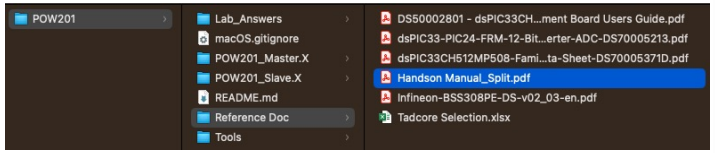


Example Code

https://github.com/EdwardLeeTW/POW201

POW201

Hands-ON Manual

動手實驗參考手冊

MICROCHIP

A Leading Provider of Smart, Connected and Secure Embedded Solutions

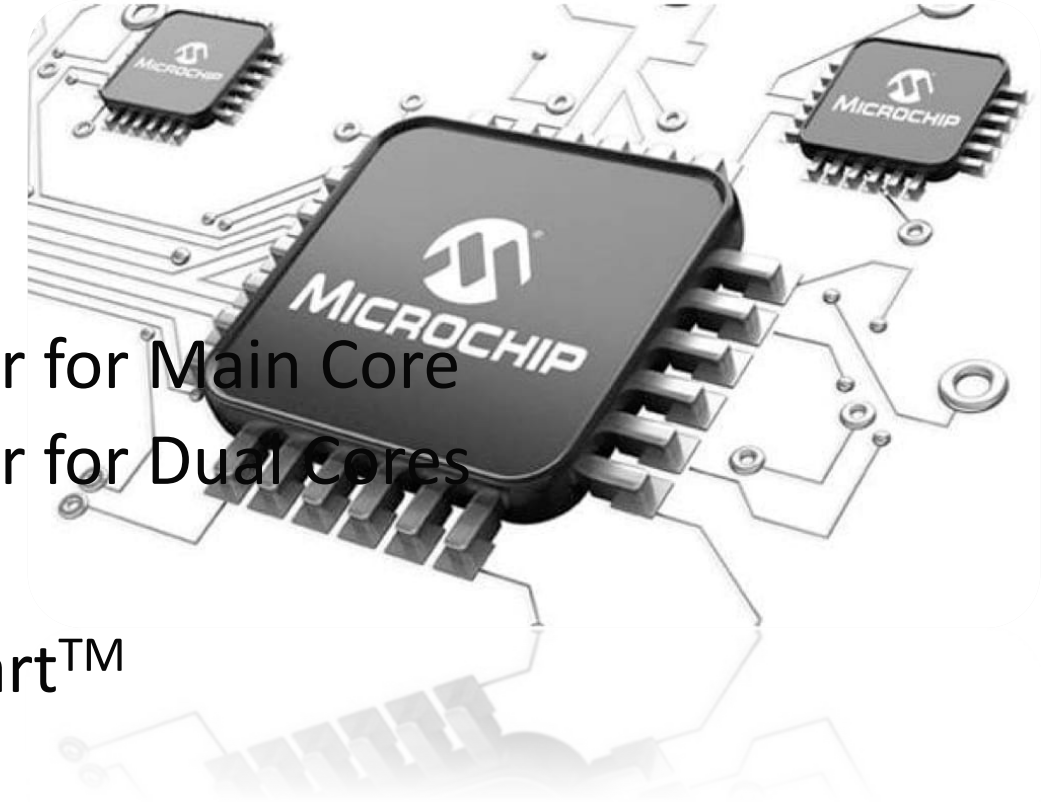
Edward Lee

Oct. 2, 2023

SMART | CONNECTED | SECURE

Agenda

- **Digital SMPS Exercises – Part-I**
  - Lab1: MCC Configuration with OS Scheduler for Main Core
  - Lab2: MCC Configuration with OS Scheduler for Dual Cores
- **Digital SMPS Exercises – Part-II**
  - Lab3: Plant Measurement Using PowerSmart™
  - Lab4: Closed-loop Control



Digital SMPS Exercises – Part-I

Hardware Setup

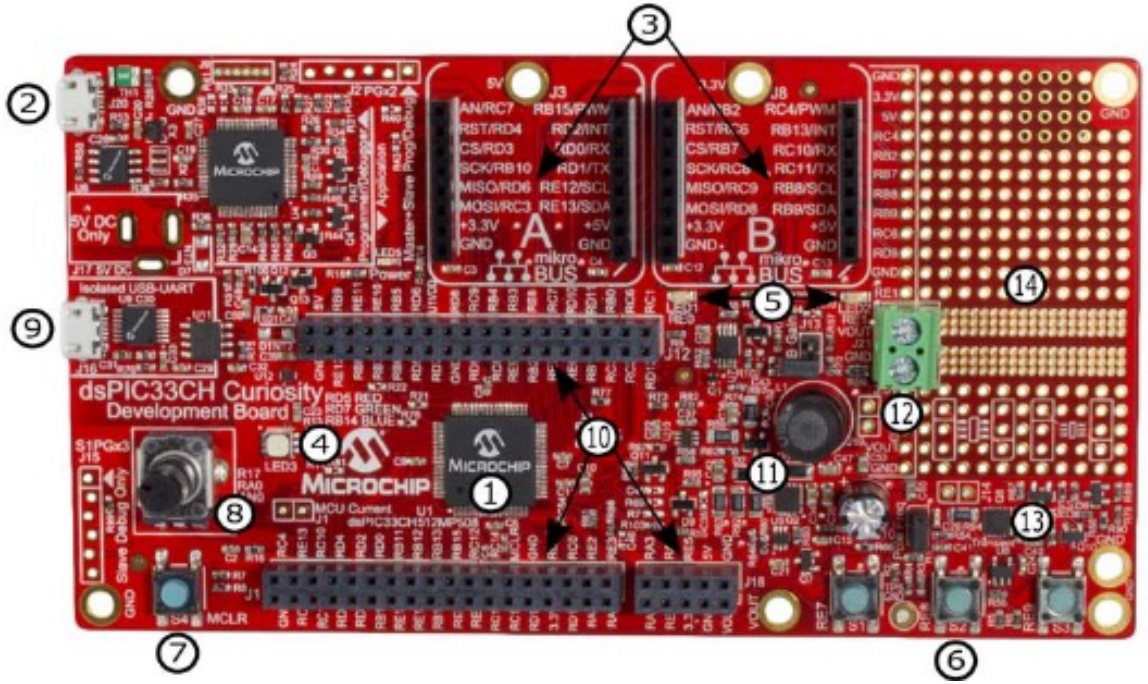
Pre-work

Curiosity DM330028-2 For dsPIC33CH512MP508

https://www.microchip.com/en-us/development-tool/dm330028-2

Hardware Features:

1. dsPIC33CH512MP508 dual core, 16-bit DSP target device.
2. Integrated PICkit™-On-Board (PKOB) programmer/debugger.
3. 2x mikroBUS™ interfaces for hardware expansion, compatible with a wide range of existing click boards™ from MikroElektronika (www.mikroe.com).
4. 1x Red/Green/Blue (RGB) LED.
5. 2x general purpose red indicator LEDs.
6. 3x general purpose push buttons.
7. 1x $\overline{\text{MCLR}}$ Reset push button.
8. 10k potentiometer.
9. Galvanically isolated USB-UART interface, capable of up to 460,800 baud.
10. Female, 100 mil pitch, I/O pin access headers for probing and connecting to all target microcontroller GPIO pins.
11. Configurable Switch Mode Power Supply (SMPS) test circuit that can be operated in Buck, Boost, or Buck-Boost modes, using either Voltage mode or Peak Current mode control.
12. Converter output voltage screw terminal.
13. Configurable load step transient generator.
14. General purpose through-hole and SMT prototyping area.



DM330028-2

dsPIC33CH Curiosity Dev. Board for Hand-ON

- Main chip: dsPIC33CH512MP508
  - Configurable SMPS test circuit that can be operated in Buck, Boost, or Buck-Boost modes, using either Voltage mode or Peak Current mode control
  - PWM: RC14_S1PWM7H

- Buck Mode：
  - Vin = 5V USB Power
  - Vout = 3.3V
  - Iout = Max 0.5A
  - L = 33 µH
  - C = 150 µF, ESR = 390 mΩ
  - f_{SW} = 250 kHz

USB Connected:

- The PICkit™ On-Board (PKOB) USB programmer
- 5V Power Source

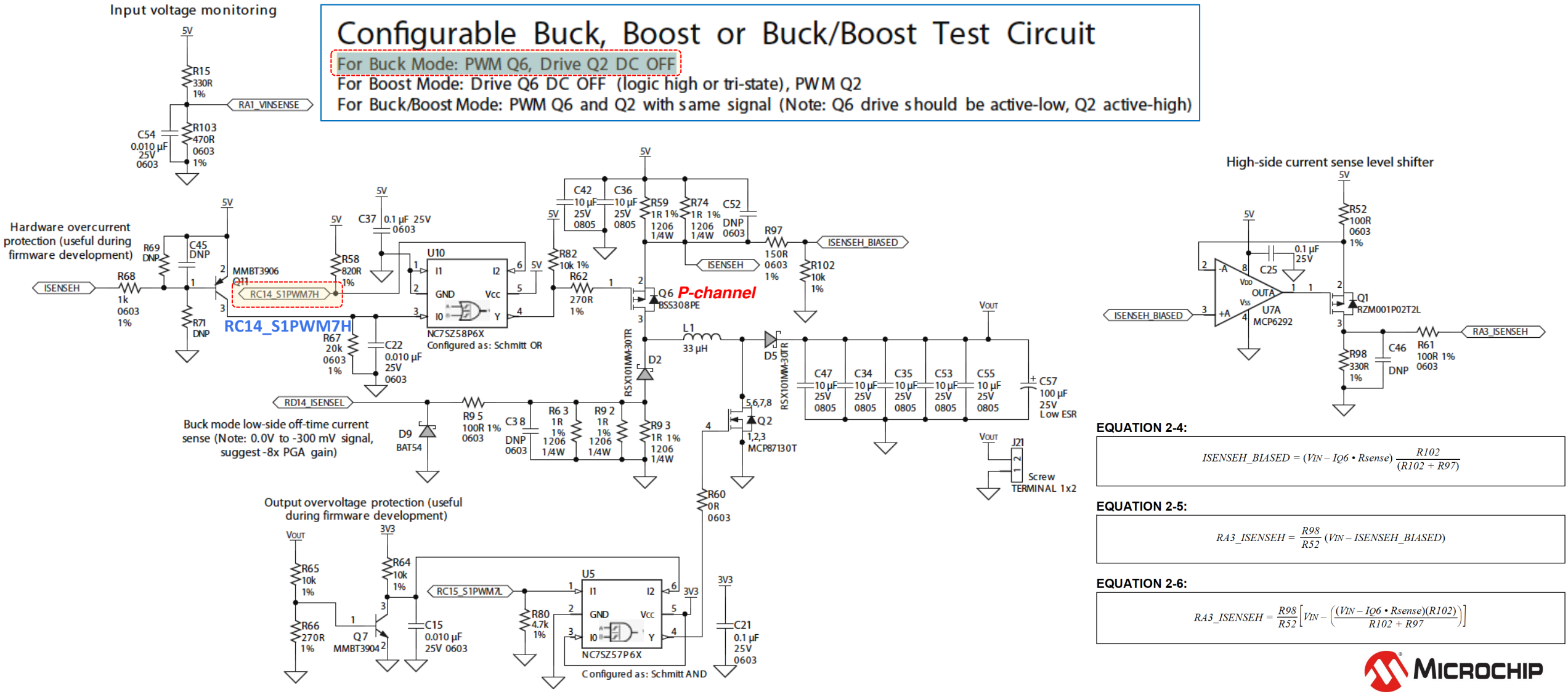
DM330028-2

dsPIC33CH Curiosity Dev. Board for Hand-ON

Main Core

Secondary Core

Power Stage with current sense, H/W OCP and HW OVP



EQUATION 2-4:

$$ISENSEH\_BIASED = (VIN - IQ6 \cdot Rsense)\frac{R102}{(R102 + R97)}$$

EQUATION 2-5:

$$RA3\_ISENSEH = \frac{R98}{R52}(VIN - ISENSEH\_BIASED)$$

EQUATION 2-6:

$$RA3\_ISENSEH = \frac{R98}{R52}\left[VIN - \left(\frac{(VIN - IQ6 \cdot Rsense)(R102)}{R102 + R97}\right)\right]$$

MICROCHIP

dsPIC33CH Curiosity Dev. Board for Hand-ON

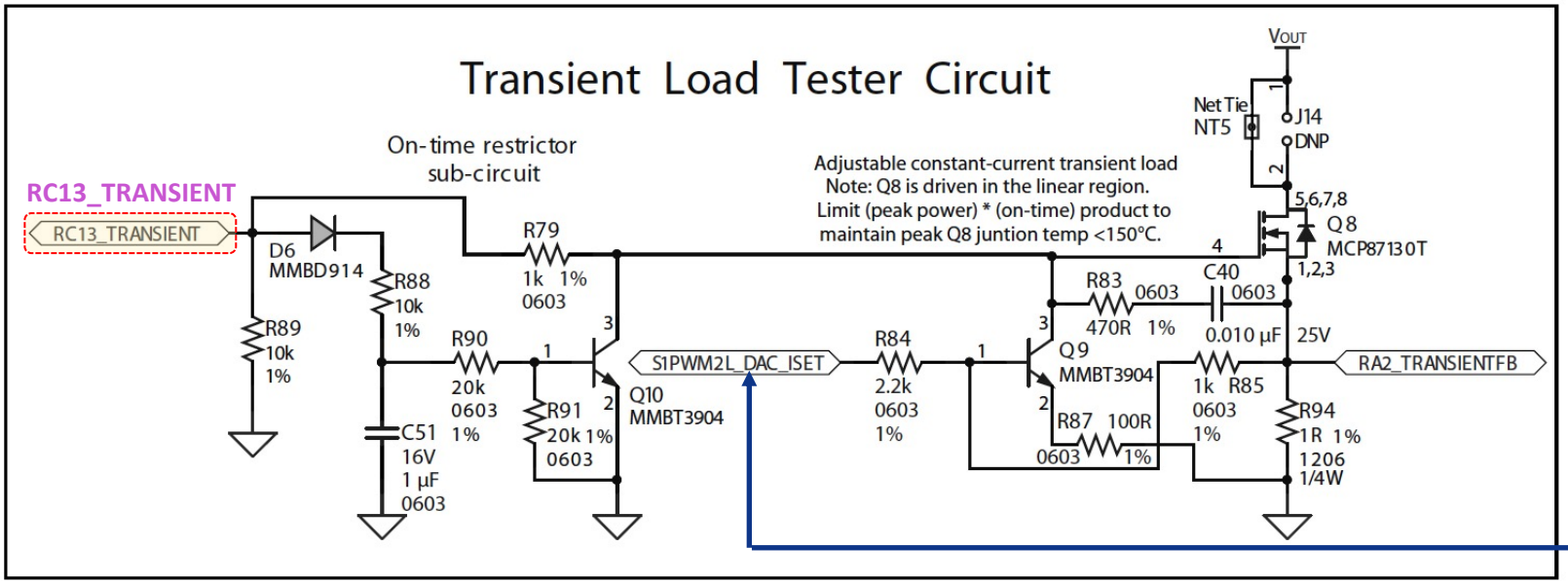
Main Core

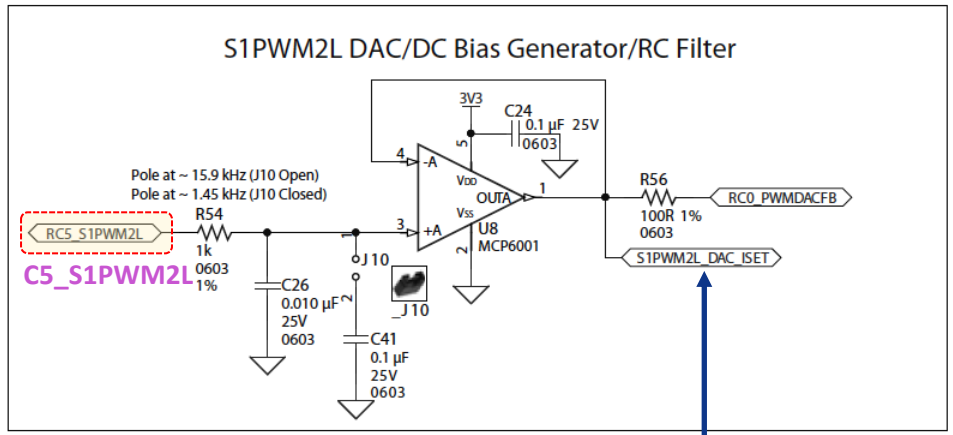
Secondary Core

Configurable load step transient generator

- **Output I/O Pin: RC13_TRANSIENT**
- **Output PWM Pin as DAC Output: RC5_S1PWM2L**
- **Reserved:**
  - Input AN Pin: RA2_TRANSIENTFB
  - Input AN Pin: RC0_PWMDACFB

Adjusting the PWM waveform duty cycle on RC5_S1PWM2L by +1.0% alters the Q8 constant-current sink value by approximately -12 mA. At 50% PWM duty cycle, the approximate current sink level is around 390 mA, but will vary somewhat between boards and at different ambient temperatures, as these will affect the Q9 turn-on voltage. For exact current sink values, it is necessary to use closed-loop control by measuring the RA2_TRANSIENTFB current sense voltage with the ADC at run time. Then, using the resulting value to fine-tune adjust the PWM duty cycle on RC5_S1PWM2L.





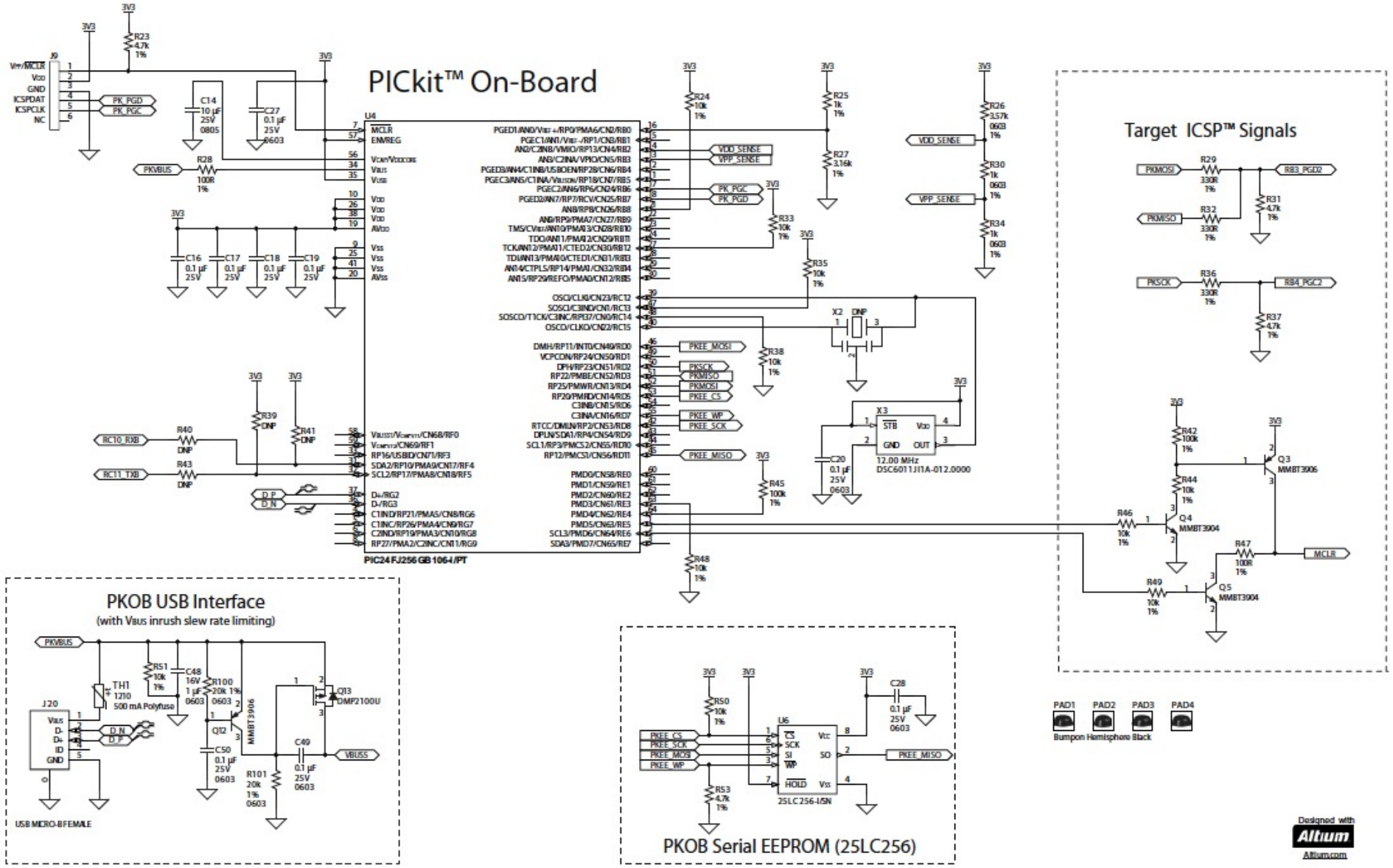
MICROCHIP

dsPIC33CH Curiosity Dev. Board for Hand-ON

Main Core

Secondary Core

PKOB

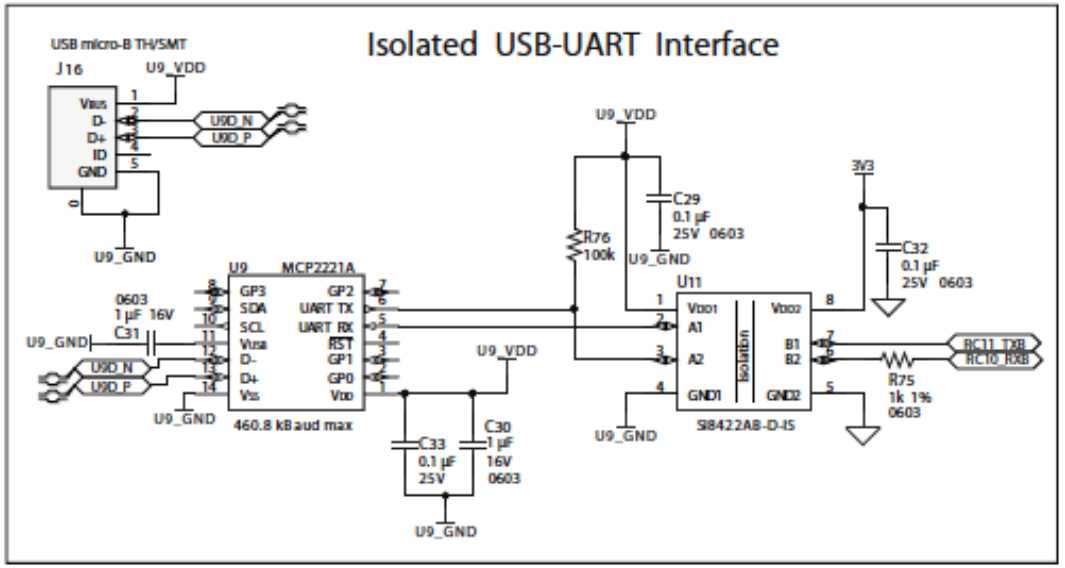


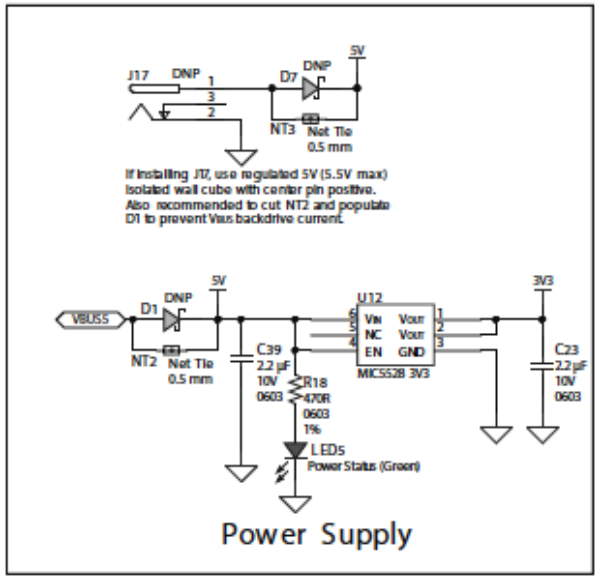
dsPIC33CH Curiosity Dev. Board for Hand-ON

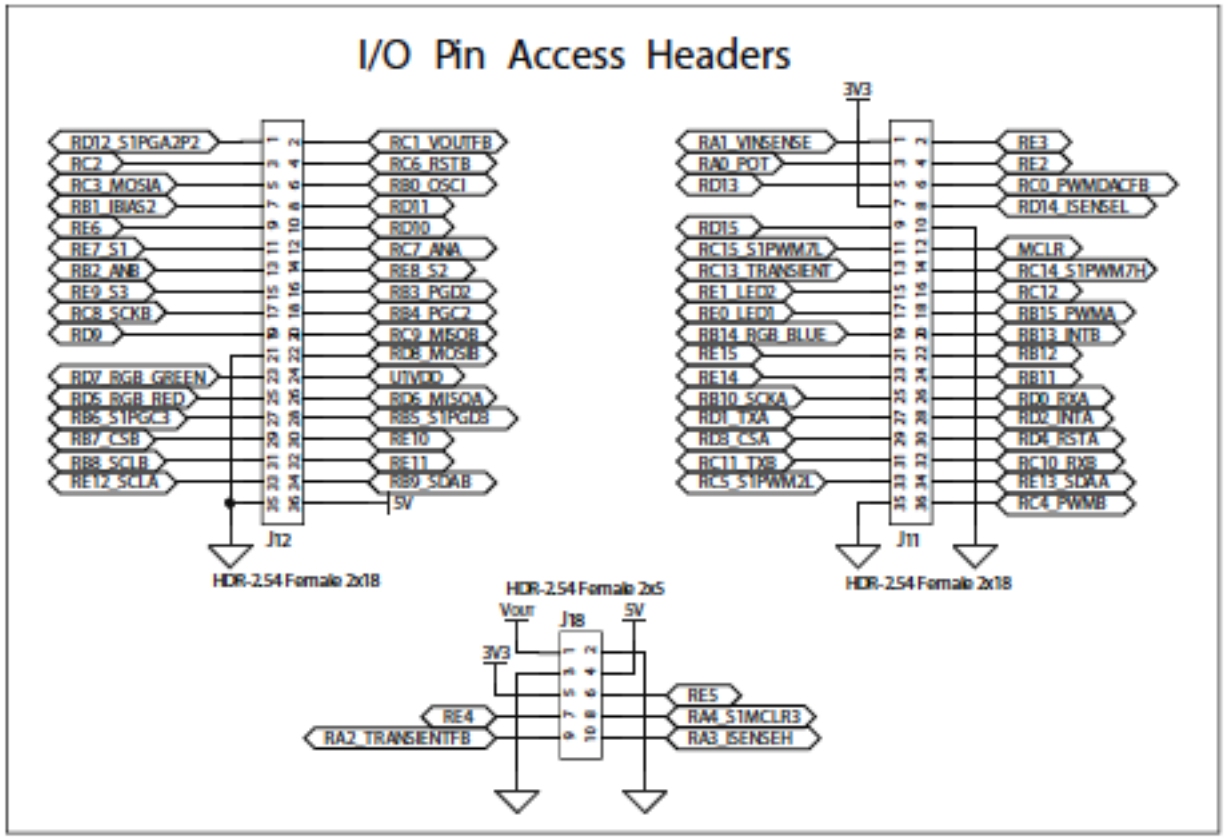
Main Core

Secondary Core

MCU & Interface



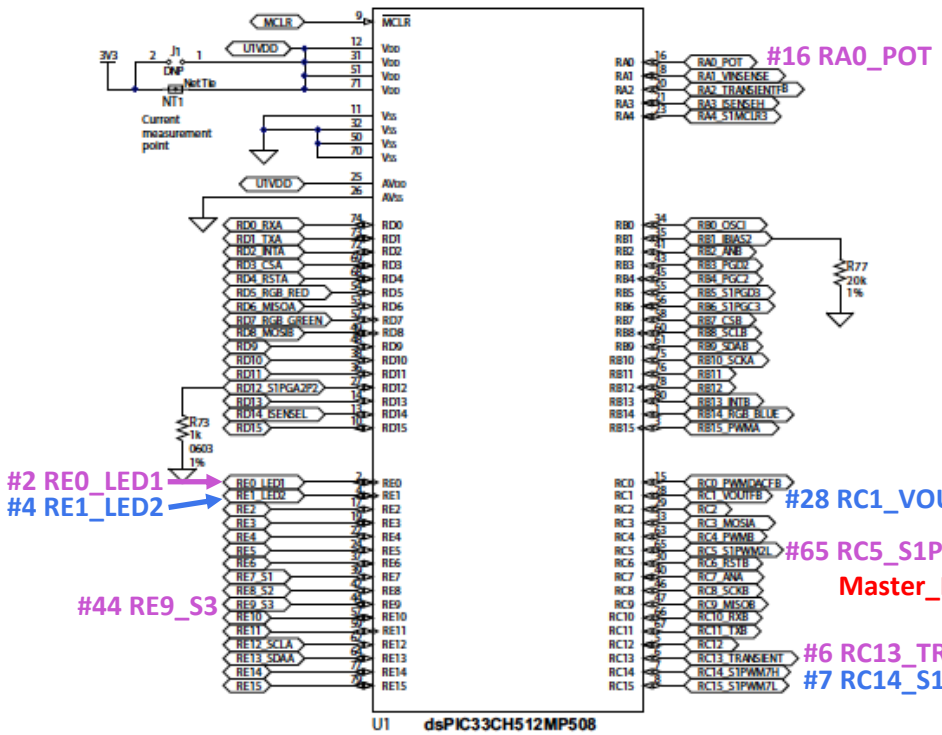


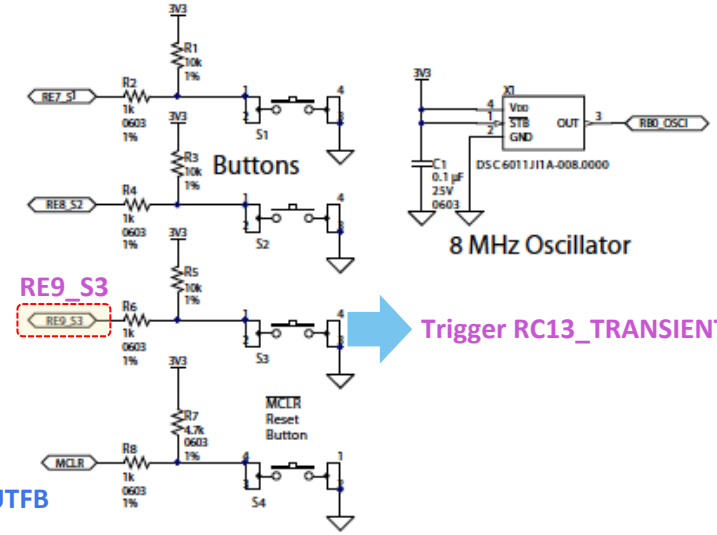


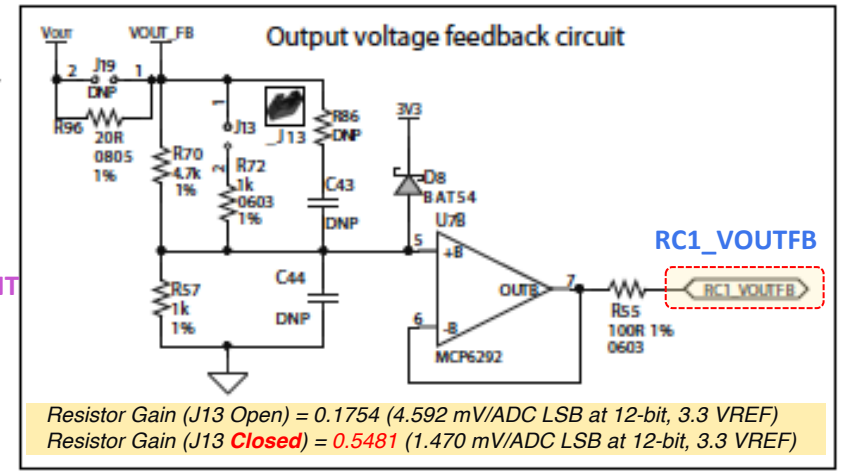
dsPICC33CH Curiosity Dev. Board for Hand-ON

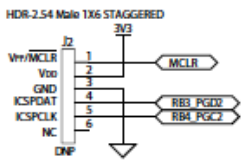
Main Core

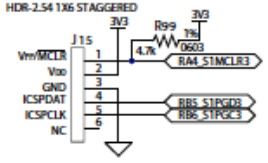
Secondary Core

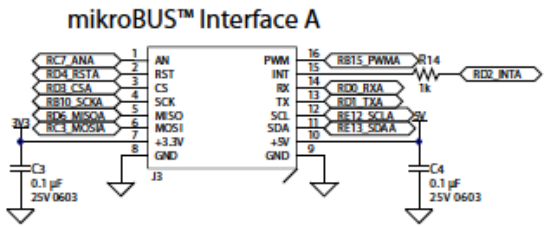


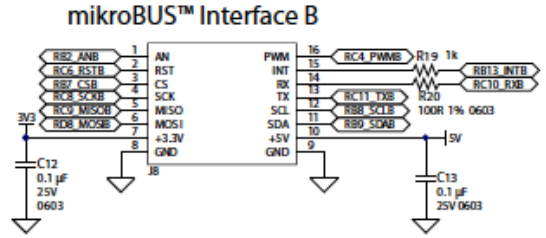


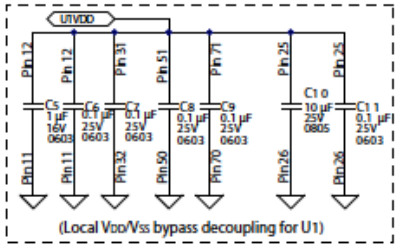


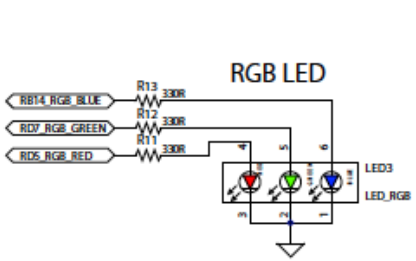


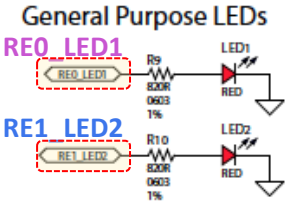


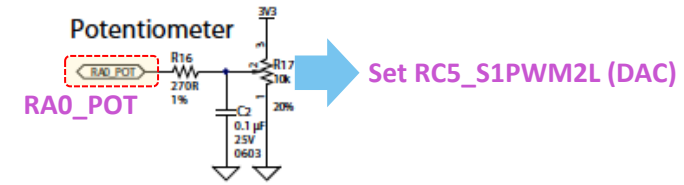


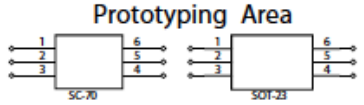


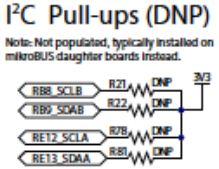












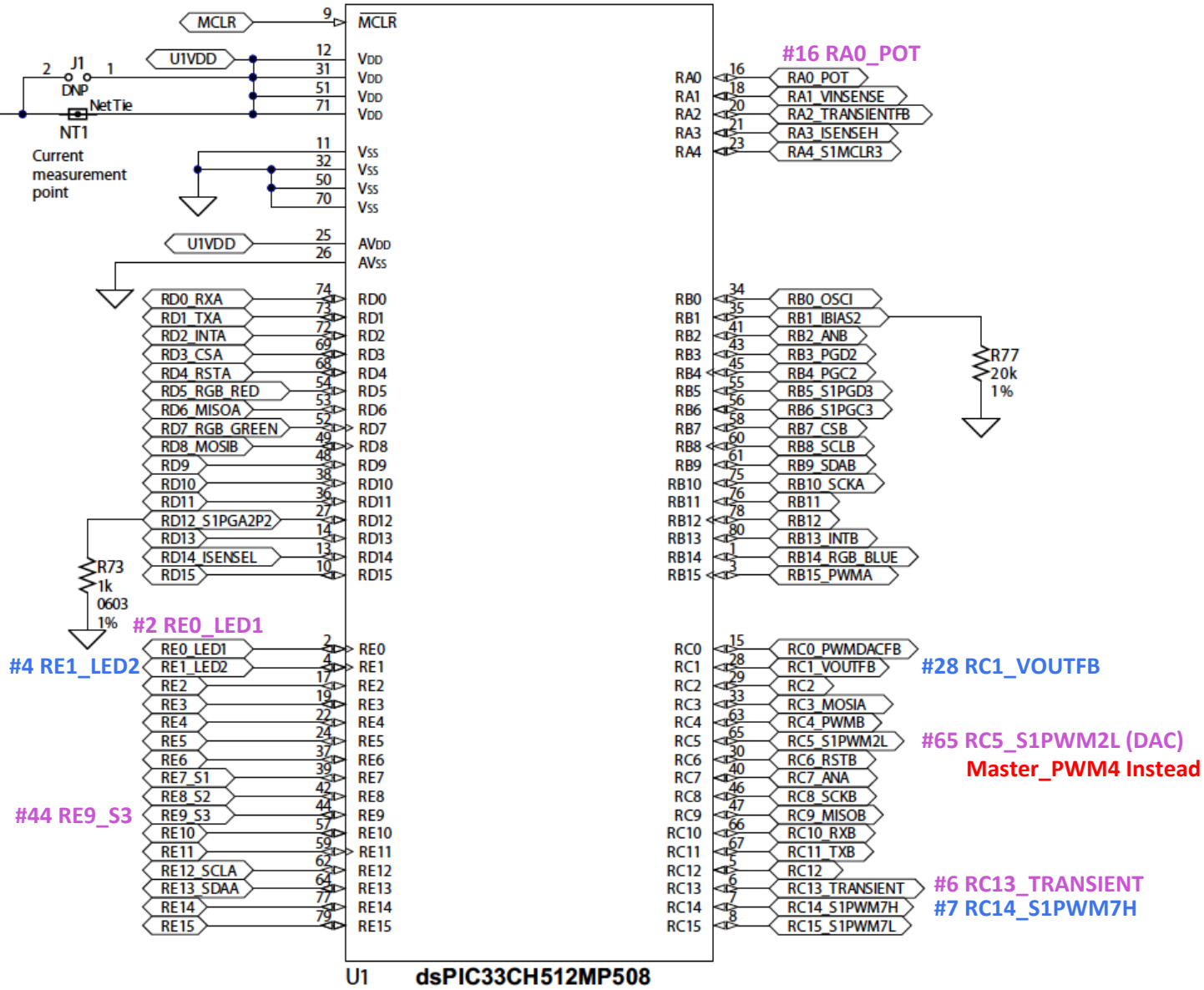
Designed with Altium

MICROCHIP

dsPIC33CH Curiosity Dev. Board for Hand-ON

Main Core

Secondary Core

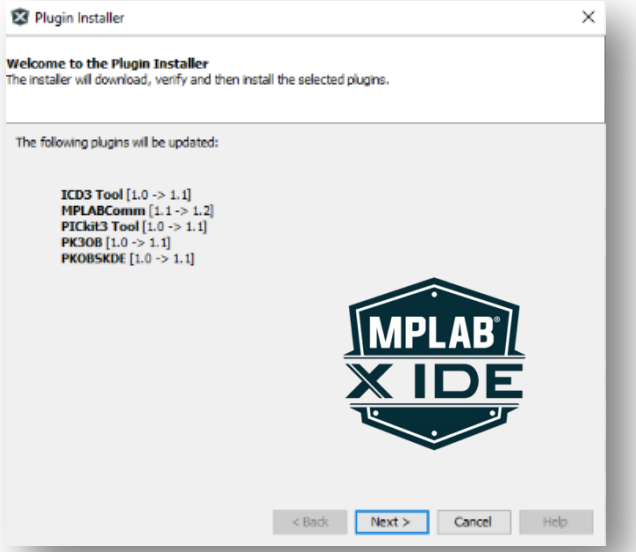


Software Tools

Pre-work

Software Tools

- **Software requirements (Versions for now)**
  - MPLAB® X IDE V6.10 https://www.microchip.com/ide
  - MPLAB XC Compiler (**XC16**) V2.1 https://www.microchip.com/xc
    - DFP v1.12.352
  - MPLAB Code Configurator (**MCC**) V5.3.7
    - Core v5.5.7
    - Device Library: dsPIC33 v1.171.2

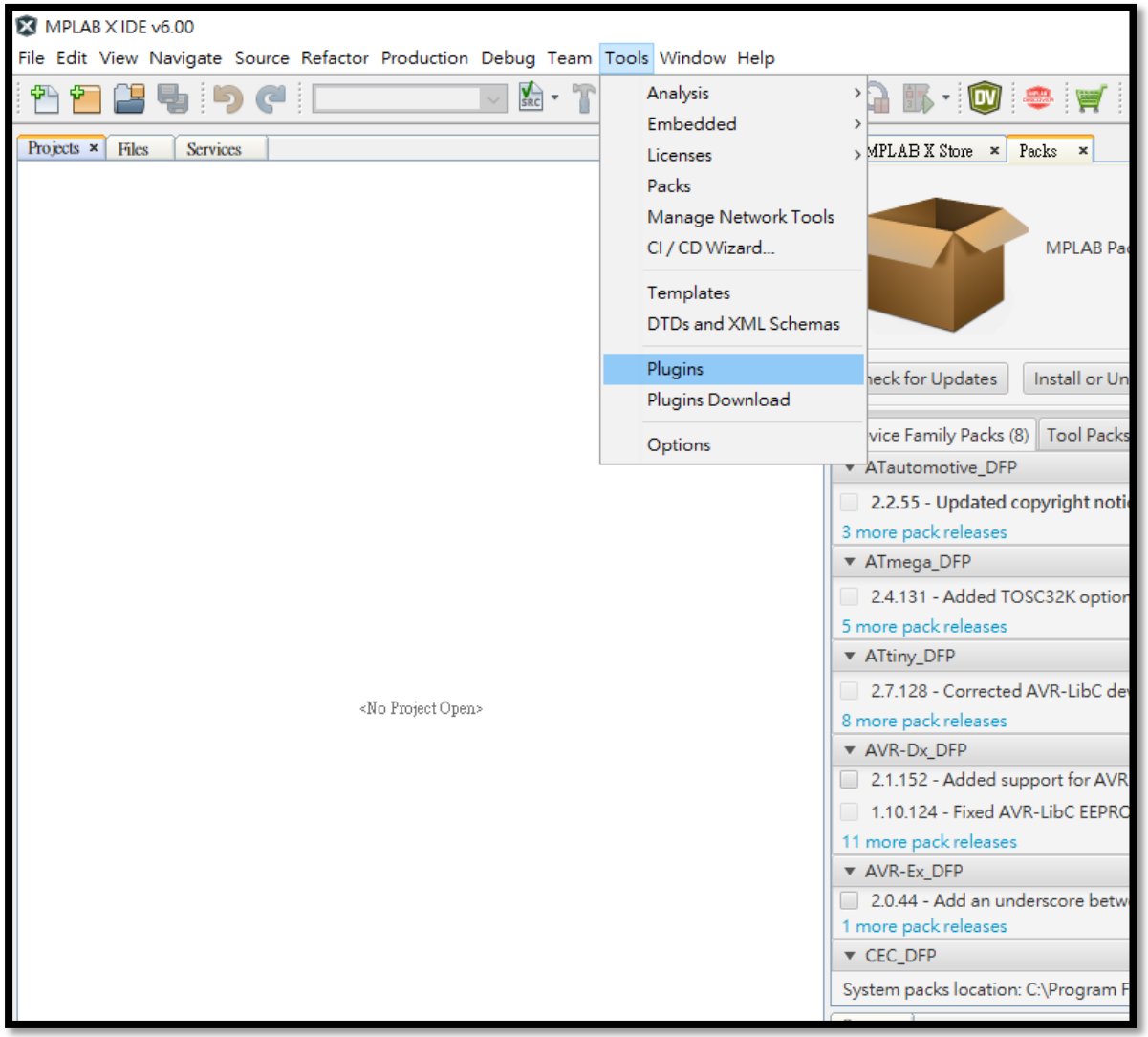


Plug-in updates for MPLAX V6.10 to fix the USB communication issues which were preventing Generation 3 tools support.

- **Optional Software**
  - MPLAB Mindi™ Analog Simulator https://www.microchip.com/mindi
  - MPLAB PowerSmart™-DCLD https://www.microchip.com/en-us/solutions/power-management-and-conversion/intelligent-power/mplab-powersmart-development-suite
    - v1.0.0.633

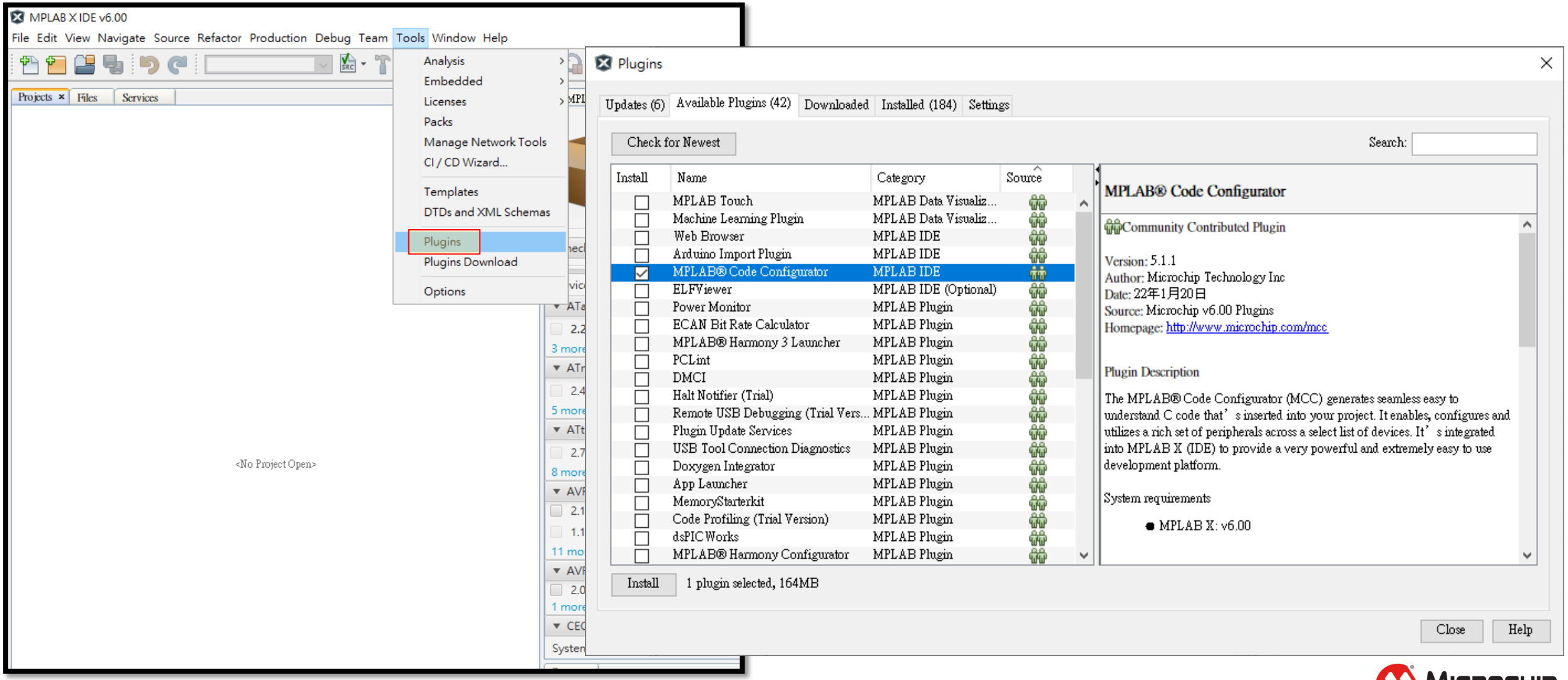
Installing MCC

MPLAB® X IDE / Tools / Plugins



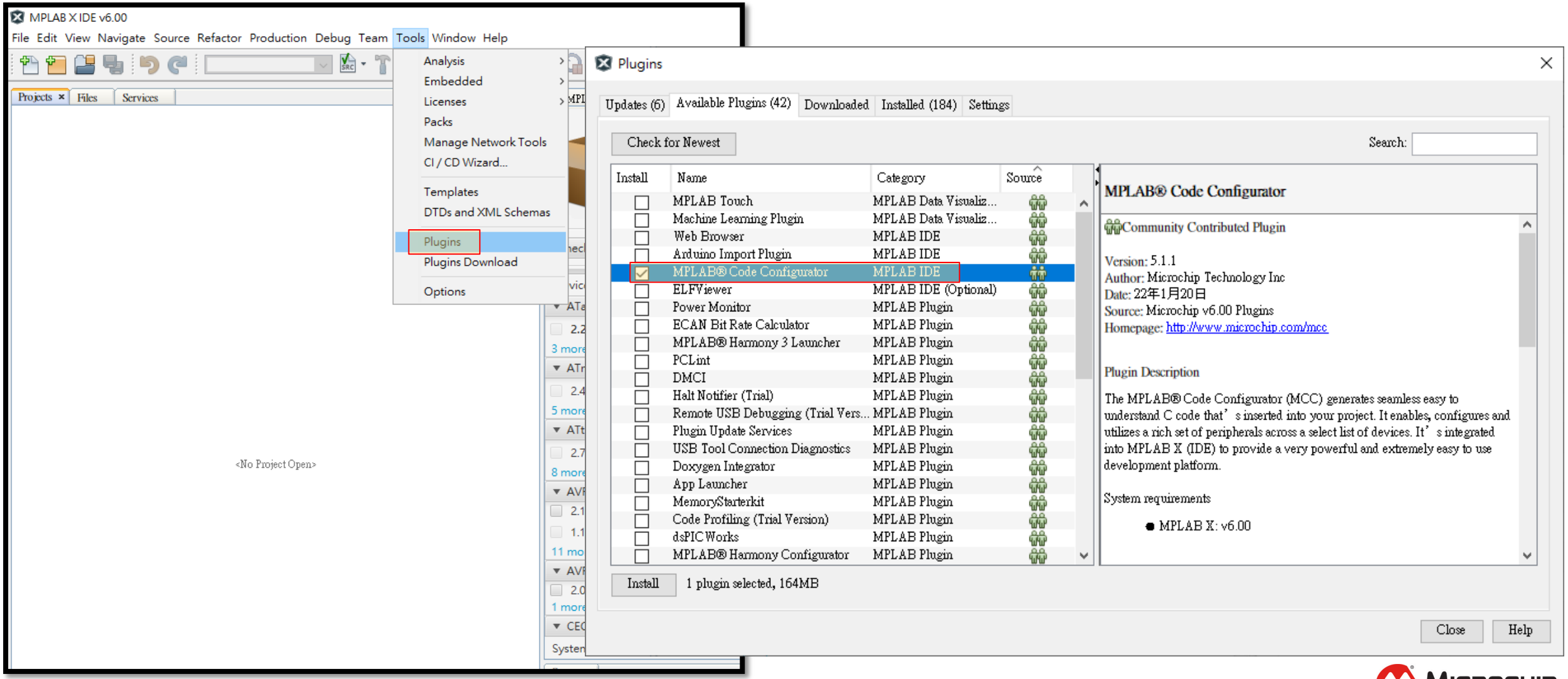
Installing MCC

MPLAB® X IDE / Tools / Plugins



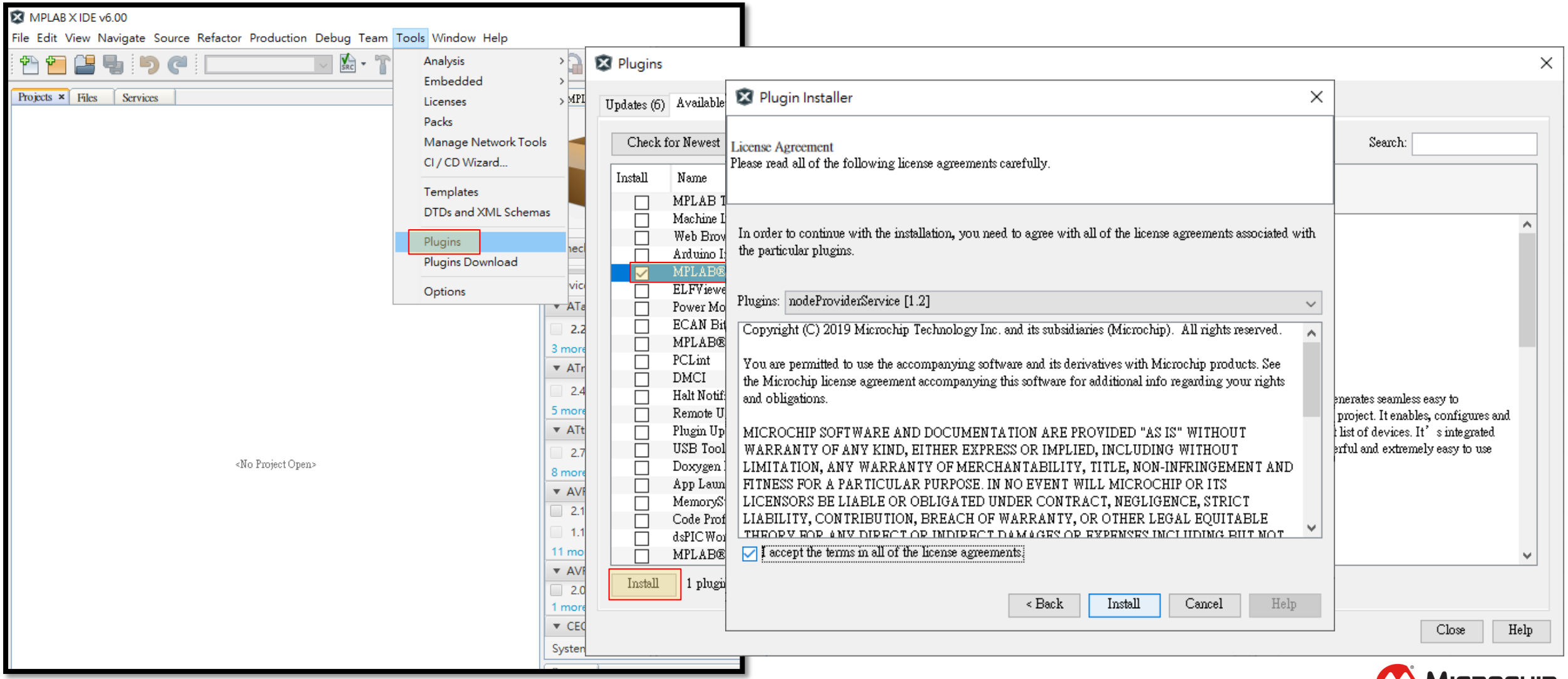
Installing MCC

MPLAB® X IDE / Tools / Plugins



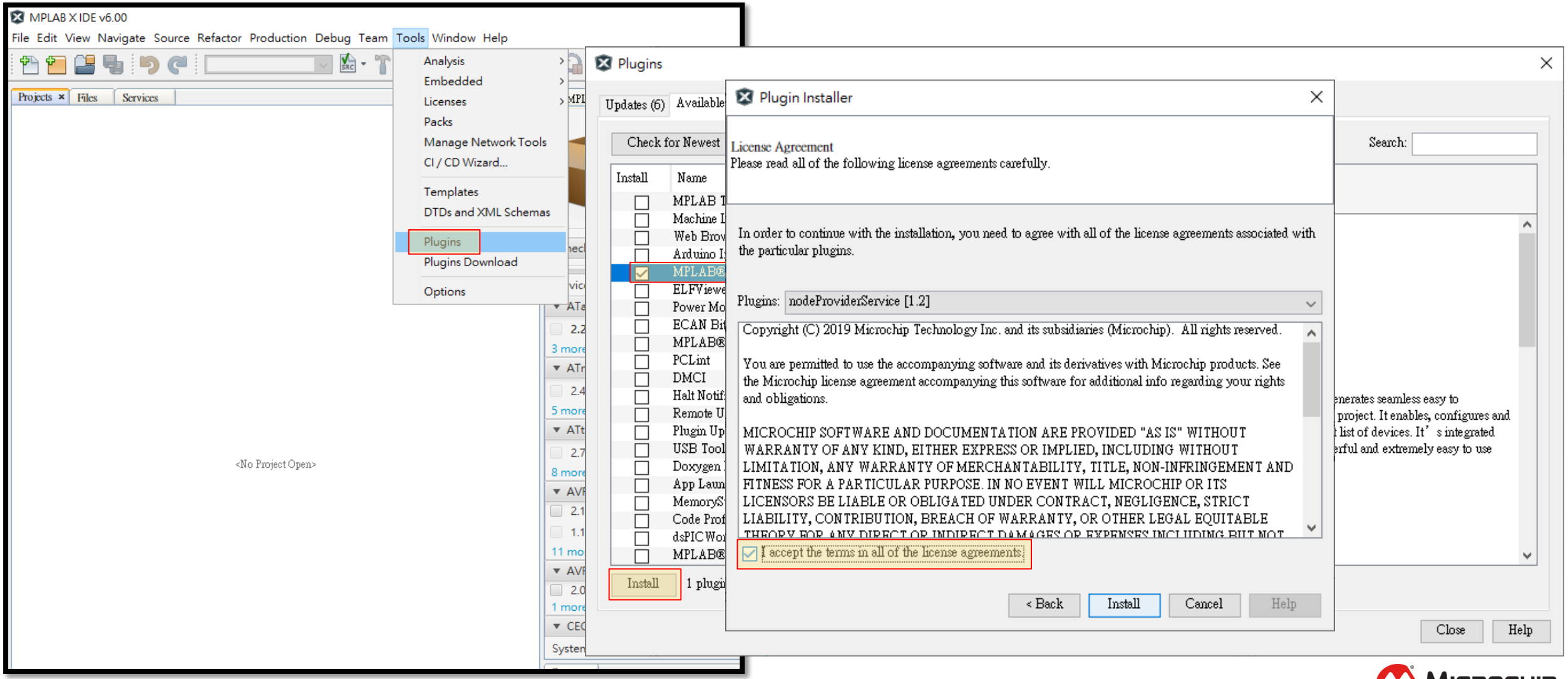
Installing MCC

MPLAB® X IDE / Tools / Plugins



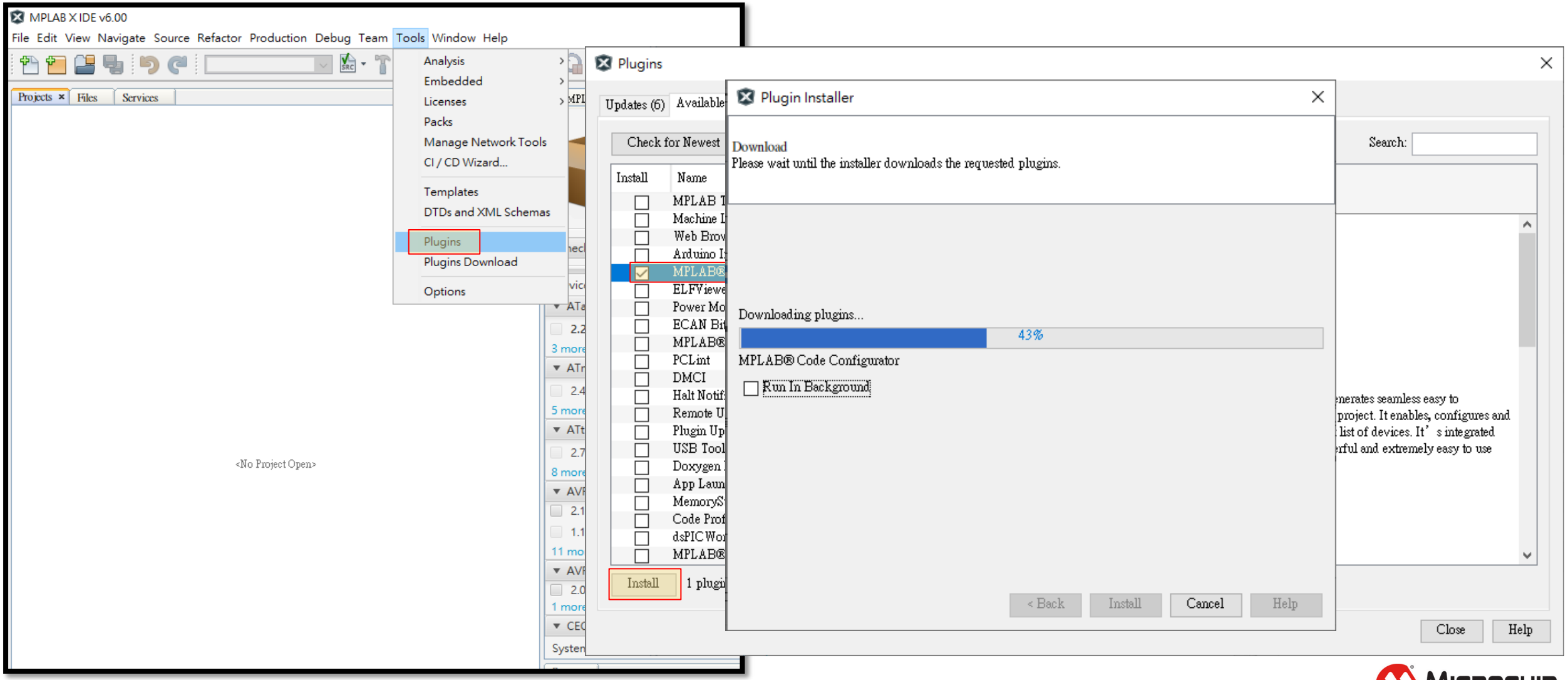
Installing MCC

MPLAB® X IDE / Tools / Plugins



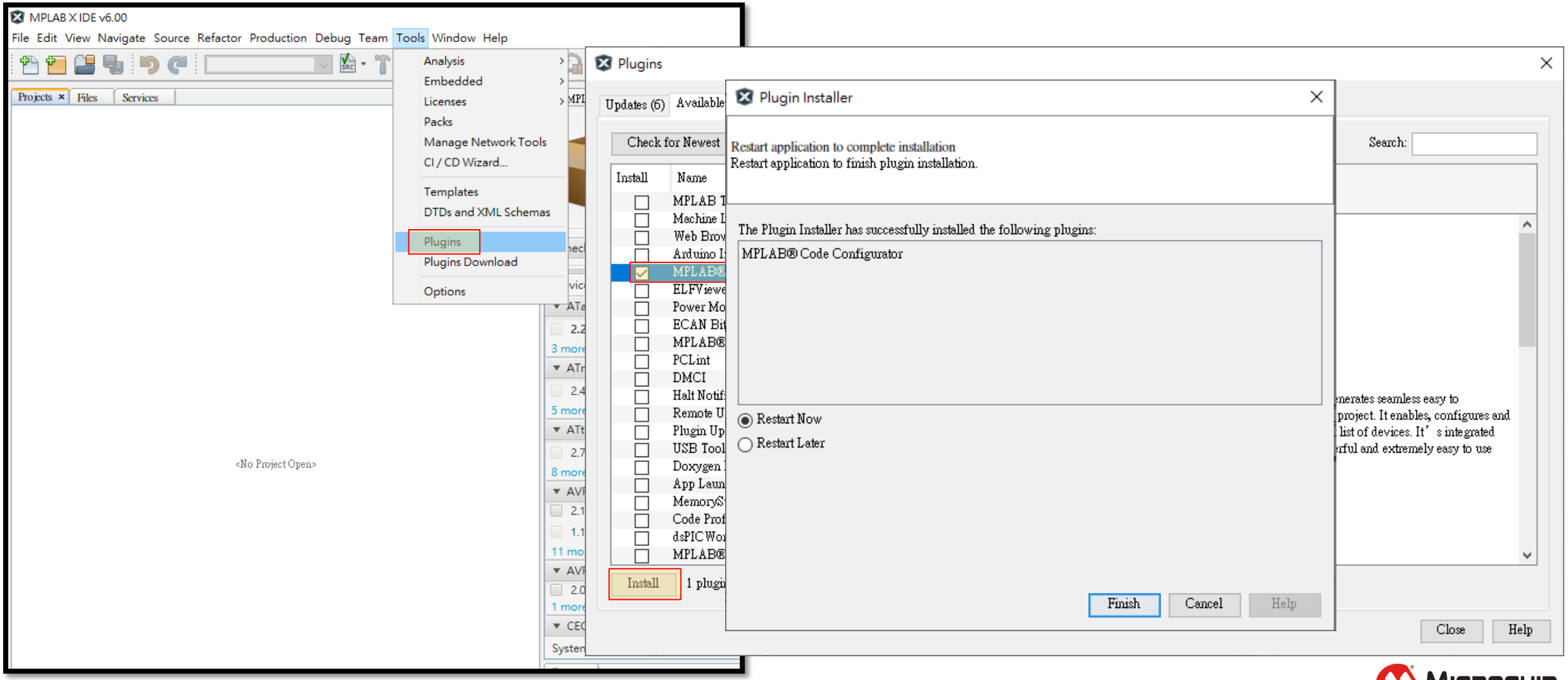
Installing MCC

MPLAB® X IDE / Tools / Plugins



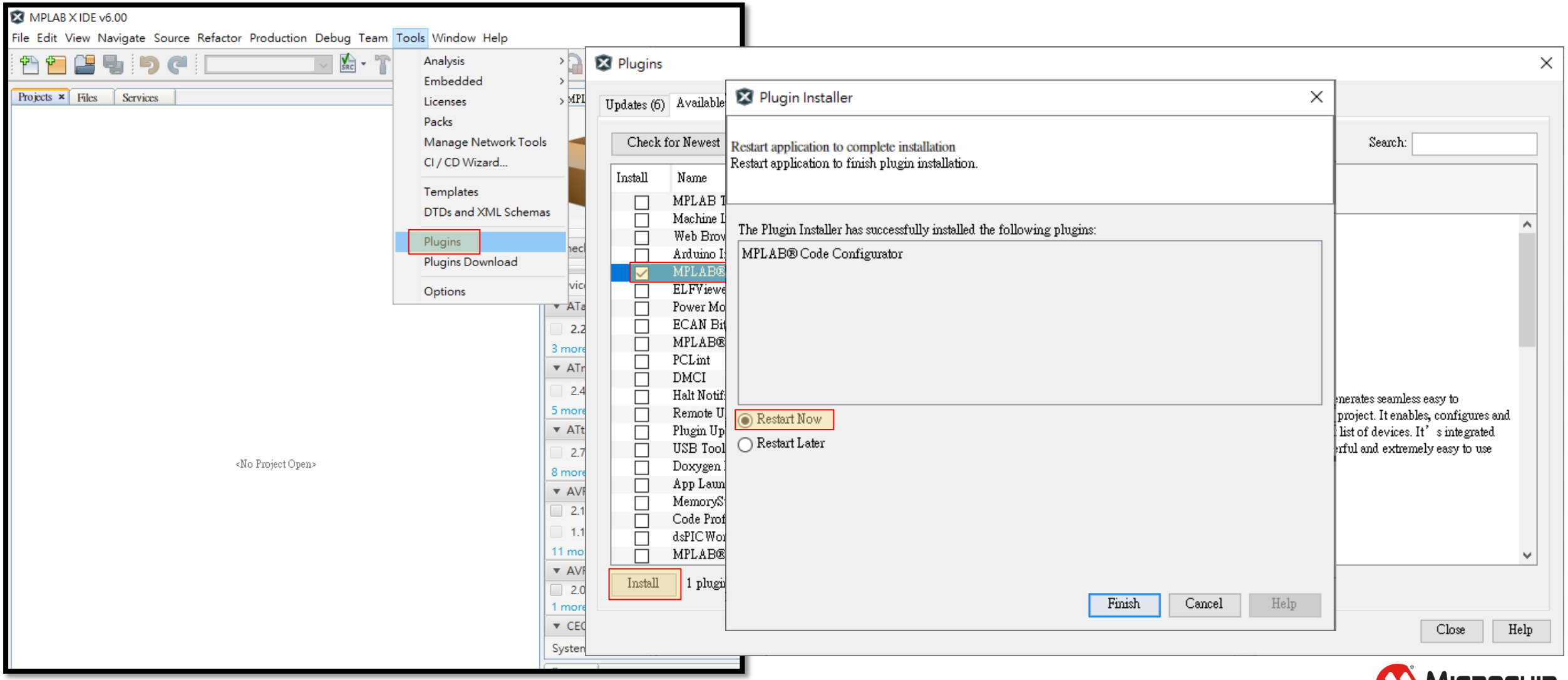
Installing MCC

MPLAB® X IDE / Tools / Plugins



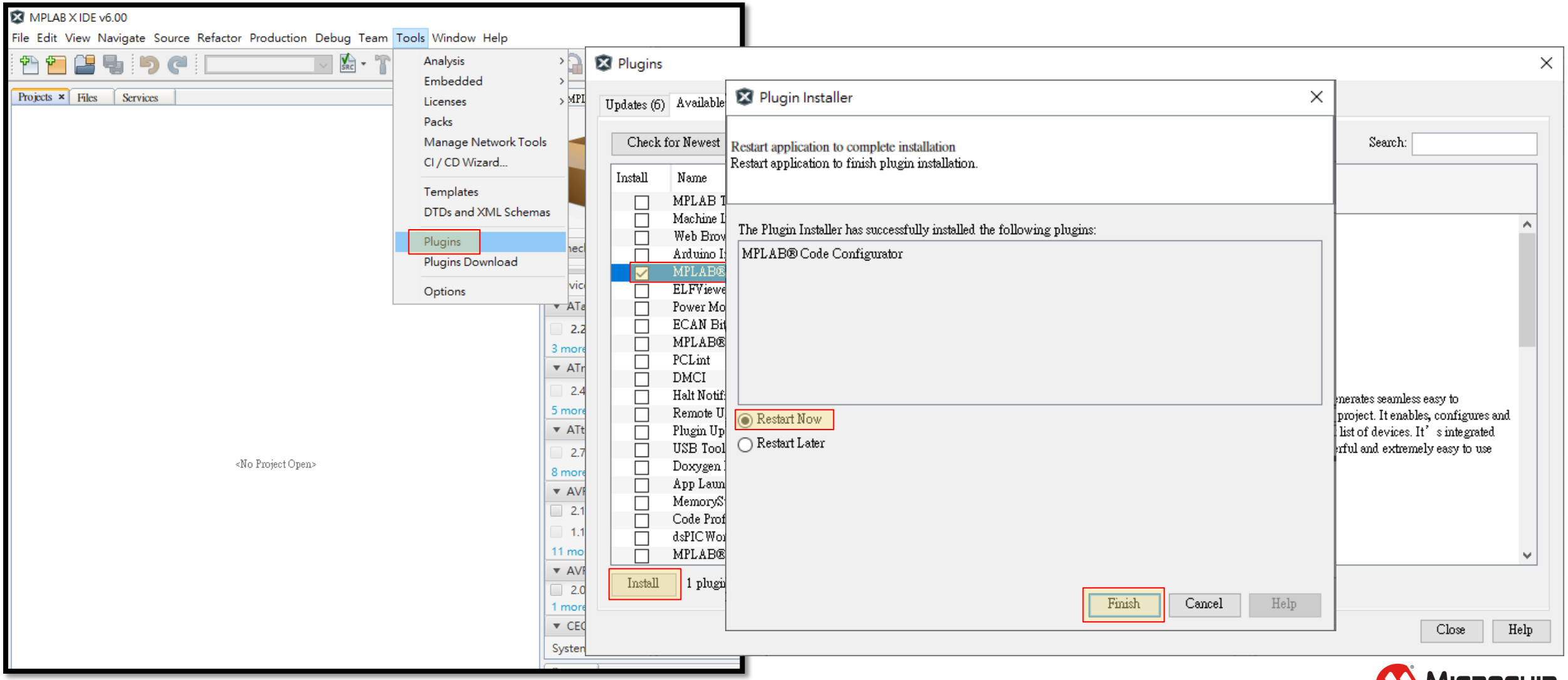
Installing MCC

MPLAB® X IDE / Tools / Plugins



Installing MCC

MPLAB® X IDE / Tools / Plugins



MCC + OS Scheduler for Main Core

Lab1

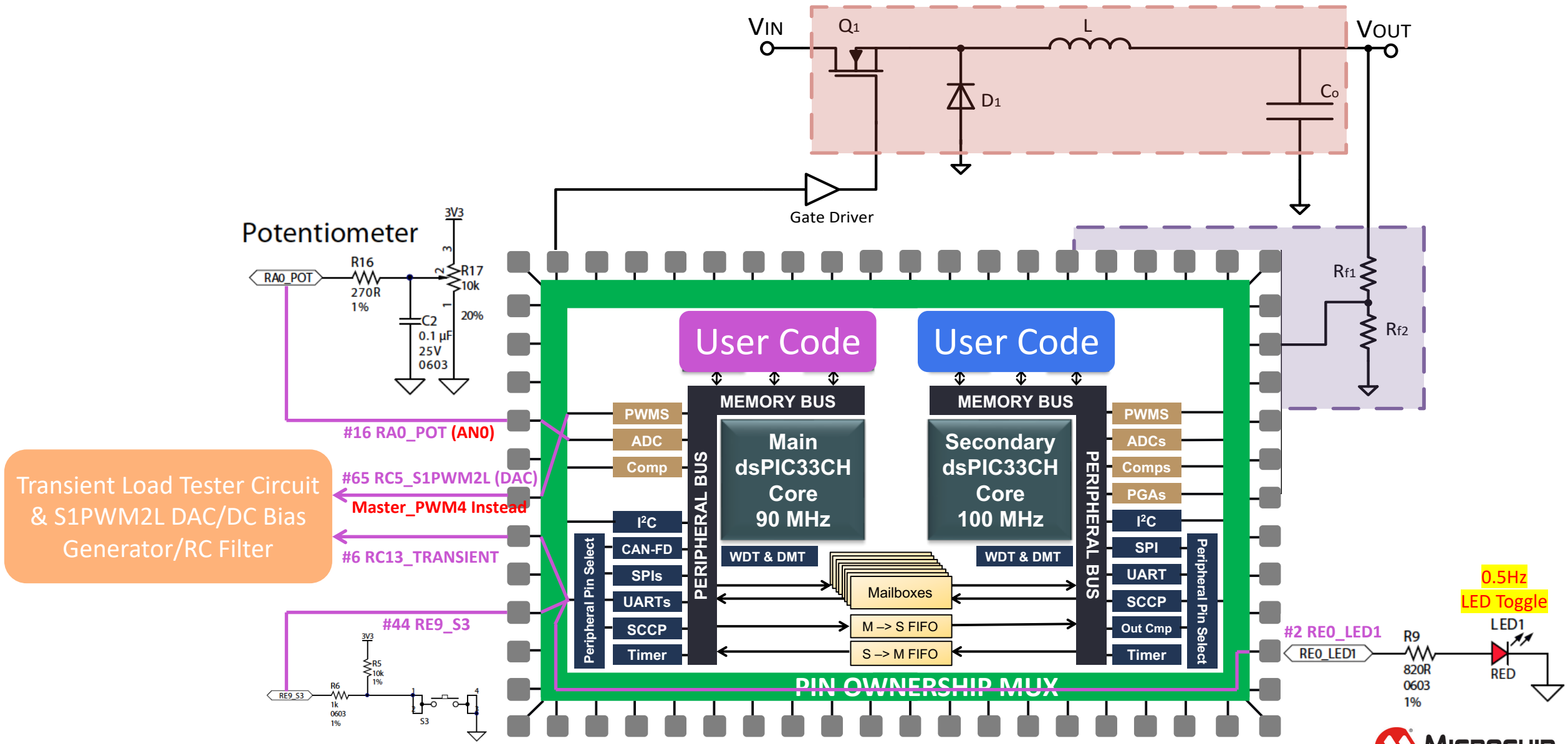
Reference manual on 

https://microchipdeveloper.com/mcc:mplab-code-configurator-support-for-dual-core-devices

Lab #1: MCC + OS Scheduler for Main Core

Main Core

Secondary Core



MICROCHIP

Lab #1: MCC + OS Scheduler for Main Core

Main Core

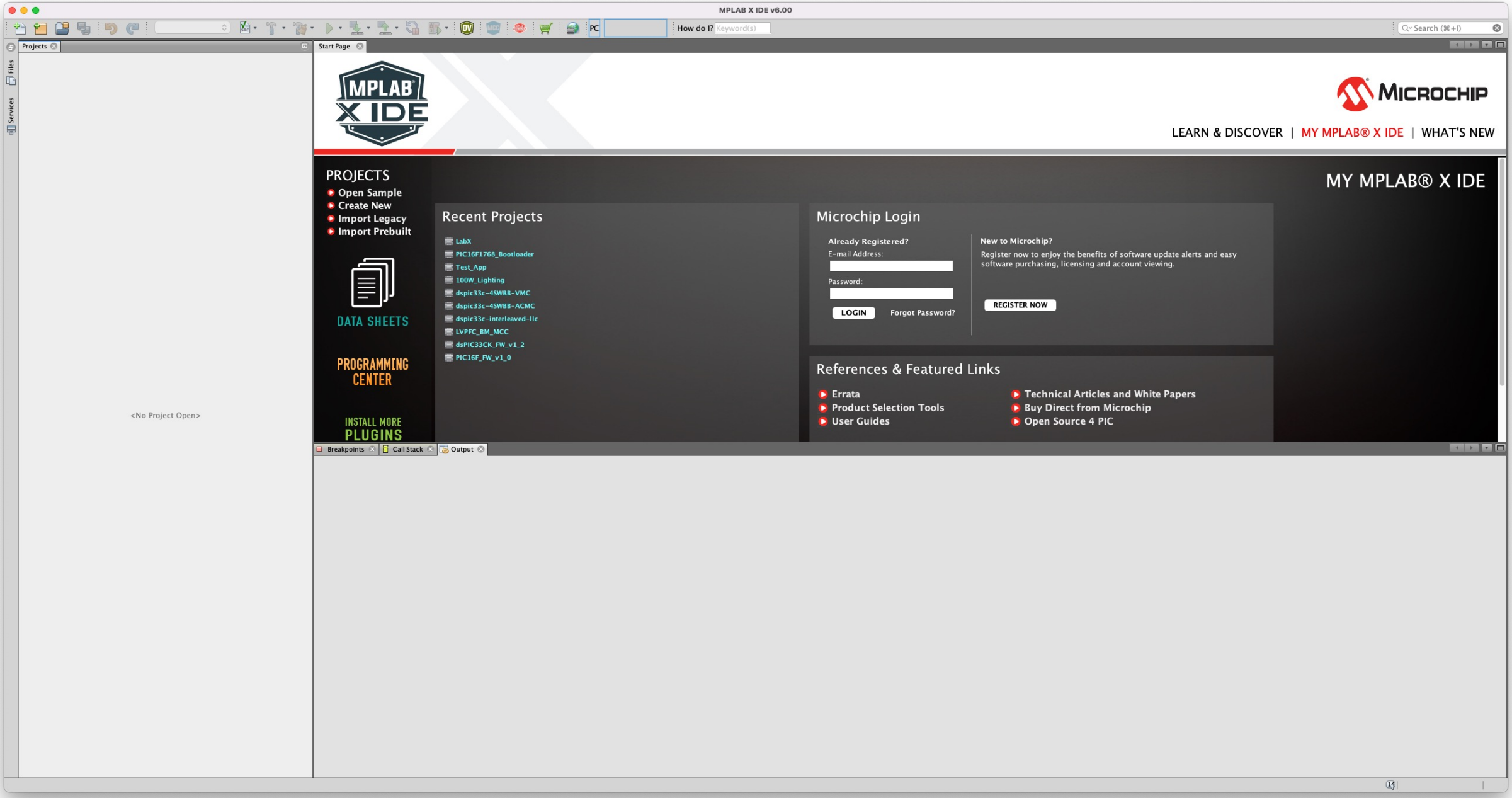
Secondary Core

User Code (90MIPS)

