



PSFB Converter Code Examples

dsPIC33CK64MP102



A Leading Provider of Smart, Connected and Secure Embedded Solutions

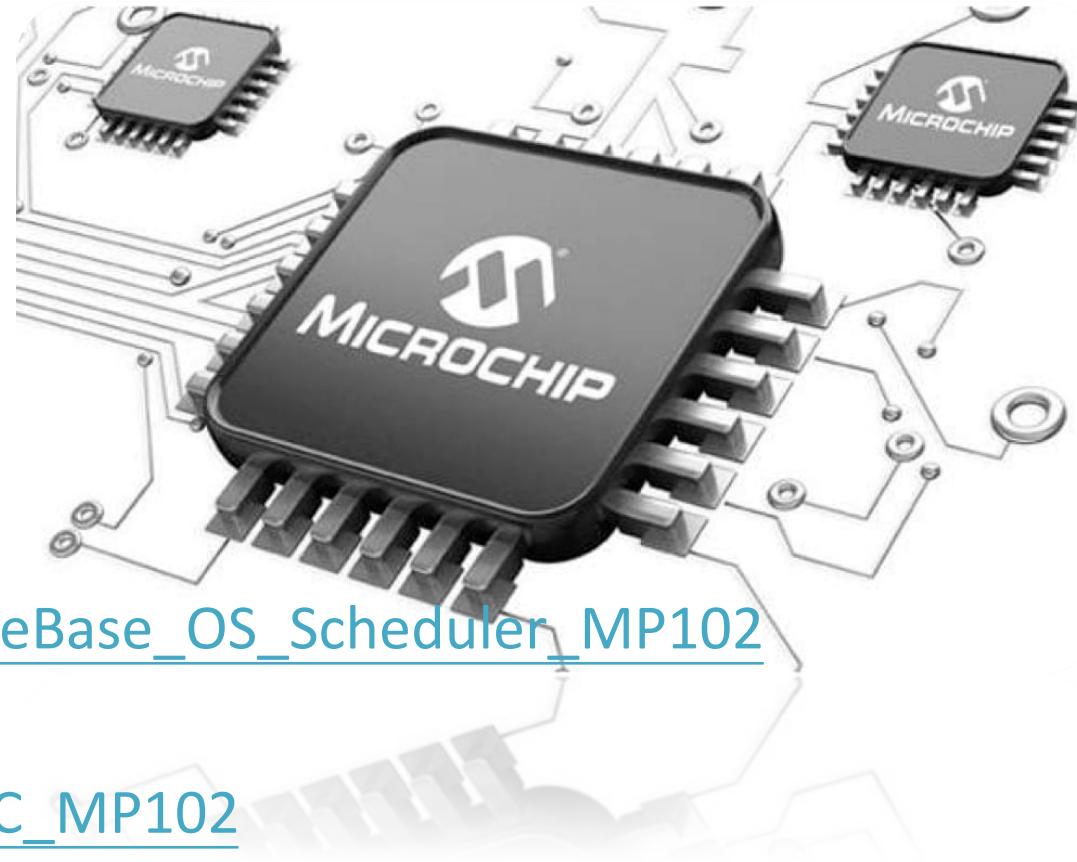


SMART | CONNECTED | SECURE

Microchip
ESE / Edward Lee
March-20, 2024

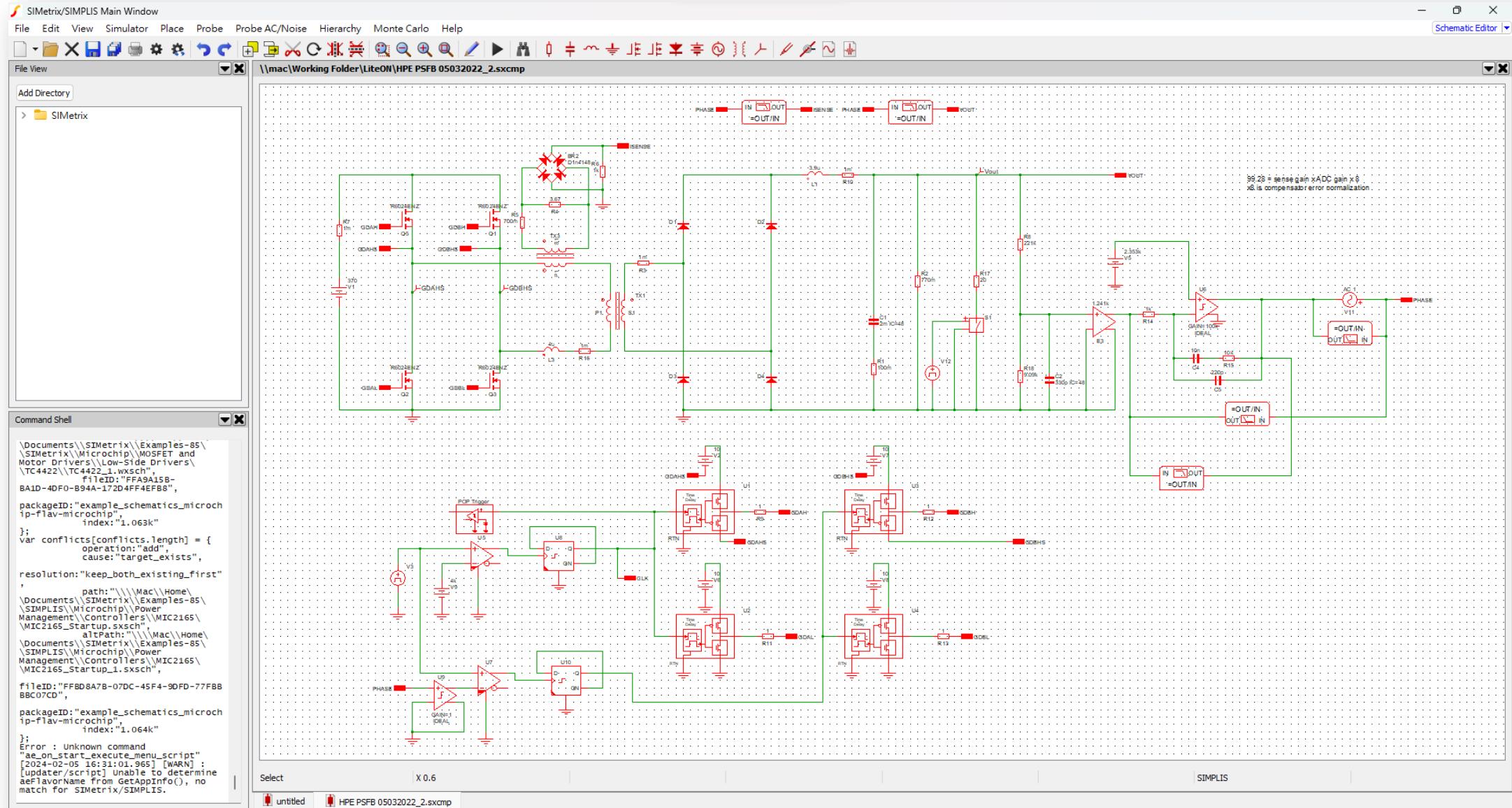
Agenda

- 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
 - PWM2 SOC from PCI and Self-trigger
 - https://github.com/EdwardLeeTW/PSFB_PCMC_MP102

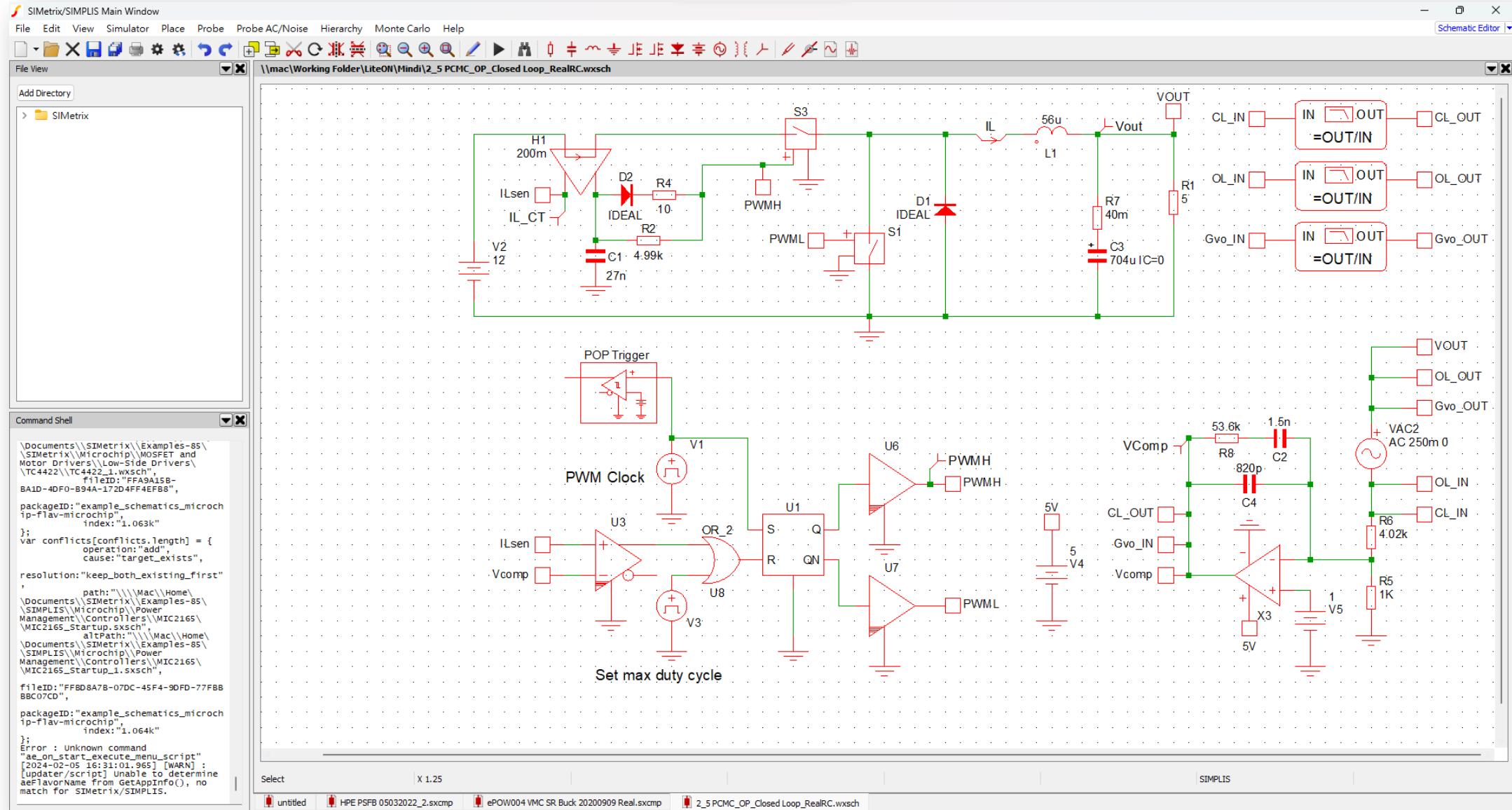


Simulation

SIMPLIS



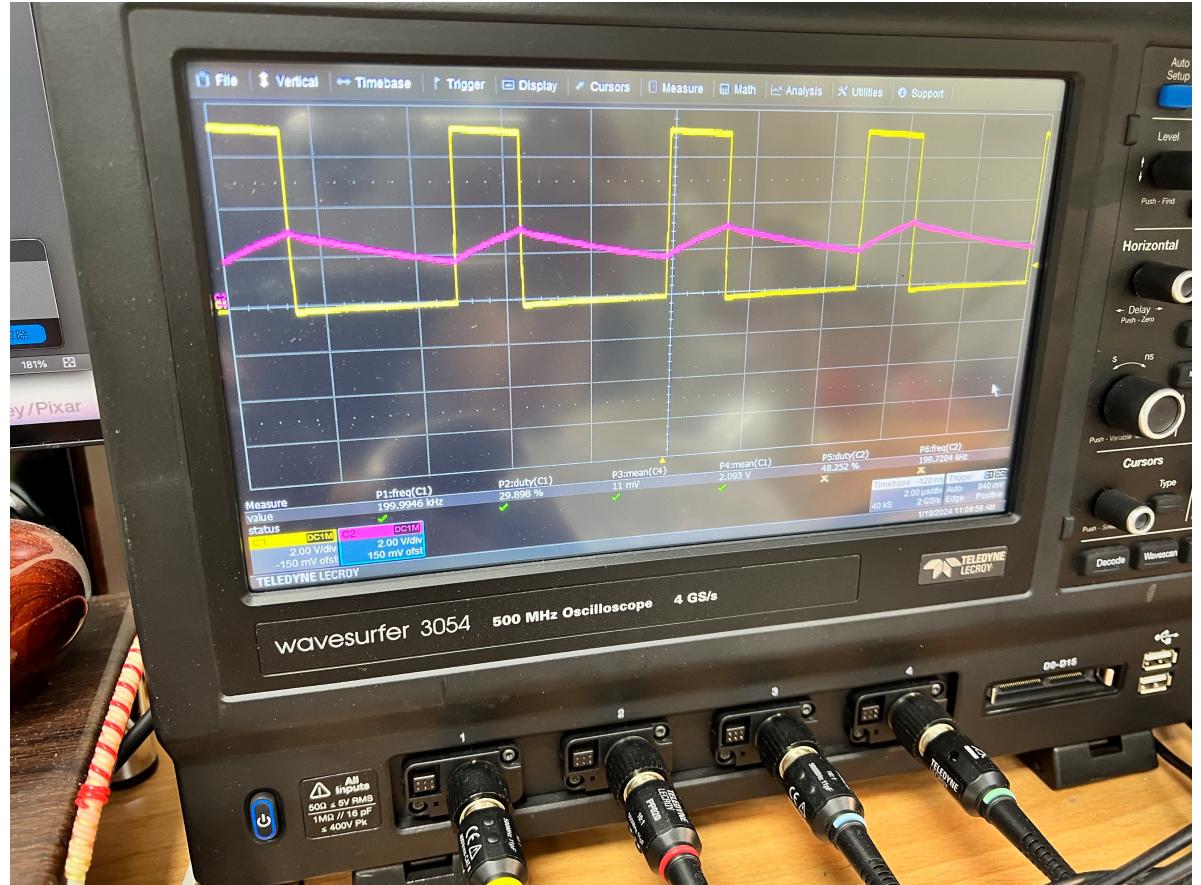
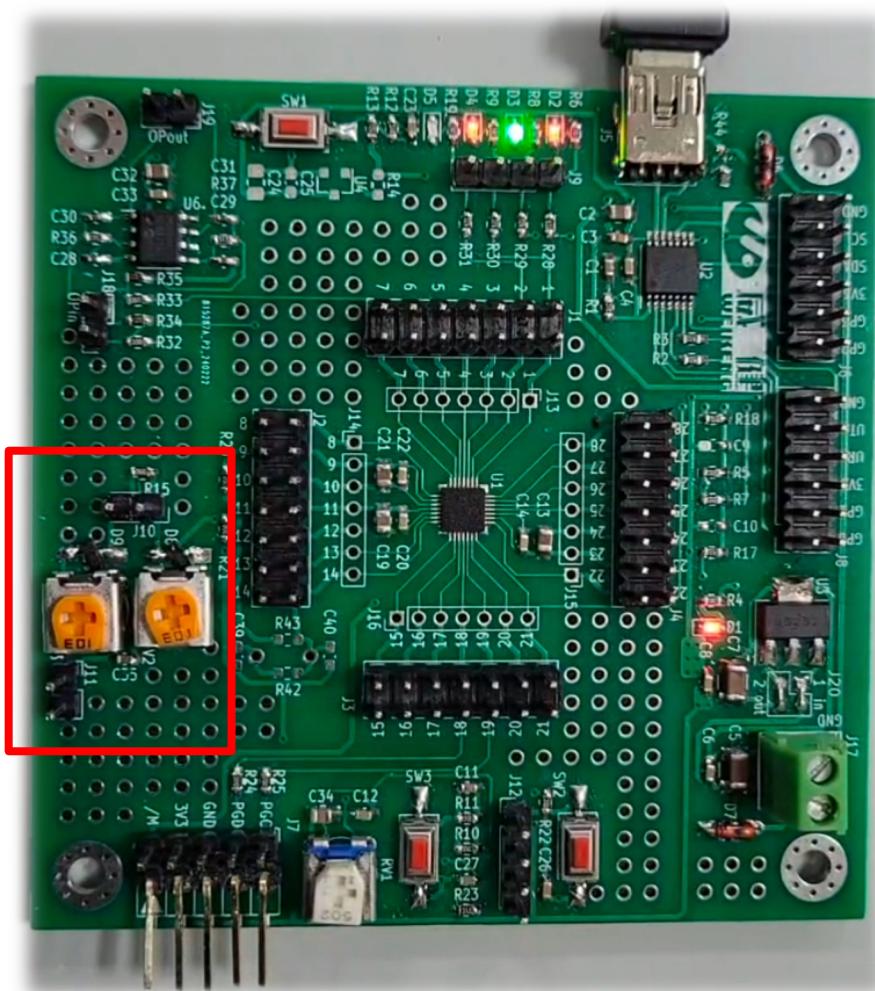
SIMPLIS



Test Environment

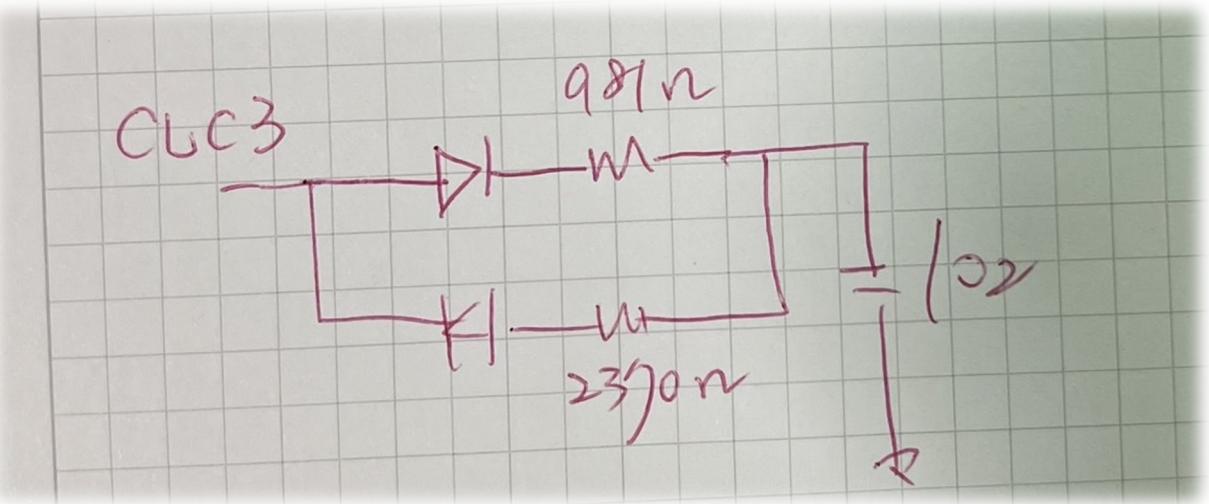
MP102 Debugging Board by Weikeng

- Peak Current Signal Simulation



Modification

- Change PG3 to PG1
 - Release PG3
- LED1
 - Removed R28
 - Connect to J9-1 (MCU Pin #17)
- Peak Current 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

OSC2 is general purpose digital I/O pin

Reference Oscillator Output

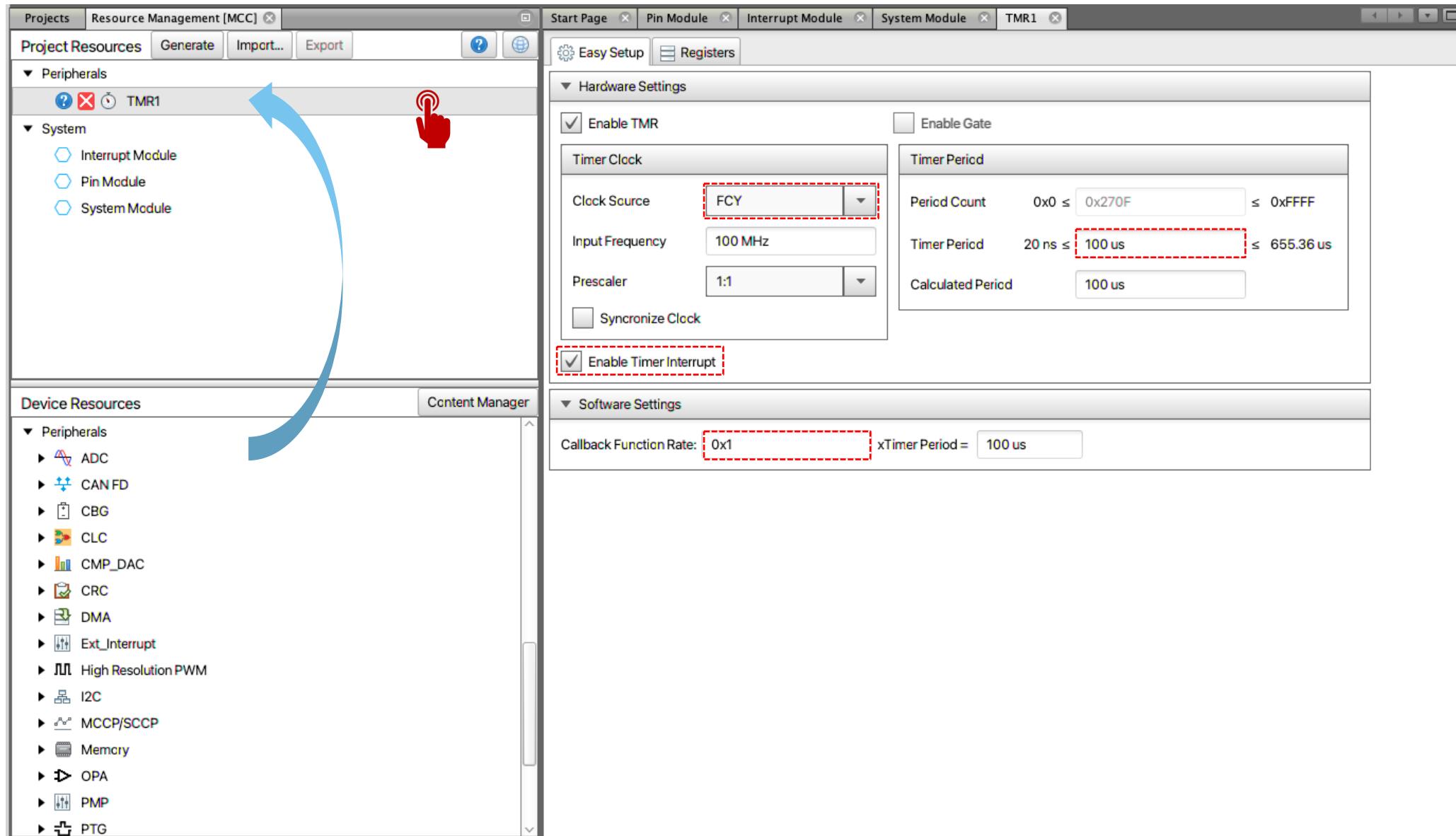
CAN FD Clock Generator

Notifications Output Pin Manager: Grid View Notifications [MCC]

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

Search Results

MCC Drivers – Timer1 (100 us)



MCC Drivers – Pin Management

MPLAB X IDE v6.20 - MP102_Codebase : default

Projects Files Classes Services Resource M... MCC v5.5.0 Project Resources Generate Import... Export Selected Package : QFN28

Pin Module Interrupt Module System Module TMR1

Easy Setup Registers

Pin Name Module Function Custom Na... Start High Analog Output WPU WPD OD IOC

Pin Name	Module	Function	Custom Na...	Start High	Analog	Output	WPU	WPD	OD	IOC
RB4	Pin Module	GPIO	LED1			<input checked="" type="checkbox"/>				n...
RB8	ICD	PGD1								n...
RB9	ICD	PGC1								n...

