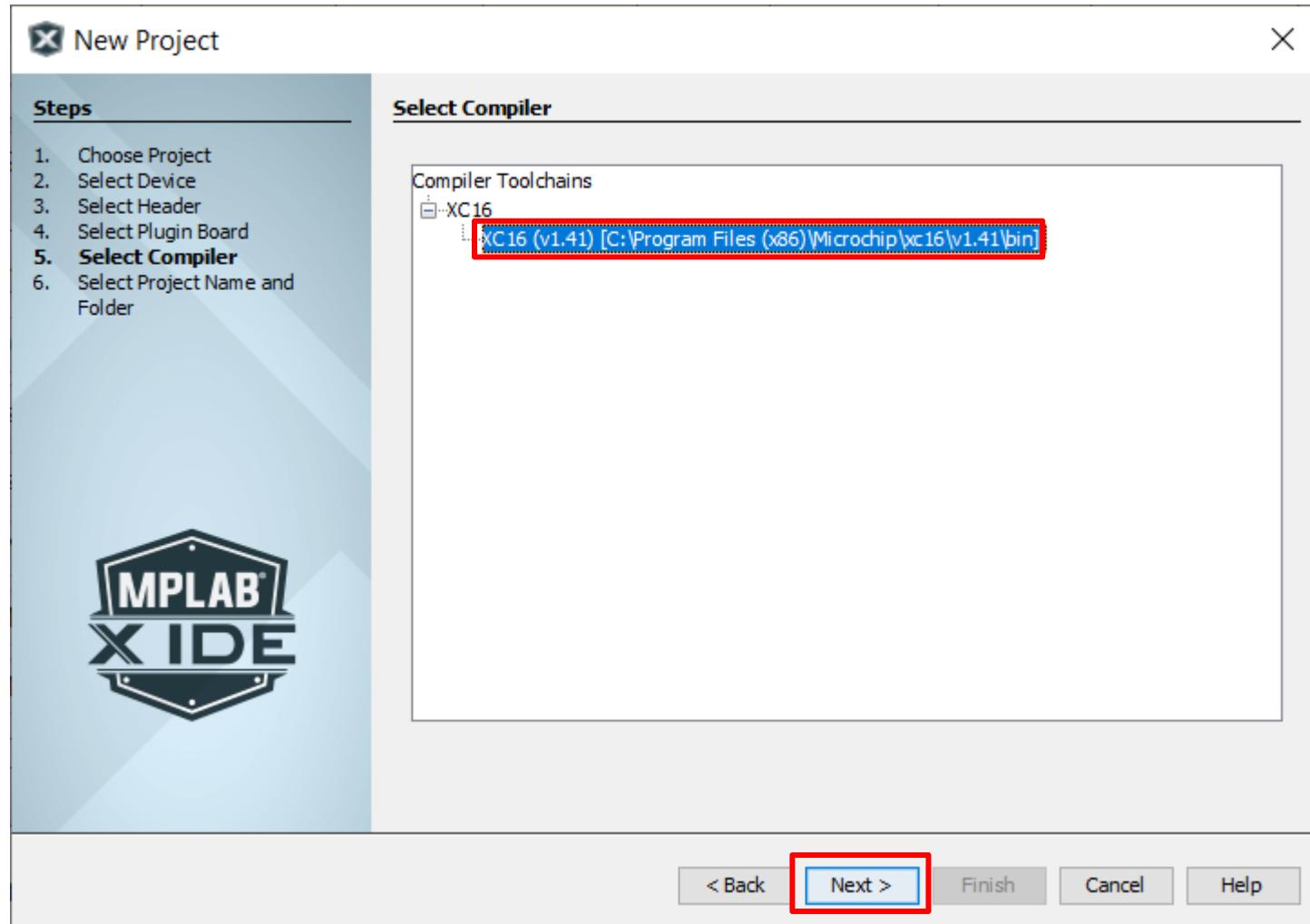
Creating the Project



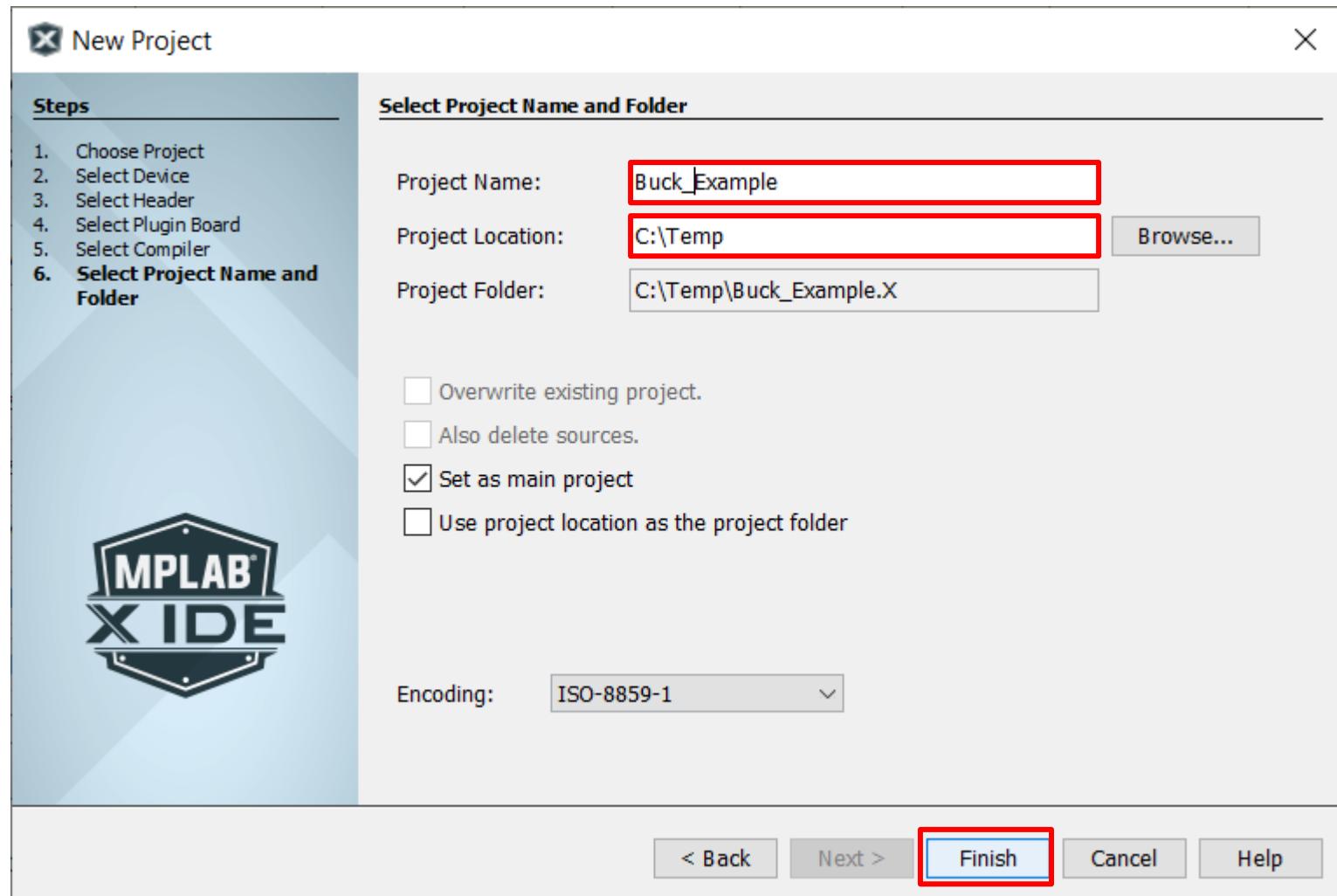
Creating the Project



Creating the Project



Creating the Project



System Configuration in MCC

1

2

3

Project Resource...

