GaN OBC SW User Document

Document Version : 265
Document Owner : Manish Bhardwaj

1 Hardware Required

https://www.ti.com/tool/TMDSCNCD28388D

Note some control cards for RevA have 25MHz crystal and some have 20MHz crystal, there is no way to tell this otherwise. Rev B are all 25MHz.

An error checking routine is included in the code to make sure you are not able to run the code at an incorrect clock.

The crystal value is defined in the device.h file

#define USE_20MHZ_XTAL

Make sure the **SW3 and SW4 are pointing to the left.**

3 Pin & Peripheral Usage Mapping

	Peripheral	Description	**HSEC Pin Number**	**F2838x**	F28003x	F28P65x
SYSTEM ISR Trigger	ECAP	Used to decouple the freq of PFC from ISR allows to do spread spectrum on the PFC freq if need be	NA	**ECAP1**	ECAP1	ECAP1
CLLLC_CONTROL_OUTPUT_DAC_PIN	DAC		14	**DACC**	**DACA?**	DACC

CLLLC_PRIM_LEG1_H/L	EPWM		49, 51	EPWM1 (A/B)	EPWM1 (A/B)	EPWM1 (A/B)
CLLLC_PRIM_LEG2_H/L	EPWM		53, 55	EPWM2 (A/B)	EPWM2 (A/B)	EPWM2 (A/B)
CLLLC_SEC_LEG1_H/L	EPWM		50, 52	EPWM3 (A/B)	EPWM3 (A/B)	EPWM3 (A/B)
CLLLC_SEC_LEG2_H/L	EPWM		54, 56	EPWM4 (A/B)	EPWM4 (A/B)	EPWM4 (A/B)
CLLLC_FAULTn	GPIO		74	GPIO-23 → INPUTXBAR2	GPIO-23 → INPUTXBAR2	GPIO-23 → INPUTXBAR2
CLLLC_LC_CHANGE	GPIO		62	GPIO-14	GPIO-14	GPIO-14
CLLLC_SEC_SIDE_DIAG	GPIO		80	GPIO-30	GPIO-30	GPIO-30
TTPLPFC_LOW_FREQ_H/L	EPWM		57, 59	EPWM5 (A/B)	EPWM5 (A/B)	EPWM5 (A/B)
TTPLPFC_HIGH_FREQ_PH1_H/L	EPWM		61, 63	EPWM6 (A/B)	EPWM6 (A/B)	EPWM6 (A/B)

TTPLPFC_HIGH_FREQ_PH2_H/L	EPWM		58, 60	EPWM7 (A/B) (GPIO12/GPIO13)	EPWM7 (A/B)	EPWM7 (A/B)
TTPLPFC_FAULTn	GPIO		72	GPIO-22 → INPUTXBAR1	GPIO-22 → INPUTXBAR1	GPIO-22 → INPUTXBAR1
TTPLPFC_INRUSH_RELAY_CTRL	GPIO		64	GPIO-15	GPIO-15	GPIO-15
ACTIVE_EMI_INJ_PU_PULSE	OUTPUTXBAR/ CLBXBAR		67	GPIO16 → OUTPUTXBAR7	GPIO16 → OUTPUTXBAR7	GPIO16 → OUTPUTXBAR7
ACTIVE_EMI_INJ_PD_PULSE	OUTPUTXBAR/ CLBXBAR		69	GPIO17 → OUTPUTXBAR8	GPIO17 → OUTPUTXBAR8	GPIO17 → OUTPUTXBAR8
ACTIVE_EMI_RESET	OUTPUTXBAR/ CLBXBAR		87	GPIO33 → CLB_OUTPUTXBAR2	GPIO33 → CLB_OUTPUTXBAR2	GPIO33 → CLB_OUTPUTXBAR2

ACTIVE_EMI_ADC_SOC	OUTPUTXBAR/ CLBXBAR		122	GPIO36 → CLB_OUTPUTXBAR5 → Looped back into INPUTXBAR5	**null**	null
ERRORSTSn	GPIO		102	NA	GPIO55	GPIO55
CANTX/CANRX	CAN		68,70	GPIO20/21	**NO CAN function GPIO20/21**	
SYSTEM_WATCHDOG_OUT SYSTEM_WATCHDOG_DISABLE SYSTEM_PMIC_SPI (resv) SYSTEM_PMIC_SPI (resv)	GPIO/SPI		75, 77, 79, 81	GPIO24 GPIO25 GPIO26 GPIO27	GPIO24 GPIO25(Resistor option) GPIO26(Resistor option) GPIO27(Resistor option)	GPIO24 GPIO25(Resistor option) GPIO26(Resistor option) GPIO27(Resistor option)
SYSTEM_DISABLE_FET_SUPPLY	GPIO		85	GPIO32	GPIO32	GPIO32

SYSTEM_TEMP_MUX_OUT1 SYSTEM_TEMP_MUX_OUT2	ECAP		91 96	GPIO41 -> **ECAP4** → **INPUTXBAR3** GPIO99 -> **ECAP5** → **INPUTXBAR4**	GPIO41 -> **ECAP2** → INPUTXBAR3 **GPIO60** -> **ECAP3** → INPUTXBAR4	GPIO41 -> **ECAP4** → INPUTXBAR3 **GPIO99 -> ECAP5** → INPUTXBAR4
SYSTEM_TEMP_MUX_SEL_1-3	GPIO		93, 94, 95	GPIO96, GPIO97, GPIO98	**GPIO47** **GPIO58** **GPIO59**	**GPIO96** **GPIO97** **GPIO98**

<table>
<tr>
<td>SYSTEM_PROFILING1
SYSTEM_PROFILING2
SYSTEM_PROFILING3</td>
<td>ECAP</td>
<td></td>
<td>89
92
127/101</td>
<td>(In the combined project we will have to choose whether to profile CLLLC code or TTPLPFC)
<table>
<tr><th></th><th colspan="2">CLLLC</th><th colspan="2">TTPLPFC</th></tr>
<tr><td>GPIO 40</td><td></td><td></td><td>INPUT XBAR 15</td><td>ECAP 2</td></tr>
<tr><td>GPIO 45</td><td>INPUTXBAR 15</td><td>ECAP 2</td><td>INPUT XBAR 16</td><td>ECAP 3</td></tr>
<tr><td>GPIO 62</td><td>INPUTXBAR 16</td><td>ECAP 3</td><td></td><td></td></tr>
</table>
</td>
<td>GPIO40 (HSEC89)
GPIO44 (HSEC92)
GPIO49 (HSEC101)</td>
<td>GPIO40 (HSEC89)
GPIO45 (HSEC92)
GPIO49 (HSEC101)</td>
</tr>
</table>

CLLLC_GAN_OC_PRIM_LEG1_H CLLLC_GAN_OC_PRIM_LEG1_L CLLLC_GAN_OC_PRIM_LEG2_H CLLLC_GAN_OC_PRIM_LEG2_L CLLLC_GAN_OC_SEC_LEG1_H CLLLC_GAN_OC_SEC_LEG1_L CLLLC_GAN_OC_SEC_LEG2_H CLLLC_GAN_OC_SEC_LEG2_L	GPIO Inputs		99 107 109 100 104 106 108 110	GPIO48 → CLBINPUTXBAR1 GPIO52 → CLBINPUTXBAR2 GPIO53 → CLBINPUTXBAR3 GPIO54 → CLBINPUTXBAR4 GPIO56 → CLBINPUTXBAR5 -Option 1 for OC pin GPIO57 → CLBINPUTXBAR6 -Option3 for OC pin GPIO58 → CLBINPUTXBAR7 -Option 1 for OC pin GPIO59 → CLBINPUTXBAR8 -Option3 for OC pin)		

TTPLPFC_GAN_OC_HIGH_FREQ_PH1_H TTPLPFC_GAN_OC_HIGH_FREQ_PH1_L TTPLPFC_GAN_OC_HIGH_FREQ_PH2_H TTPLPFC_GAN_OC_HIGH_FREQ_PH2_L **Not implemented on either device**	GPIO		88 90 71 73	GPIO39 GPIO125 GPIO18 GPIO19	**GPIO37** (Resistor option-TDI) **GPIO35** (Resistor option-TDO) GPIO18 (**Switch** option, xtal or GPIO) GPIO19 (**Switch** option, xtal or GPIO)	GPIO39 GPIO125 GPIO18 GPIO35
CLLLC_GANFAULTn_GPIO	GPIO		74	GPIO23	GPIO23	GPIO23
FSI_TX_D0 FSI_TX_D1 FSI_TX_CLK	FSI	For ETAS tool	101 103 105	GPIO-49/FSITXA_D0 GPIO-50/FSITXA_D1 GPIO-51/FSITXA_CLK	GPIO-49/FSITXA_D0 GPIO-50/FSITXA_D1 GPIO-51/FSITXA_CLK	

CLLLC_ACTIVE_SYNC_RECT_DEBUG1 CLLLC_ACTIVE_SYNC_RECT_DEBUG2	OUTPUTXBAR/ GPIO		125 126	GPIO-60 → OUTPUTXBAR 3 (CMSS) GPIO-61 → OUTPUTXBAR 4 (CMSS)	Null	Null
LED1 LED2	GPIO	LED Toggle - General Purpose LED - SFRA GUI	82 86	GPIO-31 → LED1 GPIO-34 → LED2 (SFRA)	GPIO-31 → LED1 GPIO-34 → LED2 (SFRA)	GPIO-31 → LED1 GPIO-34 → LED2 (SFRA)

4 GaN Compatibility

Functional Pin	Test Pin	F2838x			F28003x
GPIO0		-	CLB1		CLB_OUTPUTXBAR8
GPIO1		-			CLB_OUTPUTXBAR7
GPIO2		OUTPUTXBAR1	CLB2		OUTPUTXBAR1
GPIO3		OUTPUTXBAR2			OUTPUTXBAR2
GPIO4	None(output same as GPIO24)	OUTPUTXBAR3 (XBAR conflict, need CLB 3 override output on PWM3A)	CLB3	CLB5	CLB_OUTPUTXBAR6/ OUTPUTXBAR3
GPIO5	GPIO11 (63)	OUTPUTXBAR3			CLB_OUTPUTXBAR5/ OUTPUTXBAR3

Functional Pin	Test Pin	F2838x			F28003x
GPIO6	GPIO24(75)	OUTPUTXBAR4	CLB4		CLB_OUTPUTXBAR8/ OUTPUTXBAR4
GPIO7	GPIO31(82)	OUTPUTXBAR5			CLB_OUTPUTXBAR2/ OUTPUTXBAR5

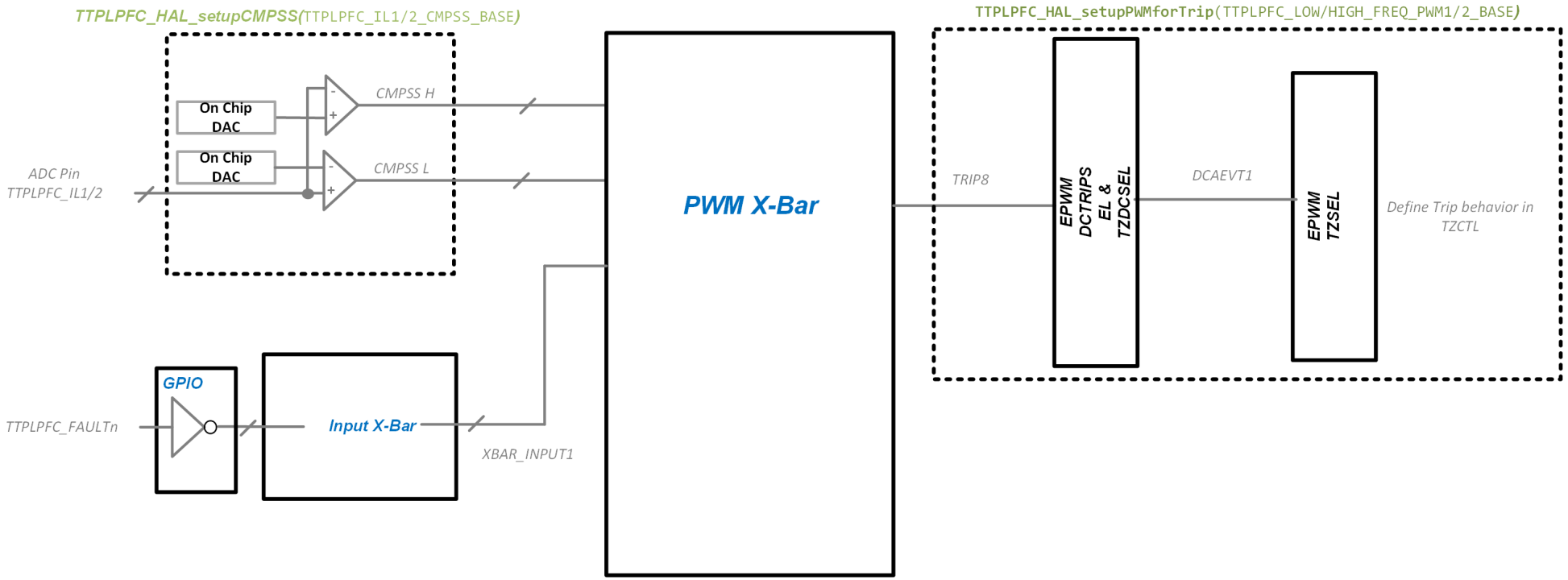
CLB 6, 7 and 8 for Active EMI

ACTIVE_EMI_INJ_PU_PULSE	OUTPUTXBAR/ CLBXBAR		67	GPIO16 → OUTPUTXBAR7, CLB 8 OUTPUT 5
ACTIVE_EMI_INJ_PD_PULSE	OUTPUTXBAR/ CLBXBAR		69	GPIO17 → OUTPUTXBAR8, CLB 8 OUTPUT 4
ACTIVE_EMI_RESET	OUTPUTXBAR/ CLBXBAR		87	GPIO33 → CLB_OUTPUTXBAR2, CLB6 OUTPUT 4

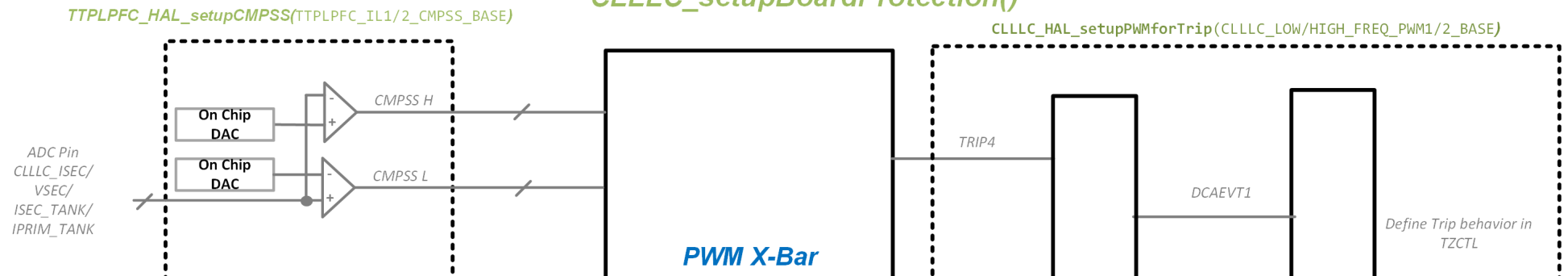
ACTIVE_EMI_ADC_SOC	OUTPUTXBAR/ CLBXBAR		122	GPIO36 → CLB_OUTPUTXBAR5 → Looped back into INPUTXBAR , CLB6 OUTPUT 4

5 Protection Scheme

TTPLPFC_setupBoardProtection()



CLLLC_setupBoardProtection()



TTPLPFC_FAULTn comes to INPUR XBAR 1

EPWM_TRIP8 is used to trip TTPLPFC

CLLLC_FAULTn comes to INPUT XBAR 2

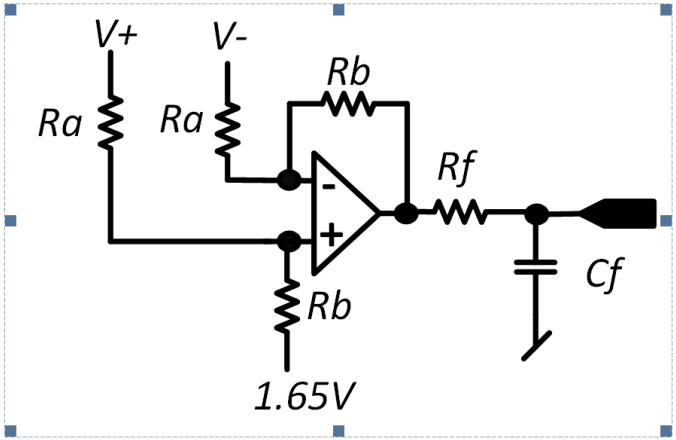
EPWM_TRIP4/5/7 are used by CLLLC, where TRIP4 is for over current trip/ fault trip etc and TRIP5/7 lines are used for active synchronous rectification

6 Sensing Scheme

(Also see GaN OBC SensingGain Worksheet.xlsx)

6.1 TPLPFC_VAC

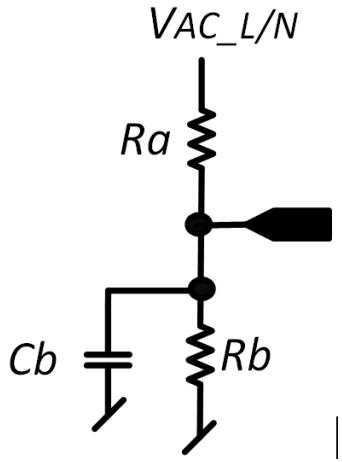
Sampling Rate: Fast (need to sample every cycle due to ZCD management)



TTPLPFC_VAC			
Ra	1095000	Ohms	
Rb	3480	Ohms	
Resitor Divider Gain	0.003178082		
Offset	1.65	Volts	
TTPLPFC_VAC_MAX_SENSE	519.1810345	Volts	
Rf	19.6	Ohms	
Cf	100	nF	
Fc	81242.68816	Hz	

6.1.1 TTPLPFC_VAC_L/N

(redundant sensing for Vac, future proofing for safety goals)



TTPLPFC_VAC_L/N			
Ra	1095000	Ohms	

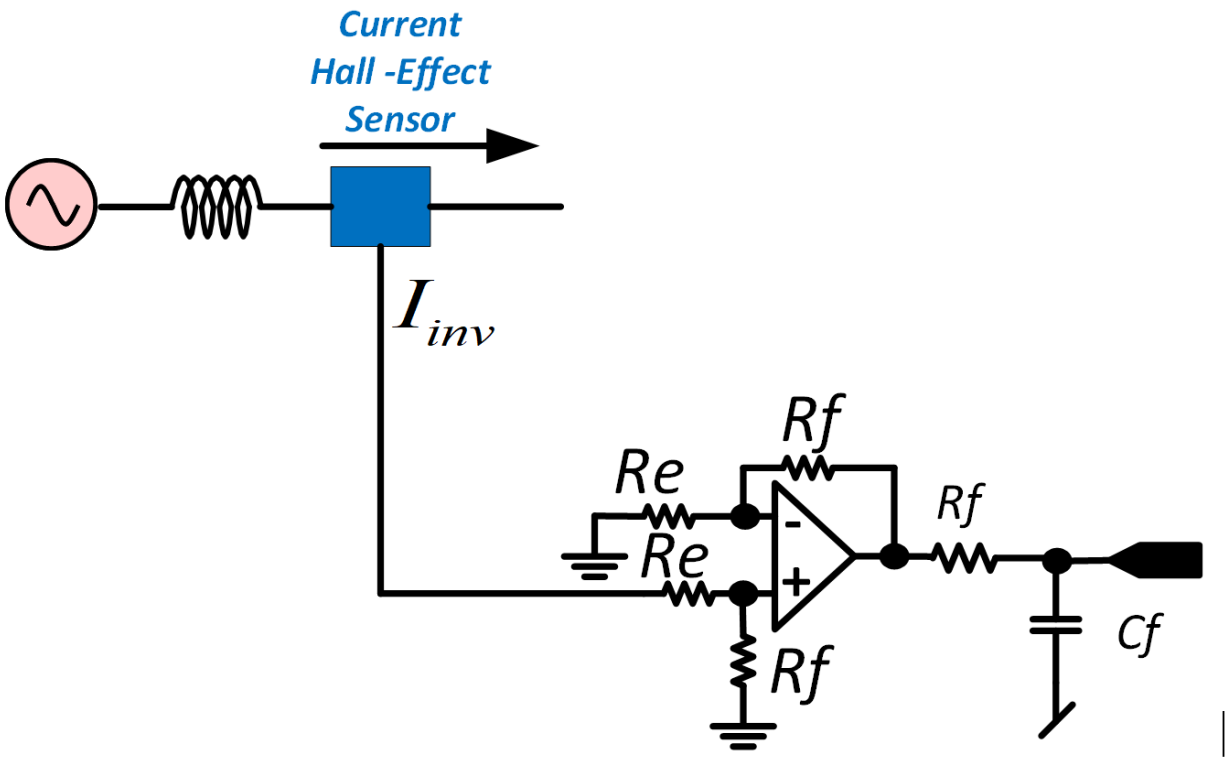
Rb	7150	Ohms	
Resitor Divider Gain	0.00648732		
Opamp Gain	1		
TTPLPFC_VAC_L_N_MAX_SENSE	508.6846154	Volts	
C	0.45	nF	
Fc	49490.49535	Hz	

6.2 TTPLPFC_IAC_PH1/PH2

Using Allegro ACS733KLATR-40AB-T

(Supply Allegro device with 3.3V rail (33mA is the consumption of the device),

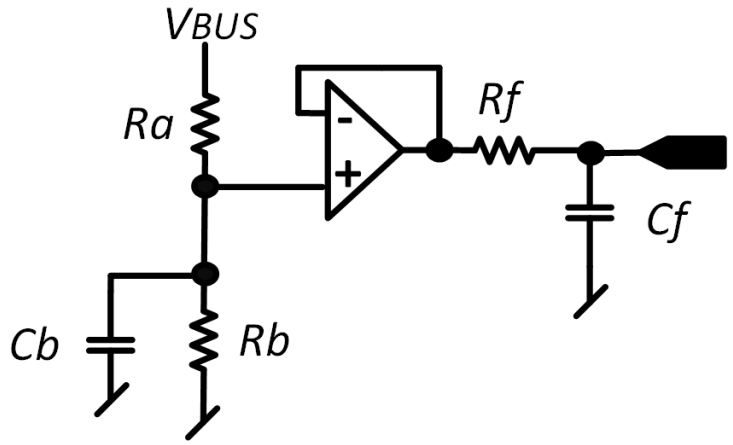
Current going into the inductor from the AC side is positive


Current
Hall -Effect
Sensor
I_{inv}
Rf
Re
Rf
Re
Rf
Cf

TTPLPFC_IAC_PH1/2, ACS733KLATR-40AB-T			
Optimized Range	**40**	**Amps**	
Sensivity	33	mV/A	
Full Range	50	Irms	
Re	10000	Ohms	
Rf	16500	Ohms	
TTPLPFC_IAC_MAX_SENSE	30.3030303	Amps	
Rfltr	20	Ohms	
Cfltr	100	nF	

Cut off	79617.83439	Hz	

6.3 TTPLPFC_VBUS



TTPLPFC_VBUS			
Ra	1095000	Ohms	
Rb	7150	Ohms	
Resitor Divider Gain	0.00648732		
TTPLPFC_VBUS_MAX_SENSE	508.6846154	**Volts**	
ADC pre filter			
Rf	10	Ohms	
Cf	100	nF	
Fc	159235.6688	Hz	
Input Aliasing Filter			