Device Resources Content Manager

- I2C
- MCCP/SCCP
- Memory
- OPA
- PTG
- QEI
- SENT
- SPI
- UART

Examples

- Explorer 16/32 Demo
- LED Blink Example

Mikro-E Clicks

Audio and Voice

MP102_Codebase - Dashboard Navigator

- MP102_Codebase
- Project Type: Application – Configuration: default
- Device dsPIC33CK32MP102
 - Checksum: Blank, no code loaded
 - CRC32: Hex file unavailable
- Packs dsPIC33CK-MP_DFP (1.13.366)
- Compiler Toolchain XC16 (v2.10) [/Applications/microchip/xc16/v2.1]
- Production Image: Optimization: gcc 0
- Device support information: dsPIC33CK-MP_DFP
- Memory Usage Symbols disabled. Click to enable Load Sym

Pin Manager: Package View

RB9|PGC1
VS2A
VDD4A
RB13
RB12
RB10
RB11
RA14 1 28 27 26 25 24 23 22 21 R88|PGD1
RA15 2 20 R87
NMCLR 3 19 R86
RA0 4 18 R85
RA1 5 dsPIC33CK32MP102 17 R84|LED1|GPIO
RA2 6 16 R83
RA3 7 15 R82
8 9 10 11 12 13 14 15 RA4
AVDDA
AVSSA
VDD2A
VSS2A
RB0
RB1

Output Pin Manager: Grid View Notifications [MCC]

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																							

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:

Pin Manager: Grid View:

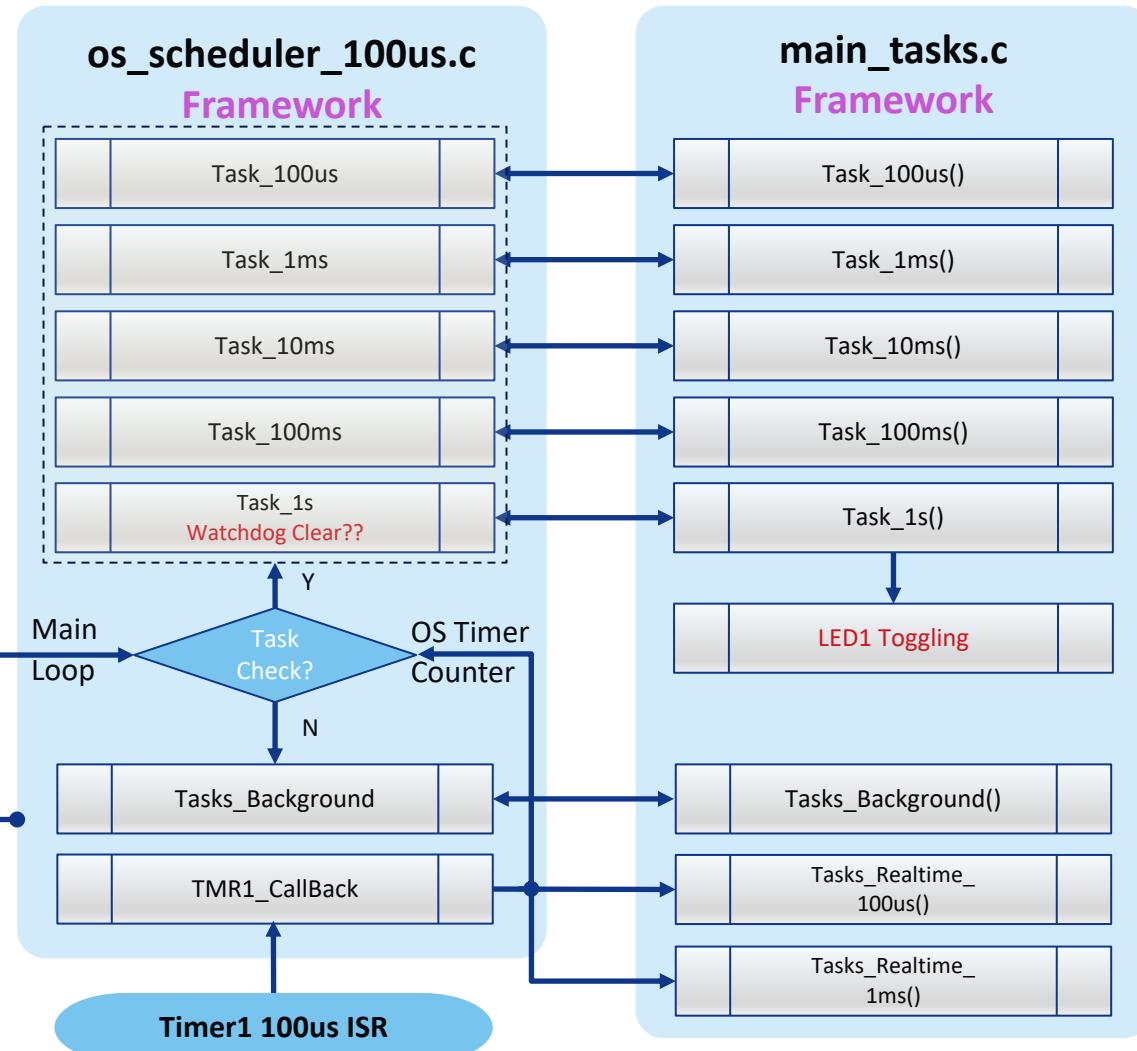
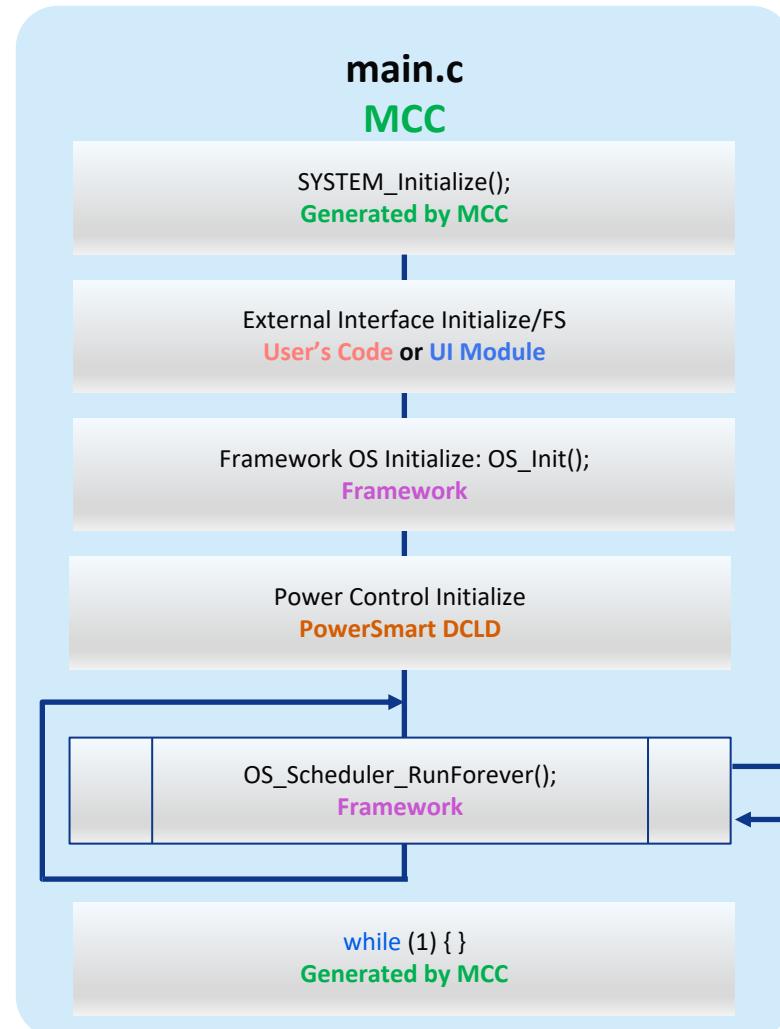
Package: QFN28	Pin No:	4	5	6	7	8	13	14	15	16	17	18	19	20	21	22	25	26	27	28	1	2		
Port A ▼		Port B ▼																						
Module	Function	Direction	0	1	2	3	4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Clock ▼	CLKI	input						🔒																
	CLKO	output						🔒																
	OSCI	input						🔒																
	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

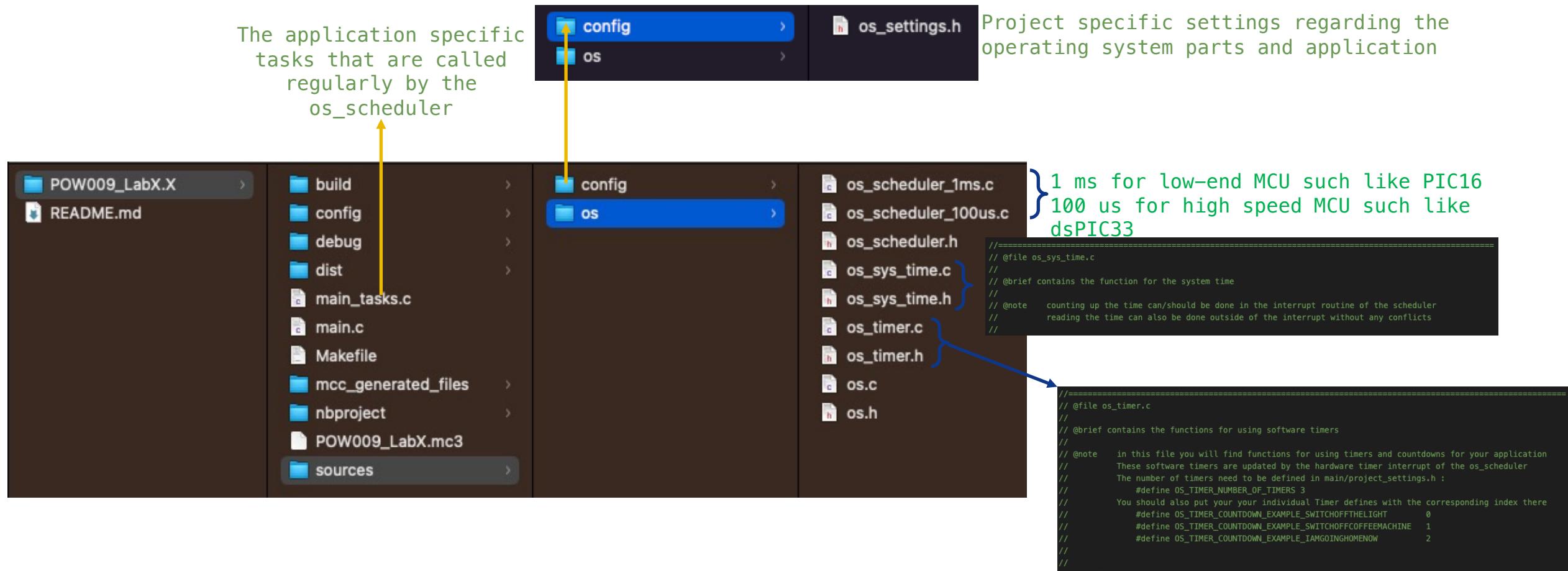


SMPS Firmware Framework

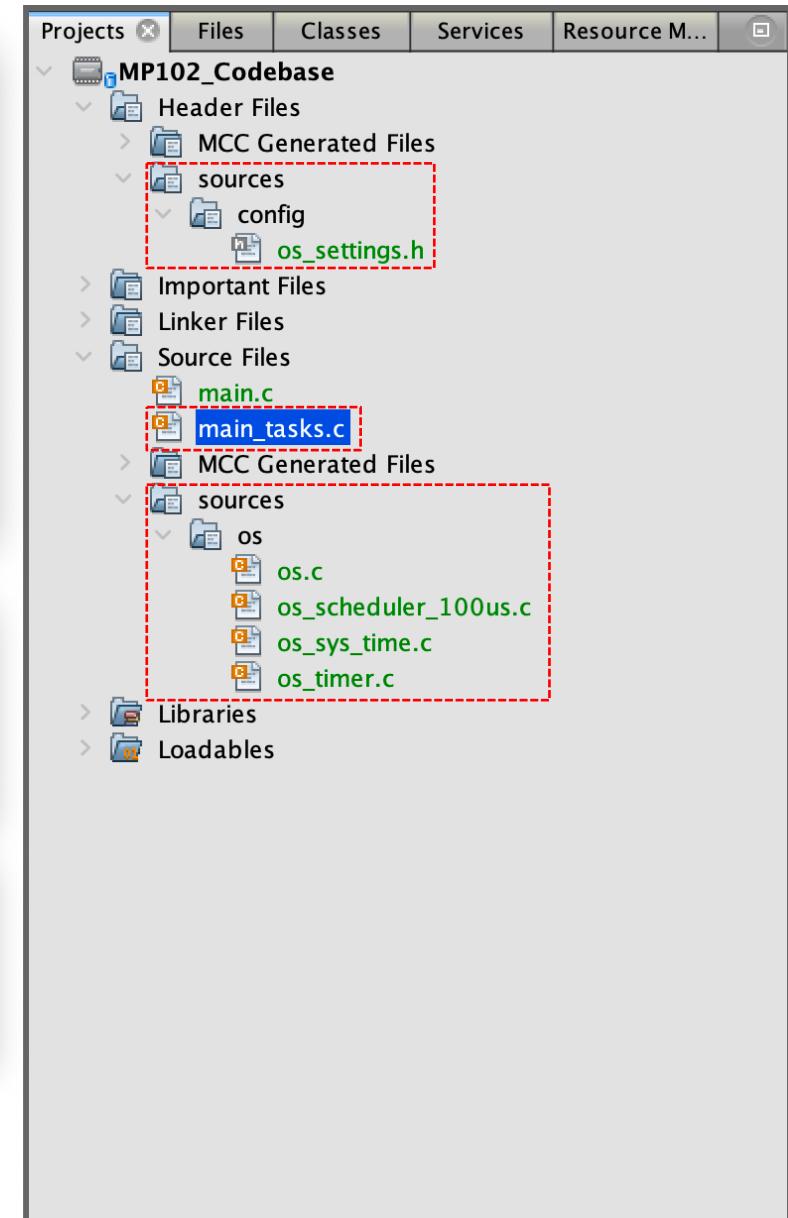
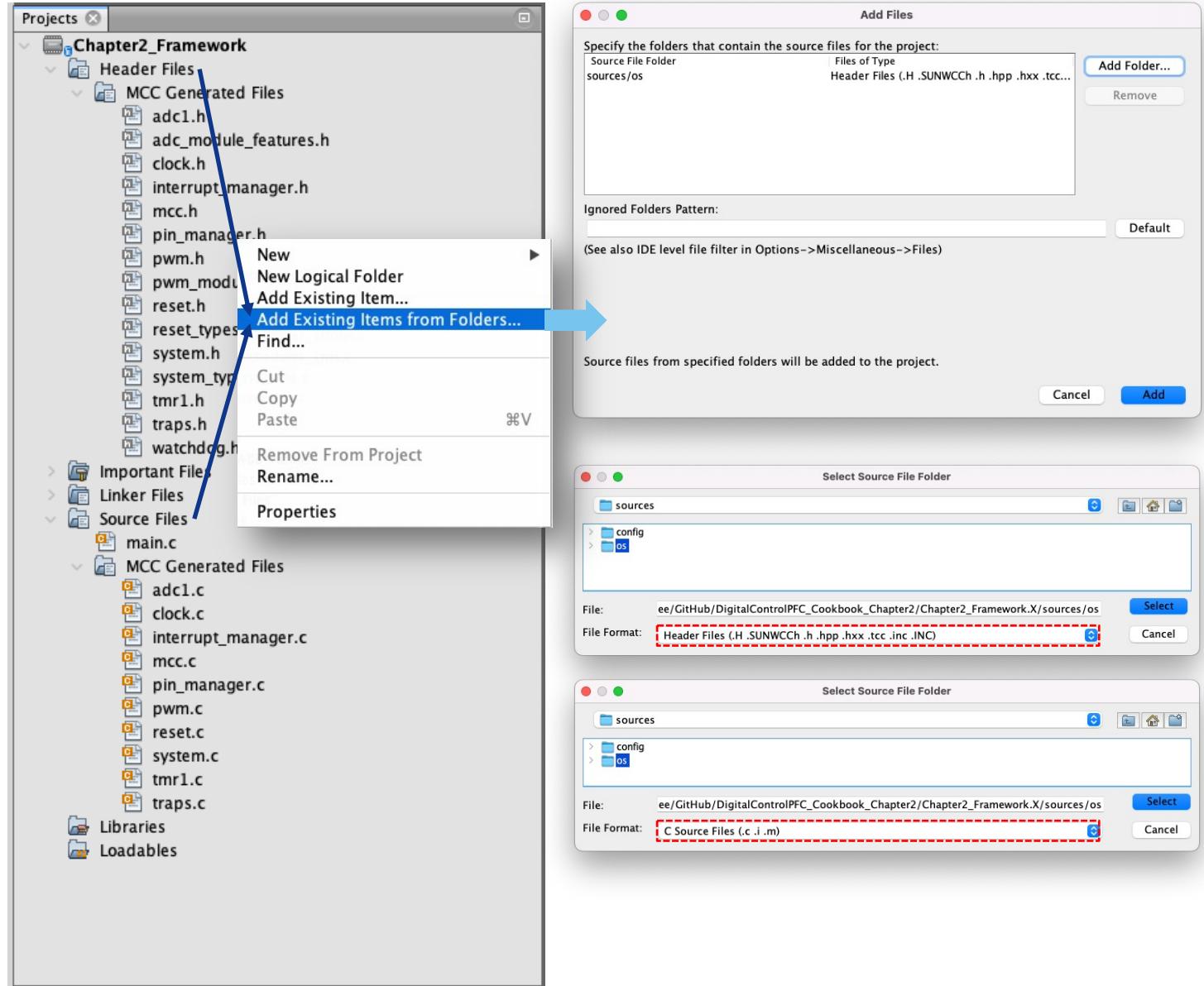
Directory Structure

