Test design :

Controllable DC/DC Buck Converter

Ju-Wei Huang

ju-wei.huang@sjsu.edu

Requirement:

a.            Focus on the controller - schematics and layout (ideally in Altium but could be any decent software that the candidate is most familiar with). Objective of layout demonstration is to show your considerations for ground returns, and noise-immune design concepts in general

b.            Select chip that can provide automatic cycle-by-cycle control of the duty cycle but at the same time take in control input from the microcontroller for target output current. Control input could be 0-5V voltage signal or digital comms (I2C / SPI, etc).

c.             Target SiC FET half-bridge modules as power devices (200A output current) - define switching frequency accordingly - such as <https://www.wolfspeed.com/cas300m12bm2>

d.            Note driver considerations for SiC FET module

e.            Note EMC considerations for SiC FET module

MATLAB Simulation

Using MATLAB code to calculate the values of the buck converter.

% Assume

fs= 100e3;

Vi=1000;

D=0.5;

R=2.5;

I\_ripple=0.1;

V\_ripple=0.05;

% Calculation

Vo=Vi\*D

IL\_avg=Vo/R

del\_iL=I\_ripple\*IL\_avg

del\_vC=V\_ripple\*Vo

L=(Vo\*(1-D))/(fs\*del\_iL)

C=(1-D)/(8\*(fs^2)\*L\*(del\_vC/Vo))

P=(Vo^2)/R

Thus, I can have the values of inductor, capacitor.

Vo = 500

IL\_avg = 200

del\_iL = 20

del\_vC = 25

L = 1.2500e-04

C = 1.0000e-06

P = 100000

With those values, I use MATLAB Simulink to do the Buck converter with current program control.

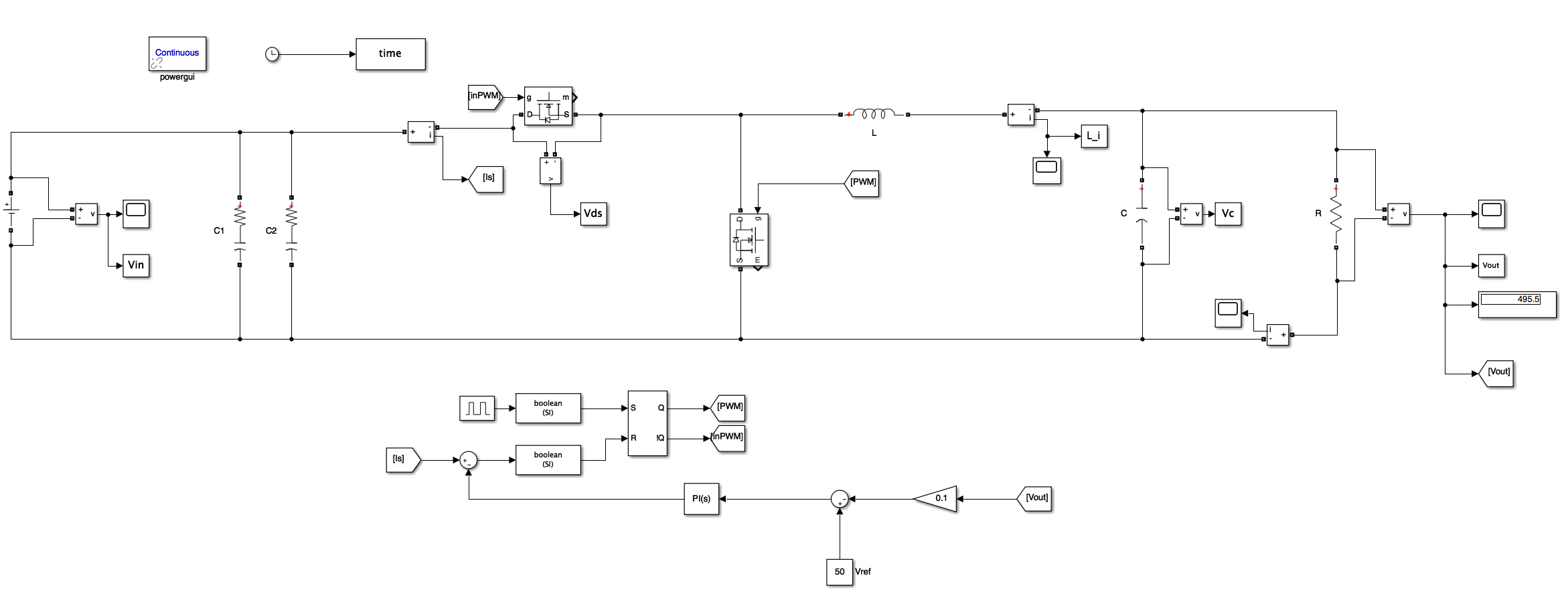


Figure 1 MATLAB Simulink model for Buck converter with current program control.



Figure 2 Input voltage and output voltage waveform.



Figure 3 Inductor current waveform.



Figure 4 Capacitor voltage waveform.

Schematic Design

1. Converter Design

OrCAD/Converter.pdf

Figure 5 Schematic for proposed converter.

The design of this converter was followed in two discrete stages. The first was to develop the topologies in Simulink and test for the functionality and the stresses. Next, this result is used to determine which system needs to be made modular in order to reduce the stress and overall cost of the converter.