System Module

Easy Setup Registers

Clock

8000000 Hz FRC Oscillator (8.0 MHz) Clock Source

FRC Postscaler

PLL Enable

8 MHz 1:1 Prescaler

800 MHz 1:100 FeedBack

400 MHz 1:2 Postscaler1

400 MHz 1:1 Postscaler2

200 MHz Fosc

100 MHz Fosc/2

Auxiliary Clock

8000000 Hz FRC Clock Source

PLL Enable

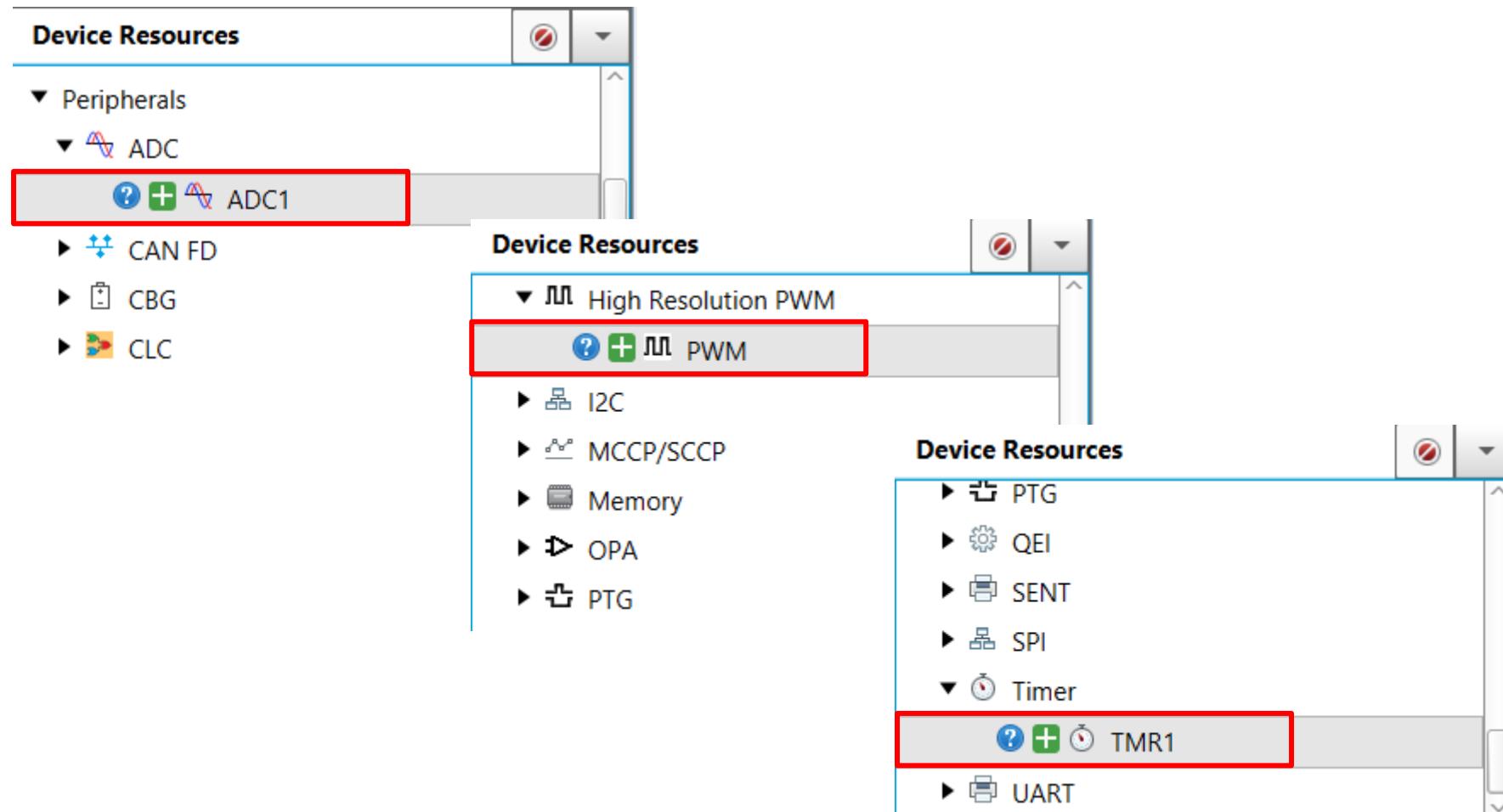
8 MHz 1:1 Prescaler

1000 MHz 1:125 FeedBack

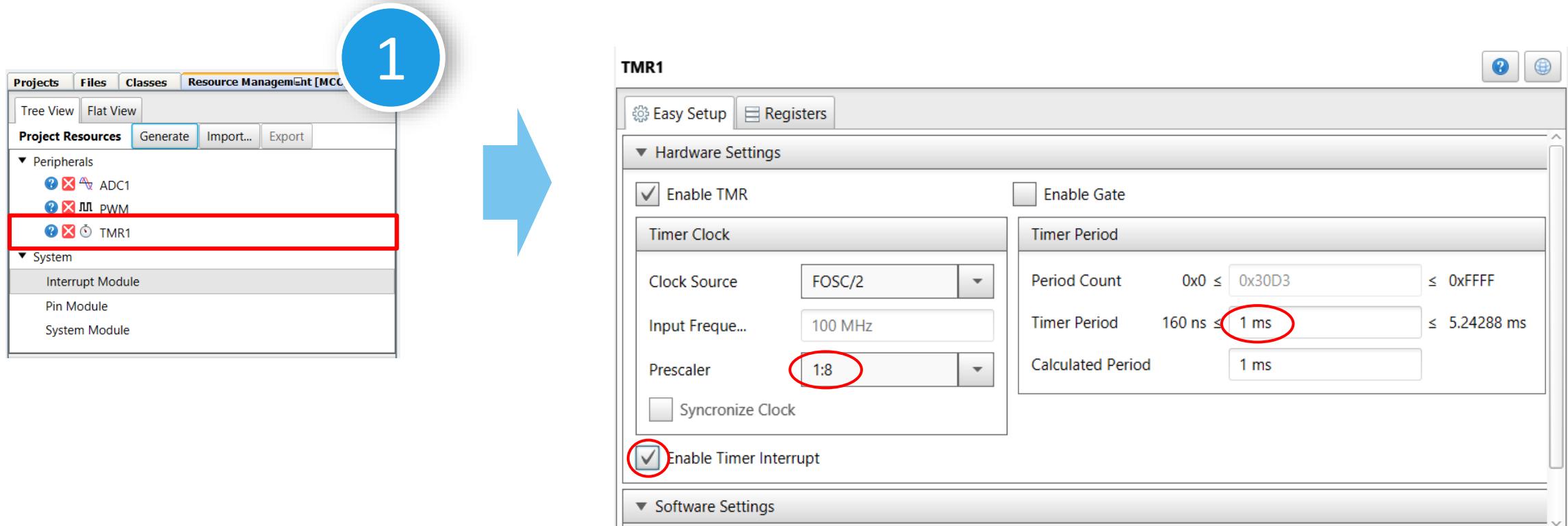
500 MHz 1:2 Postscaler1

500 MHz 1:1 Postscaler2

Peripherals Selection in MCC



Timer1 Configuration in MCC



ADC Configuration in MCC

1

The diagram illustrates the process of configuring an ADC in the Microchip Configuration Center (MCC). It starts with the 'Resource Management [MCC]' interface, where the 'Project Resources' tab is selected. A blue circle labeled '1' highlights the 'Peripherals' section, which contains three items: ADC1, PWM, and TMR1. The 'ADC1' item is highlighted with a red box. A large blue arrow points from this interface to the right, leading to the 'ADC1' configuration screen.