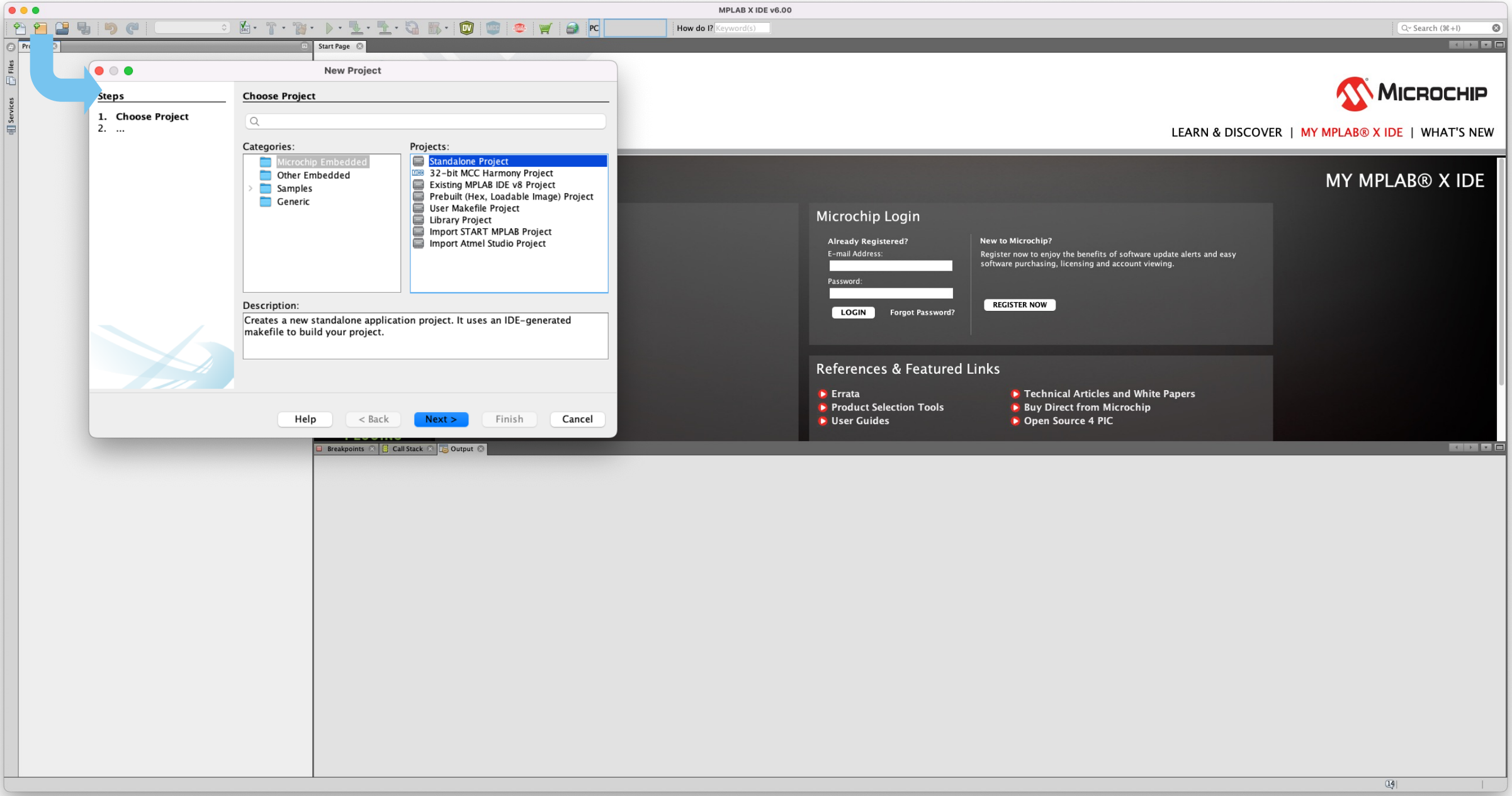
- **Main loop:**
  - OS Scheduler
    - 0.5Hz Toggle LED1
    - Check S3 to trigger RC13_TRANSIENT
- **ISR:**
  - 1ms Timer1 for OS Scheduler
  - AN0-POT to set RC5_PWM4
    - Max 50%
- **Peripheral:**
  - PWM4H – ~**152**KHz (Initial 0% Duty Cycle)
  - I/O: RE9-In, RC13-Out, RE0-Out
  - Timer1 with 1ms ISR
  - AN0 with ISR (IP=1)
    - Triggered by PWM2_Trigger1

User Code (100MIPS)

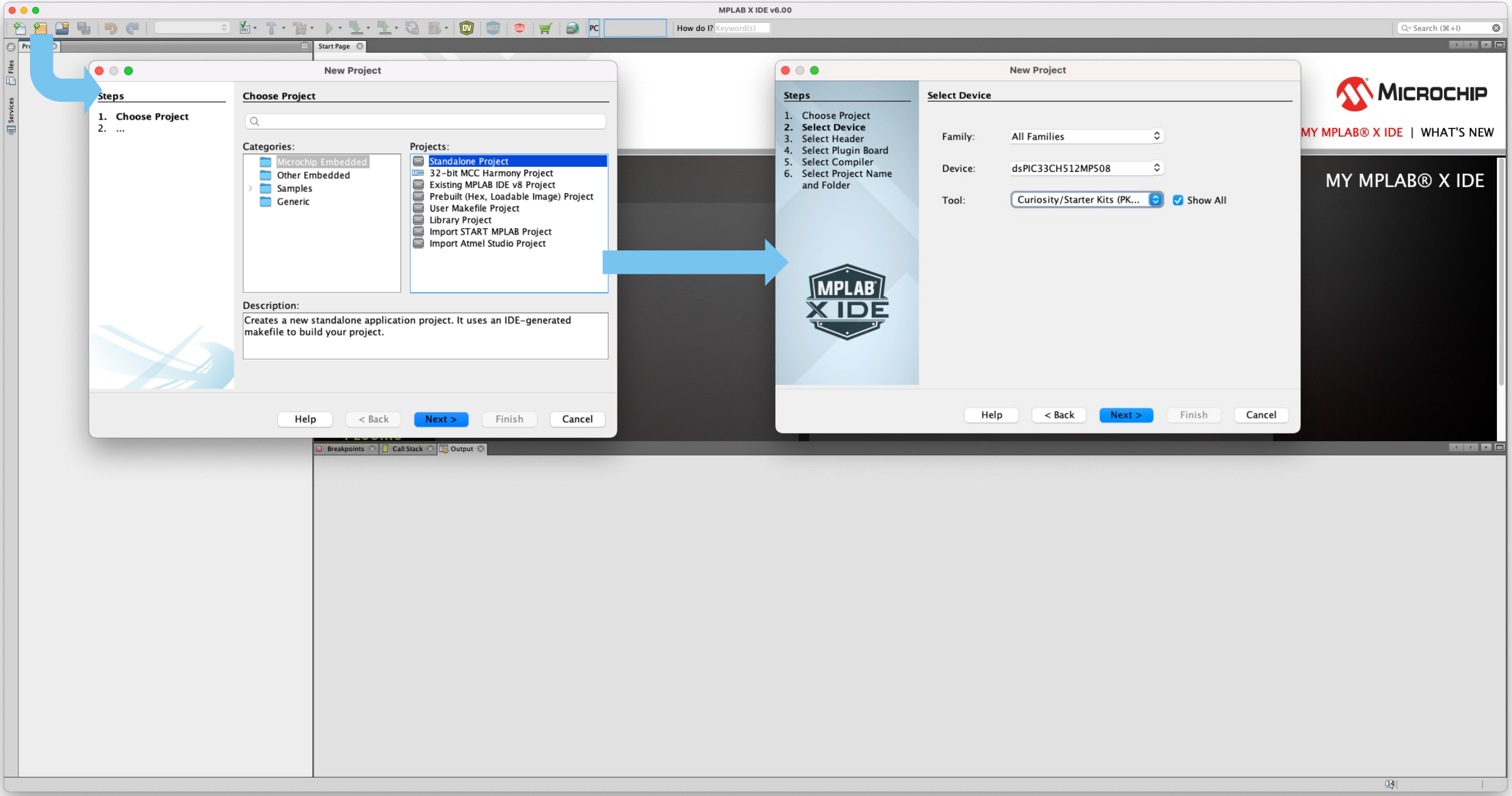
New Project as Master on Main core



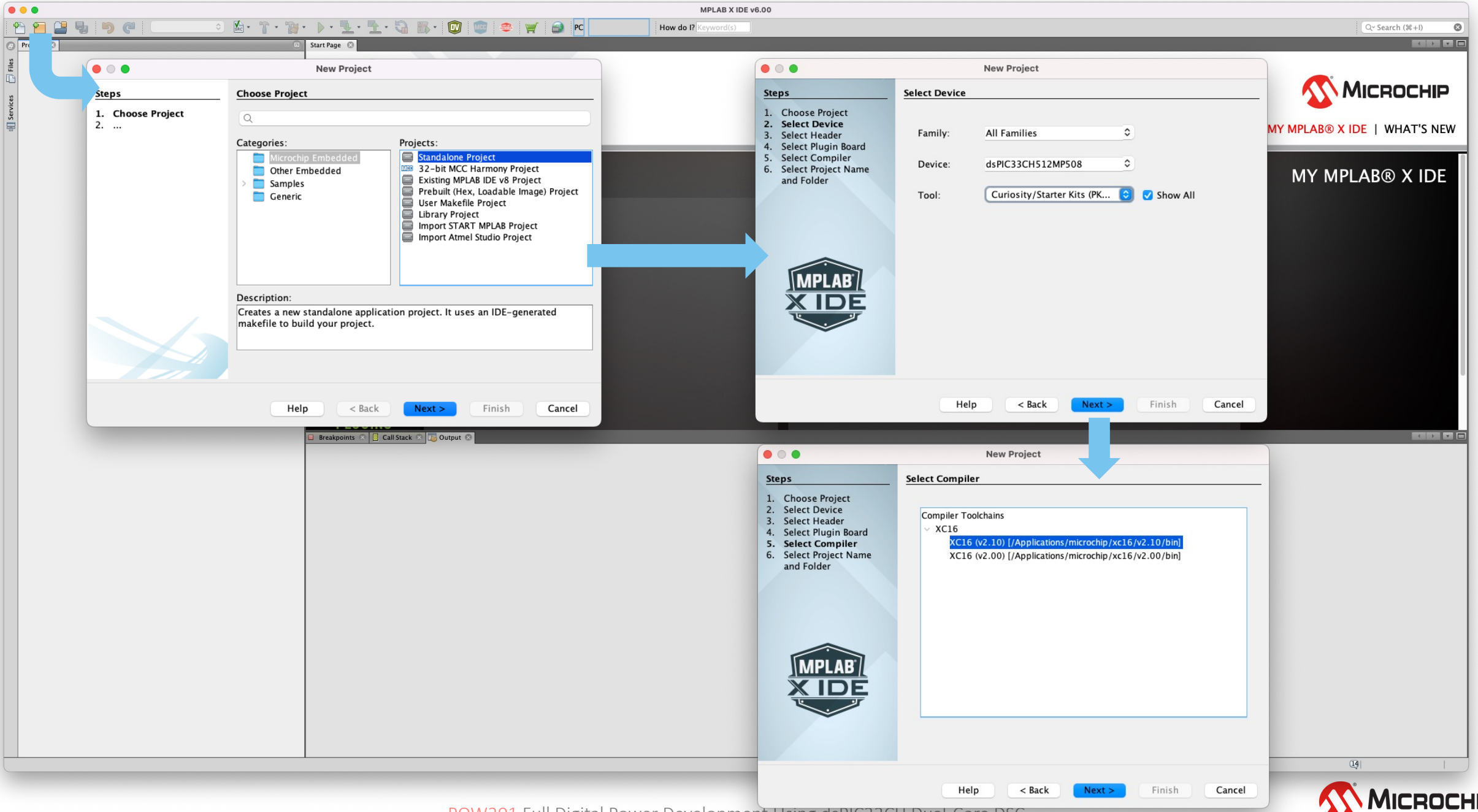
New Project as Master on Main core



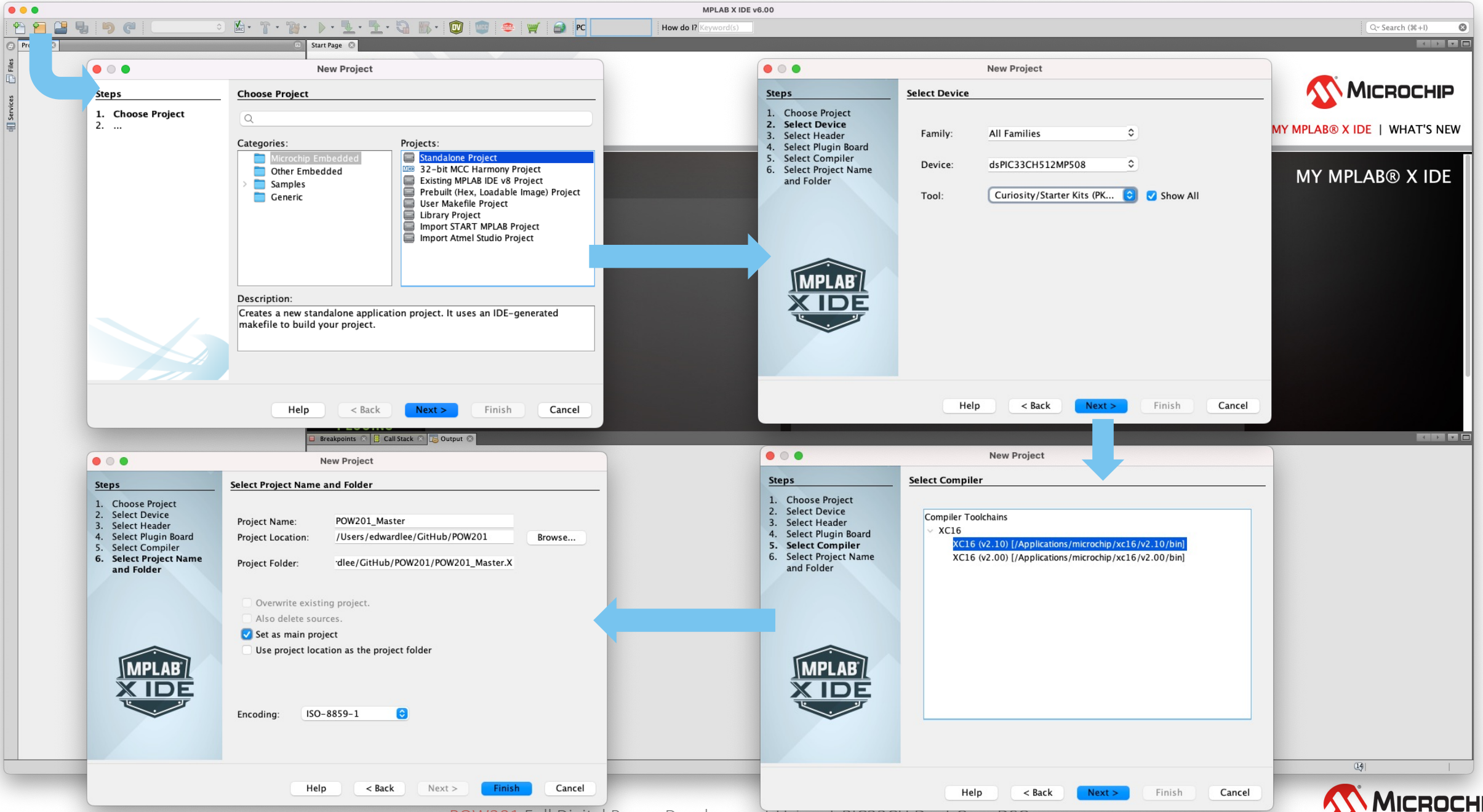
New Project as Master on Main core

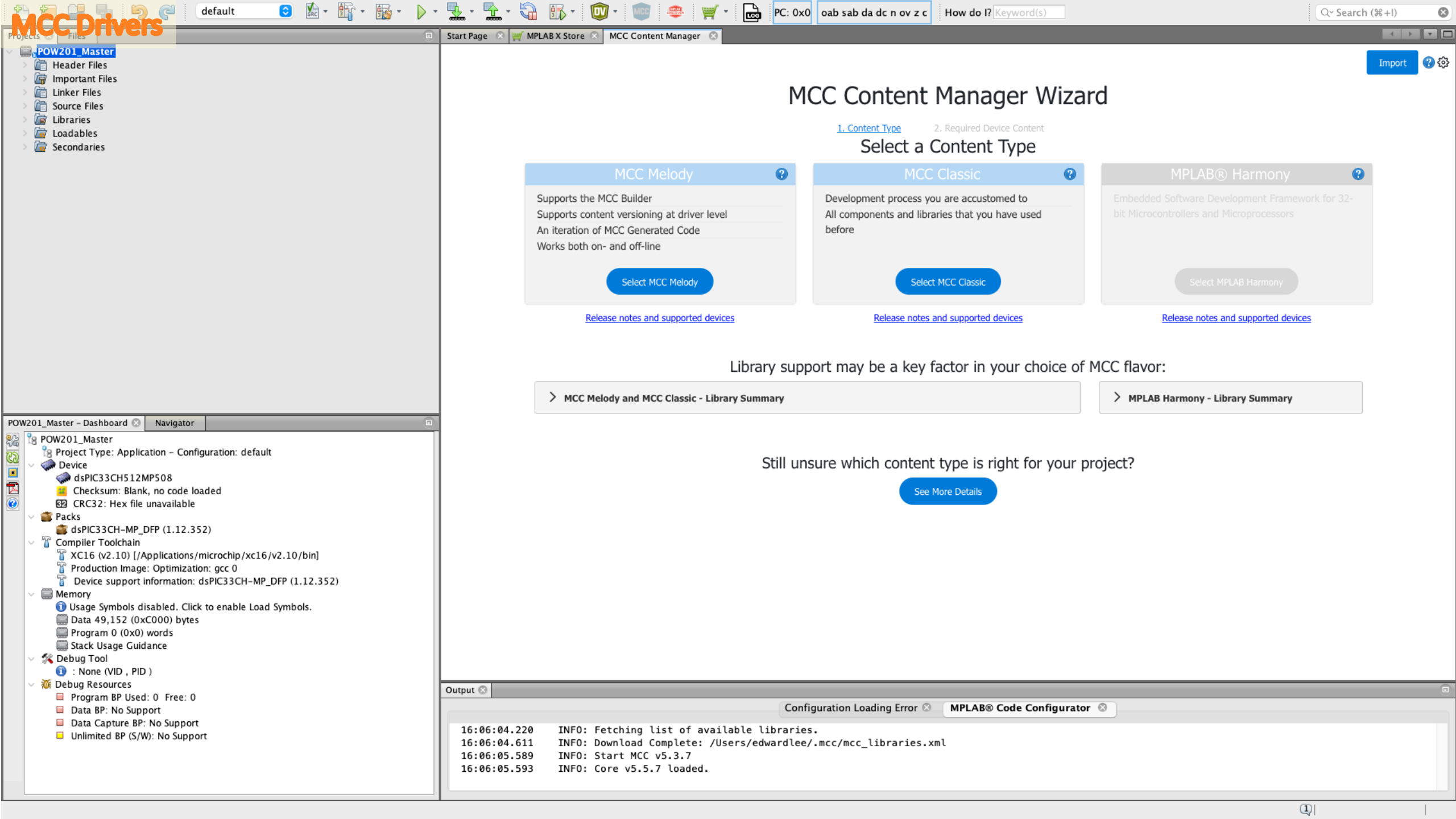


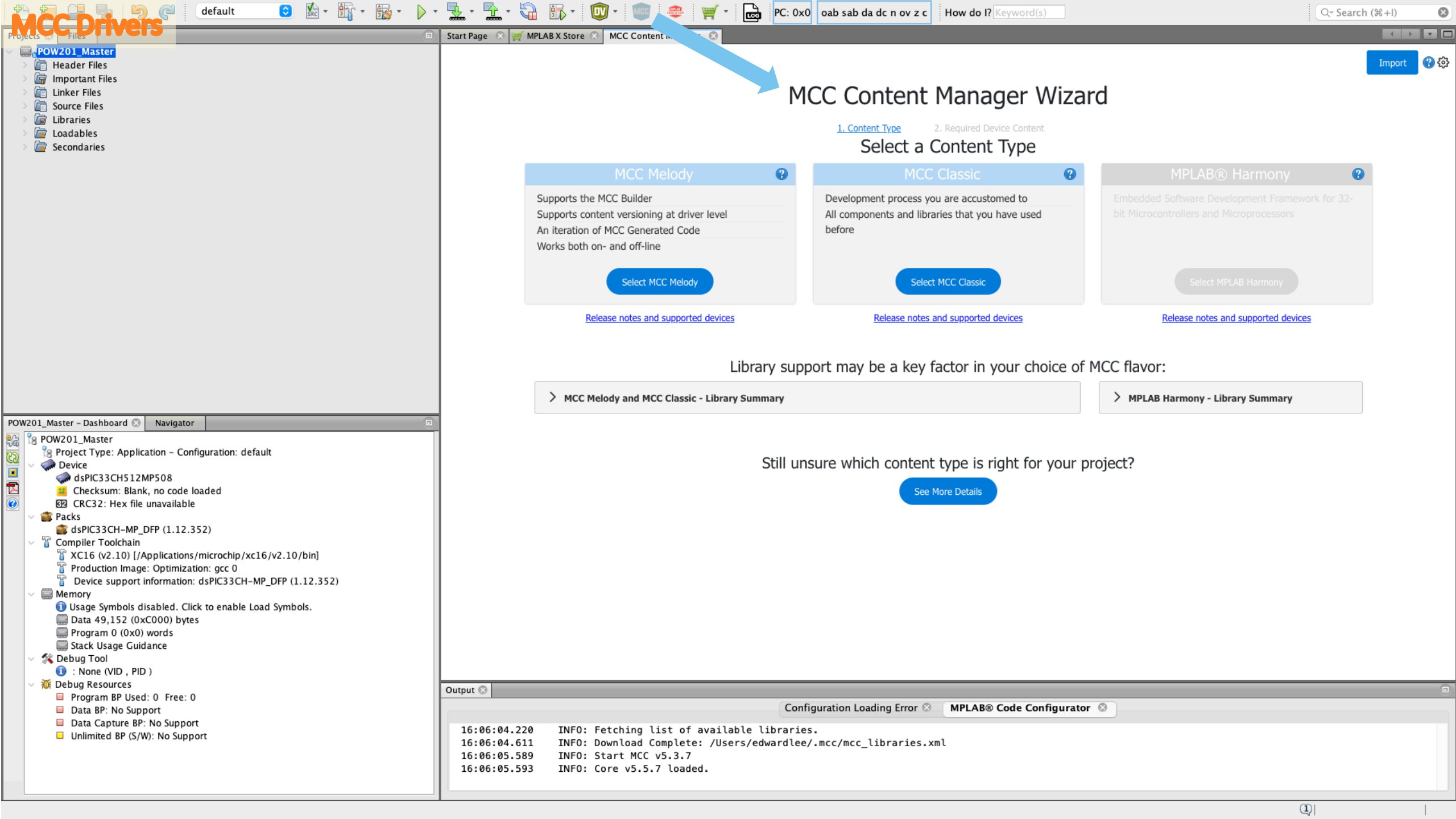
New Project as Master on Main core

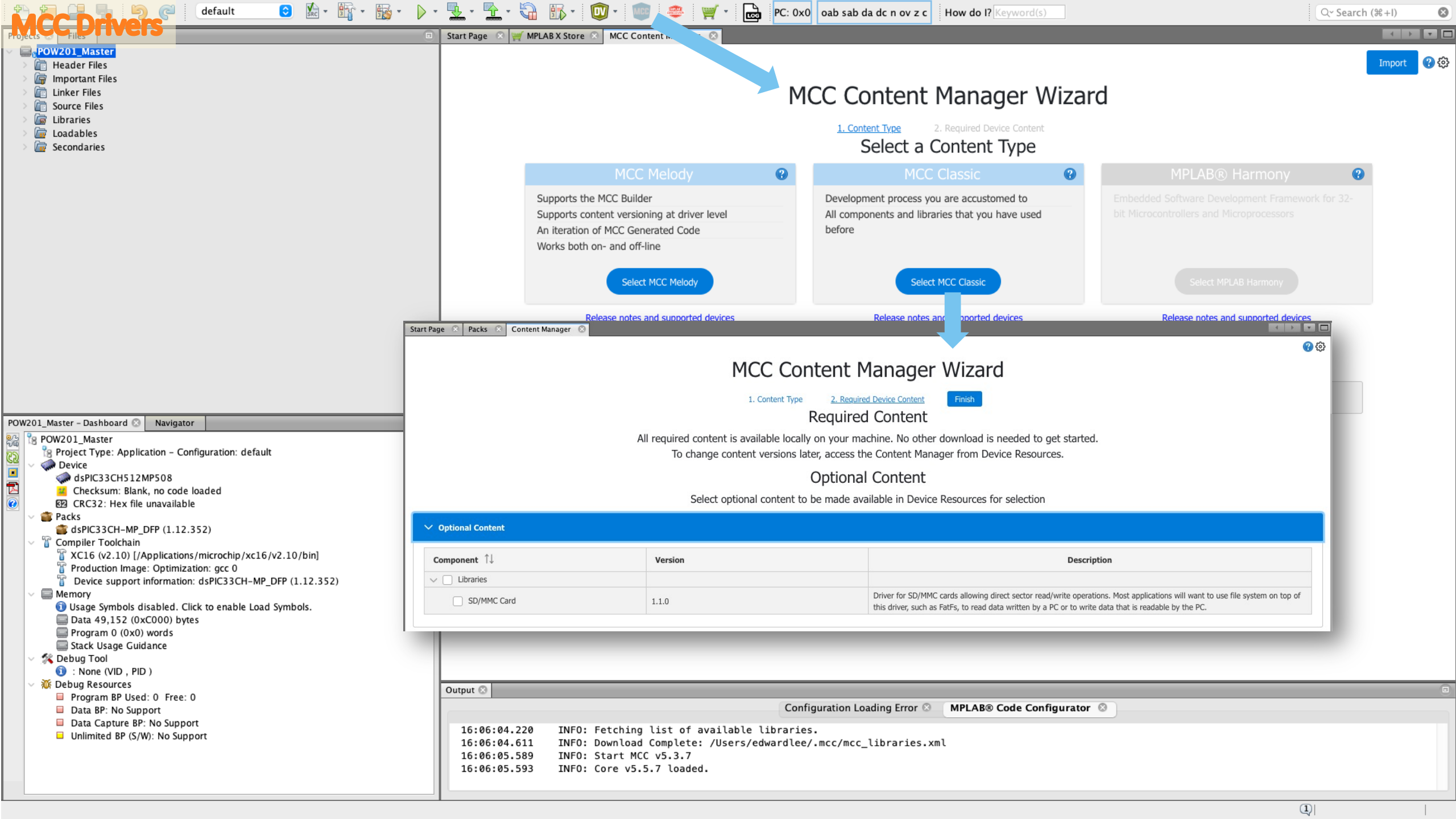


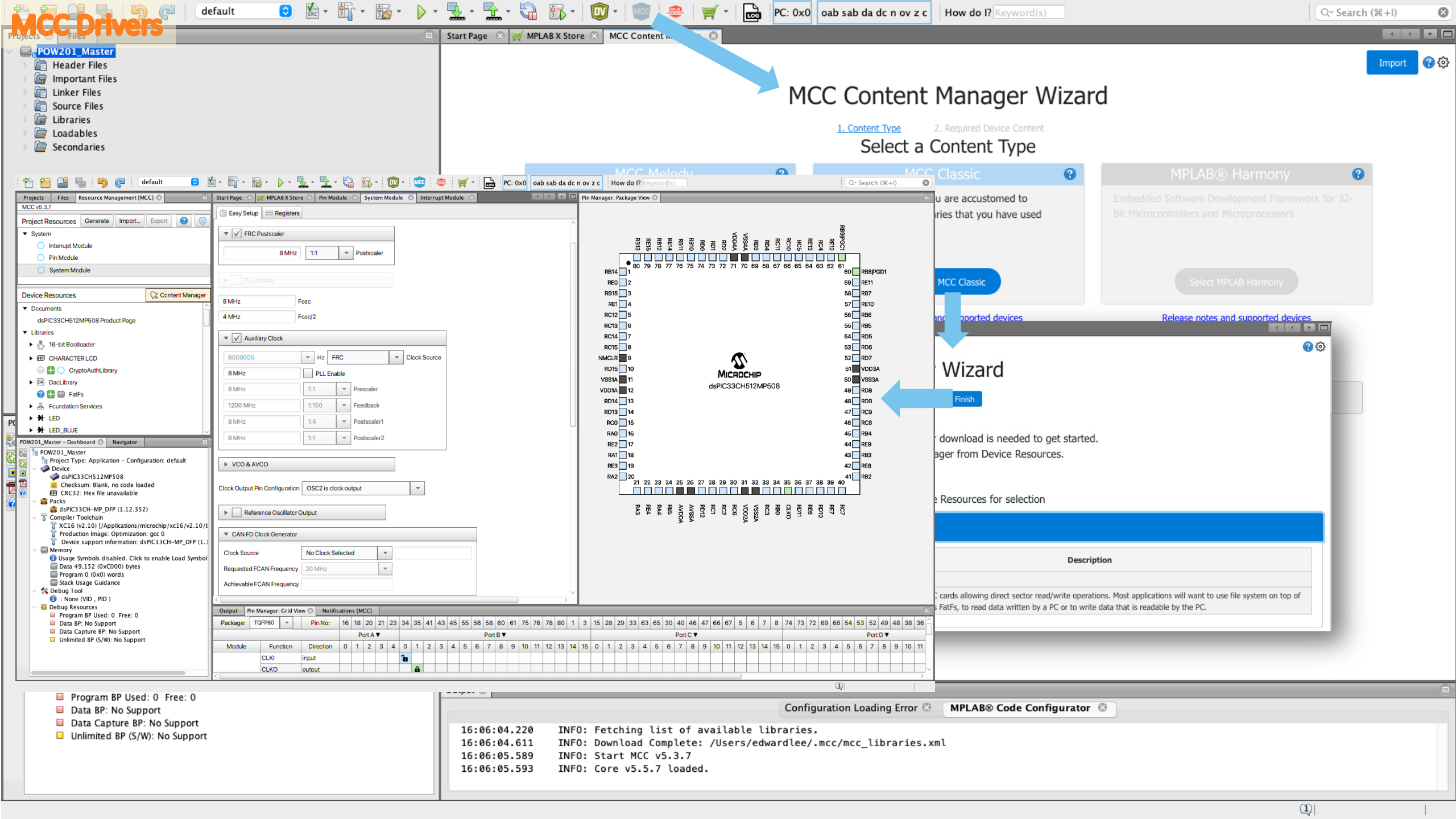
New Project as Master on Main core




MCC Drivers
default
PC: 0x0
oab sab da dc n ov z c
How do I?
Keyword(s)
Search (⌘+I)
Projects
Files
POW201_Master
Header Files
Important Files
Linker Files
Source Files
Libraries
Loadables
Secondaries
Start Page
MPLAB X Store
MCC Content Manager
Import
MCC Content Manager Wizard
1. Content Type
2. Required Device Content
Select a Content Type
MCC Melody
Supports the MCC Builder
Supports content versioning at driver level
An iteration of MCC Generated Code
Works both on- and off-line
Select MCC Melody
Release notes and supported devices
MCC Classic
Development process you are accustomed to
All components and libraries that you have used before
Select MCC Classic
Release notes and supported devices
MPLAB® Harmony
Embedded Software Development Framework for 32-bit Microcontrollers and Microprocessors
Select MPLAB Harmony
Release notes and supported devices
Library support may be a key factor in your choice of MCC flavor:
MCC Melody and MCC Classic - Library Summary
MPLAB Harmony - Library Summary
Still unsure which content type is right for your project?
See More Details
POW201_Master – Dashboard
Navigator
POW201_Master
Project Type: Application – Configuration: default
Device
dsPIC33CH512MP508
Checksum: Blank, no code loaded
CRC32: Hex file unavailable
Packs
dsPIC33CH-MP_DFP (1.12.352)
Compiler Toolchain
XC16 (v2.10) [/Applications/microchip/xc16/v2.10/bin]
Production Image: Optimization: gcc 0
Device support information: dsPIC33CH-MP_DFP (1.12.352)
Memory
Usage Symbols disabled. Click to enable Load Symbols.
Data 49,152 (0xC000) bytes
Program 0 (0x0) words
Stack Usage Guidance
Debug Tool
: None (VID , PID)
Debug Resources
Program BP Used: 0 Free: 0
Data BP: No Support
Data Capture BP: No Support
Unlimited BP (S/W): No Support
Output
Configuration Loading Error
MPLAB® Code Configurator
16:06:04.220 INFO: Fetching list of available libraries.
16:06:04.611 INFO: Download Complete: /Users/edwardlee/.mcc/mcc_libraries.xml
16:06:05.589 INFO: Start MCC v5.3.7
16:06:05.593 INFO: Core v5.5.7 loaded.


MCC Drivers
default
PC: 0x0
oab sab da dc n ov z c
How do I?
Keyword(s)
Search (⌘+I)
Projects
Files
POW201_Master
Header Files
Important Files
Linker Files
Source Files
Libraries
Loadables
Secondaries
Start Page
MPLAB X Store
MCC Content
Import
MCC Content Manager Wizard
1. Content Type
2. Required Device Content
Select a Content Type
MCC Melody
Supports the MCC Builder
Supports content versioning at driver level
An iteration of MCC Generated Code
Works both on- and off-line
Select MCC Melody
MCC Classic
Development process you are accustomed to
All components and libraries that you have used before
Select MCC Classic
MPLAB® Harmony
Embedded Software Development Framework for 32-bit Microcontrollers and Microprocessors
Select MPLAB Harmony
Release notes and supported devices
Release notes and supported devices
Release notes and supported devices
Library support may be a key factor in your choice of MCC flavor:
MCC Melody and MCC Classic - Library Summary
MPLAB Harmony - Library Summary
Still unsure which content type is right for your project?
See More Details
POW201_Master – Dashboard
Navigator
POW201_Master
Project Type: Application – Configuration: default
Device
dsPIC33CH512MP508
Checksum: Blank, no code loaded
CRC32: Hex file unavailable
Packs
dsPIC33CH-MP_DFP (1.12.352)
Compiler Toolchain
XC16 (v2.10) [/Applications/microchip/xc16/v2.10/bin]
Production Image: Optimization: gcc 0
Device support information: dsPIC33CH-MP_DFP (1.12.352)
Memory
Usage Symbols disabled. Click to enable Load Symbols.
Data 49,152 (0xC000) bytes
Program 0 (0x0) words
Stack Usage Guidance
Debug Tool
: None (VID , PID)
Debug Resources
Program BP Used: 0 Free: 0
Data BP: No Support
Data Capture BP: No Support
Unlimited BP (S/W): No Support
Output
Configuration Loading Error
MPLAB® Code Configurator
16:06:04.220 INFO: Fetching list of available libraries.
16:06:04.611 INFO: Download Complete: /Users/edwardlee/.mcc/mcc_libraries.xml
16:06:05.589 INFO: Start MCC v5.3.7
16:06:05.593 INFO: Core v5.5.7 loaded.


MCC Drivers
default
PC: 0x0
oab sab da dc n ov z c
How do I?
Keyword(s)
Search (⌘+I)
POW201_Master
Header Files
Important Files
Linker Files
Source Files
Libraries
Loadables
Secondaries
Start Page
MPLAB X Store
Import
MCC Content Manager Wizard
1. Content Type
2. Required Device Content
Select a Content Type
MCC Melody
Supports the MCC Builder
Supports content versioning at driver level
An iteration of MCC Generated Code
Works both on- and off-line
Select MCC Melody
MCC Classic
Development process you are accustomed to
All components and libraries that you have used before
Select MCC Classic
MPLAB® Harmony
Embedded Software Development Framework for 32-bit Microcontrollers and Microprocessors
Select MPLAB Harmony
Release notes and supported devices
Start Page
Packs
Content Manager
MCC Content Manager Wizard
1. Content Type
2. Required Device Content
Finish
Required Content
All required content is available locally on your machine. No other download is needed to get started.
To change content versions later, access the Content Manager from Device Resources.
Optional Content
Select optional content to be made available in Device Resources for selection
Optional Content
Component
Version
Description
Libraries
SD/MMC Card
1.1.0
Driver for SD/MMC cards allowing direct sector read/write operations. Most applications will want to use file system on top of this driver, such as FatFs, to read data written by a PC or to write data that is readable by the PC.
POW201_Master - Dashboard
Navigator
POW201_Master
Project Type: Application - Configuration: default
Device
dsPIC33CH512MP508
Checksum: Blank, no code loaded
CRC32: Hex file unavailable
Packs
dsPIC33CH-MP_DFP (1.12.352)
Compiler Toolchain
XC16 (v2.10) [/Applications/microchip/xc16/v2.10/bin]
Production Image: Optimization: gcc 0
Device support information: dsPIC33CH-MP_DFP (1.12.352)
Memory
Usage Symbols disabled. Click to enable Load Symbols.
Data 49,152 (0xC000) bytes
Program 0 (0x0) words
Stack Usage Guidance
Debug Tool
: None (VID , PID)
Debug Resources
Program BP Used: 0 Free: 0
Data BP: No Support
Data Capture BP: No Support
Unlimited BP (S/W): No Support
Output
Configuration Loading Error
MPLAB® Code Configurator
16:06:04.220 INFO: Fetching list of available libraries.
16:06:04.611 INFO: Download Complete: /Users/edwardlee/.mcc/mcc_libraries.xml
16:06:05.589 INFO: Start MCC v5.3.7
16:06:05.593 INFO: Core v5.5.7 loaded.


MCC Drivers
POW201_Master
Header Files
Important Files
Linker Files
Source Files
Libraries
Loadables
Secondaries
Start Page
MPLAB X Store
MCC Content
Import
MCC Content Manager Wizard
1. Content Type
2. Required Device Content
Select a Content Type
MCC Classic
MPLAB® Harmony
Embedded Software Development Framework for 32-bit Microcontrollers and Microprocessors
Select MPLAB Harmony
Release notes and supported devices
Wizard
Finish
download is needed to get started.
ager from Device Resources.
Resources for selection
Description
cards allowing direct sector read/write operations. Most applications will want to use file system on top of
FatFs, to read data written by a PC or to write data that is readable by the PC.
Resource Management [MCC]
Project Resources
Generate
Import...
Export
System
Interrupt Module
Pin Module
System Module
Device Resources
Content Manager
Documents
dsPIC33CH512MP508 Product Page
Libraries
16-bit Bootloader
CHARACTER LCD
CryptoAuthLibrary
DacLibrary
FatFs
Foundation Services
LED
LED_BLUE
Easy Setup
Registers
FRC Postscaler
8 MHz
1:1
Postscaler
PLL Enable
8 MHz
Fosc
4 MHz
Fosc/2
Auxiliary Clock
8000000
Hz
FRC
Clock Source
8 MHz
PLL Enable
8 MHz
1:1
Prescaler
1200 MHz
1:150
Feedback
8 MHz
1:4
Postscaler1
8 MHz
1:1
Postscaler2
VCO & AVCO
Clock Output Pin Configuration
OSC2 is clock output
Reference Oscillator Output
CAN FD Clock Generator
Clock Source
No Clock Selected
Requested FCAN Frequency
20 MHz
Achievable FCAN Frequency
Pin Manager: Package View
MICROCHIP
dsPIC33CH512MP508
POW201_Master - Dashboard
Navigator
Project Type: Application - Configuration: default
Device
dsPIC33CH512MP508
Checksum: Blank, no code loaded
CRC32: Hex file unavailable
Packs
dsPIC33CH-MP_DFP (1.12.352)
Compiler Toolchain
Memory
Usage Symbols disabled. Click to enable Load Symbol
Data 49,152 (0xC000) bytes
Program 0 (0x0) words
Stack Usage Guidance
Debug Tool
: None (VID , PID)
Debug Resources
Program BP Used: 0 Free: 0
Data BP: No Support
Data Capture BP: No Support
Unlimited BP (S/W): No Support
Output
Pin Manager: Grid View
Notifications [MCC]
Package:
TQFP80
Pin No:
Port A
Port B
Port C
Port D
Module
Function
Direction
CLKI
input
CLKO
output
Configuration Loading Error
MPLAB® Code Configurator
16:06:04.220 INFO: Fetching list of available libraries.
16:06:04.611 INFO: Download Complete: /Users/edwardlee/.mcc/mcc_libraries.xml
16:06:05.589 INFO: Start MCC v5.3.7
16:06:05.593 INFO: Core v5.5.7 loaded.

MCC Drivers – Clock@90 MIPS

MCC v5.3.7

Project Resources | Generate | Import... | Export

- System
  - Interrupt Module
  - Pin Module
  - System Module

Device Resources | Content Manager

- Documents
  - dsPIC33CH512MP508 Product Page
- Libraries
  - 16-bit Bootloader
  - CHARACTER LCD
  - CryptoAuthLibrary
  - DacLibrary
  - FatFs
  - Foundation Services
  - LED
  - LED_BLUE
  - LED_GREEN

POW201_Master – Dashboard | Navigator

- POW201_Master
  - Project Type: Application – Configuration: default
  - Device
    - dsPIC33CH512MP508
    - Checksum: Blank, no code loaded
    - CRC32: Hex file unavailable
  - Packs
    - dsPIC33CH-MP_DFP (1.12.352)
  - Compiler Toolchain
    - XC16 (v2.10) [/Applications/microchip/xc16/v2.10/bi
    - Production Image: Optimization: gcc 0
    - Device support information: dsPIC33CH-MP_DFP (1.12
  - Memory
    - Usage Symbols disabled. Click to enable Load Symbols.
    - Data 49,152 (0xC000) bytes
    - Program 0 (0x0) words
    - Stack Usage Guidance
  - Debug Tool
    - : None (VID , PID)
  - Debug Resources
    - Program BP Used: 0 Free: 0
    - Data BP: No Support
    - Data Capture BP: No Support
    - Unlimited BP (S/W): No Support

MPLAB X Store | Pin Module | System Module | Interrupt Module

Easy Setup | Registers

Clock

8000000 Hz FRC Oscillator (8.0 MHz) Clock Source

FRC Postscaler

PLL Enable

8 MHz	1:1	Prescaler
1440 MHz	1:180	FeedBack
720 MHz	1:2	Postscaler1
360 MHz	1:2	Postscaler2

180 MHz Fosc

90 MHz Fosc/2

Auxiliary Clock

8000000 Hz	FRC	Clock Source
8 MHz	PLL Enable	
8 MHz	1:1	Prescaler
1000 MHz	1:125	Feedback
500 MHz	1:2	Postscaler1
500 MHz	1:1	Postscaler2

VCO & AVCO

360 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

Pin Manager: Package View

6.1 Primary PLL

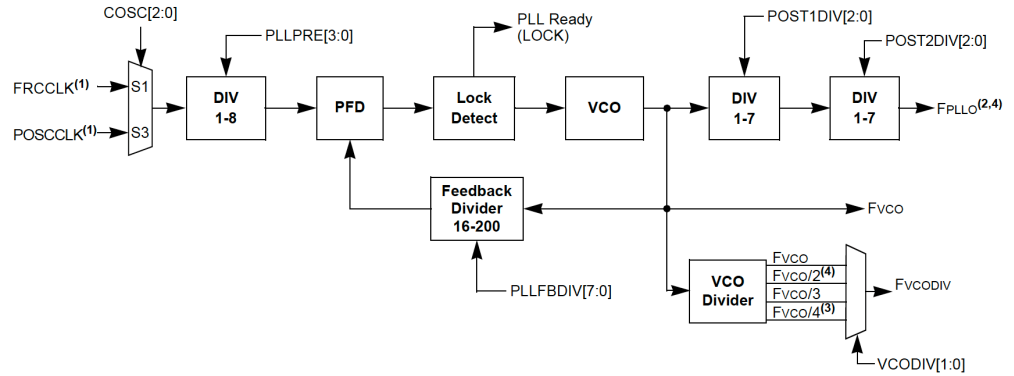
The Primary Oscillator and internal FRC Oscillator sources can optionally use an on-chip PLL to obtain higher operating speeds. There are two independent instantiations of PLL for the Master and Slave clock subsystems. Figure 6-4 illustrates a block diagram of the Master/Slave core PLL module.

For PLL operation, the following requirements must be met at all times without exception:

- The PLL Input Frequency (FPLLI) must be in the range of 8 MHz to 64 MHz
- The PFD Input Frequency (FPFD) must be in the range of 8 MHz to (FVCO/16) MHz

The VCO Output Frequency (FVCO) must be in the range of 400 MHz to 1600 MHz

FIGURE 6-4: MASTER/SLAVE CORE PLL AND VCO DETAIL



Note 1: From Master and Slave core shared oscillator source.
2: Clock option for PWM.
3: Clock option for ADC.
4: Clock option for DAC.

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

MCC Drivers – Clock@90 MIPS

MCC v5.3.7

Project Resources | Generate | Import... | Export

- System
  - Interrupt Module
  - Pin Module
  - System Module

Device Resources | Content Manager

- Documents
  - dsPIC33CH512MP508 Product Page
- Libraries
  - 16-bit Bootloader
  - CHARACTER LCD
  - CryptoAuthLibrary
  - DacLibrary
  - FatFs
  - Foundation Services
  - LED
  - LED_BLUE
  - LED_GREEN

POW201_Master – Dashboard | Navigator

- POW201_Master
  - Project Type: Application – Configuration: default
  - Device
    - dsPIC33CH512MP508
    - Checksum: Blank, no code loaded
    - CRC32: Hex file unavailable
  - Packs
    - dsPIC33CH-MP_DFP (1.12.352)
  - Compiler Toolchain
    - XC16 (v2.10) [/Applications/microchip/xc16/v2.10/bi
    - Production Image: Optimization: gcc 0
    - Device support information: dsPIC33CH-MP_DFP (1.12
  - Memory
    - Usage Symbols disabled. Click to enable Load Symbols.
    - Data 49,152 (0xC000) bytes
    - Program 0 (0x0) words
    - Stack Usage Guidance
  - Debug Tool
    - : None (VID , PID)
  - Debug Resources
    - Program BP Used: 0 Free: 0
    - Data BP: No Support
    - Data Capture BP: No Support
    - Unlimited BP (S/W): No Support

MPLAB X Store | Pin Module | System Module | Interrupt Module

Easy Setup | Registers

Clock

8000000 Hz FRC Oscillator (8.0 MHz) Clock Source

FRC Postscaler

PLL Enable

8 MHz	1:1	Prescaler
1440 MHz	1:180	FeedBack
720 MHz	1:2	Postscaler1
360 MHz	1:2	Postscaler2

180 MHz Fosc

90 MHz Fosc/2 **90MIPS**

Auxiliary Clock

8000000 Hz FRC Clock Source

8 MHz PLL Enable

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

VCO & AVCO

360 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

Pin Manager: Package View

6.1 Primary PLL

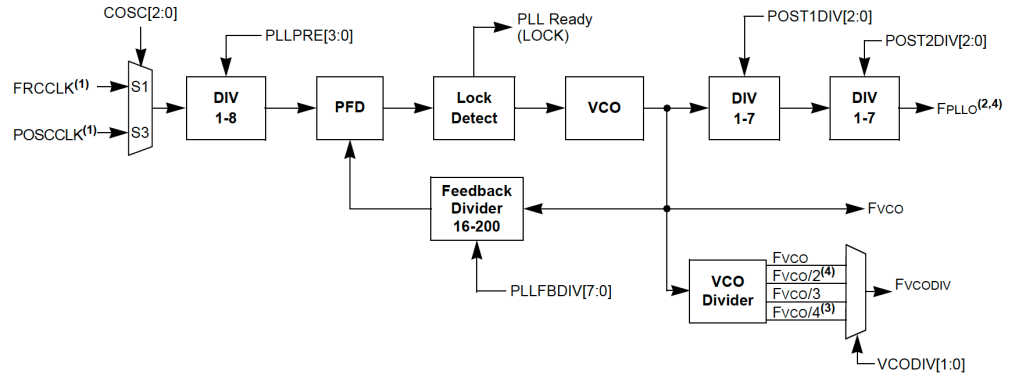
The Primary Oscillator and internal FRC Oscillator sources can optionally use an on-chip PLL to obtain higher operating speeds. There are two independent instantiations of PLL for the Master and Slave clock subsystems. Figure 6-4 illustrates a block diagram of the Master/Slave core PLL module.

For PLL operation, the following requirements must be met at all times without exception:

- The PLL Input Frequency (FPLLI) must be in the range of 8 MHz to 64 MHz
- The PFD Input Frequency (FPFD) must be in the range of 8 MHz to (FVCO/16) MHz

The VCO Output Frequency (FVCO) must be in the range of 400 MHz to 1600 MHz

FIGURE 6-4: MASTER/SLAVE CORE PLL AND VCO DETAIL



Note 1: From Master and Slave core shared oscillator source.
2: Clock option for PWM.
3: Clock option for ADC.
4: Clock option for DAC.

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

MCC Drivers – Clock@90 MIPS

MCC v5.3.7

Project Resources | Generate | Import... | Export

- System
  - Interrupt Module
  - Pin Module
  - System Module

Device Resources | Content Manager

- Documents
  - dsPIC33CH512MP508 Product Page
- Libraries
  - 16-bit Bootloader
  - CHARACTER LCD
  - CryptoAuthLibrary
  - DacLibrary
  - FatFs
  - Foundation Services
  - LED
  - LED_BLUE
  - LED_GREEN

POW201_Master - Dashboard | Navigator

- POW201_Master
  - Project Type: Application - Configuration: default
  - Device
    - dsPIC33CH512MP508
    - Checksum: Blank, no code loaded
    - CRC32: Hex file unavailable
  - Packs
    - dsPIC33CH-MP_DFP (1.12.352)
  - Compiler Toolchain
    - XC16 (v2.10) [/Applications/microchip/xc16/v2.10/bi
    - Production Image: Optimization: gcc 0
    - Device support information: dsPIC33CH-MP_DFP (1.12
  - Memory
    - Usage Symbols disabled. Click to enable Load Symbols.
    - Data 49,152 (0xC000) bytes
    - Program 0 (0x0) words
    - Stack Usage Guidance
  - Debug Tool
    - : None (VID , PID)
  - Debug Resources
    - Program BP Used: 0 Free: 0
    - Data BP: No Support
    - Data Capture BP: No Support
    - Unlimited BP (S/W): No Support

MPLAB X Store | Pin Module | System Module | Interrupt Module

Easy Setup | Registers

Clock

8000000 Hz FRC Oscillator (8.0 MHz) Clock Source

FRC Postscaler

PLL Enable

8 MHz	1:1	Prescaler
1440 MHz	1:180	FeedBack
720 MHz	1:2	Postscaler1
360 MHz	1:2	Postscaler2

180 MHz Fosc

90 MHz Fosc/2 **90MIPS**

Auxiliary Clock

8000000 Hz FRC Clock Source

8 MHz PLL Enable

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

VCO & AVCO

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

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

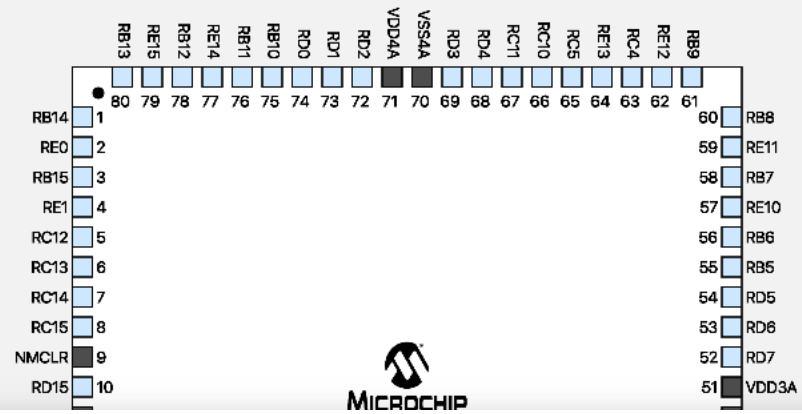
Reference Oscillator Output

CAN FD Clock Generator

ICD

Emulator Pin Placement: Communicate on PGC2 and PGD2

Pin Manager: Package View



6.1 Primary PLL

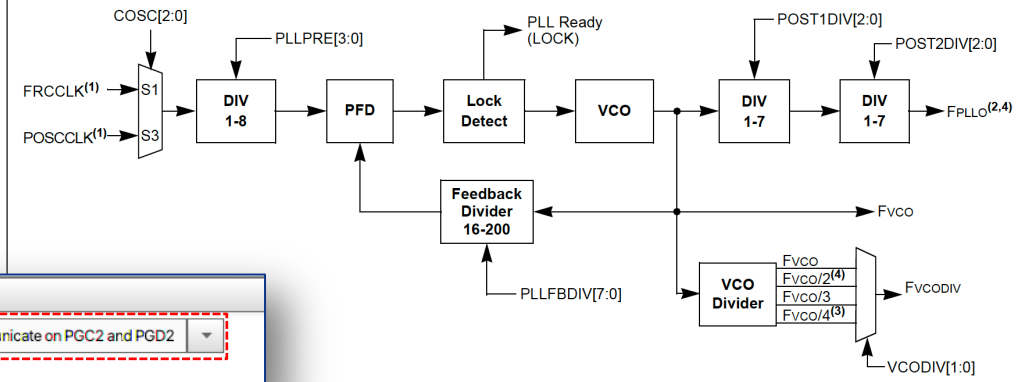
The Primary Oscillator and internal FRC Oscillator sources can optionally use an on-chip PLL to obtain higher operating speeds. There are two independent instantiations of PLL for the Master and Slave clock subsystems. Figure 6-4 illustrates a block diagram of the Master/Slave core PLL module.

For PLL operation, the following requirements must be met at all times without exception:

- The PLL Input Frequency (FPLLI) must be in the range of 8 MHz to 64 MHz
- The PFD Input Frequency (FPFD) must be in the range of 8 MHz to (FVCO/16) MHz

The VCO Output Frequency (FVCO) must be in the range of 400 MHz to 1600 MHz

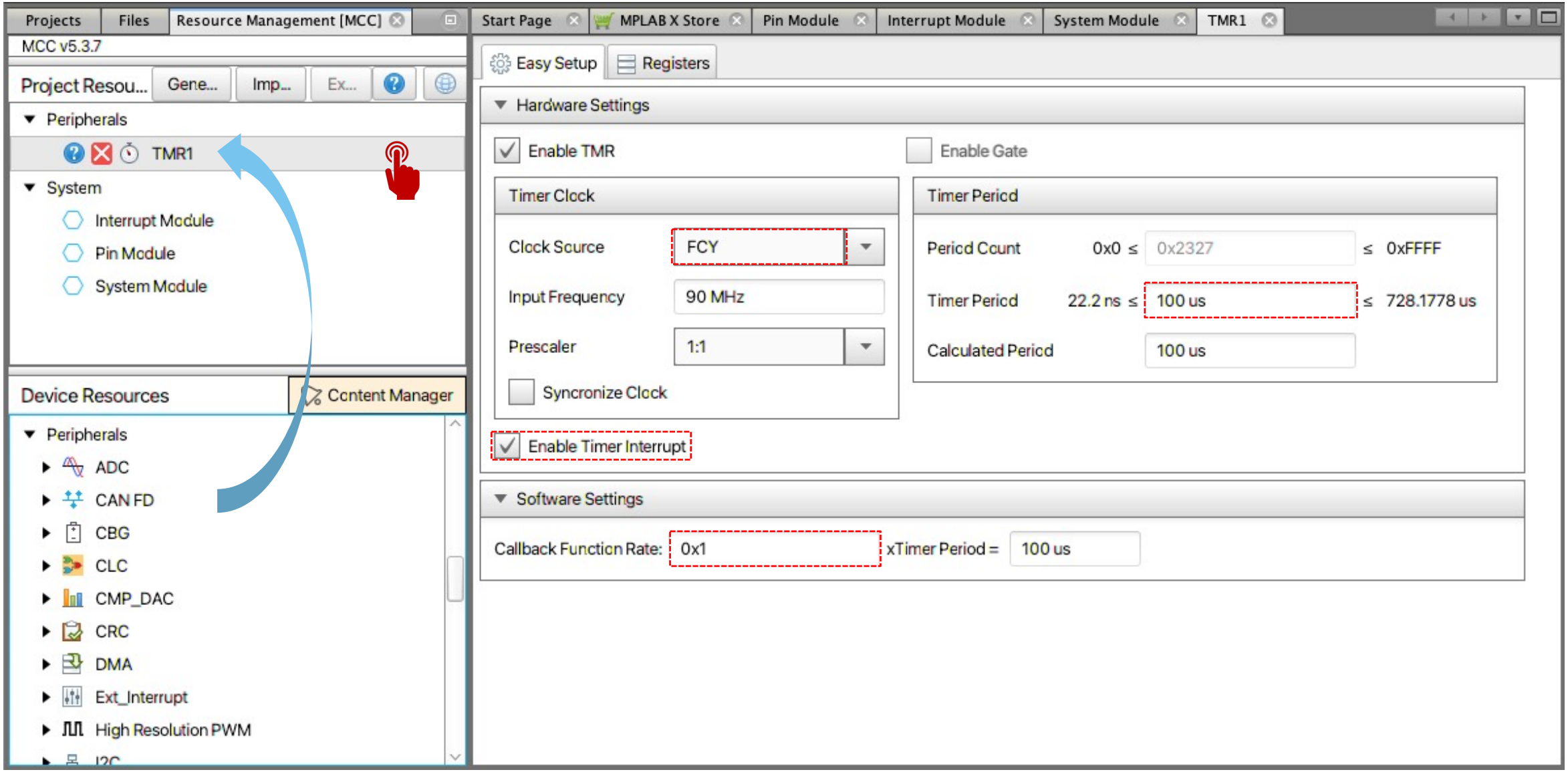
FIGURE 6-4: MASTER/SLAVE CORE PLL AND VCO DETAIL



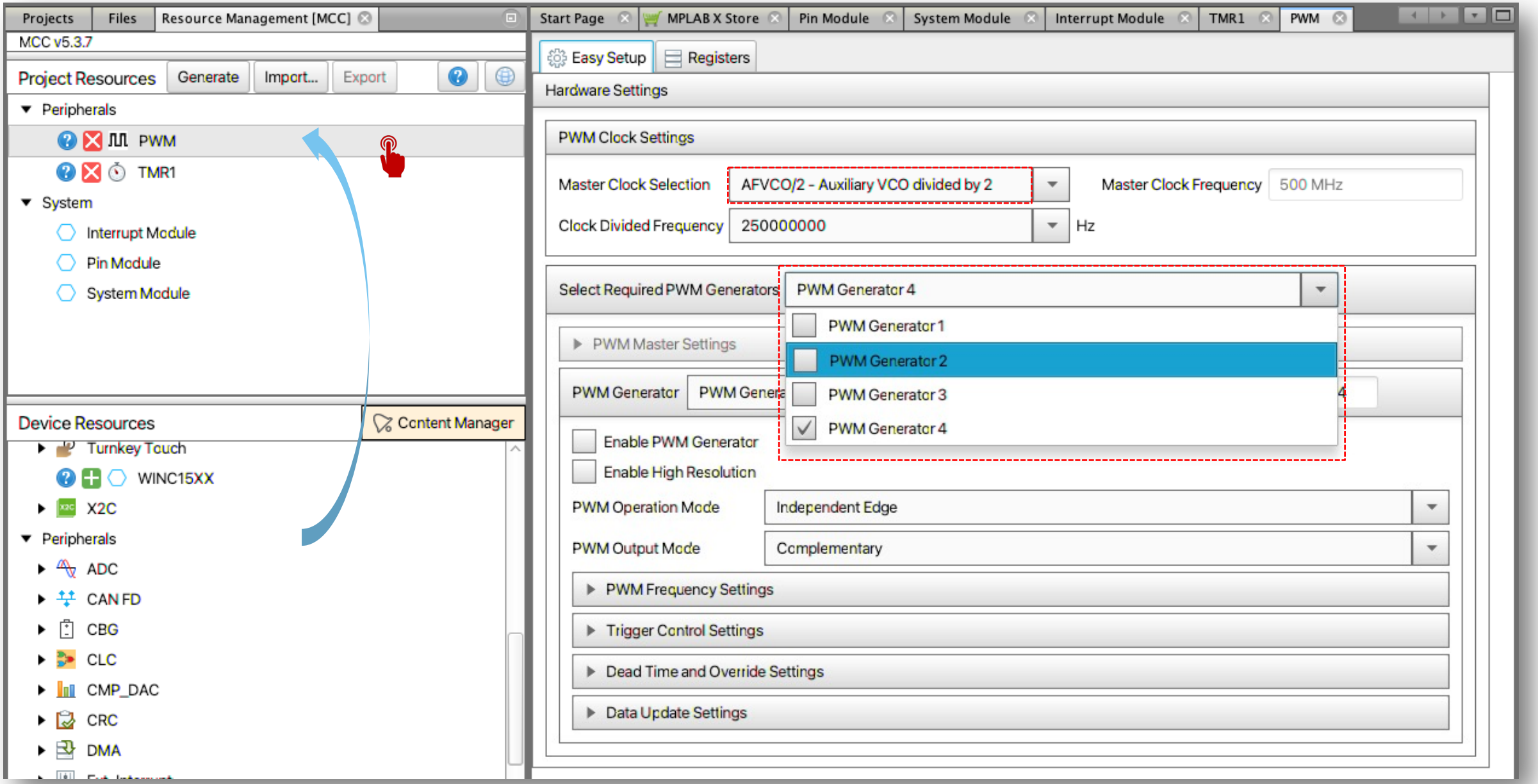
Note 1: From Master and Slave core shared oscillator source.
2: Clock option for PWM.
3: Clock option for ADC.
4: Clock option for DAC.

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

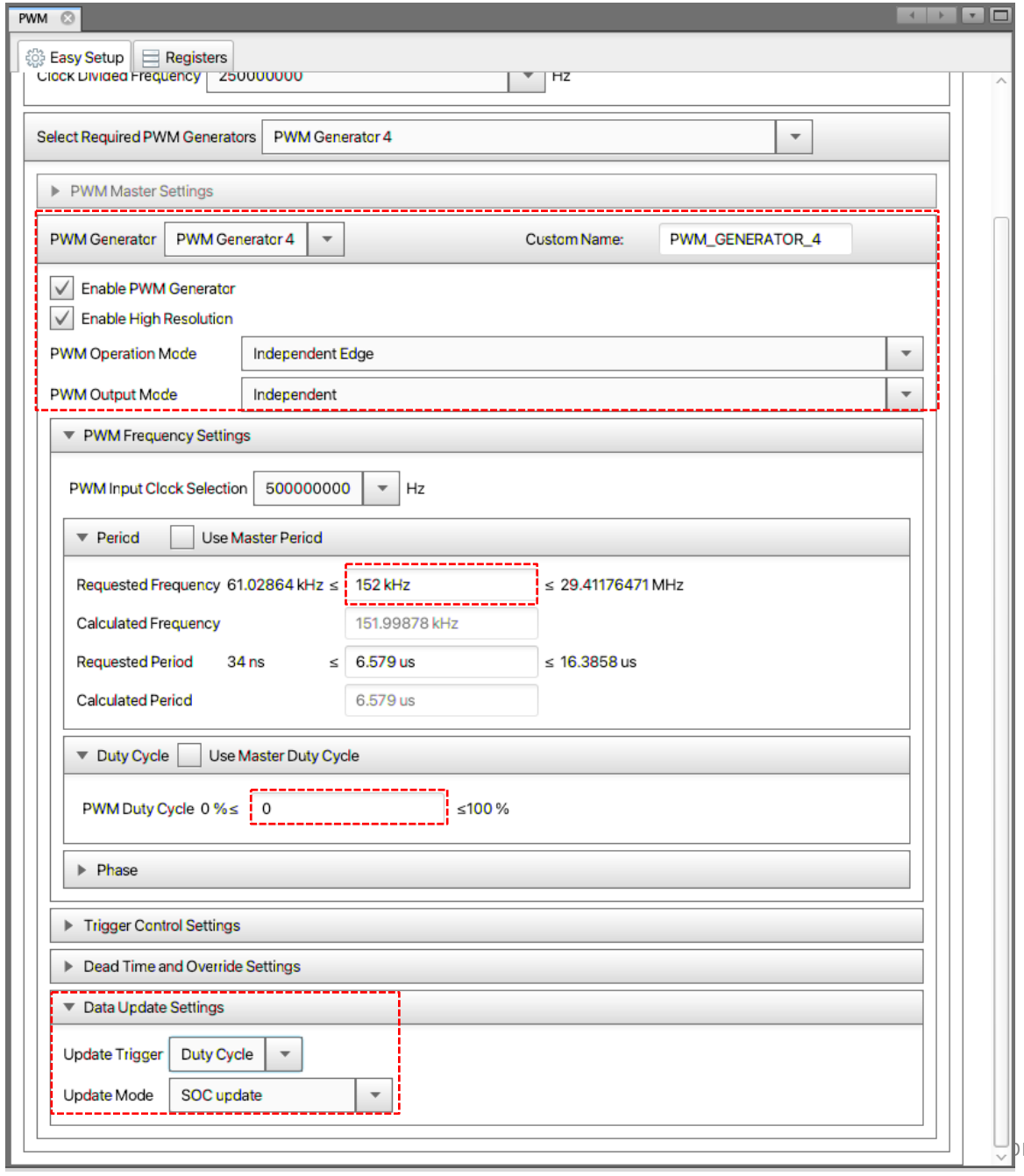
MCC Drivers – Timer1 (100 us)

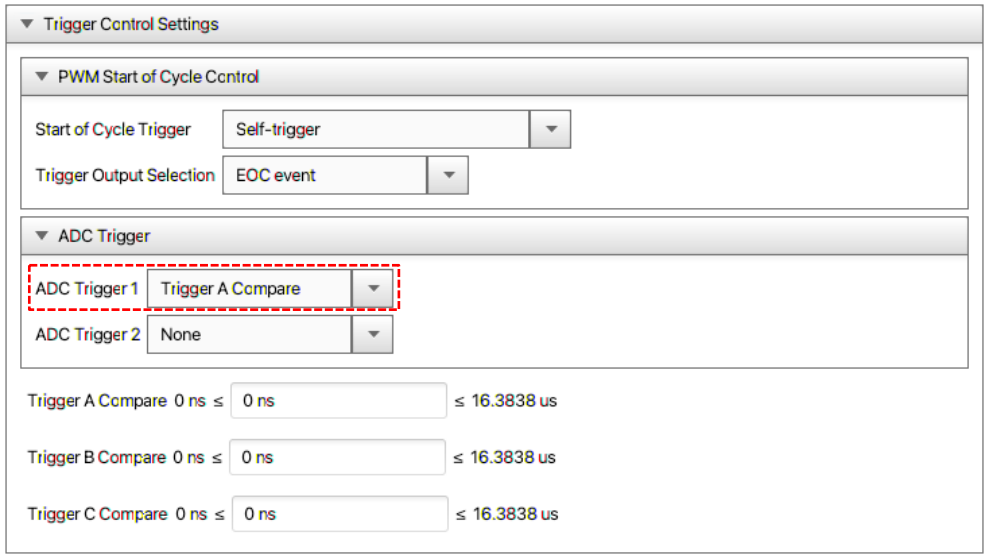


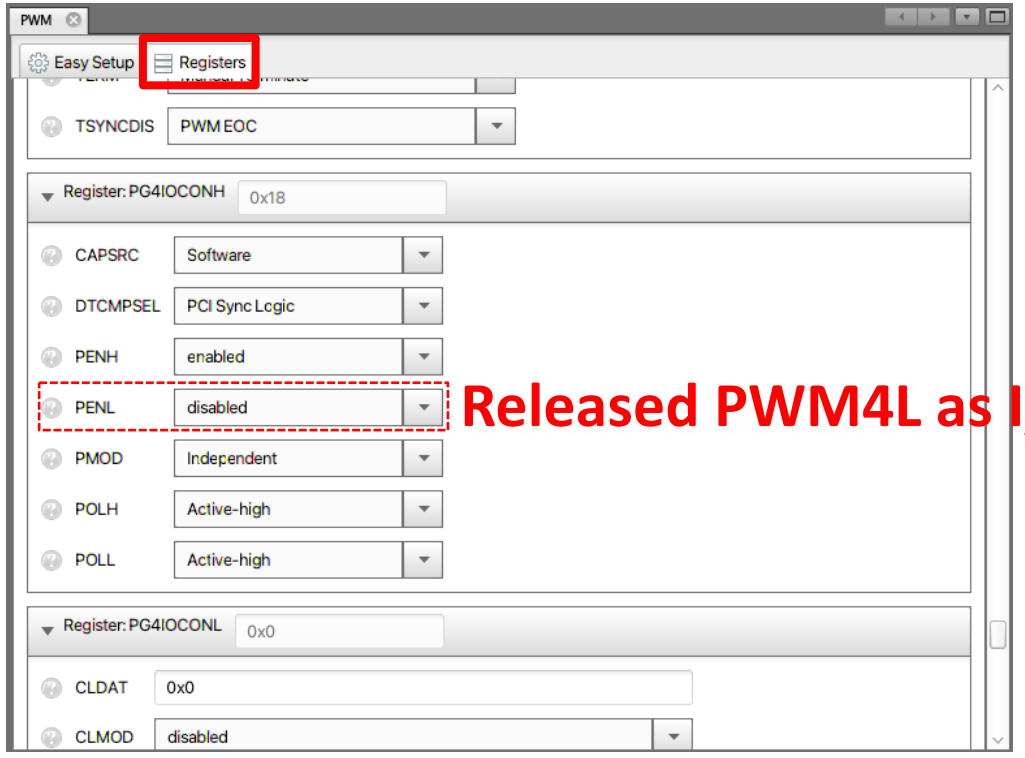
MCC Drivers – PWM4H – ~152KHz (Initial 0% Duty Cycle)



MCC Drivers – PWM4H – ~152KHz (Initial 0% Duty Cycle)

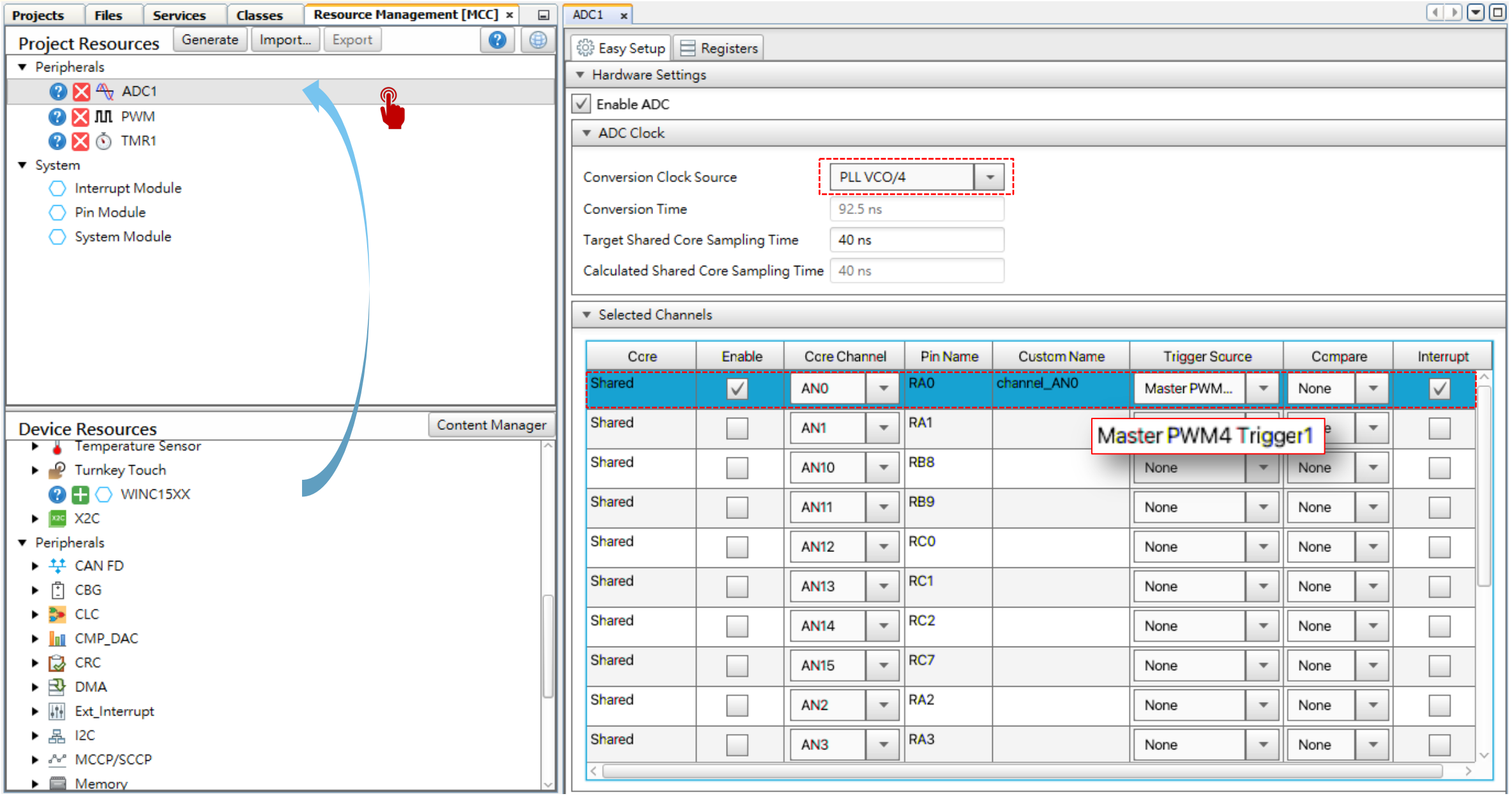




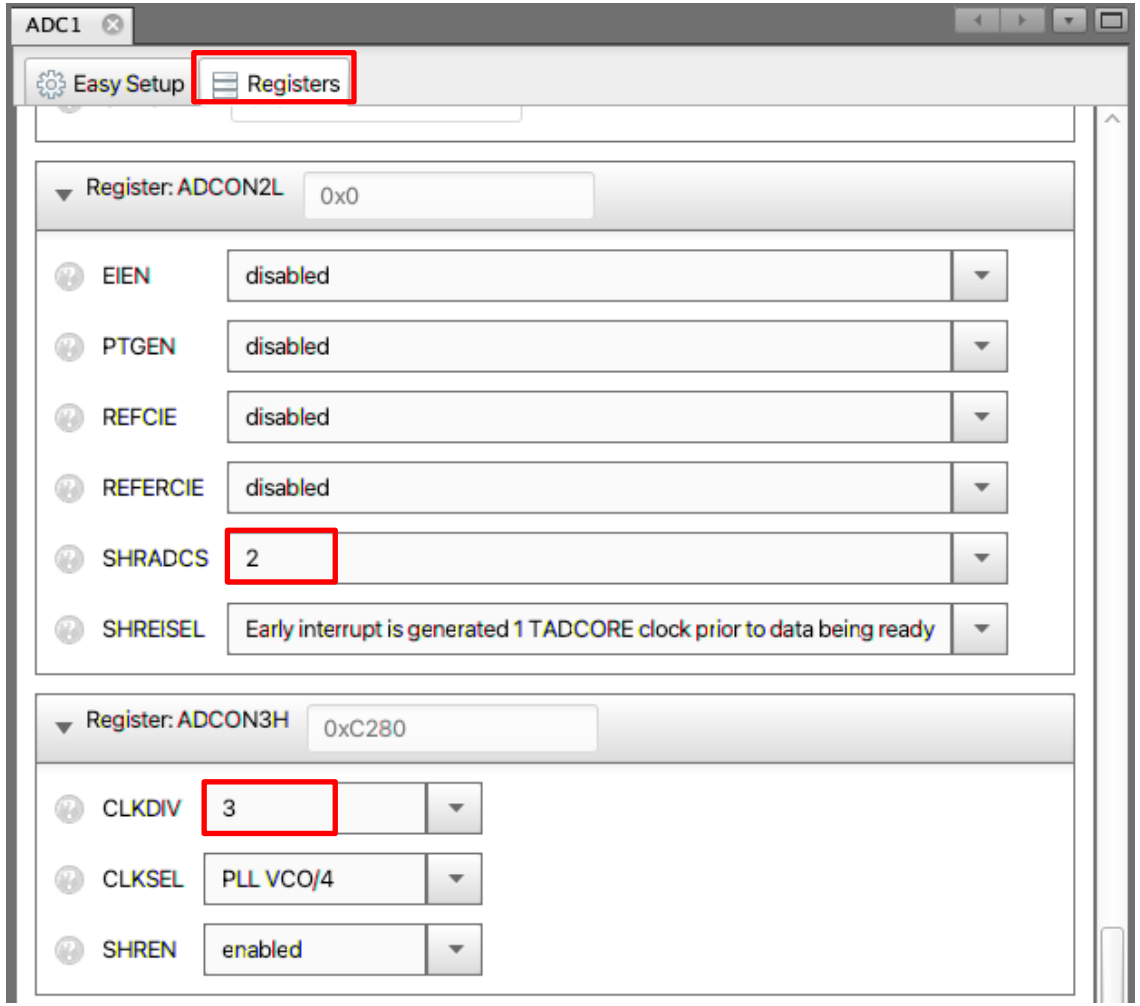


Released PWM4L as I/O

MCC Drivers – ADC/AN0 for POT

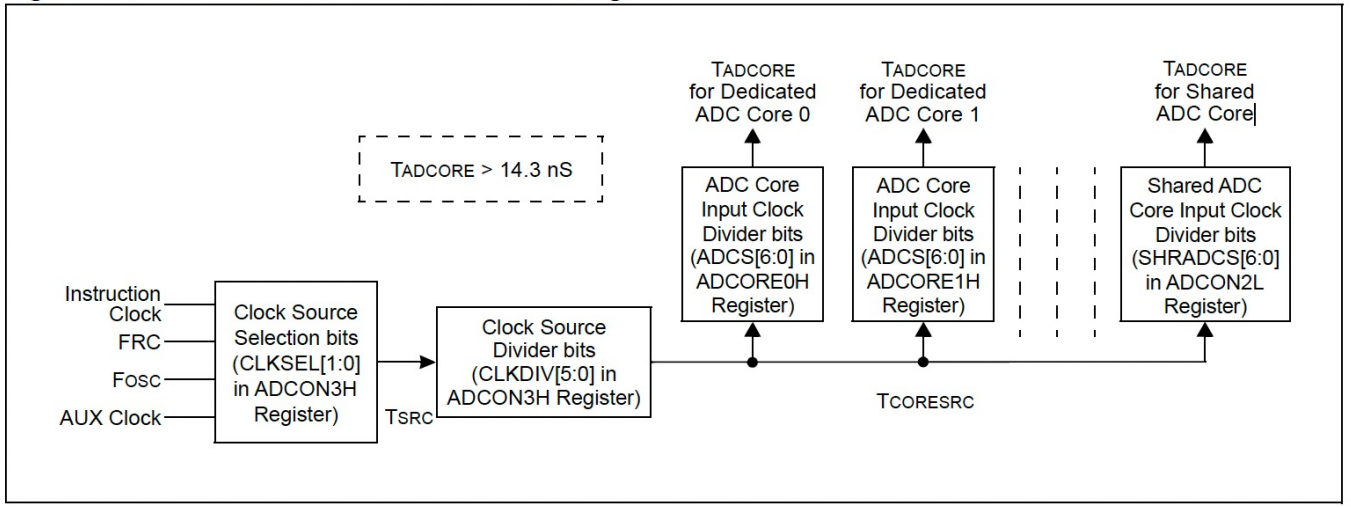


MCC Drivers – ADC/AN0 for POT

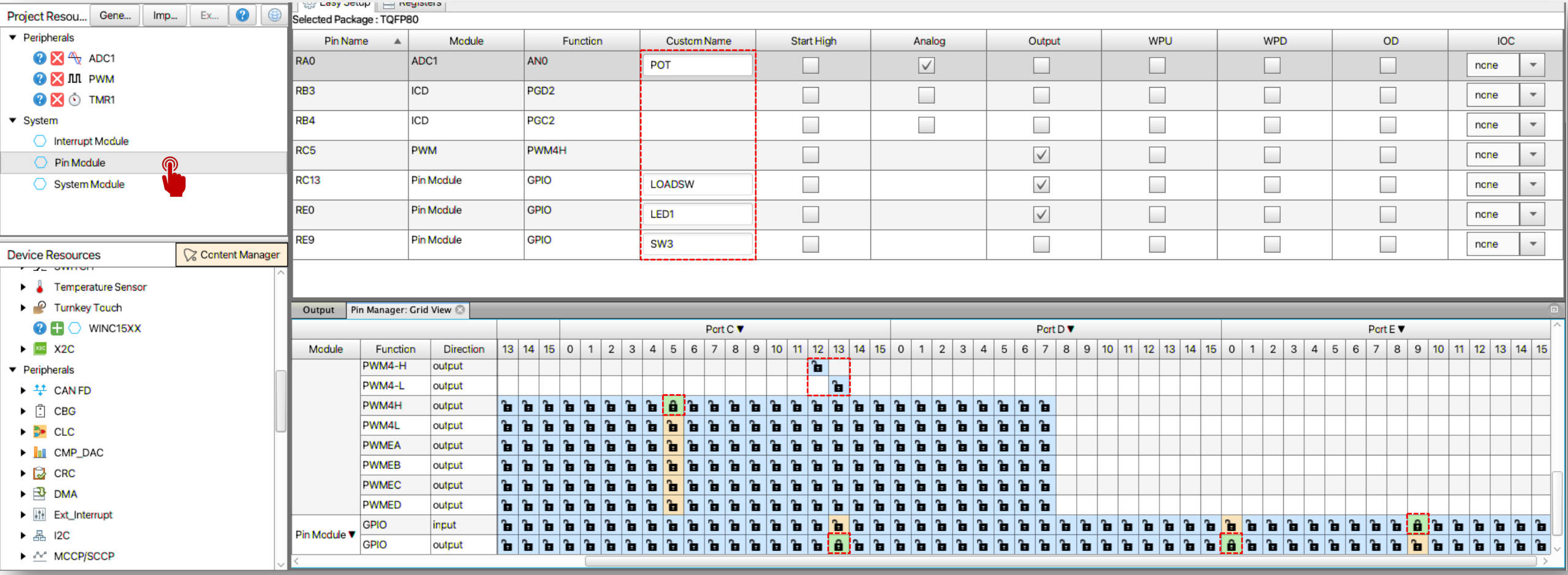


Master - ADC Shared Core			
ADC Module Clock Source	360	MHz	PLL VOC/4
ADCON2L.SHRADCS	2		Shared ADC Core Input Clock Divider bits
ADCON3H.CLKDIV	3		ADC Module Clock Source Divider bits
Fadcore	60	MHz	Clock Source / SHRADCS / CLKDIV
Tadcore	16.67	ns	1/Fadcore >= 14.3ns