C	470	pF	
Aliasing cutt off	47384.51682	Hz	

7 ADC Sampling scheme on F2838x

	ADC-A	**ADC-B**	**ADC-C**	**ADC-D**
Highest Priority Signals **(120kHz/** **Oversampled 240kHz (2x for PFC current)** **Oversampled 960kHz (8x for CLLLC)**	TTPLPFC_IAC_PH1 (A2, **CMPSS1**) **SOC0 → ADC_TRIGGER_EPWM6_SOCA,** **SOC1 → ADC_TRIGGER_EPWM6_SOCB**	TTPLPFC_IAC_PH2 (B2, **CMPSS3**) **SOC0 → ADC_TRIGGER_EPWM6_SOCA,** **SOC1 → ADC_TRIGGER_EPWM6_SOCB,**	TTPLPFC_VAC (C5) **SOC0 → ADC_TRIGGER_EPWM6_SOCA,** **SOC1 → ADC_TRIGGER_EPWM6_SOCB,**	CLLLC_IPRIM (D2, **CMPSS8**) **SOC0**

CLLLC_ISEC (A4, **CMPSS2**) **ISEC1 → SOC2 → ADC_TRIGGER_EPWM6_SOCA** **ISEC2 → SOC3 → ADC_TRIGGER_EPWM6_SOCA** **ISEC3 → SOC4 → ADC_TRIGGER_EPWM6_SOCA** **ISEC4 → SOC5 → ADC_TRIGGER_EPWM6_SOCA** **ISEC5 → SOC6 → ADC_TRIGGER_EPWM6_SOCA** **ISEC6 → SOC7 → ADC_TRIGGER_EPWM6_SOCA** **ISEC7 → SOC8 → ADC_TRIGGER_EPWM6_SOCB** **ISEC8 → SOC9 → ADC_TRIGGER_EPWM6_SOCB**	TTPLPFC_VBUS / CLLLC_VBUS (B3) **VBUS1 → SOC2 → ADC_TRIGGER_EPWM6_SOCA,** **VBUS2 → SOC3 → ADC_TRIGGER_EPWM6_SOCB,** **VBUS3 → SOC4 → ADC_TRIGGER_EPWM6_SOCA,** **VBUS4 → SOC5 → ADC_TRIGGER_EPWM6_SOCA** **VBUS5 → SOC6 → ADC_TRIGGER_EPWM6_SOCA** **VBUS6 → SOC7 → ADC_TRIGGER_EPWM6_SOCA** **VBUS7 → SOC8 → ADC_TRIGGER_EPWM6_SOCA** **VBUS8 → SOC9 → ADC_TRIGGER_EPWM6_SOCB** **VBUS9 → SOC10 → ADC_TRIGGER_EPWM6_SOCB**	CLLLC_VSEC (C2, **CMPSS6**) **VSEC1 → SOC2 → ADC_TRIGGER_EPWM6_SOCA** **VSEC2 → SOC3 → ADC_TRIGGER_EPWM6_SOCA** **VSEC3 → SOC4 → ADC_TRIGGER_EPWM6_SOCA** **VSEC4 → SOC5 → ADC_TRIGGER_EPWM6_SOCA** **VSEC5 → SOC6 → ADC_TRIGGER_EPWM6_SOCA** **VSEC6 → SOC7 → ADC_TRIGGER_EPWM6_SOCA** **VSEC7 → SOC8 → ADC_TRIGGER_EPWM6_SOCA** **VSEC8 → SOC9 → ADC_TRIGGER_EPWM6_SOCA** **VSEC9 → SOC10 → ADC_TRIGGER_EPWM6_SOCA**	ACTIVE_EMI_SENSE (D3) **SOC1**

	ISEC9 → SOC10 → ADC_TRIGGER_EPWM6_SOCB **ISEC10 → SOC11 → ADC_TRIGGER_EPWM6_SOCB** **ISEC11 → SOC12 → ADC_TRIGGER_EPWM6_SOCB** **ISEC12 → SOC13 → ADC_TRIGGER_EPWM6_SOCB**	**VBUS10 → SOC11 → ADC_TRIGGER_EPWM6_SOCB** **VBUS11 → SOC12 → ADC_TRIGGER_EPWM6_SOCB** **VBUS12 → SOC13 → ADC_TRIGGER_EPWM6_SOCB**	**VSEC10 → SOC11 → ADC_TRIGGER_EPWM6_SOCA** **VSEC11 → SOC12 → ADC_TRIGGER_EPWM6_SOCA** **VSEC12 → SOC13 → ADC_TRIGGER_EPWM6_SOCA**	
Medium Frequency Sampling Signals **(70kHz)**				
Low-Frequency Sampling Signals **(10kHz)**	TTPLPFC_VAC_L (A3) **SOC14 →ADC_TRIGGER_CPU1_TINT2**	TTPLPFC_VAC_N (B1, DACOUTC) **SOC10 → ADC_TRIGGER_CPU1_TINT2**	TTPLPFC_VBUS2 (ADCIN14, **CMPSS4**) **SOC14 → ADC_TRIGGER_CPU1_TINT2**	

	SYSTEM_ TEMP_1 (ADCIN15) **SOC15 → ADC_TRIGGER_CPU1_TINT2**	SYSTEM_VREF_1_65 (B5) **SOC11 →ADC_TRIGGER_CPU1_TINT2**	CLLLC_VSEC (C2, **CMPSS6**) **VSEC13 → SOC15 →ADC_TRIGGER_CPU1_TINT2** ***Triggers ISR3 in combined project***	
Not Sampled	ACTIVE_EMI_INJ (DAC) (A0, DACOUTA)	CLLLC_IPRIM_TANK (CMPSS must, can share with ISEC) C4, **CMPSS5**		CLLLC_ISEC_TANK (CMPSS must, can share with ISEC) D0, **CMPSS7**

F28003x Sampling scheme(differences highlighted)

	ADC-A	**ADC-B**	**ADC-C**	~~**ADC-D**~~

Highest Priority Signals **(120kHz/** **Oversampled 240kHz (2x for PFC current)** **Oversampled 960kHz (8x for CLLLC)**	TTPLPFC_IAC_PH1 (A2, CMPSS1) SOC0 → ADC_TRIGGER_EPWM6_SOCA, SOC1 → ADC_TRIGGER_EPWM6_SOCB	TTPLPFC_IAC_PH2 (B12, CMPSS3) SOC0 → ADC_TRIGGER_EPWM6_SOCA, SOC1 → ADC_TRIGGER_EPWM6_SOCB,	TTPLPFC_VAC (C7) SOC0 → ADC_TRIGGER_EPWM6_SOCA, SOC1 → ADC_TRIGGER_EPWM6_SOCB,	

CLLLC_ISEC (A5, CMPSS2) ISEC1 → SOC2 → ADC_TRIGGER_EPWM6_SOCA ISEC2 → SOC3 → ADC_TRIGGER_EPWM6_SOCA ISEC3 → SOC4 → ADC_TRIGGER_EPWM6_SOCA ISEC4 → SOC5 → ADC_TRIGGER_EPWM6_SOCA ISEC5 → SOC6 → ADC_TRIGGER_EPWM6_SOCA ISEC6 → SOC7 → ADC_TRIGGER_EPWM6_SOCA ISEC7 → SOC8 → ADC_TRIGGER_EPWM6_SOCB ISEC8 → SOC9 → ADC_TRIGGER_EPWM6_SOCB	TTPLPFC_VBUS / CLLLC_VBUS (B4) VBUS1 → SOC2 → ADC_TRIGGER_EPWM6_SOCA, VBUS2 → SOC3 → ADC_TRIGGER_EPWM6_SOCB, VBUS3 → SOC4 → ADC_TRIGGER_EPWM7_SOCA, VBUS4 → SOC5 → ADC_TRIGGER_EPWM7_SOCB ~~VBUS5 → SOC6 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VBUS6 → SOC7 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VBUS7 → SOC8 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VBUS8 → SOC9 → ADC_TRIGGER_EPWM6_SOCB~~ ~~VBUS9 → SOC10 → ADC_TRIGGER_EPWM6_SOCB~~	CLLLC_VSEC (C11, CMPSS2) VSEC1 → SOC2 → ADC_TRIGGER_EPWM6_SOCA VSEC2 → SOC3 → ADC_TRIGGER_EPWM6_SOCA VSEC3 → SOC4 → ADC_TRIGGER_EPWM6_SOCA VSEC4 → SOC5 → ADC_TRIGGER_EPWM6_SOCA VSEC5 → SOC6 → ADC_TRIGGER_EPWM6_SOCA VSEC6 → SOC7 → ADC_TRIGGER_EPWM6_SOCA VSEC7 → SOC8 → ADC_TRIGGER_EPWM6_SOCA VSEC8 → SOC9 → ADC_TRIGGER_EPWM6_SOCA VSEC9 → SOC10 → ADC_TRIGGER_EPWM6_SOCA	

	ISEC9 → SOC10 → ADC_TRIGGER_EPWM6_SOCB ISEC10 → SOC11 → ADC_TRIGGER_EPWM6_SOCB ISEC11 → SOC12 → ADC_TRIGGER_EPWM6_SOCB ~~ISEC12 → SOC13 → ADC_TRIGGER_EPWM6_SOCB~~ Updated to 11x oversampling to allow for CLLLC_IPRIM sampling	~~VBUS10 → SOC11 → ADC_TRIGGER_EPWM6_SOCB~~ ~~VBUS11 → SOC12 → ADC_TRIGGER_EPWM6_SOCB~~ ~~VBUS12 → SOC13 → ADC_TRIGGER_EPWM6_SOCB~~	VSEC10 → SOC11 → ADC_TRIGGER_EPWM6_SOCA VSEC11 → SOC12 → ADC_TRIGGER_EPWM6_SOCA ~~VSEC12 → SOC13 → ADC_TRIGGER_EPWM6_SOCA~~	
	CLLLC_IPRIM (A9, CMPSS2) SOC13		ACTIVE_EMI_SENSE (A14/B14/**C4**) SOC13 **Not implemented?**	

Medium Frequency Sampling Signals **(70kHz)**				
Low-Frequency Sampling Signals **(10kHz)**	TTPLPFC_VAC_L (A4) SOC14 →ADC_TRIGGER_CPU1_TINT2	TTPLPFC_VAC_N (B2, DACOUTC) SOC10 → ADC_TRIGGER_CPU1_TINT2	TTPLPFC_VBUS2 (C10, CMPSS2) SOC14 → ADC_TRIGGER_CPU1_TINT2	
	SYSTEM_ TEMP_1 (A11) SOC15 → ADC_TRIGGER_CPU1_TINT2	SYSTEM_VREF_1_65 (B5) SOC11 →ADC_TRIGGER_CPU1_TINT2	CLLLC_VSEC (C11, CMPSS2) VSEC13 → SOC15 →ADC_TRIGGER_CPU1_TINT2 *Triggers ISR3 in combined project*	

Not Sampled	ACTIVE_EMI_INJ (DAC) (A0, DACOUTA)	CLLLC_IPRIM_TANK (CMPSS must, can share with ISEC) A7/C3, CMPSS4 (XBAR06/07) or A12/C5, **CMPSS2(possibly jump to hsec38 EMI_sense CMPSS3 on A14)**	CLLLC_ISEC_TANK (CMPSS must, can share with ISEC) C1, CMPSS4	

F28P65x Sampling scheme(differences highlighted)

	ADC-A	**ADC-B**	**ADC-C**	~~**ADC-D**~~

Highest Priority Signals **(120kHz/** **Oversampled 240kHz (2x for PFC current)** **Oversampled 960kHz (8x for CLLLC)**	TTPLPFC_IAC_PH1 (A2, CMPSS1) SOC0 → ADC_TRIGGER_EPWM6_SOCA, SOC1 → ADC_TRIGGER_EPWM6_SOCB	TTPLPFC_IAC_PH2 (B2, CMPSS3) SOC0 → ADC_TRIGGER_EPWM6_SOCA, SOC1 → ADC_TRIGGER_EPWM6_SOCB,	TTPLPFC_VAC (C11) SOC0 → ADC_TRIGGER_EPWM6_SOCA, 8x HW oversample, ADC_REPINST2, PPB2	

CLLLC_ISEC (A4, CMPSS2) ISEC1 → SOC2 → ADC_TRIGGER_EPWM6_SOCA ADC_REPINST1, PPB1 Updated to 8x HW oversampling ~~ISEC2 → SOC3 → ADC_TRIGGER_EPWM6_SOCA~~ ~~ISEC3 → SOC4 → ADC_TRIGGER_EPWM6_SOCA~~ ~~ISEC4 → SOC5 → ADC_TRIGGER_EPWM6_SOCA~~ ~~ISEC5 → SOC6 → ADC_TRIGGER_EPWM6_SOCA~~ ~~ISEC6 → SOC7 → ADC_TRIGGER_EPWM6_SOCA~~ ~~ISEC7 → SOC8 → ADC_TRIGGER_EPWM6_SOCB~~	TTPLPFC_VBUS / CLLLC_VBUS (B3, CMPSS3) VBUS1 → SOC2 → ADC_TRIGGER_EPWM6_SOCA, ADC_REPINST1, PPB1 Updated to 8x HW oversampling ~~VBUS2 → SOC3 → ADC_TRIGGER_EPWM6_SOCB,~~ ~~VBUS3 → SOC4 → ADC_TRIGGER_EPWM7_SOCA,~~ ~~VBUS4 → SOC5 → ADC_TRIGGER_EPWM7_SOCB~~ ~~VBUS5 → SOC6 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VBUS6 → SOC7 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VBUS7 → SOC8 → ADC_TRIGGER_EPWM6_SOCA~~	CLLLC_VSEC (C2, CMPSS6) VSEC1 → SOC2 → ADC_TRIGGER_EPWM6_SOCA ADC_REPINST1, PPB1 Updated to 8x HW oversampling ~~VSEC2 → SOC3 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VSEC3 → SOC4 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VSEC4 → SOC5 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VSEC5 → SOC6 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VSEC6 → SOC7 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VSEC7 → SOC8 → ADC_TRIGGER_EPWM6_SOCA~~	

	~~ISEC8 → SOC9 → ADC_TRIGGER_EPWM6_SOCB~~ ~~ISEC9 → SOC10 → ADC_TRIGGER_EPWM6_SOCB~~ ~~ISEC10 → SOC11 → ADC_TRIGGER_EPWM6_SOCB~~ ~~ISEC11 → SOC12 → ADC_TRIGGER_EPWM6_SOCB~~ ~~ISEC12 → SOC13 → ADC_TRIGGER_EPWM6_SOCB~~	~~VBUS8 → SOC9 → ADC_TRIGGER_EPWM6_SOCB~~ ~~VBUS9 → SOC10 → ADC_TRIGGER_EPWM6_SOCB~~ ~~VBUS10 → SOC11 → ADC_TRIGGER_EPWM6_SOCB~~ ~~VBUS11 → SOC12 → ADC_TRIGGER_EPWM6_SOCB~~ ~~VBUS12 → SOC13 → ADC_TRIGGER_EPWM6_SOCB~~	~~VSEC8 → SOC9 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VSEC9 → SOC10 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VSEC10 → SOC11 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VSEC11 → SOC12 → ADC_TRIGGER_EPWM6_SOCA~~ ~~VSEC12 → SOC13 → ADC_TRIGGER_EPWM6_SOCA~~	
	CLLLC_IPRIM (A8, CMPSS8) SOC13		ACTIVE_EMI_SENSE (A14/B14/**C4**) SOC13 **Not implemented?**	

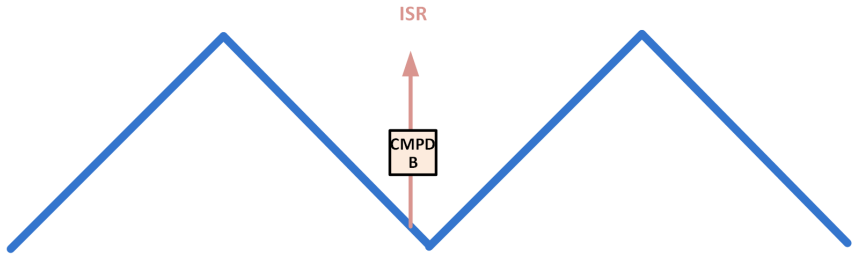
Medium Frequency Sampling Signals **(70kHz)**				
Low-Frequency Sampling Signals **(10kHz)**	TTPLPFC_VAC_L (A3) SOC14 →ADC_TRIGGER_CPU1_TINT2	TTPLPFC_VAC_N (B1, DACOUTC) SOC10 → ADC_TRIGGER_CPU1_TINT2	TTPLPFC_VBUS2 (C14, CMPSS4) SOC14 → ADC_TRIGGER_CPU1_TINT2	
	SYSTEM_ TEMP_1 (A5) SOC15 → ADC_TRIGGER_CPU1_TINT2	SYSTEM_VREF_1_65 (B5) SOC11 →ADC_TRIGGER_CPU1_TINT2	CLLLC_VSEC (C2, CMPSS6) VSEC13 → SOC15 →ADC_TRIGGER_CPU1_TINT2 *Triggers ISR3 in combined project*	

Not Sampled	ACTIVE_EMI_INJ (DAC) (A0, DACOUTA)	CLLLC_IPRIM_TANK C4, CMPSS5 jumped to HSEC30, C4, CMPSS5; Connect a blue wire from HSEC33 (17 th on front) to HSEC30 (15th on back)) AGPIO205. Enable it by setting the corresponding bit to 1	CLLLC_ISEC_TANK C6, CMPSS10	

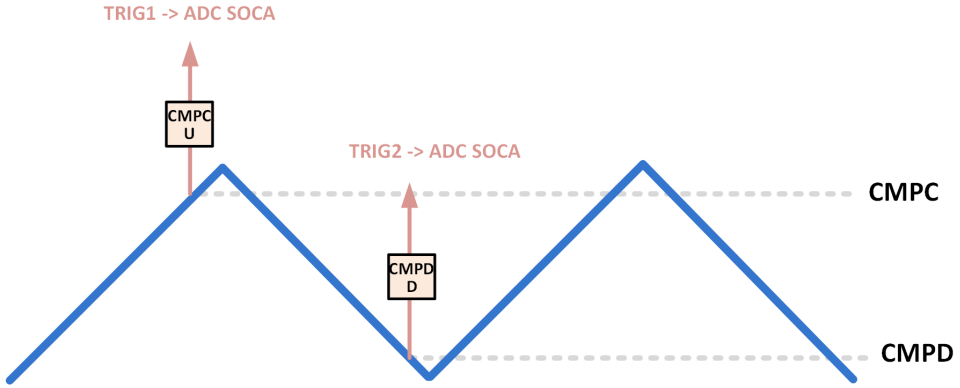
8 ADC Sampling Structure

TTPLPFC_LOW_FREQ_PWM1_BASE

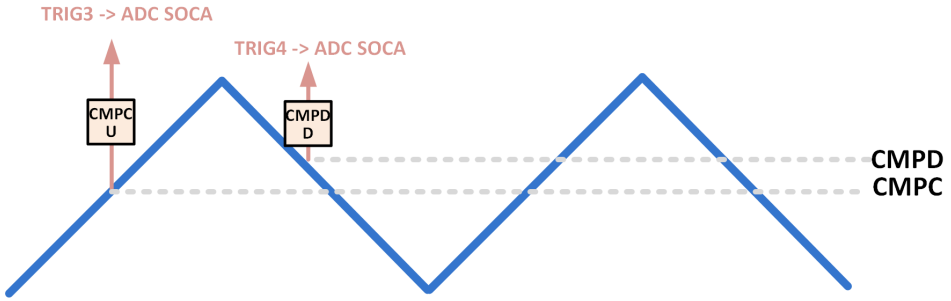
(need to replace by eCAP)



TTPLPFC_HIGH_FREQ_PWM1_BASE



TTPLPFC_HIGH_FREQ_PWM2_BASE



9 ISR Structure

The project framework can have upto three ISRs (ISR1,2 and 3) with ISR1 being the fastest and non-nestable ISR. ISR1 is reserved for the fast update to the PWM in the CLLLC.

	CLLLC	**TTPLPFC**	**Combined project**
ISR1	EPWM1_INT → GROUP3 FAST ISR synchronous to PWM time base of CLLLC	ECAP1INT - > synched with EPWM5 → GROUP4 120kHz *(not yet changed to eCAP)*	EPWM1_INT FAST ISR synchronous to PWM time base of CLLLC
ISR2	ECAP1_INT → GROUP4 EPWM5 → Issues ADC SOC 120kHz	CPU Timer → ADC INT 10kHz	ECAP1_INT → EPWM5_INT → Issues ADC SOC 120kHz
ISR3	CPU Timer → ADC INT → GROUP10 10kHz		CPU Timer → ADC INT → GROUP10 10kHz

On totempole PFC:

the ISR1 i.e. the CONTROL_ISR runs from the PWM trigger.

The following are the defines that are related to this ISR.

#define TTPLPFC_C28x_CONTROL_ISR_PIE_GROUP_NO INTERRUPT_ACK_GROUP3
#define TTPLPFC_C28x_CONTROL_ISR_TRIG_BASE EPWM5_BASE
#define TTPLPFC_C28x_CONTROL_ISR INT_EPWM5

#define TTPLPFC_CLA_CONTROLISR_TRIG CLA_TRIGGER_EPWM5INT

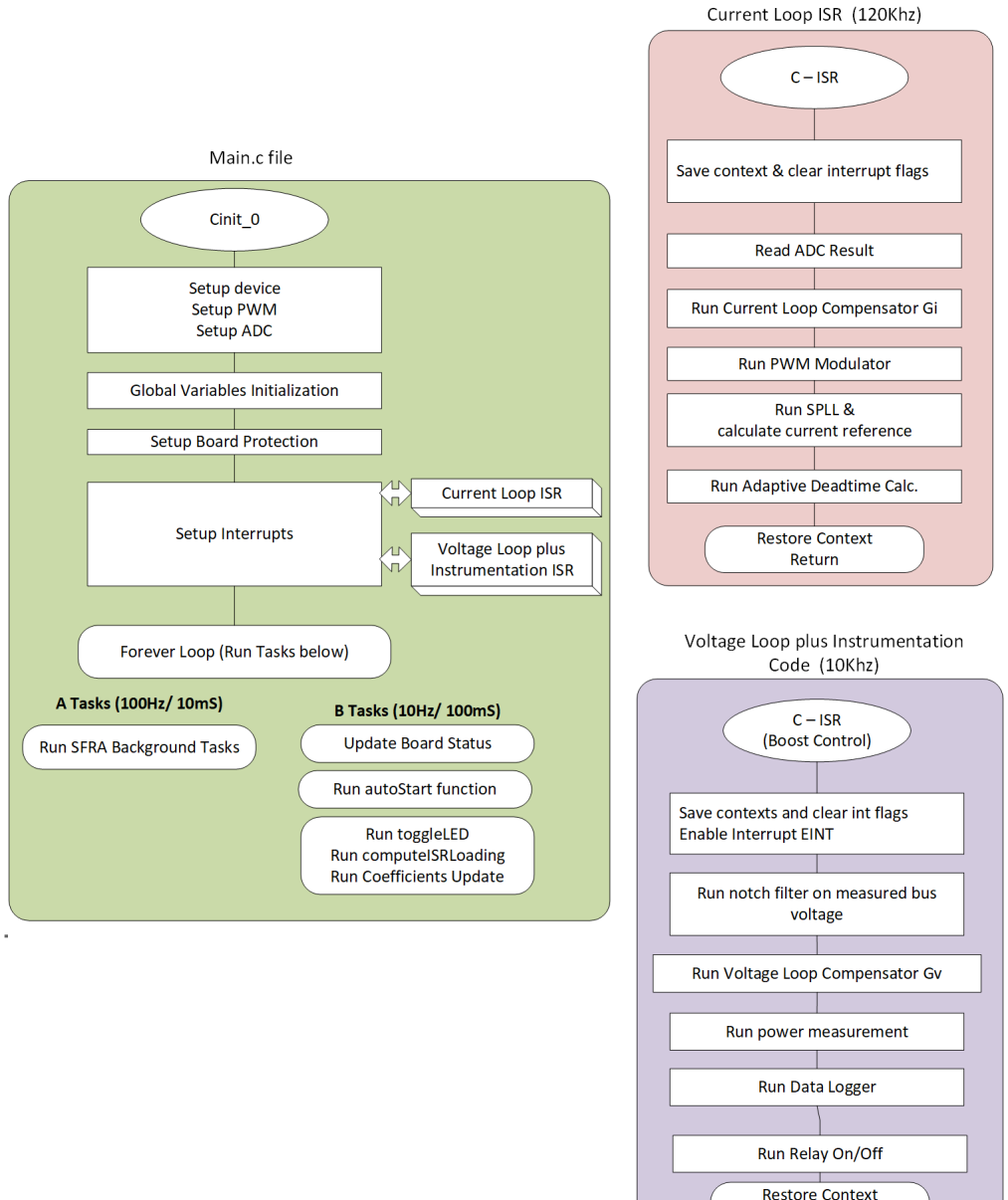
The ISR2 which is the instrumentation ISR

