**The first question is about the cmpss5h and cmpss6h operation highlighted in yellow and green：**

**Why is the configuration different during initialization and normal operation?**

void LLC\_HAL\_setupEPWM\_SYNC\_RAMP(uint32\_t base)

{

    EPWM\_setPeriodLoadMode(base,EPWM\_PERIOD\_DIRECT\_LOAD);

    EPWM\_setTimeBasePeriod(base, LLC\_EPWM1\_TBPRD);

    EPWM\_setPhaseShift(base, 0U);

    EPWM\_setTimeBaseCounter(base, 0U);

    EPWM\_setTimeBaseCounterMode(base, EPWM\_COUNTER\_MODE\_UP);

    EPWM\_enablePhaseShiftLoad(base);

    EPWM\_setCountModeAfterSync(base, EPWM\_COUNT\_MODE\_UP\_AFTER\_SYNC);

    EPWM\_setClockPrescaler(base,EPWM\_CLOCK\_DIVIDER\_1,EPWM\_HSCLOCK\_DIVIDER\_1);

    HRPWM\_setSyncPulseSource(base,HRPWM\_PWMSYNC\_SOURCE\_ZERO);

    //

    //TRIPIN7 = cmpss6h, select TRIPIN7

    //

    EPWM\_selectDigitalCompareTripInput(base,EPWM\_DC\_TRIP\_TRIPIN7,EPWM\_DC\_TYPE\_DCAL);

    //

    // DCAEVT1 = DCAL high(will become active as Comparator output goes high)

    //

    EPWM\_setTripZoneDigitalCompareEventCondition(base,EPWM\_TZ\_DC\_OUTPUT\_A1,EPWM\_TZ\_EVENT\_DCXL\_HIGH);

    EPWM\_setDigitalCompareEventSource(base, EPWM\_DC\_MODULE\_A,

                                       EPWM\_DC\_EVENT\_1,

                                       EPWM\_DC\_EVENT\_SOURCE\_ORIG\_SIGNAL);

    EPWM\_enableDigitalCompareSyncEvent(base,EPWM\_DC\_MODULE\_A);

}

void LLC\_HAL\_setupEPWMActiveHighComplementary(uint32\_t base)

{

    //

    // Set-up TBCLK

    //

#if LLC\_INCR\_BUILD == 1

    EPWM\_setPeriodLoadMode(base,EPWM\_PERIOD\_DIRECT\_LOAD);

#endif

    EPWM\_setTimeBasePeriod(base, LLC\_EPWM1\_TBPRD);

#if LLC\_INCR\_BUILD == 1

    EPWM\_setPhaseShift(base, 0U);

    EPWM\_setTimeBaseCounter(base, 0U);

    EPWM\_setTimeBaseCounterMode(base, EPWM\_COUNTER\_MODE\_UP);

    EPWM\_enablePhaseShiftLoad(base);

    EPWM\_setCountModeAfterSync(base, EPWM\_COUNT\_MODE\_UP\_AFTER\_SYNC);

    EPWM\_setClockPrescaler(base,EPWM\_CLOCK\_DIVIDER\_1,EPWM\_HSCLOCK\_DIVIDER\_1);

#endif

    //

    //select source for EPWM1SYNCPER

    //

    HRPWM\_setSyncPulseSource(base, HRPWM\_PWMSYNC\_SOURCE\_ZERO);

    //

    // setup counter compare submodule

    //

#if LLC\_INCR\_BUILD == 1

    EPWM\_setCounterCompareShadowLoadMode(base,EPWM\_COUNTER\_COMPARE\_A,EPWM\_COMP\_LOAD\_ON\_CNTR\_ZERO);

#endif

    EPWM\_setCounterCompareValue(base,EPWM\_COUNTER\_COMPARE\_B, LLC\_EPWM1\_TBPRD/2);