The converter used along with its auxiliary circuit is designed in OrCAD Capture and tested for its functionality. BOM of this designed converter is then generated. Footpads and Footprints are created in Allegro/PCB Editor for all the components. These footprints are then used in part placement. PCB stack up is then decided based on the maximum current each layer will be carrying.

The layout of the converter is then generated carefully while calculating and ensuring the trace width used on each layer. Once the layout is completed, Gerber files and NC drill files are generated in order to obtain a DFM report from the PCB manufacturer.

The input capacitor C3, C4, C67 are used as filters and to increase the bandwidth of the converter.

* 1. Isolated Gate Driver

OrCAD/Isolated%20Gate%20Driver.pdf

Figure 6 Schematic for isolated gate driver.

The UCC21520 is an isolated dual-channel gate driver with a 4-A source and 6-A sink peak current. The device is designed to drive power MOSFETs, IGBTs, and SiC MOSFETs up to 5 MHz with best-in-class propagation delay and pulse-width distortion. The input side is isolated from the two output drivers by a 5.7-kVRMS reinforced isolation barrier with a minimum of 100-V/ns common-mode transient immunity (CMTI). Internal functional isolation between the two secondary-side drivers allows a working voltage of up to 1500-V DC. A disable pin shuts down both outputs simultaneously when it is set high and allows normal operation when left floating or grounded. The device accepts VDD supply voltages up to 25 V. A wide input VCCI range from 3 V to 18 V makes the driver suitable for interfacing with both analog and digital controllers.

A bypass capacitor is connected to VCC1 supports the transient current needed for the primary logic and total current consumption i.e. a few mA. Thus, 50V MLCC > 100nF is recommended. A 1μF electrolytic capacitor is recommended to be connected in parallel to MLCC if the bias supply output is relatively long distance from the VCC1 pin.

DT Pin is used to introduce a dead time. Dead time can be introduced either through

the microcontroller or by selecting a value of resistance. Here a value of resistance is selected with the help of simulation to introduce the necessary deadtime:

RDT = 100Ω

Resistor Rin and capacitor Cin are used to filter out the ringing introduced by non-ideal layout long PCB traces. Thus, their values are selected as follows

Rin = 0 to 100Ω

Cin = 10 to 100pF

While selecting the bootstrap diode it is necessary to ensure high voltage and fast recovery diodes/Schottky diodes are chosen with low forward voltage drop and low junction capacitance in order to reduce losses.

Voltage rating of diode should be greater than the DC link voltage with a good margin. Resistor RBOOT is used to reduce inrush current in Dboot and to limit the slew rate of VDDA and VSSA during each switching cycle. Thus, RBOOT is selected in the range between 1Ω to 20Ω [Generally 2.2Ω should be fine]. VBDF is the estimated diode forward voltage drop. It is essential to know the worst case peak current through the bootstrap diode in order to ensure that the gate drive is safe during this condition.

The external gate driver resistors RON/ROFF is used to:

1. Limit ringing caused by parasitic capacitance/Inductance

2. Limit ringing caused by high voltage/current switching dv/dt, di/dt and body diode reverse recovery.

3. Peak sink and source current to optimize the switching loss

4. Reduce EMI

ROFF is the resistance connected in series with the diode and is a very small value.

R19 and R23 are used to reduce the parasitic ringing.

* 1. Voltage Sensor

OrCAD/Input%20Voltage%20Sensor.pdfFigure 7 Schematic for voltage sensor.

The ACPL-7900 precision miniature isolation amplifiers are designed for voltage sensing in electronic power converter applications such as motor drives and renewable energy systems. In a typical converter implementation, current flows through an external resistor and the resulting analog voltage drop is sensed by the isolation amplifier. A differential output voltage that is proportional to the current is created on the other side of the optical isolation barrier. For general applications, ACPL-7900 (±3% gain tolerance) are recommended. The voltage sensor operates from a single 5V supply and provides excellent linearity and dynamic performance of 60 dB SNR.

It has a 200 kHz bandwidth and 1.6 μs response time. Combined with superior optical coupling technology, the ACPL-7900 precision miniature isolation amplifiers implement sigma-delta analog-to-digital converter, chopper stabilized amplifiers, and a fully differential circuit topology to provide unequaled isolation-mode noise rejection, low offset, high gain accuracy and stability. This performance is delivered in a compact, auto-insertable DIP-8 package that meets worldwide regulatory safety standards. These voltage sensors are used to sense the input and the output voltage in the Buck converter.

The maximum allowable input voltage of the voltage sensor is 200mV. Thus, a resistor divider circuit with R24 = 50M and R25 = 1K is used. The maximum input voltage is assumed to be 1000V. Using the resistor divider calculation:

Vout / Vin = R2/(R1+R2)

Vout=1000\*1k/(50M+1k) = 200mV

The gain of the sensor is 8.2(max). Thus, the maximum achievable output is = 200mV × 8.2 = 1.64V. Thus, in the op-amp of the double ended to single ended circuit ahead of the output of the voltage sensor a gain of 2 is used to send a voltage of 2 × 1.64 = 3.28V which is within the acceptable limits of the DSP input which is 3.3V.

