



PSFB Converter Code Examples

dsPIC33CK64MP102



A Leading Provider of Smart, Connected and Secure Embedded Solutions



Microchip
ESE / Edward Lee
March-20, 2024

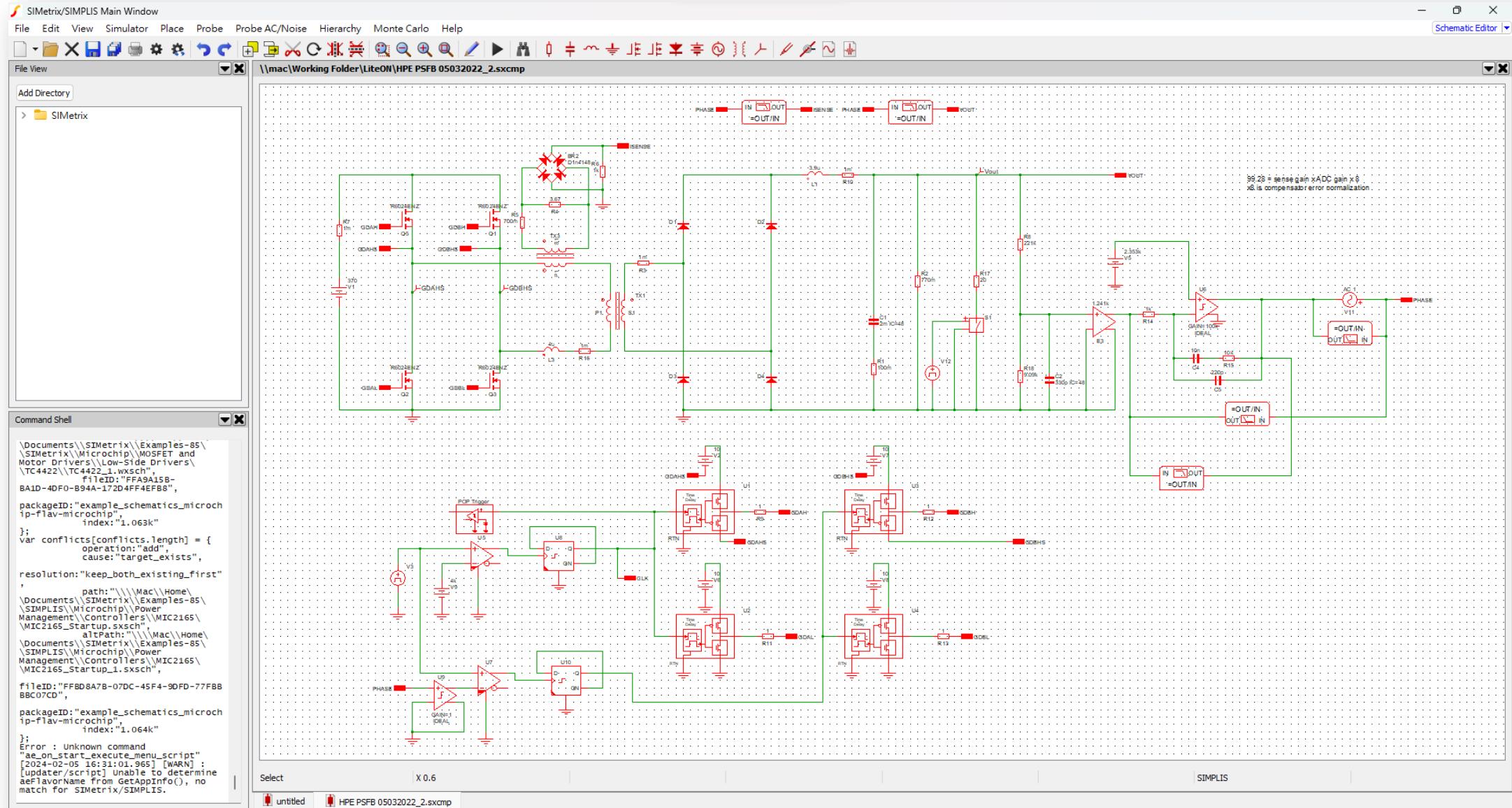
Agenda

- Simulation
- Test Environment
- Code Examples
 - Codebase with OS Scheduler
 - https://github.com/EdwardLeeTW/PSFB_CodeBase_OS_Scheduler_MP102
 - VMC PSFB with SR
 - https://github.com/EdwardLeeTW/PSFB_VMC_MP102
 - PCMC PSFB with SR

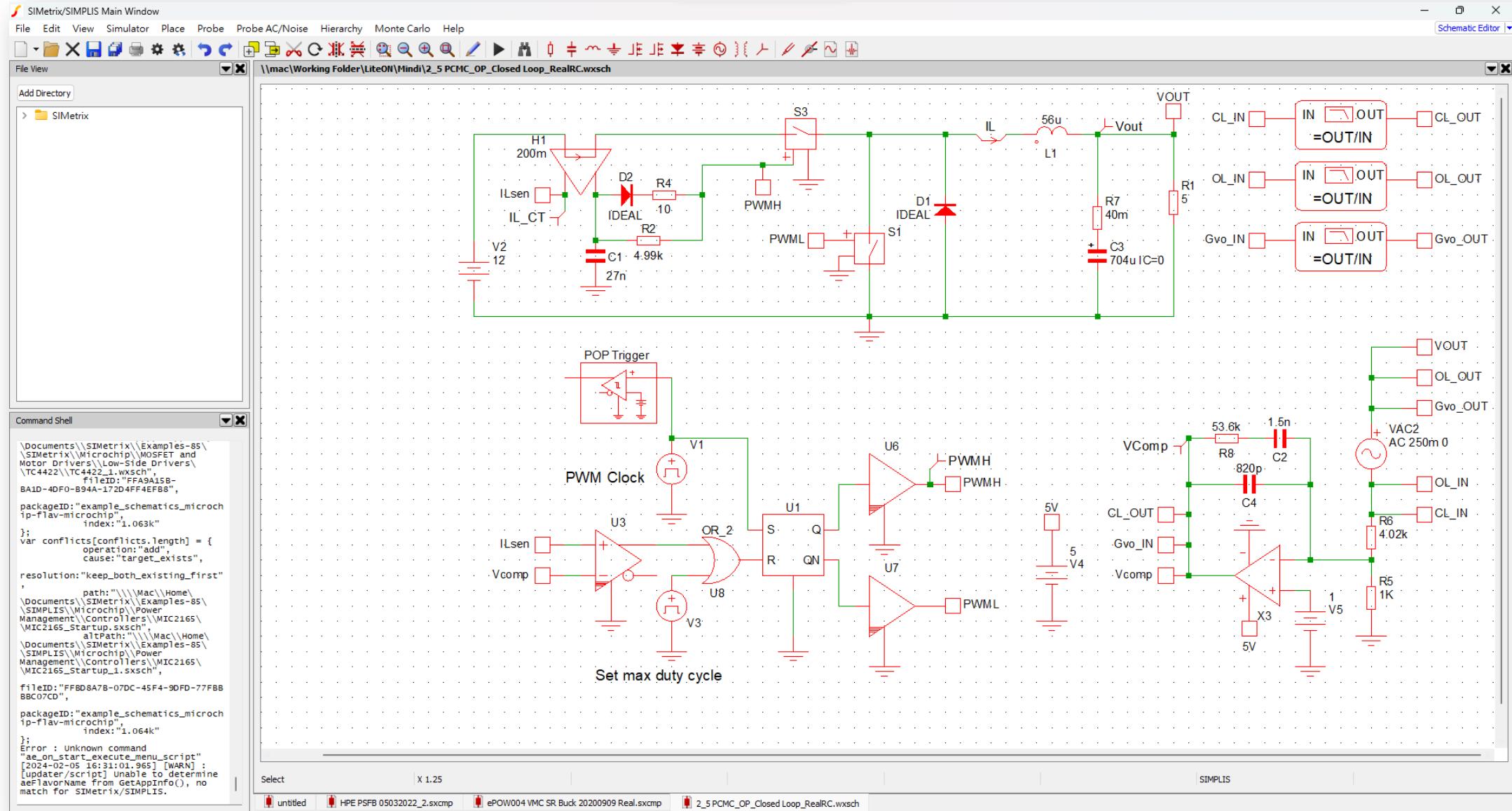


Simulation

SIMPLIS



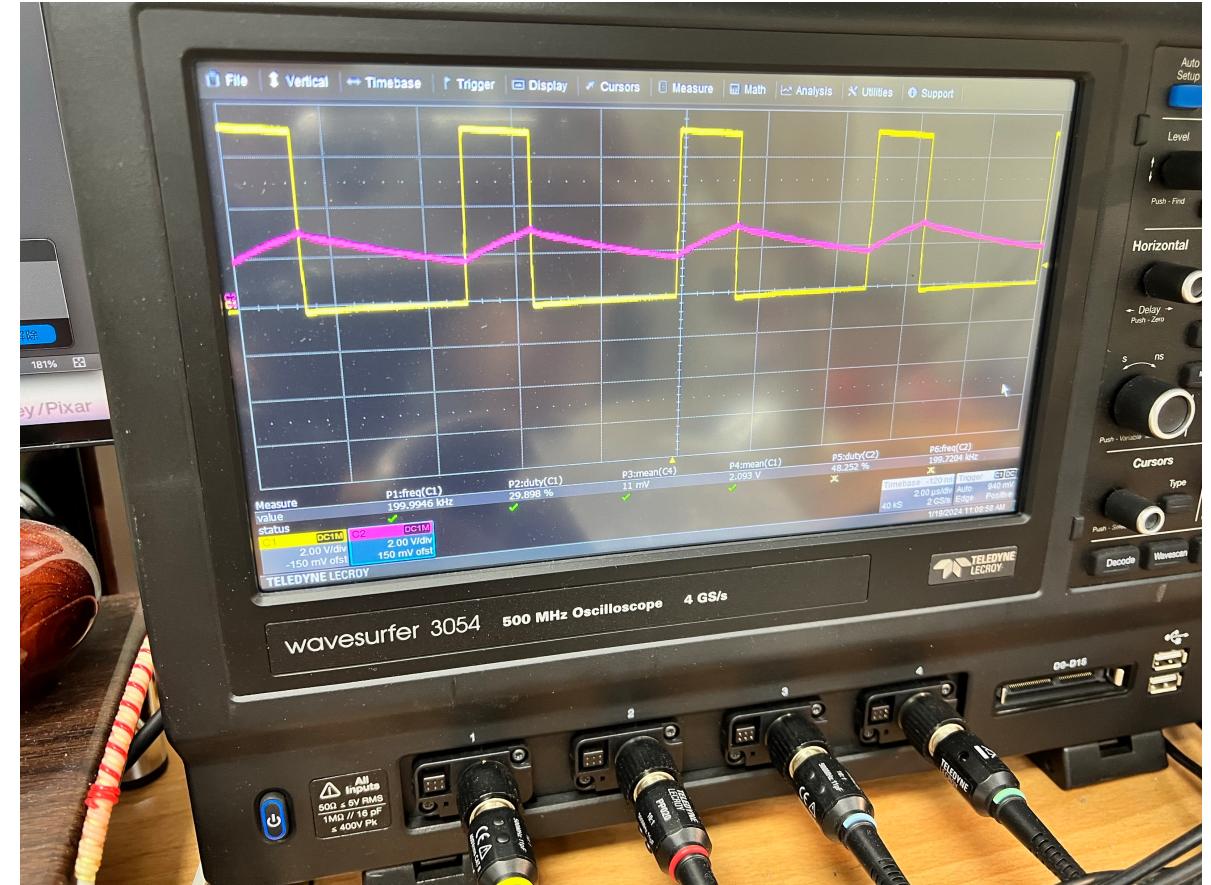
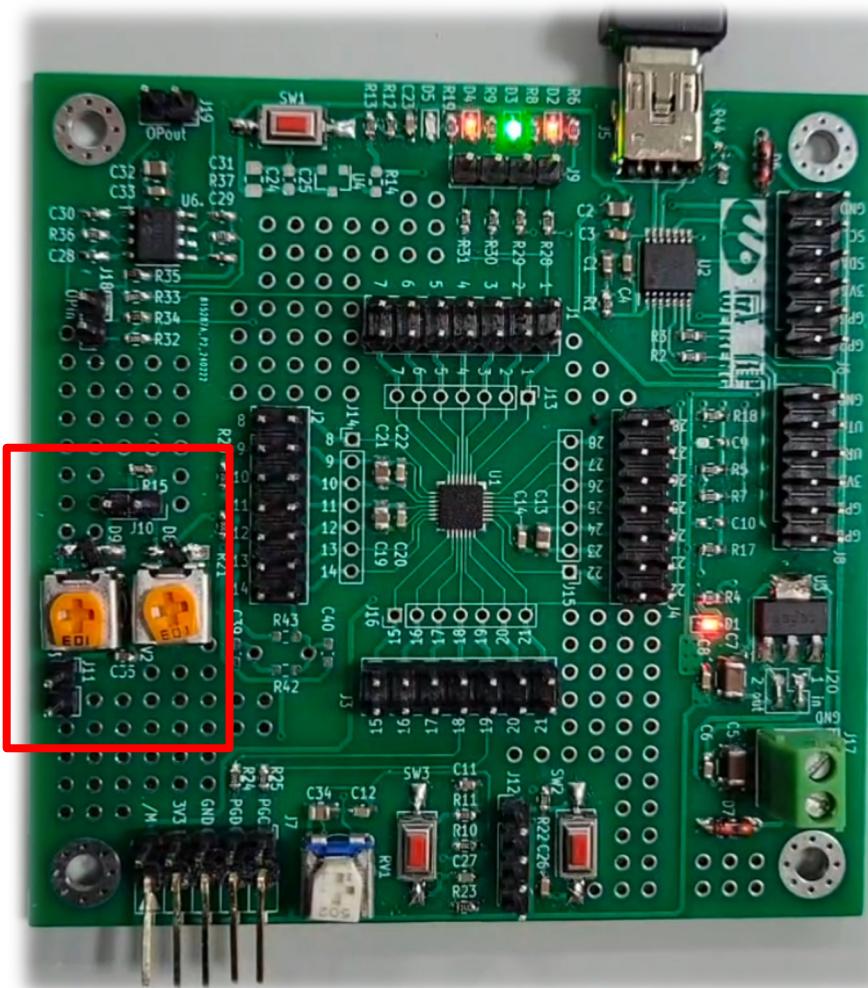
SIMPLIS



Test Environment

MP102 Debugging Board by Weikeng

- Peak Current Signal Simulation



Basic F/W Framework with OS Scheduler

Peripheral & Tool Descriptions

- Peripherals (in RED)
 - Timer1 100us
 - IP = 3 & NO Context
 - LED1 toggling in Tasks_1s
 - I/O
 - LED1 – RB4
- Chip: dsPIC33CK64MP102
- S/W Versions
 - MPLAB X IDE v6.20
 - DFP v1.13.366
 - Compiler XC16 v2.1
 - MCC v5.5.0
 - Core v5.7.0
 - MCU Lib v1.171.4

MCC Drivers – Clock@100 MIPS

Project Resources Generate Import... Export ?

System Interrupt Module Pin Module System Module Content Manager Pin Manager: Package View

8000000 Hz FRC Oscillator (8.0 MHz) Clock Source

FRC Postscaler

PLL Enable

Prescaler: 1:1, Feedback: 1:200, Postscaler1: 1:2, Postscaler2: 1:2

200 MHz Fosc
100 MHz Fosc/2 **100MIPS**

Auxiliary Clock

PLL Enable

Prescaler: 1:1, Feedback: 1:125, Postscaler1: 1:2, Postscaler2: 1:1

VCO & AVCO

VCO Divider: FVCO/4, AVCO Divider: FVCO/2

Clock Output Pin Configuration: OSC2 is general purpose digital I/O pin

Reference Oscillator Output

CAN FD Clock Generator

RB9|PGC1 VSS4A RB13 RB12 RB11 RB10 VDD4A

RB14 1 RB15 2 NMCLR 3 RA0 4 RA1 5 RA2 6 RA3 7

28 27 26 25 24 23 22 21 RB8|PGD1

20 RB7

19 RB6

18 RB5

17 RB4|LED1|GPIO

16 RB3

15 RB2

8 9 10 11 12 13 14

MICROCHIP
dsPIC33CK64MP102

Core Versions [MCC] Navigator

Device Resources Content Manager

Documents dsPIC33CK256MP508 Product Page

Libraries 16-bit Bootloader CHARACTER LCD CryptoAuthLibrary DacLibrary FatFs Foundation Services LED LED_BLUE LED_GREEN LED_RED MCP802X motorBench® Development Suite Pac193xLibrary POTENTIOMETER RGB LED

FIGURE 9-3: PLL AND VCO DETAIL

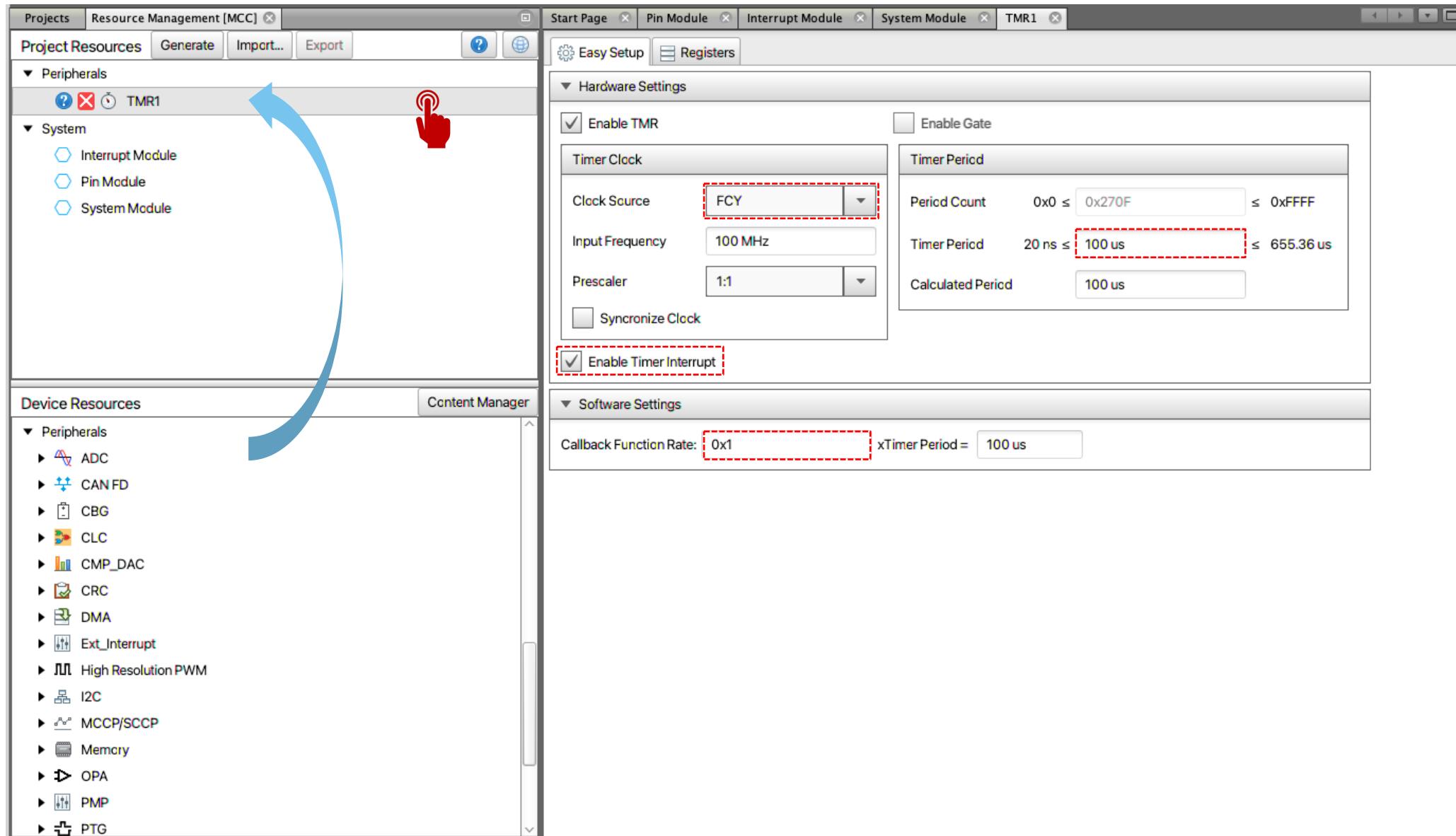
Note 1: Clock option for PWM.
 2: Clock option for ADC.
 3: Clock option for DAC.
 4: PLL source is always FRC unless FNOSC is the Primary Oscillator with PLL.

Emulator Pin Placement Communicate on PGC1 and PGD1

PC: 0x0 oab sab da dc n ov z c How do I? Keyword(s) Search (%+!)

Notifications Output Pin Manager: Grid View Notifications [MCC]

MCC Drivers – Timer1 (100 us)



MCC Drivers – Pin Management

MCC Drivers – Interrupt Management & Code Generating

Screenshot of MPLAB X IDE v6.20 showing the Pin Manager and Interrupt Manager for the dsPIC33CK32MP102 device.

Interrupt Manager:

| Module | Interrupt | Description | IRQ Number | 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 |
| DMT | DMTI | Dead Man Timer | 45 | <input type="checkbox"/> | 1 | OFF |
| TMR1 | TI | Timer 1 | 1 | <input checked="" type="checkbox"/> | 3 | OFF |

Pin Manager: Package View:

The diagram shows the pinout for the dsPIC33CK32MP102 device, which is a QFN28 package. The pins are numbered 1 through 28. Functions include:

- Pins 1-7: RA0-RA3, AVDDA, VSS2A, VDD2A, RBO, RB1, RB2
- Pins 8-14: RA4-RB13, RB8|PGD1, RB7, RB6, RB5, RB4|LED1|GPIO
- Pins 15-21: NMCLR, RA1, RA2, RA3, RB12, RB11, RB10, RB9, RB8|PGC1, RB7, RB6, RB5
- Pins 22-28: RB13, RB12, RB11, RB10, RB9, RB8|PGC1, RB7, RB6, RB5, RB4|LED1|GPIO, RB3, RB2, RB1, VSS2A, VDD2A, RBO, RB1

Pin Manager: Grid View:

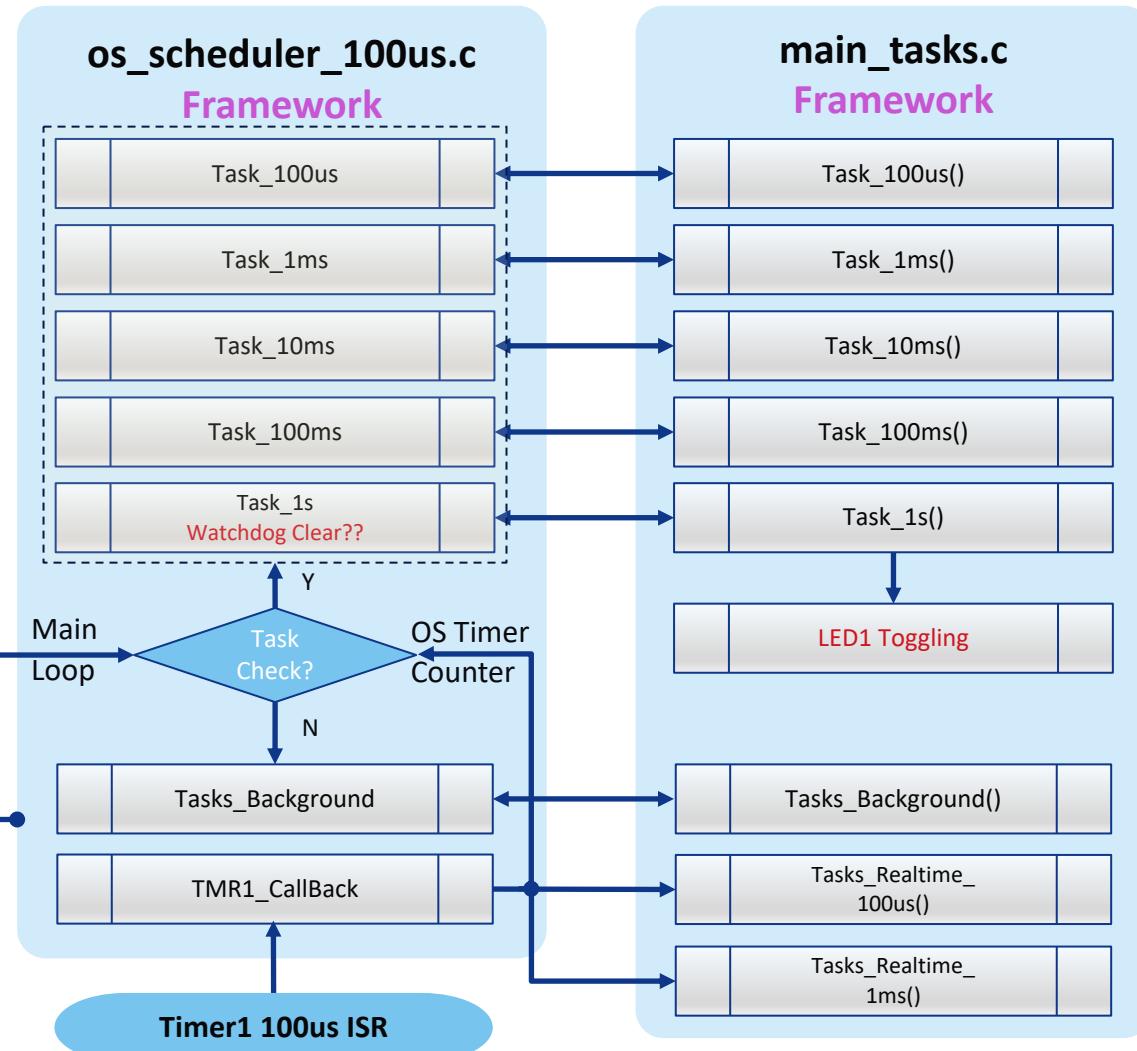
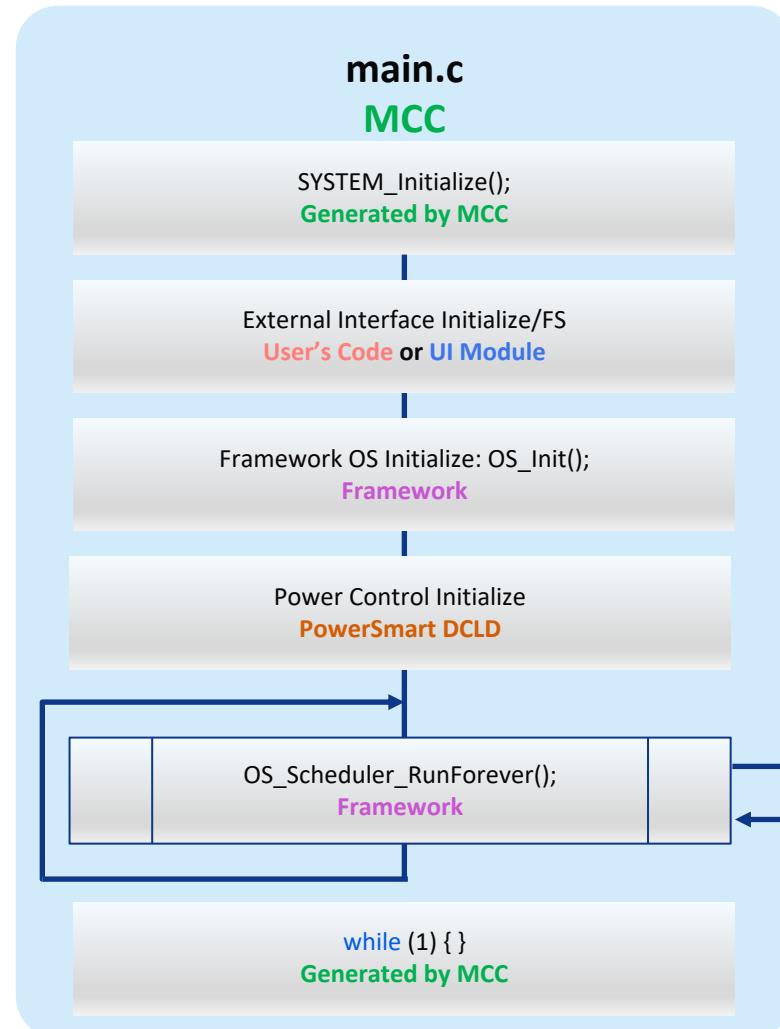
| Package: QFN28 | Pin No: | 4 | 5 | 6 | 7 | 8 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 25 | 26 | 27 | 28 | 1 | 2 | | | |
|----------------|----------|-----------|---|---|---|---|----|----|----|----|----|----|----|----------|----|----|----|----|----|----|----|----|----|--|--|
| | | Port A ▼ | | | | | | | | | | | | Port B ▼ | | | | | | | | | | | |
| Module | Function | Direction | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | | |
| Clock ▾ | CLKI | input | | | | | | | | | | | | | | | | | | | | | | | |
| | CLKO | output | | | | | | | | | | | | | | | | | | | | | | | |
| | OSCI | input | | | | | | | | | | | | | | | | | | | | | | | |
| | OSCO | output | | | | | | | | | | | | | | | | | | | | | | | |
| | REFI | input | | | | | | | | | | | | | | | | | | | | | | | |
| | REFO | output | | | | | | | | | | | | | | | | | | | | | | | |
| ICD ▾ | PGCx | input | | | | | | | | | | | | | | | | | | | | | | | |
| | PGDx | input | | | | | | | | | | | | | | | | | | | | | | | |
| Pin Module ▾ | GPIO | input | | | | | | | | | | | | | | | | | | | | | | | |
| | GPIO | output | | | | | | | | | | | | | | | | | | | | | | | |
| TMR1 | T1CK | input | | | | | | | | | | | | | | | | | | | | | | | |

MP102_Codebase - Dashboard:

- Project Type: Application – Configuration: default
- Device: dsPIC33CK32MP102
 - Checksum: Blank, no code loaded
 - CRC32: Hex file unavailable
- Packs: dsPIC33CK-MP_DFP (1.13.366), PICKit 4 (2.3.1848)
- Compiler Toolchain: XC16 (v2.10) [/Applications/microchip/xc16/v2.1], Production Image: Optimization: gcc 0
- Memory: Usage Symbols disabled. Click to enable Load Sym

SMPS Firmware Framework

Firmware Structure

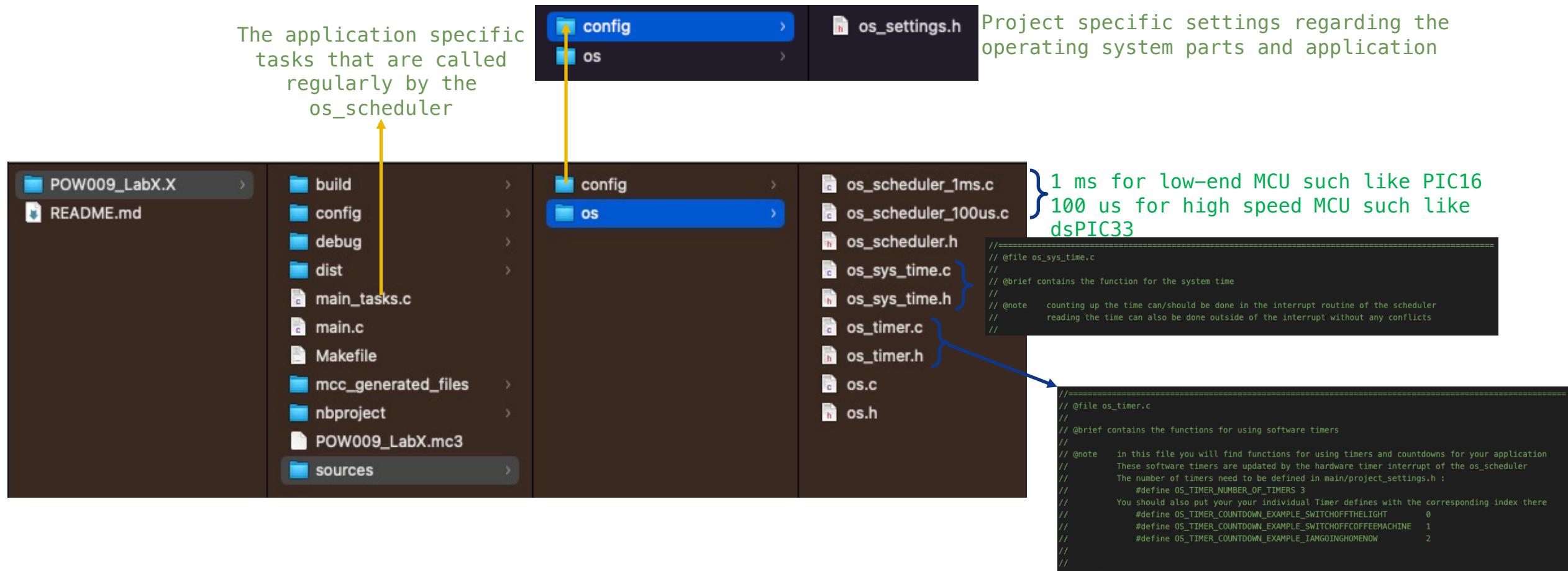


SMPS Firmware Framework

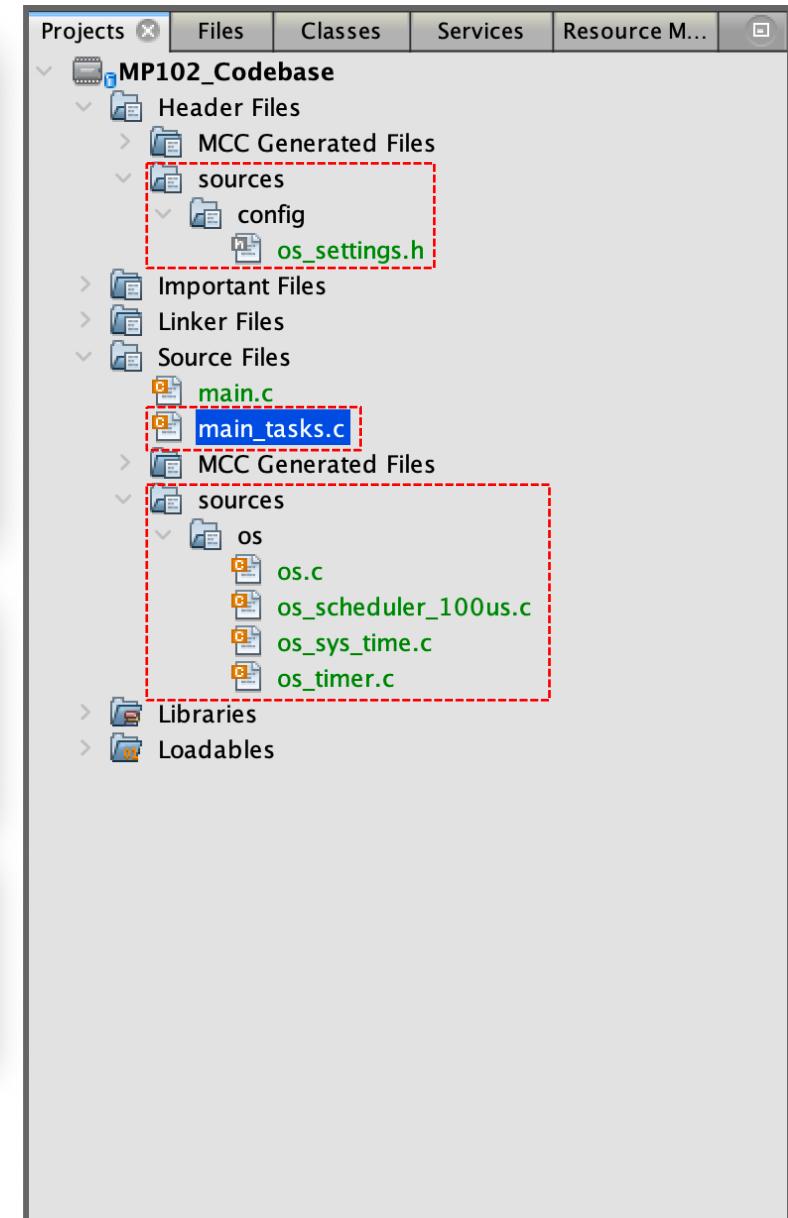
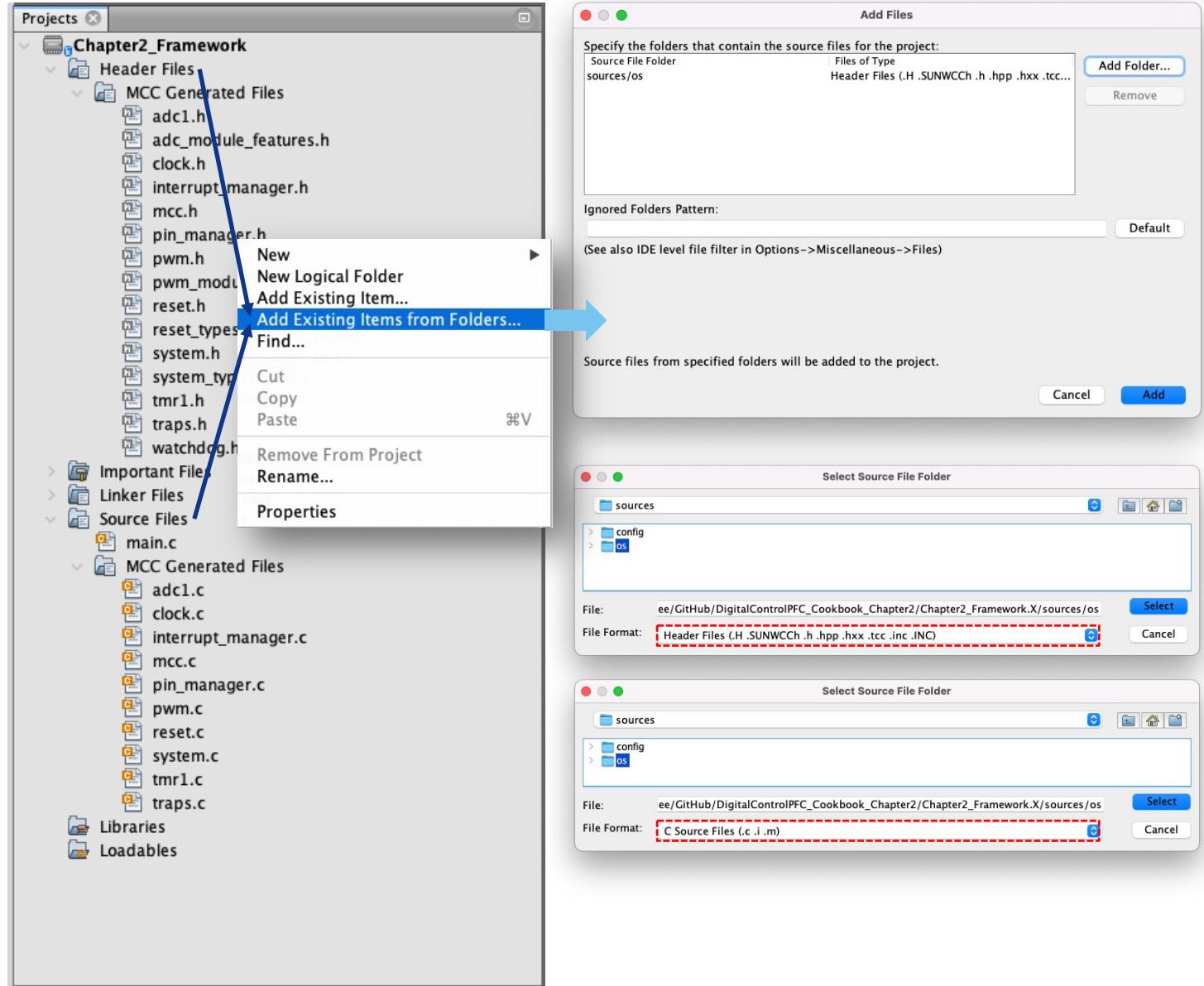
Directory Structure

| Directory path | Description |
|--|--|
| ExampleProject.X | main project directory |
| ExampleProject.X\main.c | the main.c file should be in the main project directory because MCC puts it in here. |
| ExampleProject.X\main_tasks.c | the main_tasks.c file that the MCOS (Microchip Cooperative Operating System) requires should be in the same directory where the main.c file resides |
| ExampleProject.X\sources | all the other sub directories for sources and includes files should be under the directory 'sources' |
| ExampleProject.X\sources\app | the 'app' directory should contain the application logic. this is just a suggestion. depending on the project there might be a reason to have the application logic somewhere else. filenames in this directory should start with "app_" |
| ExampleProject.X\sources\config 系統設定檔案，例如: os_settings.h | the sub directory config is for storing different kind of configuration files. For configuring the MCOS (Microchip Cooperative Operating System) we use the file 'os_settings.h' (former 'project_settings.h' in older versions) In the case of the MCOS, is must be here, or else it cannot be found by the operating system files. |
| ExampleProject.X\sources\device 對外通訊連結，例如: dev_gui_comm.c/.h.h | the 'device' directory should contain drivers that handle I/O streams similar to a serial interface. Usually that devices have a send and receive function. In some of our projects there is the 'dev_gui_comm.c' communication layer for communication with the GUI here. filenames in this directory should start with "dev_" |
| ExampleProject.X\sources\driver 通訊以外的週邊驅動功能，例如: LED, I/O..... | the 'driver' directory should contain all the drivers that do not fit in the device directory. filenames in this directory should start with "drv_". Examples: drv_led.c; drv_button.c; drv_adc.c, |
| ExampleProject.X\sources\driver\power_controller 較為複雜的週邊驅動功能，例如: 電源控制 | for more complex drivers that consists of multiple files it makes sense to create a sub directory. Example: In our power controller development boards we use the sub directory "power_controllers" |
| ExampleProject.X\sources\misc 共用的雜項檔案，例如: fault_common.c | directory to store miscellaneous files. Examples: global.h; fault/fault_common.c; ... |
| ExampleProject.X\sources\os OS管理目錄，例如: os_scheduler.c | directory for individual Operating System files, like the MCOS (Microchip Cooperative Operating System) |

OS Scheduler Example



Integrate OS_Scheduler

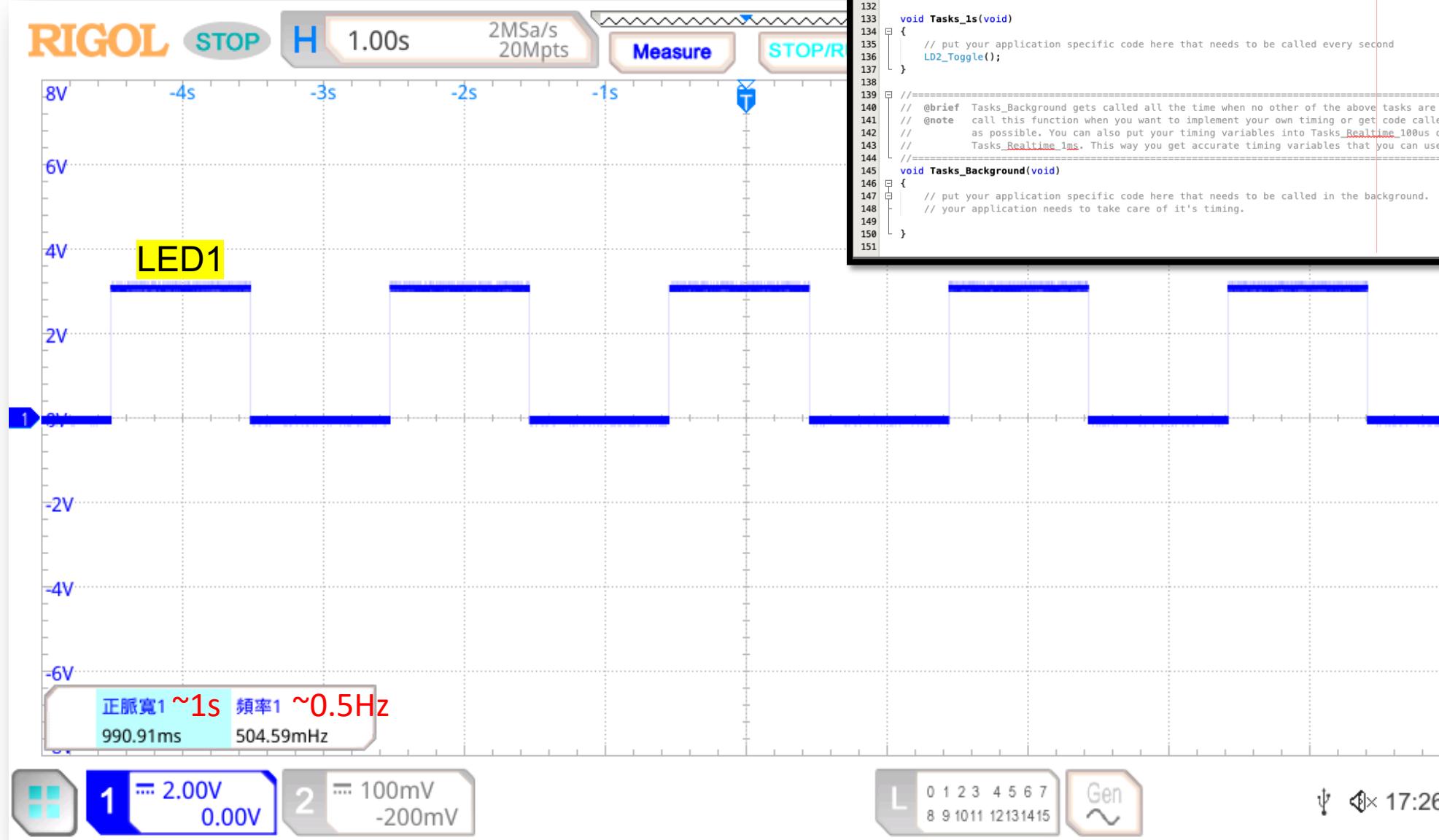


Main & Tasks

```
main.c
47  */
48  #include "mcc_generated_files/system.h"
49  #include "sources/os/os.h"
50
51  /*
52   *          Main application
53   */
54  int main(void)
55  {
56      // initialize the device
57      SYSTEM_Initialize();
58
59      // OS
60      OS_Init();
61      OS_Scheduler_RunForever();
62
63      while (1)
64      {
65          // Add your application code
66      }
67      return 1;
68 }
69 /**
70 | End of File
71 */
```

```
main_tasks.c
130 // @note there could be some jitter here because it is not called directly by a timer
131 //=====
132
133 void Tasks_1s(void)
134 {
135     // put your application specific code here that needs to be called every second
136     LED1_Toggle();
137 }
138
139 //=====
140 // @brief Tasks_Background gets called all the time when no other of the above tasks
```

0.5Hz LED1 Toggle



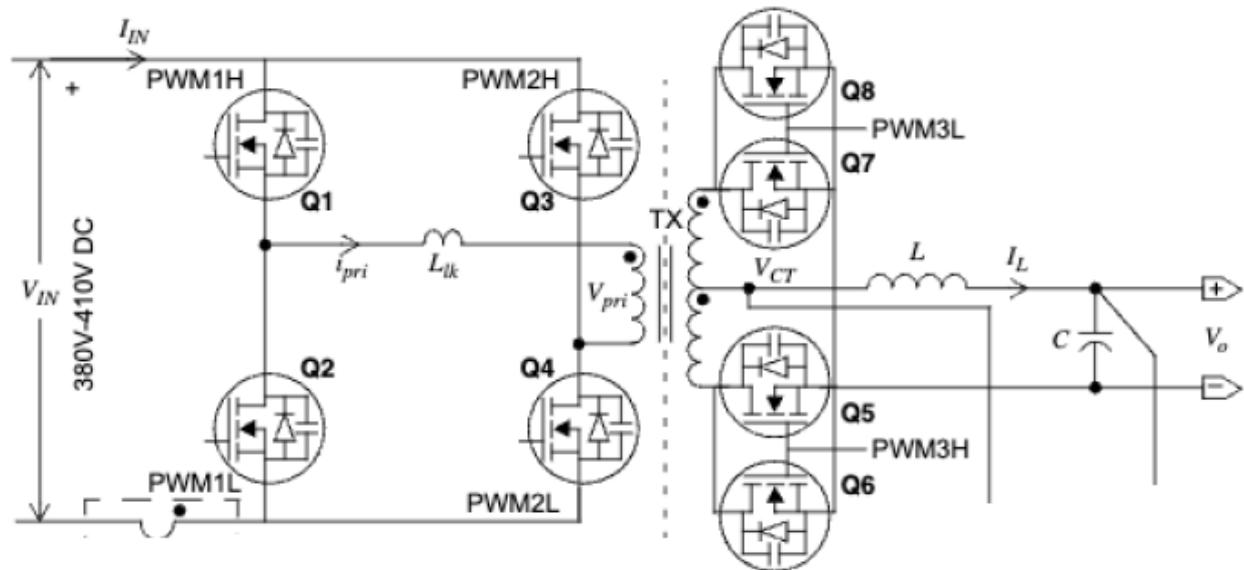
VMC PSFB Converter With SR

Peripheral & Tool Descriptions

- Peripherals (in RED)
 - Timer1 100us
 - IP = 3 & NO Context
 - LED1 toggling in Tasks_1s
 - I/O
 - LED1 – RB4
- Chip: dsPIC33CK64MP102
- S/W Versions
 - MPLAB X IDE v6.20
 - DFP v1.13.366
 - Compiler XC16 v2.1
 - MCC v5.5.0
 - Core v5.7.0
 - MCU Lib v1.171.4