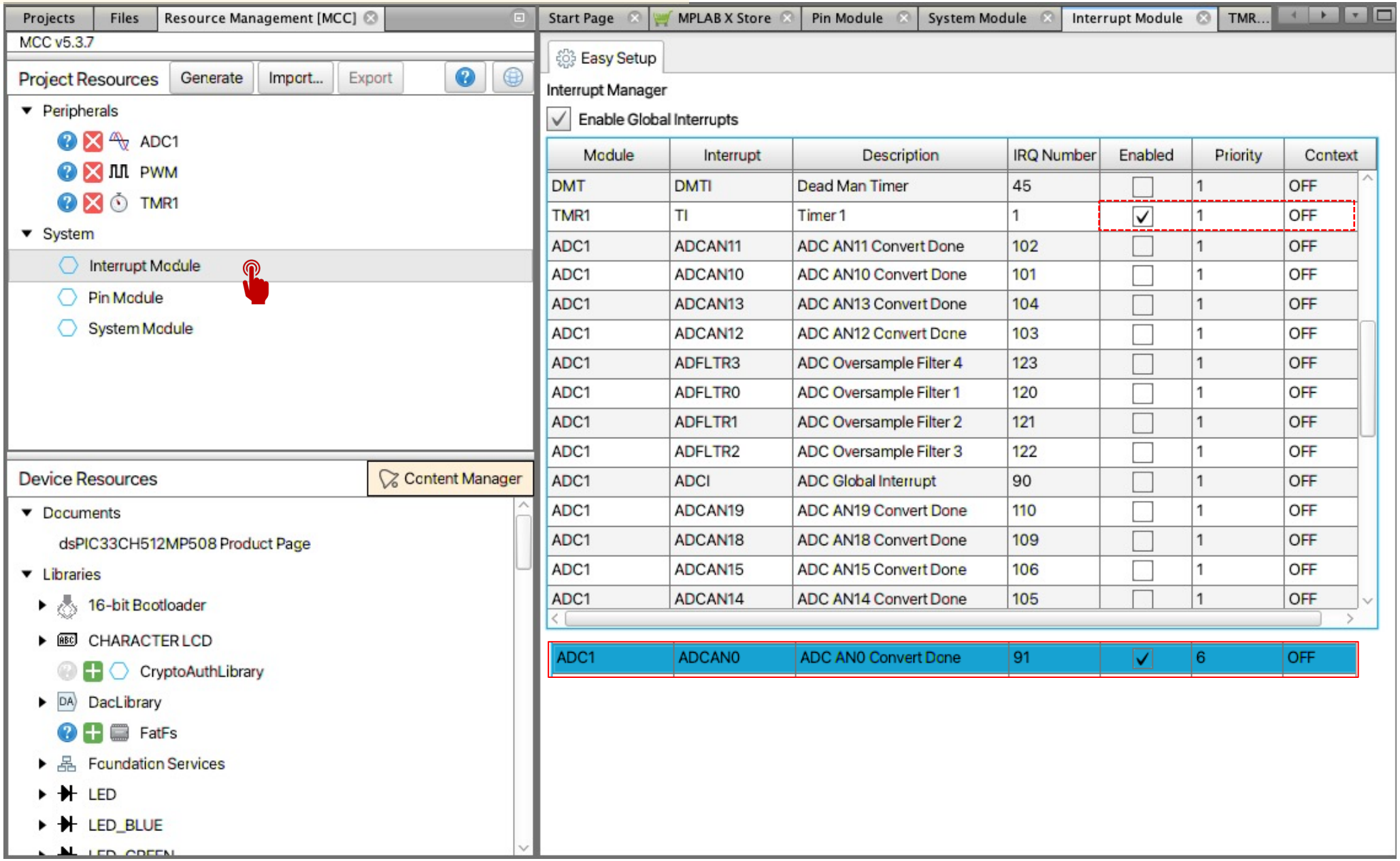
Figure 5-1: ADC Module Clock Path Block Diagram



MCC Drivers – Pin Management

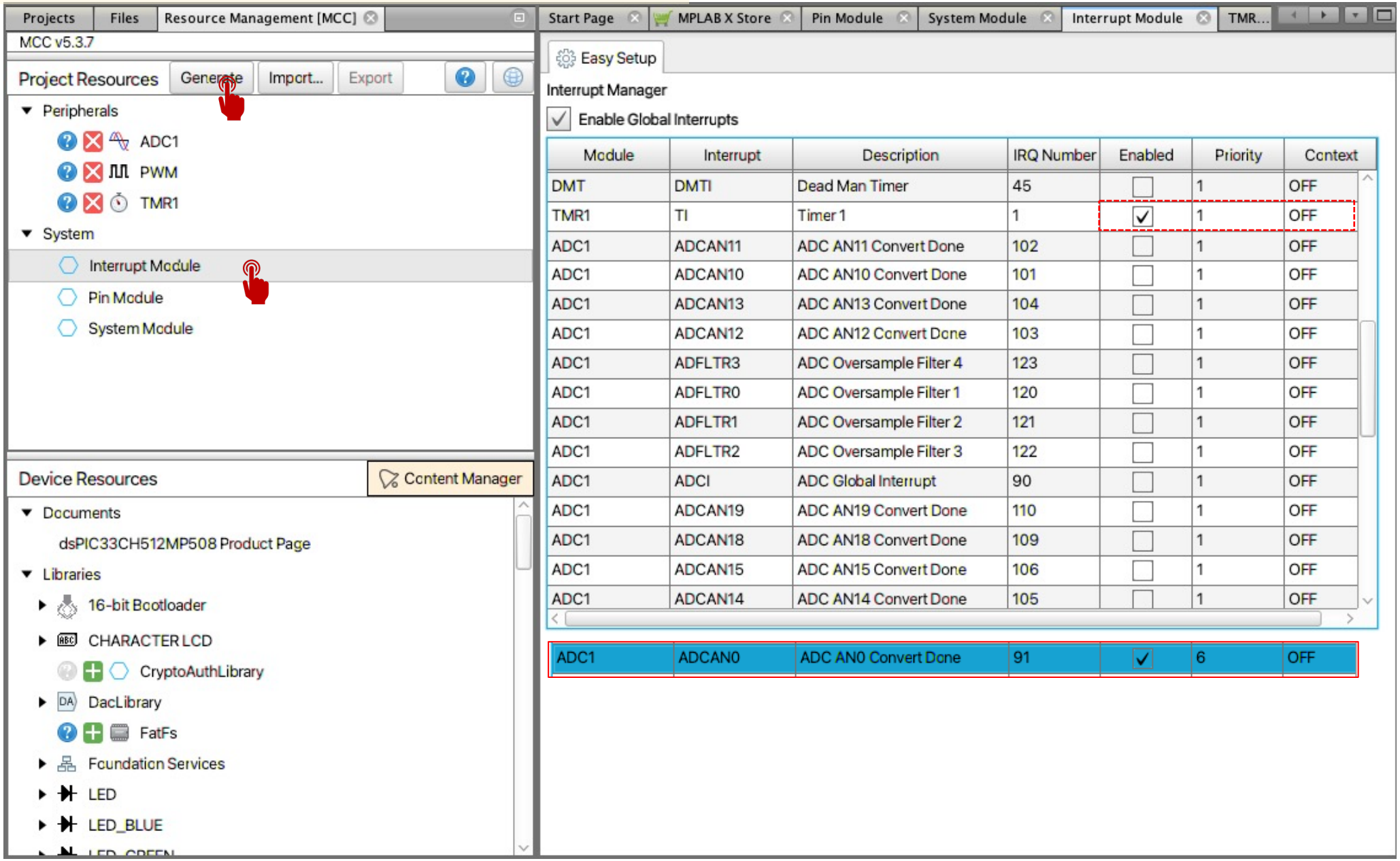


MCC Drivers – Interrupt Management & Code Generating



Module	Interrupt	Description	IRQ Number	Enabled	Priority	Context
DMT	DMTI	Dead Man Timer	45	☐	1	OFF
TMR1	TI	Timer 1	1	☑	1	OFF
ADC1	ADCAN11	ADC AN11 Convert Done	102	☐	1	OFF
ADC1	ADCAN10	ADC AN10 Convert Done	101	☐	1	OFF
ADC1	ADCAN13	ADC AN13 Convert Done	104	☐	1	OFF
ADC1	ADCAN12	ADC AN12 Convert Done	103	☐	1	OFF
ADC1	ADFLTR3	ADC Oversample Filter 4	123	☐	1	OFF
ADC1	ADFLTR0	ADC Oversample Filter 1	120	☐	1	OFF
ADC1	ADFLTR1	ADC Oversample Filter 2	121	☐	1	OFF
ADC1	ADFLTR2	ADC Oversample Filter 3	122	☐	1	OFF
ADC1	ADCI	ADC Global Interrupt	90	☐	1	OFF
ADC1	ADCAN19	ADC AN19 Convert Done	110	☐	1	OFF
ADC1	ADCAN18	ADC AN18 Convert Done	109	☐	1	OFF
ADC1	ADCAN15	ADC AN15 Convert Done	106	☐	1	OFF
ADC1	ADCAN14	ADC AN14 Convert Done	105	☐	1	OFF
ADC1	ADCAN0	ADC AN0 Convert Done	91	☑	6	OFF

MCC Drivers – Interrupt Management & Code Generating



Module	Interrupt	Description	IRQ Number	Enabled	Priority	Context
DMT	DMTI	Dead Man Timer	45	☐	1	OFF
TMR1	TI	Timer 1	1	☑	1	OFF
ADC1	ADCAN11	ADC AN11 Convert Done	102	☐	1	OFF
ADC1	ADCAN10	ADC AN10 Convert Done	101	☐	1	OFF
ADC1	ADCAN13	ADC AN13 Convert Done	104	☐	1	OFF
ADC1	ADCAN12	ADC AN12 Convert Done	103	☐	1	OFF
ADC1	ADFLTR3	ADC Oversample Filter 4	123	☐	1	OFF
ADC1	ADFLTR0	ADC Oversample Filter 1	120	☐	1	OFF
ADC1	ADFLTR1	ADC Oversample Filter 2	121	☐	1	OFF
ADC1	ADFLTR2	ADC Oversample Filter 3	122	☐	1	OFF
ADC1	ADCI	ADC Global Interrupt	90	☐	1	OFF
ADC1	ADCAN19	ADC AN19 Convert Done	110	☐	1	OFF
ADC1	ADCAN18	ADC AN18 Convert Done	109	☐	1	OFF
ADC1	ADCAN15	ADC AN15 Convert Done	106	☐	1	OFF
ADC1	ADCAN14	ADC AN14 Convert Done	105	☐	1	OFF
ADC1	ADCAN0	ADC AN0 Convert Done	91	☑	6	OFF

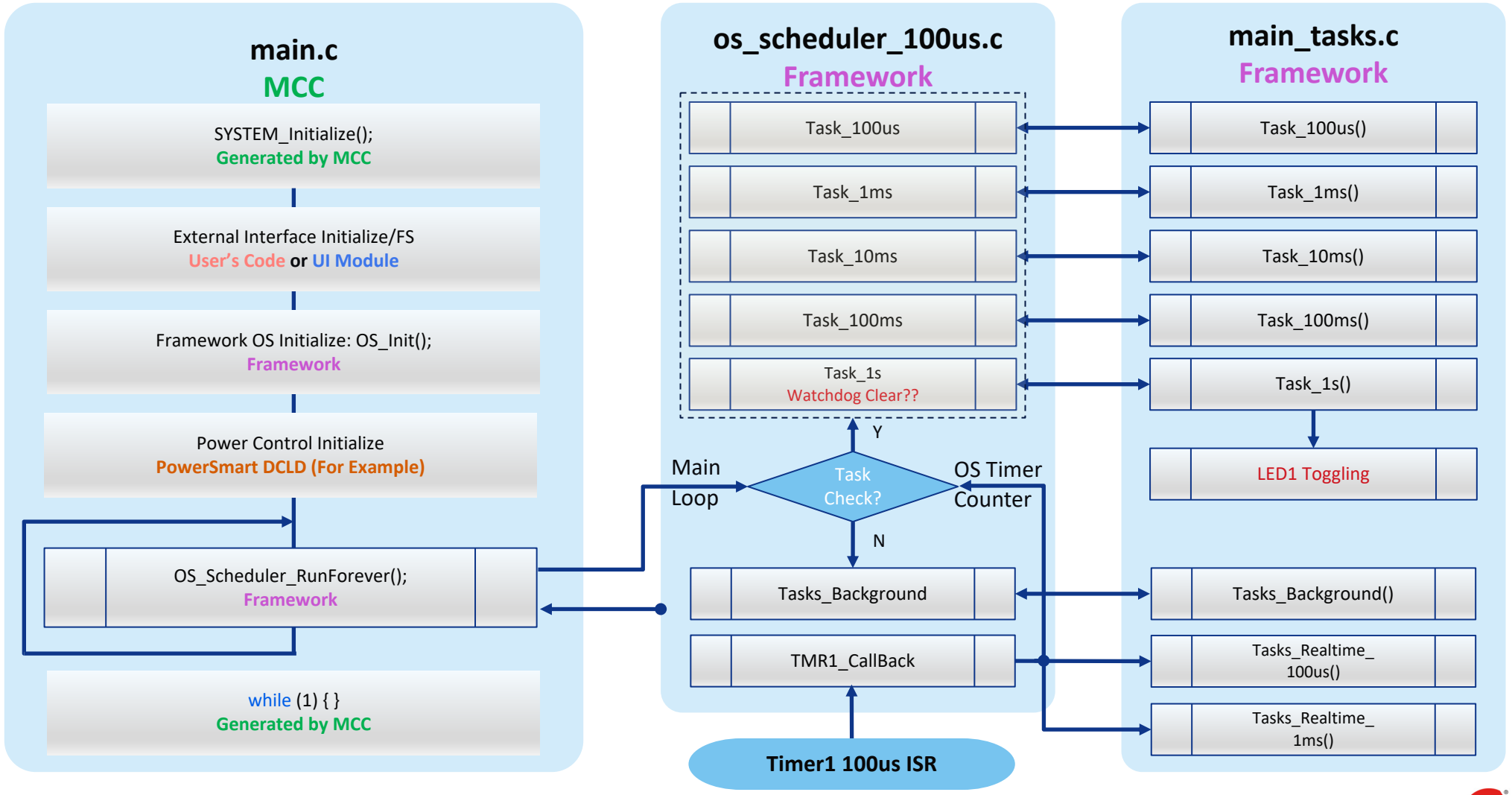
Basic MCC Codebase

```
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

/**
  Section: Included Files
*/
#include "mcc_generated_files/system.h"

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

```
17:00:17.886    INFO: mcc_generated_files/reset.h                Success. New file.
17:00:17.886    INFO: mcc_generated_files/reset_types.h          Success. New file.
17:00:17.886    INFO: mcc_generated_files/system.c               Success. New file.
17:00:17.886    INFO: mcc_generated_files/system.h               Success. New file.
17:00:17.886    INFO: mcc_generated_files/system_types.h         Success. New file.
17:00:17.886    INFO: mcc_generated_files/tmr1.c                 Success. New file.
17:00:17.886    INFO: mcc_generated_files/tmr1.h                 Success. New file.
17:00:17.887    INFO: mcc_generated_files/traps.c                Success. New file.
17:00:17.887    INFO: mcc_generated_files/traps.h                Success. New file.
17:00:17.887    INFO: mcc_generated_files/watchdog.h             Success. New file.
17:00:17.919    INFO: ************************************************
17:00:17.919    INFO:  Generation complete (total time: 718 milliseconds)
17:00:17.919    INFO: ************************************************
17:00:17.919    INFO: Generation complete.
```

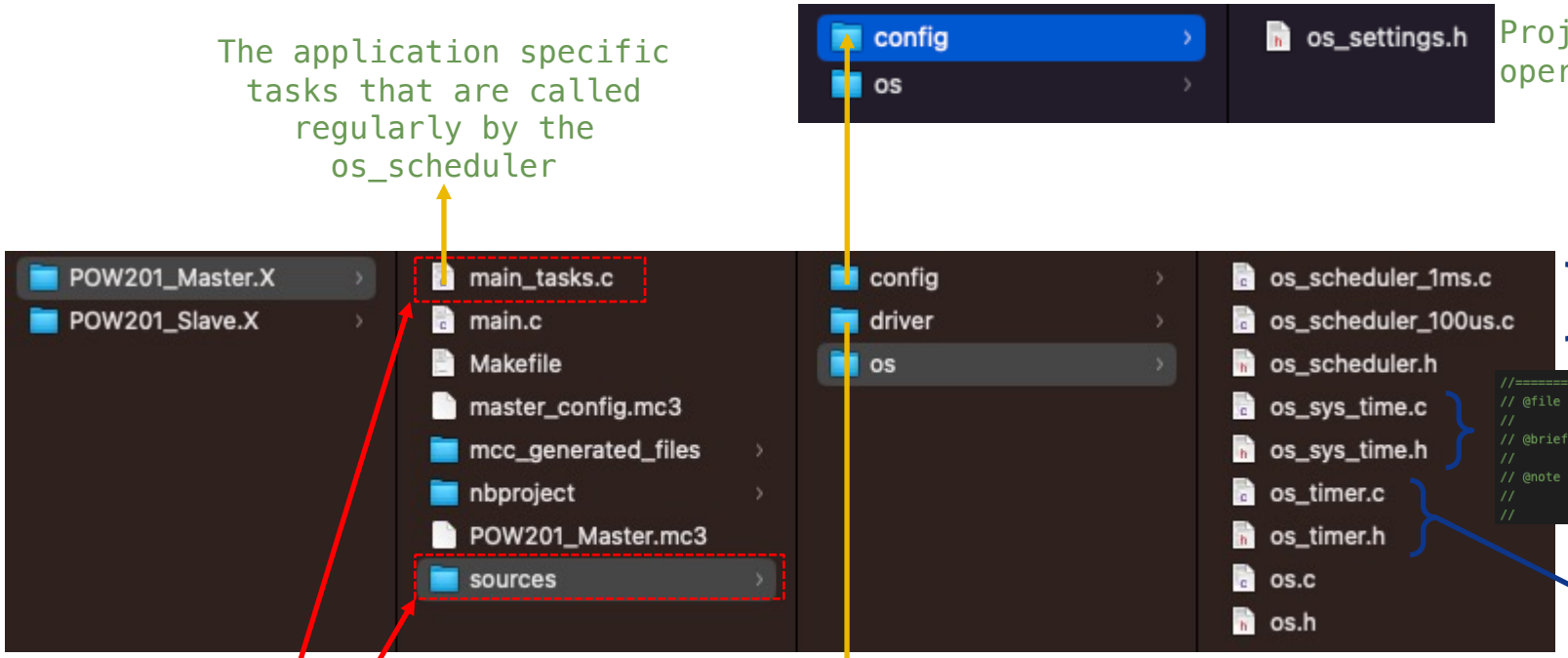
The SMPS Firmware Framework

Firmware Structure



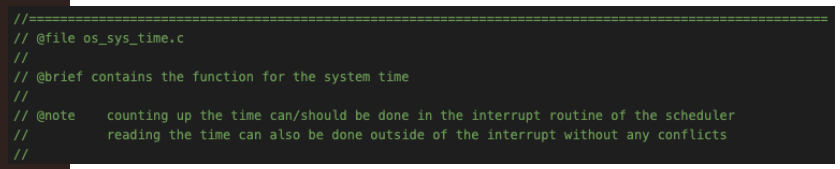
OS Scheduler for Master

The application specific tasks that are called regularly by the os_scheduler

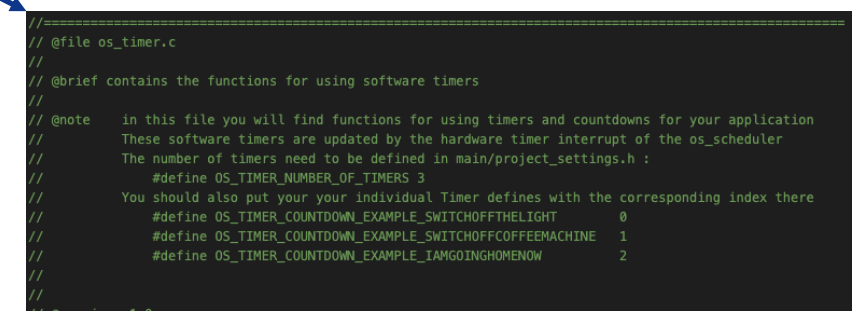


Project specific settings regarding the operating system parts and application

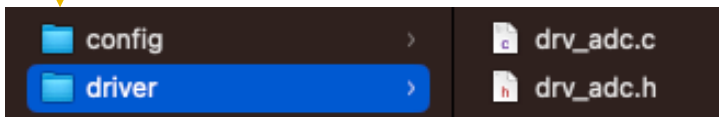
1 ms for low-end MCU such like PIC16
100 us for high speed MCU such like dsPIC33



```
//==========================================================================================
// @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
//
```



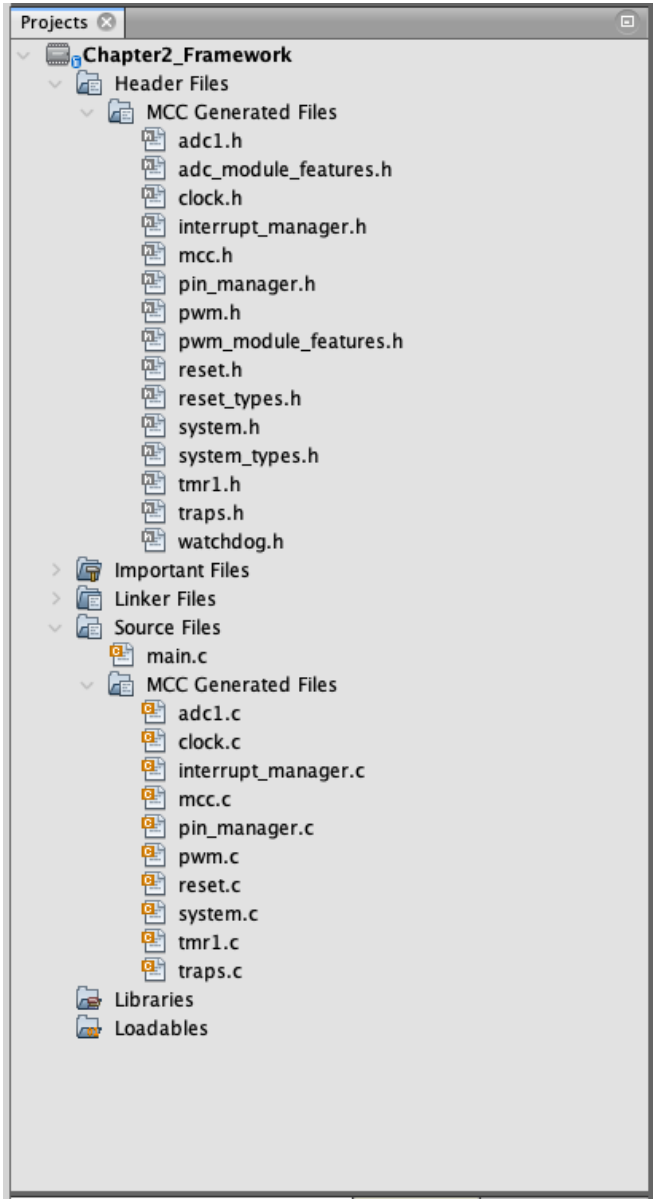
```
//==========================================================================================
// @file os_timer.c
//
// @brief contains the functions for using software timers
//
// @note    in this file you will find functions for using timers and countdowns for your application
//          These 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
//
//
```



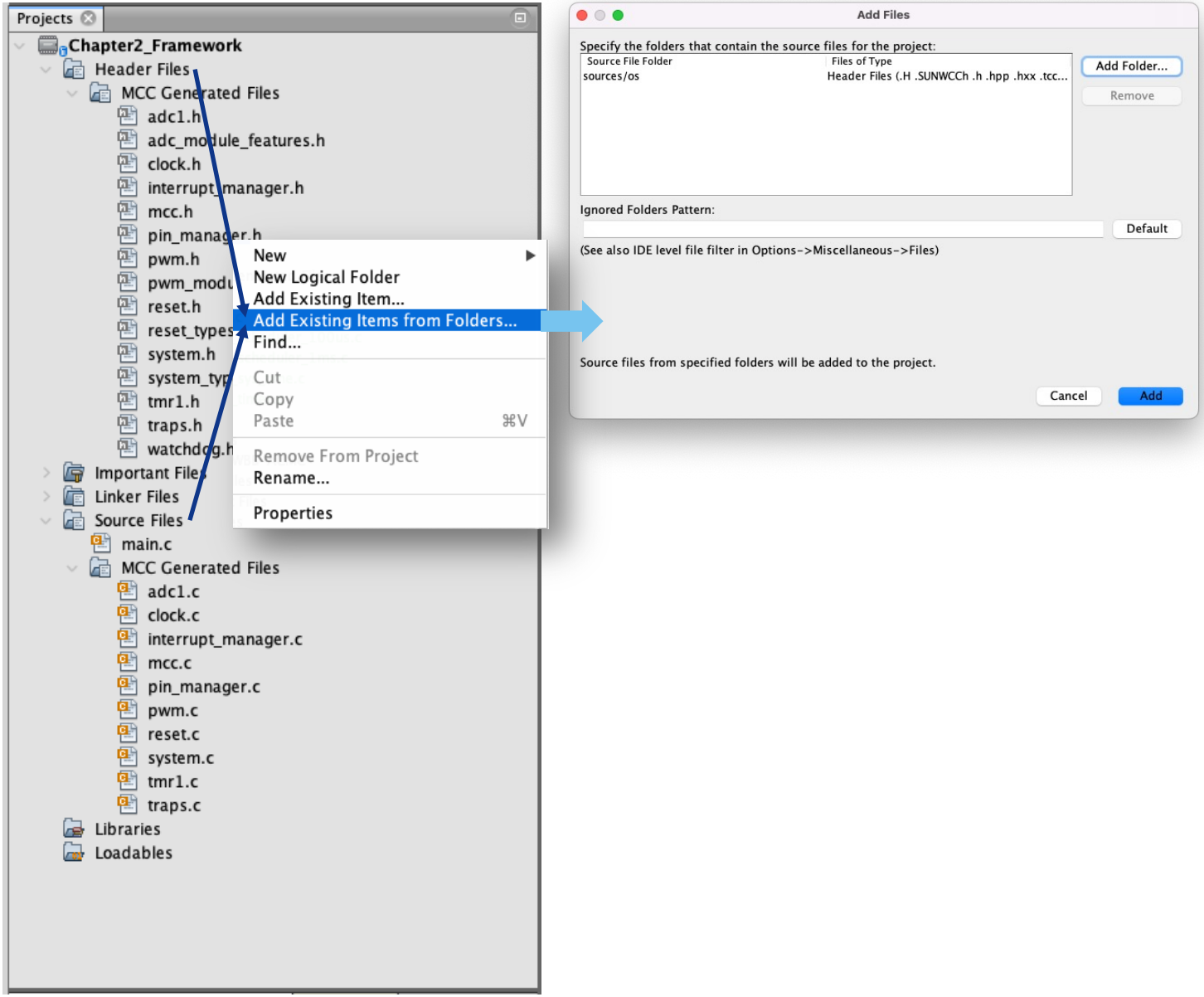
Additional driver code example for ADC

此檔案與目錄皆可在Lab Answer中對應的Lab目錄中找到。

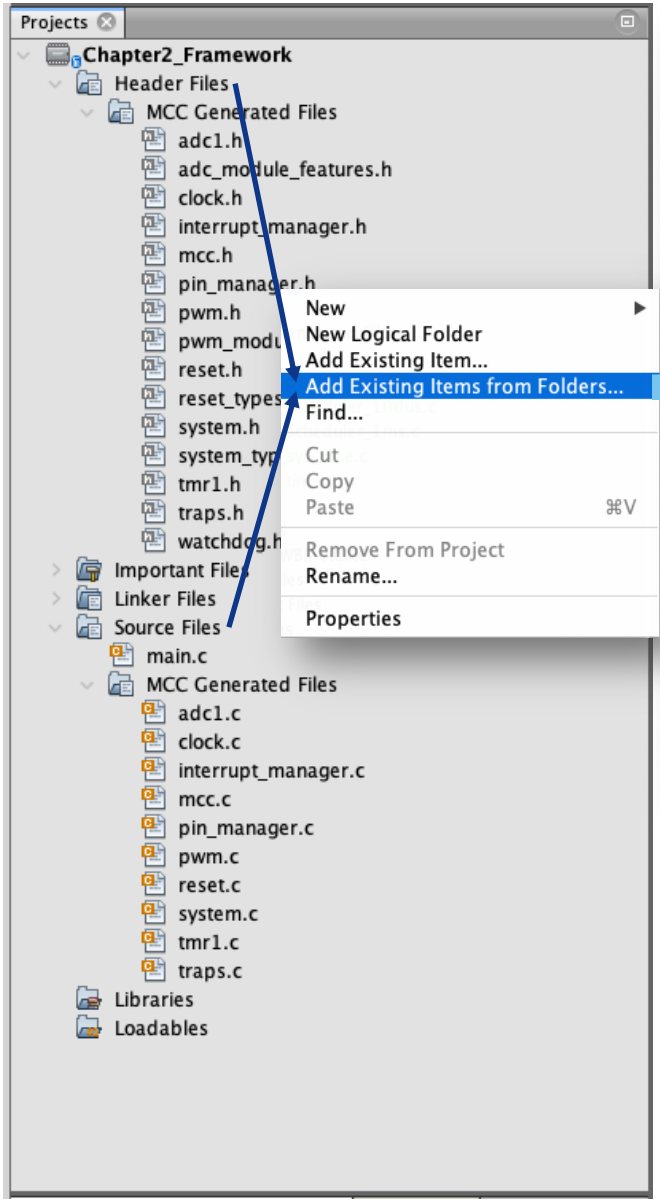
Import OS_Scheduler

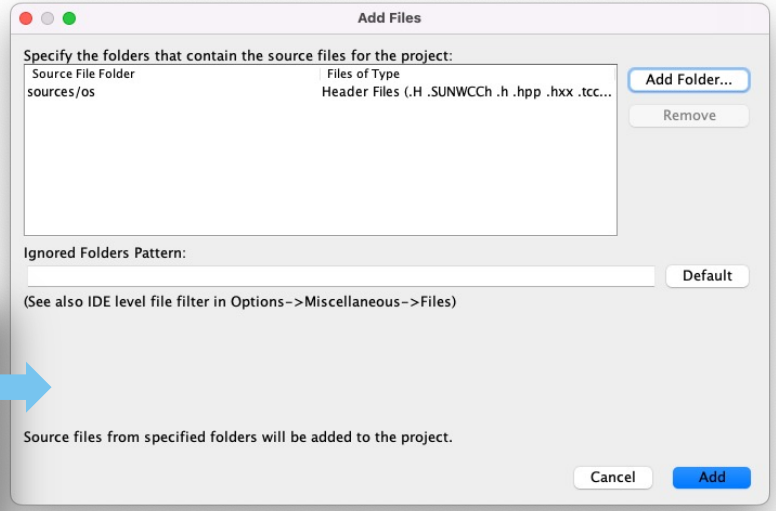


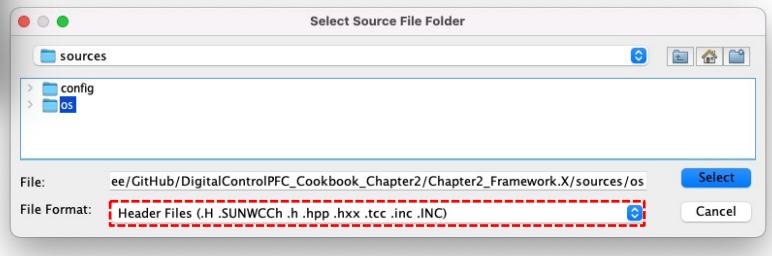
Import OS_Scheduler



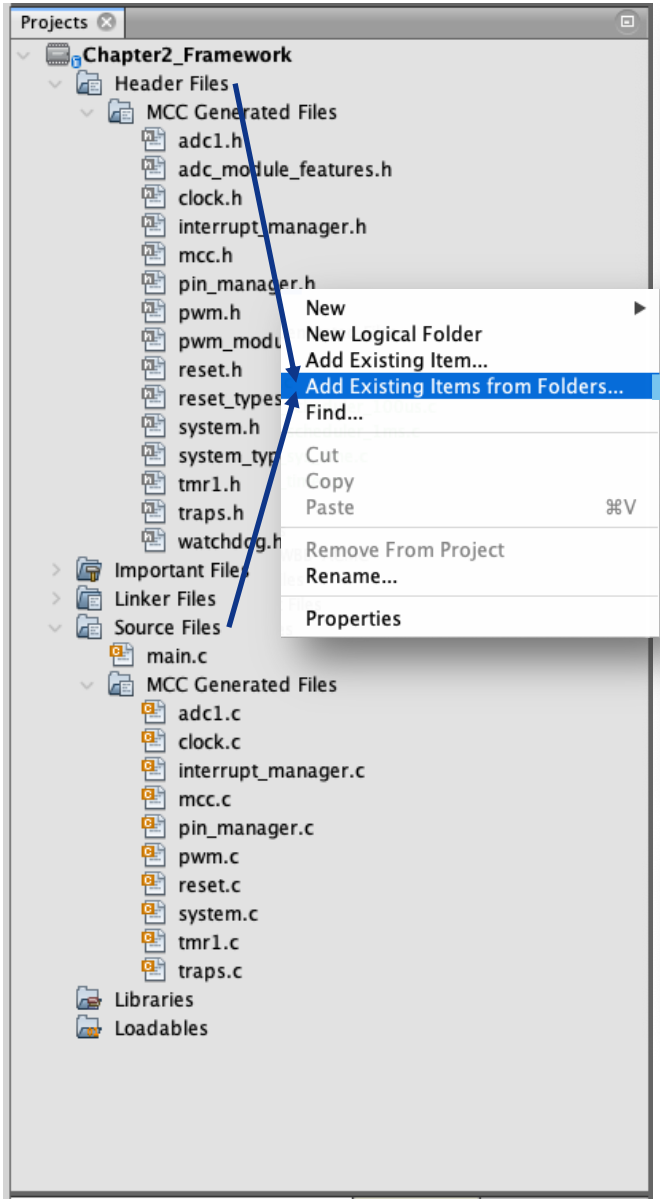
Import OS_Scheduler

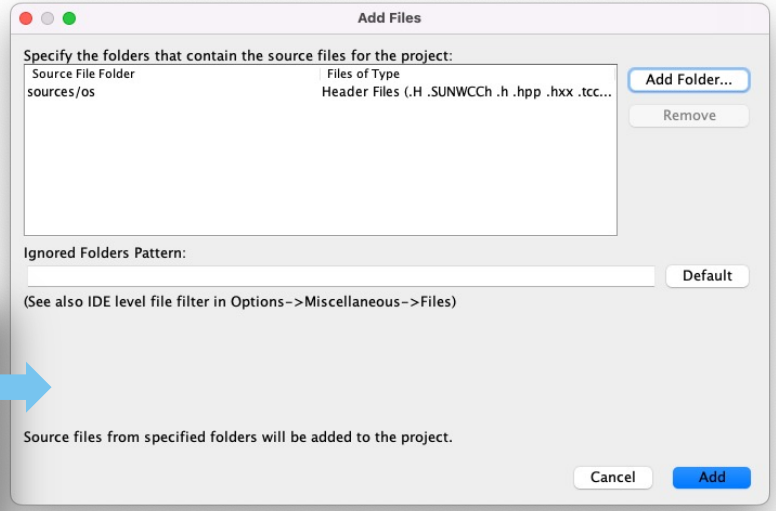


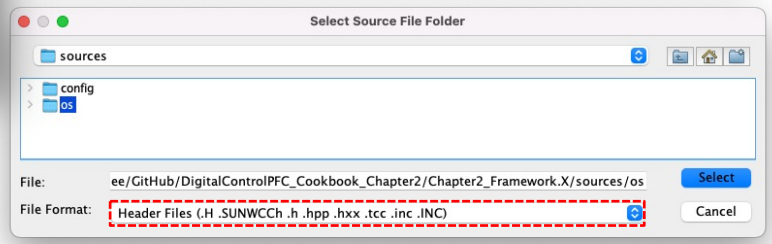


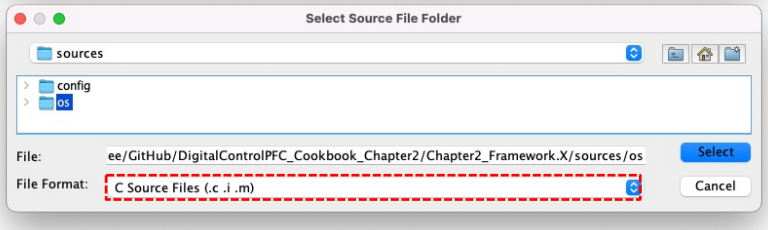


Import OS_Scheduler

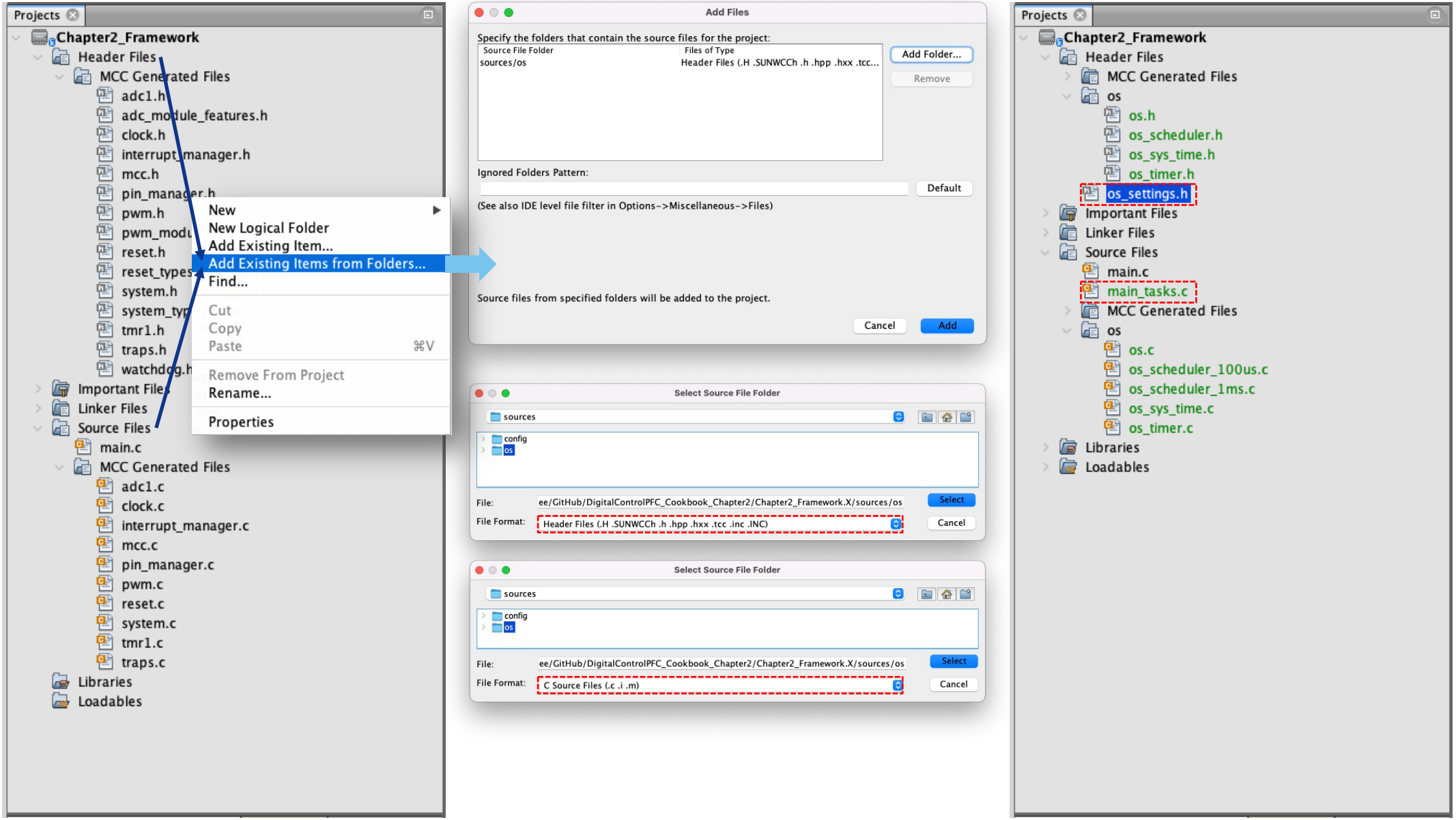








Import OS_Scheduler



Main & Tasks

main.c

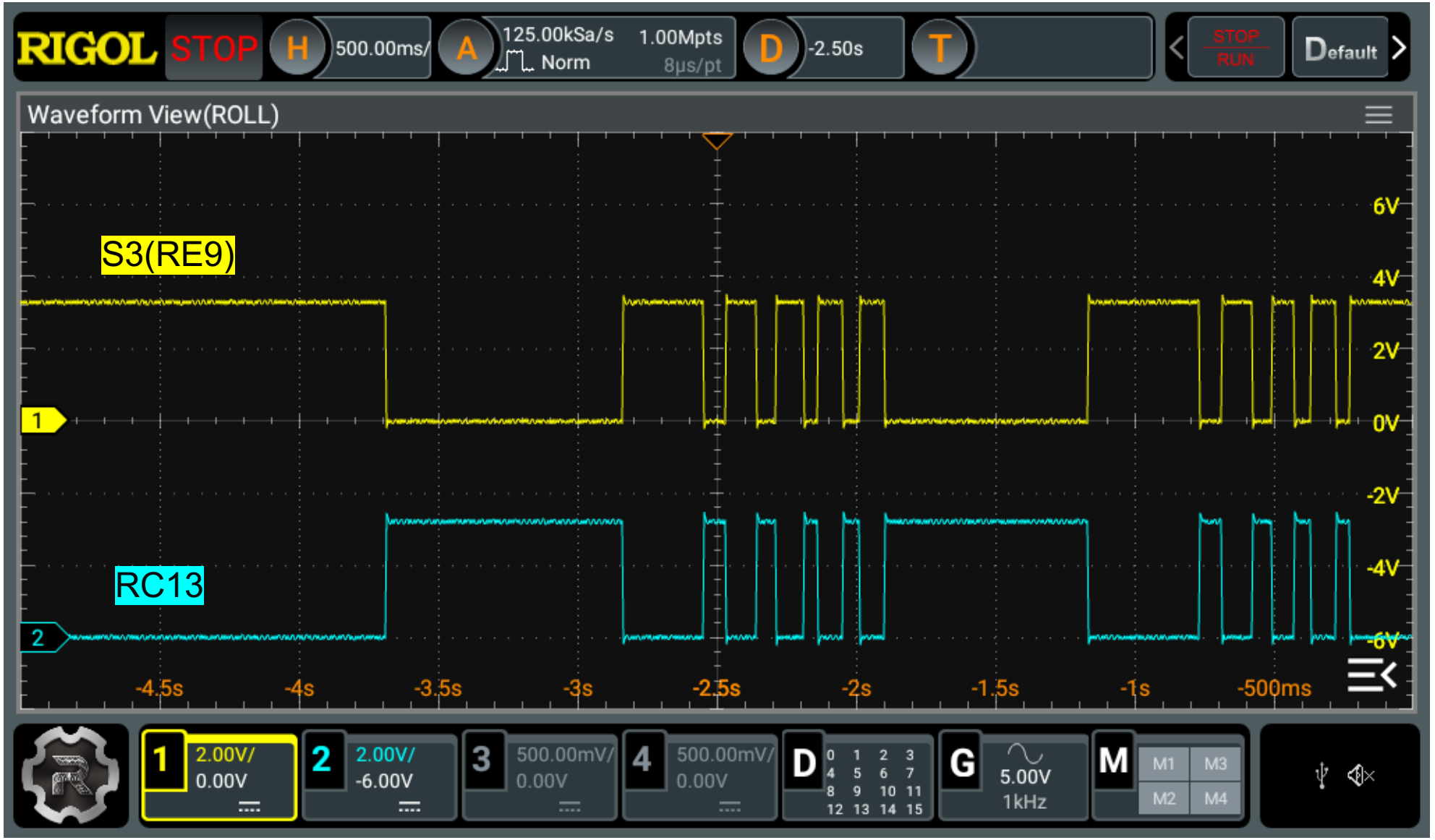
```
47  */
48  #include "mcc_generated_files/system.h"
49  #include "sources/os/os.h"
50
51  /*
52                          Main application
53   */
54  int main(void)
55  {
56      // initialize the device
57      SYSTEM_Initialize();
58
59      // OS
60      OS_Init();
61      OS_Scheduler_RunForever();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

main_tasks.c

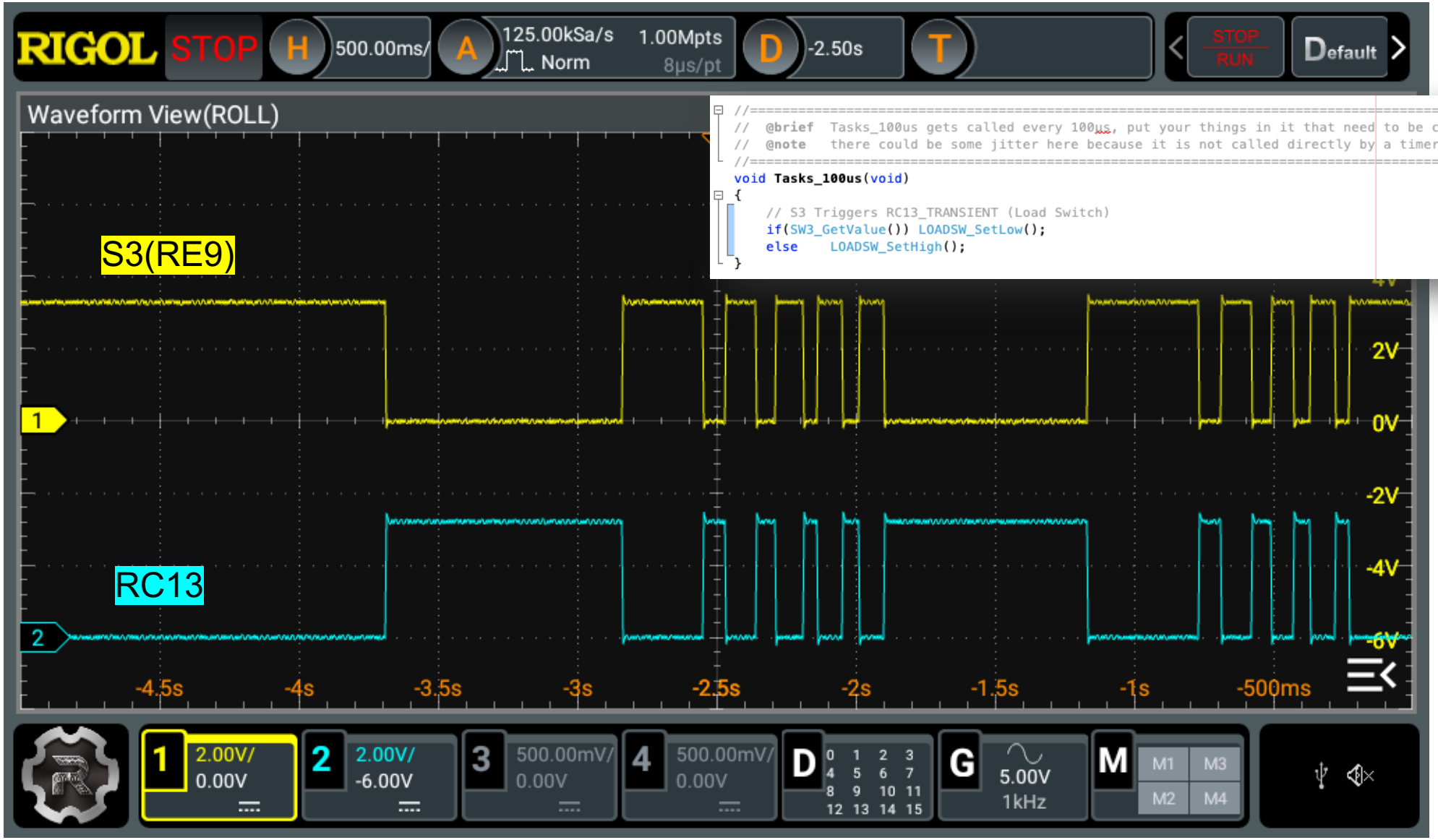
```
130 //  @note   there could be some jitter here because it is not called directly by a tim
131 //=========================================================================
132
133 void Tasks_1s(void)
134 {
135     // put your application specific code here that needs to be called every second
136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

0.5 Hz LED1 Toggle

S3(RE9) Triggers RC13_TRANSIENT



S3(RE9) Triggers RC13_TRANSIENT




```
//=====================================================================================
//  @brief  Tasks_100us gets called every 100µs, put your things in it that need to be called regularly
//  @note   there could be some jitter here because it is not called directly by a timer interrupt
//=====================================================================================
void Tasks_100us(void)
{
    // S3 Triggers RC13_TRANSIENT (Load Switch)
    if(SW3_GetValue()) LOADSW_SetLow();
    else    LOADSW_SetHigh();
}
```

AN0-POT Sets RC5_Master PWM4H

AN0-POT Sets RC5_Master PWM4H

drv_adc.c

Source History

```c
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include "../config/os_settings.h"

#include "drv_adc.h"
#include "../../mcc_generated_files/pwm.h"

void __attribute__ ( ( __interrupt__ , auto_psv) ) _ADCAN0Interrupt ( void )
{
    uint16_t valPOT;
    uint16_t PWMLimit;
    //Read the ADC value from the ADCBUF
    valPOT = ADCBUF0<<3;

    //Limit with Max 50% DC
    if(valPOT > (PG4PER >> 1)) PWMLimit = PG4PER >>1;
    else PWMLimit = valPOT;

    PWM_DutyCycleSet(4, PWMLimit);

    //clear the POT interrupt flag
    IFS5bits.ADCAN0IF = 0;
}
```

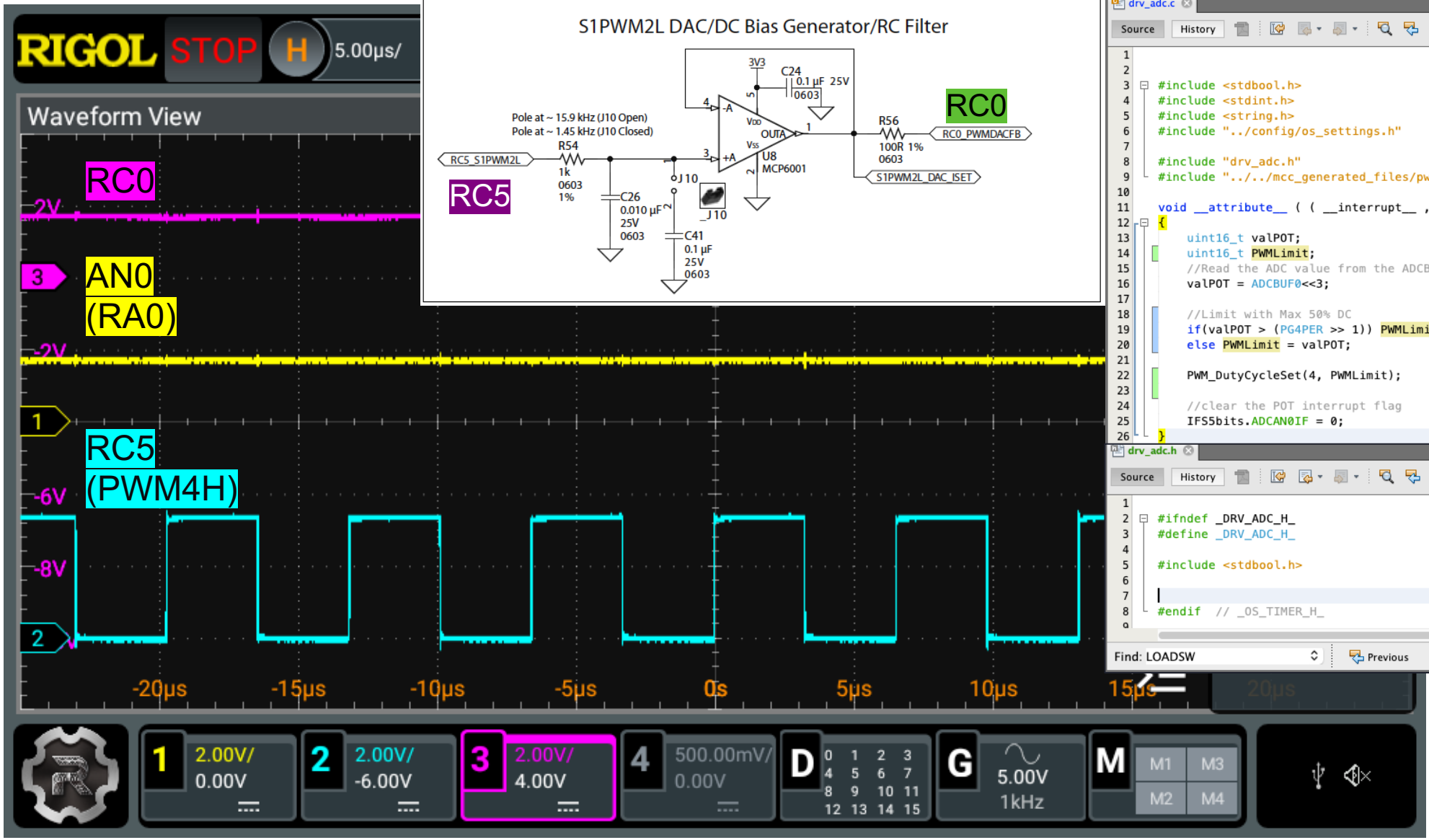
drv_adc.h

Source History

```c
#ifndef _DRV_ADC_H_
#define _DRV_ADC_H_

#include <stdbool.h>

#endif  // _OS_TIMER_H_
```

Find: LOADSW Previous Next No matches

AN0-POT Sets RC5_Master PWM4H



drv_adc.c

Source | History

```
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include "../config/os_settings.h"

#include "drv_adc.h"
#include "../../mcc_generated_files/pwm.h"

void __attribute__ ( ( __interrupt__ , auto_psv) ) _ADCAN0Interrupt ( void )
{
    uint16_t valPOT;
    uint16_t PWMLimit;
    //Read the ADC value from the ADCBUF
    valPOT = ADCBUF0<<3;

    //Limit with Max 50% DC
    if(valPOT > (PG4PER >> 1)) PWMLimit = PG4PER >>1;
    else PWMLimit = valPOT;

    PWM_DutyCycleSet(4, PWMLimit);

    //clear the POT interrupt flag
    IFS5bits.ADCAN0IF = 0;
}
```

drv_adc.h

Source | History