Two resistors R26 and R27 of 1KΩ each are used and a 4KΩ resistor R30 is connected between the output and the input of the op-amp. This gives us the gain of the op-amp:

Gain= R30 /(R26 + R27 )=4/(1+1)=2

Capacitor C45 is calculated using the formula

f = 2πRC. C = 100K/2π × (1+1) K = 0.8nF ≈1nF.

* 1. Current Sensor

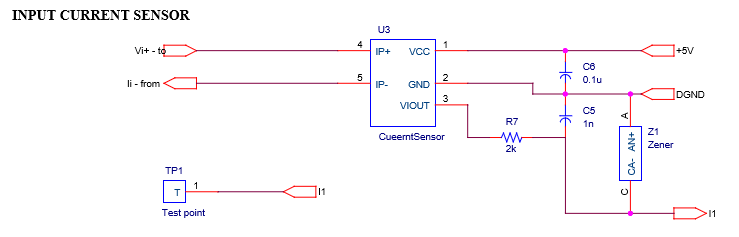


Figure 8 Schematic for current sensor.

The current sensor ACS770ECB-200U-PFF-T consists of a precision, low-offset linear Hall circuit with a copper conduction path located near the die. The sensor provides economical and precise solutions for AC or DC current sensing. Typical applications include motor control, load detection and management, power supply and DC-to-DC converter control, inverter control, and overcurrent fault detection. The current is sensed differentially in order to reject common-mode fields, improving accuracy in magnetically noisy environments. The inherent device accuracy is optimized through the close proximity of the magnetic field to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC. The output of the device has a positive slope when an increasing current flows through the primary copper conduction path. This sensor is used to sense the input current and the output current.

* 1. Buffer

The SN74LV1T34 device is a low voltage CMOS gate logic that operates at a wider voltage range for industrial, portable, and telecom applications. The output level is referenced to the supply voltage and supports 1.8-V, 2.5-V, 3.3-V, and 5-V CMOS levels. The input has a lower threshold circuit to match 1.8 V input logic at VCC = 3.3 V and can be used in 1.8 V to 3.3 V level up translation. In addition, the 5 V tolerant input pins enable down translation (that is, 3.3 V to 2.5 V output at VCC = 2.5 V). The wide VCC range of 1.8 V to 5.5 V allows generation of desired output levels to connect to controllers or processors. The SN74LV1T34 device is designed with current drive capability of 8 mA to reduce line reflections, overshoot, and undershoot caused by high-drive outputs. The buffers are used to provide additional isolation between DSP and the isolated gate driver.

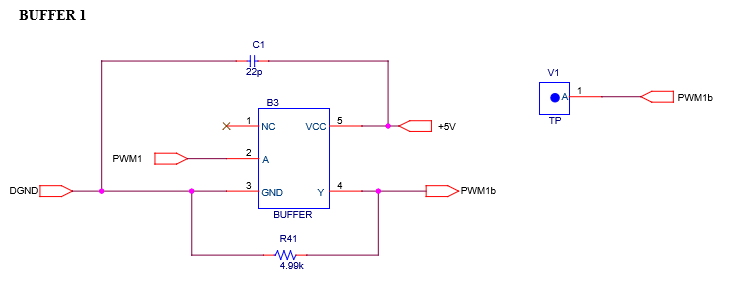


Figure 9 Schematic for buffer.

* 1. 5V Regulator

An isolated DC-DC converter is selected whose input is 5V and provides a 5V fixed output.

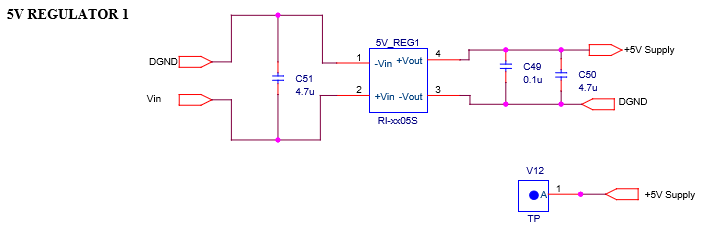


Figure 10 Schematic for 5V regulator.

* 1. 15V Regulator

An isolated DC-DC converter is selected whose input is 5V and provides a 15V boosted output.

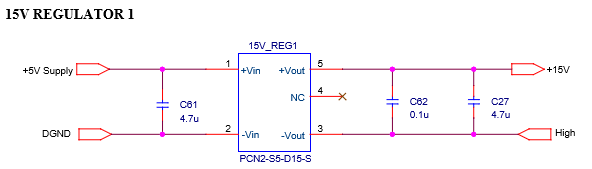


Figure 11 Schematic for 15V regulator.