ADC1

Easy Setup **Registers**

Hardware Settings

Enable ADC

ADC Clock

Conversion Clock Source: FOSC/2

Conversion Time: 370 ns

Target Shared Core Sampling Time: 40 ns

Calculated Shared Core Sampling Time: 40 ns

Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Shared	<input type="checkbox"/>	AN10	RB8		None	No...	
Shared	<input type="checkbox"/>	AN11	RB9		None	No...	
Shared	<input checked="" type="checkbox"/>	AN12	RC0	channel_AN12	PWM1 Trigger1	No...	
Shared	<input checked="" type="checkbox"/>	AN13	RC1	channel_AN13	PWM1 Trigger2	No...	

PWM Configuration in MCC (1/3)

1

Projects Files Classes Resource Management [MCC]

Tree View Flat View

Project Resources Generate Import... Export

Peripherals

- ADC1
- PWM**
- TMR1

System

- Interrupt Module
- Pin Module
- System Module

PWM

Easy Setup Registers

Hardware Settings

PWM Clock Settings

Master Clock Selection AFPLL0 - Auxiliary Clock with PLL Enabled Master Clock Frequency 500 MHz

Clock Divided Frequency 250000000 Hz

Select Required PWM Generators PWM Generator 1

PWM Master Settings

PWM Generator PWM Generator 1 Custom Name PWM_GENERATOR_1

Enable PWM Generator
 Enable High Resolution

PWM Operation Mode Independent Edge

PWM Output Mode Complementary

PWM Configuration in MCC (2/3)

1

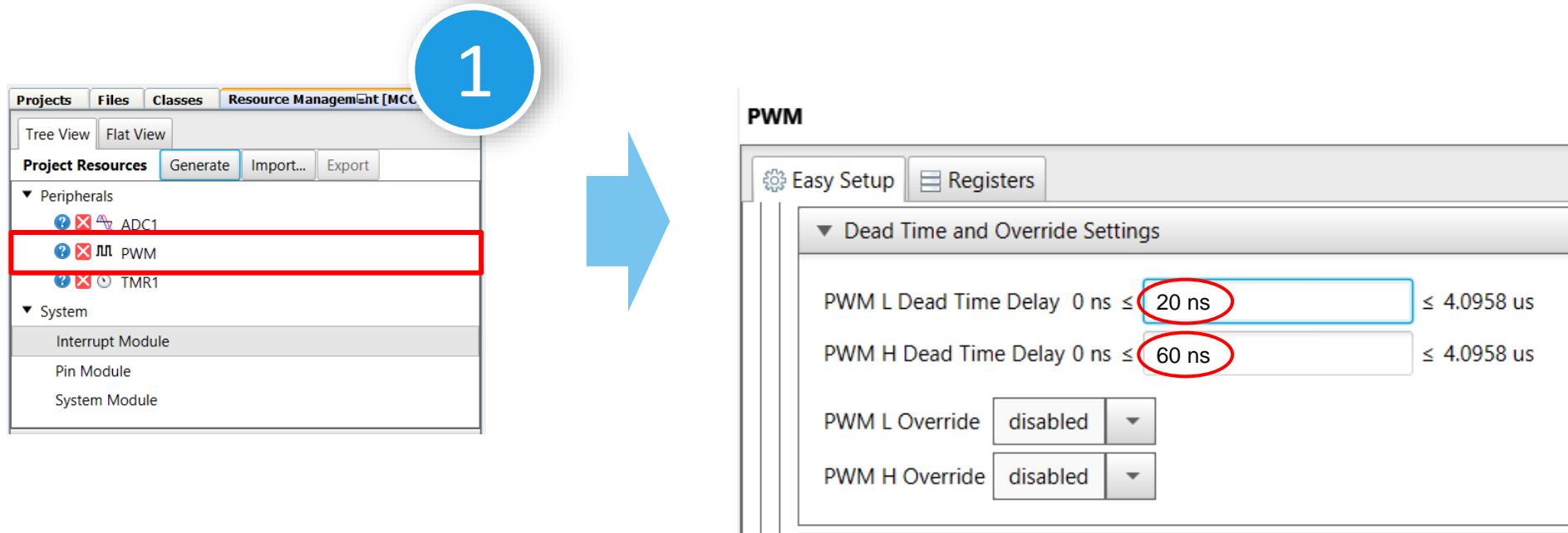
The diagram illustrates the process of configuring a PWM module. It starts with the 'Resource Management [MCC]' interface on the left, where the 'Project Resources' tab is selected. A blue circle labeled '1' highlights the 'Peripherals' section, specifically the 'PWM' resource, which is highlighted with a red box. A large blue arrow points from this interface to the right. On the right is the 'PWM' configuration dialog. This dialog has two tabs: 'Easy Setup' (selected) and 'Registers'. Under 'Easy Setup', there are sections for 'PWM Frequency Settings', 'Duty Cycle', and 'Phase'. In the 'Period' section of 'Frequency Settings', the 'Requested Frequency' is set to 61.02864 kHz, which is circled in red. Below it, the 'Calculated Frequency' is shown as 500 kHz. In the 'Duty Cycle' section, the 'PWM Duty Cycle' is set to 0%. In the 'Phase' section, the 'PWM Phase' is set to 0 ns.

PWM Configuration in MCC (3/3)

1

The diagram illustrates the process of configuring PWM in the Microchip Configuration (MCC) tool. It starts with the 'Resource Management' interface on the left, where the 'Project Resources' tab is selected. A blue circle with the number '1' is placed over the 'Project Resources' tab. Below it, the 'Peripherals' section is expanded, showing three resources: ADC1, PWM, and TMR1. The 'PWM' resource is highlighted with a red rectangle and a large blue arrow points to the right, indicating the transition to the next step. The second part of the diagram shows the 'PWM' configuration interface. At the top, there are two tabs: 'Easy Setup' (selected) and 'Registers'. The 'Easy Setup' tab displays several configuration sections. One section, 'PWM Start of Cycle Control', includes fields for 'Start of Cycle Trigger' (set to 'Self-trigger') and 'Trigger Output Selection' (set to 'EOC event'). Another section, 'ADC Trigger', contains two dropdown menus: 'ADC Trigger 1' (set to 'Trigger A Compare') and 'ADC Trigger 2' (set to 'Trigger B Compare'). Both of these dropdowns are circled in red. Below these are three numerical input fields: 'Trigger A Compare 0 ns' (set to '1000 ns'), 'Trigger B Compare 0 ns' (set to '1400 ns'), and 'Trigger C Compare 0 ns' (set to '0 ns'). Each of these three fields is also circled in red. Further down, there are sections for 'Dead Time and Override Settings' and 'Data Update Settings'. In the 'Data Update Settings' section, the 'Update Trigger' dropdown is set to 'Duty Cycle' and is circled in red. The 'Update Mode' dropdown is set to 'SOC update'.

