```
#include "F28x_Project.h"
void Sysctl(void);
void Init_Pwm(void);
__interrupt void EPwm1ISR(void);
__interrupt void EPwm2ISR(void);
int i,p,q;
long j;


int
SineTableA[]={851,1665,2406,3043,3546,3894,4072,4095,4072,3894,3546,3043,2406,1665,85
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int
SineTableB[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,851,1665,2406,3043,3546,3894,4072,4095,40
72,3894,3546,3043,2406,1665,851};
void main(void)

{




      DINT;                                // Mask interrupts for configuration
Sysctl();                            // Initialize system clock, watchdog
Init_Pwm();                                // Initialize PWM 1,2

EALLOW;
PieCtrlRegs.PIECTRL.bit.ENPIE=1;
PieVectTable.EPWM1_INT = &EPwm1ISR;  // directs ISR of EPWM 1
PieVectTable.EPWM2_INT = &EPwm2ISR;  // directs ISR of EPWM 2
PieCtrlRegs.PIEIER3.bit.INTx1 = 1;    // enable PIE interrupt - EPWM 1
PieCtrlRegs.PIEIER3.bit.INTx2 = 1;    // enable PIE interrupt - EPWM 2


PieCtrlRegs.PIEACK.all=0xffff;
EDIS;

IER =0x0fff;                                // Enable third group

EINT;                                 // Remove interrupt masking

while(1);



}


void Init_Pwm(void)

{


EALLOW;
```

```
ClkCfgRegs.PERCLKDIVSEL.bit.EPWMCLKDIV = 1;   //EPWMCLK=PLLSYSCLK/1

GpioCtrlRegs.GPAGMUX1.bit.GPIO0 = 0;      //Select PWM mode(ePWM1 module  GPIO0-
EPWM1A and GPIO1-EPWM1B)
GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;
GpioCtrlRegs.GPAGMUX1.bit.GPIO1 = 0;
GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1;

GpioCtrlRegs.GPAGMUX1.bit.GPIO2 = 0;        // select PWM mode in MUX(ePWM2 module
GPIO2-EPWM2A and GPIO3-EPWM2B)
GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 1;
GpioCtrlRegs.GPAGMUX1.bit.GPIO3 = 0;
GpioCtrlRegs.GPAMUX1.bit.GPIO3= 1;



CpuSysRegs.PCLKCR2.bit.EPWM1=1;             // Enable clock for PWM1
CpuSysRegs.PCLKCR2.bit.EPWM2=1;             // Enable clock for PWM2


CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 1;



// * EPWM 1 ** //

EPwm1Regs.TBCTL.bit.CLKDIV = 010;          //Divides the EPWMCLK by 4 (3 bits -
binary)
EPwm1Regs.TBCTL.bit.HSPCLKDIV = 101;       //Divides the EPWMCLK by 10 : TBCLK =
EPWMCLK / (HSPCLKDIV x CLKDIV)

EPwm1Regs.TBCTL.bit.CTRMODE = 2;        // 0 for up counter
EPwm1Regs.TBCTR = 0x0000;              // Clear counter
EPwm1Regs.TBPRD = 4095;                 // EPWM period

EPwm1Regs.AQCTLA.bit.CAU = 2;           // Action qualifier Count up
EPwm1Regs.AQCTLA.bit.CAD = 1;           // Action qualifier Count down
EPwm1Regs.AQCTLB.bit.CBU = 1;           // Action qualifier Count up
EPwm1Regs.AQCTLB.bit.CBD = 2;           // Action qualifier Count down


EPwm1Regs.ETSEL.bit.INTEN = 1;          // EPWM 1 interrupt enable
EPwm1Regs.ETSEL.bit.INTSEL = 1;
EPwm1Regs.ETPS.bit.INTPRD = 1;          // Interrupt at period
EPwm1Regs.ETCLR.bit.INT = 1;            // Clear interrupt

EPwm1Regs.DBCTL.bit.OUT_MODE=3;         // DBM Fully enabled RED & FED active.
