PSFB Converter Code Examples

dsPIC33CK64MP102

A Leading Provider of Smart, Connected and Secure Embedded Solutions

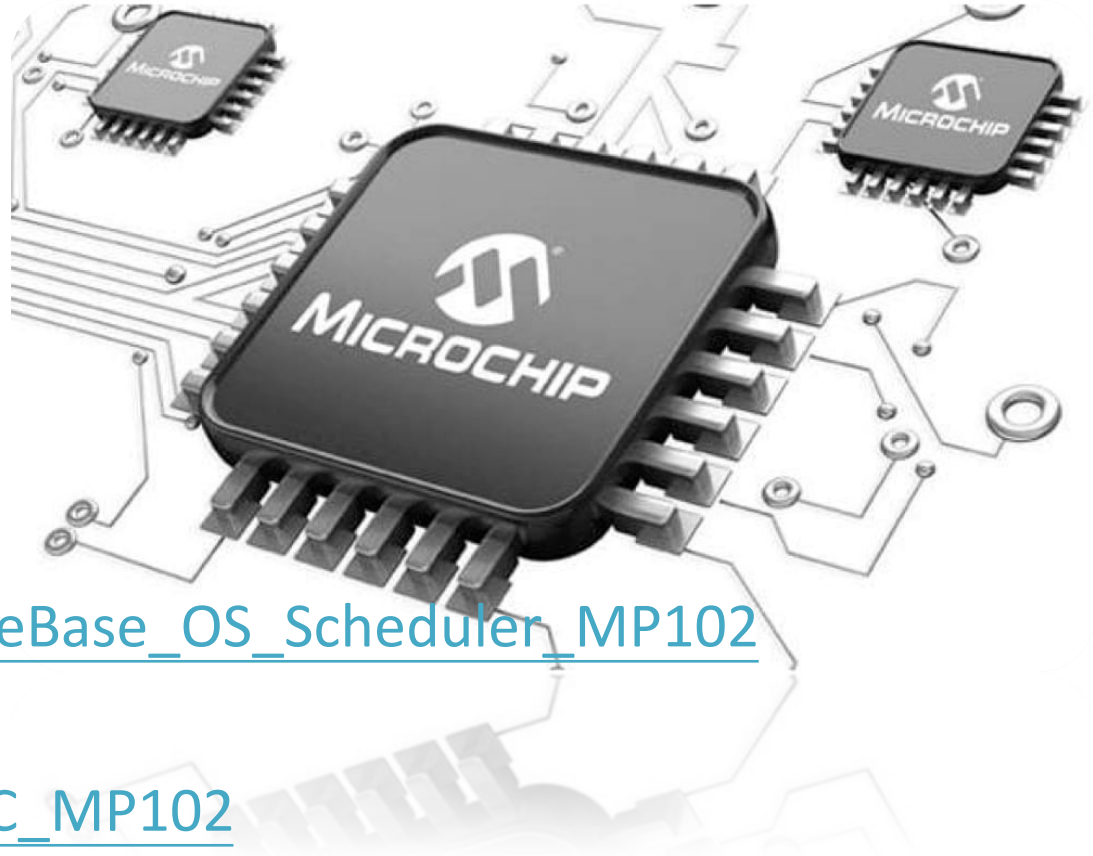
Microchip
ESE / Edward Lee

March-20, 2024

SMART | CONNECTED | SECURE

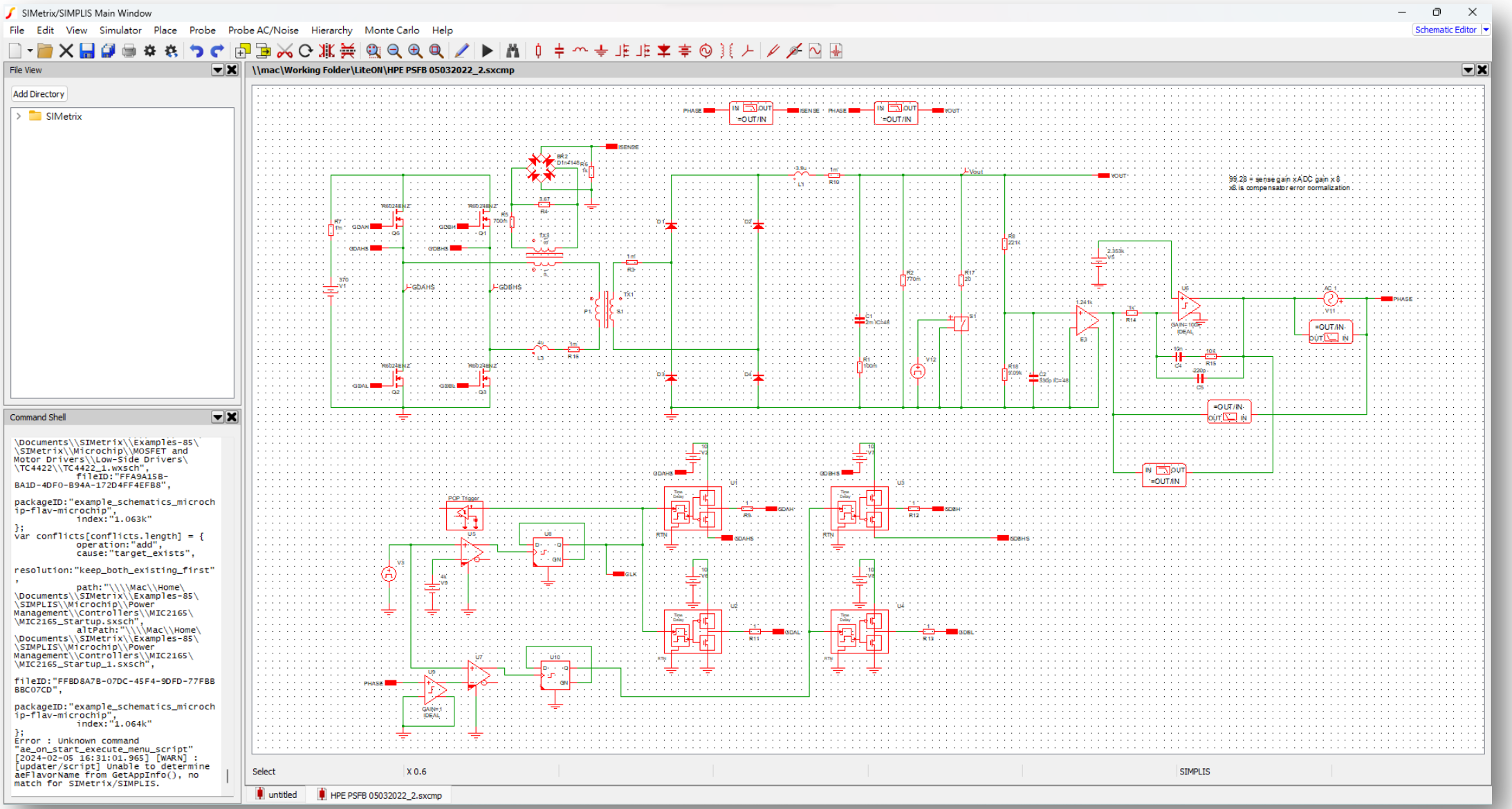
Agenda

- **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

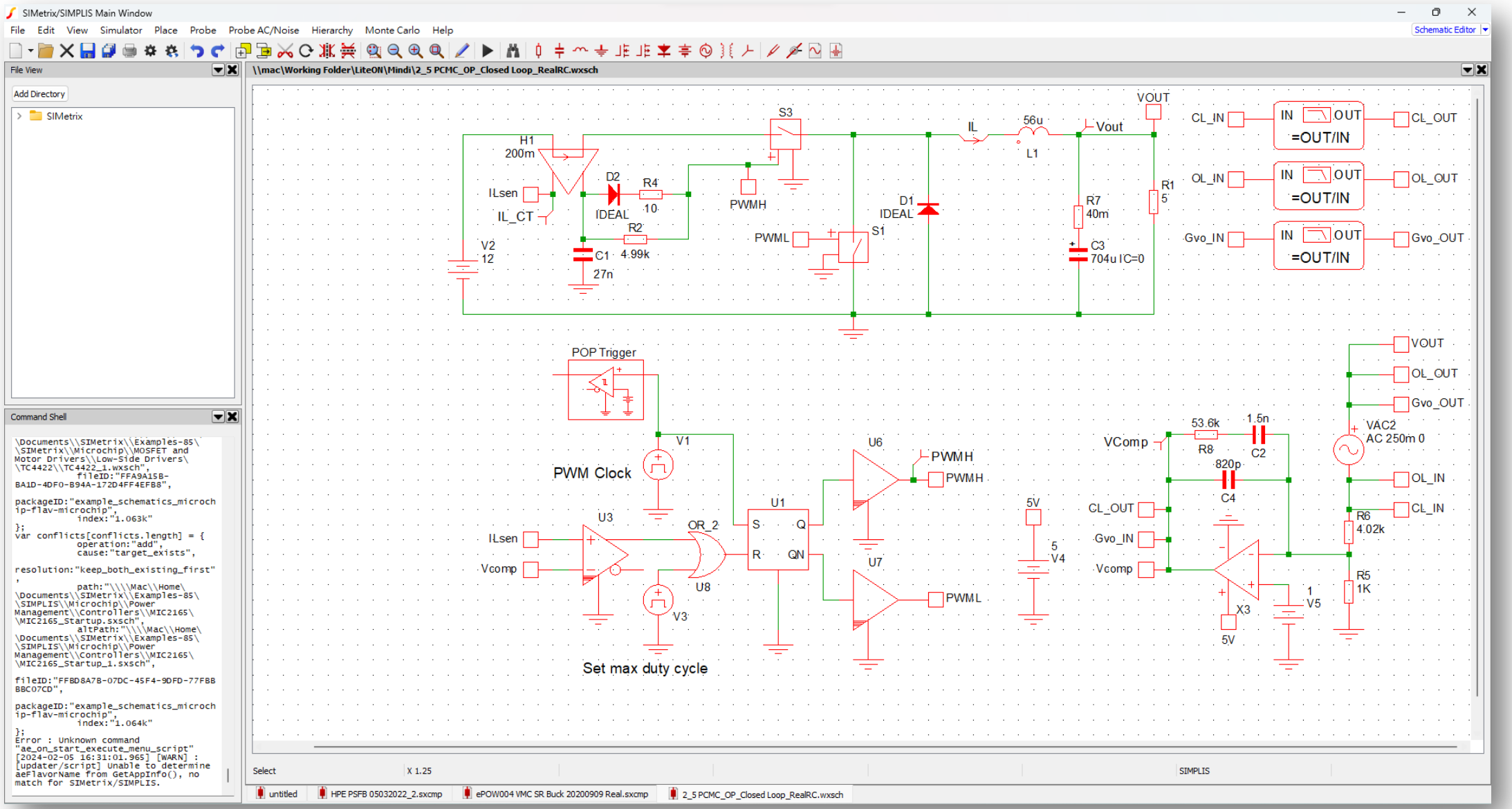


Simulation

SIMPLIS



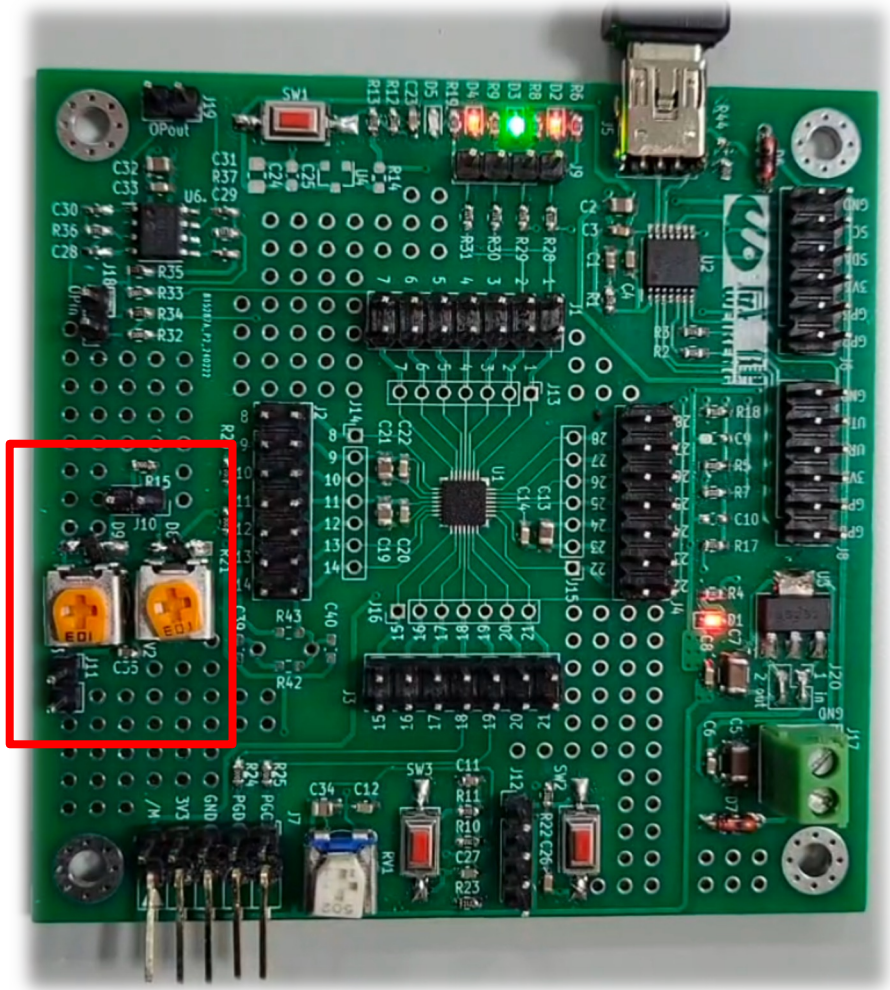
SIMPLIS

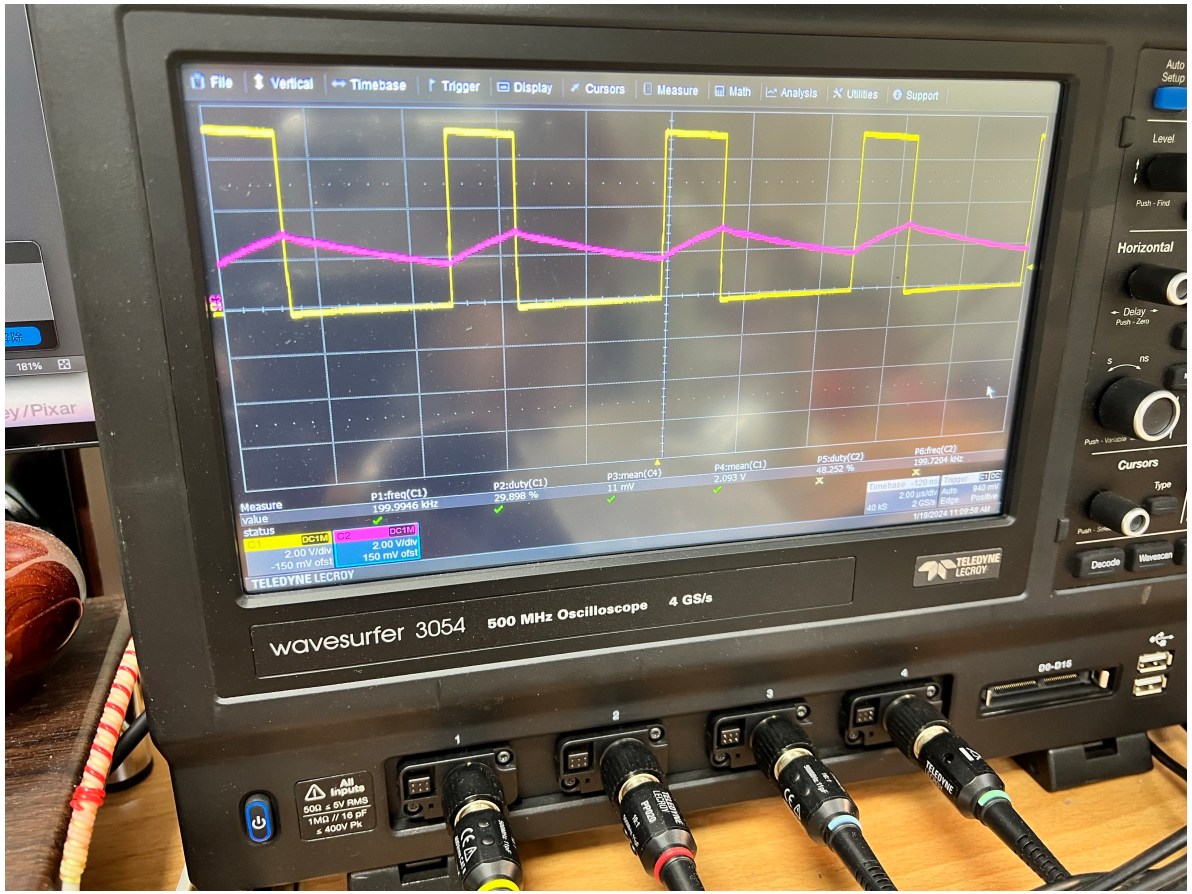


Test Environment

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

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

Pin Module | Interrupt Module | System Module | Content Manager

Easy Setup | Registers

Clock

8000000 Hz | FRC Oscillator | (8.0 MHz) Clock Source

FRC Postscaler

PLL Enable

8 MHz	1:1	Prescaler
1600 MHz	1:200	FeedBack
800 MHz	1:2	Postscaler1
400 MHz	1:2	Postscaler2

200 MHz Fosc

100 MHz Fosc/2 **100MIPS**

Auxiliary Clock

8000000 Hz | FRC | Clock Source

PLL Enable

8 MHz	1:1	Prescaler
1000 MHz	1:125	Feedback
500 MHz	1:2	Postscaler1
500 MHz	1:1	Postscaler2

VCO & AVCO

400 MHz	FVCO/4	VCO Divider
500 MHz	FVCO/2	AVCO Divider

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

Reference Oscillator Output

CAN FD Clock Generator

ICD

Emulator Pin Placement: Communicate on PGC1 and PGD1

Pin Manager: Package View

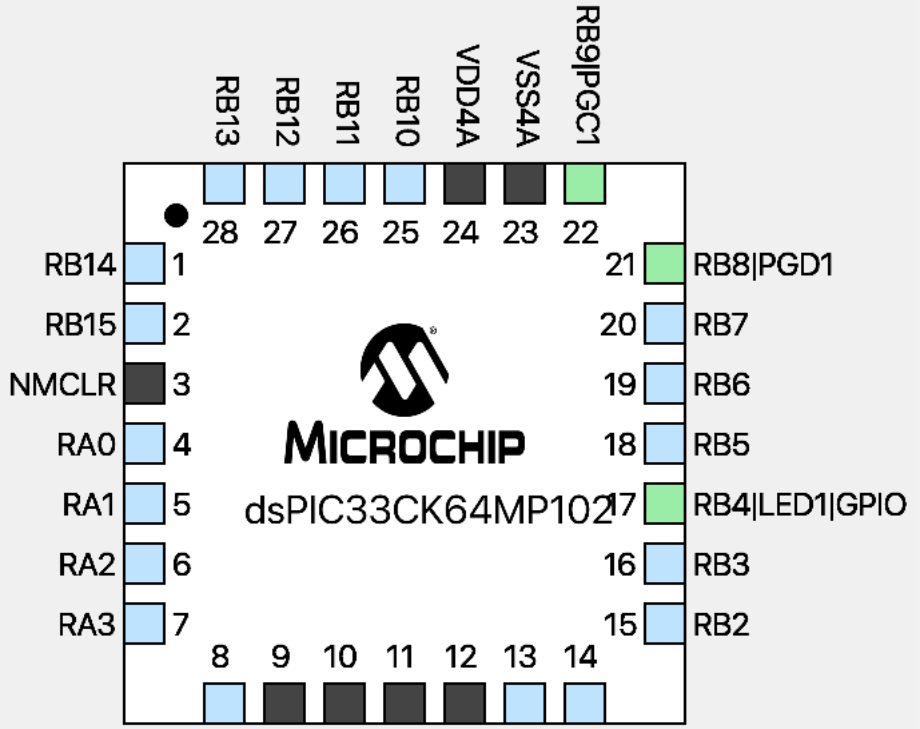
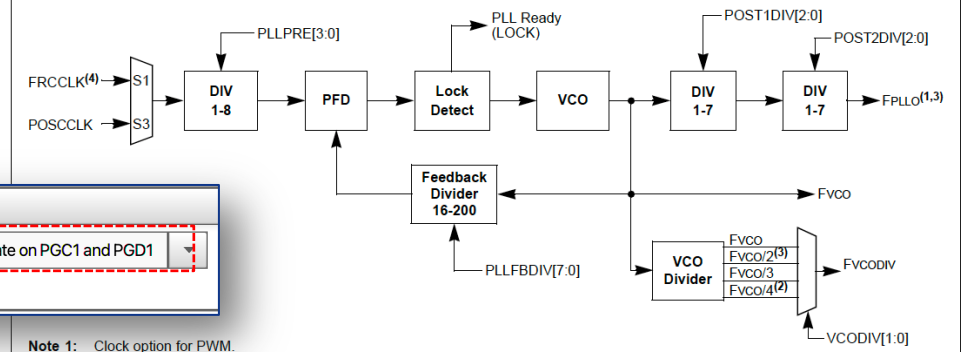


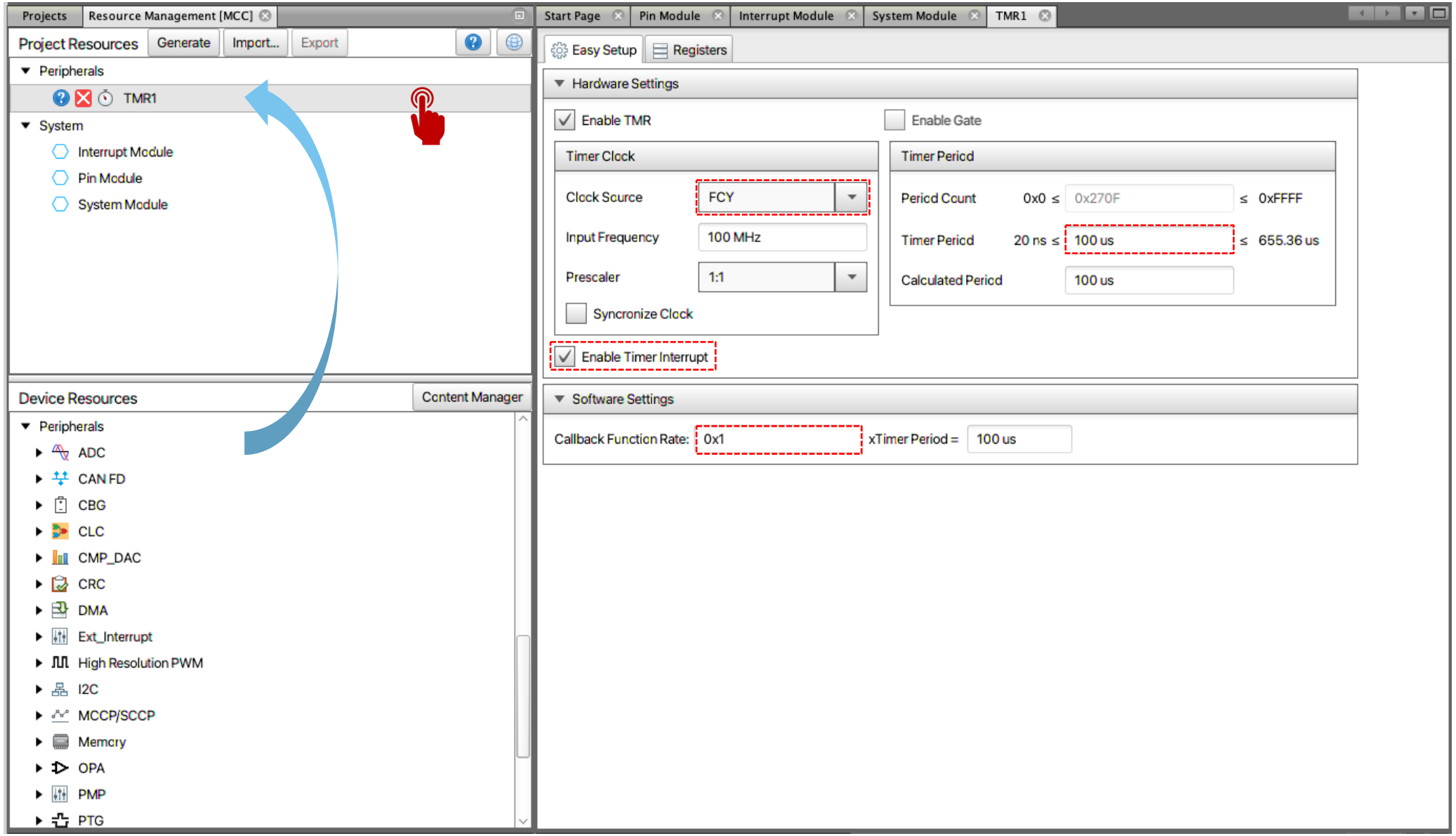
FIGURE 9-3: PLL AND VCO DETAIL



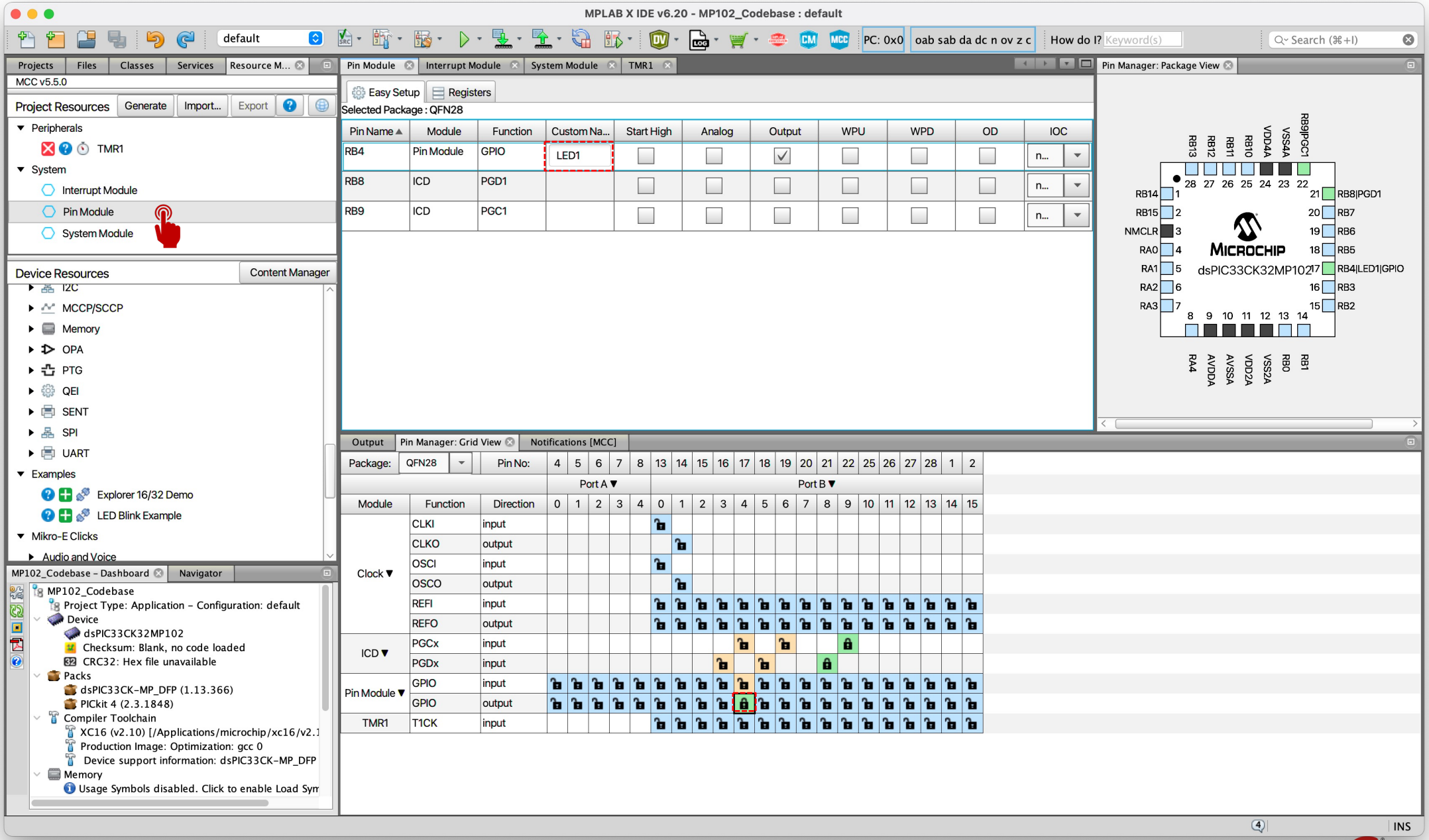
Note 1: Clock option for PWM.
2: Clock option for ADC.
3: Clock option for DAC.
4: PLL source is always FRC unless FNOSC is the Primary Oscillator with PLL.