```
#ifndef _DRV_ADC_H_
#define _DRV_ADC_H_

#include <stdbool.h>

#endif  // _OS_TIMER_H_
```

Find: LOADSW Previous Next No matches

MCC + OS Scheduler for Dual Cores

Lab2

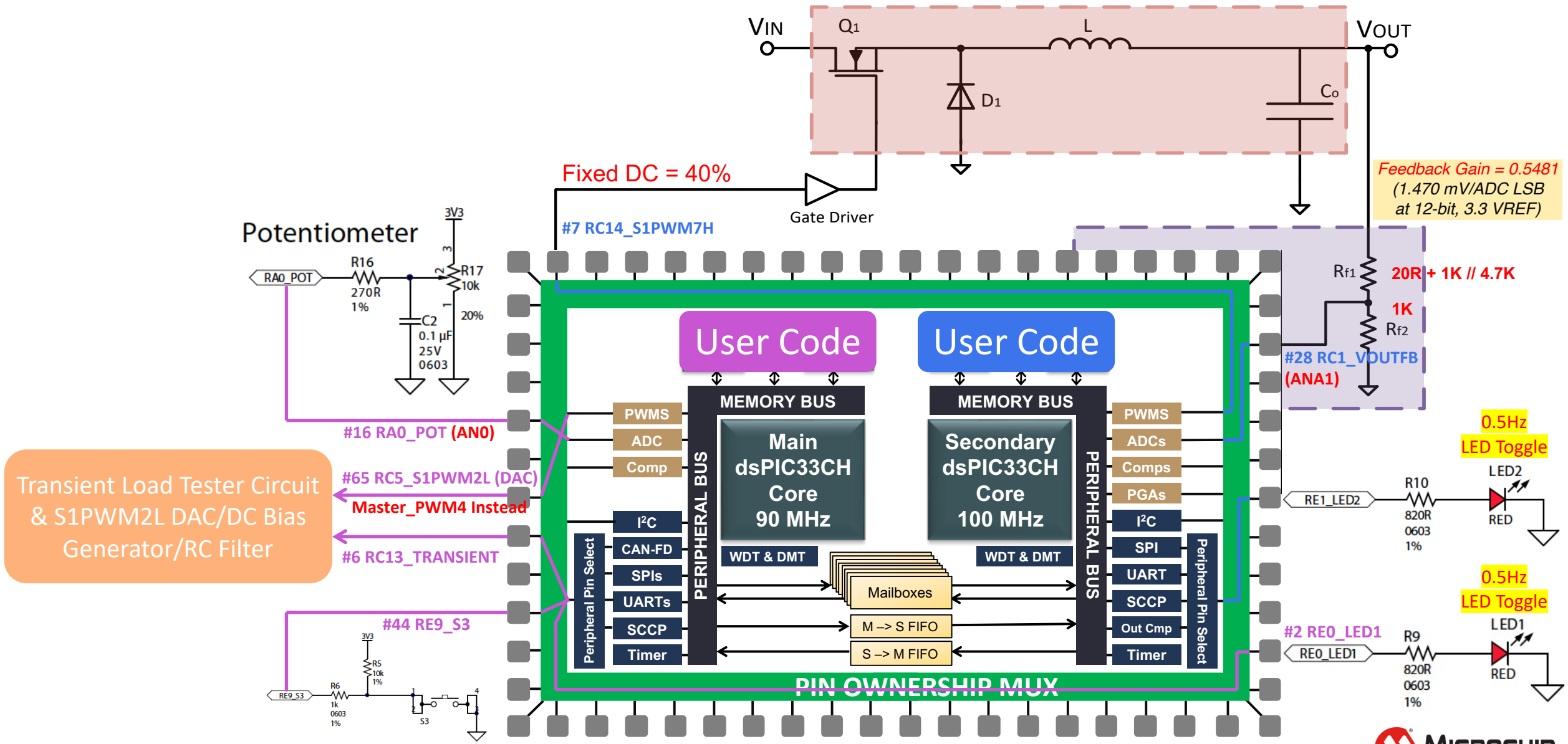
Reference manual on 

https://microchipdeveloper.com/mcc:mplab-code-configurator-support-for-dual-core-devices

Lab #2: MCC + OS Scheduler for Dual Cores

Main Core

Secondary Core



MICROCHIP

Lab #2: MCC + OS Scheduler for Dual Cores

Main Core

Secondary Core

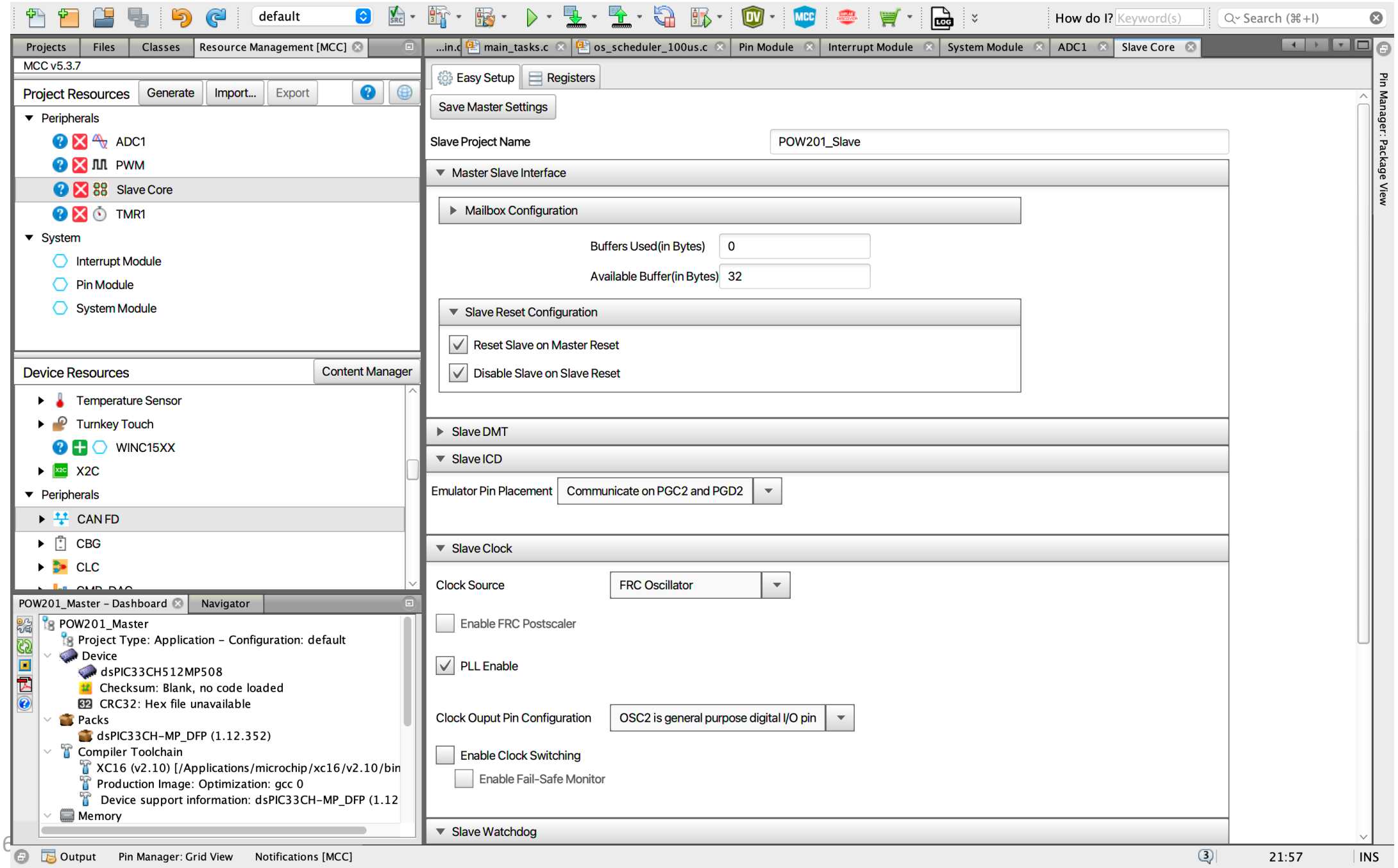
User Code (90MIPS)

- **Main loop:**
  - OS Scheduler
    - 0.5Hz Toggle LED1
    - Check S3 to trigger RC13_TRANSIENT
- **ISR:**
  - 1ms Timer1 for OS Scheduler
  - AN0-POT to set RC5_S1PWM2L(M-PWM4)
    - Max 50%
- **Peripheral:**
  - PWM4H– ~**152**KHz (Initial 0% Duty Cycle)
  - I/O: RE9-In, RC13-Out, RE0-Out
  - Timer1 with 1ms ISR
  - AN0 with ISR (IP=1)
    - Triggered by PWM2_Trigger1

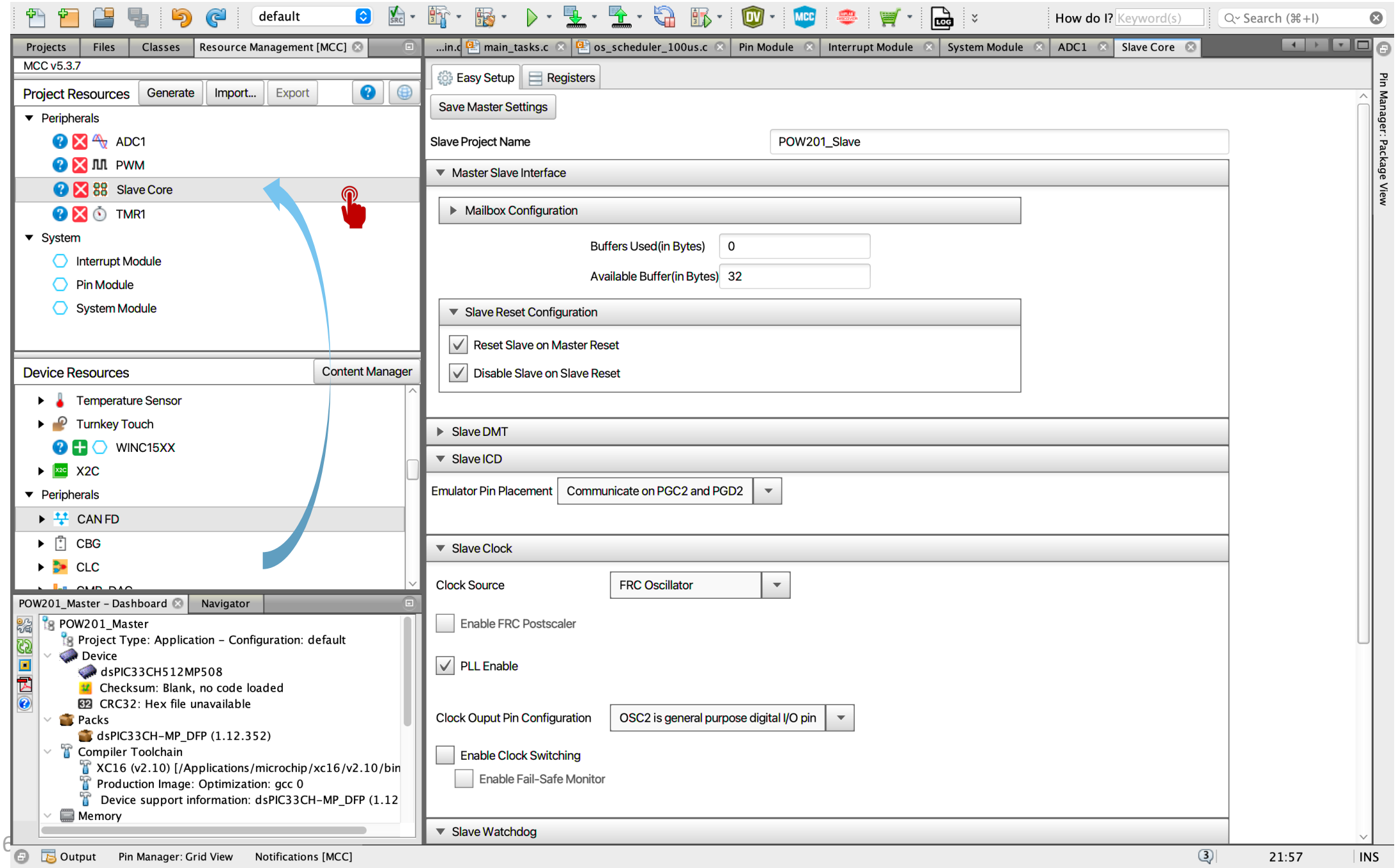
User Code (100MIPS)

- **Main loop:**
  - OS Scheduler
    - 0.5Hz Toggle LED2
- **ISR:**
  - 1ms Timer1 for OS Scheduler
  - ANA1-Vfb without user code.
- **Peripheral:**
  - PWM7H – 250KHz & Fixed 40% Duty Cycle
  - I/O: RE1-Out
  - Timer1 with 1ms ISR
  - ANA1 with ISR (IP=6)
    - Triggered by PWM7_Trigger1

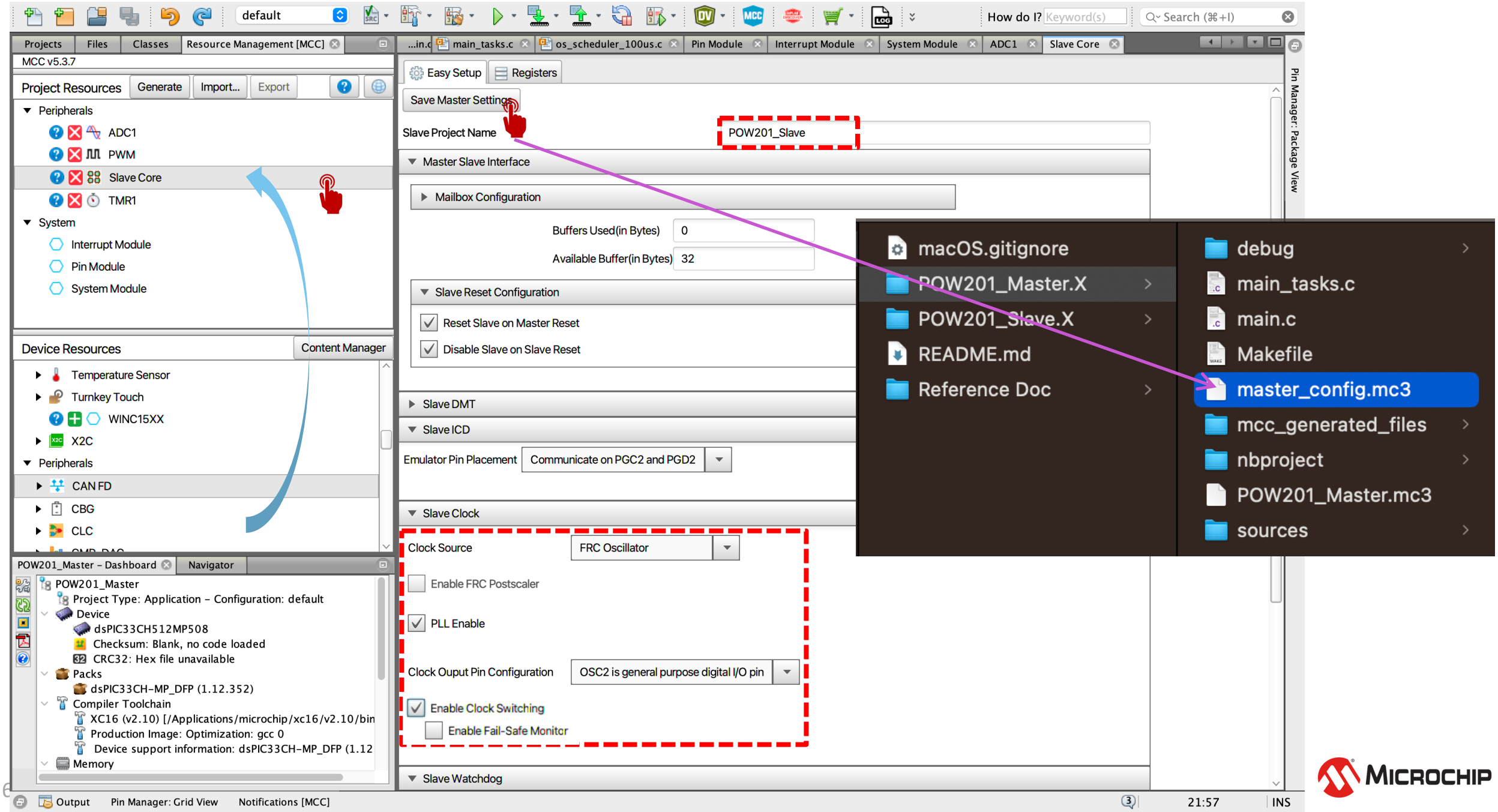
Set Secondary Core on Master-MCC



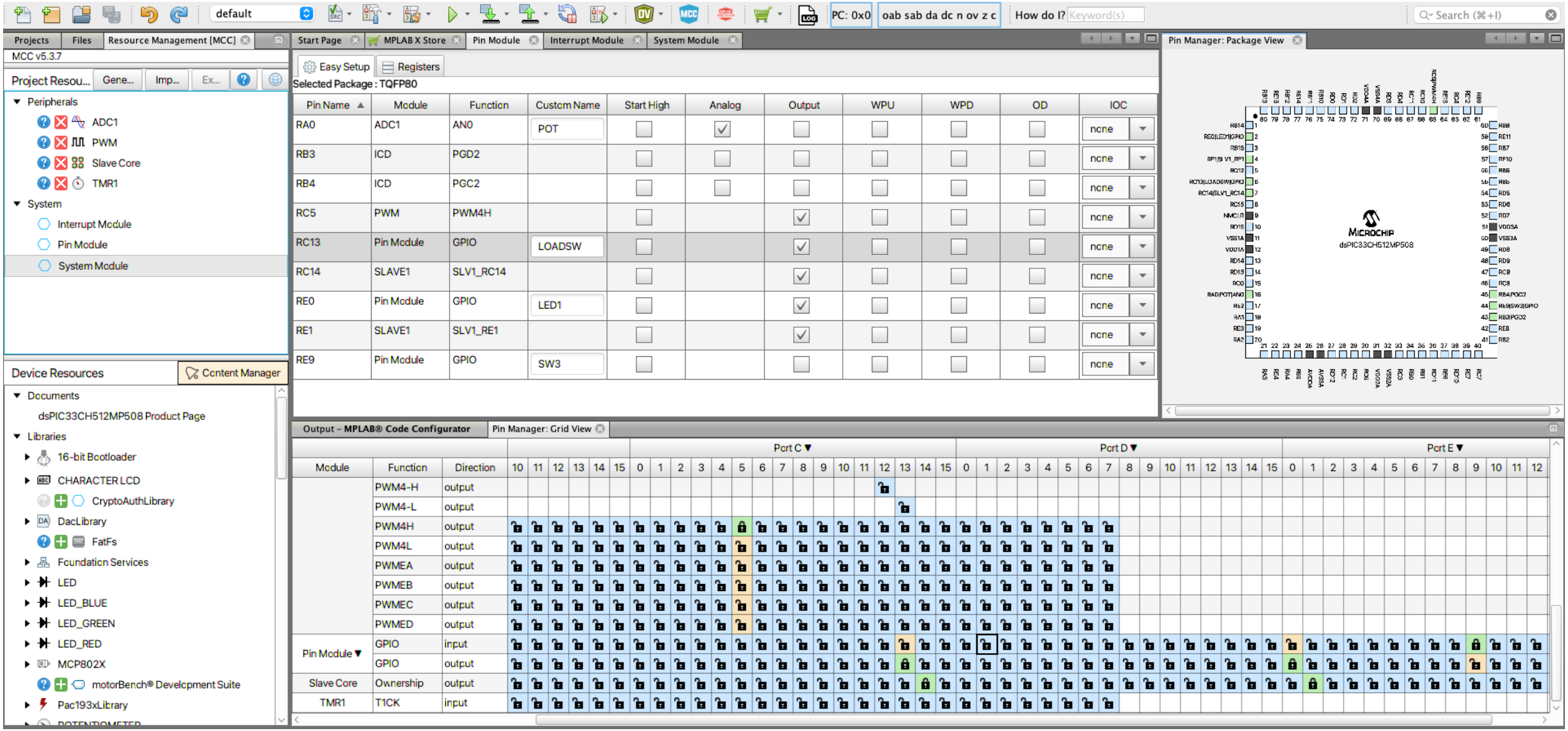
Set Secondary Core on Master-MCC



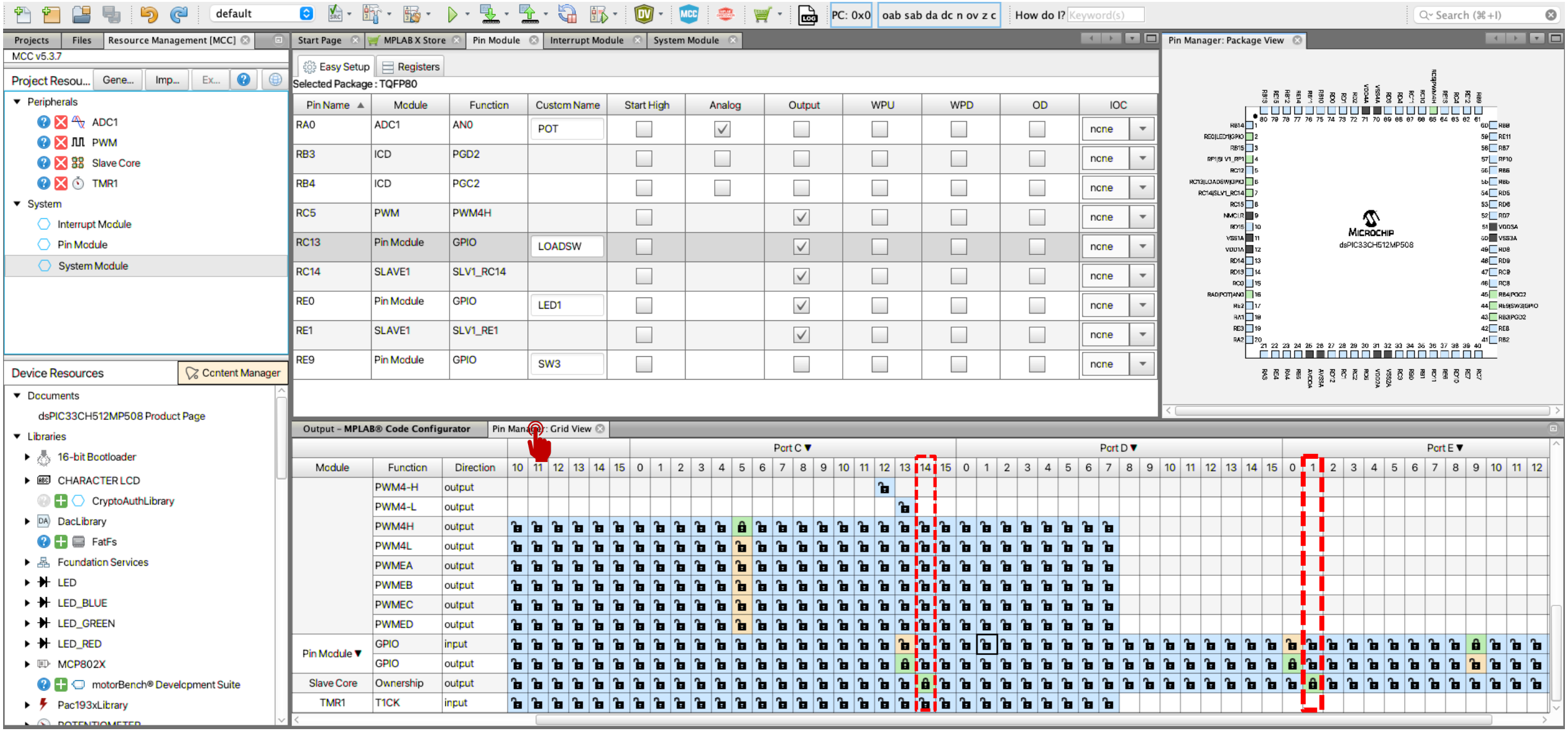
Set Secondary Core on Master-MCC



Important: Output Pin Ownership Settings on Master-MCC



Important: Output Pin Ownership Settings on Master-MCC

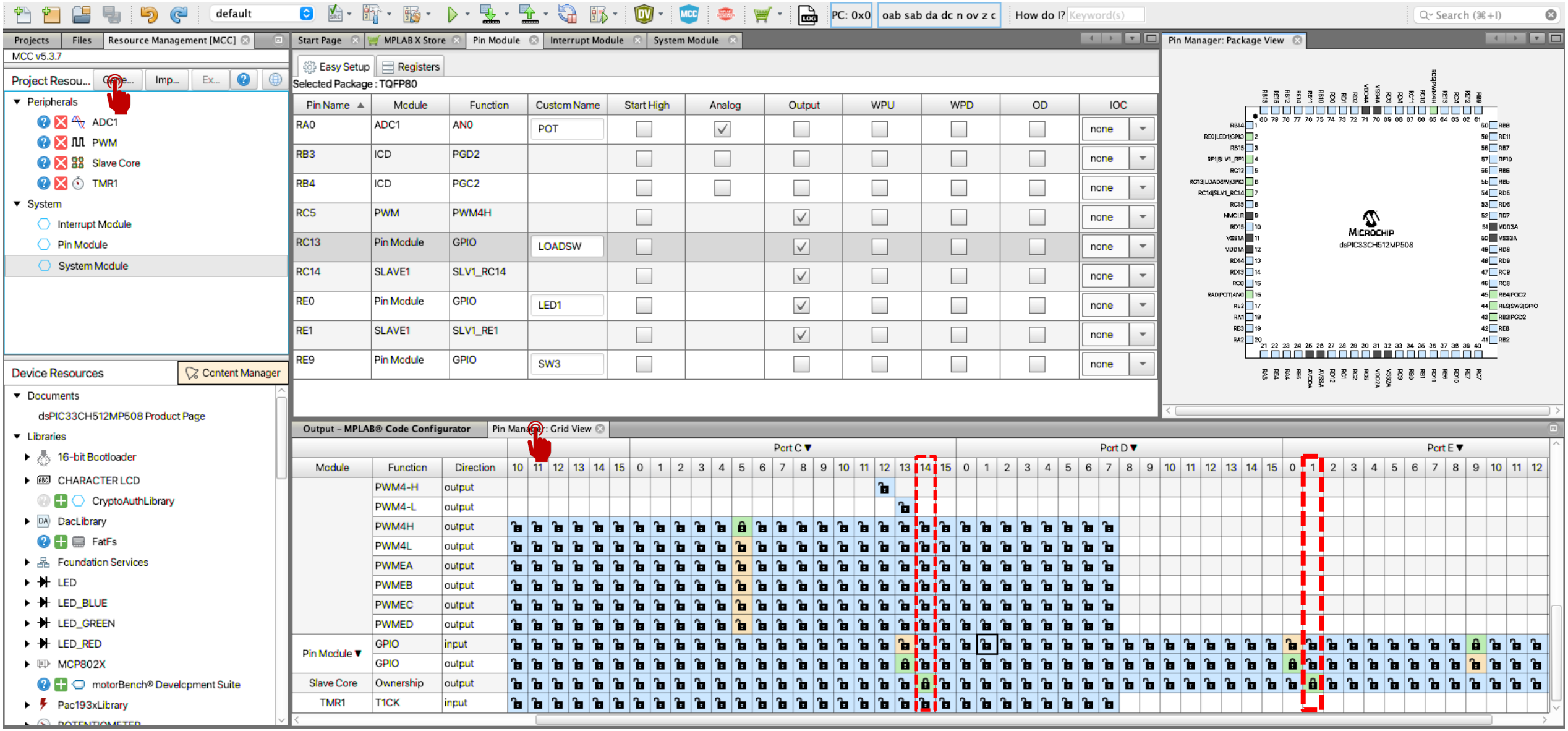


LED2

PWM7H

MICROCHIP

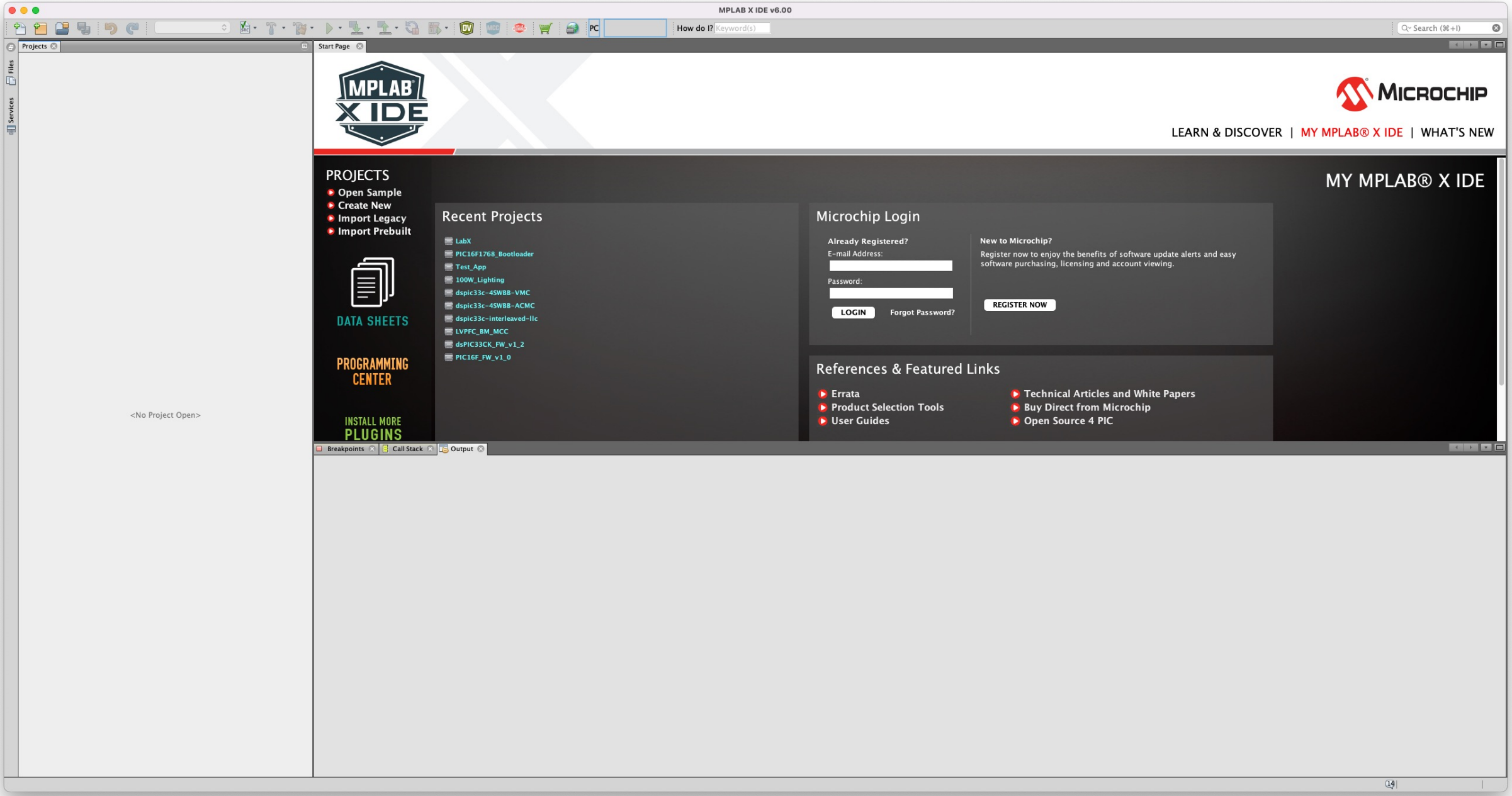
Important: Output Pin Ownership Settings on Master-MCC



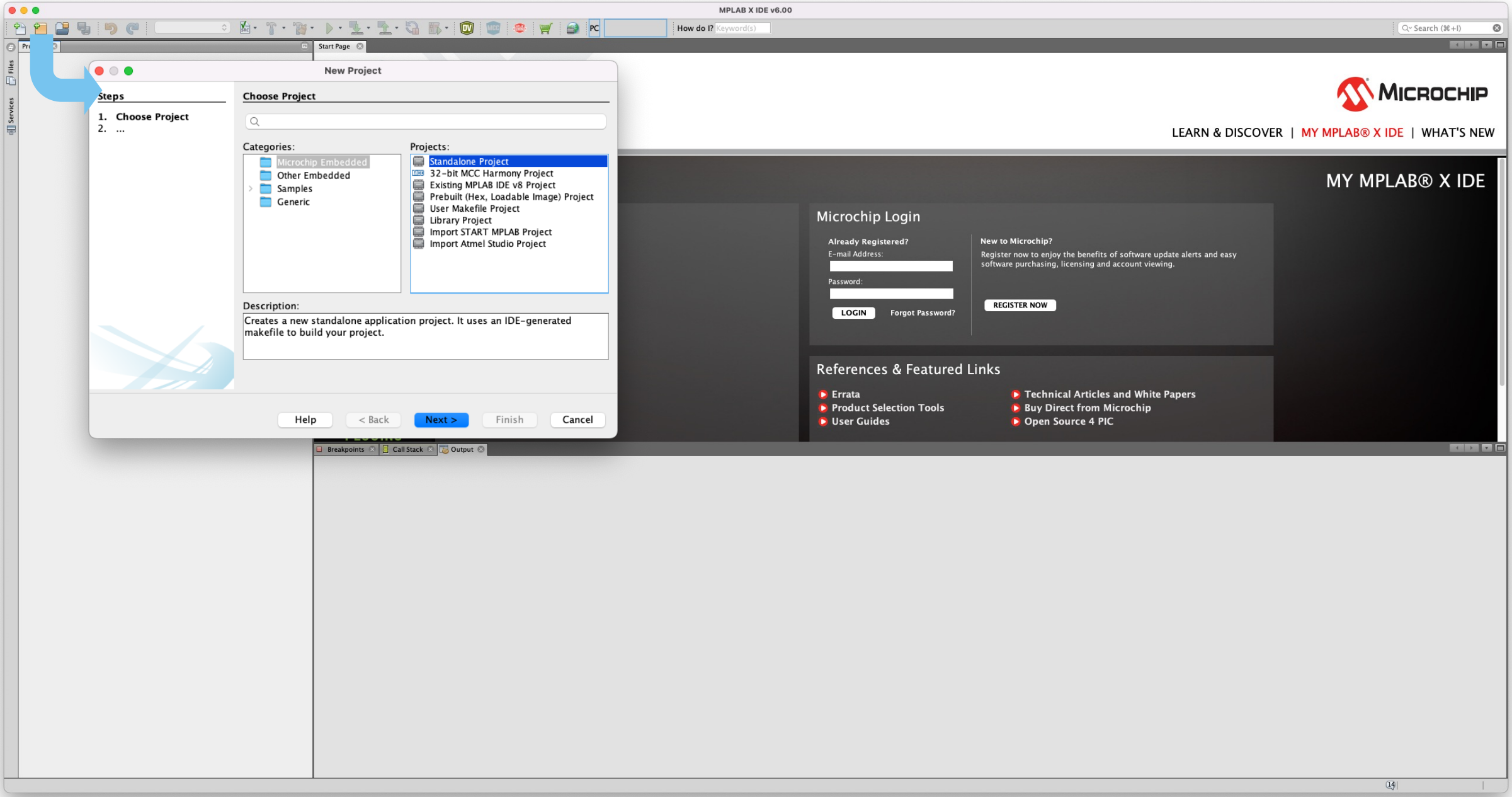
LED2

PWM7H

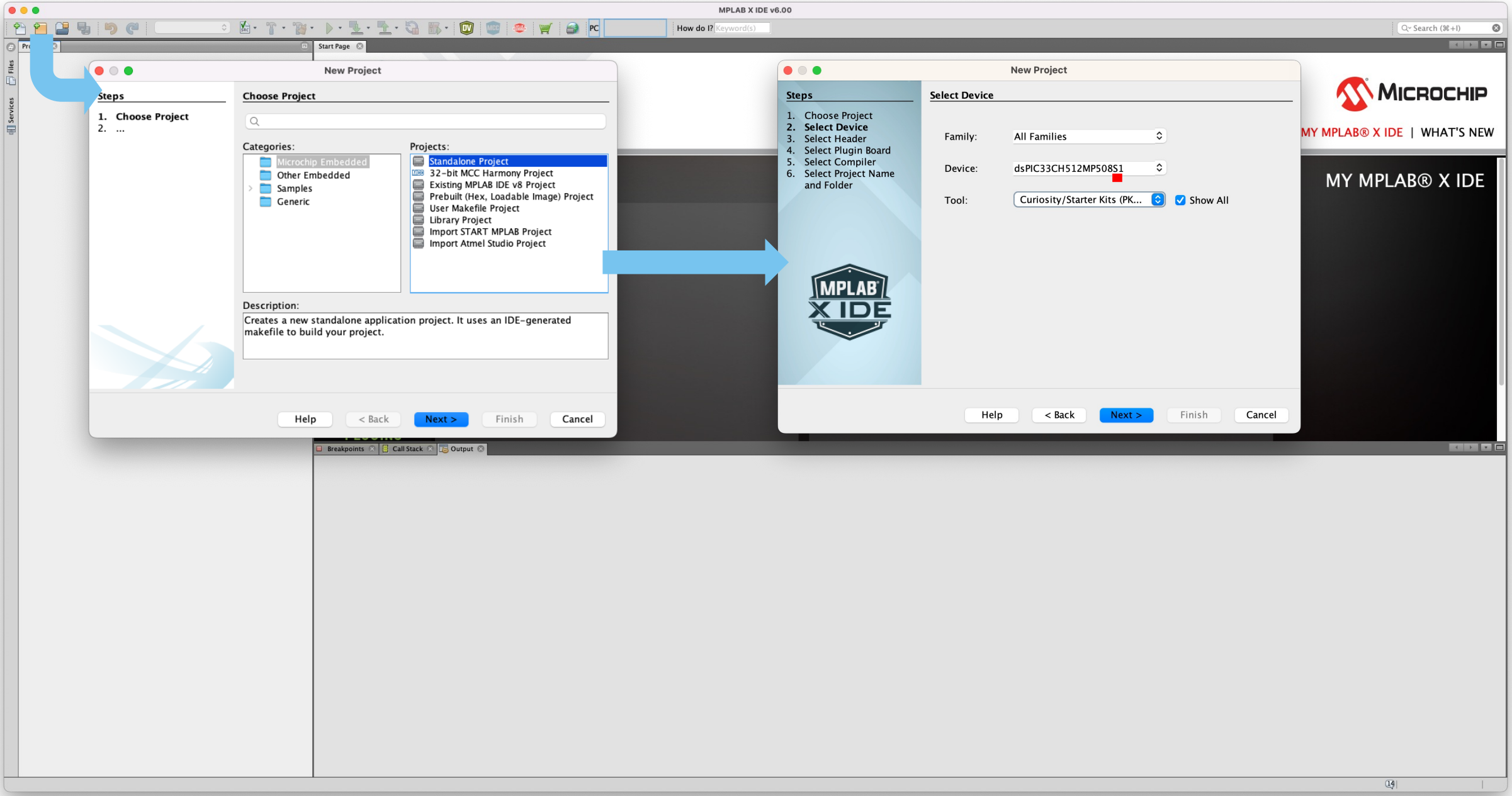
MICROCHIP


MPLAB X IDE v6.00
How do I?
Keyword(s)
Search (⌘+I)
Projects
Files
Services
<No Project Open>
Start Page
MPLAB X IDE
MICROCHIP
LEARN & DISCOVER | MY MPLAB® X IDE | WHAT'S NEW
PROJECTS
Open Sample
Create New
Import Legacy
Import Prebuilt
DATA SHEETS
PROGRAMMING CENTER
INSTALL MORE PLUGINS
Recent Projects
LabX
PIC16F1768_Bootloader
Test_App
100W_Lighting
dspic33c-4SWBB-VMC
dspic33c-4SWBB-ACMC
dspic33c-interleaved-llc
LVPFC_BM_MCC
dsPIC33CK_FW_v1_2
PIC16F_FW_v1_0
MY MPLAB® X IDE
Microchip Login
Already Registered?
E-mail Address:
Password:
LOGIN
Forgot Password?
New to Microchip?
Register now to enjoy the benefits of software update alerts and easy software purchasing, licensing and account viewing.
REGISTER NOW
References & Featured Links
Errata
Product Selection Tools
User Guides
Technical Articles and White Papers
Buy Direct from Microchip
Open Source 4 PIC
Breakpoints
Call Stack
Output

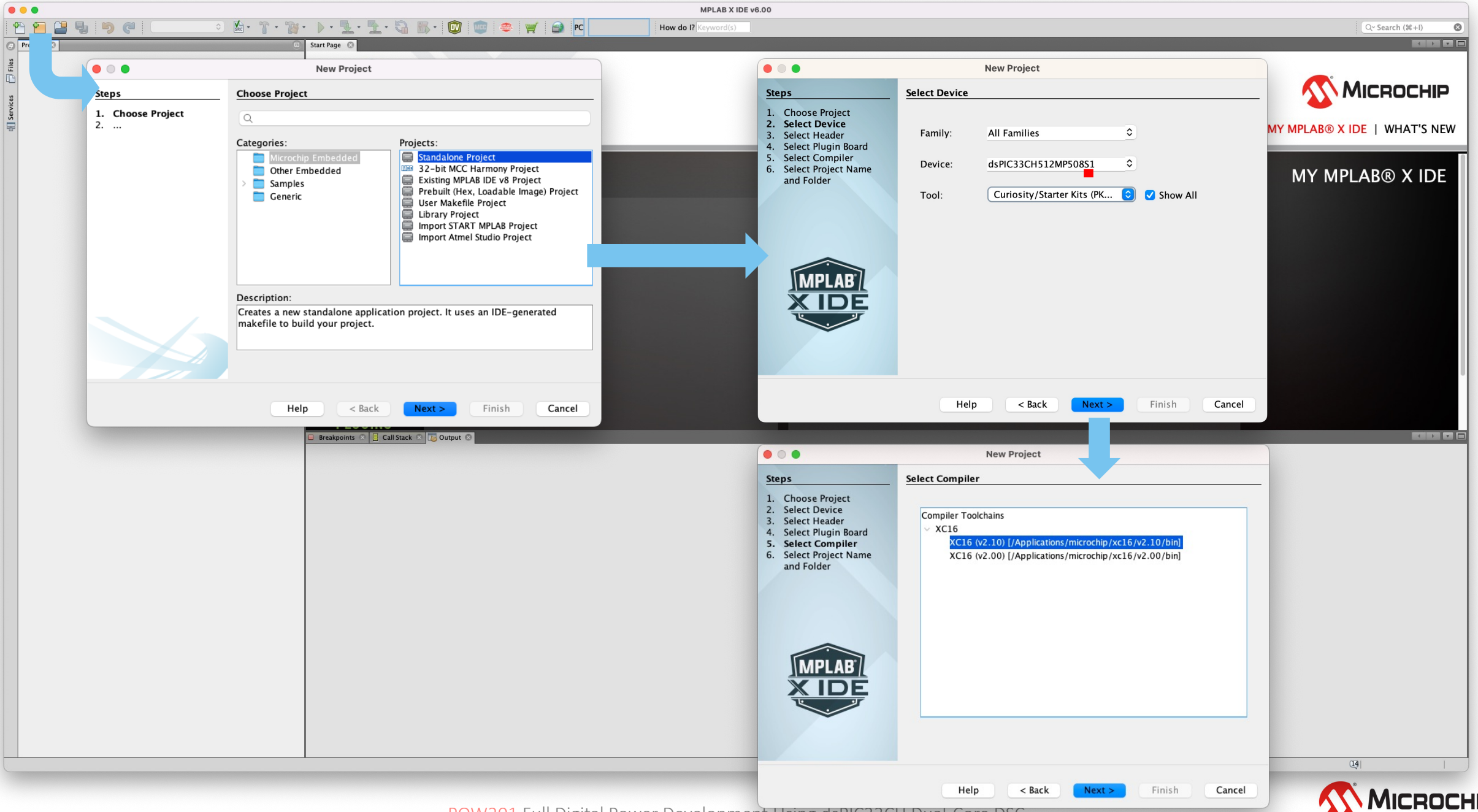
New Project as Slave on Secondary core



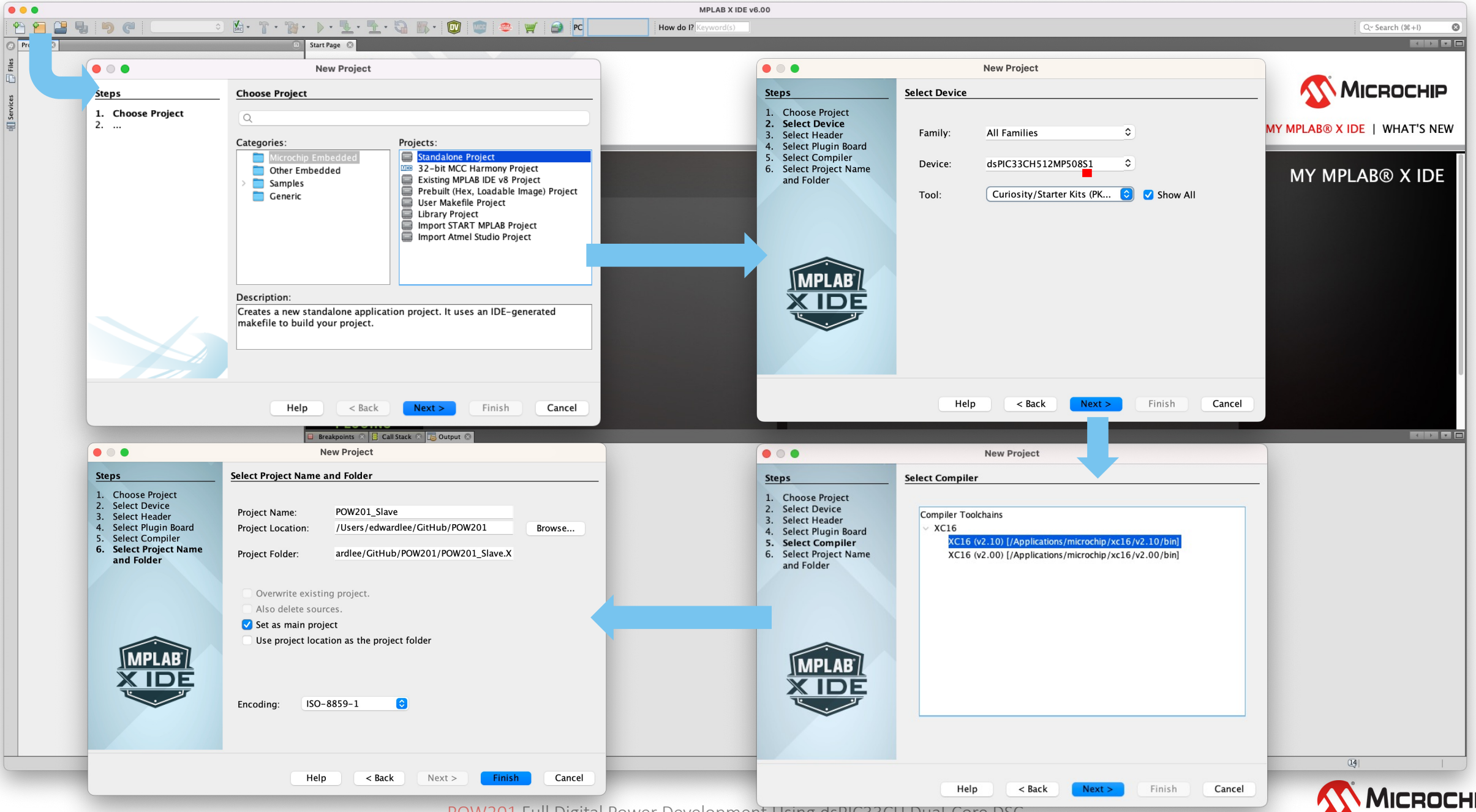
New Project as Slave on Secondary core



New Project as Slave on Secondary core



New Project as Slave on Secondary core



MCC Drivers

default

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

Projects | Files | Classes

- POW201_Master
  - Header Files
  - Important Files
  - Linker Files
  - Source Files
    - main.c
    - main_tasks.c
    - MCC Generated Files
    - os
  - Libraries
  - Loadables
  - Secondaries
- POW201_Slave
  - Header Files
  - Important Files
  - Linker Files
  - Source Files
  - Libraries
  - Loadables

POW201_Slave – Dashboard | main() – Navigator

- POW201_Slave
  - Project Type: Application – Configuration: default
  - Device
    - dsPIC33CH512MP508S1
    - Checksum: Blank, no code loaded
    - CRC32: Hex file unavailable
  - Packs
    - dsPIC33CH-MP_DFP (1.12.352)
    - Curiosity/Starter Kits (PKOB4) (1.11.1054)
  - Compiler Toolchain
    - XC16 (v2.10) [/Applications/microchip/xc16/v2.10/bin]
    - Production Image: Optimization: gcc 0
    - Device support information: dsPIC33CH-MP_DFP (1.12.35

Start Page | MPLAB X Store | MCC Content Manager

Import

MCC Content Manager Wizard

1. Content Type 2. Required Device Content

Select a Content Type

MCC Melody

Supports the MCC Builder
Supports content versioning at driver level
An iteration of MCC Generated Code
Works both on- and off-line

Select MCC Melody

Release notes and supported devices

MCC Classic

Development process you are accustomed to
All components and libraries that you have used before

Select MCC Classic

Release notes and supported devices

MPLAB® Harmony

Embedded Software Development Framework for 32-bit Microcontrollers and Microprocessors

Select MPLAB Harmony

Release notes and supported devices

Library support may be a key factor in your choice of MCC flavor:

> MCC Melody and MCC Classic - Library Summary

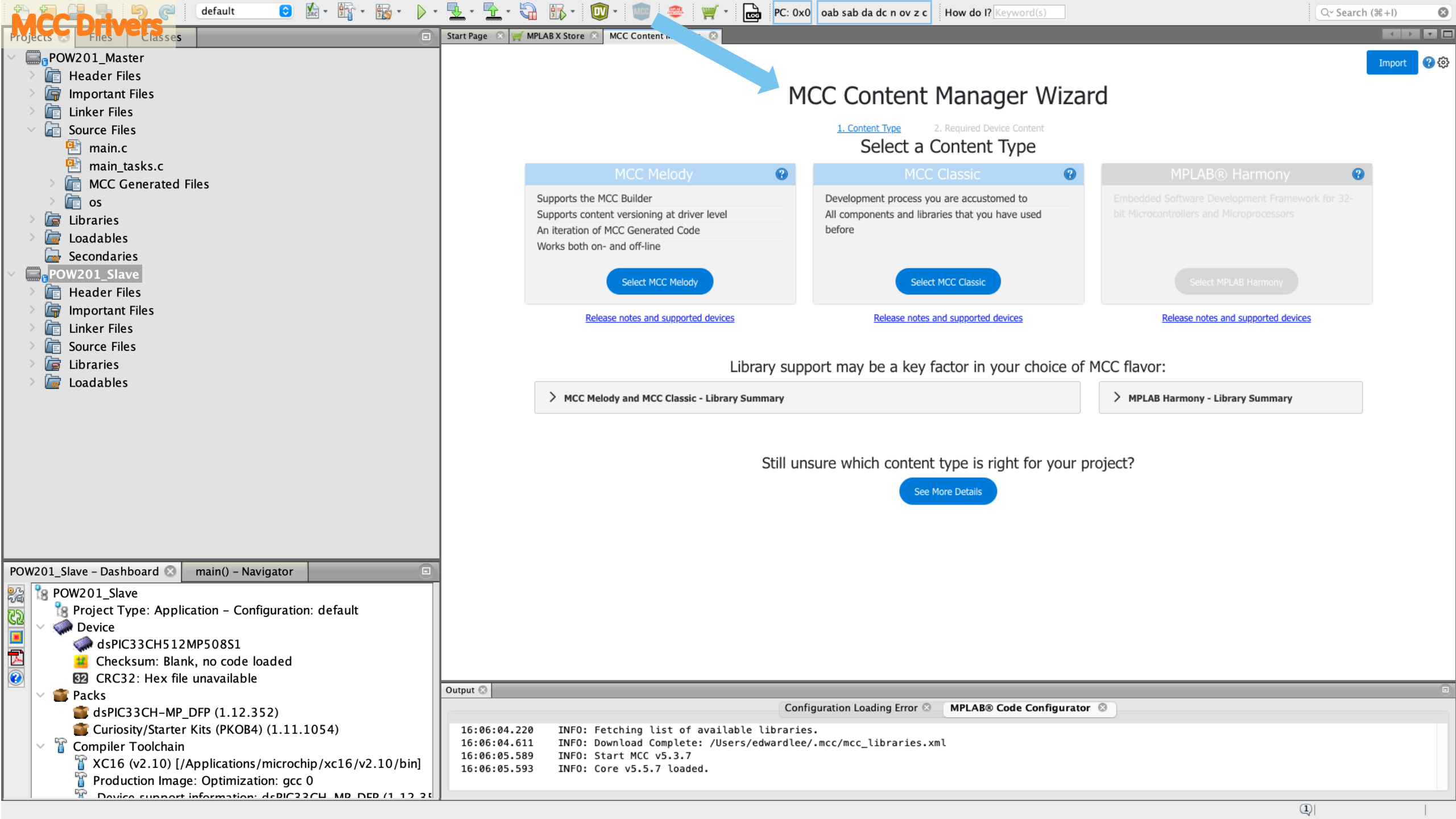
> MPLAB Harmony - Library Summary

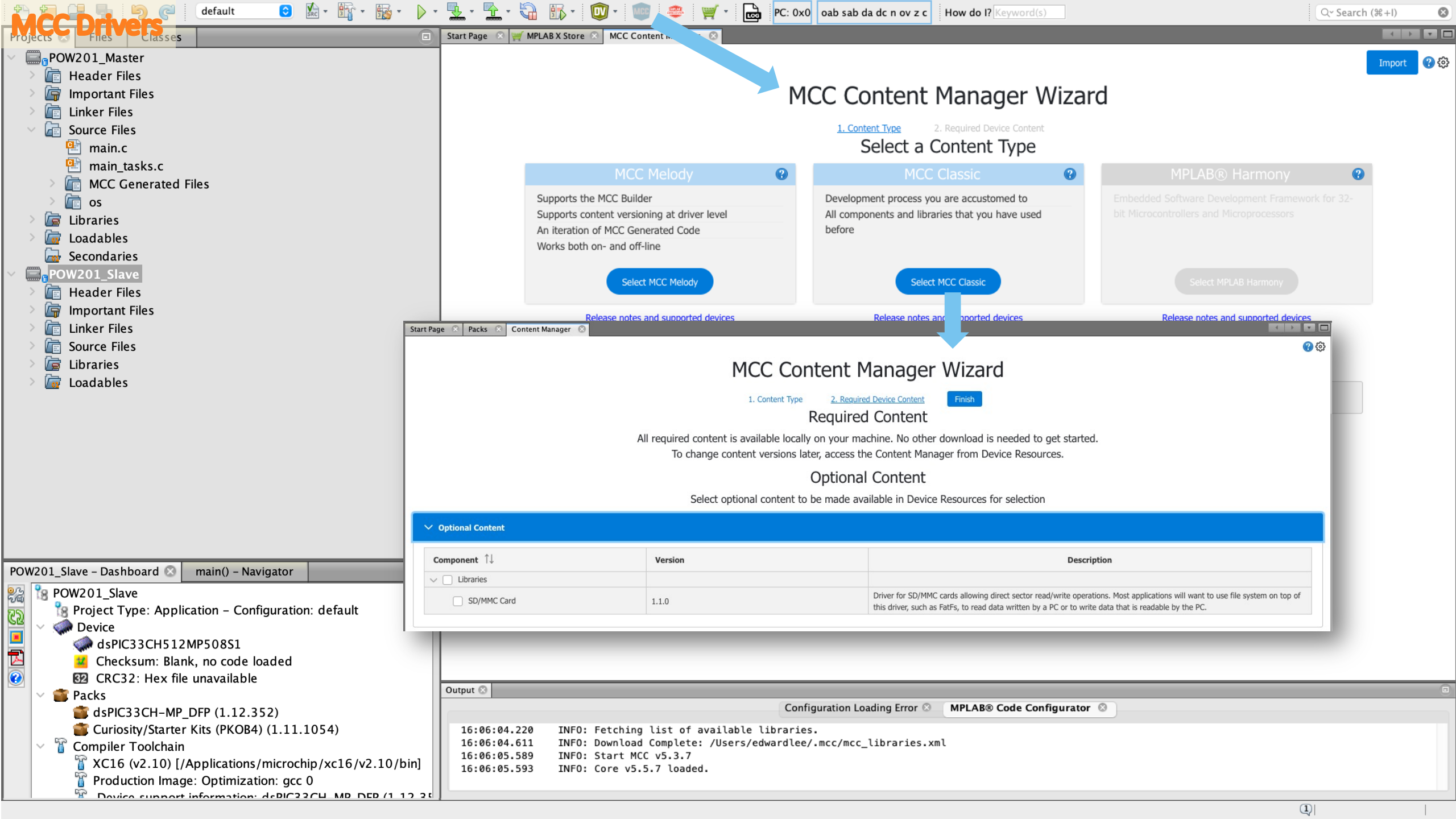
Still unsure which content type is right for your project?

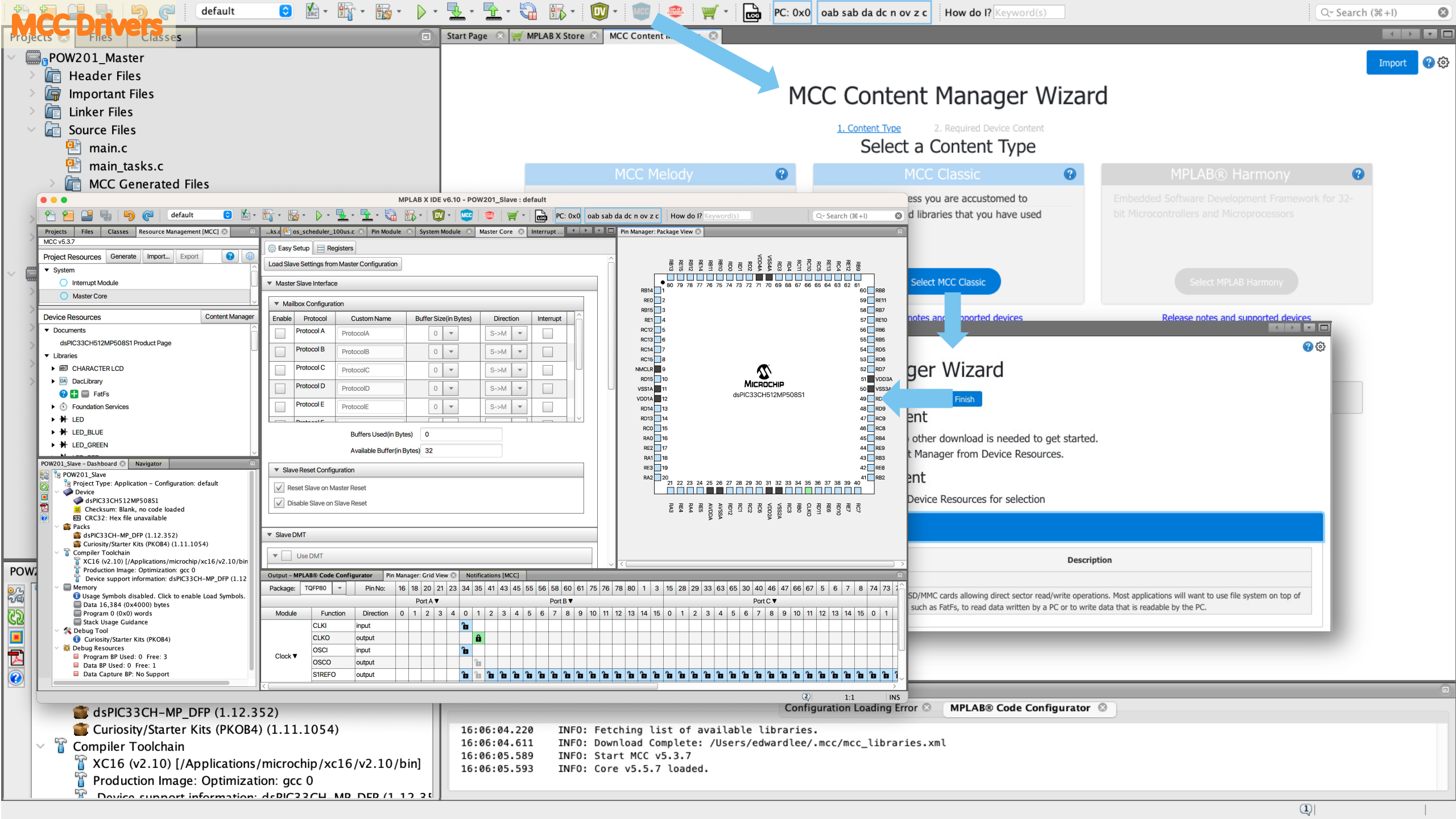
See More Details

Output | Configuration Loading Error | MPLAB® Code Configurator

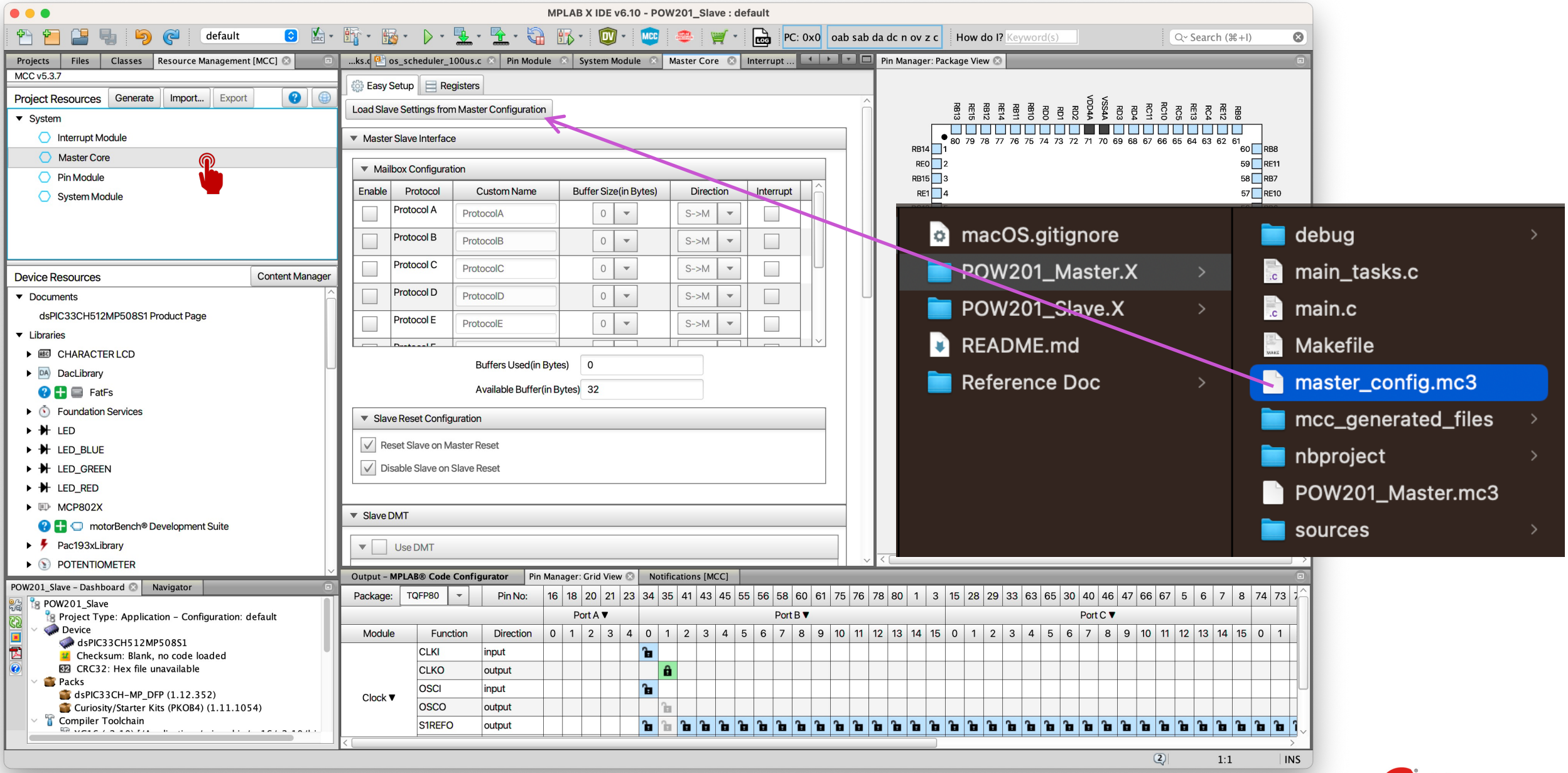
```
16:06:04.220    INFO: Fetching list of available libraries.
16:06:04.611    INFO: Download Complete: /Users/edwardlee/.mcc/mcc_libraries.xml
16:06:05.589    INFO: Start MCC v5.3.7
16:06:05.593    INFO: Core v5.5.7 loaded.
```


MCC Drivers
default
PC: 0x0
oab sab da dc n ov z c
How do I?
Keyword(s)
Search (⌘+I)
Projects
Files
Classes
POW201_Master
Header Files
Important Files
Linker Files
Source Files
main.c
main_tasks.c
MCC Generated Files
os
Libraries
Loadables
Secondaries
POW201_Slave
Header Files
Important Files
Linker Files
Source Files
Libraries
Loadables
Start Page
MPLAB X Store
MCC Content
Import
MCC Content Manager Wizard
1. Content Type
2. Required Device Content
Select a Content Type
MCC Melody
Supports the MCC Builder
Supports content versioning at driver level
An iteration of MCC Generated Code
Works both on- and off-line
Select MCC Melody
Release notes and supported devices
MCC Classic
Development process you are accustomed to
All components and libraries that you have used before
Select MCC Classic
Release notes and supported devices
MPLAB® Harmony
Embedded Software Development Framework for 32-bit Microcontrollers and Microprocessors
Select MPLAB Harmony
Release notes and supported devices
Library support may be a key factor in your choice of MCC flavor:
MCC Melody and MCC Classic - Library Summary
MPLAB Harmony - Library Summary
Still unsure which content type is right for your project?
See More Details
POW201_Slave - Dashboard
main() - Navigator
POW201_Slave
Project Type: Application - Configuration: default
Device
dsPIC33CH512MP508S1
Checksum: Blank, no code loaded
CRC32: Hex file unavailable
Packs
dsPIC33CH-MP_DFP (1.12.352)
Curiosity/Starter Kits (PKOB4) (1.11.1054)
Compiler Toolchain
XC16 (v2.10) [/Applications/microchip/xc16/v2.10/bin]
Production Image: Optimization: gcc 0
Output
Configuration Loading Error
MPLAB® Code Configurator
16:06:04.220 INFO: Fetching list of available libraries.
16:06:04.611 INFO: Download Complete: /Users/edwardlee/.mcc/mcc_libraries.xml
16:06:05.589 INFO: Start MCC v5.3.7
16:06:05.593 INFO: Core v5.5.7 loaded.


MCC Drivers
default
PC: 0x0
oab sab da dc n ov z c
How do I?
Keyword(s)
Search (⌘+I)
Projects
Files
Classes
POW201_Master
Header Files
Important Files
Linker Files
Source Files
main.c
main_tasks.c
MCC Generated Files
os
Libraries
Loadables
Secondaries
POW201_Slave
Header Files
Important Files
Linker Files
Source Files
Libraries
Loadables
Start Page
MPLAB X Store
MCC Content
Import
MCC Content Manager Wizard
1. Content Type
2. Required Device Content
Select a Content Type
MCC Melody
Supports the MCC Builder
Supports content versioning at driver level
An iteration of MCC Generated Code
Works both on- and off-line
Select MCC Melody
MCC Classic
Development process you are accustomed to
All components and libraries that you have used before
Select MCC Classic
MPLAB® Harmony
Embedded Software Development Framework for 32-bit Microcontrollers and Microprocessors
Select MPLAB Harmony
Release notes and supported devices
Start Page
Packs
Content Manager
MCC Content Manager Wizard
1. Content Type
2. Required Device Content
Finish
Required Content
All required content is available locally on your machine. No other download is needed to get started.
To change content versions later, access the Content Manager from Device Resources.
Optional Content
Select optional content to be made available in Device Resources for selection
Optional Content
Component
Version
Description
Libraries
SD/MMC Card
1.1.0
Driver for SD/MMC cards allowing direct sector read/write operations. Most applications will want to use file system on top of this driver, such as FatFs, to read data written by a PC or to write data that is readable by the PC.
POW201_Slave - Dashboard
main() - Navigator
POW201_Slave
Project Type: Application - Configuration: default
Device
dsPIC33CH512MP508S1
Checksum: Blank, no code loaded
CRC32: Hex file unavailable
Packs
dsPIC33CH-MP_DFP (1.12.352)
Curiosity/Starter Kits (PKOB4) (1.11.1054)
Compiler Toolchain
XC16 (v2.10) [/Applications/microchip/xc16/v2.10/bin]
Production Image: Optimization: gcc 0
Output
Configuration Loading Error
MPLAB® Code Configurator
16:06:04.220 INFO: Fetching list of available libraries.
16:06:04.611 INFO: Download Complete: /Users/edwardlee/.mcc/mcc_libraries.xml
16:06:05.589 INFO: Start MCC v5.3.7
16:06:05.593 INFO: Core v5.5.7 loaded.


MCC Drivers
POW201_Master
Header Files
Important Files
Linker Files
Source Files
main.c
main_tasks.c
MCC Generated Files
Start Page
MPLAB X Store
MCC Content
MCC Content Manager Wizard
1. Content Type
2. Required Device Content
Select a Content Type
MCC Melody
MCC Classic
MPLAB® Harmony
Import
Select MCC Classic
Select MPLAB Harmony
Release notes and supported devices
MPLAB X IDE v6.10 - POW201_Slave : default
Resource Management [MCC]
Project Resources
Generate
Import...
Export
System
Interrupt Module
Master Core
Device Resources
Content Manager
Documents
dsPIC33CH512MP508S1 Product Page
Libraries
CHARACTER LCD
DacLibrary
FatFs
Foundation Services
LED
LED_BLUE
LED_GREEN
Easy Setup
Registers
Load Slave Settings from Master Configuration
Master Slave Interface
Mailbox Configuration
Enable
Protocol
Custom Name
Buffer Size(in Bytes)
Direction
Interrupt
Protocol A
Protocol B
Protocol C
Protocol D
Protocol E
Buffers Used(in Bytes)
Available Buffer(in Bytes)
Slave Reset Configuration
Reset Slave on Master Reset
Disable Slave on Slave Reset
Slave DMT
Use DMT
Pin Manager: Package View
MICROCHIP
dsPIC33CH512MP508S1
Pin Manager: Grid View
Package:
TQFP80
Pin No:
Port A
Port B
Port C
Module
Function
Direction
Clock
CLKI
CLKO
OSCI
OSCO
S1REFO
POW201_Slave - Dashboard
Navigator
Project Type: Application - Configuration: default
Device
dsPIC33CH512MP508S1
Checksum: Blank, no code loaded
CRC32: Hex file unavailable
Packs
dsPIC33CH-MP_DFP (1.12.352)
Curiosity/Starter Kits (PKOB4) (1.11.1054)
Compiler Toolchain
XC16 (v2.10) [/Applications/microchip/xc16/v2.10/bin]
Production Image: Optimization: gcc 0
Memory
Usage Symbols disabled. Click to enable Load Symbols.
Data 16,384 (0x4000) bytes
Program 0 (0x0) words
Stack Usage Guidance
Debug Tool
Curiosity/Starter Kits (PKOB4)
Debug Resources
Program BP Used: 0 Free: 3
Data BP Used: 0 Free: 1
Data Capture BP: No Support
Output - MPLAB® Code Configurator
Notifications [MCC]
Configuration Loading Error
MPLAB® Code Configurator
16:06:04.220 INFO: Fetching list of available libraries.
16:06:04.611 INFO: Download Complete: /Users/edwardlee/.mcc/mcc_libraries.xml
16:06:05.589 INFO: Start MCC v5.3.7
16:06:05.593 INFO: Core v5.5.7 loaded.
Finish
Description

Load Slave Settings from Master Configuration



MCC Drivers – Clock@100 MIPS

MCC v5.3.7

Project Resources | Generate | Import... | Export

- System
  - Interrupt Module
  - Master Core
  - Pin Module
  - System Module

Device Resources | Content Manager

- Documents
  - dsPIC33CH512MP508S1 Product Page
- Libraries
  - CHARACTER LCD
  - DacLibrary
  - FatFs
  - Foundation Services
  - LED
  - LED_BLUE
  - LED_GREEN
  - LED_RED
  - MCP802X
  - motorBench® Development Suite
  - Pac193xLibrary
  - POTENTIOMETER
  - RGB LED
  - SERIALUART
  - SWITCH
  - Temperature Sensor
  - Turnkey Touch
  - WINC15XX
  - X2C
- Peripherals
  - ADC
  - 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

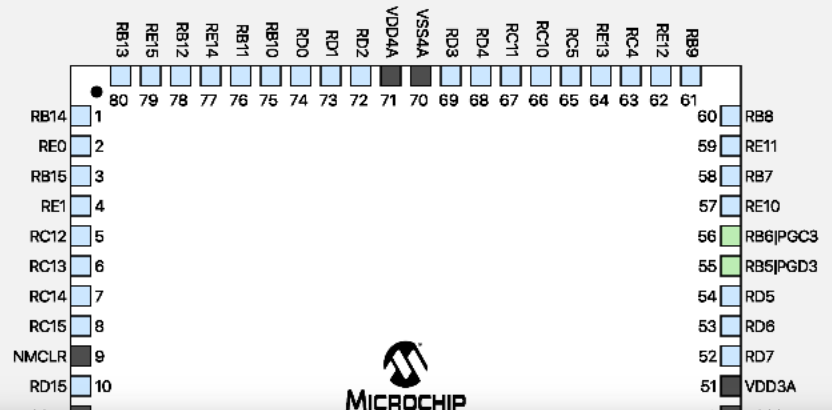
400 MHz	FVCO/4	VCO Divider
500 MHz	FVCO/2	AVCO Divider

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

Reference Oscillator Output

CAN FD Clock Generator

Pin Manager: Package View



6.1 Primary PLL

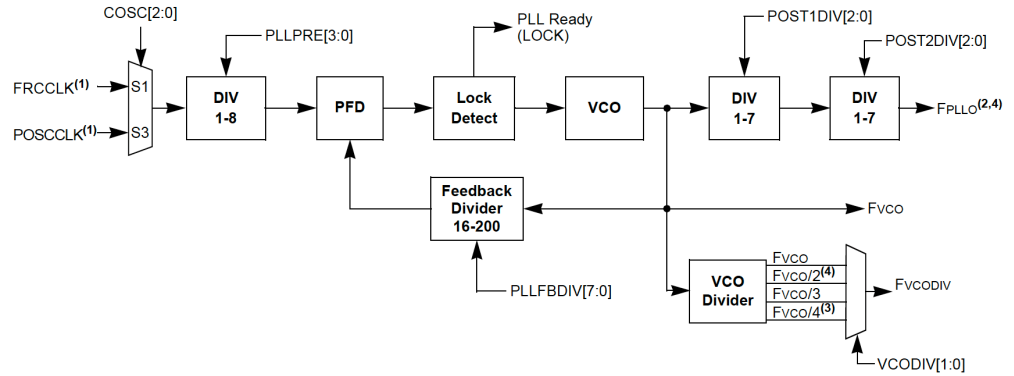
The Primary Oscillator and internal FRC Oscillator sources can optionally use an on-chip PLL to obtain higher operating speeds. There are two independent instantiations of PLL for the Master and Slave clock subsystems. Figure 6-4 illustrates a block diagram of the Master/Slave core PLL module.

For PLL operation, the following requirements must be met at all times without exception:

- The PLL Input Frequency (FPLLI) must be in the range of 8 MHz to 64 MHz
- The PFD Input Frequency (FPFD) must be in the range of 8 MHz to (FVCO/16) MHz

The VCO Output Frequency (FVCO) must be in the range of 400 MHz to 1600 MHz

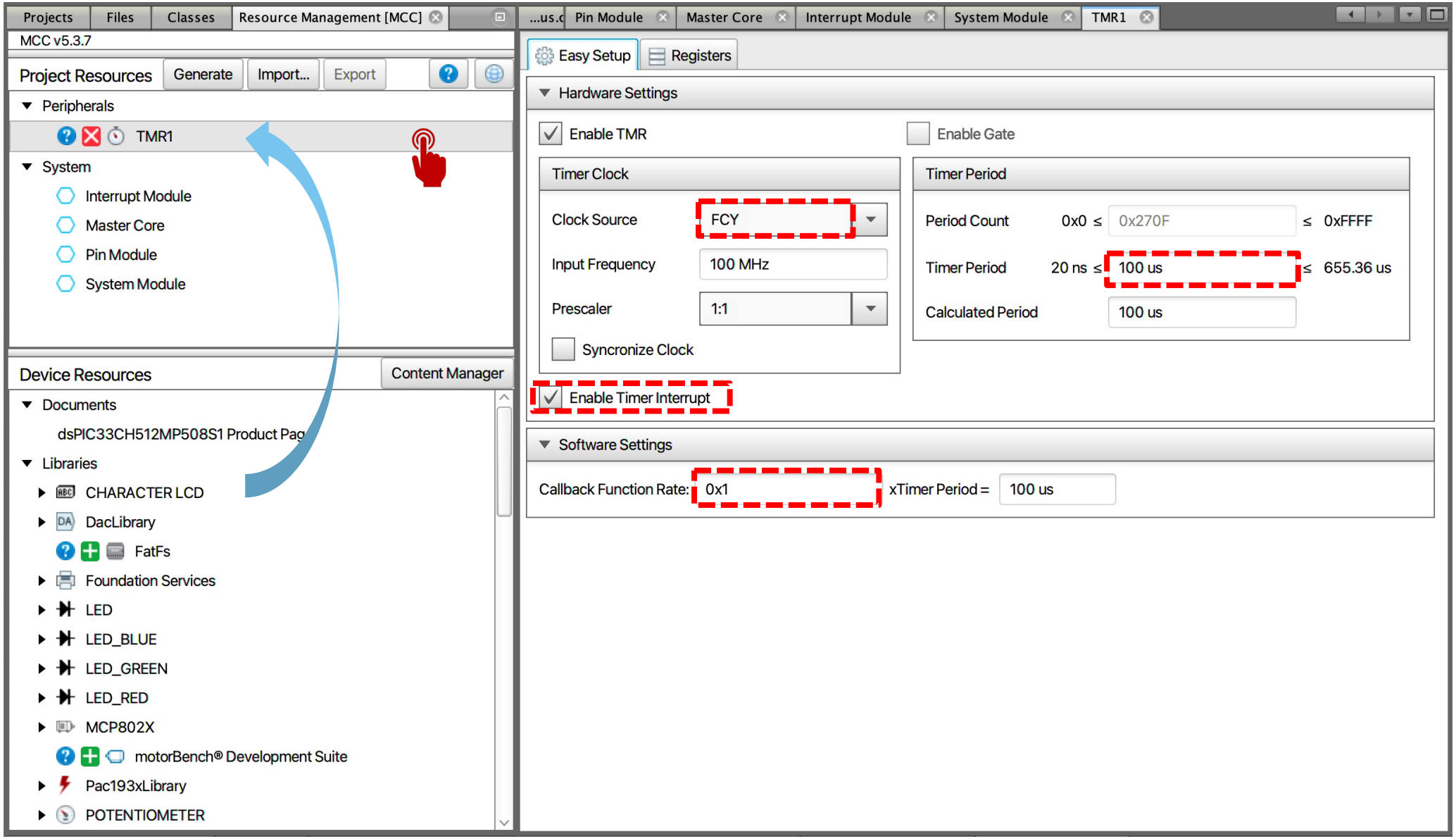
FIGURE 6-4: MASTER/SLAVE CORE PLL AND VCO DETAIL



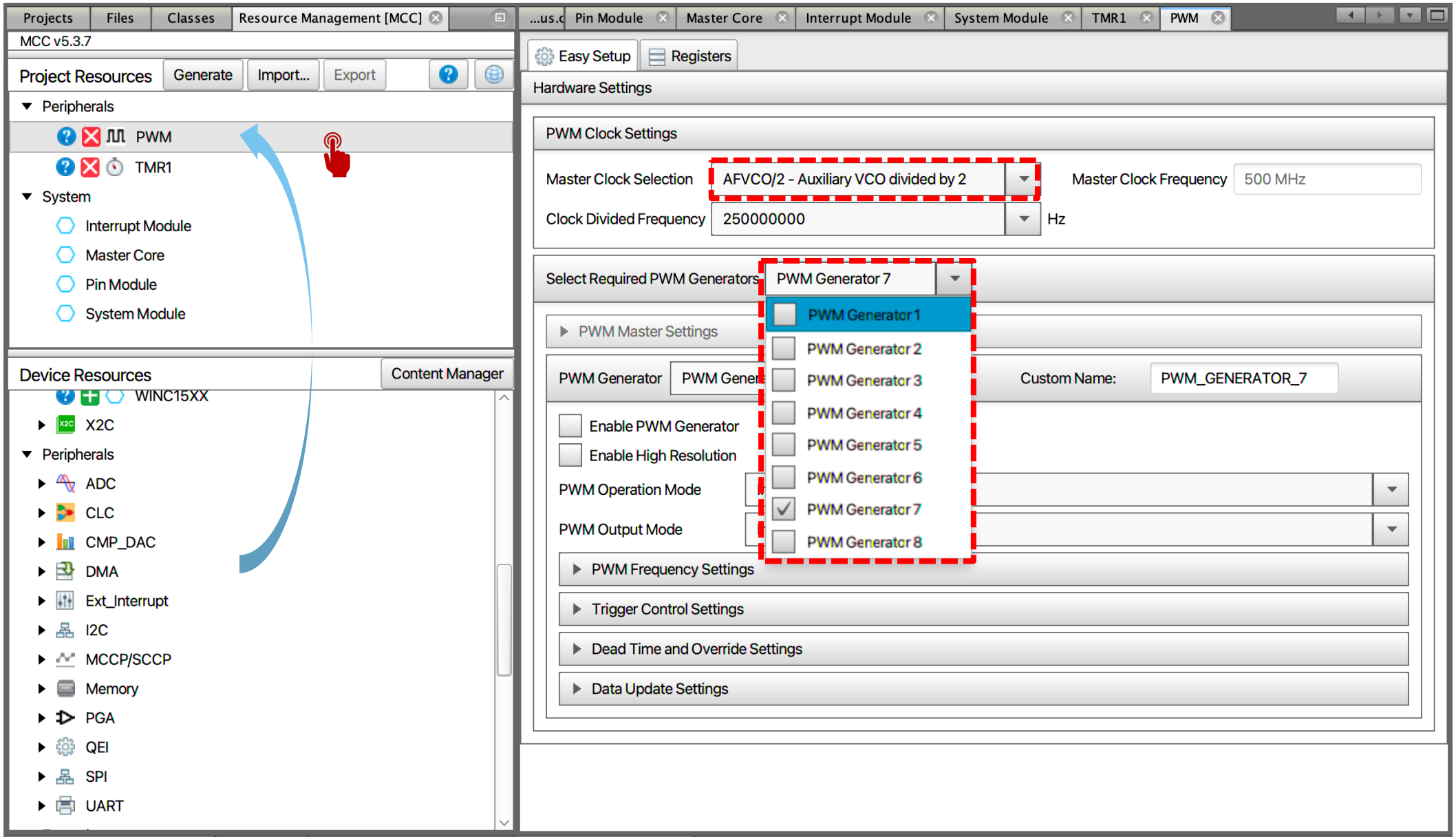
Note 1: From Master and Slave core shared oscillator source.
2: Clock option for PWM.
3: Clock option for ADC.
4: Clock option for DAC.

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

MCC Drivers – Timer1 (100 us)



MCC Drivers – PWM7H – 250KHz & Fixed 40% Duty Cycle



MCC Drivers – PWM7H – 250KHz & Fixed 40% Duty Cycle

Pin Module | Master Core | Interrupt Module | System Module | TMR1 | PWM

Easy Setup | Registers

PWM Generator: PWM Generator 7 — Custom Name: PWM_GENERATOR_7

- [x] Enable PWM Generator
- [x] Enable High Resolution

PWM Operation Mode: Independent Edge

PWM Output Mode: Independent

PWM Frequency Settings

PWM Input Clock Selection: 500000000 Hz

Period — [ ] Use Master Period

Requested Frequency 61.02864 kHz ≤ 250 kHz ≤ 29.41176471 MHz

Calculated Frequency: 250 kHz

Requested Period 34 ns ≤ 4 us ≤ 16.3858 us

Calculated Period: 4 us

Duty Cycle — [ ] Use Master Duty Cycle

PWM Duty Cycle 0 % ≤ 40 ≤100 %

Phase

PWM Phase 0 ns ≤ 0 ns ≤ 16.3838 us

Trigger Control Settings

Dead Time and Override Settings

Data Update Settings

Update Trigger: Duty Cycle

Update Mode: SOC update

Trigger Control Settings

PWM Start of Cycle Control

Start of Cycle Trigger: Self-trigger

Trigger Output Selection: EOC event

ADC Trigger

ADC Trigger 1: Trigger A Compare

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

PWM

Easy Setup | Registers

Register: PG7IOCONH 0x1E

CAPSRC: Software

DTCMPSEL: PCI Sync Logic

PENH: enabled

PENL: enabled

PMOD: Independent

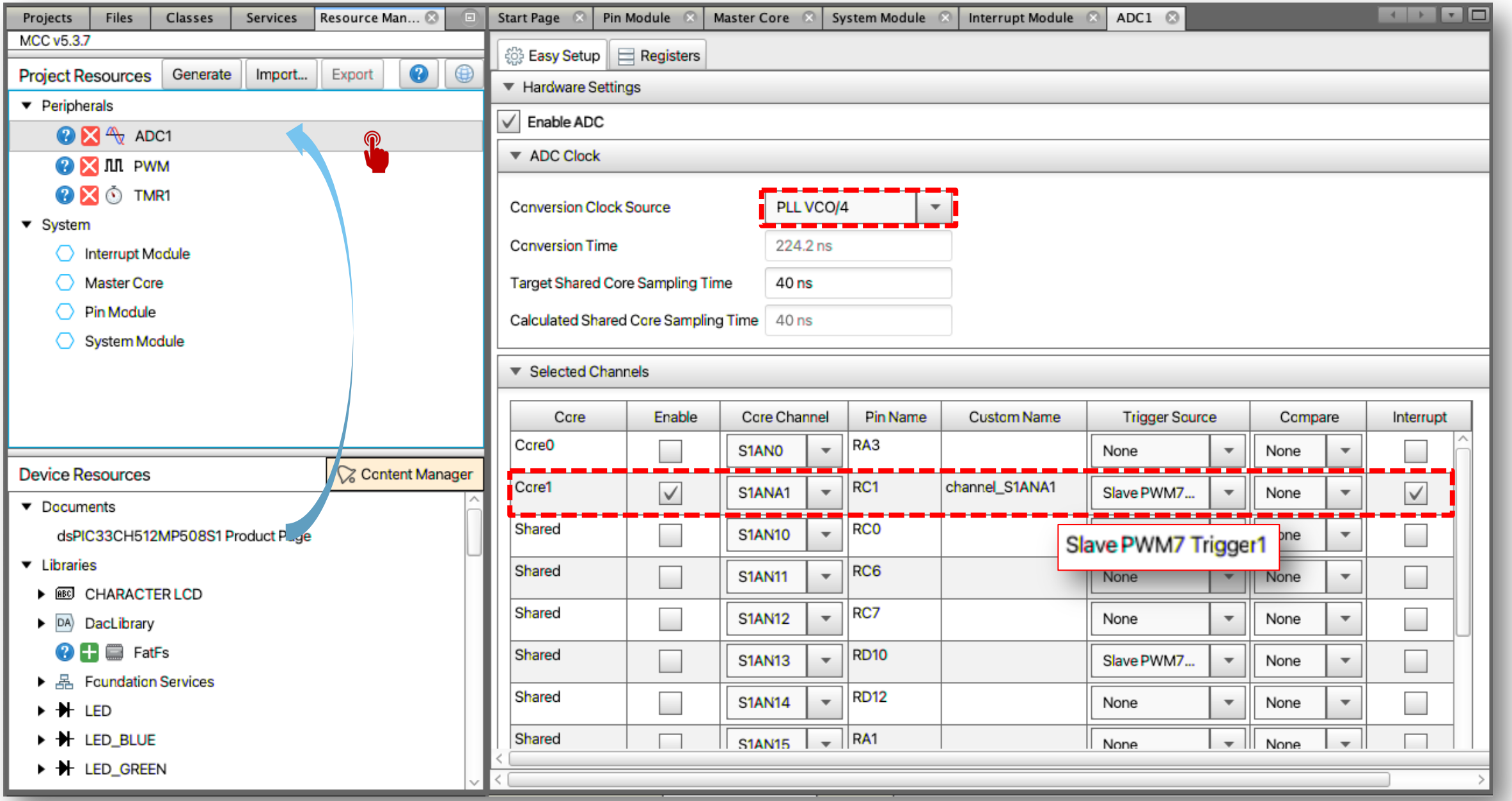
POLH: Active-low — *For P-channel*

POLL: Active-high

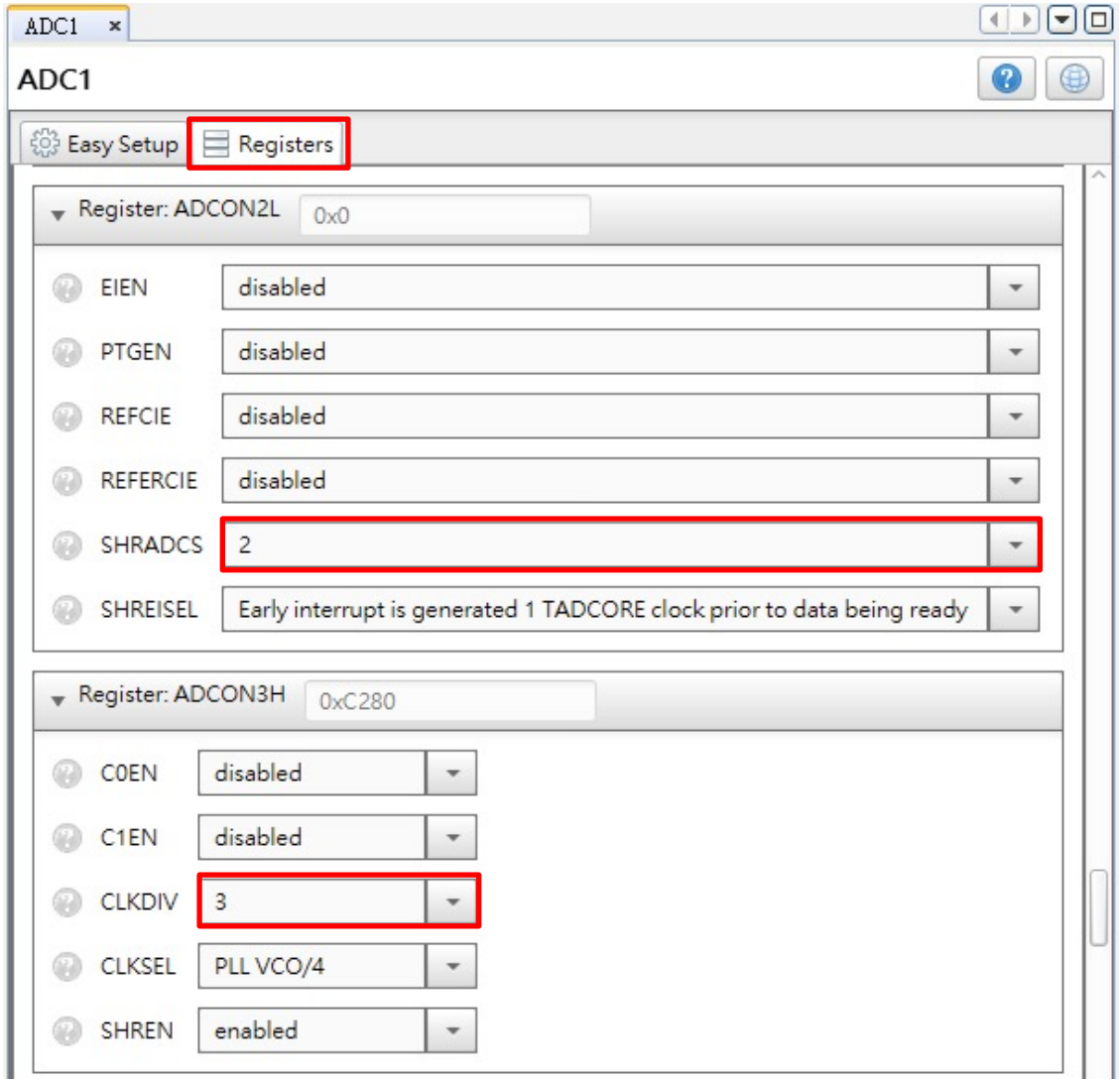
Register: PG7IOCONL 0x0

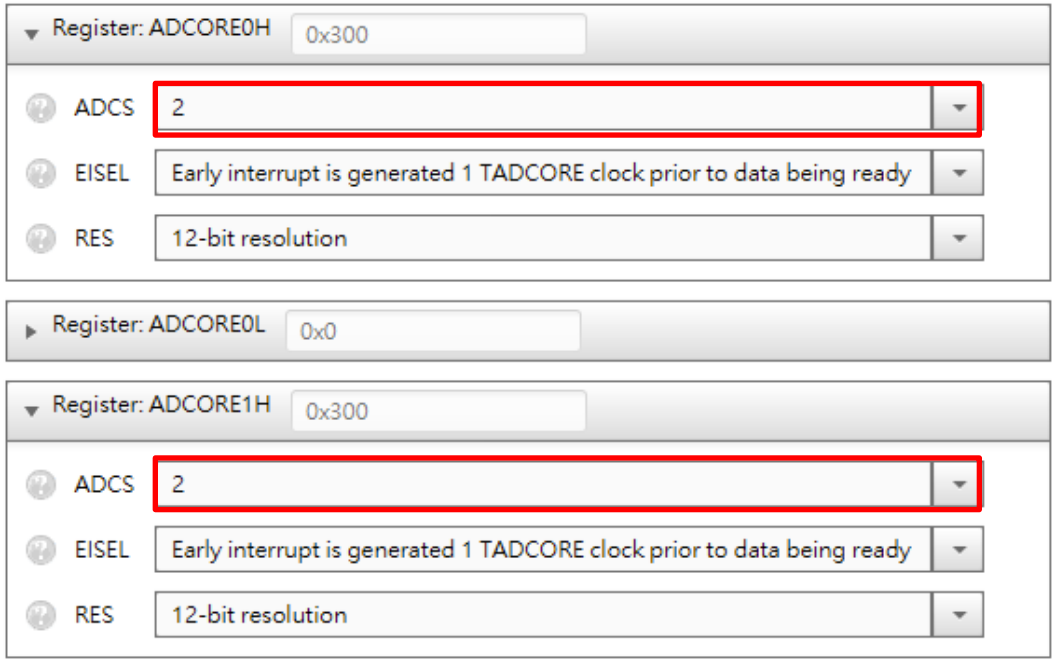
MICROCHIP

MCC Drivers – ADC/AN13 for Vout Feedback



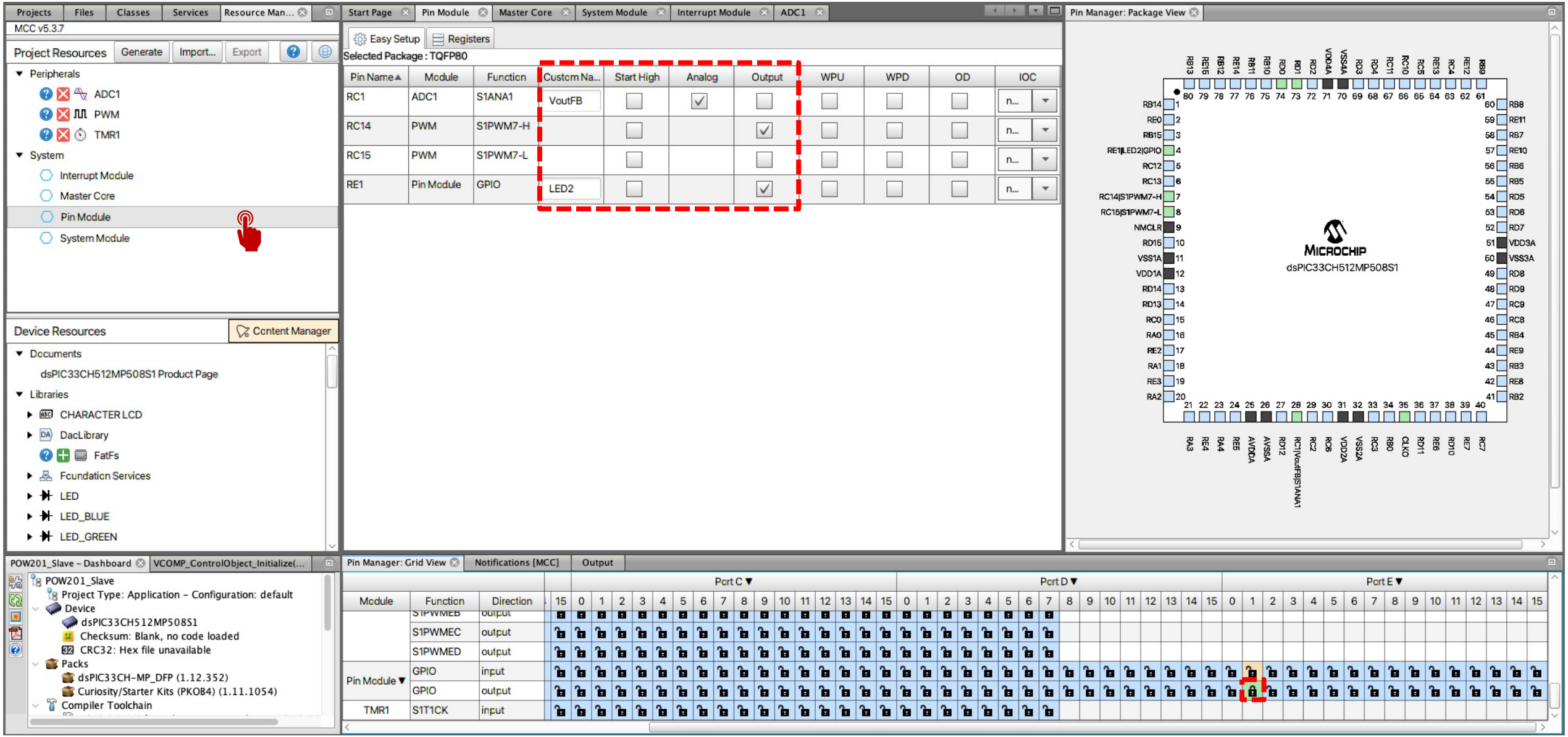
MCC Drivers – ADC/AN13 for Vout Feedback



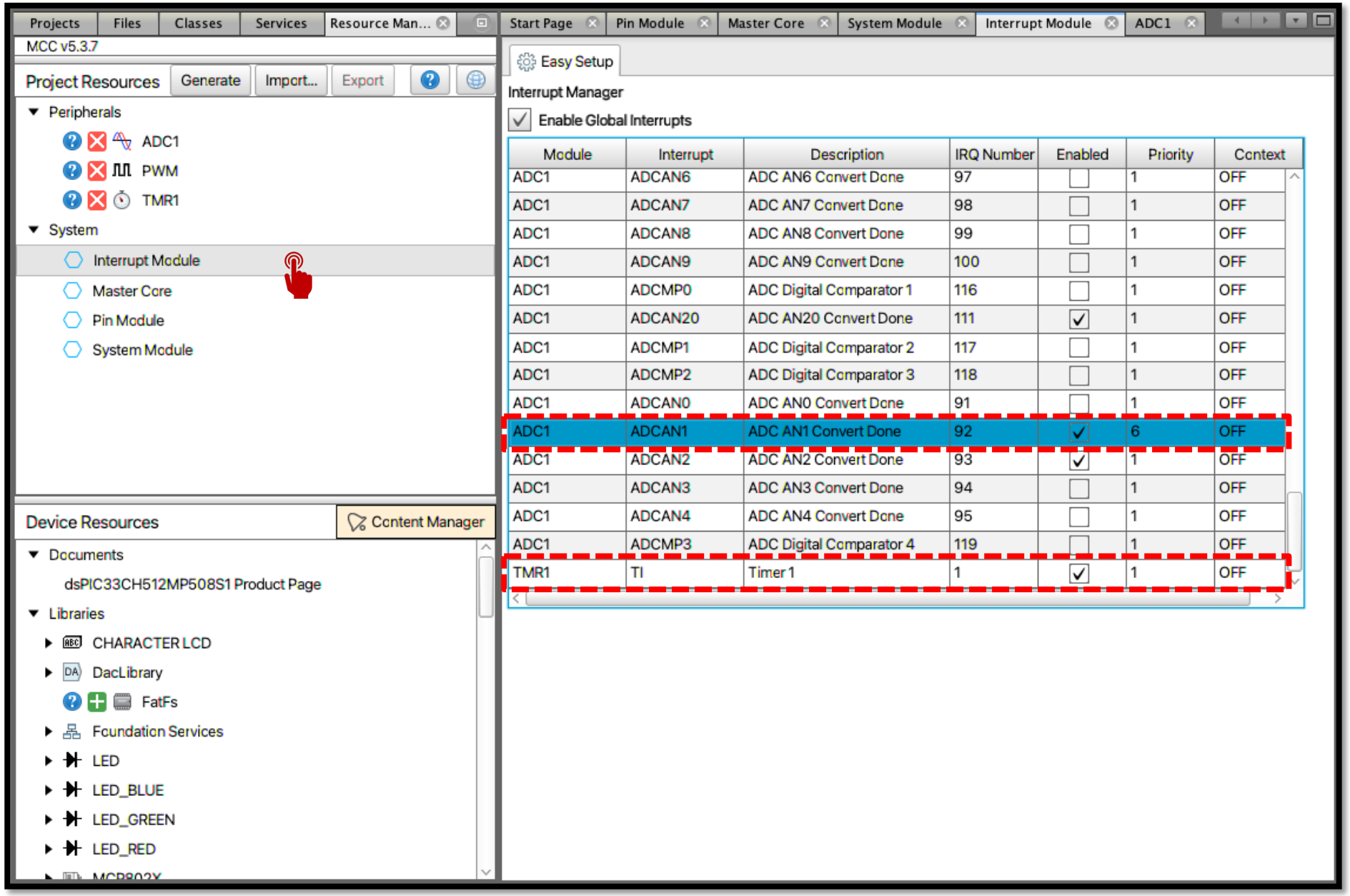


Slave - ADC Shared Core			
ADC Module Clock Source	400	MHz	PLL VOC/4
ADCON2L.SHRADCS	2		Shared ADC Core Input Clock Divider bits
ADCON3H.CLKDIV	3		ADC Module Clock Source Divider bits
Fadcore	66.67	MHz	Clock Source / SHRADCS / CLKDIV
Tadcore	15.00	ns	1/Fadcore >= 14.3ns

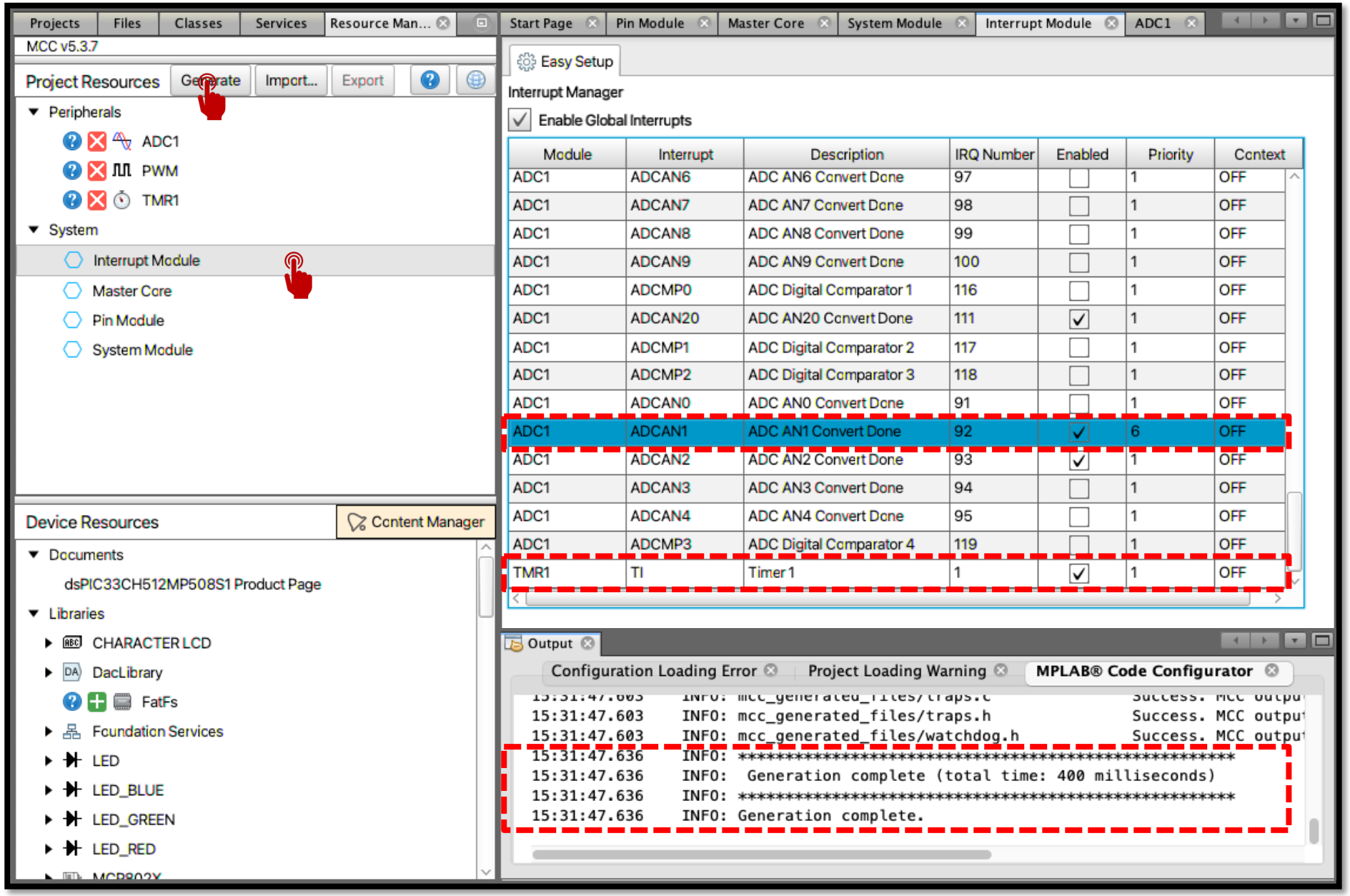
MCC Drivers – Pin Management



MCC Drivers – Interrupt Management & Code Generating

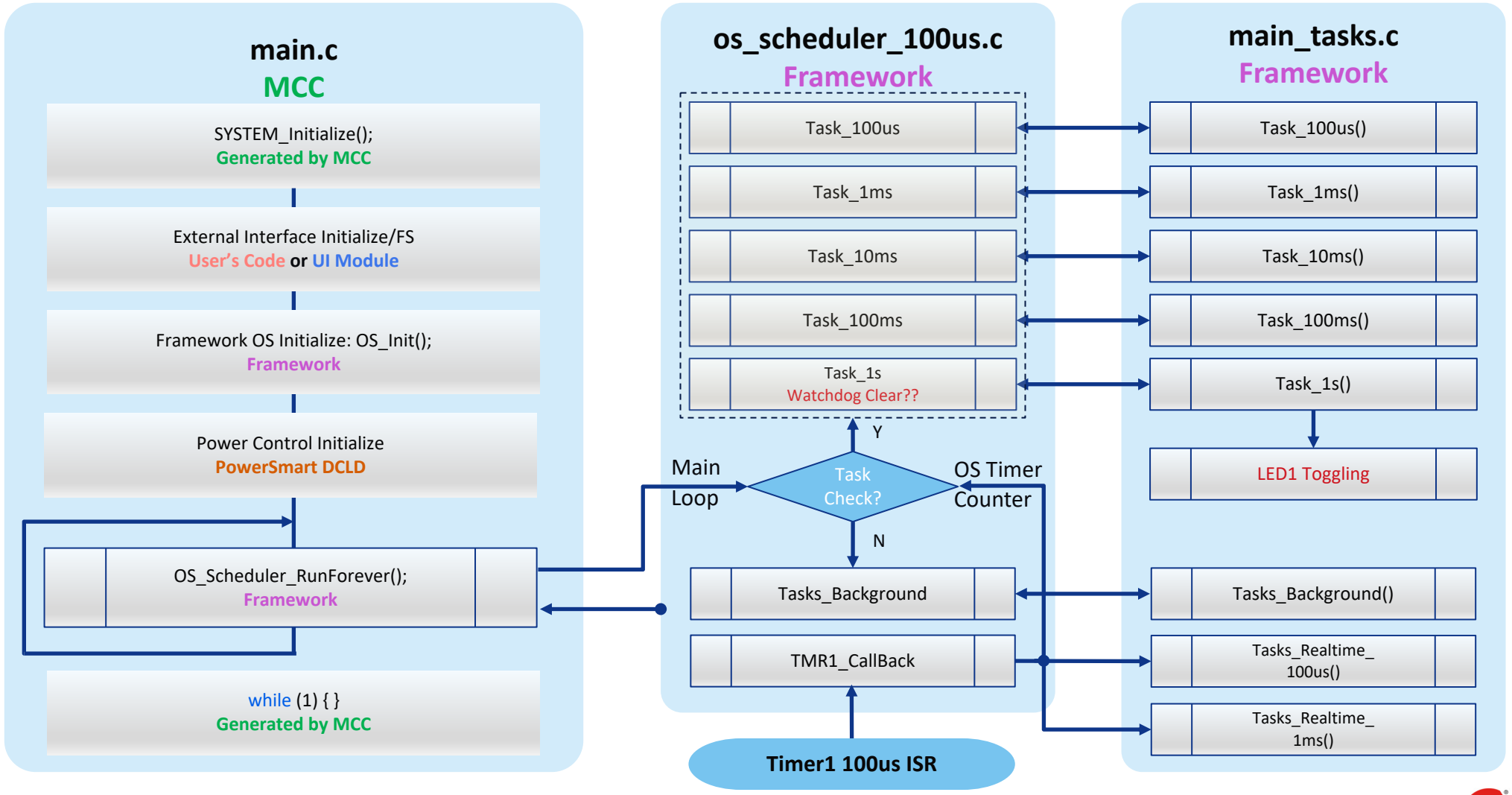


MCC Drivers – Interrupt Management & Code Generating



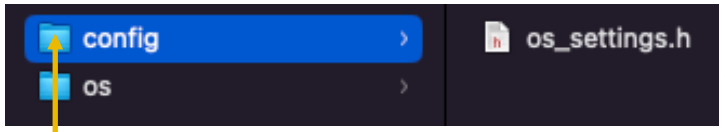
SMPS Firmware Framework

Firmware Structure

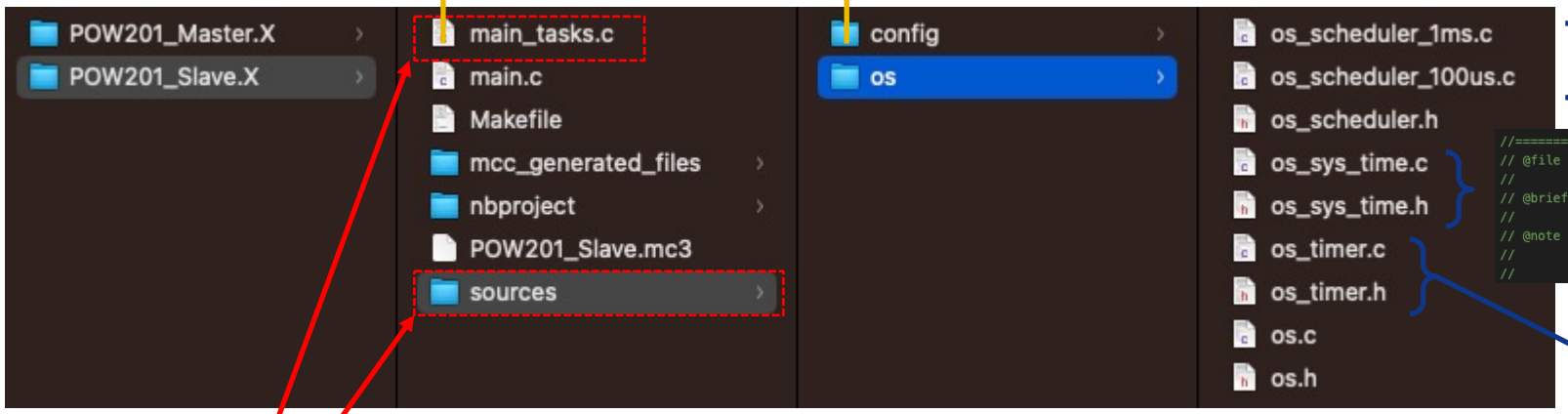


OS Scheduler for Slave

The application specific tasks that are called regularly by the os_scheduler



Project specific settings regarding the operating system parts and application



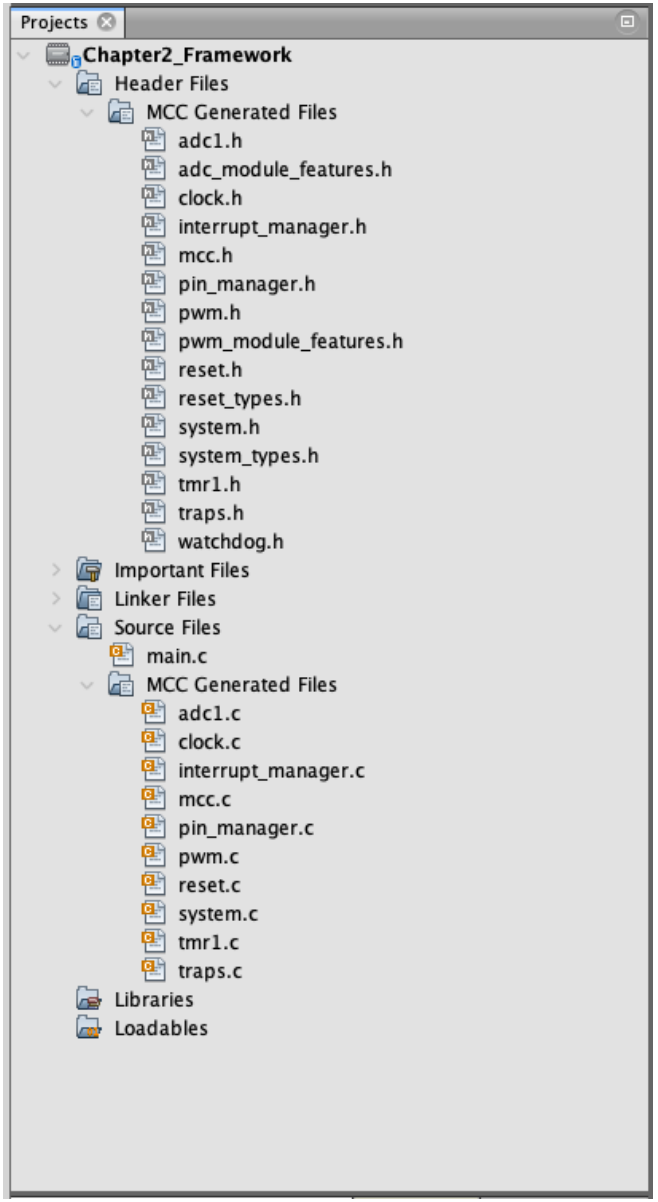
1 ms for low-end MCU such like PIC16
100 us for high speed MCU such like dsPIC33