PCB Layout

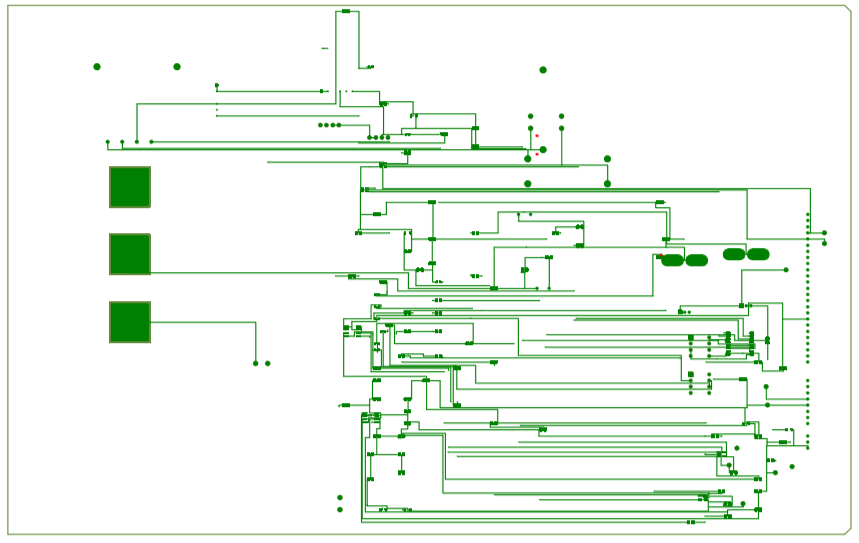


Figure 12 Top layer of the PCBA.

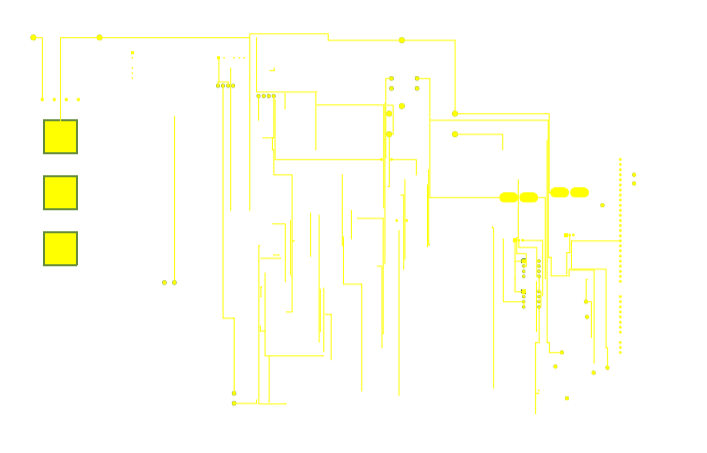


Figure 13 Bottom layer of the PCBA.

Future Work

1. EMC Consideration

To prevent the EMC in SiC MOSFET, we can use snubber circuit. A simple RC snubber uses a small resistor (R) in series with a small capacitor (C). This combination can be used to suppress the rapid rise in voltage across a thyristor, preventing the erroneous turn-on of the thyristor; it does this by limiting the rate of rise in voltage (dV/dt) across the thyristor to a value which will not trigger it. An appropriately-designed RC snubber can be used with either DC or AC loads. This sort of snubber is commonly used with inductive loads such as electric motors. The voltage across a capacitor cannot change instantaneously, so a decreasing transient current will flow through it for a small fraction of a second, allowing the voltage across the switch to increase more slowly when the switch is opened. Determination of voltage rating can be difficult owing to the nature of transient waveforms, and may be defined simply by the power rating of the snubber components and the application.

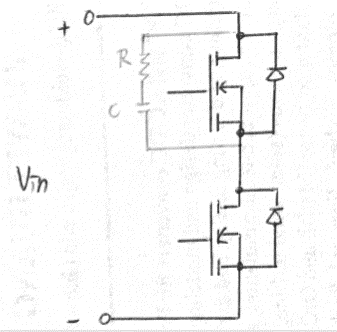


Figure 12 Proposed RC snubber circuit.

1. 3 Level Switching Circuit

In order to reduce the ripple in inductor current and the current stress on the components, we can use the 3 level switching circuit. The advantage of using this circuit is that the switch loss will decrease.

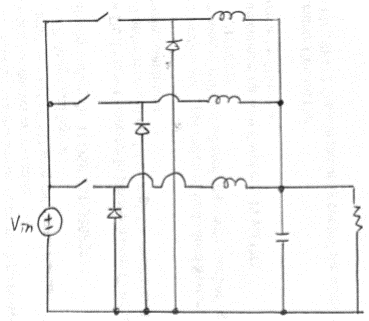


Figure 1 Proposed 3 level switching circuit.

DSP Coding

// Included Files

//

#include "F28x\_Project.h"

//

// Defines

//

#define EPWM1\_TIMER\_TBPRD 588 // Period register

#define EPWM1\_CMPA 294

#define EPWM2\_TIMER\_TBPRD 588 // Period register

#define EPWM2\_CMPA 294

#define EPWM3\_TIMER\_TBPRD 588 // Period register

#define EPWM3\_CMPA 294

#define EPWM4\_TIMER\_TBPRD 588 // Period register

#define EPWM4\_CMPA 294

//

// Globals

//

Uint16 AdcaResult;

Uint16 AdcaResult\_offsetAdjusted;

Uint16 AdcbResult;

Uint16 AdcbResult\_offsetAdjusted;

//Uint16 ConversionCount;

Uint16 LoopCount;

Uint16 ConversionCount;

Uint16 Voltage1[10];

Uint16 Voltage2[10];

Uint16 Voltage3[10];

Uint16 Voltage4[10];

// Function Prototypes

//

void InitEPwm1Example(void);

void InitEPwm2Example(void);

void InitEPwm3Example(void);

void InitEPwm4Example(void);

void ConfigureADC(void);

void SetupADCSoftware(void);

interrupt void adc\_isr(void);

//

// Main

//

void main(void)

{

//

// Step 1. Initialize System Control:

// PLL, WatchDog, enable Peripheral Clocks

// This example function is found in the F2837xD\_SysCtrl.c file.