#define TTPLPFC_INSTRUMENTATION_ISR_PIE_GROUP_NO INTERRUPT_ACK_GROUP10
#define TTPLPFC_INSTRUMENTATION_ISR_TRIG_BASE ADCA_BASE
#define TTPLPFC_INSTRUMENTATION_ISR INT_ADCA2

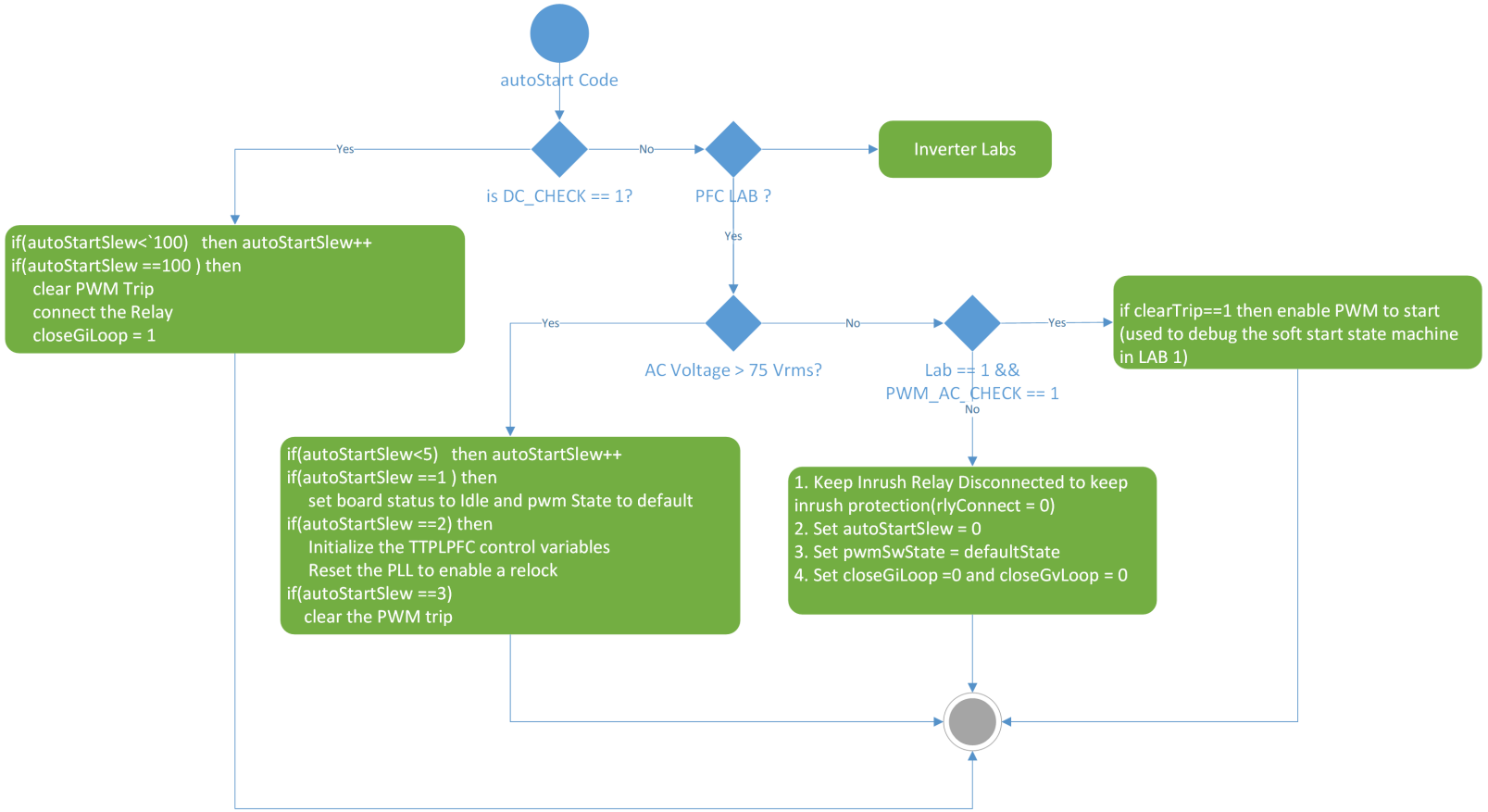
#define TTPLPFC_INSTRUMENTATION_ISR_FREQUENCY 10000
#define TTPLPFC_INSTRUMENTATION_ISR_TIME_BASE CPUTIMER2_BASE

10 PFC

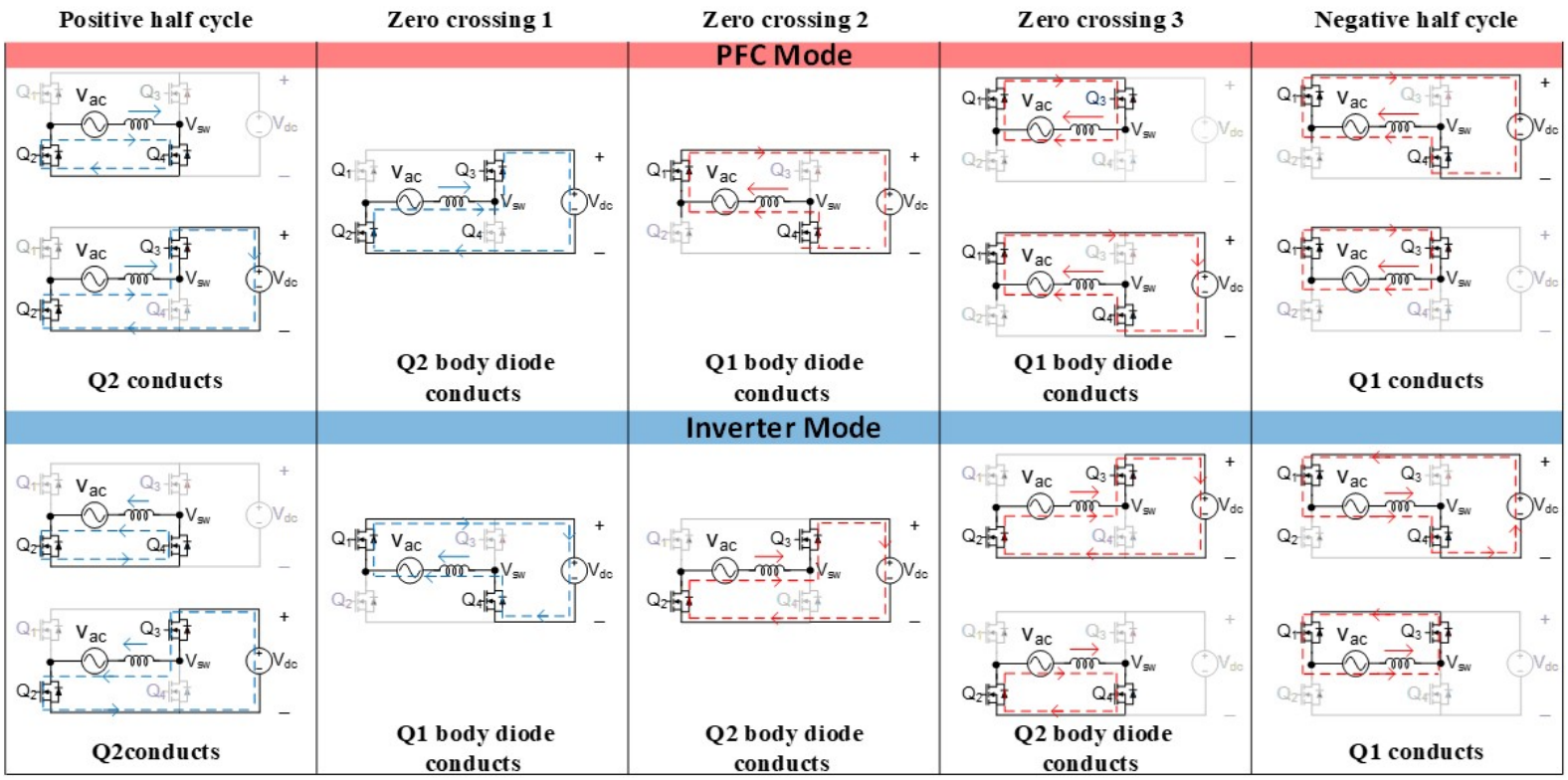
The overall flow of the software is as below

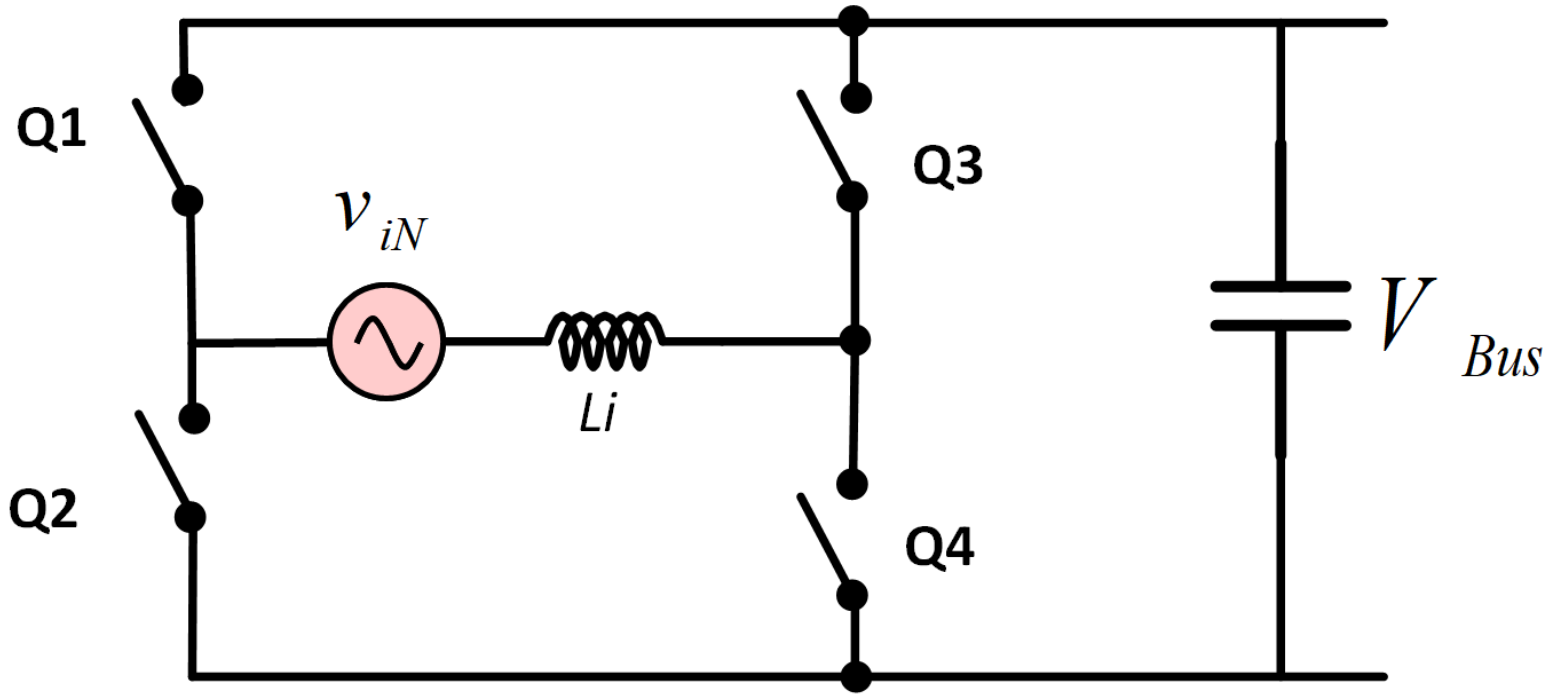

Main.c file
Cinit_0
Setup device
Setup PWM
Setup ADC
Global Variables Initialization
Setup Board Protection
Setup Interrupts
Current Loop ISR
Voltage Loop plus
Instrumentation ISR
Forever Loop (Run Tasks below)
A Tasks (100Hz/ 10mS)
Run SFRA Background Tasks
B Tasks (10Hz/ 100mS)
Update Board Status
Run autoStart function
Run toggleLED
Run computeISRLoading
Run Coefficients Update
Current Loop ISR (120Khz)
C – ISR
Save context & clear interrupt flags
Read ADC Result
Run Current Loop Compensator Gi
Run PWM Modulator
Run SPLL &
calculate current reference
Run Adaptive Deadtime Calc.
Restore Context
Return
Voltage Loop plus Instrumentation
Code (10Khz)
C – ISR
(Boost Control)
Save contexts and clear int flags
Enable Interrupt EINT
Run notch filter on measured bus
voltage
Run Voltage Loop Compensator Gv
Run power measurement
Run Data Logger
Run Relay On/Off
Restore Context

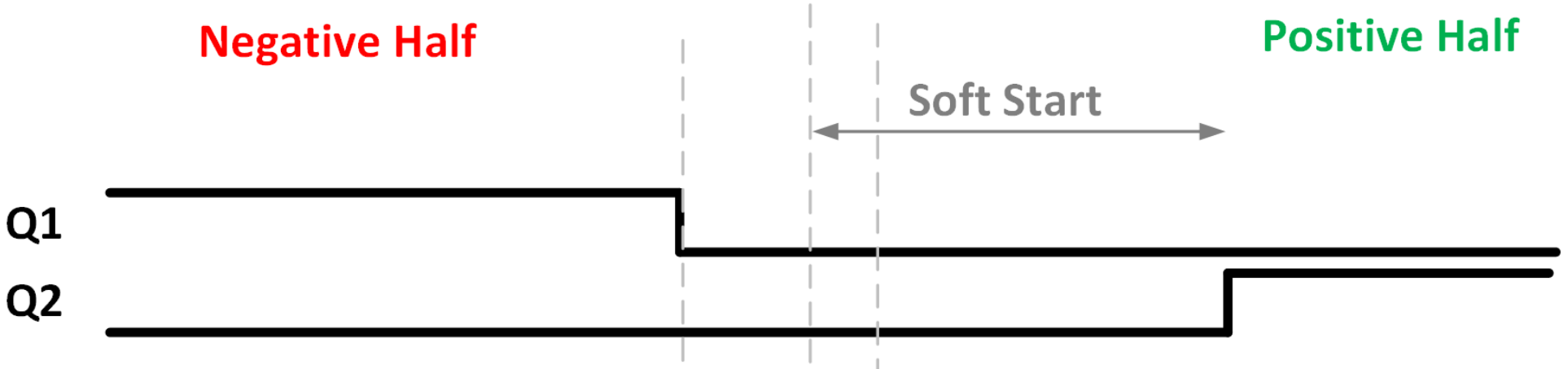
The code is equipped with a state machine than can start PFC on its own when the condition detected are correct. For this the following state machine is run which is called in the slow tasks in the background loop.


autoStart Code
is DC_CHECK == 1?
Yes
No
PFC LAB ?
Inverter Labs
if(autoStartSlew<`100) then autoStartSlew++
if(autoStartSlew ==100) then
clear PWM Trip
connect the Relay
closeGiLoop = 1
Yes
AC Voltage > 75 Vrms?
No
Lab == 1 &&
PWM_AC_CHECK == 1
Yes
if clearTrip==1 then enable PWM to start
(used to debug the soft start state machine
in LAB 1)
No
if(autoStartSlew<5) then autoStartSlew++
if(autoStartSlew ==1) then
set board status to Idle and pwm State to default
if(autoStartSlew ==2) then
Initialize the TTPLPFC control variables
Reset the PLL to enable a relock
if(autoStartSlew ==3)
clear the PWM trip
1. Keep Inrush Relay Disconnected to keep
inrush protection(rlyConnect = 0)
2. Set autoStartSlew = 0
3. Set pwmSwState = defaultState
4. Set closeGiLoop =0 and closeGvLoop = 0

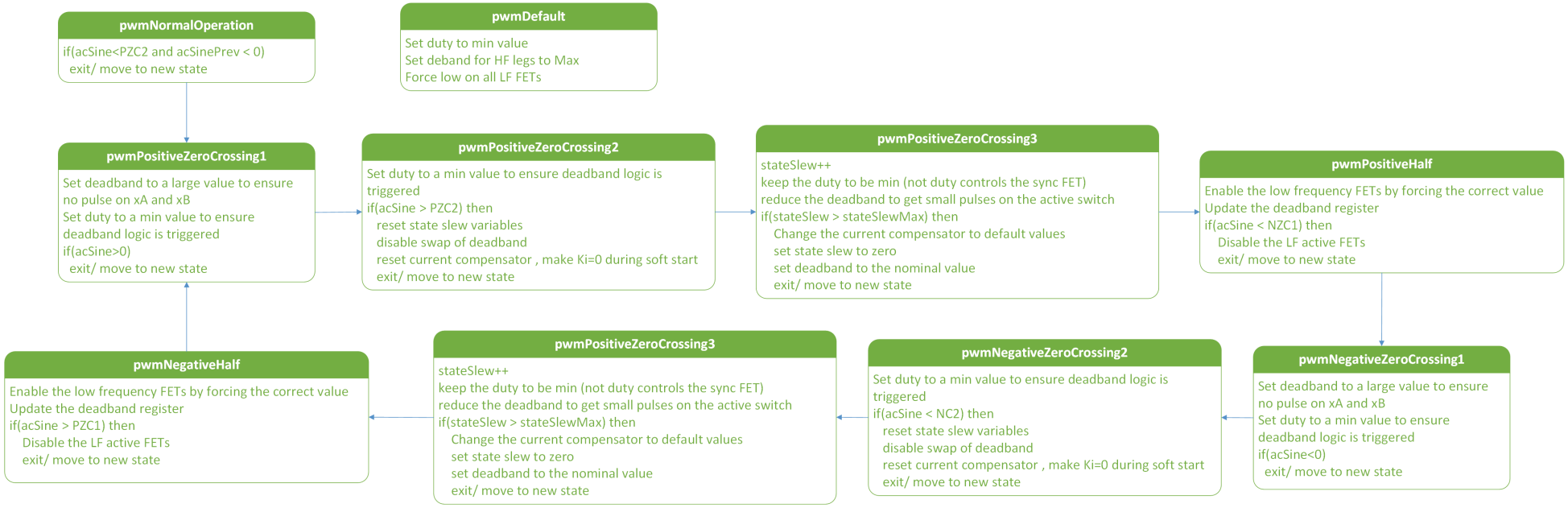
In totempole PFC to avoid the ZC spike a soft start is implemented, this soft start is controlled using a state machine


Positive half cycle
Zero crossing 1
Zero crossing 2
Zero crossing 3
Negative half cycle
PFC Mode
Q2 conducts
Q2 body diode conducts
Q1 body diode conducts
Q1 body diode conducts
Q1 conducts
Inverter Mode
Q2conducts
Q1 body diode conducts
Q2 body diode conducts
Q2 body diode conducts
Q1 conducts


Q1
Q3
v_{iN}
Li
V_{Bus}
Q2
Q4


Negative Half
Positive Half
Soft Start
Q1
Q2

below is the UML diagram of the state machine



10.1 Labs

Lab		What are we checking for	
1	PFC : ISR, PWM & ADC Check	Some checks can be run on a control card (Check 1- 5), with no protection or limited protection. • Check 1 Test ISR Structure • Check 2 Test ADC mapping and reading of conversion data. • Check 3 Test PWM driver DC, • Check 4 Test Board Protection • Check 5 Test PWM under AC The following checks can be run on the power board • Check 6 Open Loop Running with DC on the power board • Check 7 Calibration if any needed • Check 8 AC measurement verification • Check 9 , pulses of PWM for a short duration of time to debug hardware during startup under DC conditions	

Lab		What are we checking for	
2	PFC: Closed current loop check DC input		
3	PFC: Closed current loop check AC input		
4	PFC: Closed voltage loop check AC input		
5	INV: Open Loop Check (DC Output)		(Not scoped for demo project)
6	INV: Open Loop Check (AC output)		(Not scoped for demo project)
7	INV: Closed current loop check (DC Output, res load)		(Not scoped for demo project)
8	INV: Closed Current Loop CHeck (AC output, Res load)		(Not scoped for demo project)
9	INV: Closed Current Loop Check (AC output, Grid Connected)		(Not scoped for demo project)

10.1.1 Lab 1

Set the project to Lab 1 by changing the Lab Number in the <settings.h> file, (this will be changed by powerSUITE GUI when using powerSUITE project)

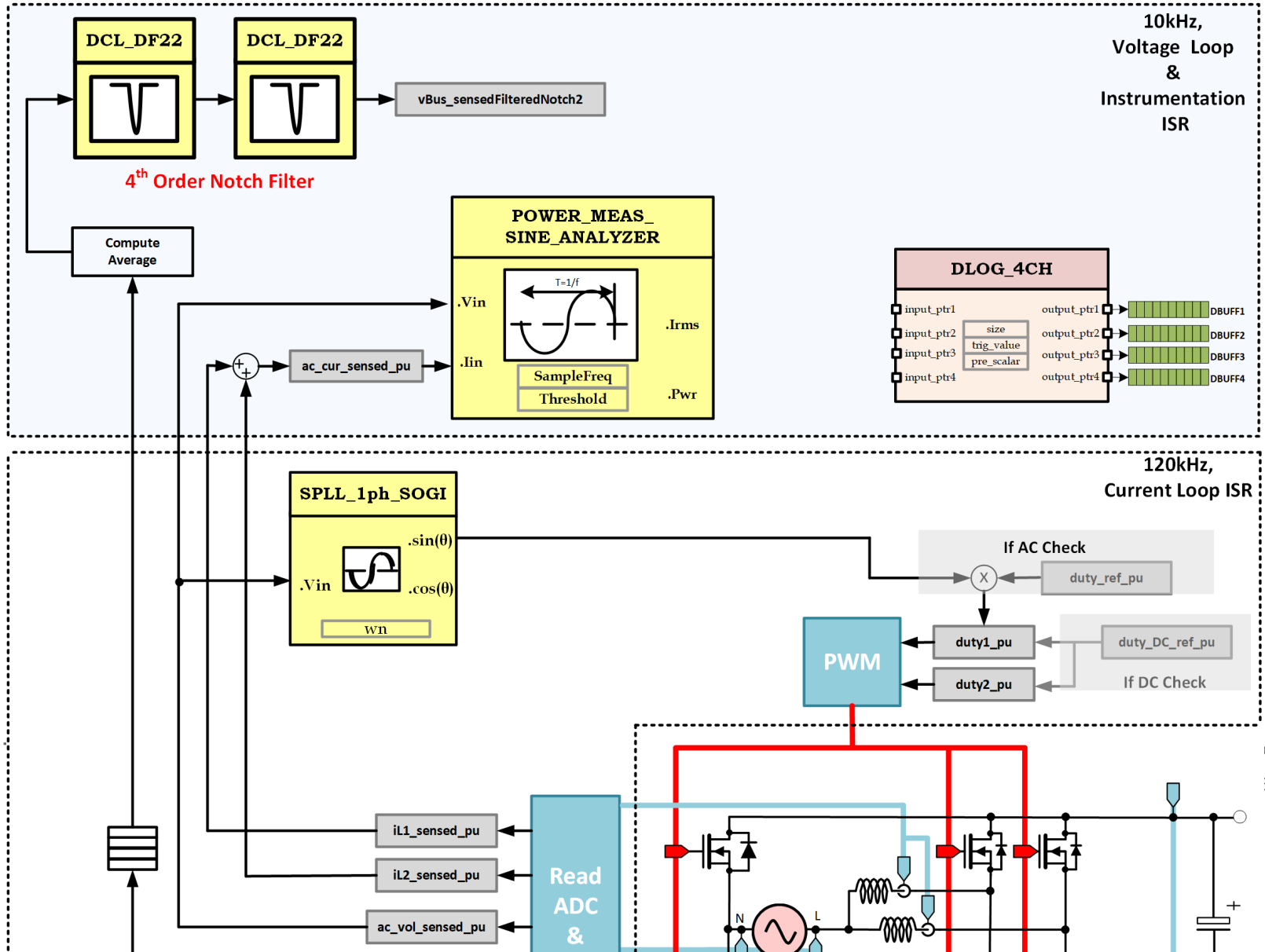
#define TTPLPFC_LAB 1

All the other options can be left at default for now in the user_settings.h file

#if TTPLPFC_LAB == 1
#define TTPLPFC_DC_CHECK 1
#define TTPLPFC_PWM_AC_CHECK 0
#define TTPLPFC_AUTO_START 1
#define TTPLPFC_IL1_PROTECTION_ENABLE 0
#define TTPLPFC_IL2_PROTECTION_ENABLE 0
#define TTPLPFC_FAULTn_PROTECTION_ENABLE 0

#define TTPLPFC_SPLL_METHOD_SELECT TTPLPFC_SPLL_1PH_SOGI_SEL
#define TTPLPFC_CONTROL_RUNNING_ON 1
#define TTPLPFC_CURRENT_LOOP_CLOSED 0
#define TTPLPFC_VOLTAGE_LOOP_CLOSED 0
#define TTPLPFC_ADAPTIVE_DEADTIME 0
#define TTPLPFC_PHASE_SHEDDING_ENABLED 0
#define TTPLPFC_SFRA_TYPE 1
#endif

The SW diagram of the code running is as below


10kHz,
Voltage Loop
&
Instrumentation
ISR
DCL_DF22
DCL_DF22
vBus_sensedFilteredNotch2
4th Order Notch Filter
Compute Average
POWER_MEAS_
SINE_ANALYZER
.Vin
.Iin
T=1/f
.Irms
.Pwr
SampleFreq
Threshold
ac_cur_sensed_pu
DLOG_4CH
input_ptr1
input_ptr2
input_ptr3
input_ptr4
size
trig_value
pre_scalar
output_ptr1
output_ptr2
output_ptr3
output_ptr4
DBUFF1
DBUFF2
DBUFF3
DBUFF4
120kHz,
Current Loop ISR
SPLL_1ph_SOGI
.Vin
.sin(θ)
.cos(θ)
wn
If AC Check
X
duty_ref_pu
PWM
duty1_pu
duty2_pu
duty_DC_ref_pu
If DC Check
iL1_sensed_pu
iL2_sensed_pu
ac_vol_sensed_pu
Read
ADC
&
N
L

10.1.1.1 **Lab 1 - Check 1**

First check whether the ISRs are executing correctly and at the correct rate, for this, we use GPIO toggle. The pins were listed on the table above on this page as SYSTEM_PROFILING1/2. Connect oscilloscope probes to each of these pins.

