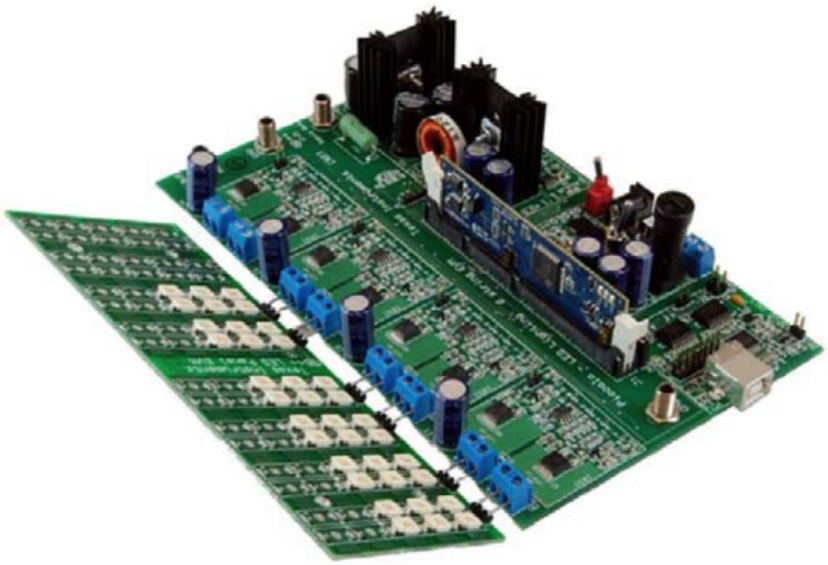
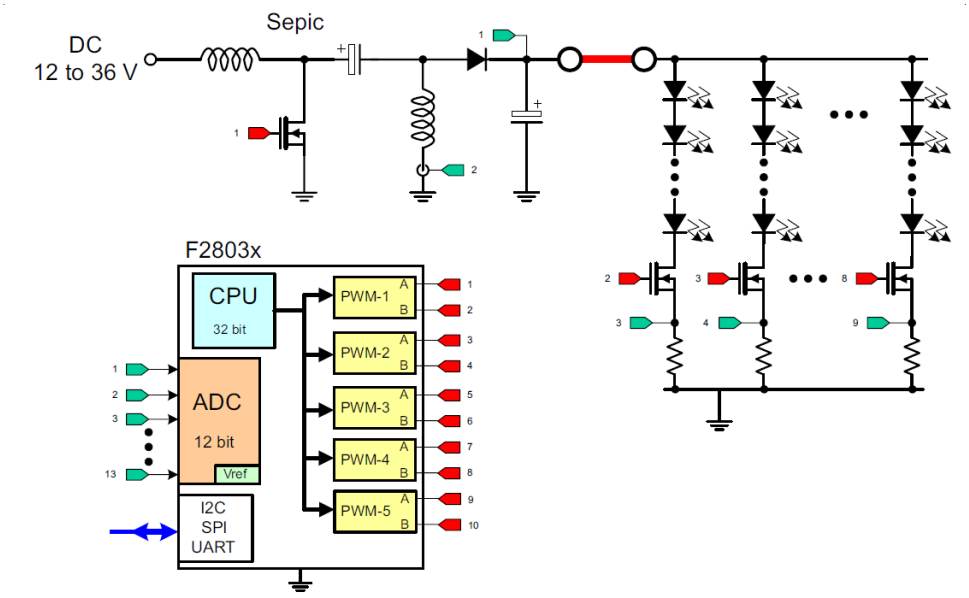
TMDSDCDCLEDKIT



- DC/DC SEPIC power stage;
- C2000 Piccolo TMS320F28035 Microcontroller with **CPU frequency equal to 60MHz**;
- 4 LED dimming stages PWM each managing a pair of LED strings for a total of 8 strings (see Figure below)

DC-DC LED Lighting Hardware Block Diagram



SEPIC MOSFET (red label 1) opens and closes with a frequency that is the **switching frequency**= 100kHz.