//

InitSysCtrl();

//

// Step 2. Initialize GPIO:

// This example function is found in the F2837xD\_Gpio.c file and

// illustrates how to set the GPIO to it's default state.

//

InitGpio();

//

// enable PWM1, PWM2 and PWM3

//

CpuSysRegs.PCLKCR2.bit.EPWM1=1;

CpuSysRegs.PCLKCR2.bit.EPWM2=1;

CpuSysRegs.PCLKCR2.bit.EPWM3=1;

CpuSysRegs.PCLKCR2.bit.EPWM4=1;

//

// For this case just init GPIO pins for ePWM1, ePWM2, ePWM3, and ePWM4

// These functions are in the F2837xD\_EPwm.c file

//

InitEPwm1Gpio();

InitEPwm2Gpio();

InitEPwm3Gpio();

InitEPwm4Gpio();

//

// Step 3. Clear all interrupts and initialize PIE vector table:

// Disable CPU interrupts

//

DINT;

//

// Initialize the PIE control registers to their default state.

// The default state is all PIE interrupts disabled and flags

// are cleared.

// This function is found in the F2837xD\_PieCtrl.c file.

//

InitPieCtrl();

//

// Disable CPU interrupts and clear all CPU interrupt flags:

//

IER = 0x0000;

IFR = 0x0000;

//

// Initialize the PIE vector table with pointers to the shell Interrupt

// Service Routines (ISR).

// This will populate the entire table, even if the interrupt

// is not used in this example. This is useful for debug purposes.

// The shell ISR routines are found in F2837xD\_DefaultIsr.c.

// This function is found in F2837xD\_PieVect.c.

//

InitPieVectTable();

// Interrupts that are used in this example are re-mapped to

// ISR functions found within this file.

EALLOW; // This is needed to write to EALLOW protected register

PieVectTable.ADCINT = &adc\_isr;

EDIS; // This is needed to disable write to EALLOW protected registers

// Step 4.Initialize all the Device Peripherals:

// This function is found in DSP2833x\_InitPeripherals.c

// InitPeripherals(); // Not required for this example

InitAdc(); // For this example, init the ADC

EALLOW;

CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 0;

EDIS;

// Step 5. User specific code,

// Enable ADCINT in PIE

PieCtrlRegs.PIEIER1.bit.INTx6 = 1;

IER |= M\_INT1; // Enable CPU Interrupt 1

EINT; // Enable Global interrupt INTM

ERTM; // Enable Global realtime interrupt DBGM

LoopCount = 0;

ConversionCount = 0;

ConfigureADC();

// Wait for ADC interrupt

for(;;)

{

LoopCount++;

}

}

//

// InitEPwm1Example - Initialize EPWM1 configuration

//

void InitEPwm1Example()

{

//

// Setup TBCLK

//

EPwm1Regs.TBPRD = EPWM1\_TIMER\_TBPRD; // Set timer period

EPwm1Regs.TBPHS.bit.TBPHS = 0x0000; // Phase is 0

EPwm1Regs.TBCTR = 0x0000; // Clear counter

//

// Set Compare values

//

EPwm1Regs.CMPA.bit.CMPA = EPWM1\_CMPA; // Set compare A value

//

// Setup counter mode

//

EPwm1Regs.TBCTL.bit.CTRMODE = TB\_COUNT\_UPDOWN; // Count up and down

EPwm1Regs.TBCTL.bit.PHSEN = TB\_DISABLE; // Disable phase loading

EPwm1Regs.TBCTL.bit.PRDLD = TB\_SHADOW;

EPwm1Regs.TBCTL.bit.SYNCOSEL = TB\_CTR\_ZERO; // Sync down-stream module

EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB\_DIV1; // Clock ratio to SYSCLKOUT

EPwm1Regs.TBCTL.bit.CLKDIV = TB\_DIV1;

//

// Setup shadowing

//

EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC\_SHADOW;

EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC\_SHADOW;

EPwm1Regs.CMPCTL.bit.LOADAMODE = CC\_CTR\_ZERO; // Load on Zero

EPwm1Regs.CMPCTL.bit.LOADBMODE = CC\_CTR\_ZERO;

//

// Set actions

//

EPwm1Regs.AQCTLA.bit.CAU = AQ\_SET; // Set PWM1A on event A, up

// count

EPwm1Regs.AQCTLA.bit.CAD = AQ\_CLEAR; // Clear PWM1A on event A,

// down count

// Dead Band

EPwm1Regs.DBCTL.bit.OUT\_MODE = DB\_FULL\_ENABLE; // enable Dead-band module

EPwm1Regs.DBCTL.bit.POLSEL = DB\_ACTV\_HIC; // Active Hi complementary

EPwm1Regs.DBFED.bit.DBFED = 50; // FED = 150 ns

EPwm1Regs.DBRED.bit.DBRED = 50; // RED = 150 ns

}

//

// InitEPwm2Example - Initialize EPWM2 configuration

//

void InitEPwm2Example()

