Lab 1 Lecture

Monday, January 10, 2022 7:29 PM

Agenda:
- Logistics and Remote Options
- CCS
- EPWM (week 1)
- ADC (week 2)
- OWON USB Oscilloscope

Logistics:
- Please check Announcements on Canvas frequently
- In person lab every Tuesday, remote option offered asynchronously
    - Use OWON oscilloscope and personal PC for remote option
    - Watch Lab Lecture (same material covered during in-person lab section)
    - If doing fully remote, recommend forming a pair with others doing fully remote
    - Please LET ISHAAN KNOW if you are doing fully remote
    - If feeling unwell or exposed to COVID, do not attend class.
    - Ishaan will be flexible if you get sick or need to quarantine.

CCS (11.0 with C2000 support):
- Instructions to get starter code is in Canvas files (Lab 1 CCS.pdf).
- Walkthrough

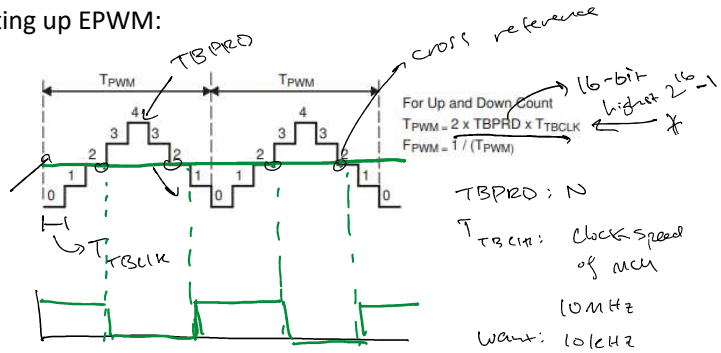
EPWM:
- Peak Valley Sampling
- Use EPWM1A/B: 10kHz, up-down counter mode, 100ns deadtime
- Sync ADC
- Registers

ADC:
- Interrupt
- Registers

DAC
- Used to validate ADC
- Registers

Look in reference user manual

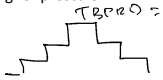
Setting up EPWM:



Registers:
Look in Reference Manual for details and to determine values if I do not give to you.

Set using: EPwm1Regs.REGISTER.bit.BITFIELD

TBCTL: Time Base Control Register
- CLKDIV/HSPCLKDIV: control clock period of EPWM (how long does a single count take) T_{TBCLK}
  - Limits minimum frequency. To get lower frequency, use higher prescale
  - Set such that Fepwm= System Clock = 100MHz
- CTRMODE: determines count mode (Up, Down, or Up-Down)
  - Use Up-Down to implement peak-valley sampling



TBPRD: Set Nmax, or the maximum count value (16-bit)

DBCTL: configure deadtime settings (deadtime loops)
- IN_MODE: determines source of delay, set to 0b0
- POLSEL: used to make complimentary EPWM. Use Active High Complementary (EPWMB is inverted)
- OUTMODE: enable DBFED/DBRED

DBFED/DBRED: configure deadtime values ----------

AQCTLA: set what happens when the PWM counter crosses the reference on up and down count
- Set such that PWM is high when reference higher than carrier

ADC starts converting.

Sync ADC with EPWM:
- **ETSEL**: Enable Start of Conversion (SOC) sync with EPWM
  - SOCAEN bit (enable SOC on EPWM pulse)
  - SOCASEL bit (set to enable peak-valley sampling)
- **ETPS**: control how many EPWM events (peaks/valleys) necessary to start conversion
  - SOCAPRD: set to 0b1
  - INTPRD: 0b1 (may not be necessary)

Enable EPWM on GPIO pins: GPIO pins may have multiple functions. Choose functions with these registers
- **GPAMUX1**:
  - Set such that EPWM is selected on desired pins (EPWM1A/B)
- **GPAPUD**:
  - Disable internal pullup resistor

Binary: 0b10
Hex: 0x2
decimal: 2

How to set Register values?

RegName.REGISTER.bit.BITFIELD = _____ ; TBPRD: use decimal.

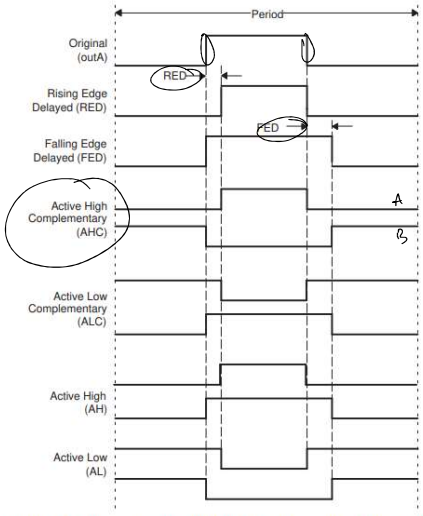


Figure 18-35. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)

(deadtime loops)

The dead-band submodule supports independent values for rising-edge (RED) and falling-edge (FED) delays. The amount of delay is programmed using the DBRED and DBFED registers. These are 10-bit registers and their value represents the number of time-base clock, TBCLK, periods by which a signal edge is delayed. For example, the formula to calculate falling-edge-delay and rising-edge-delay is:

$FED = DBFED \times T_{TBCLK}$

$RED = DBRED \times T_{TBCLK}$

10-bit

Where T_{TBCLK} is the period of TBCLK, the prescaled version of EPWMCLK.