```
//==========================================================================================
// @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
//
```

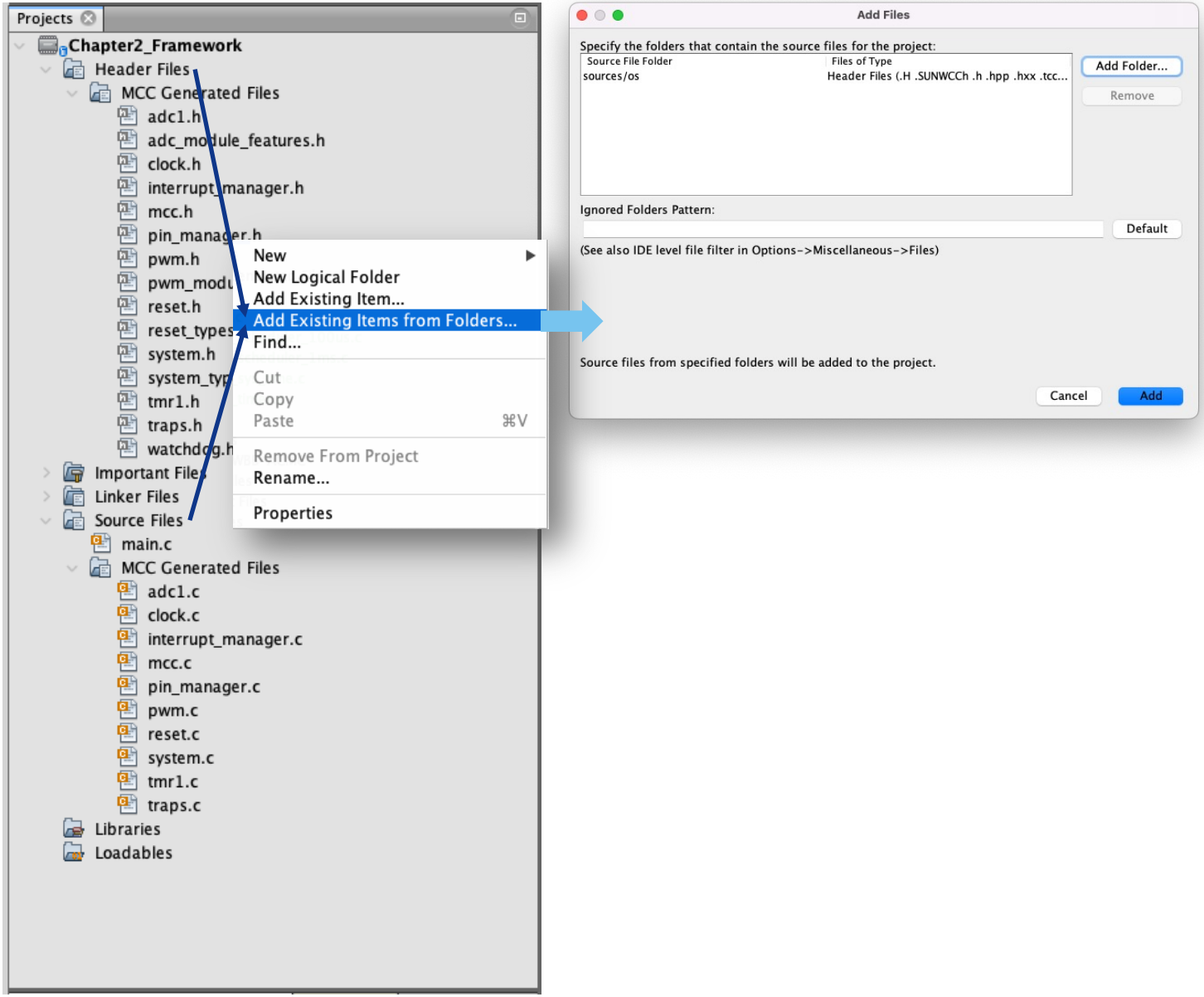
```
//==========================================================================================
// @file os_timer.c
//
// @brief contains the functions for using software timers
//
// @note    in this file you will find functions for using timers and countdowns for your application
//          These 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
//
//
```

此檔案與目錄皆可在Lab Answer中對應的Lab目錄中找到。

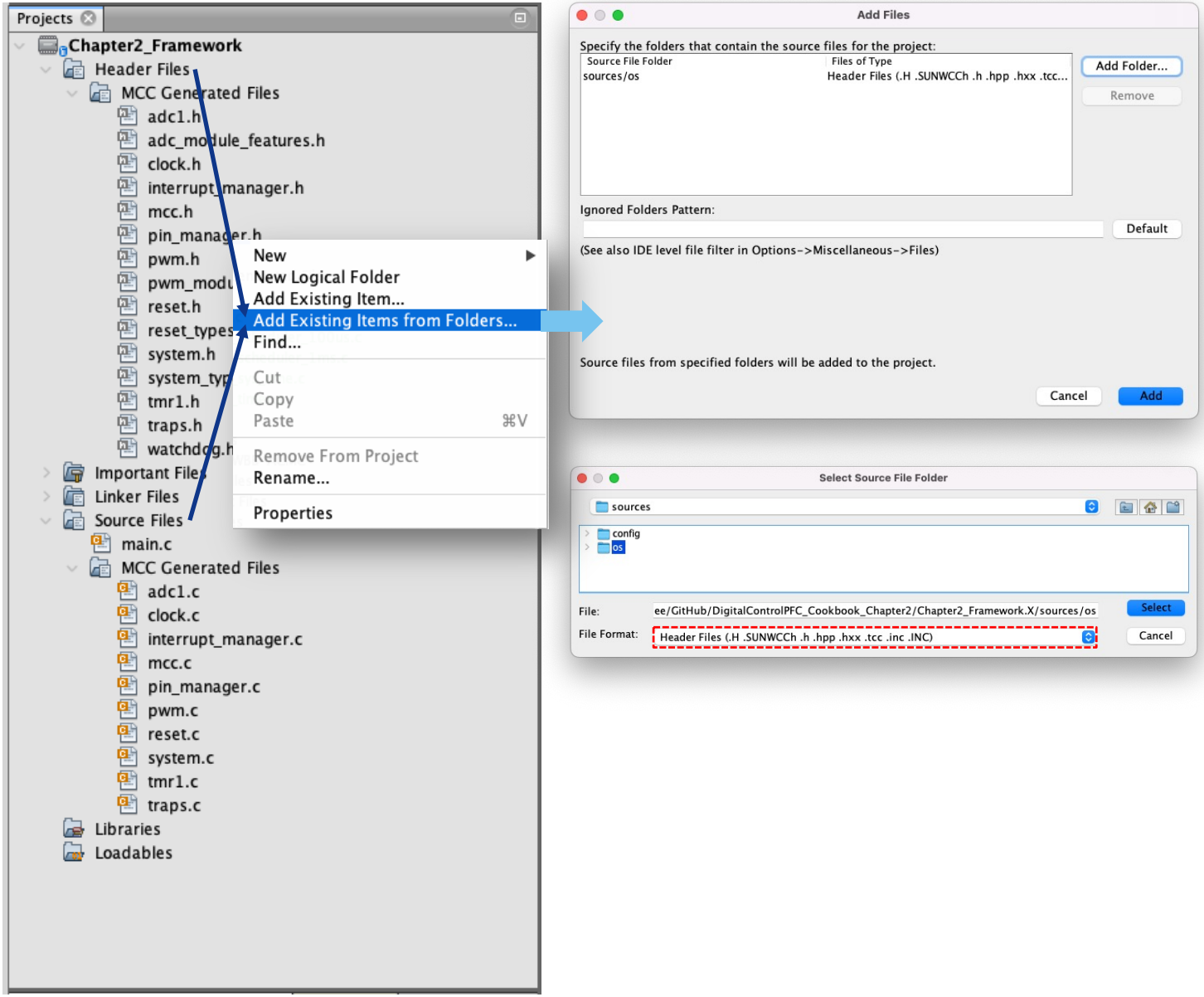
Import OS_Scheduler



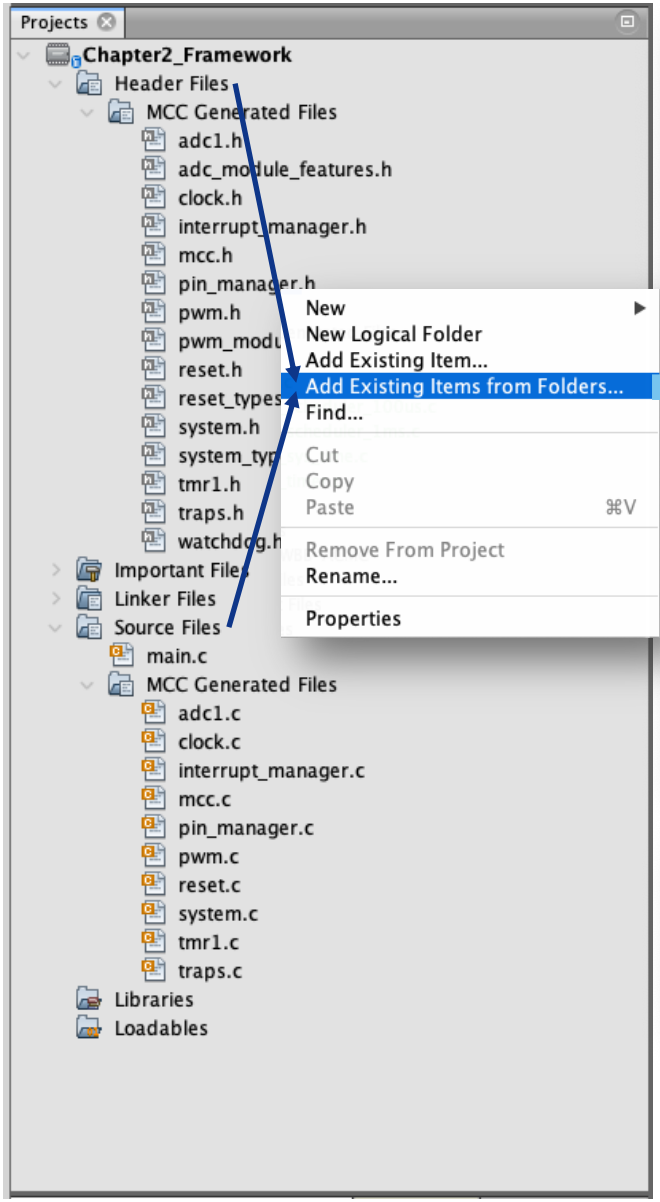
Import OS_Scheduler

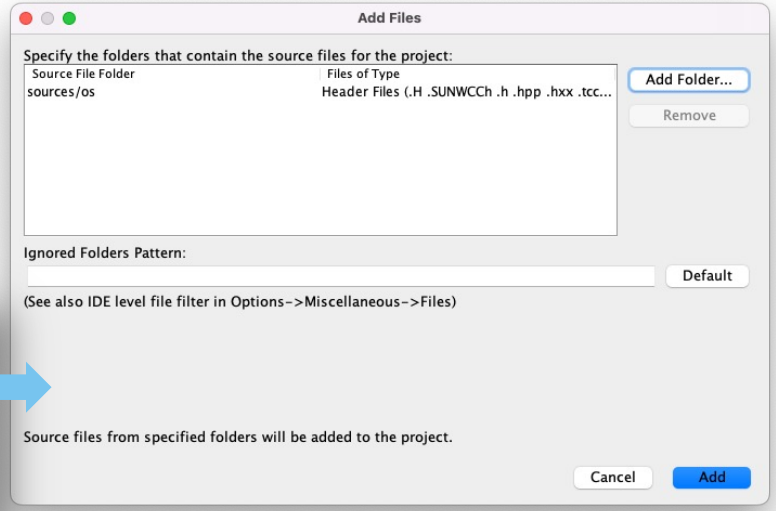


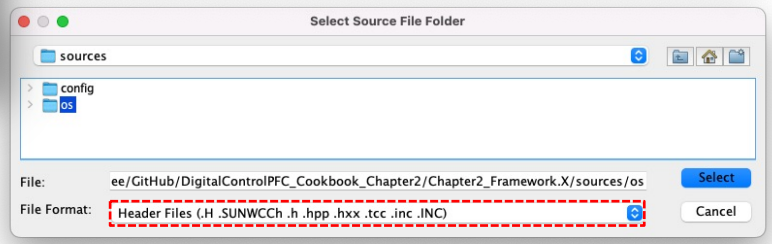
Import OS_Scheduler

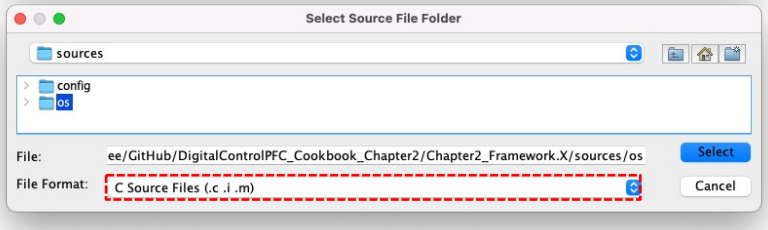


Import OS_Scheduler

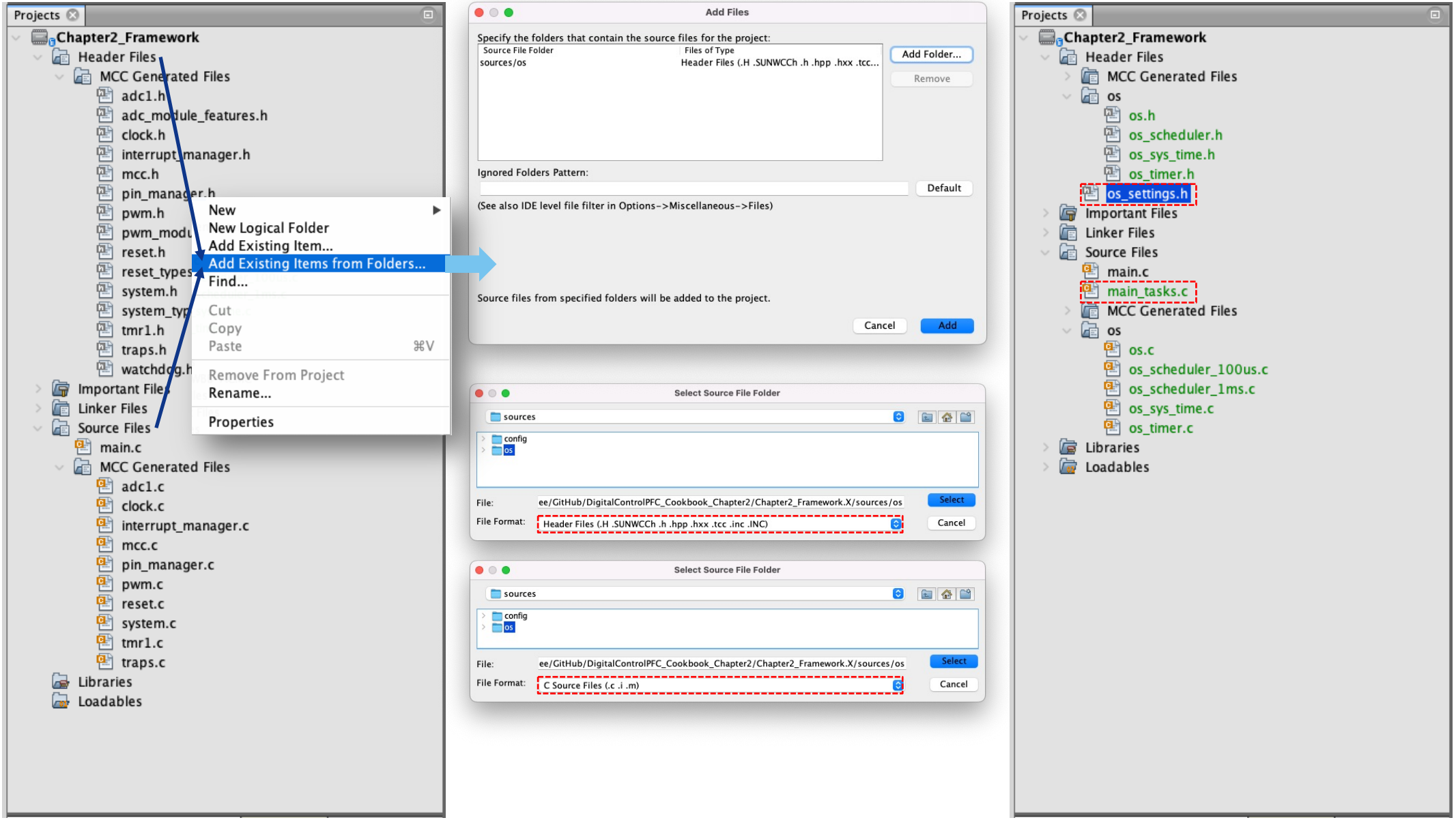








Import OS_Scheduler



Main & Tasks

main.c

Source | History

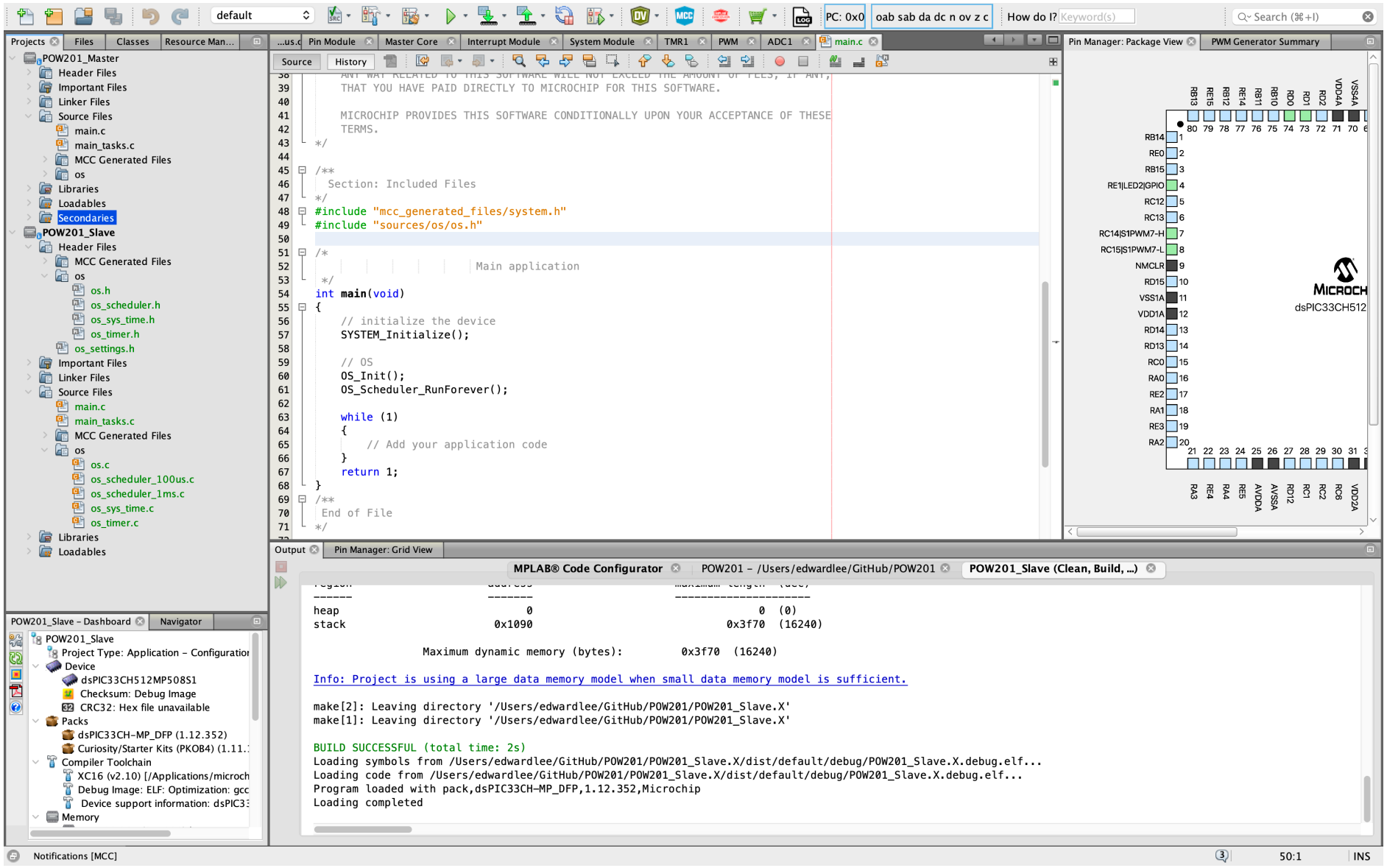
```
47  */
48  #include "mcc_generated_files/system.h"
49  #include "sources/os/os.h"
50
51  /*
52                          Main application
53   */
54  int main(void)
55  {
56      // initialize the device
57      SYSTEM_Initialize();
58
59      // OS
60      OS_Init();
61      OS_Scheduler_RunForever();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

main_tasks.c

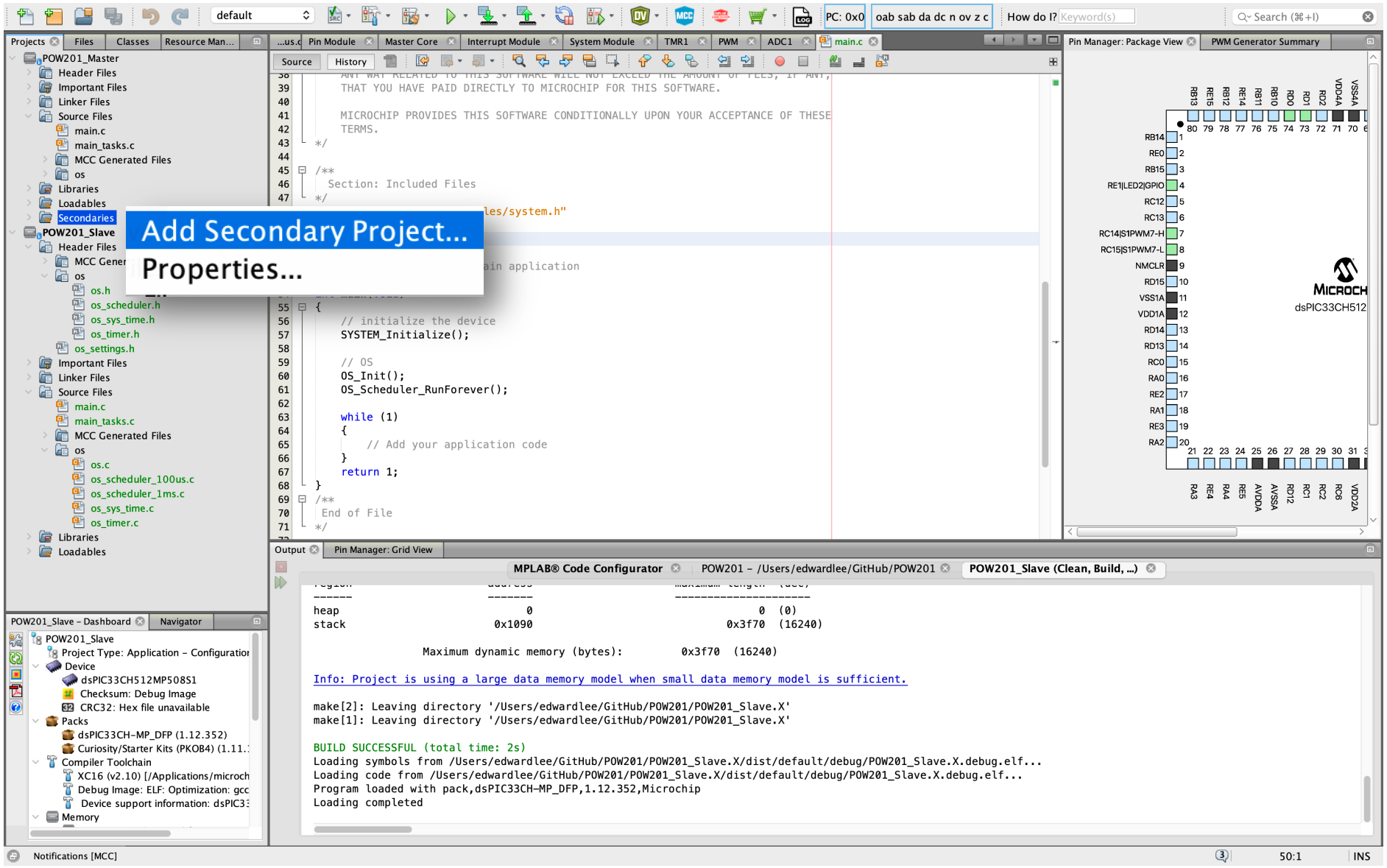
Source | History

```
127
128 //=============================================================================
129 //  @brief  Tasks_1s gets called every second, put your things in it that need to be 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     LED2_Toggle();
137 }
```

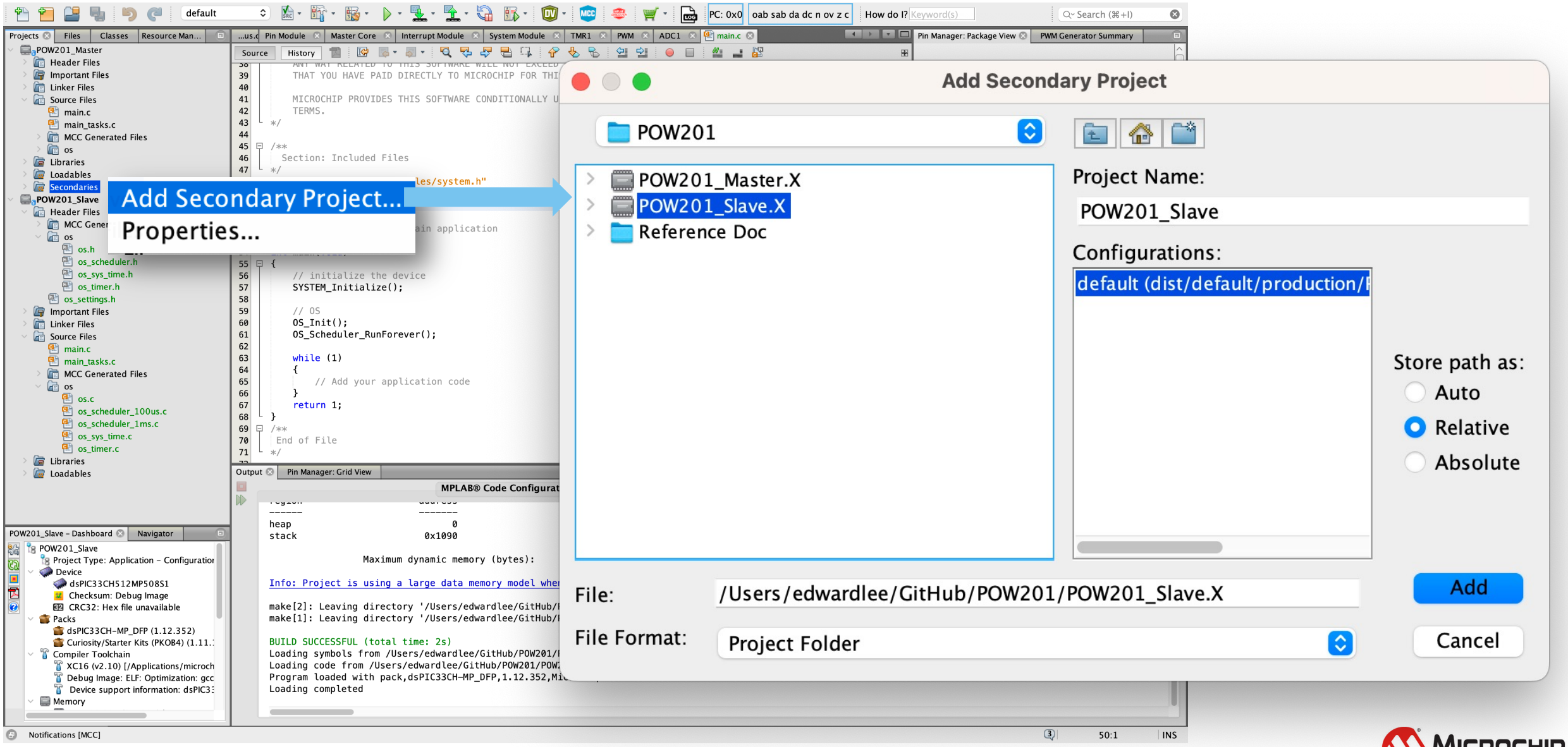
Back to Master and Add Secondary Project



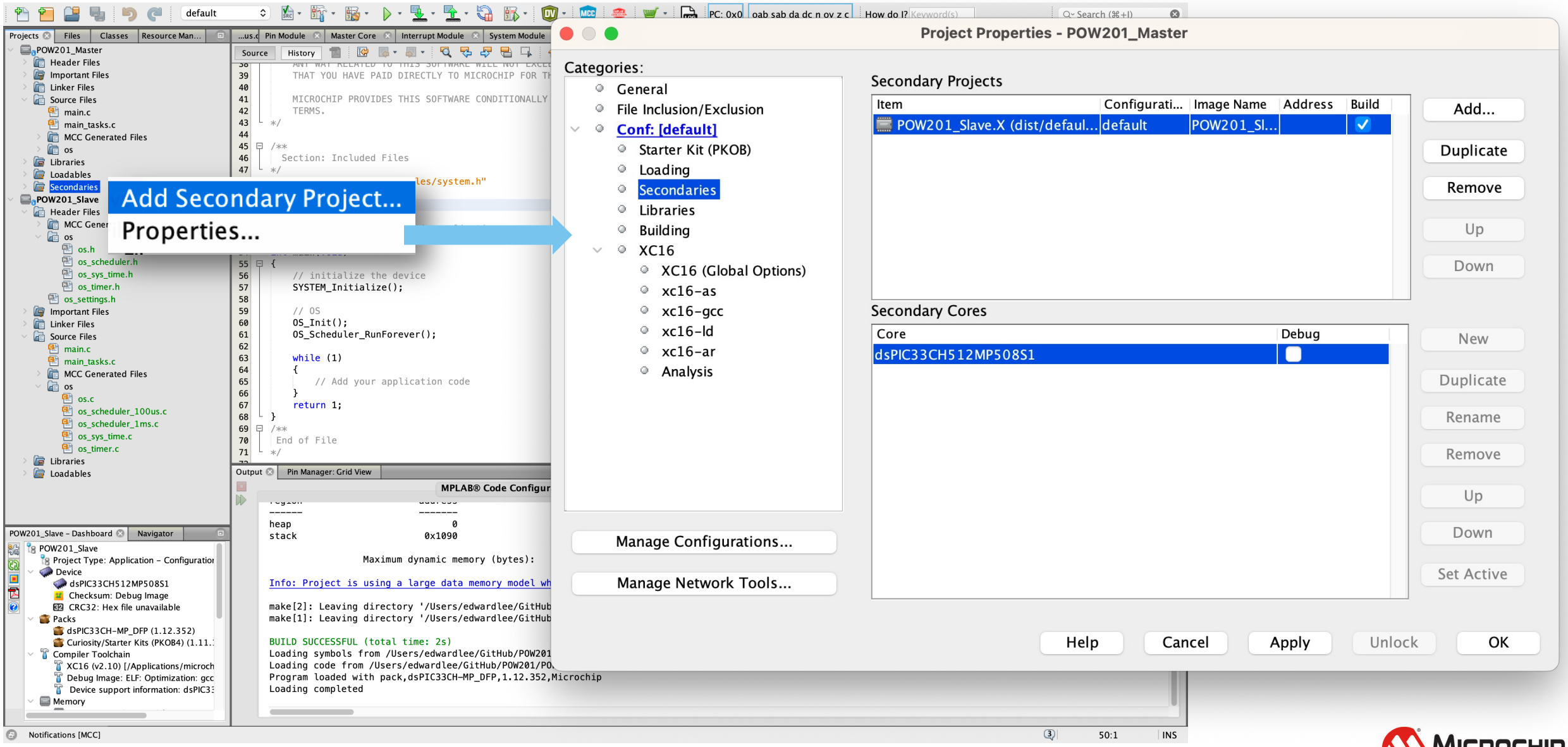
Back to Master and Add Secondary Project



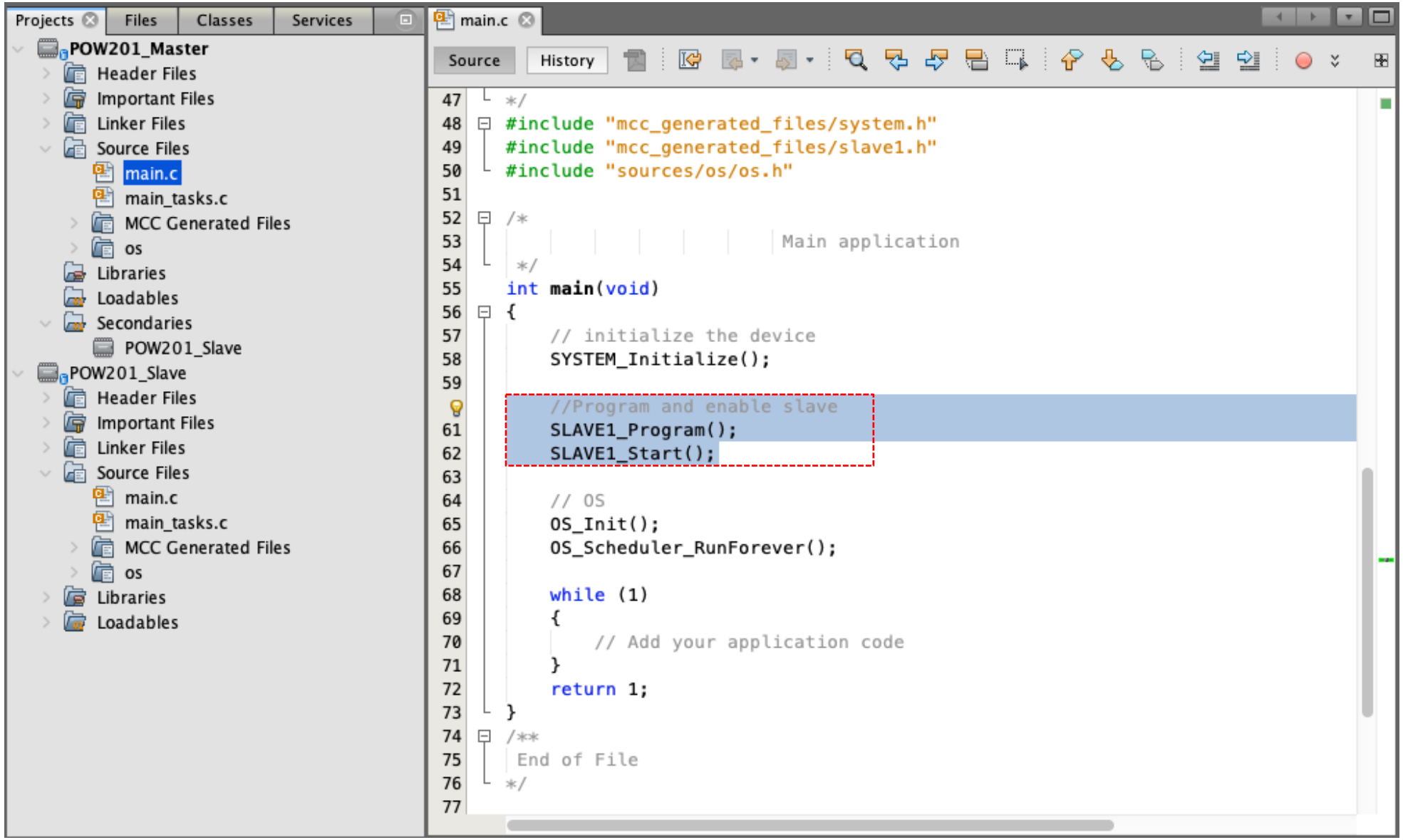
Back to Master and Add Secondary Project



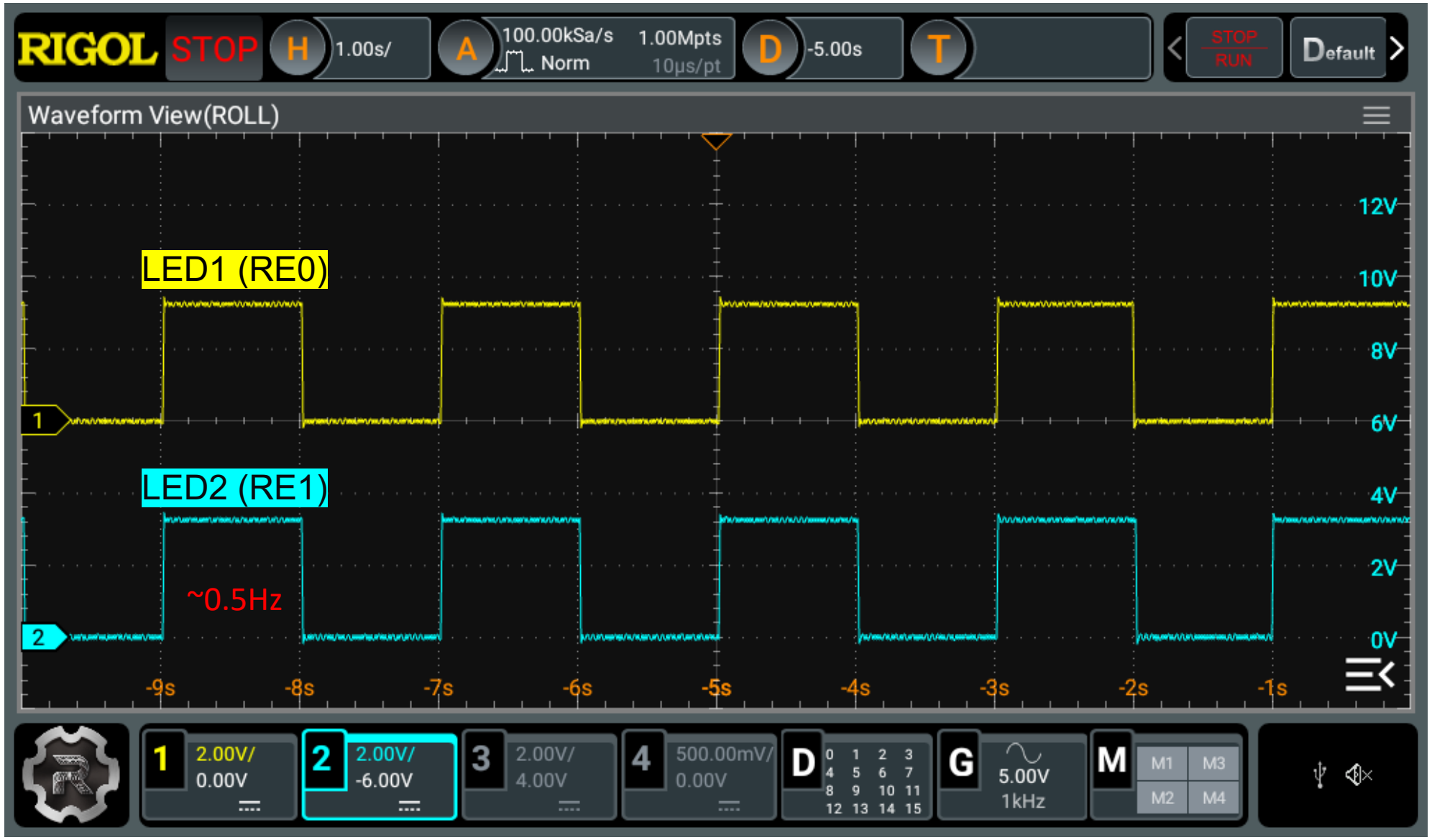
Back to Master and Add Secondary Project



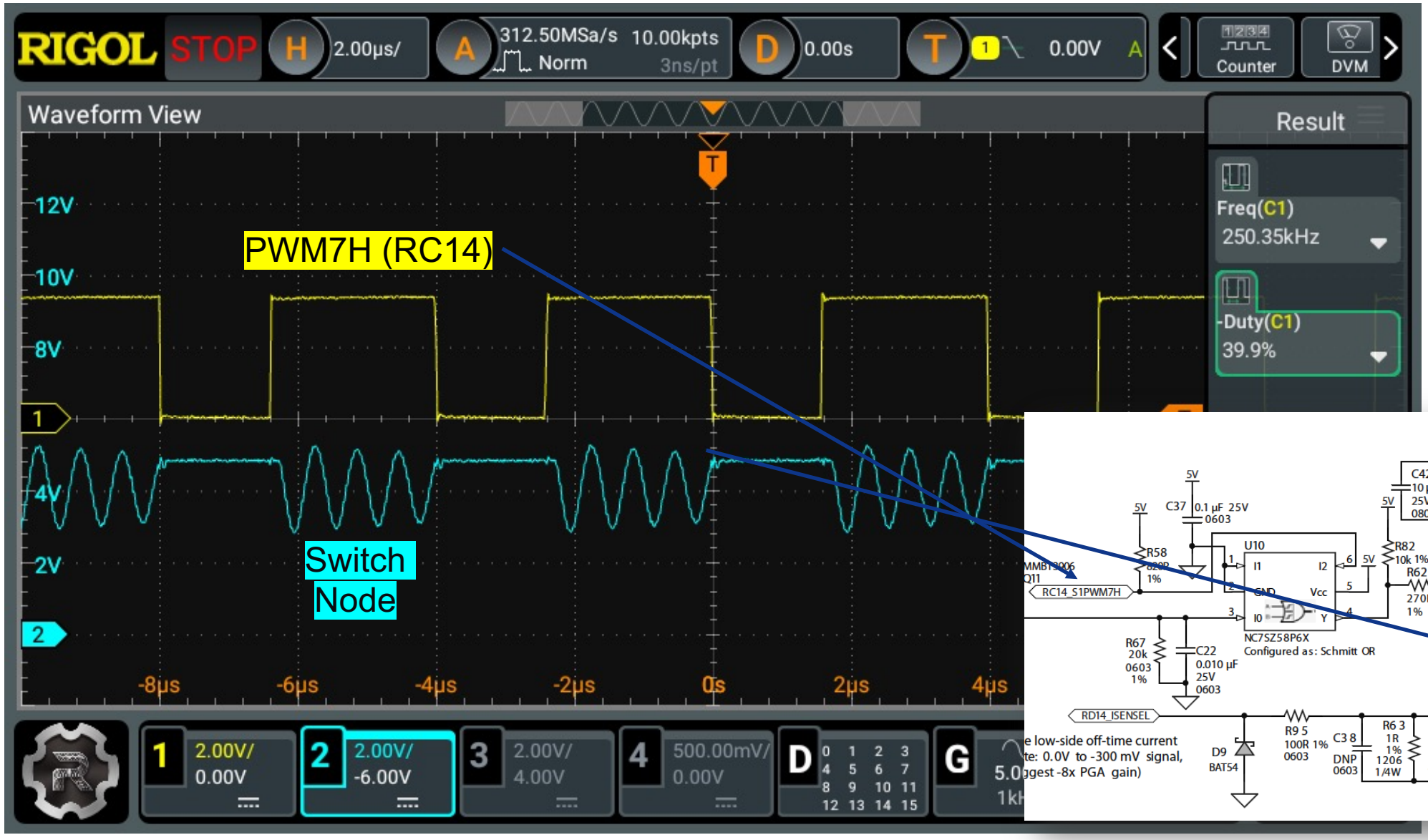
Back to Master and “Program&Start” Slave Core



0.5 Hz LED1 & LED2 Toggle



PWM7H (RC14) - 40% Duty (Active Low)



~250kHz

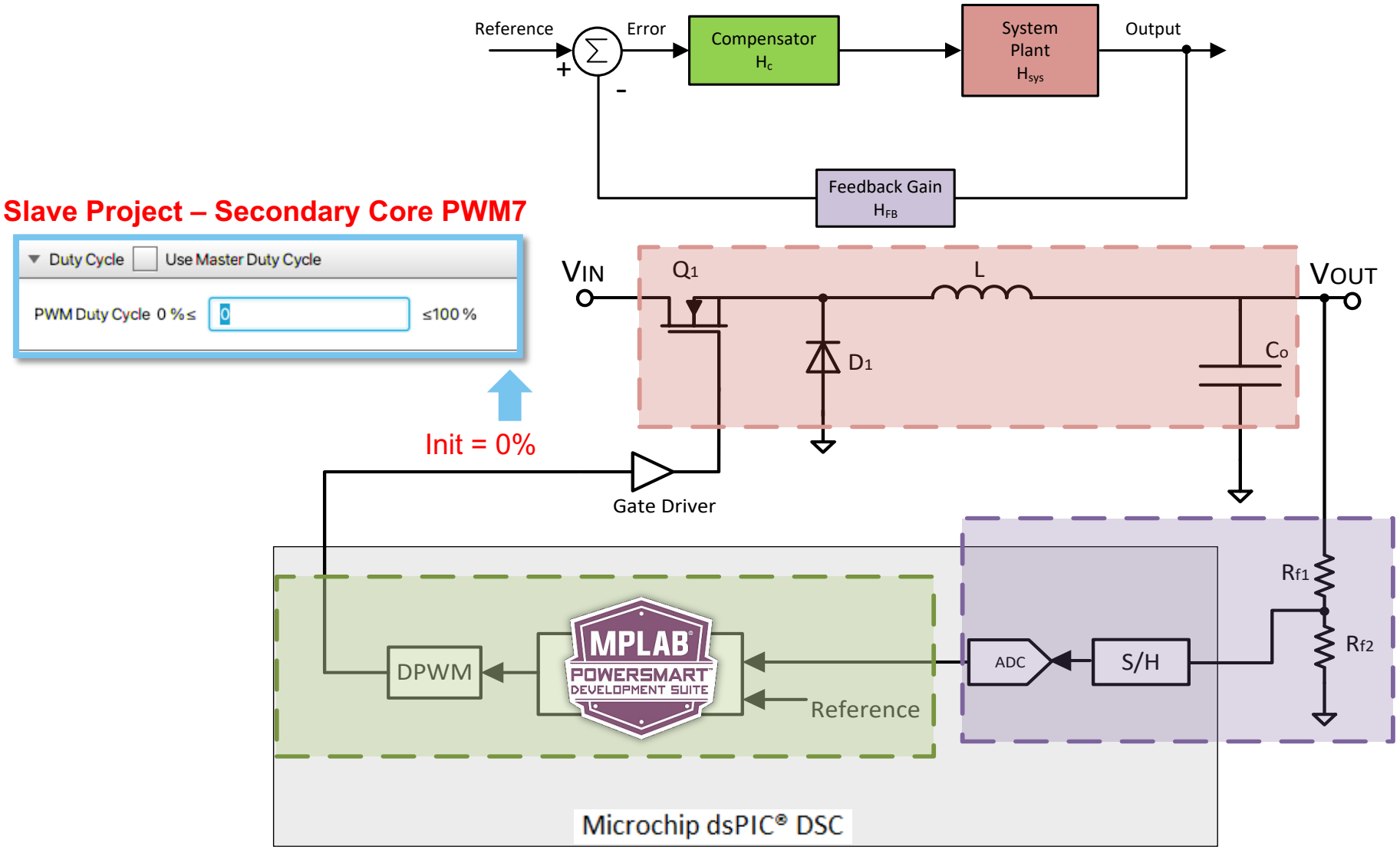
~40% DC

Digital SMPS Exercises – Part-II

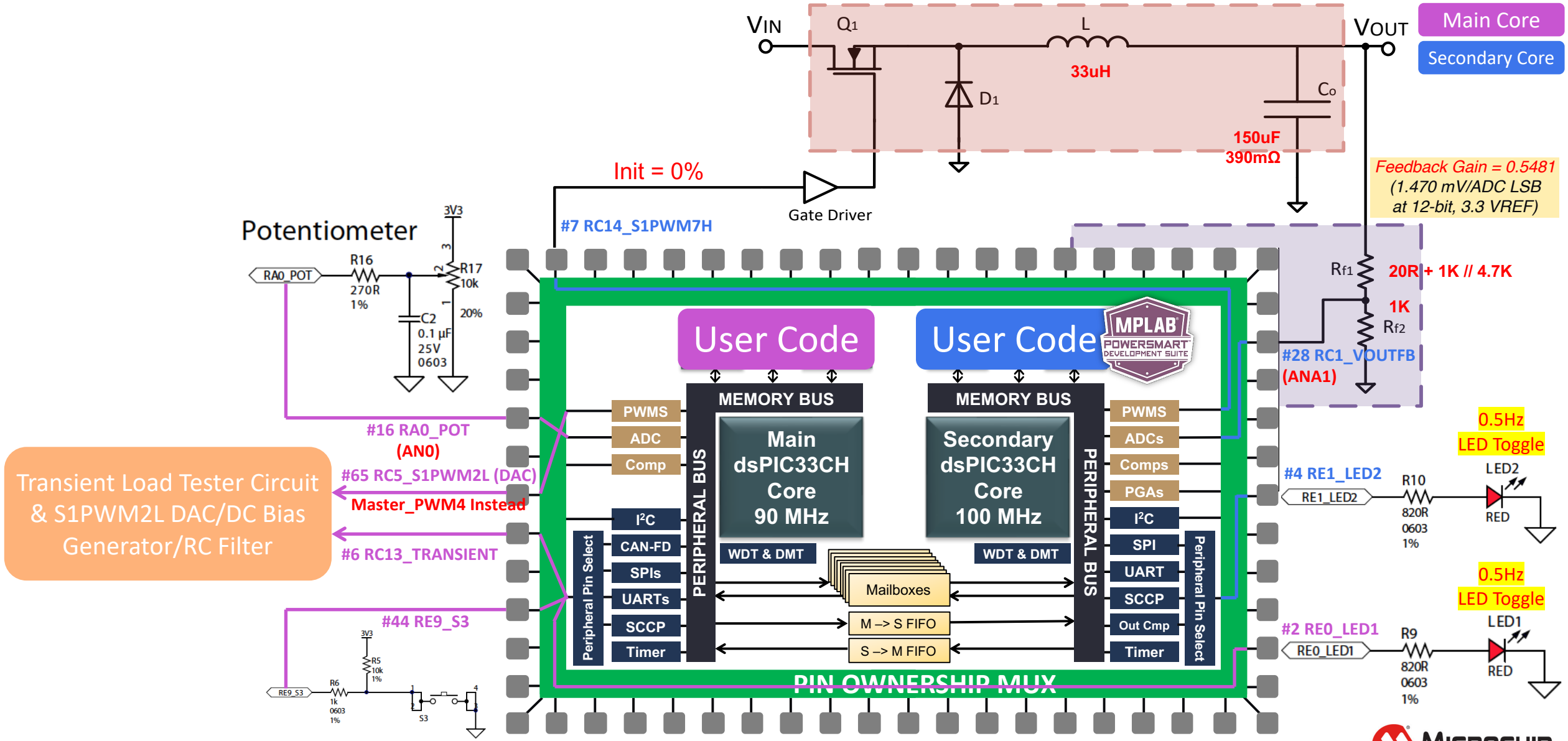
Plant Measurement

Lab3

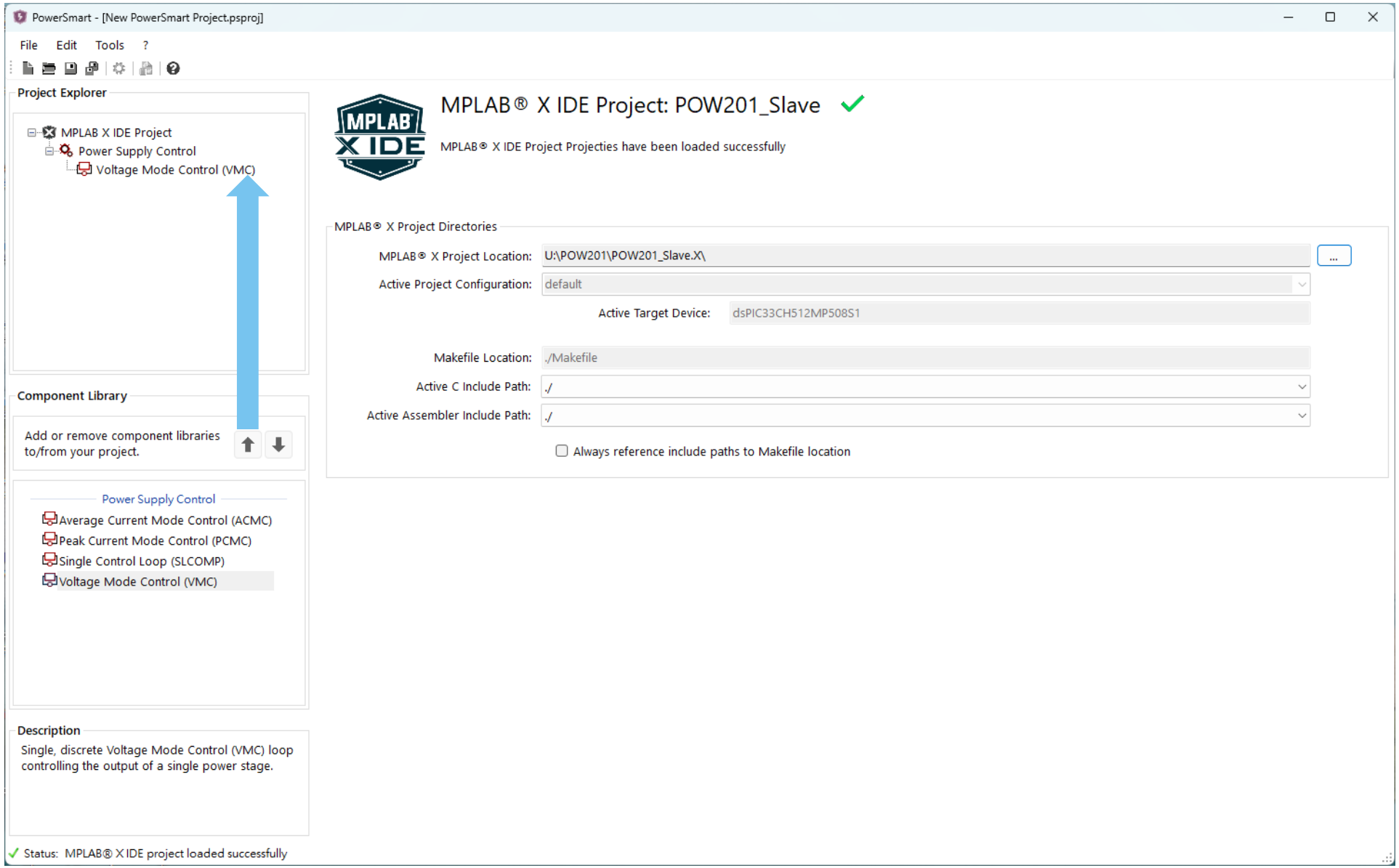
Lab #3: Plant Measurement Using PowerSmart™



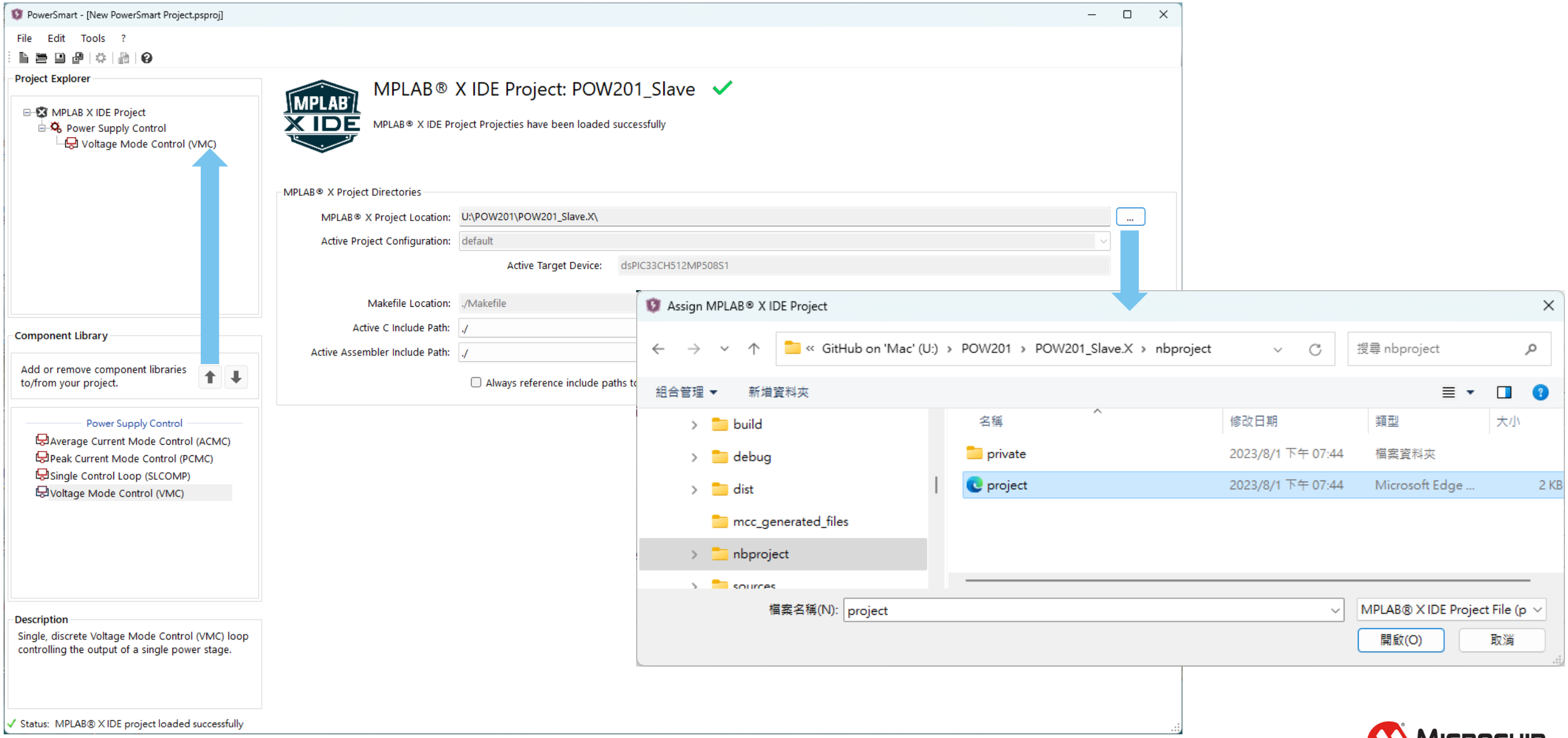
Lab #3: Plant Measurement Using PowerSmart™