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

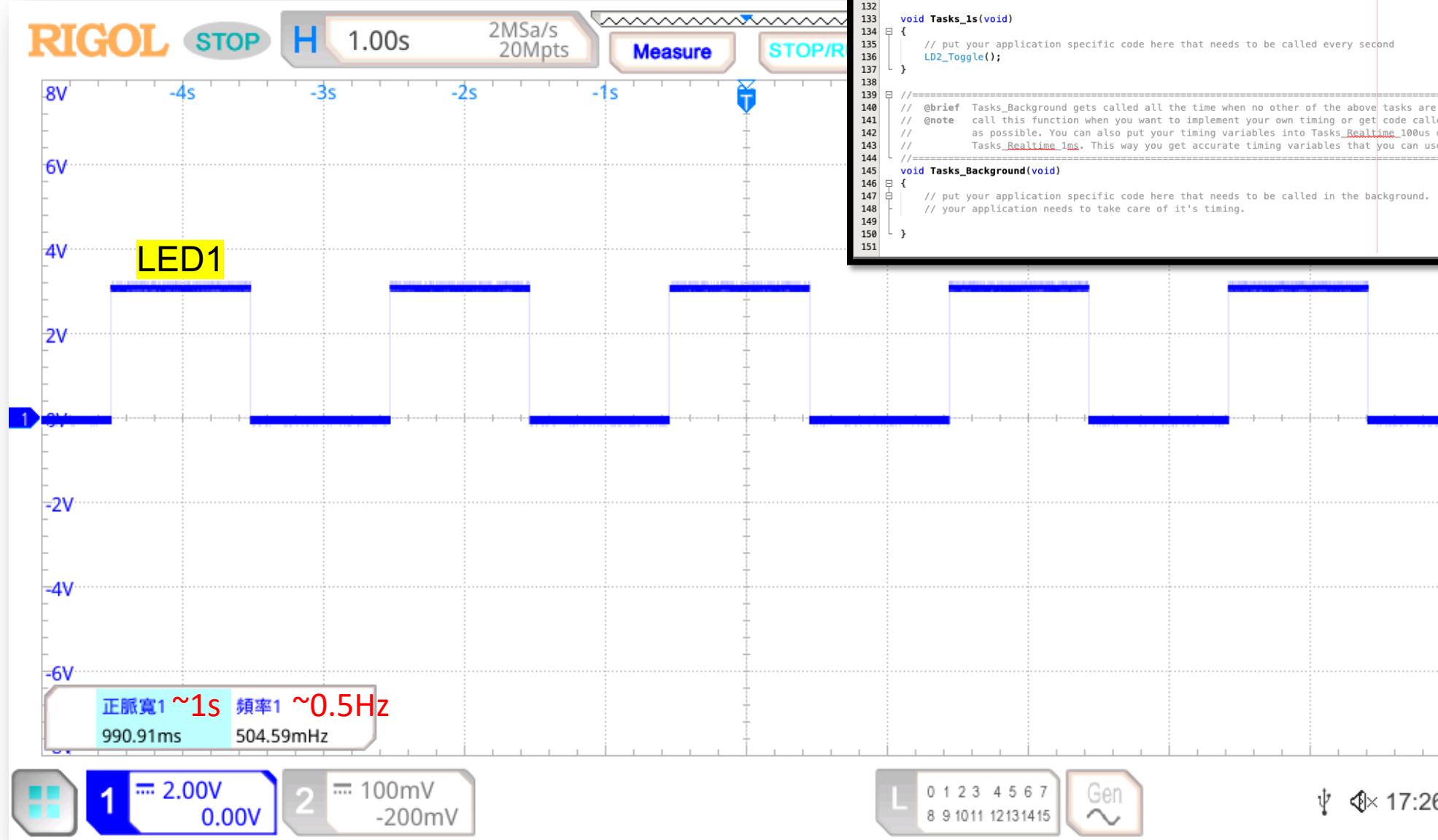


Main & Tasks

```
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



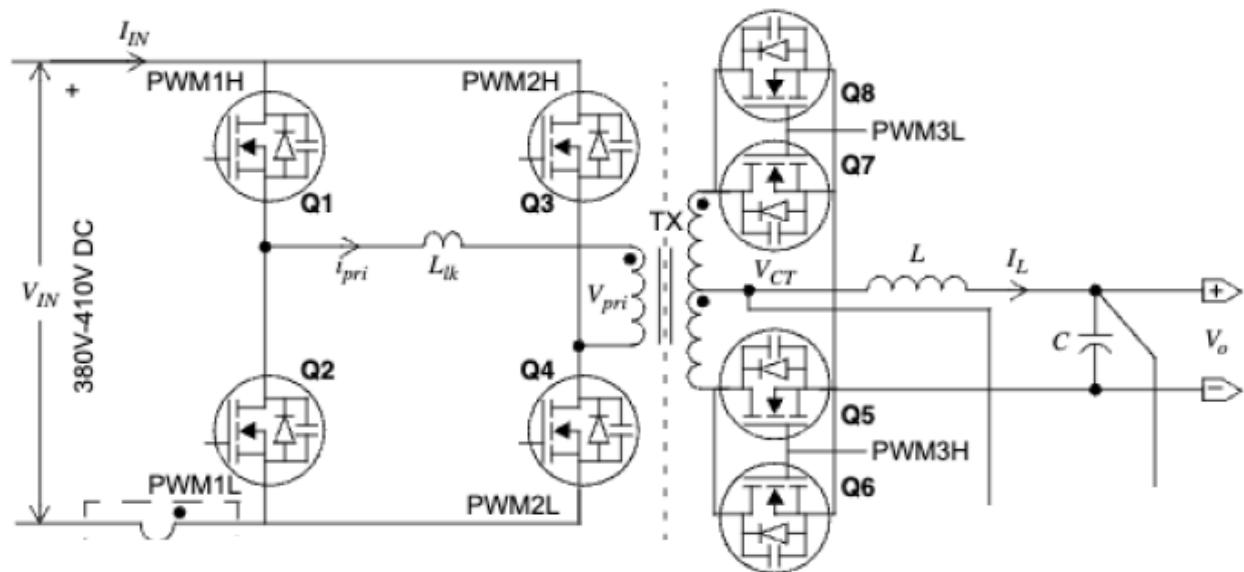
VMC PSFB Converter With SR

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

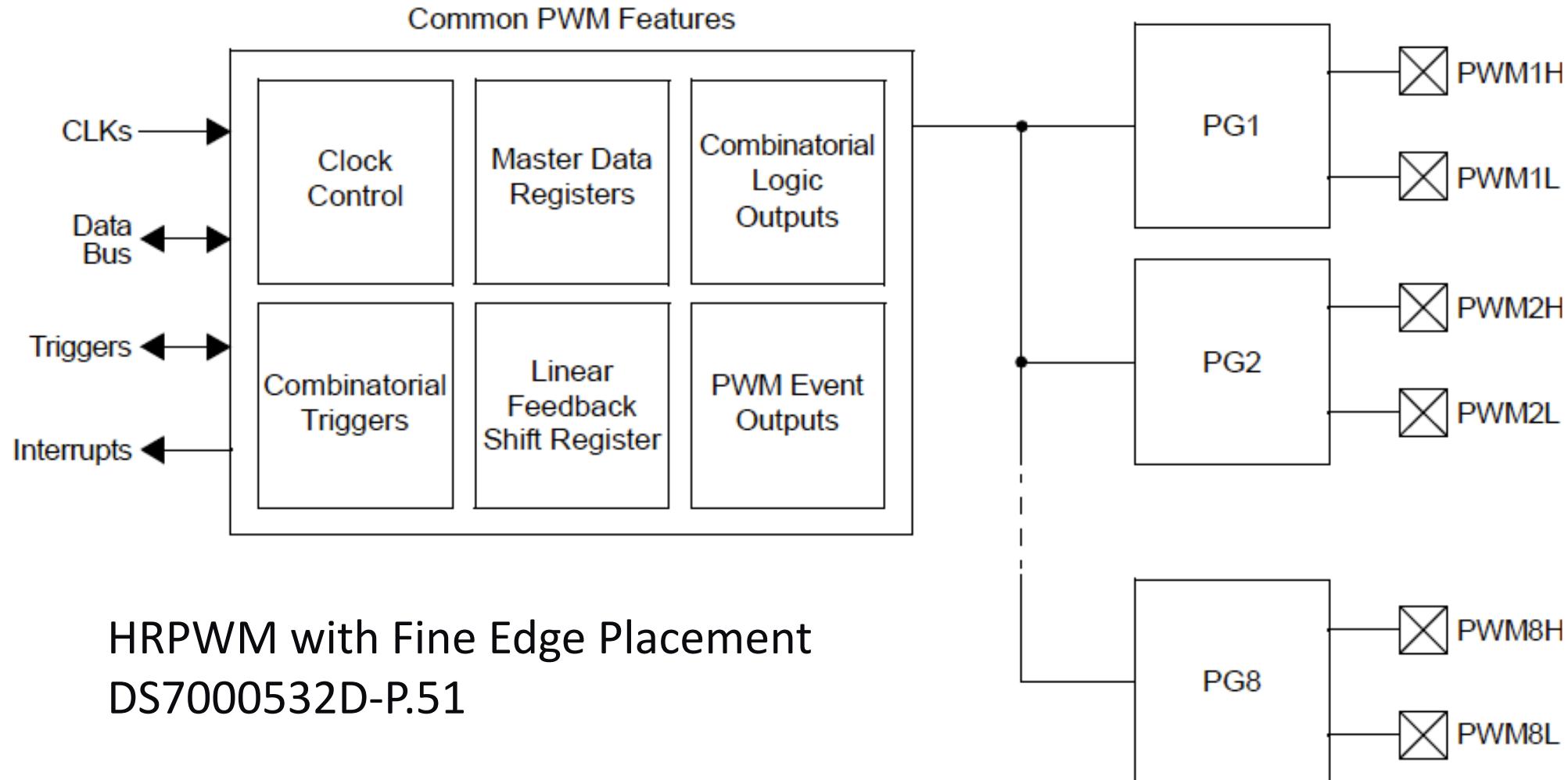
- Peripherals (in RED)
 - PWM1 (Fixed leg)
 - 100kHz (Complementary) with fixed 50% DC
 - Trigger A -> Phase-shifted value to PWM3
 - Trigger B -> ANO Trigger
 - PWM2 (Phase-shifted leg)
 - 100kHz (Complementary) with fixed 50% DC
 - SOC from PC1 (PWM1 Trigger A)
 - Trigger A -> 200ns reserved for SR 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



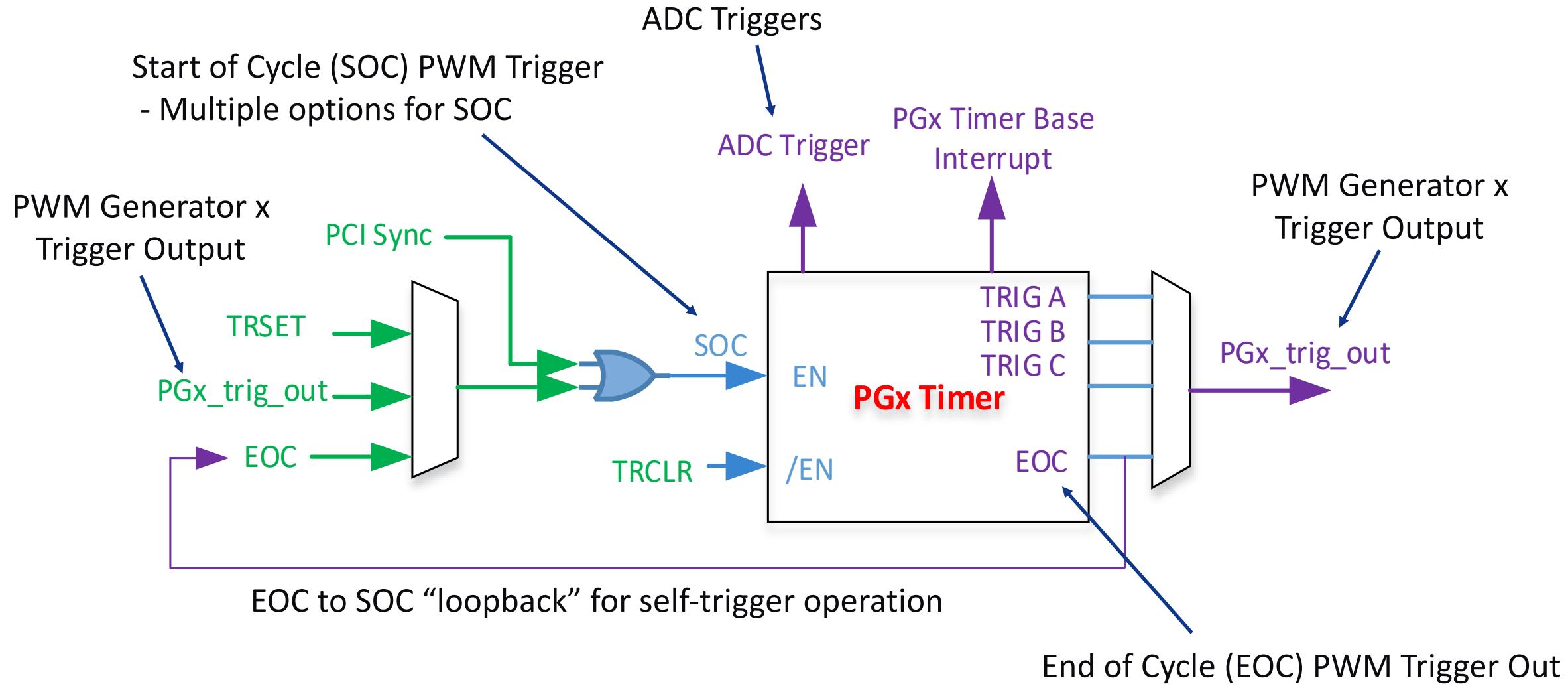
PWM Ideas

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**

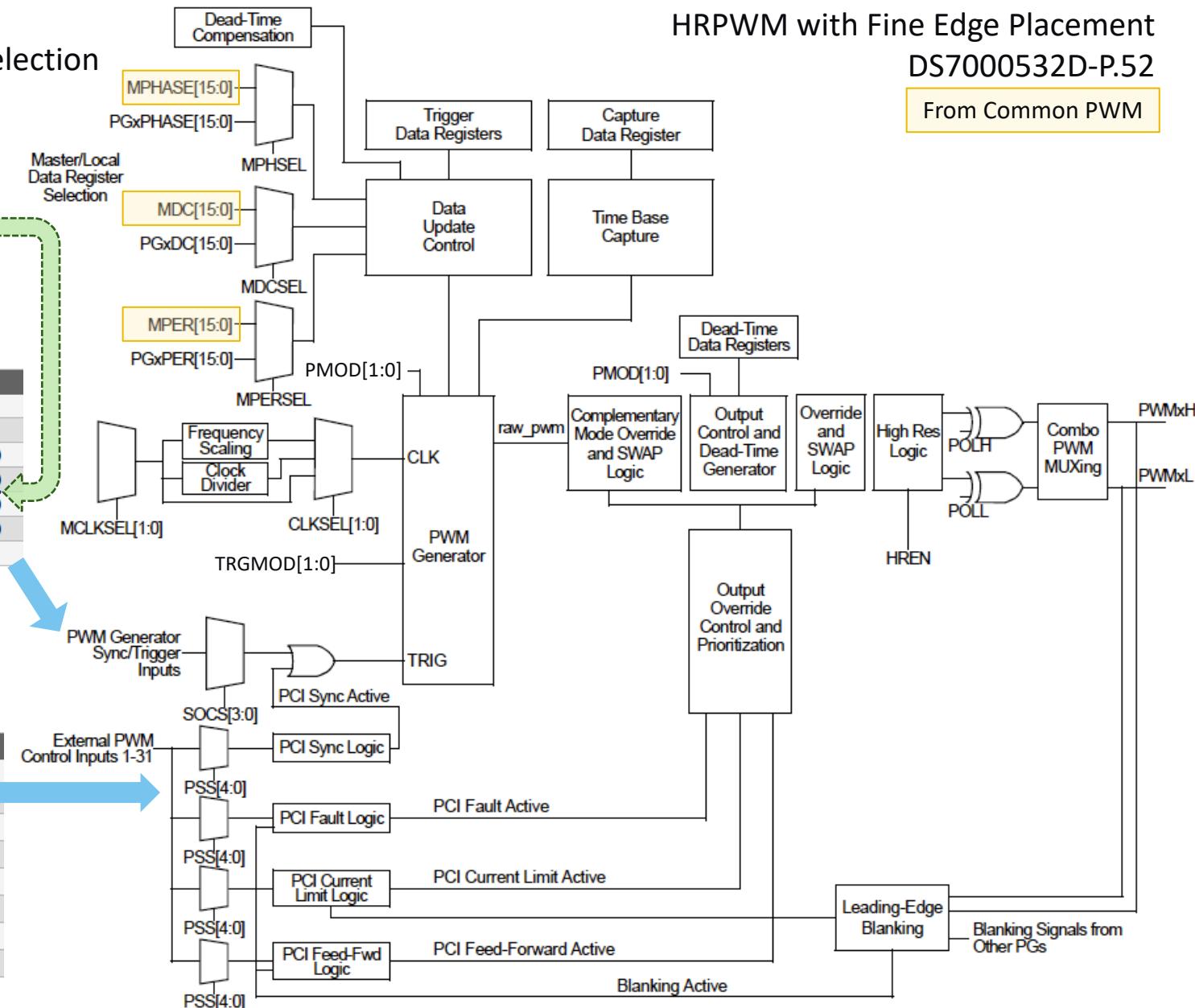
Value	Description
1111	TRIG bit or PCI Sync function only (no hardware trigger source is selected)
1110	Reserved
0100	Trigger output selected by PG4 or PG8 PGTRGSEL[2:0] bits (PGxEVTL[2:0])
0011	Trigger output selected by PG3 or PG7 PGTRGSEL[2:0] bits (PGxEVTL[2:0])
0010	Trigger output selected by PG2 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

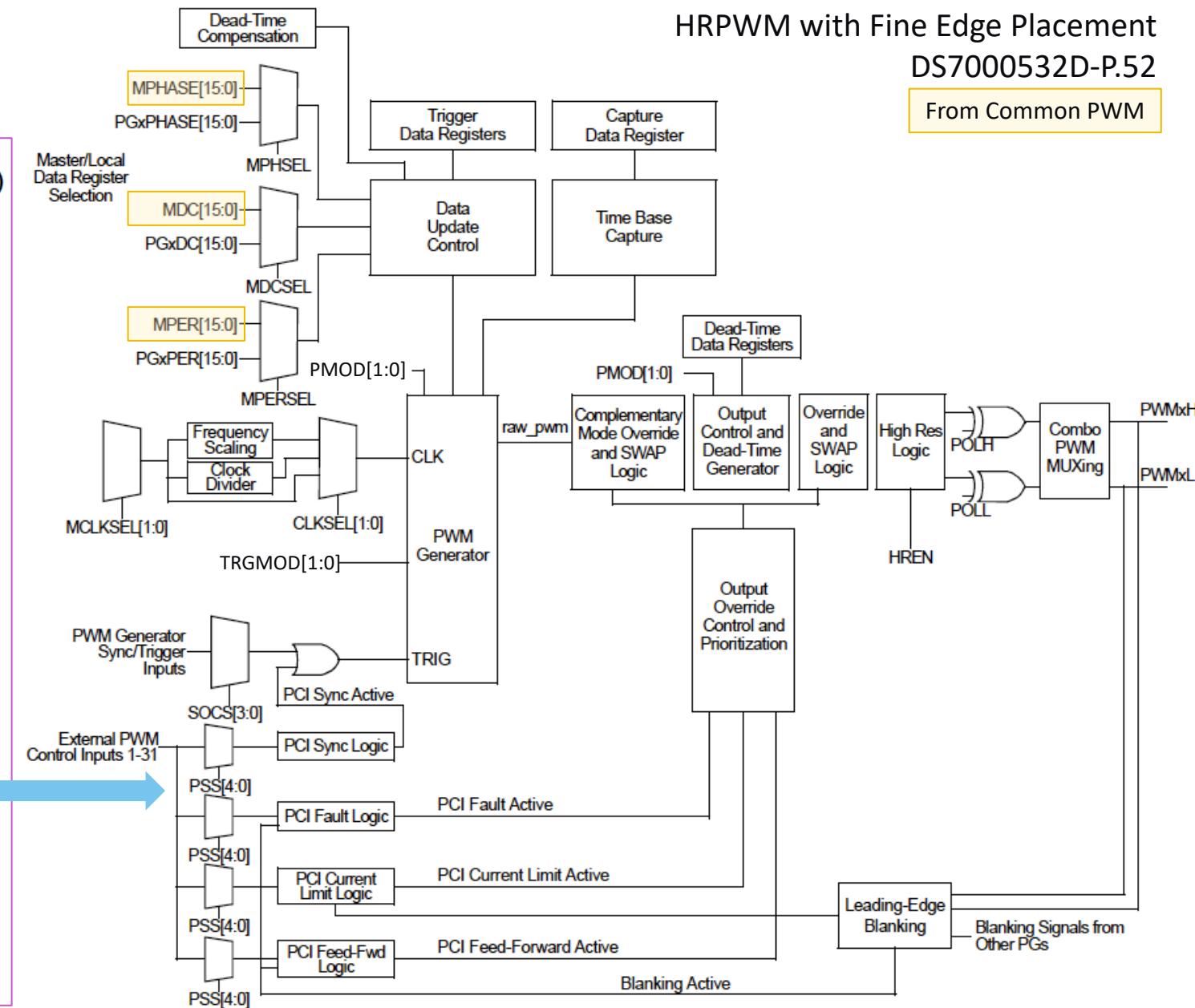
Bits 4:0 – PSS[4:0] PCI Source Selection

**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'

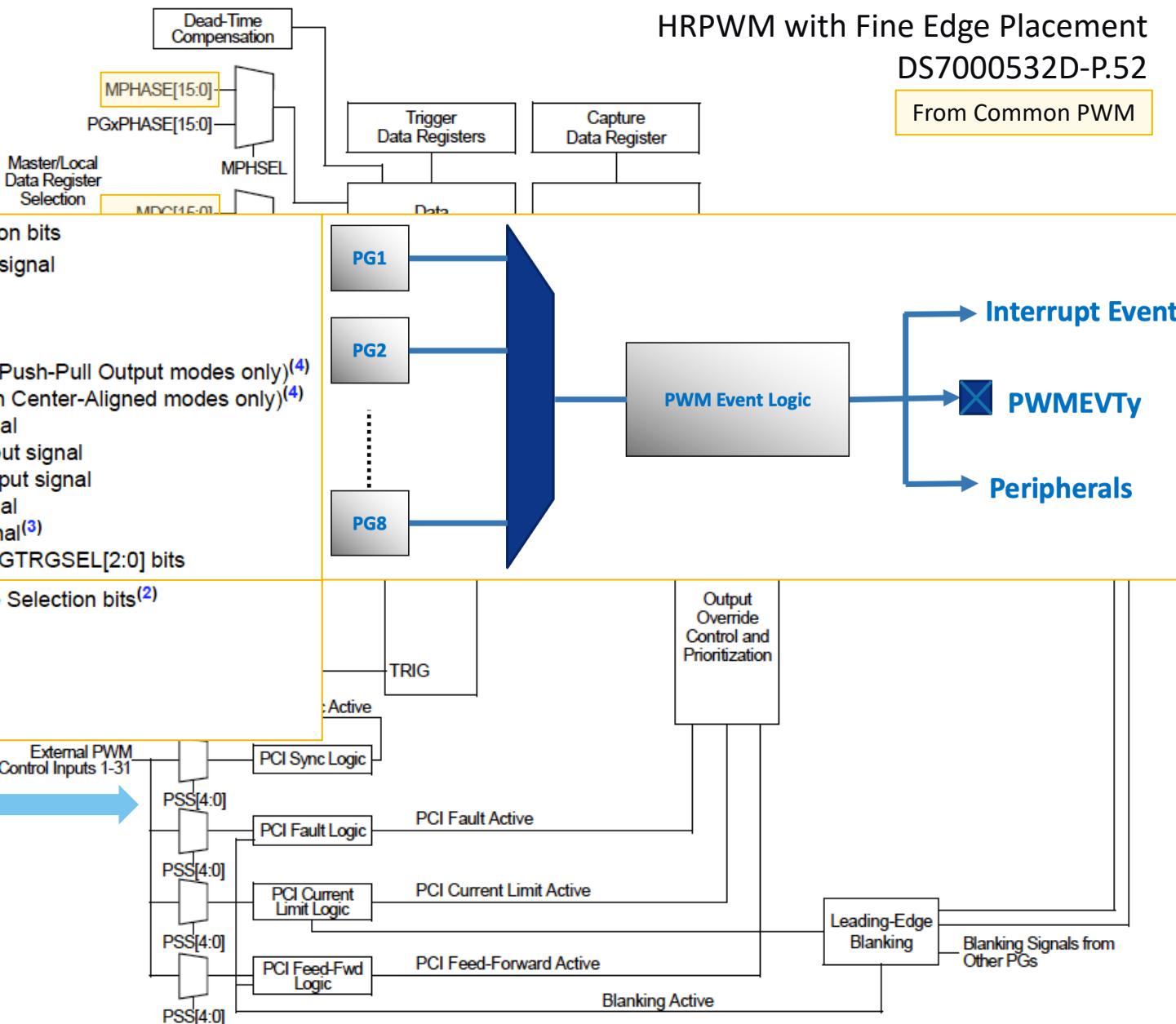
EVTySEL[3:0]: PWM Event Selection bits

- 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

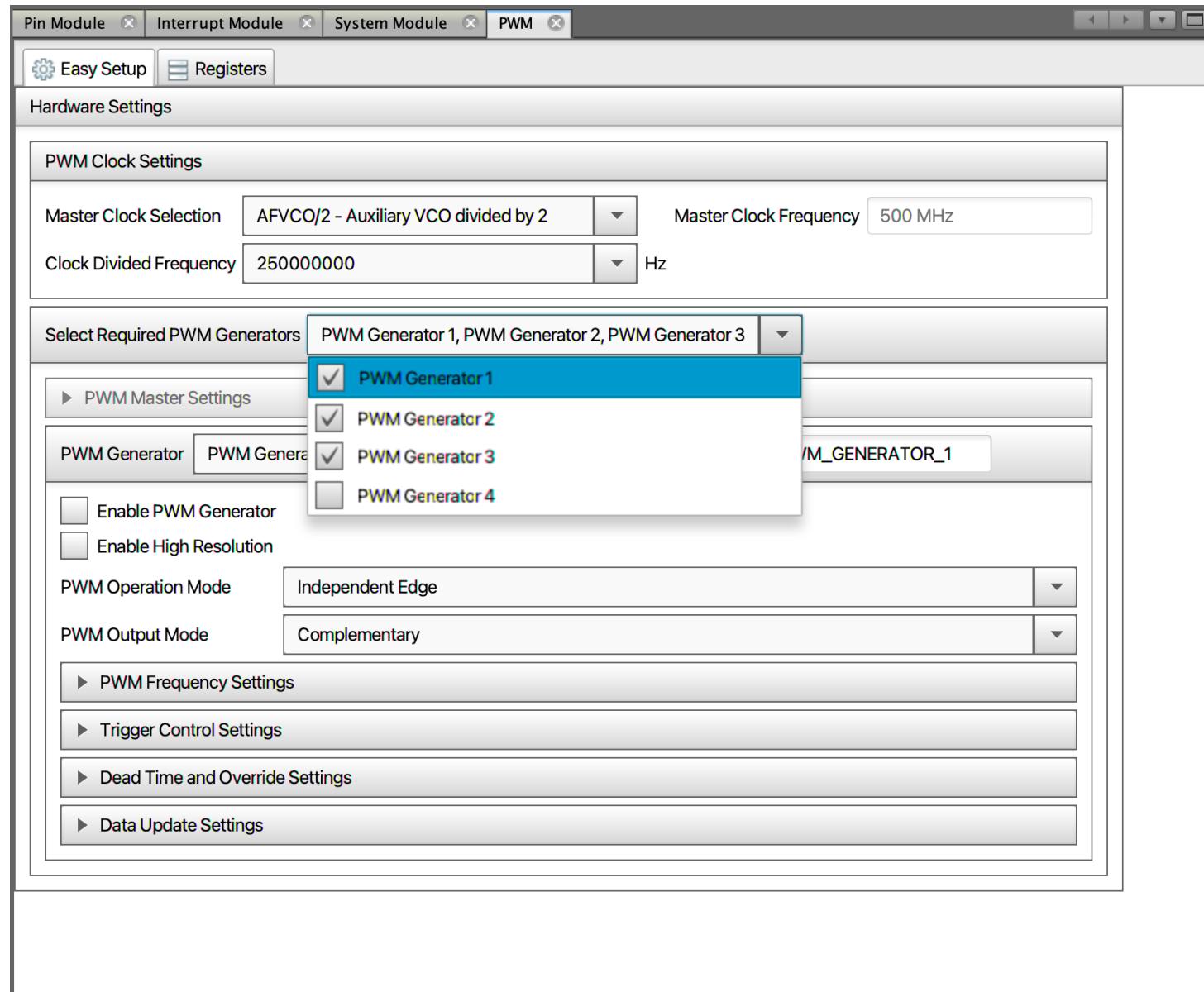
EVTyPGS[2:0]: PWM Event Source Selection bits⁽²⁾

- 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	\leq 100 kHz	\leq 29.41176471 MHz
Calculated Frequency	100 kHz	
Requested Period: 34 ns	\leq 10 us	\leq 16.3858 us
Calculated Period	10 us	

Duty Cycle Use Master Duty Cycle

PWM Duty Cycle: 0 %	\leq 50	\leq 100 %
---------------------	-----------	--------------

Phase

PWM Phase: 0 ns	\leq 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 \leq 0 ns \leq 16.3838 us Default phase-shifted value

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

Trigger C Compare: 0 ns \leq 0 ns \leq 16.3838 us

Dead Time and Override Settings

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

PWM L Override: disabled
PWM H Override: disabled

Data Update Settings

Update Trigger: Trigger A
Update Mode: SOC update

Updating Trig1A with updating Duty/Period/...in sync

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

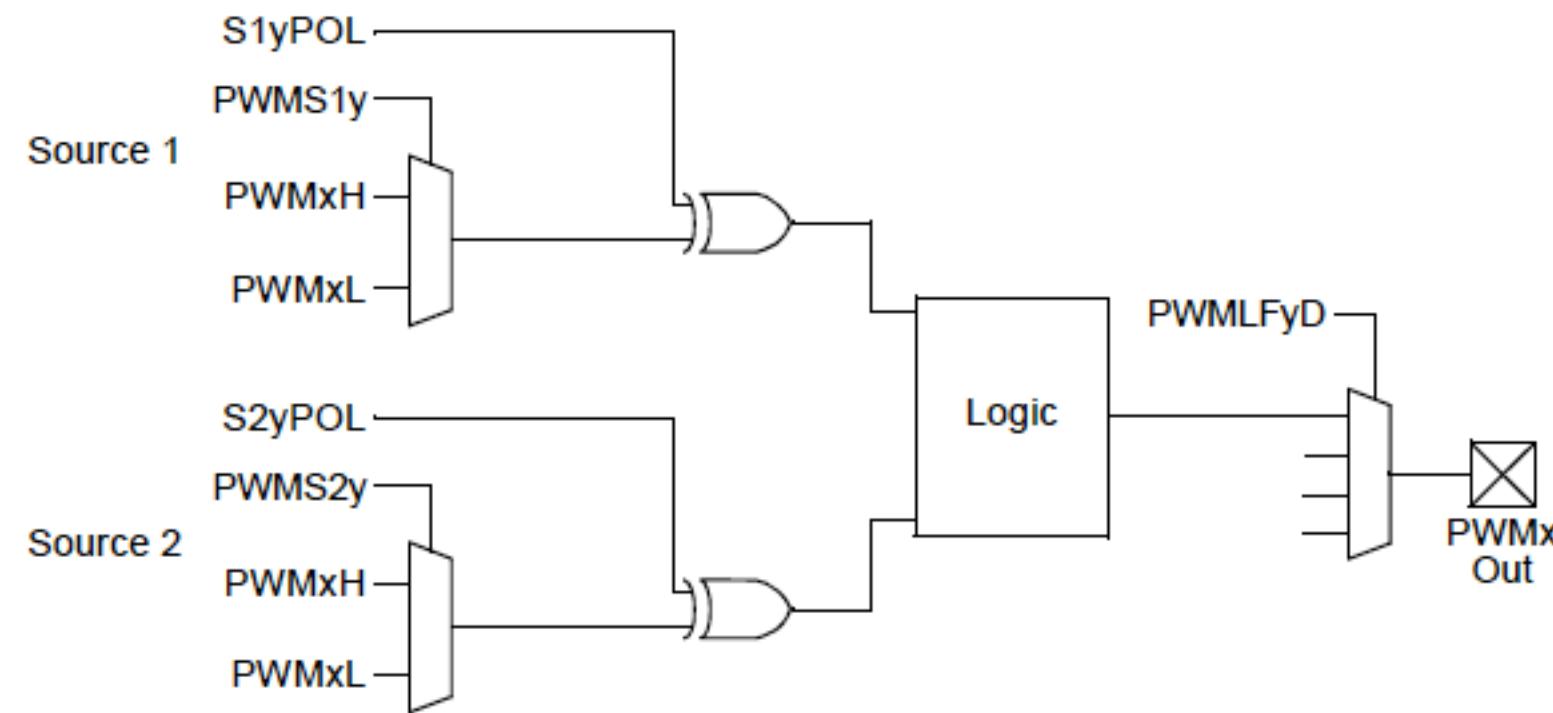
PWM L Override: disabled
PWM H Override: disabled

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

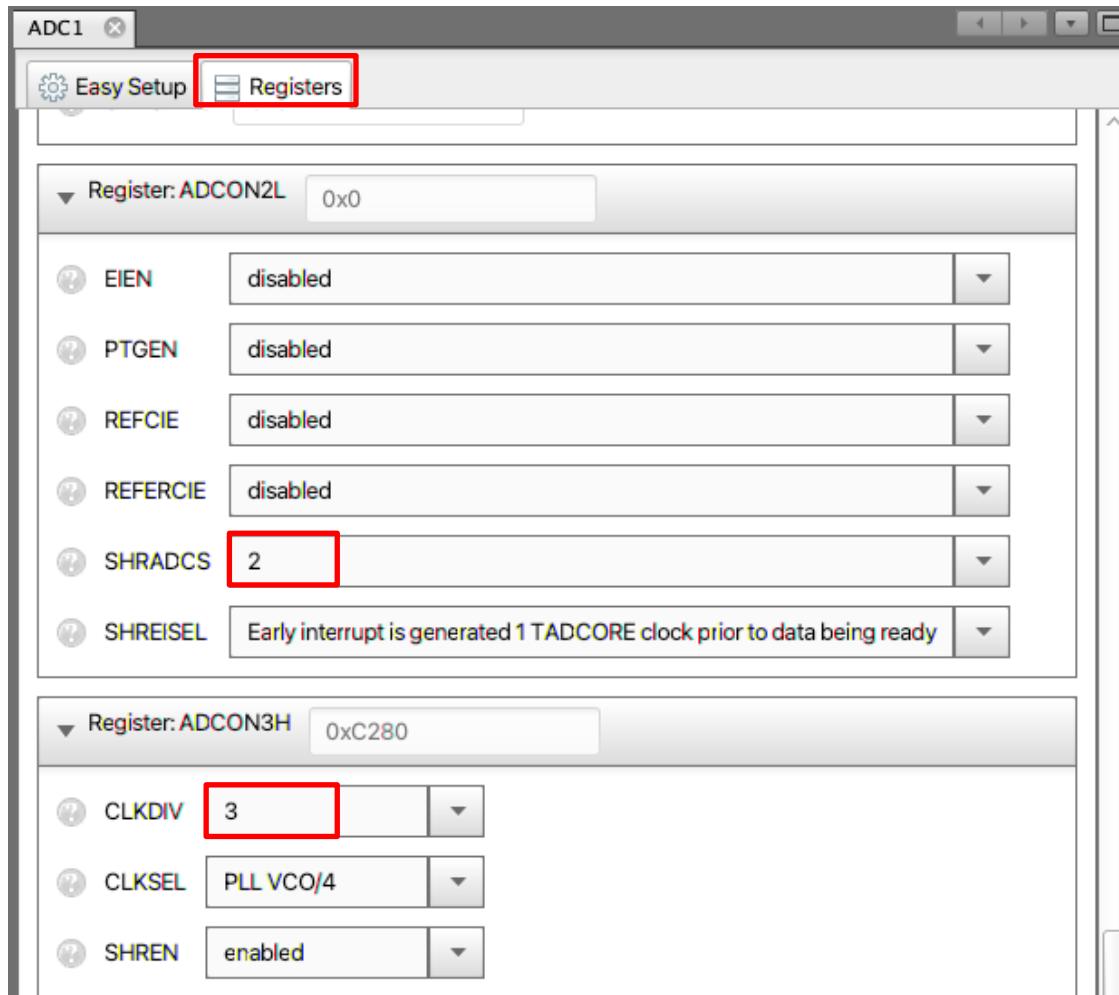
- Conversion Clock Source: PLL VCO/4 (highlighted with a red dashed box)
- Conversion Time: 92.5 ns
- Target Shared Core Sampling Time: 40 ns
- Calculated Shared Core Sampling Time: 40 ns

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"/>

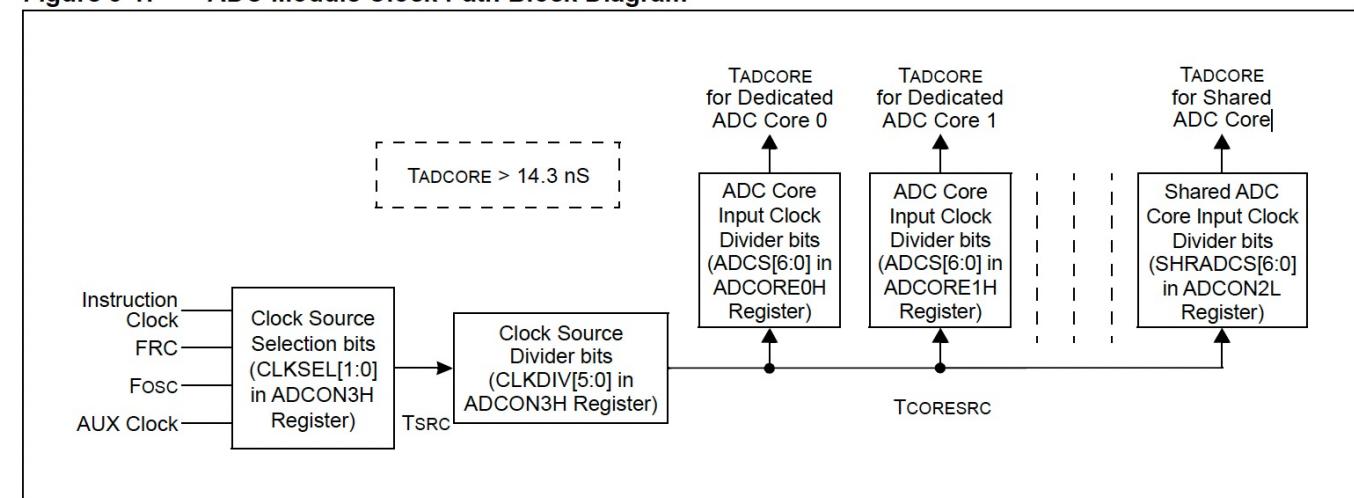
A blue callout box highlights the "Trigger Source" dropdown for the Ccore0 row, which is set to "PWM1 Trigger2".

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

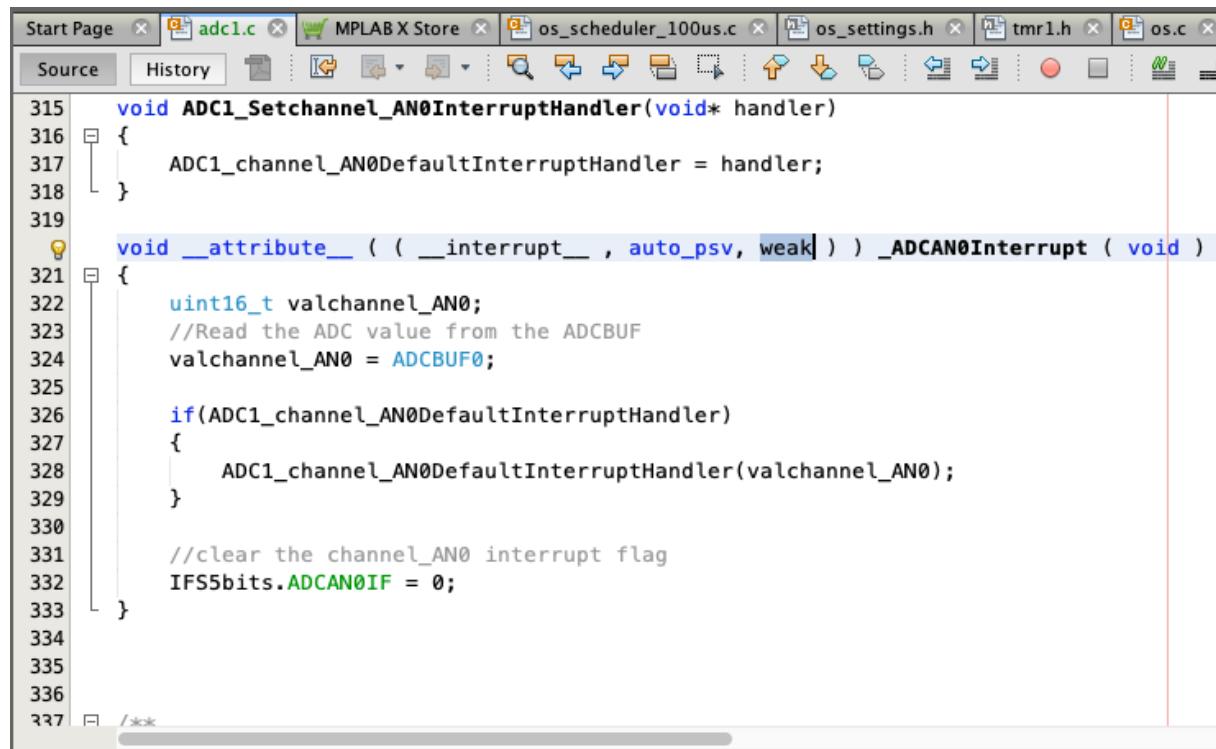
Interrupt Manager

Enable Global Interrupts

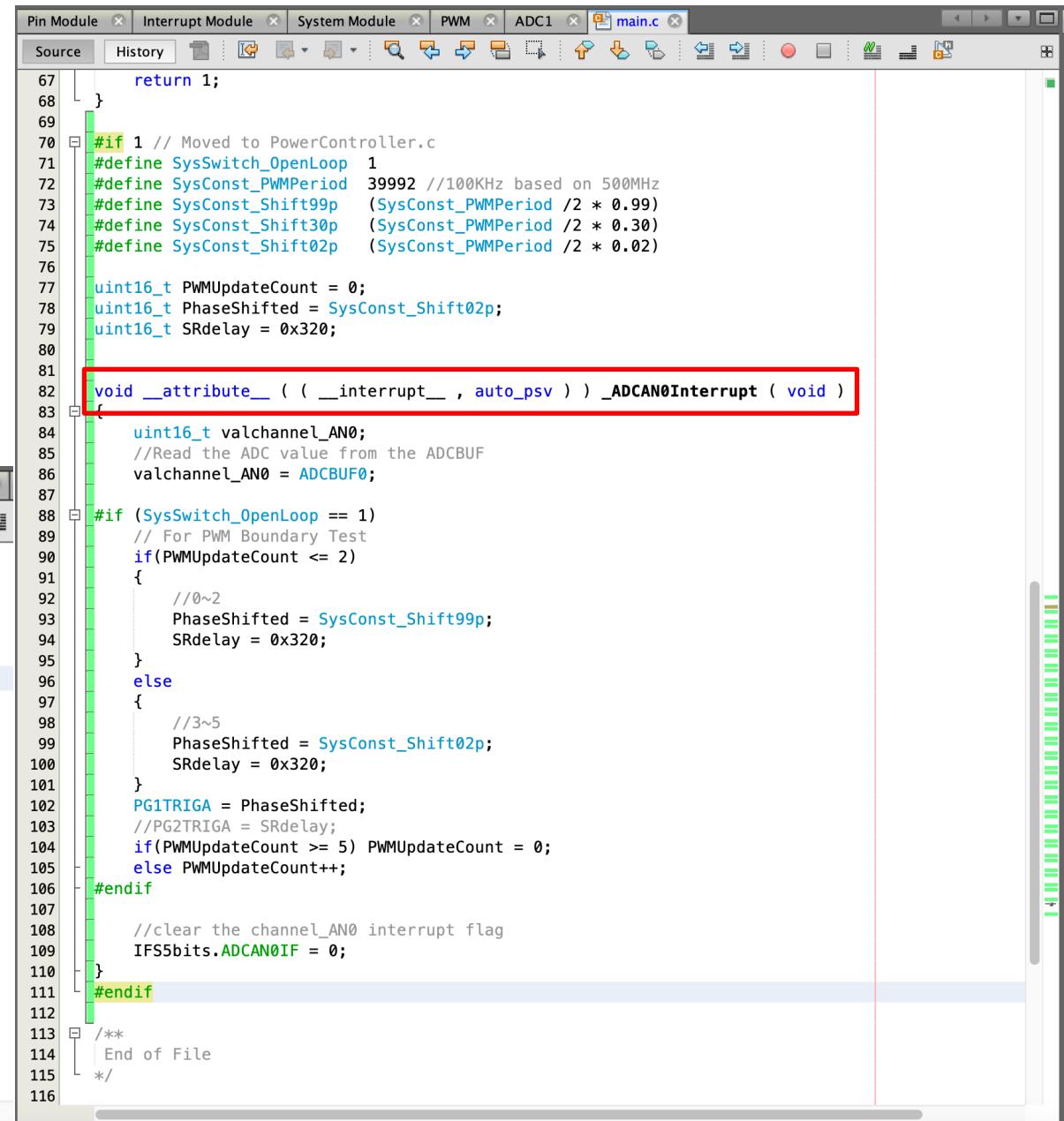
Module	Interrupt	Description	IRQ Number	Enabled	Priority	Context
ADC1	ADCAN6	ADC AN6 Convert Done	97	<input type="checkbox"/>	1	OFF
ADC1	ADCAN7	ADC AN7 Convert Done	98	<input type="checkbox"/>	1	OFF
ADC1	ADCAN24	ADC AN24 Convert Done	192	<input type="checkbox"/>	1	OFF
ADC1	ADCAN8	ADC AN8 Convert Done	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”

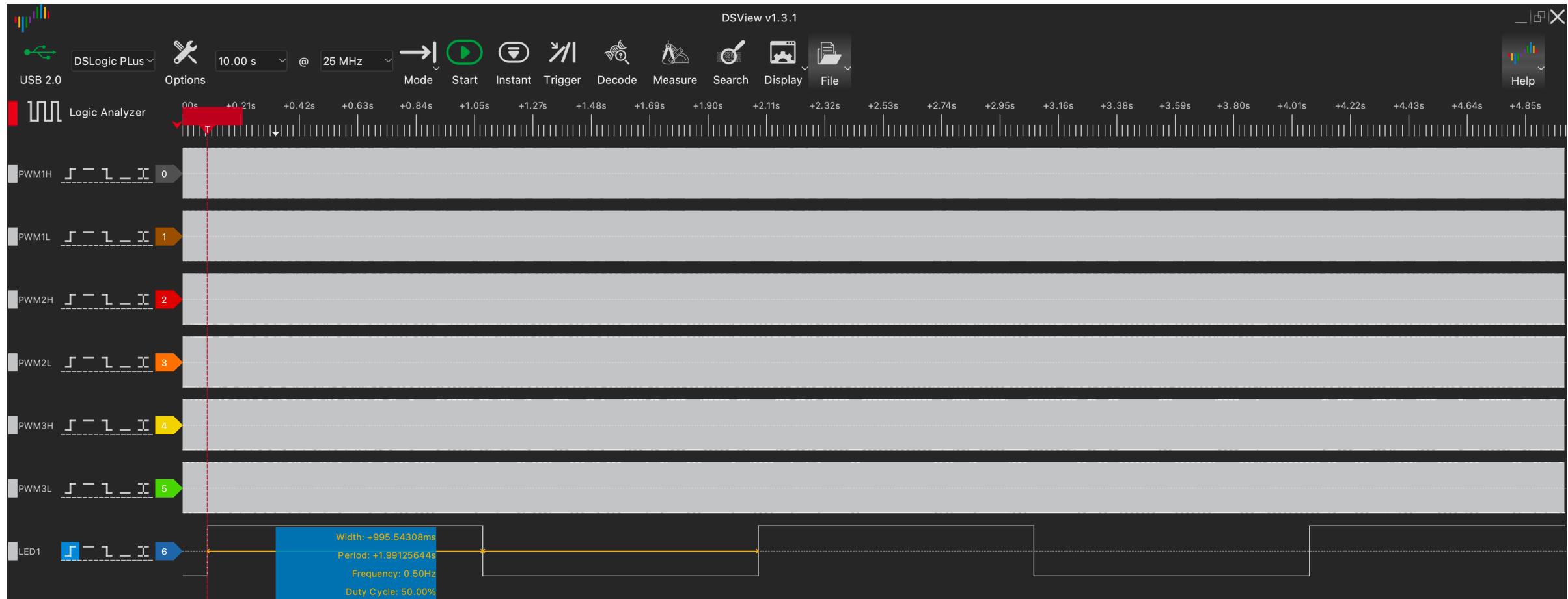


