Green trace – debug trace (pulses alternate in length for debug purpose)

* First pulse – enters ISR
* Second pulse – right before switch
* Third pulse – after turning one module off, and setting second to pwm
* Forth pulse – end of switch
* Fifth pulse – end of ISR

Note that whenever switch statement is reached while “pwm to off” pwm module is low I get a delay in new pwm settings.



Note that whenever switch statement is reached while “pwm to off” pwm modules is high, new pwm settings go into effect OK.



So I played around to see if there is a magic count when it is OK to introduce new pwm settings. When I reset the counter to one of the compare values things seem to work much better. I would say 99.9% of my commutations are clean, but once in a while and for some reason same commutation state still has this problem. Using debug gpio it looks like this problem pops up whenever counter is in one specific state.



Following is abbreviated code I use with debug pulses and counter reset fix…………

// Init Timer-Base Control Register for EPWM1-EPWM3

 EPwm1Regs.TBCTL.bit.FREE\_SOFT = SOFT\_STOP\_FLAG;

 EPwm1Regs.TBCTL.bit.PHSDIR = PHSDIR\_CNT\_UP;

 EPwm1Regs.TBCTL.bit.CLKDIV = CLKDIV\_PRESCALE\_X\_1;

 EPwm1Regs.TBCTL.bit.HSPCLKDIV = HSPCLKDIV\_PRESCALE\_X\_1;

 EPwm1Regs.TBCTL.bit.SYNCOSEL = 0; //send synch down stream

 EPwm1Regs.TBCTL.bit.PRDLD = 1; //0-shadow mode, 1-immidiate mode

 EPwm1Regs.TBCTL.bit.PHSEN = 1; //no sync, make this pwm master

 EPwm1Regs.TBCTL.bit.CTRMODE = TIMER\_CNT\_UP;

 EPwm2Regs.TBCTL.bit.FREE\_SOFT = SOFT\_STOP\_FLAG;

 EPwm2Regs.TBCTL.bit.PHSDIR = PHSDIR\_CNT\_UP;

 EPwm2Regs.TBCTL.bit.CLKDIV = CLKDIV\_PRESCALE\_X\_1;

 EPwm2Regs.TBCTL.bit.HSPCLKDIV = HSPCLKDIV\_PRESCALE\_X\_1;

 EPwm2Regs.TBCTL.bit.SYNCOSEL = 0; //sync to master

 EPwm2Regs.TBCTL.bit.PRDLD = 1; //0-shadow mode, 1-immidiate mode

 EPwm2Regs.TBCTL.bit.PHSEN = 1; //sync enable

 EPwm2Regs.TBCTL.bit.CTRMODE = TIMER\_CNT\_UP;

 EPwm3Regs.TBCTL.bit.FREE\_SOFT = SOFT\_STOP\_FLAG;

 EPwm3Regs.TBCTL.bit.PHSDIR = PHSDIR\_CNT\_UP;

 EPwm3Regs.TBCTL.bit.CLKDIV = CLKDIV\_PRESCALE\_X\_1;

 EPwm3Regs.TBCTL.bit.HSPCLKDIV = HSPCLKDIV\_PRESCALE\_X\_1;

 EPwm3Regs.TBCTL.bit.SYNCOSEL = 0; //sync to master

 EPwm3Regs.TBCTL.bit.PRDLD = 1; //0-shadow mode, 1-immidiate mode

 EPwm3Regs.TBCTL.bit.PHSEN = 1; //sync enable

 EPwm3Regs.TBCTL.bit.CTRMODE = TIMER\_CNT\_UP;

 //EPwm2Regs.TBCTL.all = EPwm1Regs.TBCTL.all;

 //EPwm3Regs.TBCTL.all = EPwm1Regs.TBCTL.all;

 // Init Compare Control Register for EPWM1-EPWM3

 EPwm1Regs.CMPCTL.bit.SHDWBMODE = 1; //0-shadow mode, 1-immidiate mode

 EPwm1Regs.CMPCTL.bit.SHDWAMODE = 1; //0-shadow mode, 1-immidiate mode

 EPwm1Regs.CMPCTL.bit.LOADBMODE = 0; //0-load on zero, 1-load on PRD

 EPwm1Regs.CMPCTL.bit.LOADAMODE = 0; //0-load on zero, 1-load on PRD

 EPwm2Regs.CMPCTL.all = EPwm1Regs.CMPCTL.all;

 EPwm3Regs.CMPCTL.all = EPwm1Regs.CMPCTL.all;