Notifications | Output | Pin Manager: Grid View | Notifications [MCC]

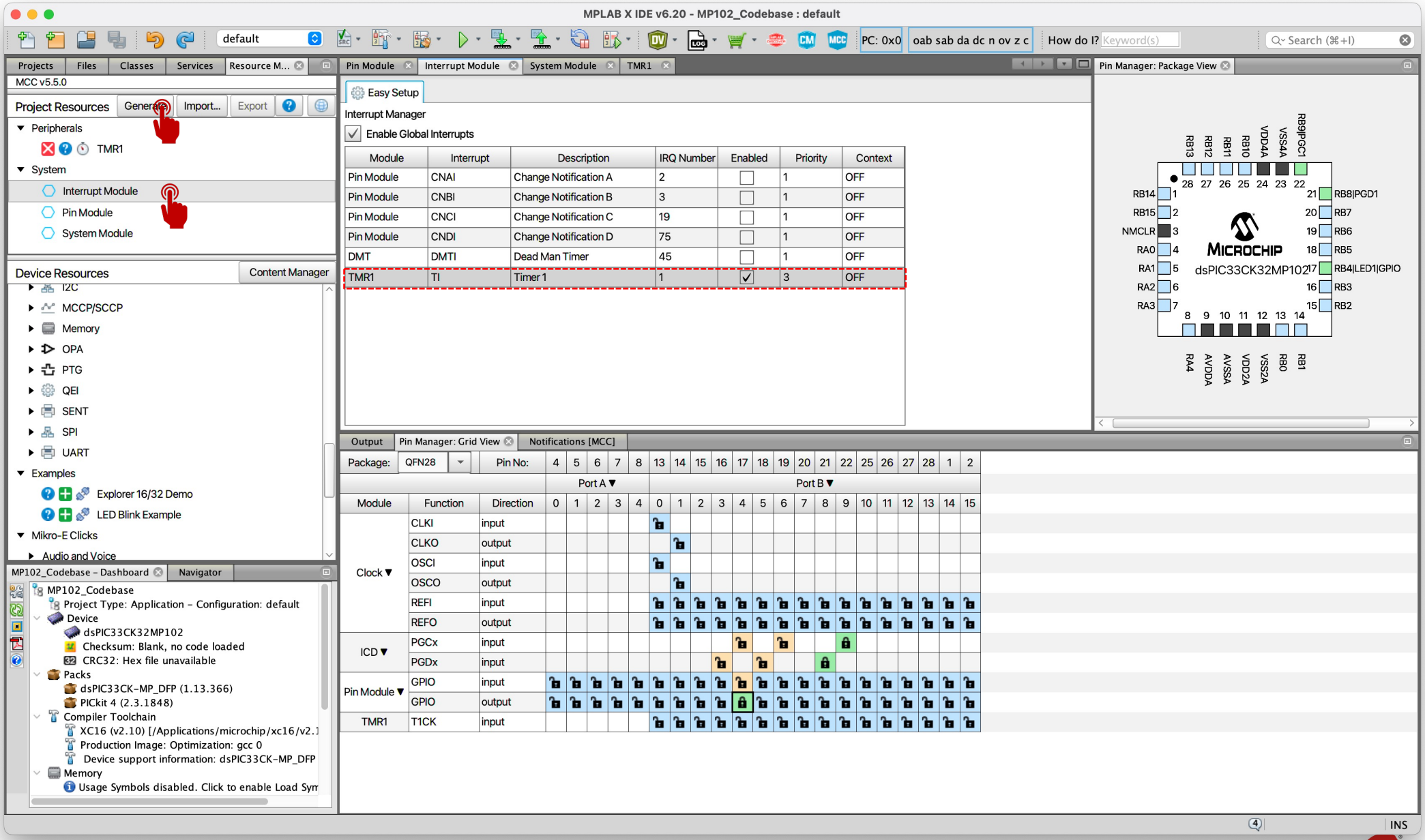
MCC Drivers – Timer1 (100 us)



MCC Drivers – Pin Management

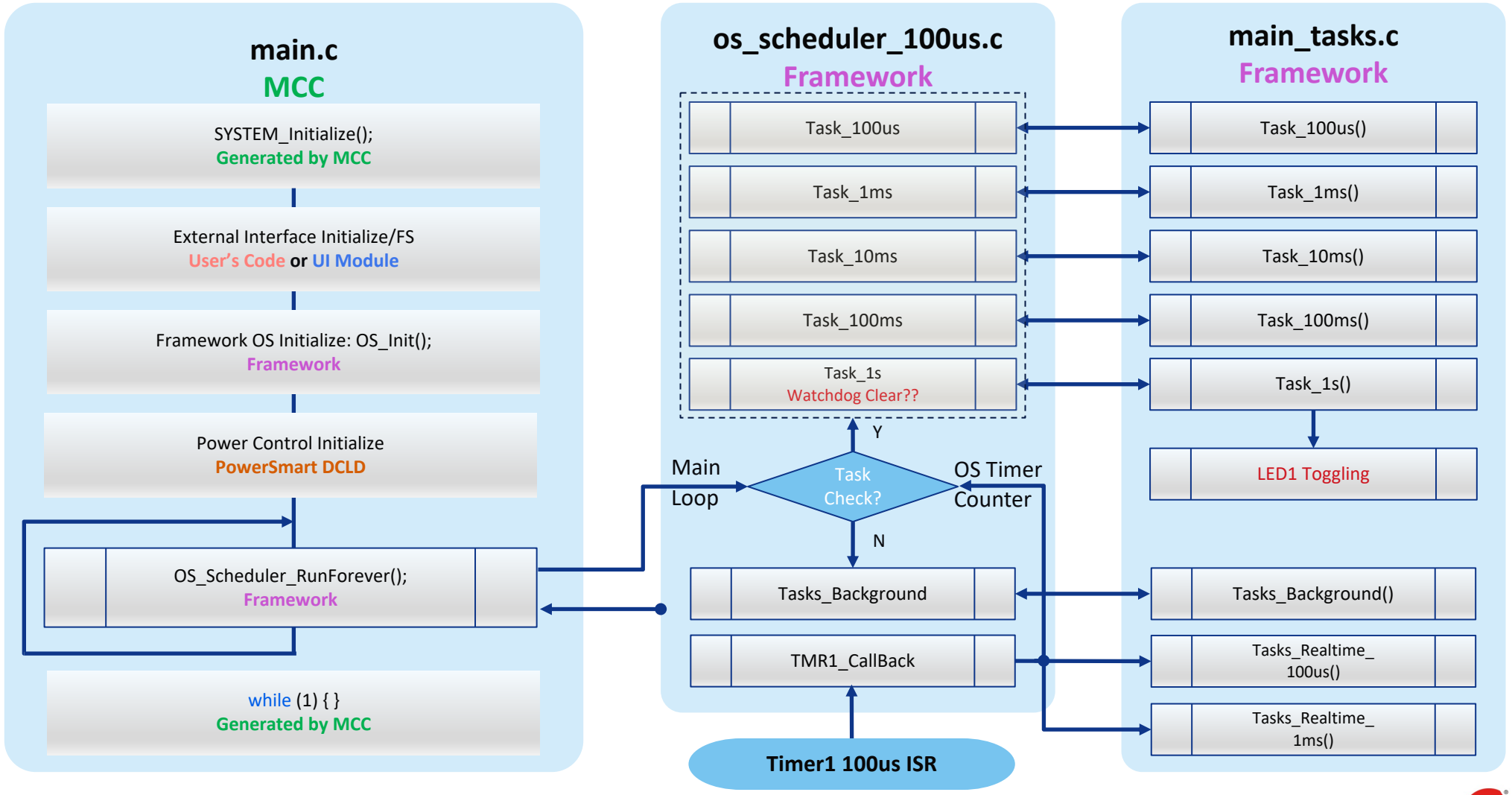


MCC Drivers – Interrupt Management & Code Generating



SMPS Firmware Framework

Firmware Structure



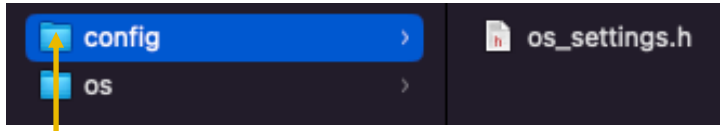
SMPS Firmware Framework

Directory Structure

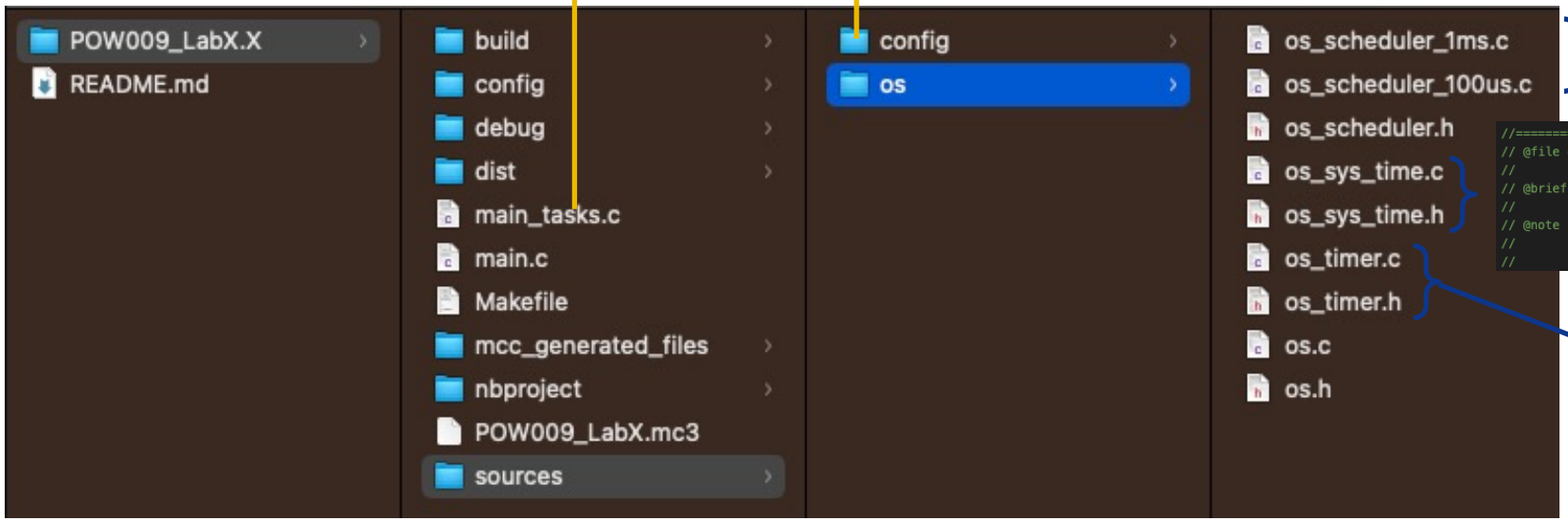
Directory path	Description
ExampleProject.X	main project directory
ExampleProject.X\main.c	the main.c file should be in the main project directory because MCC puts it in here.
ExampleProject.X\main_tasks.c	the main_tasks.c file that the MCOS (Microchip Cooperative Operating System) requires should be 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

The application specific tasks that are called regularly by the os_scheduler



Project specific settings regarding the operating system parts and application

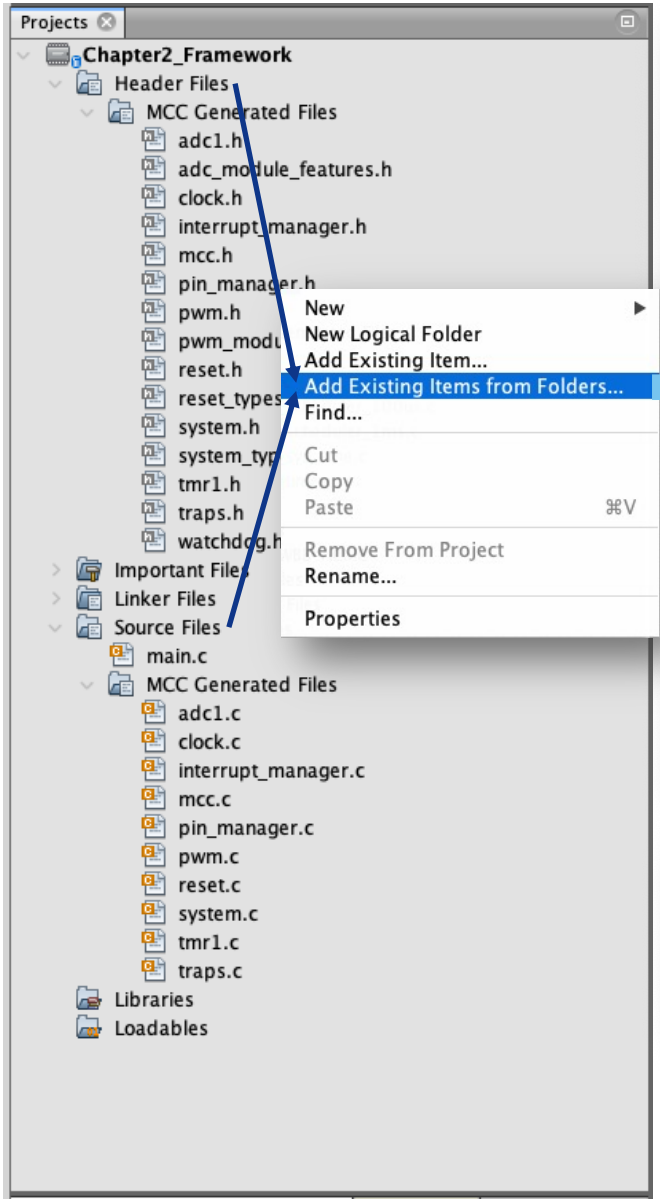


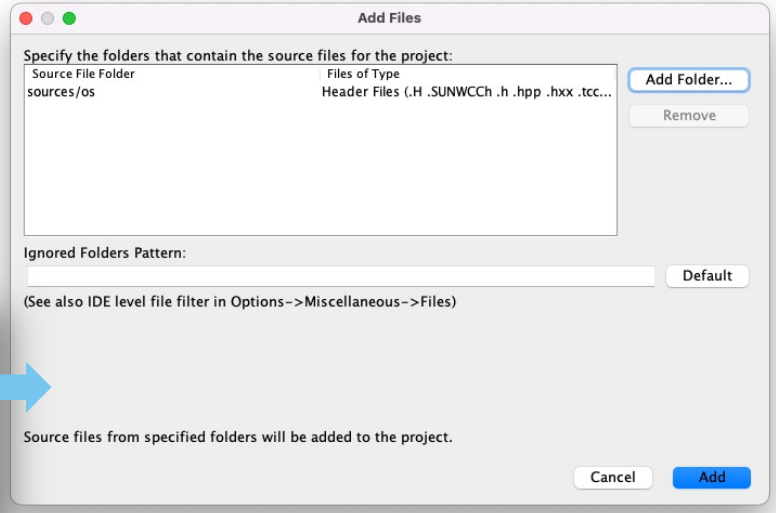
1 ms for low-end MCU such like PIC16
100 us for high speed MCU such like dsPIC33

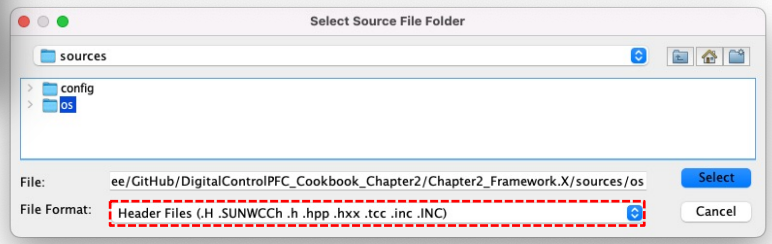
```
//==========================================================================================
// @file os_sys_time.c
//
// @brief contains the function for the system time
//
// @note    counting up the time can/should be done in the interrupt routine of the scheduler
//          reading the time can also be done outside of the interrupt without any conflicts
//
```

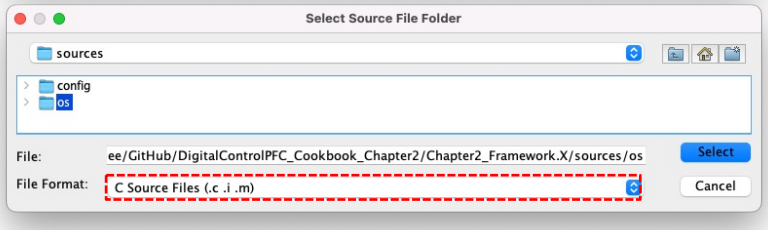
```
//==========================================================================================
// @file os_timer.c
//
// @brief contains the functions for using software timers
//
// @note    in this file you will find functions for using timers and countdowns for your application
//          These software timers are updated by the hardware timer interrupt of the os_scheduler
//          The number of timers need to be defined in main/project_settings.h :
//              #define OS_TIMER_NUMBER_OF_TIMERS 3
//          You should also put your your individual Timer defines with the corresponding index there
//              #define OS_TIMER_COUNTDOWN_EXAMPLE_SWITCHOFFTHELIGHT        0
//              #define OS_TIMER_COUNTDOWN_EXAMPLE_SWITCHOFFCOFFEEMACHINE   1
//              #define OS_TIMER_COUNTDOWN_EXAMPLE_IAMGOINGHOMENOW          2
//
//
```

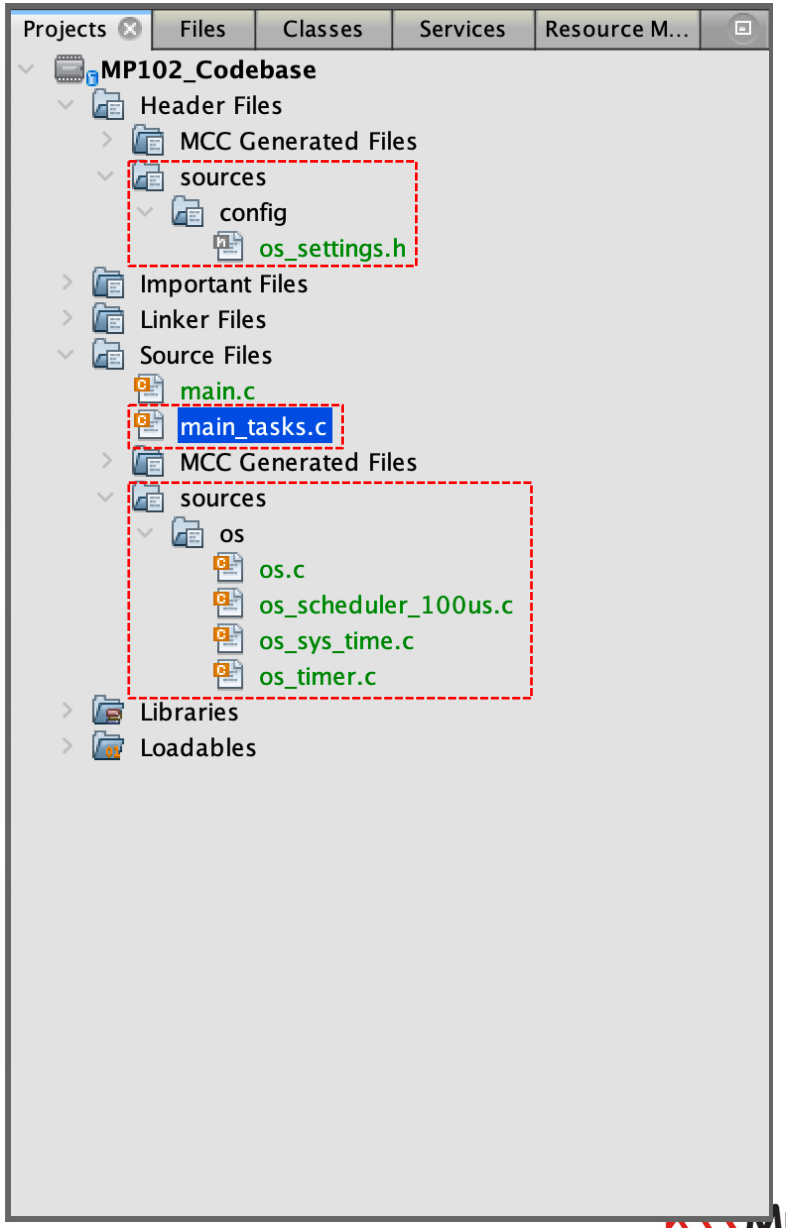
Integrate OS_Scheduler











Main & Tasks

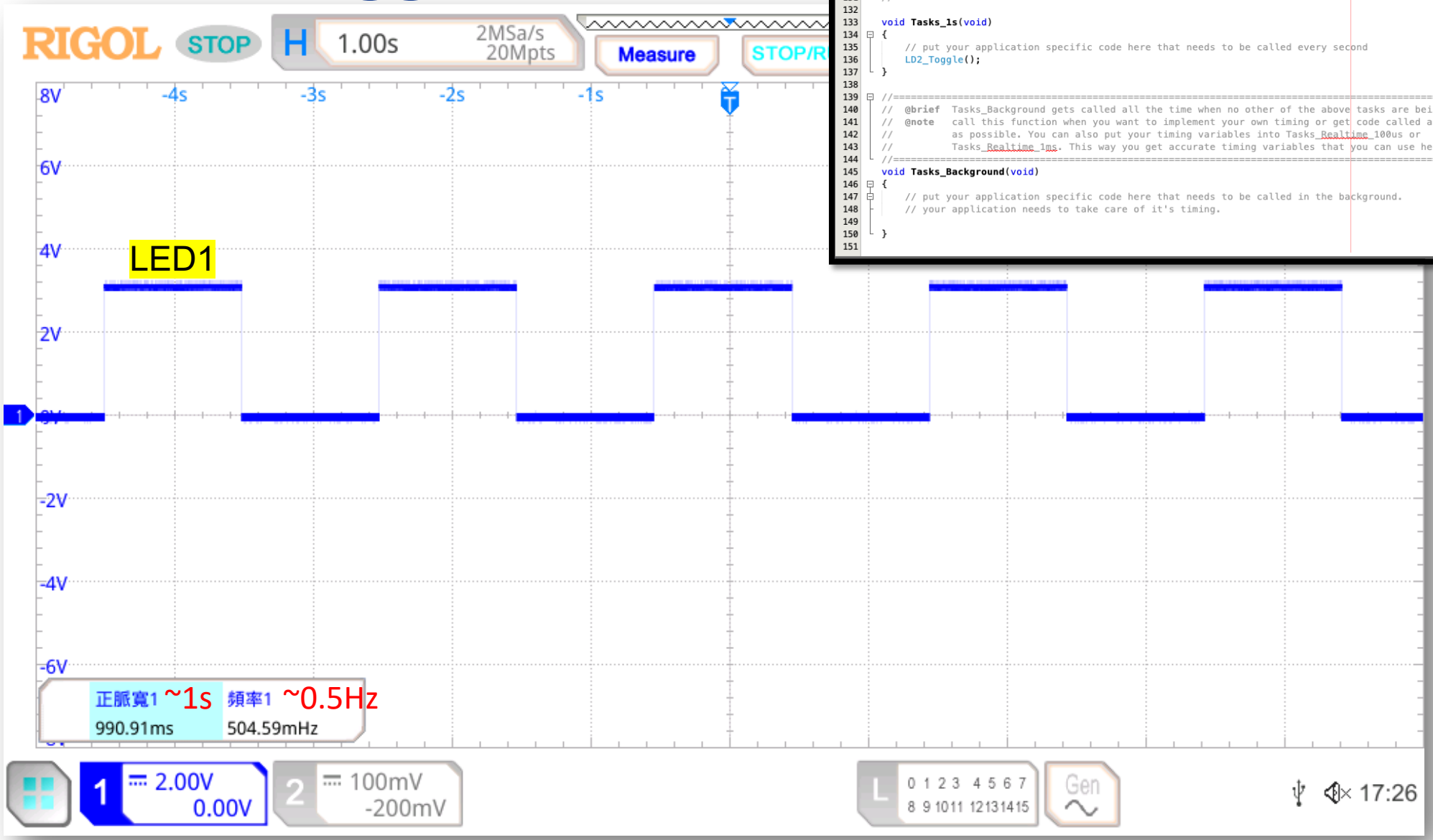
main.c

```
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 tim
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