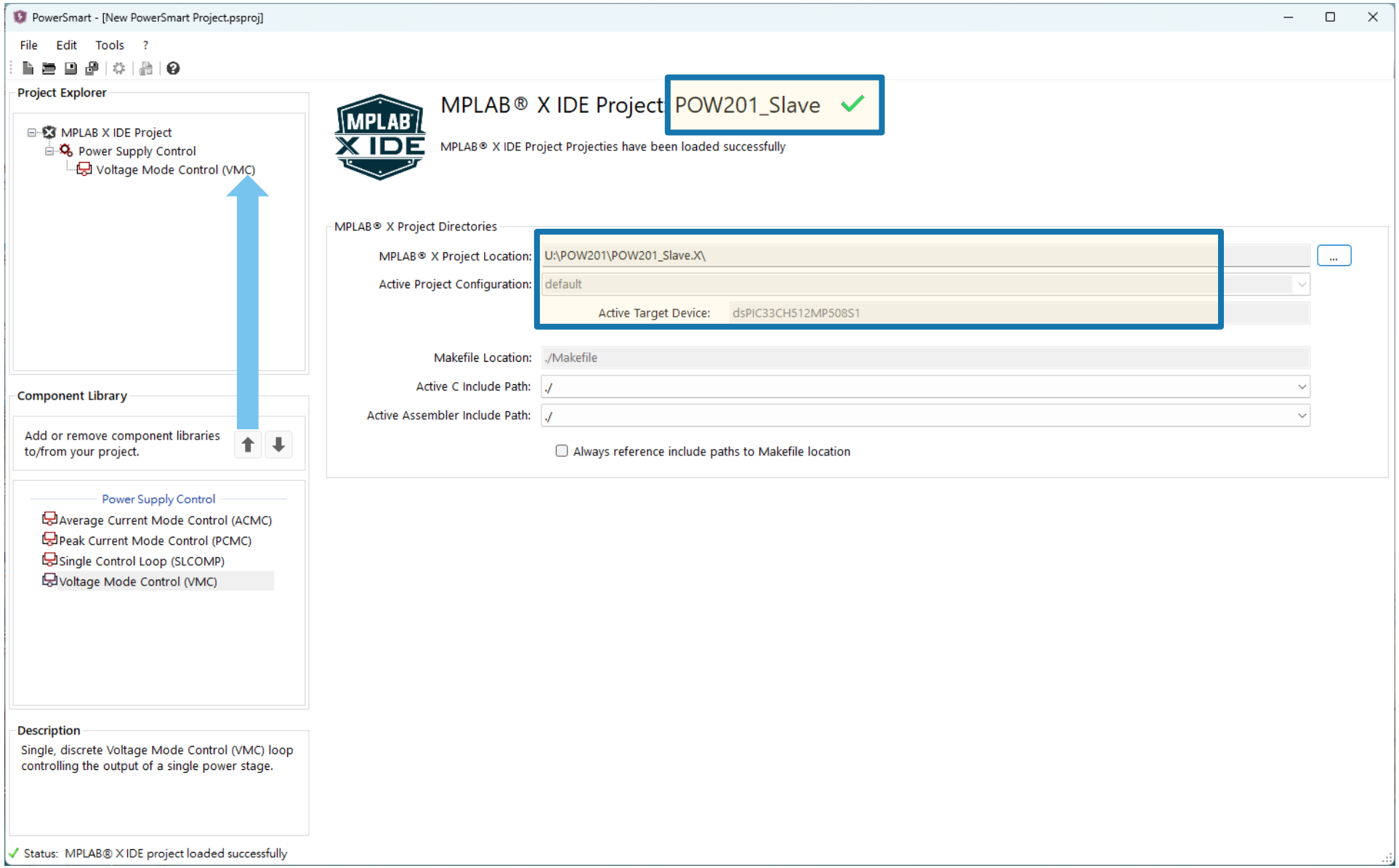
Add Control Mode for the Slave Project



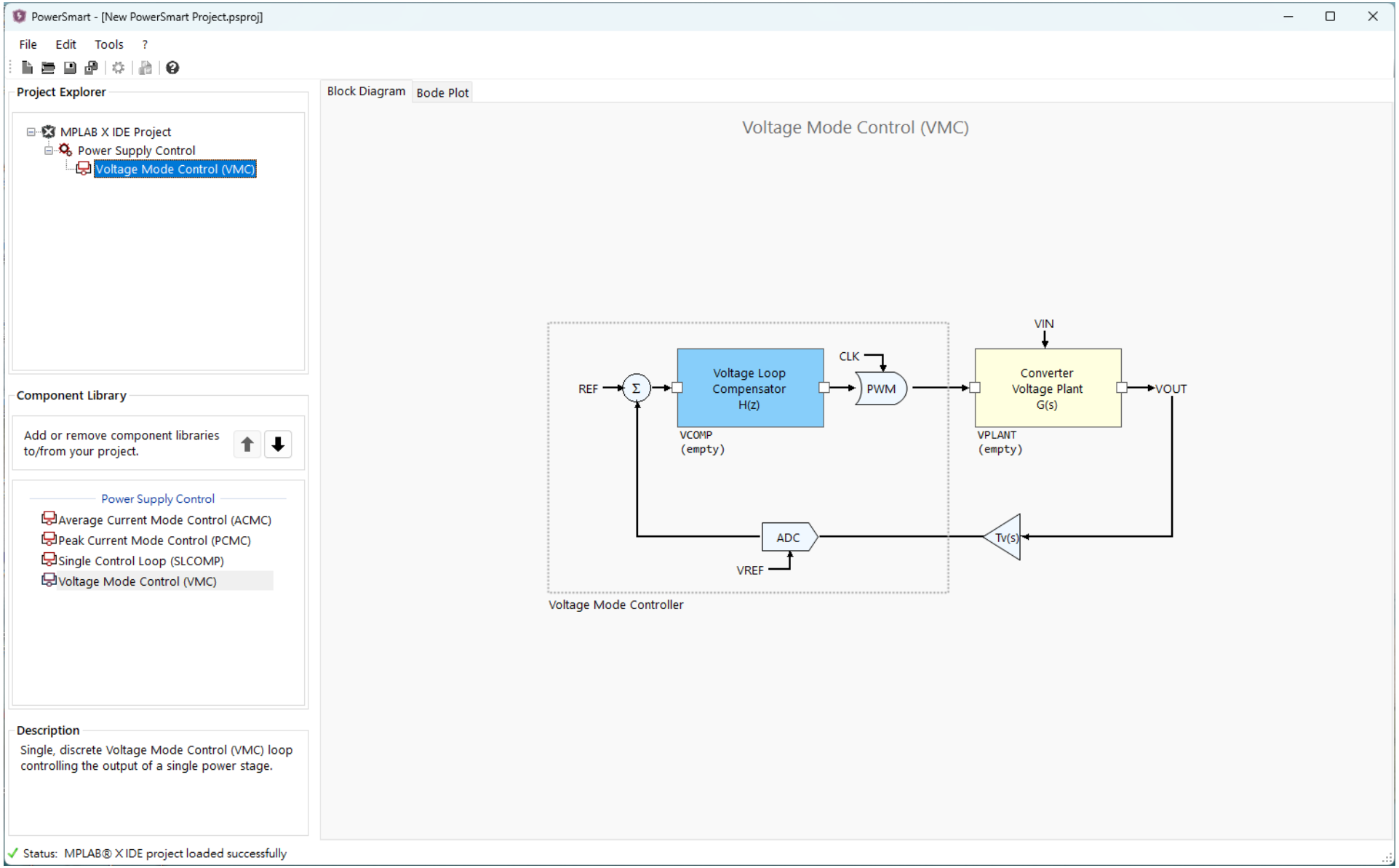
Add Control Mode for the Slave Project



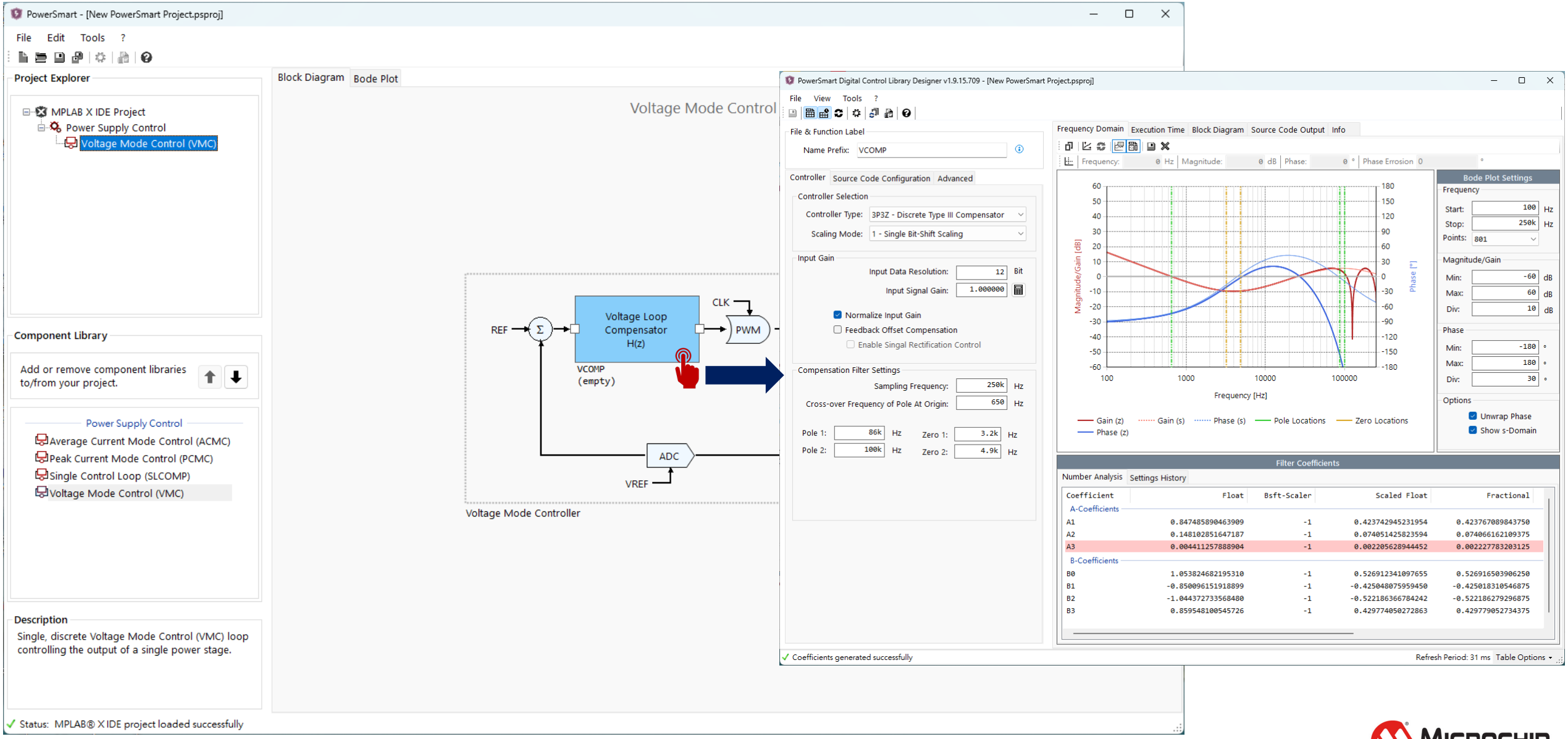
Add Control Mode for the Slave Project



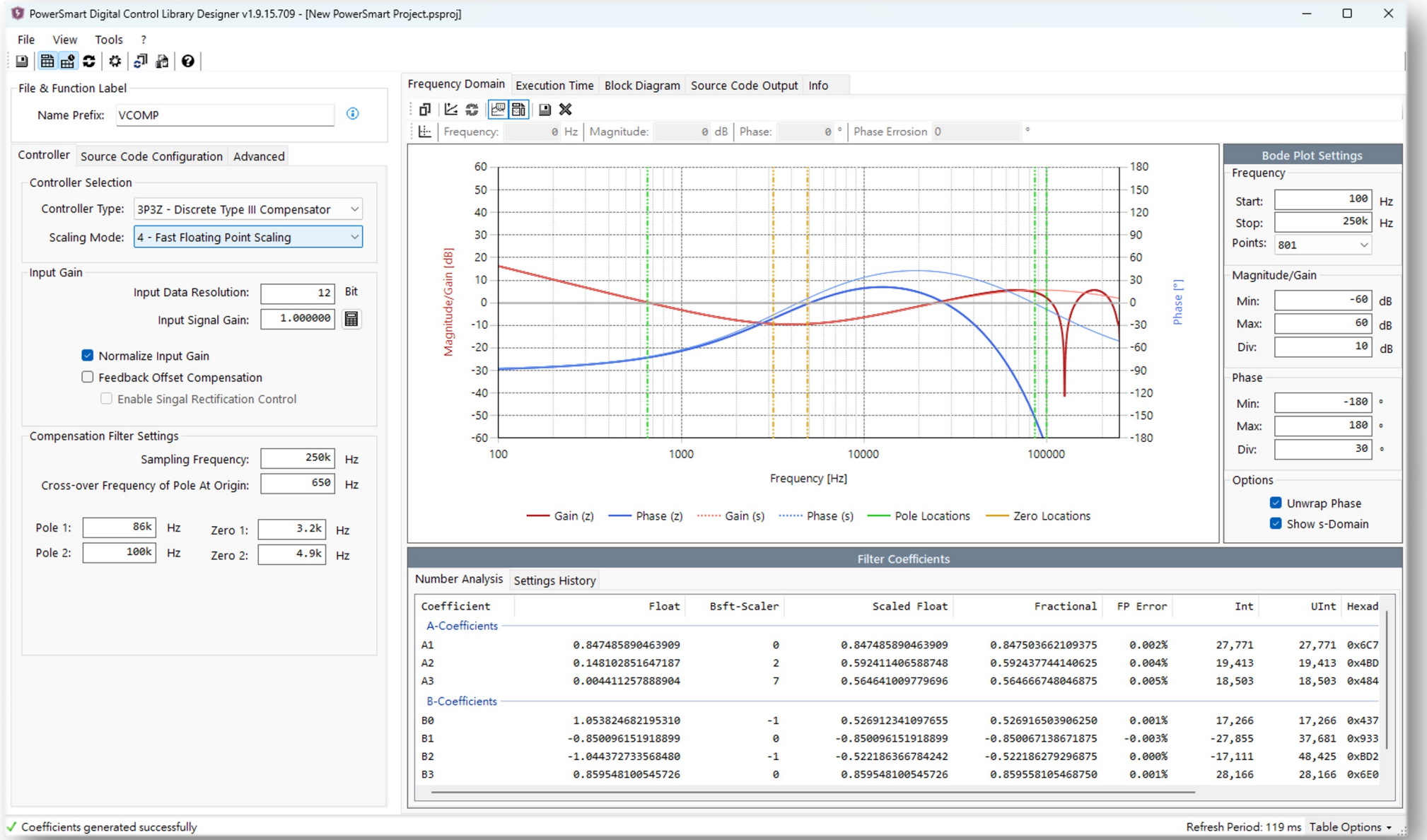
Switch to DCLD Window



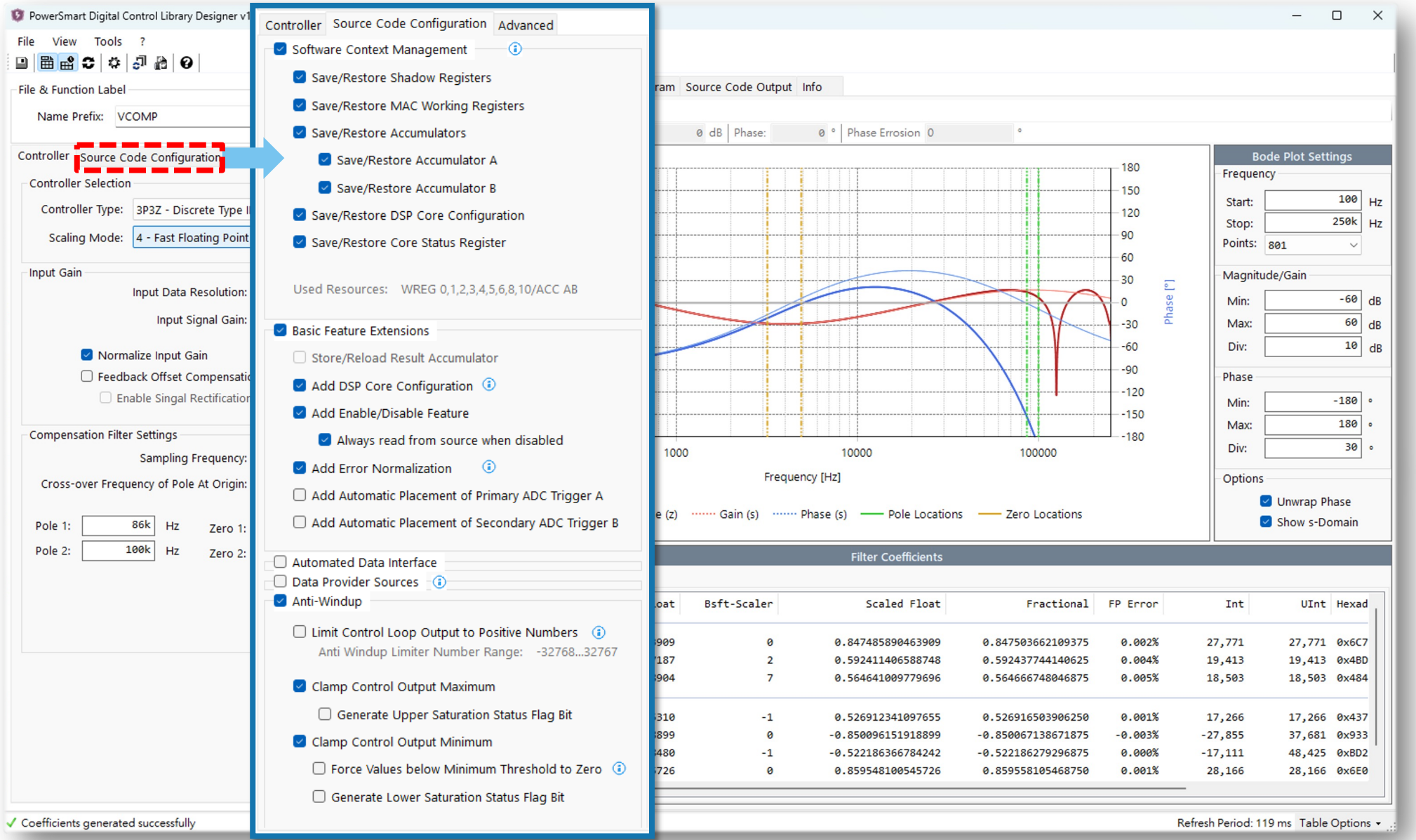
Switch to DCLD Window



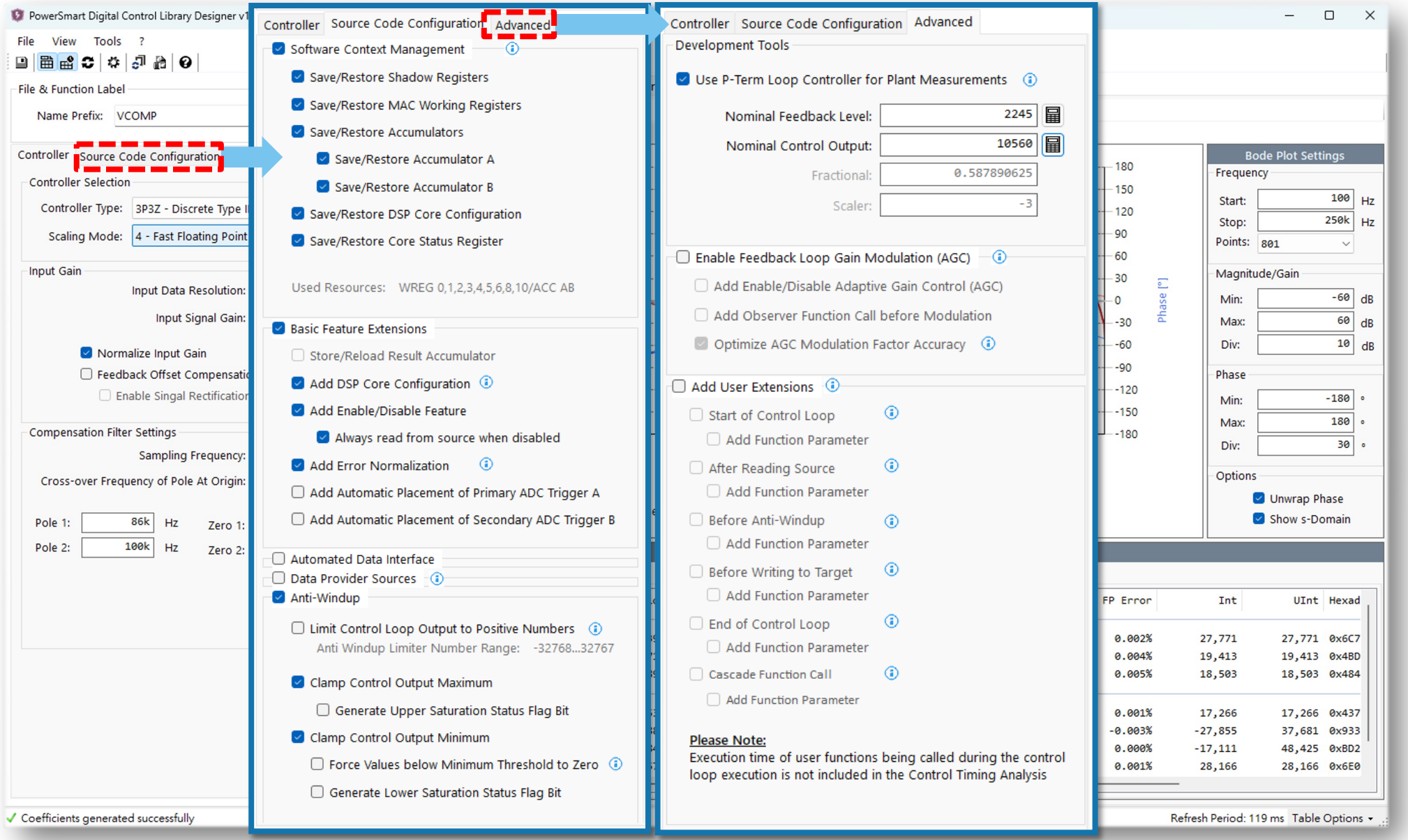
Config K_p (P-Term) Gain



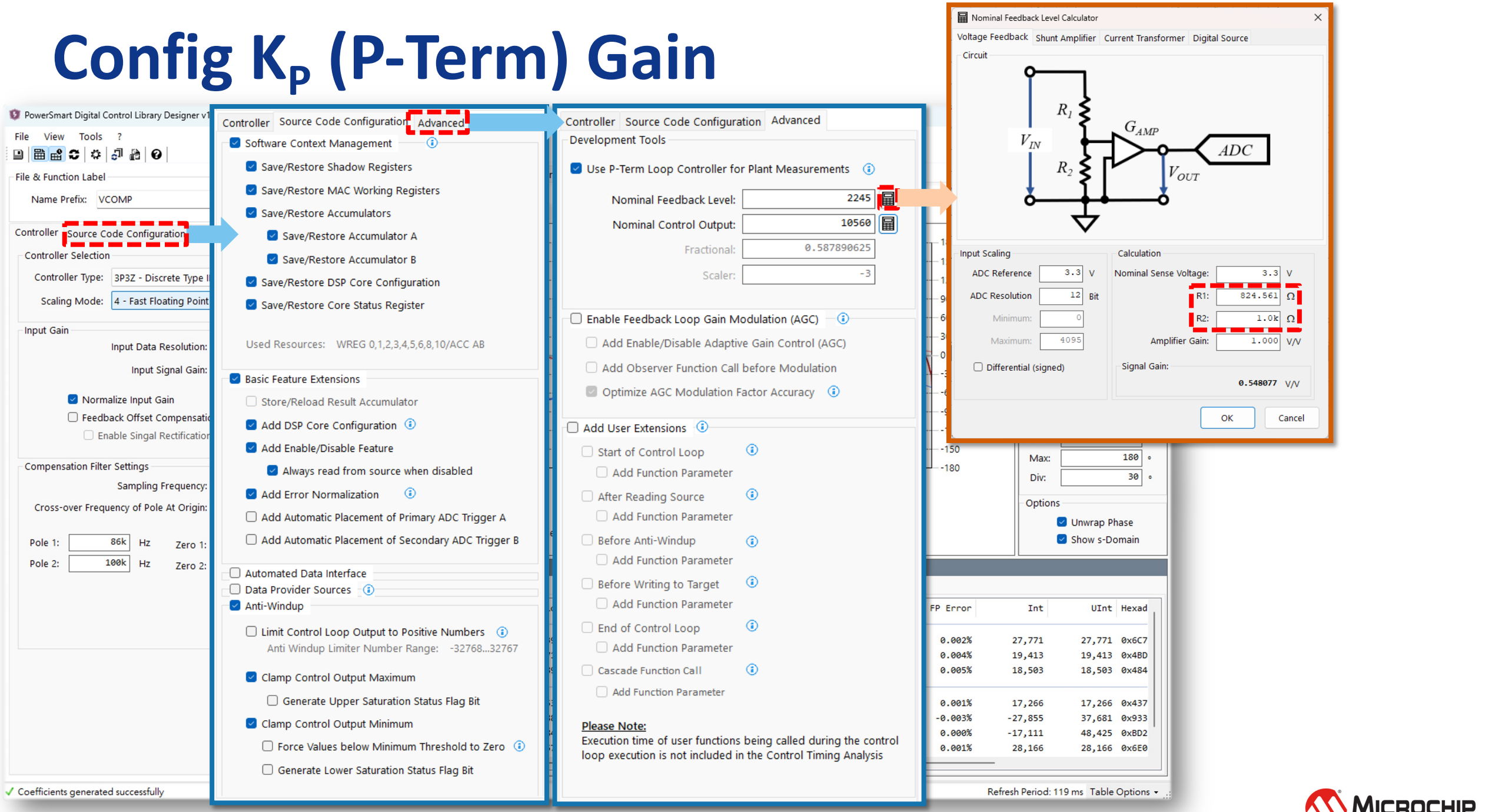
Config K_p (P-Term) Gain



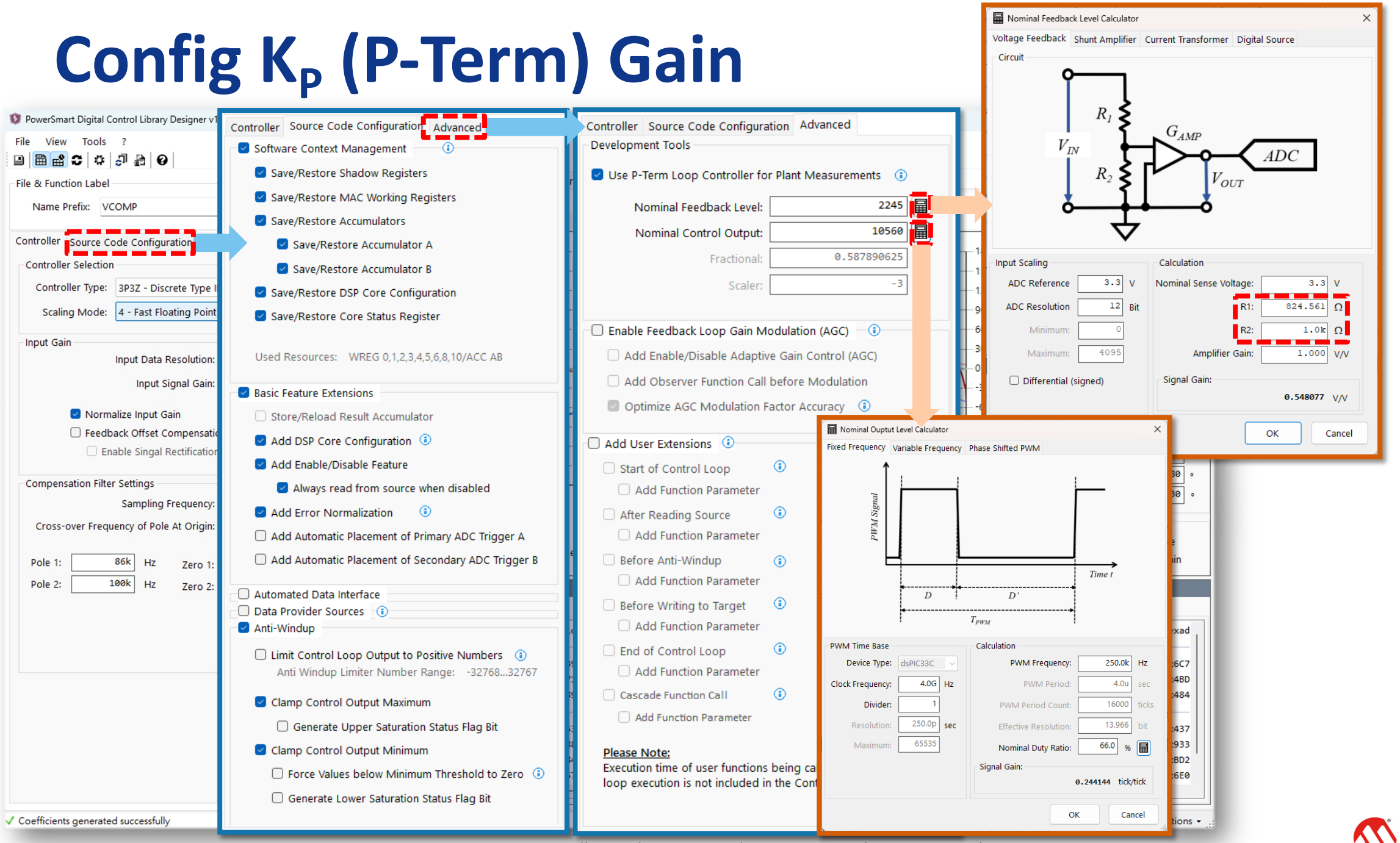
Config K_p (P-Term) Gain



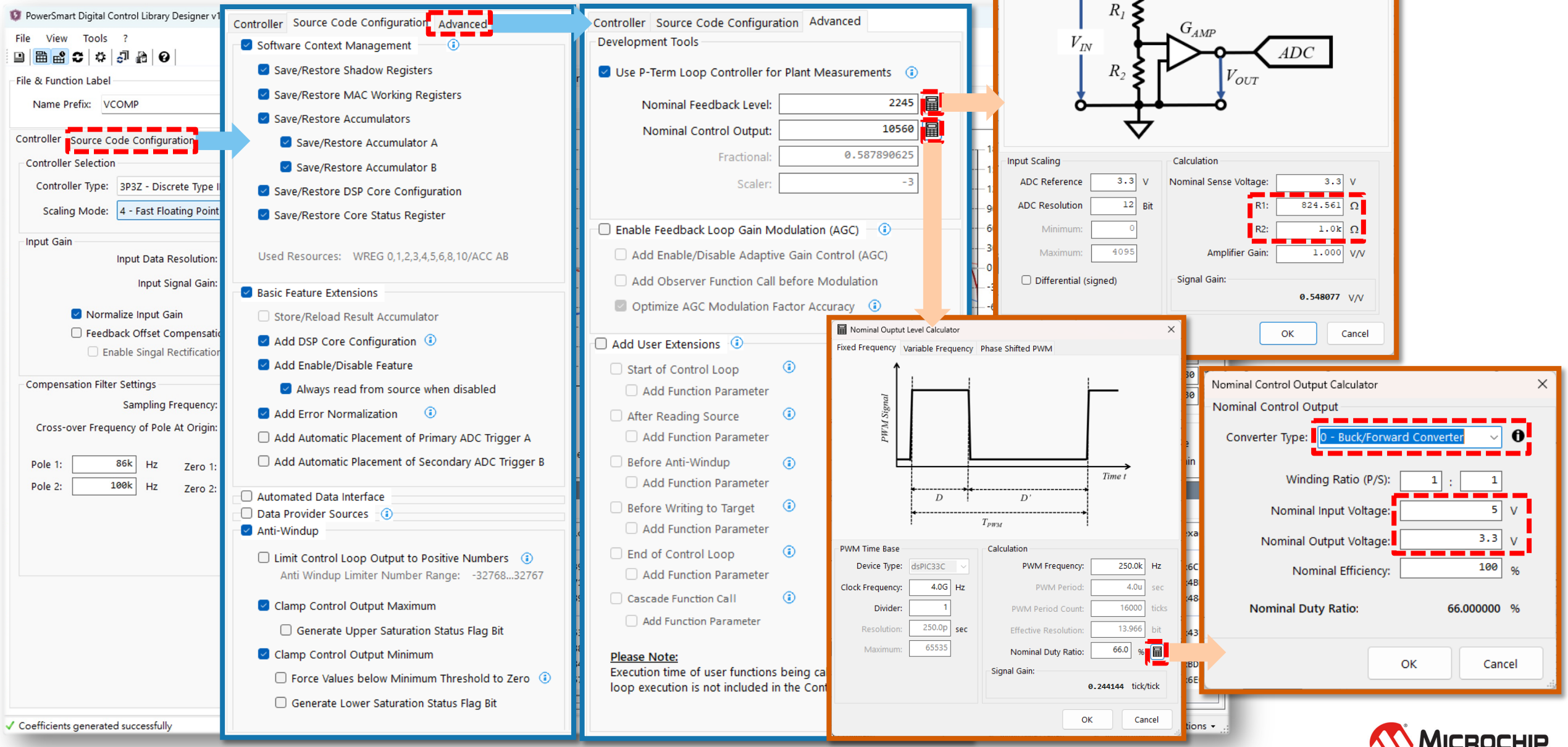
Config K_p (P-Term) Gain



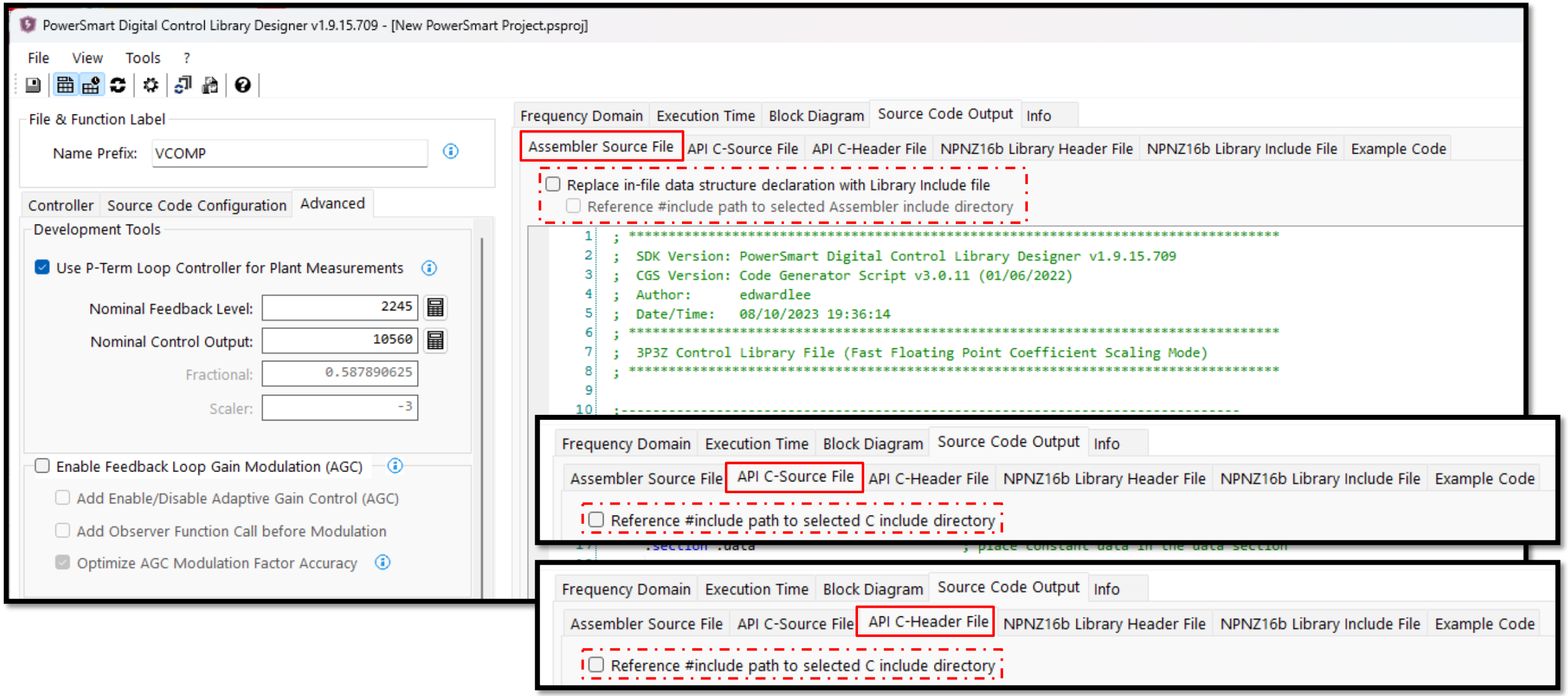
Config K_p (P-Term) Gain



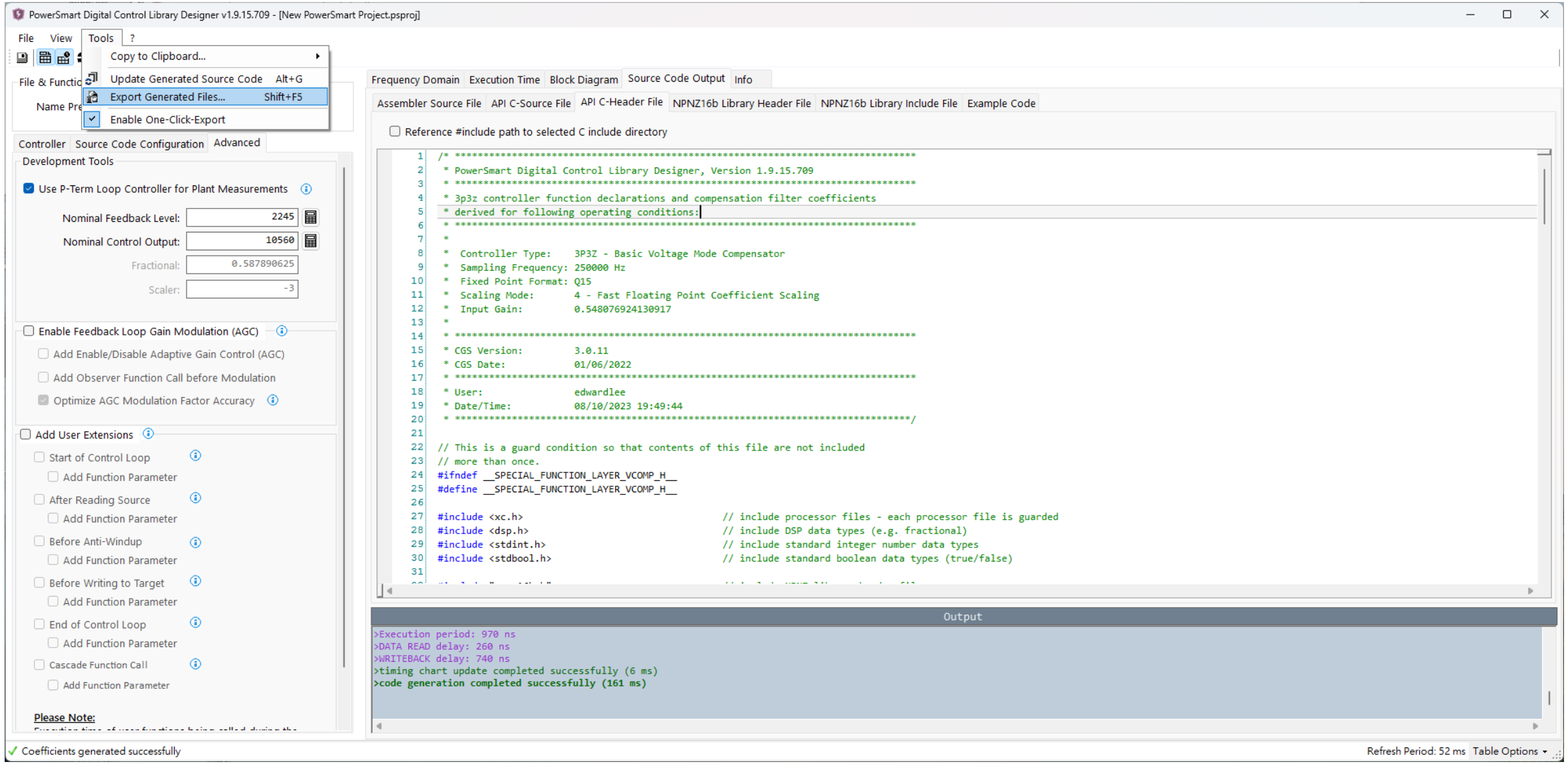
Config K_p (P-Term) Gain



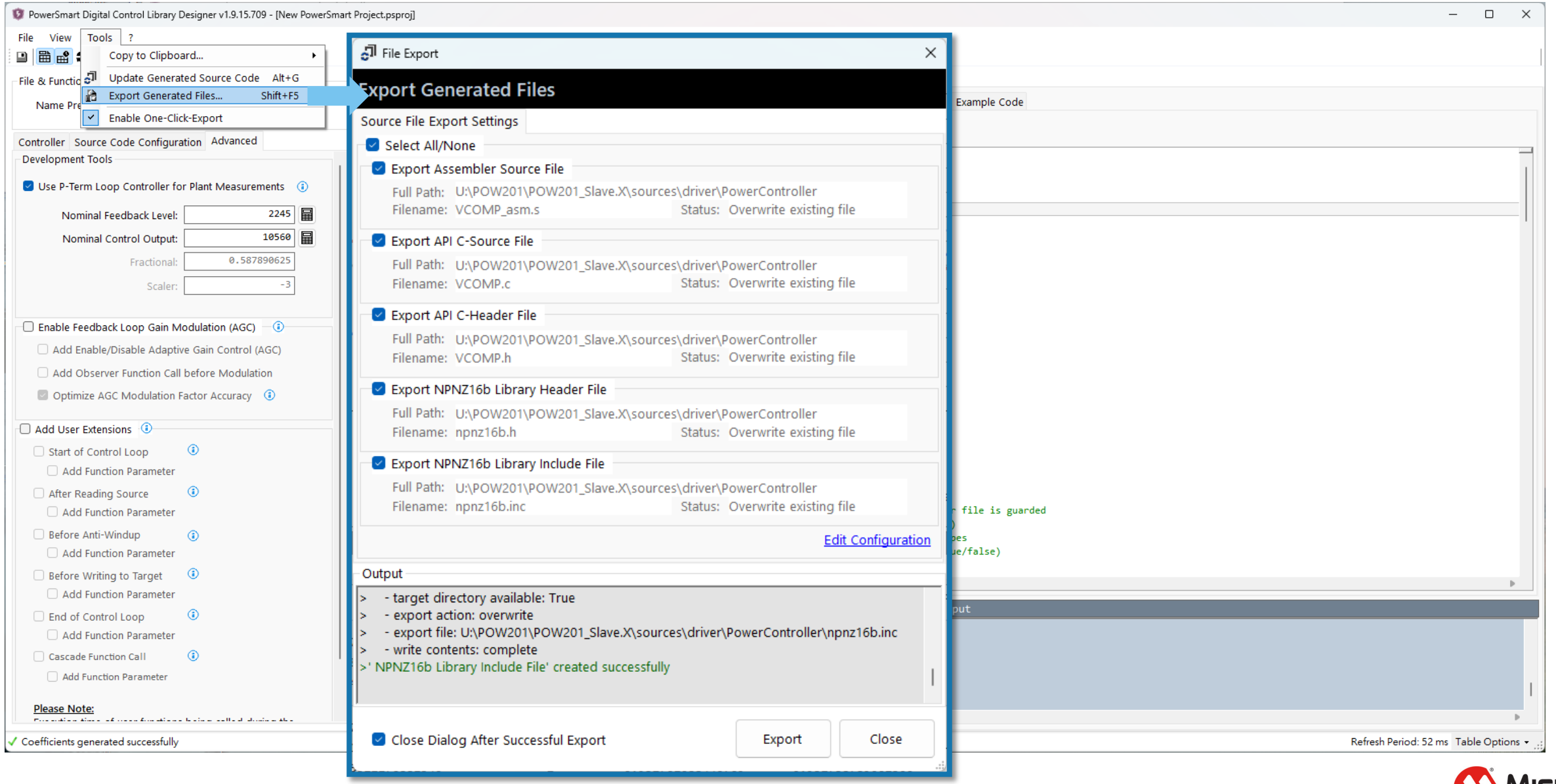
Setting for C-include Directory



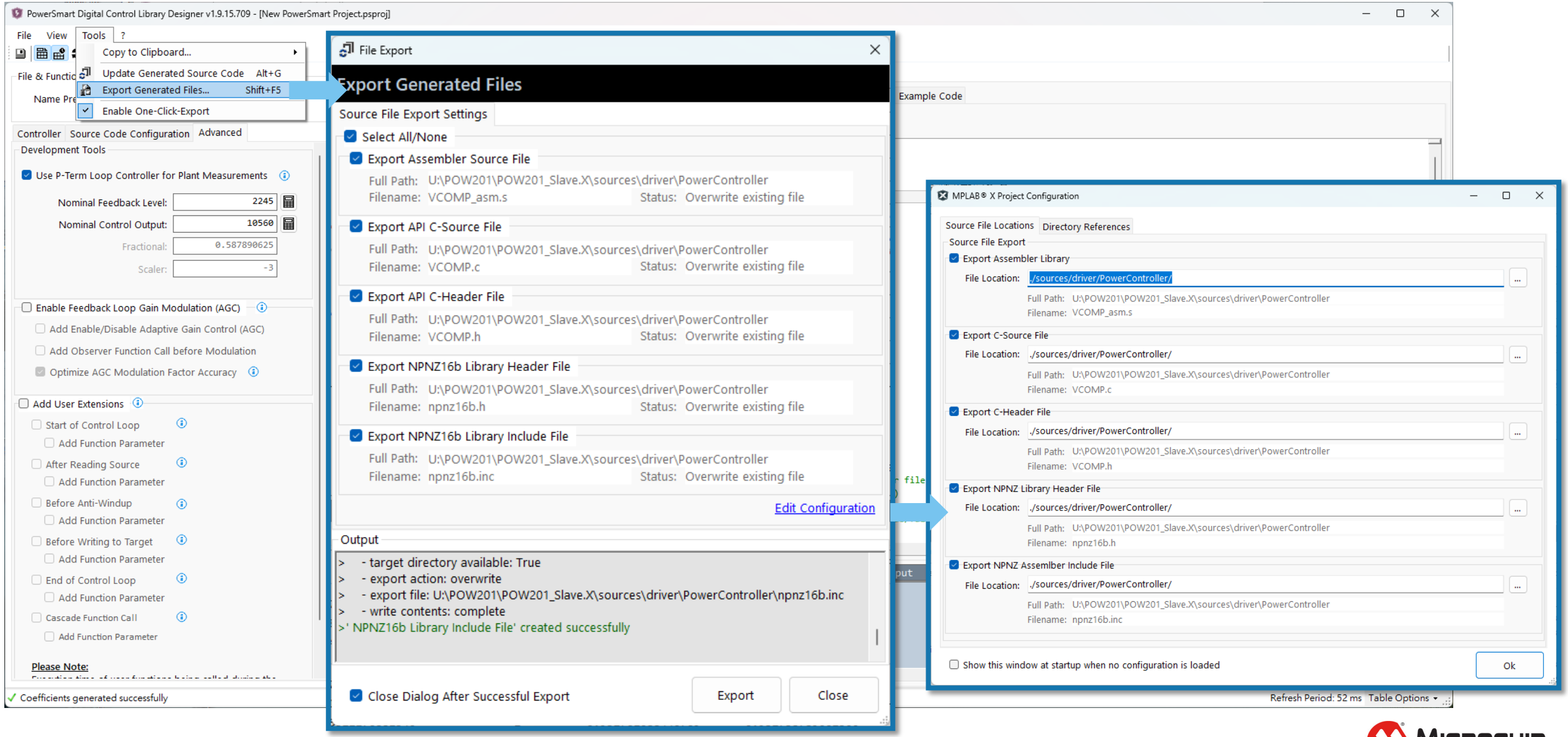
Export Compensator Sources Codes



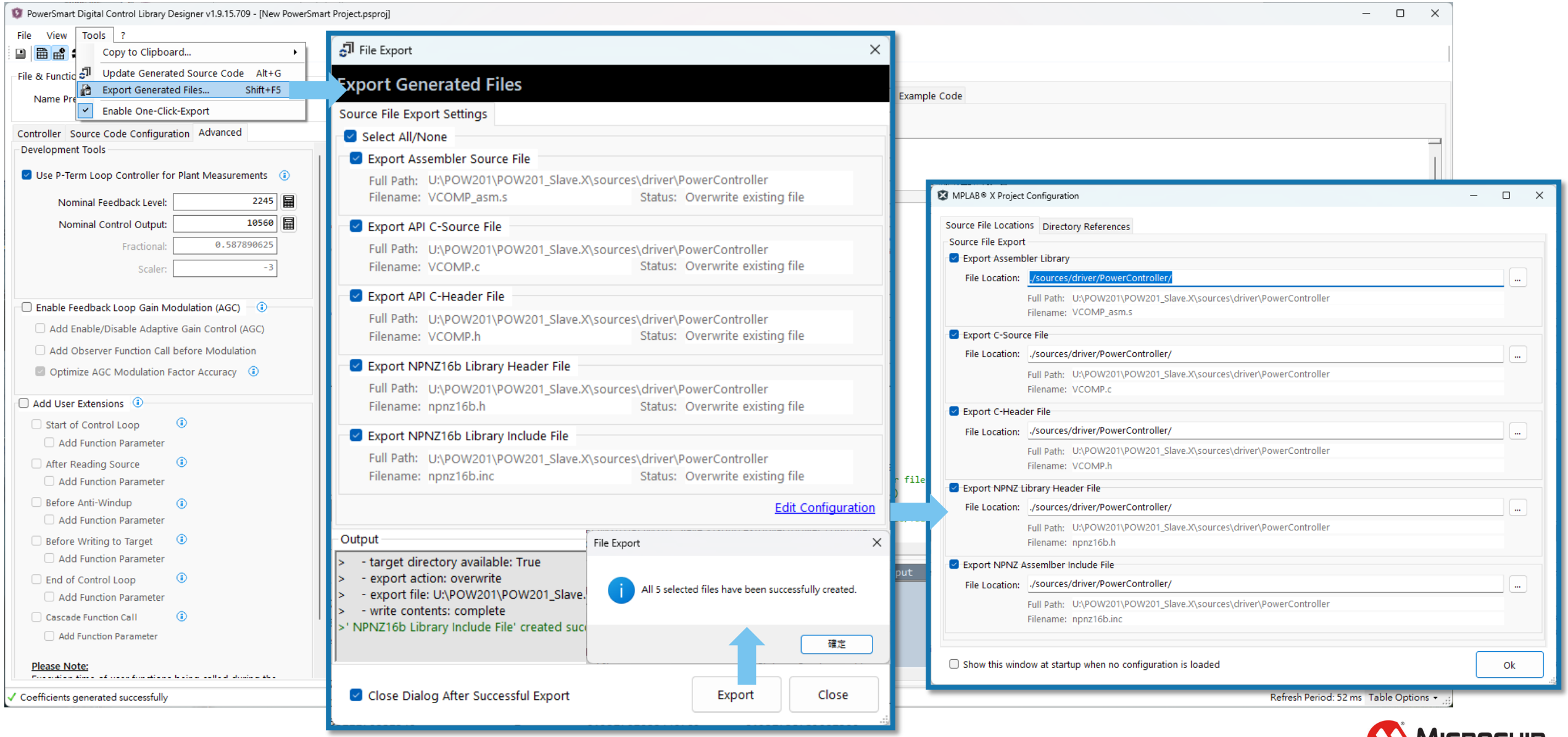
Export Compensator Sources Codes



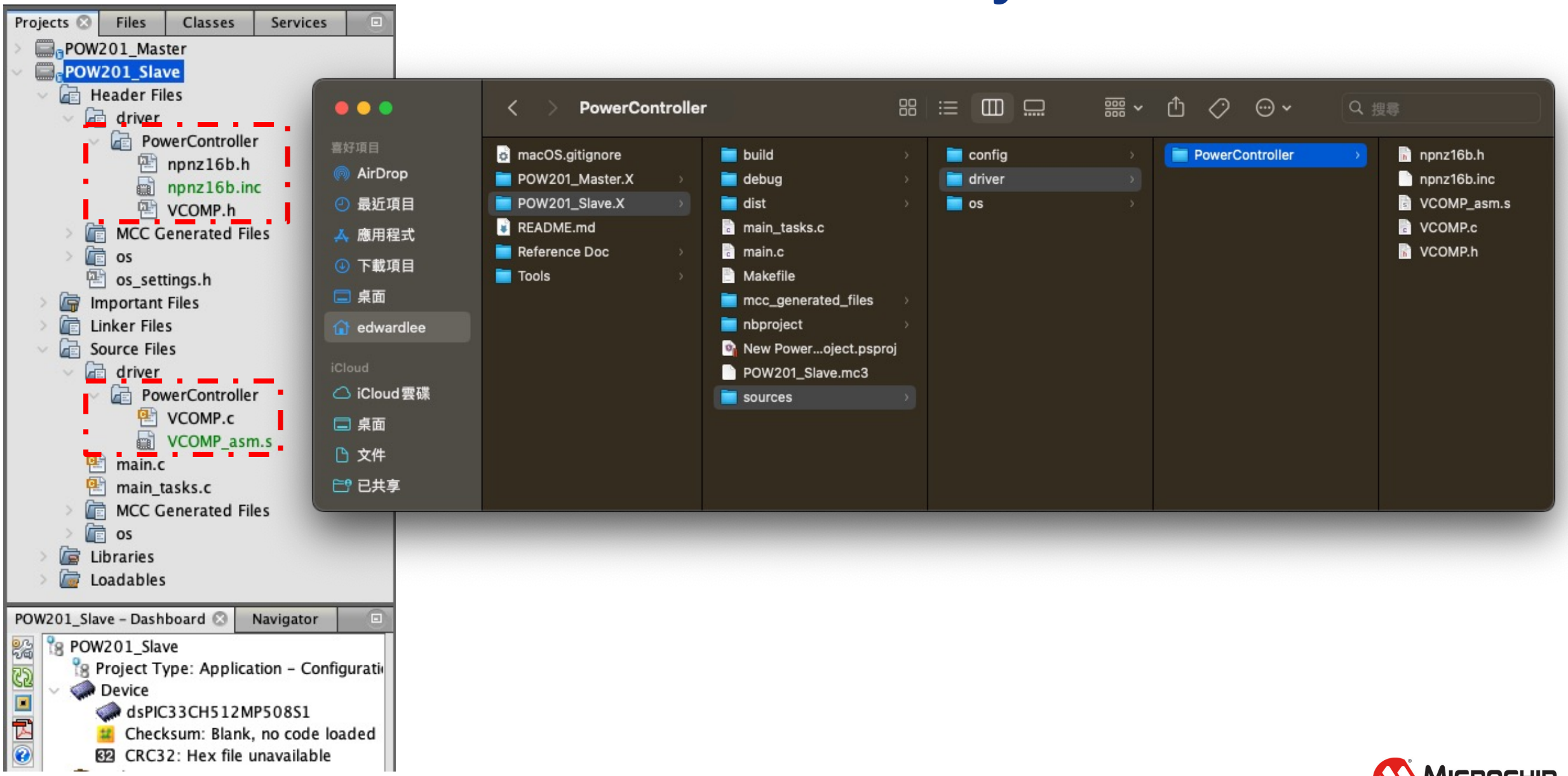
Export Compensator Sources Codes



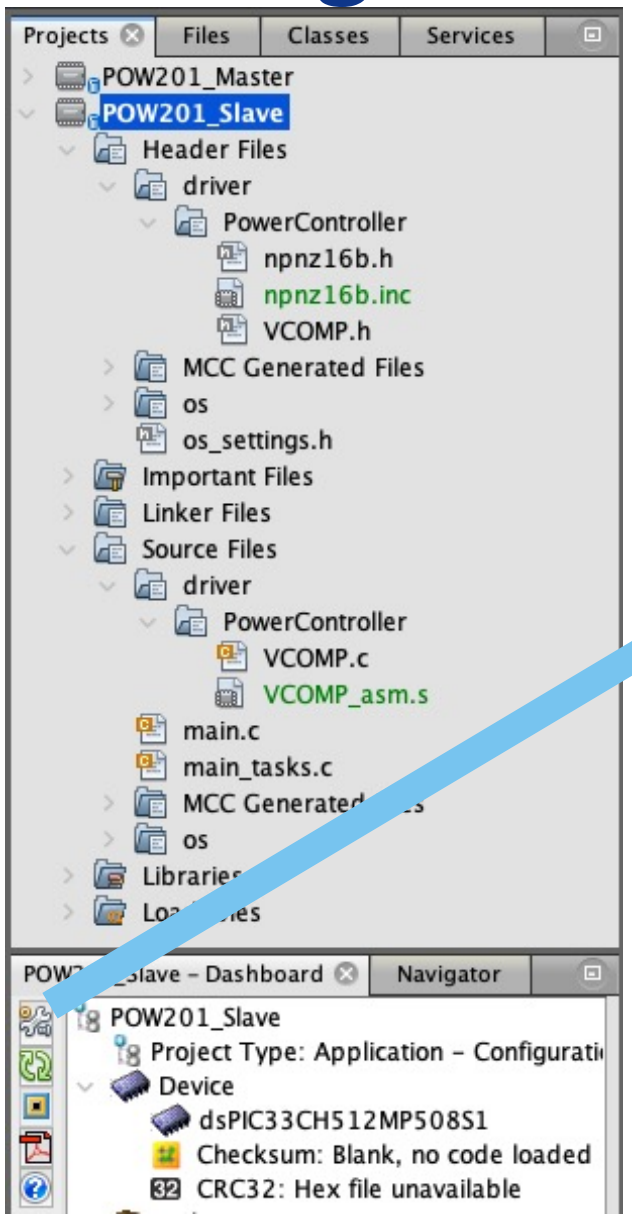
Export Compensator Sources Codes

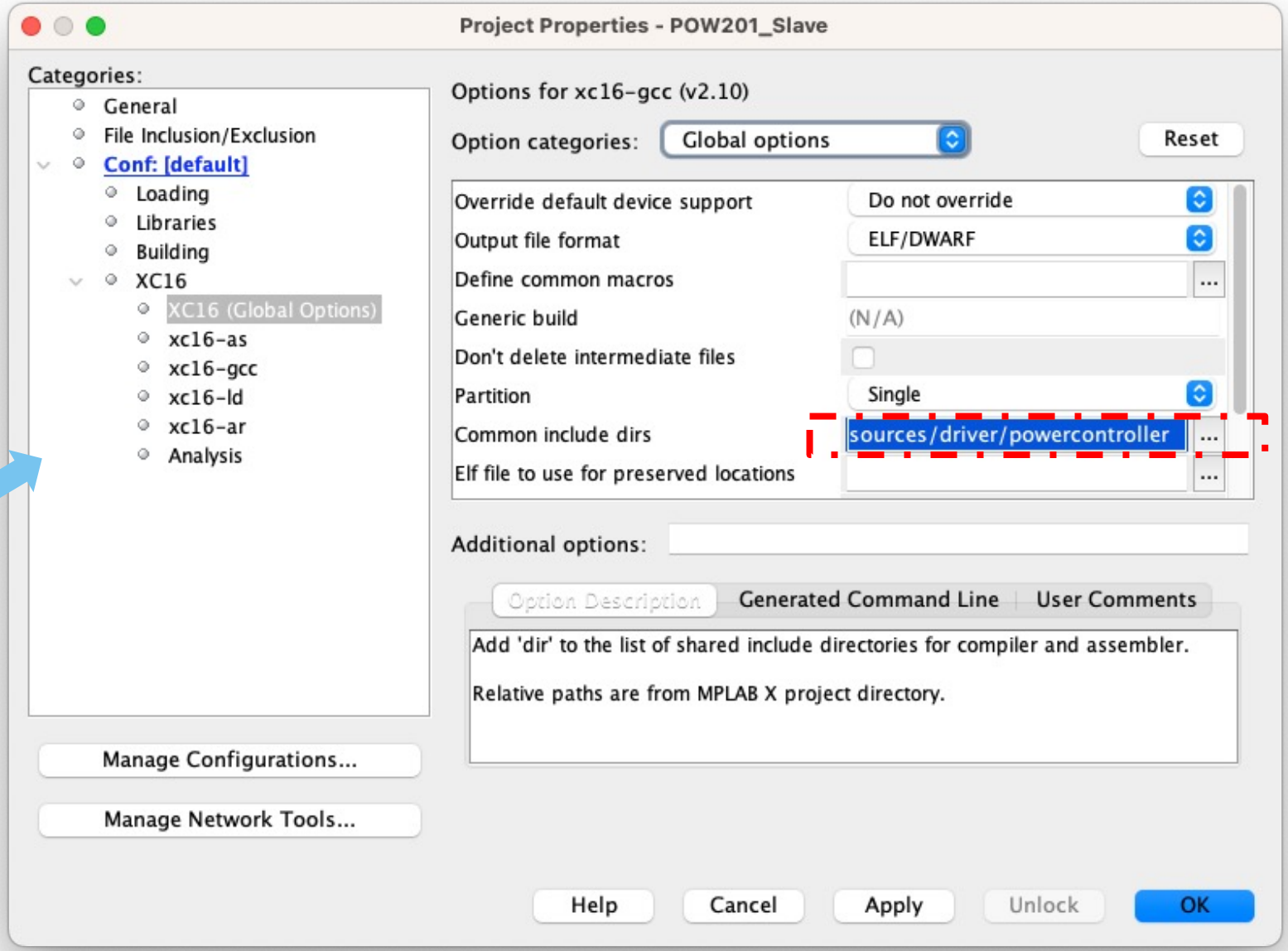


Add Control Libraries to the Project

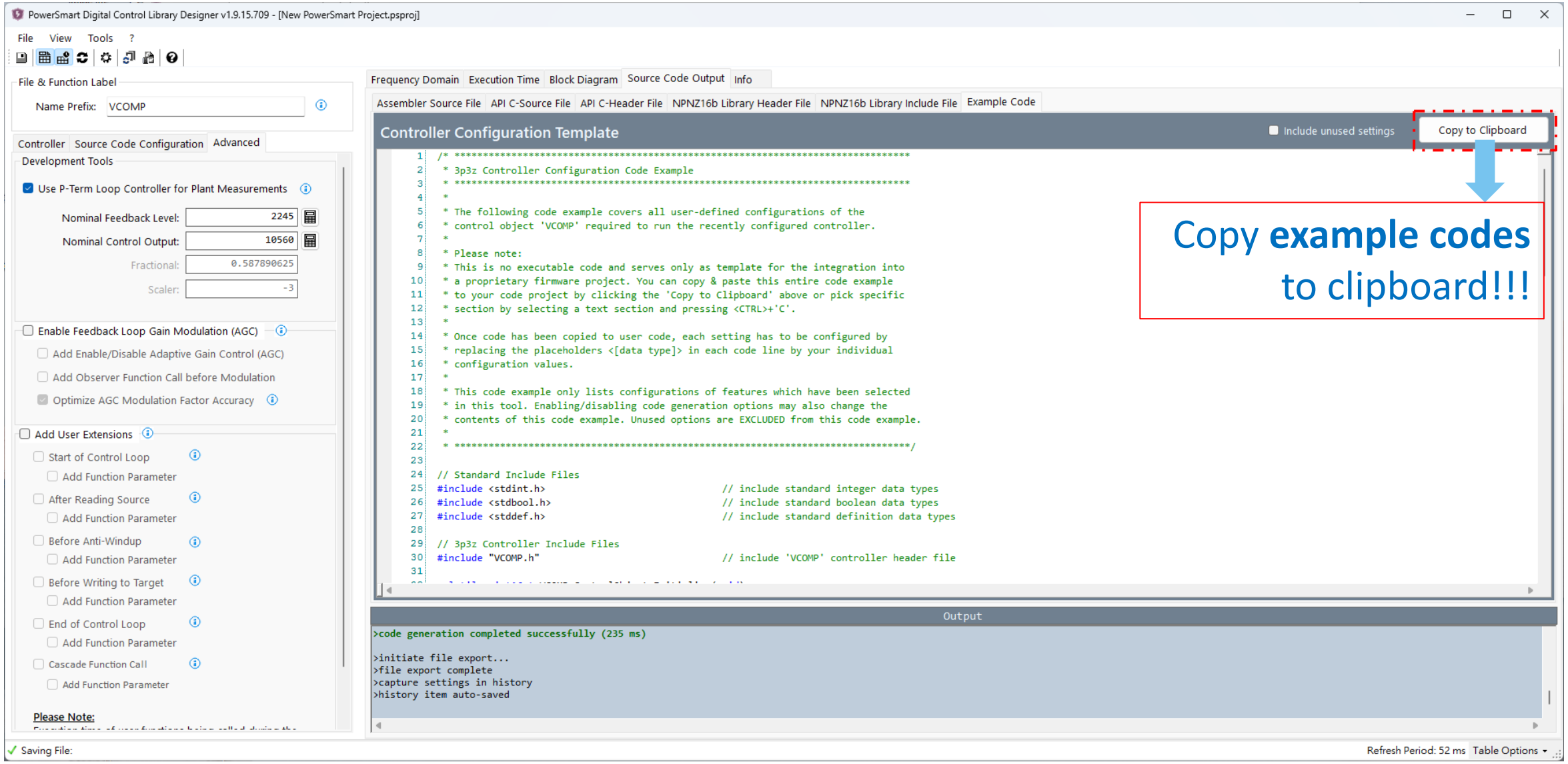


Setting for C-include Directory

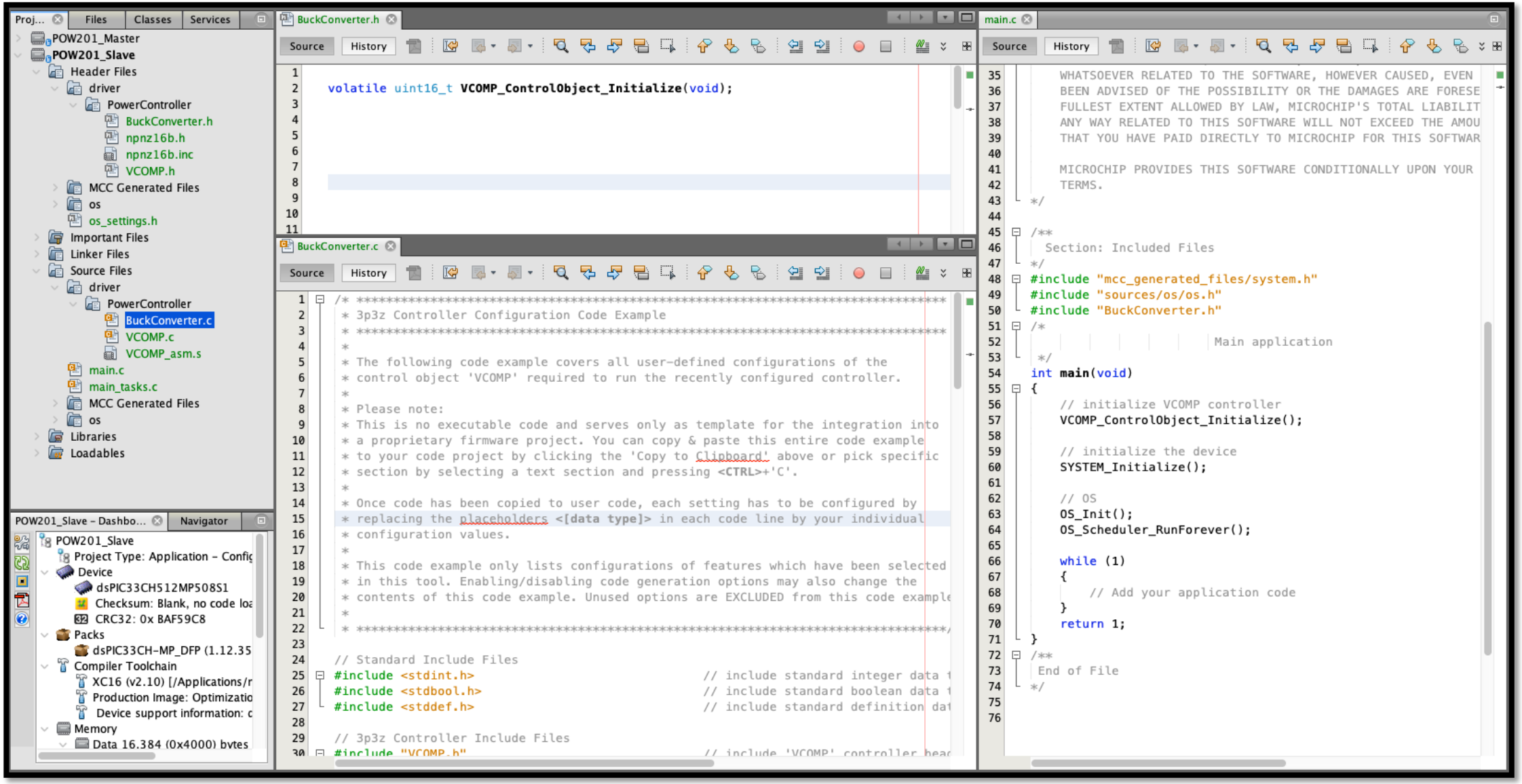




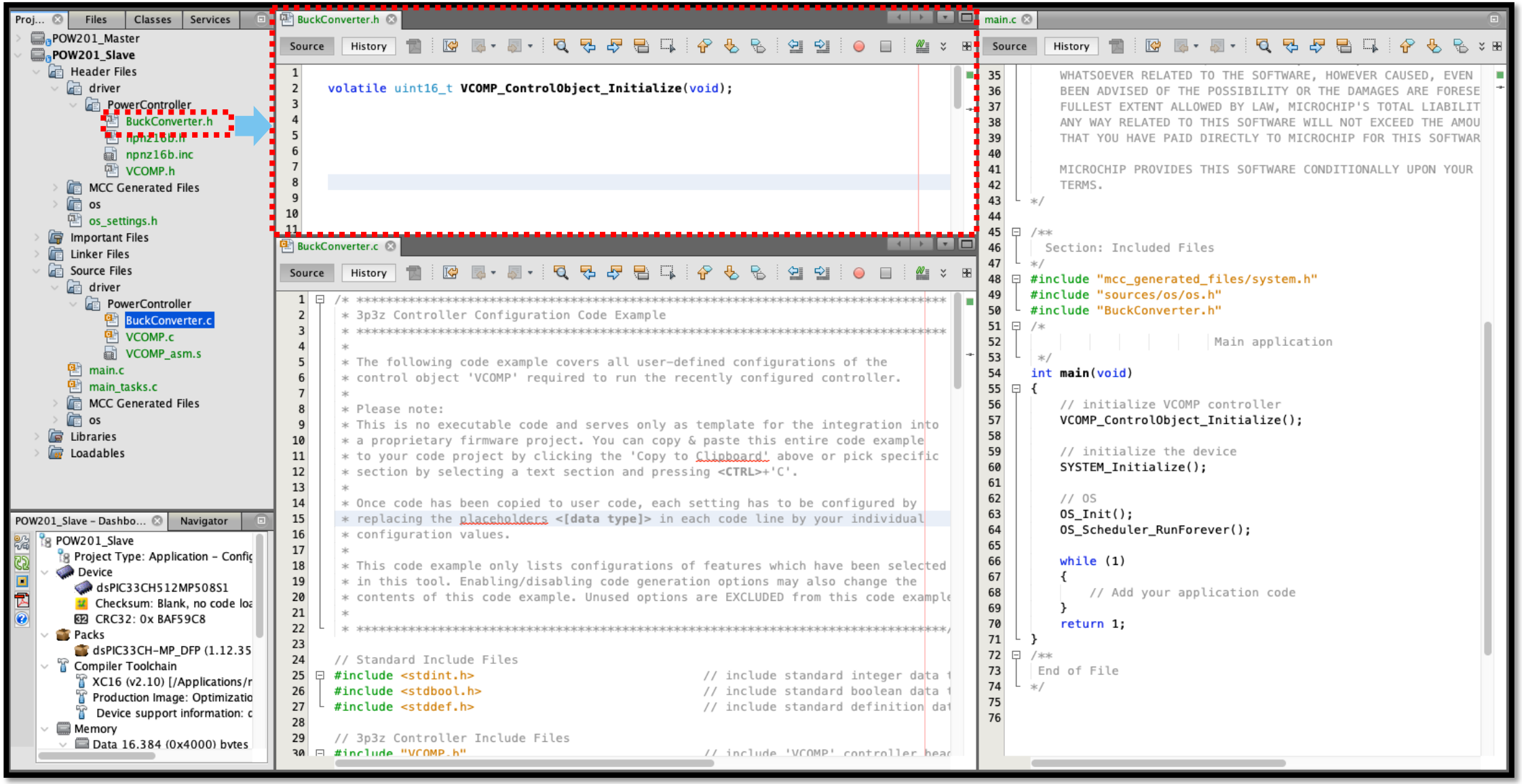
Copy example codes



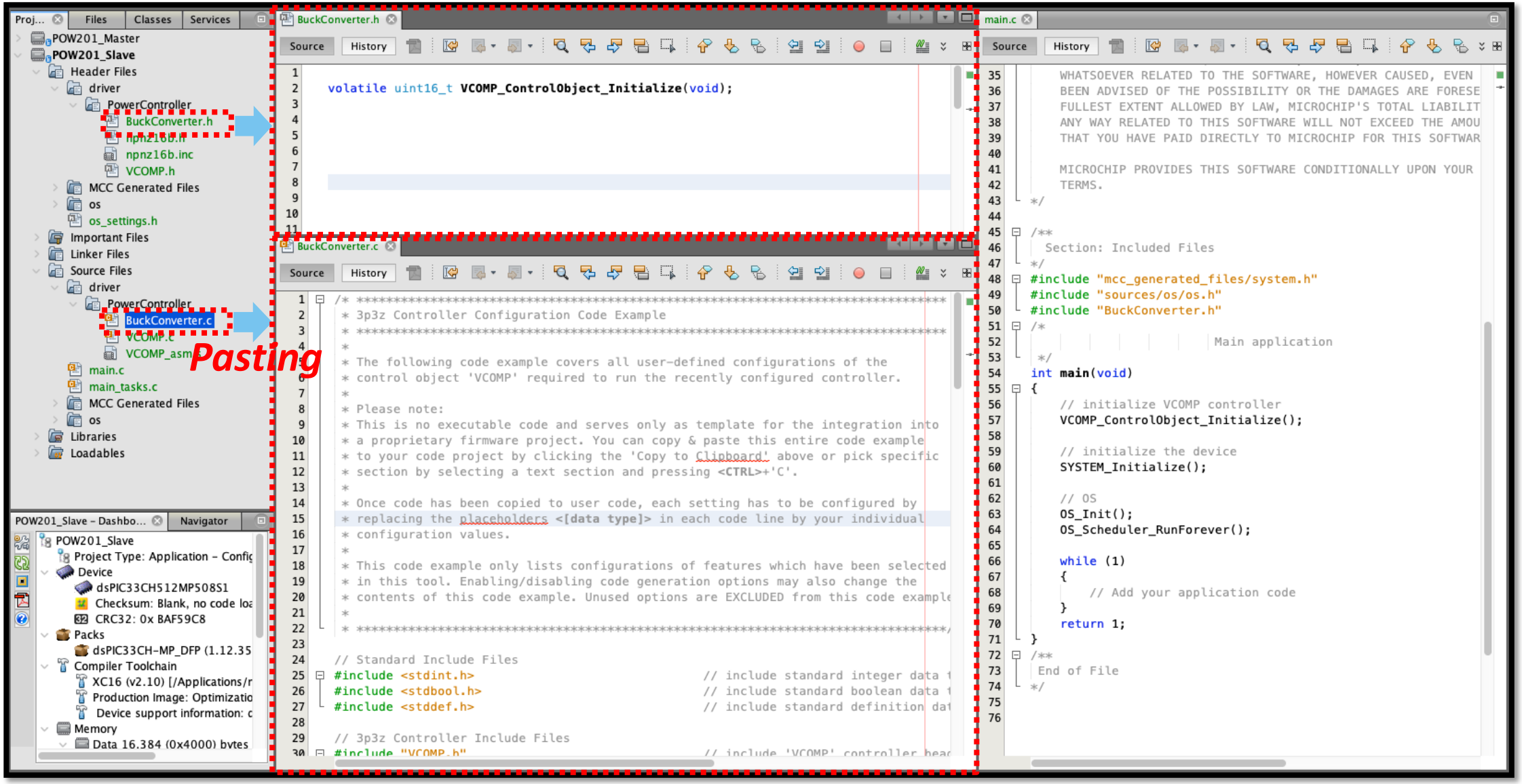
New file with pasting example codes!



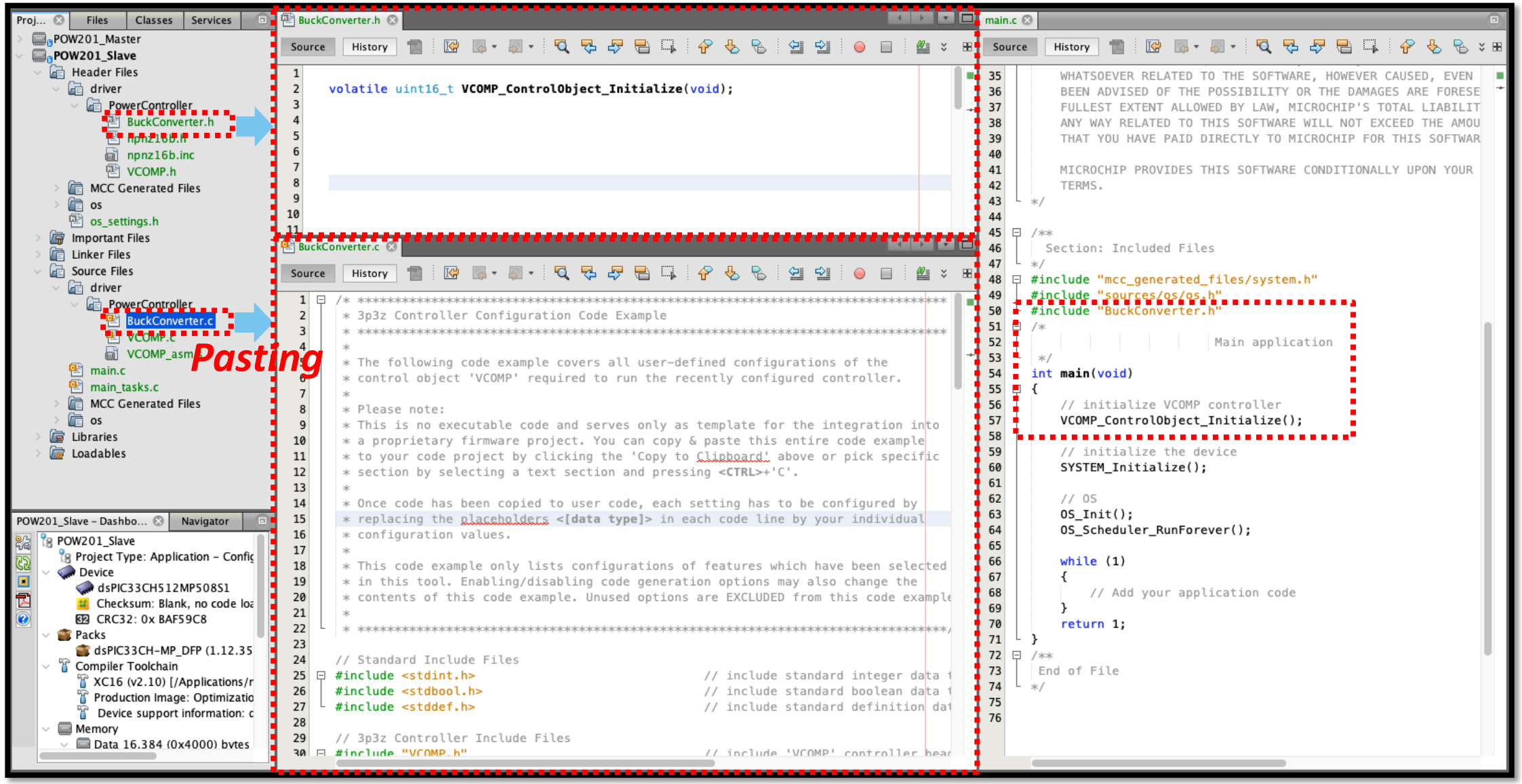
New file with pasting example codes!