EPwm1Regs.DBCTL.bit.POLSEL = 2;         //00 means independent; 01 means AL
Complimentary; 02 means Active high complimentary; PWMB is inverted

EPwm1Regs.DBFED.bit.DBFED=200;
EPwm1Regs.DBRED.bit.DBRED=200;


// * EPWM 2 ** //
```

```
EPwm2Regs.TBCTL.bit.CLKDIV = 010;          //Divides the EPWMCLK by 4 (3 bits - 
binary)
EPwm2Regs.TBCTL.bit.HSPCLKDIV = 101;       //Divides the EPWMCLK by 10 : TBCLK = 
EPWMCLK / (HSPCLKDIV x CLKDIV)2
EPwm2Regs.TBCTL.bit.CTRMODE = 2;        // 0 for up counte2
EPwm2Regs.TBCTR = 0x0000;              // Clear counter
EPwm2Regs.TBPRD = 4095;                 // EPWM period

EPwm2Regs.AQCTLA.bit.CAU = 2;           // Action qualifier Count up
EPwm2Regs.AQCTLA.bit.CAD = 1;           // Action qualifier Count down
EPwm2Regs.AQCTLB.bit.CBU = 1;           // Action qualifier Count up
EPwm2Regs.AQCTLB.bit.CBD = 2;           // Action qualifier  Count down


EPwm2Regs.ETSEL.bit.INTEN = 1;          // EPWM 1 interrupt enable
EPwm2Regs.ETSEL.bit.INTSEL = 1;
EPwm2Regs.ETPS.bit.INTPRD = 1;          // Interrupt at period
EPwm2Regs.ETCLR.bit.INT = 1;            // Clear interrupt

EPwm2Regs.DBCTL.bit.OUT_MODE=3;         // DBM Fully enabled RED & FED active.
EPwm2Regs.DBCTL.bit.POLSEL = 2;         //00 means independent; 01 means AL 
Complimentary; 02 means Active high complimentary; PWMB is inverted


EPwm2Regs.DBFED.bit.DBFED=200;
EPwm2Regs.DBRED.bit.DBRED=200;

EDIS;

}
void Sysctl(void)
{
    int i;
    EALLOW;
        ClkCfgRegs.CLKSRCCTL1.bit.OSCCLKSRCSEL=0; // primary oscillator select
        ClkCfgRegs.SYSPLLCTL1.bit.PLLCLKEN=0; // bypass pll
        for(i=0;i<120;i++);
        ClkCfgRegs.SYSCLKDIVSEL.bit.PLLSYSCLKDIV=0;   //wait for 120 oscillator cycle
        for(i=0;i<=4;i++)
        {
            ClkCfgRegs.SYSPLLCTL1.bit.PLLEN= 0;  // lock PLL
            ClkCfgRegs.SYSPLLMULT.bit.IMULT=16;  // 162Mhz PLL raw clock
            ClkCfgRegs.SYSPLLMULT.bit.FMULT=1;   //0.25 fractional mult
            while(ClkCfgRegs.SYSPLLSTS.bit.LOCKS != 1);
        }
        ClkCfgRegs.SYSCLKDIVSEL.bit.PLLSYSCLKDIV=1;  //Desired+1
        ClkCfgRegs.SYSPLLCTL1.bit.PLLCLKEN=1; // switch to PLL
        ClkCfgRegs.SYSCLKDIVSEL.bit.PLLSYSCLKDIV=0;   //divides by 1, 162Mhz clock
        ClkCfgRegs.LOSPCP.bit.LSPCLKDIV=1;  // 80Mhz LSP clock
        EDIS;

}
```

```
__interrupt void EPwm1ISR(void)

{

EPwm1Regs.ETCLR.bit.INT = 1;            // Clear interrupt

       if(j++>=30)j=0;
       {
        //EPwm1Regs.CMPA.bit.CMPA= p*q*SineTableA[j];
       EPwm1Regs.CMPA.bit.CMPA=0.2 *SineTableA[j];

        }

PieCtrlRegs.PIEACK.bit.ACK3 = 1; //acknowledge interrupt
}


__interrupt void EPwm2ISR(void)

{

EPwm2Regs.ETCLR.bit.INT = 1;            // Clear interrupt
              {

          EPwm2Regs.CMPA.bit.CMPA=0.2*SineTableB[j];
          }

PieCtrlRegs.PIEACK.bit.ACK3=1;//acknowledge interrupt

}
```