{

//

// Setup TBCLK

//

EPwm2Regs.TBPRD = EPWM2\_TIMER\_TBPRD; // Set timer period

EPwm2Regs.TBPHS.bit.TBPHS = 0x0000; // Phase is 0

EPwm2Regs.TBCTR = 0x0000; // Clear counter

//

// Set Compare values

//

EPwm2Regs.CMPA.bit.CMPA = EPWM2\_CMPA; // Set compare A value

//

// Setup counter mode

//

EPwm2Regs.TBCTL.bit.CTRMODE = TB\_COUNT\_UPDOWN; // Count up and down

EPwm2Regs.TBCTL.bit.PHSEN = TB\_ENABLE; // Enable phase loading

EPwm2Regs.TBCTL.bit.PRDLD = TB\_SHADOW;

EPwm2Regs.TBCTL.bit.SYNCOSEL = TB\_SYNC\_IN; // sync flow-through

EPwm2Regs.TBCTL.bit.HSPCLKDIV = TB\_DIV1; // Clock ratio to SYSCLKOUT

EPwm2Regs.TBCTL.bit.CLKDIV = TB\_DIV1;

//

// Setup shadowing

//

EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC\_SHADOW;

EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC\_SHADOW;

EPwm2Regs.CMPCTL.bit.LOADAMODE = CC\_CTR\_ZERO; // Load on Zero

EPwm2Regs.CMPCTL.bit.LOADBMODE = CC\_CTR\_ZERO;

//

// Set actions

//

EPwm2Regs.AQCTLA.bit.CAU = AQ\_CLEAR; // Clear PWM2A on event A, up

// count

EPwm2Regs.AQCTLA.bit.CAD = AQ\_SET; // Set PWM2A on event B, down

// count

// Dead Band

EPwm2Regs.DBCTL.bit.OUT\_MODE = DB\_FULL\_ENABLE; // enable Dead-band module

EPwm2Regs.DBCTL.bit.POLSEL = DB\_ACTV\_HIC; // Active Hi complementary

EPwm2Regs.DBFED.bit.DBFED = 50; // FED = 150 ns

EPwm2Regs.DBRED.bit.DBRED = 50; // RED = 150 ns

}

//

// InitEPwm3Example - Initialize EPWM3 configuration

//

void InitEPwm3Example(void)

{

//

// Setup TBCLK

//

EPwm3Regs.TBPRD = EPWM3\_TIMER\_TBPRD; // Set timer period

EPwm3Regs.TBPHS.bit.TBPHS = 0x0000; // Phase is 0

EPwm3Regs.TBCTR = 0x0000; // Clear counter

//

// Set Compare values

//

EPwm3Regs.CMPA.bit.CMPA = EPWM3\_CMPA; // Set compare A value

//

// Setup counter mode

//

EPwm3Regs.TBCTL.bit.CTRMODE = TB\_COUNT\_UPDOWN; // Count up and down

EPwm3Regs.TBCTL.bit.PHSEN = TB\_ENABLE; // Enable phase loading

EPwm3Regs.TBCTL.bit.PRDLD = TB\_SHADOW;

EPwm3Regs.TBCTL.bit.SYNCOSEL = TB\_SYNC\_IN; // sync flow-through

EPwm3Regs.TBCTL.bit.HSPCLKDIV = TB\_DIV1; // Clock ratio to SYSCLKOUT

EPwm3Regs.TBCTL.bit.CLKDIV = TB\_DIV1;

//

// Setup shadowing

//

EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC\_SHADOW;

EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC\_SHADOW;

EPwm3Regs.CMPCTL.bit.LOADAMODE = CC\_CTR\_ZERO; // Load on Zero

EPwm3Regs.CMPCTL.bit.LOADBMODE = CC\_CTR\_ZERO;

//

// Set actions

//

EPwm3Regs.AQCTLA.bit.CAU = AQ\_SET; // Set PWM3A on event A, up

// count

EPwm3Regs.AQCTLA.bit.CAD = AQ\_CLEAR; // Clear PWM3A on event B, down

// count

// Dead Band

EPwm3Regs.DBCTL.bit.OUT\_MODE = DB\_FULL\_ENABLE; // enable Dead-band module

EPwm3Regs.DBCTL.bit.POLSEL = DB\_ACTV\_HIC; // Active Hi complementary

EPwm3Regs.DBFED.bit.DBFED = 50; // FED = 150 ns

EPwm3Regs.DBRED.bit.DBRED = 50; // RED = 150 ns

}

//

// InitEPwm4Example - Initialize EPWM4 configuration

//

void InitEPwm4Example(void)

{

//

// Setup TBCLK

//

EPwm4Regs.TBPRD = EPWM4\_TIMER\_TBPRD; // Set timer period

EPwm4Regs.TBPHS.bit.TBPHS = 0x0000; // Phase is 0

EPwm4Regs.TBCTR = 0x0000; // Clear counter

//

// Set Compare values

//

EPwm4Regs.CMPA.bit.CMPA = EPWM4\_CMPA; // Set compare A value

//

// Setup counter mode

//

EPwm4Regs.TBCTL.bit.CTRMODE = TB\_COUNT\_UPDOWN; // Count up and down

EPwm4Regs.TBCTL.bit.PHSEN = TB\_ENABLE; // Enable phase loading

EPwm4Regs.TBCTL.bit.PRDLD = TB\_SHADOW;