```
Start Page x adc1.c x MPLAB X Store x os_scheduler_100us.c x os_settings.h x tmr1.h x os.c x
Source History ... _ADCAN0Interrupt ( void )
315 void ADC1_Setchannel_AN0InterruptHandler(void* handler)
316 {
317     ADC1_channel_AN0DefaultInterruptHandler = handler;
318 }
319
320 void __attribute__ (( __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 }
```

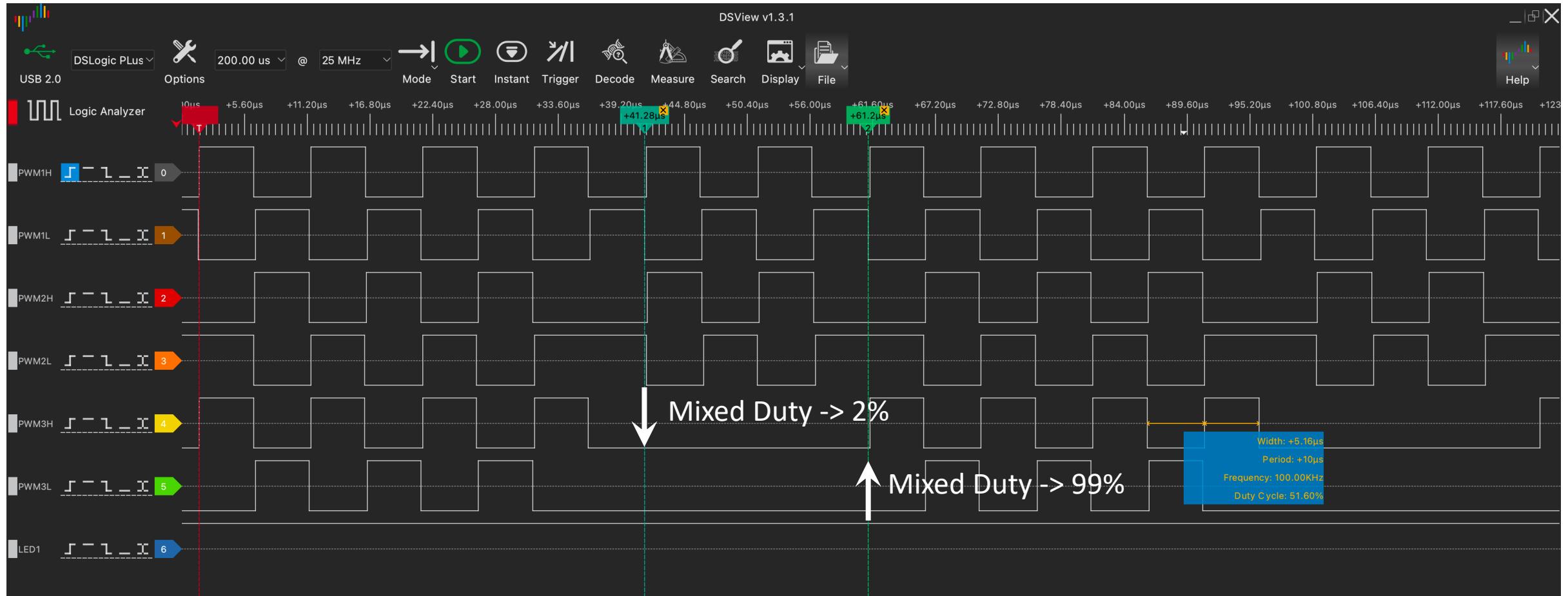


```
Pin Module x Interrupt Module x System Module x PWM x ADC1 x main.c x
Source History ... _ADCAN0Interrupt ( void )
67     return 1;
68 }
69
70 #if 1 // Moved to PowerController.c
71 #define SysSwitch_OpenLoop 1
72 #define SysConst_PWMPeriod 39992 //100KHz based on 500MHz
73 #define SysConst_Shift99p (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

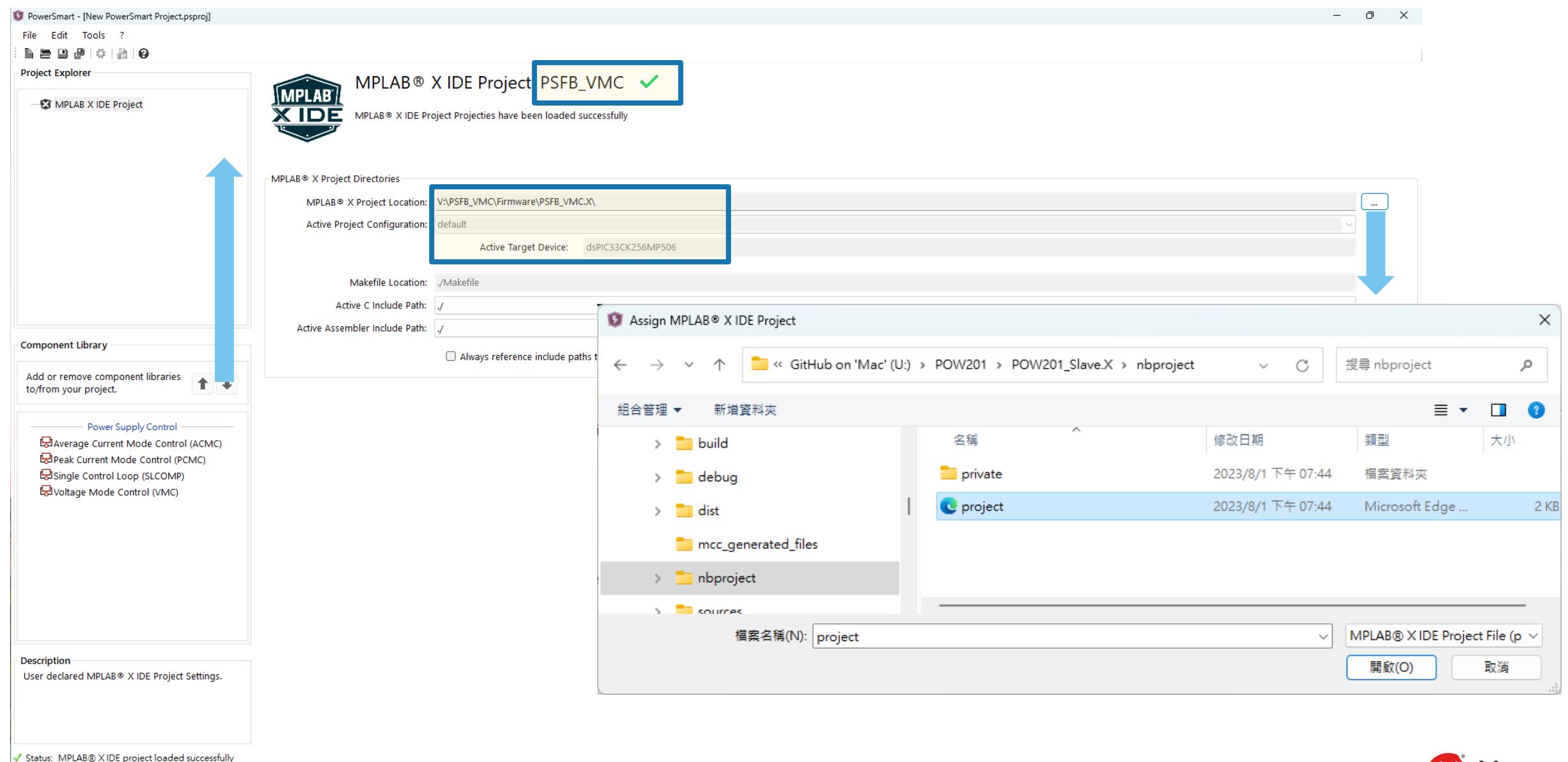


Boundary Test Between 2% ↔ 99% shifted



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) → H(z) → PWM → CLK → Blue Arrow → Feedback from ADC through VREF to Σ

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

File View Tools ?

File & Function Label

Name Prefix: VCOMP

Controller 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

Frequency [Hz] 100 1000 10000 100000

Magnitude/Gain [dB] 60 50 40 30 20 10 0 -10 -20 -30 -40 -50 -60

Phase [°] 180 150 120 90 60 30 0 -30 -60 -90 -120 -150 -180

Gain (z) Gain (s) Phase (s) Pole Locations Zero Locations

Phase (z)

Filter Coefficients

Number Analysis Settings History

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 ::

Status: MPLAB® X IDE project loaded successfully

Microchip Proprietary and Confidential

MICROCHIP

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: Zero 1:

Pole 2: Zero 2:

Source Code Configuration

Development Tools

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

- 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

Nominal Feedback Level Calculator

Voltage Feedback Shunt Amplifier Current Transformer Digital Source

Circuit

Input Scaling

ADC Reference: ADC Resolution: Nominal Sense Voltage:

ADC Minimum: ADC Maximum: R1: R2:

Calculation

Amplifier Gain: Signal Gain:

OK Cancel

Nominal Output Level Calculator

Fixed Frequency Variable Frequency Phase Shifted PWM

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:

OK Cancel

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:

OK Cancel

Setting for C-include Directory

The screenshot shows the PowerSmart Digital Control Library Designer interface with the following details:

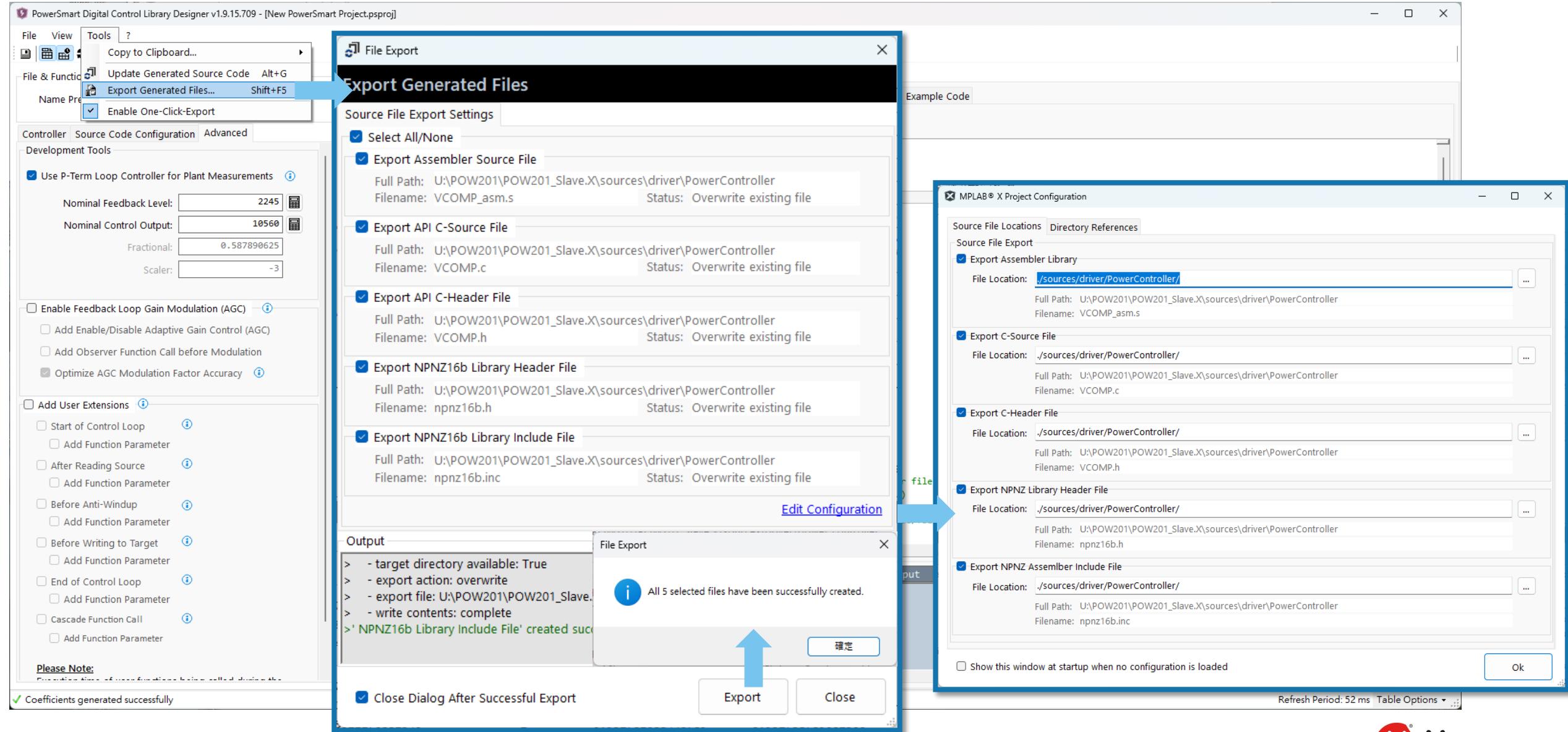
- File & Function Label:** Name Prefix: VCOMP
- Controller:** Use P-Term Loop Controller for Plant Measurements (checked).
 - Nominal Feedback Level: 2245
 - Nominal Control Output: 10560
 - Fractional: 0.587890625
 - Scaler: -3
- Development Tools:**
 - Enable Feedback Loop Gain Modulation (AGC) (unchecked)
 - Add Enable/Disable Adaptive Gain Control (AGC) (unchecked)
 - Add Observer Function Call before Modulation (unchecked)
 - Optimize AGC Modulation Factor Accuracy (checked)
- Source Code Output Tabs:** Frequency Domain, Execution Time, Block Diagram, Source Code Output, Info.
- Assembler Source File Tab:** Contains code comments:

```
1 ; ****
2 ; SDK Version: PowerSmart Digital Control Library Designer v1.9.15.709
3 ; CGS Version: Code Generator Script v3.0.11 (01/06/2022)
4 ; Author: edwardlee
5 ; Date/Time: 08/10/2023 19:36:14
6 ;
7 ; 3P3Z Control Library File (Fast Floating Point Coefficient Scaling Mode)
8 ; ****
9 ;
10 ;
```
- API C-Source File Tab:** Contains code comments:

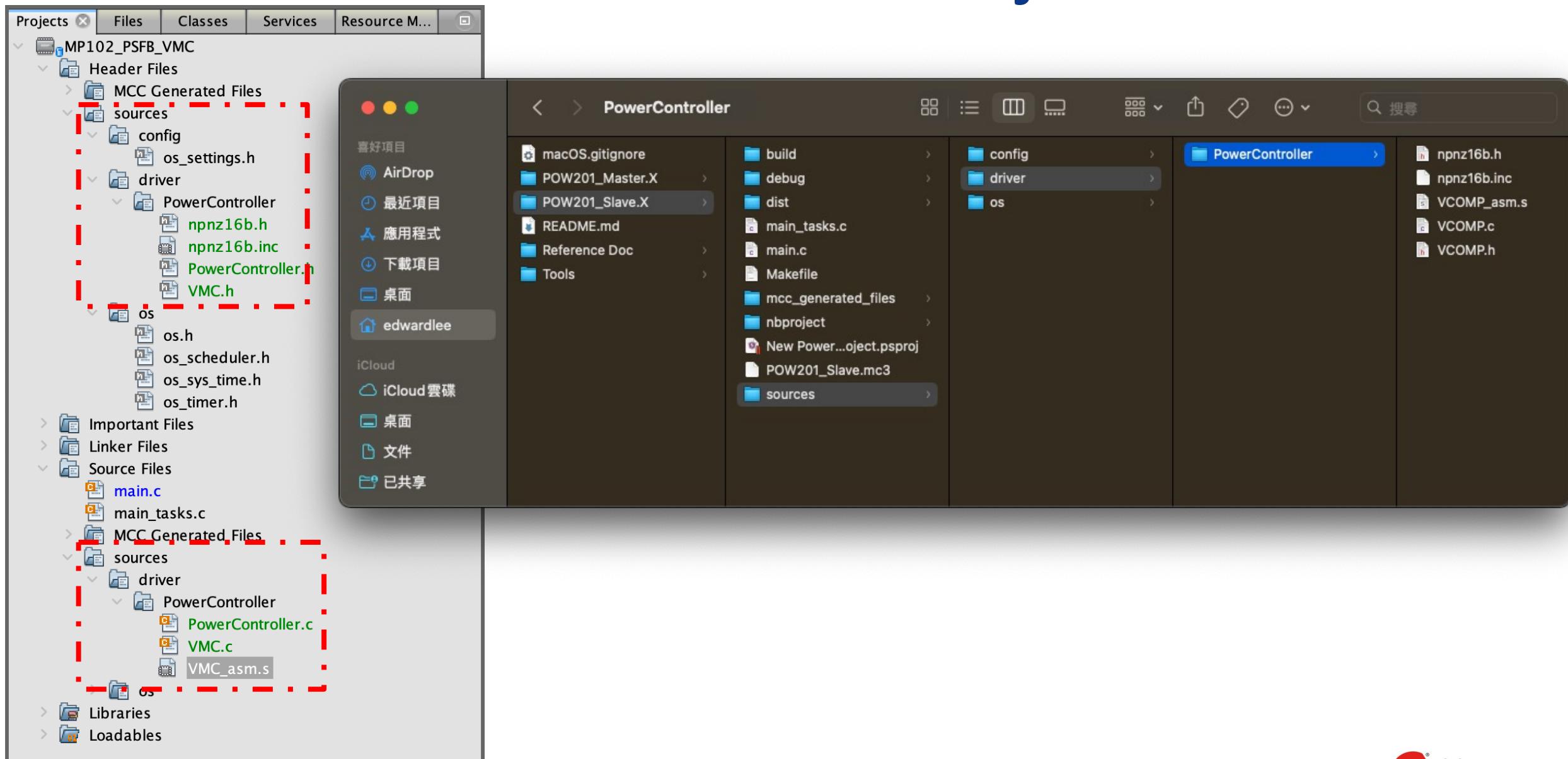
```
17 .SECTION .data
18 ; place constant data in the data section
```
- API C-Header File Tab:** Contains code comments:

```
17 .SECTION .data
18 ; place constant data in the data section
```