The project will build without errors, a default target configuration file is already assigned to the project hence simply click on the debug icon in CCS. It may ask you whether you want to load to CPU1, if this is the first time running this project. As in this project, because we only use CPU1, you only need to load to CPU1, you can uncheck CPU2.

Once the code is loaded, hit run.

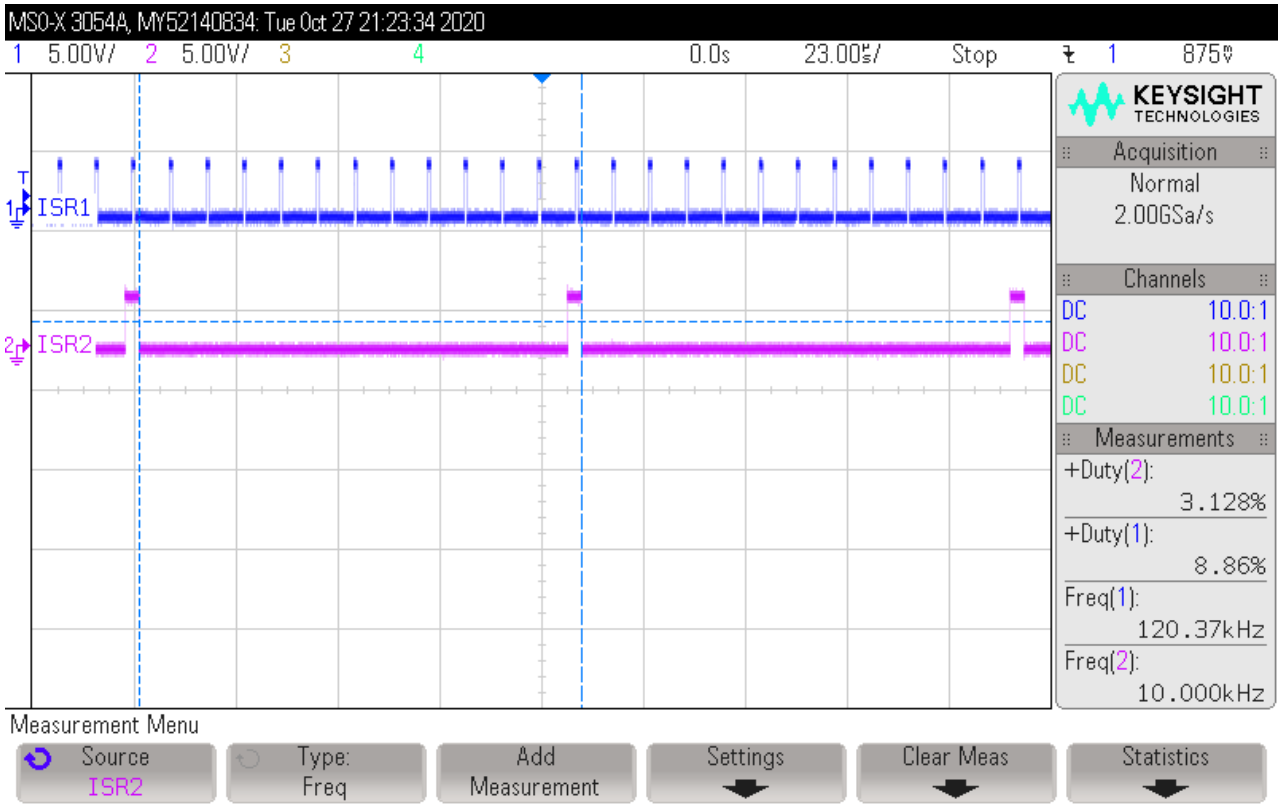
To add the variables in the watch and expressions window, click View → Scripting Console to open the scripting console dialog box. On the upper-right corner of this console, click on open then browse to the setupdebugenv_lab1.js script file located inside the project folder. This script file populates the watch window with appropriate variables required to debug the system. Click on the Continuous Refresh button on the watch window to enable continuous updates of values from the controller. The watch window appears as shown below:

Variables | Expressions | Registers

Expression	Type	Value	Address
TTPLPFC_lab.enum_lab	enum <unnamed>	Lab1	0x00008002@Data
TTPLPFC_vBus_sensed_Volts	float	504.073303	0x0000801E@Data
TTPLPFC_ac_vol_sensed_Volts	float	512.969116	0x00008026@Data
TTPLPFC_iL1_sensed_Amps	float	-30.3029995	0x00008010@Data
TTPLPFC_iL2_sensed_Amps	float	-7.94400215	0x00008012@Data
TTPLPFC_vBus_sensed_pu	float	0.990966797	0x0000801A@Data
TTPLPFC_ac_vol_sensed_pu	float	0.984863281	0x00008024@Data
TTPLPFC_iL1_sensed_pu	float	-1.0	0x00008008@Data
TTPLPFC_iL2_sensed_pu	float	-0.379882813	0x0000800A@Data
TTPLPFC_ac_vol_senseOffset_pu	float	0.5	0x0000802C@Data
TTPLPFC_clearTrip	long	1	0x00008094@Data
EPwm5Regs.TZFLG	Register	0x0000	
EPwm6Regs.TZFLG	Register	0x0000	
EPwm7Regs.TZFLG	Register	0x0000	
TTPLPFC_duty1_DC_ref_pu	float	0.200000003	0x0000806E@Data
TTPLPFC_duty2_DC_ref_pu	float	0.400000006	0x00008070@Data
TTPLPFC_duty_ref_pu	float	0.5	0x00008072@Data
TTPLPFC_system_temp_pu	float	0.0	0x0000802E@Data
TTPLPFC_system_vref_165_pu	float	0.0341796875	0x00008030@Data
TTPLPFC_vbus2_pu	float	0.0	0x00008032@Data
TTPLPFC_autoStartSlew	long	101	0x00008088@Data
TTPLPFC_rlyConnect	long	1	0x00008096@Data
TTPLPFC_ISR1_LoadingMax	float	0.227607369	0x000080DA@Data
TTPLPFC_ISR2_LoadingAvg_accountingForNesting	float	0.024914993	0x000080D6@Data
Add new expression			

If the SW is set up correctly you should see IO toggles on those profiling GPIOs. To **verify/pass** this check following must be completed:

1. Match the rate of the ISR1 and ISR2 by measuring frequency to be what is desired in the application. In this solution, the rate is 120kHz for ISR1 and 10kHz for ISR2.
2. Confirm the nesting behavior matches the expectation i.e. ISR2 nests ISR1. To verify this single step the trigger, a few times. On some of these triggers, an instance when ISR1 is triggered while ISR2 is executing will occur. In this case, the ISR2 will nest ISR1.


MSO-X 3054A, MY52140834: Tue Oct 27 21:23:34 2020
1 5.00V/ 2 5.00V/ 3 4 0.0s 23.00ks/ Stop 1 875
ISR1
ISR2
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
2.00GSa/s
Channels
DC 10.0:1
DC 10.0:1
DC 10.0:1
DC 10.0:1
Measurements
+Duty(2):
3.128%
+Duty(1):
8.86%
Freq(1):
120.37kHz
Freq(2):
10.000kHz
Measurement Menu
Source ISR2
Type: Freq
Add Measurement
Settings
Clear Meas
Statistics

One can also observe the LED 1 toggle periodically as the code is ran.

You can now halt the execution, and disable real-time mode if you had enabled it.

10.1.1.2 **Lab 1 - Check 2**

ADC Reading, verify measurement variables in the watch window to ensure the conversion is happening and the scale range is close to what is expected. Note because the sampling happens as a very fast rate some times the circuitry used to drive the voltage on the pin may not have enough strength and absolute accuracy cannot be checked in this lab. Only the pin mapping, sw configuration, and the rough range.

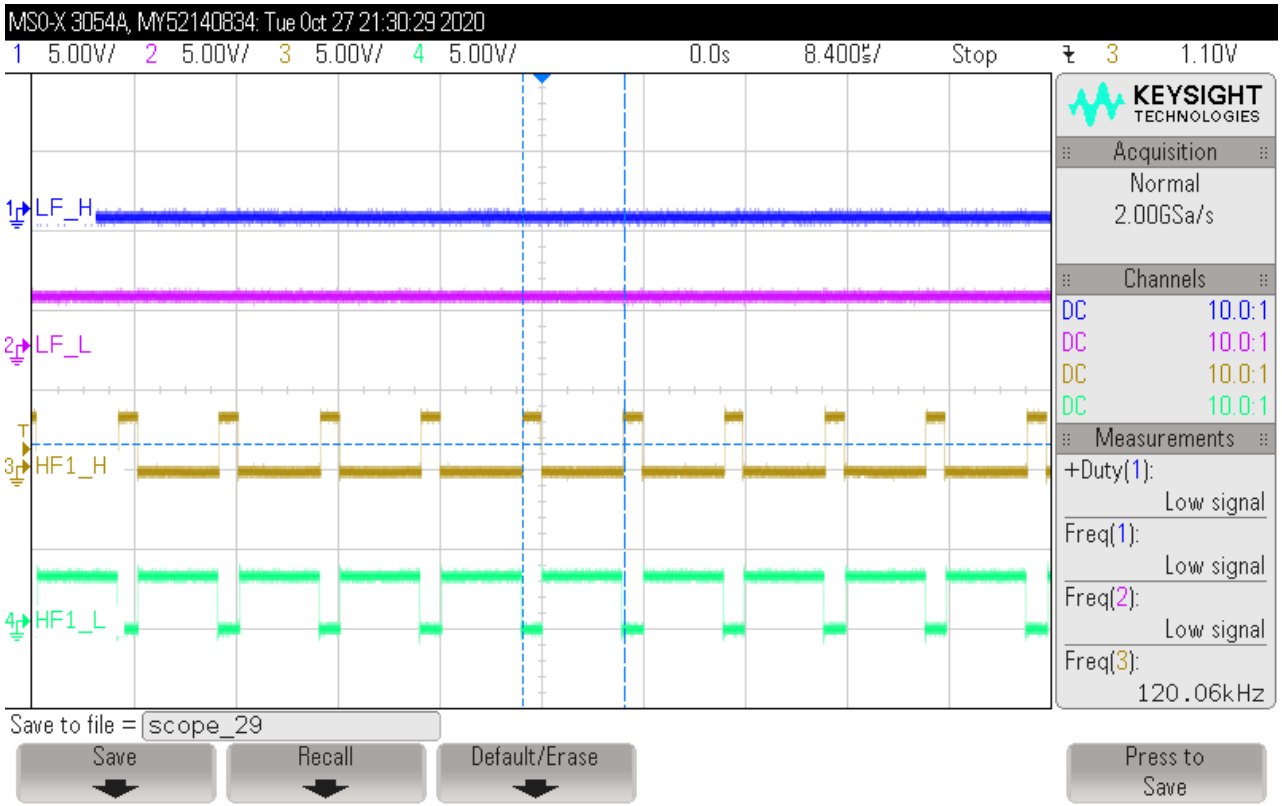
		3.3V	**0V**	**HSEC pin**
ttplpfc_ac_vol_sensed_Volts	TTPLPFC_VAC_MAX_SENSE	519 V	-519 V	39
ttplpfc_vBus_sensed_Volts	TTPLPFC_VBUS_MAX_SENSE	508 V	0 V	20
ttplpfc_iL1_sensed_Amps ttplpfc_iL2_sensed_Amps	TTPLPFC_IL_MAX_SENSE	30 Amps	-30 Amps	15 18 Pins inverted
ttplpfc_system_temp_pu		0.99	0	27

		3.3V	0V	HSEC pin
ttplpfc_system_vref_165_pu		0.999	0	26
ttplpfc_vbus2_pu		0.999	0	25

10.1.1.3 Lab 1 - Check 3

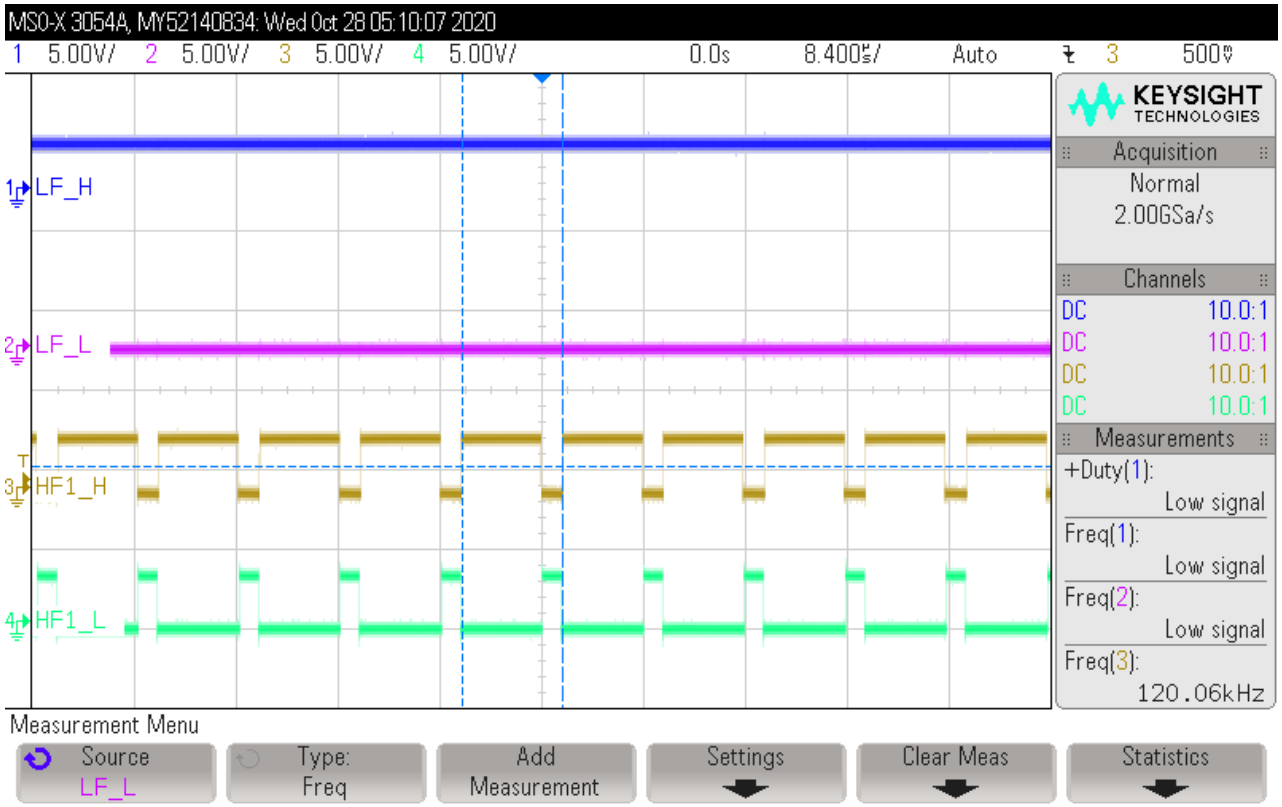
Next, we will check PWM for every phase, for this, we begin with the build option of DC Check. This way we have a steady PWM being generated which is easy to observe on an oscilloscope. For this go back to the settings.h file, and change the TTTPLPFC_DC_CHECK needs to be set to 1, the settings are the same as used for Lab1 Check 1 and not repeated here. Now rebuild the project and reload the project. As you can see from the SW diagram above if the DC_CHECK is set to 1, a fixed duty cycle can be output on any of the phases by entering a value into **TTPLPFC_duty_DC_ref_pu**. (It's not available in .js file. Please copy and paste it in to debug watch window)

Connect oscilloscope pins on TTPLPFC_LOW_FREQ_PWM_H/L(57, 59) , TTPLPFC_HIGH_FREQ_PWM1_H/L(PH1 61, 63 or PH2 58, 60) . Also, connect the TTPLPFC_VAC signal to 3.3V so a positive voltage is present on ac_vol_sensed. After loading and running the project you should see the PWM duty cycle, by default duty is set to 0.5 hence PWM will be observed. Change it to 0.2 to ensure the correct pulse is being modulated. The PWM on the scope will look as follows.


MSO-X 3054A, MY52140834: Tue Oct 27 21:30:29 2020
1 5.00V/ 2 5.00V/ 3 5.00V/ 4 5.00V/ 0.0s 8.400ks/ Stop 3 1.10V
1 LF_H
2 LF_L
3 HF1_H
4 HF1_L
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
2.00GSa/s
Channels
DC 10.0:1
DC 10.0:1
DC 10.0:1
DC 10.0:1
Measurements
+Duty(1):
Low signal
Freq(1):
Low signal
Freq(2):
Low signal
Freq(3):
120.06kHz
Save to file = scope_29
Save
Recall
Default/Erase
Press to Save

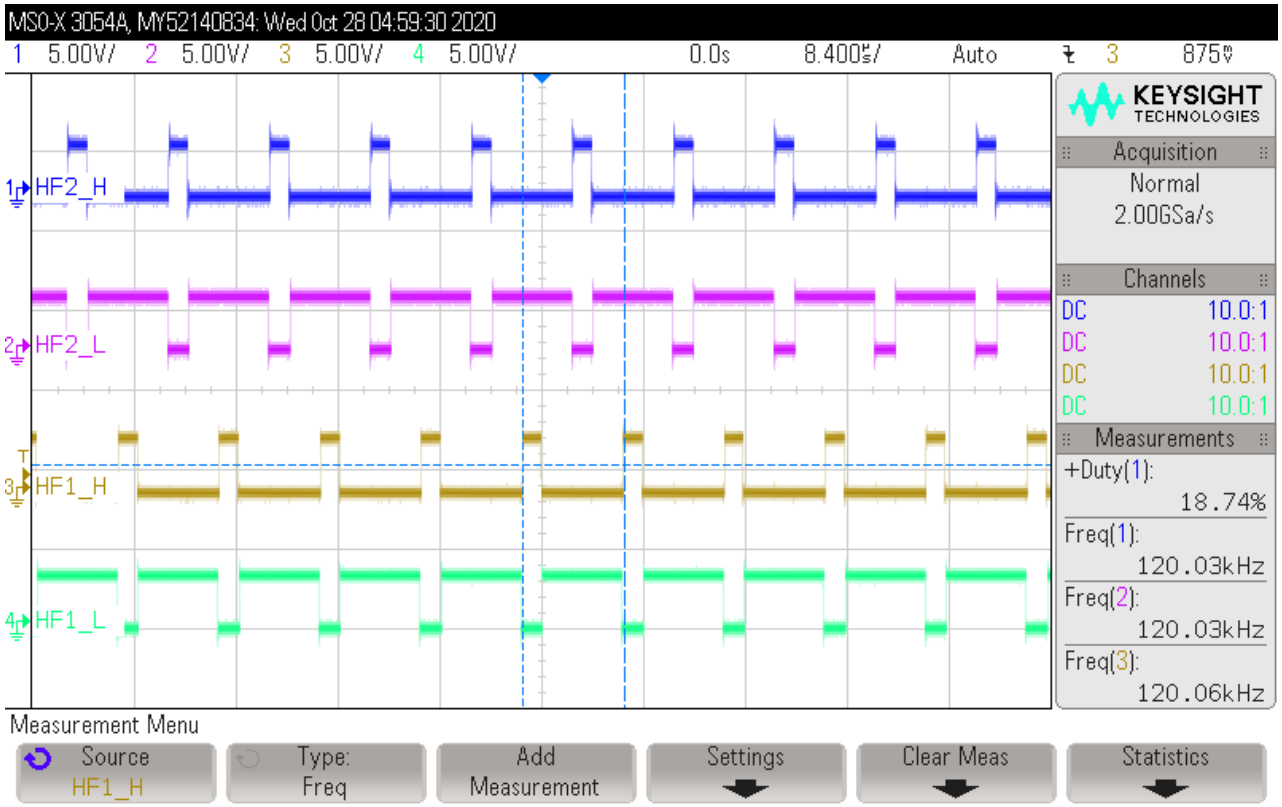
Next, change the ac_vol_sensed to 0V, this will trip the PWM.

Now restart the code, set the duty cycle to be -0.2, and once the trip is cleared after auto_start_slew reaches 100.