PWM Configuration in MCC (3/3)



IO Configuration in MCC

The diagram illustrates the process of configuring IO pins in the Microchip Configuration Center (MCC). It consists of three main sections connected by arrows:

- Project Resources:** On the left, under the "Resource Management [MCC]" tab, the "Pin Module" item is highlighted with a red box. A large blue circle with the number "1" is positioned above it.
- Pin Module:** A table titled "Pin Module" shows pin assignments. The "Custom Name" column for pin RB6 is circled in red and contains the value "USER_LED". This table is also highlighted with a red box.
- Pin Manager:** On the right, a grid-based interface shows the physical pin layout for a QFN48 package. The "Pin Manager: Grid View" tab is active. The "Port B" row, which includes pin RB6, is highlighted with a red box. The "Pin Module" section of the grid, which includes the "Module", "Function", and "Direction" columns, is also highlighted with a red box. A green lock symbol is visible in the "LOCK" column for pin RB6.

To make device pin RB6 an output, click on LOCK symbol
representing device pin function

Interrupt Configuration in MCC

1

The diagram illustrates the process of configuring interrupts in the Microchip Configuration Center (MCC). It shows two main windows: the 'Resource Management [MCC]' interface and the 'Easy Setup' window.

Resource Management [MCC] Interface:

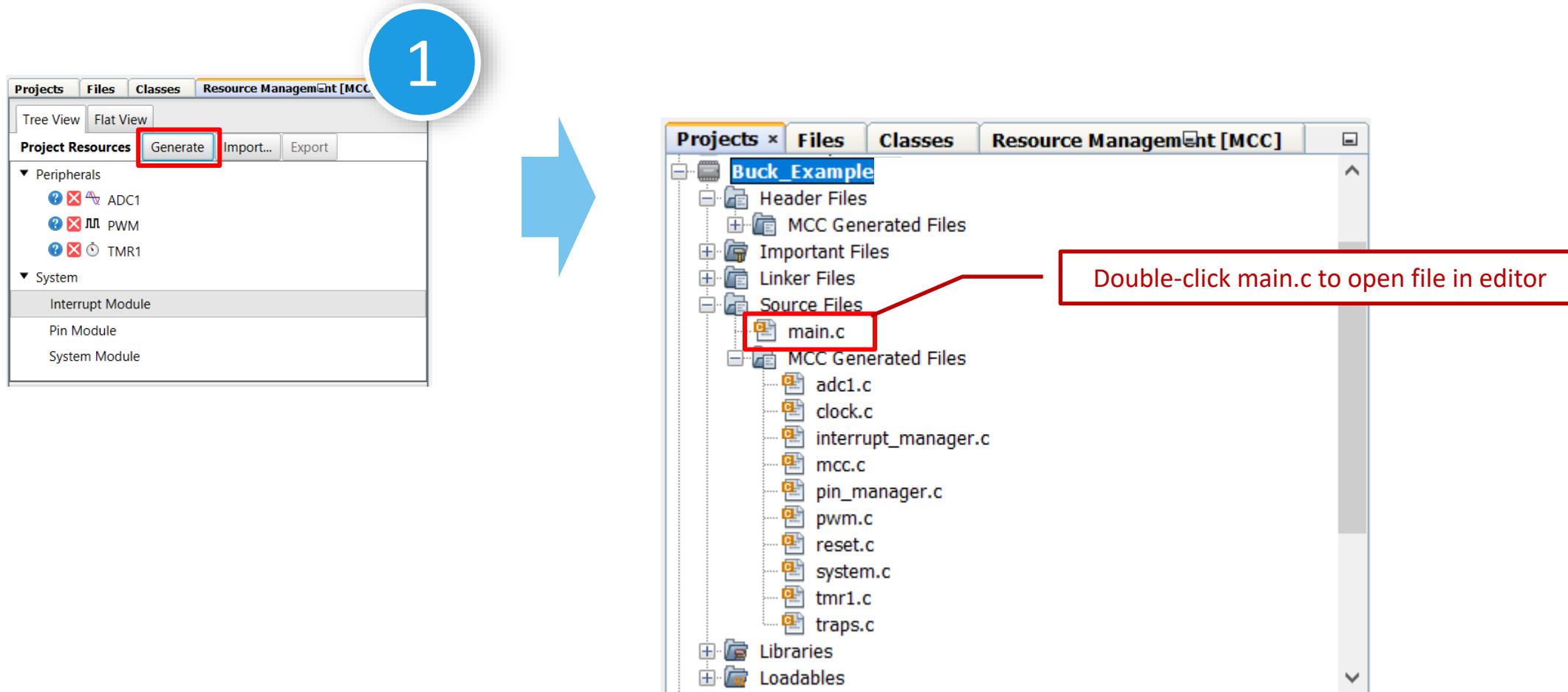
- Project Resources tab is selected.
- Tree View and Flat View buttons are available.
- Peripherals section lists ADC1, PWM, and TMR1.
- System section lists Interrupt Module, Pin Module, and System Module. The 'Interrupt Module' item is highlighted with a red border.

Easy Setup - Interrupt Manager:

- Enable Global Interrupts checkbox is checked.
- Table displays interrupt configurations:

Module	Interrupt	Description	IRQ Nu...	Enabled	Priority	Context
Pin Module	CNAI	Change Notification A	2	<input type="checkbox"/>	1	OFF
Pin Module	CNBI	Change Notification B	3	<input type="checkbox"/>	1	OFF
Pin Module	CNCI	Change Notification C	19	<input type="checkbox"/>	1	OFF
Pin Module	CNDI	Change Notification D	75	<input type="checkbox"/>	1	OFF
TMR1	TI	Timer 1	1	<input checked="" type="checkbox"/>	1	OFF
ADC1	ADCAN11	ADC AN11 Convert Done	102	<input type="checkbox"/>	1	OFF
ADC1	ADCAN10	ADC AN10 Convert Done	101	<input type="checkbox"/>	1	OFF
ADC1	ADCAN13	ADC AN13 Convert Done	104	<input checked="" type="checkbox"/>	6	OFF
ADC1	ADCAN12	ADC AN12 Convert Done	103	<input type="checkbox"/>	1	OFF

Generating the Project



Setting Up Interrupts

File main.c

```
void TMR1_Int(void)
{
    static int led_tmr = 0;

    // blink LED 1Hz
    if(++led_tmr>=500) {
        led_tmr = 0;
        USER_LED_Toggle();
    }
}
```