LED strings MOSFETs (red labels 2..8) switch at a frequency that is the **dimming frequency=**20kHz.

SETUP $f_{switching}$ e $f_{dimming}$ in the CODE

```
//---------------------------------------------------------------------
#if (INCR_BUILD == 3)   // Closed Loop SEPIC + Closed Loop LED Strings
//---------------------------------------------------------------------
    #if (DSP2802x_DEVICE_H || DSP2803x_DEVICE_H)
        #define     prd_sepic       600     // Period count = 100 KHz @ 60 MHz
        #define     prd_led         3000    // Period count = 20 KHz @ 60 MHz
    #endif
```

- ✓ $f_{sw} = \frac{f_{cpu}}{\#ticks} = \frac{60M}{600} = 100kHz$
- ✓ $f_{dim} = \frac{f_{cpu}}{\#ticks} = \frac{60M}{3000} = 20kHz$

PWM MODULES CONFIGURATION

There are 5 PWM modules whose configuration is made by two distinct functions.

The function which generates the **ePWM1A** signal to regulate the SEPIC is BuckSingle_CNF. This function takes in input the period, the mode (1 for MASTER and 0 for SLAVE) and the phase.

```
    BuckSingle_CNF(1, prd_sepic, 1, 0);     // ePWM1A    Period=prd, Master, Phase=Don't Care
```

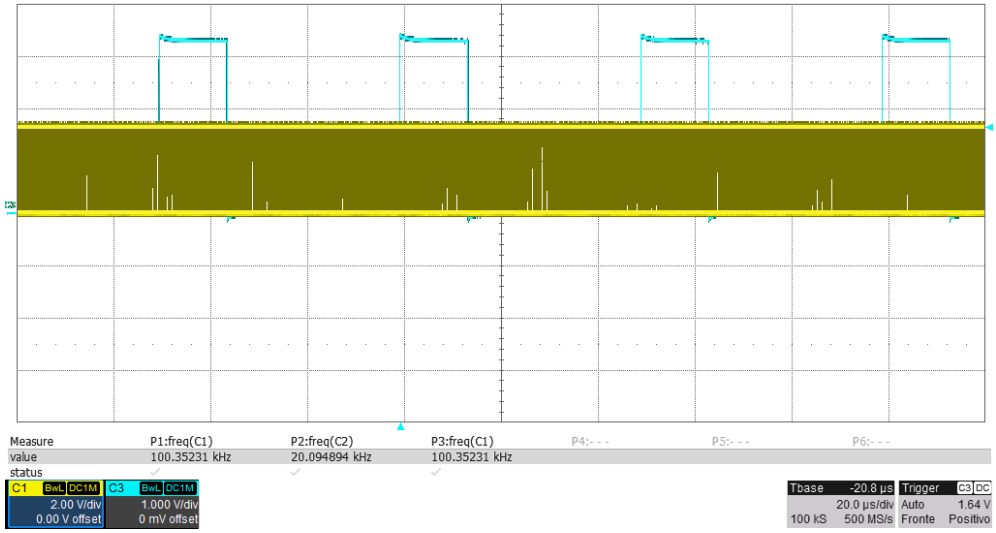
The other function BuckDualCNF configures the 4 modules that control the strings by generating the signals **ePWMxA, B**. The LED strings aren't turned on and off together but they are out of phase by ¼ of the dimming period. The first PWM module is the MASTER and so the first to be activated. The SLAVE modules are synchronized with it.

```
    // Spread LED power draw over full 50us (1/20kHz);
    BuckDual_CNF(2, prd_led, 1, 0);         // ePWM2A,B, Period=prd, Master,  Phase =  0degrees
    BuckDual_CNF(3, prd_led, 0, 750);       // ePWM3A,B, Period=prd, Slave,   Phase =  90degrees
    BuckDual_CNF(4, prd_led, 0, 1500);      // ePWM4A,B, Period=prd, Slave,   Phase = 180degrees
    BuckDual_CNF(5, prd_led, 0, 2250);      // ePWM5A,B, Period=prd, Slave,   Phase = 270degrees
```