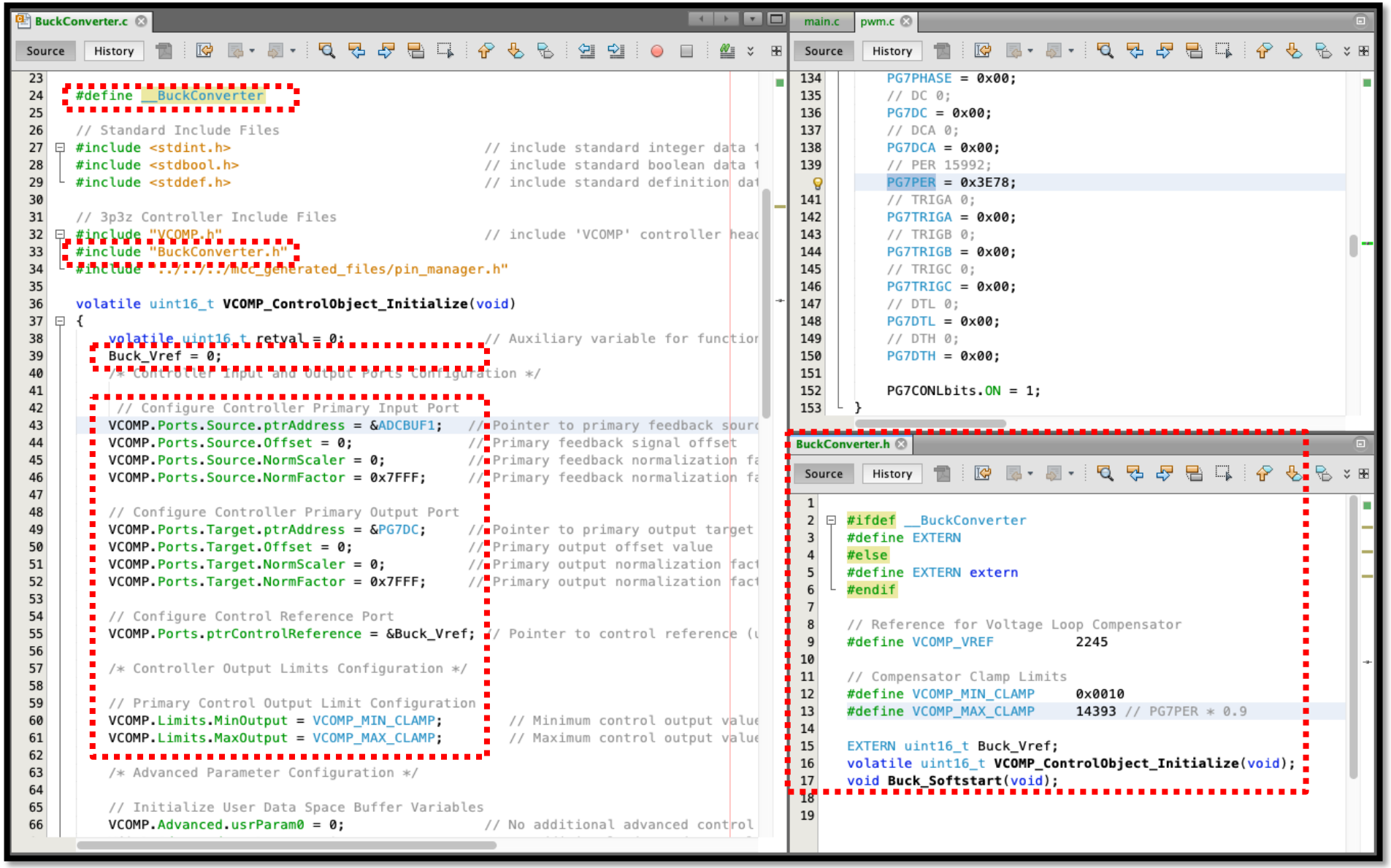
New file with pasting example codes!



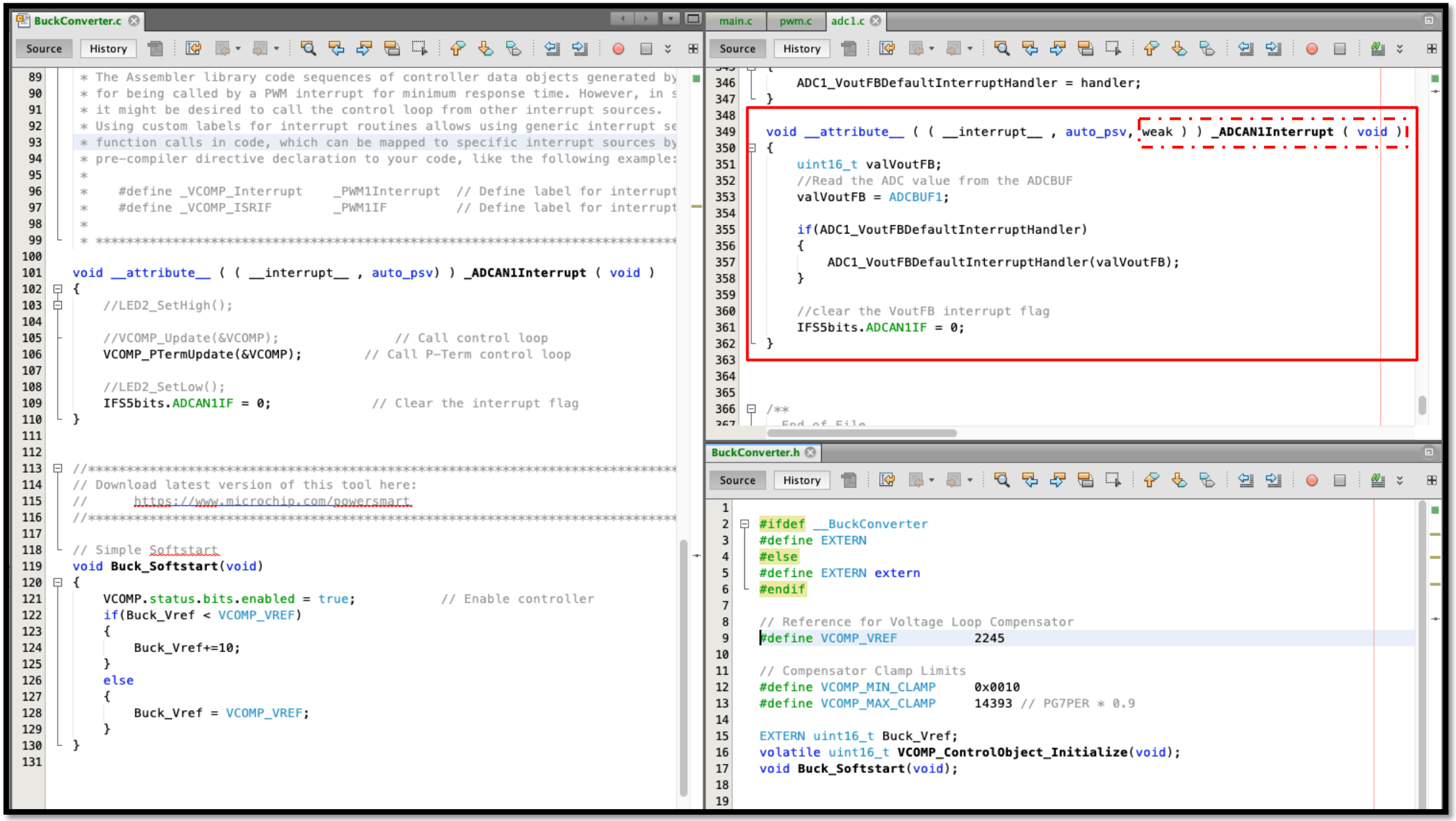
New file with pasting example codes!



Setting Initial Registers



Setting ADC ISR



BuckConverter.c

```
89  * The Assembler library code sequences of controller data objects generated by
90  * for being called by a PWM interrupt for minimum response time. However, in s
91  * it might be desired to call the control loop from other interrupt sources.
92  * Using custom labels for interrupt routines allows using generic interrupt se
93  * function calls in code, which can be mapped to specific interrupt sources by
94  * pre-compiler directive declaration to your code, like the following example:
95  *
96  *    #define _VCOMP_Interrupt    _PWM1Interrupt  // Define label for interrupt
97  *    #define _VCOMP_ISRIF        _PWM1IF         // Define label for interrupt
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 //*****************************************************************************
114 // Download latest version of this tool here:
115 //      https://www.microchip.com/powersmart
116 //*****************************************************************************
117
118 // Simple Softstart
119 void Buck_Softstart(void)
120 {
121     VCOMP.status.bits.enabled = true;           // Enable controller
122     if(Buck_Vref < VCOMP_VREF)
123     {
124         Buck_Vref+=10;
125     }
126     else
127     {
128         Buck_Vref = VCOMP_VREF;
129     }
130 }
131
```

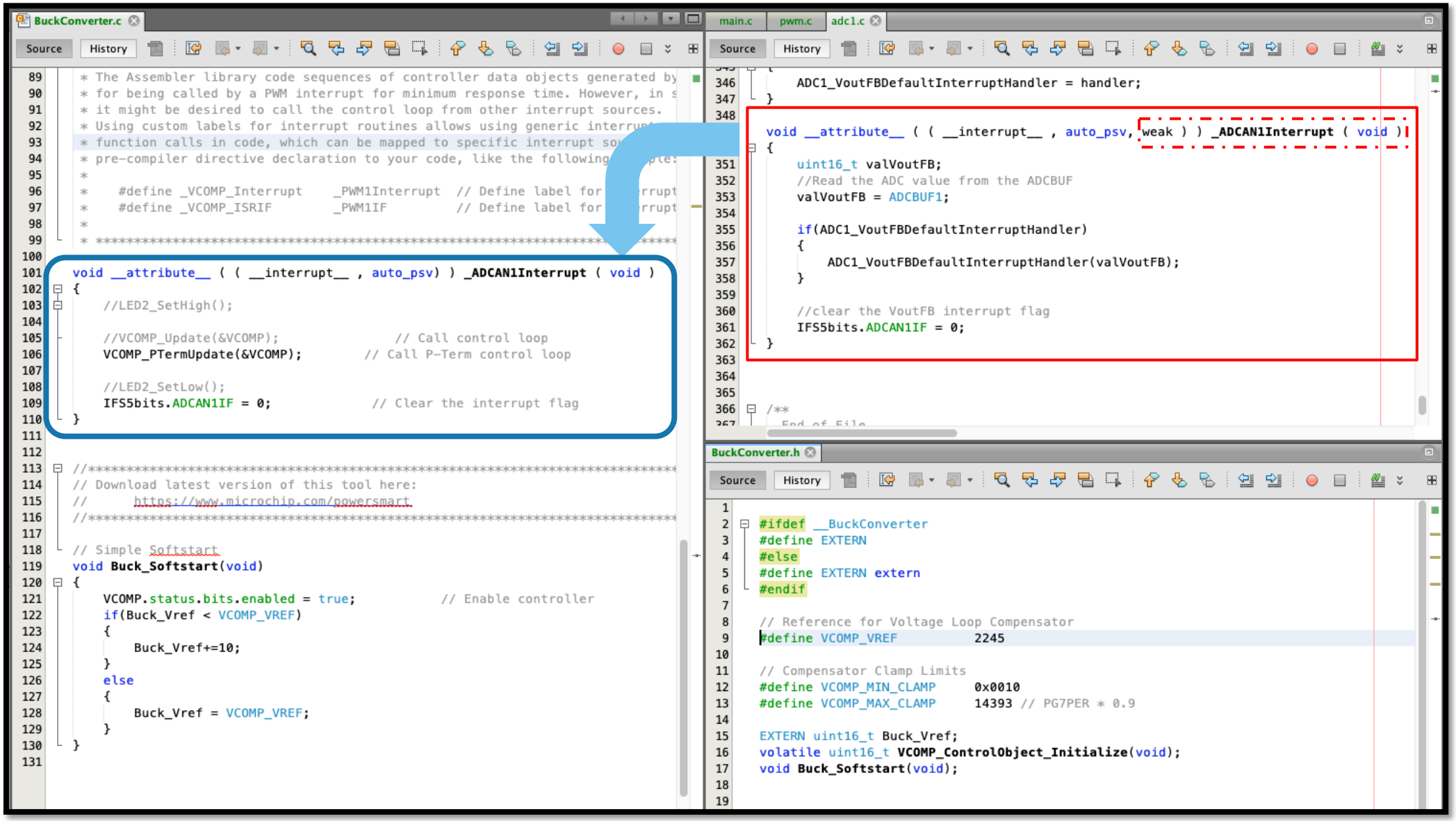
main.c | pwm.c | adc1.c

```
346     ADC1_VoutFBDefaultInterruptHandler = handler;
347 }
348
349 void __attribute__ ( ( __interrupt__ , auto_psv, weak ) ) _ADCAN1Interrupt ( void )
350 {
351     uint16_t valVoutFB;
352     //Read the ADC value from the ADCBUF
353     valVoutFB = ADCBUF1;
354
355     if(ADC1_VoutFBDefaultInterruptHandler)
356     {
357         ADC1_VoutFBDefaultInterruptHandler(valVoutFB);
358     }
359
360     //clear the VoutFB interrupt flag
361     IFS5bits.ADCAN1IF = 0;
362 }
363
364
365
366 /**
367   End of File
```

BuckConverter.h

```
1
2  #ifdef __BuckConverter
3  #define EXTERN
4  #else
5  #define EXTERN extern
6  #endif
7
8  // Reference for Voltage Loop Compensator
9  #define VCOMP_VREF          2245
10
11 // Compensator Clamp Limits
12 #define VCOMP_MIN_CLAMP     0x0010
13 #define VCOMP_MAX_CLAMP     14393 // PG7PER * 0.9
14
15 EXTERN uint16_t Buck_Vref;
16 volatile uint16_t VCOMP_ControlObject_Initialize(void);
17 void Buck_Softstart(void);
18
19
```

Setting ADC ISR



BuckConverter.c

```
89  * The Assembler library code sequences of controller data objects generated by
90  * for being called by a PWM interrupt for minimum response time. However, in s
91  * it might be desired to call the control loop from other interrupt sources.
92  * Using custom labels for interrupt routines allows using generic interrupt
93  * function calls in code, which can be mapped to specific interrupt so
94  * pre-compiler directive declaration to your code, like the following
95  *
96  *    #define _VCOMP_Interrupt    _PWM1Interrupt  // Define label for
97  *    #define _VCOMP_ISRIF        _PWM1IF         // Define label for
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 //*****************************************************************************
114 // Download latest version of this tool here:
115 //      https://www.microchip.com/powersmart
116 //*****************************************************************************
117
118 // Simple Softstart
119 void Buck_Softstart(void)
120 {
121     VCOMP.status.bits.enabled = true;           // Enable controller
122     if(Buck_Vref < VCOMP_VREF)
123     {
124         Buck_Vref+=10;
125     }
126     else
127     {
128         Buck_Vref = VCOMP_VREF;
129     }
130 }
131
```

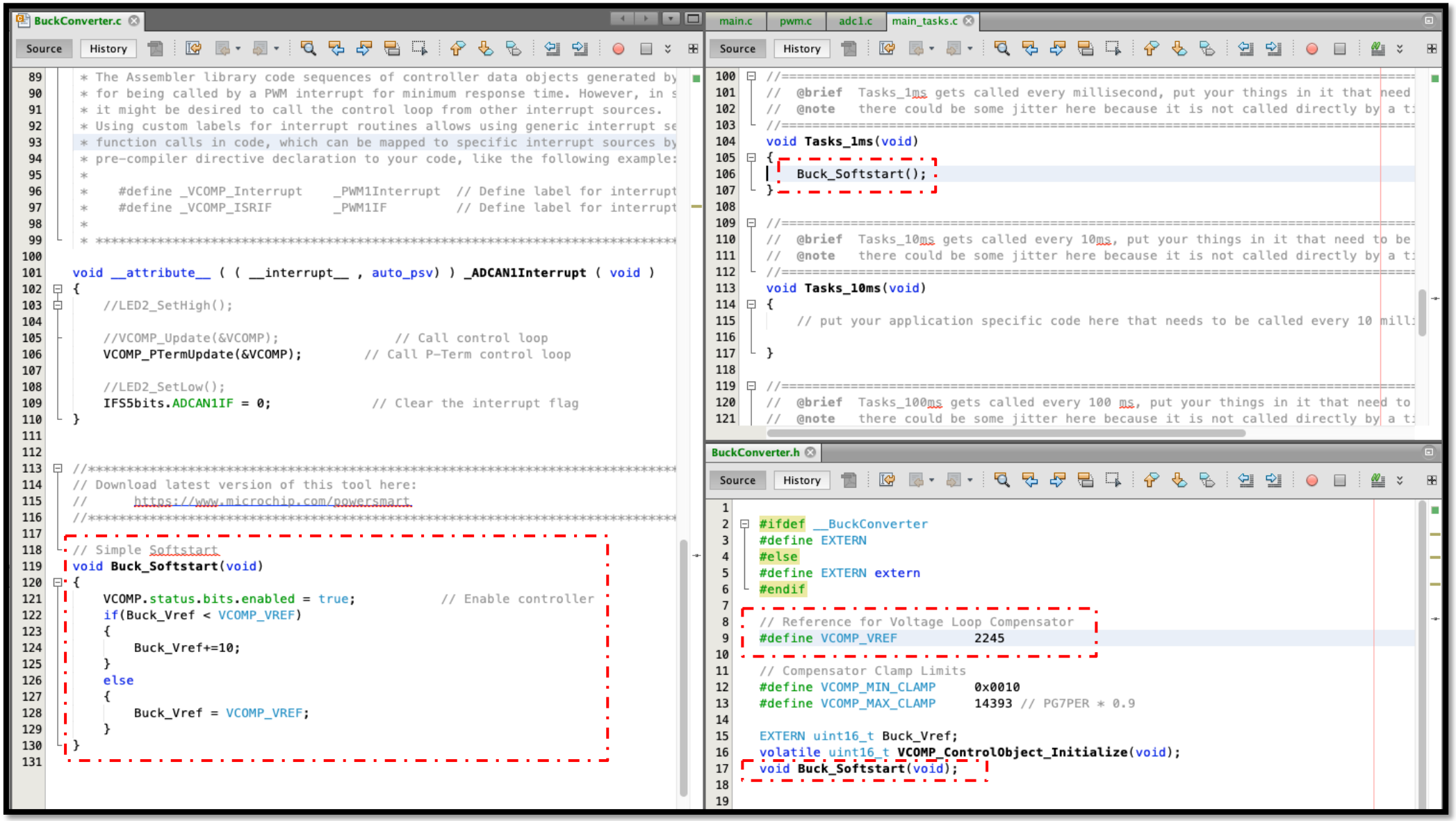
main.c | pwm.c | adc1.c

```
346     ADC1_VoutFBDefaultInterruptHandler = handler;
347 }
348
    void __attribute__ ( ( __interrupt__ , auto_psv, weak ) ) _ADCAN1Interrupt ( void )
{
351     uint16_t valVoutFB;
352     //Read the ADC value from the ADCBUF
353     valVoutFB = ADCBUF1;
354
355     if(ADC1_VoutFBDefaultInterruptHandler)
356     {
357         ADC1_VoutFBDefaultInterruptHandler(valVoutFB);
358     }
359
360     //clear the VoutFB interrupt flag
361     IFS5bits.ADCAN1IF = 0;
362 }
363
364
365
366 /**
```

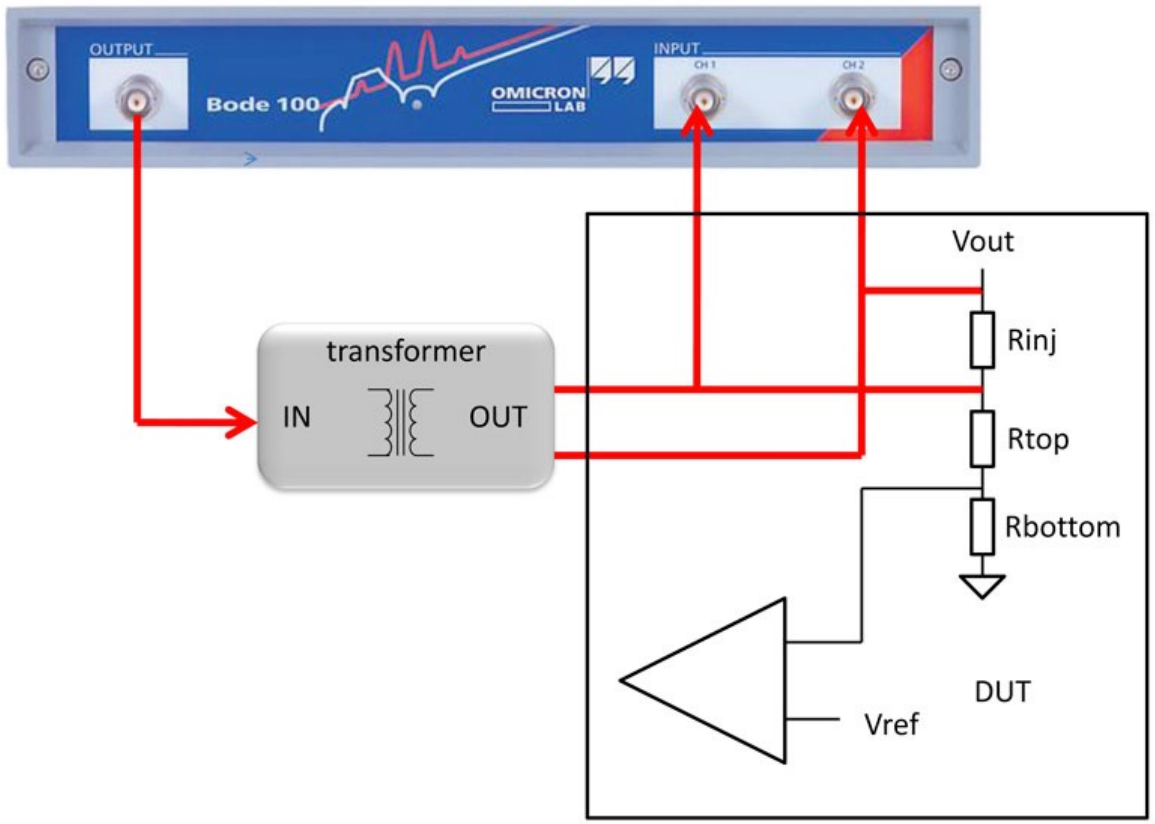
BuckConverter.h

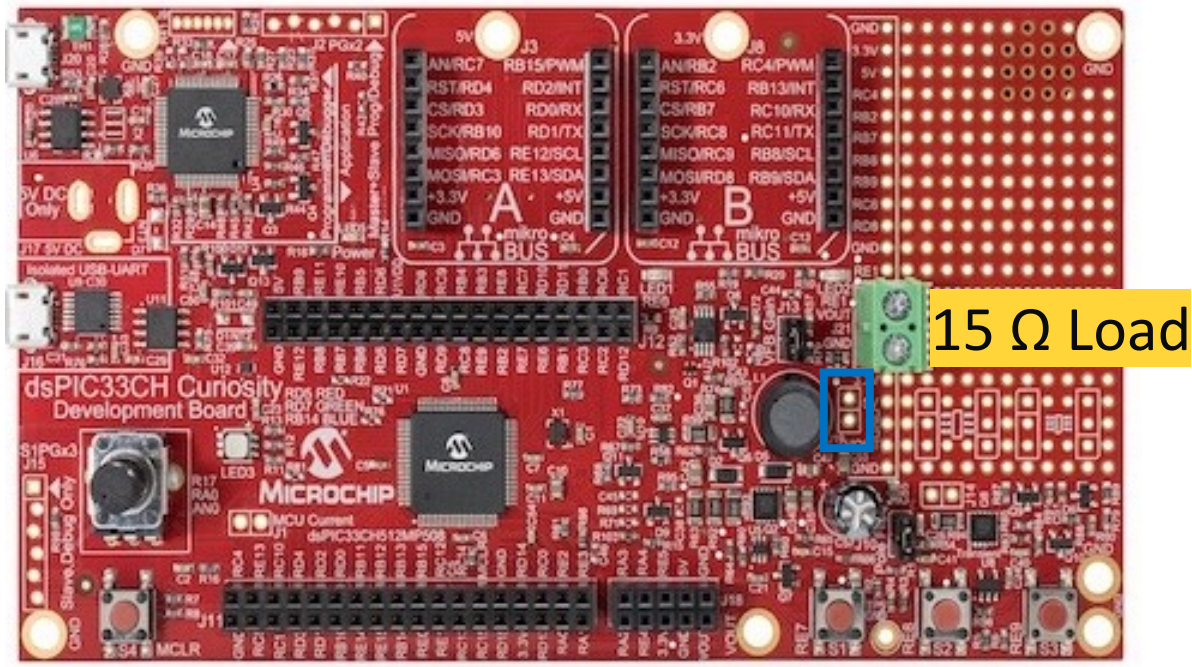
```
1
2   #ifdef __BuckConverter
3   #define EXTERN
4   #else
5   #define EXTERN extern
6   #endif
7
8   // Reference for Voltage Loop Compensator
9   #define VCOMP_VREF          2245
10
11  // Compensator Clamp Limits
12  #define VCOMP_MIN_CLAMP     0x0010
13  #define VCOMP_MAX_CLAMP     14393 // PG7PER * 0.9
14
15  EXTERN uint16_t Buck_Vref;
16  volatile uint16_t VCOMP_ControlObject_Initialize(void);
17  void Buck_Softstart(void);
18
19
```

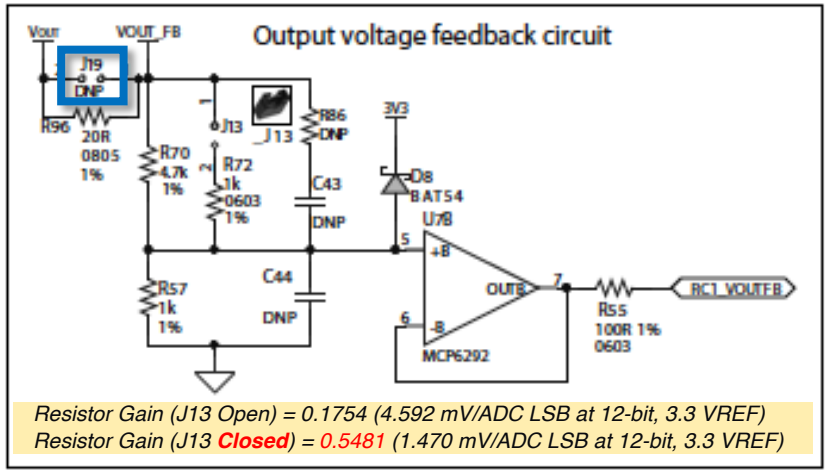
Implement Soft-start THEN Program BY Master!



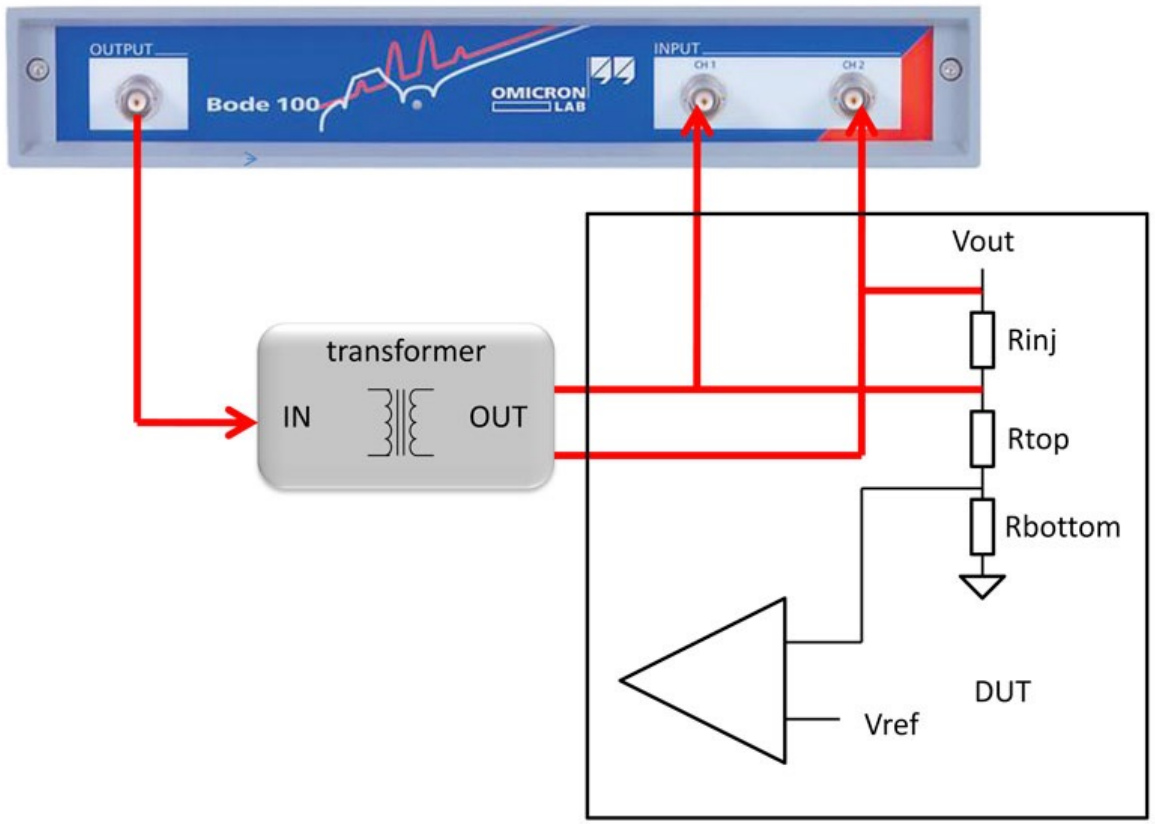
Plant Measurement Using Bode-100

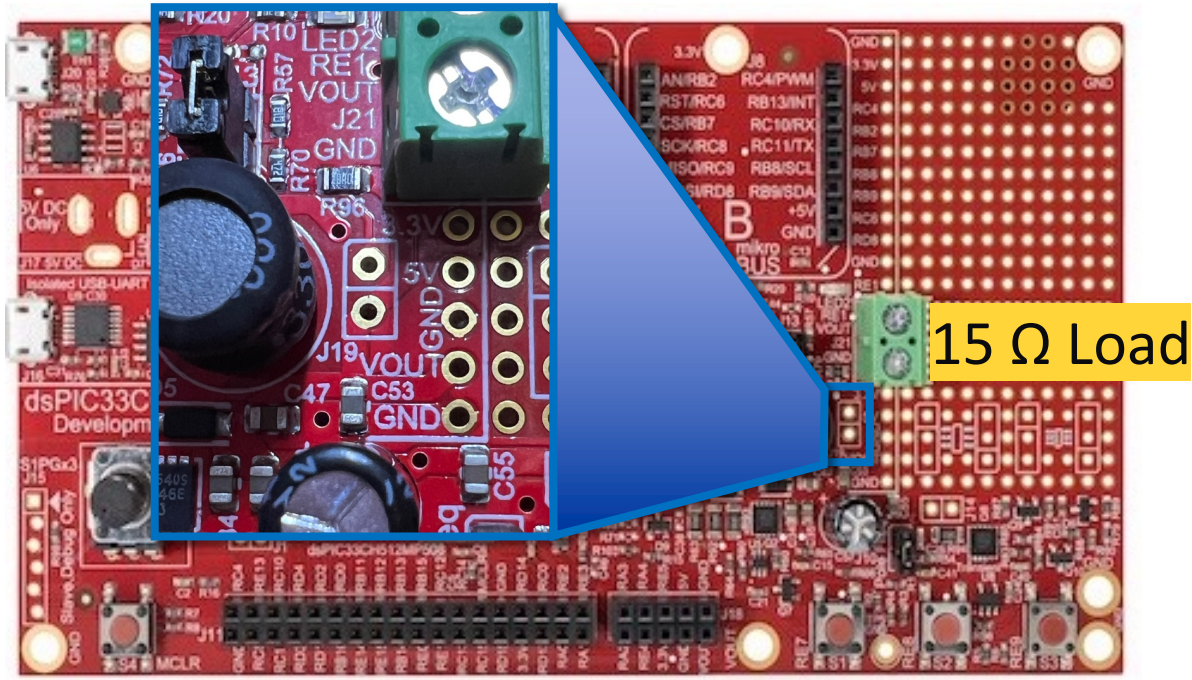


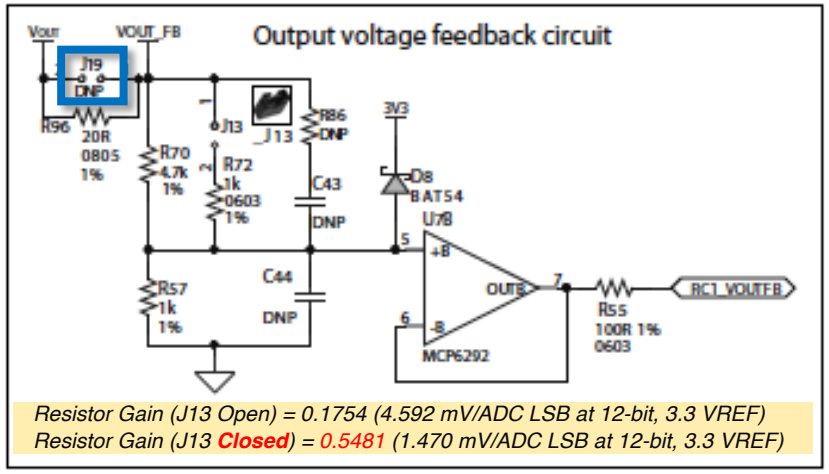




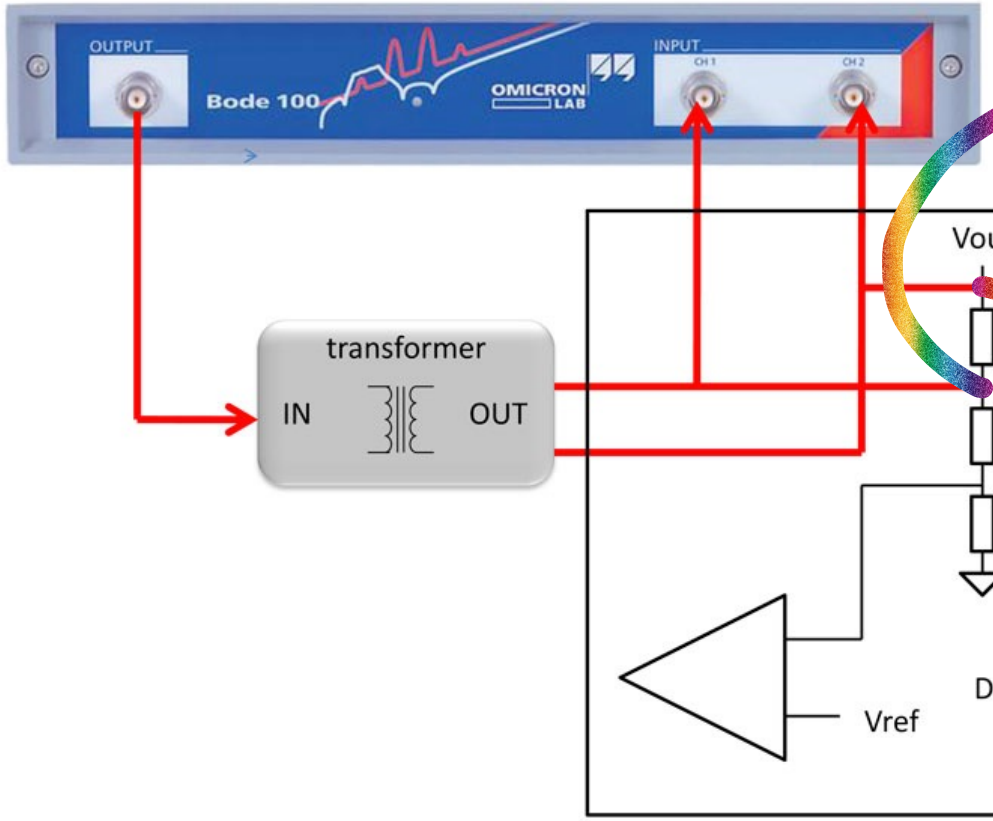
Plant Measurement Using Bode-100

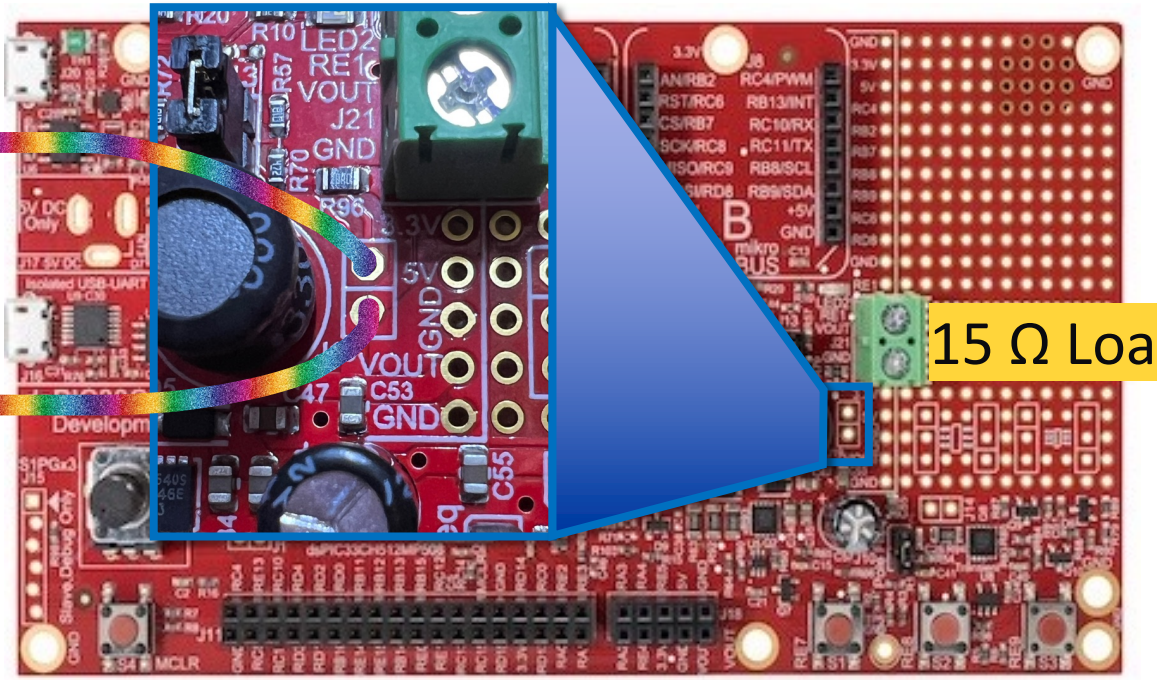


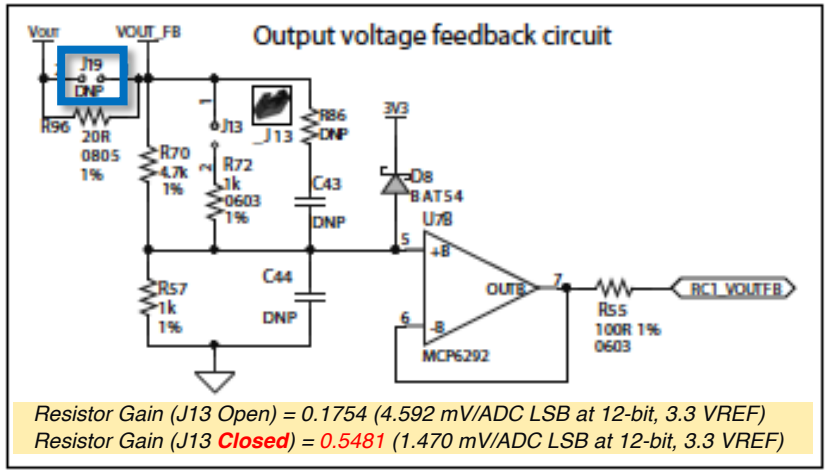




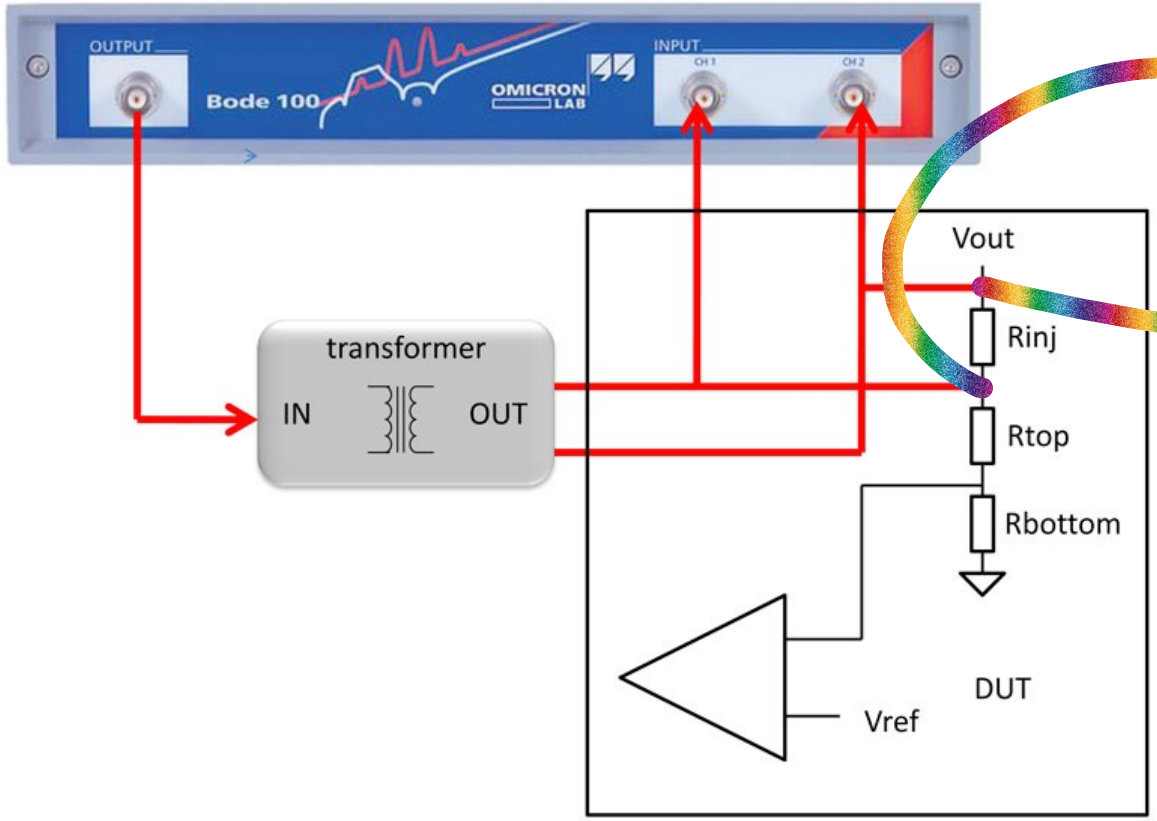
Plant Measurement Using Bode-100

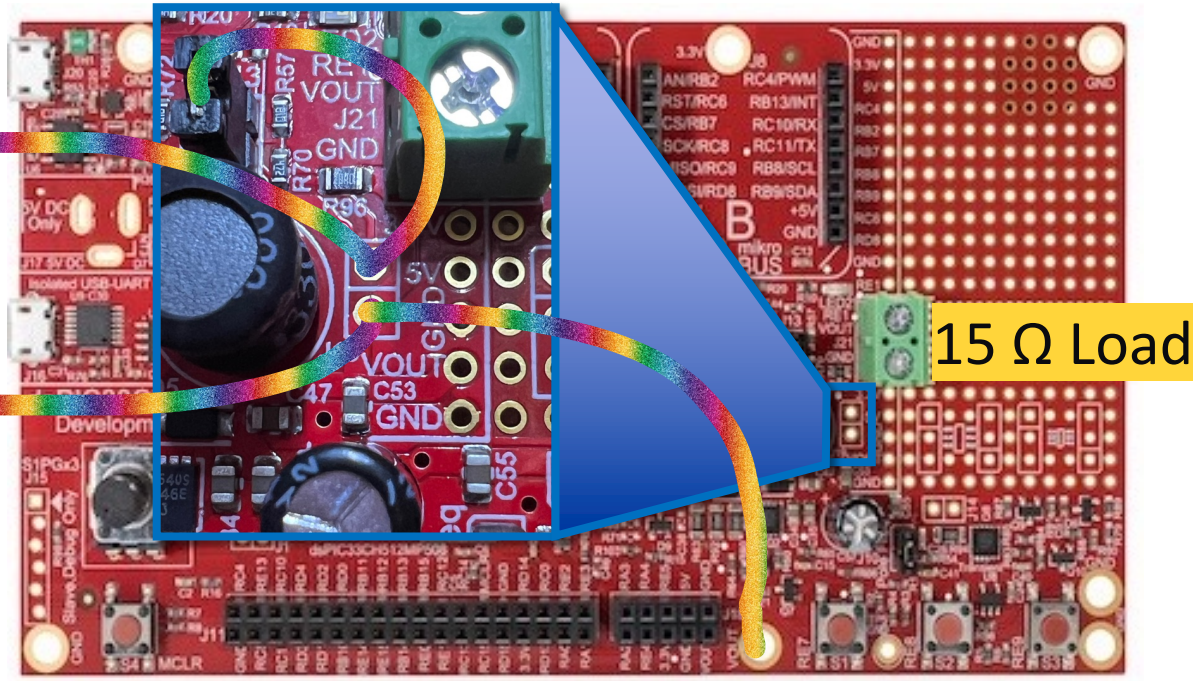


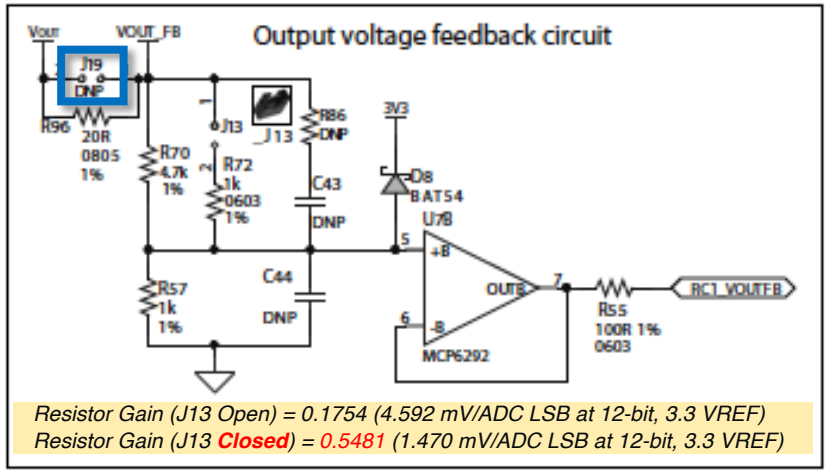




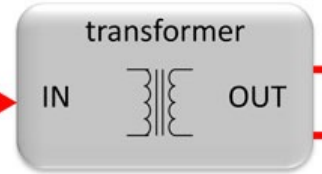
Plant Measurement Using Bode-100

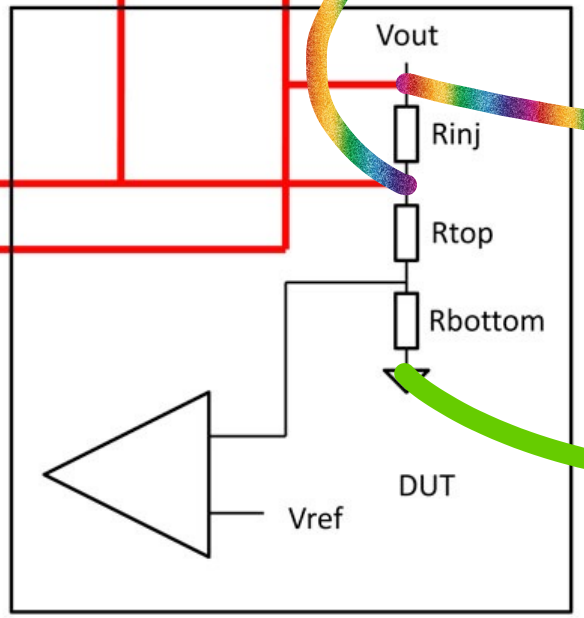


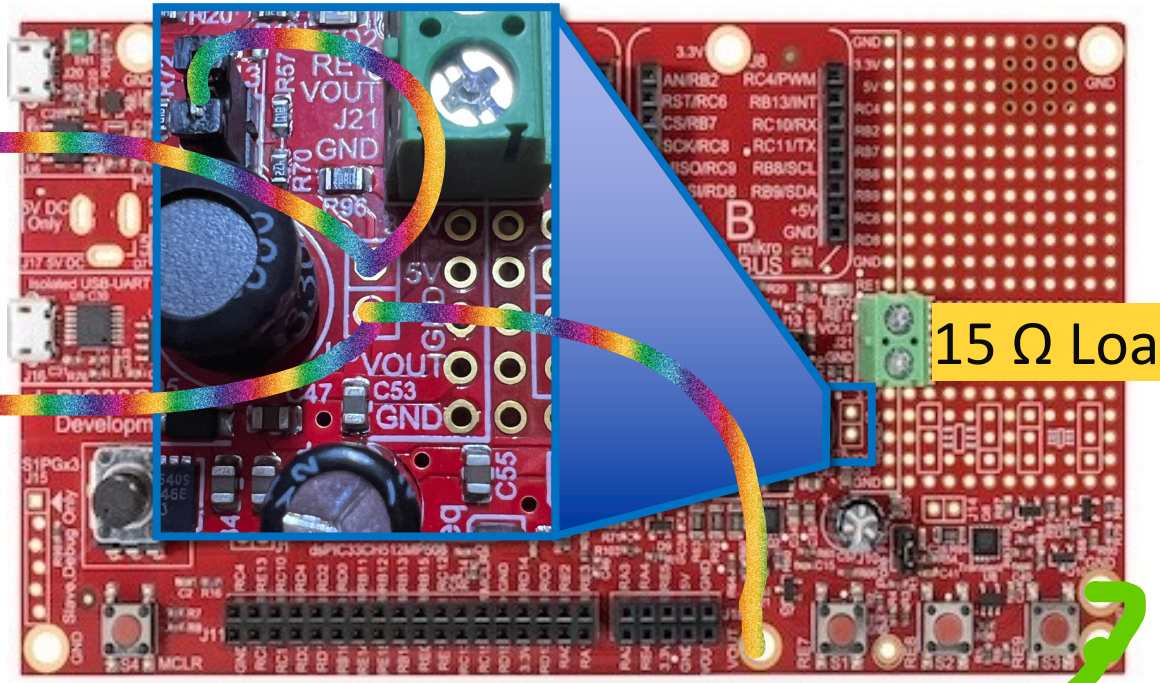


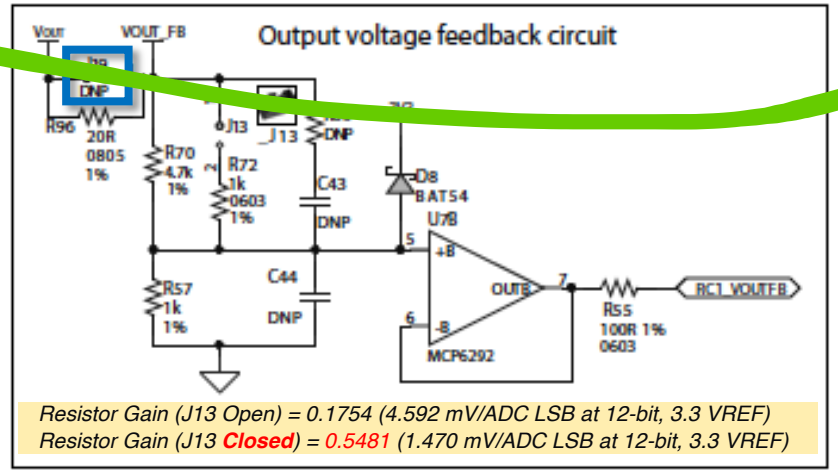


Plant Measurement Using Bode-100

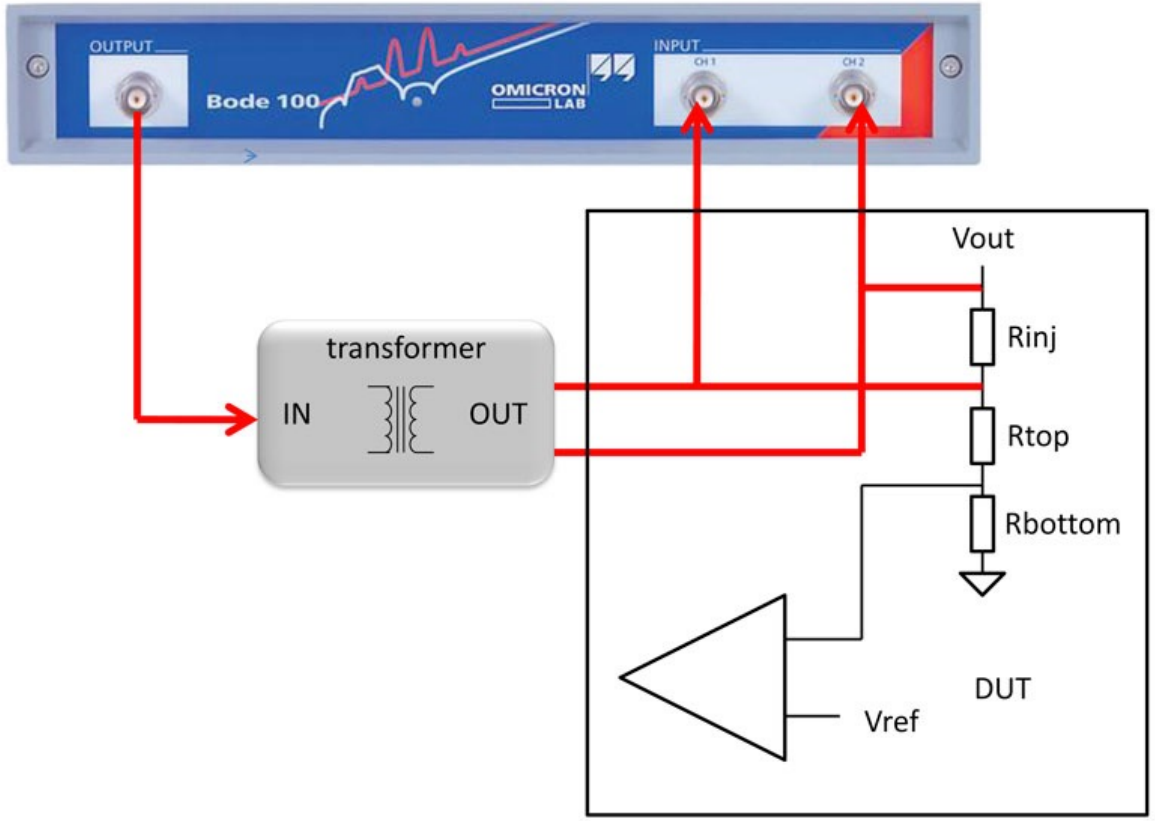




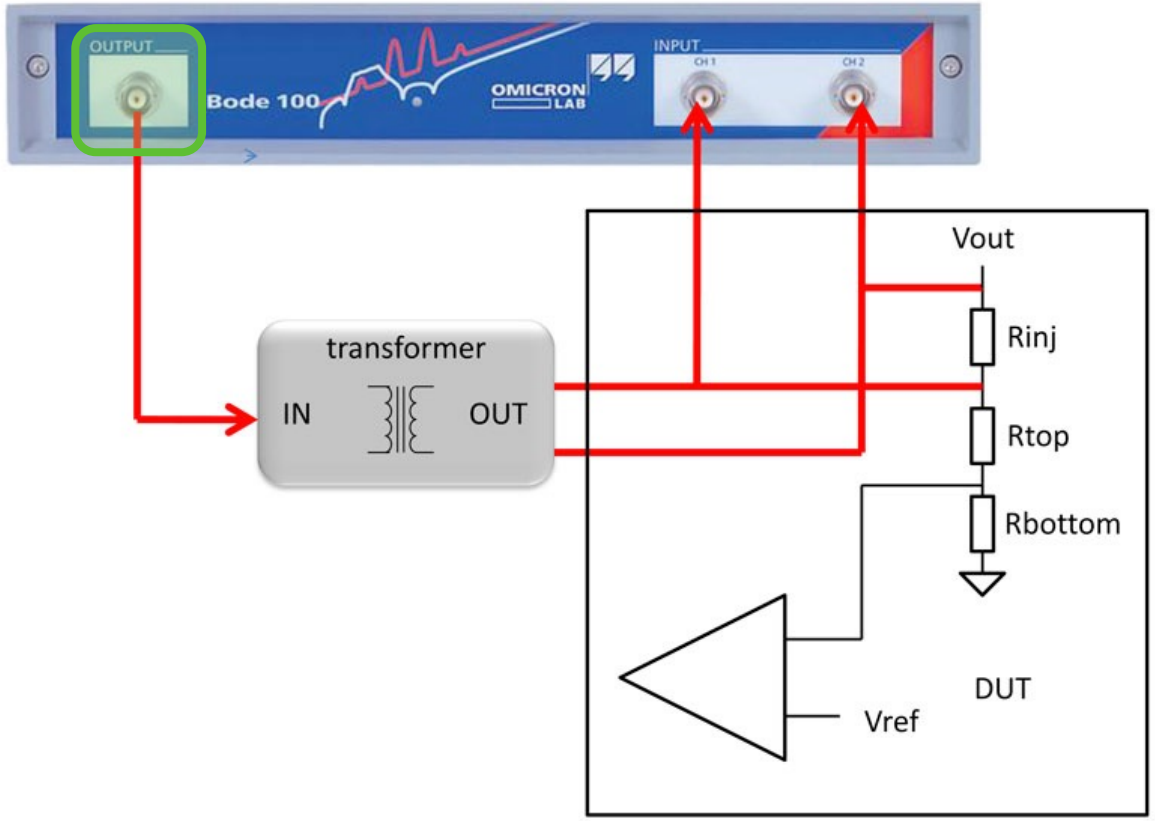




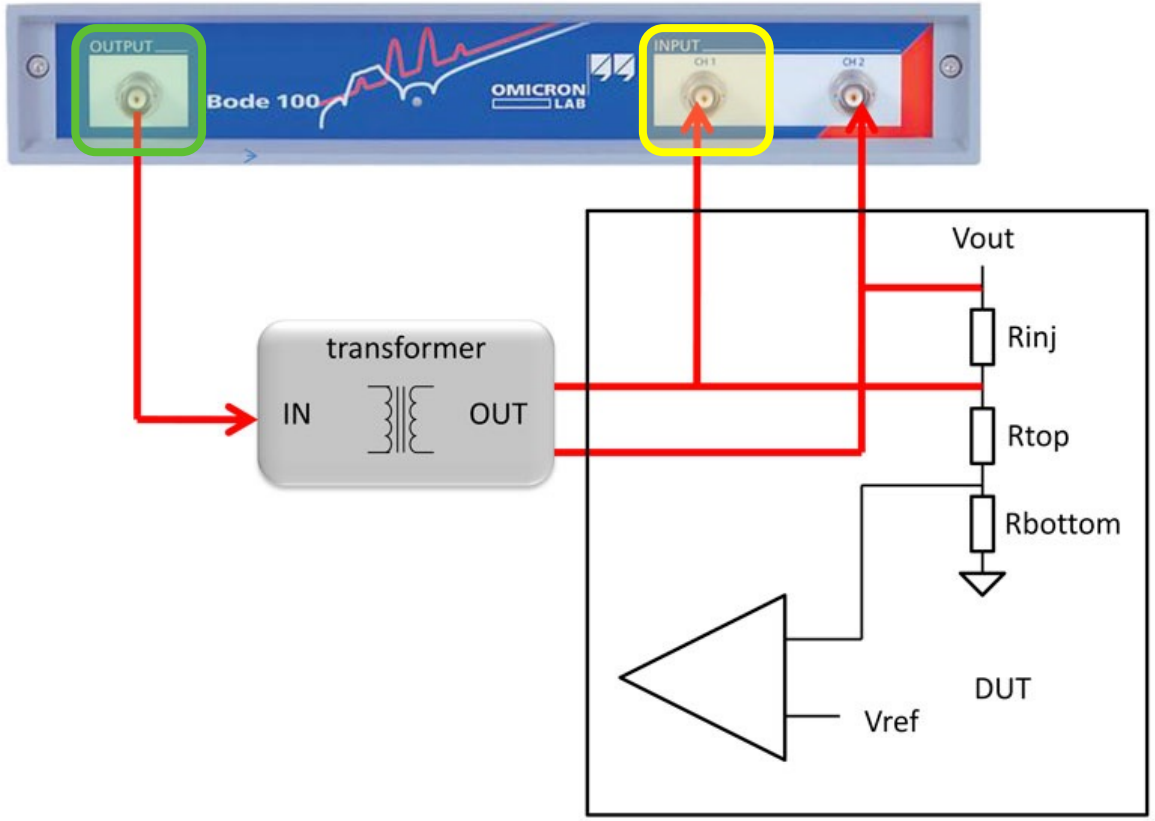
Plant Measurement Using DHO924S



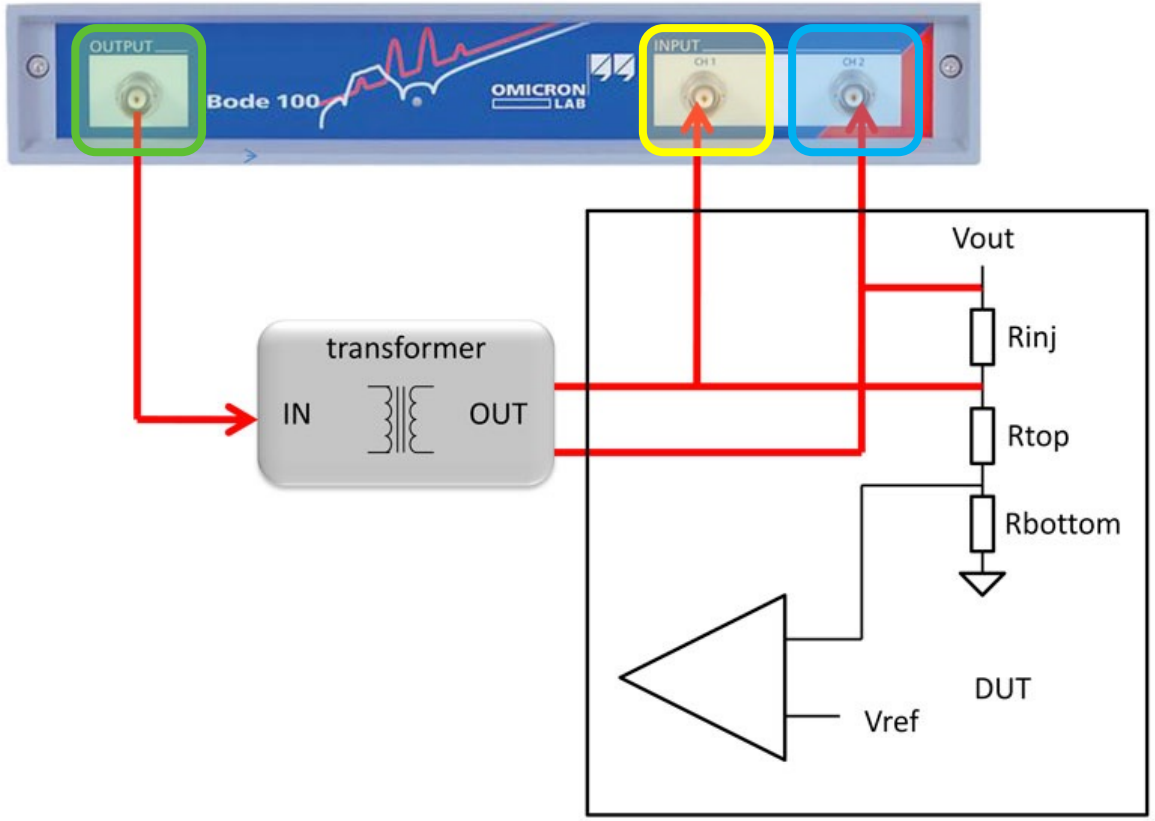
Plant Measurement Using DHO924S



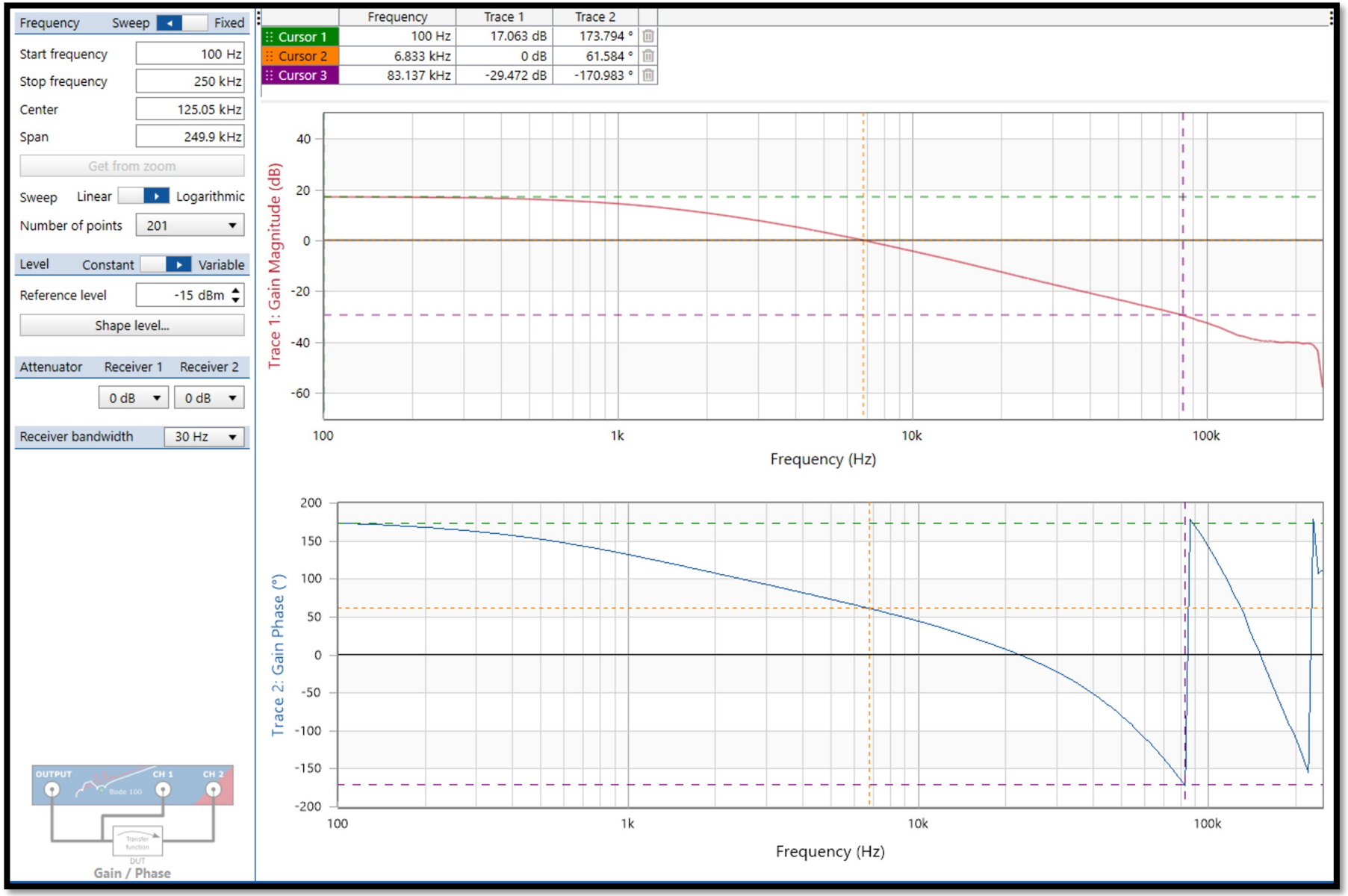
Plant Measurement Using DHO924S



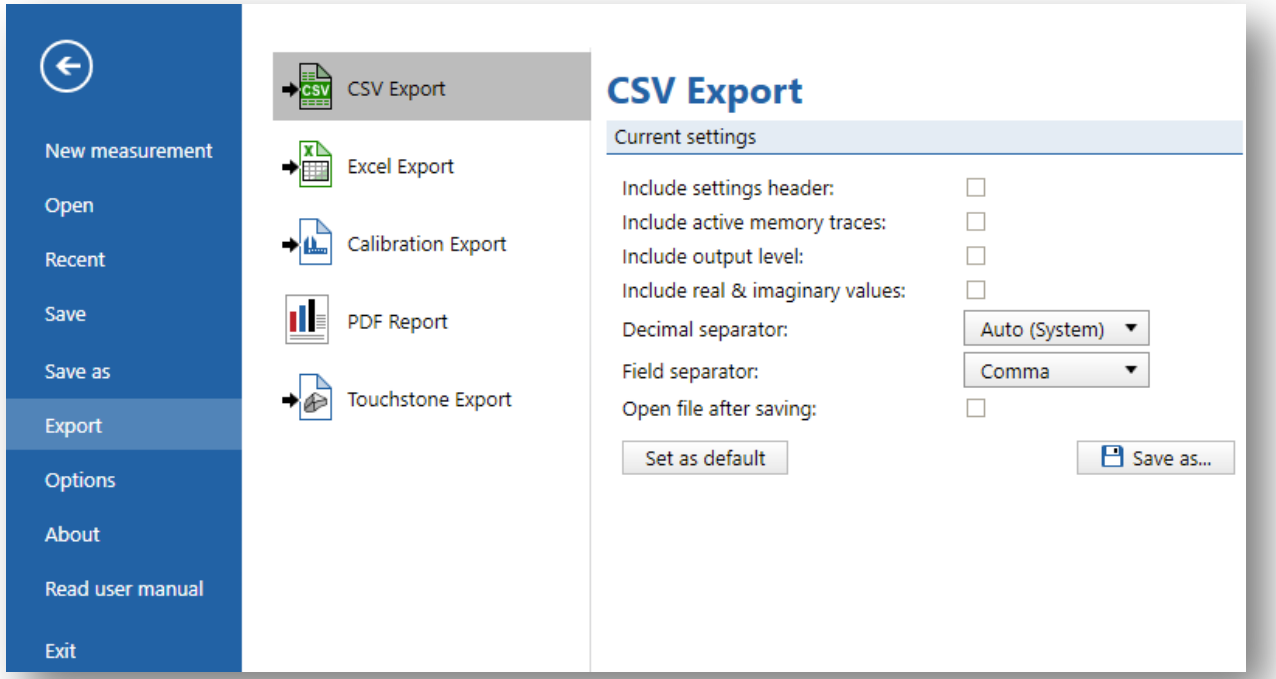
Plant Measurement Using DHO924S



Lab #3: Plant Measurement Using PowerSmart™



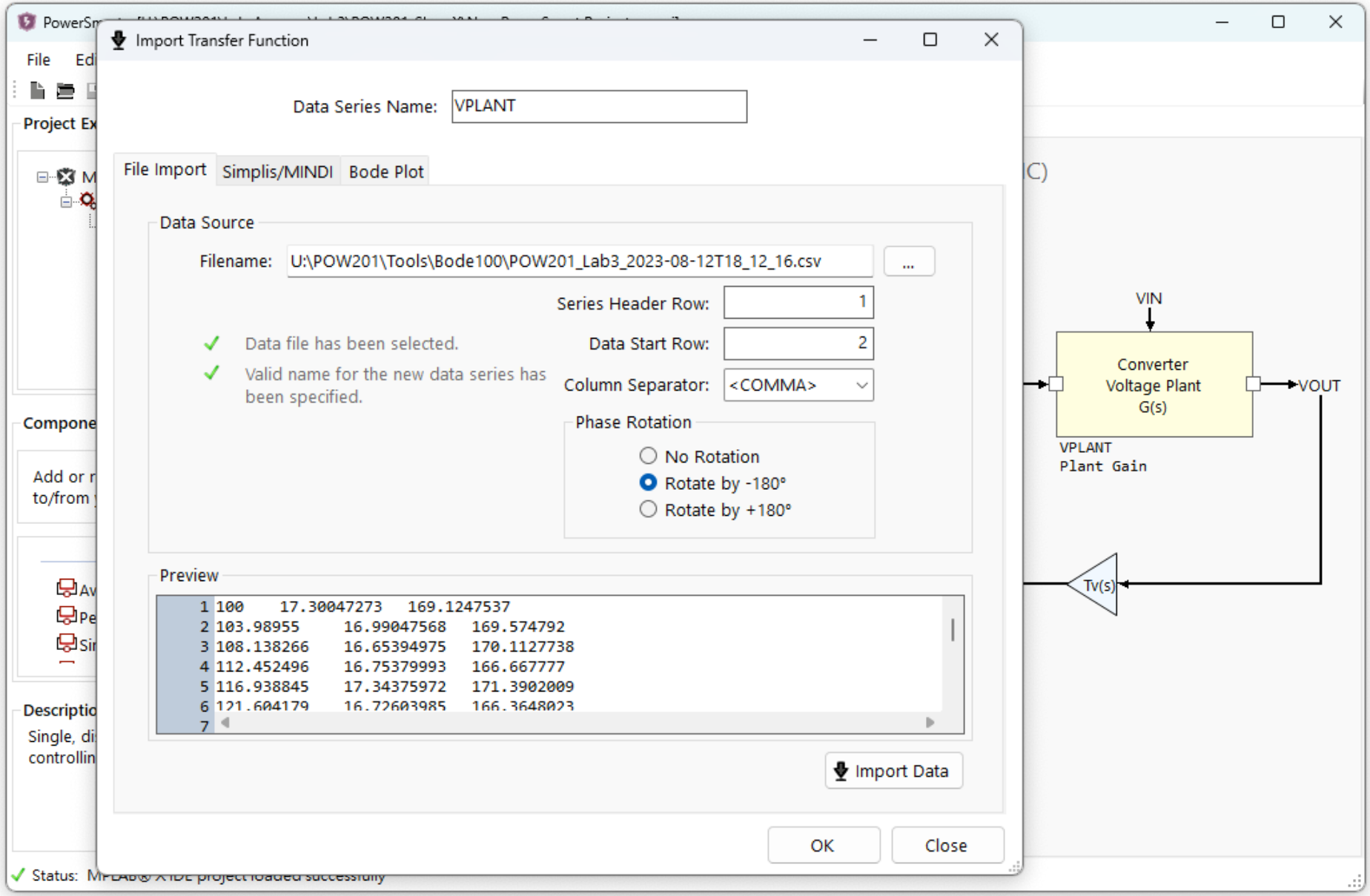
Export Bode Plot to CSV



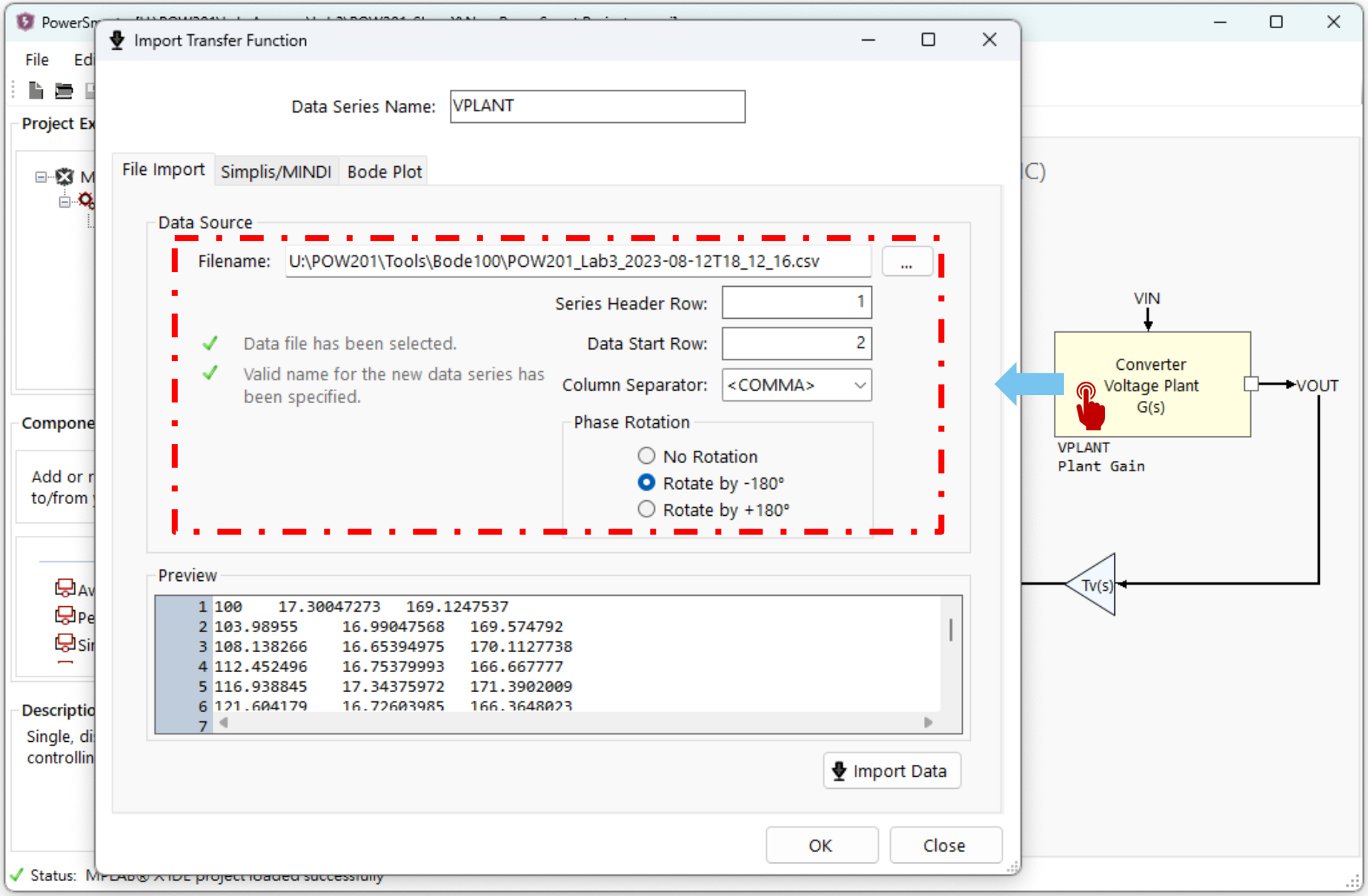
	A	B	C	D
1	Frequency (H	Trace 1: Gain	Trace 2: Gain: Phase (°)	
2	100	17.30047273	169.1247537	
3	103.98955	16.99047568	169.574792	
4	108.138266	16.65394975	170.1127738	
5	112.452496	16.75379993	166.667777	
6	116.938845	17.34375972	171.3902009	
7	121.604179	16.72603985	166.3648023	
8	126.455639	17.30457827	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!