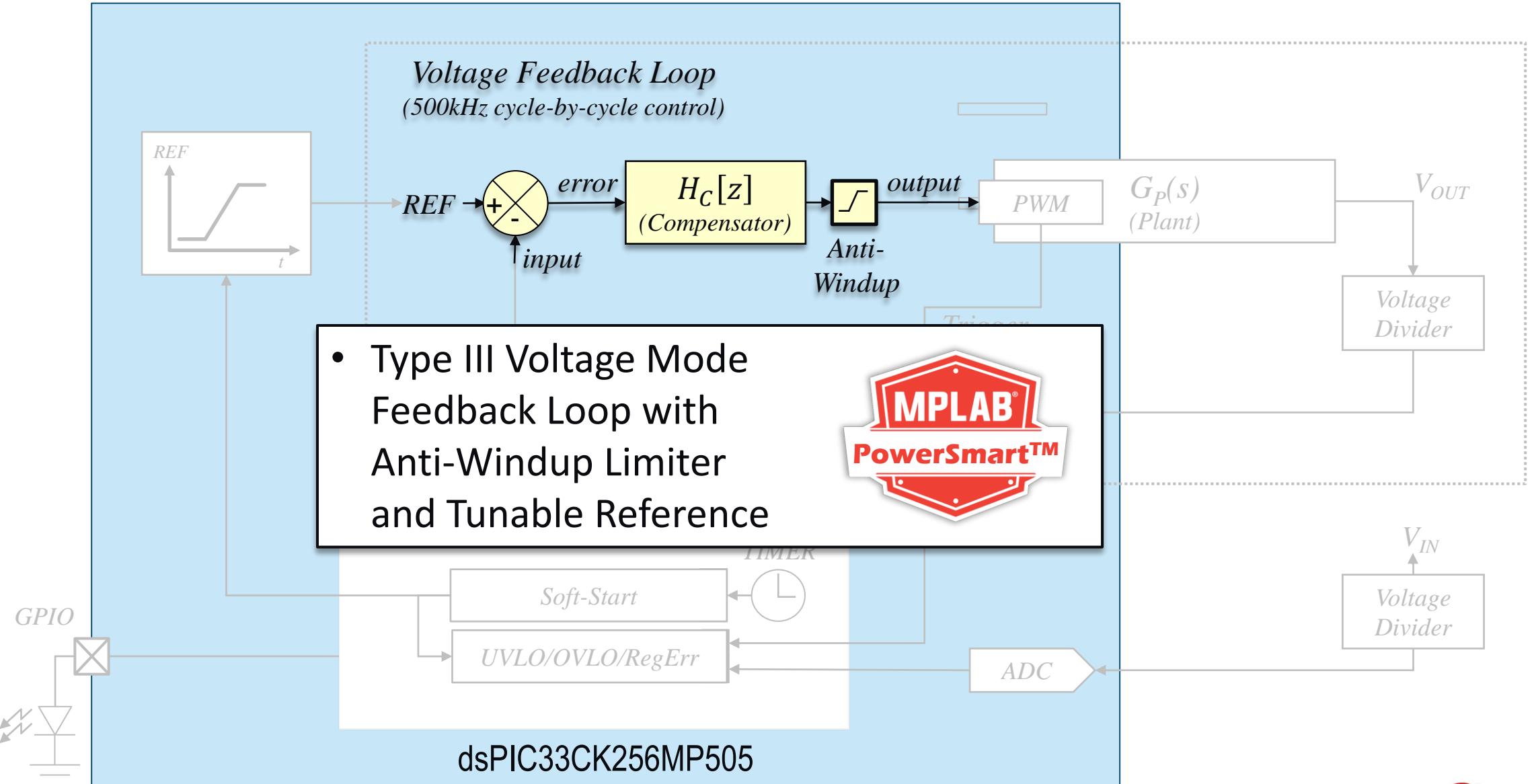
```
void AN13_Int(void)
{
    adc_vin = ADCBUF12;
    adc_vbuck = ADCBUF13;
}
```

Recommended Intermediary Step:

If you program this code into the target device,
The on-board LED will start blinking at a
frequency of 1 Hz

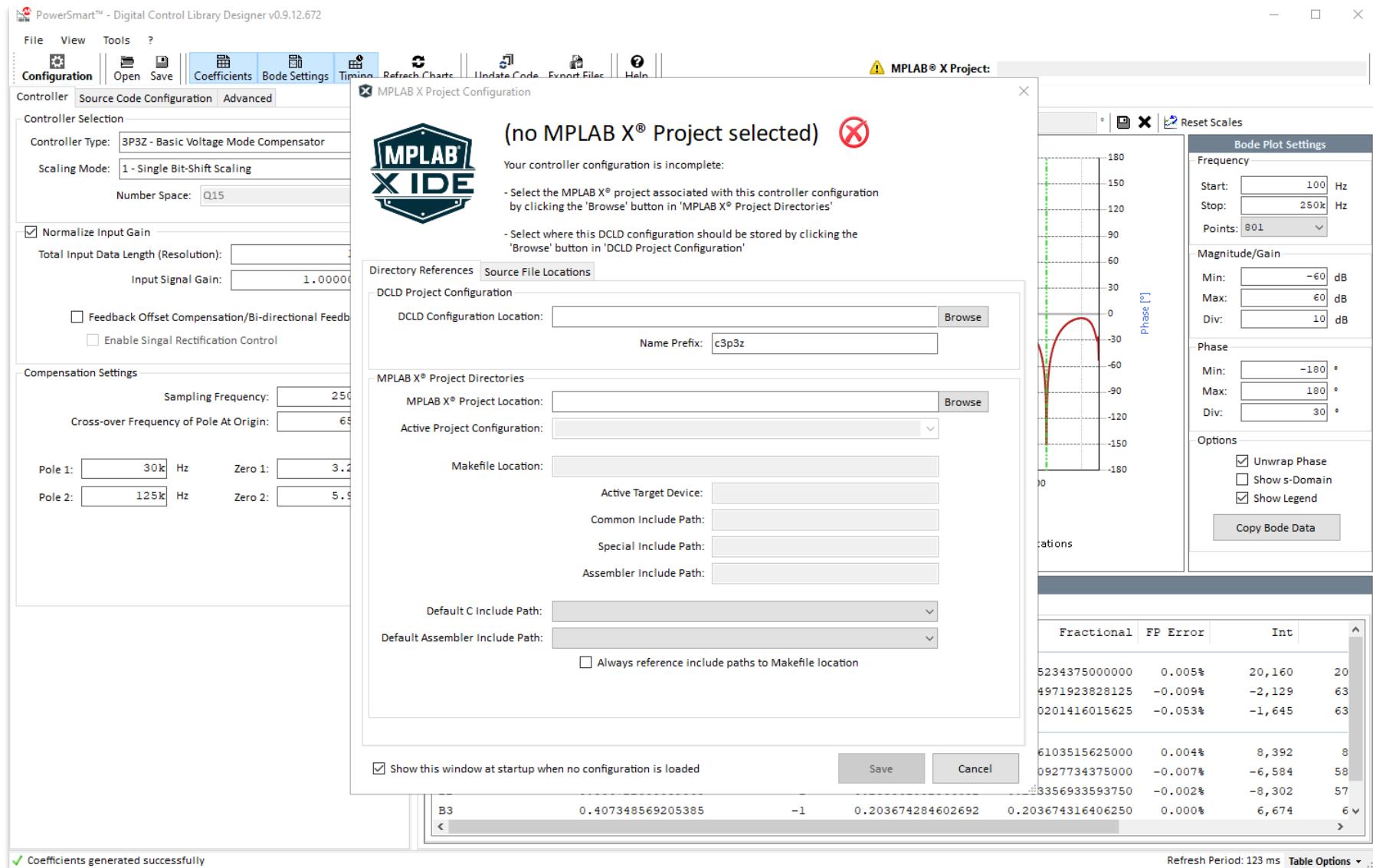
```
/**  
 * Section: Included Files  
 */  
#include "mcc_generated_files/system.h"  
#include "mcc_generated_files/mcc.h"  
  
// Define Global Variables:  
int adc_vin = 0;      // Scaling: 155.1 steps/V  
int adc_vbuck = 0;    // Scaling: 620.5 steps/V  
int ref_vbuck = 0;    // Scaling: 620.5 steps/V  
[...]  
  
add red colored code to file main.c  
  
int main(void)  
{  
    // initialize the device  
    SYSTEM_Initialize();  
    ADC1_Setchannel_AN13InterruptHandler(&AN13_Int);  
    TMR1_SetInterruptHandler(&TMR1_Int);  
    while (1)  
    {  
        // Add your application code  
    }  
    return 1;  
}
```