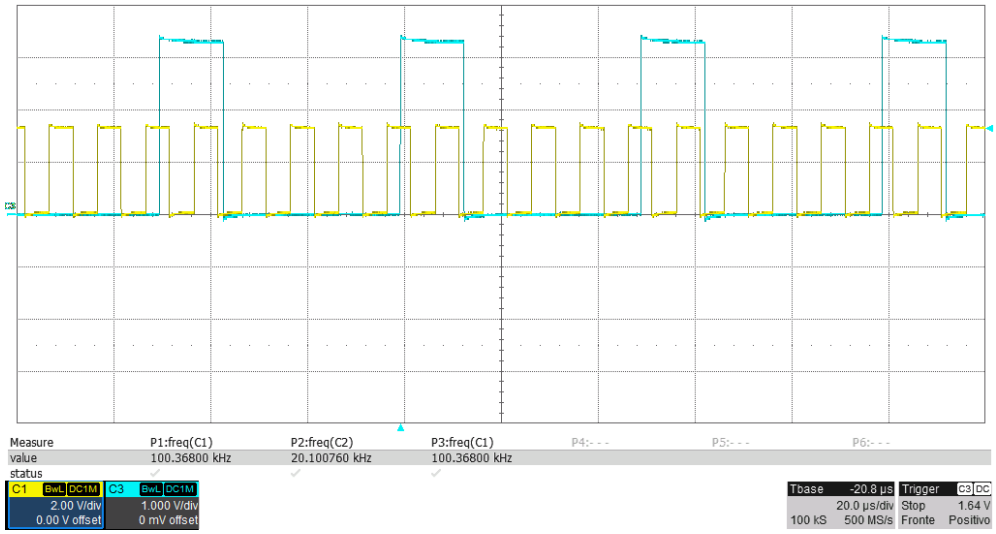
We expect that the ePWM1A (SEPIC) and ePWM2A or ePWM2B signals (MASTER module that manages the first two strings of LEDs) are synchronized. This does not happen, just look at the screenshots of the oscilloscope below.

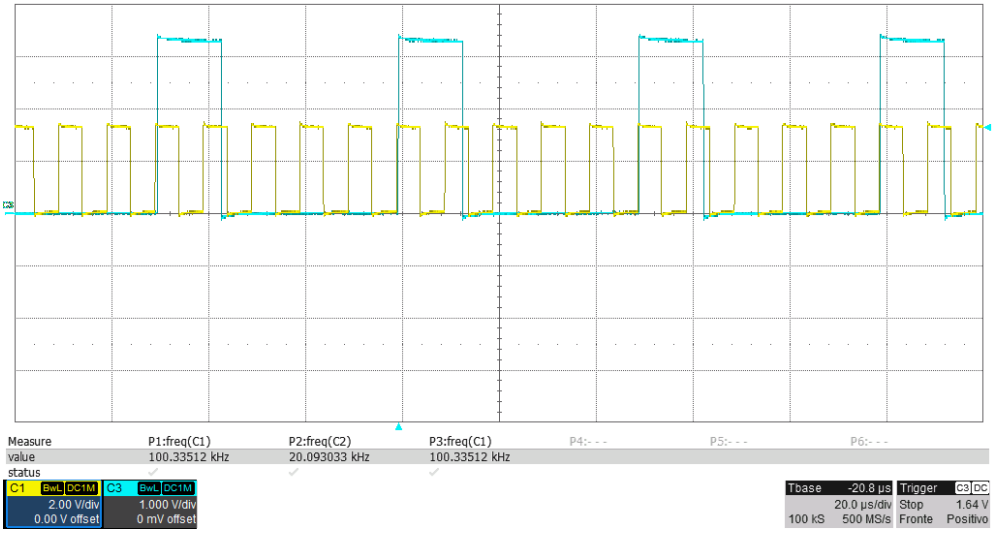
Blu signal: ePWM2A output (1st LED string)
Yellow signal: ePWM1A output (SEPIC)

The image was captured with the oscilloscope in **AUTO MODE**.