#if LLC\_INCR\_BUILD == 1

    //

    // setup dead band submodule

    // Use EPWMA as the input for both RED and FED

    //

    EPWM\_setDeadBandCounterClock(base, EPWM\_DB\_COUNTER\_CLOCK\_FULL\_CYCLE);

    EPWM\_setRisingEdgeDeadBandDelayInput(base, EPWM\_DB\_INPUT\_EPWMA);

    EPWM\_setFallingEdgeDeadBandDelayInput(base, EPWM\_DB\_INPUT\_EPWMA);

    //

    // Set the RED and FED values

    //

    EPWM\_setFallingEdgeDelayCount(base, LLC\_priFED\_10ns);

    EPWM\_setRisingEdgeDelayCount(base, LLC\_priRED\_10ns);

    //

    // Invert only the Falling Edge delay output (AHC)

    //

    EPWM\_setDeadBandDelayPolarity(base, EPWM\_DB\_RED, EPWM\_DB\_POLARITY\_ACTIVE\_HIGH);

    EPWM\_setDeadBandDelayPolarity(base, EPWM\_DB\_FED, EPWM\_DB\_POLARITY\_ACTIVE\_LOW);

    //

    // Use the delayed signals instead of the original signals

    //

    EPWM\_setDeadBandDelayMode(base, EPWM\_DB\_RED, true);

    EPWM\_setDeadBandDelayMode(base, EPWM\_DB\_FED, true);

    //

    // DO NOT Switch Output A with Output B

    //

    EPWM\_setDeadBandOutputSwapMode(base, EPWM\_DB\_OUTPUT\_A, false);

    EPWM\_setDeadBandOutputSwapMode(base, EPWM\_DB\_OUTPUT\_B, false);

#endif

    //

    //  Setup EPWM XBAR

    //  Link CMPSS events to EPWM modules

    //  Link CMPSS5 to EPWM TZ event

    //  Link CMPSS6 to EPWM TZ event

    //

    XBAR\_setEPWMMuxConfig(XBAR\_TRIP5, XBAR\_EPWM\_MUX08\_CMPSS5\_CTRIPH);

    XBAR\_enableEPWMMux(XBAR\_TRIP5, XBAR\_MUX08);

    XBAR\_setEPWMMuxConfig(XBAR\_TRIP7, XBAR\_EPWM\_MUX10\_CMPSS6\_CTRIPH);

    XBAR\_enableEPWMMux(XBAR\_TRIP7, XBAR\_MUX10);

    //

    // DCAH = TRIPIN5 = cmpss5h

    // DCAL = TRIPIN7 = cmpss6h

    //

    EPWM\_selectDigitalCompareTripInput(base,EPWM\_DC\_TRIP\_TRIPIN5,EPWM\_DC\_TYPE\_DCAH);

    EPWM\_selectDigitalCompareTripInput(base,EPWM\_DC\_TRIP\_TRIPIN7,EPWM\_DC\_TYPE\_DCAL);

    //

    // DCAEVT1 = DCAH high(will become active as Comparator output goes high)

    // DCAEVT2= DCAL high(will become active as Comparator output goes high)

    //

    EPWM\_setTripZoneDigitalCompareEventCondition(base,EPWM\_TZ\_DC\_OUTPUT\_A1,EPWM\_TZ\_EVENT\_DCXH\_HIGH);

    EPWM\_setTripZoneDigitalCompareEventCondition(base,EPWM\_TZ\_DC\_OUTPUT\_A2,EPWM\_TZ\_EVENT\_DCXL\_HIGH);

    //

    // max clamping debug DCAEVT2 = DCEVTFILT

    // DCAEVT2 = DCAEVT2 (not filtered)

    // DCAEVT1 = DCAEVT1 (not filtered)

    //

    EPWM\_setDigitalCompareEventSource(base, EPWM\_DC\_MODULE\_A,

                                      EPWM\_DC\_EVENT\_1,

                                      EPWM\_DC\_EVENT\_SOURCE\_ORIG\_SIGNAL);