Requirements

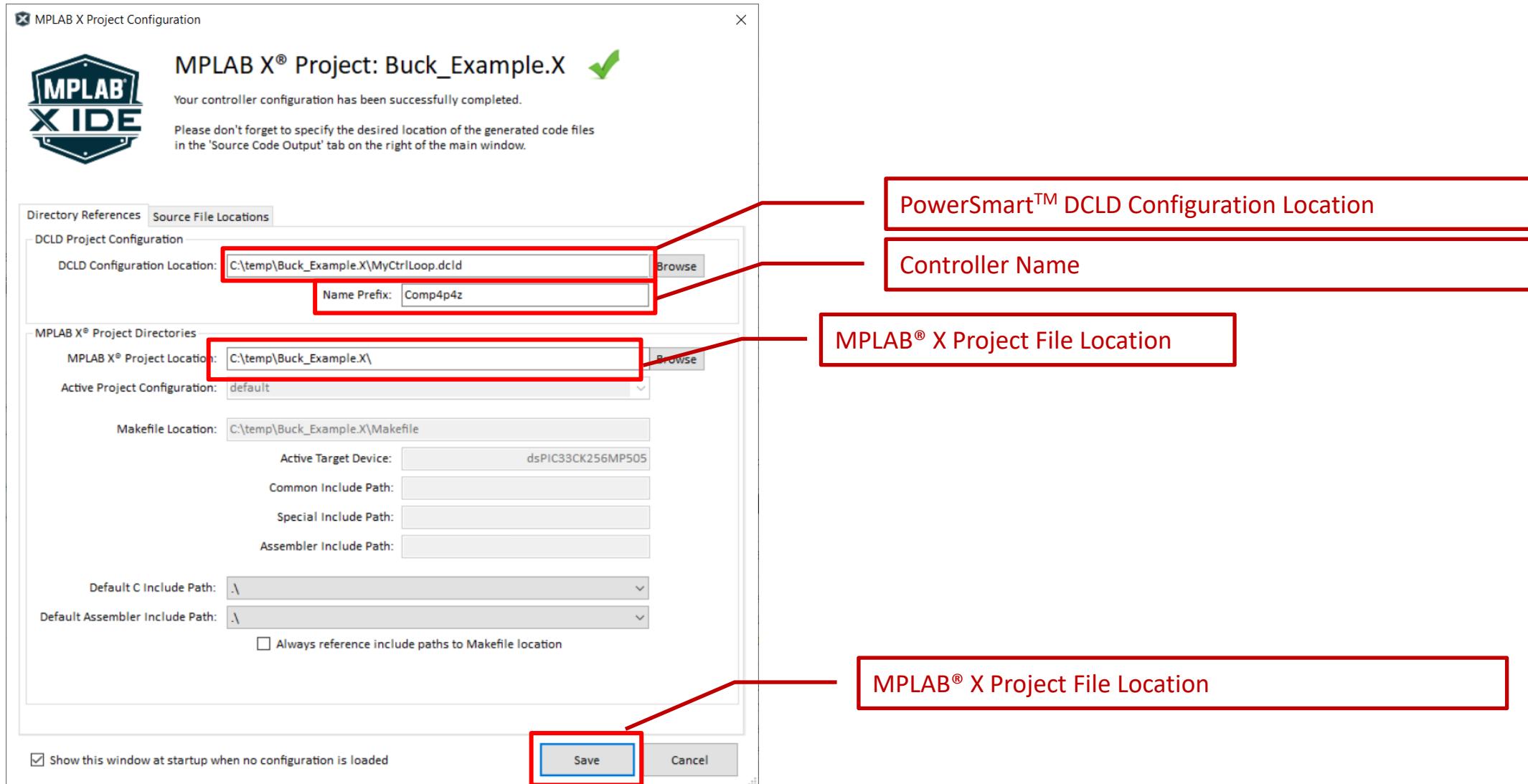


MPLAB® PowerSmart™ Digital Control Library

Designer



Initial Project Configuration



Compensation Filter Selection and Adjustment

Controller Source Code Configuration Advanced

Controller Selection

Controller Type: 4P4Z - Advanced High-Q Compensator
 Scaling Mode: 4 - Fast Floating Point Coefficient Scaling
 Number Space: Q15

Normalize Input Gain
 Total Input Data Length (Resolution): 12 Bit
 Input Signal Gain: 0.500000

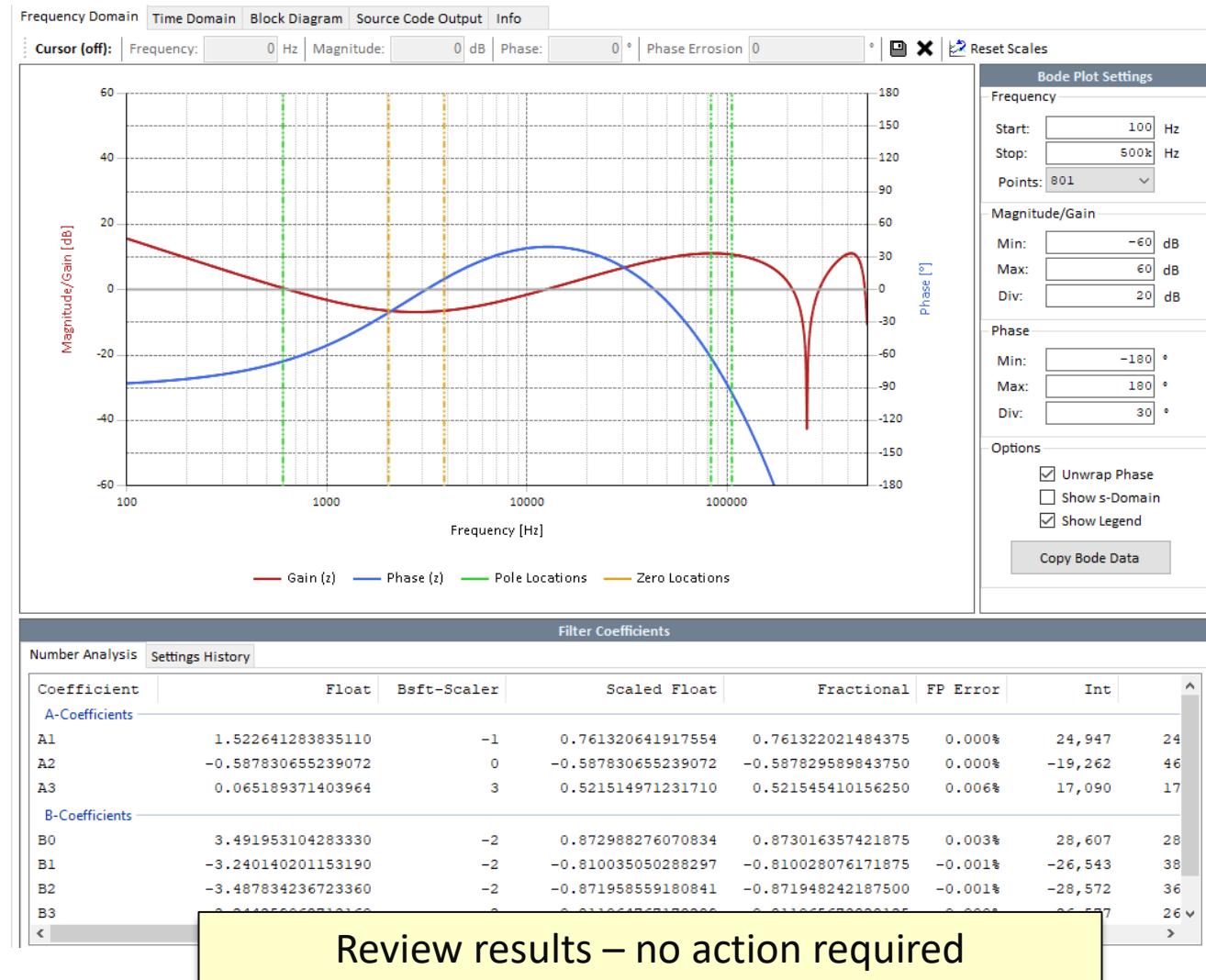
Feedback Offset Compensation/Bi-directional Feedback
 Enable Singal Rectification Control