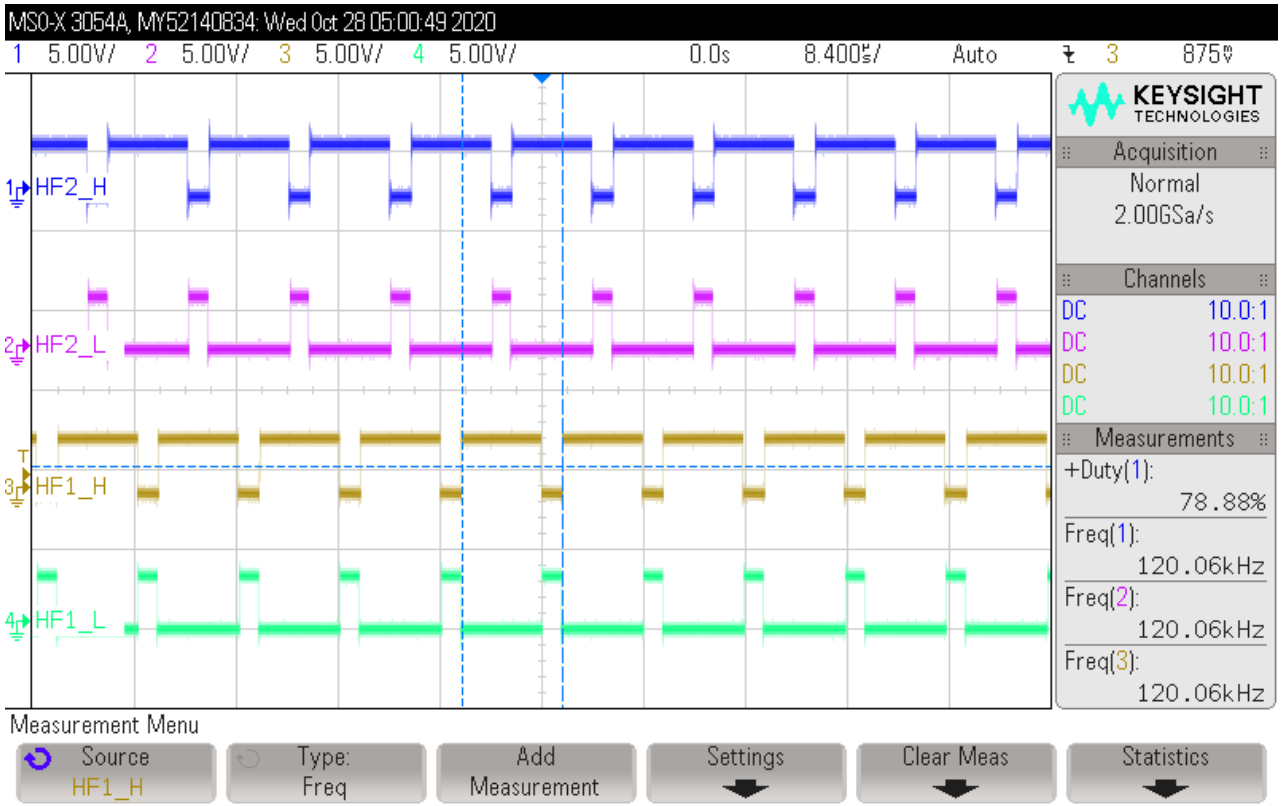

MSO-X 3054A, MY52140834: Wed Oct 28 05:10:07 2020
1 5.00V/ 2 5.00V/ 3 5.00V/ 4 5.00V/ 0.0s 8.400ks/ Auto 3 500m
1 LF_H
2 LF_L
3 HF1_H
4 HF1_L
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
2.00GSa/s
Channels
DC 10.0:1
DC 10.0:1
DC 10.0:1
DC 10.0:1
Measurements
+Duty(1):
Low signal
Freq(1):
Low signal
Freq(2):
Low signal
Freq(3):
120.06kHz
Measurement Menu
Source LF_L
Type: Freq
Add Measurement
Settings
Clear Meas
Statistics

Next check the interleaving operation, and ensure the ph1 and ph2 pwm are 180 degrees apart.

for 0.2 duty, make sure to restart the code with the ac_vol_sensed signal to zero. duty_DC_ref_pu variable is adjusted for the duty change.


MSO-X 3054A, MY52140834: Wed Oct 28 04:59:30 2020
1 5.00V/ 2 5.00V/ 3 5.00V/ 4 5.00V/ 0.0s 8.400ks/ Auto 3 875
1 HF2_H
2 HF2_L
3 HF1_H
4 HF1_L
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
2.00GSa/s
Channels
DC 10.0:1
DC 10.0:1
DC 10.0:1
DC 10.0:1
Measurements
+Duty(1):
18.74%
Freq(1):
120.03kHz
Freq(2):
120.03kHz
Freq(3):
120.06kHz
Measurement Menu
Source HF1_H
Type: Freq
Add Measurement
Settings
Clear Meas
Statistics

for -0.2 duty , make sure to restart the code with the ac_vol_sensed signal to zero


MSO-X 3054A, MY52140834: Wed Oct 28 05:00:49 2020
1 5.00V/ 2 5.00V/ 3 5.00V/ 4 5.00V/ 0.0s 8.400ks/ Auto 3 875
1 HF2_H
2 HF2_L
3 HF1_H
4 HF1_L
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
2.00GSa/s
Channels
DC 10.0:1
DC 10.0:1
DC 10.0:1
DC 10.0:1
Measurements
+Duty(1):
78.88%
Freq(1):
120.06kHz
Freq(2):
120.06kHz
Freq(3):
120.06kHz
Measurement Menu
Source HF1_H
Type: Freq
Add Measurement
Settings
Clear Meas
Statistics

One can also check the Relay connection, when the PWM trip is cleared by the Auto Start routine, the Relay connect is also set to 1,

Probing the Relay pin will observe relay connect go to 1 when the PWM trip is cleared.

Verifying all the waveforms **passes this check**.

You can then halt the execution of the code and return to the editor's view.

10.1.1.4 Lab 1 - Check 4

Before moving on to the real HW, the protection schemes must also be verified.

So far the protection was disabled, to verify the trip mechanisms are working correctly change the protection to enabled in the user settings.h file for lab1. These were set to disable in the previous checks

Select the specific trip to check, these are controller by the #defines below.

```
#if TTPLPFC_LAB == 1
#define TTPLPFC_DC_CHECK 1
#define TTPLPFC_PWM_AC_CHECK 0
#define TTPLPFC_AUTO_START 1
#define TTPLPFC_IL1_PROTECTION_ENABLE 1
#define TTPLPFC_IL2_PROTECTION_ENABLE 0
#define TTPLPFC_FAULTn_PROTECTION_ENABLE 0

#define TTPLPFC_SPLL_METHOD_SELECT TTPLPFC_SPLL_1PH_SOGI_SEL
#define TTPLPFC_CONTROL_RUNNING_ON 1
#define TTPLPFC_CURRENT_LOOP_CLOSED 0
#define TTPLPFC_VOLTAGE_LOOP_CLOSED 0
```

#define TTPLPFC_ADAPTIVE_DEADTIME 0
#define TTPLPFC_PHASE_SHEDDING_ENABLED 0
#define TTPLPFC_SFRA_TYPE 1
#endif

Now build and load the code. Put 1.65V on IL1 sensed pin. And now run the code. The VAC sensed pin is set to 3.3V.

The PWM will be started once the autoStartSlew reaches 100.

Now move the IL1_sensed to "0"V or 3.3V , this will trip the PWM.

Next, change the #defines to check IL2 only and repeat the steps.

Next check FAULTn trip, FAULTn being 0 indicates FAULT has occurred. enable the protection from FAULTn by modifying the #define and connect the pin to 3.3V. The PWM will appear once the code is loaded and ran and autoStartSlew reaches 100.

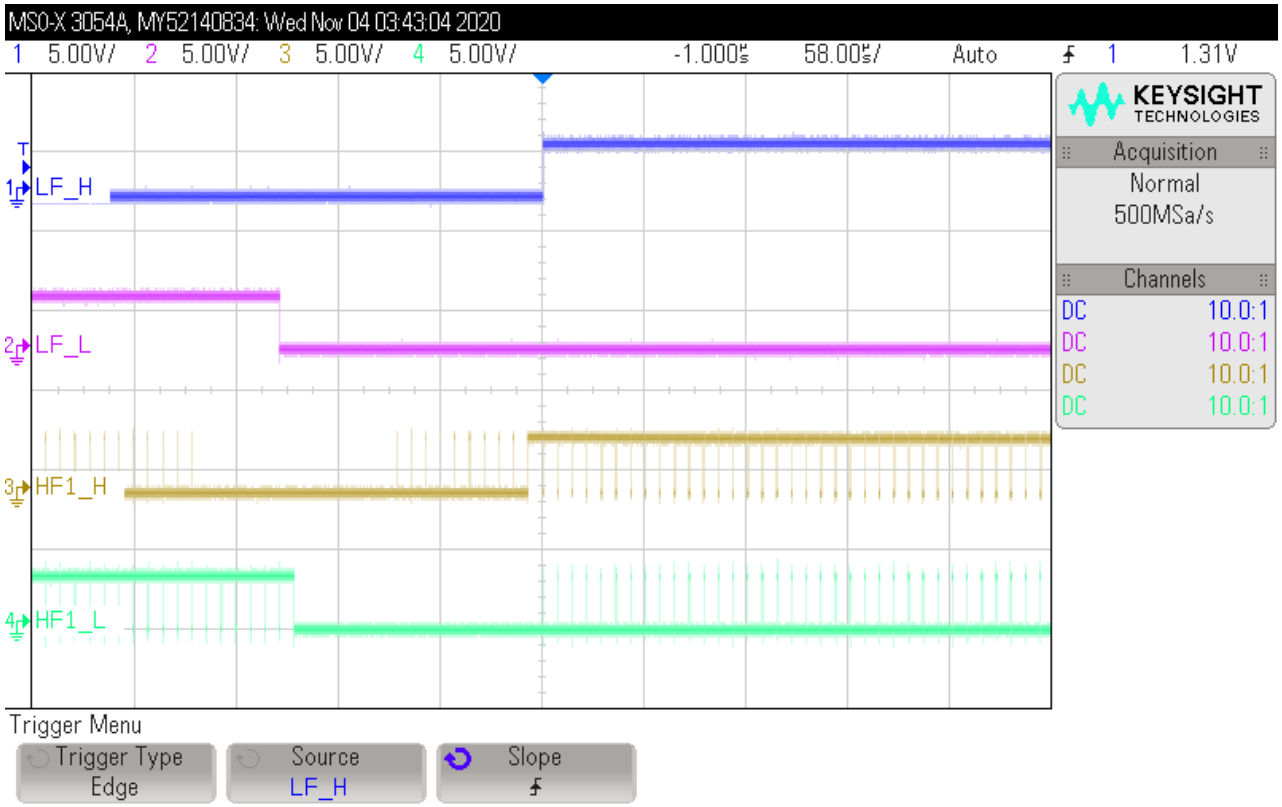
This completes the verification of this build.

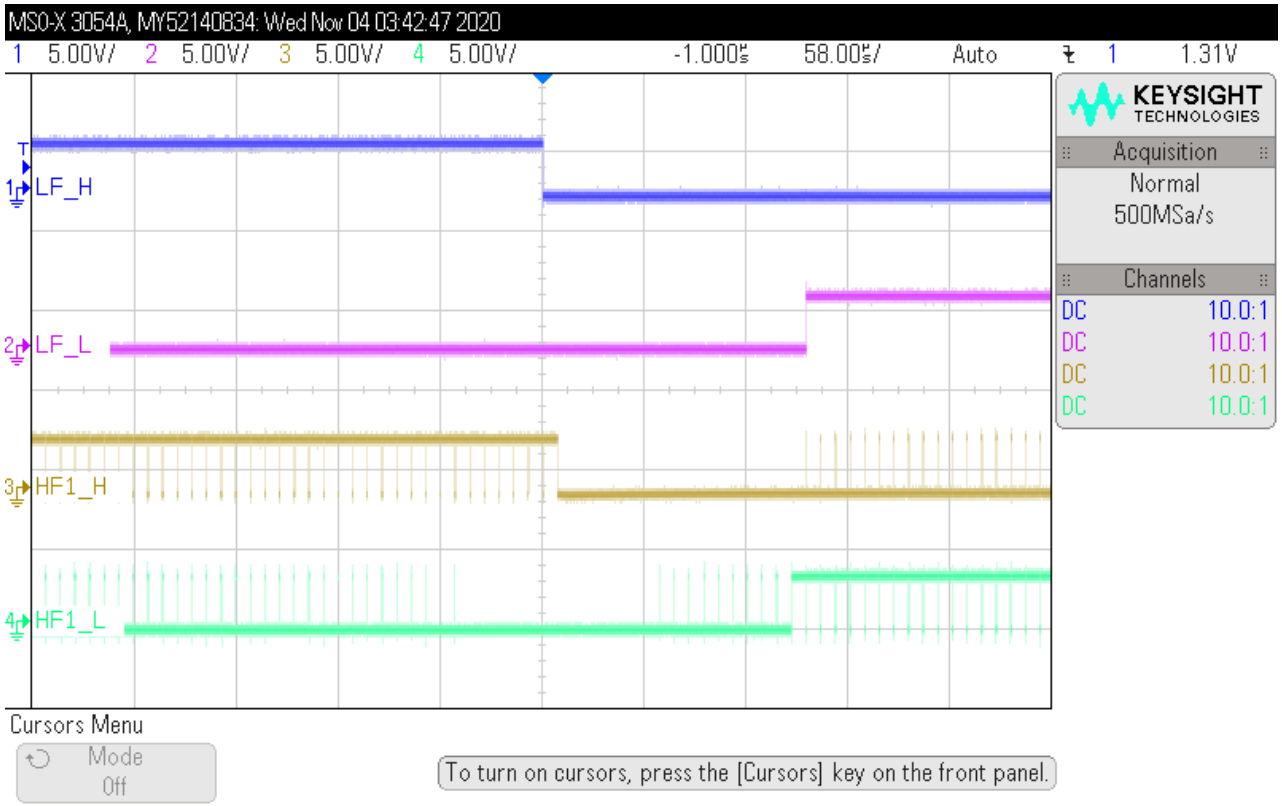
10.1.1.5 **Lab 1 - Check 5**

This lab can be used to check the PWM waveform under a forced AC duty cycle. This allows to check for the soft starting scheme around zero crossing. To force clear the PWM trip set TTPLPFC_clearTrip to "1"

#if TTPLPFC_LAB == 1
#define TTPLPFC_DC_CHECK 0
#define TTPLPFC_PWM_AC_CHECK 1
#define TTPLPFC_AUTO_START 1
#define TTPLPFC_IL1_PROTECTION_ENABLE 0

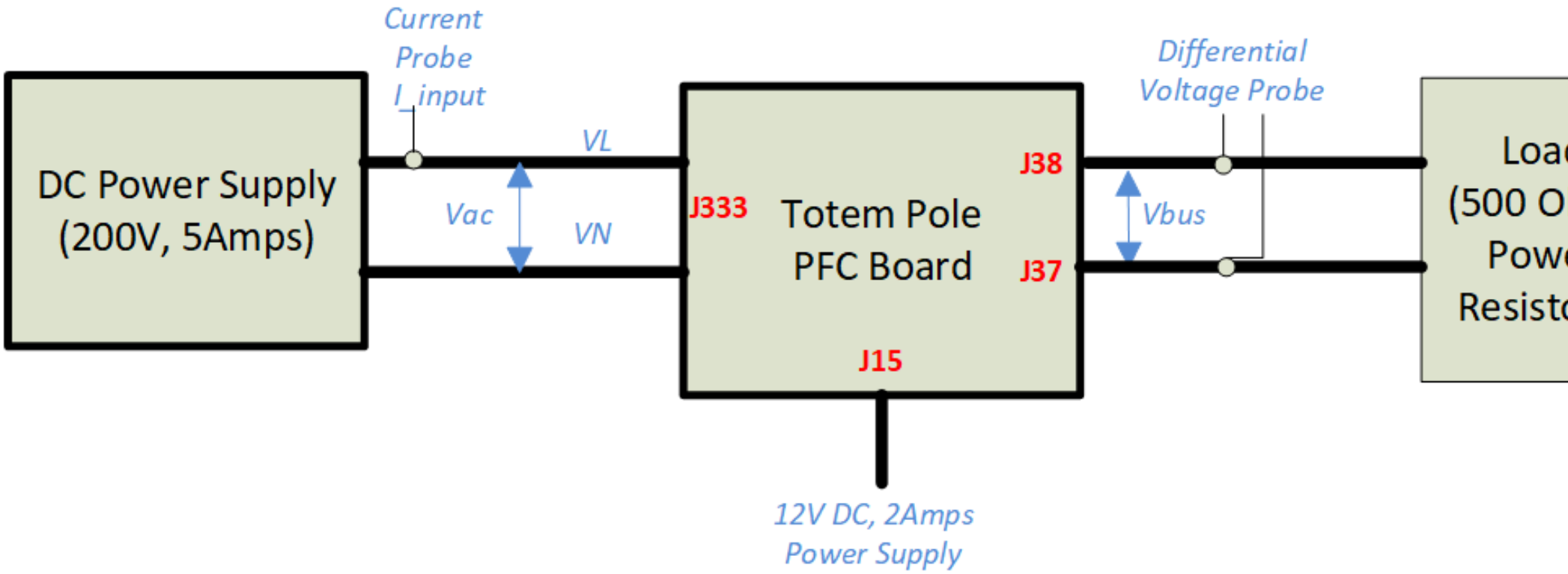
```
#define TTPLPFC_IL2_PROTECTION_ENABLE 0
#define TTPLPFC_FAULTn_PROTECTION_ENABLE 0

#define TTPLPFC_SPLL_METHOD_SELECT TTPLPFC_SPLL_1PH_SOGI_SEL
#define TTPLPFC_CONTROL_RUNNING_ON 1
#define TTPLPFC_CURRENT_LOOP_CLOSED 0
#define TTPLPFC_VOLTAGE_LOOP_CLOSED 0
#define TTPLPFC_ADAPTIVE_DEADTIME 0
#define TTPLPFC_PHASE_SHEDDING_ENABLED 0
#define TTPLPFC_SFRA_TYPE 1
#endif
```


MSO-X 3054A, MY52140834: Wed Nov 04 03:43:04 2020
1 5.00V/ 2 5.00V/ 3 5.00V/ 4 5.00V/ -1.000s 58.00s/ Auto 1 1.31V
LF_H
LF_L
HF1_H
HF1_L
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
500MSa/s
Channels
DC 10.0:1
DC 10.0:1
DC 10.0:1
DC 10.0:1
Trigger Menu
Trigger Type Edge
Source LF_H
Slope


MSO-X 3054A, MY52140834: Wed Nov 04 03:42:47 2020
1 5.00V/ 2 5.00V/ 3 5.00V/ 4 5.00V/
-1.000s 58.00s/ Auto 1 1.31V
LF_H
LF_L
HF1_H
HF1_L
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
500MSa/s
Channels
DC 10.0:1
DC 10.0:1
DC 10.0:1
DC 10.0:1
Cursors Menu
Mode
Off
To turn on cursors, press the [Cursors] key on the front panel.

10.1.1.6 **Lab1 - Check 6**

In this lab, the board is excited in open-loop fashion with a fixed duty cycle. The duty cycle is controlled with dutyPU_DC variable. This build verifies the sensing of feedback values from the power stage and also the operation of the PWM gate driver and ensures there are no hardware issues. Additionally, calibration of input and output voltage sensing can be performed in this build. The software structure for this build is shown below. There are two ISR in the system: a fast ISR for the current loop and a slower ISR to run the voltage loop and instrumentation functions.


Current
Probe
I_input
DC Power Supply
(200V, 5Amps)
VL
Vac
VN
J333
Totem Pole
PFC Board
J38
J37
J15
Differential
Voltage Probe
Vbus
12V DC, 2Amps
Power Supply

The following defines must be verified

```
#if TTPLPFC_LAB == 1
#define TTPLPFC_CONTROL_RUNNING_ON 1
#define TTPLPFC_DC_CHECK 1
#define TTPLPFC_CURRENT_LOOP_CLOSED 0
#define TTPLPFC_VOLTAGE_LOOP_CLOSED 0
#define TTPLPFC_ADAPTIVE_DEADTIME 0
#define TTPLPFC_PHASE_SHEDDING_ENABLED 0
#define TTPLPFC_SFRA_TYPE 1
#define TTPLPFC_PWM_AC_CHECK 0
#define TTPLPFC_AUTO_START 1
#define TTPLPFC_IL1_PROTECTION_ENABLE 1
#define TTPLPFC_IL2_PROTECTION_ENABLE 1
#define TTPLPFC_FAULTn_PROTECTION_ENABLE 1
#define TTPLPFC_SPLL_METHOD_SELECT TTPLPFC_SPLL_1PH_SOGI_SEL
#endif
```

Run the project. In a few seconds the inrush relay clicks, the software is programmed to do so in the lab1 once the autoStartSlew reaches zero. The trip clears, and a duty cycle of 0.5 is applied.

In the watch view, check if the TTPLPFC_ac_vol_sensed_Volts, TTPLPFC_vBus_sensed_Volts, TTPLPFC_iL1_sensed_Amps, variables are updating, periodically.

Now slowly increase the input DC voltage from zero to 120 V.

The output voltage shows a boosted voltage as a steady duty cycle of 0.5 PU is applied as a default setting. (the code is designed to look at the ac_vol_sensed and if the amplitude does not match that of the duty cycle a trip is issued on the PWM.

If a high current is drawn, reduce the voltage to zero first and correct the issue before resuming the test

Verifying the voltage sensing: Make sure TTPLPFC_ac_vol_sensed_Volts and TTPLPFC_vBus_sensed_Volts display the correct values, for 120-V DC input, TTPLPFC_vBus_sensed_Volts is close to 240V. This verifies the voltage sensing of the board in some manner.

Verifying the current sensing: Notice the TTPLPFC_iL1_sensed_Amps for the given test condition; this value is close to 1 A.

This verifies at a basic level the PWM driver and connection of hardware, user can change the dutyPU_DC variable to see operation under various boost conditions.

Once finished, reduce the input voltage to zero and watch for the bus voltages to reduce down to zero.

This completes the check for positive half of the AC cycle with DC input. The controller can now be halted, and reset.

Now swap the DC voltage terminal such that a negative voltage can be applied to the DUT.

Now run the project, before the inrush relay clicks in set TTPLPFC_duty_pu_DC to be -0.5. In a few seconds the inrush relay clicks, the software is programmed to do so in the lab1 once the autoStartSlew reaches zero. The trip clears, and a duty cycle is applied.

In the watch view, check if the TTPLPFC_ac_vol_sensed_Volts, TTPLPFC_vBus_sensed_Volts, TTPLPFC_iL1_sensed_Amps, variables are updating, periodically.

Now slowly increase the input DC voltage from zero to -120 V.

The output voltage shows a boosted voltage as a steady duty cycle of 0.5 PU is applied as a default setting. (the code is designed to look at the ac_vol_sensed and if the amplitude does not match that of the duty cycle a trip is issued on the PWM.

If a high current is drawn, reduce the voltage to zero first and correct the issue before resuming the test

Verifying the voltage sensing: Make sure TTPLPFC_ac_vol_sensed_Volts and TTPLPFC_vBus_sensed_Volts display the correct values, for -120-V DC input, TTPLPFC_vBus_sensed_Volts is close to 240V. This verifies the voltage sensing of the board in some manner.

Verifying the current sensing: Notice the TTPLPFC_iL1_sensed_Amps for the given test condition; this value is close to -1 A.

This verifies at a basic level the PWM driver and connection of hardware, user can change the dutyPU_DC variable to see operation under various boost conditions.

Once finished, reduce the input voltage to zero and watch for the bus voltages to reduce down to zero.

This completes the check for negative half of the AC cycle with DC input. The controller can now be halted, and reset.

Fully halting the MCU when in real-time mode is a two-step process. First halt the processor by using the Halt button on the toolbar or by using Target → Halt. Then take the MCU out of real-time mode. Finally, reset the MCU and Close CCS debug session by clicking on Terminate Debug Session (Target → Terminate all).

10.1.1.7 **Lab1 - Check 7**

This lab is used to check the quality of measurement on the actual board and the accuracy of the measurement.

The setup is exactly as outlined in Lab1-Check6.

The following tables must be filled to make sure the sensing circuitry is correct and if any calibration is required. The offset values needed for the ac_vol_sensed, ac_cur_sensed etc need to be identified in this build and the code modified to reflect those values.

These are programmed into the code in the main.c file with the following #defines

```
//
// note the value below is determined in Lab1 Check 7
//
TTPLPFC_iL1_senseOffset_pu = TTPLPFC_IL1_OFFSET_PU;
TTPLPFC_iL2_senseOffset_pu = TTPLPFC_IL2_OFFSET_PU;
TTPLPFC_ac_vol_senseOffset_pu = TTPLPFC_VAC_OFFSET_PU;
```

these defines are in the user_settings.h file

#define TTPLPFC_IL1_OFFSET_PU ((float32_t)0.50)
#define TTPLPFC_IL2_OFFSET_PU ((float32_t)0.50)
#define TTPLPFC_VAC_OFFSET_PU ((float32_t)0.50

	From Multimeter	From watch window	%error
TTPLPFC_ac_vol_sensed_Volts TTPLPFC_ac_vol_senseOffset_pu = 0.5	0		Need to adjust ac_vol_senseOffset_pu ?
TTPLPFC_vBus_sensed_Volts			

	From Multimeter	From watch window	%error
TTPLPFC_iL1_sensed_Amps TTPLPFC_iL1_senseOffset_pu = 0.5	0		Need to adjust iL1_senseOffset_pu ?