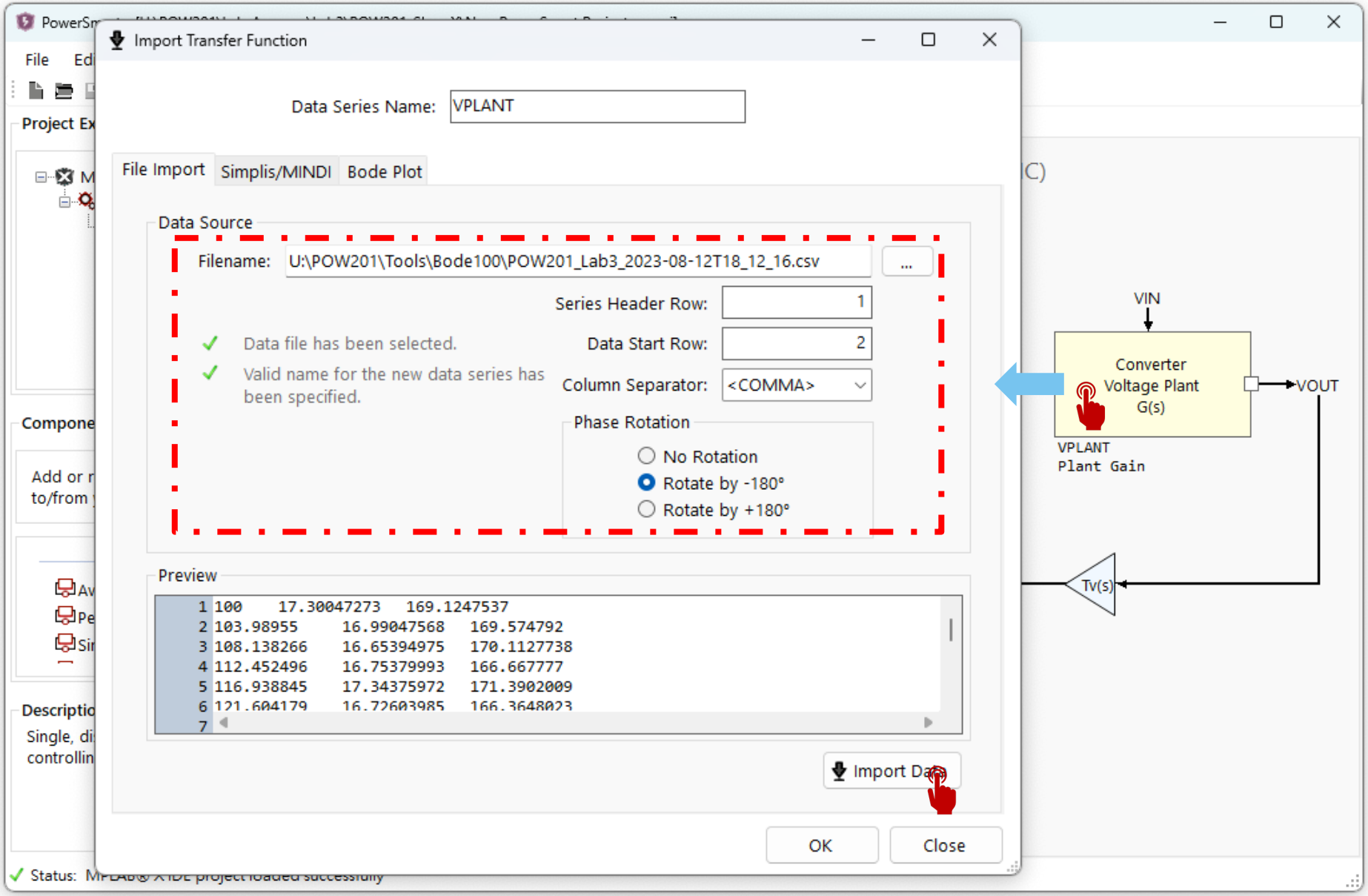
Lab #3: Plant Measurement Using PowerSmart™



Lab #3: Plant Measurement Using PowerSmart™



Lab #3: Plant Measurement Using PowerSmart™

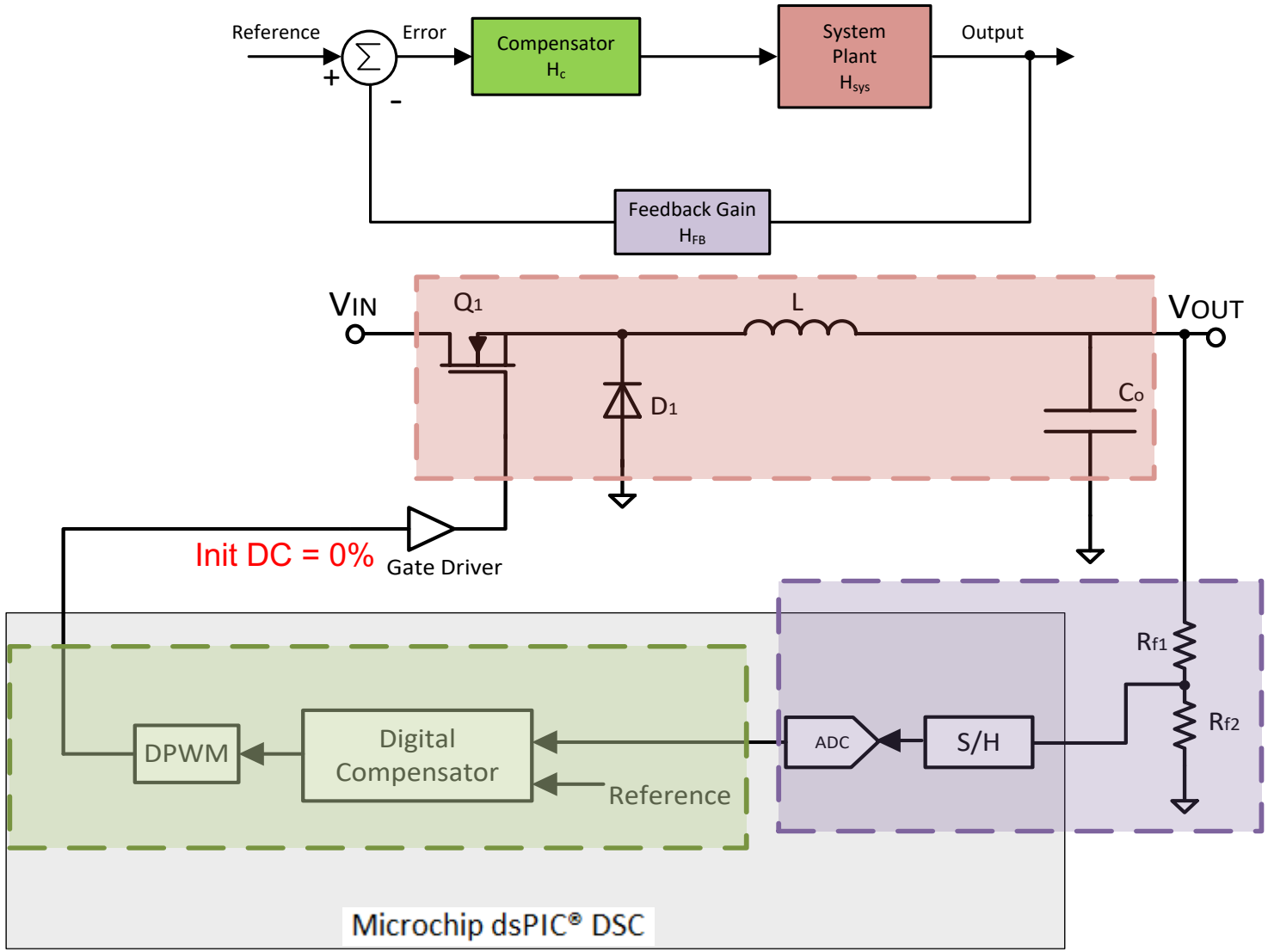


NOW, Actual Plant is Imported!!!

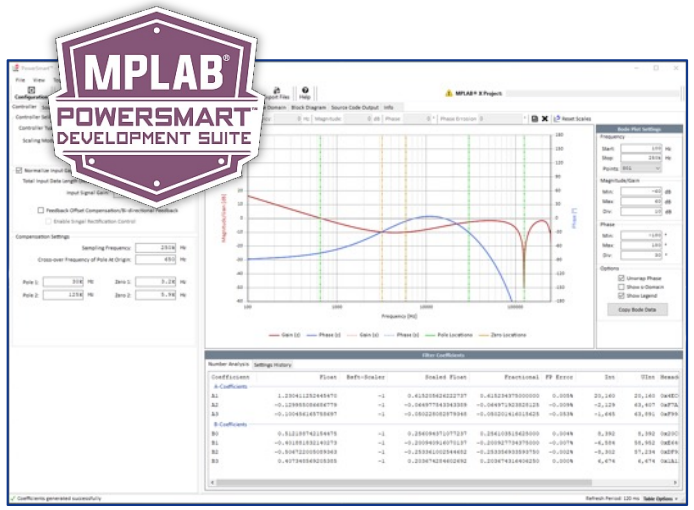
Closed-loop Control

Lab4

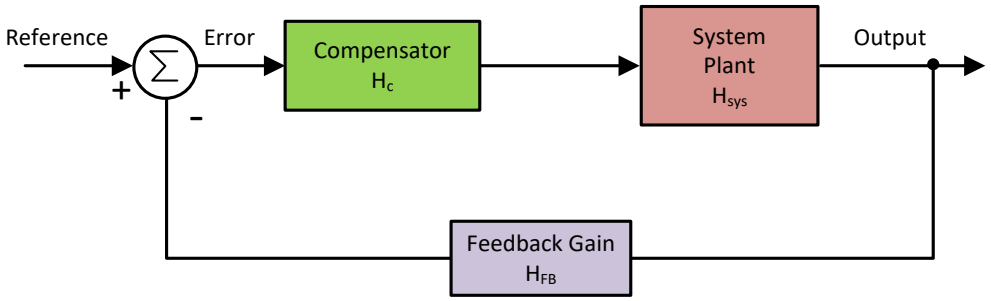
Lab #4: Closed-loop Control

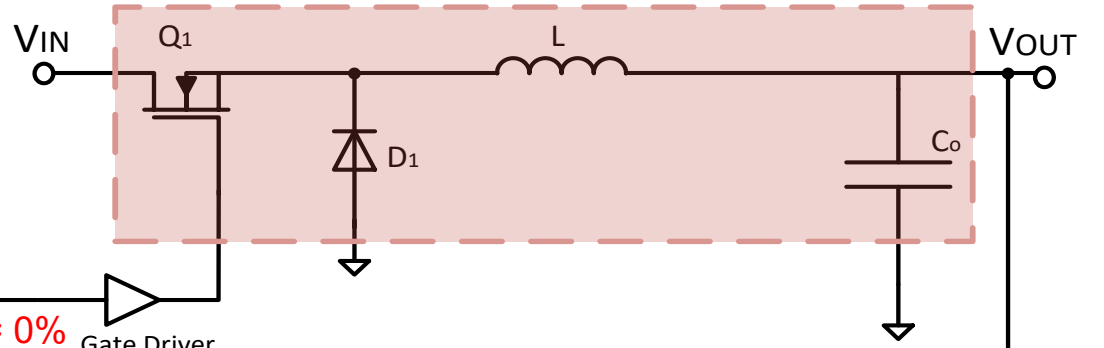


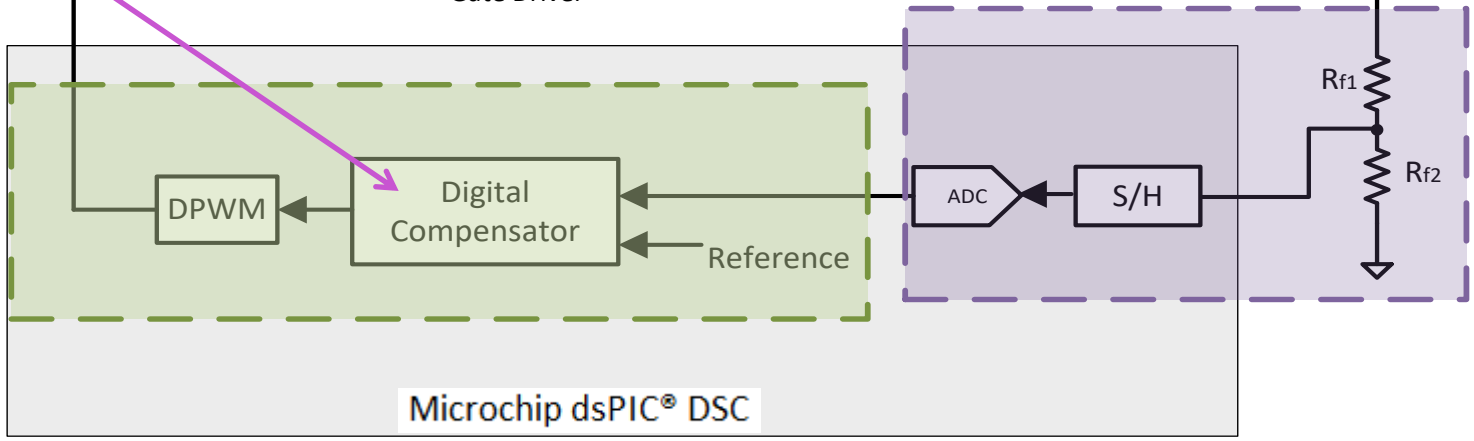
Lab #4: Closed-loop Control



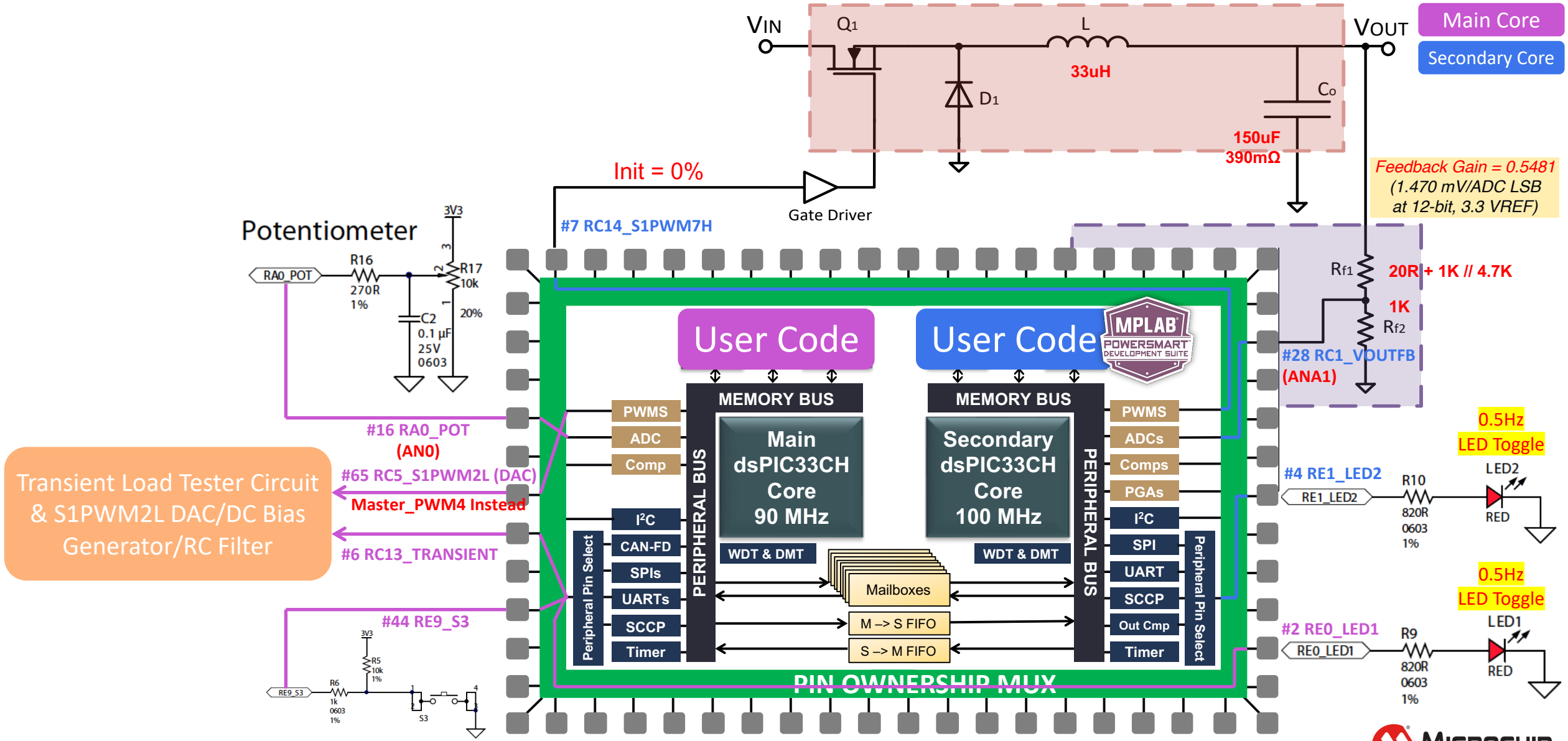
Closed-Loop Control





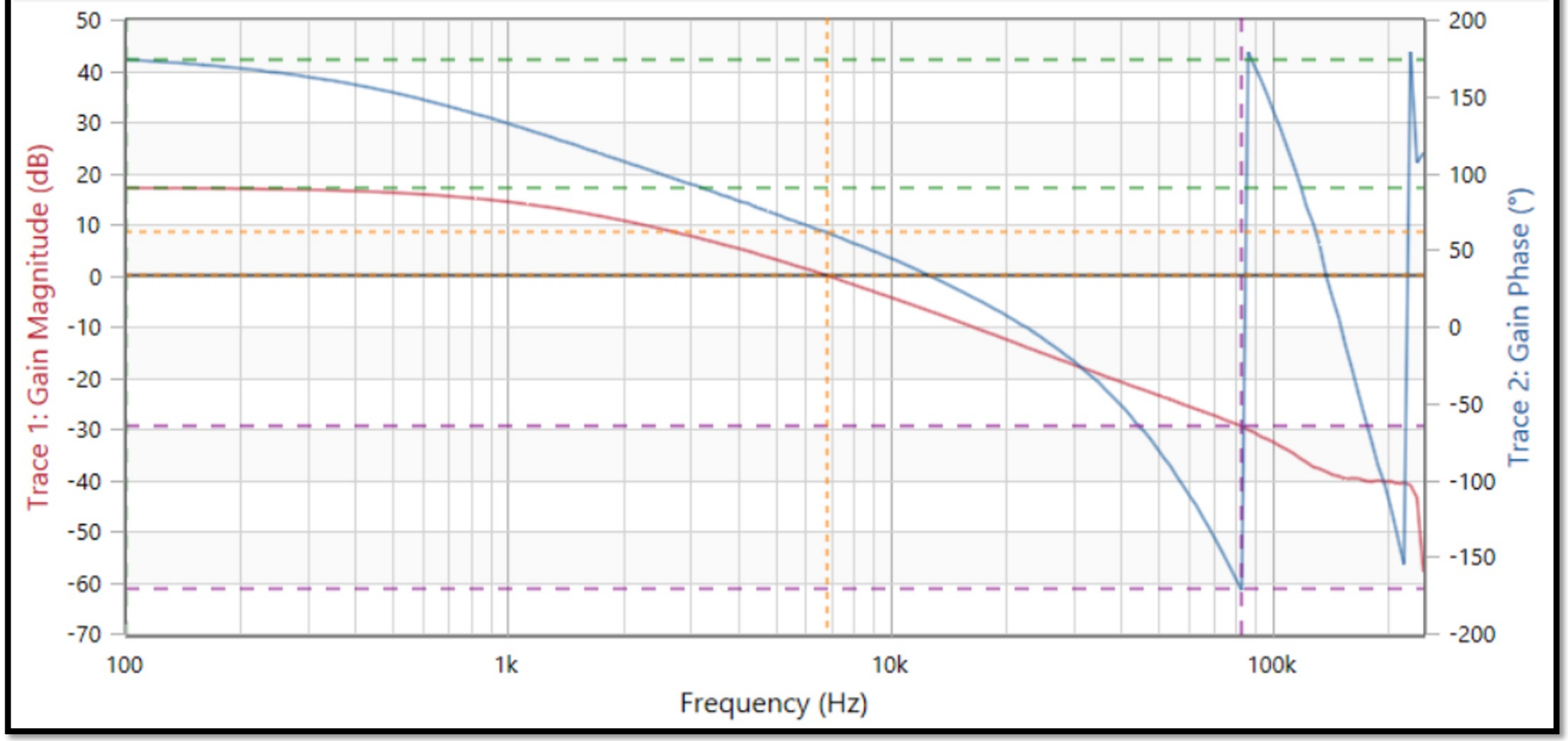


Lab #4: Closed-loop Control



The Actual Plant

	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 °



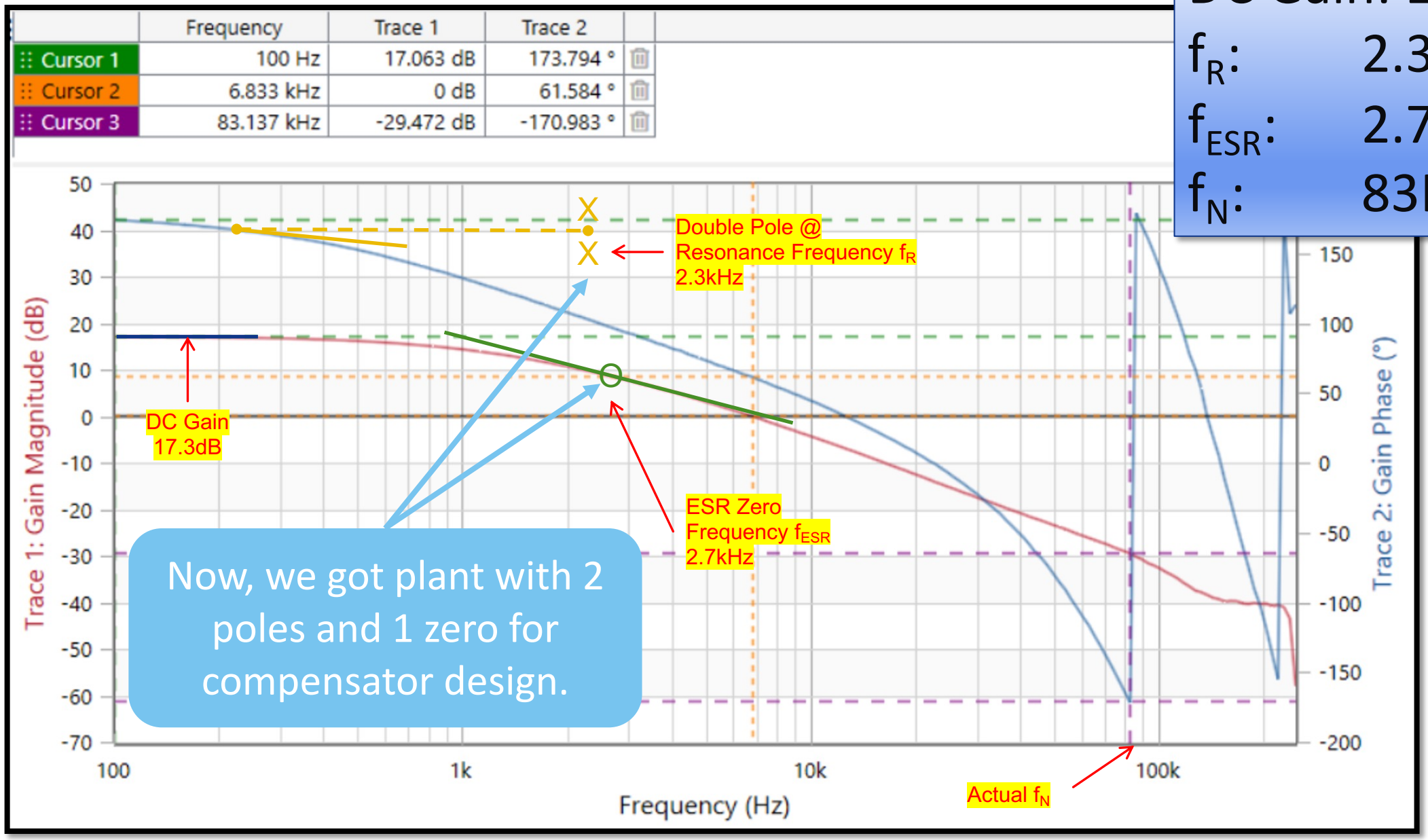
The Actual Plant

DC Gain: 17dB

f_R: 2.3Hz

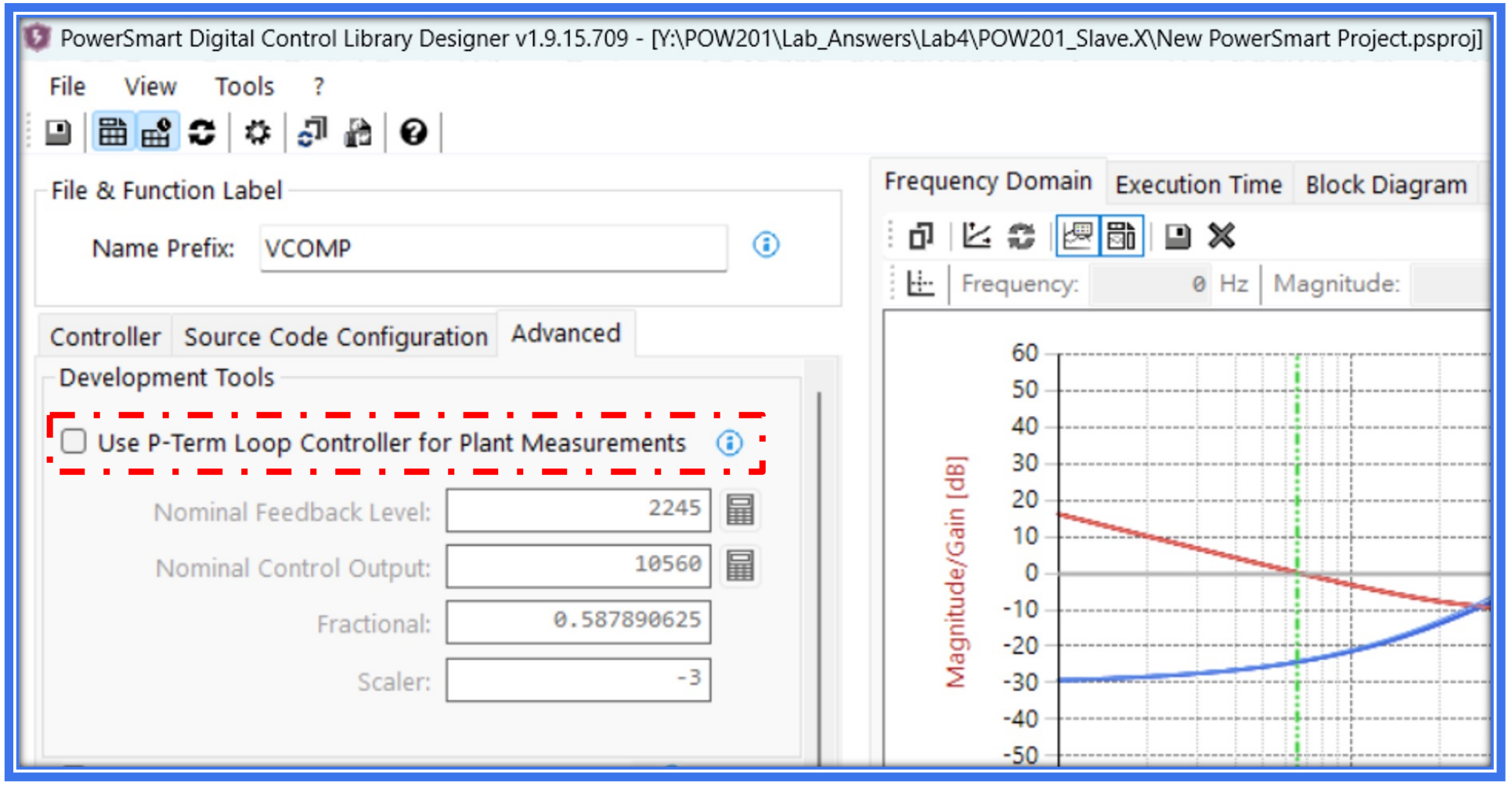
f_{ESR}: 2.7KHz

f_N: 83KHz

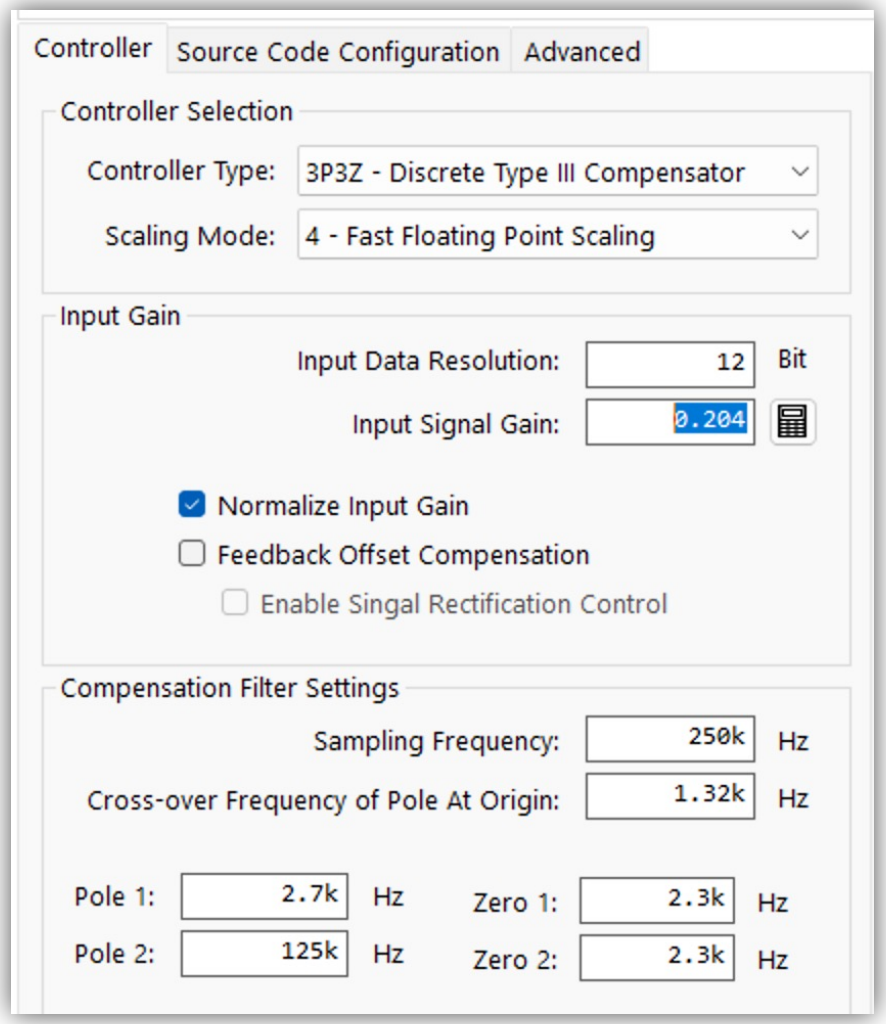


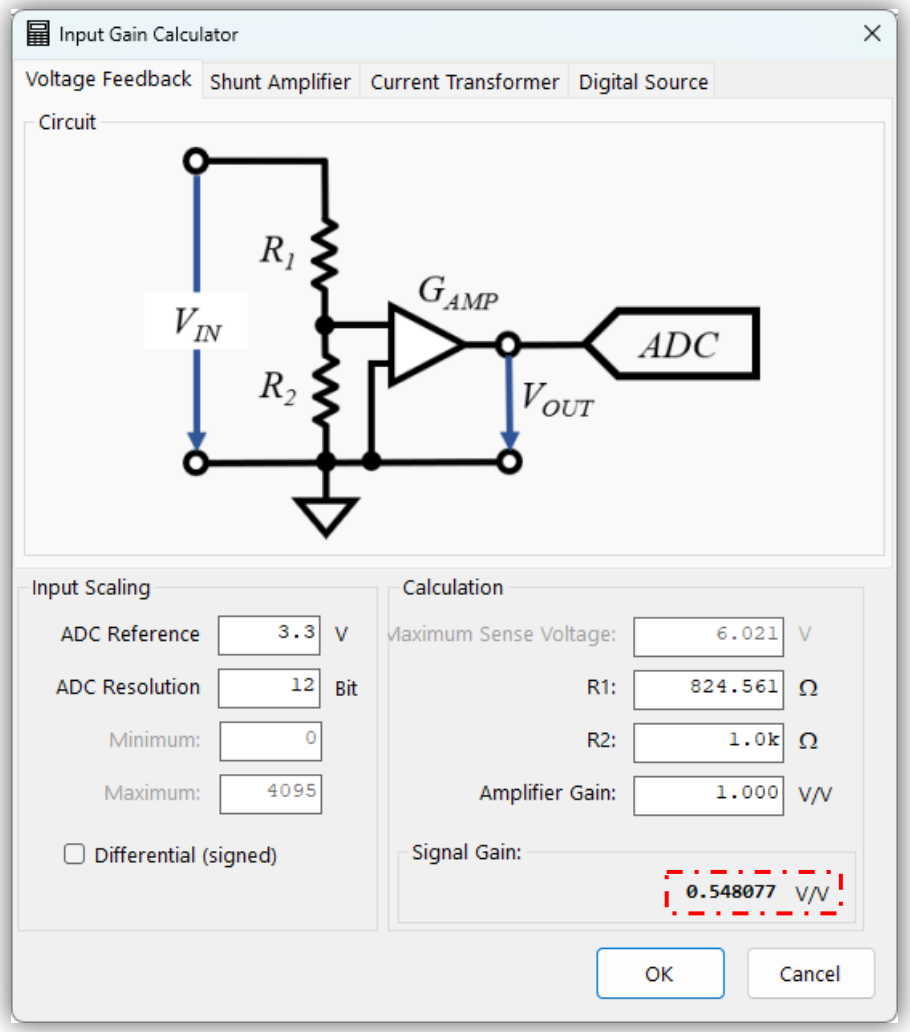
Uncheck Kp (P-Term) Control on DCLD

- **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

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

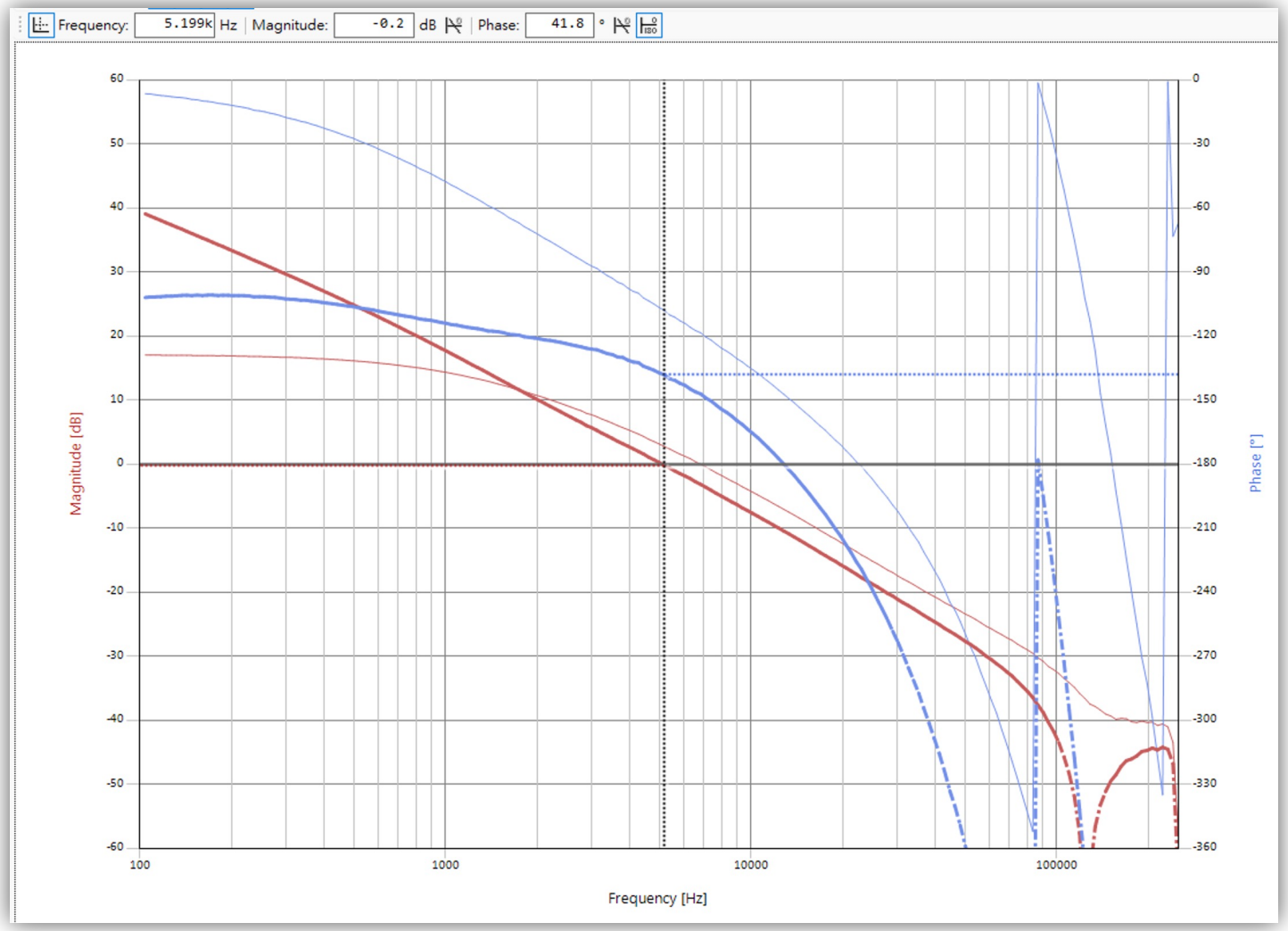
☐ Enable Singal Rectification Control

Compensation Filter Settings

Sampling Frequency: 250k Hz

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

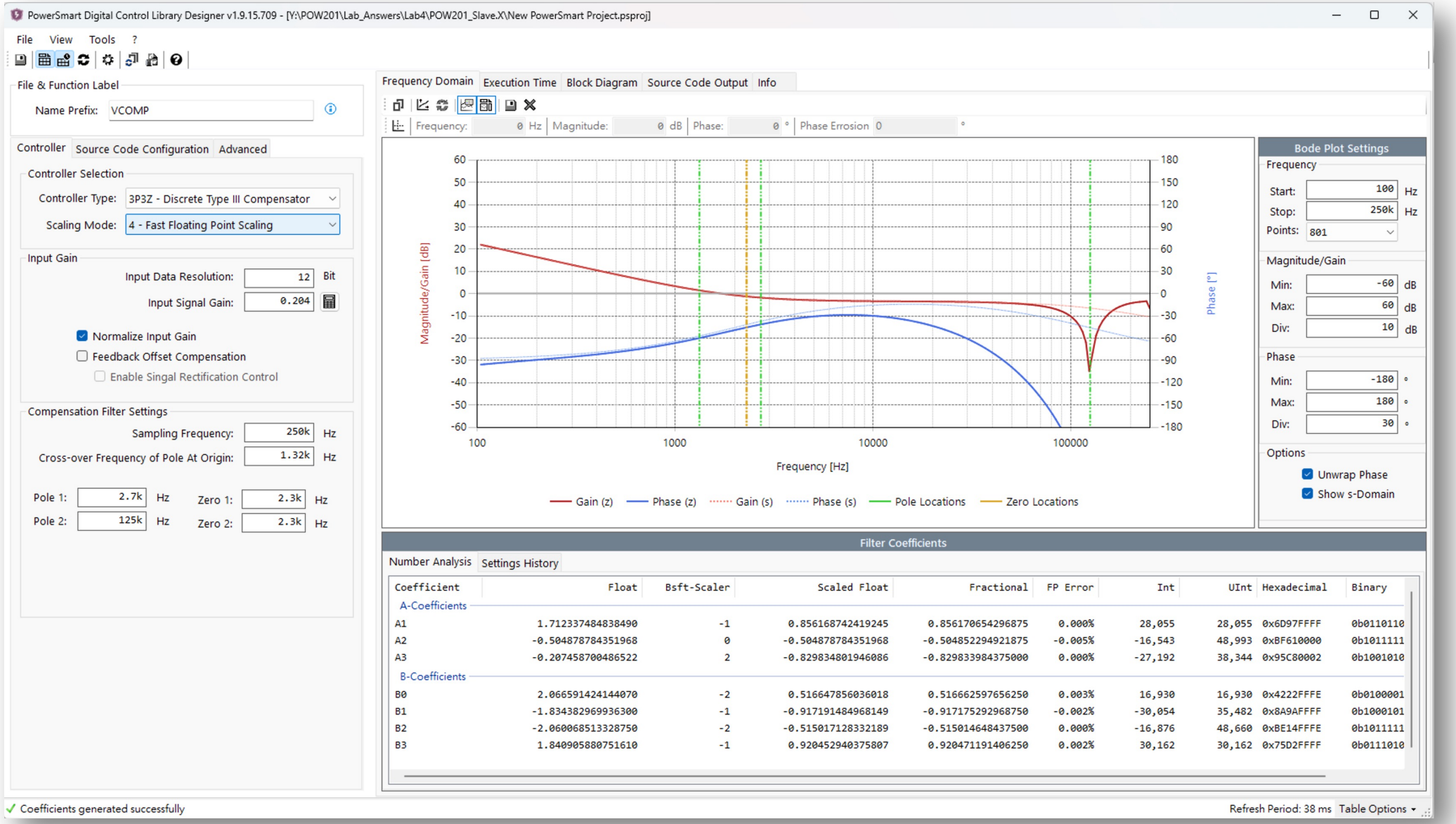
Pole 1:	2.7k	Hz	Zero 1:	2.3k	Hz
Pole 2:	125k	Hz	Zero 2:	2.3k	Hz



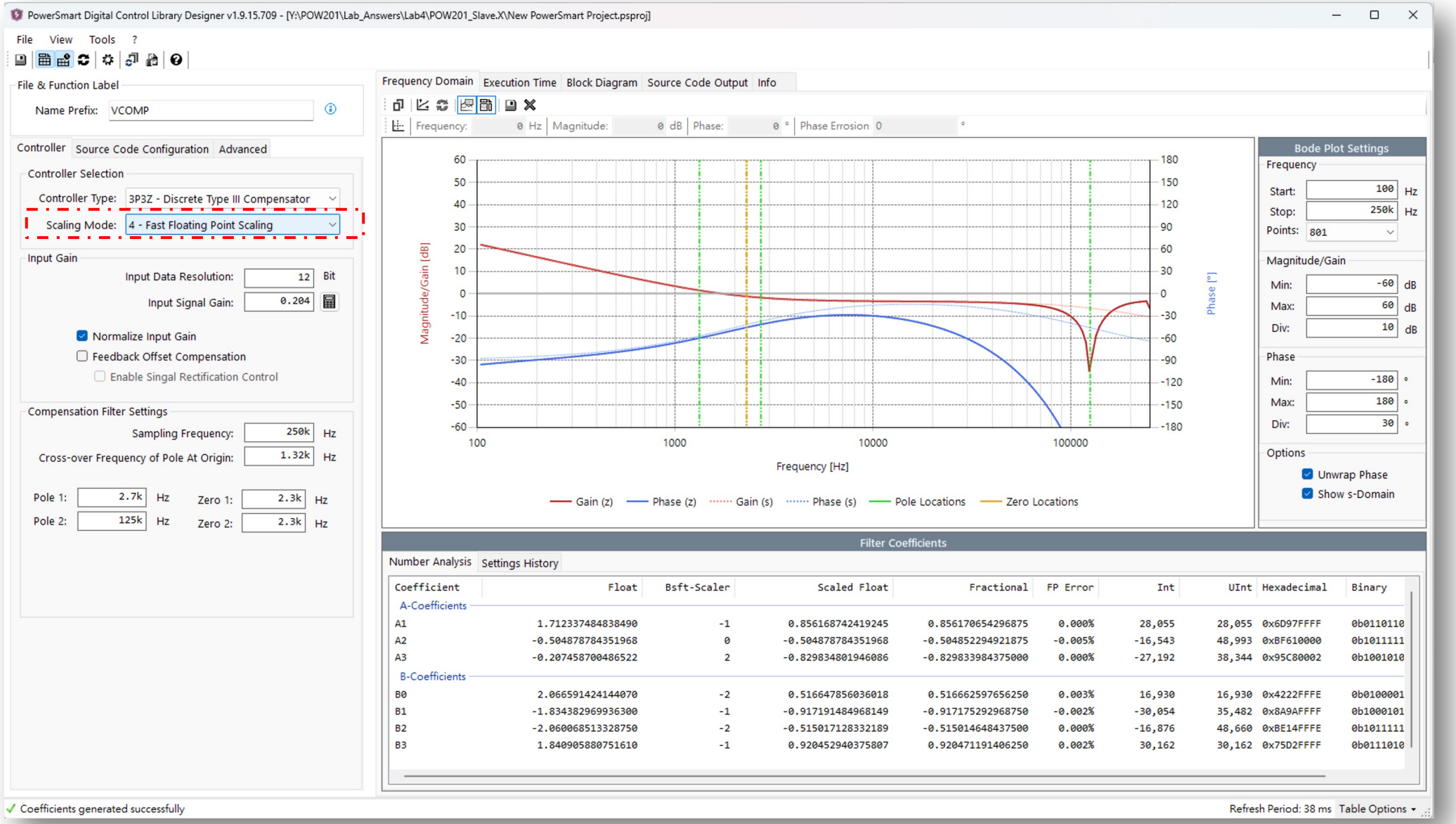
DCLD

PowerSmart™

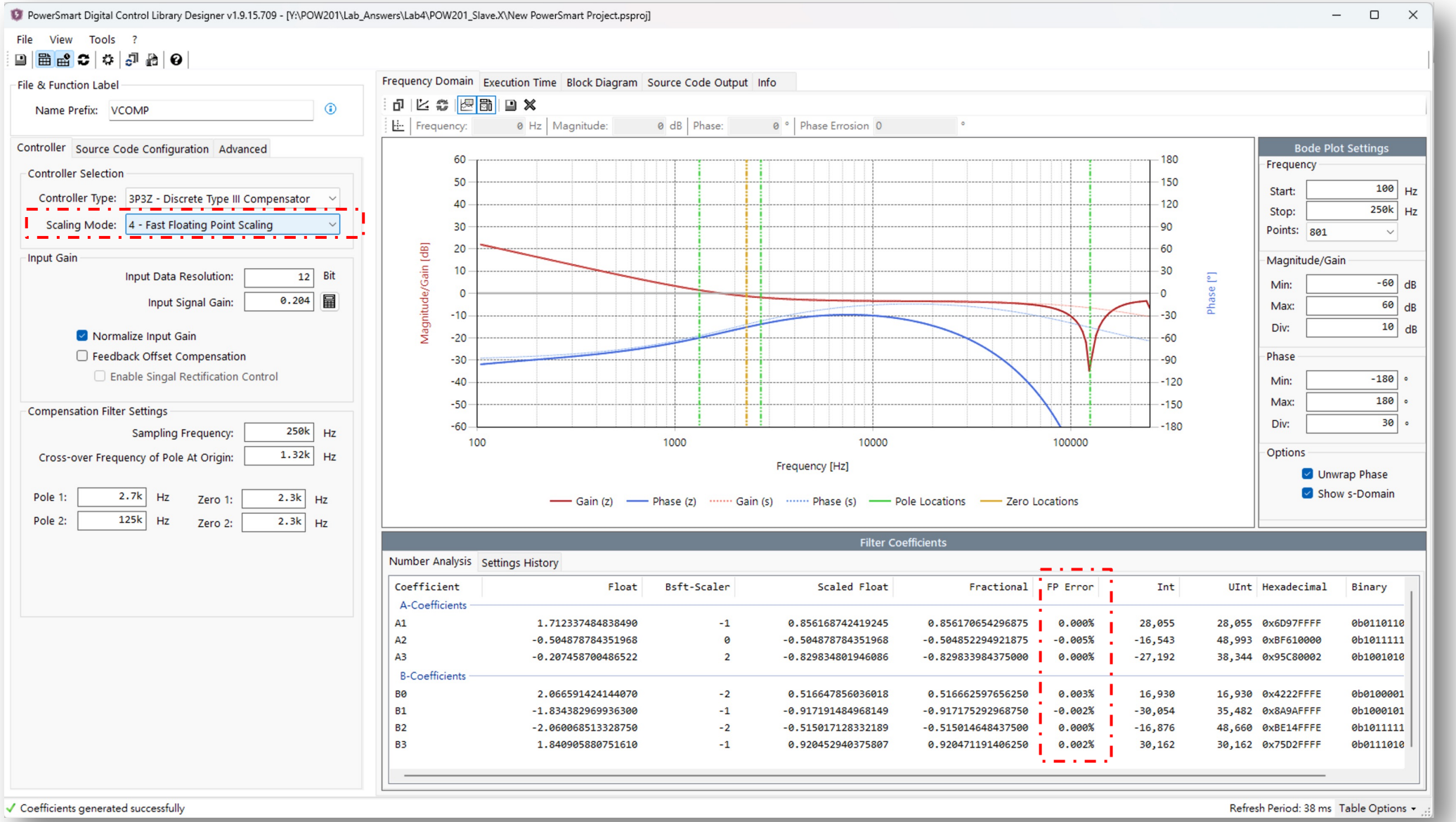
Check Coefficients with Suitable Scaling Mode



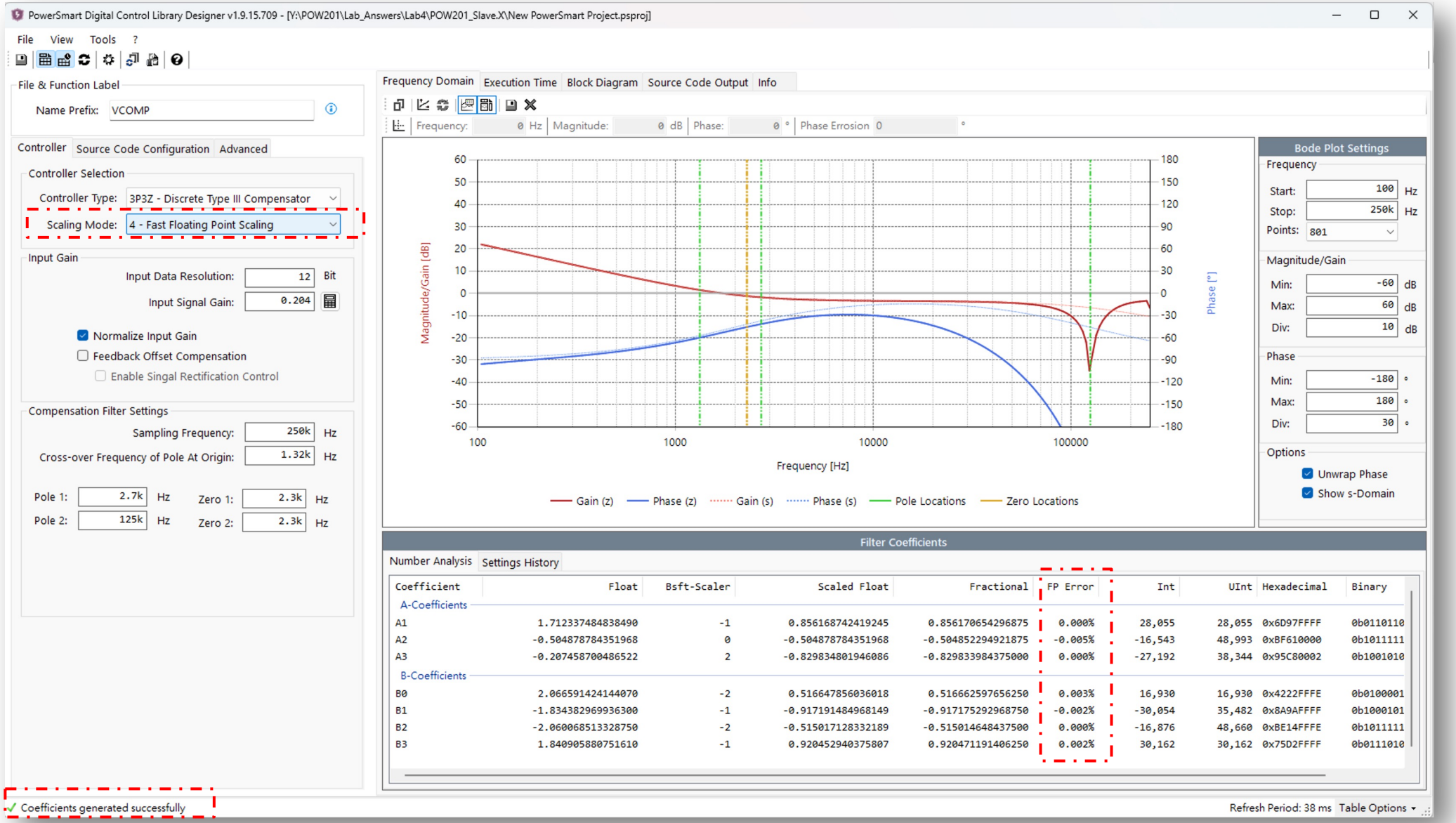
Check Coefficients with Suitable Scaling Mode



Check Coefficients with Suitable Scaling Mode

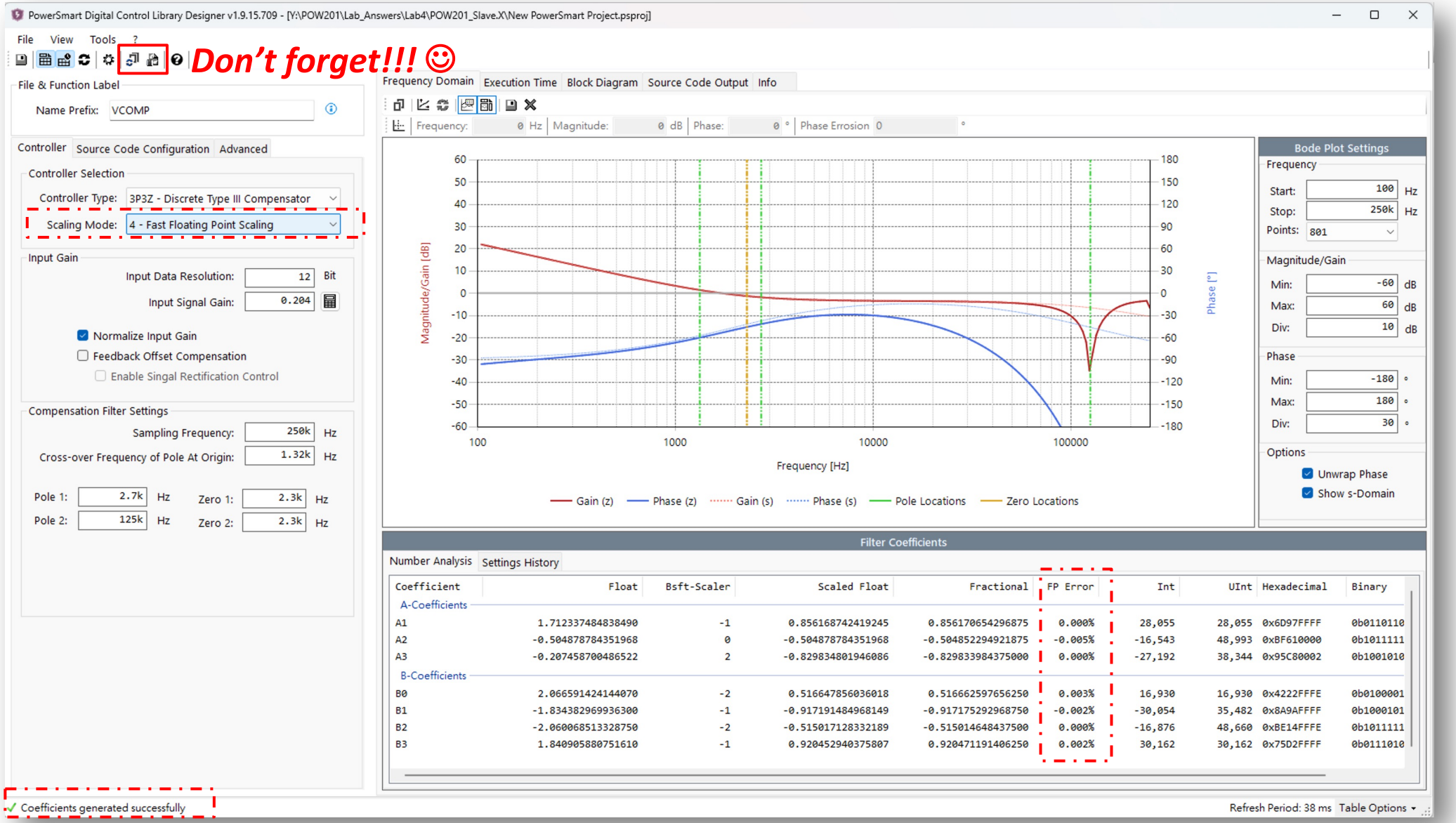


Check Coefficients with Suitable Scaling Mode



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

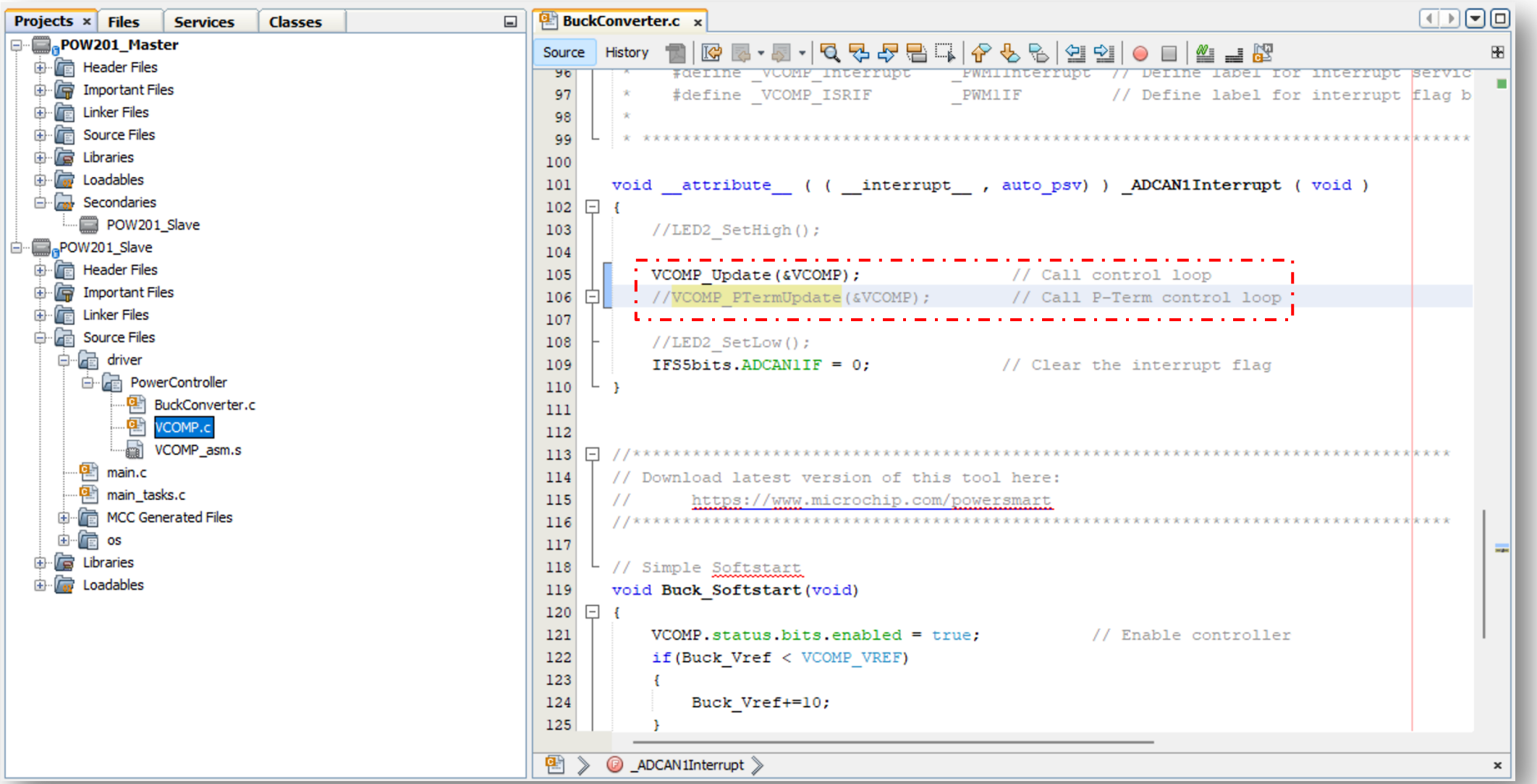
Check Coefficients with Suitable Scaling Mode



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

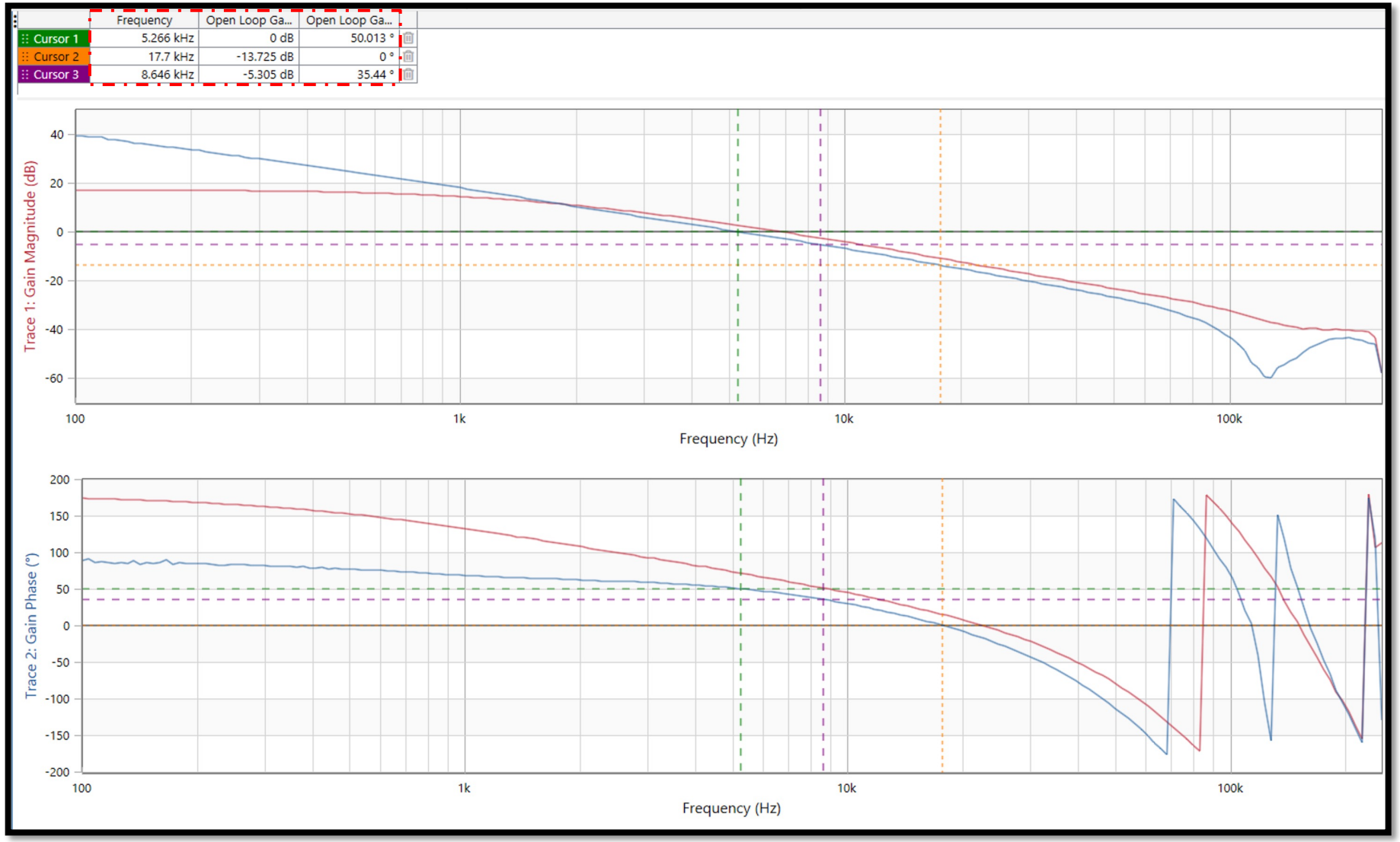
Lab #4: Closed-loop Control

Switch to closed-loop control in AN1 ISR



Lab #4: Closed-loop Control

Bode Plot Measurement

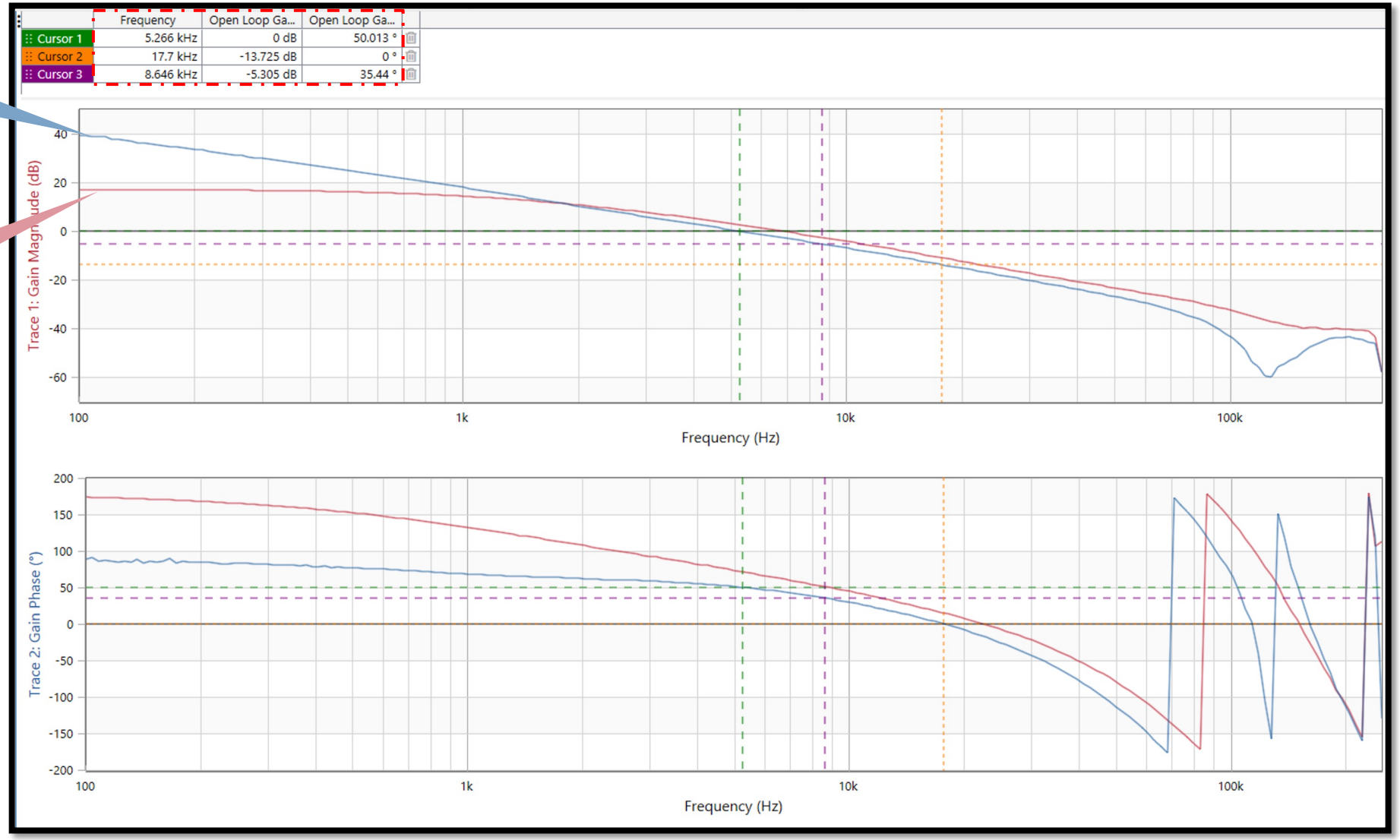


Lab #4: Closed-loop Control

Bode Plot Measurement

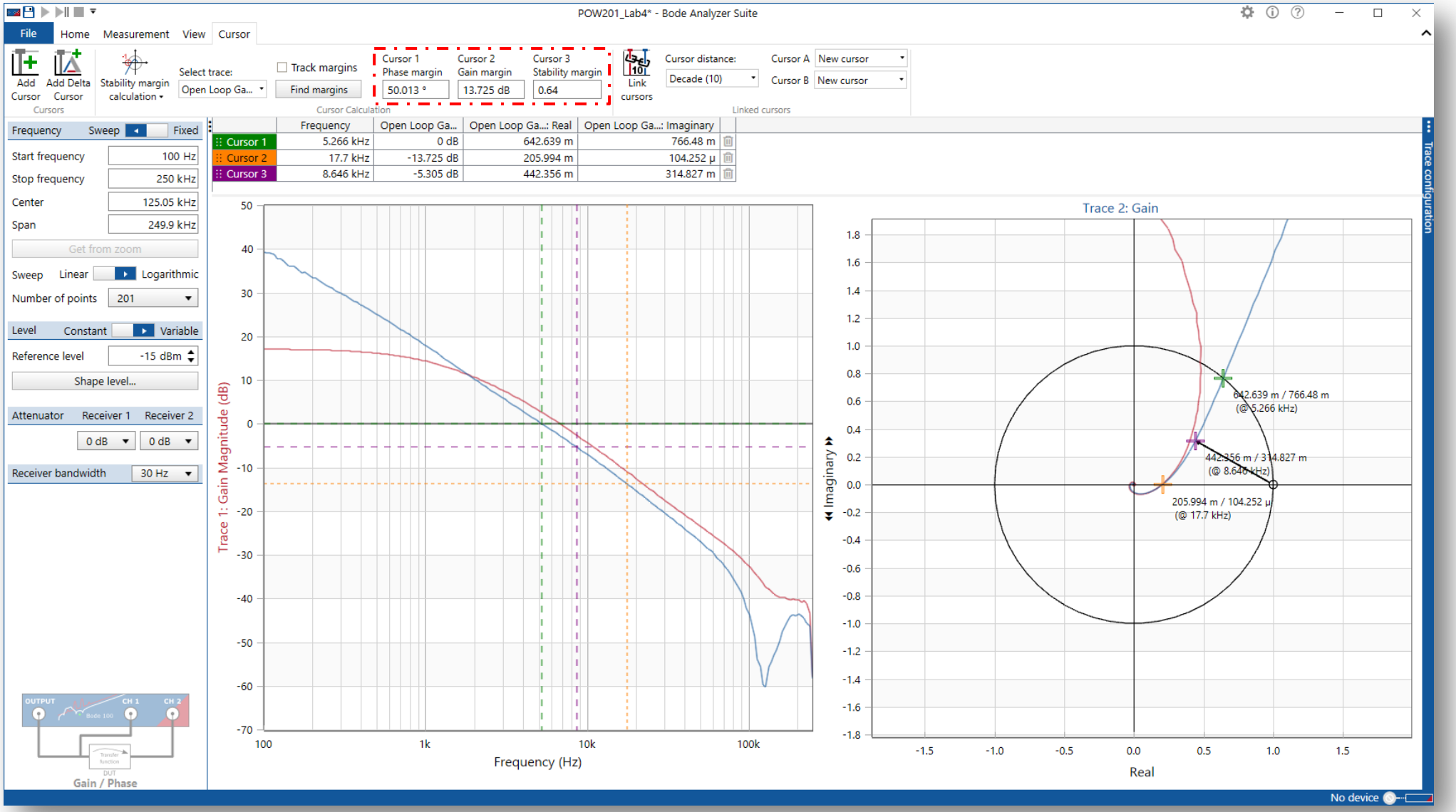
OL TF

Plant

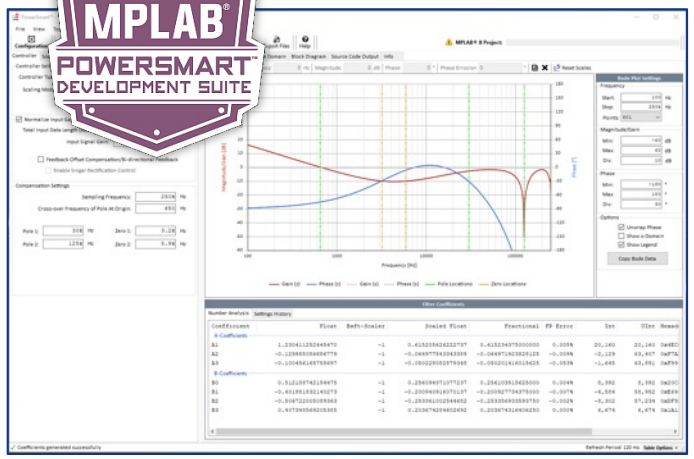


Lab #4: Closed-loop Control

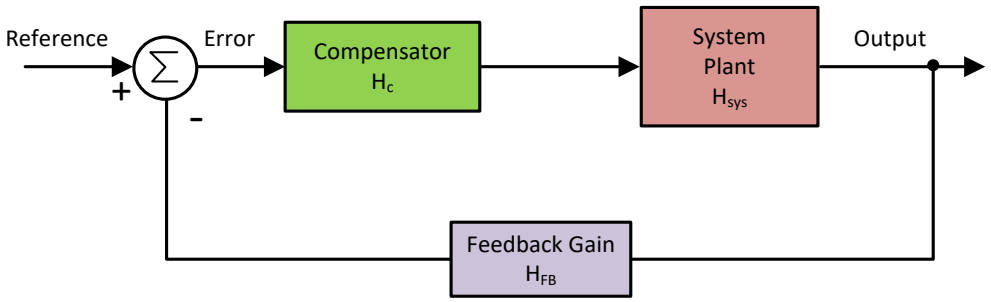
Stability Margin & Nyquist Plot



Now, You have known that …



Step3. Closed-Loop Control

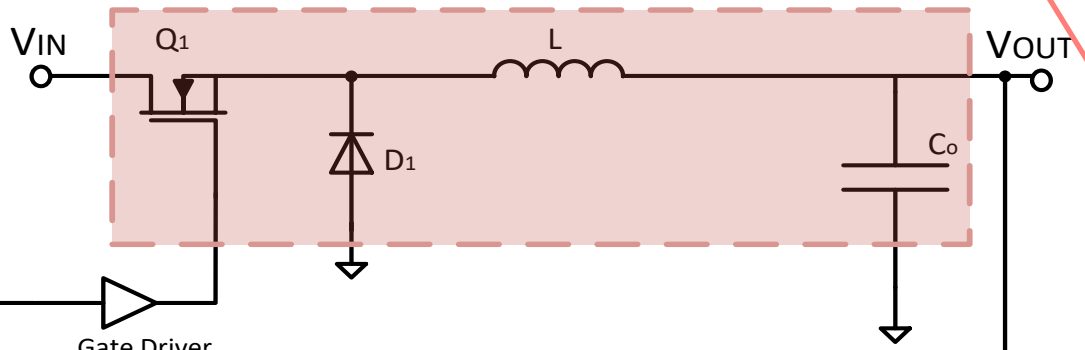


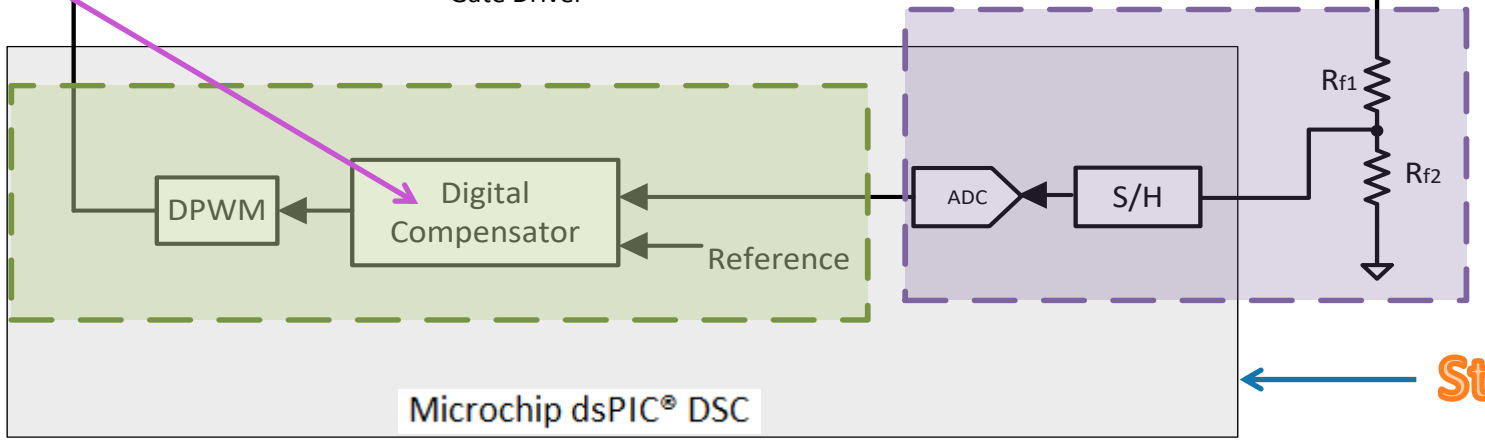
Step2. Plant Modeling

MPLAB MINDI ANALOG SIMULATOR — Simulation

OR

MPLAB POWERSMART DEVELOPMENT SUITE — Measurement Using P-term Control





MPLAB CODE CONFIGURATOR

Step1. MCC Codebase

May The *Power* Be With You