The following images were captured with the oscilloscope in **STOP MODE** in two different moments.





For completeness, we report the functions that deal with the configuration of the PWM modules and the code of the interruption in which the control laws are calculated and the ePWM signals are generated.

The interruption occurs at each switching frequency, such as every 100 kHz (10µs). At each interruption, only one of the ePWMxA, B signals relating to the LED strings is generated. In contrast, the ePWM1A signal for SEPIC regulation is always generated every 10µs.

BuckSingle_CNF function

```
void BuckSingle_CNF(int16 n, int16 period, int16 mode, int16 phase)
{
    (*ePWM[n]).TBCTL.bit.PRDLD = TB_IMMEDIATE;  // set Immediate load
    (*ePWM[n]).TBPRD = period;                  // PWM frequency = 1 / period
    (*ePWM[n]).TBPHS.half.TBPHS = 0;
    (*ePWM[n]).CMPA.half.CMPA = 0;              // set duty 0% initially
    (*ePWM[n]).CMPB = 0;                        // set duty 0% initially
    (*ePWM[n]).TBCTR = 0;
    (*ePWM[n]).TBCTL.bit.CTRMODE = TB_COUNT_UP;
    (*ePWM[n]).TBCTL.bit.HSPCLKDIV = TB_DIV1;
    (*ePWM[n]).TBCTL.bit.CLKDIV = TB_DIV1;

    if(mode == 1) // config as a Master
    {
        (*ePWM[n]).TBCTL.bit.PHSEN = TB_DISABLE;
        (*ePWM[n]).TBCTL.bit.SYNCOSEL = TB_CTR_ZERO; // sync "down-stream"
    }

    if(mode == 0) // config as a Slave (Note: Phase+2 value used to compensate for logic
    {
        (*ePWM[n]).TBCTL.bit.PHSEN = TB_ENABLE;
        (*ePWM[n]).TBCTL.bit.SYNCOSEL = TB_SYNC_IN;

        if( 0 <= phase <= 2)    (*ePWM[n]).TBPHS.half.TBPHS = (2-phase);
        if( phase > 2 )         (*ePWM[n]).TBPHS.half.TBPHS = (period-phase+2);
    }

    (*ePWM[n]).CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    (*ePWM[n]).CMPCTL.bit.LOADAMODE = CC_CTR_PRD;

    (*ePWM[n]).AQCTLA.bit.ZRO = AQ_SET;
    (*ePWM[n]).AQCTLA.bit.CAU = AQ_CLEAR;

    (*ePWM[n]).AQCTLB.bit.ZRO = AQ_NO_ACTION;
    (*ePWM[n]).AQCTLB.bit.CAU = AQ_NO_ACTION;
    (*ePWM[n]).AQCTLB.bit.PRD = AQ_NO_ACTION;
```

```
/  Enable HiRes option
   EALLOW;
   (*ePWM[n]).HRCNFG.all = 0x0;
   (*ePWM[n]).HRCNFG.bit.EDGMODE = HR_FEP;
   (*ePWM[n]).HRCNFG.bit.CTLMODE = HR_CMP;
   (*ePWM[n]).HRCNFG.bit.HRLOAD = HR_CTR_PRD;
   //(*ePWM[n]).HRCNFG.bit.HRLOAD = HR_CTR_ZERO;
   EDIS;
```