Peripheral & Tool Descriptions

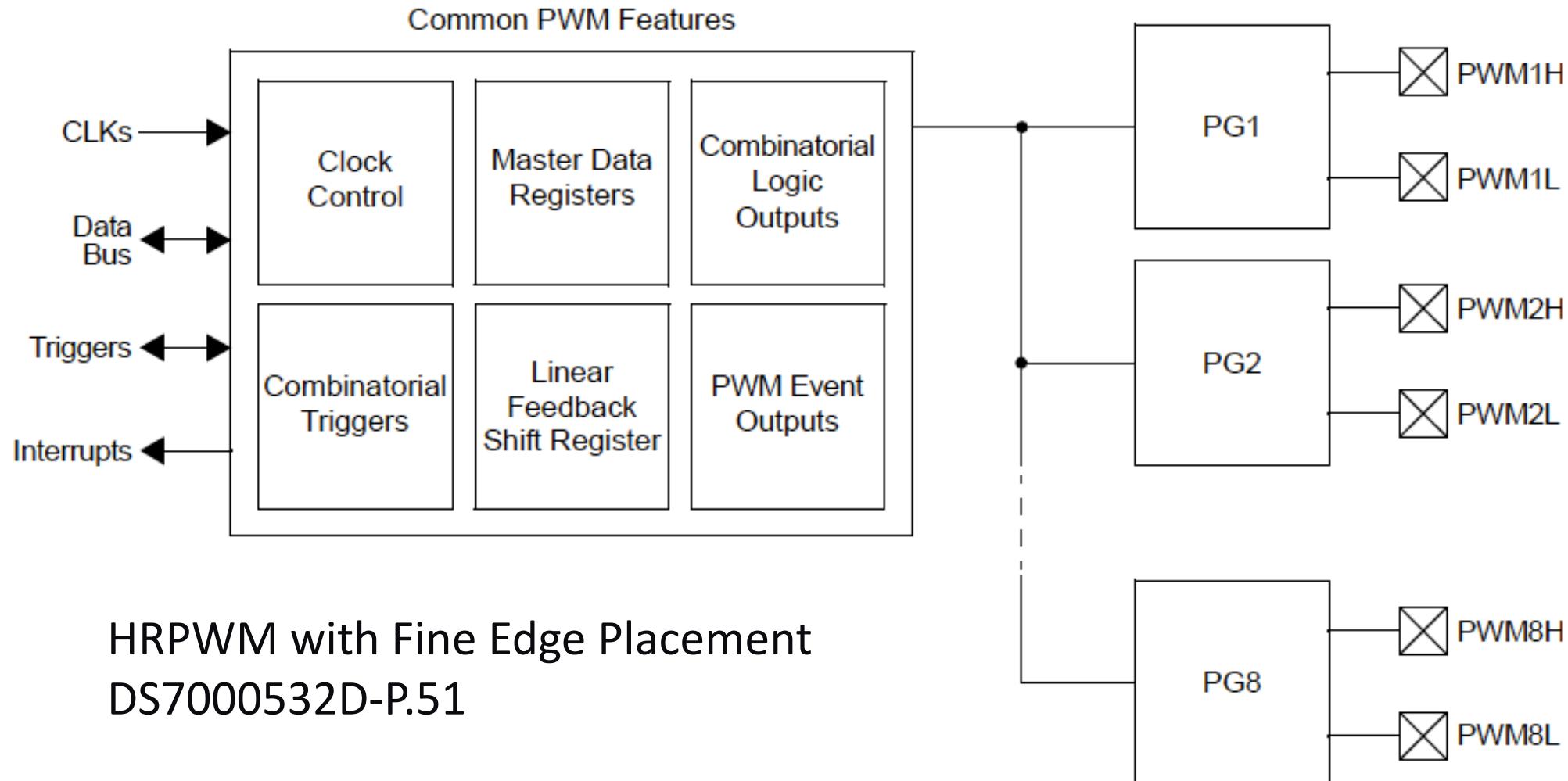
- Peripherals (in RED)
 - PWM1 (Fixed leg)
 - 100kHz (Complementary) with fixed 50% DC
 - Trigger A -> Phase-shifted value to PWM3
 - Trigger B -> ANO Trigger
 - PWM2 (Phase-shifted leg)
 - 100kHz (Complementary) with fixed 50% DC
 - SOC from PC1 (PWM1 Trigger A)
 - Trigger A -> 200ns reserved for SR PWM
 - Trigger B -> Reserved
 - PWM3 (SR)
 - 100kHz (Complementary) with fixed 0% DC
 - SOC from PC2 (PWM2 Trigger A)
 - Trigger A -> Reserved
 - Trigger B -> Reserved
 - Combinatorial Logic Output
 - PWM3H = PWM1H && PWM2L
 - PWM3L = PWM1L && PWM2H
- AN0 (Port Rxx)
 - IP = 5 & NO Context
 - Triggered by PWM1 Trigger2 (B)
 - ISR 100KHz



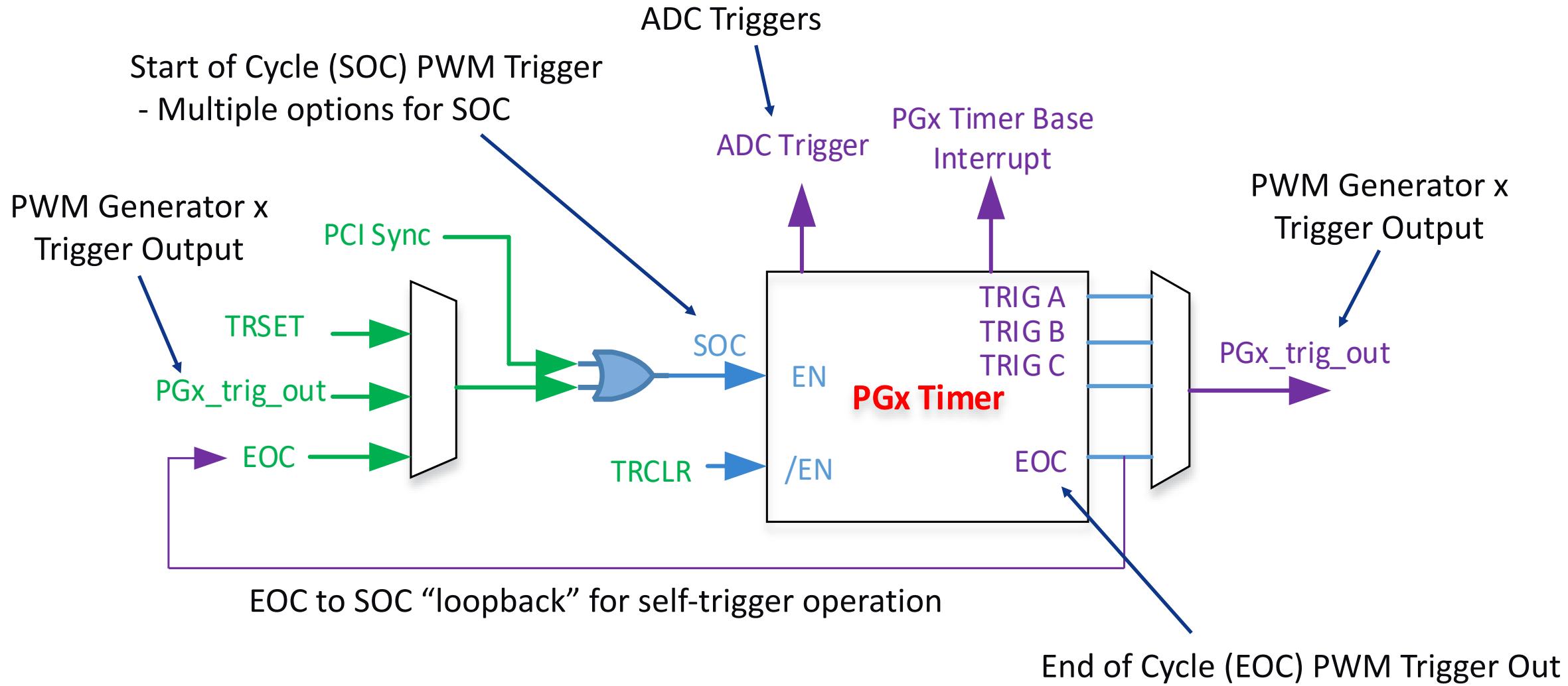
PWM Ideas

PWM High-Level Block Diagram

Figure 3-1. PWM High-Level Block Diagram



dsPIC33C PGx Triggers



Single PWM Generator

HRPWM with Fine Edge Placement P.32

Bits 2:0 – PGTRGSEL[2:0] **PWM Generator Trigger Output Selection**

| Value | Description |
|-------|---|
| 011 | PGxTRIGC compare event is the PWM Generator trigger |
| 010 | PGxTRIGB compare event is the PWM Generator trigger |
| 001 | PGxTRIGA compare event is the PWM Generator trigger |
| 000 | EOC event is the PWM Generator trigger |

HRPWM with Fine Edge Placement P.25

Bits 3:0 – SOCS[3:0] **Start-of-Cycle (SOC) Selection bits**

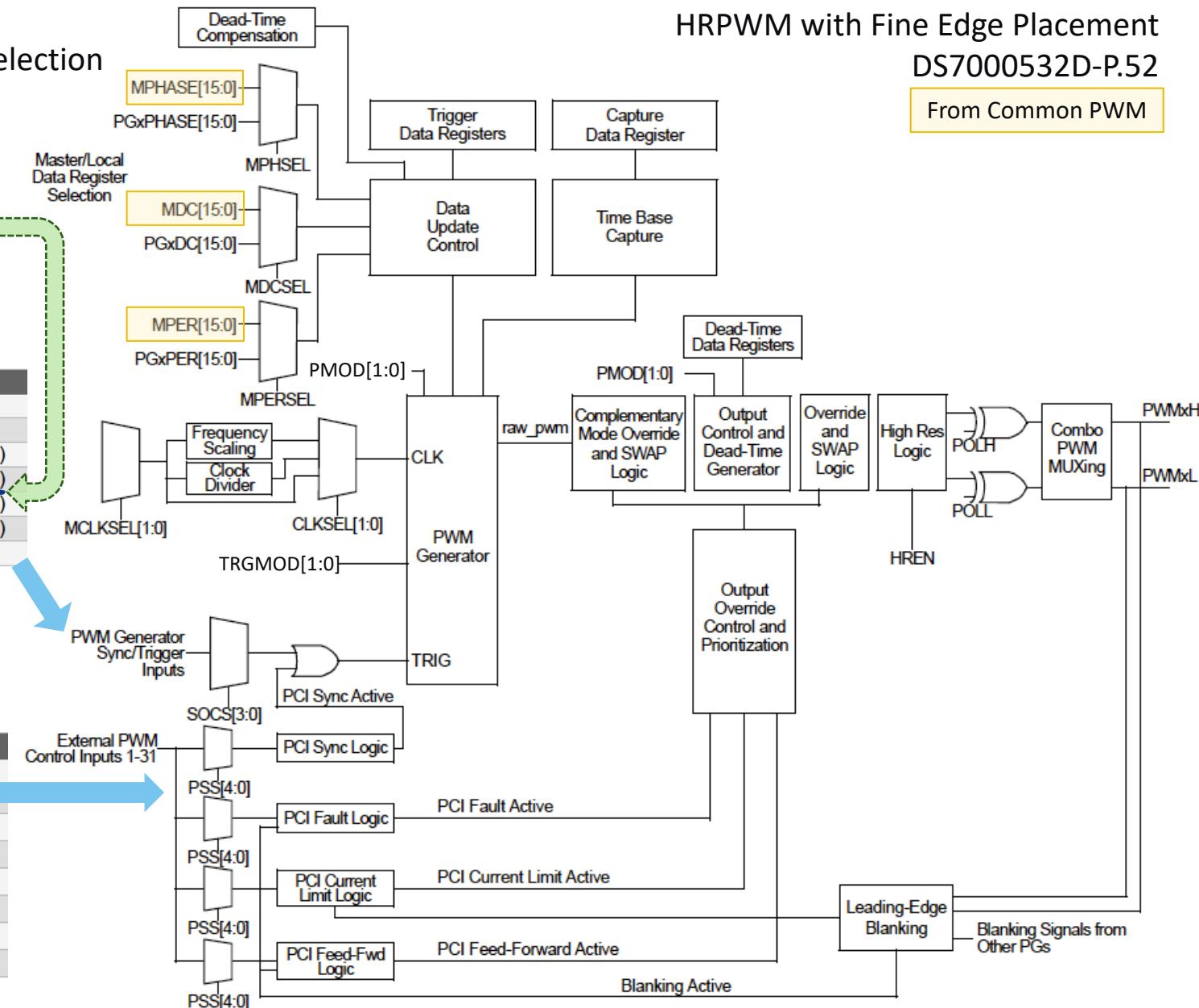
| Value | Description |
|-------|---|
| 1111 | TRIG bit or PCI Sync function only (no hardware trigger source is selected) |
| 1110 | Reserved |
| 0100 | Trigger output selected by PG4 or PG8 PGTRGSEL[2:0] bits (PGxEVTL[2:0]) |
| 0011 | Trigger output selected by PG3 or PG7 PGTRGSEL[2:0] bits (PGxEVTL[2:0]) |
| 0010 | Trigger output selected by PG2 or PG6 PGTRGSEL[2:0] bits (PGxEVTL[2:0]) |
| 0001 | Trigger output selected by PG1 or PG5 PGTRGSEL[2:0] bits (PGxEVTL[2:0]) |
| 0000 | Local EOC – PWM Generator is self-triggered |

HRPWM with Fine Edge Placement P.36

Bits 4:0 – PSS[4:0] **PCI Source Selection**

Note: PCI sources are device-dependent; refer to the device data sheet for availability.

| Value | Description |
|-------|---|
| 11111 | PCI Source #31 (reserved) |
| ... | ... |
| 00101 | PCI Source #5 (reserved) |
| 00100 | PCI Source #4 (reserved) |
| 00011 | PCI Source #3 (internally connected to Combinatorial Trigger B) |
| 00010 | PCI Source #2 (internally connected to Combinatorial Trigger A) |
| 00001 | PCI Source #1 (internally connected to PWMPCI[2:0] output MUX) |
| 00000 | Software PCI control bit (SWPCI) only |



Single PWM Generator

dsPIC33CK256MP508

Data Sheet DS70005349H P.288

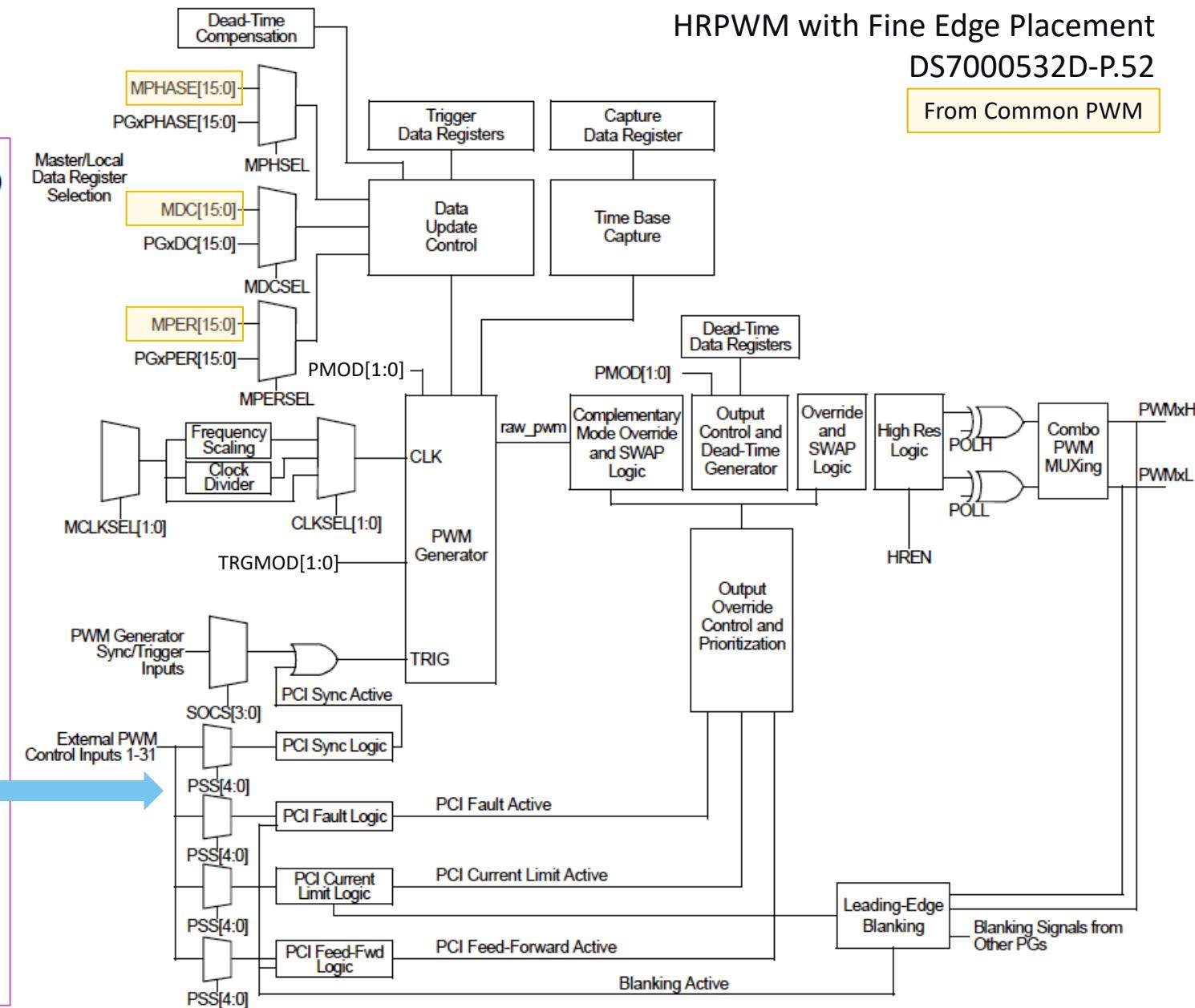
Bits 4:0 – PSS[4:0] PCI Source Selection

**REGISTER 12-19: PGxyPCIL: PWM GENERATOR xy PCI REGISTER LOW
(x = PWM GENERATOR #; y = F, CL, FF OR S) (CONTINUED)**

bit 4-0 **PSS[4:0]**: PCI Source Selection bits

- 11111 = CLC1
- 11110 = Reserved
- 11101 = Comparator 3 output
- 11100 = Comparator 2 output
- 11011 = Comparator 1 output
- 11010 = PWM Event D
- 11001 = PWM Event C
- 11000 = PWM Event B
- 10111 = PWM Event A
- 10110 = Device pin, PCI[22]
- 10101 = Device pin, PCI[21]
- 10100 = Device pin, PCI[20]
- 10011 = Device pin, PCI[19]
- 10010 = RPN input, PCI18R
- 10001 = RPN input, PCI17R
- 10000 = RPN input, PCI16R
- 01111 = RPN input, PCI15R
- 01110 = RPN input, PCI14R
- 01101 = RPN input, PCI13R
- 01100 = RPN input, PCI12R
- 01011 = RPN input, PCI11R
- 01010 = RPN input, PCI10R
- 01001 = RPN input, PCI9R
- 01000 = RPN input, PCI8R
- 00111 = Reserved
- 00110 = Reserved
- 00101 = Reserved
- 00100 = Reserved
- 00011 = Internally connected to Combo Trigger B
- 00010 = Internally connected to Combo Trigger A
- 00001 = Internally connected to the output of PWMPCI[2:0] MUX
- 00000 = Tied to '0'

PGx PCI Sync



Single PWM Generator

dsPIC33CK256MP508

Data Sheet DS70005349H P.288

Bits 4:0 – PSS[4:0] PCI Source Selection

**REGISTER 12-19: PGxyPCIL: PWM GENERATOR xy PCI REGISTER LOW
(x = PWM GENERATOR #; y = F, CL, FF OR S) (CONTINUED)**

bit 4-0 **PSS[4:0]: PCI Source Selection bits**

- 11111 = CLC1
- 11110 = Reserved
- 11101 = Comparator 3 output
- 11100 = Comparator 2 output
- 11011 = Comparator 1 output
- 11010 = PWM Event D
- 11001 = PWM Event C
- 11000 = PWM Event B
- 10111 = PWM Event A
- 10110 = Device pin, PCI[22]
- 10101 = Device pin, PCI[21]
- 10100 = Device pin, PCI[20]
- 10011 = Device pin, PCI[19]
- 10010 = RPh input, PCI18R
- 10001 = RPh input, PCI17R
- 10000 = RPh input, PCI16R
- 01111 = RPh input, PCI15R
- 01110 = RPh input, PCI14R
- 01101 = RPh input, PCI13R
- 01100 = RPh input, PCI12R
- 01011 = RPh input, PCI11R
- 01010 = RPh input, PCI10R
- 01001 = RPh input, PCI9R
- 01000 = RPh input, PCI8R
- 00111 = Reserved
- 00110 = Reserved
- 00101 = Reserved
- 00100 = Reserved
- 00011 = Internally connected to Combo Trigger B
- 00010 = Internally connected to Combo Trigger A
- 00001 = Internally connected to the output of PWMPCI[2:0] MUX
- 00000 = Tied to '0'

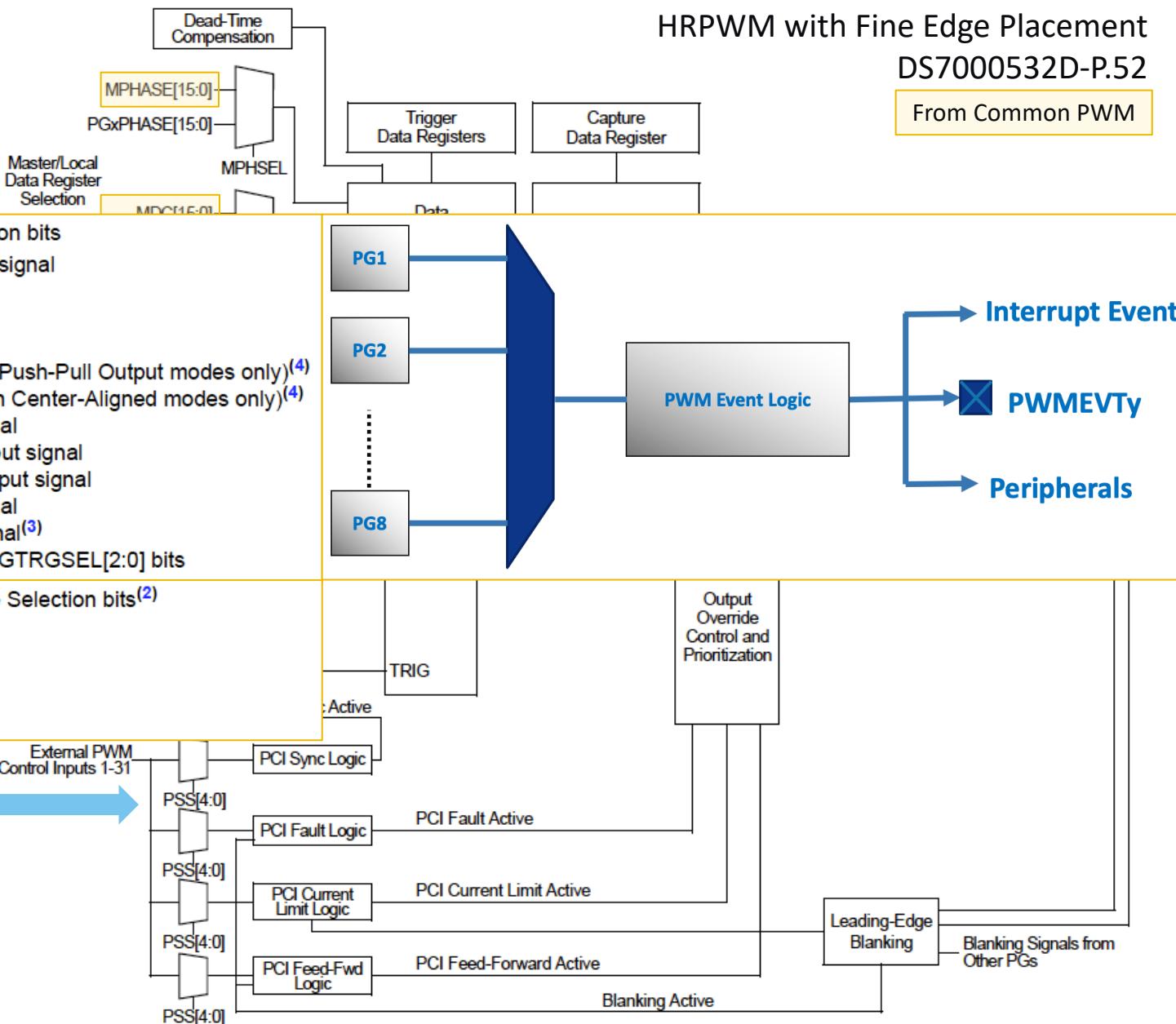
EVTySEL[3:0]: PWM Event Selection bits

- 1111 = High-resolution error event signal
- 1110-1010 = Reserved
- 1001 = ADC Trigger 2 signal
- 1000 = ADC Trigger 1 signal
- 0111 = STEER signal (available in Push-Pull Output modes only)⁽⁴⁾
- 0110 = CAHALF signal (available in Center-Aligned modes only)⁽⁴⁾
- 0101 = PCI Fault active output signal
- 0100 = PCI current-limit active output signal
- 0011 = PCI feed-forward active output signal
- 0010 = PCI Sync active output signal
- 0001 = PWM Generator output signal⁽³⁾
- 0000 = Source is selected by the PGTRGSEL[2:0] bits