    EPWM\_setDigitalCompareEventSource(base, EPWM\_DC\_MODULE\_A,

                                       EPWM\_DC\_EVENT\_2,

                                       EPWM\_DC\_EVENT\_SOURCE\_ORIG\_SIGNAL);

    //

    // DCAEVT1/2 event as AQ T1/2

    // T1 = DCAEVT1

    // T2 = DCAEVT2

    //

    EPWM\_setActionQualifierT1TriggerSource(base, EPWM\_AQ\_TRIGGER\_EVENT\_TRIG\_DCA\_1);

    EPWM\_setActionQualifierT2TriggerSource(base, EPWM\_AQ\_TRIGGER\_EVENT\_TRIG\_DCA\_2);

    //

    // This is to avoid impact of high impedance state

    //

    EPWM\_setTripZoneAction(base, EPWM\_TZ\_ACTION\_EVENT\_DCAEVT1, EPWM\_TZ\_ACTION\_DISABLE);

    EPWM\_setTripZoneAction(base, EPWM\_TZ\_ACTION\_EVENT\_DCAEVT2, EPWM\_TZ\_ACTION\_DISABLE);

    //

    // set T1/T2 action setting for EPWM1A

    //

    EPWM\_setActionQualifierAction(base,EPWM\_AQ\_OUTPUT\_A, EPWM\_AQ\_OUTPUT\_HIGH, EPWM\_AQ\_OUTPUT\_ON\_T1\_COUNT\_UP);

    EPWM\_setActionQualifierAction(base,EPWM\_AQ\_OUTPUT\_A, EPWM\_AQ\_OUTPUT\_LOW, EPWM\_AQ\_OUTPUT\_ON\_T2\_COUNT\_UP);

    //

    // if T2 is missing, CMPB = clamp period/2;

    //

    EPWM\_setActionQualifierAction(base,EPWM\_AQ\_OUTPUT\_A,EPWM\_AQ\_OUTPUT\_HIGH, EPWM\_AQ\_OUTPUT\_ON\_TIMEBASE\_ZERO);

    EPWM\_setActionQualifierAction(base,EPWM\_AQ\_OUTPUT\_A,EPWM\_AQ\_OUTPUT\_LOW, EPWM\_AQ\_OUTPUT\_ON\_TIMEBASE\_UP\_CMPB);

    EPWM\_enableDigitalCompareSyncEvent(base, EPWM\_DC\_MODULE\_A);

}

**The second question is about the phase operation highlighted in purple：**

**Is the phase operation just to accomplish the synchronization of EPWM1 and EPWM8 under certain conditions?**

**Or is there any other purpose?**

//

//control Code

//

inline void LLC\_ControlCode(void)

{

    LLC\_HAL\_setProfilingGPIO();

    //

    // Read epwm8 counter value

    //

    ASSERT(EPWM\_isBaseValid(LLC\_HHC\_CTRL\_PWM\_BASE));

    LLC\_epwm8counter = HWREGH(LLC\_HHC\_CTRL\_PWM\_BASE + EPWM\_O\_TBCTR);

    //

    // normal mode

    //

    if (LLC\_epwm8counter < LLC\_minFreqClamping\_theshld2 && LLC\_epwm8counter > LLC\_minFreqClamping\_theshld1)

    {

        EPWM\_setActionQualifierAction(LLC\_PRI\_PWM\_BASE,EPWM\_AQ\_OUTPUT\_A, EPWM\_AQ\_OUTPUT\_HIGH, EPWM\_AQ\_OUTPUT\_ON\_T1\_COUNT\_UP);

        EPWM\_setActionQualifierAction(LLC\_PRI\_PWM\_BASE,EPWM\_AQ\_OUTPUT\_A, EPWM\_AQ\_OUTPUT\_LOW, EPWM\_AQ\_OUTPUT\_ON\_T2\_COUNT\_UP);