	From Multimeter	From watch window	%error

	From Multimeter	From watch window	%error
TTPLPFC_vBus_sensed_Volts	0		

10.1.1.8 **Lab1 - Check 8**

Check measurements under AC input, no switching action on the board. The following defines are used under Lab1 options. The auto start is disabled.

#if TTPLPFC_LAB == 1
#define TTPLPFC_CONTROL_RUNNING_ON 1

#define TTPLPFC_DC_CHECK 0
#define TTPLPFC_CURRENT_LOOP_CLOSED 0
#define TTPLPFC_VOLTAGE_LOOP_CLOSED 0
#define TTPLPFC_ADAPTIVE_DEADTIME 0
#define TTPLPFC_PHASE_SHEDDING_ENABLED 0
#define TTPLPFC_SFRA_TYPE 1
#define TTPLPFC_PWM_AC_CHECK 0
#define TTPLPFC_AUTO_START 0
#define TTPLPFC_IL1_PROTECTION_ENABLE 1
#define TTPLPFC_IL2_PROTECTION_ENABLE 1
#define TTPLPFC_FAULTn_PROTECTION_ENABLE 1
#define TTPLPFC_SPLL_METHOD_SELECT TTPLPFC_SPLL_1PH_SOGI_SEL
#endif

The data logger is used to check the AC input measurement, the PLL locking and the current sense.

See in ISR2 under datalogger routine for Lab 1

```
//
// check output voltage and inverter current meas.
//
TTPLPFC_dVal1 = TTPLPFC_ac_vol_sensed_pu;
TTPLPFC_dVal2 = TTPLPFC_spll_sine;
TTPLPFC_dVal3 = TTPLPFC_iL1_sensed_pu;
TTPLPFC_dVal4 = TTPLPFC_iL2_sensed_pu;
```

This build is used to check the AC voltage sensed and the AC current sensed to be correct.

Re-build and load the project. Enable real time and run the project.

Close the inrush relay by setting TTPLPFC_relayConnect to "1"

Now add the dataloggger by importing graph1.graphprop to see the buffer through Tools→ Graph → Dual Time

10.1.1.9 **Lab1 - Check 9**

Soft start the PFC,

```
#if TTPLPFC_LAB == 1
#define TTPLPFC_DC_CHECK 1
#define TTPLPFC_PWM_AC_CHECK 0
#define TTPLPFC_AUTO_START 1
#define TTPLPFC_DEBUG_STARTUP 1

#define TTPLPFC_TIMED_STARTUP 1
#define TTPLPFC_STARTUP_TIME_MS 2

#define TTPLPFC_AC_HALF_CHECK     TTPLPFC_AC_NEGATIVE_HALF_DC_CHECK
#define TTPLPFC_IL1_PROTECTION_ENABLE 0
#define TTPLPFC_IL2_PROTECTION_ENABLE 0
#define TTPLPFC_FAULTn_PROTECTION_ENABLE 0
#define TTPLPFC_SPLL_METHOD_SELECT TTPLPFC_SPLL_1PH_SOGI_SEL
#define TTPLPFC_CONTROL_RUNNING_ON 1
#define TTPLPFC_CURRENT_LOOP_CLOSED 0
#define TTPLPFC_VOLTAGE_LOOP_CLOSED 0
```

#define TTPLPFC_SFRA_TYPE 1
#define TTPLPFC_ADAPTIVE_DEADTIME 0
#define TTPLPFC_PHASE_SHEDDING_ENABLED 0
#endif

With the settings above, if

#define TTPLPFC_TIMED_STARTUP 1

is set to 1 , then the PFC is only turned on for a small duration of time controller by

#define TTPLPFC_STARTUP_TIME_MS 2

The initial duty cycle and the final duty cycle are controlled by the following piece of code in the ttplpfc.c

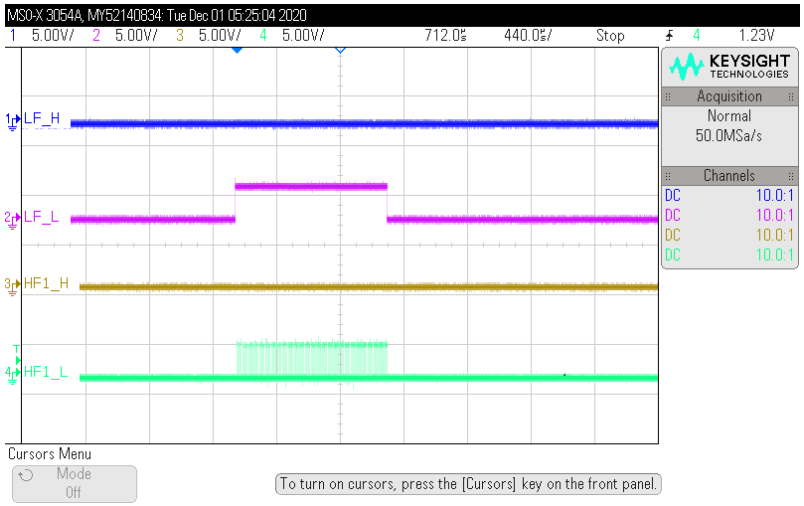
TTPLPFC_duty1_DC_ref_pu = 1.0 - 0.03f;
TTPLPFC_duty2_DC_ref_pu = 1.0 - 0.03f;

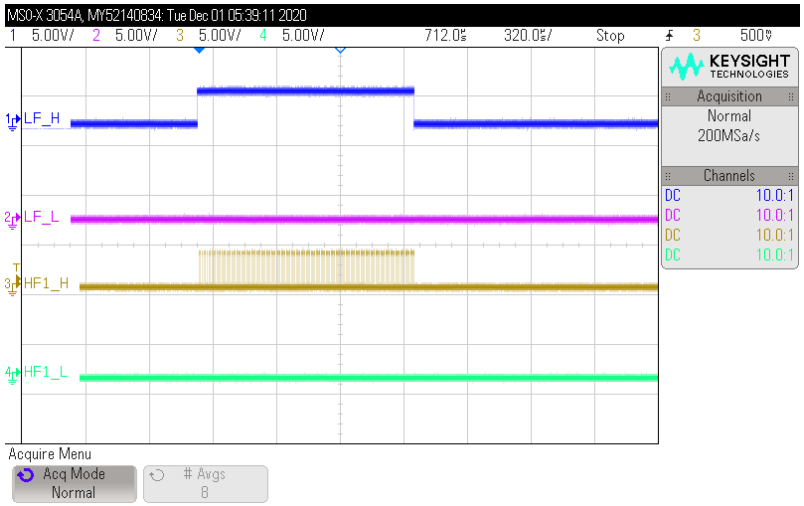
TTPLPFC_duty1_DC_ref_final_pu = 0.5f;
TTPLPFC_duty2_DC_ref_final_pu = 0.5f;

this will also vary based on the AC HALF that is selected for the duty output.

The code will sense the ac_voltage and depending on the sign adjust the duty cycle command appropriately.

The images below show the debug startup of the PFC under DC conditions.



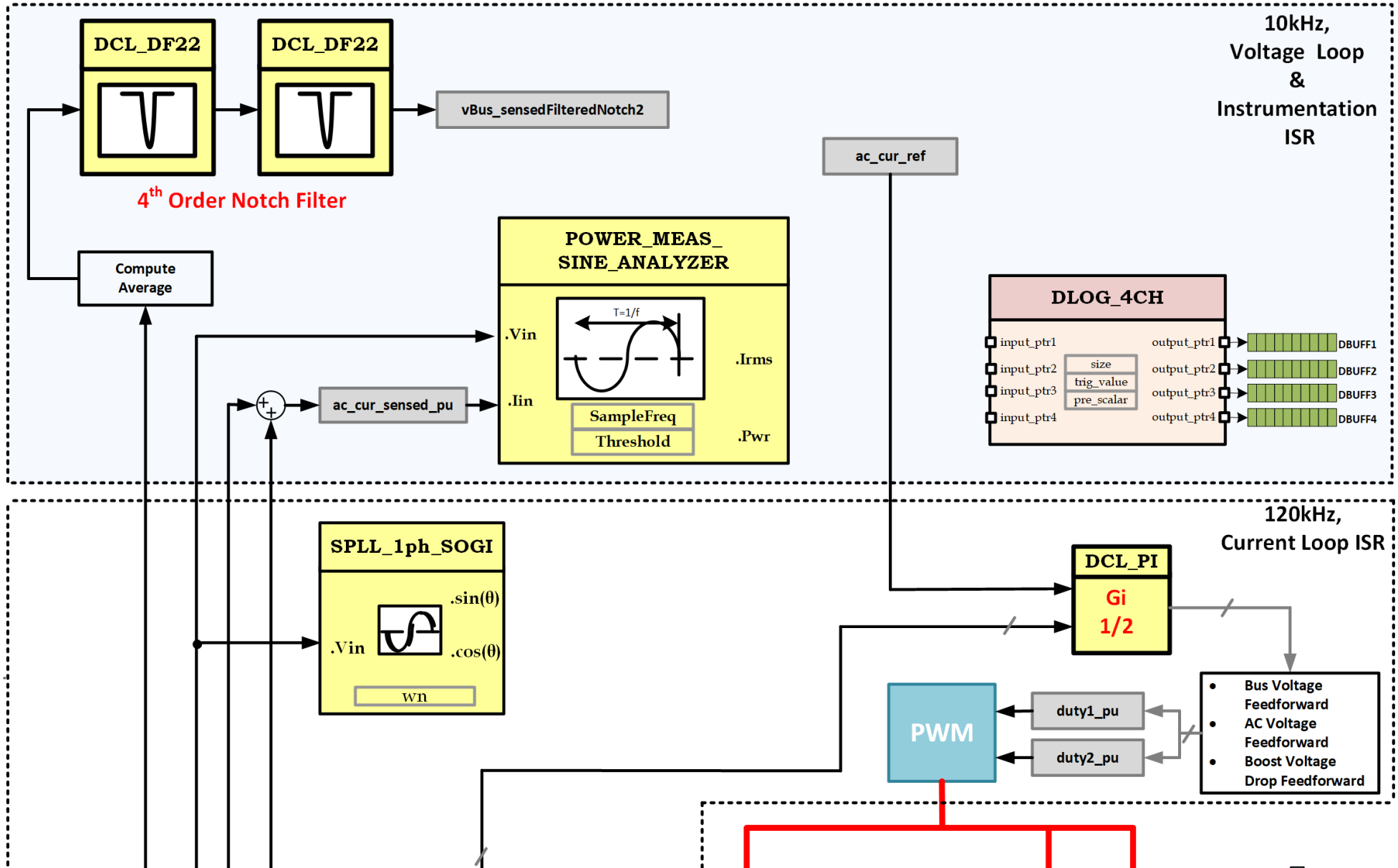


To re-initiate the PWM pulse, just write TTPLPFC_autoStartSlew variable to a value less than 100, this will push another set of pulses.

To increase the ms the pulses are on change the TTPLPFC_timedRun_ms (to make this change for next time the code is ran, change the #define **TTPLPFC_STARTUP_TIME_MS** and recompile and load the code).

To disable the timed pulse train of PWM and let the PWM run continuously, change the #define TTPLPFC_TIMED_STARTUP to be zero and recompile, and load the code.

10.1.2 Lab2


10kHz,
Voltage Loop
&
Instrumentation
ISR
DCL_DF22
DCL_DF22
vBus_sensedFilteredNotch2
4th Order Notch Filter
ac_cur_ref
Compute
Average
POWER_MEAS_
SINE_ANALYZER
.Vin
.Iin
T=1/f
.Irms
.Pwr
SampleFreq
Threshold
DLOG_4CH
input_ptr1
input_ptr2
input_ptr3
input_ptr4
size
trig_value
pre_scalar
output_ptr1
output_ptr2
output_ptr3
output_ptr4
DBUFF1
DBUFF2
DBUFF3
DBUFF4
ac_cur_sensed_pu
120kHz,
Current Loop ISR
SPLL_1ph_SOGI
.Vin
.sin(θ)
.cos(θ)
wn
DCL_PI
Gi
1/2
PWM
duty1_pu
duty2_pu
Bus Voltage Feedforward
AC Voltage Feedforward
Boost Voltage Drop Feedforward

Take a look at “ttplpfc_settings.h”

obc_6_6kw_main.c | ttplpfc_settings.h

```
43 /* Device Related Defines */
44 #define CPU_SYS_CLOCK (200*1000000)
45 #define PWMSYSCLOCK_FREQ (200*1000000)
46 #define ECAPSYSCLOCK_FREQ   (200*1000000)
47
48 //
49 // Project Options
50 // Labs for System check-out
51 // LAB 1:PFC Open Loop Check (DC Input , with auto start)
52 //
53 //       PFC Check AC voltage sensing under AC condition (without auto start)
54 // LAB 2:PFC Closed Current Loop Check (DC Input)
55 // LAB 3:PFC Closed Current Loop Check (AC Input)
56 // LAB 4:PFC Closed Voltage Loop Check (AC Input)
57 // LAB 5:INV Open Loop Check (DC Output)
58 // LAB 6:INV Open Loop Check (AC Output)
59 // LAB 7:INV Closed Current Loop Check (DC Output)
60 // LAB 8:INV Closed Current Loop Check (AC Output)
61 // LAB 9:INV Closed Current Loop Check (Grid Connection)
62 //
63 #define TTPLPFC_LAB 4
```

To update the coefficients at run time the TTPLPFC_updateCoefficientsCommand can be set to "1" , this will load the values from TTPLPFC_gi_kp_ref to TTPLPFC_gi_kp which is what is used for the current compensators

10.1.2.1 Lab2 - Check 1

Lab 1 and Lab 2 are both labs that run with DC voltage and low voltage.

Lab 1 will configure the unit to run open voltage loop and open current loop.

Lab 2 will configure the unit to run open voltage loop with the current loop closed.

Lab 2 instructions.

1. Set the input voltage to DC with a value of zero voltage.
2. Apply bias power.
3. Set the lab to 2
4. Build and run the code.
5. You can confirm which lab you running by looking at these variables in the expressions window.

1. In the expressions you will see this variable count to 100. Once it reaches 100 the controller will close the relay and start outputting pulses. If you don't want to wait for it to reach 100 you can type a number in the yellow box and the counting will start from that number. For example if you enter 95 it will count to 100 in 5 counts.

1. Duty cycle in this code is really (1-D). That is it is the duty cycle of the synchronous rectifier. Since the current loop is closed you can't directly control the duty cycle. However there are 2 duty cycle clamps that you can use to prevent the controller from putting out a "bad" duty cycle. For a positive input voltage set the clamps as shown below.

1. Now you know that the duty cycle will always output a "safe" value for a positive input voltage.
2. Next you can use this variable to set the current reference. 0.03 might be too big for the load you have. You'll be able to tell if the current loop is regulating when the current sense signals shown below match the TTPLPFC_ac_cur_ref_pu. This won't happen until the output voltage starts to rise a bit.

1. Once the unit is running you can dial the input voltage up. At this point it doesn't really even need a load but you can use 100mA or so pretty easily without the coldplate attached all the way up to a 200V input with 400V output.

2. Make sure the dc voltage you apply to the input has a polarity of + to – where the positive is on the line connection and the negative is on the neutral connection.
3. You can repeat the above steps for a negative input voltage if desired. Start by dialing the input voltage all the way to 0.
4. Set the clamps up like this.

1. Set TTPLPFC_ac_cur_ref_pu to -0.03 (or something smaller in magnitude)
2. Slowly dial the input voltage to a larger negative voltage.
3. When running the test with either positive or negative input voltage stop increasing the input voltage immediately if you see an abnormally large current flow from the source. This means something in code or hardware is not right and will need to be corrected.

10.1.3 Lab 3

To run the Lab3 at low voltage build the code with

→ AUTO_START = 0

→ TTPLPFC_RlyConnect = 1

→ TTPLPFC_pwm_SwState.enum_pwmSwState = pwmSwState_normalOperation; (or 0 for F29x)

I recommend starting with Lab 1.

1. Set the input voltage to DC with a value of zero voltage.
2. Apply bias power.
3. Set the lab to 1
4. Build and run the code.

5. You can confirm which lab you running by looking at these variables in the expressions window.

1. In the expressions you will see this variable count to 100. Once it reaches 100 the controller will close the relay and start outputting pulses. If you don't want to wait for it to reach 100 you can type a number in the yellow box and the counting will start from that number. For example if you enter 95 it will count to 100 in 5 counts.

1. You can configure the duty cycle in open loop with this variable. Duty cycle in this code is really (1-D). That is it is the duty cycle of the synchronous rectifier.

1. Once the unit is running you can dial the input voltage up. At this point it doesn't really even need a load but you can use 100mA or so pretty easily without the coldplate attached all the way up to a 200V input with 400V output.
2. Make sure the dc voltage you apply to the input has a polarity of + to – where the positive is on the line connection and the negative is on the neutral connection.

L

11 CLLLC

11.1 Rogowski Coil Active Sync Rect

11.1.1 Rogowski coil polarity change

The follow code connects the TRIP5 to be be coming from CMPSSH and TRIP7 to be coming from CMPSSL

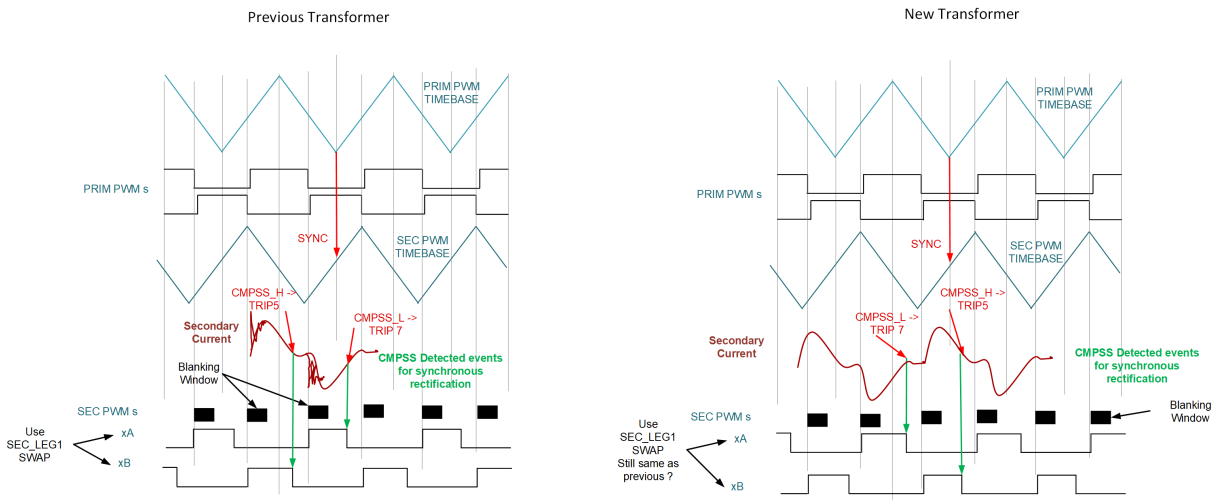
```
827         XBAR_setEPWMMuxConfig(XBAR_TRIP5,
828                               CLLLC_ISEC_TANK_H_PWM_XBAR_MUX_VAL);
829         XBAR_enableEPWMMux(XBAR_TRIP5,
830                            CLLLC_ISEC_TANK_H_XBAR_MUX);
831
832         XBAR_setEPWMMuxConfig(XBAR_TRIP7,
833                               CLLLC_ISEC_TANK_L_PWM_XBAR_MUX_VAL);
834         XBAR_enableEPWMMux(XBAR_TRIP7,
835                            CLLLC_ISEC_TANK_L_XBAR_MUX);
```

However if the rogowski coil polarity is different , the CMPSS even will not happen on the right spot as expected by the synchronous rectification driver.

A simple fix to comprehend this inversion is to flip the selection to be TRIP7 to select CMPSSH, and TRIP5 to select CMPSSL as shown below

```
827         XBAR_setEPWMMuxConfig(XBAR_TRIP7,
828                               CLLLC_ISEC_TANK_H_PWM_XBAR_MUX_VAL);
829         XBAR_enableEPWMMux(XBAR_TRIP7,
830                            CLLLC_ISEC_TANK_H_XBAR_MUX);
831
832         XBAR_setEPWMMuxConfig(XBAR_TRIP5,
833                               CLLLC_ISEC_TANK_L_PWM_XBAR_MUX_VAL);
834         XBAR_enableEPWMMux(XBAR_TRIP5,
835                            CLLLC_ISEC_TANK_L_XBAR_MUX);
```

11.1.2 Transformer Polarity Change



For the new transformer because the terminals are swapped i.e. the return leg is not connected to SEC Side LEG1 the active sync rectification scheme needs to change

The following changes are necessary, these are much more involved than what is needed for the rogowski coil inversion alone.

1. The below changes are necessary because TRIP5 which is qualified as DCAEVT2, the "D" need to be changed to "U" as the phase of PWM in which these event happen is changed

```
927         //
928         // no describe the behavior in case when DCAEVT2 and
929         //
930         EPWM_setTripZoneAdvDigitalCompareActionA(CLLLC_SEC_LEG1_PWM_BASE,
931                                                  EPWM_TZ_ADV_ACTION_EVENT_DCxEVT2_U,
932                                                  EPWM_TZ_ADV_ACTION_LOW);
933
934         EPWM_setTripZoneAdvDigitalCompareActionB(CLLLC_SEC_LEG1_PWM_BASE,
935                                                  EPWM_TZ_ADV_ACTION_EVENT_DCxEVT2_D,
936                                                  EPWM_TZ_ADV_ACTION_LOW);
937
938         EPWM_setTripZoneAdvDigitalCompareActionA(CLLLC_SEC_LEG2_PWM_BASE,
939                                                  EPWM_TZ_ADV_ACTION_EVENT_DCxEVT2_D,
940                                                  EPWM_TZ_ADV_ACTION_LOW);
941
942         EPWM_setTripZoneAdvDigitalCompareActionB(CLLLC_SEC_LEG2_PWM_BASE,
943                                                  EPWM_TZ_ADV_ACTION_EVENT_DCxEVT2_U,
944                                                  EPWM_TZ_ADV_ACTION_LOW);
945
```

2. For proper generation of the PWM pulse for the secondary side the action qualifier event needs to change

```
2728    //
2729    // Action Qualifier SubModule Registers
2730    // CTR = CMPA@period , xA set to 1
2731    //
2732    EPWM_setActionQualifierAction(base1, EPWM_AQ_OUTPUT_A ,
2733       EPWM_AQ_OUTPUT_HIGH, EPWM_AQ_OUTPUT_ON_TIMEBASE_PERIOD );
2734    //
2735    // CTR = CMPA@Down , xA set to 0
2736    //
2737    EPWM_setActionQualifierAction(base1, EPWM_AQ_OUTPUT_A ,
2738       EPWM_AQ_OUTPUT_LOW, EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPA);
2739
2740    //
2741    // CTR = CMPB@0, xB set to 1
2742    //
2743    EPWM_setActionQualifierAction(base1, EPWM_AQ_OUTPUT_B ,
2744       EPWM_AQ_OUTPUT_HIGH, EPWM_AQ_OUTPUT_ON_TIMEBASE_ZERO);
2745    //
2746    // CTR = CMPB@UP , xB set to 0
2747    //
2748    EPWM_setActionQualifierAction(base1, EPWM_AQ_OUTPUT_B ,
2749       EPWM_AQ_OUTPUT_LOW, EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPB);
2750
2751
```

3. Correspondingly the duty computation will need to flip for xA and xB to get the right PWM to come out

```
1043    //
1044    // For secondary side the PWM is not centered around zero or period
1045    // hence multiply by 2 (<<1) for period*duty
1046    //
1047    CLLLC_pwmDutyBSec_ticks = (uint32_t)((float32_t)CLLLC_pwmPeriod_ticks *
1048                                         (float32_t)
1049                                         (fabsf(CLLLC_pwmDutySec_pu))) << 1;
1050
1051    //
1052    // for secondary side B ticks = period - duty_a
1053    //
1054    CLLLC_pwmDutyASec_ticks = (CLLLC_pwmPeriod_ticks) -
1055                              CLLLC_pwmDutyASec_ticks;
1056
```

11.2 Adding More Filtering on CLLLC GaN Fault Signals

Due to system noise, additional filtering may be needed. This can be adjusted by changing the qualification period. Note a set of PINS will be affected by this qualification. For example CLLLC GaN Fault is connected to GPIO23, so the new qualification will apply to GPIO16-GPIO23

```
#if CLLLC_BOARD_PROTECTION_GANFAULT == 1
    GPIO_setDirectionMode(CLLLC_GANFAULTn_GPIO, GPIO_DIR_MODE_IN);
    GPIO_setQualificationMode(CLLLC_GANFAULTn_GPIO, GPIO_QUAL_6SAMPLE);
    GPIO_setPinConfig(CLLLC_GANFAULTn_GPIO_PIN_CONFIG);
    GPIO_setPadConfig(CLLLC_GANFAULTn_XBAR_BASE, GPIO_PIN_TYPE_STD);
    GPIO_setQualificationPeriod(CLLLC_GANFAULTn_GPIO, 6);
```

This is disruptive for other functions such as CAN etc that may share the GPIO block. Hence we may need to find an alternative approach later. Lets use this to see if additional filtering helps resolve the startup trip.