```
main_tasks.c
Source  History
127
128 //=============================================================================
129 //  @brief  Tasks_1s gets called every second, put your things in it that need to be called regularly
130 //  @note   there could be some jitter here because it is not called directly by a timer interrupt
131 //=============================================================================
132
133 void Tasks_1s(void)
134 {
135     // put your application specific code here that needs to be called every second
136     LD2_Toggle();
137 }
138
139 //=============================================================================
140 //  @brief  Tasks_Background gets called all the time when no other of the above tasks are being called
141 //  @note   call this function when you want to implement your own timing or get code called as often
142 //          as possible. You can also put your timing variables into Tasks_Realtime_100us or
143 //          Tasks_Realtime_1ms. This way you get accurate timing variables that you can use here.
144 //=============================================================================
145 void Tasks_Background(void)
146 {
147     // put your application specific code here that needs to be called in the background.
148     // your application needs to take care of it's timing.
149
150 }
151
```

VMC PSFB Converter With SR

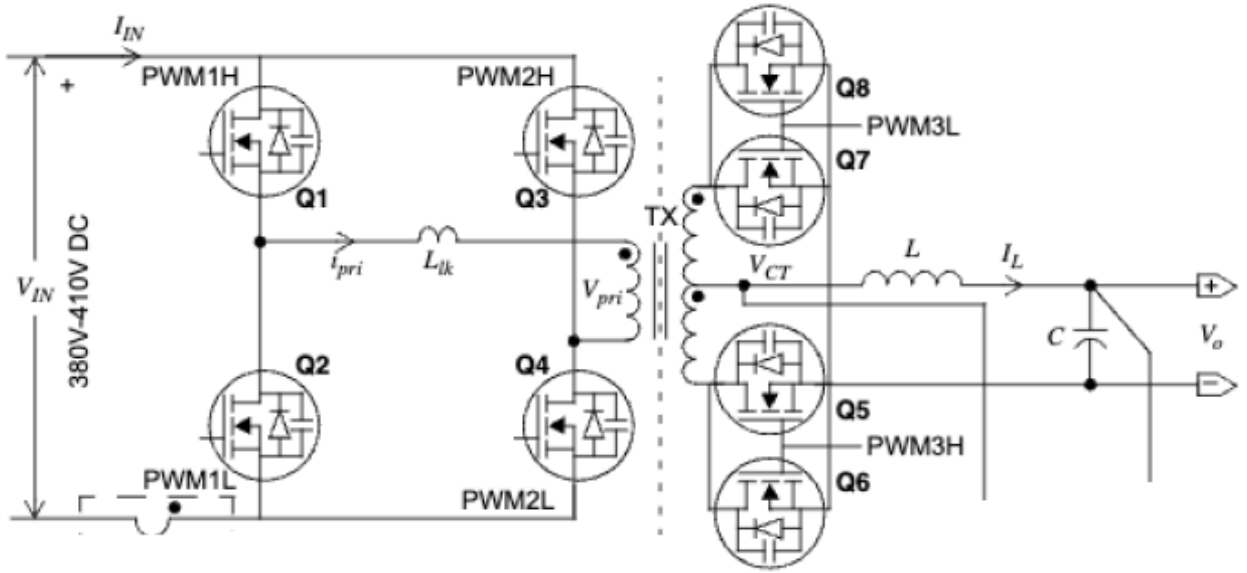
Peripheral & Tool Descriptions

- **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 -> 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

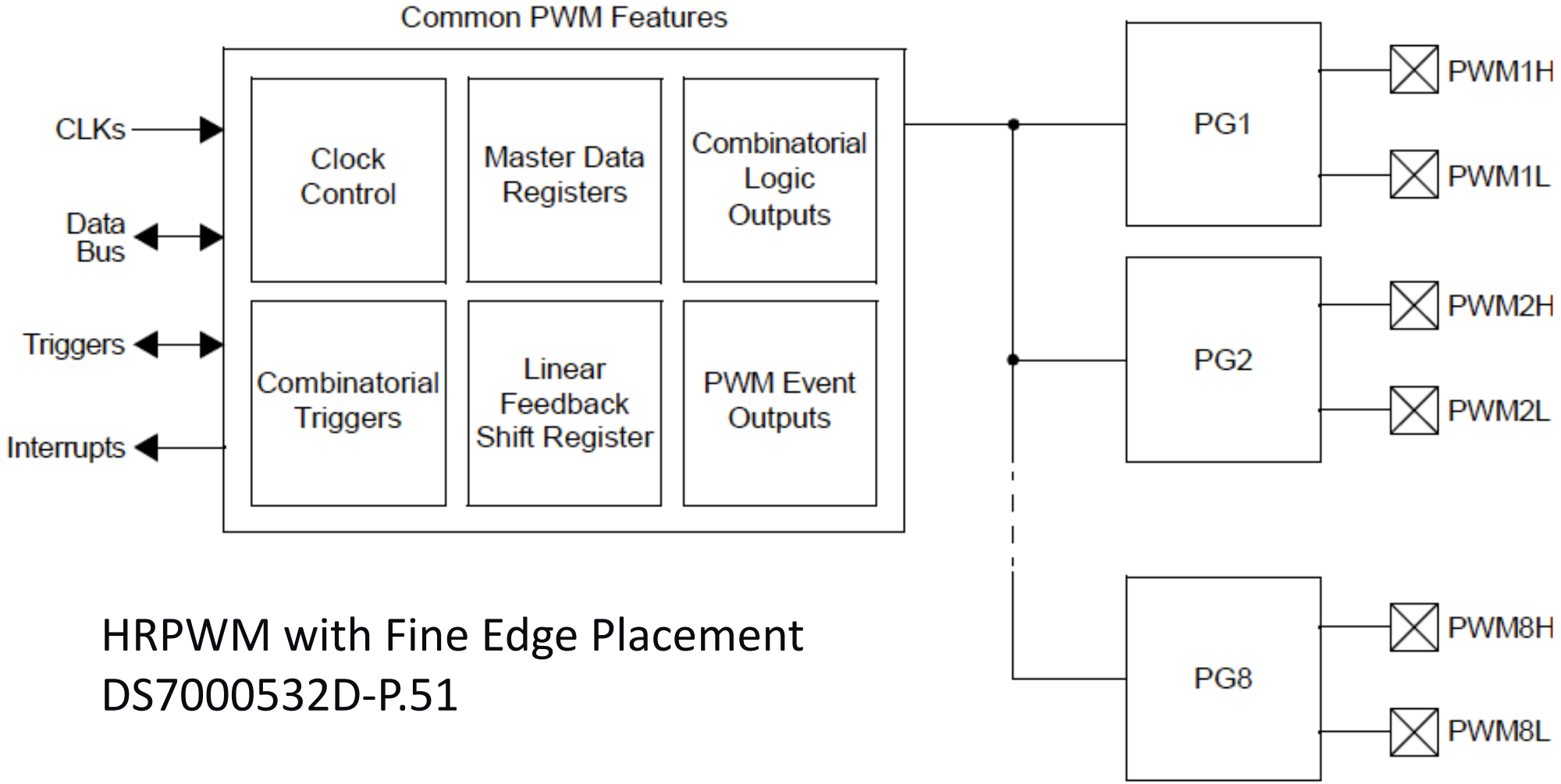
- AN0 (Port Rxx)
  - IP = 5 & NO Context
  - Triggered by PWM1 Trigger2 (B)
  - ISR 100KHz



PWM Ideas

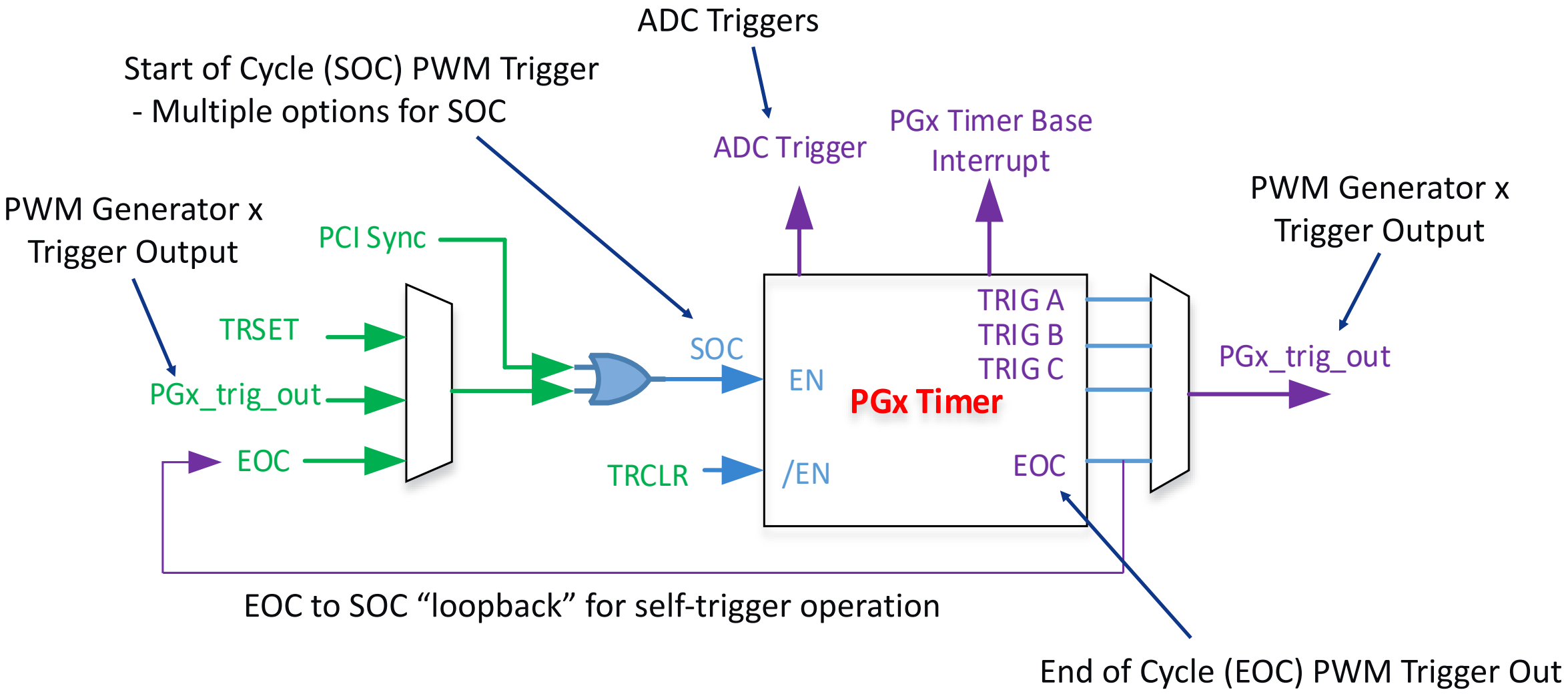
PWM High-Level Block Diagram

Figure 3-1. PWM High-Level Block Diagram



HRPWM with Fine Edge Placement
DS7000532D-P.51

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 – 0101	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

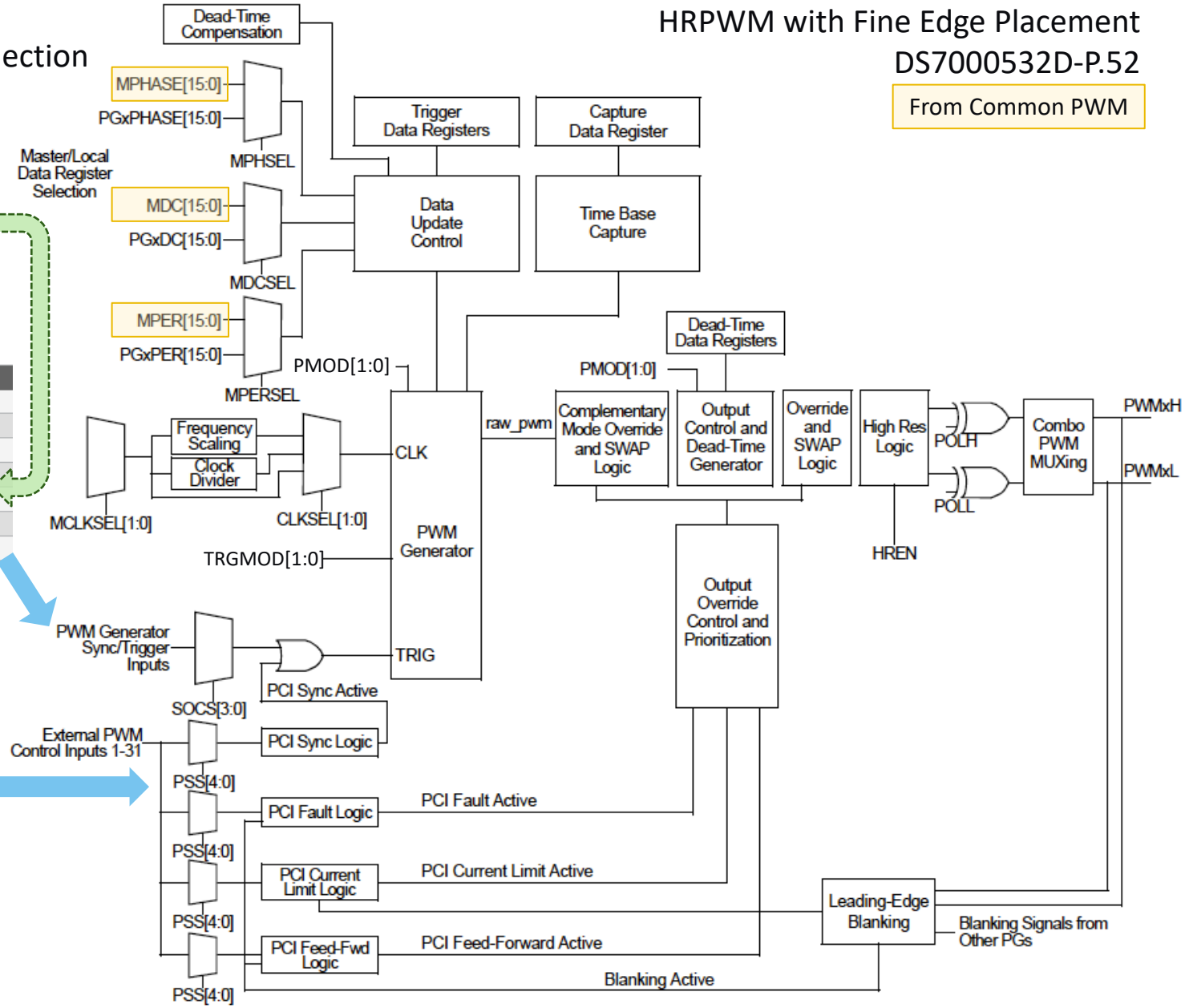
Note: PCI sources are device-dependent; refer to the device data sheet for availability.

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

HRPWM with Fine Edge Placement

DS7000532D-P.52

From Common PWM



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)

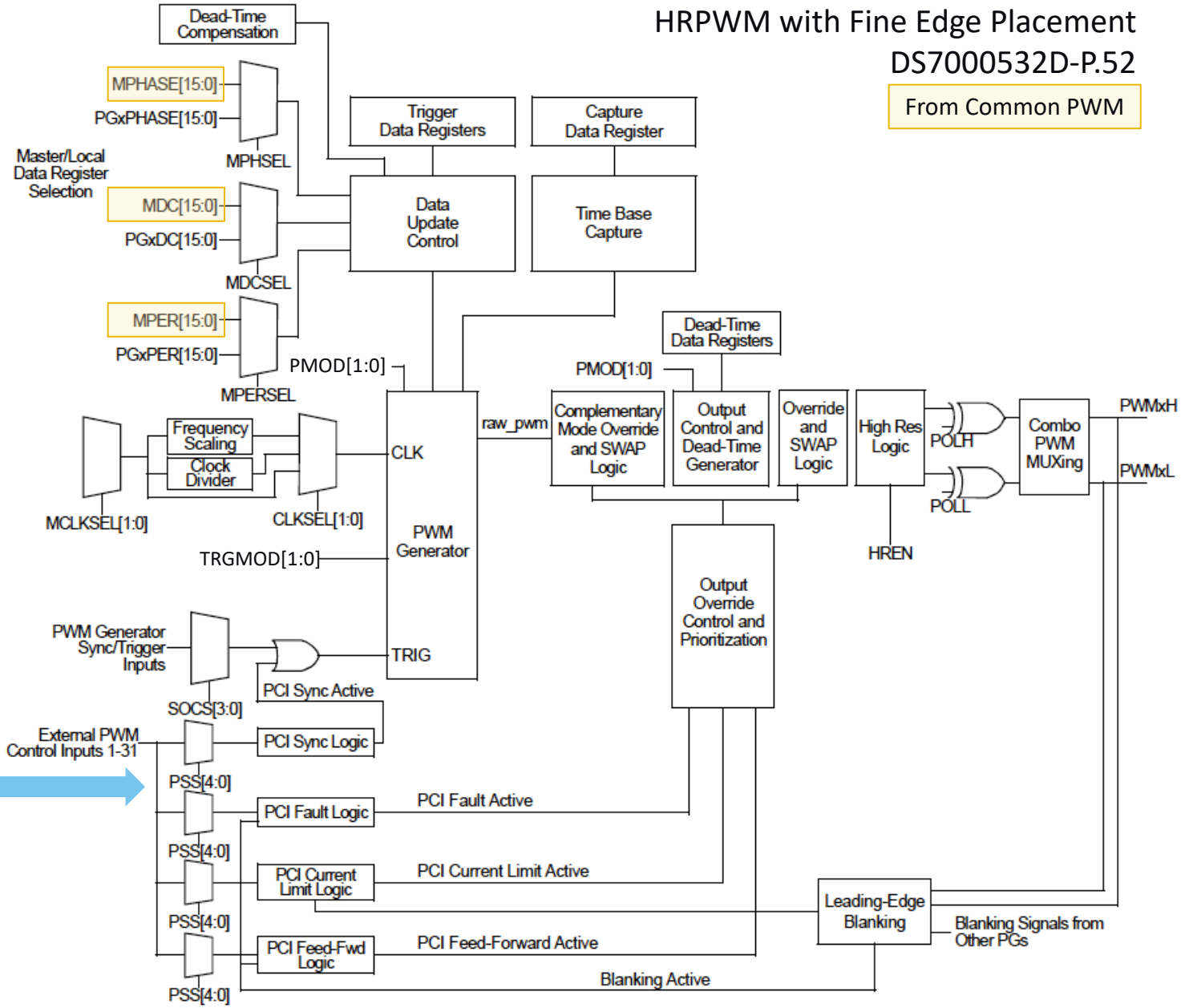
bit 4-0 **PSS[4:0]:** PCI Source Selection bits

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

HRPWM with Fine Edge Placement
DS7000532D-P.52

From Common PWM



Single PWM Generator

dsPIC33CK256MP508
Data Sheet DS70005349H P.288
Bits 4:0 – PSS[4:0] PCI Source Selection

HRPWM with Fine Edge Placement
DS7000532D-P.52

From Common PWM

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'

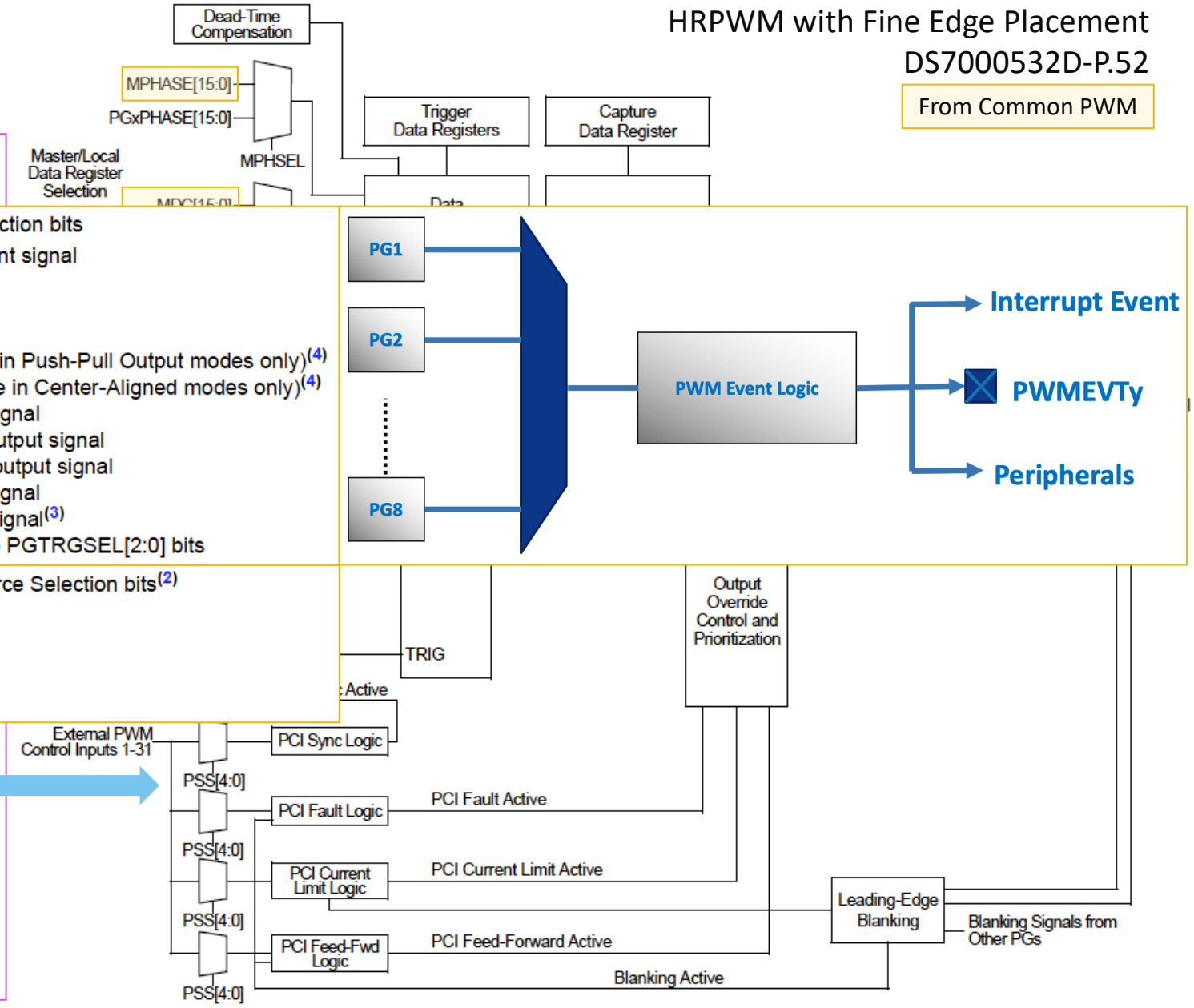
