28335PWM_ADC

```
//###########################################################################
//16 通道  在定时器 0 中断中软件启动转换，同步采样 连续转换，级连模式；运行通过；定时器中断控制其采样频率，一个 T0 中断服务程序；
//     Watcn Variables:
//
//               Voltage1[128]         Last 128 ADCRESULT0 values
//               ConversionCount    Current result number 0-9
//               LoopCount             Idle loop counter
//###########################################################################
#include "DSP28x_Project.h"         // Device neaderfile and Examples Include File

// Global variables used in this example:
Uint16 LoopCount;
Uint16 ConversionCount;
Uint16 Voltage1[1024]={0};
Uint16 Voltage2[1024]={0};
#define ADC_MODCLK 0x3 // HSPCLK = SYSCLKOUT/2*ADC_MODCLK2 = 150/(2*3)     = 25.0 MHz
void InitAdc(void);
void InitEPwm1Example();

volatile unsigned int adconvover=0;

interrupt void epwm1_timer_isr(void);


Uint16   EPwm1TimerIntCount;
```

```
#define ADC_MODCLK 0x3 // nSPCLK = SYSCLKOUT/2*ADC_MODCLK2 = 150/(2*3)      =
25.0 Mnz

main()
{
    InitSysCtrl();

    InitGpio();    // Skipped for this example

    DINT;

    InitPieCtrl();

    IER = 0x0000;
    IFR = 0x0000;

    InitPieVectTable();
    EALLOW;    // This is needed to write to EALLOW protected register
    //PieVectTable.ADCINT = &adc_isr;
    PieVectTable.EPWM1_INT = &epwm1_timer_isr;

    EDIS;

    InitAdc();
    InitEPwm1Example();

    ConversionCount = 0;
    EPwm1TimerIntCount=0;

   // Enable ADCINT in PIE
    //PieCtrlRegs.PIEIER1.bit.INTx6 = 1;
    //IER |= M_INT1; // Enable CPU Interrupt 1

    PieCtrlRegs.PIEIER3.bit.INTx1 = 1;
    IER |= M_INT3;

    EINT;                 // Enable Global interrupt INTM
    ERTM;                  // Enable Global realtime interrupt DBGM

// Wait for ADC interrupt
    for(;;)
    {
```

```
    }

}

void InitEPwm1Example()
{
    EALLOW;
    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;// Disable TBCLK within the ePWM
    EDIS;


    EPwm1Regs.TBPRD = 18750;    //2ms                                   // Set timer    Period =600
TBCLK counts
    EPwm1Regs.CMPA.half.CMPA = 12500;                                 // Setup Compare A = 400
TBCLK counts
    EPwm1Regs.CMPB = 500;                                             // Compare B = 500 TBCLK
counts

    EPwm1Regs.TBPHS.half.TBPHS = 0x0000;                 // Phase is 0
    EPwm1Regs.TBCTR = 0x0000;                                   // Clear    TB counter

// Setup TBCLK = SYSCLKOUT / (HSPCLKDIV * CLKDIV)=150/4*4
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; //00 Up-count mode;01
Down-count mode;10 Up-down-count mode
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE; // Phase loading disabled
    EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
    EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
    EPwm1Regs.TBCTL.bit.HSPCLKDIV =0x05;//TB_DIV2;
    EPwm1Regs.TBCTL.bit.CLKDIV = 0x03;//TB_DIV2;

    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;          // Load registers every
ZERO
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;    // load on CTR = Zero
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;

// Set actions

    EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
    EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
    EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;
    EPwm1Regs.AQCTLB.bit.CBD = AQ_CLEAR;
```

```
    EPwm1Regs.ETSEL.bit.SOCAEN = 1;            // Enable SOC on A group
    EPwm1Regs.ETSEL.bit.SOCASEL = 2;           // Select SOC from from CPMA on upcount
    EPwm1Regs.ETPS.bit.SOCAPRD = 3;            // Generate pulse on 1st event
  // EPwm1Regs.TBCTL.bit.CTRMODE = 0;            // count up and start

    EPwm1Regs.ETSEL.bit.INTSEL = ET_CTR_PRD;        // Select INT on PRD event
    EPwm1Regs.ETSEL.bit.INTEN = 1;    // Enable INT
    EPwm1Regs.ETPS.bit.INTPRD = ET_1ST;

    EALLOW;
    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;// Enable TBCLK within the ePWM
    EDIS;
}

void InitAdc(void)
{
        Uint16 i;

         AdcRegs.ADCTRL1.bit.RESET=1; //复位 AD 模块
           for(i=0;i<9;i++)
         {asm(" NOP");}

         AdcRegs.ADCTRL1.bit.RESET=0;

         AdcRegs.ADCTRL1.bit.SUSMOD=0;    //Emulation suspend is ignored.

         AdcRegs.ADCTRL1.bit.ACQ_PS=15;    //设定窗口大小  SC=ADCTRL1[11:8] + 1 ; 16ADC CLOCKS
         AdcRegs.ADCTRL1.bit.CPS=1;        //ADCCLK = Fclk/1.Fclk = Prescaled nSPCLK (ADCCLKPS[3:0])

         AdcRegs.ADCTRL1.bit.CONT_RUN=1;// 0:Start-stop mode. 1:Continuous conversion mode.

         AdcRegs.ADCTRL1.bit.SEQ_CASC=1; //0 Dual-sequencer mode;
                                         //1 Cascaded mode 16 级联排序

         AdcRegs.ADCTRL3.bit.ADCBGRFDN=3;
         for(i=0;i<10000;i++)
         {asm(" NOP");}
         AdcRegs.ADCTRL3.bit.ADCPWDN=1;
         for(i=0;i<5000;i++)
         {asm(" NOP");}
```