Compensation Settings

Sampling Frequency: 500k Hz
 Cross-over Frequency of Pole At Origin: 600 Hz

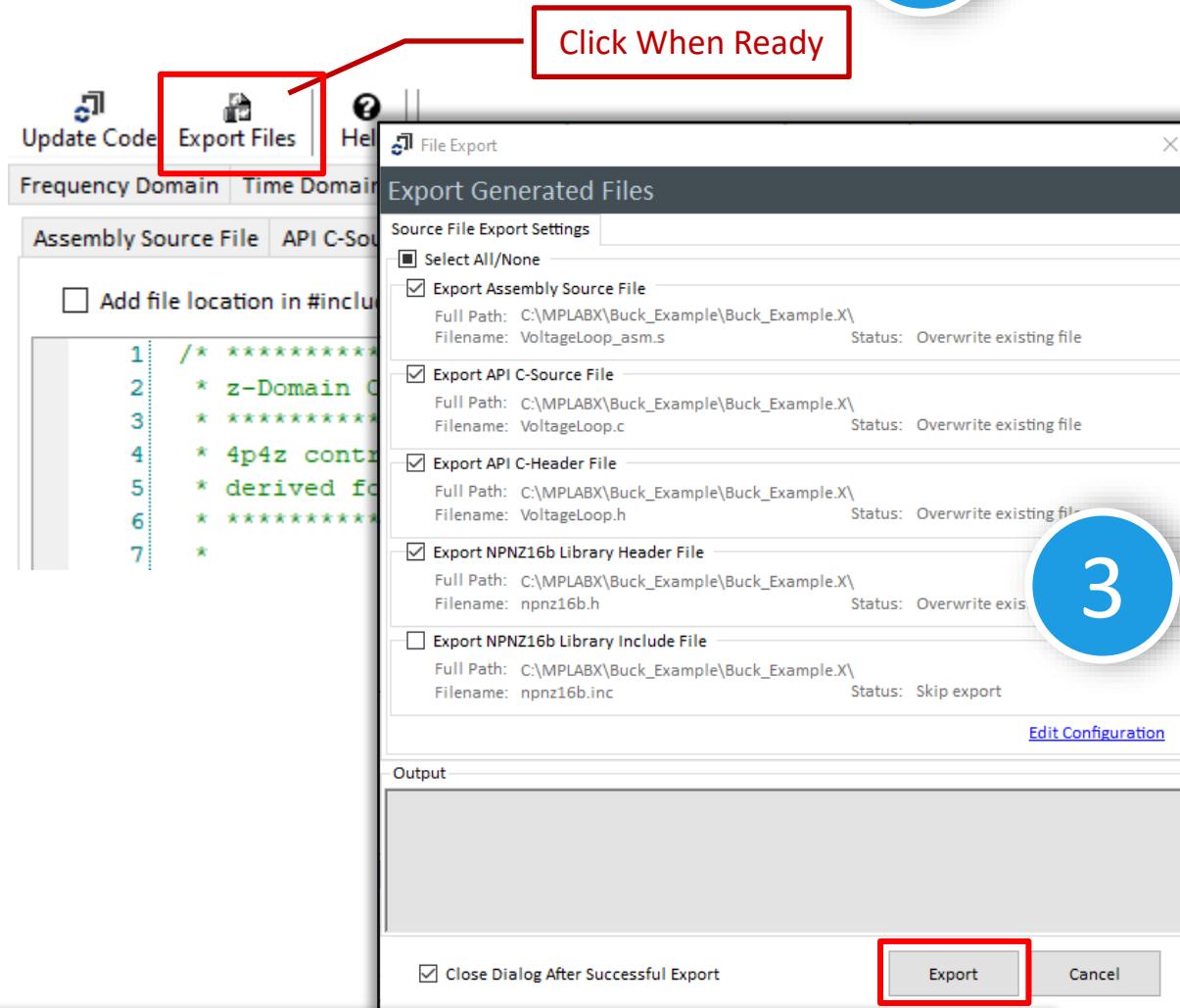
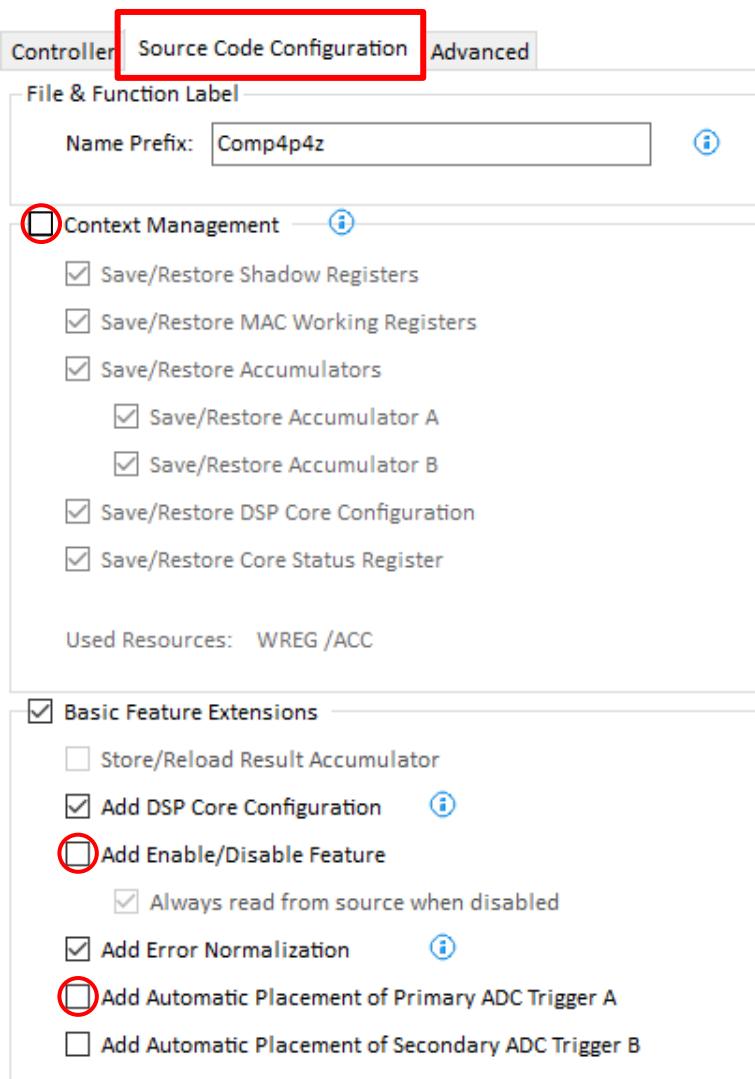
Pole 1:	50k Hz	Zero 1:	2k Hz
Pole 2:	100k Hz	Zero 2:	4k Hz
Pole 3:	180k Hz	Zero 3:	18k Hz