EVTySEL[3:0]: PWM Event Selection bits

- 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

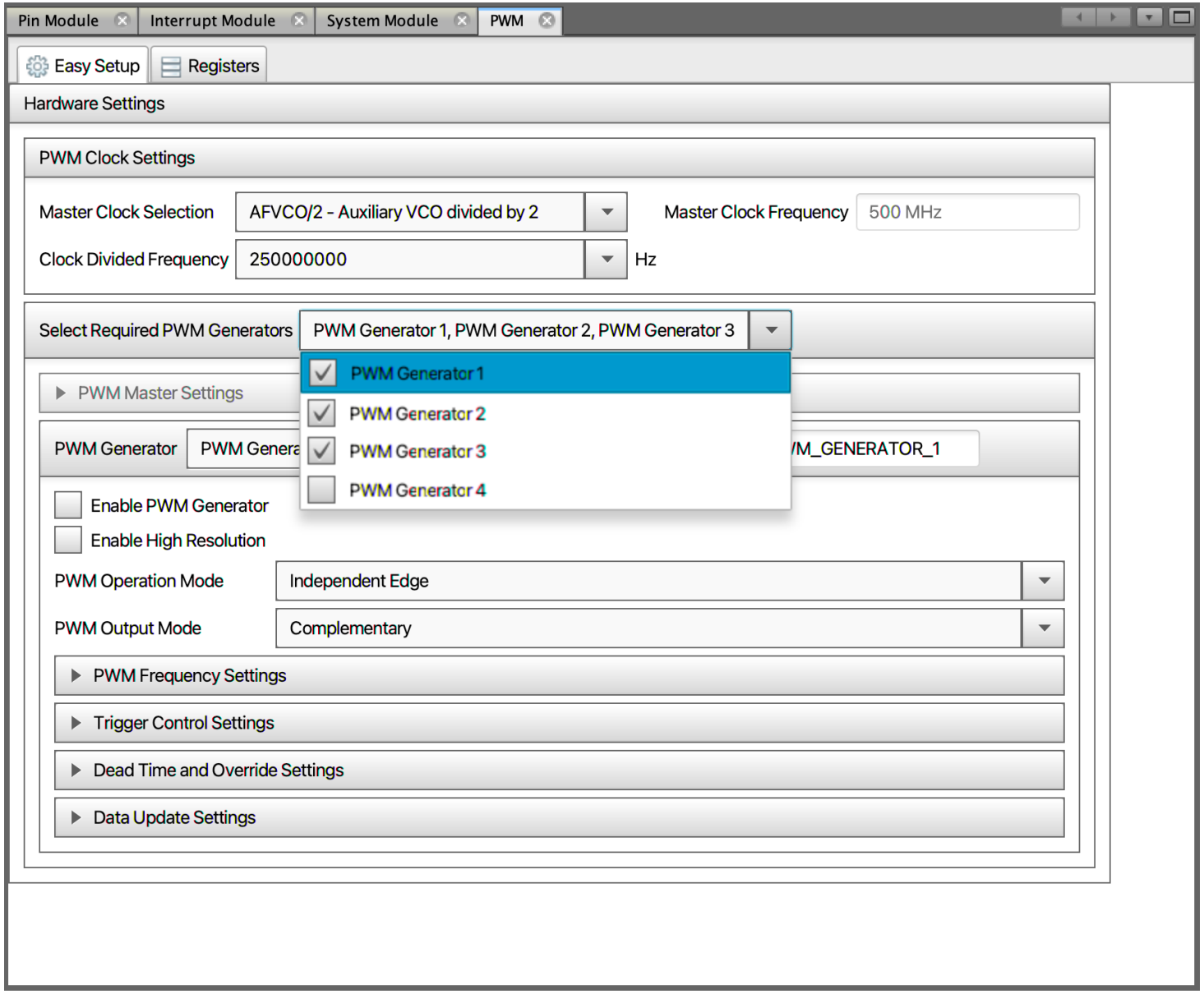
EVTyPGS[2:0]: PWM Event Source Selection bits(2)

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

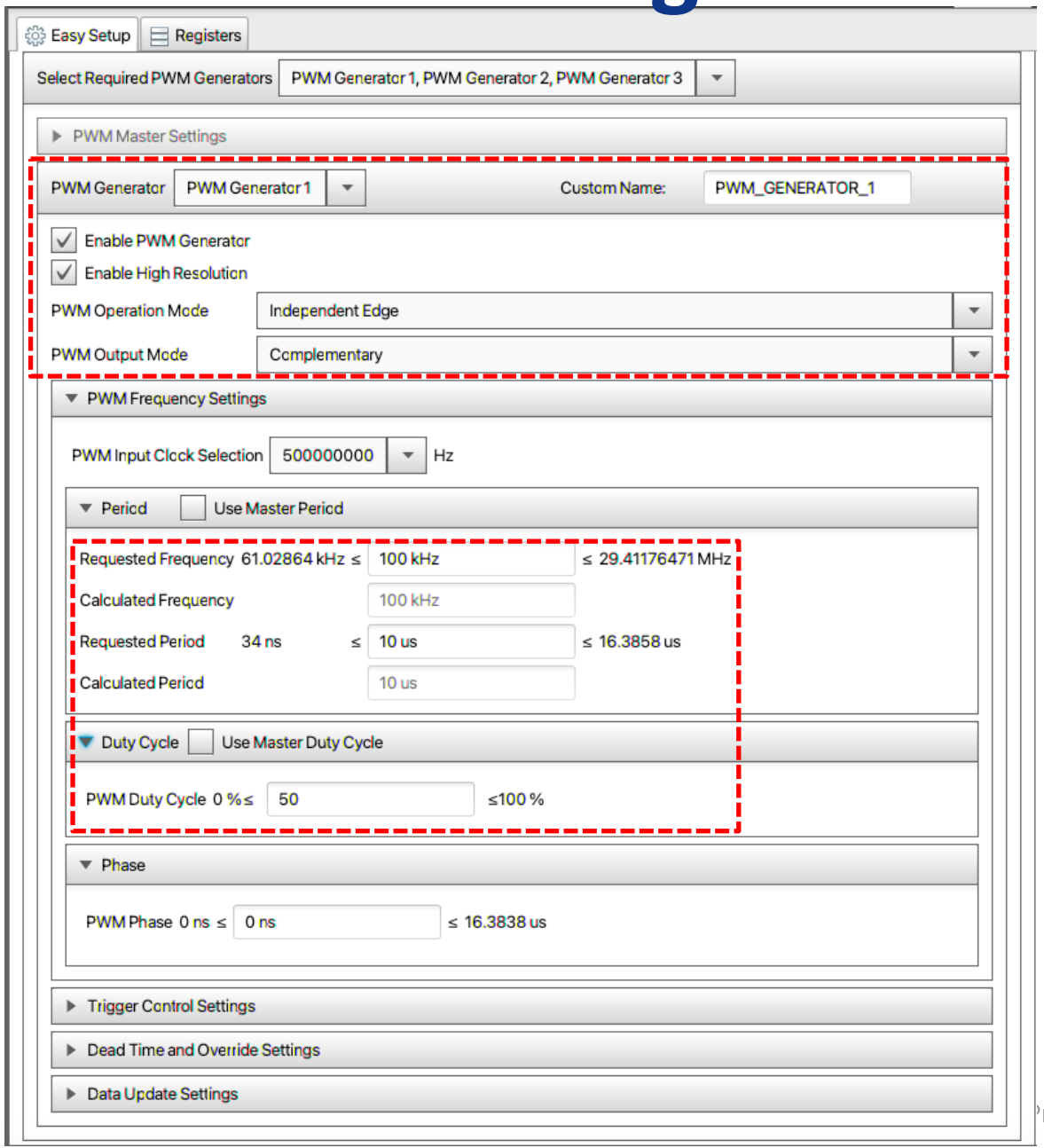
PGx PCI Sync

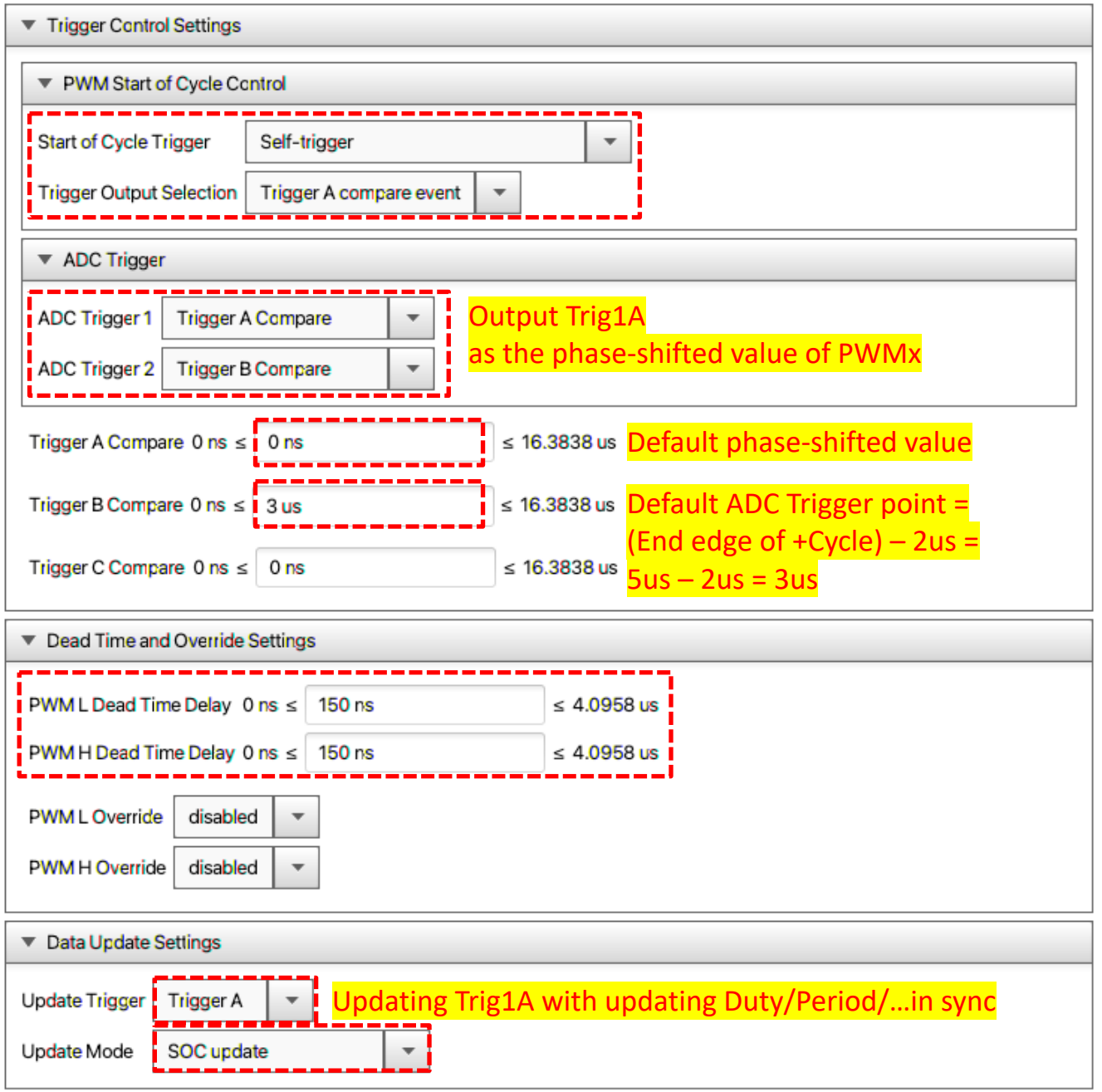


Select PWM1~3



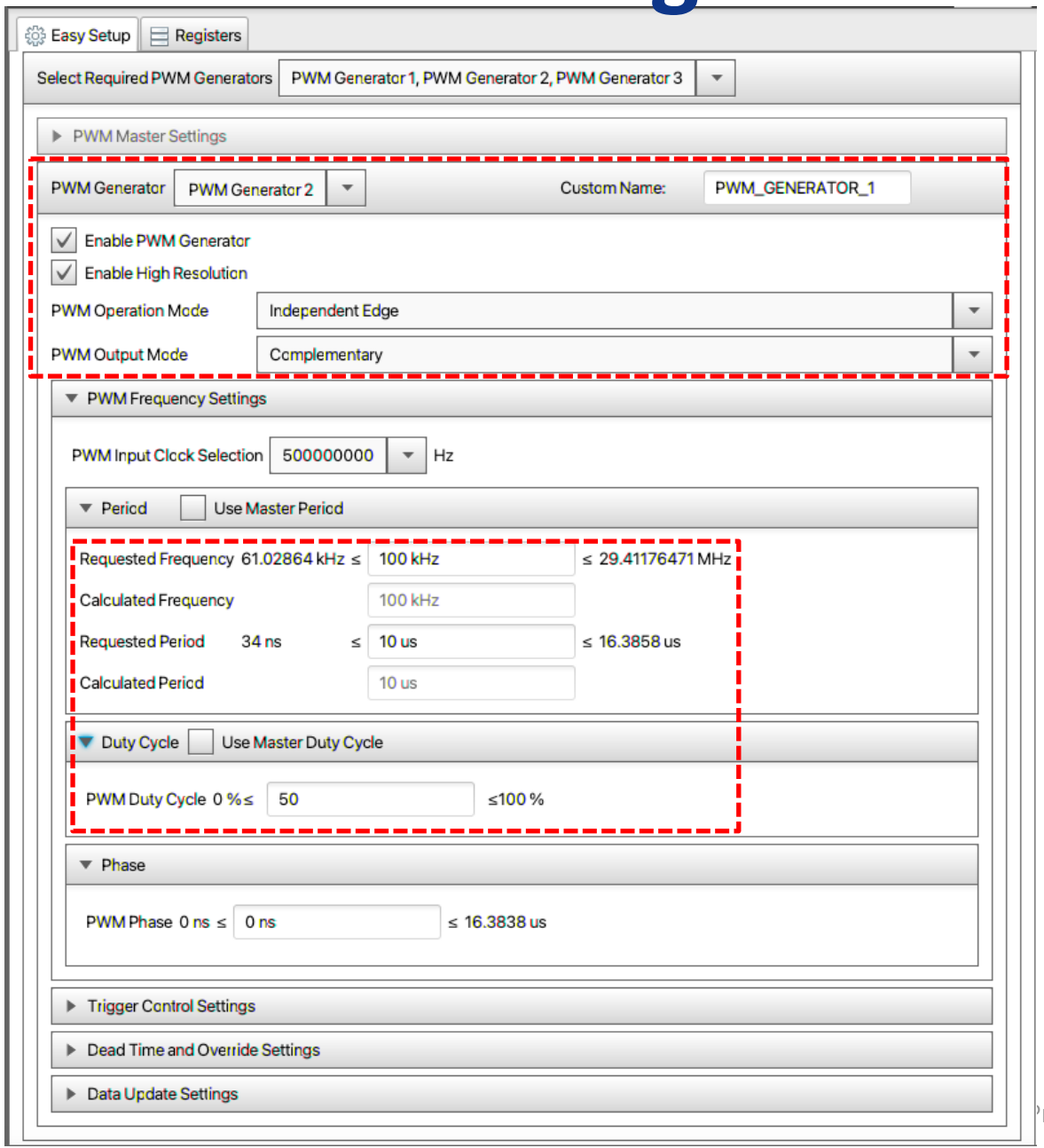
PWM1 Settings

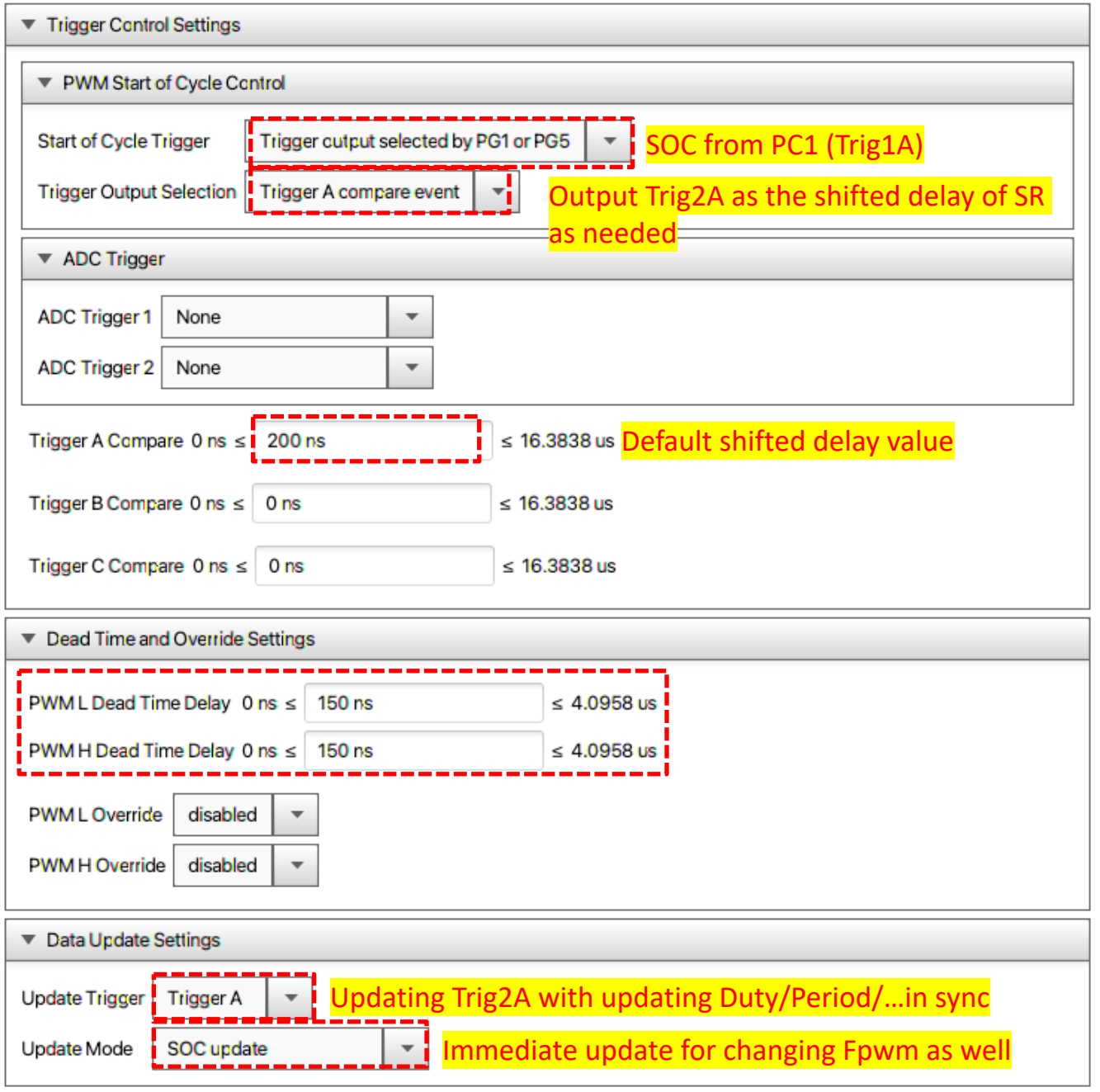




MICROCHIP

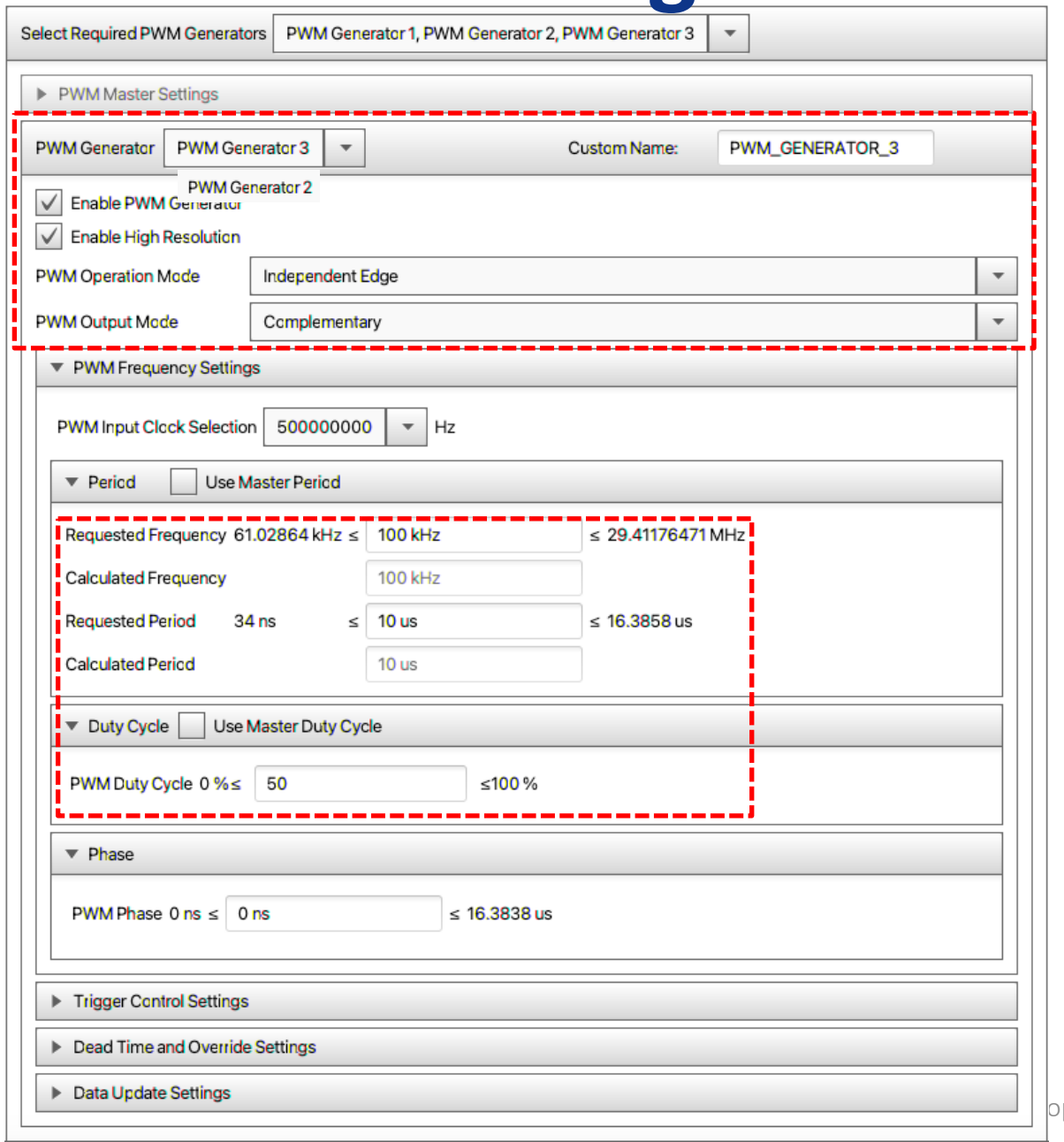
PWM2 Settings

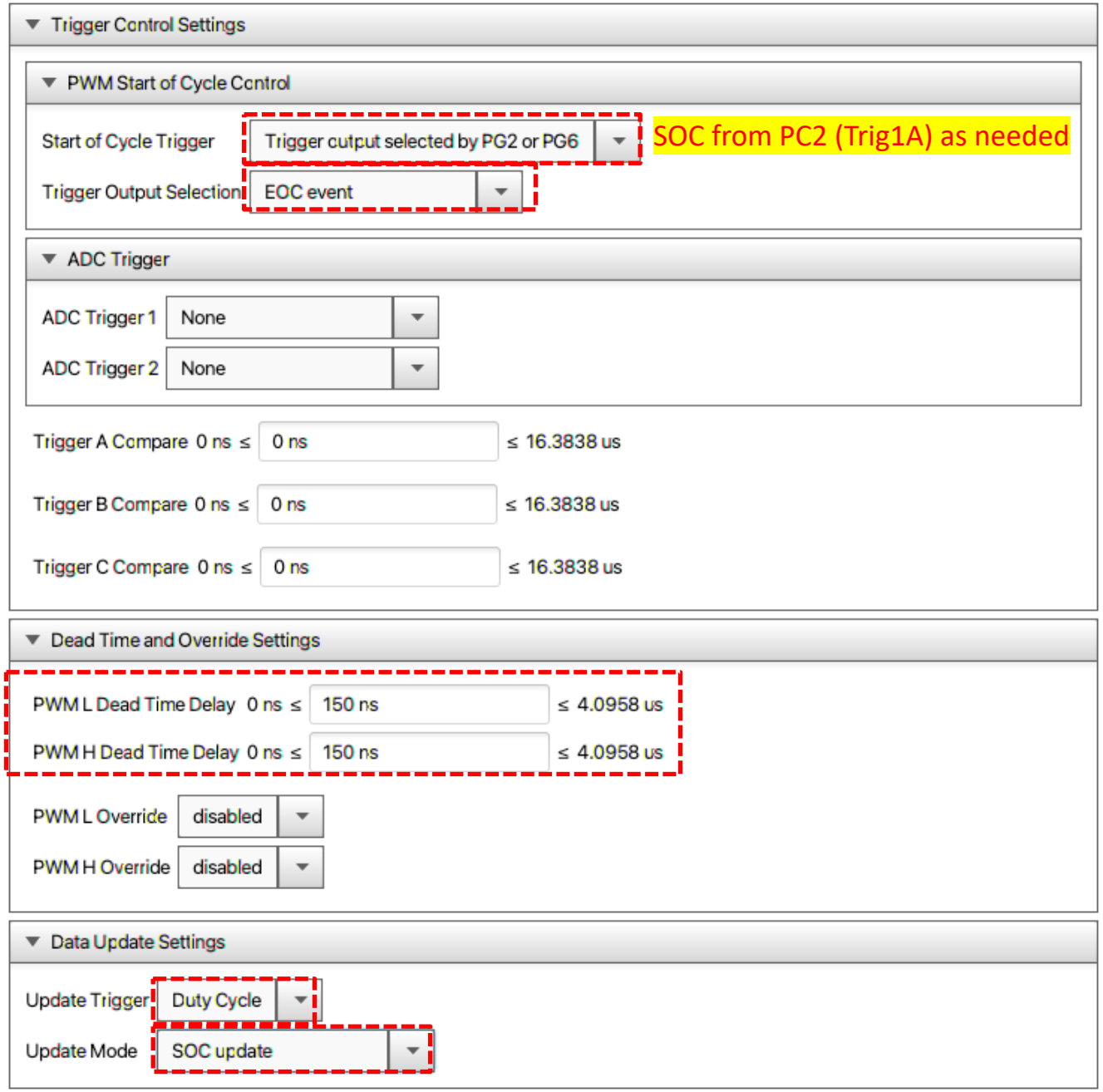




MICROCHIP

PWM3 Settings

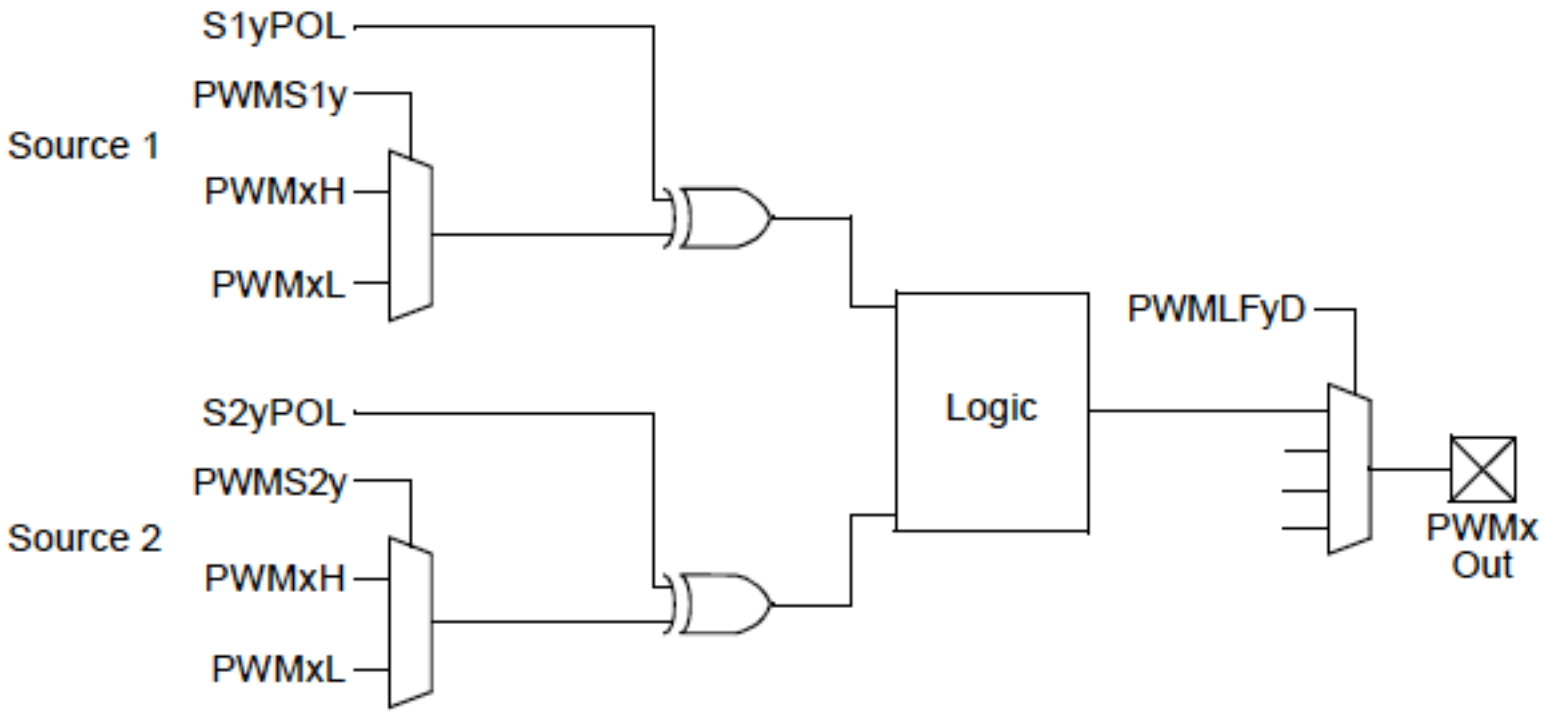


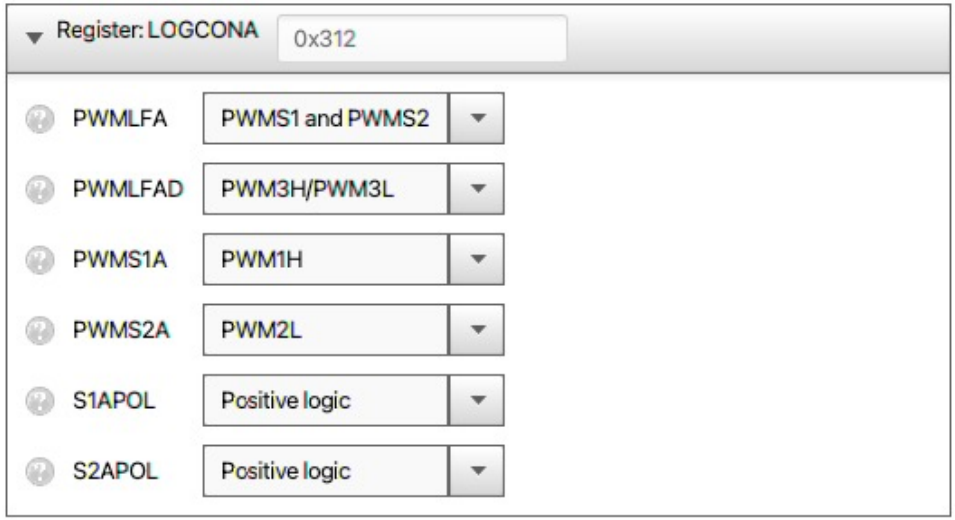


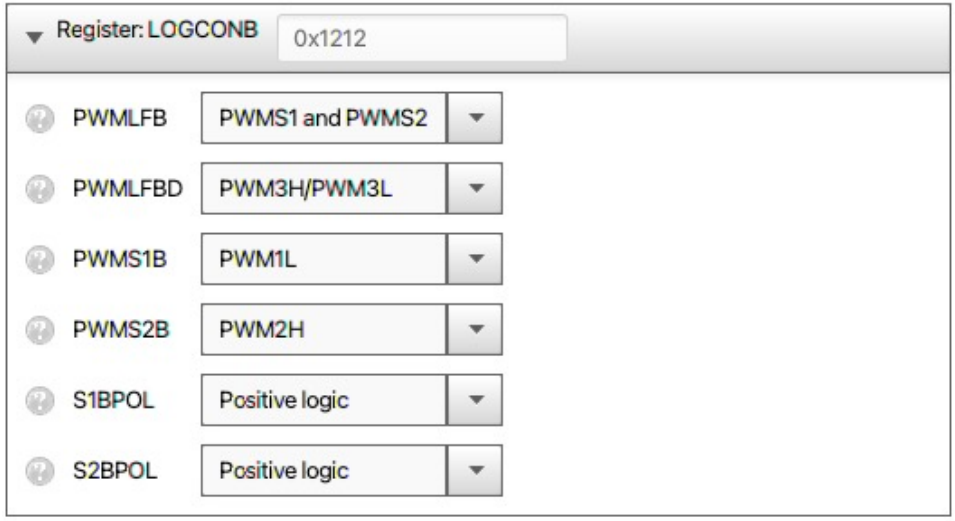
MICROCHIP

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

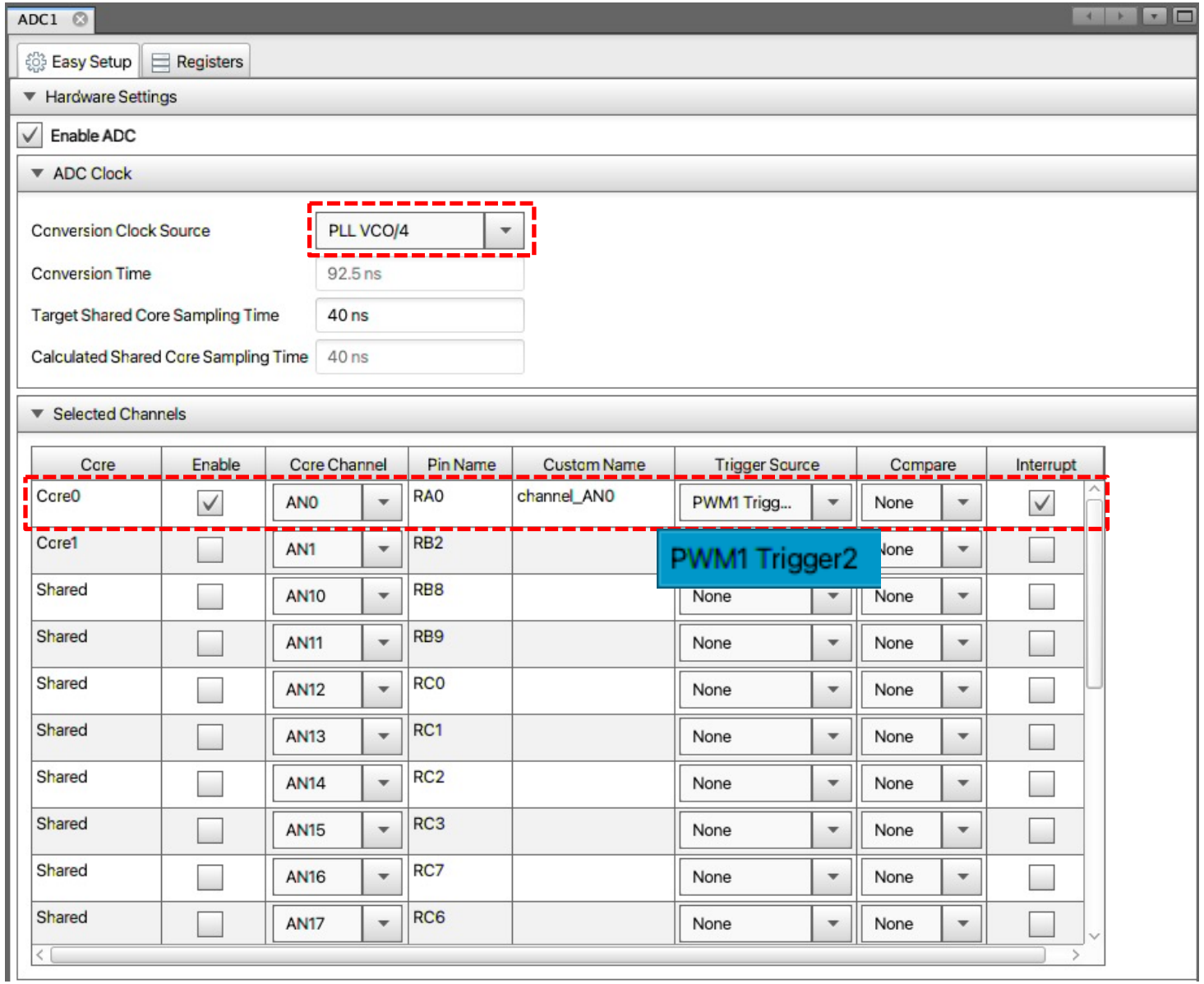




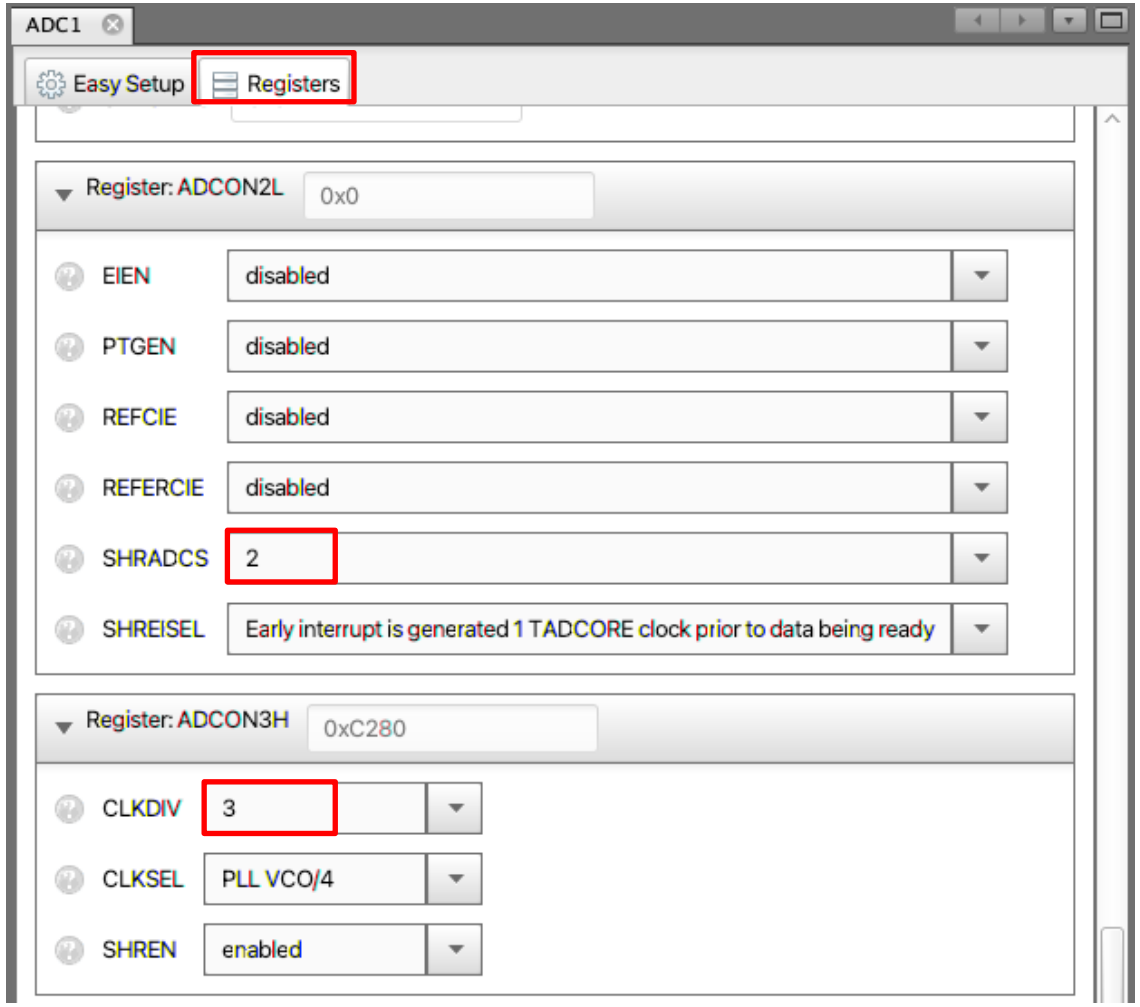


MICROCHIP

ADC/AN0 Settings

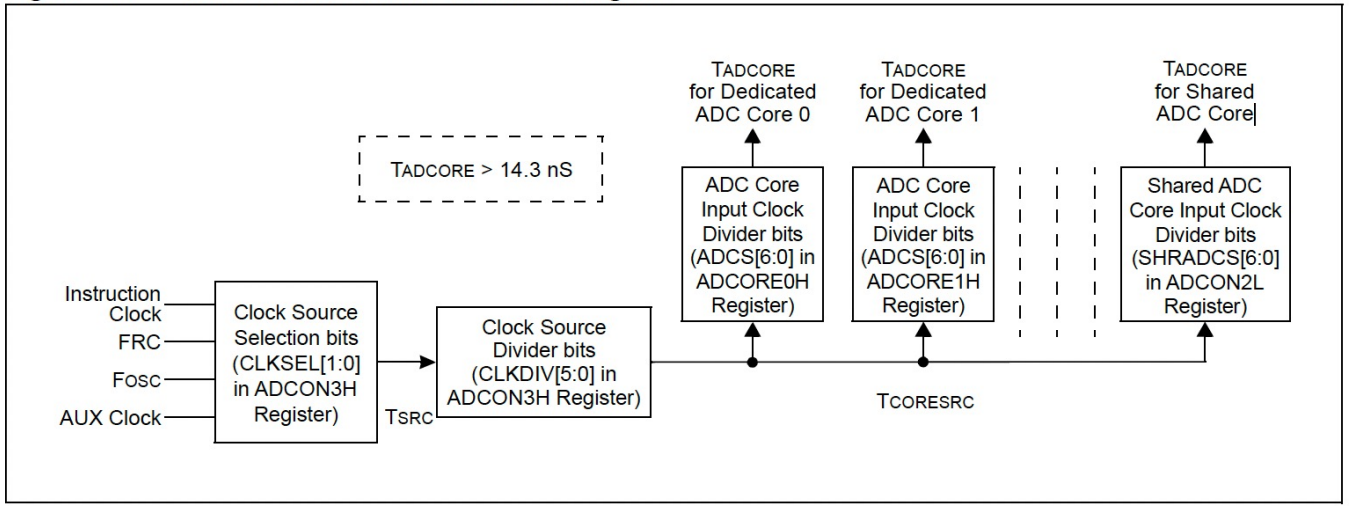


ADC/AN0 Settings

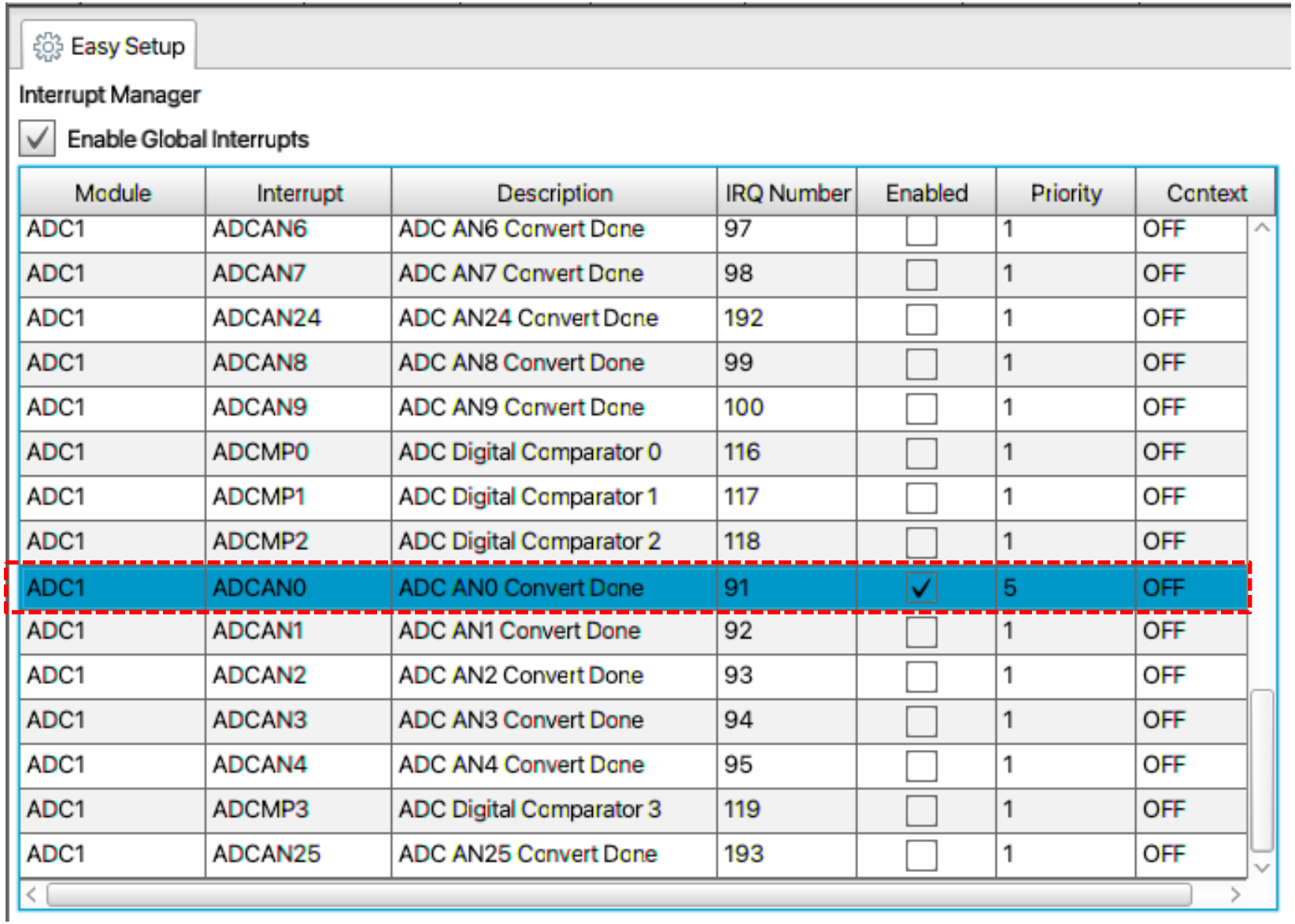


ADC Shared Core			
ADC Module Clock Source	400	MHz	PLL VOC/4
ADCON2L.SHRADCS	2		Shared ADC Core Input Clock Divider bits
ADCON3H.CLKDIV	3		ADC Module Clock Source Divider bits
Fadcore	66.67	MHz	Clock Source / SHRADCS / CLKDIV
Tadcore	15.00	ns	1/Fadcore >= 14.3ns

Figure 5-1: ADC Module Clock Path Block Diagram



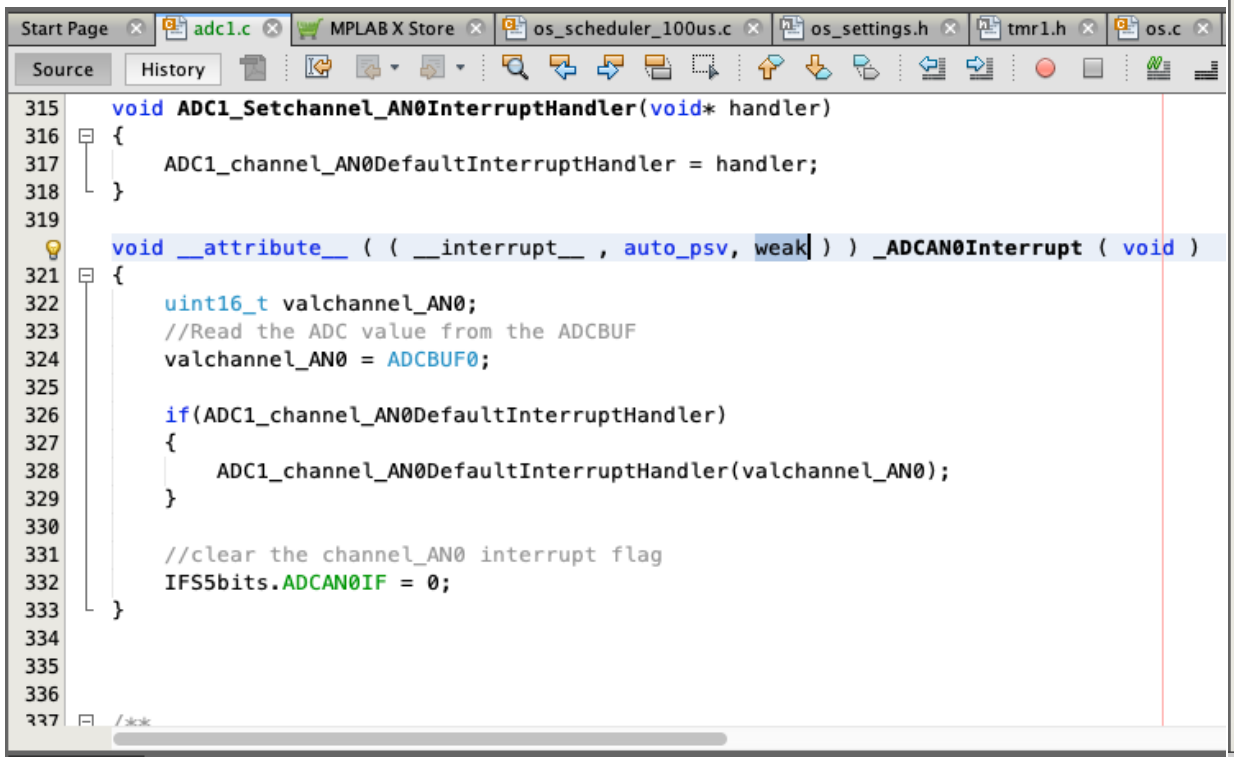
Interrupt Manager