EPwm4Regs.TBCTL.bit.SYNCOSEL = TB\_SYNC\_IN; // sync flow-through

EPwm4Regs.TBCTL.bit.HSPCLKDIV = TB\_DIV1; // Clock ratio to SYSCLKOUT

EPwm4Regs.TBCTL.bit.CLKDIV = TB\_DIV1;

//

// Setup shadowing

//

EPwm4Regs.CMPCTL.bit.SHDWAMODE = CC\_SHADOW;

EPwm4Regs.CMPCTL.bit.SHDWBMODE = CC\_SHADOW;

EPwm4Regs.CMPCTL.bit.LOADAMODE = CC\_CTR\_ZERO; // Load on Zero

EPwm4Regs.CMPCTL.bit.LOADBMODE = CC\_CTR\_ZERO;

//

// Set actions

//

EPwm4Regs.AQCTLA.bit.CAU = AQ\_CLEAR; // Clear PWM4A on event A, up

// count

EPwm4Regs.AQCTLA.bit.CAD = AQ\_SET; // Set PWM4A on event B, down

// count

// Dead Band

EPwm4Regs.DBCTL.bit.OUT\_MODE = DB\_FULL\_ENABLE; // enable Dead-band module

EPwm4Regs.DBCTL.bit.POLSEL = DB\_ACTV\_HIC; // Active Hi complementary

EPwm4Regs.DBFED.bit.DBFED = 50; // FED = 150ns

EPwm4Regs.DBRED.bit.DBRED = 50; // RED = 150ns

}

void ConfigureADC(void)

{

EALLOW;

//

//write configurations

//

AdcaRegs.ADCCTL2.bit.PRESCALE = 6; //set ADCCLK divider to /4

AdcbRegs.ADCCTL2.bit.PRESCALE = 6; //set ADCCLK divider to /4

AdcSetMode(ADC\_ADCA, ADC\_RESOLUTION\_12BIT, ADC\_SIGNALMODE\_SINGLE);

AdcSetMode(ADC\_ADCB, ADC\_RESOLUTION\_12BIT, ADC\_SIGNALMODE\_SINGLE);

//

//Set pulse positions to late

//

AdcaRegs.ADCCTL1.bit.INTPULSEPOS = 1;

AdcbRegs.ADCCTL1.bit.INTPULSEPOS = 1;

//

//power up the ADCs

//

AdcaRegs.ADCCTL1.bit.ADCPWDNZ = 1;

AdcbRegs.ADCCTL1.bit.ADCPWDNZ = 1;

//

//delay for 1ms to allow ADC time to power up

//

DELAY\_US(1000);

EDIS;

}

//

// SetupADCSoftware - Configure ADC SOC and acquisition window

//

void SetupADCSoftware(void)

{

Uint16 acqps;

//

//determine minimum acquisition window (in SYSCLKS) based on resolution

//

if(ADC\_RESOLUTION\_12BIT == AdcaRegs.ADCCTL2.bit.RESOLUTION)

{

acqps = 14; //75ns

}

else //resolution is 16-bit

{

acqps = 63; //320ns

}

//

//Select the channels to convert and end of conversion flag

//ADCA

//

EALLOW;

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 0; //SOC0 will convert pin A0

AdcaRegs.ADCSOC0CTL.bit.ACQPS = acqps; //sample window is 100 SYSCLK cycles

AdcaRegs.ADCSOC1CTL.bit.CHSEL = 0; //SOC1 will convert pin A0

AdcaRegs.ADCSOC1CTL.bit.ACQPS = acqps; //sample window is 100 SYSCLK cycles

AdcaRegs.ADCINTSEL1N2.bit.INT1SEL = 1; //end of SOC1 will set INT1 flag

AdcaRegs.ADCINTSEL1N2.bit.INT1E = 1; //enable INT1 flag

AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1; //make sure INT1 flag is cleared

//

//ADCB

//

AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0; //SOC0 will convert pin B0

AdcbRegs.ADCSOC0CTL.bit.ACQPS = acqps; //sample window is 100 SYSCLK cycles

AdcbRegs.ADCSOC1CTL.bit.CHSEL = 0; //SOC1 will convert pin B0

AdcbRegs.ADCSOC1CTL.bit.ACQPS = acqps; //sample window is 100 SYSCLK cycles

AdcbRegs.ADCINTSEL1N2.bit.INT1SEL = 1; //end of SOC1 will set INT1 flag

AdcbRegs.ADCINTSEL1N2.bit.INT1E = 1; //enable INT1 flag

AdcbRegs.ADCINTFLGCLR.bit.ADCINT1 = 1; //make sure INT1 flag is cleared

EDIS;

}

interrupt void adc\_isr(void)

{

Voltage1[ConversionCount] = AdcRegs.ADCRESULT0 >>4;

Voltage2[ConversionCount] = AdcRegs.ADCRESULT1 >>4;

Voltage3[ConversionCount] = AdcRegs.ADCRESULT2 >>4;

Voltage4[ConversionCount] = AdcRegs.ADCRESULT3 >>4;

Voltage5[ConversionCount] = AdcRegs.ADCRESULT4 >>4;