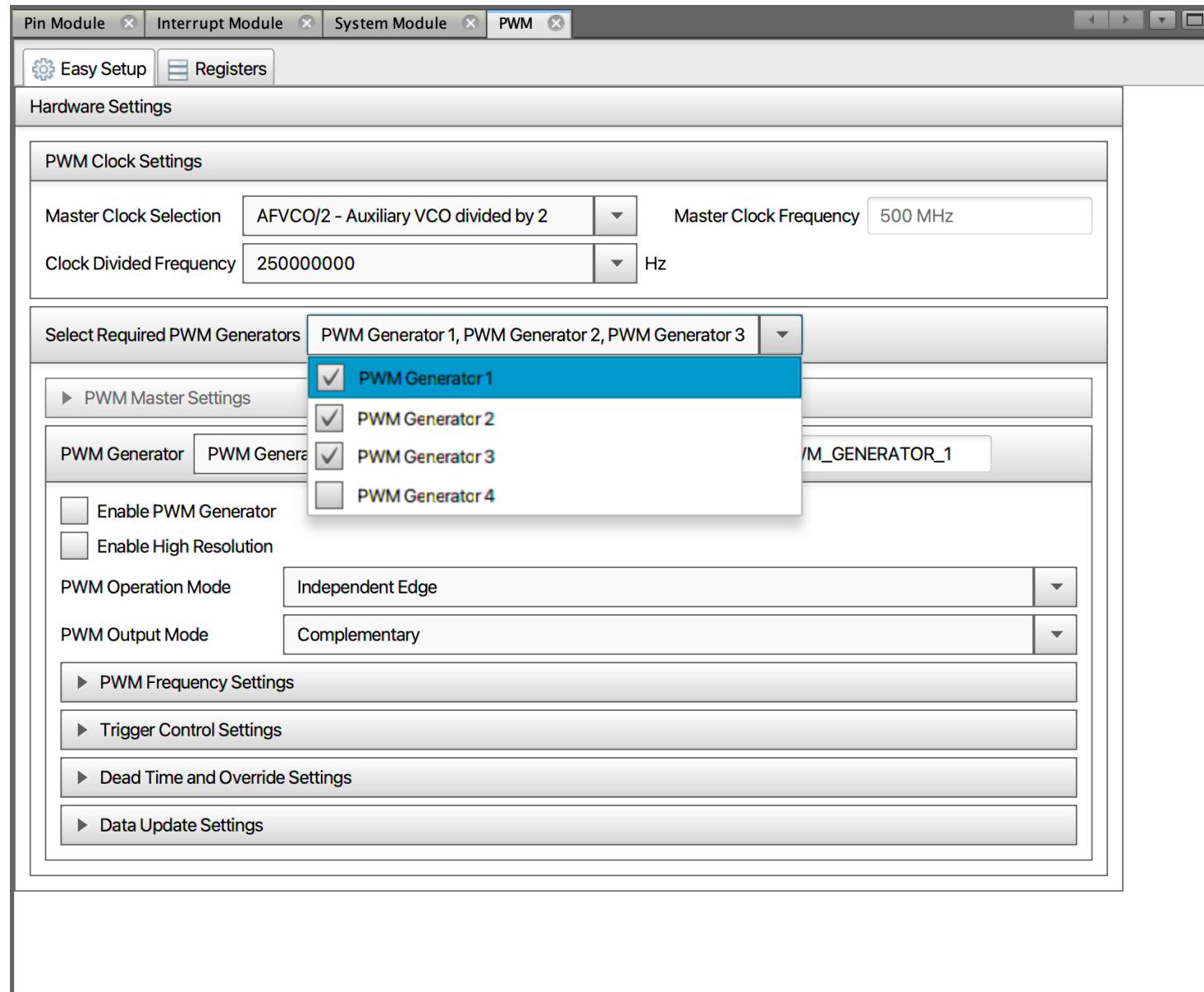
EVTyPGS[2:0]: PWM Event Source Selection bits⁽²⁾

- 111 = PWM Generator 8
- 110 = PWM Generator 7
- ...
- 000 = PWM Generator 1

PGx PCI Sync



Select PWM1~3



PWM1 Settings

Easy Setup **Registers**

Select Required PWM Generators: PWM Generator 1, PWM Generator 2, PWM Generator 3

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 Frequency Settings

PWM Input Clock Selection: 500000000 Hz

Period Use Master Period

| | | |
|------------------------------------|---------|-------------------|
| Requested Frequency 61.02864 kHz ≤ | 100 kHz | ≤ 29.41176471 MHz |
| Calculated Frequency | 100 kHz | |
| Requested Period 34 ns | ≤ 10 us | ≤ 16.3858 us |
| Calculated Period | 10 us | |

Duty Cycle Use Master Duty Cycle

| | | |
|----------------------|----|---------|
| PWM Duty Cycle 0 % ≤ | 50 | ≤ 100 % |
|----------------------|----|---------|

Phase

| | | |
|------------------|------|--------------|
| PWM Phase 0 ns ≤ | 0 ns | ≤ 16.3838 us |
|------------------|------|--------------|

Trigger Control Settings

Dead Time and Override Settings

Data Update Settings

Trigger Control Settings

PWM Start of Cycle Control

Start of Cycle Trigger: Self-trigger
Trigger Output Selection: Trigger A compare event

ADC Trigger

ADC Trigger 1: Trigger A Compare
ADC Trigger 2: Trigger B Compare

Output Trig1A as the phase-shifted value of PWMx

Trigger A Compare: 0 ns ≤ 0 ns ≤ 16.3838 us Default phase-shifted value

Trigger B Compare: 0 ns ≤ 3 us ≤ 16.3838 us Default ADC Trigger point = (End edge of +Cycle) – 2us = 5us – 2us = 3us

Trigger C Compare: 0 ns ≤ 0 ns ≤ 16.3838 us

Dead Time and Override Settings

PWM L Dead Time Delay: 0 ns ≤ 150 ns ≤ 4.0958 us
PWM H Dead Time Delay: 0 ns ≤ 150 ns ≤ 4.0958 us

PWM L Override: disabled
PWM H Override: disabled

Data Update Settings

Update Trigger: Trigger A Updating Trig1A with updating Duty/Period/...in sync
Update Mode: SOC update

PWM2 Settings

Easy Setup Registers

Select Required PWM Generators: PWM Generator 1, PWM Generator 2, PWM Generator 3

PWM Master Settings

PWM Generator: PWM Generator 2 Custom Name: PWM_GENERATOR_1

Enable PWM Generator
 Enable High Resolution

PWM Operation Mode: Independent Edge
PWM Output Mode: Complementary

PWM Frequency Settings

PWM Input Clock Selection: 500000000 Hz

Period: Use Master Period

Requested Frequency: 61.02864 kHz ≤ 100 kHz ≤ 29.41176471 MHz
Calculated Frequency: 100 kHz

Requested Period: 34 ns ≤ 10 us ≤ 16.3858 us
Calculated Period: 10 us

Duty Cycle: Use Master Duty Cycle

PWM Duty Cycle: 0 % ≤ 50 ≤ 100 %

Phase

PWM Phase: 0 ns ≤ 0 ns ≤ 16.3838 us

Trigger Control Settings

Dead Time and Override Settings

Data Update Settings

Trigger Control Settings

PWM Start of Cycle Control

Start of Cycle Trigger: Trigger output selected by PG1 or PG5 → SOC from PC1 (Trig1A)
Trigger Output Selection: Trigger A compare event → Output Trig2A as the shifted delay of SR as needed

ADC Trigger

ADC Trigger 1: None
ADC Trigger 2: None

Trigger A Compare: 0 ns ≤ 200 ns ≤ 16.3838 us → Default shifted delay value

Trigger B Compare: 0 ns ≤ 0 ns ≤ 16.3838 us

Trigger C Compare: 0 ns ≤ 0 ns ≤ 16.3838 us

Dead Time and Override Settings

PWM L Dead Time Delay: 0 ns ≤ 150 ns ≤ 4.0958 us
PWM H Dead Time Delay: 0 ns ≤ 150 ns ≤ 4.0958 us

PWM L Override: disabled
PWM H Override: disabled

Data Update Settings

Update Trigger: Trigger A → Updating Trig2A with updating Duty/Period/...in sync
Update Mode: SOC update → Immediate update for changing Fpwm as well

PWM3 Settings

Select Required PWM Generators: PWM Generator 1, PWM Generator 2, PWM Generator 3

PWM Master Settings

PWM Generator: PWM Generator 3
Custom Name: PWM_GENERATOR_3

Enable PWM Generator
 Enable High Resolution

PWM Operation Mode: Independent Edge
Pulse Width Modulation Output Mode: Complementary

PWM Frequency Settings

PWM Input Clock Selection: 500000000 Hz

Period: Use Master Period

Requested Frequency: 61.02864 kHz ≤ 100 kHz ≤ 29.41176471 MHz
Calculated Frequency: 100 kHz

Requested Period: 34 ns ≤ 10 us ≤ 16.3858 us
Calculated Period: 10 us

Duty Cycle: Use Master Duty Cycle

PWM Duty Cycle: 0 % ≤ 50 ≤ 100 %

Phase

PWM Phase: 0 ns ≤ 0 ns ≤ 16.3838 us

Trigger Control Settings

Dead Time and Override Settings

Data Update Settings

Trigger Control Settings

PWM Start of Cycle Control

Start of Cycle Trigger: Trigger output selected by PG2 or PG6
Trigger Output Selection: EOC event

ADC Trigger

ADC Trigger 1: None
ADC Trigger 2: None

Trigger A Compare: 0 ns ≤ 0 ns ≤ 16.3838 us
Trigger B Compare: 0 ns ≤ 0 ns ≤ 16.3838 us
Trigger C Compare: 0 ns ≤ 0 ns ≤ 16.3838 us

Dead Time and Override Settings

PWM L Dead Time Delay: 0 ns ≤ 150 ns ≤ 4.0958 us
PWM H Dead Time Delay: 0 ns ≤ 150 ns ≤ 4.0958 us

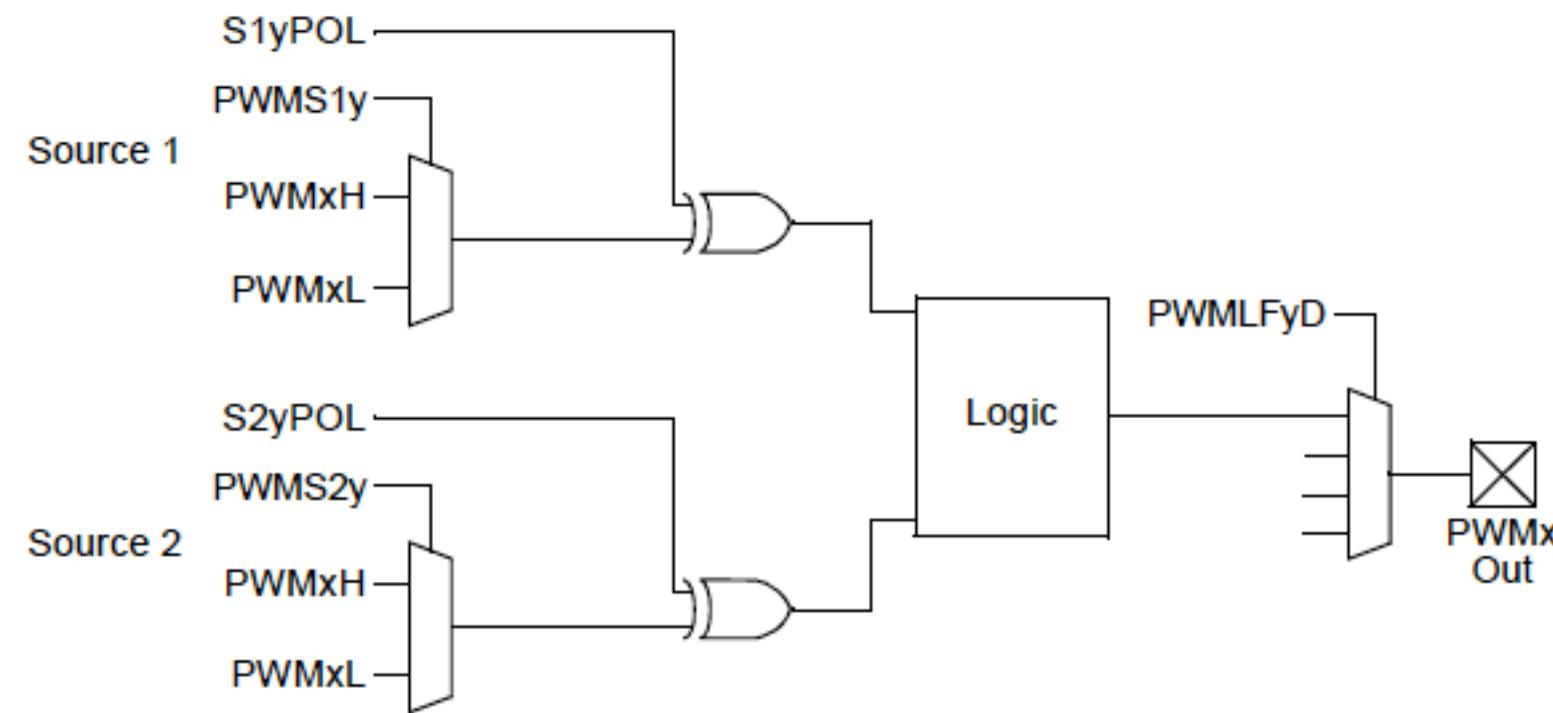
PWM L Override: disabled
PWM H Override: disabled

Data Update Settings

Update Trigger: Duty Cycle
Update Mode: SOC update

Combinatorial Logic Output

| Register | Combinatorial Logic Instance | Available Output Pin Selection |
|----------|------------------------------|--------------------------------|
| LOGCONA | A | PWM2H-PWM8H |
| LOGCONB | B | PWM2L-PWM8L |
| LOGCONC | C | PWM2H-PWM8H |
| LOGCOND | D | PWM2L-PWM8L |
| LOGCONE | E | PWM2H-PWM8H |
| LOGCONF | F | PWM2L-PWM8L |



▼ Register: LOGCONA 0x312

| | | |
|--|-----------------|---|
| <input checked="" type="radio"/> PWMLFA | PWMS1 and PWMS2 | ▼ |
| <input checked="" type="radio"/> PWMLFAD | PWM3H/PWM3L | ▼ |
| <input checked="" type="radio"/> PWMS1A | PWM1H | ▼ |
| <input checked="" type="radio"/> PWMS2A | PWM2L | ▼ |
| <input checked="" type="radio"/> S1APOL | Positive logic | ▼ |
| <input checked="" type="radio"/> S2APOL | Positive logic | ▼ |

▼ Register: LOGCONB 0x1212

| | | |
|--|-----------------|---|
| <input checked="" type="radio"/> PWMLFB | PWMS1 and PWMS2 | ▼ |
| <input checked="" type="radio"/> PWMLFBD | PWM3H/PWM3L | ▼ |
| <input checked="" type="radio"/> PWMS1B | PWM1L | ▼ |
| <input checked="" type="radio"/> PWMS2B | PWM2H | ▼ |
| <input checked="" type="radio"/> S1BPOL | Positive logic | ▼ |
| <input checked="" type="radio"/> S2BPOL | Positive logic | ▼ |

ADC/AN0 Settings

The screenshot shows the ADC1 configuration window with the following settings:

ADC Clock

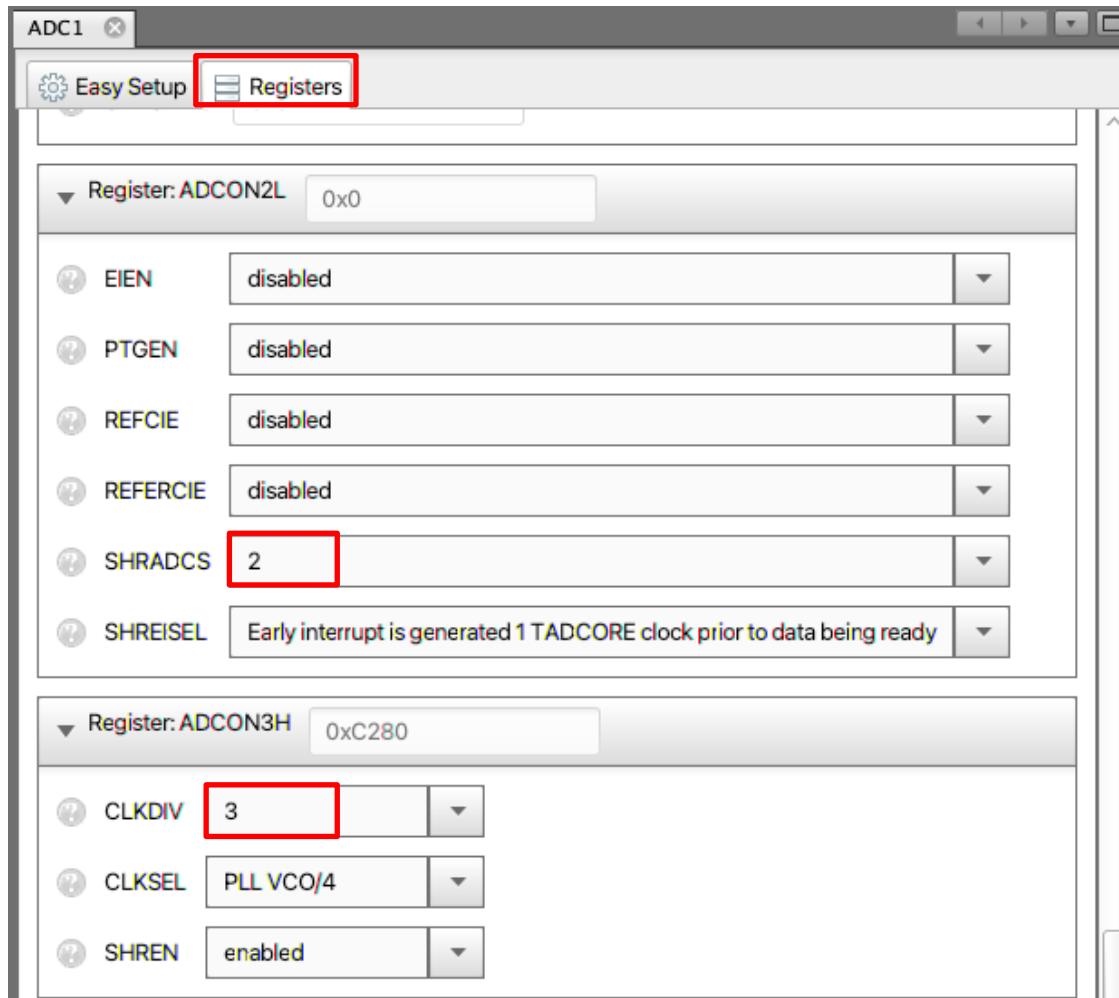
- Conversion Clock Source: PLL VCO/4 (highlighted with a red dashed box)
- Conversion Time: 92.5 ns
- Target Shared Core Sampling Time: 40 ns
- Calculated Shared Core Sampling Time: 40 ns

Selected Channels

| Core | Enable | Core Channel | Pin Name | Custom Name | Trigger Source | Compare | Interrupt |
|--------|-------------------------------------|--------------|----------|-------------|----------------|---------|-------------------------------------|
| Ccore0 | <input checked="" type="checkbox"/> | AN0 | RA0 | channel_AN0 | PWM1 Trigg... | None | <input checked="" type="checkbox"/> |
| Ccore1 | <input type="checkbox"/> | AN1 | RB2 | | | None | <input type="checkbox"/> |
| Shared | <input type="checkbox"/> | AN10 | RB8 | | None | None | <input type="checkbox"/> |
| Shared | <input type="checkbox"/> | AN11 | RB9 | | None | None | <input type="checkbox"/> |
| Shared | <input type="checkbox"/> | AN12 | RC0 | | None | None | <input type="checkbox"/> |
| Shared | <input type="checkbox"/> | AN13 | RC1 | | None | None | <input type="checkbox"/> |
| Shared | <input type="checkbox"/> | AN14 | RC2 | | None | None | <input type="checkbox"/> |
| Shared | <input type="checkbox"/> | AN15 | RC3 | | None | None | <input type="checkbox"/> |
| Shared | <input type="checkbox"/> | AN16 | RC7 | | None | None | <input type="checkbox"/> |
| Shared | <input type="checkbox"/> | AN17 | RC6 | | None | None | <input type="checkbox"/> |

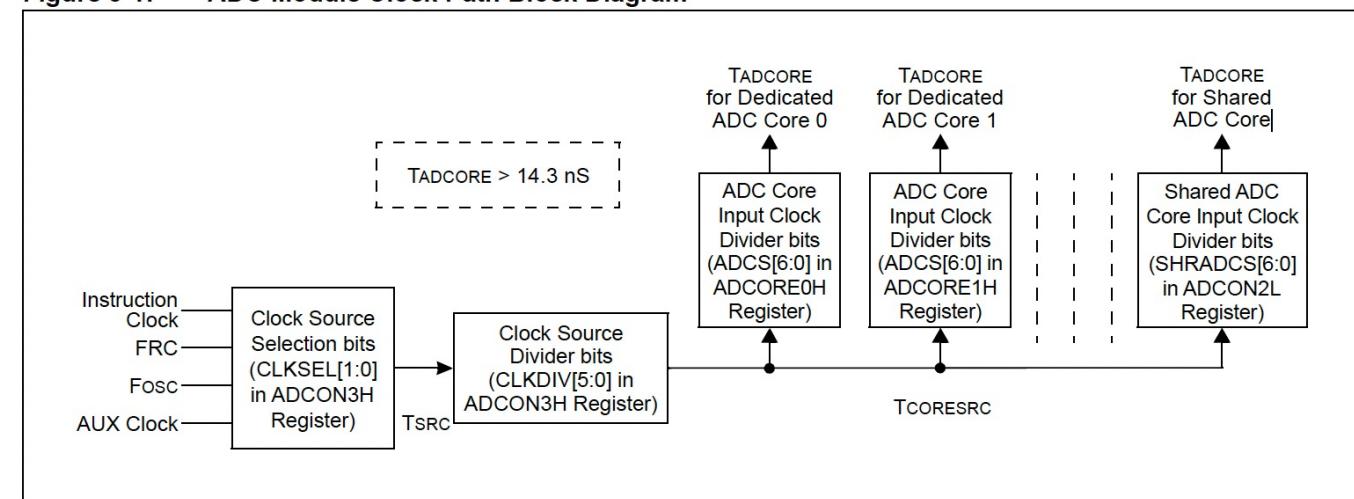
A blue callout box highlights the "Trigger Source" dropdown for Ccore0, which is set to "PWM1 Trigger2".

ADC/AN0 Settings



| ADC Shared Core | | |
|-------------------------|-----------|--|
| ADC Module Clock Source | 400 MHz | PLL VOC/4 |
| ADCON2L.SHRADCS | 2 | Shared ADC Core Input Clock Divider bits |
| ADCON3H.CLKDIV | 3 | ADC Module Clock Source Divider bits |
| Fadcore | 66.67 MHz | Clock Source / SHRADCS / CLKDIV |
| Tadcore | 15.00 ns | 1/Fadcore $\geq 14.3\text{ns}$ |