AdcRegs.ADCTRL3.bit.ADCCLKPS=7; // ADC Core clock divider ：nSPCLK/[2*(ADCTRL1[7]：CPS + 1)]=12.5MnZ

//AdcRegs.ADCTRL3.bit.SMODE_SEL=0; //0 Sequential sampling mode is selected. 顺序采样

AdcRegs.ADCTRL3.bit.SMODE_SEL=1; //1 Simultaneous sampling mode is selected. 同步采样

//AdcRegs.ADCMAXCONV.bit.MAX_CONV1=0; //设定转换数量 1

AdcRegs.ADCMAXCONV.all=0x33; //ConversionCount=8dui;33:4dui

AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x0; // Setup conv from ADCINA0 & ADCINB0

AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x1; // Setup conv from ADCINA1 & ADCINB1

AdcRegs.ADCCHSELSEQ1.bit.CONV02 = 0x2; // Setup conv from ADCINA2 & ADCINB2

AdcRegs.ADCCHSELSEQ1.bit.CONV03 = 0x3; // Setup conv from ADCINA3 & ADCINB3

AdcRegs.ADCCHSELSEQ3.bit.CONV08 = 0x4; // Setup conv from ADCINA4 & ADCINB4

AdcRegs.ADCCHSELSEQ3.bit.CONV09 = 0x5; // Setup conv from ADCINA5 & ADCINB5

AdcRegs.ADCCHSELSEQ3.bit.CONV10 = 0x6; // Setup conv from ADCINA6 & ADCINB6

AdcRegs.ADCCHSELSEQ3.bit.CONV11 = 0x7; // Setup conv from ADCINA7 & ADCINB7*/

AdcRegs.ADCST.bit.INT_SEQ1_CLR=1;

AdcRegs.ADCST.bit.INT_SEQ2_CLR=1;

//AdcRegs.ADCTRL2.all=0x2800;

AdcRegs.ADCTRL2.bit.EPWM_SOCB_SEQ=0;

AdcRegs.ADCTRL2.bit.RST_SEQ1=0;// Restart sequencer 1

AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1=0;//1;//SEQ1 interrupt enable.

AdcRegs.ADCTRL2.bit.INT_MOD_SEQ1=0;

AdcRegs.ADCTRL2.bit.EPWM_SOCA_SEQ1=1;//ePWM SOCA enable bit for SEQ1

AdcRegs.ADCTRL2.bit.EXT_SOC_SEQ1=0;//External signal start-of-conversion bit for SEQ1

```
            AdcRegs.ADCTRL2.bit.RST_SEQ2=0;
            AdcRegs.ADCTRL2.bit.SOC_SEQ2=0;
            AdcRegs.ADCTRL2.bit.INT_ENA_SEQ2=0;
            AdcRegs.ADCTRL2.bit.INT_MOD_SEQ2=0;
            AdcRegs.ADCTRL2.bit.EPWM_SOCB_SEQ2=0;

//          AdcRegs.ADCTRL2.bit.SOC_SEQ1=1; //软件启动 AD 转化

}

interrupt void    epwm1_timer_isr(void)//1S 采样
{

 EPwm1TimerIntCount++;
if(EPwm1TimerIntCount==50)
{
   EPwm1TimerIntCount=0;
   //DELAY_US(5L);
   Voltage1[ConversionCount] = AdcRegs.ADCRESULT0 >>4;
   Voltage2[ConversionCount] = AdcRegs.ADCRESULT1 >>4;

   // If 40 conversions have been logged, start over
   if(ConversionCount == 1024)
   {
       ConversionCount = 0;
   }
   else ConversionCount++;
}
   //AdcRegs.ADCTRL2.bit.SOC_SEQ1=0;
   //EPwm1Regs.ETSEL.bit.SOCAEN = 0;

   // Reinitialize for next ADC sequence
   AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;               // Reset SEQ1

     // Clear INT flag for this timer
     EPwm1Regs.ETCLR.bit.INT = 1;

     PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
  // AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;            // Clear INT SEQ1 bit
   //PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;      // Acknowledge interrupt to PIE
   return;
}
```

相信对你有帮助的：

[最实惠的 f28335 系列开发板]

[基于 TMS320F28335 的 SVPWM 实现方法]

[TMS320F28335 入门视频教程之 PWM 脉冲调制详述——研旭原创]

[28335 pwm 介绍]

介绍 **dsp** 知识，为大家提供最新的 **dsp** 资讯，更多内容可以去南京研旭电气科技有限公司的官网 **www.njyxdq.com** **www.f28335.com** 或者官方论坛，嵌嵌 **dsp** 论坛 **www.armdsp.net** 进行交流学习

欢迎大家收听嵌嵌 **dsp** 论坛的官方微博

http://t.qq.com/qianqiandsp

还需要什么 **dsp** 资料欢迎加 **QQ：1318571484**