Task 2: Motor Current Sensing with Synchronized PWM Outputs

Hotzenblitz EV Re-Engineering Project

Mohamed Ismail

December 19, 2025

1 Objective

The objective of Task 2 is to extend Task 1 by generating an additional high-frequency PWM signal while maintaining deterministic ADC sampling. Two PWM modules are used:

- ePWM1 operating at 1 kHz to trigger ADC sampling
- ePWM2 operating at 10 kHz with a 50% duty cycle

Both PWM signals are started simultaneously to ensure perfect synchronization and are verified using an oscilloscope.

2 Hardware and Software Setup

2.1 Hardware

- MCU: TI TMS320F280039C
- Development Board: LAUNCHXL-F280039C
- ADC Inputs:
  - ADCINA2 – Sensor 1
  - ADCINA14 – Sensor 2
- PWM Outputs:
  - GPIO0 → ePWM1A (1 kHz ADC timing signal)
  - GPIO2 → ePWM2A (10 kHz PWM output)

2.2 Software

- Code Composer Studio (CCS)
- C2000Ware DriverLib

3 System Working Principle

- ePWM1 runs at 1 kHz and generates ADC Start-of-Conversion triggers.
- ADC samples two current sensors every 1 ms.
- ePWM2 runs continuously at 10 kHz with a 50% duty cycle.
- Both PWMs are started simultaneously using the TBCLKSYNC mechanism.

This design guarantees deterministic sampling and synchronized PWM generation.

4 PWM Synchronization using TBCLKSYNC

TBCLKSYNC is a global control that enables the time-base clock for all ePWM modules.

- PWM configuration is performed while TBCLKSYNC is disabled.
- Both PWM counters are reset to zero.
- TBCLKSYNC is enabled once, starting all PWMs at the same instant.

This ensures that ePWM1 and ePWM2 begin operation with perfect phase alignment.

5 Sampling and PWM Frequencies

$$f_{\text{ADC}} = 1\,\text{kHz}, \quad f_{\text{ePWM1}} = 1\,\text{kHz}, \quad f_{\text{ePWM2}} = 10\,\text{kHz}$$

Five cycles of the 10 kHz PWM signal occur within one 1 kHz sampling period.