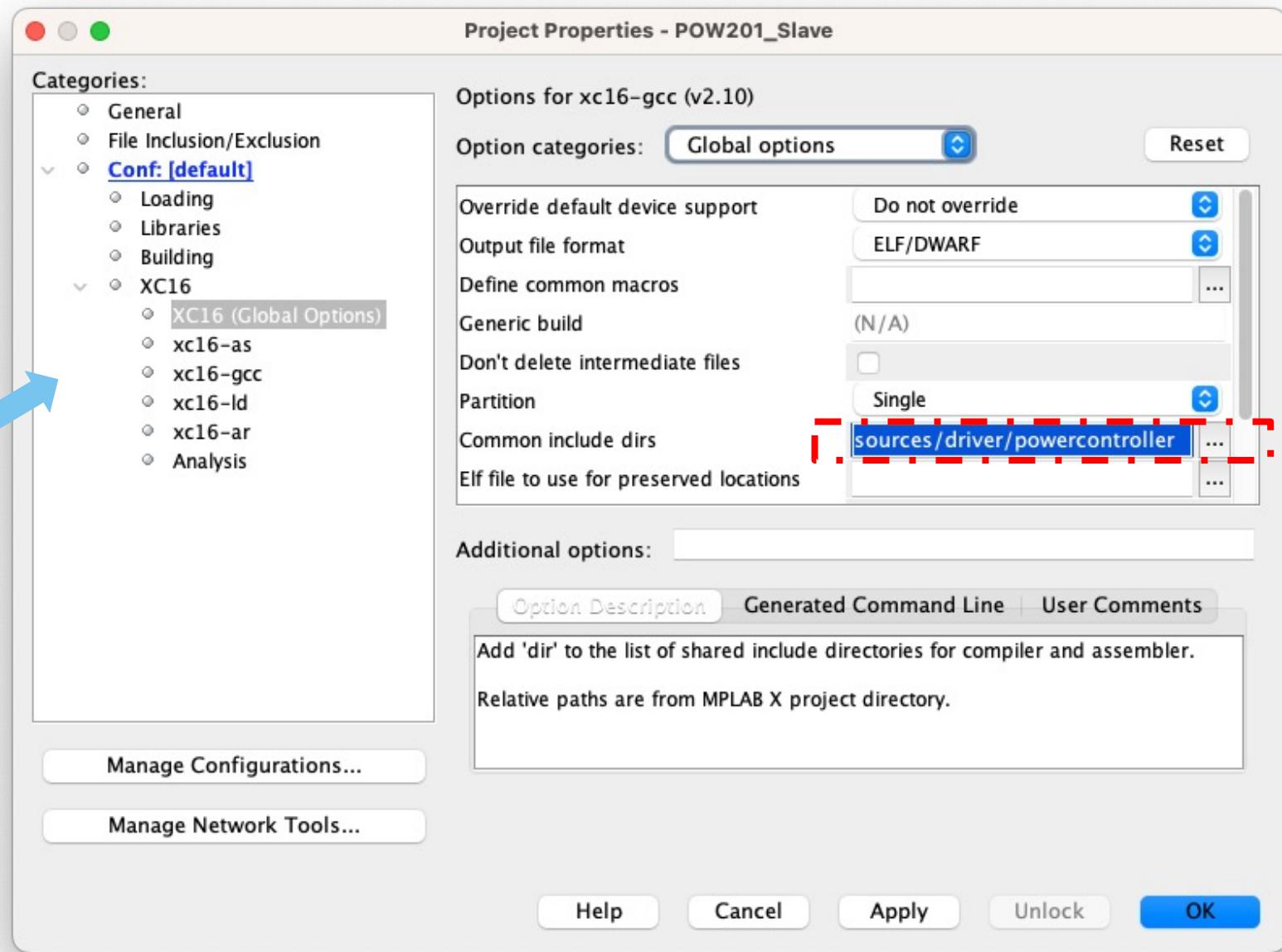
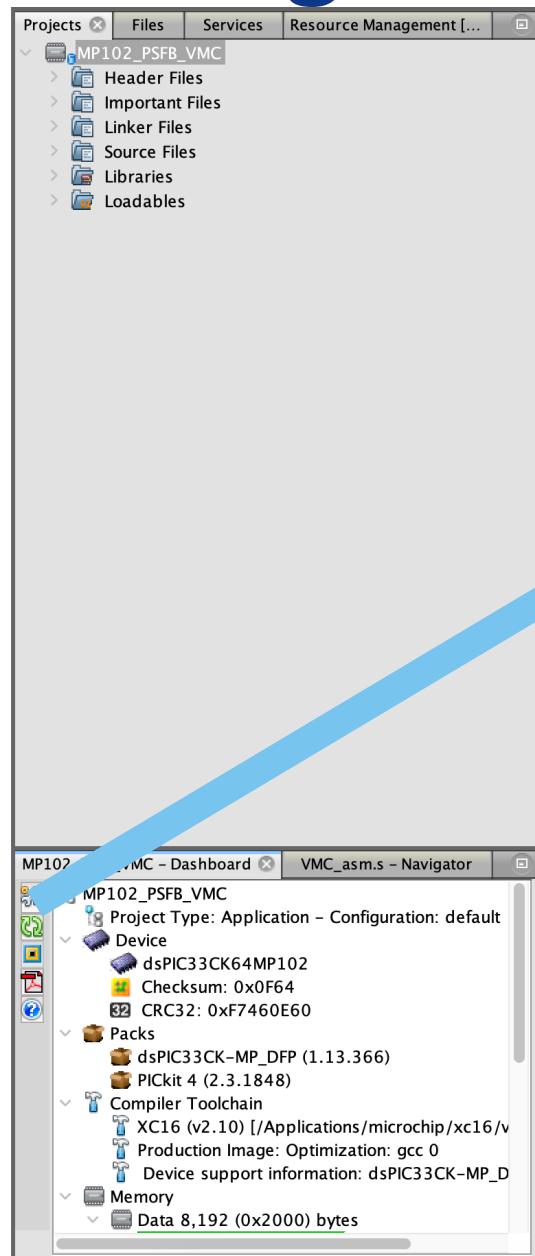
Export Compensator Sources Codes



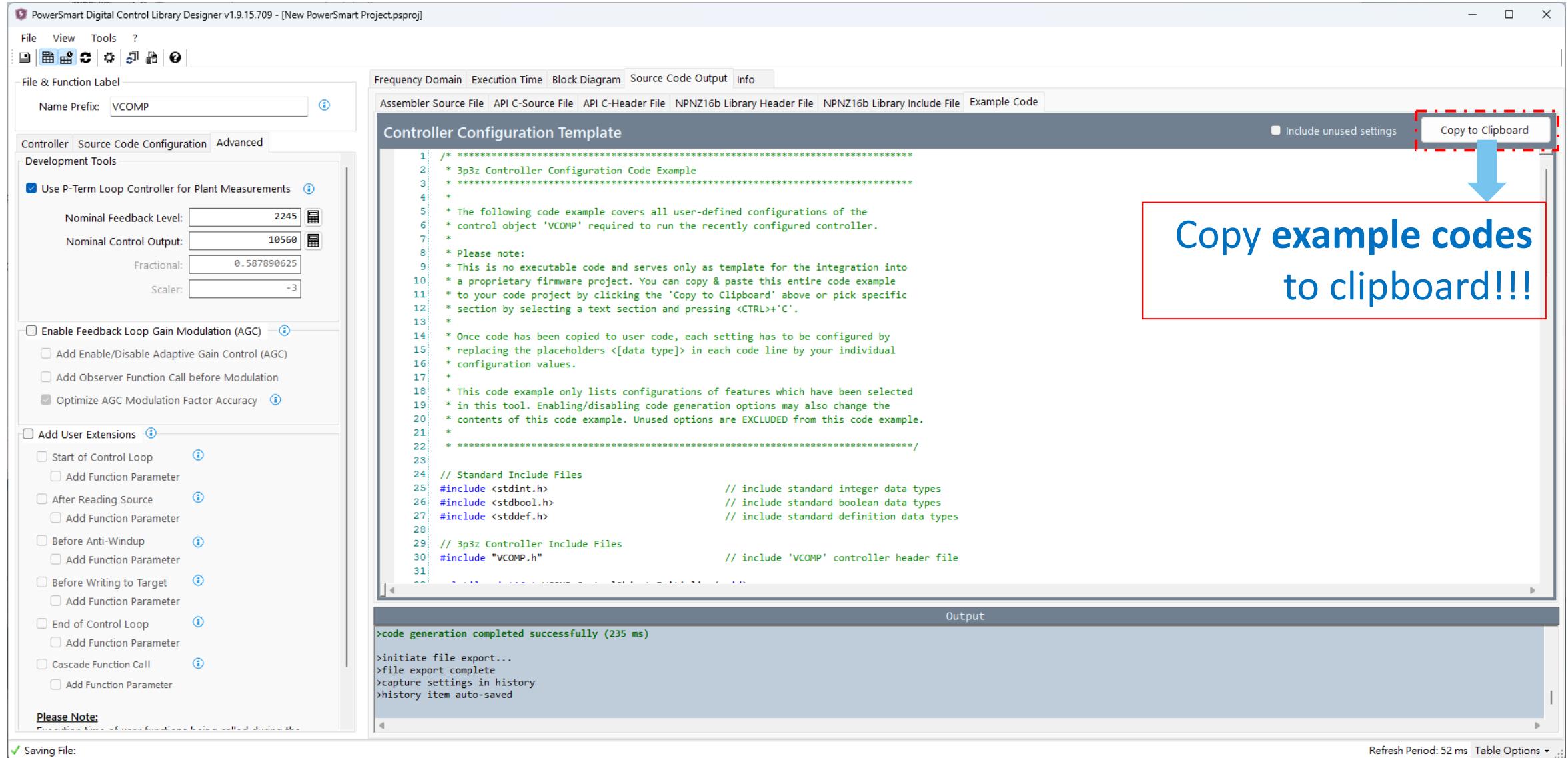
Add Control Libraries to the Project



Setting for C-include Directory



Copy example codes



Copy example codes
to clipboard!!!

New file with pasting example codes!

The screenshot shows the MPLAB X IDE interface with the project "MP102_PSFb_VMC" open. The left pane displays the project structure, and the bottom pane shows the "Dashboard" and "Navigator". The main area contains three code editors:

- PowerController.h**: A header file containing defines for the PowerController module, including #defines for SysSwitch_OpenLoop, SysConst_PWMPeriod, SysConst_Shift99p, SysConst_Shift30p, and SysConst_Shift02p.
- PowerController.c**: A source file containing interrupt service routines for ADCAN0Interrupt and ADCAN1Interrupt. It includes pre-compiler directives for interrupt labels and handles PWM boundary jumping tests.
- Main application**: A large source file containing the main() function. It initializes the VCOMP controller, the device, and the OS, then enters a loop with application-specific code.

Red dashed boxes highlight the code in the PowerController.h and PowerController.c files, indicating they are being copied or pasted. A large blue arrow points from the PowerController.h code towards the Main application code, illustrating the process of transferring code between files. The word "Pasting" is overlaid in red text on the left side of the PowerController.c editor.

```
#ifndef __PowerController
#define EXTERN
#else
#define EXTERN extern
#endif

#define SysSwitch_OpenLoop 1 // 0:Disable Open-loop control 1:Enable Open-loop control

// PWM Module
#define SysConst_PWMPeriod 39992 //100KHz based on 500MHz
#define SysConst_Shift99p (SysConst_PWMPeriod /2 * 0.99)
#define SysConst_Shift30p (SysConst_PWMPeriod /2 * 0.30)
#define SysConst_Shift02p (SysConst_PWMPeriod /2 * 0.02)

/* pre-compiler directive declaration to your code, like the following example:
 *
 * #define _VCOMP_Interrupt _PWM1Interrupt // Define label for interrupt service routine
 * #define _VCOMP_ISRIF _PWM1IF // Define label for interrupt flag bit
 */
 */

void __attribute__ ( ( __interrupt__ , auto_psv ) ) _ADCAN0Interrupt ( void )
{
    #if (SysSwitch_OpenLoop == 1)
        //Read the ADC value from the ADCBUF
        uint16_t valchannel_AN0;
        valchannel_AN0 = ADCBUF0;
        // For PWM Boundary Jumping Test
        if(PWMUpdateCount <= 2)
        {
            //0~2
            PhaseShifted = SysConst_Shift99p;
            SRdelay = 0x320;
        }
        else
        {
            //3~5
            PhaseShifted = SysConst_Shift02p;
            SRdelay = 0x320;
        }
        PG1TRIGA = PhaseShifted;
        //PG2TRIGA = SRdelay;
        if(PWMUpdateCount >= 5) PWMUpdateCount = 0;
        else PWMUpdateCount++;
    #else
        uint16_t valchannel_AN0;
        //Read the ADC value from the ADCBUF
        valchannel_AN0 = ADCBUF0;
        #if (SysSwitch_OpenLoop == 1)
            // For PWM Boundary Test
            if(PWMUpdateCount <= 2)
            {
                // ...
            }
        #endif
    #endif
}

int main(void)
{
    // initialize VCOMP controller
    VMC_ControlObject_Initialize();

    // initialize the device
    SYSTEM_Initialize();

    // OS
    OS_Init();
    OS_Scheduler_RunForever();

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

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 `main.c`.

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 ( )
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 (usually VCOMPUF0)
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 and memory
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
```

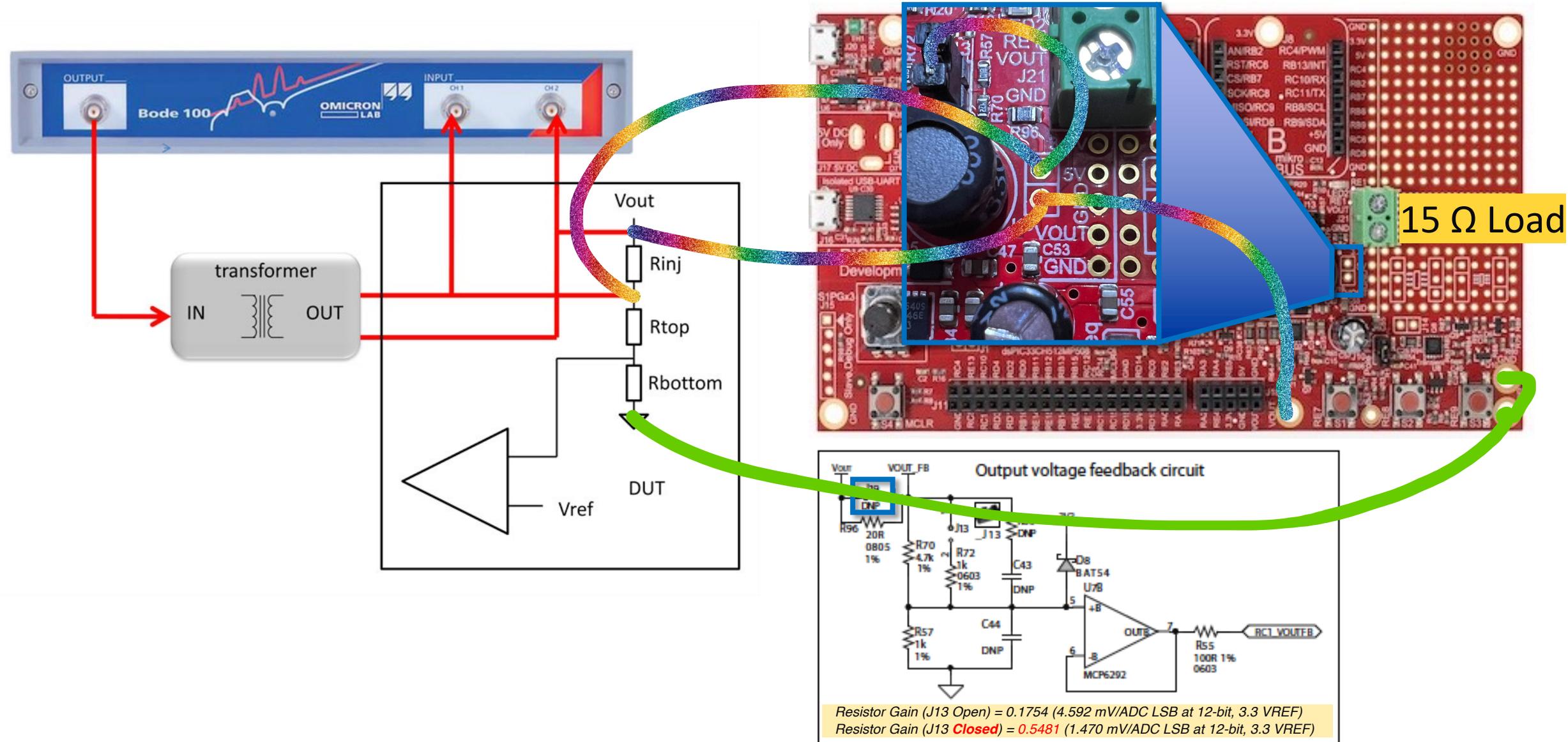
Control Loop

The screenshot shows a software development environment with a window titled "PowerController.c". The code is written in assembly language with some C-like syntax for interrupt service routines. The code is annotated with comments explaining its purpose, such as "3p3z Control Loop Interrupt Service Routine for Controller Object 'VCOMP'" and "Using custom labels for interrupt routines allows using generic interrupt service function calls in code, which can be mapped to specific interrupt sources by adding pre-compiler directive declaration to your code, like the following example". The code handles ADC conversion results and updates PWM update counts. A specific section of the code, starting at line 126 and ending at line 129, is highlighted with a blue rounded rectangle. This section contains code to call the VMC control loop and P-Term control loop. The code ends with a "#endif" directive.