//===========================================================================

//

// Global: gMotorControlECapIsr

//

Void gMotorControlECapIsr(UArg capture\_number)

{

 **static** **int** i = 0;

 **static** int16 pwm\_period\_temp = 0;

 **asm**(" EALLOW");

 GpioCtrlRegs.GPBMUX2.bit.GPIO58 = 0; // 0=GPIO, 1=MCLKR-A, 2=XD21, 3=Resv

 GpioCtrlRegs.GPBDIR.bit.GPIO58 = 1; // 1=OUTput, 0=INput

 **asm**(" EDIS");

 GpioDataRegs.GPBCLEAR.bit.GPIO58 = 1; // uncomment if --> Set Low initially

 GpioDataRegs.GPBSET.bit.GPIO58 = 1; // uncomment if --> Set High initially

 //GpioDataRegs.GPBTOGGLE.bit.GPIO58 = 1;

 **for**(i = 0; i < 5; i++)

 {

 **asm**(" NOP");

 }

 GpioDataRegs.GPBTOGGLE.bit.GPIO58 = 1;

//-----------

Init Code….

//------------

 GpioDataRegs.GPBTOGGLE.bit.GPIO58 = 1;

 **for**(i = 0; i < 1; i++)

 {

 **asm**(" NOP");

 }

 GpioDataRegs.GPBTOGGLE.bit.GPIO58 = 1;

 pwm\_period\_temp = pwm\_shifted\_cmp\_a;

 **switch**(CommutationState)

 {

 **case** COMMU\_STATE\_0:

 {

 // Unused

 EPwm3Regs.AQCSFRC.bit.CSFA = 1; // Low

 EPwm3Regs.AQCSFRC.bit.CSFB = 1; // Low

 EPwm3Regs.DBCTL.bit.OUT\_MODE = BP\_DISABLE;

 **asm**(" EALLOW");

 SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;

 **asm**(" EDIS");

 EPwm1Regs.TBCTR = pwm\_period\_temp;

 EPwm2Regs.TBCTR = pwm\_period\_temp;

 EPwm3Regs.TBCTR = pwm\_period\_temp;

 **asm**(" EALLOW");

 SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;

 **asm**(" EDIS");

 // Power In

 EPwm1Regs.AQCSFRC.bit.CSFA = 0; // PWM 1a (Ah)

 EPwm1Regs.AQCTLA.bit.CAU = 2; // high when CTR = CMPA on UP

 EPwm1Regs.AQCTLA.bit.PRD = 1; // low when CTR = PRD

 EPwm1Regs.CMPA.half.CMPA = pwm\_duty\_ticks; // PWM duty cycle

 EPwm1Regs.AQCSFRC.bit.CSFB = 0; // Comp PWM with Deadband on 1B

 EPwm1Regs.DBCTL.bit.OUT\_MODE =BP\_ENABLE;//deadband(overides CSFB)

 GpioDataRegs.GPBTOGGLE.bit.GPIO58 = 1;

 **for**(i = 0; i < 5; i++)

 {

 **asm**(" NOP");

 }

 GpioDataRegs.GPBTOGGLE.bit.GPIO58 = 1;

 // Power Out

 EPwm2Regs.AQCSFRC.bit.CSFA = 0; // Shifted Complement PWM

 EPwm2Regs.AQCTLA.bit.CAU = 1; // Set Low CMPA on UP-count

 EPwm2Regs.AQCTLA.bit.CBU = 2; // Set high CPMB on UP-count

 EPwm2Regs.CMPA.half.CMPA = pwm\_shifted\_cmp\_a; // PWM duty cycle

 EPwm2Regs.CMPB = pwm\_shifted\_zero; // PWM duty cycle

 EPwm2Regs.AQCSFRC.bit.CSFB = 0;//1; // OverRidden by DeadBand

 EPwm2Regs.DBCTL.bit.OUT\_MODE =BP\_ENABLE;//deadband(overides CSFB)

 GpioDataRegs.GPBTOGGLE.bit.GPIO58 = 1;

 **for**(i = 0; i < 1; i++)

 {

 **asm**(" NOP");

 }

 GpioDataRegs.GPBTOGGLE.bit.GPIO58 = 1;

 **break**;

 }

} // switch

 // force pwm sync

 EPwm1Regs.TBCTL.bit.SWFSYNC = 1;

 // More code…

 GpioDataRegs.GPBTOGGLE.bit.GPIO58 = 1;

 **for**(i = 0; i < 5; i++)

 {

 **asm**(" NOP");

 }

 GpioDataRegs.GPBTOGGLE.bit.GPIO58 = 1;

}