Figure 5-1: ADC Module Clock Path Block Diagram



Interrupt Manager

Easy Setup

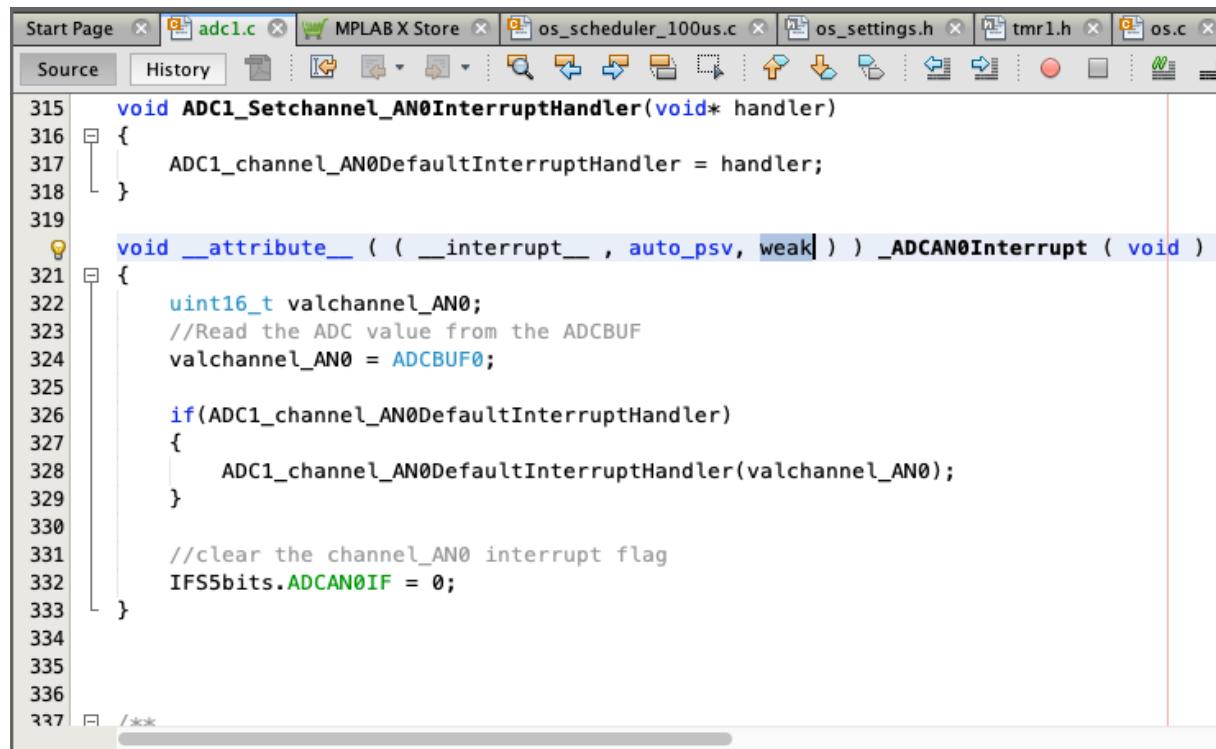
Interrupt Manager

Enable Global Interrupts

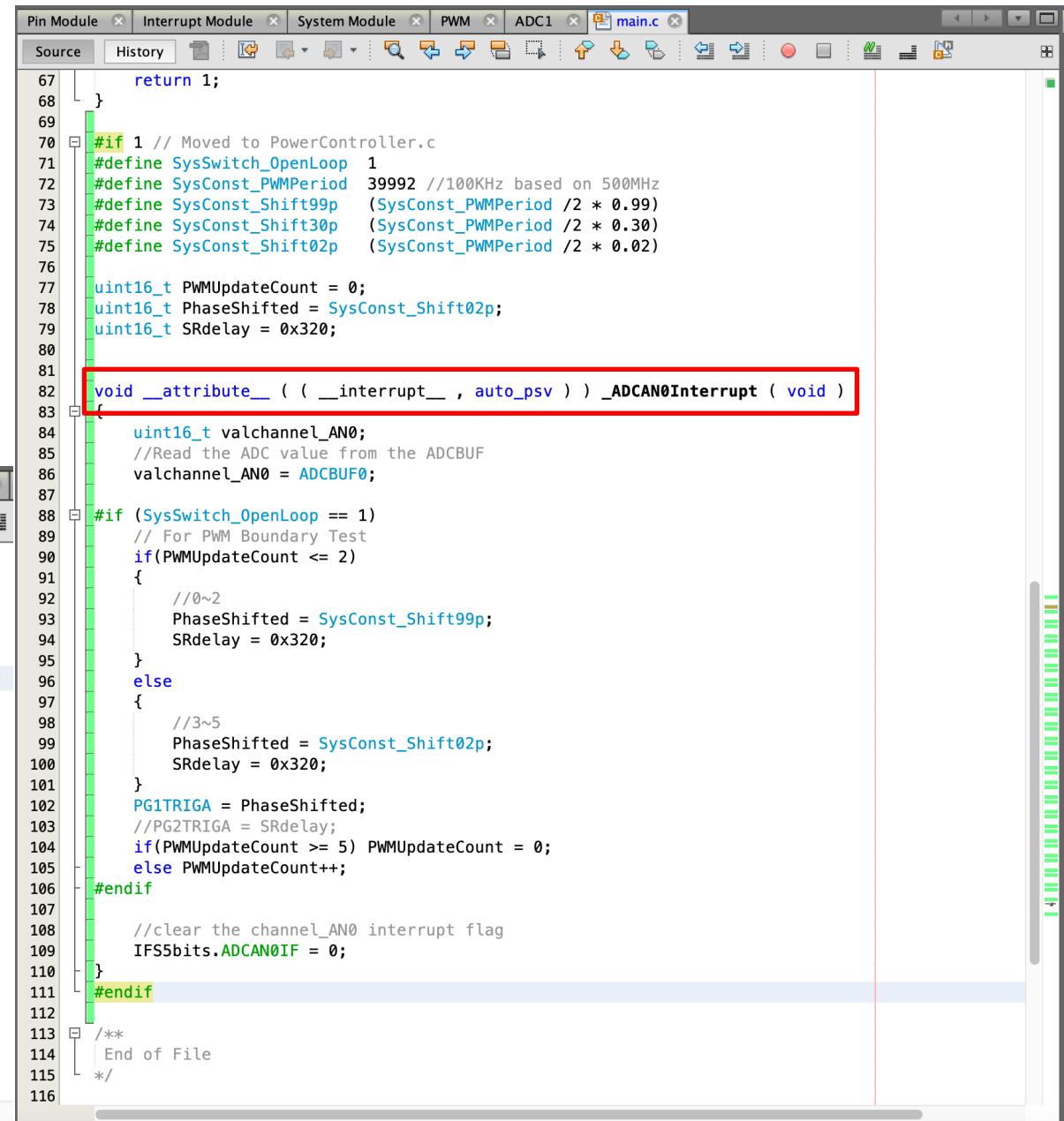
| Module | Interrupt | Description | IRQ Number | Enabled | Priority | Context |
|--------|-----------|--------------------------|------------|-------------------------------------|----------|---------|
| ADC1 | ADCAN6 | ADC AN6 Convert Done | 97 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCAN7 | ADC AN7 Convert Done | 98 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCAN24 | ADC AN24 Convert Done | 192 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCAN8 | ADC AN8 Convert Done | 99 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCAN9 | ADC AN9 Convert Done | 100 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCMP0 | ADC Digital Comparator 0 | 116 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCMP1 | ADC Digital Comparator 1 | 117 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCMP2 | ADC Digital Comparator 2 | 118 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCAN0 | ADC AN0 Convert Done | 91 | <input checked="" type="checkbox"/> | 5 | OFF |
| ADC1 | ADCAN1 | ADC AN1 Convert Done | 92 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCAN2 | ADC AN2 Convert Done | 93 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCAN3 | ADC AN3 Convert Done | 94 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCAN4 | ADC AN4 Convert Done | 95 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCMP3 | ADC Digital Comparator 3 | 119 | <input type="checkbox"/> | 1 | OFF |
| ADC1 | ADCAN25 | ADC AN25 Convert Done | 193 | <input type="checkbox"/> | 1 | OFF |

AN0 ISR in main.c

- Copied from adc1.c
- Removed “weak”

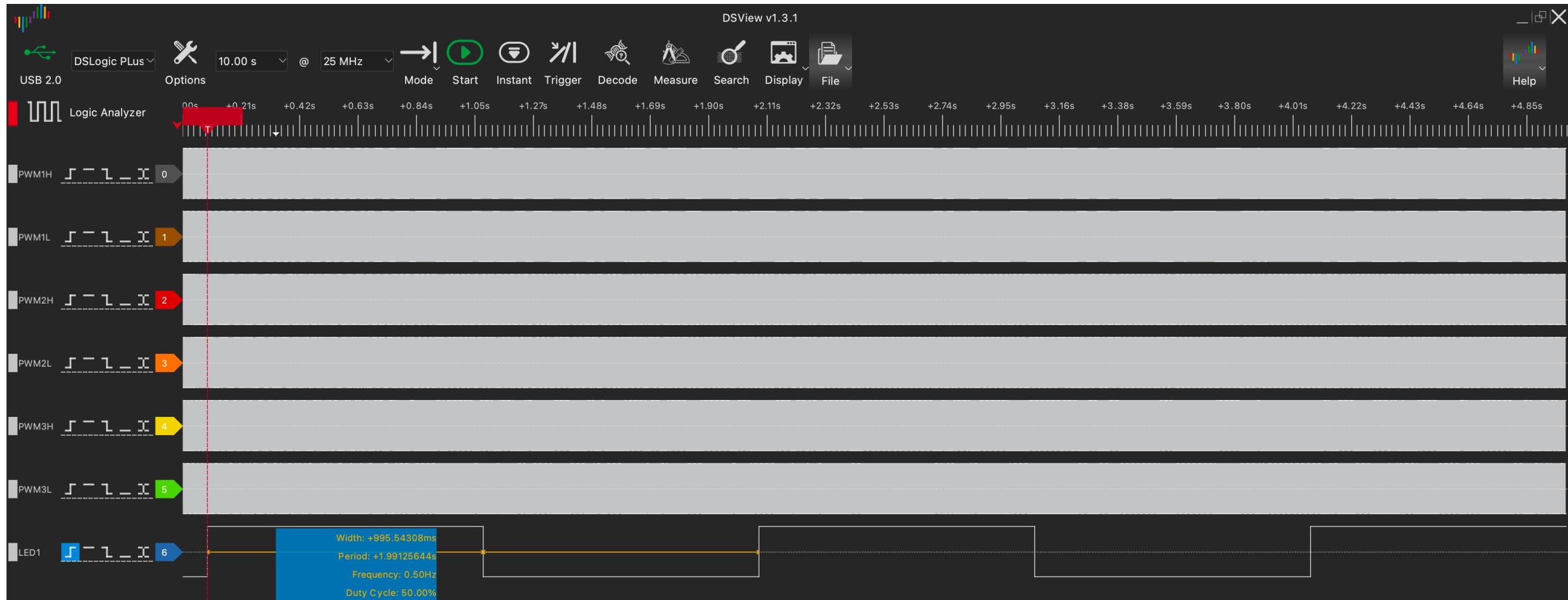


```
Start Page x adc1.c x MPLAB X Store x os_scheduler_100us.c x os_settings.h x tmr1.h x os.c x
Source History ... _ADCAN0Interrupt ( void )
315 void ADC1_Setchannel_AN0InterruptHandler(void* handler)
316 {
317     ADC1_channel_AN0DefaultInterruptHandler = handler;
318 }
319
320 void __attribute__ (( __interrupt__ , auto_psv, weak )) _ADCAN0Interrupt ( void )
321 {
322     uint16_t valchannel_AN0;
323     //Read the ADC value from the ADCBUF
324     valchannel_AN0 = ADCBUF0;
325
326     if(ADC1_channel_AN0DefaultInterruptHandler)
327     {
328         ADC1_channel_AN0DefaultInterruptHandler(valchannel_AN0);
329     }
330
331     //clear the channel_AN0 interrupt flag
332     IFS5bits.ADCAN0IF = 0;
333 }
```

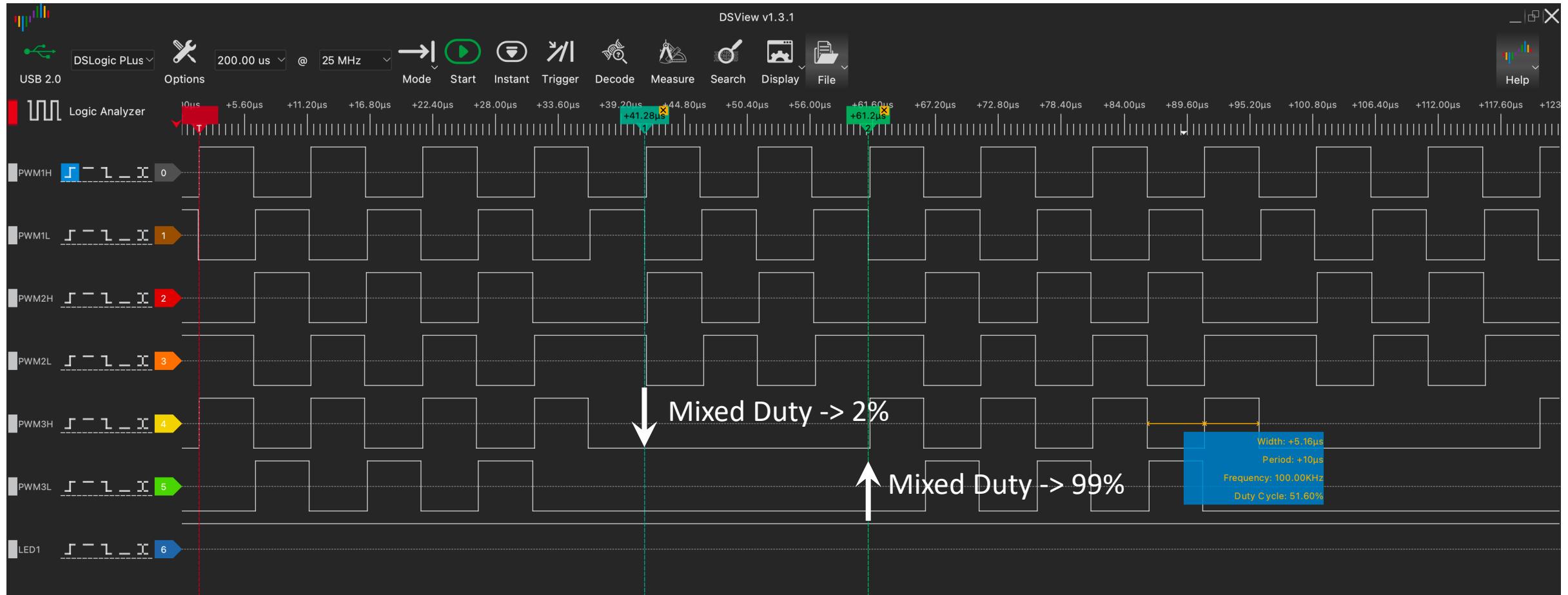


```
Pin Module x Interrupt Module x System Module x PWM x ADC1 x main.c x
Source History ... _ADCAN0Interrupt ( void )
67     return 1;
68 }
69
70 #if 1 // Moved to PowerController.c
71 #define SysSwitch_OpenLoop 1
72 #define SysConst_PWMPeriod 39992 //100KHz based on 500MHz
73 #define SysConst_Shift99p (SysConst_PWMPeriod /2 * 0.99)
74 #define SysConst_Shift30p (SysConst_PWMPeriod /2 * 0.30)
75 #define SysConst_Shift02p (SysConst_PWMPeriod /2 * 0.02)
76
77 uint16_t PWMUpdateCount = 0;
78 uint16_t PhaseShifted = SysConst_Shift02p;
79 uint16_t SRdelay = 0x320;
80
81 void __attribute__ (( __interrupt__ , auto_psv )) _ADCAN0Interrupt ( void )
82 {
83     uint16_t valchannel_AN0;
84     //Read the ADC value from the ADCBUF
85     valchannel_AN0 = ADCBUF0;
86
87 #if (SysSwitch_OpenLoop == 1)
88     // For PWM Boundary Test
89     if(PWMUpdateCount <= 2)
90     {
91         //0~2
92         PhaseShifted = SysConst_Shift99p;
93         SRdelay = 0x320;
94     }
95     else
96     {
97         //3~5
98         PhaseShifted = SysConst_Shift02p;
99         SRdelay = 0x320;
100    }
101
102    PG1TRIGA = PhaseShifted;
103    //PG2TRIGA = SRdelay;
104    if(PWMUpdateCount >= 5) PWMUpdateCount = 0;
105    else PWMUpdateCount++;
106 #endif
107
108    //clear the channel_AN0 interrupt flag
109    IFS5bits.ADCAN0IF = 0;
110 }
111
112 /**
113  * End of File
114 */
115
116
```

LED1 Toggling

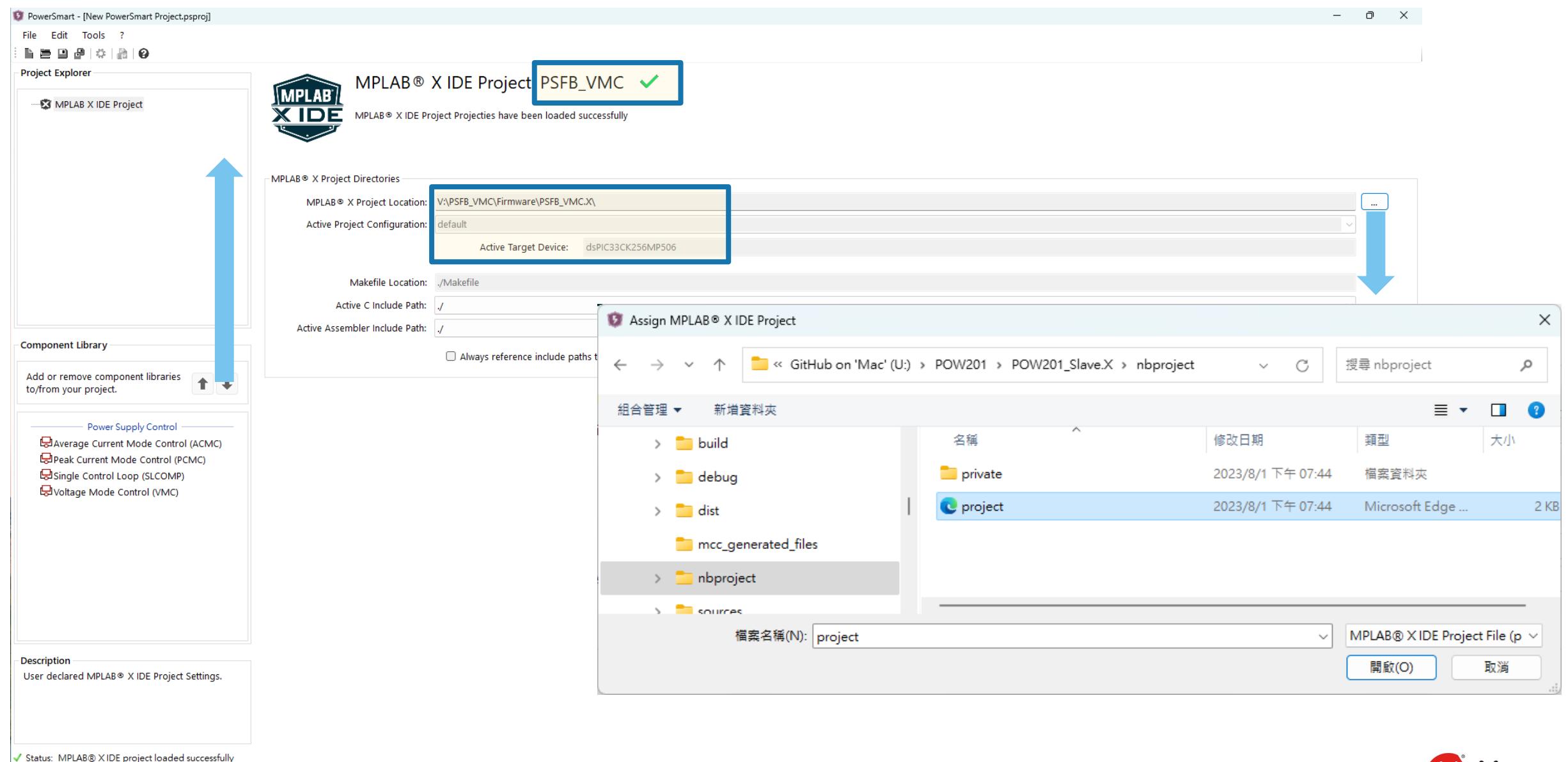


Boundary Test Between 2% ↔ 99% shifted



PowerSmart™-DCLD

Add Control Mode to The Project



Switch to DCLD Window

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

Voltage Mode Controller

REF → Σ → VCOMP (empty) → PWM → CLK → ADC → VREF

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

PowerSmart Digital Control Library Designer v1.9.15.709 - [New PowerSmart Project.psproj]

File View Tools ?

File & Function Label

Name Prefix: VCOMP

Controller Source Code Configuration Advanced

Controller Selection

Controller Type: 3P3Z - Discrete Type III Compensator

Scaling Mode: 1 - Single Bit-Shift Scaling

Input Gain

Input Data Resolution: 12 Bit

Input Signal Gain: 1.000000

Normalize Input Gain

Feedback Offset Compensation

Enable Singal Rectification Control

Compensation Filter Settings

Sampling Frequency: 250k Hz

Cross-over Frequency of Pole at Origin: 650 Hz

Pole 1: 86k Hz Zero 1: 3.2k Hz

Pole 2: 100k Hz Zero 2: 4.9k Hz

Bode Plot Settings

Frequency Domain Execution Time Block Diagram Source Code Output Info

Frequency: 0 Hz Magnitude: 0 dB Phase: 0 ° Phase Erosion 0

Magnitude/Gain

Min: -60 dB Max: 60 dB Div: 10 dB

Phase

Min: -180 ° Max: 180 ° Div: 30 °

Options

Unwrap Phase Show s-Domain

Bode Plot

Magnitude/Gain [dB] vs Frequency [Hz]

Phase [°] vs Frequency [Hz]

Filter Coefficients

| Coefficient | Float | Bsft-Scaler | Scaled Float | Fractional |
|----------------|--------------------|-------------|--------------------|--------------------|
| A-Coefficients | | | | |
| A1 | 0.847485890463909 | -1 | 0.423742945231954 | 0.423767089843758 |
| A2 | 0.148102851647187 | -1 | 0.074051425823594 | 0.074066162109375 |
| A3 | 0.004411257888904 | -1 | 0.00220562894452 | 0.002227783203125 |
| B-Coefficients | | | | |
| B0 | 1.053824682195310 | -1 | 0.526916503906250 | 0.526916503906250 |
| B1 | -0.850096151918899 | -1 | -0.425048075959450 | -0.425018310546875 |
| B2 | -1.044372733568480 | -1 | -0.522186366784242 | -0.522186366784242 |
| B3 | 0.859548100545726 | -1 | 0.429774050272863 | 0.429774050272863 |

Coefficients generated successfully

Refresh Period: 31 ms Table Options ::

Config K_P (P-Term) Gain

PowerSmart Digital Control Library Designer v1.0

Controller **Source Code Configuration** **Advanced**

- Software Context Management
 - Save/Restore Shadow Registers
 - Save/Restore MAC Working Registers
 - Save/Restore Accumulators
 - Save/Restore Accumulator A
 - Save/Restore Accumulator B
 - Save/Restore DSP Core Configuration
 - Save/Restore Core Status Register

Input Gain

Input Data Resolution:

Input Signal Gain:

- Normalize Input Gain
- Feedback Offset Compensation
- Enable Singal Rectification

Compensation Filter Settings

Sampling Frequency:

Pole 1: Pole 2:

Zero 1: Zero 2:

Basic Feature Extensions

- Add DSP Core Configuration
- Add Enable/Disable Feature
 - Always read from source when disabled
 - Add Error Normalization
 - Add Automatic Placement of Primary ADC Trigger A
 - Add Automatic Placement of Secondary ADC Trigger B
- Add User Extensions
 - Start of Control Loop
 - After Reading Source
 - Before Anti-Windup
 - Before Writing to Target
 - End of Control Loop
 - Cascade Function Call
- Anti-Windup
 - Limit Control Loop Output to Positive Numbers

Anti Windup Limiter Number Range: -32768...32767
 - Clamp Control Output Maximum
 - Generate Upper Saturation Status Flag Bit
 - Clamp Control Output Minimum
 - Force Values below Minimum Threshold to Zero
 - Generate Lower Saturation Status Flag Bit

Coefficients generated successfully

Source Code Configuration

Advanced

Use P-Term Loop Controller for Plant Measurements

Nominal Feedback Level: Nominal Control Output:

Fractional: Scaler:

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

Please Note: Execution time of user functions being called during loop execution is not included in the Control Loop Time.

Nominal Feedback Level Calculator

Circuit

Input Scaling

ADC Reference: ADC Resolution: Minimum: Maximum:

Calculation

Nominal Sense Voltage: R1: R2: Amplifier Gain:

Signal Gain:

Nominal Output Level Calculator

PWM Signal

PWM Time Base

Device Type: dsPIC33C Clock Frequency: Divider: Resolution: Maximum:

Calculation

PWM Frequency: PWM Period: PWM Period Count: Effective Resolution: Nominal Duty Ratio: Signal Gain:

Nominal Control Output Calculator

Nominal Control Output

Converter Type: D - Buck/Forward Converter Winding Ratio (P/S): Nominal Input Voltage: Nominal Output Voltage: Nominal Efficiency:

Nominal Duty Ratio:

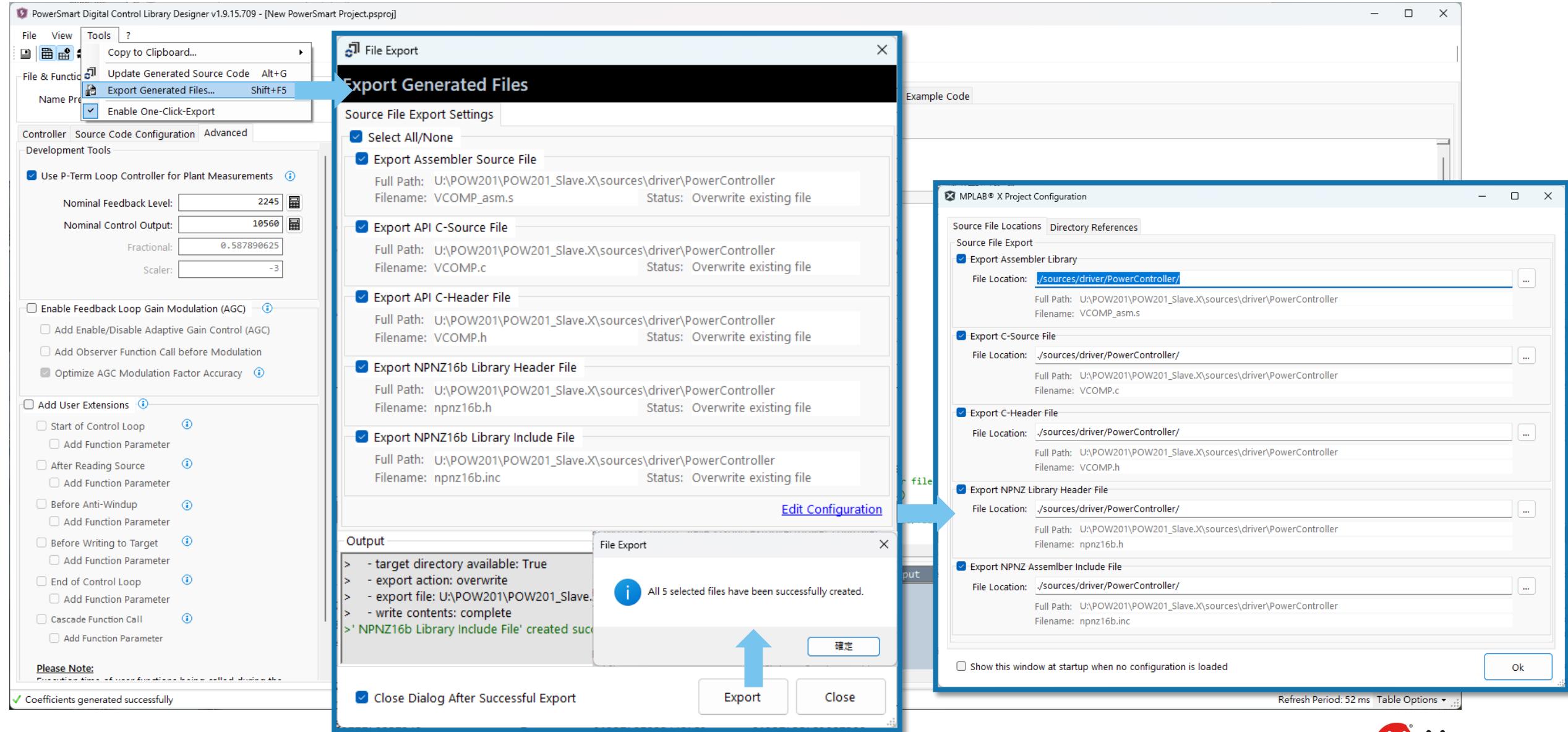
Setting for C-include Directory

The screenshot shows the PowerSmart Digital Control Library Designer interface. On the left, there's a sidebar with settings for a controller, including a 'Name Prefix' field set to 'VCOMP'. Below it are sections for 'Nominal Feedback Level' (2245), 'Nominal Control Output' (10560), 'Fractional' (0.587890625), and 'Scaler' (-3). There are also checkboxes for AGC-related options like 'Enable Feedback Loop Gain Modulation (AGC)' and 'Optimize AGC Modulation Factor Accuracy'. The main area consists of three stacked tabs under 'Source Code Output':