6 Firmware Implementation (Task 2)

```
//*************************************
// FILE: hotzenblitz_current_sensing_with_synced_pwm.c
//*************************************

#include "driverlib.h"
#include "device.h"

//*************************************
// ADC Configuration
//*************************************
#define SENSOR1_ADC_MODULE      ADCA_BASE
#define SENSOR1_ADC_CHANNEL     ADC_CH_ADCIN2
#define SENSOR1_RESULT_BASE     ADCARESULT_BASE
#define SENSOR1_SOC_NUMBER      ADC_SOC_NUMBER0

#define SENSOR2_ADC_MODULE      ADCA_BASE
#define SENSOR2_ADC_CHANNEL     ADC_CH_ADCIN14
#define SENSOR2_RESULT_BASE     ADCARESULT_BASE
```

```
19 #define SENSOR2_SOC_NUMBER      ADC_SOC_NUMBER1
20
21 //*************************************
22 // PWM Configuration
23 //*************************************
24 #define PWM1_OUTPUT_GPIO        0
25 #define PWM1_OUTPUT_PIN_CONFIG  GPIO_0_EPWM1_A
26 #define PWM2_OUTPUT_GPIO        2
27 #define PWM2_OUTPUT_PIN_CONFIG  GPIO_2_EPWM2_A
28
29 //*************************************
30 // Global Variables
31 //*************************************
32 uint16_t sensor1_raw = 0;
33 uint16_t sensor2_raw = 0;
34
35 float current_phase_a = 0.0f;
36 float current_phase_b = 0.0f;
37 float total_current = 0.0f;
38
39 uint32_t sample_count = 0;
40
41 //*************************************
42 // Function Prototypes
43 //*************************************
44 void InitEPWM1(void);
45 void InitEPWM2_10kHz(void);
46 void StartPWMsSynchronized(void);
47 void InitADC(void);
48 void InitADCSOC(void);
49 void ConfigurePWMOutputGPIO(void);
50 __interrupt void ADCA1_ISR(void);
51
52 //*************************************
53 // Main Function
54 //*************************************
55 void main(void)
56 {
57     Device_init();
58     Device_initGPIO();
59
60     ConfigurePWMOutputGPIO();
61
62     Interrupt_initModule();
63     Interrupt_initVectorTable();
64
65     InitADC();
66     InitEPWM1();
67     InitEPWM2_10kHz();
68
69     StartPWMsSynchronized();
```

```
70
71      InitADCSOC();
72
73      Interrupt_register(INT_ADCA1, &ADCA1_ISR);
74      Interrupt_enable(INT_ADCA1);
75
76      EINT;
77      ERTM;
78
79      while(1)
80      {
81          NOP;
82      }
83  }
84
85  //*************************************
86  // GPIO Configuration
87  //*************************************
88  void ConfigurePWMOutputGPIO(void)
89  {
90      GPIO_setPadConfig(PWM1_OUTPUT_GPIO, GPIO_PIN_TYPE_STD);
91      GPIO_setPinConfig(PWM1_OUTPUT_PIN_CONFIG);
92
93      GPIO_setPadConfig(PWM2_OUTPUT_GPIO, GPIO_PIN_TYPE_STD);
94      GPIO_setPinConfig(PWM2_OUTPUT_PIN_CONFIG);
95  }
96
97  //*************************************
98  // ePWM1 Configuration
99  //*************************************
100 void InitEPWM1(void)
101 {
102     SysCtl_enablePeripheral(SYSCTL_PERIPH_CLK_EPWM1);
103     SysCtl_disablePeripheral(SYSCTL_PERIPH_CLK_TBCLKSYNC);
104
105     EPWM_setTimeBasePeriod(EPWM1_BASE, 60000);
106     EPWM_setTimeBaseCounter(EPWM1_BASE, 0);
107
108     EPWM_setClockPrescaler(EPWM1_BASE,
109                            EPWM_CLOCK_DIVIDER_1,
110                            EPWM_HSCLOCK_DIVIDER_1);
111
112     EPWM_setTimeBaseCounterMode(EPWM1_BASE,
        EPWM_COUNTER_MODE_UP_DOWN);
113     EPWM_setPhaseShift(EPWM1_BASE, 0);
114     EPWM_disablePhaseShiftLoad(EPWM1_BASE);
115     EPWM_setEmulationMode(EPWM1_BASE, EPWM_EMULATION_FREE_RUN);
116
117     EPWM_setCounterCompareValue(EPWM1_BASE,
118                                 EPWM_COUNTER_COMPARE_A,
119                                 59000);
```

```
120
121     EPWM_setActionQualifierAction(EPWM1_BASE,
122                                   EPWM_AQ_OUTPUT_A,
123                                   EPWM_AQ_OUTPUT_HIGH,
124                                   EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPA
                                          );
125
126     EPWM_setActionQualifierAction(EPWM1_BASE,
127                                   EPWM_AQ_OUTPUT_A,
128                                   EPWM_AQ_OUTPUT_LOW,
129                                   EPWM_AQ_OUTPUT_ON_TIMEBASE_PERIOD
                                          );
130
131     EPWM_setADCTriggerSource(EPWM1_BASE,
132                              EPWM_SOC_A,
133                              EPWM_SOC_TBCTR_PERIOD);
