F28069 ControlCard Lab2_2

976 Hz Sinusoidal Signal
carrier: 500 kHz ePWM1A Signal

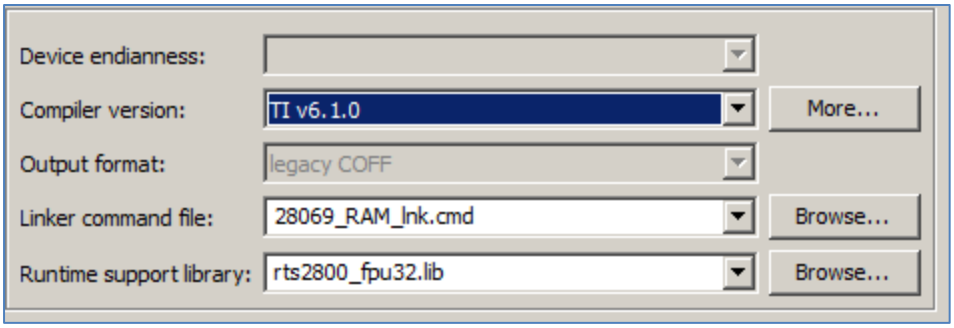
1. Project Dependencies

The project expects the following support files:

- Support files of controlSUITE installed in:

  C:\TI\controlSUITE\device_support\f28069\v135

- Code Generation Tools Version 6.1.0:



- CodeComposerStudio Version 5.3.0:

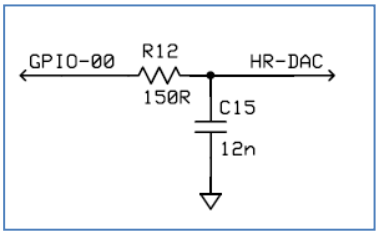
2. Project Objective

- Generate a 500 kHz PWM signal at ePWM1A (GPIO00) of the F28069 controlCARD Docking Station.
- symmetric PWM mode
- Modulate pulse width of ePWM1A as sinusoidal signal
- Access to I2Q30 sine look-up table in ROM (address 0x3FE000); 512 values for 360 degree
- Watchdog unit enabled and serviced in endless loop of "main()".

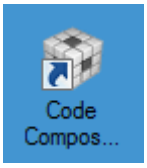
3. Hardware Setup

No special setup required. Connect the F28069 Docking Station with a USB cable to your computer. Make sure the main switch SW1 is in position "USB". The LED LD1 (green) of the F28069 controlCARD should be "on".

The 500 kHz PWM – signal can be measured at pin GPIO00 of the F28069 Docking Station. To produce the 976 Hz sinusoidal signal (500 kHz / 512 = 976 Hz) an external low-pass RC – filter (not populated at the Docking Station) is necessary:



4. Start Code Composer Studio V5

Start Code Composer Studio V5: 

When CCS loads, a dialog box will prompt you for the location of a workspace folder. Use the default location for the workspace and click OK. This folder contains all CCS custom settings, which includes project settings and views when CCS is closed so that the same projects and settings will be available when CCS is opened again. The workspace is saved automatically when CCS is closed.

Note: The first time CCS opens a "Welcome to Code Composer Studio v5" page appears.

The term workbench refers to the desktop development environment. The workbench will open in the "CCS Edit" Perspective view. Notice the "CCS EDIT" icon in the upper right-hand corner. A perspective defines the initial layout views of the workbench windows, toolbars, and menus which are appropriate for a specific type of task (i.e. code development or debugging). The "CCS Edit" Perspective is used to create or build C/C++ or any other languages projects. A "Debug Perspective" view will automatically be enabled when the debug session is started.

5. Code Modification

5.1. Create a new source code “Lab2_2.c”

- In the “C/C++” perspective of project “Lab2”, open file “Lab2_1.c” and save it as “Lab2_2.c”.
- Exclude the file “Lab2_1.c” from build.