11.3 Labs

1	Charging : ISR, PWM & ADC Check	Check 1 - Test ISR Structure Check 2 - Test ADC mapping and reading of conversion data. Check 3 - Test PWM driver Check 4 - Test Board Protection Check 5 - GPIO configuration	These checks can be run on a control card, with no protection or limited protection.
2	Charging : Open-loop: sensing and actuation		

3	Charging : Voltage loop		
4	Charging : Current Loop		
5	Charging : Battery Charging Mode		not scoped
6	V2L :PWM check		
7	V2L: Open Loop		
8	V2L: Closed loop voltage		
9			

11.3.1 Lab 1

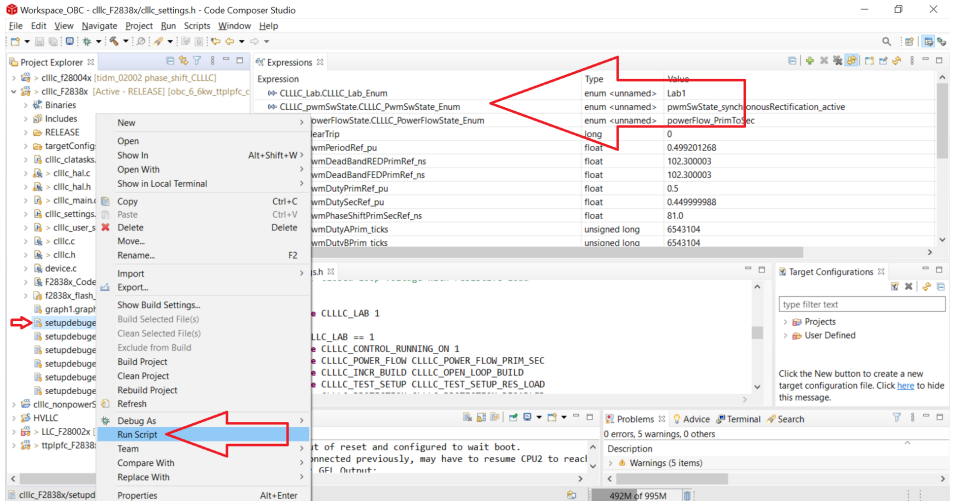
The checks in lab 1 are meant to be run on a controlCARD in a docking station with no protection or limited protection.

Set the project to Lab 1 by changing the Lab Number in the <settings.h> file, (this will be changed by powerSUITE GUI when using powerSUITE project).

All the other options can be left at default for now in the user_settings.h file.

```
#define CLLLC_LAB 1

#if CLLLC_LAB == 1
#define CLLLC_CONTROL_RUNNING_ON 1
#define CLLLC_POWER_FLOW CLLLC_POWER_FLOW_PRIM_SEC
#define CLLLC_INCR_BUILD CLLLC_OPEN_LOOP_BUILD
#define CLLLC_TEST_SETUP CLLLC_TEST_SETUP_RES_LOAD
#define CLLLC_PROTECTION CLLLC_PROTECTION_DISABLED
#define CLLLC_SFRA_TYPE 0
#define CLLLC_SFRA_AMPLITUDE (float32_t)CLLLC_SFRA_INJECTION_AMPLITUDE_LEVEL2
#endif
```

Load the variables for this lab into the watch window by running the *setupdebugenv_lab1and6.js* script.



11.3.1.1 **Lab1 - Check 1 (ISRs)**

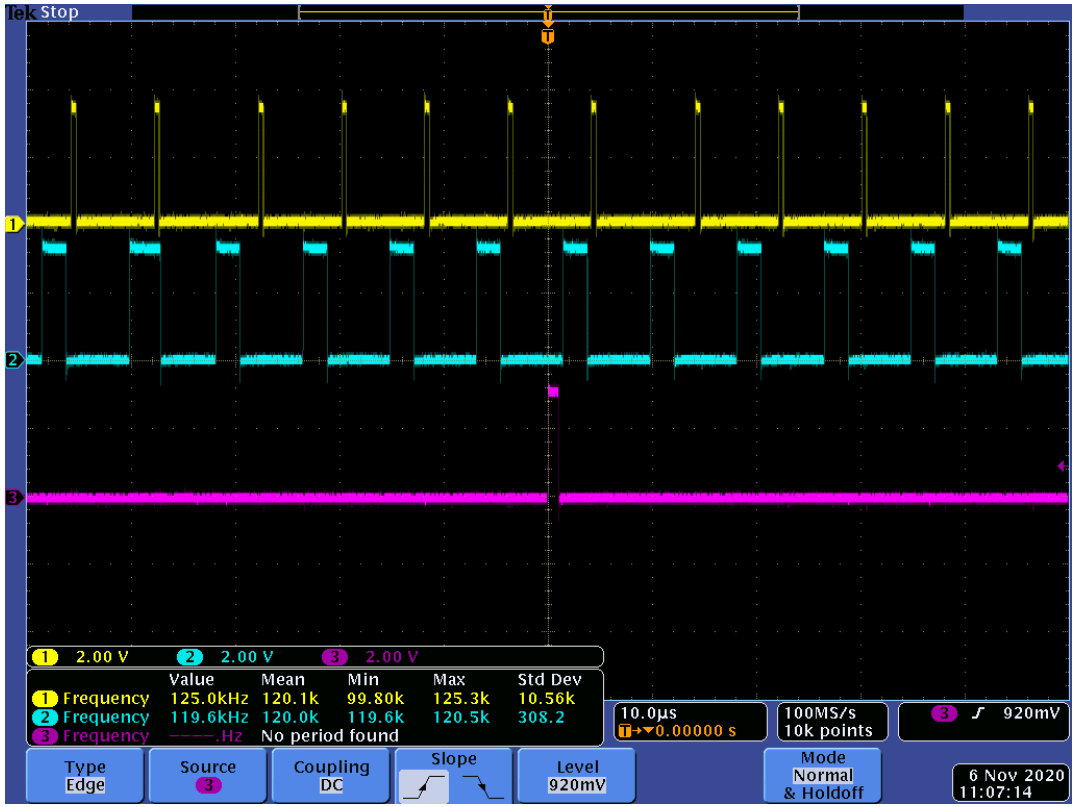
ISR Check

Here you will check to verify the frequency and nesting of the ISRs.

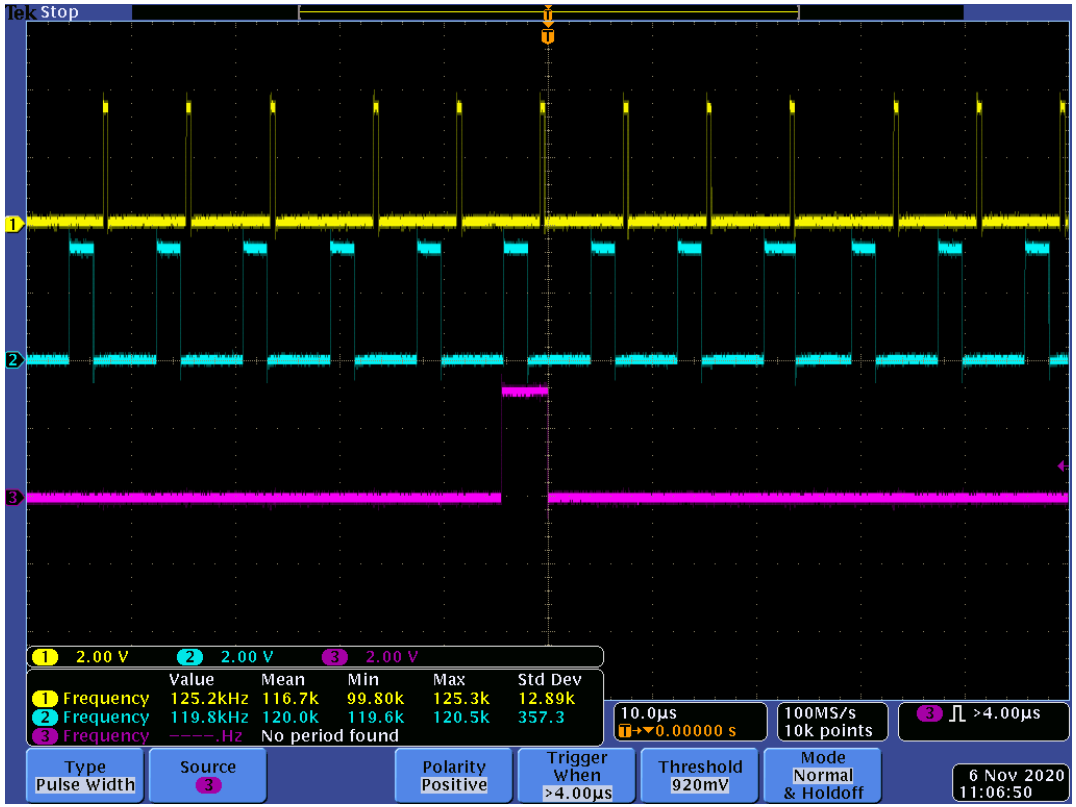
1. Probe GPIO-40 (ISR1), GPIO-45 (ISR2), GPIO-62 (ISR3) on the docking station.
    a. For F28388D ControlCARD this cooresponds to HSEC pins 89, 92, and 127.
2. Check that the frequency of the ISRs is correct.
    a. ISR1 = 120 kHz

   b. ISR2 = 120 kHz
   c. ISR3 = 10 kHz
3. Check the nesting behavior of the ISRs. To do this you will need to single trigger the scope a few times (or use a pulse width trigger on ISR3, pulse >4uS) until you can see the ISR1 / ISR2 interrupting during ISR3. The figure below shows such a capture.
   a. ISR1 should always happen after ISR2.
   b. ISR1 should be able to nest in ISR3.
   c. ISR2 should be able to nest in ISR3.
      i. ISR3 will take longer to execute when another ISR is nesting in it.

On some cycles there will be no ISR nesting


Tek Stop
1 2.00 V 2 2.00 V 3 2.00 V
Value Mean Min Max Std Dev
1 Frequency 125.0kHz 120.1k 99.80k 125.3k 10.56k
2 Frequency 119.6kHz 120.0k 119.6k 120.5k 308.2
3 Frequency ----.Hz No period found
10.0µs
T→▼0.00000 s
100MS/s
10k points
3 920mV
Type Edge
Source 3
Coupling DC
Slope
Level 920mV
Mode Normal & Holdoff
6 Nov 2020
11:07:14

On some cycles ISR1 and ISR2 may nest in ISR3


Tek Stop
1 2.00 V
2 2.00 V
3 2.00 V
Value Mean Min Max Std Dev
1 Frequency 125.2kHz 116.7k 99.80k 125.3k 12.89k
2 Frequency 119.8kHz 120.0k 119.6k 120.5k 357.3
3 Frequency ----.Hz No period found
10.0µs
T→▼0.00000 s
100MS/s
10k points
3 ⎍ >4.00µs
Type
Pulse Width
Source
3
Polarity
Positive
Trigger
When
>4.00µs
Threshold
920mV
Mode
Normal
& Holdoff
6 Nov 2020
11:06:50

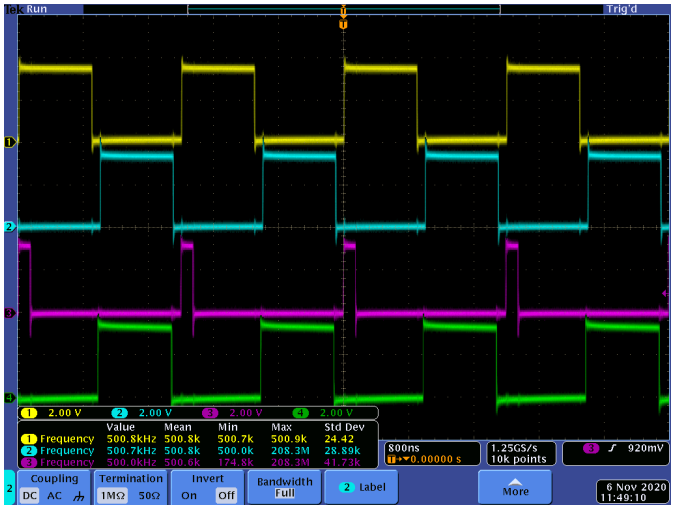
11.3.1.2 **Lab1 - Check 2 (ADCs)**

Verify measurement variables in the watch window to ensure the conversion is happening and the scale range is close to what is expected.

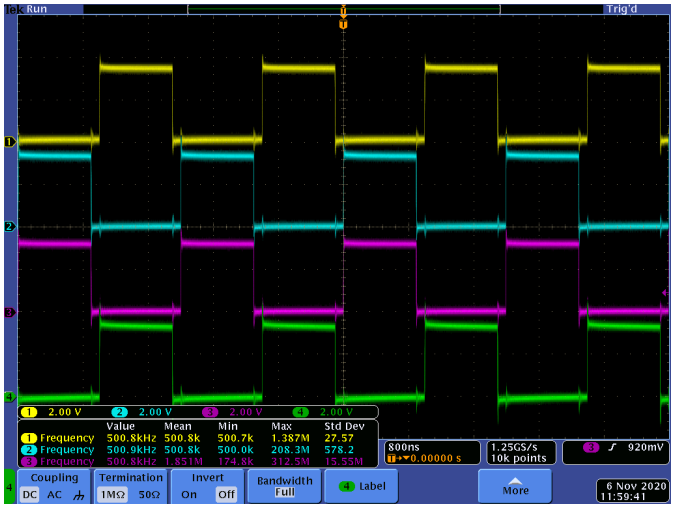
Note because the sampling happens as a very fast rate some times the circuitry used to drive the voltage on the pin may not have enough strength and absolute accuracy cannot be checked in this lab. Only the pin mapping, sw configuration, and the rough range.

		ADC Ch.	HSEC Pin	3.3V	1.65V	0V
CLLLC_iPrimSensed_Amps	CLLLC_IPRIM_MAX_SENSE_AMPS	D2	34	-33 A	0 A	33 A
CLLLC_iSecSensed_Amps	CLLLC_ISEC_MAX_SENSE_AMPS	A4	21	42 A	0 A	-42 A
CLLLC_vSecSensed_Volts	CLLLC_VSEC_MAX_SENSE_VOLTS	C2	31	482 V	240 V	0 V
CLLLC_vPrimSensed_Volts	CLLLC_VPRIM_MAX_SENSE_VOLTS	B3	20	508 V	254 V	0 V

11.3.1.3 **Lab1 - Check 3 (PWMs)**

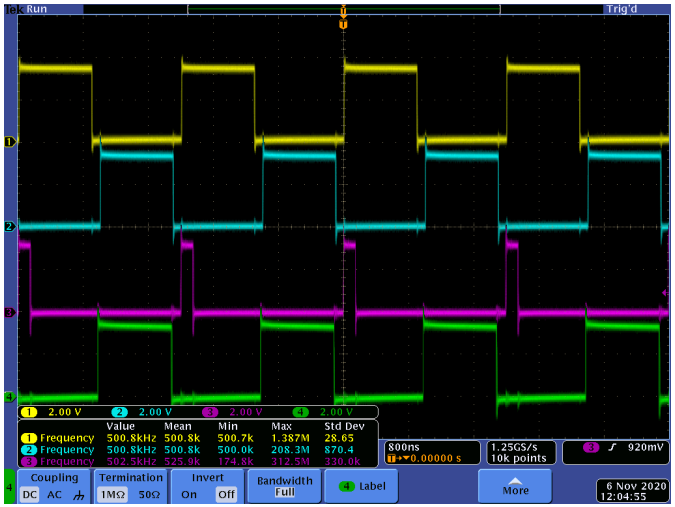
1	Initialization 1. Load code onto the controller and run. 2. Probe the following signals: a. PWM1A (GPIO-00) b. PWM2A (GPIO-02) c. PWM3A (GPIO-04) d. PWM4A (GPIO-06) e. On F28338D controlCARD these will be HSEC pins 49, 53, 50, 52 respectively. 3. Set the ***CLLLC_clearTrip*** variable to 1, this will clear the trip zone and allow the PWM signals to propagate to the pins. 4. At startup the switching frequency should be about 500 kHz. 5. The active SR signals will be assymetric because the voltage on the feedback pin is below 1.65V. 6. Probe the other PWM signals to make sure they look okay.	**PWMs at initialization** *Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM2A (Cyan), Ch.3 - PWM3A (Violet), Ch. 4 - PWM4A (Green)* *Comment By Yerra: Ch3 - is CLLLC_SEC_LEG2_H and Ch4 is CLLLC_SEC_LEG1_H in F28P65x design.*  **PWM1, 2**

Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM1B (Cyan), Ch.3 - PWM2A (Violet), Ch. 4 - PWM2B (Green)



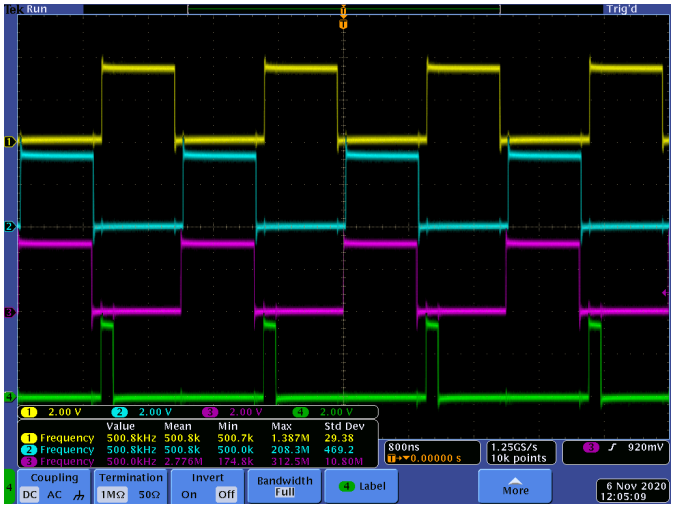
PWM3

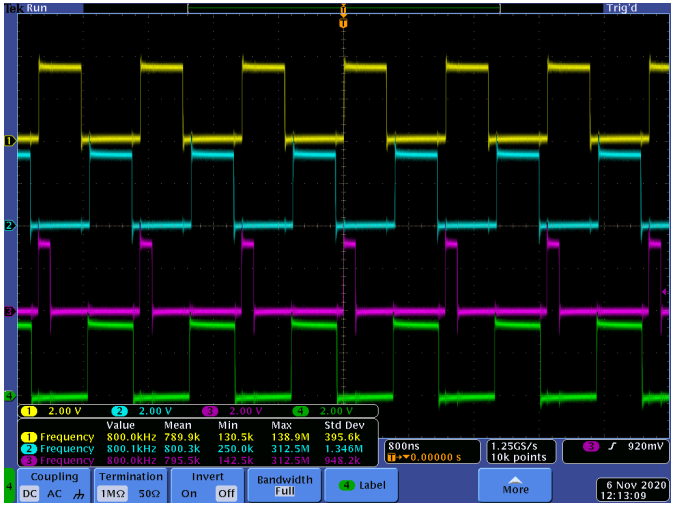
Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM2A (Cyan), Ch.3 - PWM3A (Violet), Ch. 4 - PWM3B (Green)

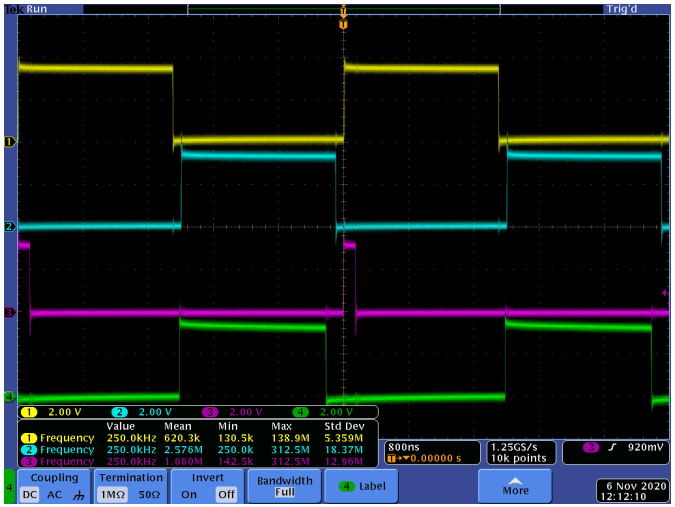


PWM4

Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM2A (Cyan), Ch.3 - PWM4A (Violet), Ch. 4 - PWM4B (Green)


Tek Run
Trig'd
2.00 V
2.00 V
2.00 V
2.00 V
Value Mean Min Max Std Dev
Frequency 500.8kHz 500.8k 500.7k 1.387M 29.38
Frequency 500.8kHz 500.8k 500.0k 208.3M 469.2
Frequency 500.0kHz 2.776M 174.8k 312.5M 10.80M
800ns
0.00000 s
1.25GS/s
10k points
920mV
Coupling
DC AC
Termination
1MΩ 50Ω
Invert
On Off
Bandwidth
Full
4 Label
More
6 Nov 2020
12:05:09

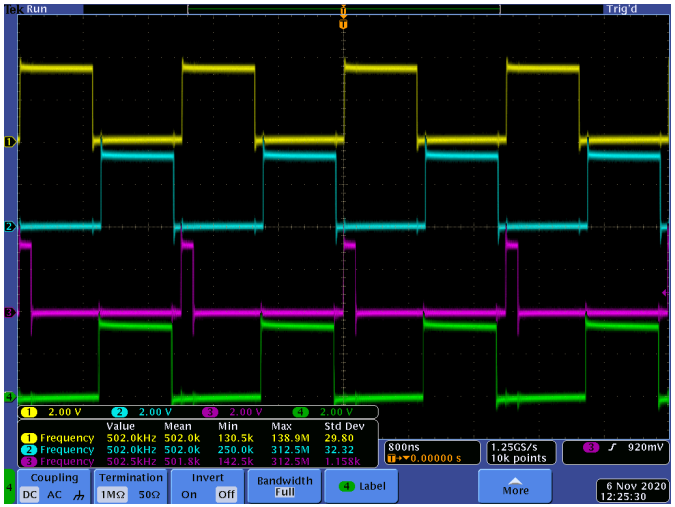
2	Frequency modulation 1. In the watch window use the ***CLLLC_pwmPeriodRef_pu*** variable to set the switching frequency. 2. Check the minimum and maximum switching frequency. a. ***CLLLC_pwmPeriodRef_pu*** = 0.3125 (800 kHz) b. ***CLLLC_pwmPeriodRef_pu*** = 1.0 (250 kHz) 3. Check high-resolution frequency. a. Set ***CLLLC_pwmPeriodRef_pu*** = 0.500 i. Observe the PWM frequency, should be 500 kHz. b. Set ***CLLLC_pwmPeriodRef_pu*** = 0.499, 0.498,0.497 i. The frequency should increase by about 1 kHz for each step up. ii. If high-res mode is not working, the frequency will have a resolution of about ~5 kHz.	**Max. switching frequency** *Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM2A (Cyan), Ch.3 - PWM3A (Violet), Ch. 4 - PWM4A (Green)*  **Min. switching frequency** *Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM2A (Cyan), Ch.3 - PWM3A (Violet), Ch. 4 - PWM4A (Green)*