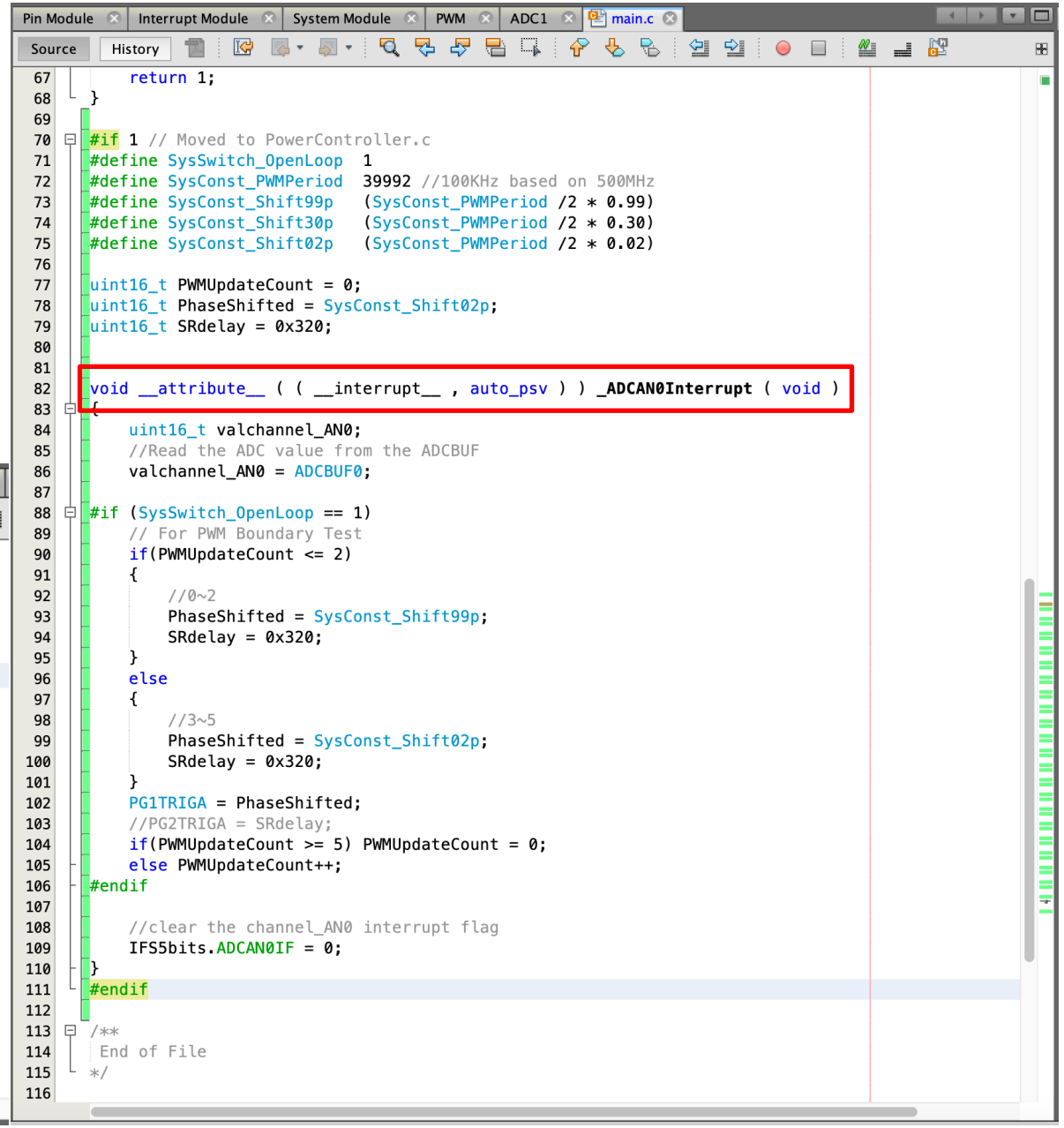


Module	Interrupt	Description	IRQ Number	Enabled	Priority	Context
ADC1	ADCAN6	ADC AN6 Convert Done	97	☐	1	OFF
ADC1	ADCAN7	ADC AN7 Convert Done	98	☐	1	OFF
ADC1	ADCAN24	ADC AN24 Convert Done	192	☐	1	OFF
ADC1	ADCAN8	ADC AN8 Convert Done	99	☐	1	OFF
ADC1	ADCAN9	ADC AN9 Convert Done	100	☐	1	OFF
ADC1	ADCMP0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

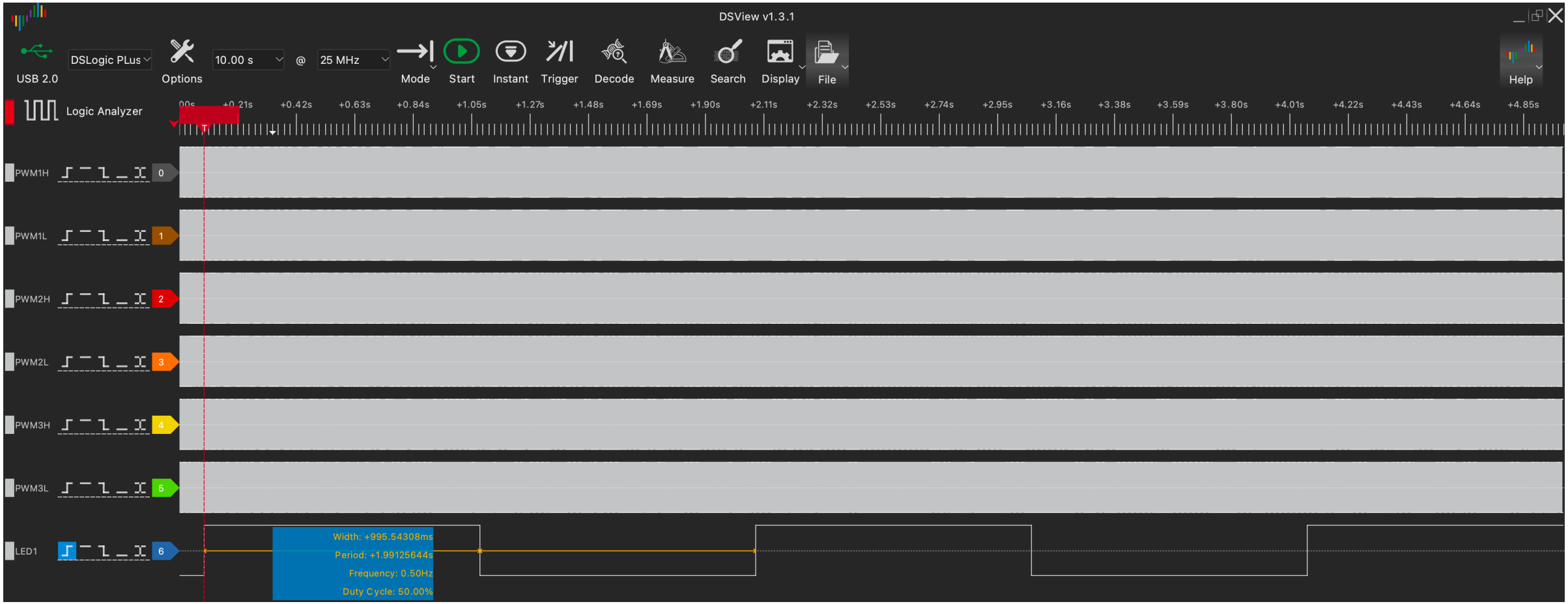
AN0 ISR in main.c

- Copied from adc1.c
- Removed “weak”

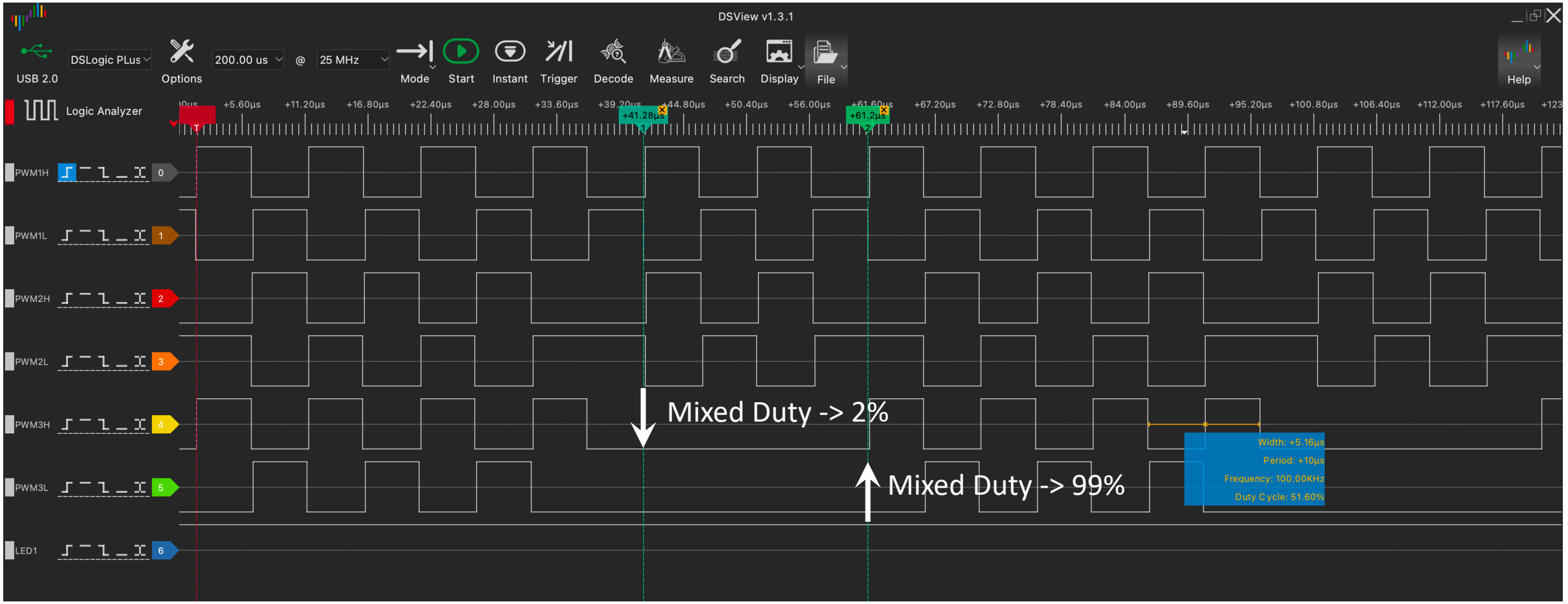




LED1 Toggling

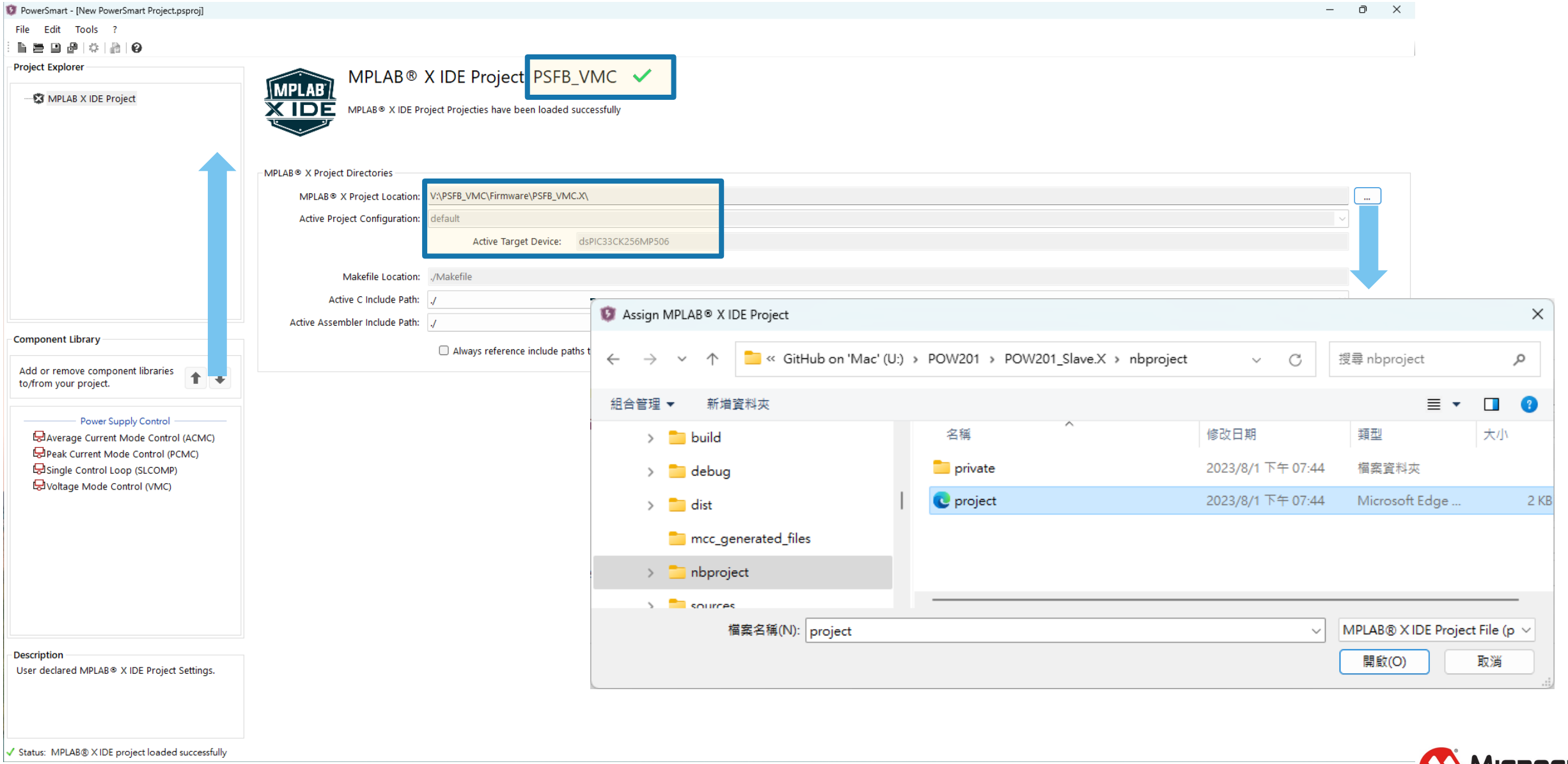


Boundary Test Between 2% ⇔ 99% shifted

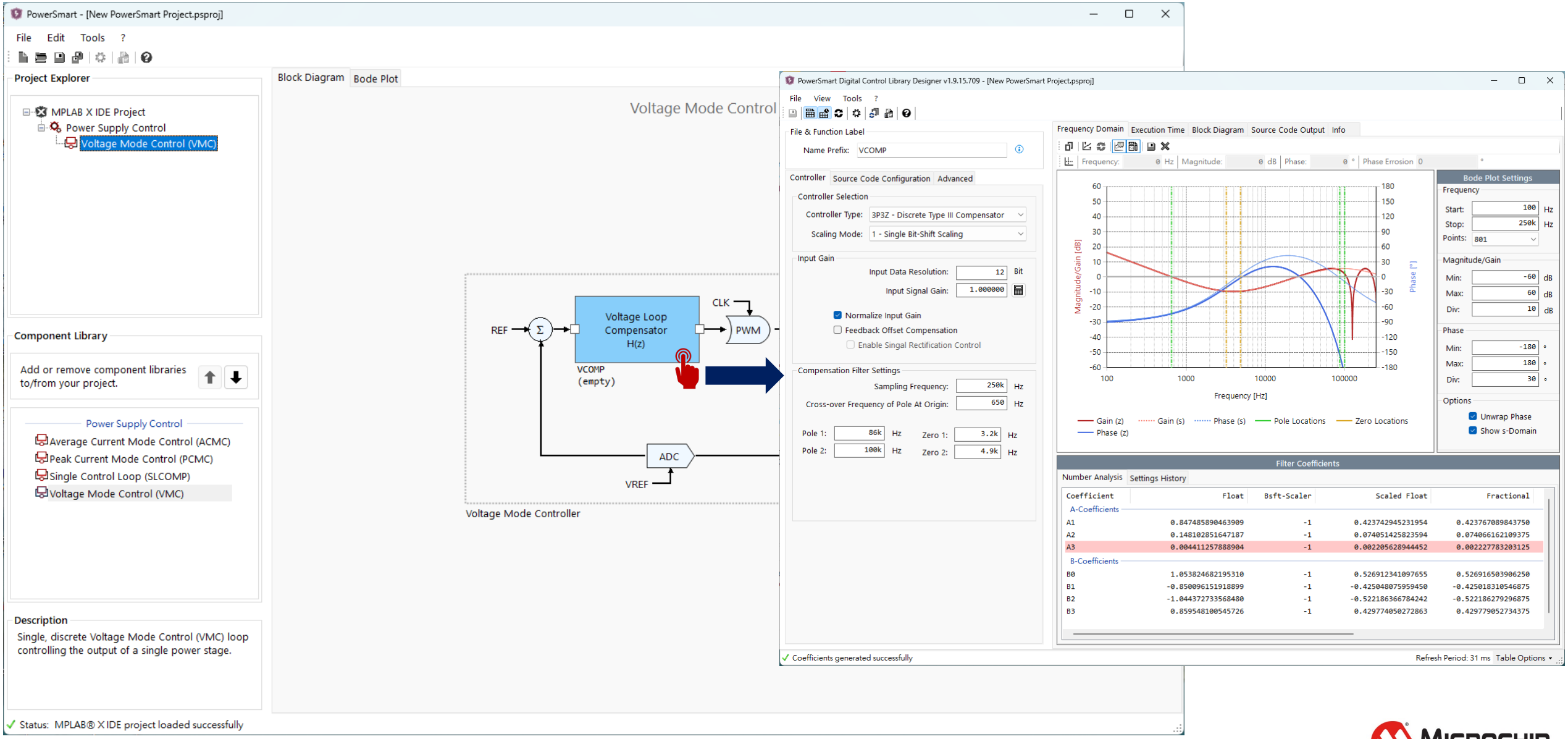


PowerSmart™-DCLD

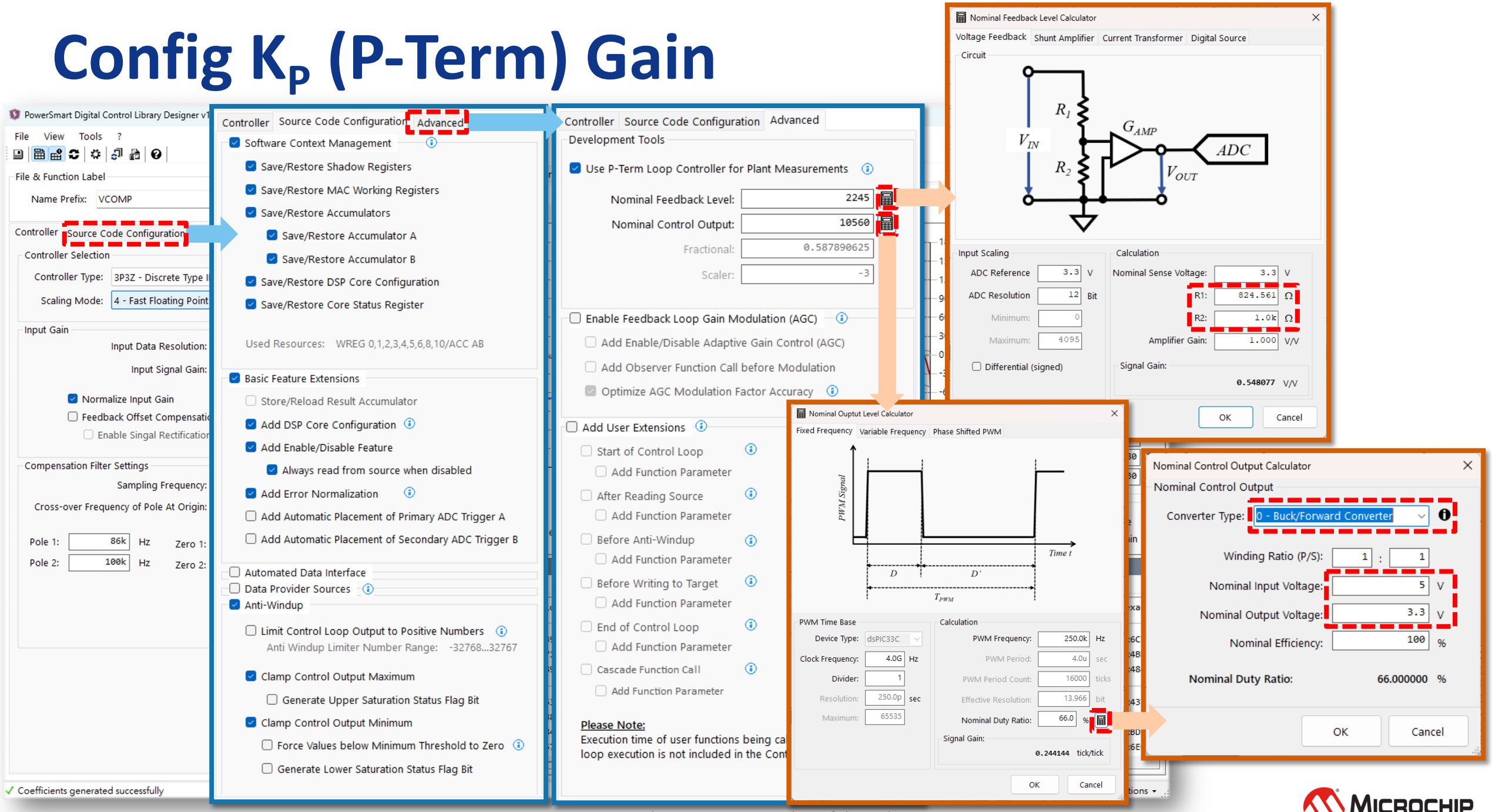
Add Control Mode to The Project



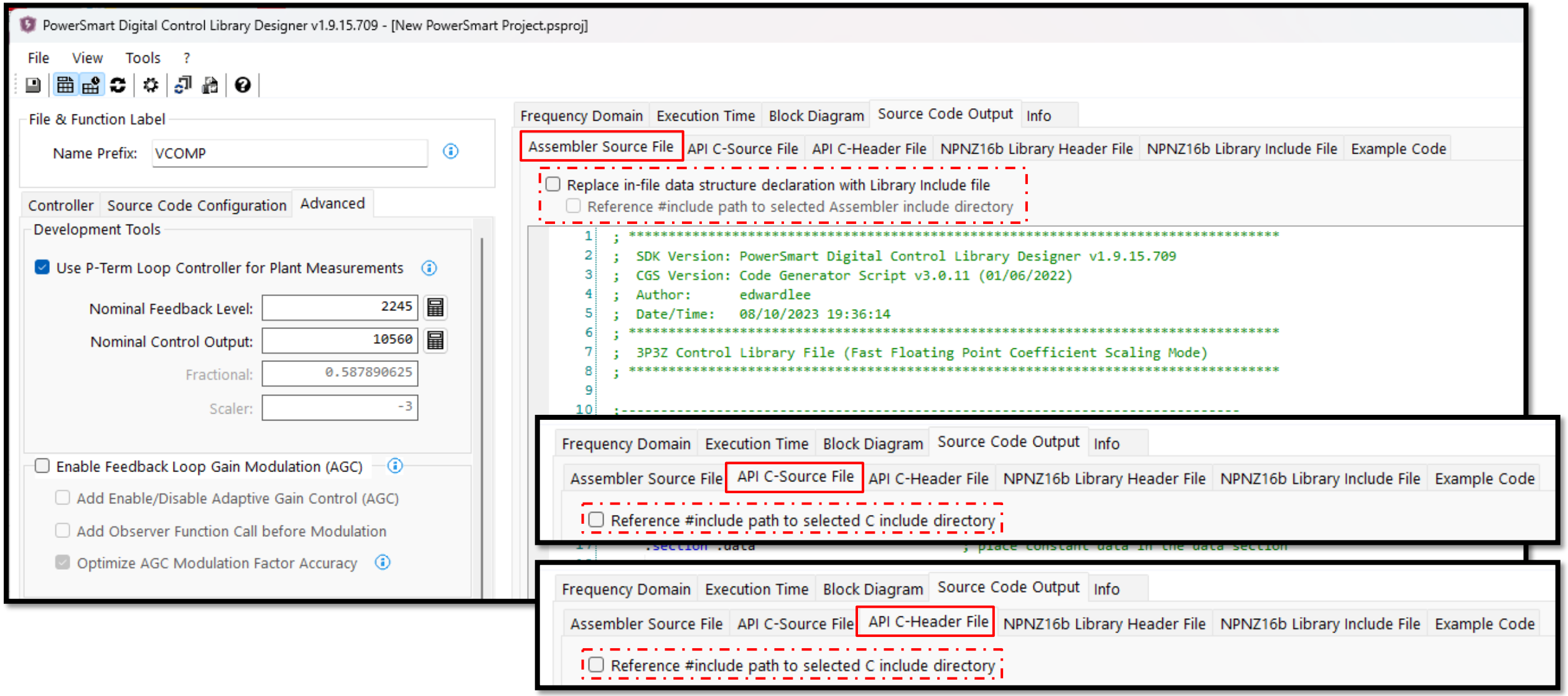
Switch to DCLD Window



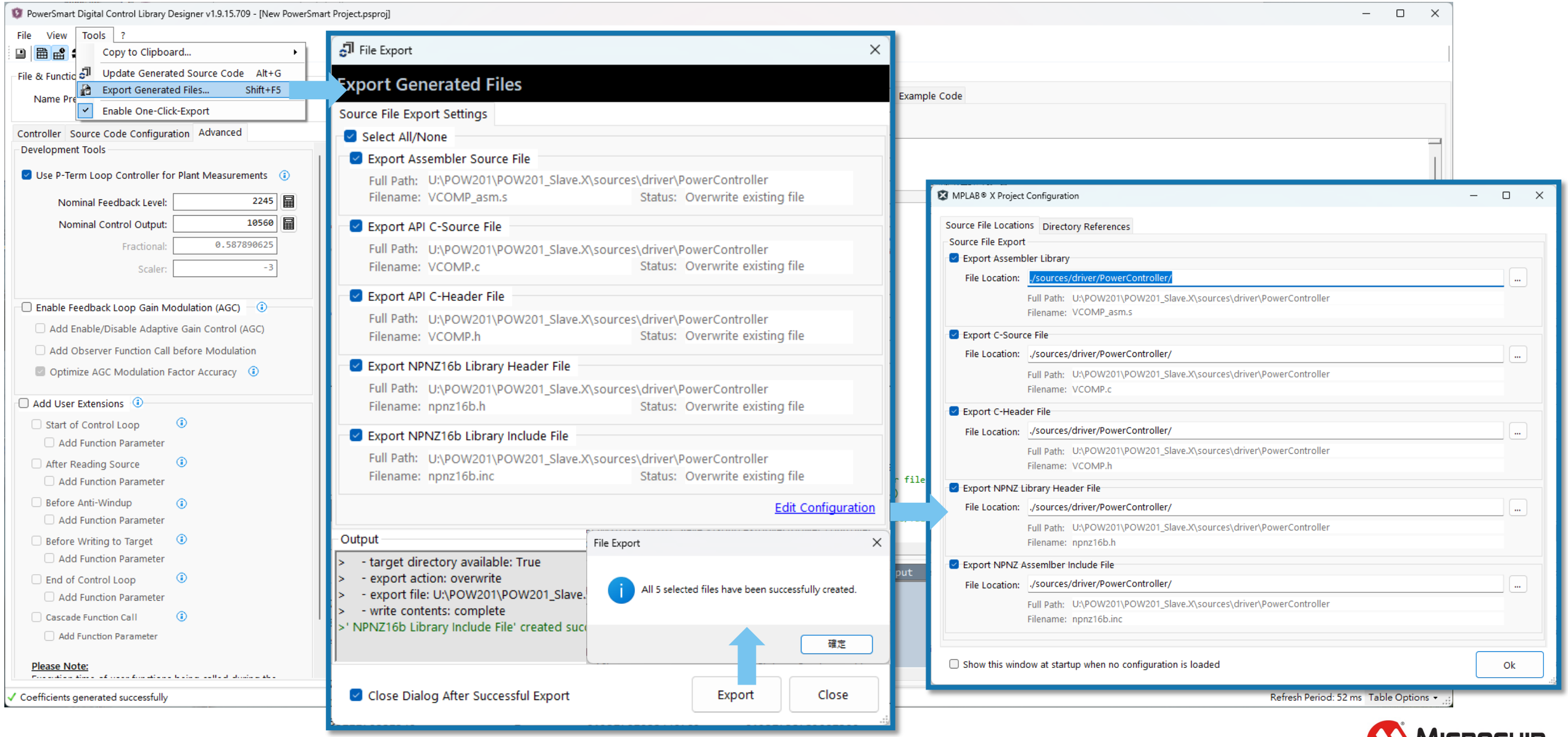
Config K_p (P-Term) Gain



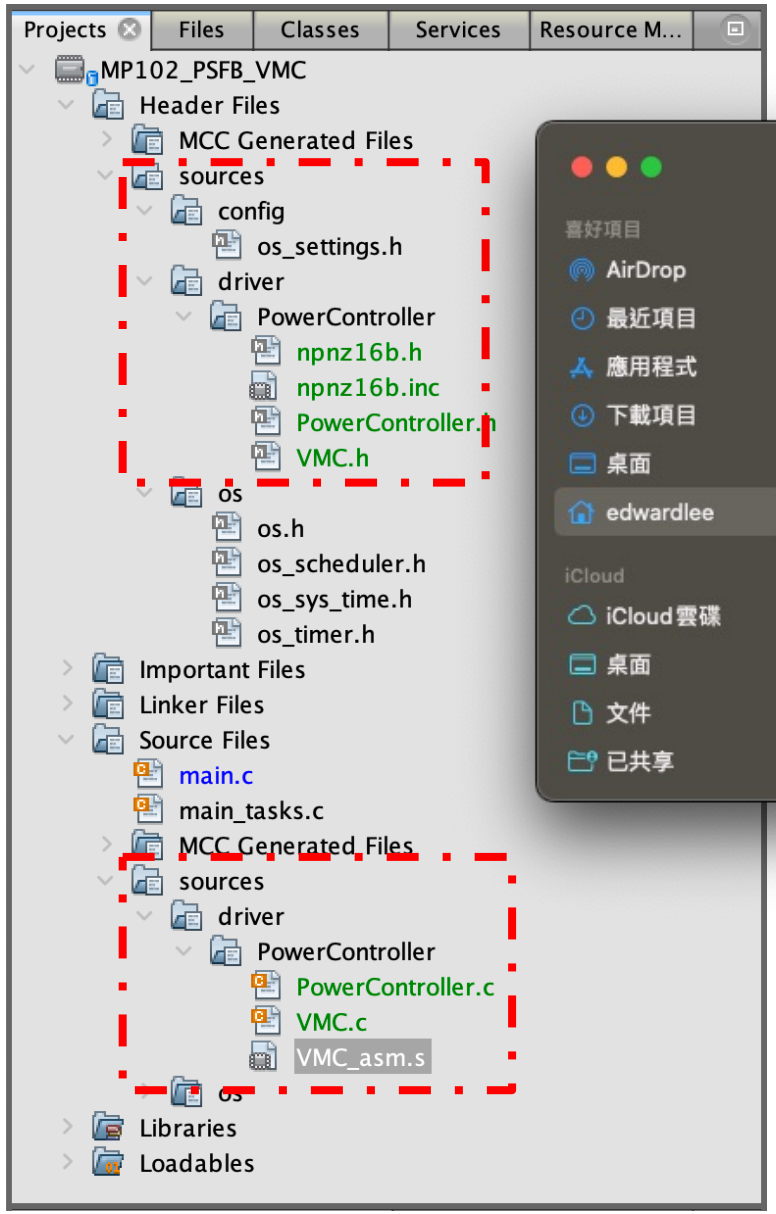
Setting for C-include Directory

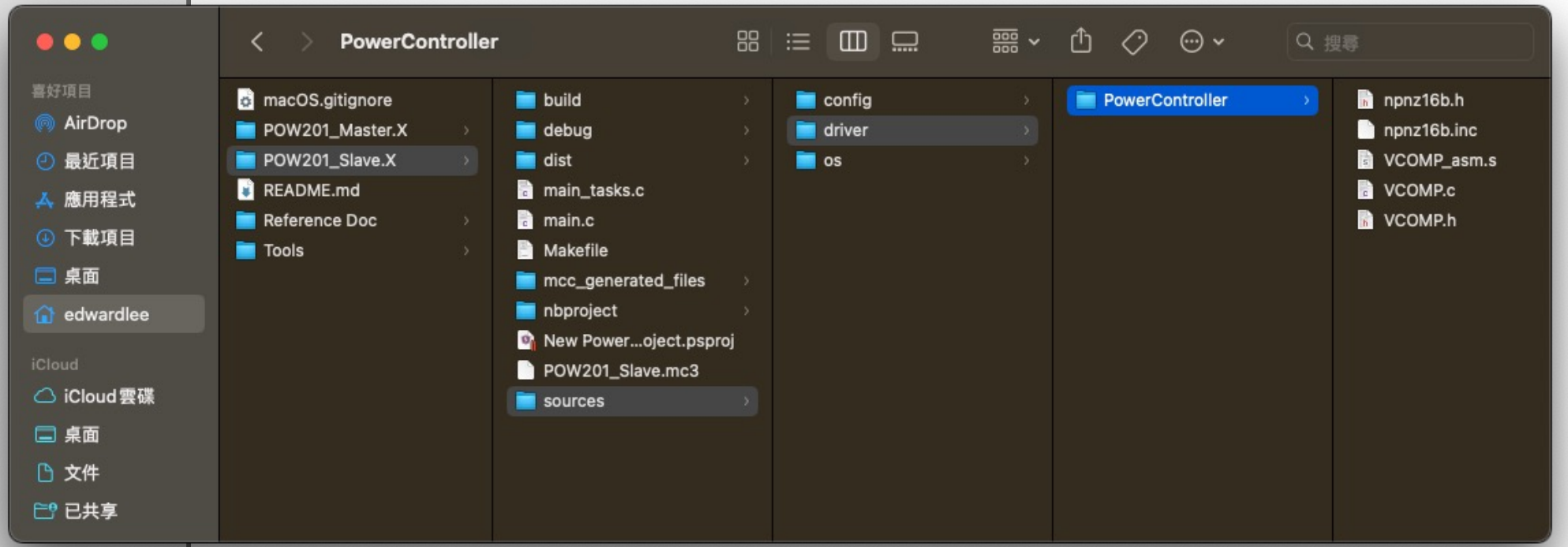


Export Compensator Sources Codes

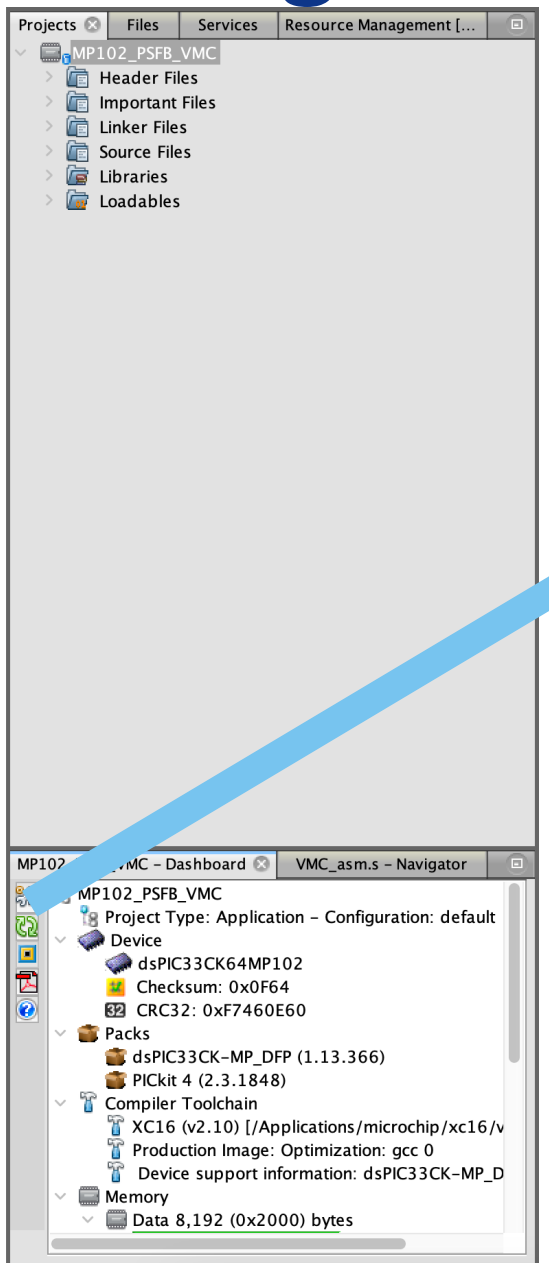


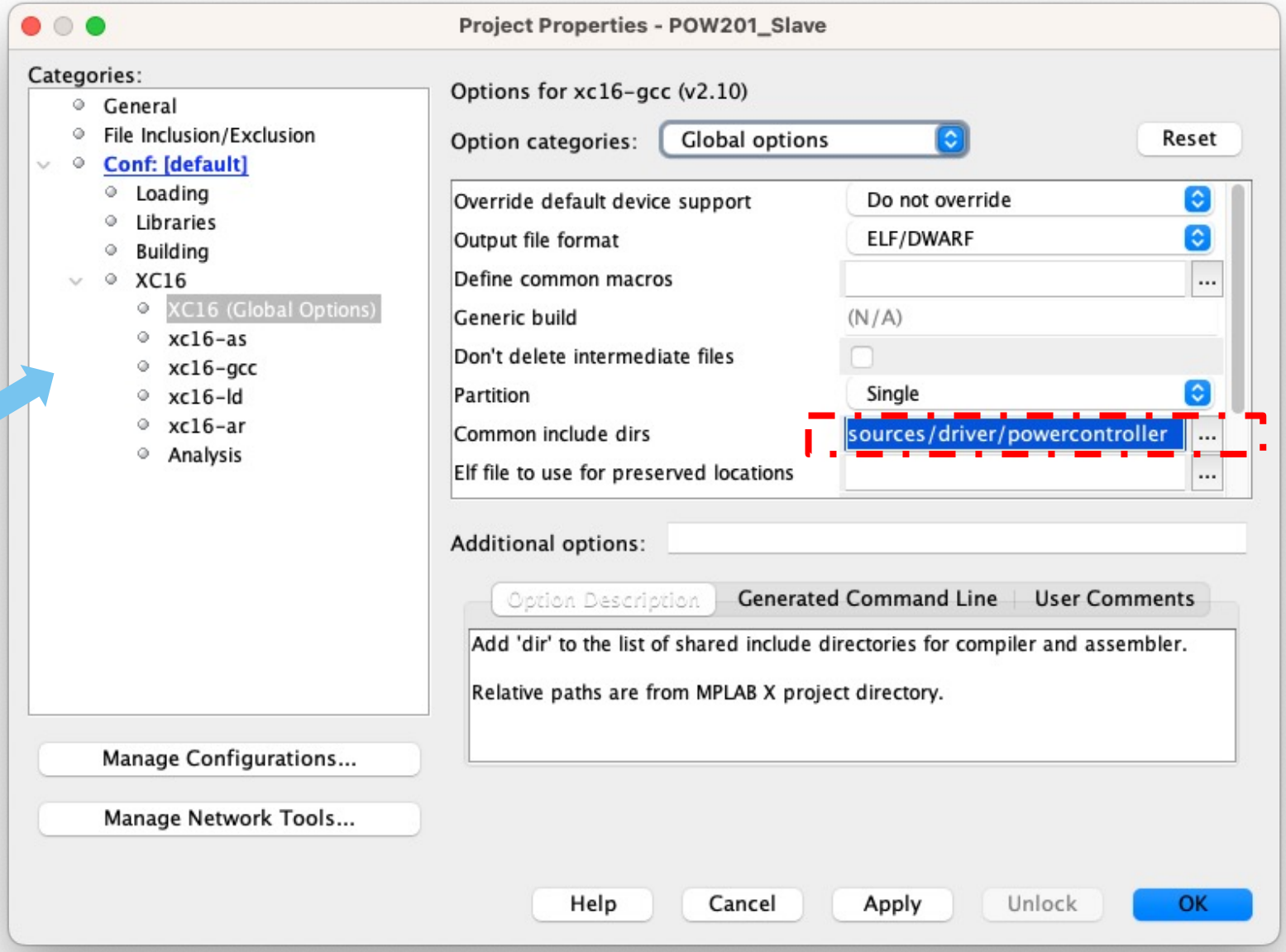
Add Control Libraries to the Project



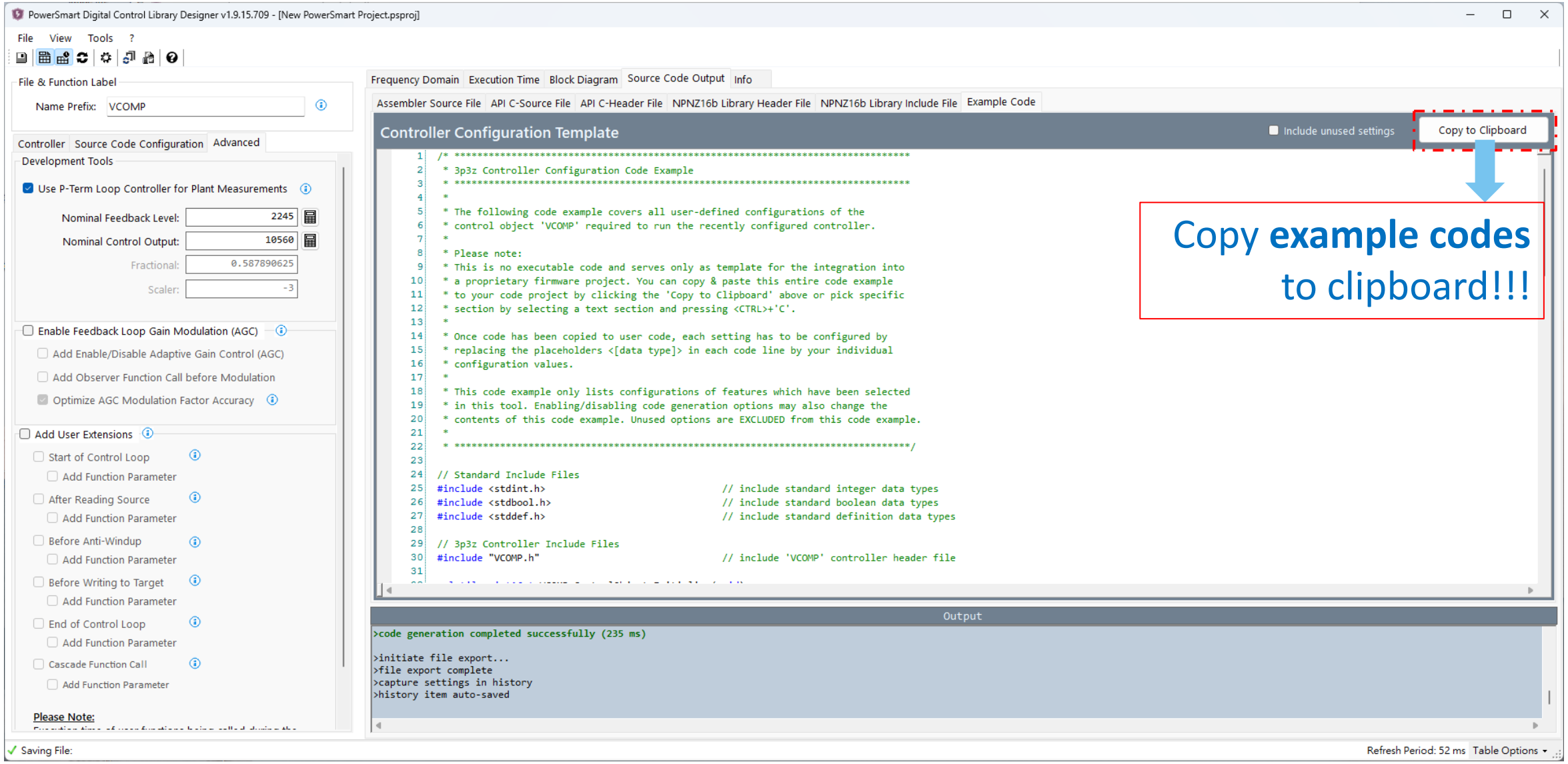


Setting for C-include Directory

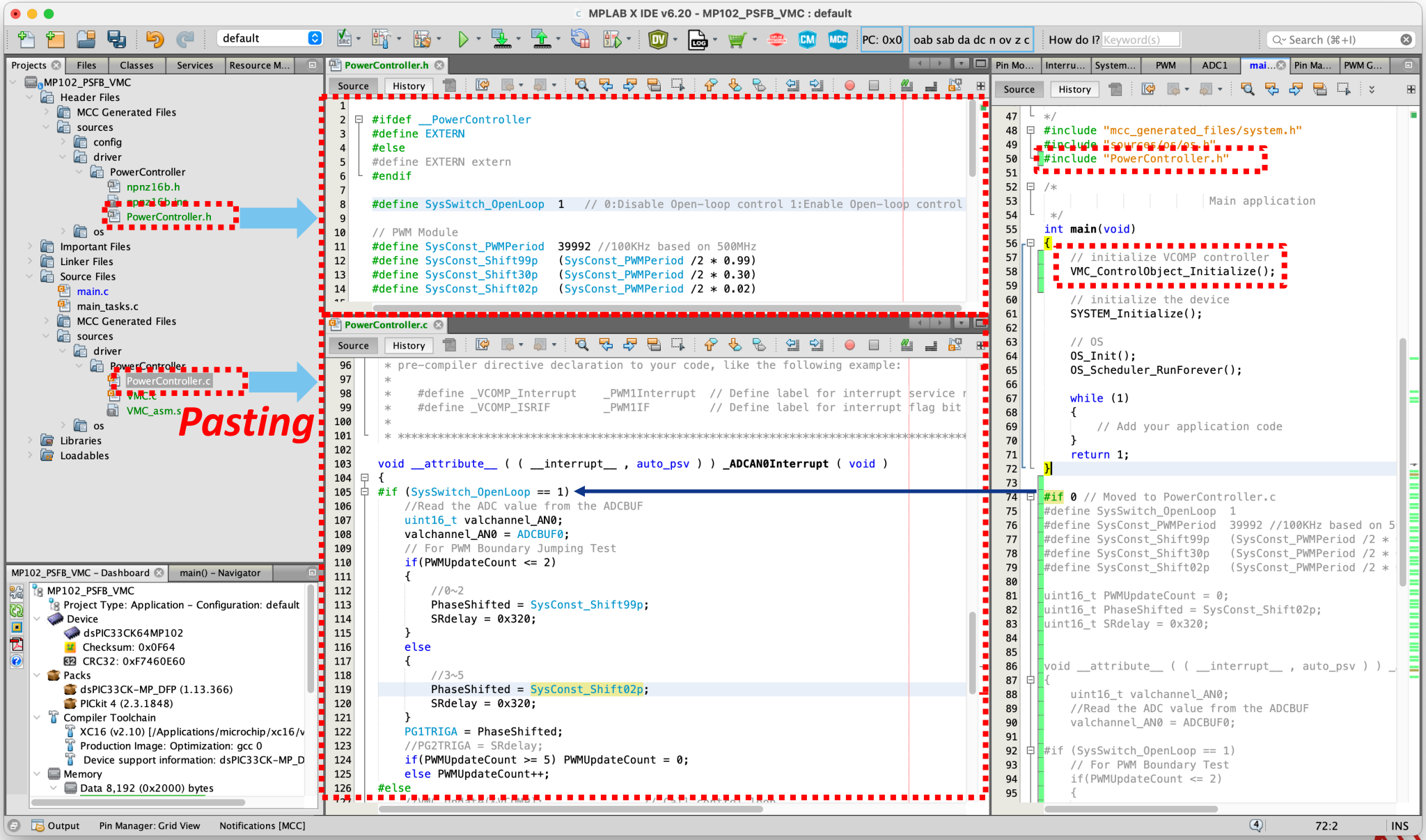




Copy example codes



New file with pasting example codes!



Setting Initial Registers

PowerController.c