Because we will use the ePWM1A compare interrupt to update the pulse width, we have to install the PIE interrupt system of the F28069. First, **link** 3 more files to the project:

```
C:\TI\controlSUITE\device_support\f2806x\v135\F2806x_common\source\F2806x_DefaultIsr.c
C:\TI\controlSUITE\device_support\f2806x\v135\F2806x_common\source\F2806x_PieCtrl.c
C:\TI\controlSUITE\device_support\f2806x\v135\F2806x_common\source\F2806x_PieVect.c
```

At the beginning of “Lab2_2.c” add a new include command:

```
#include "IQmathLib.h"
```

Open the properties of project “Lab2” and go to “C2000 compiler”, “Include Options” and add the include search path:

C:\TI\controlSUITE\libs\math\IQmath\v160\include

Also at the beginning of “Lab2_2.c” add 3 more function prototypes:

```
extern void InitPieVectTable(void);
extern void InitPieCtrl(void);
interrupt void ePWM1A_compare_isr(void);
```

Define a global variable “sine_table”:

```
#pragma DATA_SECTION(sine_table,"IQmathTables");
_iq30 sine_table[512];
```

The data type “_iq30” is defined in “IQmathLib.h”; The symbol “IQmathTables” will be linked to address 0x3FE000 (the beginning of the sine look-up table).

Next, in “main()”, directly after the call of “InitSysCtrl()”, call:

```
InitPieCtrl();          // basic setup of PIE table
InitPieVectTable();     // copy default ISR's into PIE
EALLOW;
PieVectTable.EPWM1_INT = &ePWM1A_compare_isr;
EDIS;
```

The PIE – entry for ePWM1 is replaced by our own function “ePWM1A_compare_isr”.

Change the initialization for register “EPwm1Regs.TBPRD” to a 500 kHz period:

```
EPwm1Regs.TBPRD = 90;    // 500 kHz PWM frequency
// TBPRD = fcpu / (2* fpwm * CLKDIV * HSPCLKDIV)
// TBPRB = 90 MHz / (2 * 500 kHz *  1 * 1)
```

Add an initial duty cycle of 50%:

```
EPwm1Regs.CMPA.half.CMPA = 45;      // initial duty 50 %
```

Change the Action Qualification Register to:

```
EPwm1Regs.AQCTLA.all = 0x0060;
// CMPA up = set; CMPA down = clear
```

Add code to enable the "CMPA – down" match interrupt for ePWM1A:

```
EPwm1Regs.ETSEL.all = 0;
EPwm1Regs.ETSEL.bit.INTEN = 1;      // enable ePWM1 int
EPwm1Regs.ETSEL.bit.INTSEL = 5;     // CMPA down match
EPwm1Regs.ETPS.bit.INTPRD = 1;      // 1st event
```

Just before the "while(1)" – loop of "main()", enable the interrupt system:

```
PieCtrlRegs.PIEIER3.bit.INTx1 = 1;  // ePWM1
IER |= 4;                           // enable INT3
EINT;                               // global int enable
```

At the end of file "Lab2_2.c" add the interrupt service function for ePWM1A:

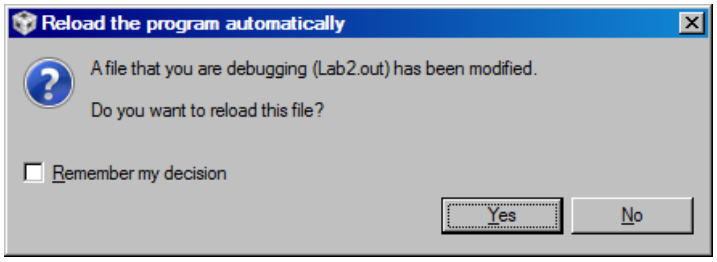
```
interrupt void ePWM1A_compare_isr(void)
{
      static unsigned int index = 0;
      EPwm1Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD -

      _IQsat(_IQ30mpy((sine_table[index]+_IQ30(0.9999))/2,
                        EPwm1Regs.TBPRD),EPwm1Regs.TBPRD,0);
      if (index++ >511) index = 0;
      EPwm1Regs.ETCLR.bit.INT = 1;  // clear ePWM1 int flag
      PieCtrlRegs.PIEACK.all = 4;   // ACK for PIE3 int
}
```