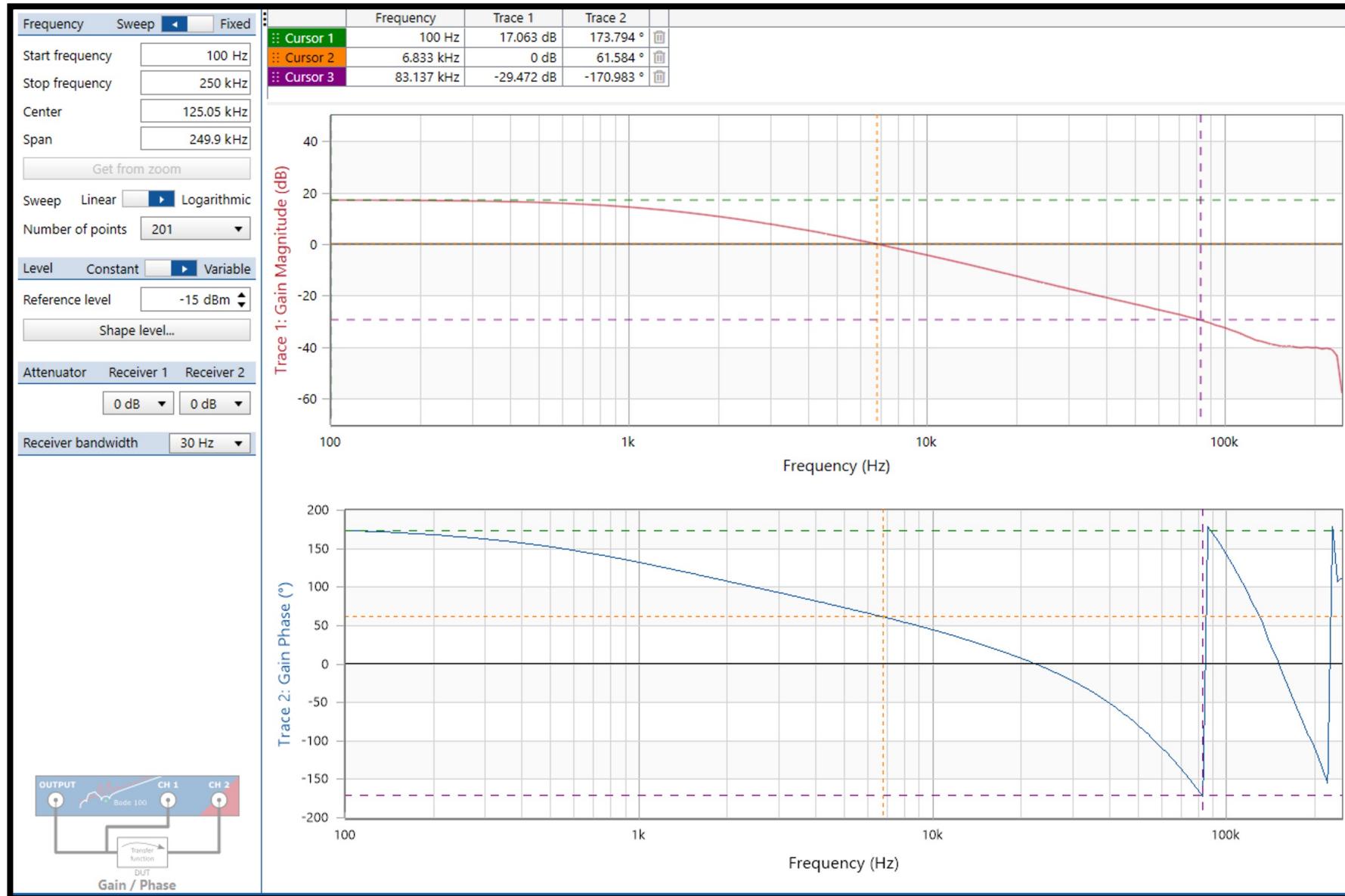
```
87
88  /* 3p3z Control Loop Interrupt Service Routine for Controller Object 'VCOMP'
89  ****
90  * This code example of a interrupt service routine uses the tailored name label '_VC
91  * The Assembler library code sequences of controller data objects generated by PS-DC
92  * for being called by a PWM interrupt for minimum response time. However, in some ap
93  * it might be desired to call the control loop from other interrupt sources.
94  * Using custom labels for interrupt routines allows using generic interrupt service
95  * function calls in code, which can be mapped to specific interrupt sources by addin
96  * pre-compiler directive declaration to your code, like the following example:
97  *
98  * #define _VCOMP_Interrupt      _PWM1Interrupt // Define label for interrupt servi
99  * #define _VCOMP_ISRIF         _PWM1IF        // Define label for interrupt flag
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     //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

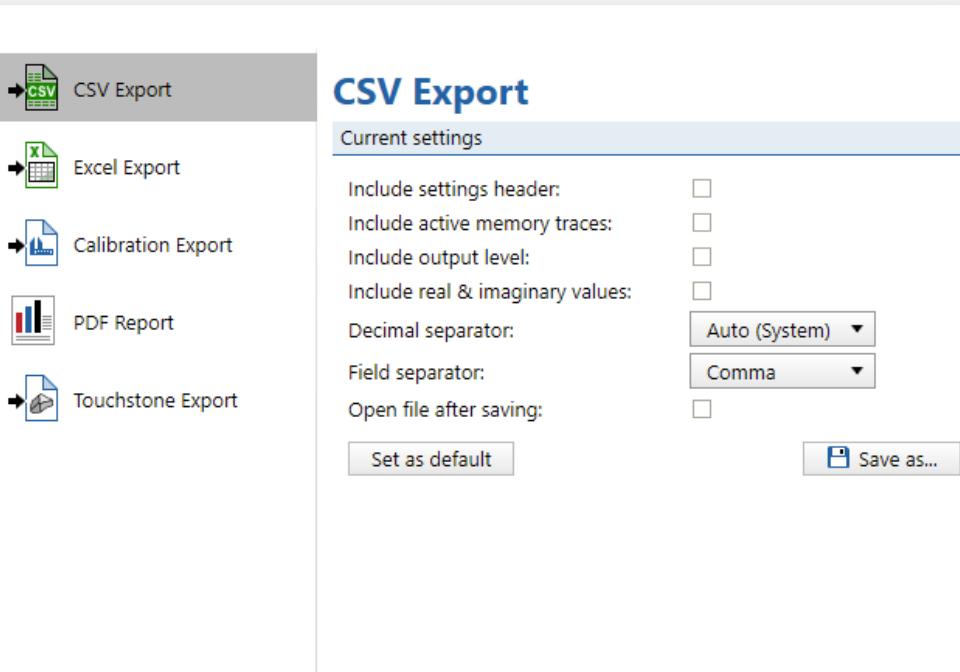
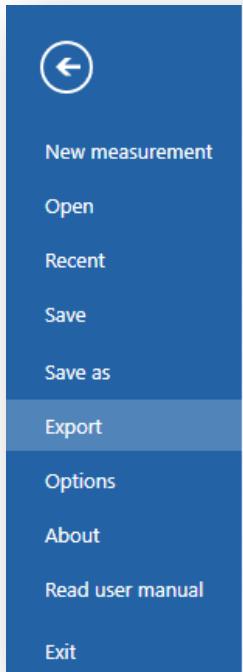
Plant Measurement Using Bode-100



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 title is '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 the 'File Import' tab, there is a 'Data Source' section with a 'Filename' field set to 'U:\POW201\Tools\Bode100\POW201_Lab3_2023-08-12T18_12_16.csv', a 'Series Header Row' of '1', a 'Data Start Row' of '2', and a 'Column Separator' of '<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.

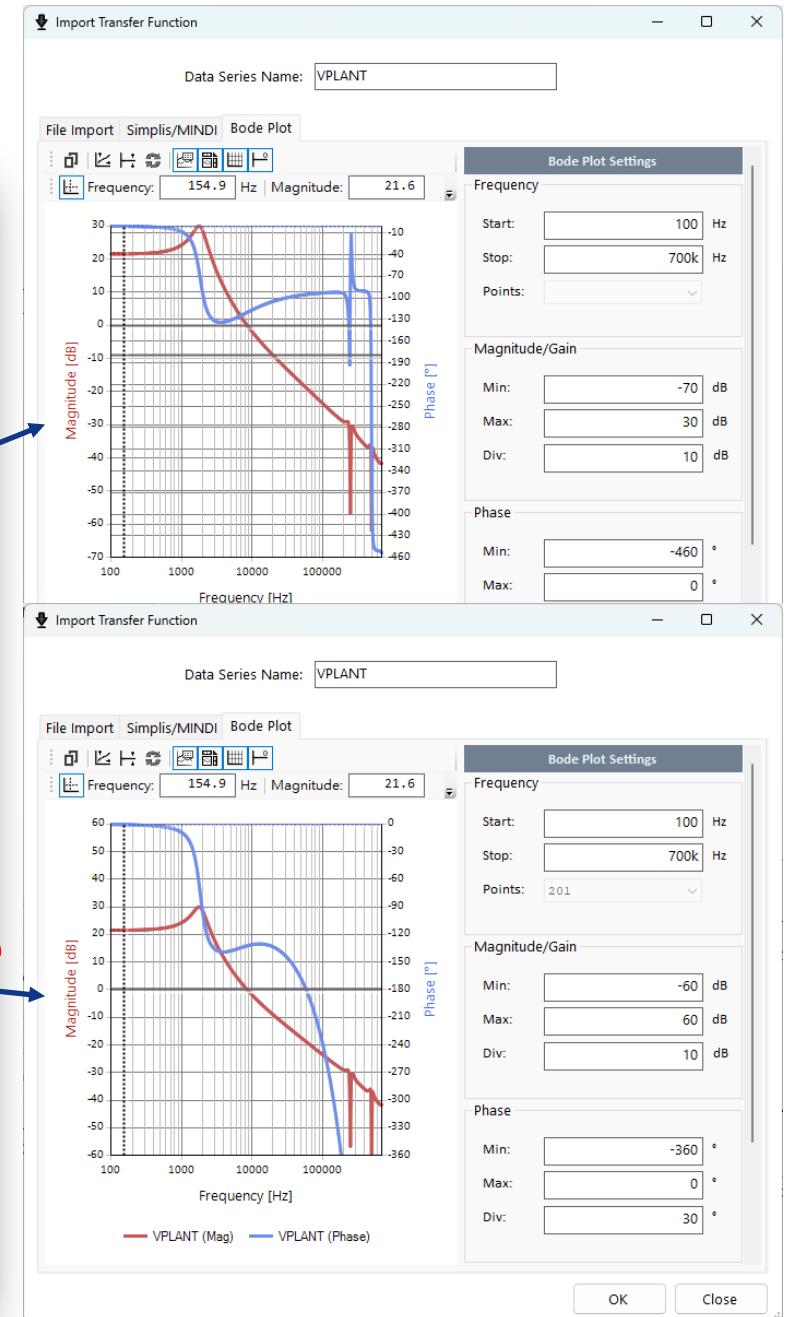
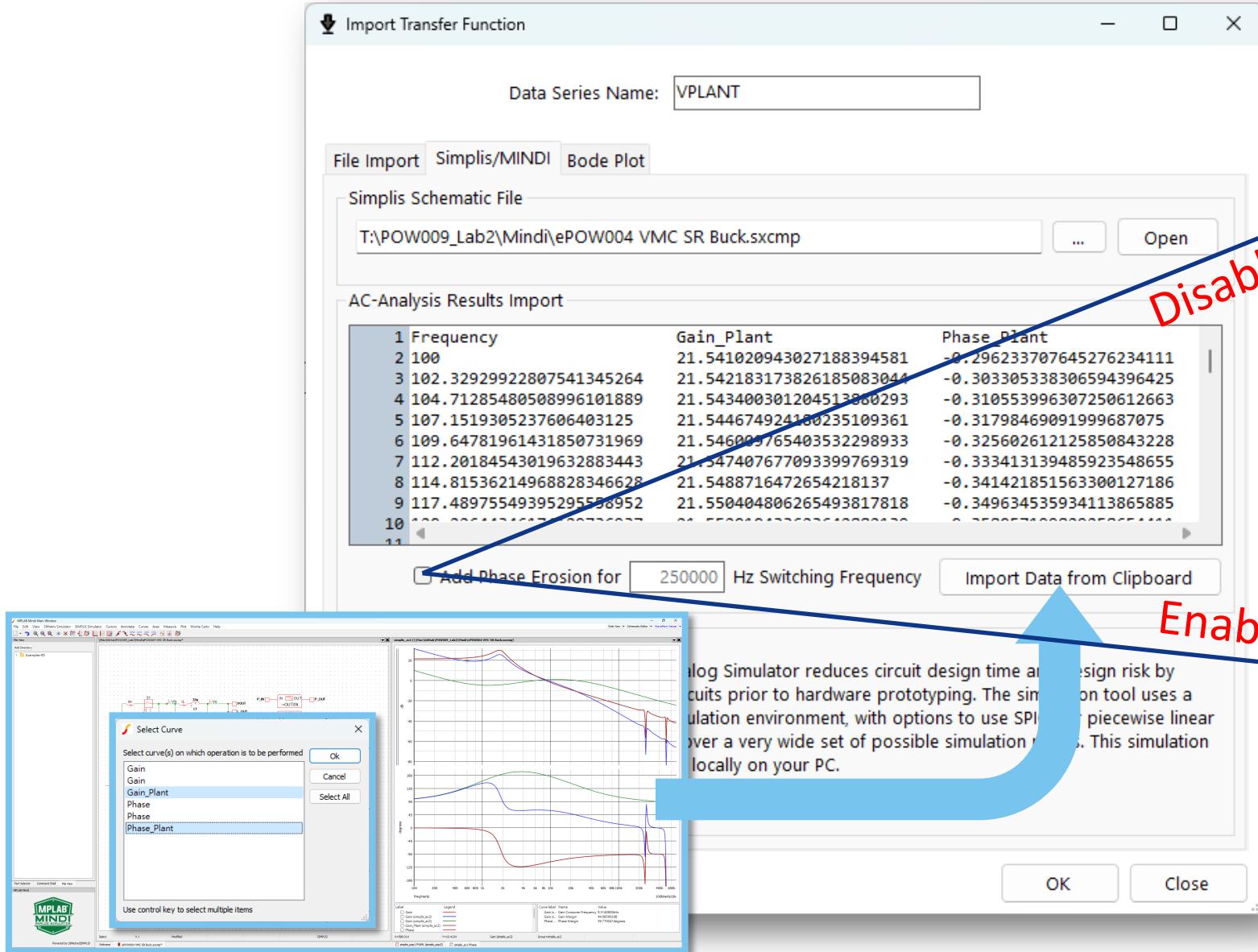
To the right of the dialog, there is a block diagram window titled 'C)'. It contains a block labeled 'Converter Voltage Plant G(s)' with 'VIN' at the top and 'VOUT' at the bottom. A blue arrow points from the 'VOUT' terminal of this block to the 'VOUT' terminal of a 'Plant Gain' block. Below the 'Plant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



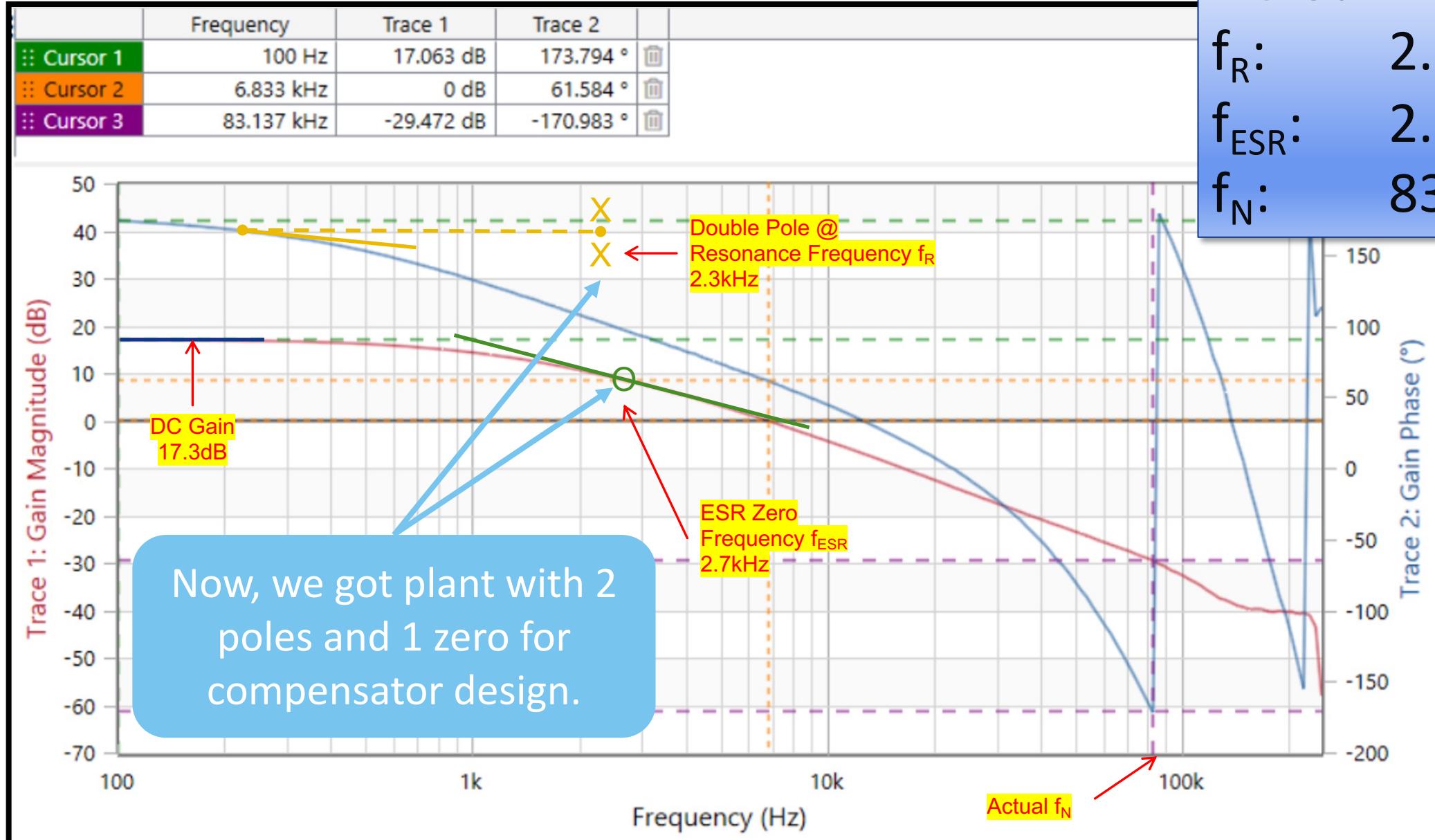
Design Tip



Closed-loop Control

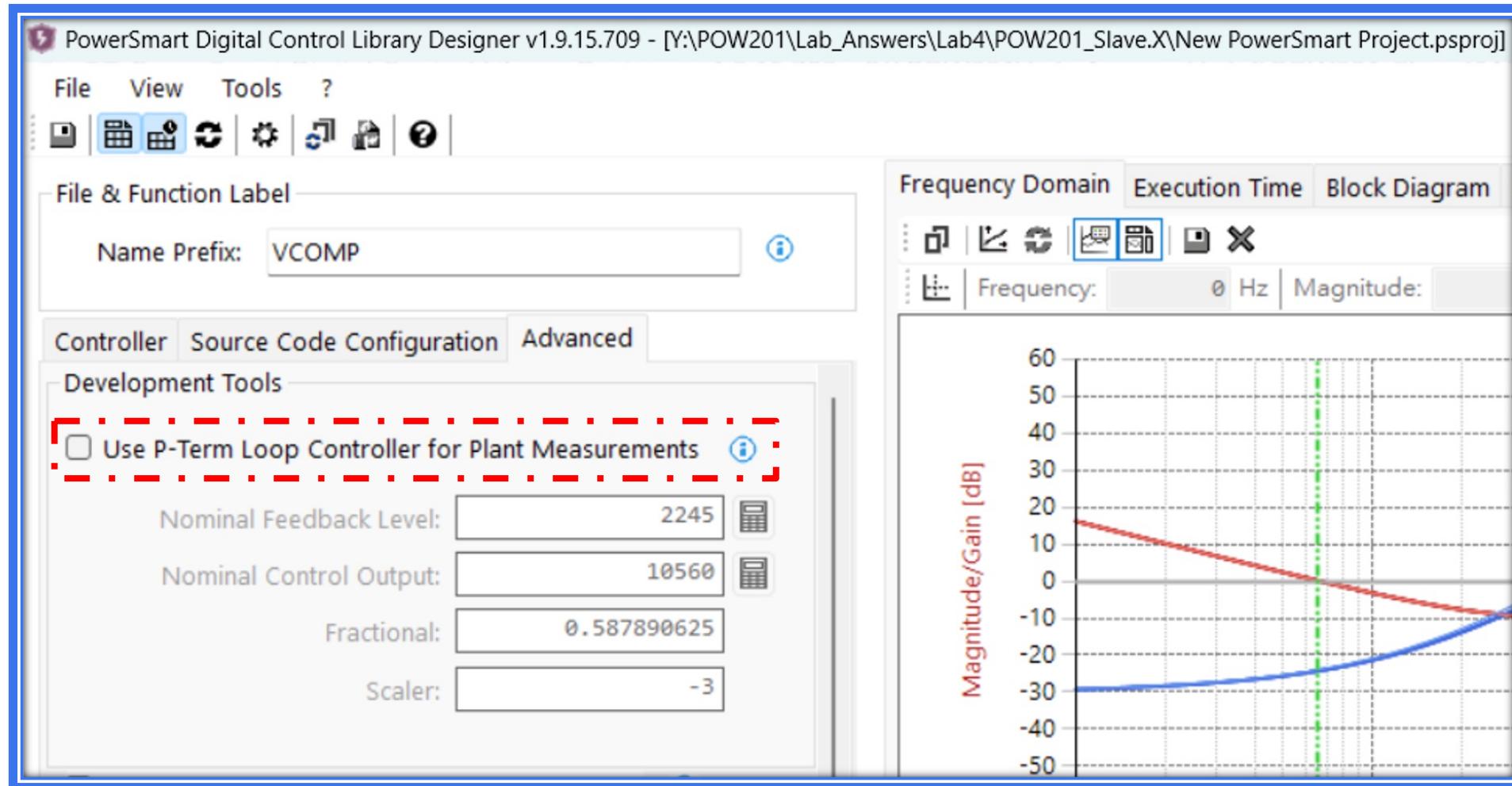
Lab4

The Actual Plant

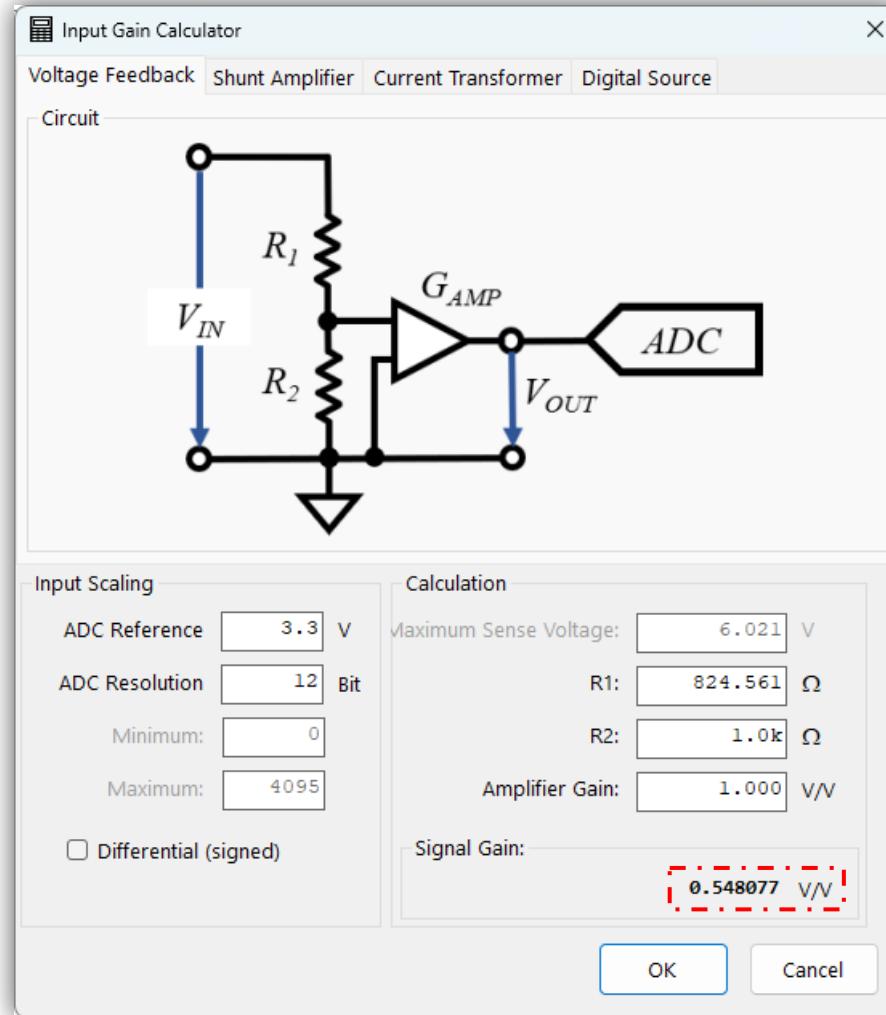
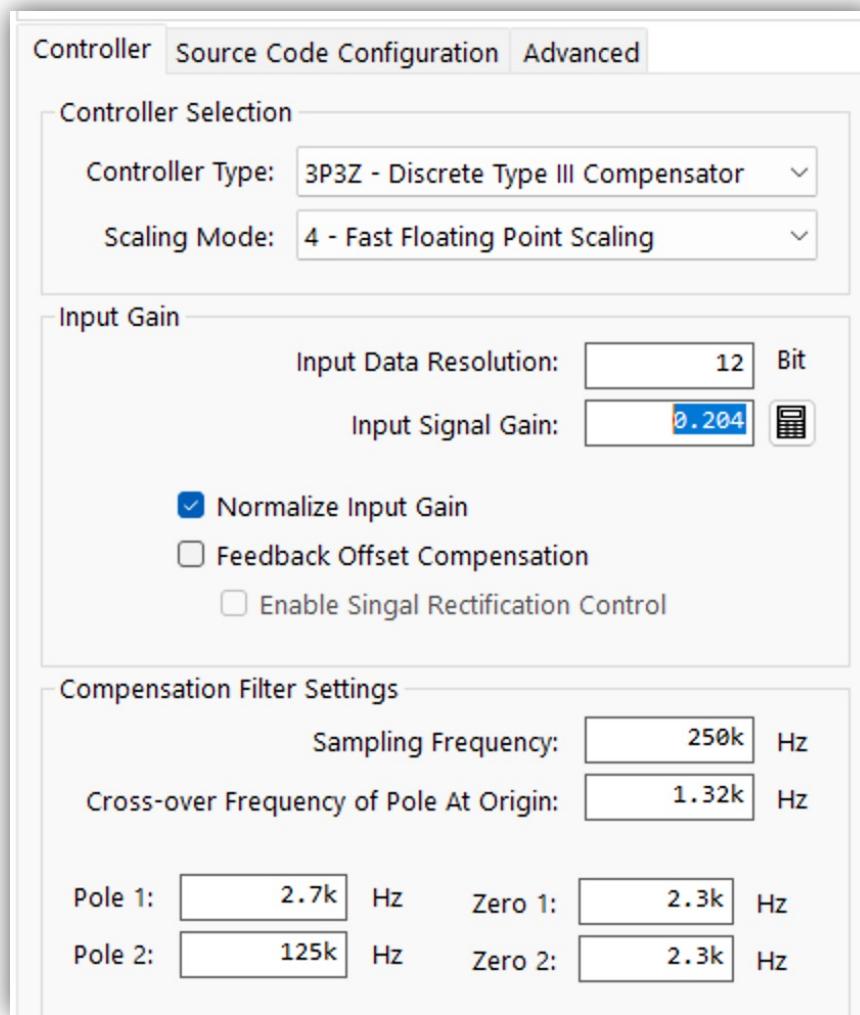


Uncheck Kp (P-Term) Control on DCLD

- **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

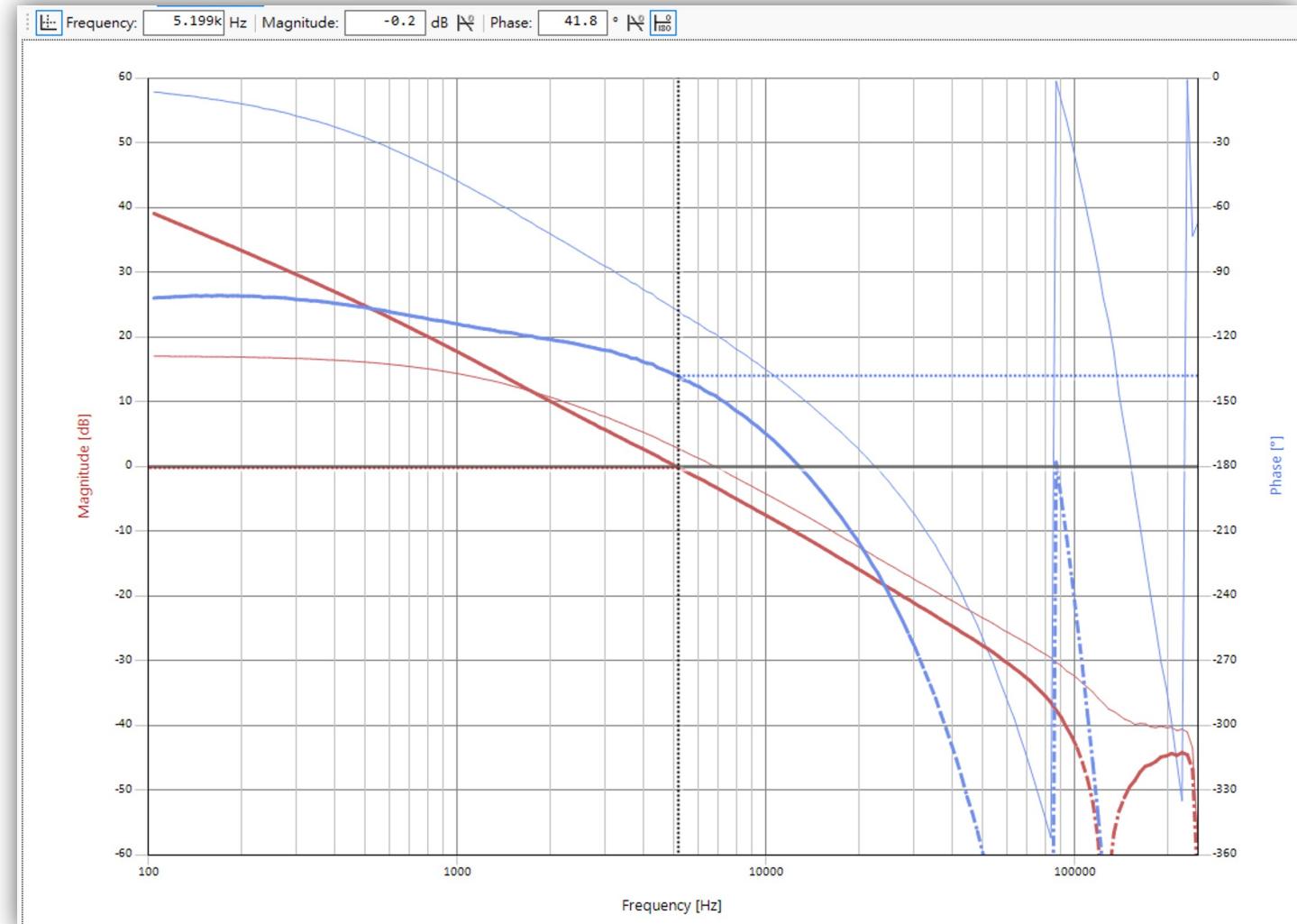
Normalize Input Gain
 Feedback Offset Compensation
 Enable Singal Rectification Control

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

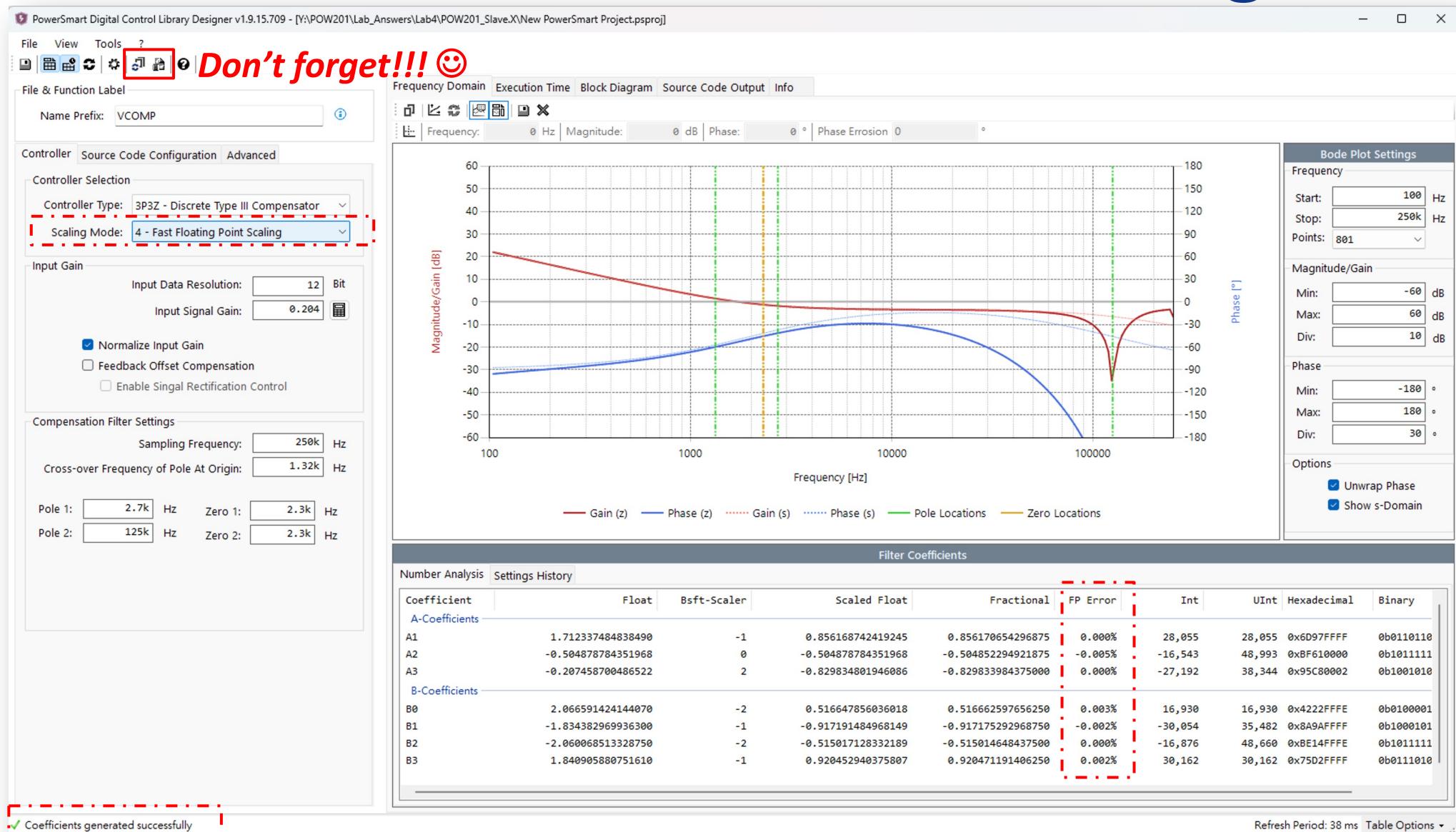
Observe Open Loop Gain



DCLD

PowerSmart™
Microchip Proprietary and Confidential

Check Coefficients with Suitable Scaling Mode



Closed-loop Control

Switch to closed-loop control in ANx ISR

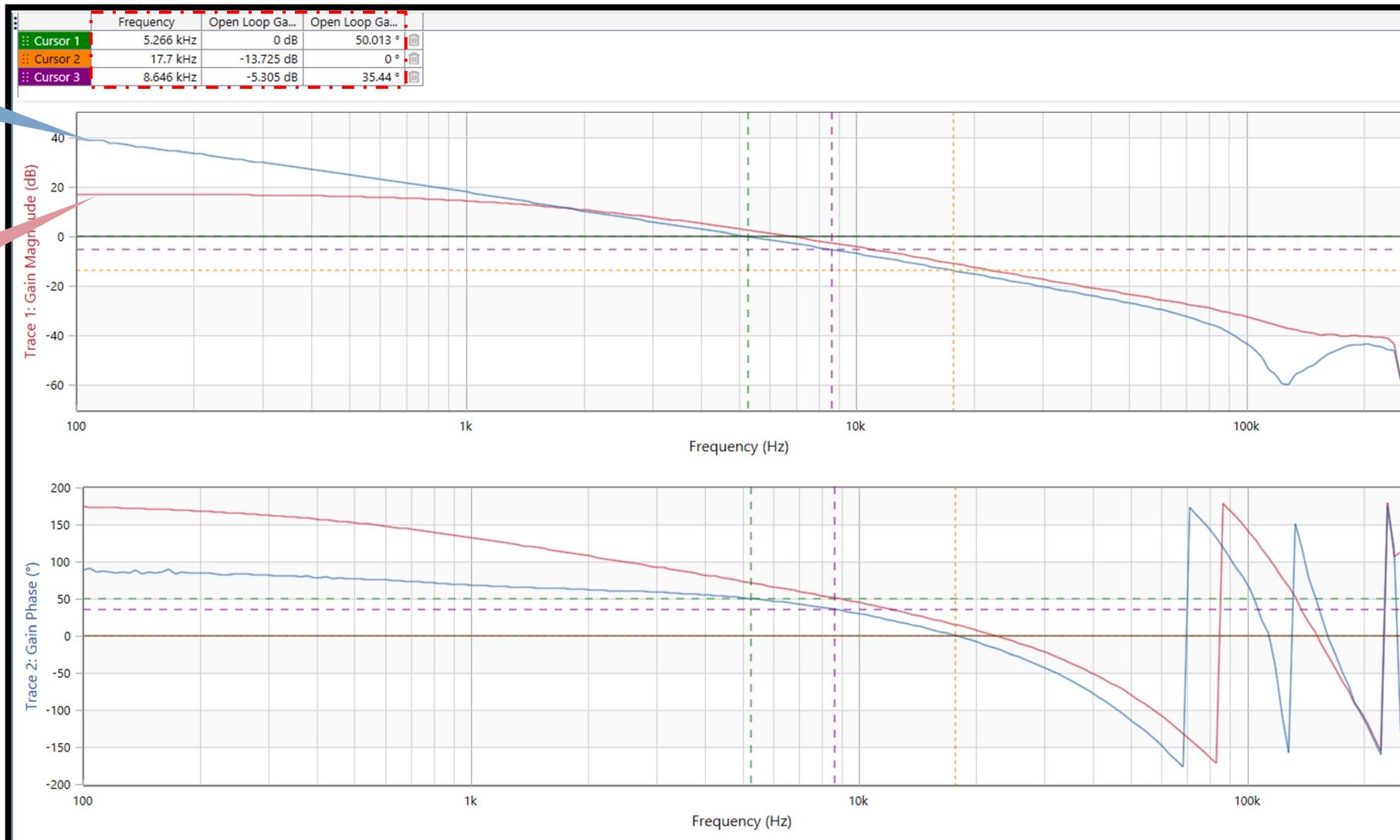
```
Projects x Files Services Classes BuckConverter.c x
Source History
96 * #define _VCOMP_INTERRUPT _PWM1Interrupt // Define label for interrupt service
97 * #define _VCOMP_ISRIF _PWM1IF // Define label for interrupt flag b
98 *
99 ****
100
101 void __attribute__ (( __interrupt__ , auto_psv )) _ADCAN1Interrupt ( 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