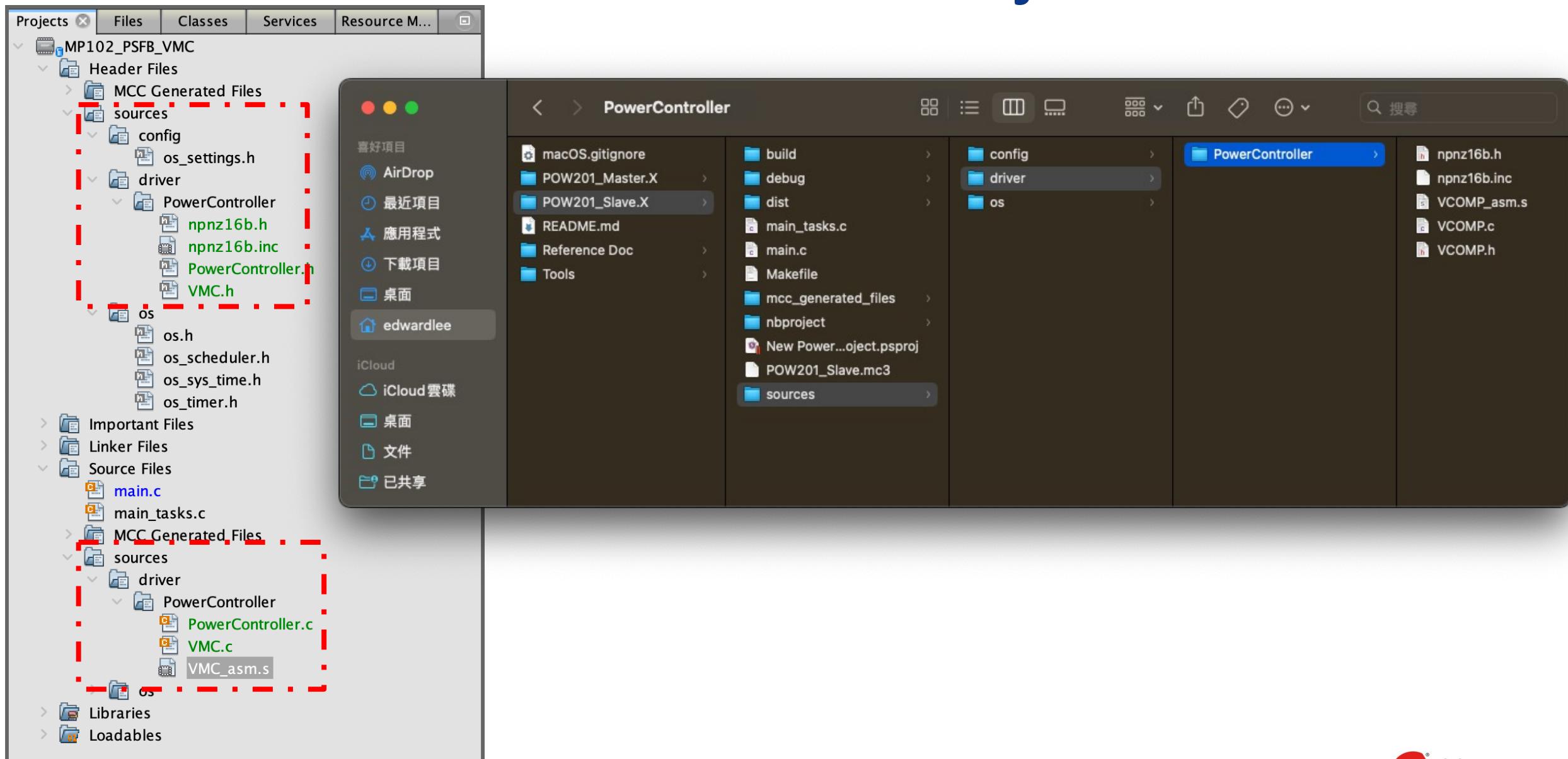
- Assembler Source File:** Contains assembly code with comments about the SDK and CGS versions, and a note about coefficient scaling mode.
- API C-Source File:** Contains C code for a data section, including a comment about placing constant data in the data section.
- API C-Header File:** Contains C code for a header section, including a comment about placing constant data in the data section.

In all three tabs, there is a red-bordered checkbox labeled 'Reference #include path to selected C include directory'.

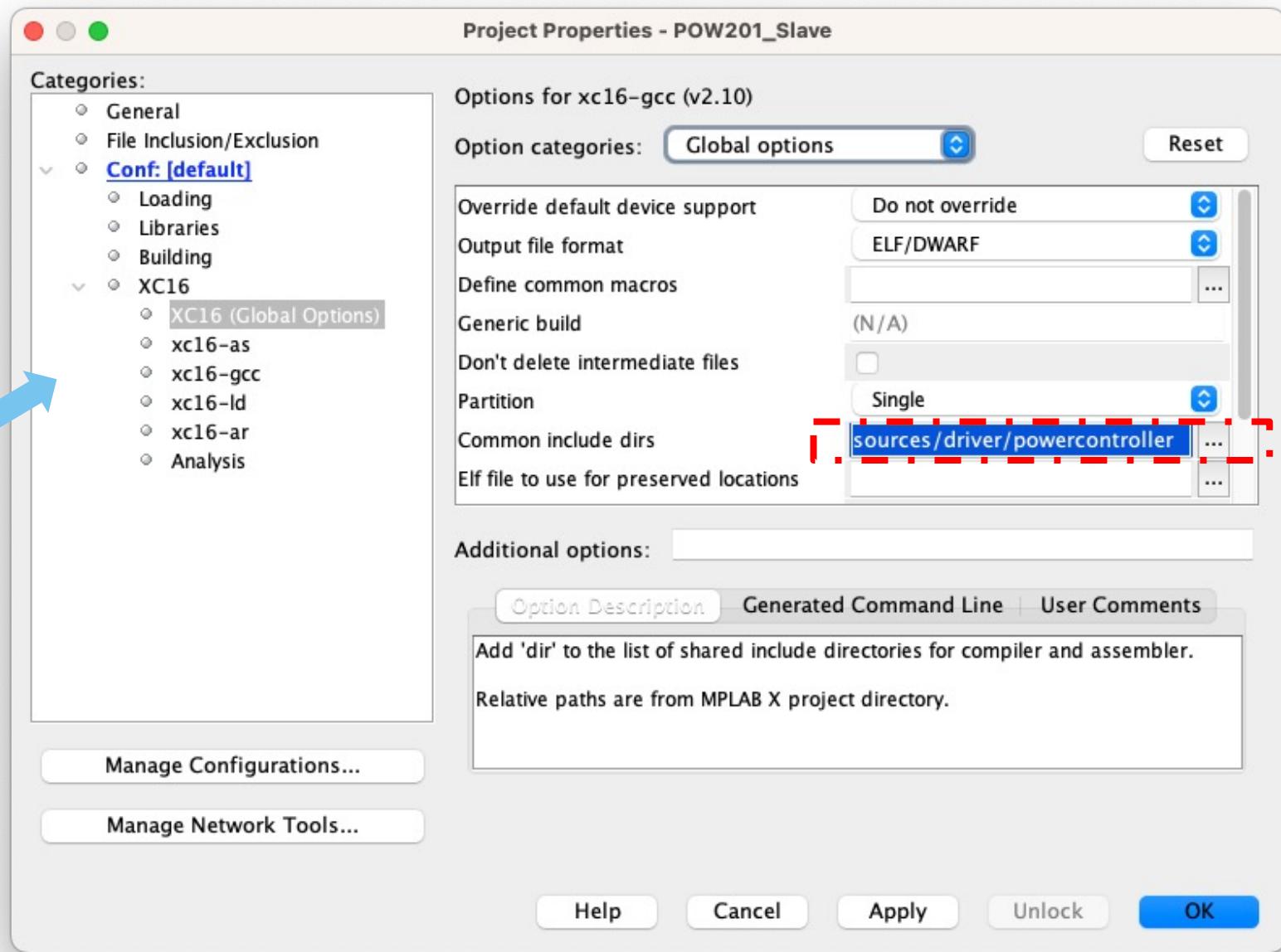
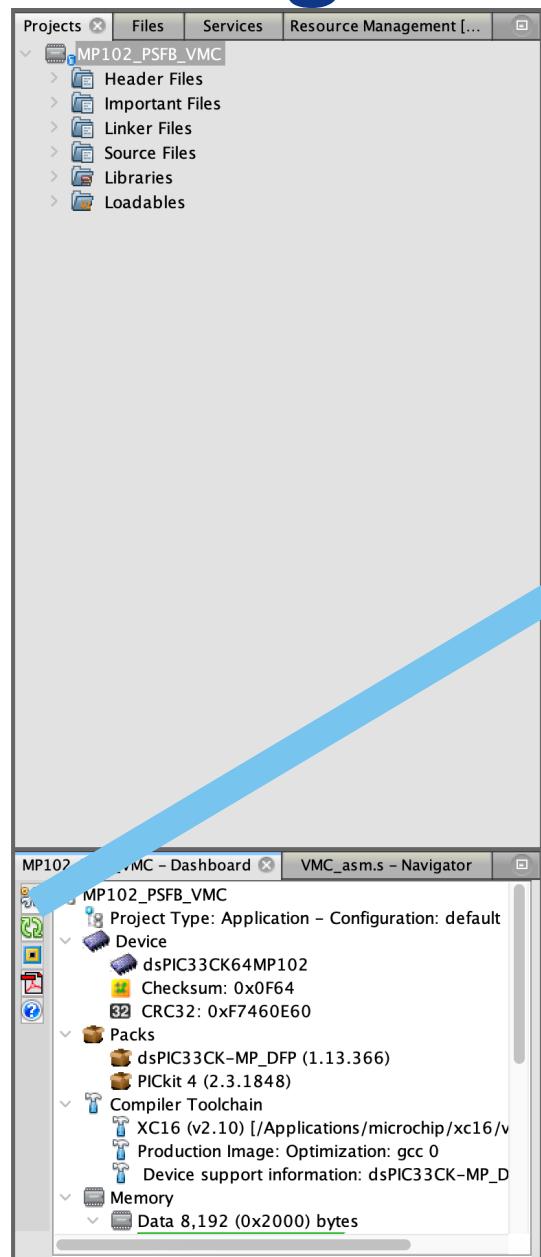
Export Compensator Sources Codes



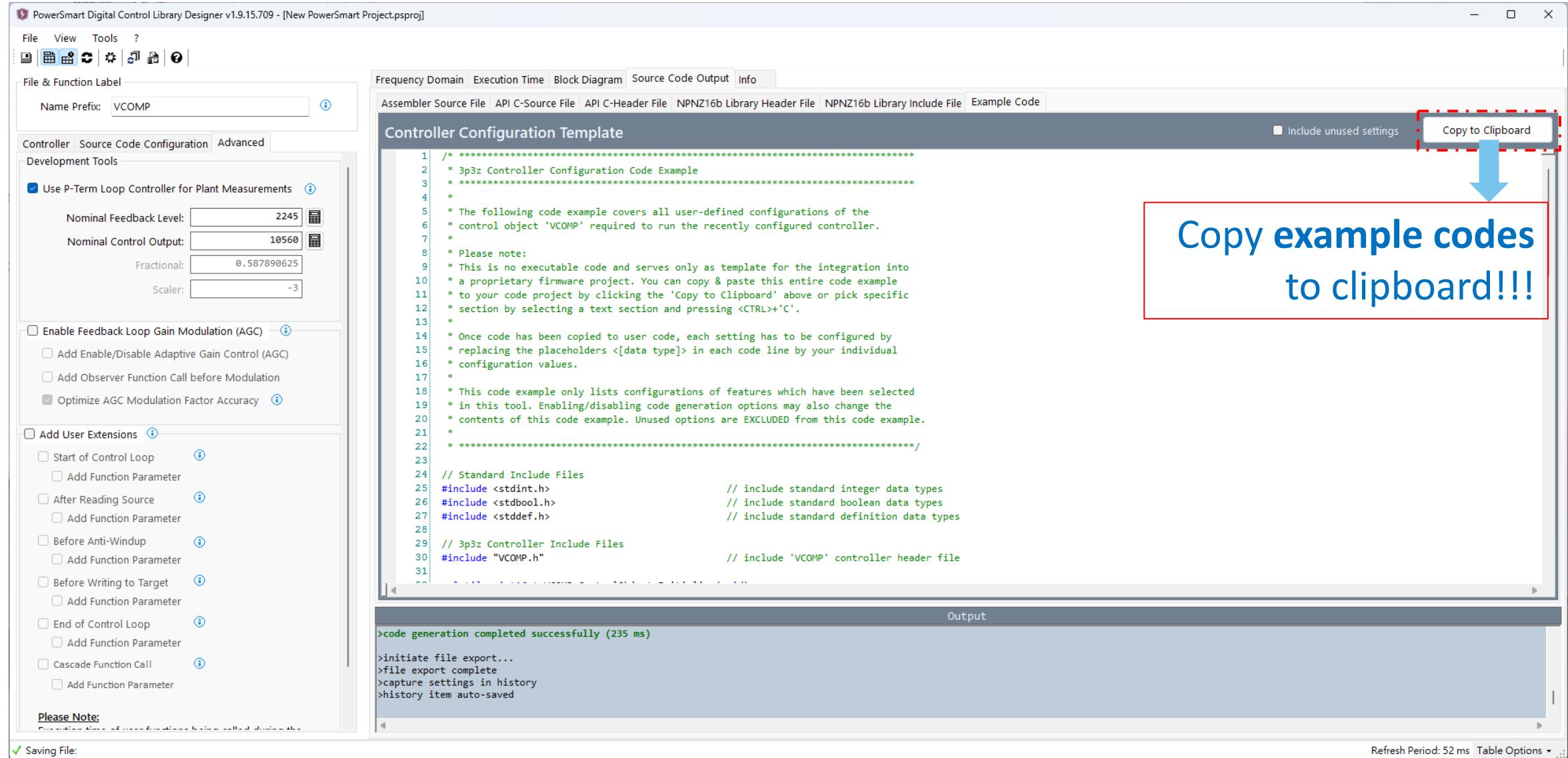
Add Control Libraries to the Project



Setting for C-include Directory



Copy example codes



New file with pasting example codes!

The screenshot shows the MPLAB X IDE interface with two code editors open. The left code editor displays `PowerController.h` and the right code editor displays `main.c`. Both files are annotated with red dashed boxes highlighting specific code snippets. A large blue arrow with the word "Pasting" written on it points from the left code editor towards the right code editor, indicating the process of copying and pasting code between them.

`PowerController.h` content:

```
1 #ifndef __PowerController
2 #define EXTERN
3 #else
4 #define EXTERN extern
5 #endif
6
7 #define SysSwitch_OpenLoop 1 // 0:Disable Open-loop control 1:Enable Open-loop control
8
9 // PWM Module
10 #define SysConst_PWMPeriod 39992 //100KHz based on 500MHz
11 #define SysConst_Shift99p (SysConst_PWMPeriod /2 * 0.99)
12 #define SysConst_Shift30p (SysConst_PWMPeriod /2 * 0.30)
13 #define SysConst_Shift02p (SysConst_PWMPeriod /2 * 0.02)
```

`main.c` content:

```
47 /*
48  *include "mcc_generated_files/system.h"
49  *include "sources/os/os.h"
50  *include "PowerController.h"
51 */
52 /*
53  * Main application
54 */
55 int main(void)
56 {
57     // initialize VCOMP controller
58     VMC_ControlObject_Initialize();
59
60     // initialize the device
61     SYSTEM_Initialize();
62
63     // OS
64     OS_Init();
65     OS_Scheduler_RunForever();
66
67     while (1)
68     {
69         // Add your application code
70     }
71     return 1;
72 }
```

`PowerController.c` content:

```
96     * pre-compiler directive declaration to your code, like the following example:
97     *
98     *     #define _VCOMP_Interrupt    _PWM1Interrupt // Define label for interrupt service routine
99     *     #define _VCOMP_ISRIF        _PWM1IF           // Define label for interrupt flag bit
100
101
102
103 void __attribute__ ( ( __interrupt__ , auto_psv ) ) _ADCAN0Interrupt ( void )
104 {
105     #if (SysSwitch_OpenLoop == 1) ←
106         //Read the ADC value from the ADCBUF
107         uint16_t valchannel_AN0;
108         valchannel_AN0 = ADCBUF0;
109         // For PWM Boundary Jumping Test
110         if(PWMUpdateCount <= 2)
111         {
112             //0~2
113             PhaseShifted = SysConst_Shift99p;
114             SRdelay = 0x320;
115         }
116         else
117         {
118             //3~5
119             PhaseShifted = SysConst_Shift02p;
120             SRdelay = 0x320;
121         }
122         PG1TRIGA = PhaseShifted;
123         //PG2TRIGA = SRdelay;
124         if(PWMUpdateCount >= 5) PWMUpdateCount = 0;
125         else PWMUpdateCount++;
126     #else
127         //Read the ADC value from the ADCBUF
128         valchannel_AN0 = ADCBUF0;
129     #if (SysSwitch_OpenLoop == 1)
130         // For PWM Boundary Test
131         if(PWMUpdateCount <= 2)
132         {
```

Dashboard information:

- Project Type: Application - Configuration: default
- Device: dsPIC33CK64MP102
- CRC32: 0xF64E60
- Packs: dsPIC33CK-MP_DFP (1.13.366), PICkit 4 (2.3.1848)
- Compiler Toolchain: XC16 (v2.10) [/Applications/microchip/xc16/v2.10]
- Memory: Data 8,192 (0x2000) bytes

Setting Initial Registers

The screenshot shows two code editors side-by-side. The left editor displays the file `PowerController.c`, and the right editor displays the file `Pow.h`. Both files are part of a project named `PowerController`.

PowerController.c (Left Editor):

```
37 volatile uint16_t VMC_ControlObject_Initialize(void)
38 {
39     volatile uint16_t retval = 0; // Auxiliary variable for function
40     Output_Vref = 0;
41     PhaseShifted = SysConst_Shift02p;
42     SRdelay = 0x320;
43     /* Controller Input and Output Ports Configuration */
44
45     // Configure Controller Primary Input Port
46     VMC.Ports.Source.ptrAddress = &ADCBUF0; // Pointer to primary feedback source (ADCBUF0)
47     VMC.Ports.Source.Offset = 0; // Primary feedback signal offset
48     VMC.Ports.Source.NormScaler = 0; // Primary feedback normalization factor
49     VMC.Ports.Source.NormFactor = 0x7FFF; // Primary feedback normalization factor
50
51     // Configure Controller Primary Output Port
52     VMC.Ports.Target.ptrAddress = &PhaseShifted; // Pointer to primary output target
53     VMC.Ports.Target.Offset = 0; // Primary output offset value
54     VMC.Ports.Target.NormScaler = 0; // Primary output normalization factor
55     VMC.Ports.Target.NormFactor = 0x7FFF; // Primary output normalization factor
56
57     // Configure Control Reference Port
58     VMC.Ports.ptrControlReference = &Output_Vref; // Pointer to control reference (Output_Vref)
59
60     /* Controller Output Limits Configuration */
61
62     // Primary Control Output Limit Configuration
63     VMC.Limits.MinOutput = VCOMP_MIN_CLAMP; // Minimum control output value
64     VMC.Limits.MaxOutput = VCOMP_MAX_CLAMP; // Maximum control output value
65
66     /* Advanced Parameter Configuration */
67
68     // Initialize User Data Space Buffer Variables
69     VMC.Advanced usrParam0 = 0; // No additional advanced control opt
70     VMC.Advanced usrParam1 = 0; // No additional advanced control opt
71     VMC.Advanced usrParam2 = 0; // No additional advanced control opt
72     VMC.Advanced usrParam3 = 0; // No additional advanced control opt
73     VMC.Advanced usrParam4 = 0; // No additional advanced control opt
74     VMC.Advanced usrParam5 = 0; // No additional advanced control opt
75     VMC.Advanced usrParam6 = 0; // No additional advanced control opt
76     VMC.Advanced usrParam7 = 0; // No additional advanced control opt
77
78     /* Controller Status Word Configuration */
79
80     VMC.status.bits.enabled = false; // Keep controller disabled
81
82     // Call Assembler Control Library Initialization Function
83     retval = VMC_Initialize(&VMC); // Initialize controller data arrays
84
85     return(retval);
86 }
```

Pow.h (Right Editor):

```
1 #ifndef __PowerController
2 #define EXTERN
3 #else
4 #define EXTERN extern
5 #endif
6
7 #define SysSwitch_OpenLoop 1 // 0:Disable Open-loop control
8
9 // PWM Module
10 #define SysConst_PWMPeriod 39992 //100KHz based on 500MHz
11 #define SysConst_Shift99p (SysConst_PWMPeriod /2 * 0.99)
12 #define SysConst_Shift30p (SysConst_PWMPeriod /2 * 0.30)
13 #define SysConst_Shift02p (SysConst_PWMPeriod /2 * 0.02)
14
15 // Reference for Voltage Loop Compensator
16 #define VCOMP_VREF 3023 //
17
18 // Compensator Clamp Limits
19 #define VCOMP_MIN_CLAMP SysConst_Shift02p
20 #define VCOMP_MAX_CLAMP SysConst_Shift99p
21
22
23 EXTERN uint16_t PWMUpdateCount;
24 EXTERN uint16_t PhaseShifted;
25 EXTERN uint16_t SRdelay;
26 EXTERN uint16_t Output_Vref;
27
28
29 volatile uint16_t VMC_ControlObject_Initialize(void);
30 void VMC_Softstart(void);
31
32
```

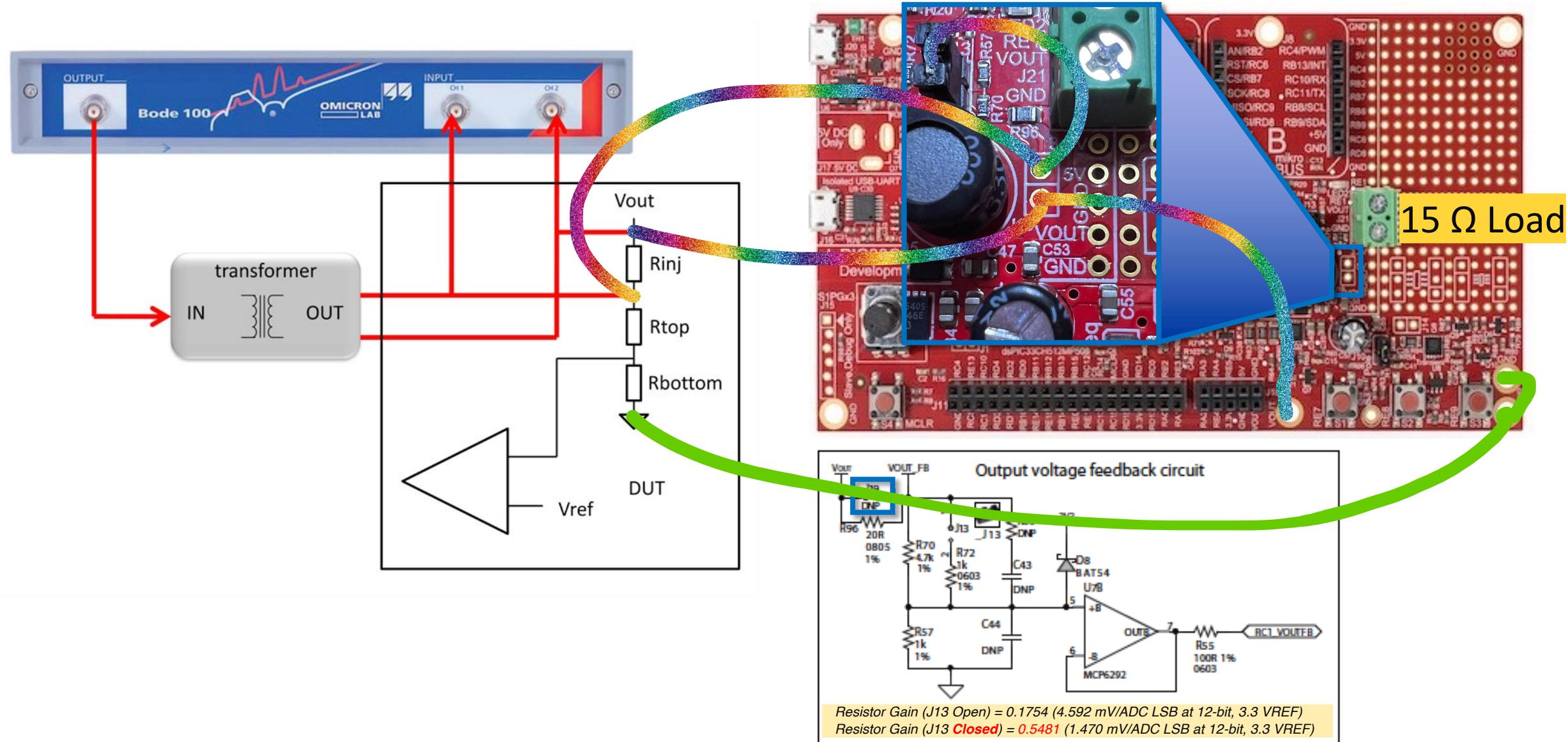
Control Loop

The screenshot shows a software development environment with a window titled "PowerController.c". The window has tabs for "Source" and "History". The "Source" tab is active, displaying C code for a control loop. The code is annotated with comments and includes several preprocessor directives (#define, #if, #else, #endif). A blue rectangular box highlights a section of the code starting with "#else" and ending with "#endif". The code is as follows:

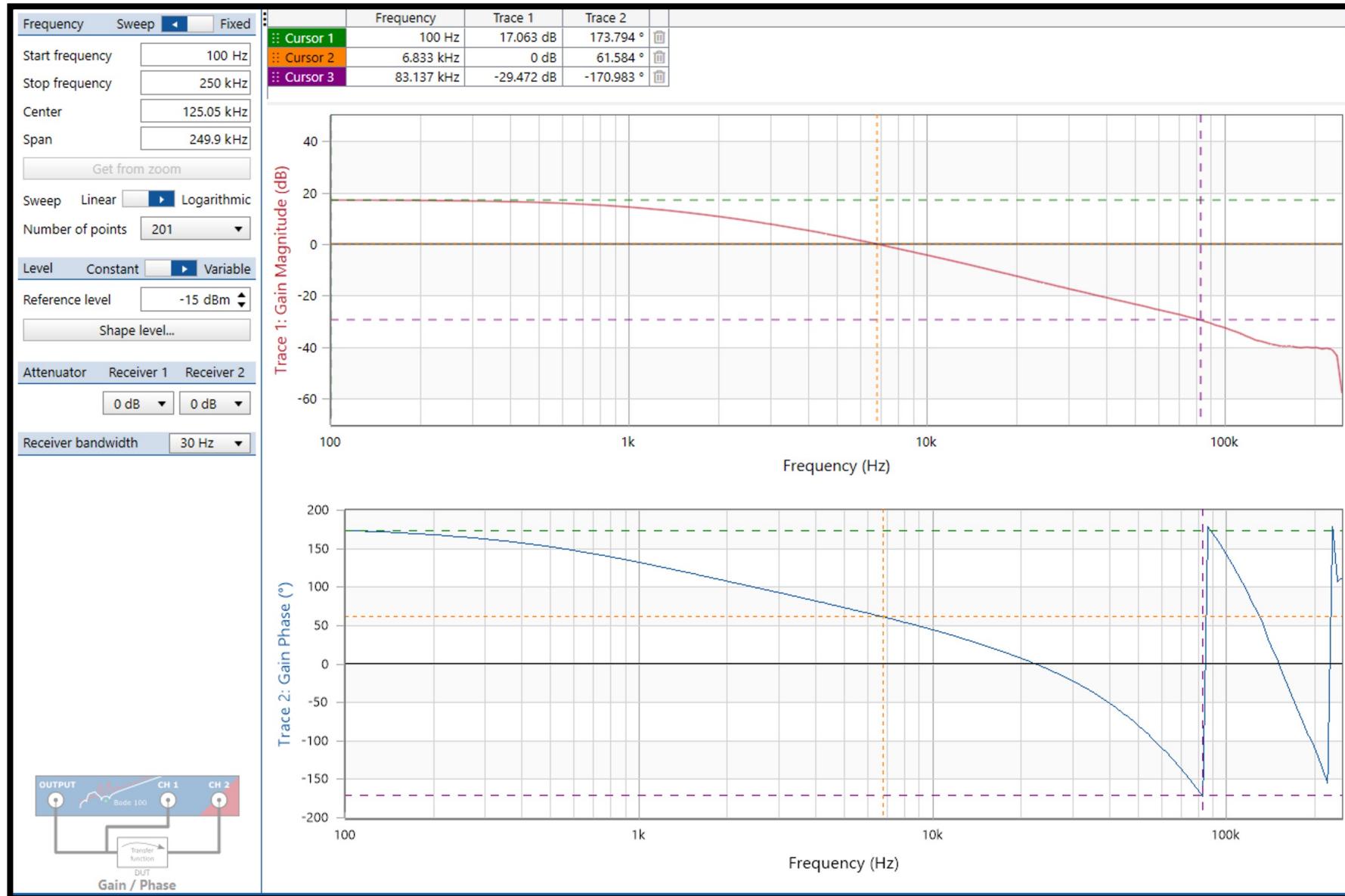
```
87  /* 3p3z Control Loop Interrupt Service Routine for Controller Object 'VCOMP'
88  ****
89  * This code example of a interrupt service routine uses the tailored name label '_VC
90  * The Assembler library code sequences of controller data objects generated by PS-DC
91  * for being called by a PWM interrupt for minimum response time. However, in some ap
92  * it might be desired to call the control loop from other interrupt sources.
93  * Using custom labels for interrupt routines allows using generic interrupt service
94  * function calls in code, which can be mapped to specific interrupt sources by addin
95  * pre-compiler directive declaration to your code, like the following example:
96  *
97  *     #define _VCOMP_Interrupt      _PWM1Interrupt // Define label for interrupt servi
98  *     #define _VCOMP_ISRIF         _PWM1IF        // Define label for interrupt flag
99  *
100 ****
101
102 void __attribute__ ( ( __interrupt__ , auto_psv ) ) _ADCAN0Interrupt ( void )
103 {
104 #if (SysSwitch_OpenLoop == 1)
105     //Read the ADC value from the ADCBUF
106     uint16_t valchannel_AN0;
107     valchannel_AN0 = ADCBUF0;
108     // For PWM Boundary Jumping Test
109     if(PWMUpdateCount <= 2)
110     {
111         //0~2
112         PhaseShifted = SysConst_Shift99p;
113         SRdelay = 0x320;
114     }
115     else
116     {
117         //3~5
118         PhaseShifted = SysConst_Shift02p;
119         SRdelay = 0x320;
120     }
121     PG1TRIGA = PhaseShifted;
122     //PG2TRIGA = SRdelay;
123     if(PWMUpdateCount >= 5) PWMUpdateCount = 0;
124     else PWMUpdateCount++;
125
126 #else
127     //VMC_Update(&VCOMP);           // Call control loop
128     VMC_PTermUpdate(&VMC);        // Call P-Term control loop
129 #endif
130
131     //clear the channel_AN0 interrupt flag
132     IFS5bits.ADCAN0IF = 0;
133 }
134
135
136
```

Soft-start

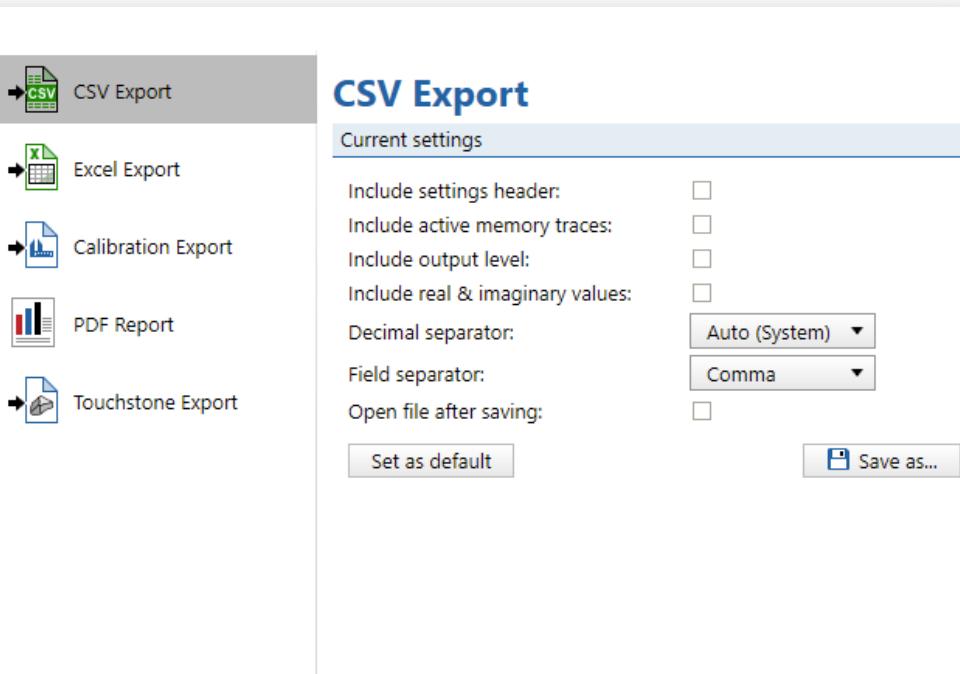
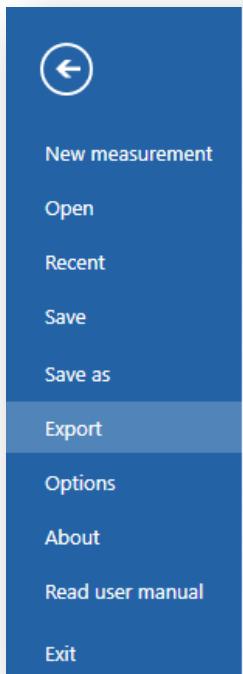
Plant Measurement Using Bode-100



Plant Measurement Using PowerSmart™



Export Bode Plot to CSV



| | A | B | C | D |
|----|----------------|---------------|--------------------------|---|
| 1 | Frequency (Hz) | Trace 1: Gain | Trace 2: Gain: Phase (°) | |
| 2 | 100 | 17.30047273 | 169.1247537 | |
| 3 | 103.98955 | 16.99047568 | 169.574792 | |
| 4 | 108.138266 | 16.65394975 | 170.1127738 | |
| 5 | 112.452496 | 16.75379993 | 166.667777 | |
| 6 | 116.938845 | 17.34375972 | 171.3902009 | |
| 7 | 121.604179 | 16.72603985 | 166.3648023 | |
| 8 | 126.455639 | 17.30457827 | 167.1115465 | |
| 9 | 131.50065 | 17.30183334 | 167.5736927 | |
| 10 | 136.746935 | 17.15930463 | 168.4839062 | |
| 11 | 142.202523 | 17.48557814 | 167.3003502 | |
| 12 | 147.875764 | 16.78895798 | 166.5875587 | |
| 13 | 153.775342 | 16.79352515 | 167.3131347 | |
| 14 | 159.910286 | 16.84065133 | 162.3407847 | |
| 15 | 166.289987 | 16.86709837 | 161.8181398 | |
| 16 | 172.92421 | 17.16188517 | 166.2776363 | |
| 17 | 179.823108 | 16.58615684 | 163.233949 | |
| 18 | 186.997242 | 16.82301431 | 159.8557044 | |

Lab #3: Plant Measurement Using PowerSmart™

The screenshot shows the PowerSmart software interface. On the left, there is a sidebar with various icons and a 'Project Explorer' section. The main window has a title bar 'Import Transfer Function'. In the center, there is a 'Data Series Name' input field containing 'VPLANT'. Below it, there are three tabs: 'File Import' (selected), 'Simplis/MINDI', and 'Bode Plot'. Under 'File Import', there is a 'Data Source' section with a 'Filename' field set to 'U:\POW201\Tools\Bode100\POW201_Lab3_2023-08-12T18_12_16.csv', 'Series Header Row' set to '1', 'Data Start Row' set to '2', and 'Column Separator' set to '<COMMA>'. There are two green checkmarks indicating that a data file has been selected and a valid name for the new data series has been specified. Below this, there is a 'Phase Rotation' section with three radio button options: 'No Rotation' (unselected), 'Rotate by -180°' (selected), and 'Rotate by +180°'. To the right of this section is a preview window showing a portion of the CSV data:

| | 1 | 100 | 17.30047273 | 169.1247537 |
|---|------------|-------------|-------------|-------------|
| 2 | 103.98955 | 16.99047568 | 169.574792 | |
| 3 | 108.138266 | 16.65394975 | 170.1127738 | |
| 4 | 112.452496 | 16.75379993 | 166.667777 | |
| 5 | 116.938845 | 17.34375972 | 171.3902009 | |
| 6 | 121.604179 | 16.72603985 | 166.3648023 | |
| 7 | | | | |

At the bottom of the dialog, there is an 'Import Data' button with a red hand cursor icon pointing to it, and 'OK' and 'Close' buttons.

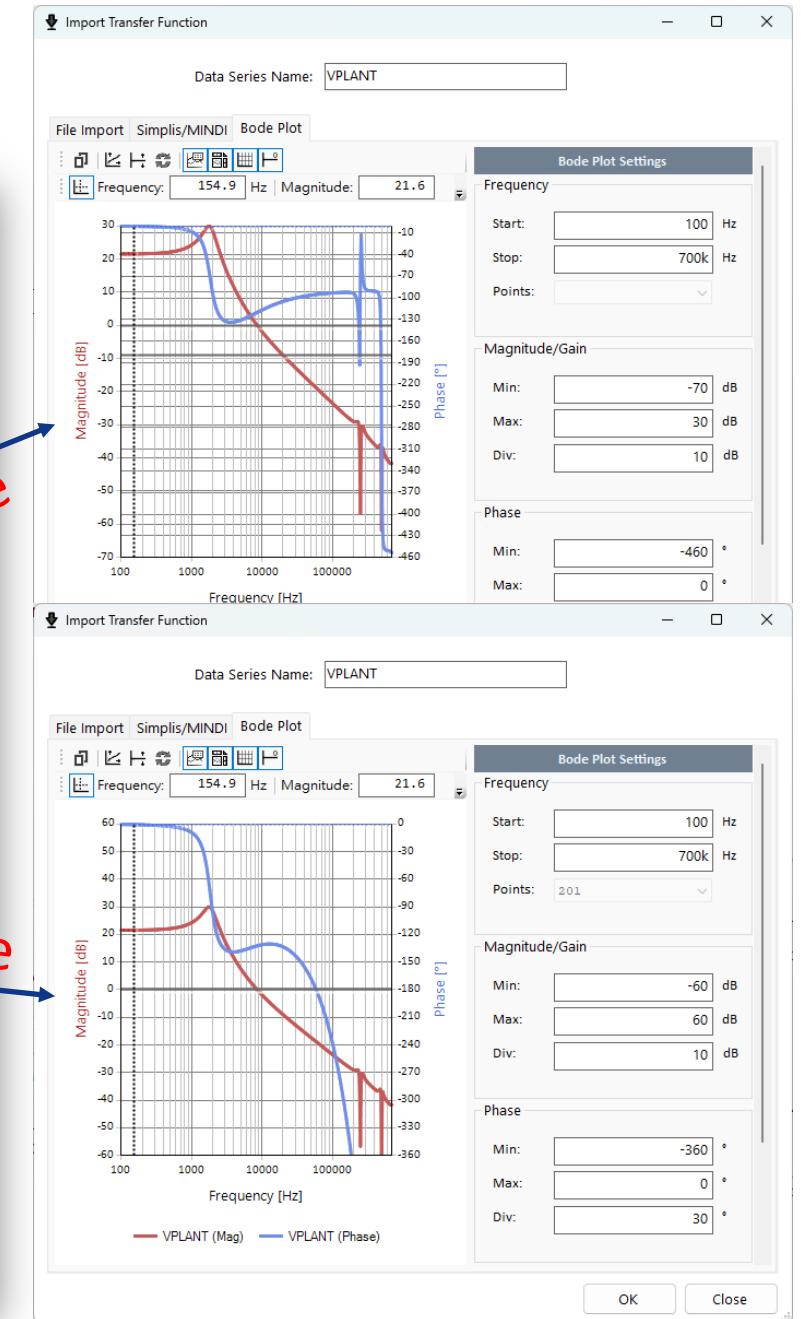
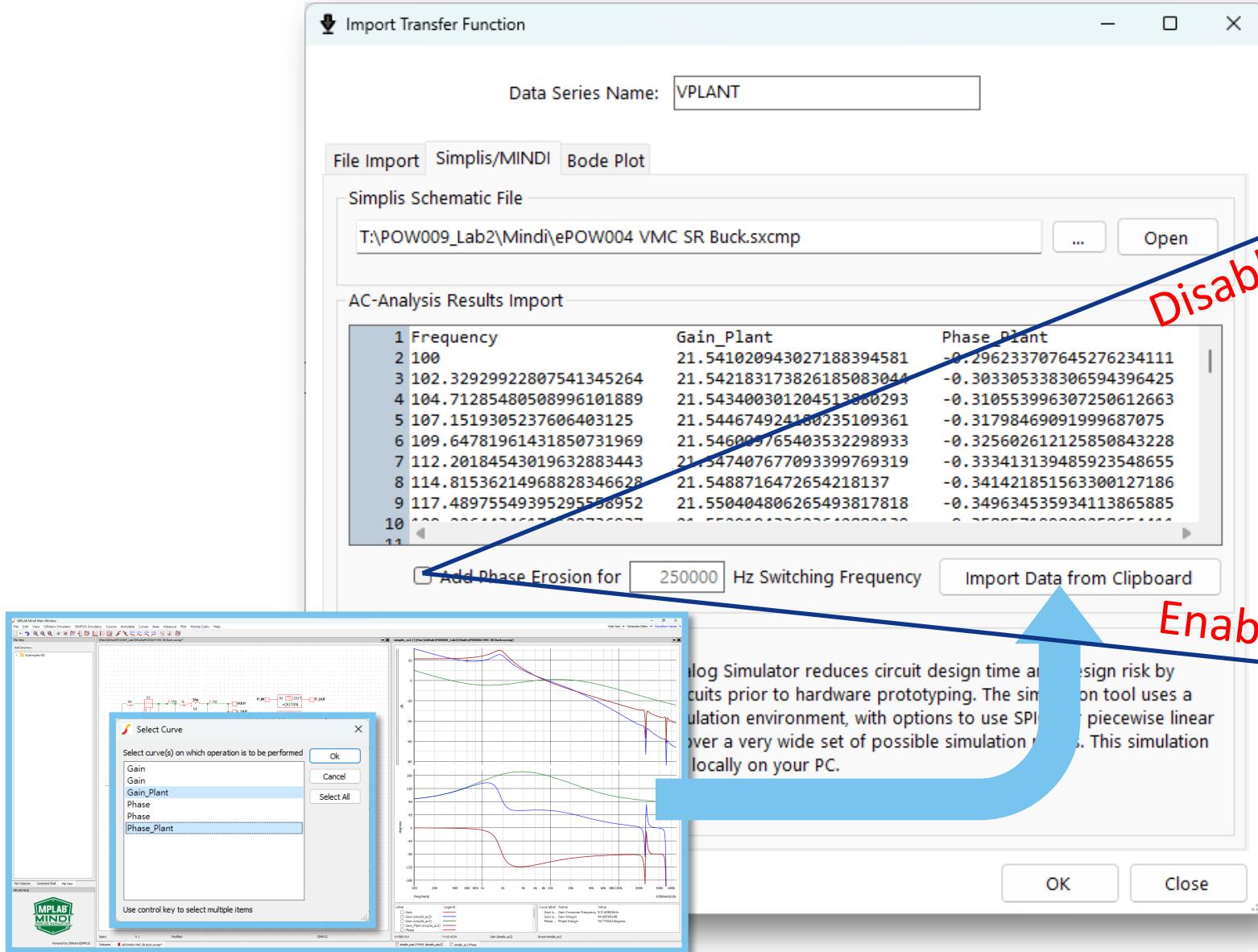
To the right of the dialog, there is a block diagram window titled 'C)'. It contains a block labeled 'Converter Voltage Plant G(s)' with 'VIN' at the top and 'VOUT' at the bottom. A blue arrow points from the 'VOUT' terminal of this block to the 'VOUT' terminal of a 'Plant Gain' block. Below the 'Plant Gain' block is a 'Tv(s)' block. The entire diagram is enclosed in a light gray frame.

At the bottom left of the dialog, there is a status message: 'Status: Microchip® XDS100 project loaded successfully'.

NOW, Actual Plant is Imported!!!