5.2. Rebuild Project

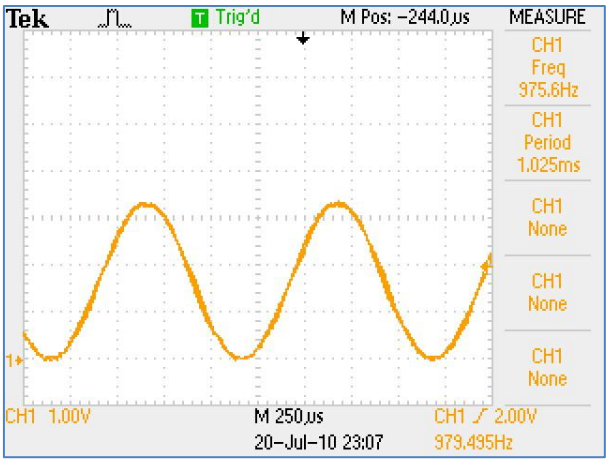
➔Project ➔ Build Project

A window will pop up to indicate that the machine code for the target has changed:



- Answer: "Yes". If the compilation and the download were successful, the instruction pointer (blue arrow) is set to the beginning of main.
- If the compilation and the download were successful, the instruction pointer (blue arrow) is again set to the beginning of main.
- Now perform a Run (F8)!

Use an oscilloscope to monitor the signal waveform at the output of the external low-pass filter:



6. Solution for Lab2_2.c

```
#include "F2806x_Device.h"
#include "IQmathLib.h"
extern void InitSysCtrl(void);
extern void InitPieVectTable(void);
extern void InitPieCtrl(void);
interrupt void ePWM1A_compare_isr(void);
#pragma DATA_SECTION(sine_table,"IQmathTables");
_iq30 sine_table[512];

void main(void)
{
       InitSysCtrl();
       InitPieCtrl();             // basic setup of PIE table
       InitPieVectTable(); // copy default ISR's into PIE
       EALLOW;
       PieVectTable.EPWM1_INT = &ePWM1A_compare_isr;
       SysCtrlRegs.WDCR = 0x00AF;
       GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;     // ePWM1A
       GpioCtrlRegs.GPAPUD.bit.GPIO0 = 0;      // enable pull-up
       EDIS;
       EPwm1Regs.TBCTL.all = 0; // default values
       EPwm1Regs.TBCTL.bit.CLKDIV = 0;  // CLKDIV = 1;
       EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0; // HSPCLKDIV = 1
       EPwm1Regs.TBCTL.bit.CTRMODE = 2;  // up-down mode
       EPwm1Regs.AQCTLA.all = 0x0006; // ZRO=set; PRD=clear
       EPwm1Regs.TBPRD = 90;    // 500 kHz PWM frequency
       // TBPRD = fcpu / (2* fpwm * CLKDIV * HSPCLKDIV)
       // TBPRB = 90 MHz / (2 * 500 kHz *  1 * 1)
       EPwm1Regs.CMPA.half.CMPA = 45;    // initial duty 50 %
       EPwm1Regs.AQCTLA.all = 0x0060;
       // CMPA up = set; CMPA down = clear
       EPwm1Regs.ETSEL.all = 0;
       EPwm1Regs.ETSEL.bit.INTEN = 1;    // enable ePWM1 int
       EPwm1Regs.ETSEL.bit.INTSEL = 5;   // CMPA down match
       EPwm1Regs.ETPS.bit.INTPRD = 1;    // 1st event
       PieCtrlRegs.PIEIER3.bit.INTx1 = 1;             // ePWM1
       IER |= 4;                         // enable INT3
       EINT;                             // global int enable
       while(1)
       {
              EALLOW;
              SysCtrlRegs.WDKEY = 0x55;  // service key #1
              SysCtrlRegs.WDKEY = 0xAA;  // service key #2
              EDIS;
       }
}

interrupt void ePWM1A_compare_isr(void)
{
       static unsigned int index = 0;
       EPwm1Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD -
       _IQsat(_IQ30mpy((sine_table[index]+_IQ30(0.9999))/2,
              EPwm1Regs.TBPRD),EPwm1Regs.TBPRD,0);
       if (index++ >511) index = 0;
       EPwm1Regs.ETCLR.bit.INT = 1;      // clear ePWM1 interrupt flag
       PieCtrlRegs.PIEACK.all = 4;    // ACK for PIE group 3 int

}
```