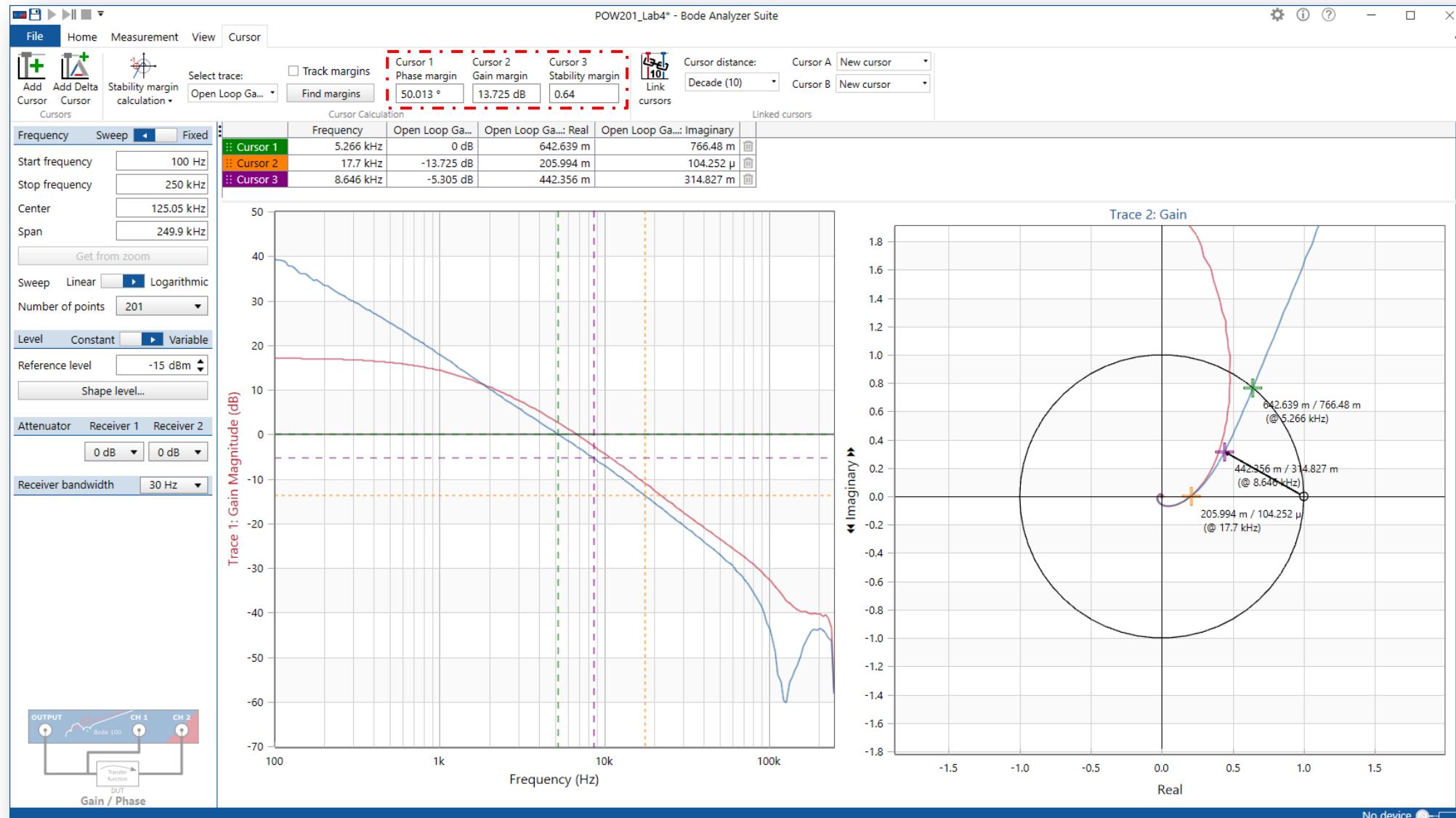
OL TF

Plant



Closed-loop Control

Stability Margin & Nyquist Plot



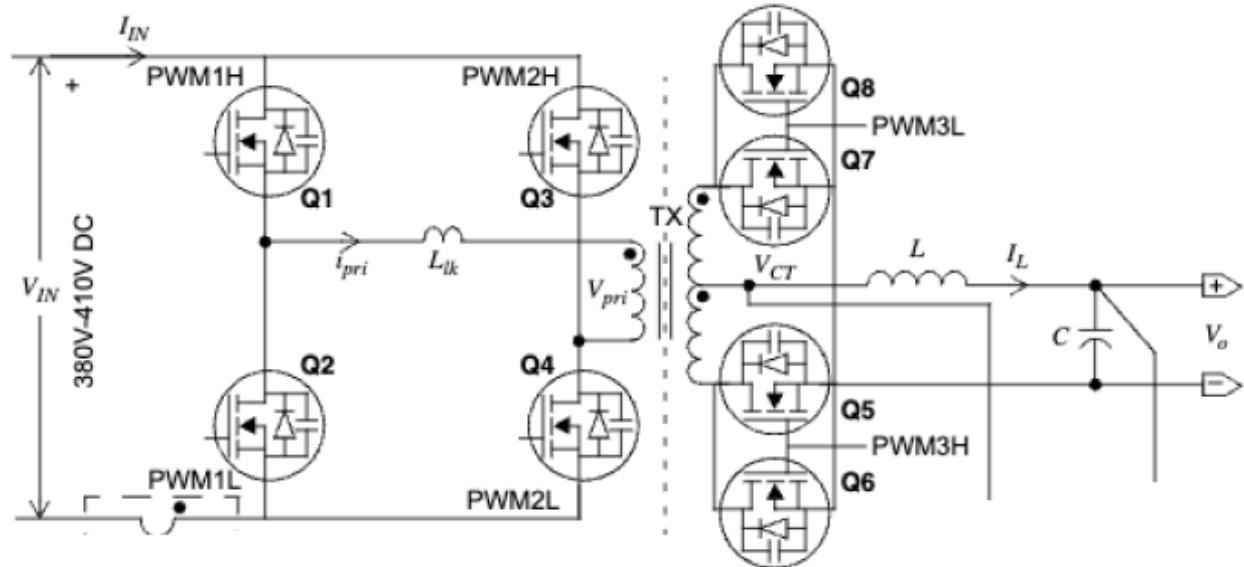
**PCMC PSFB Converter
With SR
(PWM2 SOC from
Trigger A of PWM1)**

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 -> AN0 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



Peripheral & Tool Descriptions

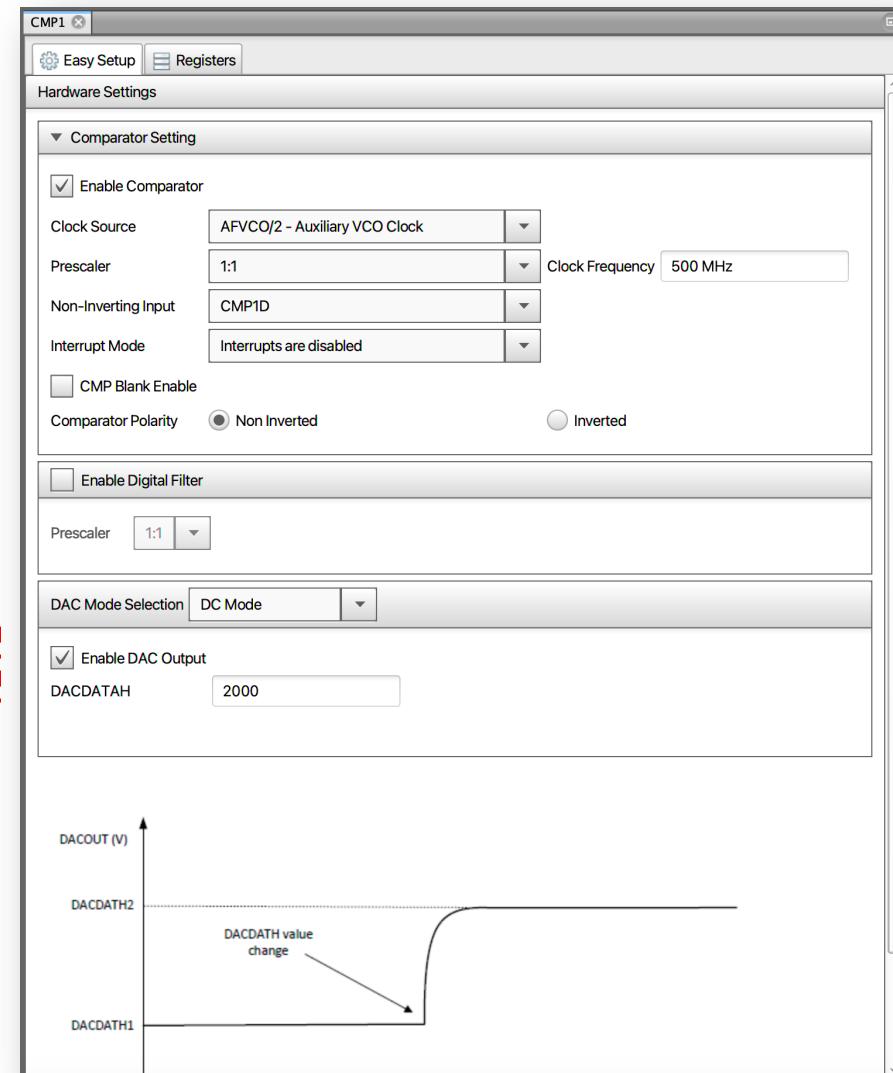
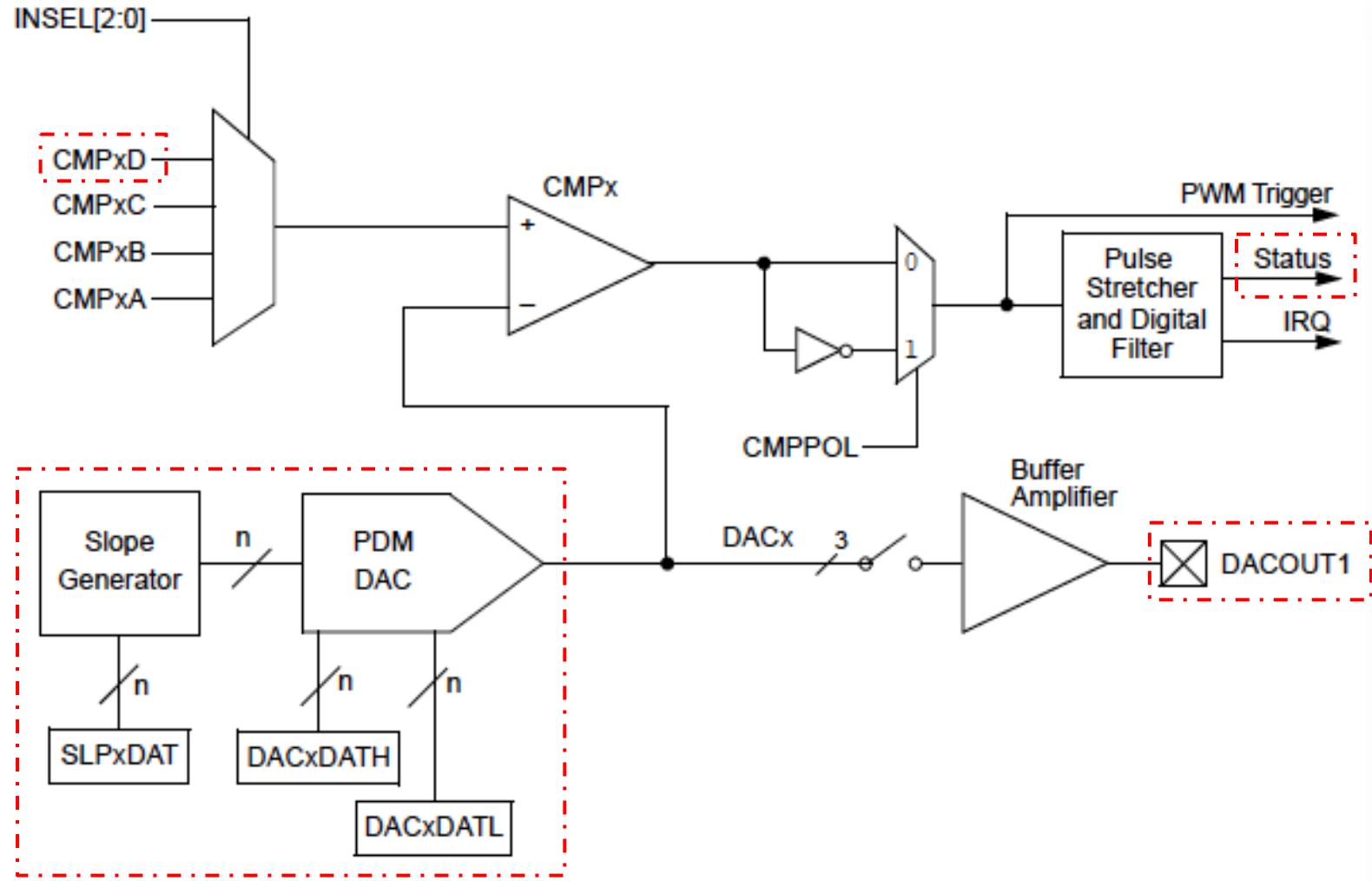
- Peripherals (in RED)

- Comparator 1
 - CMP1D – Pin15 (RB2)
 - DAC without slope for pre-testing
- CLC3
 - PWM3H (CLCINA) // PWM3L (CLCINB)
 - CLC3 O/P on Pin13(RB0) for checking
- CLC2
 - PWM1L (CLCIND) && CMP1 O/P
 - CLC2 O/P on Pin22(RB9) for checking
- CLC1
 - CMP1D OR'd PWM1H (CLCINC) Inverted
 - CLC1 O/P on Pin21(RB8) for checking

Peripheral & Tool Descriptions

- Peripherals (in RED)
 - Comparator 1
 - CMP1D – Pin15 (RB2)
 - DAC without slope for pre-testing

Comparator 1 (CMP1) with DAC



CLC3 for Duty Cycle TX simulation

- Peripherals (in RED)

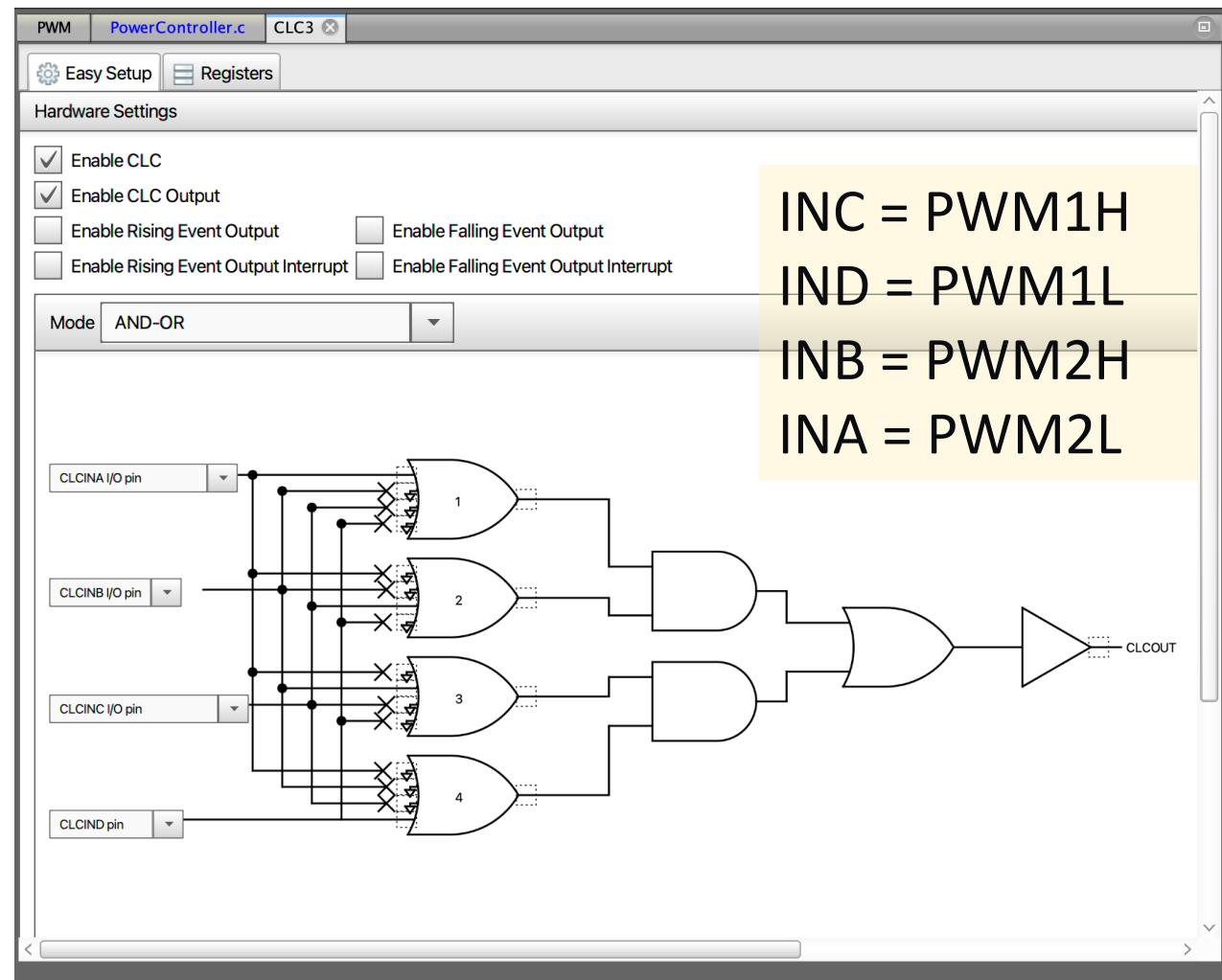
- Comparator 1

- CMP1D – Pin15 (RB2)
 - DAC without slope for pre-testing

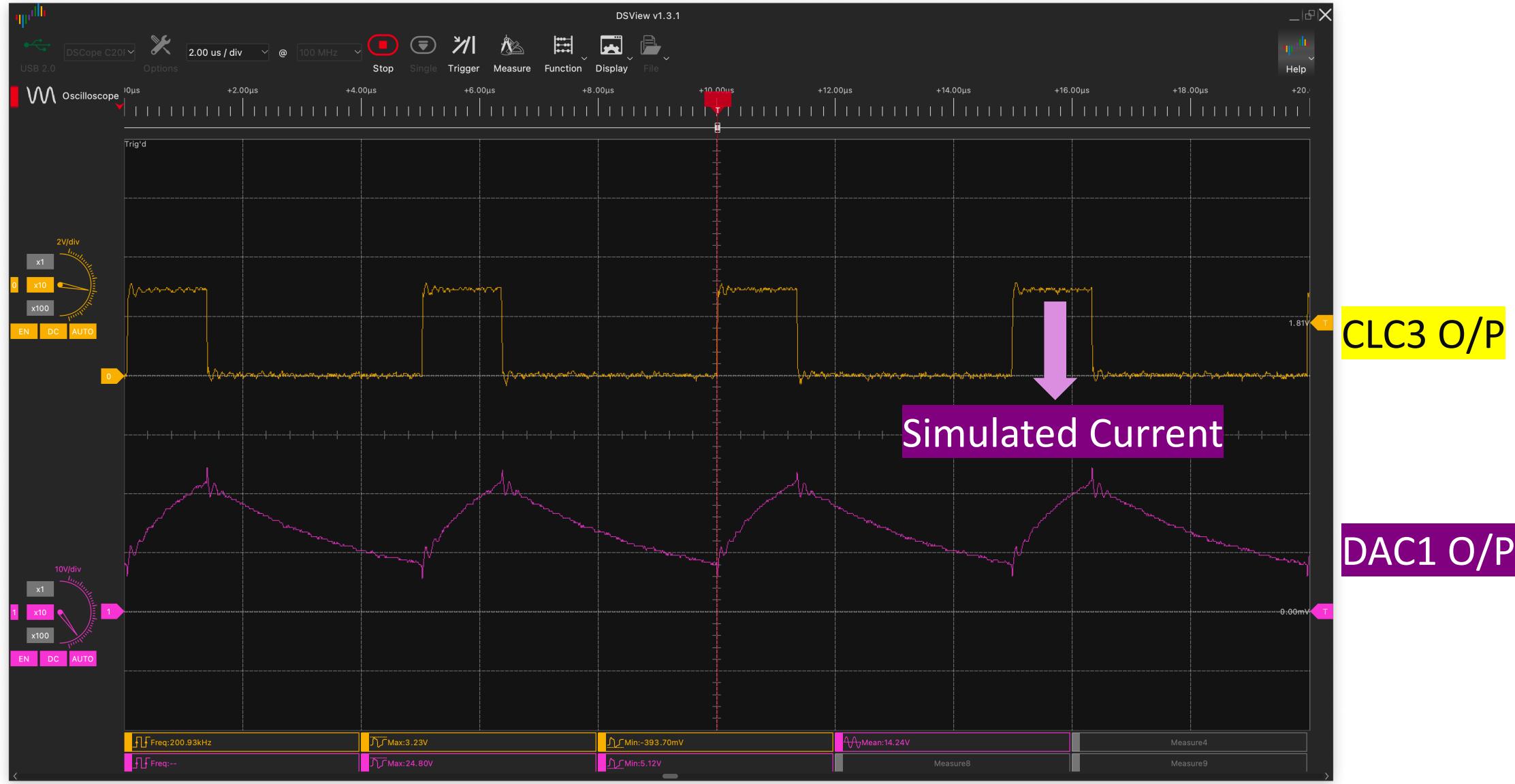
- CLC3

- PWM3H (CLCINA) && PWM3L (CLCINB)
 - CLC3 O/P on Pin13(RB0) for checking

			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
ADC1 ▾	ADCTRGS1	input																					
	ANx	input																					
CLC3 ▾	CLC3OUT	output																					
	CLCINA	input																					
	CLCINB	input																					
	CLCINC	input																					
	CLCIND	input																					
	CMP1	output																					



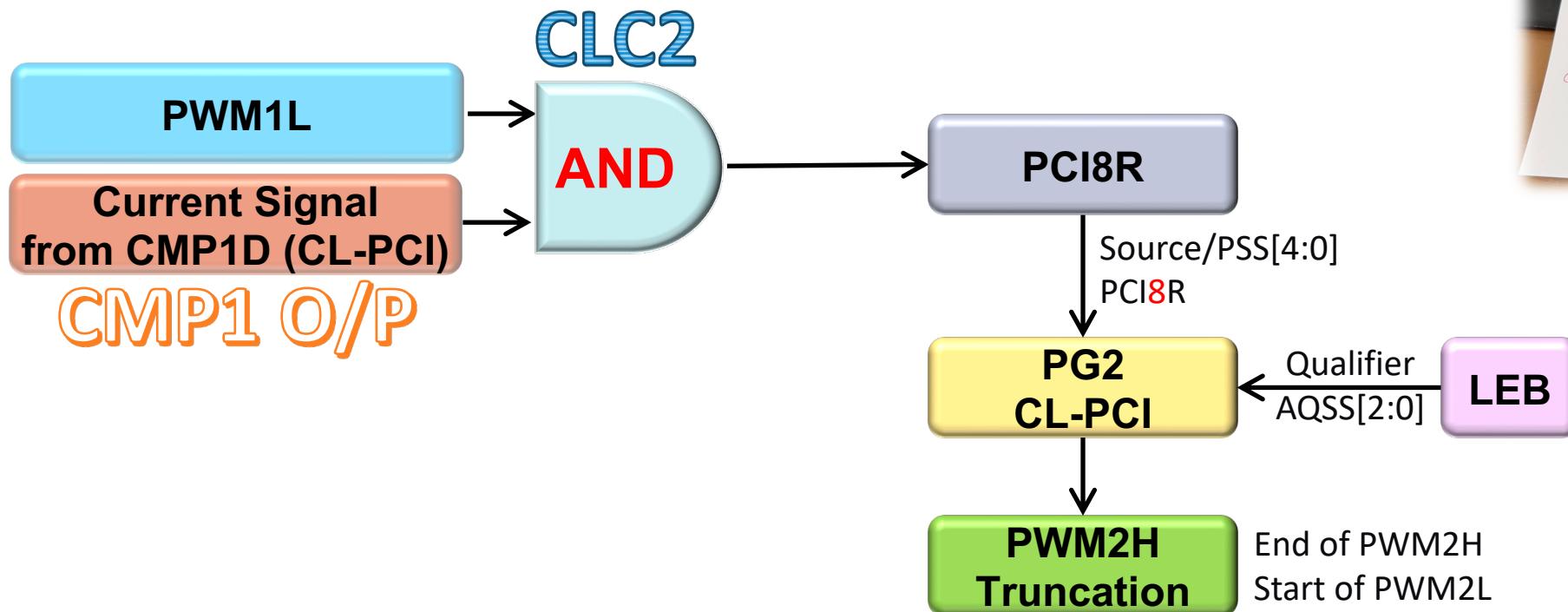
CLC3 for Duty Cycle TX simulation



Design Idea

- PWM2H Truncation (Current Limit-PCI to truncate PWM2H)

- Peripherals
 - CMP1-D/DAC with enabling DACOUT function
 - CLC2



A handwritten diagram on a piece of paper showing the flow of data from a keyboard to a monitor. The flow starts at the keyboard, goes through a buffer (BUF), then to CPU1 (CPU1), then to PCI (PCI). From PCI, the data goes to RAM (RAM), then to CPU2 (CPU2), then to PCI (PCI). Finally, the data is sent to a monitor (MONITOR).