BuckDual_CNF function

```
void BuckDual_CNF(int16 n, int16 period, int16 mode, int16 phase)
{
    (*ePWM[n]).TBCTL.bit.PRDLD = TB_IMMEDIATE;  // set Immediate load
    (*ePWM[n]).TBPRD = period;                  // PWM frequency = 1 / period
    (*ePWM[n]).TBPHS.half.TBPHS = 0;
    (*ePWM[n]).CMPA.half.CMPA = 0;              // set duty 0% initially
    (*ePWM[n]).CMPB = 0;                        // set duty 0% initially
    (*ePWM[n]).TBCTR = 0;
    (*ePWM[n]).TBCTL.bit.CTRMODE = TB_COUNT_UP;
    (*ePWM[n]).TBCTL.bit.HSPCLKDIV = TB_DIV1;
    (*ePWM[n]).TBCTL.bit.CLKDIV = TB_DIV1;

    if(mode == 1) // config as a Master
    {
        (*ePWM[n]).TBCTL.bit.PHSEN = TB_DISABLE;
        (*ePWM[n]).TBCTL.bit.SYNCOSEL = TB_CTR_ZERO; // sync "down-stream"
    }

    if(mode == 0) // config as a Slave
    {
        (*ePWM[n]).TBCTL.bit.PHSEN = TB_ENABLE;
        (*ePWM[n]).TBCTL.bit.SYNCOSEL = TB_SYNC_IN;
        (*ePWM[n]).TBPHS.half.TBPHS = (period - phase); // set phase = 360 - lag
    }

    (*ePWM[n]).CMPCTL.bit.LOADAMODE = CC_CTR_PRD;
    (*ePWM[n]).CMPCTL.bit.LOADBMODE = CC_CTR_PRD;
    (*ePWM[n]).CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    (*ePWM[n]).CMPCTL.bit.SHDWBMODE = CC_SHADOW;

    (*ePWM[n]).AQCTLA.bit.ZRO = AQ_SET;
    (*ePWM[n]).AQCTLA.bit.CAU = AQ_CLEAR;
    (*ePWM[n]).AQCTLB.bit.ZRO = AQ_SET;
    (*ePWM[n]).AQCTLB.bit.CBU = AQ_CLEAR;
}
```

INTERRUPTION TO MANAGE CONTROL ACTIONS

```
    .if(INCR_BUILD = 3)
        ;ISR will always run the Sepic controller, but will only run two LED string controllers
        ;  each time an interrupt fires
        ADC_NchDRV      12      ; Convert 12 Adc Channels, (i.e. N=12)
        Control_3P3Z 1      ; Perform calculations for SEPIC controller
        BuckSingleHR_DRV 1      ; update EPWM1A duty cycle
;LED control is "time-sliced" to save cycles. This can be done because the LED
        ;  control does not need to run at 100kHz
        MOVW    DP, #tsPtr
        CMP     @tsPtr, #1
        B       TS1, EQ
        CMP     @tsPtr, #2
        B       TS2, EQ
        CMP     @tsPtr, #3
        B       TS3, EQ
        CMP     @tsPtr, #4
        B       TS4, EQ
        CMP     @tsPtr, #5
        B       TS5, GEQ

        MOV     @tsPtr,#1
        LB      TS_SKIP
TS1:
        ControlLaw_2P2Z 2
        ControlLaw_2P2Z 3
        BuckDual_DRV    2       ; EPWM2A, 2B
        LB      TS_SKIP
TS2:
        ControlLaw_2P2Z 4
        ControlLaw_2P2Z 5
        BuckDual_DRV    3       ; EPWM3A, 3B
        LB      TS_SKIP


TS3:
       ControlLaw_2P2Z 6
       ControlLaw_2P2Z 7
       BuckDual_DRV    4       ; EPWM4A, 4B
       LB      TS_SKIP
TS4:
       ControlLaw_2P2Z 8
       ControlLaw_2P2Z 9
       BuckDual_DRV    5       ; EPWM5A, 5B
       LB      TS_SKIP
TS5:    ; DO NOTHING

TS_SKIP:
       MOVW    DP, #tsPtr
       INC     @tsPtr
       CMP     @tsPtr, #5      ; if new value of tsPtr is 6 or greater reset tsPtr to 1
       B       TS_FIN, LEQ
       MOV     @tsPtr, #1

TS_FIN:



;ADC_Reset:
;       NOP
;       MOVW    DP,#ADCTRL2>>6      ; Reset ADC SEQ
;       MOV     @ADCTRL2,#0x4101    ; RST_SEQ1=1, SOCA-SEQ1=1, SOCB-SEQ2=1
    .endif
;   ;-----------------------------------------------------------
```