134
135     EPWM_setADCTriggerEventPrescale(EPWM1_BASE, EPWM_SOC_A, 1);
136     EPWM_enableADCTrigger(EPWM1_BASE, EPWM_SOC_A);
137 }
138
139 //*************************************
140 // ePWM2 Configuration
141 //*************************************
142 void InitEPWM2_10kHz(void)
143 {
144     SysCtl_enablePeripheral(SYSCTL_PERIPH_CLK_EPWM2);
145
146     EPWM_setTimeBasePeriod(EPWM2_BASE, 6000);
147     EPWM_setTimeBaseCounter(EPWM2_BASE, 0);
148
149     EPWM_setClockPrescaler(EPWM2_BASE,
150                            EPWM_CLOCK_DIVIDER_1,
151                            EPWM_HSCLOCK_DIVIDER_1);
152
153     EPWM_setTimeBaseCounterMode(EPWM2_BASE,
          EPWM_COUNTER_MODE_UP_DOWN);
154     EPWM_setPhaseShift(EPWM2_BASE, 0);
155     EPWM_disablePhaseShiftLoad(EPWM2_BASE);
156
157     EPWM_setCounterCompareValue(EPWM2_BASE,
158                                 EPWM_COUNTER_COMPARE_A,
159                                 3000);
160
161     EPWM_setActionQualifierAction(EPWM2_BASE,
162                                   EPWM_AQ_OUTPUT_A,
163                                   EPWM_AQ_OUTPUT_LOW,
164                                   EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPA
                                          );
165
166     EPWM_setActionQualifierAction(EPWM2_BASE,
```

```
                                        EPWM_AQ_OUTPUT_A,
                                        EPWM_AQ_OUTPUT_HIGH,
                                        EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPA
                                            );

    EPWM_setEmulationMode(EPWM2_BASE, EPWM_EMULATION_FREE_RUN);
}

//*************************************
// PWM Synchronization
//*************************************
void StartPWMsSynchronized(void)
{
    EPWM_setTimeBaseCounter(EPWM1_BASE, 0);
    EPWM_setTimeBaseCounter(EPWM2_BASE, 0);

    SysCtl_enablePeripheral(SYSCTL_PERIPH_CLK_TBCLKSYNC);
}

//*************************************
// ADC Configuration
//*************************************
void InitADC(void)
{
    ADC_setPrescaler(ADCA_BASE, ADC_CLK_DIV_4_0);
    ADC_setInterruptPulseMode(ADCA_BASE, ADC_PULSE_END_OF_CONV);
    ADC_enableConverter(ADCA_BASE);
    DEVICE_DELAY_US(1000);
}

void InitADCSOC(void)
{
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER0,
                 ADC_TRIGGER_EPWM1_SOCA,
                 ADC_CH_ADCIN2, 15);

    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER1,
                 ADC_TRIGGER_EPWM1_SOCA,
                 ADC_CH_ADCIN14, 15);

    ADC_setInterruptSource(ADCA_BASE,
                           ADC_INT_NUMBER1,
                           ADC_SOC_NUMBER1);

    ADC_enableContinuousMode(ADCA_BASE, ADC_INT_NUMBER1);
    ADC_enableInterrupt(ADCA_BASE, ADC_INT_NUMBER1);
    ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1);
}

//*************************************
// ADC Interrupt Service Routine
```

```
217 //*************************************
218 __interrupt void ADCA1_ISR(void)
219 {
220     sensor1_raw = ADC_readResult(ADCARESULT_BASE, ADC_SOC_NUMBER0
        );
221     sensor2_raw = ADC_readResult(ADCARESULT_BASE, ADC_SOC_NUMBER1
        );
222
223     current_phase_a = ((float)sensor1_raw / 4095.0f) * 100.0f;
224     current_phase_b = ((float)sensor2_raw / 4095.0f) * 100.0f;
225
226     total_current = current_phase_a + current_phase_b;
227     sample_count++;
228
229     ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1);
230     Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP1);
231 }
232
233 //*************************************
234 // End of File
235 //*************************************
```

7 Oscilloscope Verification

Two channels were used:

- Channel 1: GPIO0 (ePWM1A, 1 kHz)
- Channel 2: GPIO2 (ePWM2A, 10 kHz)

Both signals start simultaneously at $t = 0$, and five cycles of the 10 kHz PWM are visible within one 1 kHz period.

8 Conclusion

Task 2 successfully demonstrates synchronized multi-frequency PWM generation together with deterministic ADC sampling. The use of ePWM hardware triggering and TBCLKSYNC ensures precise timing, making the system suitable for real-time motor-control applications.

Reference

Texas Instruments, *C2000™ F28003x Series LaunchPad™ Development Kit User's Guide*, SPRUJ31.