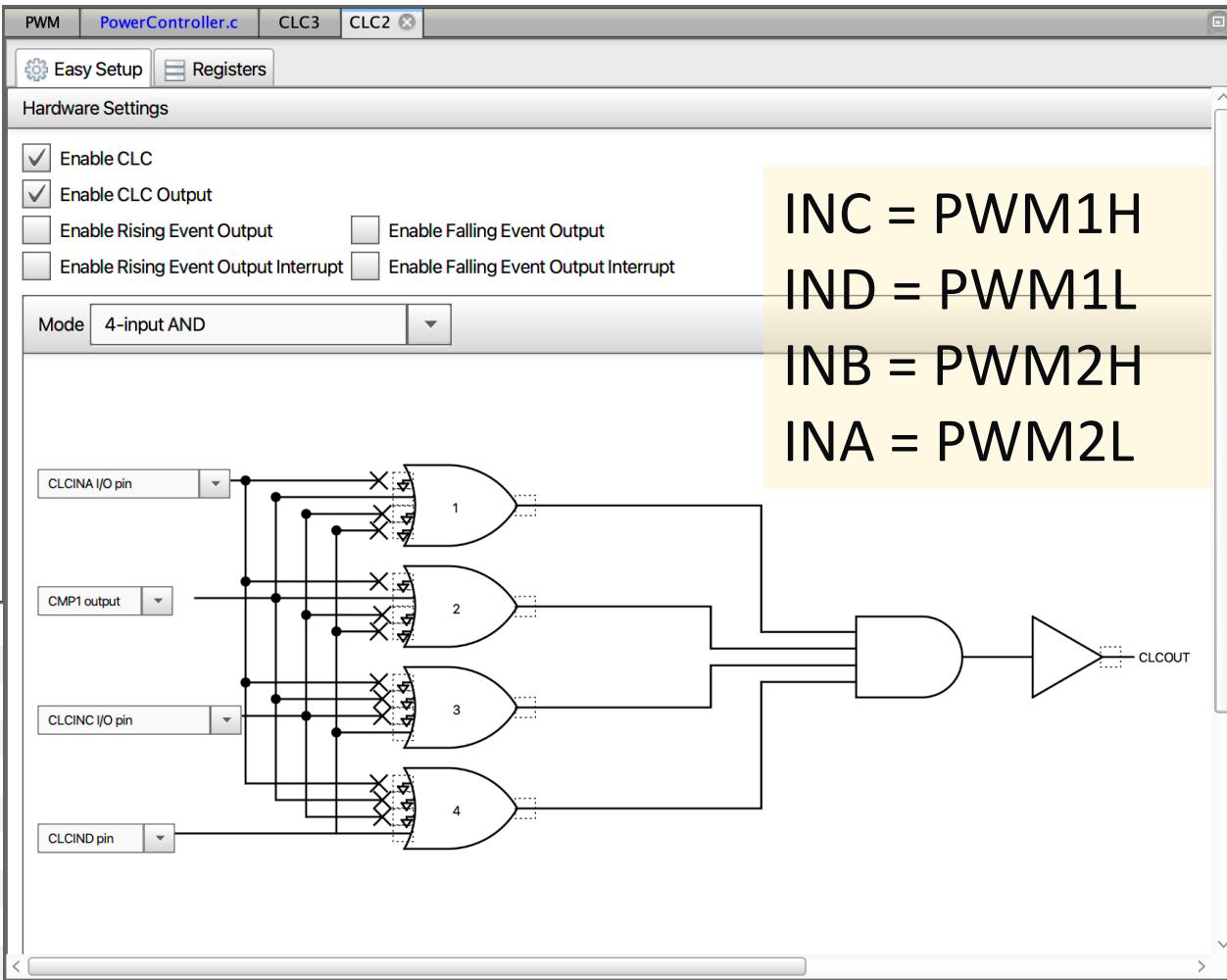
Peripheral & Tool Descriptions

- Peripherals (in RED)

- CLC2

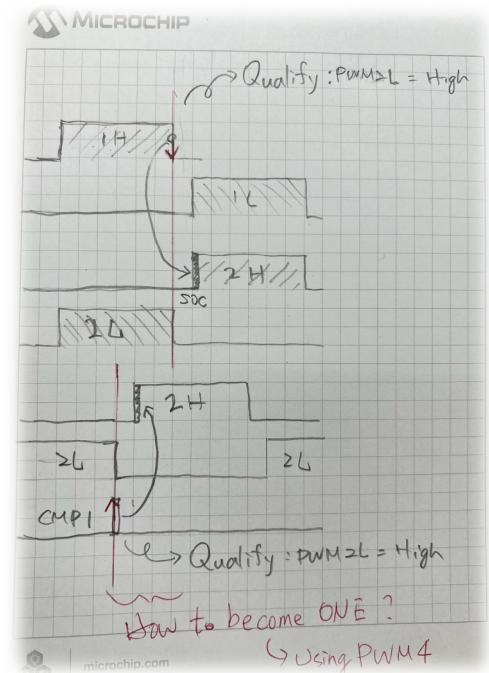
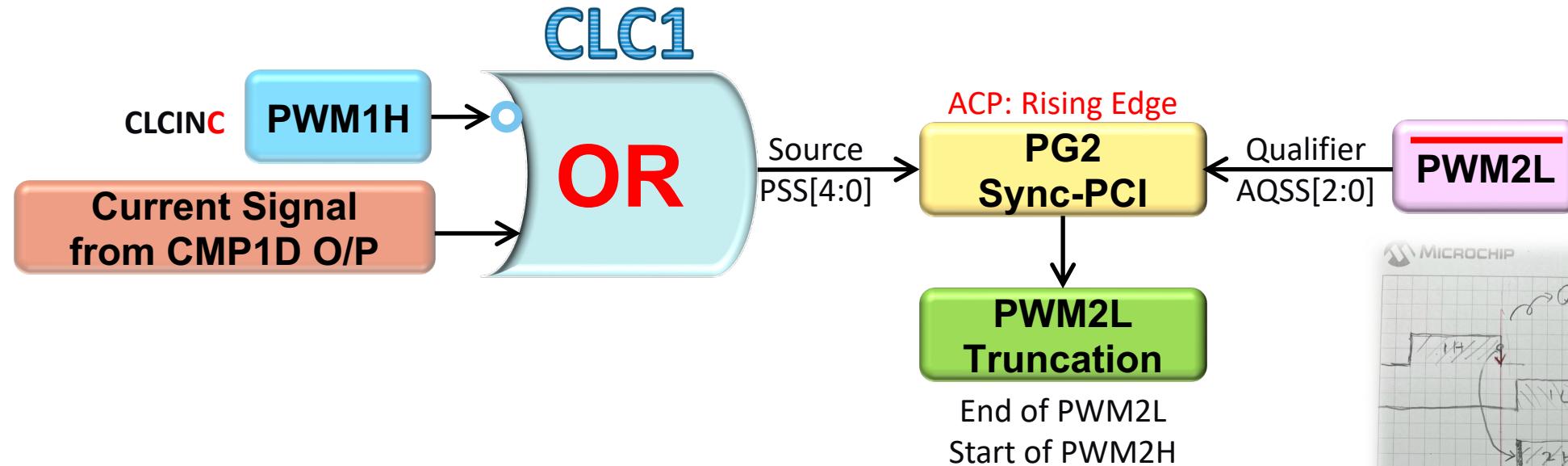
- PWM1L (CLCIND) && CMP1 O/P
 - CLC2 O/P on Pin22(RB9) for checking

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
ADC1 ▾	ADCTRG31	input																					
	ANx	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
CLC2 ▾	CLC2OUT	output							🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	CLCINA	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	CLCINB	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	CLCINC	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	CLCIND	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒



Design Idea

- PWM2L Truncation (Sync-PCI to reset PG2 SOC)

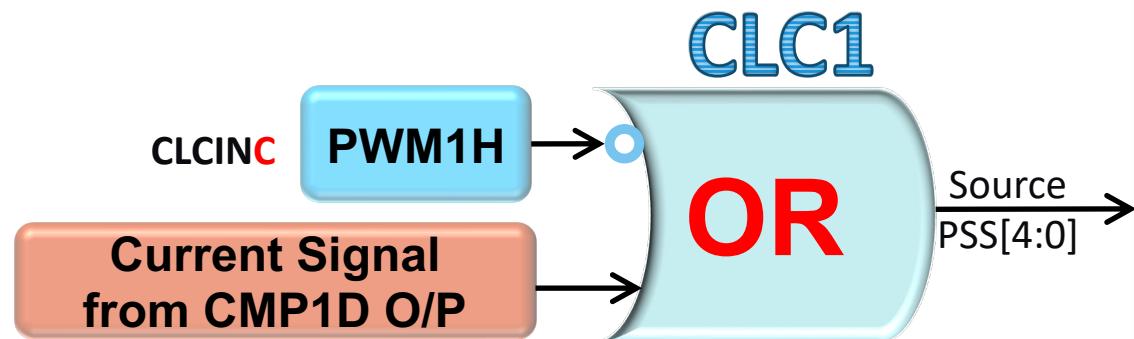


Peripheral & Tool Descriptions

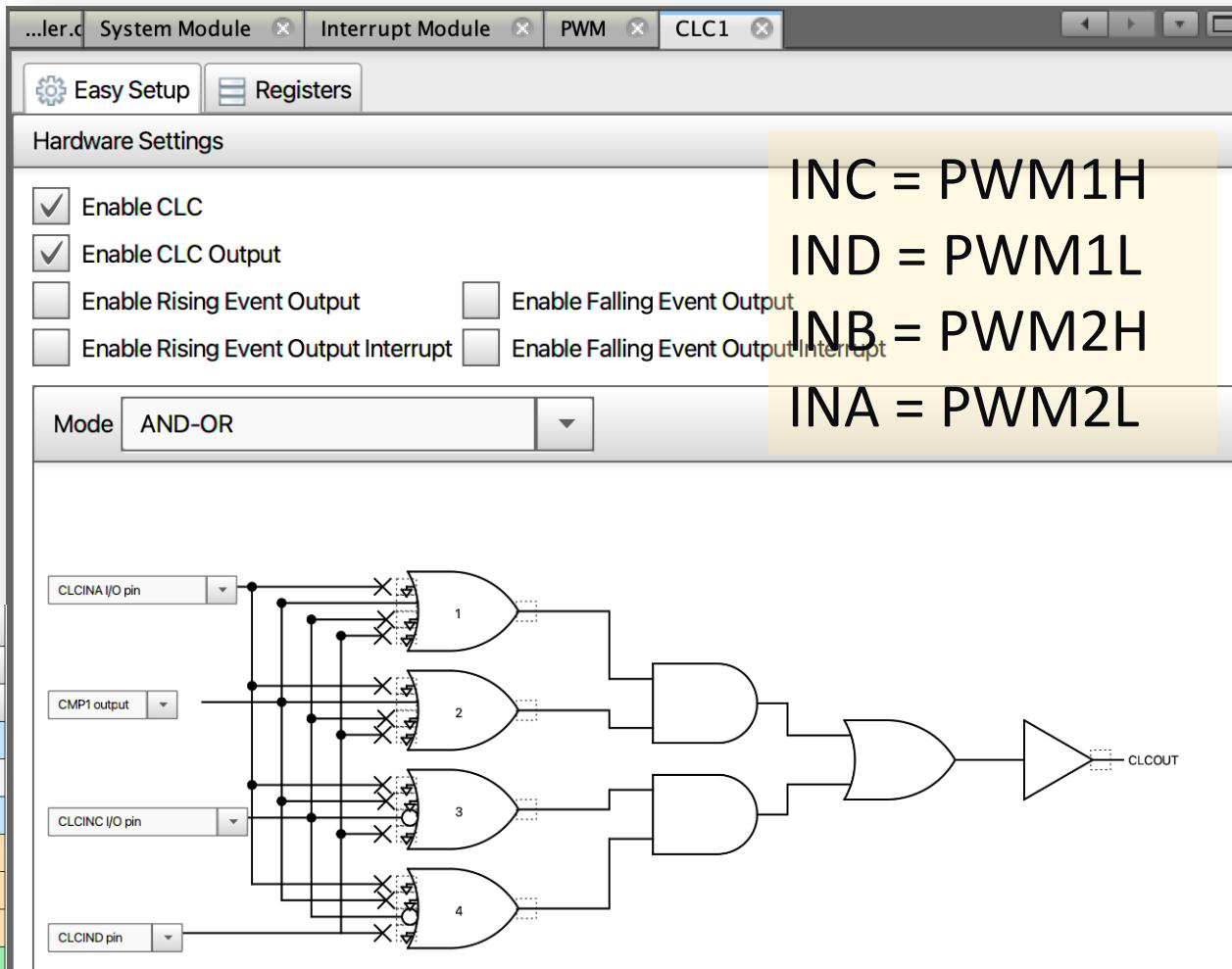
- Peripherals (in RED)

- CLC1

- CMP1D OR'd PWM1H (CLCINC) Inverted
 - CLC1 O/P on Pin21(RB8) for checking



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	
ADC1 ▾	ADCTRG31	input																						
	ANx	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
CLC1 ▾	CLC1OUT	output						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
	CLCINA	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
	CLCINB	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
	CLCINC	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
	CLCIND	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	

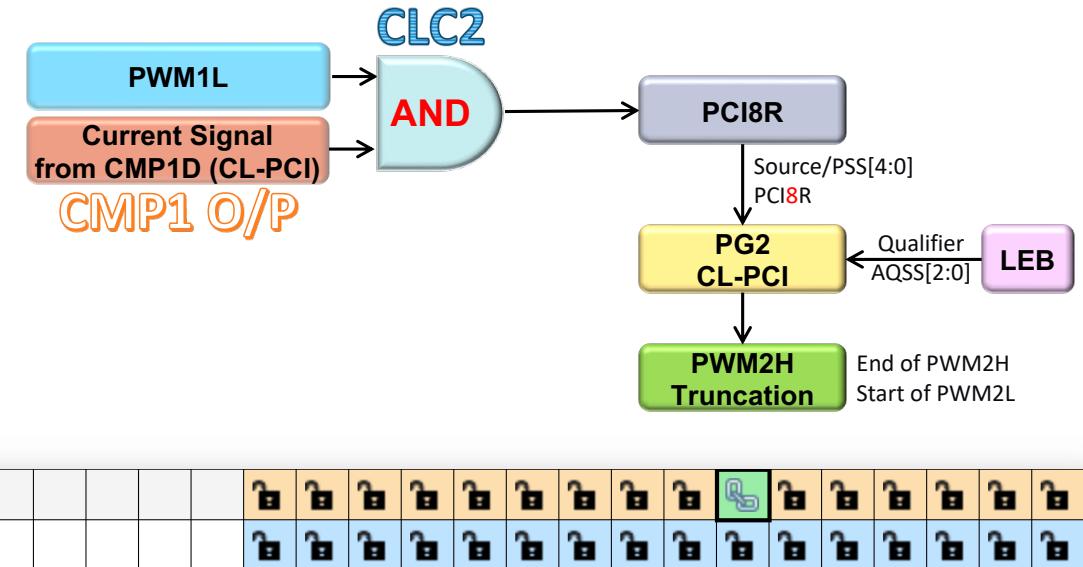


PG2CLPCI

PWM Generator xy PCI Register Low (x = PWM Generator #; y = F, CL, FF or S)

Name: PGxyPCIL

- Peripherals (in RED)
 - 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
 - PG2CLPCI



Register: PG2CLPCIH		0x300
ACP	Latched	▼
BPEN	disabled	▼
BPSEL	PWM Generator 1	▼
PCIGT	disabled	▼
SWPCI	Drives '0'	▼
SWPCIM	PCI acceptance logic	▼
TQPS	Not inverted	▼
TQSS	None	▼

Register: PG2CLPCIL	0x1A08
AQPS	Inverted
AQSS	LEB is active
PPS	Not inverted
PSS	RPN input, PCI8R
PSYNC	disabled
SWTERM	disabled
TERM	Auto-Terminate
TSYNCDIS	PWM EOC

Register: PG2IOCONL		0x10
CLDAT	0x1	
CLMOD	disabled	
DBDAT	0x0	
FFDAT	0x0	
FLTDAT	0x0	
OSYNC	User output overrides are synchronized to the local PWM time base	
OVRDAT	0x0	
OVRENH	disabled	
OVRENL	disabled	
SWAP	disabled	

Register: PG2LEBH		0x10A
	PHF	disabled ▾
	PHR	enabled ▾
	PLF	disabled ▾
	PLR	enabled ▾
	PWMPCI	PWM2 ▾
Register: PG2LEBL		0x4F
	LEB	79

$$F_{PGx_clk} = 500 \text{ MHz}$$

$$LEB \text{ Resolution} = 8/500 \text{ MHz} = 16 \text{ ns Steps}$$

$$\text{Desired LEB Time} = T_{LEB} = 1 \mu\text{s}$$

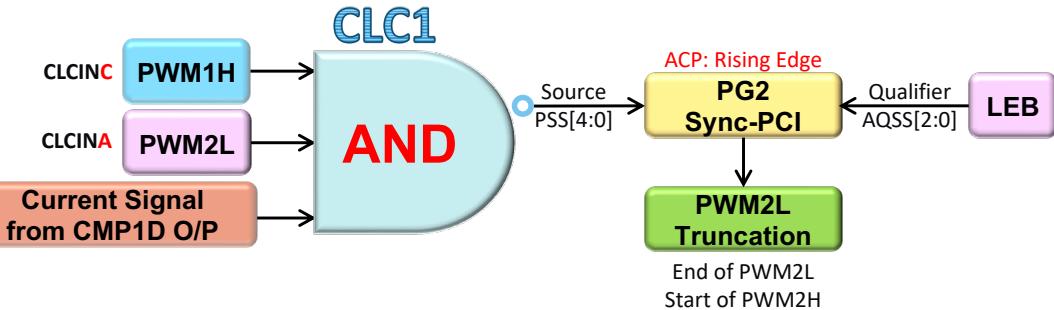
$$LEB[15:3] = \left(\frac{F_{PGx_clk} * T_{LEB}}{8} \right) - 1$$

Name: PGxyPCIL

- Peripherals (in RED)

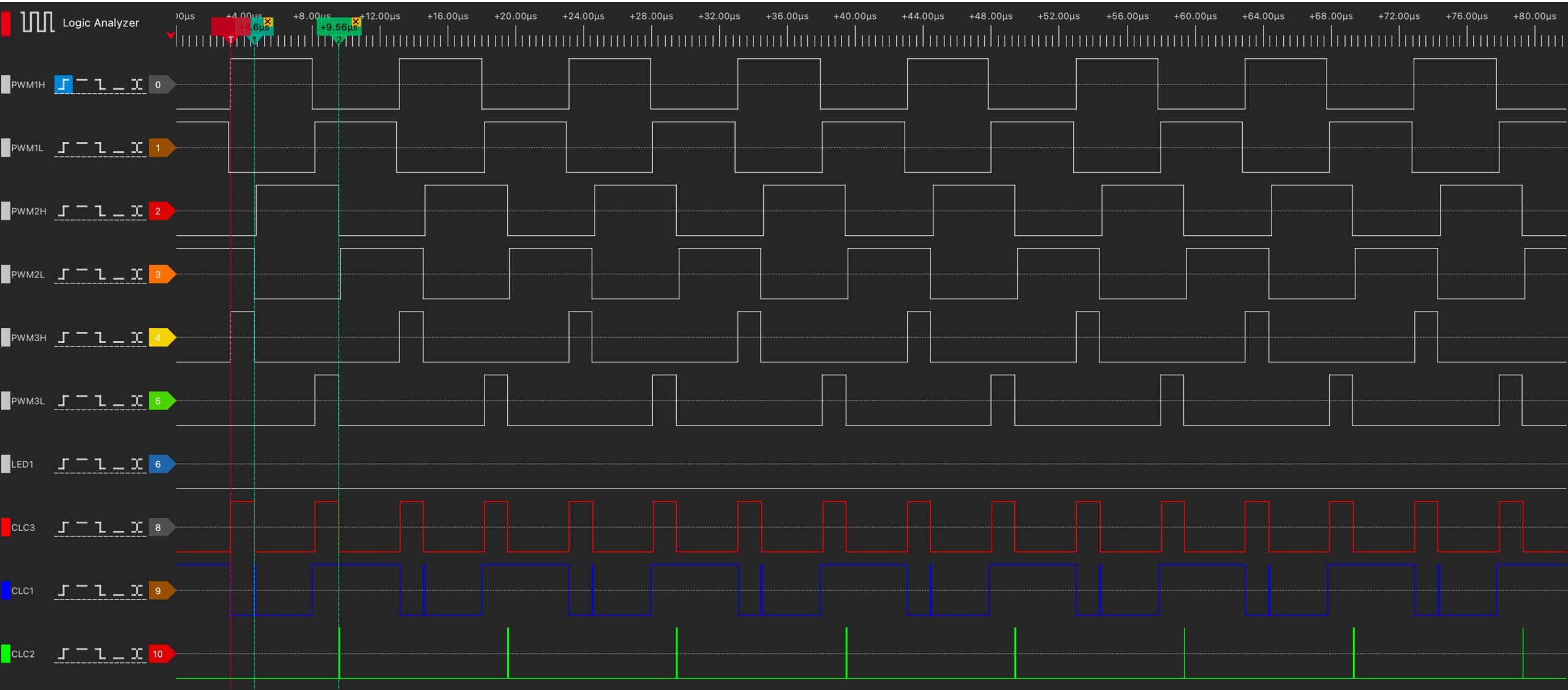
- PWM2 (Phase-shifted leg)

- 100kHz (Complementary) with fixed 50% DC
- SOCS from Sync-PCI & Self-trigger
- Trigger A -> 200ns reserved for SR PWM
- Trigger B -> Reserved
- PG2CLPCI
- PG2SPIC

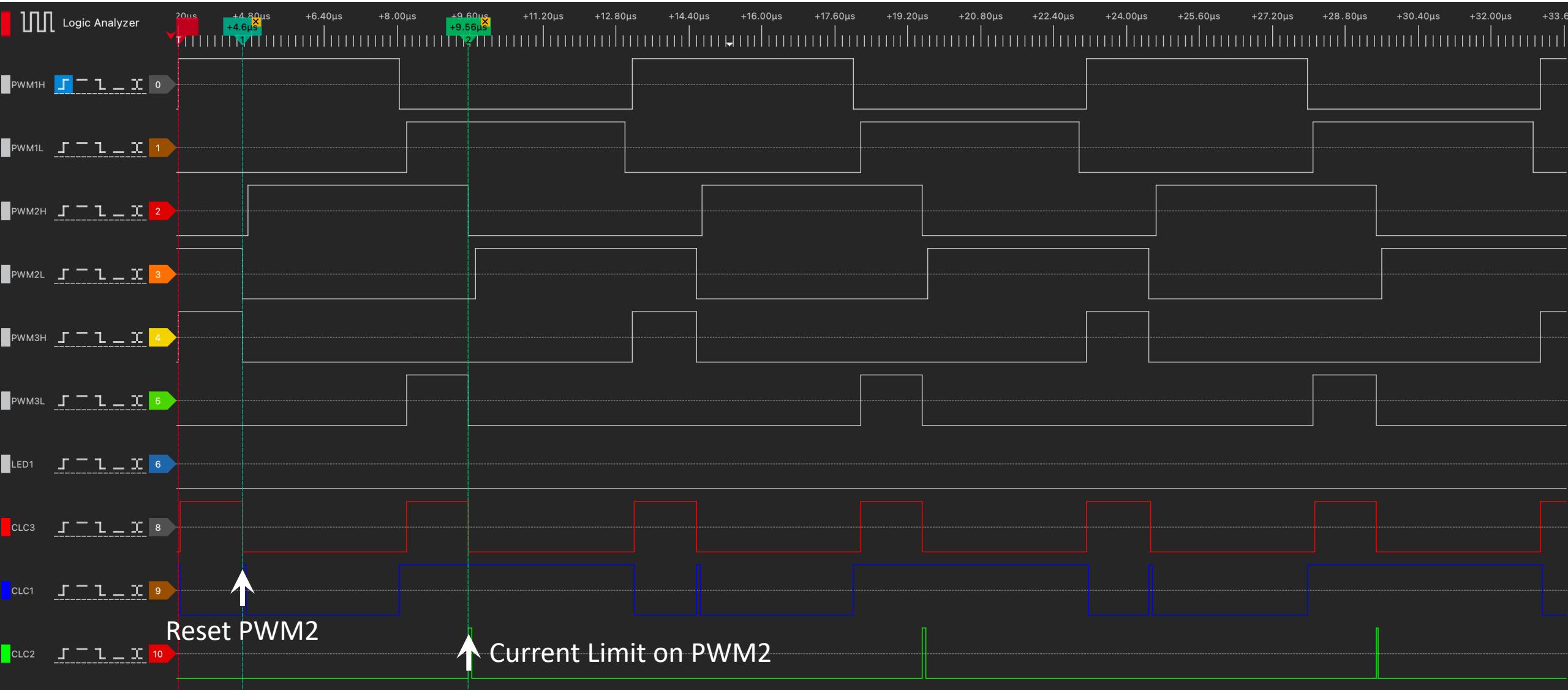


Register: PG2SPCIH 0x100		Register: PG2SPCIL 0x991F		Register: PG1CONH 0x0	
ACP	Rising edge	AQPS	Inverted	MDCSEL	disabled
B PEN	disabled	AQSS	Duty cycle is active	MPERSEL	disabled
BPSEL	PWM Generator 1	PPS	Not inverted	MPHSEL	disabled
PCIGT	disabled	PSS	CLC1 output	MSTEN	disabled
SWPCI	Drives '0'	PSYNC	disabled	SOCS	Self-trigger
SWPCIM	PCI acceptance logic	SWTERM	disabled	TRGMOD	Single trigger mode
TQPS	Not inverted	TERM	Auto-Terminate	UPDMOD	SOC update
TQSS	None	TSYNCDIS	Immediately		

PCMC Timing



PCMC Timing





May The *Power* Be With You

**KNOWLEDGE IS
POWER**

Massive power density in the smallest packages

Thanks!