Voltage6[ConversionCount] = AdcRegs.ADCRESULT5 >>4;

Voltage7[ConversionCount] = AdcRegs.ADCRESULT6 >>4;

Voltage8[ConversionCount] = AdcRegs.ADCRESULT7 >>4;

Voltage9[ConversionCount] = AdcRegs.ADCRESULT8 >>4;

// If 40 conversions have been logged, start over

if(ConversionCount == 9)

{

ConversionCount = 0;

}

else ConversionCount++;

// Reinitialize for next ADC sequence

AdcRegs.ADCTRL2.bit.RST\_SEQ1 = 1; // Reset SEQ1

AdcRegs.ADCST.bit.INT\_SEQ1\_CLR = 1; // Clear INT SEQ1 bit

AdcRegs.ADCTRL2.bit.RST\_SEQ2 = 1; // Reset SEQ1

AdcRegs.ADCST.bit.INT\_SEQ2\_CLR = 1; // Clear INT SEQ1 bit

PieCtrlRegs.PIEACK.all = PIEACK\_GROUP1; // Acknowledge interrupt to PIE

return;

}

## BOM

|  |  |  |  |
| --- | --- | --- | --- |
| Index | Quantity | Manufacturer Part Number | Description |
| 1 | 3 | OSTYC022150 | TERMINAL BLOCK 5.08MM 2POS PCB |
| 2 | 4 | SV12AC105KAR | CAP CER 1UF 1KV X7R RADIAL |
| 3 | 1 | CAS300M12BM2 | MOSFET 2N-CH 1200V 404A MODULE |
| 4 | 2 | ACS770ECB-200U-PFF-T | SENSOR CURRENT HALL 200A DC |
| 5 | 2 | RN73C2A2K0BTDF | RES SMD 2K OHM 0.1% 1/10W 0805 |
| 6 | 9 | C0805C102K5RACTU | CAP CER 1000PF 50V X7R 0805 |
| 7 | 18 | GCM21BR72A104KA37L | CAP CER 0.1UF 100V X7R 0805 |
| 8 | 6 | MM3Z3V3B | DIODE ZENER 3.3V 200MW SOD323F |
| 9 | 2 | ACPL-790B-300E | IC OPAMP ISOLATION 200KHZ 8DIPGW |
| 10 | 2 | OPA320AIDBVR | C OPAMP GP 20MHZ RRO SOT23-5 |
| 11 | 2 | TLC352IDR | IC DIFF VOLT COMP DUAL 8-SOIC |
| 12 | 1 | RH73H2A50MKTN | RES SMD 50M OHM 10% 1/8W 0805 |
| 13 | 10 | TNPW08051K00BEEA | RES 1K OHM 0.1% 1/5W 0805 |
| 14 | 4 | ERA-6AEB4021V | RES SMD 4.02K OHM 0.1% 1/8W 0805 |
| 15 | 2 | RR1220P-751-D | RES SMD 750 OHM 0.5% 1/10W 0805 |
| 16 | 2 | ERA-6AEB153V | RES SMD 15K OHM 0.1% 1/8W 0805 |
| 17 | 2 | RR1220P-103-D | RES SMD 10K OHM 0.5% 1/10W 0805 |
| 18 | 2 | RNCP0805FTD604R | RES 604 OHM 1% 1/4W 0805 |
| 19 | 1 | HVC0805T2504JET | RES SMD 2.5M OHM 5% 1/8W 0805 |
| 20 | 2 | PCN2-S5-D15-S | DC DC CONVERTER +/-15V 2W |
| 21 | 8 | GRM21BR61H475KE51L | CAP CER 4.7UF 50V X5R 0805 |
| 22 | 2 | RI-xx05S | DC DC CONVERTER 5V 2W |
| 23 | 2 | SN74LV1T34DCKR | IC BUF NON-INVERT 5.5V SC70-5 |
| 24 | 2 | RG2012P-4991-B-T5 | RES SMD 4.99K OHM 0.1% 1/8W 0805 |
| 25 | 2 | C0805C220J5GACTU | CAP CER 22PF 50V C0G/NP0 0805 |
| 26 | 1 | UCC21520DWR | DGTL ISO 5.7KV GATE DRVR 16SOIC |
| 27 | 2 | RL1220T-R010-J | RES 0.01 OHM 5% 1/4W 0805 |
| 28 | 2 | RCS08052R20JNEA | RES SMD 2.2 OHM 5% 0.4W 0805 |
| 29 | 2 | TNPW080510K0BEEA | RES 10K OHM 0.1% 1/5W 0805 |
| 30 | 2 | ERA-6AEB101V | RES SMD 100 OHM 0.1% 1/8W 0805 |
| 31 | 2 | GRM21BR61E106KA73L | CAP CER 10UF 25V X5R 0805 |
| 32 | 3 | UMK212B7224KG-T | CAP CER 0.22UF 50V X7R 0805 |
| 33 | 1 | CC0805KKX7R8BB105 | CAP CER 1UF 25V X7R 0805 |
| 34 | 1 | 61300311121 | CONN HEADER 3 POS 2.54 |
| 35 | 1 | 61302511121 | CONN HEADER 25 POS 2.54 |
| 36 | 1 | 61300811121 | CONN HEADER 8 POS 2.54 |