```
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 factc
49      VMC.Ports.Source.NormFactor = 0x7FFF;      // Primary feedback normalization factc
50
51      // Configure Controller Primary Output Port
52      VMC.Ports.Target.ptrAddress = &PhaseShifted;    // Pointer to primary output tar
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 (us
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 ar
84
85      return(retval);
```

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

Control Loop

PowerController.c

Source | History

```
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 addir
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
```

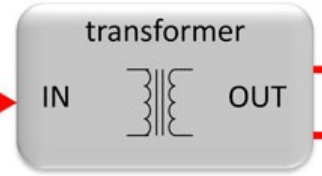
Soft-start

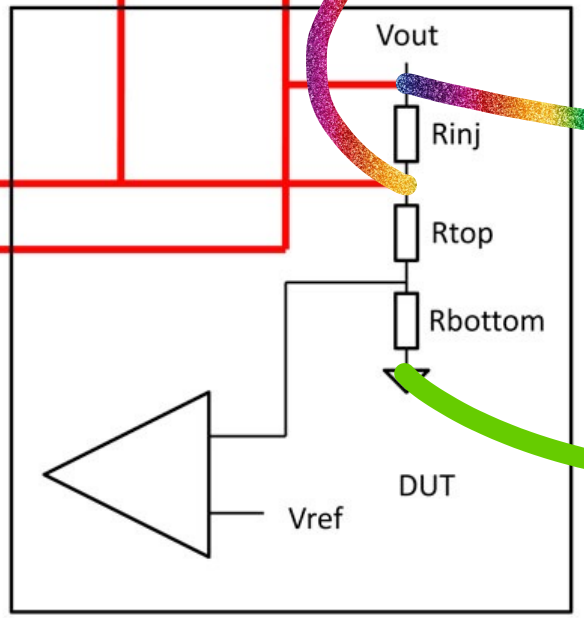
PowerController.c

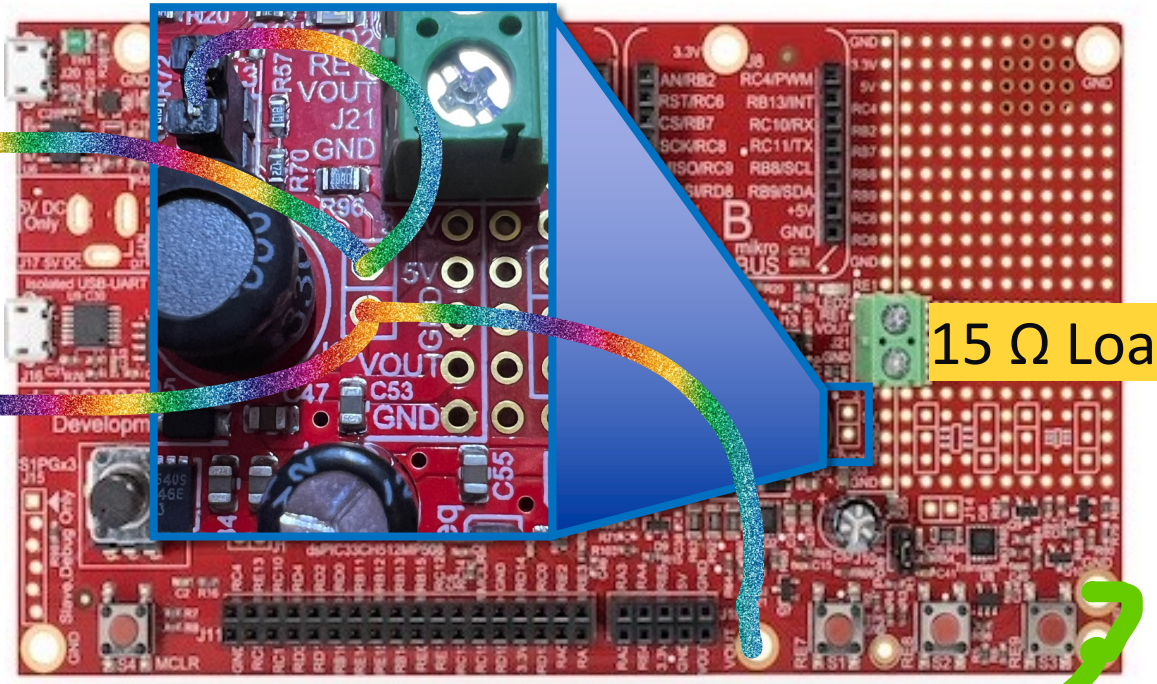
Source | History

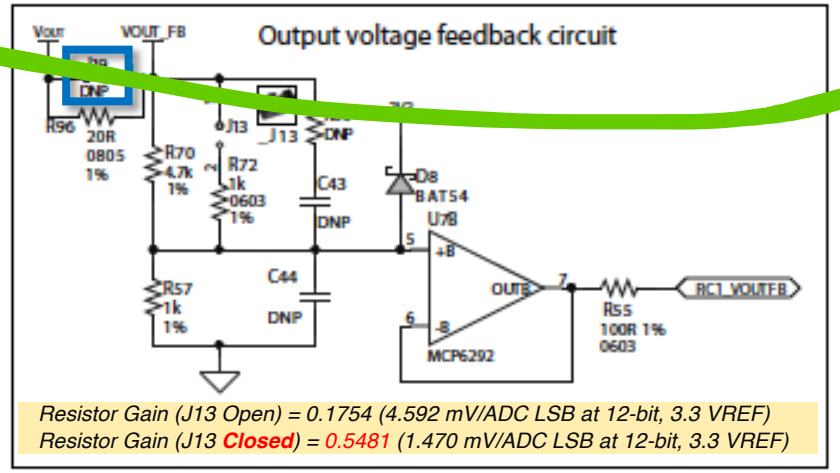
```
122         PG1TRIGA = PhaseShifted;
123         //PG2TRIGA = SRdelay;
124         if(PWMUpdateCount >= 5) PWMUpdateCount = 0;
125         else PWMUpdateCount++;
126     #else
127         //VMC_Update(&VCOMP);          // Call control loop
128         VMC_PTermUpdate(&VMC);         // Call P-Term control loop
129     #endif
130
131         //clear the channel_AN0 interrupt flag
132         IFS5bits.ADCAN0IF = 0;
133     }
134
135
136     //*****************************************************************************************
137     // Download latest version of this tool here:
138     //      https://www.microchip.com/powersmart
139     //*****************************************************************************************
140
141     // Simple Softstart
142     void VMC_Softstart(void)
143     {
144         VMC.status.bits.enabled = true;          // Enable controller
145         if(Output_Vref < VCOMP_VREF)
146         {
147             Output_Vref+=10;
148         }
149         else
150         {
151             Output_Vref = VCOMP_VREF;
152         }
153     }
154
```

Plant Measurement Using Bode-100

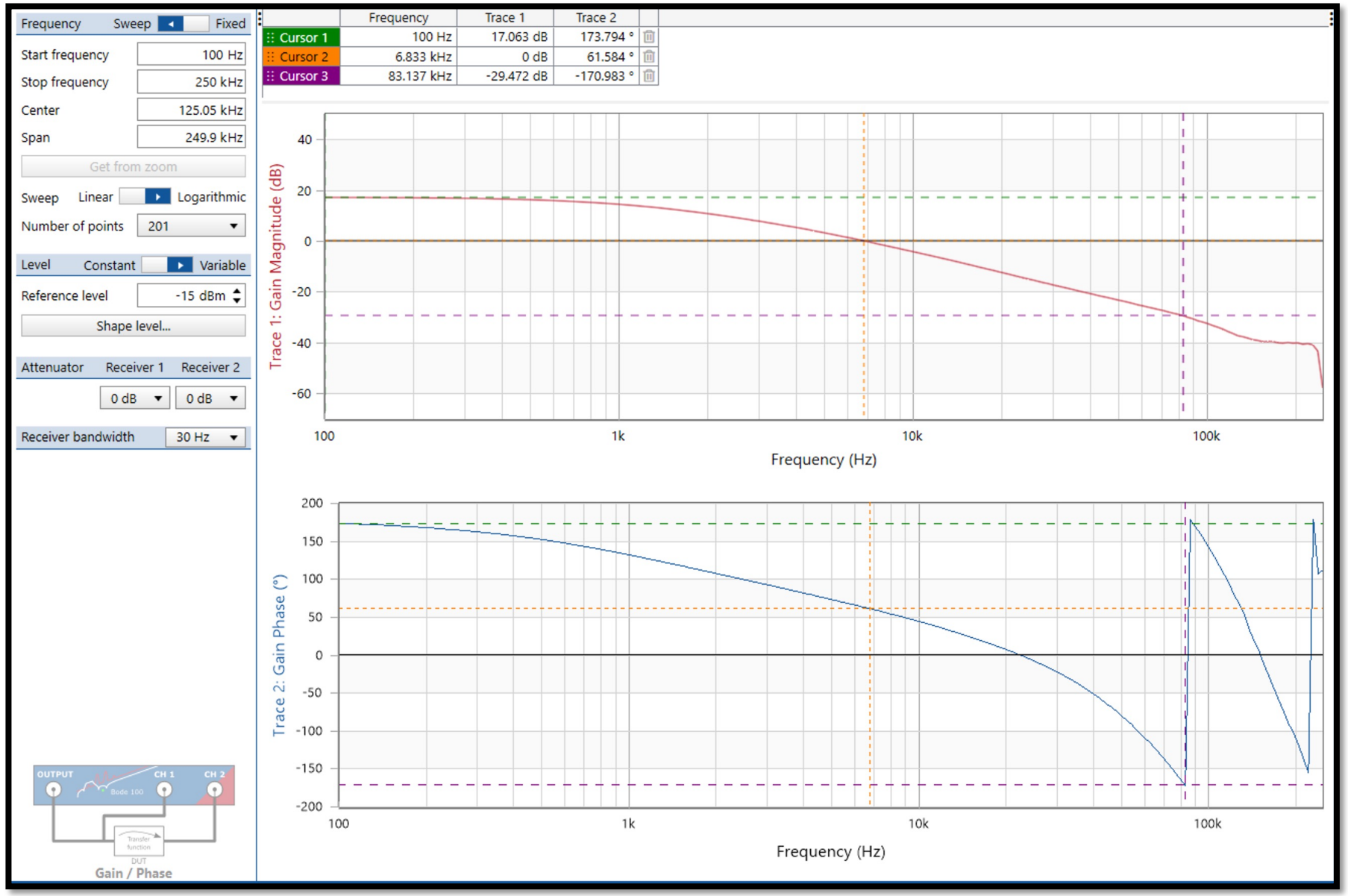




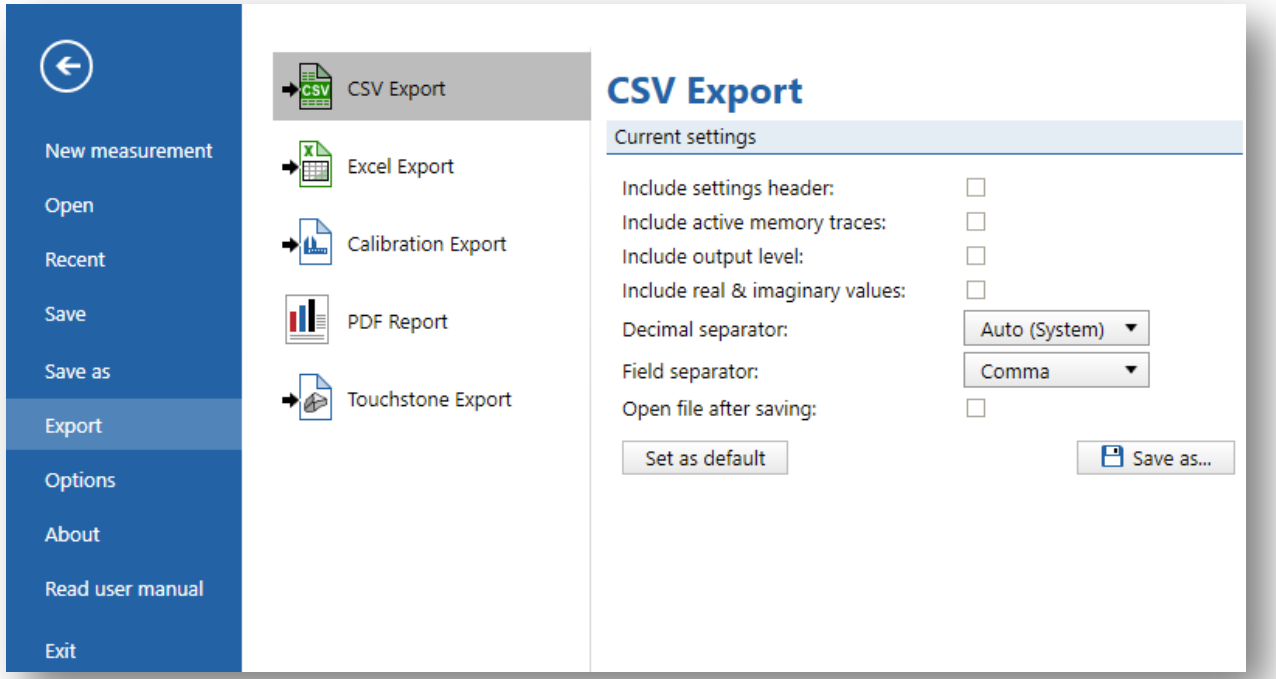




Plant Measurement Using PowerSmart™



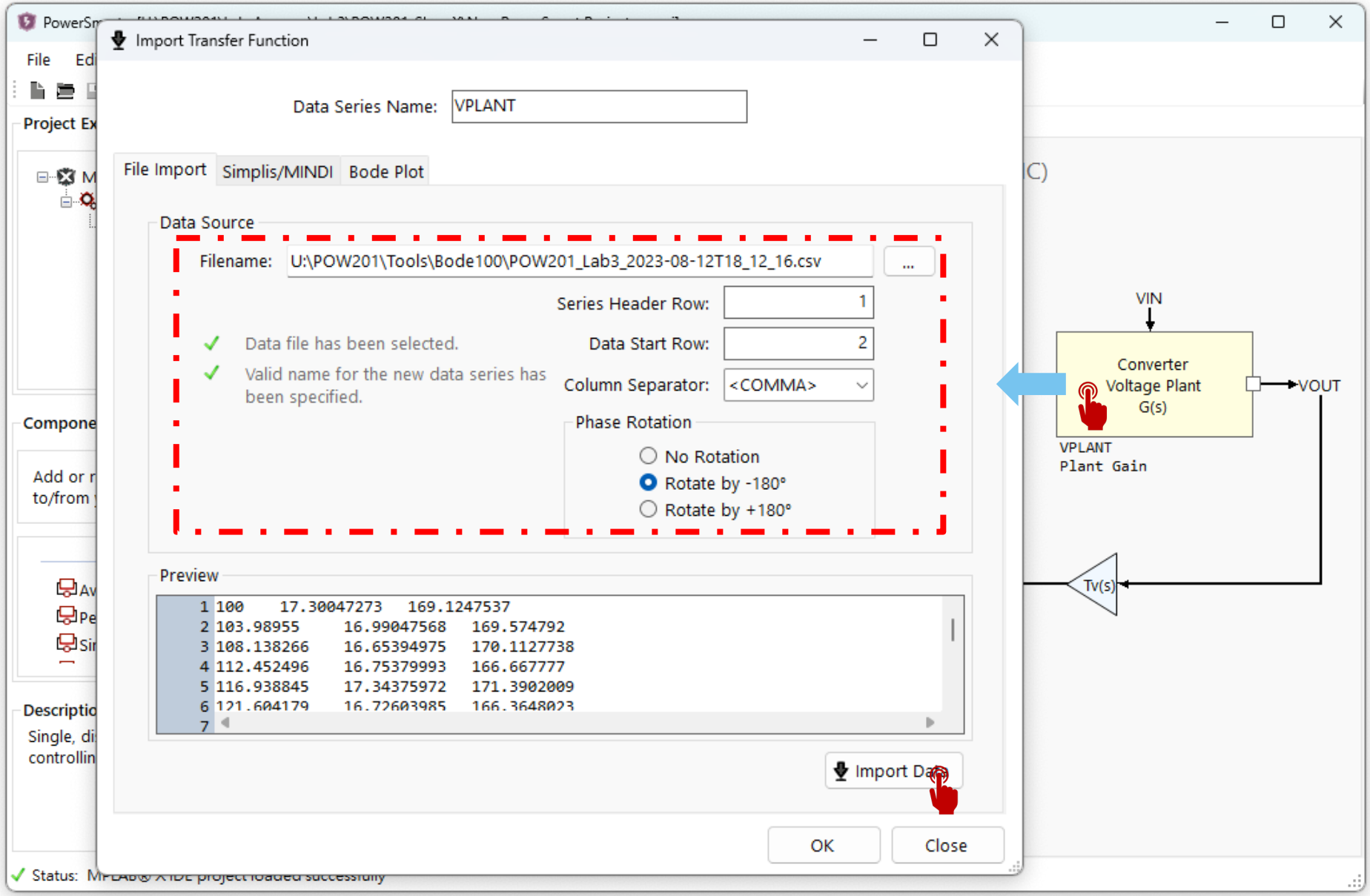
Export Bode Plot to CSV



	A	B	C	D
1	Frequency (H	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	

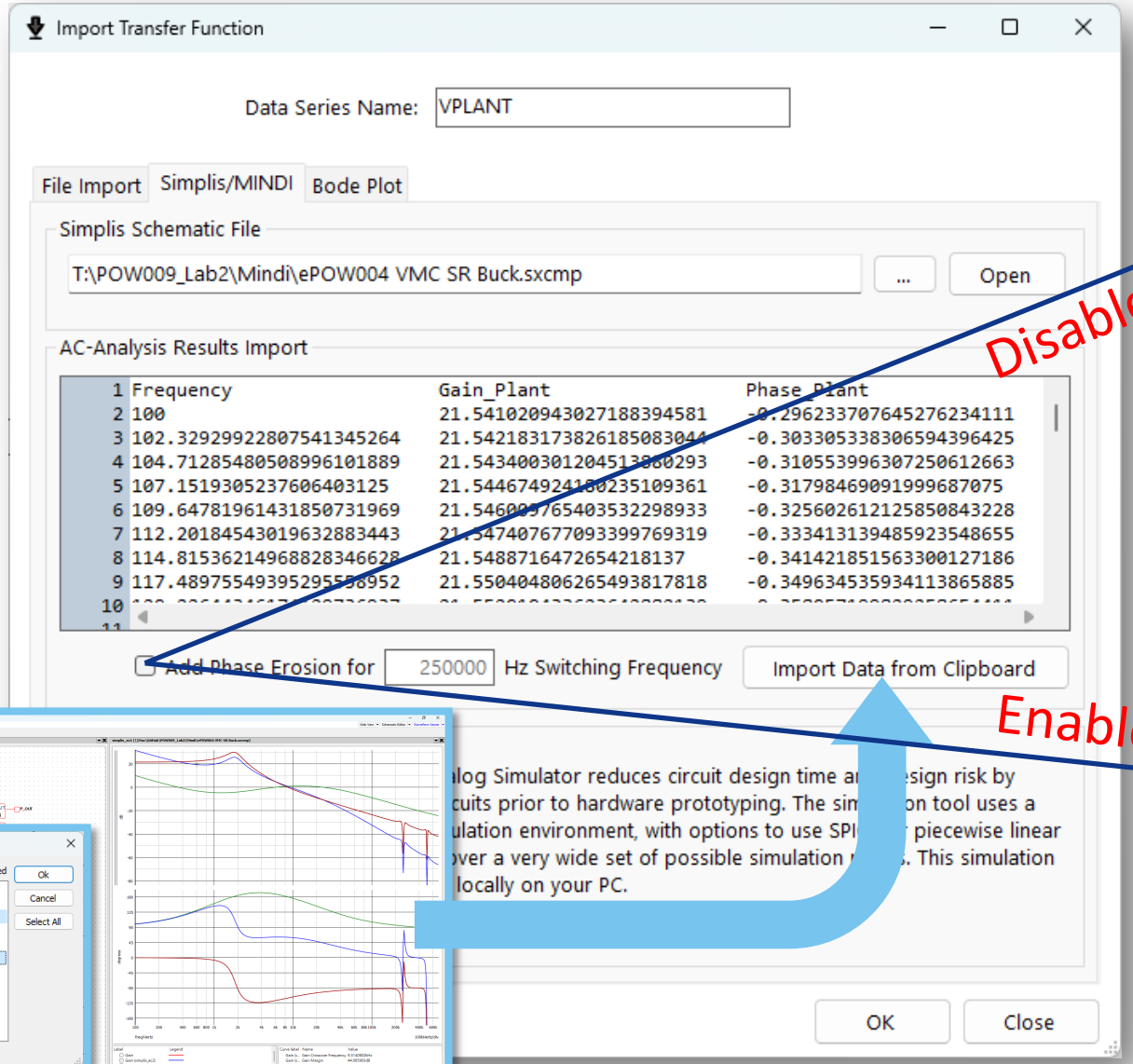