1



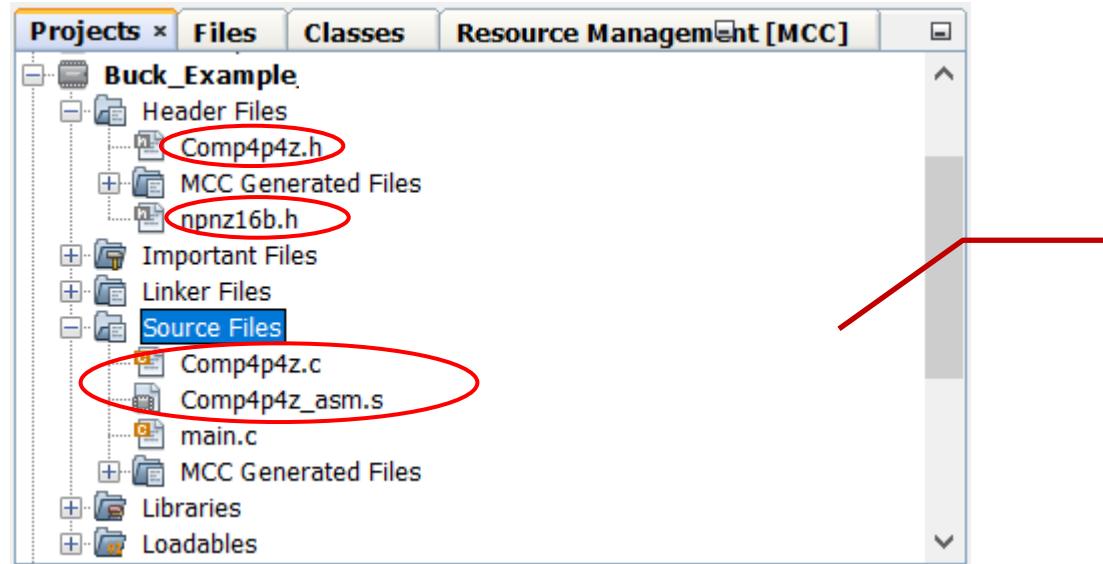
Configuring Software-Options

2



MPLAB® PowerSmart™ Digital Control Library Designer
is generating files in project directory...

Add PS-DCLD Generated Files To Project



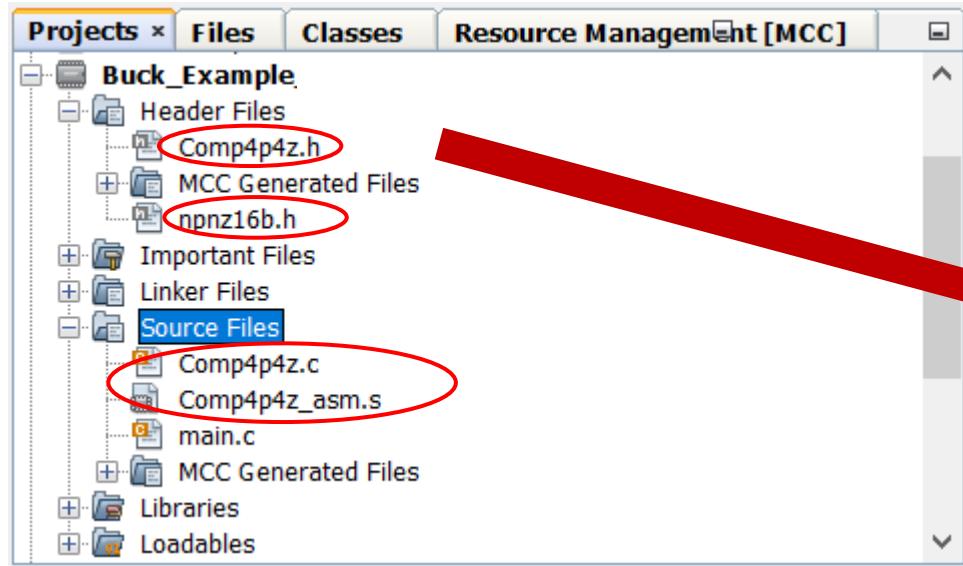
Add generated files to Project by Right-Click on *Header Files* → *Add Existing File*:

- *Source Files:*
 - Assembly Library File (<my_controller>.asm.s)
 - C-Source File (<my_controller>.c)
- *Header Files:*
 - C-Header File (<my_controller>.h)
 - Library C-Header File (npnz16b.h)

Recommended:

Also add the PS-DCLD Configuration file (*.dcl) to folder *Important Files* in the *Project Manager*. Thus, the current configuration can be conveniently opened from MPLAB® X by right-click on the configuration file, selecting *Open In System*

Including DCLD Generated Files



```
/**  
 * Section: Included Files  
 */  
#include "mcc_generated_files/system.h"  
#include "mcc_generated_files/pin_manager.h"  
#include "mcc_generated_files/pwm.h"  
#include "mcc_generated_files/adc1.h"  
#include "mcc_generated_files/tmr1.h"  
#include "Comp4p4z.h"
```

```
void AN13_Int(void)  
{  
    adc_vin = ADCBUF12;  
    adc_vbuck = ADCBUF13;  
    Comp4p4z_Update(&Comp4p4z);  
}
```

*Add call of newly added controller function to ADC
Interrupt Service Routine*

Adding DCLD Controller Initialization Routine

File main.c

```
void CONTROL_Initialize(void)
{
    // initialize compensator
    Comp4p4z_Initialize(&Comp4p4z);
    Comp4p4z.Ports.Source.ptrAddress = (unsigned*)&adc_vbuck;
    Comp4p4z.Ports.Target.ptrAddress = (unsigned*)&PG1DC;
    Comp4p4z.Ports.ptrControlReference = (unsigned*)&ref_vbuck;
    Comp4p4z.Limits.MinOutput = 0;
    Comp4p4z.Limits.MaxOutput = (int)(0.8*PG1PER); // 80% DC
}
```

*Add call of newly added controller initialization
function to main(void)*

```
/**  
 * Section: Included Files  
 */
```

[...]

Add code before main(void)

```
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    CONTROL_Initialize();

    ADC1_Setchannel_AN13InterruptHandler(&AN13_Int);
    TMR1_SetInterruptHandler(&TMR1_Int);

    while (1)
    {
        // Add your application code
    }
    return 1;
}
```

You are all set to program
the device now...

May the Power be with you