        EPWM\_enablePhaseShiftLoad(LLC\_PRI\_PWM\_BASE);

        EPWM\_enableDigitalCompareSyncEvent(LLC\_HHC\_CTRL\_PWM\_BASE, EPWM\_DC\_MODULE\_A);

        #ifdef BUILD\_F28004X

        EPWM\_setSyncOutPulseMode(LLC\_PRI\_PWM\_BASE, EPWM\_SYNC\_OUT\_PULSE\_DISABLED);

        #endif

        #ifdef BUILD\_F28003X

        EPWM\_disableSyncOutPulseSource(LLC\_PRI\_PWM\_BASE, EPWM\_SYNC\_OUT\_PULSE\_ON\_ALL);

        #endif

        EPWM\_setPhaseShift(LLC\_HHC\_CTRL\_PWM\_BASE, 0);

        LLC\_mode = 2;

    }

    //

    // min freq clamping mode

    //

    else if (LLC\_epwm8counter > LLC\_minFreqClamping\_theshld3)

    {

        EPWM\_disablePhaseShiftLoad(EPWM1\_BASE);

        EPWM\_disableDigitalCompareSyncEvent(LLC\_HHC\_CTRL\_PWM\_BASE, EPWM\_DC\_MODULE\_A);

        #ifdef BUILD\_F28004X

        EPWM\_setSyncOutPulseMode(LLC\_PRI\_PWM\_BASE, EPWM\_SYNC\_OUT\_PULSE\_ON\_COUNTER\_ZERO);

        SysCtl\_setSyncInputConfig(SYSCTL\_SYNC\_IN\_EPWM7, SYSCTL\_SYNC\_IN\_SRC\_EPWM1SYNCOUT);

        #endif

        #ifdef BUILD\_F28003X

        EPWM\_enableSyncOutPulseSource(LLC\_PRI\_PWM\_BASE, EPWM\_SYNC\_OUT\_PULSE\_ON\_CNTR\_ZERO);

        EPWM\_setSyncInPulseSource(LLC\_HHC\_CTRL\_PWM\_BASE, EPWM\_SYNC\_IN\_PULSE\_SRC\_SYNCOUT\_EPWM1);

        #endif

        EPWM\_setPhaseShift(LLC\_HHC\_CTRL\_PWM\_BASE, LLC\_controlpwm\_phaseshift);

        EPWM\_setActionQualifierAction(LLC\_PRI\_PWM\_BASE,EPWM\_AQ\_OUTPUT\_A, EPWM\_AQ\_OUTPUT\_NO\_CHANGE, EPWM\_AQ\_OUTPUT\_ON\_T1\_COUNT\_UP);

        EPWM\_setActionQualifierAction(LLC\_PRI\_PWM\_BASE,EPWM\_AQ\_OUTPUT\_A, EPWM\_AQ\_OUTPUT\_NO\_CHANGE, EPWM\_AQ\_OUTPUT\_ON\_T2\_COUNT\_UP);

        LLC\_mode = 3;

    }

    CMPSS\_clearFilterLatchHigh(LLC\_HHC\_CH1\_CMPSS\_BASE);

    EPWM\_clearTripZoneFlag(LLC\_HHC\_CTRL\_PWM\_BASE, EPWM\_TZ\_FLAG\_DCAEVT1);

    //

    // EPwm1Regs.TZCLR.INT = 1;

    //

    ASSERT(EPWM\_isBaseValid(LLC\_HHC\_CTRL\_PWM\_BASE));

    EALLOW;

    //

    // clear INT bit of TZCLR register

    //

    HWREGH(LLC\_HHC\_CTRL\_PWM\_BASE + EPWM\_O\_TZCLR) |= EPWM\_TZCLR\_INT;

    EDIS;

    LLC\_HAL\_resetProfilingGPIO();

    ADC\_clearInterruptStatus(LLC\_VBUS\_ADC\_MODULE, ADC\_INT\_NUMBER1);