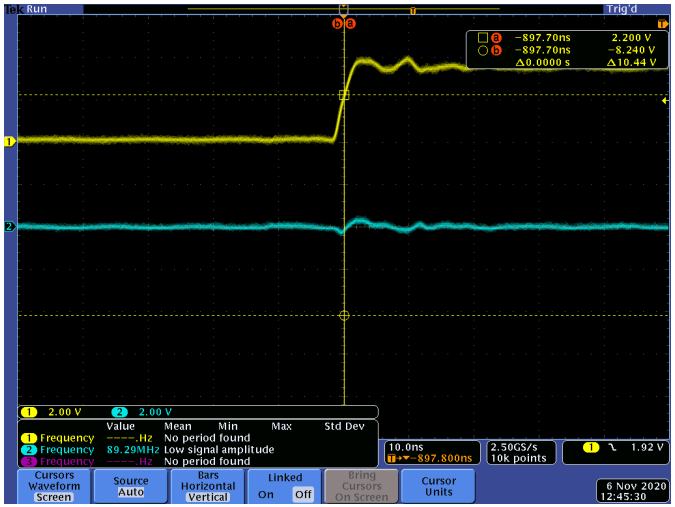
High-resolution frequency

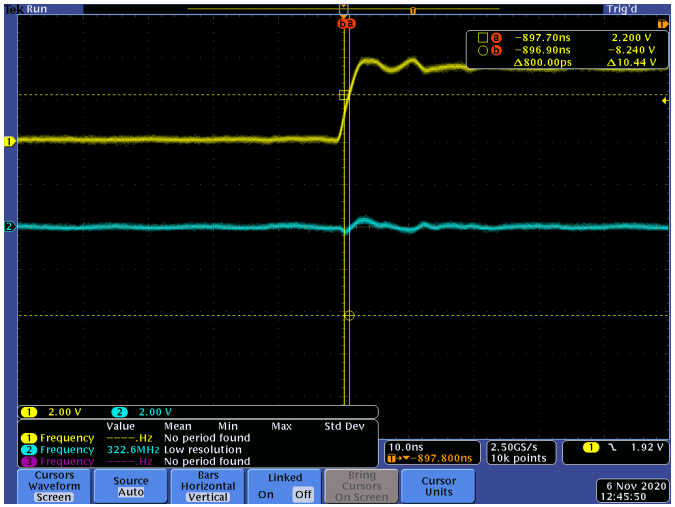
CLLLC_pwmPeriodRef_pu = 0.498**,** Fsw = 502 kHz

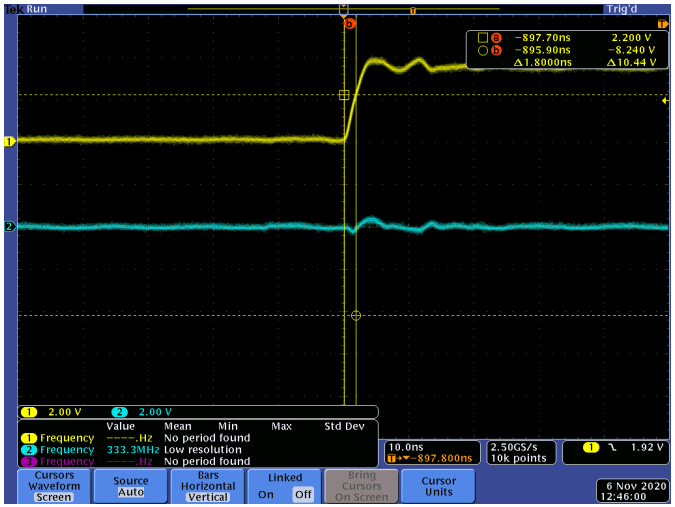
Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM2A (Cyan), Ch.3 - PWM3A (Violet), Ch. 4 - PWM4A (Green)

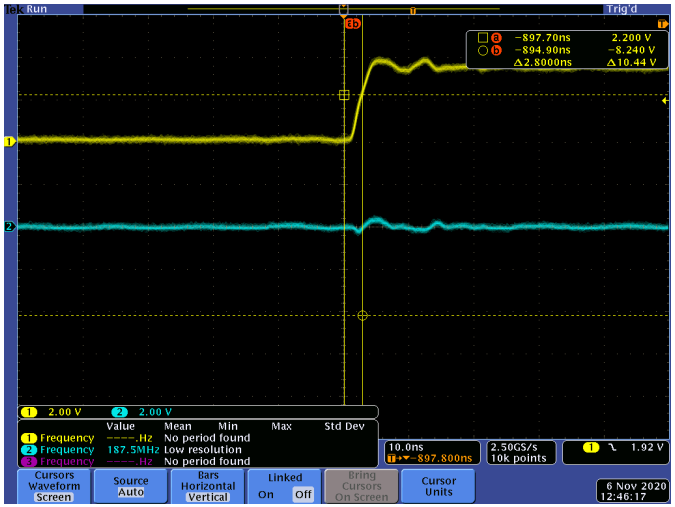

Tek Run
Trig'd
2.00 V
2.00 V
2.00 V
2.00 V
Value Mean Min Max Std Dev
Frequency 502.0kHz 502.0k 130.5k 138.9M 29.80
Frequency 502.0kHz 502.0k 250.0k 312.5M 32.32
Frequency 502.5kHz 501.8k 142.5k 312.5M 1.158k
800ns
0.00000 s
1.25GS/s
10k points
920mV
Coupling
DC AC
Termination
1MΩ 50Ω
Invert
On Off
Bandwidth
Full
Label
More
6 Nov 2020
12:25:30

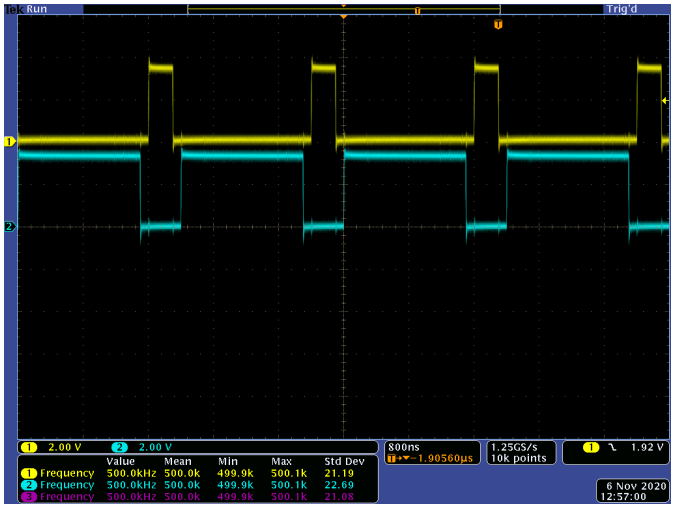
3	Primary legs dead-band 1. Probe PWM1A and PWM2A. 2. Verify that the high-resolution dead-band is working. a. The default value for the dead-band is 102.3ns. b. Adjust the ***CLLLC_pwmDeadBandREDPrimRef_ns*** value in 1ns steps and confirm the 1ns steps on the rising edge of the PWM1A signal. c. Adjust the ***CLLLC_pwmDeadBandFEDPrimRef_ns*** value in 1ns steps and confirm the 1ns steps on the rising edge of the PWM2A signal.	**High-resolution dead-band** 1. ***CLLLC_pwmDeadBandREDPrimRef_ns*** = 102.3ns 2. ***CLLLC_pwmDeadBandREDPrimRef_ns*** = 103ns 3. ***CLLLC_pwmDeadBandREDPrimRef_ns*** = 104ns 4. ***CLLLC_pwmDeadBandREDPrimRef_ns*** = 105ns *Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM2A (Cyan), Ch.3 - PWM3A (Violet), Ch. 4 - PWM4A (Green)*

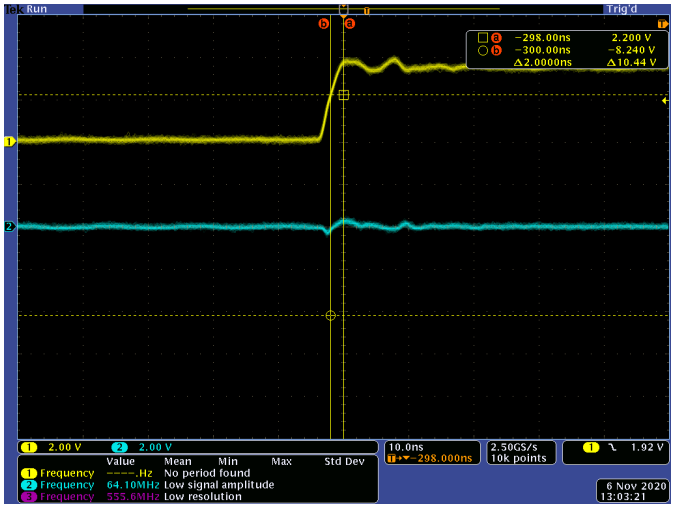

Tek Run
Trig'd
-897.70ns 2.200 V
-897.70ns -8.240 V
Δ0.0000 s Δ10.44 V
1 2.00 V 2 2.00 V
Value Mean Min Max Std Dev
1 Frequency ----.Hz No period found
2 Frequency 89.29MHz Low signal amplitude
3 Frequency ----.Hz No period found
10.0ns
-897.800ns
2.50GS/s
10k points
1.92 V
Cursors Waveform Screen
Source Auto
Bars Horizontal Vertical
Linked On Off
Bring Cursors On Screen
Cursor Units
6 Nov 2020
12:45:30


Tek Run
Trig'd
-897.70ns 2.200 V
-896.90ns -8.240 V
Δ800.00ps Δ10.44 V
1 2.00 V 2 2.00 V
Value Mean Min Max Std Dev
1 Frequency ----.Hz No period found
2 Frequency 322.6MHz Low resolution
3 Frequency ----.Hz No period found
10.0ns
-897.800ns
2.50GS/s
10k points
1.92 V
Cursors Waveform Screen
Source Auto
Bars Horizontal Vertical
Linked On Off
Bring Cursors On Screen
Cursor Units
6 Nov 2020
12:45:50


Tek Run
Trig'd
-897.70ns 2.200 V
-895.90ns -8.240 V
Δ1.8000ns Δ10.44 V
1 2.00 V 2 2.00 V
Value Mean Min Max Std Dev
1 Frequency ----.Hz No period found
2 Frequency 333.3MHz Low resolution
3 Frequency ----.Hz No period found
10.0ns
-897.800ns
2.50GS/s
10k points
1.92 V
Cursors Waveform Screen
Source Auto
Bars Horizontal Vertical
Linked On Off
Bring Cursors On Screen
Cursor Units
6 Nov 2020
12:46:00


Tek Run
Trig'd
-897.70ns 2.200 V
-894.90ns -8.240 V
Δ2.8000ns Δ10.44 V
1 2.00 V 2 2.00 V
Value Mean Min Max Std Dev
1 Frequency ----.Hz No period found
2 Frequency 187.5MHz Low resolution
3 Frequency ----.Hz No period found
10.0ns
-897.800ns
2.50GS/s
10k points
1.92 V
Cursors Waveform Screen
Source Auto
Bars Horizontal Vertical
Linked On Off
Bring Cursors On Screen
Cursor Units
6 Nov 2020
12:46:17

4	Primary legs duty-cycle 1. Probe PWM1A and PWM2A. 2. Verify that duty is updating in high-res. a. Set ***CLLLC_pwmPeriodRef_pu*** = 0.5 b. Set ***CLLLC_pwmDutyPrimRef_pu*** = 0.2 i. Observe that the PWMs update accordingly. c. Set ***CLLLC_pwmDutyPrimRef_pu*** = 0.201 i. Confirm that the duty changes by about 2ns.	**<u>High-resolution duty-cycle control</u>** *Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM2A (Cyan), Ch.3 - PWM3A (Violet), Ch. 4 - PWM4A (Green)* ***CLLLC_pwmDutyPrimRef_pu*** = 0.2  ***CLLLC_pwmDutyPrimRef_pu*** = 0.2 → ***CLLLC_pwmDutyPrimRef_pu*** = 0.201


Tek Run
Trig'd
-298.00ns 2.200 V
-300.00ns -8.240 V
Δ2.0000ns Δ10.44 V
2.00 V
2.00 V
10.0ns
-298.000ns
2.50GS/s
10k points
1.92 V
Value Mean Min Max Std Dev
Frequency -----.Hz No period found
Frequency 64.10MHz Low signal amplitude
Frequency 555.6MHz Low resolution
6 Nov 2020
13:03:21

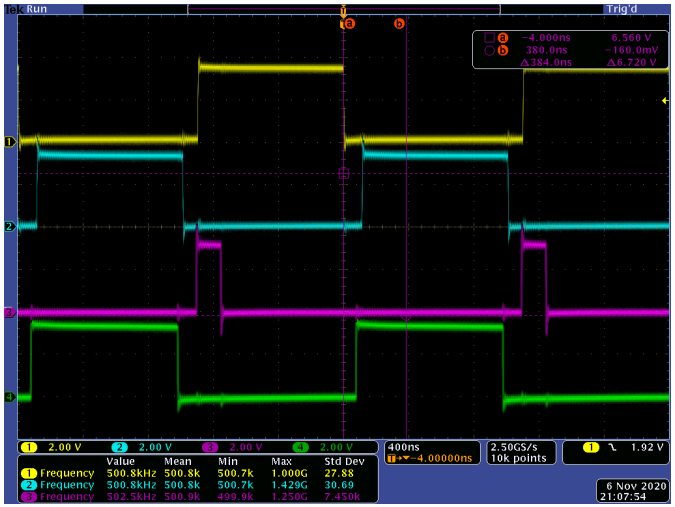
5	*Notes:* • *high-res primary phase shift is not working yet* • *there is a little glitch when going from frequency mode to phase shift mode and back to frequency mode* • *negative phase shifts are not supported* • *phase shifts greater than 0.5 are not supported**(due to new updates the phase shift limit is now smaller than 0.5 need to update documentation)*** Phase shift mode 1. Probe the following signals a. PWM1A b. PWM2A c. PWM3A d. PWM4A 2. Check values of ***CLLLC_pwmPhaseShiftPrimLegsRef_pu*** = 0, 0.1, 0.4, 0.5 a. Verify that the phase shift of the primary PWMs updates. b. Verify that as phase shift increases from 0 → 0.5 the falling edge of PWM2A is shifting toward the rising edge of PWM1A.	**Primary Legs Phase Shift** 1. ***CLLLC_pwmPhaseShiftPrimLegsRef_pu*** = 0.0 2. ***CLLLC_pwmPhaseShiftPrimLegsRef_pu*** = 0.1 3. ***CLLLC_pwmPhaseShiftPrimLegsRef_pu*** = 0.4 4. ***CLLLC_pwmPhaseShiftPrimLegsRef_pu*** = 0.5 *Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM2A (Cyan), Ch.3 - PWM3A (Violet), Ch. 4 - PWM4A (Green)*

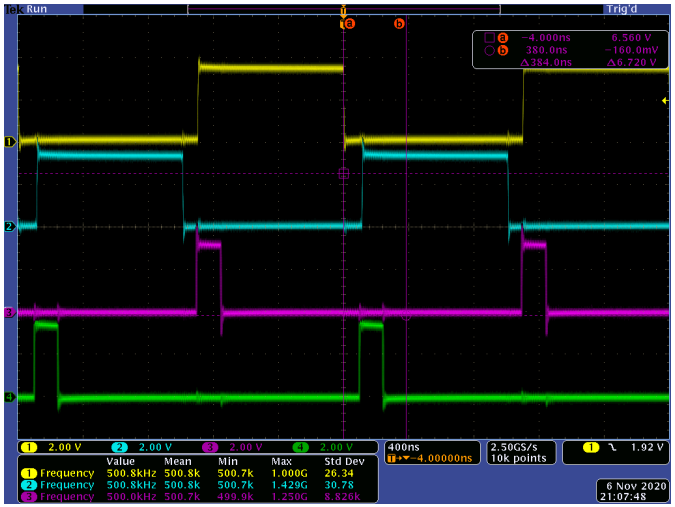
c. Verify that the SR PWMs (EPWM3A and EPWM4A) shift with PWM2A.
d. Verify that the SR PWMs don't get stuck high, or stuck low.
    i. This can happen if the count direction for EPWM3/4 isn't getting updated correctly.
e. Verify that as the phase shift increases the duty cycle of the SR PWMs is limited. (I.e. the duty will get smaller and eventually go to zero.


Tek Run
Trig'd
-4.000ns 6.560 V
380.0ns -160.0mV
Δ384.0ns Δ6.720 V
1 2.00 V 2 2.00 V 3 2.00 V 4 2.00 V
Value Mean Min Max Std Dev
1 Frequency 502.5kHz 501.8k 500.7k 1.000G 842.5
2 Frequency 502.5kHz 501.8k 500.7k 1.429G 842.5
3 Frequency 502.5kHz 501.8k 500.0k 1.250G 1.138k
400ns
-4.00000ns
2.50GS/s
10k points
1 1.92 V
Coupling
DC AC
Termination
1MΩ 50Ω
Invert
On Off
Bandwidth
Full
3 Label
More
6 Nov 2020
20:52:04

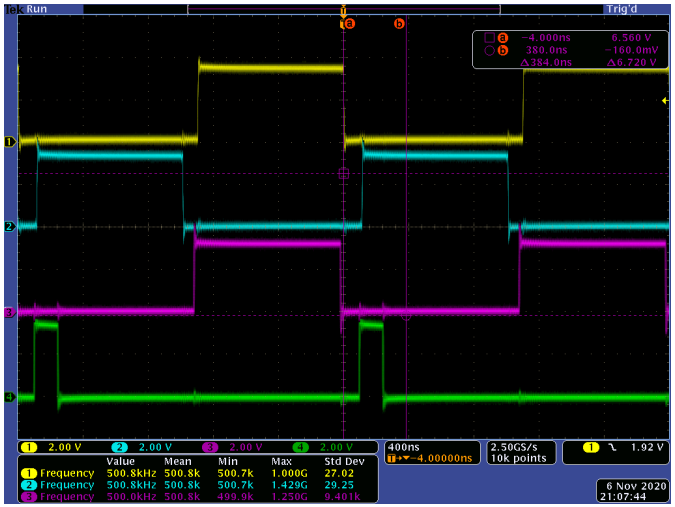

Tek Run
Trig'd
a -4.000ns 6.560 V
b 380.0ns -160.0mV
Δ384.0ns Δ6.720 V
1 2.00 V
2 2.00 V
3 2.00 V
4 2.00 V
Value Mean Min Max Std Dev
1 Frequency 502.5kHz 502.4k 500.7k 1.000G 416.4
2 Frequency 502.5kHz 502.4k 500.7k 1.429G 420.0
3 Frequency 502.5kHz 502.4k 500.0k 1.250G 523.2
400ns
-4.00000ns
2.50GS/s
10k points
1.92 V
Coupling
DC AC
Termination
1MΩ 50Ω
Invert
On Off
Bandwidth
Full
3 Label
More
6 Nov 2020
20:52:29


Tek Run
Trig'd
-4.000ns 6.560 V
380.0ns -160.0mV
Δ384.0ns Δ6.720 V
1 2.00 V 2 2.00 V 3 2.00 V 4 2.00 V
Value Mean Min Max Std Dev
1 Frequency 502.5kHz 502.5k 500.7k 1.000G 280.6
2 Frequency 502.5kHz 502.5k 500.7k 1.429G 283.3
3 Frequency 85.47MHz Low signal amplitude
400ns
-4.00000ns
2.50GS/s
10k points
1 1.92 V
Coupling
DC AC
Termination
1MΩ 50Ω
Invert
On Off
Bandwidth
Full
3 Label
More
6 Nov 2020
20:52:39

6	Active SR feedback 1. Probe the following signals a. PWM1A b. PWM2A c. PWM3A d. PWM4A 2. Connect a power supply to the CLLLC_ISEC_TANK pin (D0, HSEC 28) 3. Sweep the power supply from 0 → 3.3V and confirm that the duty of the SR PWMs behaves appropriately. a. 0 → 1.5V: PWM4A has max. duty b. 1.65V: PWM3A & PWM4A have min. duty determined by length of the blanking window. c. 1.8 → 3.3V: PWM3A has max. duty	**Active Synchronous Rectification Feedback** *Ch. 1 - PWM1A (Yellow), Ch. 2 - PWM2A (Cyan), Ch.3 - PWM3A (Violet), Ch. 4 - PWM4A (Green)* *CLLLC_ISEC_TANK (D0)* = 0 → 1.5V  *CLLLC_ISEC_TANK (D0)* = 1.65V



CLLLC_ISEC_TANK (D0) = 1.8 → 3.3V



11.3.1.4 **Lab1 - Check 4 (Board Protection)**

Find the CLLLC protection selection in the *clllc_user_settings.h* file.

```
#if CLLLC_PROTECTION == CLLLC_PROTECTION_ENABLED
//
// set 1 to enable the appropriate protection scheme
```

```
//
#define CLLLC_BOARD_PROTECTION_IPRIM 1
#define CLLLC_BOARD_PROTECTION_ISEC 1
#define CLLLC_BOARD_PROTECTION_VSEC 1
#define CLLLC_BOARD_PROTECTION_GANFAULT 1
#else
//
// set 0 to disable the appropriate protection scheme
//
#define CLLLC_BOARD_PROTECTION_IPRIM 0
#define CLLLC_BOARD_PROTECTION_ISEC 0
#define CLLLC_BOARD_PROTECTION_VSEC 0
#define CLLLC_BOARD_PROTECTION_GANFAULT 0
#endif
```

Check that the PWMs trip when the protection is enabled

1. Enable one protection at a time by writing 1 to the #define.
2. Probe the PWM signals (EPWM1A, EPWM2A, EPWM3A, EPWM4A)
3. Connect a power supply to the appropriate pin for the enabled protection.
    a. Ensure the power supply voltage is set to a value within the trip limits.
4. Rebuild and launch the code on the controlCARD.
5. Run the code.
6. Set ***CLLLC_clearTrip*** = 1 to enable the PWMs.
    a. The PWM signals should be visible on the scope.

   b. Check the ***CLLLC_tripFlag*** variable in the watch window and ensure it displays *no trip.*
7. Adjust the power supply voltage until the PWMs trip.
8. Check the ***CLLLC_tripFlag*** variable in the watch window and ensure it displays the correct trip value.
9. Connect each of the PWM pins to 3.3V with a 5k resistor and verify that the voltage on the pin remains forced low in the trip state.
   a. If the advanced trip zone configuration is not done correctly the PWMs will go into Hi-Z instead of force low, which can allow noise to couple onto the gate driver inputs and cause spurious switching events.

Protection	Pin	Trip thresholds	Trip Flag Value
CLLLC_BOARD_PROTECTION_IPRIM	D2 (HSEC 34)	< -30A > +30A	primOverCurrentTrip
CLLLC_BOARD_PROTECTION_ISEC	A4 (HSEC 21)	< -30A > +30A	secOverCurrentTrip
CLLLC_BOARD_PROTECTION_VSEC	C2 (HSEC 31)	> +450V	secOverVoltageTrip

Protection	Pin	Trip thresholds	Trip Flag Value
CLLLC_BOARD_PROTECTION_GANFAULT	GPIO-23 (HSEC74)	Logic low	ganFault

11.3.1.5 **Lab1 - Check 5 (GPIOs)**

Verify that the following GPIOs work as expected:

- Verify that GPIO14 (HSEC 62) may be set or cleared by writing the ***CLLLC_enableLC*** .
- Verify that ***CLLLC_iSecSenseDiag*** variable in the watch window follows voltage applied to GPIO30 (HSEC 80)

11.3.2 Lab 2

12 PFC+ CLLLC

12.1 Device Usage Information

MIPS

Memory

RAM, FLASH