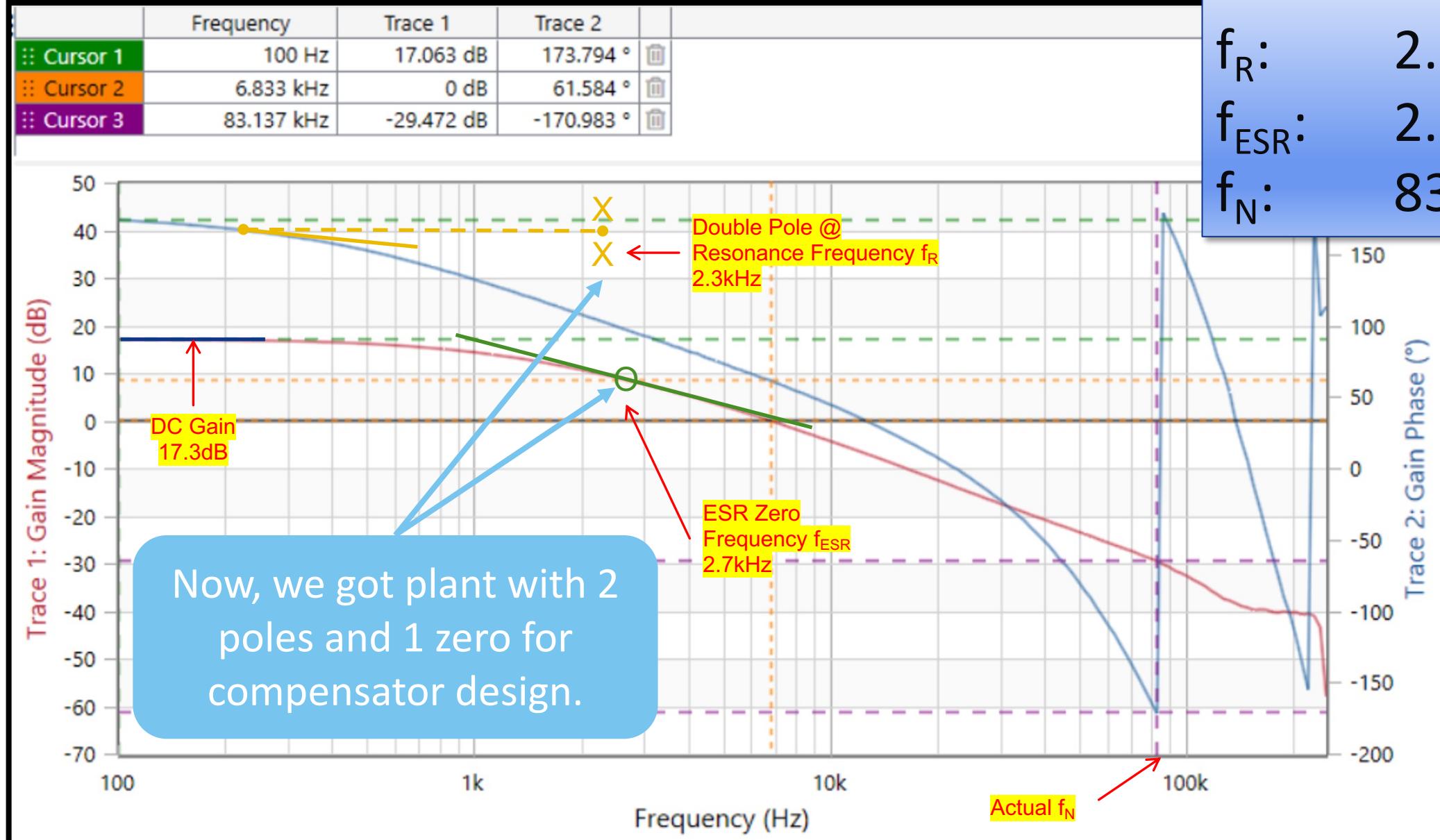
Design Tip



Closed-loop Control

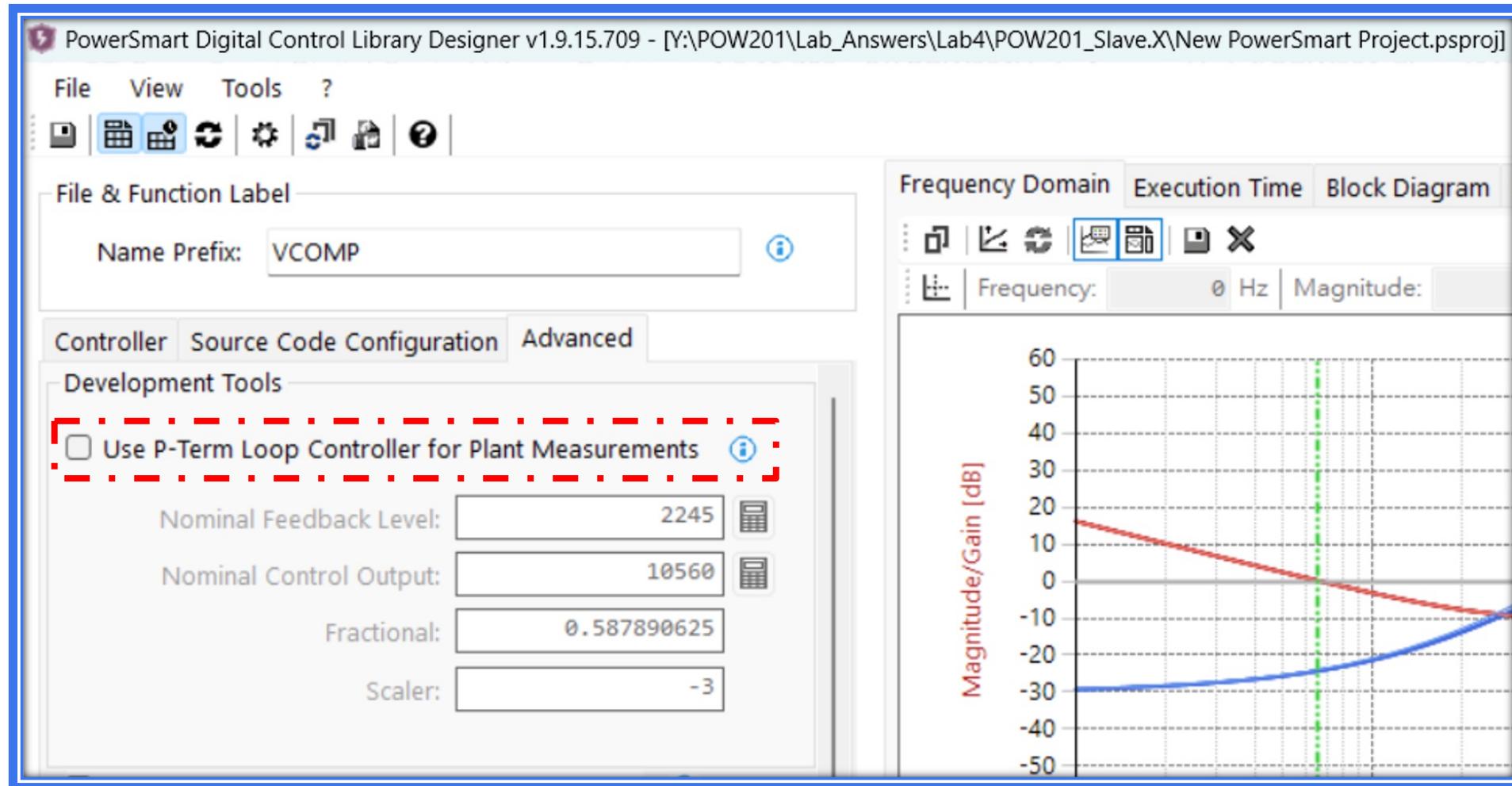
Lab4

The Actual Plant

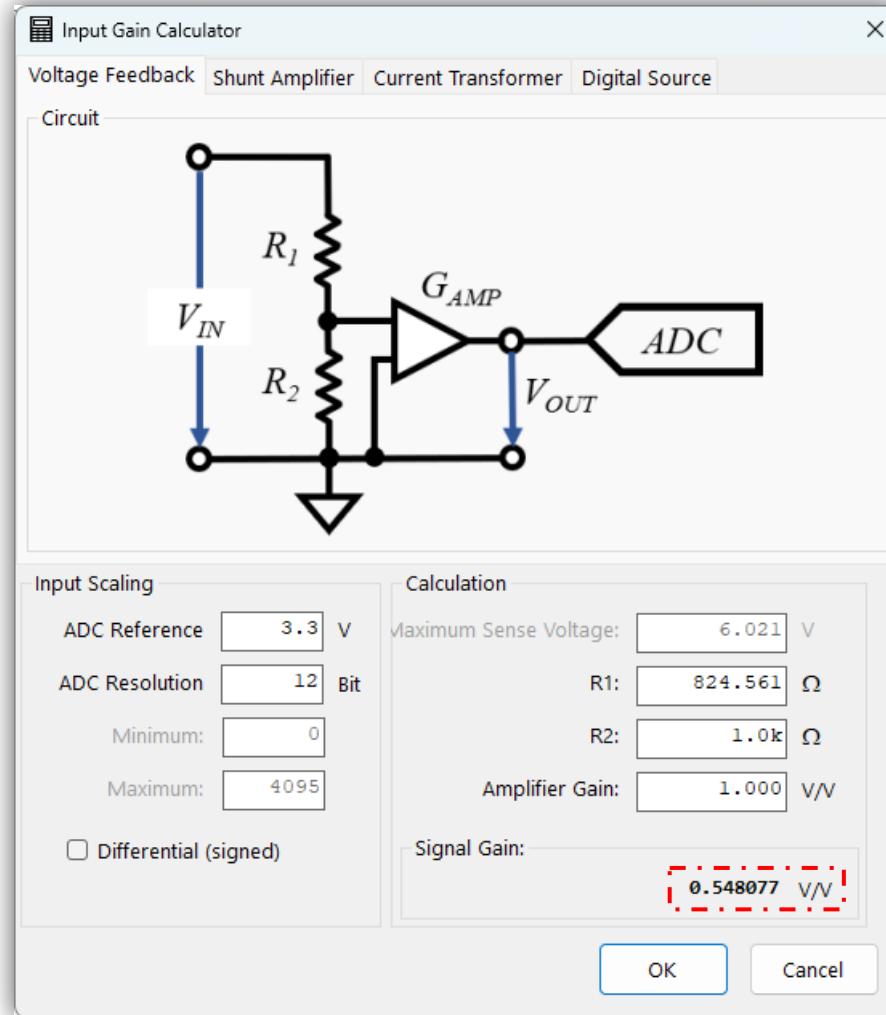
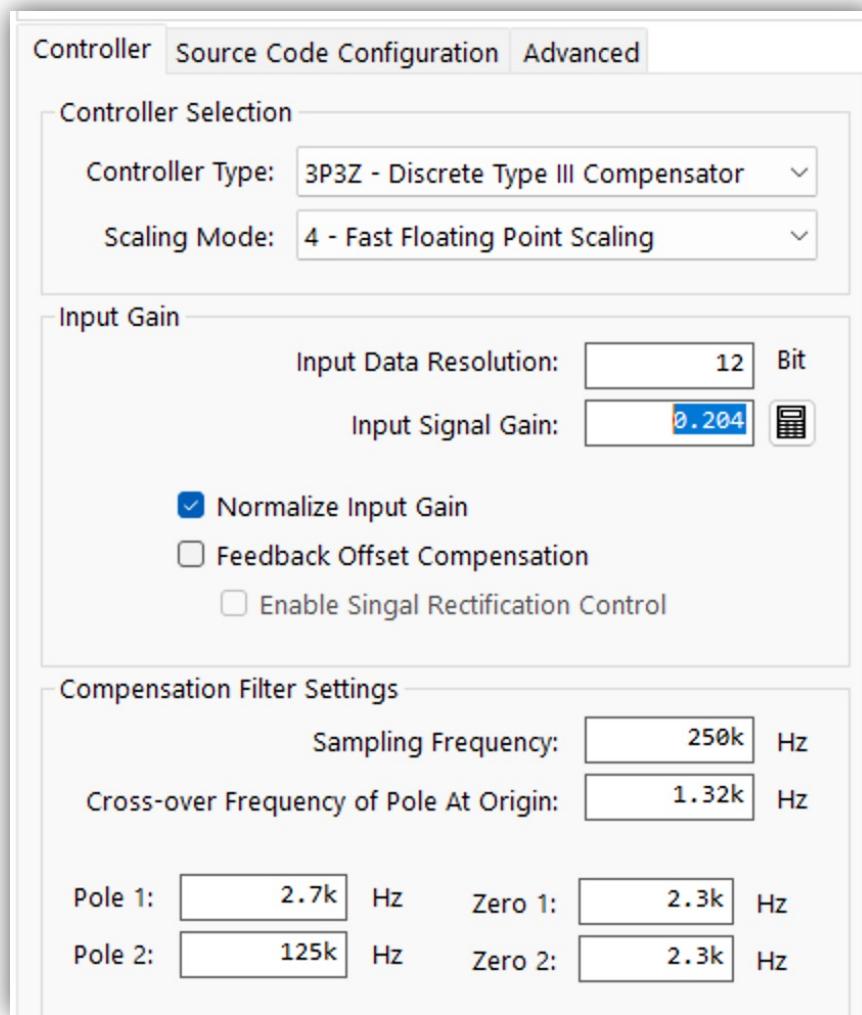


Uncheck Kp (P-Term) Control on DCLD

- **Uncheck P-Term loop controller if needed to remove the code.**



Place Poles-Zeros



$$\begin{aligned}K_{FB} &= 1 * R_1 / (R_1 + R_2) \\&= 0.4519\end{aligned}$$

$$\begin{aligned}\text{Input Signal Gain} &= K_{FB} * K_{FB} \\&= 0.204\end{aligned}$$

DCLD

Place Poles-Zeros

Controller Source Code Configuration Advanced

Controller Selection

Controller Type: 3P3Z - Discrete Type III Compensator
Scaling Mode: 4 - Fast Floating Point Scaling

Input Gain

Input Data Resolution: 12 Bit
Input Signal Gain: 0.204

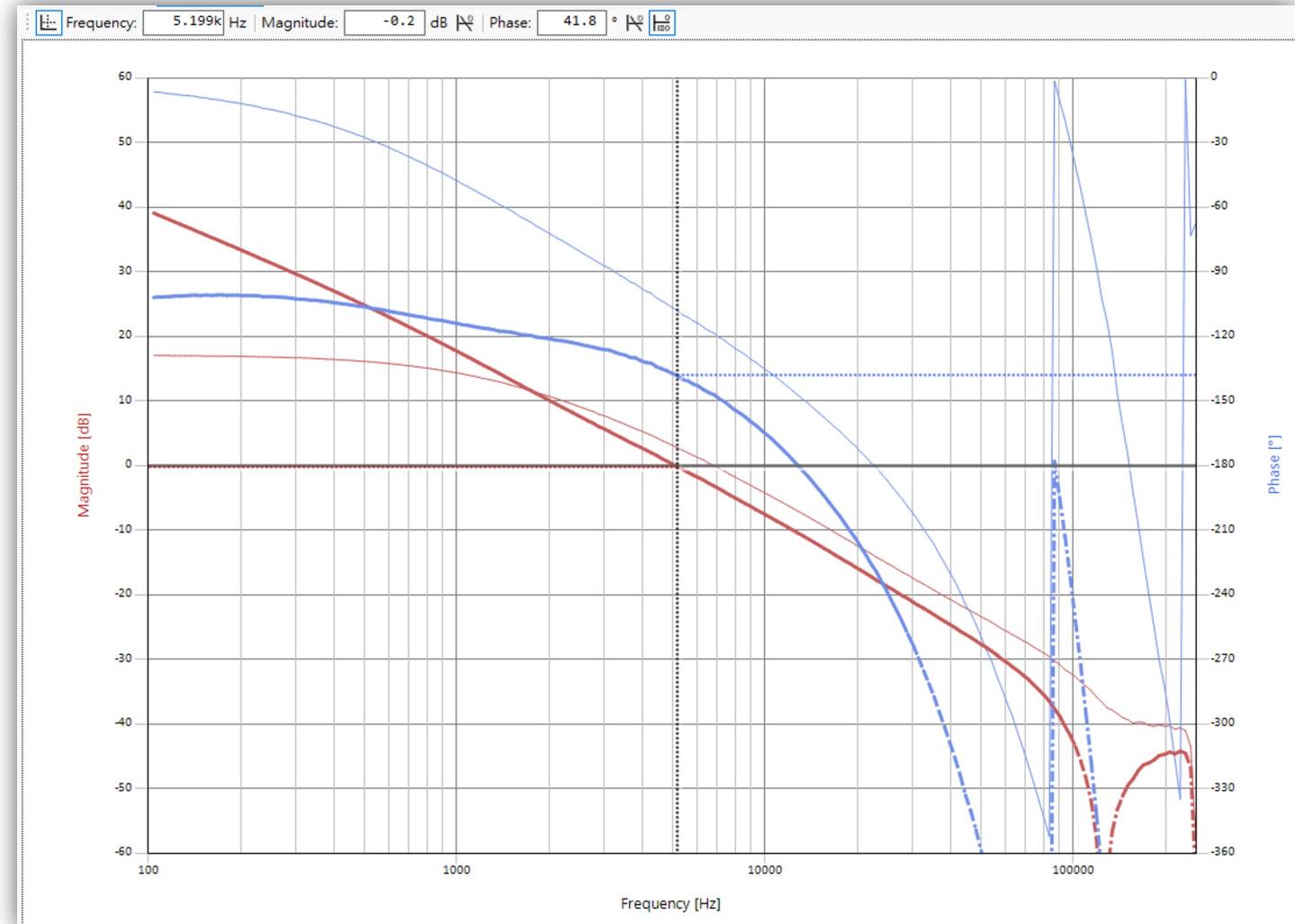
Normalize Input Gain
 Feedback Offset Compensation
 Enable Singal Rectification Control

Compensation Filter Settings

Sampling Frequency: 250k Hz
Cross-over Frequency of Pole At Origin: 1.32k Hz

Pole 1: 2.7k Hz Zero 1: 2.3k Hz
Pole 2: 125k Hz Zero 2: 2.3k Hz

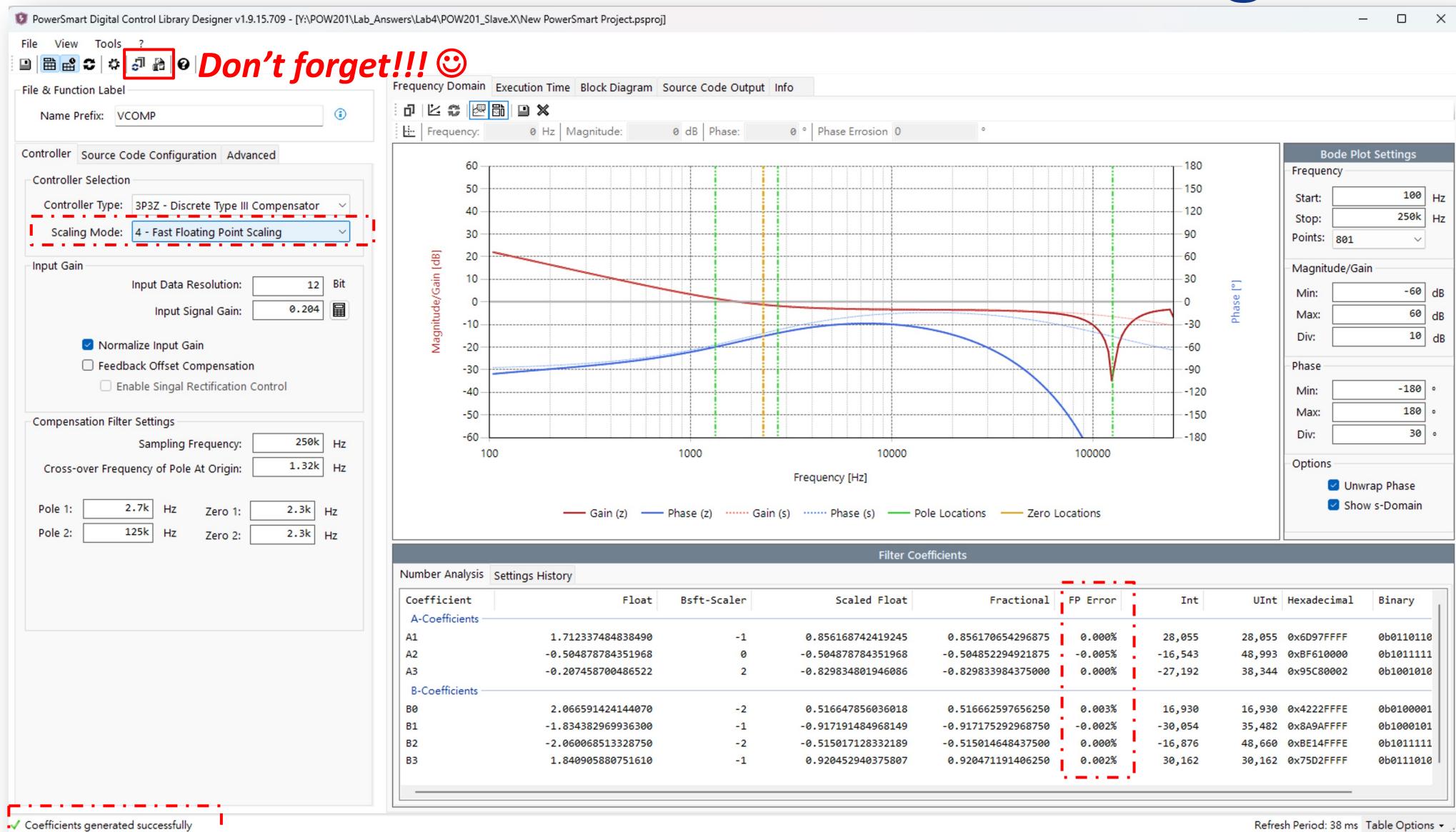
Observe Open Loop Gain



DCLD

PowerSmart™
Microchip Proprietary and Confidential

Check Coefficients with Suitable Scaling Mode



Closed-loop Control

Switch to closed-loop control in ANx ISR

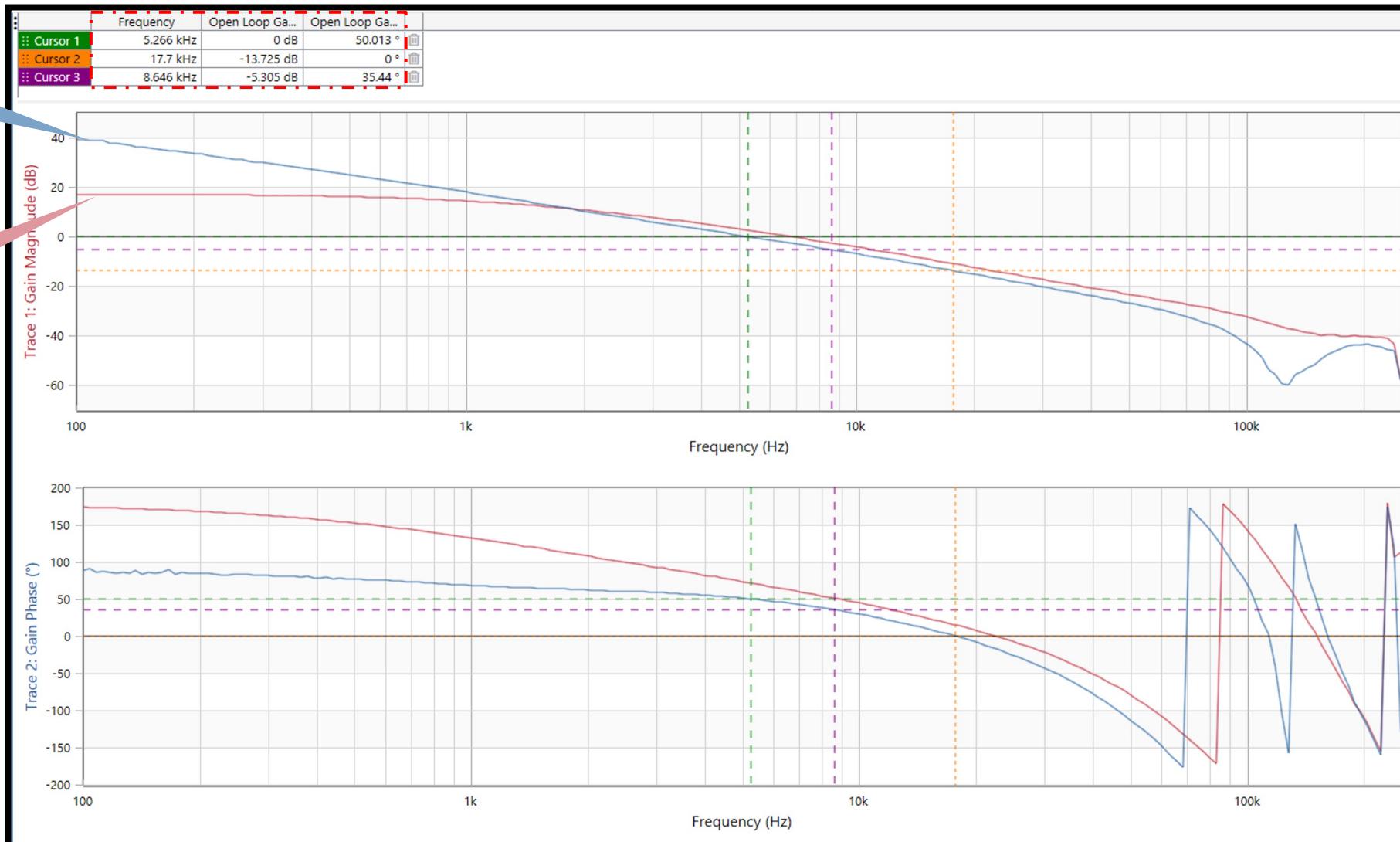
```
Projects x Files Services Classes BuckConverter.c x
Source History
96 * #define _VCOMP_INTERRUPT _PWM1Interrupt // Define label for interrupt service
97 * #define _VCOMP_ISRIF _PWM1IF // Define label for interrupt flag b
98 *
99 ****
100
101 void __attribute__ (( __interrupt__ , auto_psv )) _ADCAN1Interrupt ( void )
102 {
103     //LED2_SetHigh();
104
105     VCOMP_Update(&VCOMP); // Call control loop
106     //VCOMP_PTermUpdate(&VCOMP); // Call P-Term control loop
107
108     //LED2_SetLow();
109     IFS5bits.ADCAN1IF = 0; // Clear the interrupt flag
110 }
111
112 ****
113 // Download latest version of this tool here:
114 // https://www.microchip.com/powersmart
115 ****
116
117 // Simple Softstart
118 void Buck_Softstart(void)
119 {
120     VCOMP.status.bits.enabled = true; // Enable controller
121     if(Buck_Vref < VCOMP_VREF)
122     {
123         Buck_Vref+=10;
124     }
125 }
```

Closed-loop Control

Bode Plot Measurement

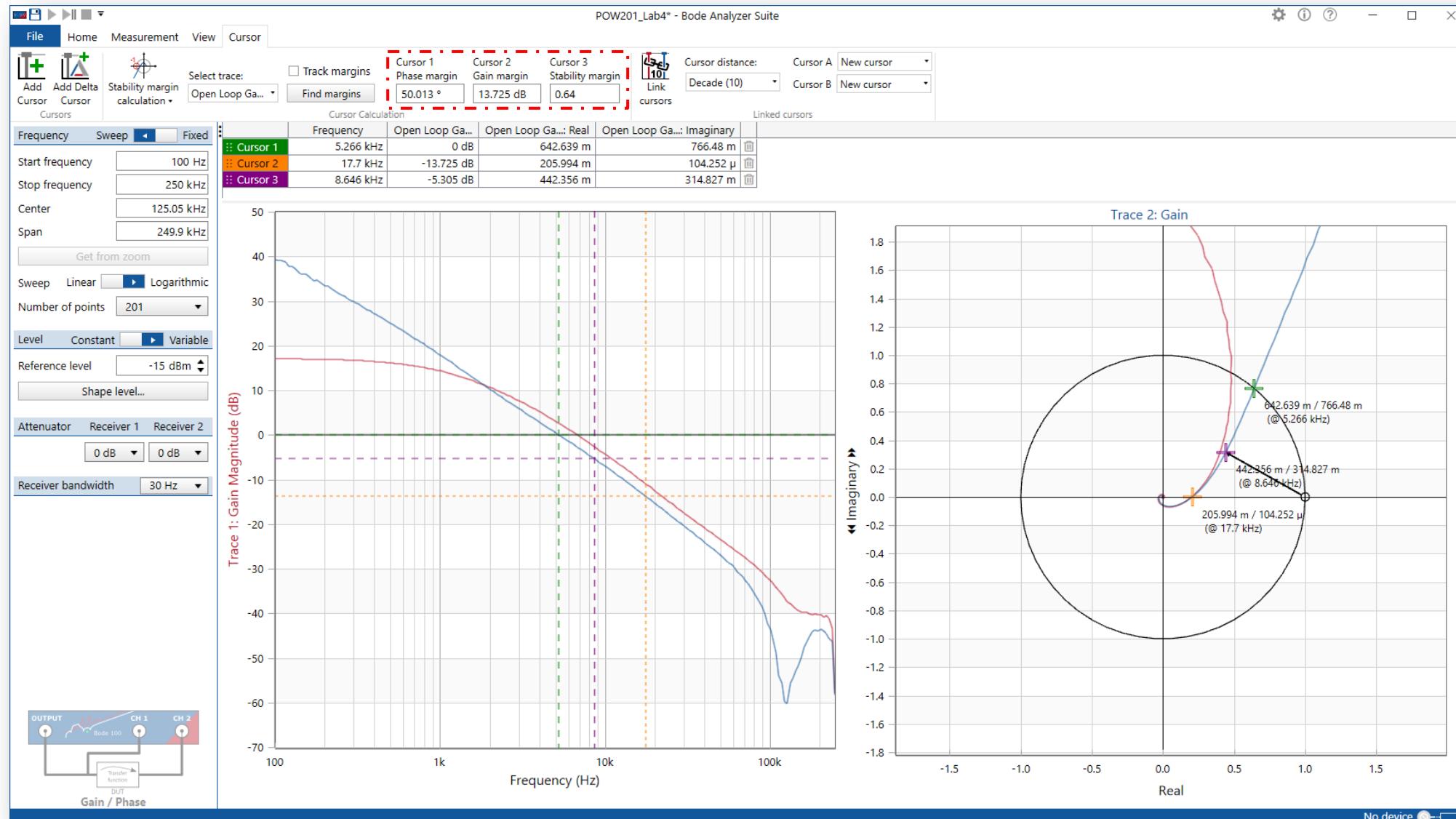
OL TF

Plant



Closed-loop Control

Stability Margin & Nyquist Plot





May The *Power* Be With You

**KNOWLEDGE IS
POWER**

Massive power density in the smallest packages

Thanks!