For convenience, delay values for various TBCLK options are shown in Table 18-10. The ePWM input clock frequency that these delay values been computed by is 100 MHz.

Table 18-10. Dead-Band Delay Values in µS as a Function of DBFED and DBRED

Dead-Band Value	Dead-Band Delay in µS		
DBFED, DBRED	TBCLK = EPWMCLK/1	TBCLK = EPWMCLK /2	TBCLK = EPWMCLK/4
1	0.01 µS	0.02 µS	0.04 µS
5	0.05 µS	0.10 µS	0.20 µS
10	0.10 µS	0.20 µS	0.40 µS
100	1.00 µS	2.00 µS	4.00 µS
200	2.00 µS	4.00 µS	8.00 µS
400	4.00 µS	8.00 µS	16.00 µS
500	5.00 µS	10.00 µS	20.00 µS
600	6.00 µS	12.00 µS	24.00 µS
700	7.00 µS	14.00 µS	28.00 µS
800	8.00 µS	16.00 µS	32.00 µS
900	9.00 µS	18.00 µS	36.00 µS
1000	10.00 µS	20.00 µS	40.00 µS

Setting up ADC

- Set with 100ns sample window, ADC clock = System Clock, sampling frequency = 2 x Fepwm
- ADC conversion triggers an interrupt. This means that the interrupt service routine periodically runs after every ADC conversion. You will want to read the ADCRESULT into a variable in the interrupt in order to see what it is.

Registers:

ADCCTL2: control ADC clock (and other settings)

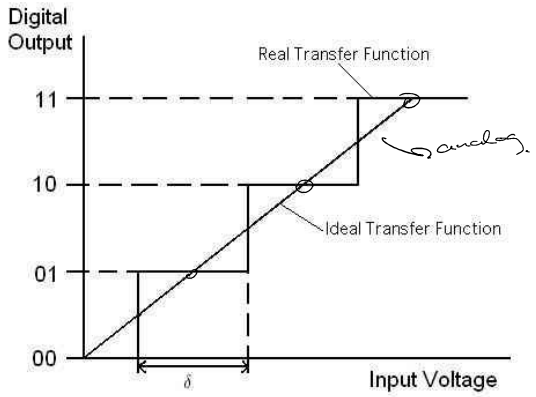
- Prescale: set such that ADC clock = System Clock

ADCSOCxCTL: control ADC start of conversion trigger and channel

- CHSEL: used to select ADC channel (e.g. ADCINA3)
- ACQPS: used to select ADC sample window based on number of cycles of ADC clock. Set to achieve 100ns sample window.
- TRIGSEL: used to select trigger for ADC SOC, use EPWM1.

Interrupt for ADCA should already be set up in the sample code. No need to worry about this.

Test ADC using DC power supply ---------->



Setting up DAC:

Set up Digital Analog Converter to output the SAME values as the ADC input. This can test your ADC, including sampling window and period.

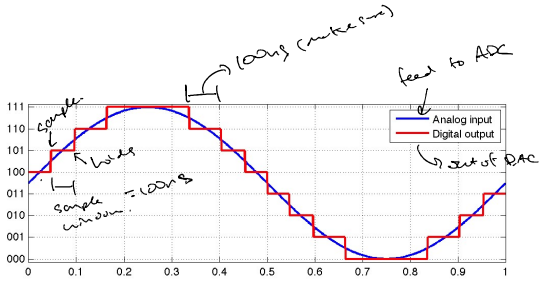
Registers:

DACCTL: Control settings for DAC

- DACREFSEL: 0b1 to match ADC reference
- LOADMODE: 0b0 to update DAC value on SYSCLK
- MODE: 0b1

DACOUTEN: enable DAC output, set to 1

DACVALS: clear to start, this is the value loaded into the DAC (numerical). Set this value in the ADC interrupt service routine to the same value as the ADCRESULT in order to pass directly to output.



Feeding an analog signal to the ADC to validate with the DAC:

- Use function generator (3V 1kHZ sine wave)
- Make sure grounds are connected, avoid shorting ground to power pins on the MCU
- Start function generator at ~0V. Turn on, and increase the voltage slowly.
- Probe DAC and input function and observe.
- If you use OWON scope, use RC filter

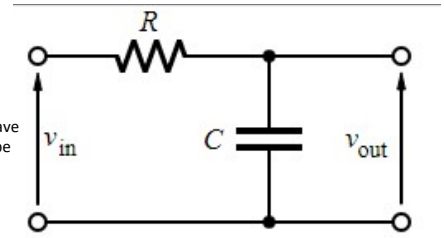
Using the OWON Oscilloscope:

- USB oscilloscope
- Use normal oscilloscope probes
- Only ever reference (alligator clip) to ground.
- Must download software for VDS1022I: https://www.owon.com.hk/products_owon_vds_series_pc_oscilloscope

RC Filter to provide Analog Signal:

- Use R = 100kOhm, C = 0.01uF
  - Selected by simulating in PLECS, and I had a bunch of 0.01uF capacitors. R high to limit current
- Produce Triangle Wave from 1.9-3.1V
- Make sure grounds connected correctly (ground bottom of capacitor and the bottom pin of the square wave generator)

Square wave From scope



Triangle wave to ADC