Removed!

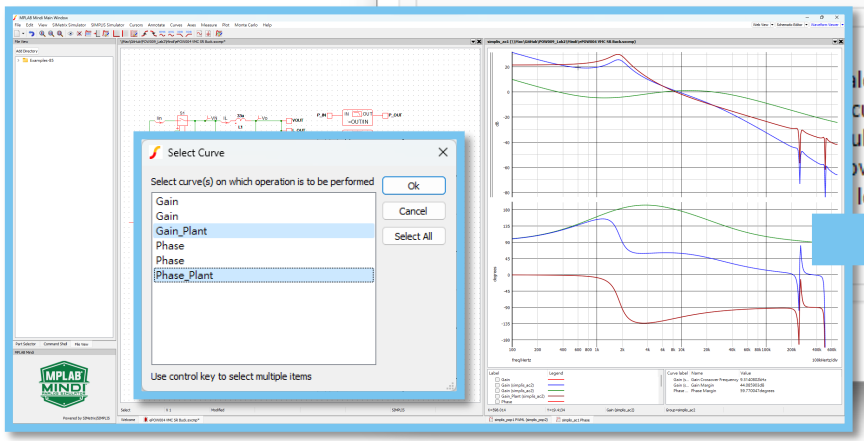
Lab #3: Plant Measurement Using PowerSmart™

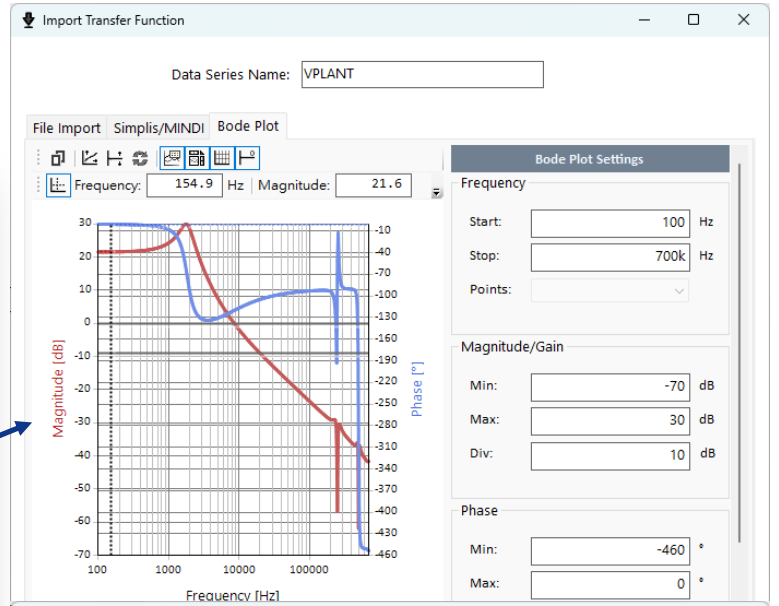


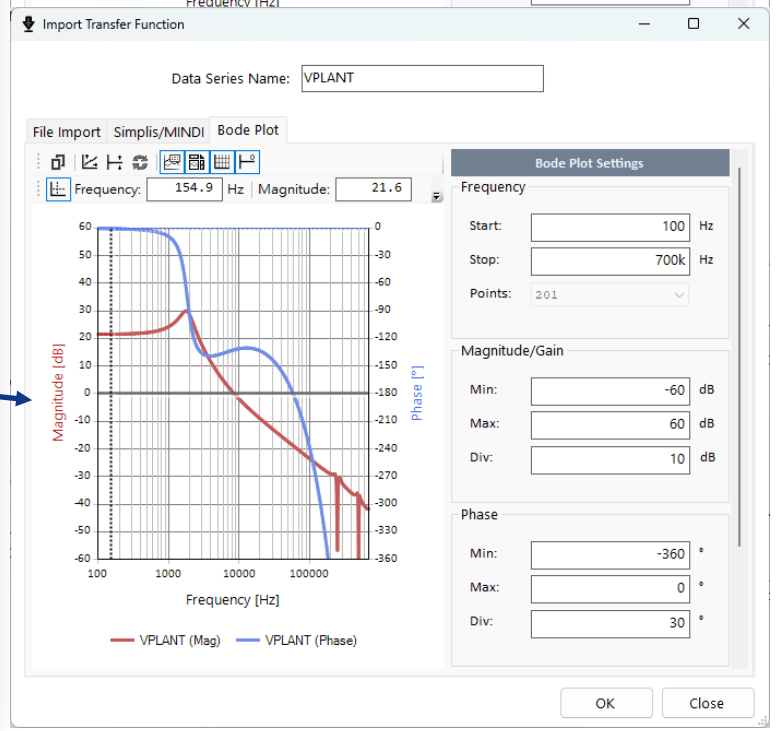
NOW, Actual Plant is Imported!!!

Design Tip









Disable

Enable

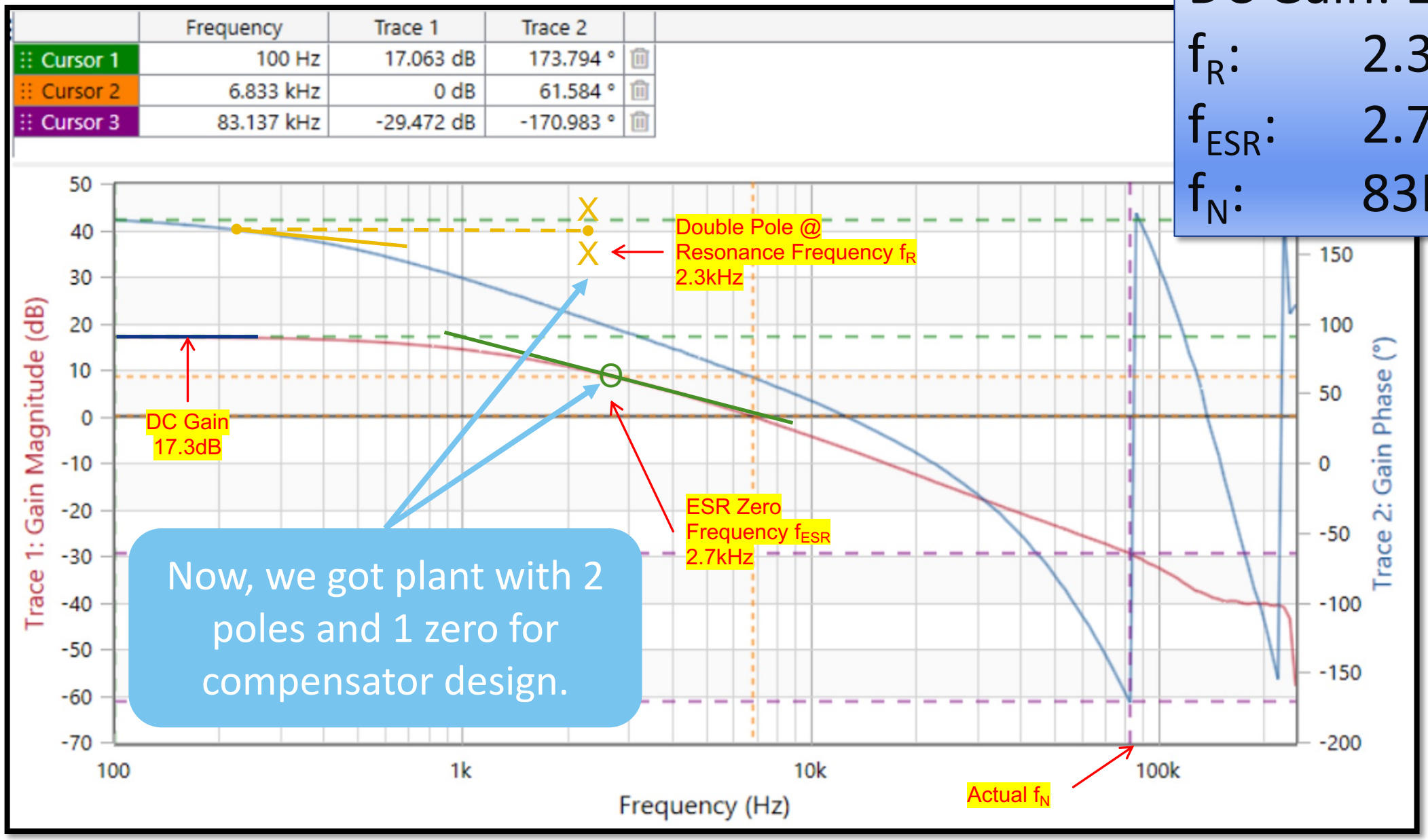
Closed-loop Control

Lab4

The Actual Plant

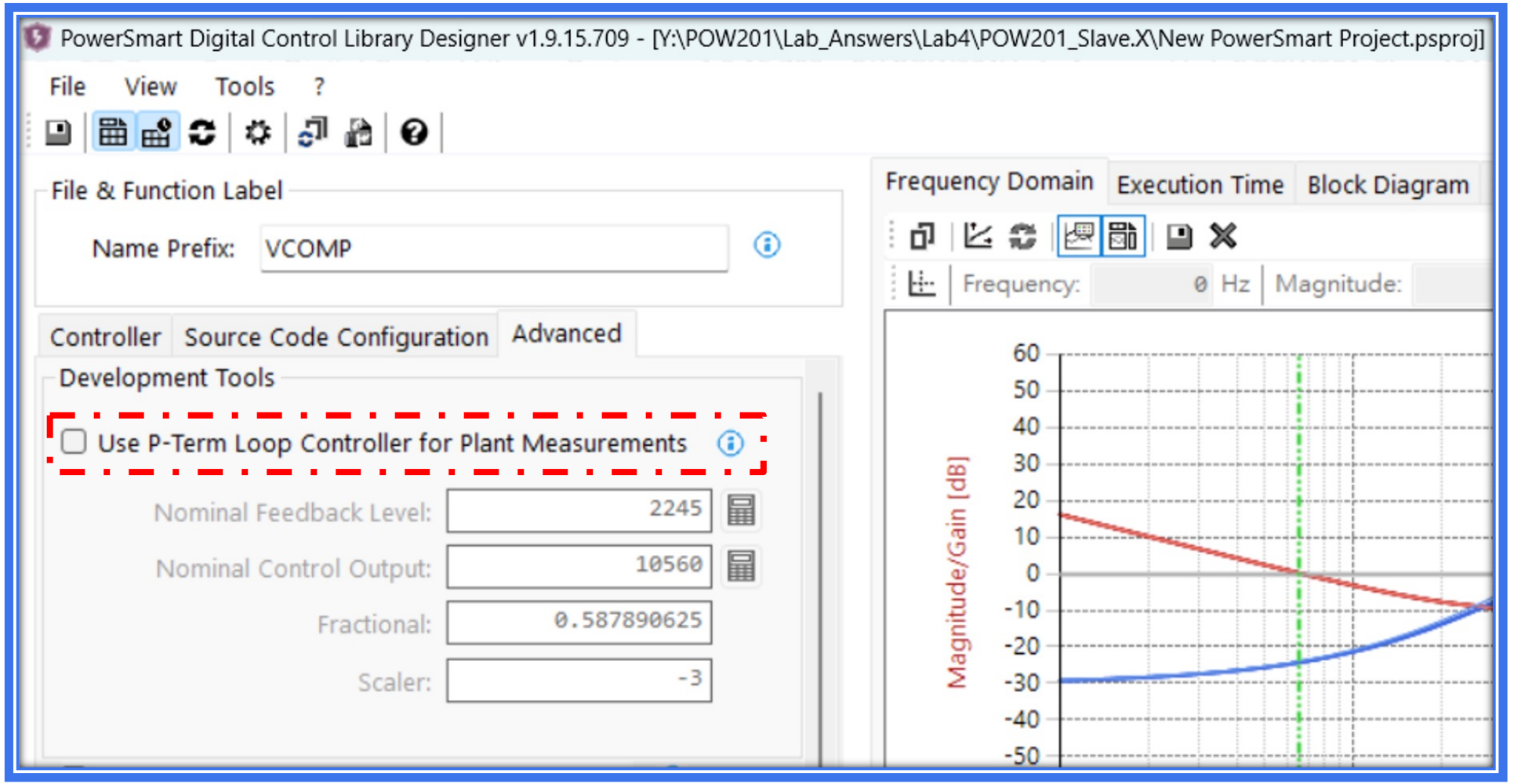
DC Gain: 17dB
f_R: 2.3Hz
f_{ESR}: 2.7KHz
f_N: 83KHz

	Frequency	Trace 1	Trace 2
Cursor 1	100 Hz	17.063 dB	173.794 °
Cursor 2	6.833 kHz	0 dB	61.584 °
Cursor 3	83.137 kHz	-29.472 dB	-170.983 °

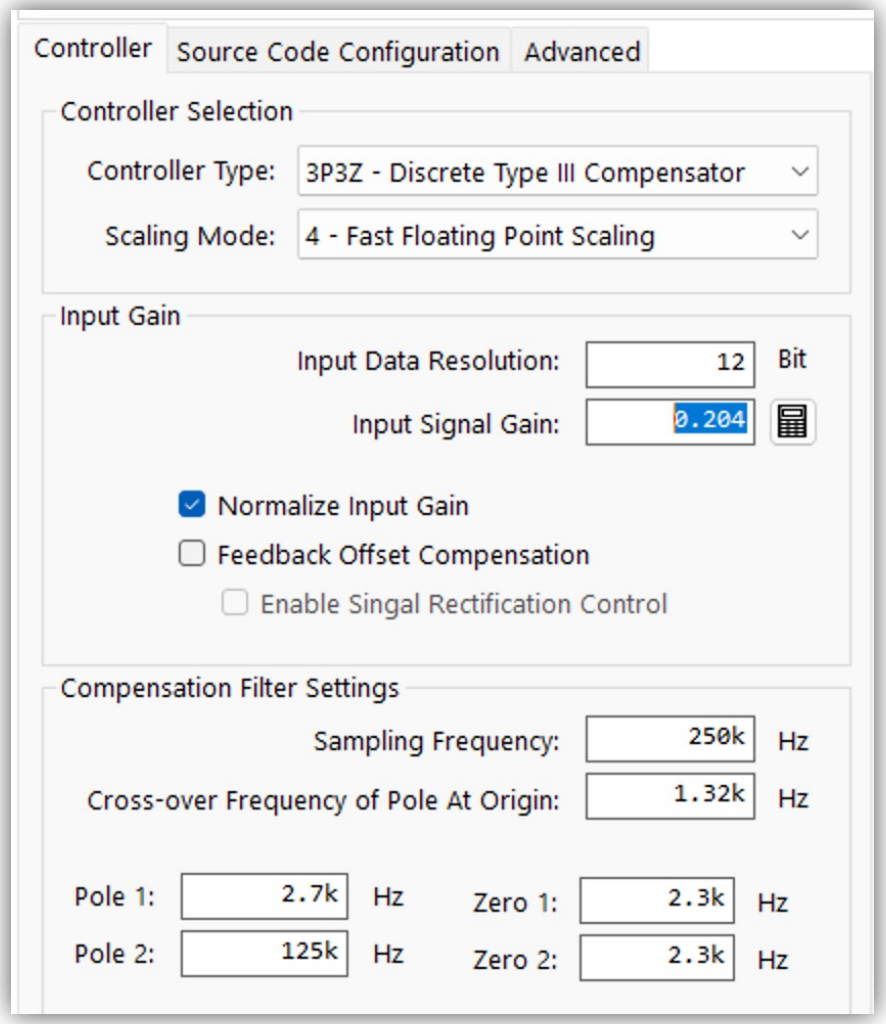


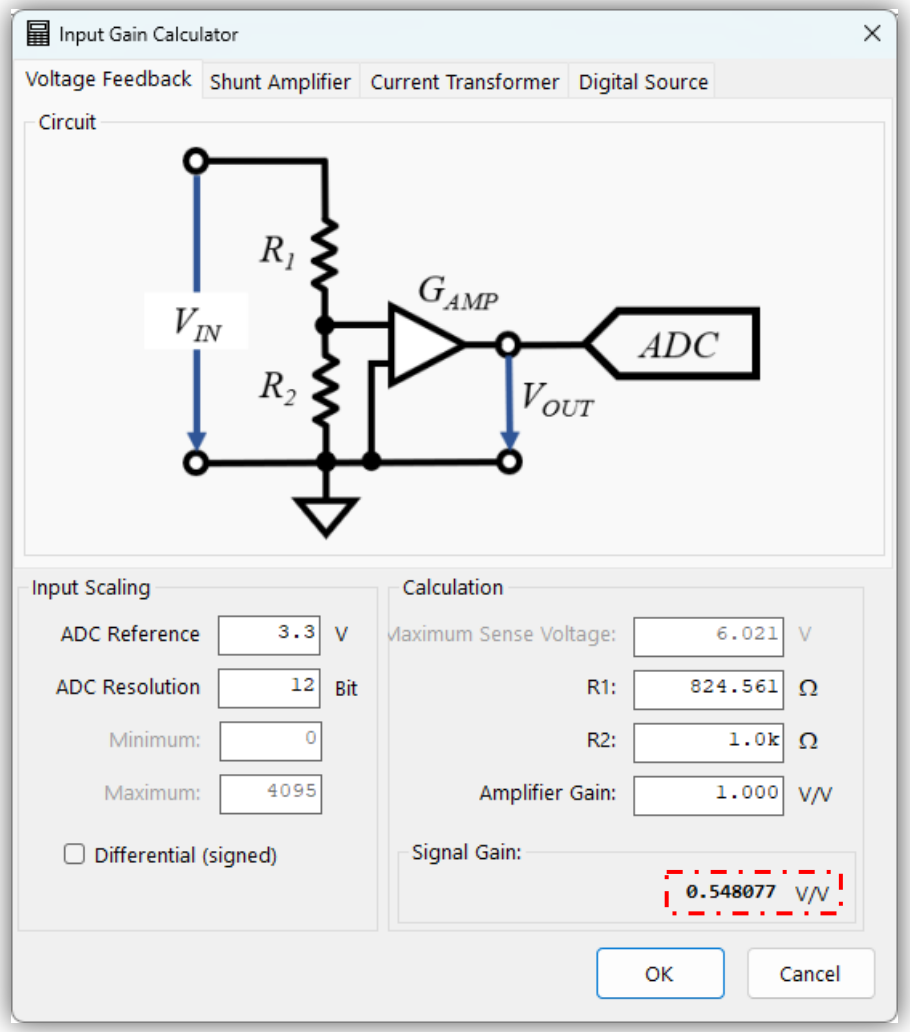
Uncheck Kp (P-Term) Control on DCLD

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



Place Poles-Zeros





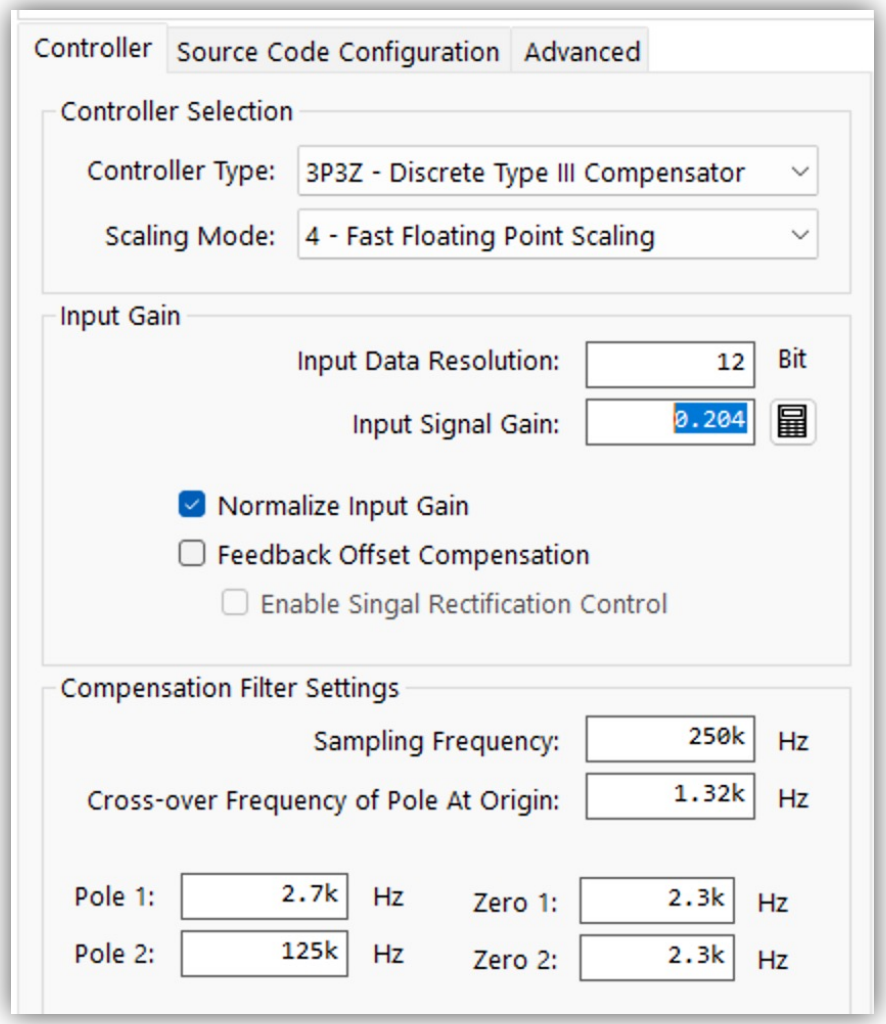
K_{FB}
= 1*R1/(R1+R2)
= 0.4519

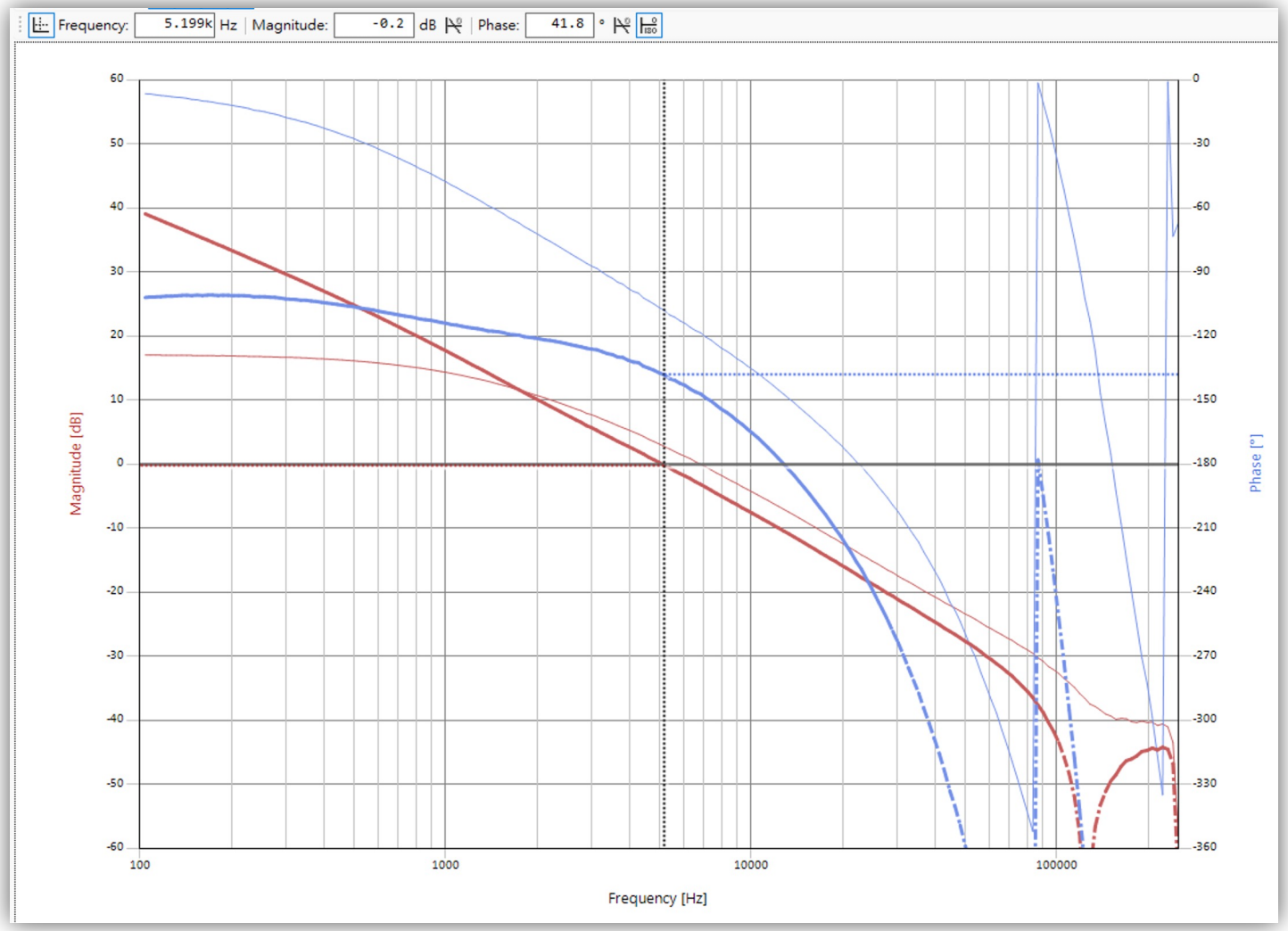
Input Signal Gain
= K_{FB} * K_{FB}
= 0.204

DCLD

Place Poles-Zeros

Observe Open Loop Gain



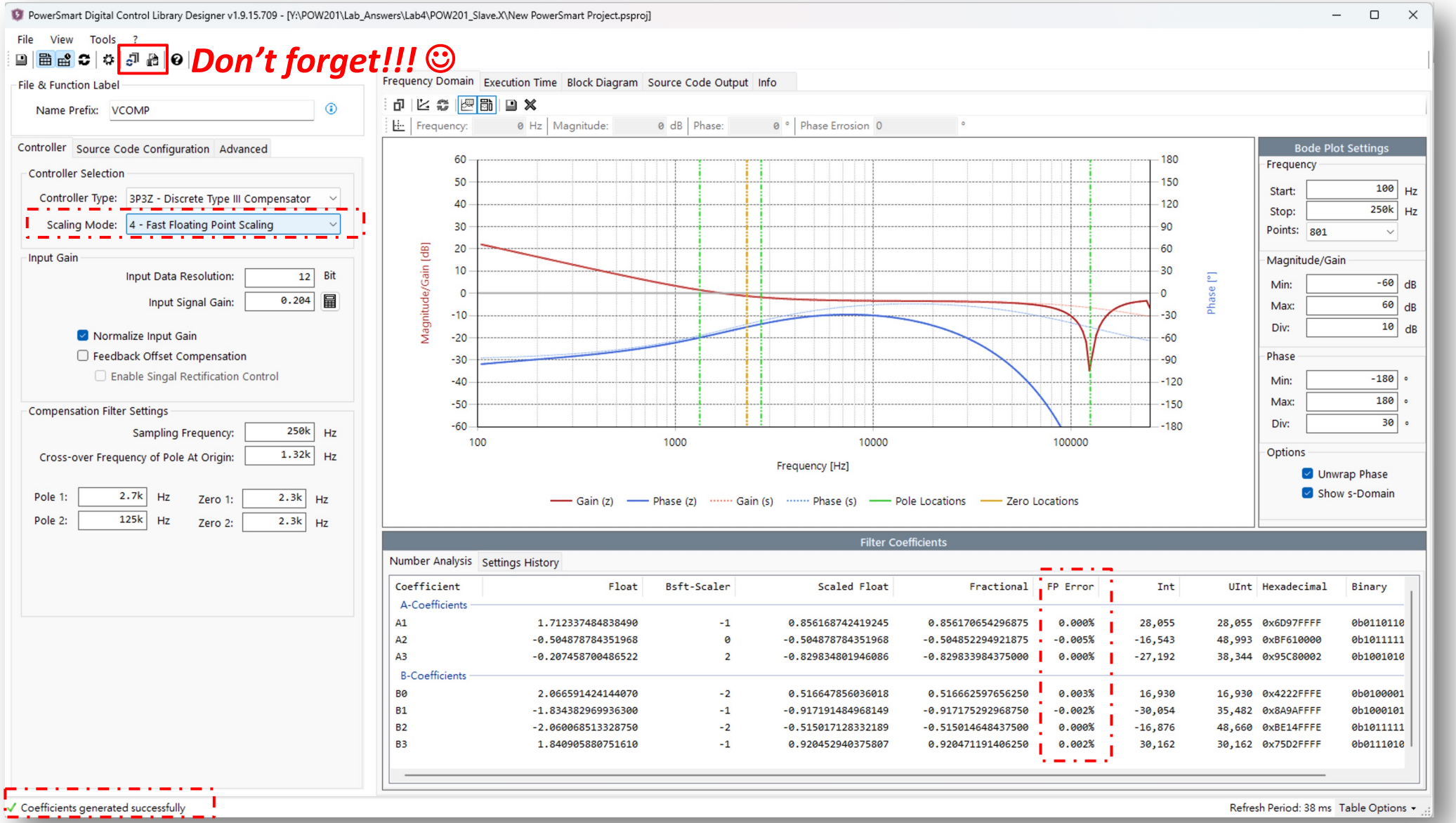


DCLD

PowerSmart™

MICROCHIP

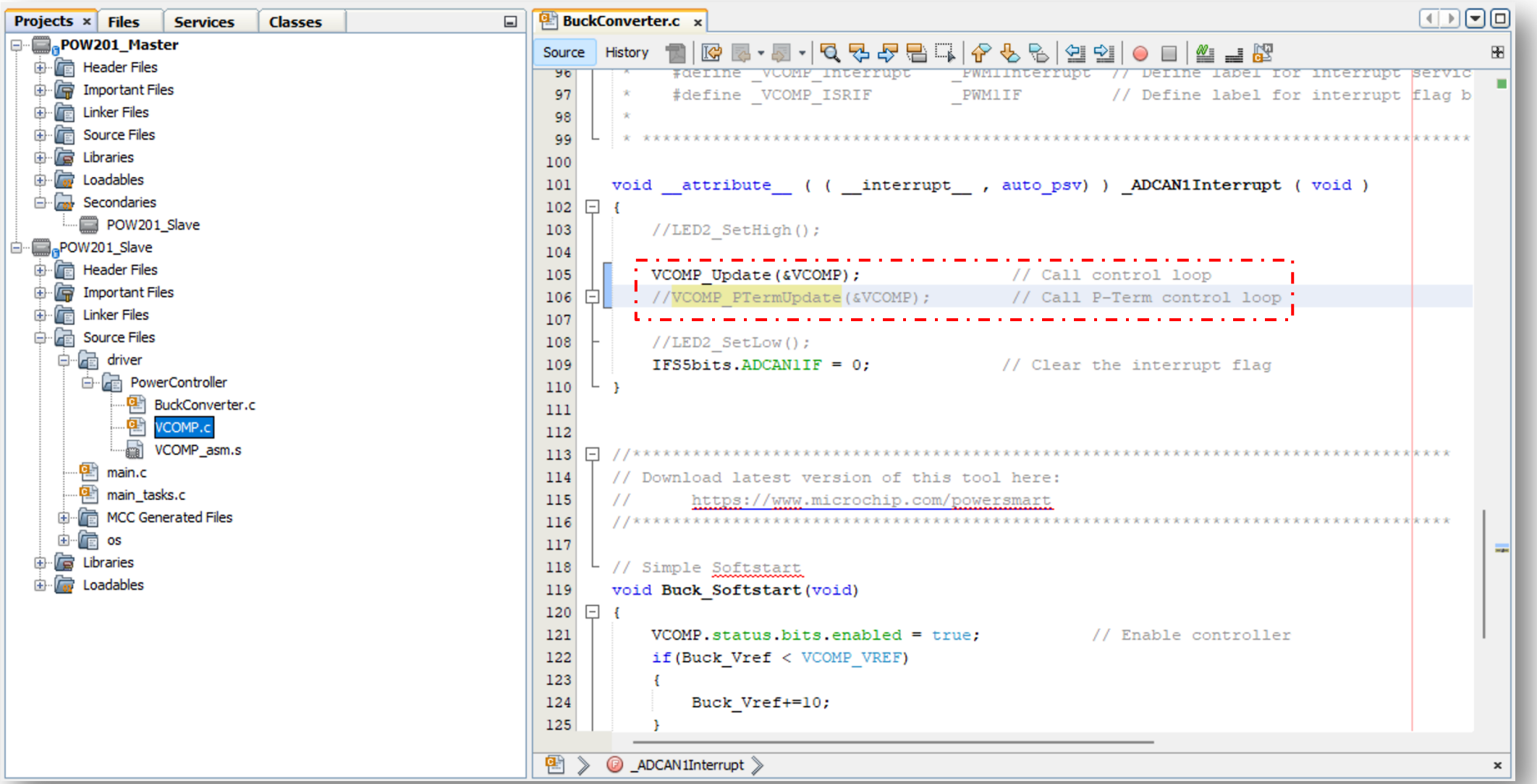
Check Coefficients with Suitable Scaling Mode



Coefficient	Float	Bsft-Scaler	Scaled Float	Fractional	FP Error	Int	UInt	Hexadecimal	Binary
A-Coefficients									
A1	1.712337484838490	-1	0.856168742419245	0.856170654296875	0.000%	28,055	28,055	0x6D97FFFF	0b0110110
A2	-0.504878784351968	0	-0.504878784351968	-0.504852294921875	-0.005%	-16,543	48,993	0xBF610000	0b1011111
A3	-0.207458700486522	2	-0.829834801946086	-0.829833984375000	0.000%	-27,192	38,344	0x95C80002	0b1001010
B-Coefficients									
B0	2.066591424144070	-2	0.516647856036018	0.516662597656250	0.003%	16,930	16,930	0x4222FFFE	0b0100001
B1	-1.834382969936300	-1	-0.917191484968149	-0.917175292968750	-0.002%	-30,054	35,482	0x8A9AFFFF	0b1000101
B2	-2.060068513328750	-2	-0.515017128332189	-0.515014648437500	0.000%	-16,876	48,660	0xBE14FFFE	0b1011111
B3	1.840905880751610	-1	0.920452940375807	0.920471191406250	0.002%	30,162	30,162	0x75D2FFFF	0b0111010

Closed-loop Control

Switch to closed-loop control in ANx ISR

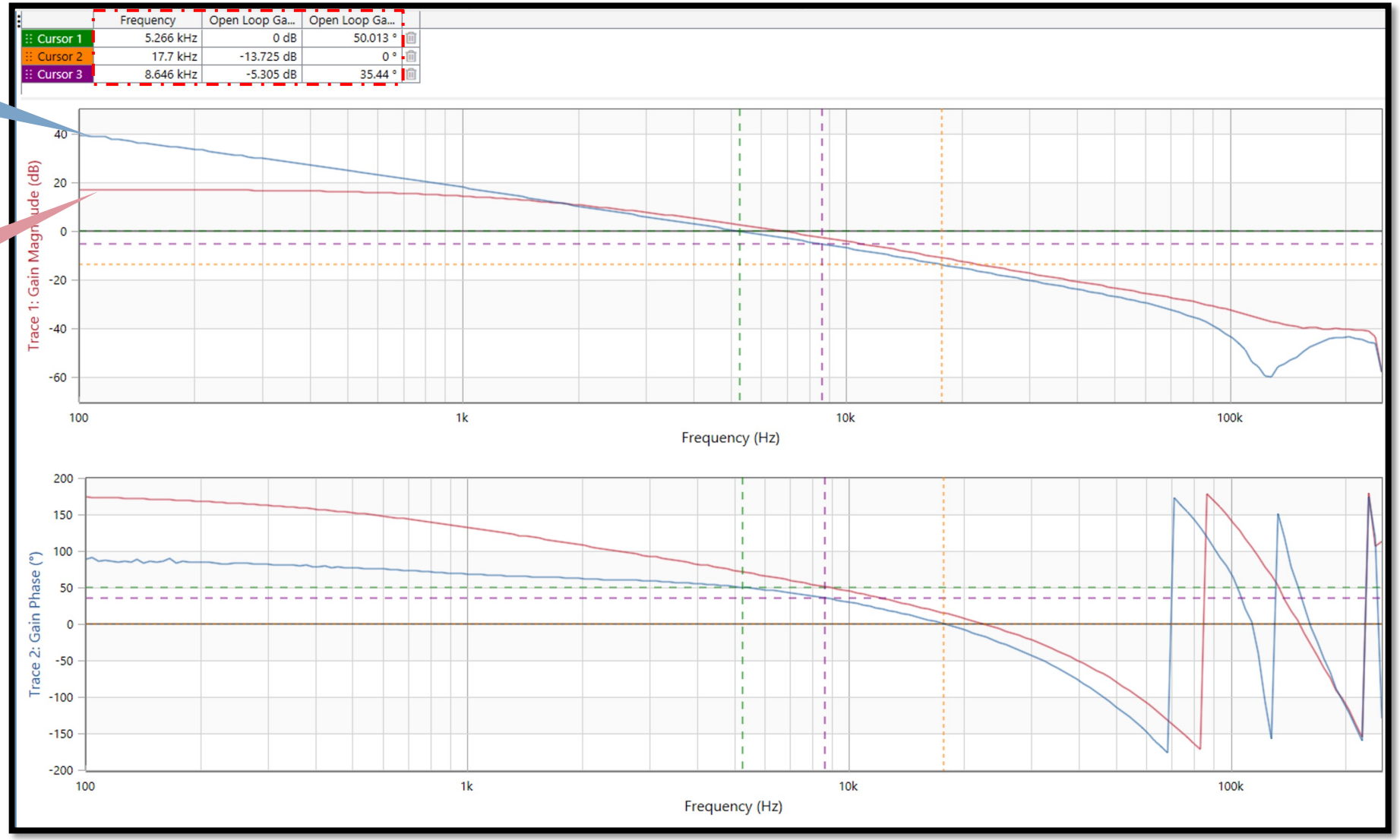


Closed-loop Control

Bode Plot Measurement

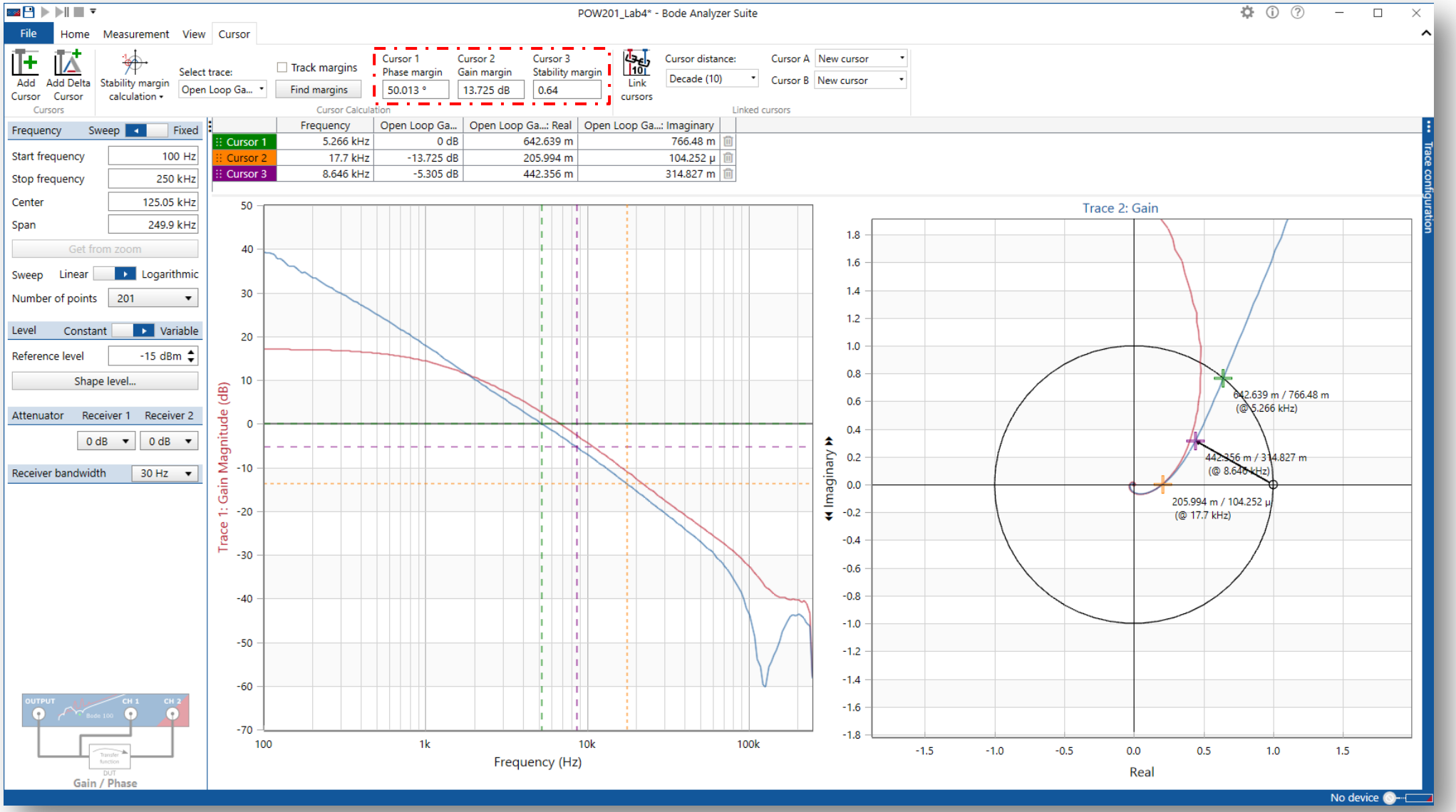
OL TF

Plant



Closed-loop Control

Stability Margin & Nyquist Plot



May The *Power* Be With You