Dsp28335 中断的方法

步骤 1

```
void InitPieCtrl(void)    //初化 PIE 控制寄存器
{
      DINT; //关闭所有 CPU 标准中断 Disable Interrupts at the CPU level:
      PieCtrlRegs.PIECTRL.bit.ENPIE = 0; //  关闭所有 PIE 中断 Disable the PIE

       // Clear all PIEIER registers:  清除所有中断使能位
       PieCtrlRegs.PIEIER1~12(省了中间部分).all = 0;
       // Clear all PIEIFR registers:  清除所有中断标志位
       PieCtrlRegs.PIEIFR1~12.all = 0;
}
```

步骤 2

```
   // Disable CPU interrupts and clear all CPU interrupt flags:
     IER = 0x0000;
     IFR = 0x0000;
```

步骤 3 初始化中断向量表

```
void InitPieVectTable(void)
{      int16 i;
       Uint32 *Source = (void *) &PieVectTableInit;
       Uint32 *Dest = (void *) &PieVectTable;
       EALLOW;
       for(i=0; i < 128; i++)
             *Dest++ = *Source++;
       EDIS;
       // Enable the PIE Vector Table
       PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
```

```
}
//下面这个原中断数据地向量位置
const struct PIE_VECT_TABLE PieVectTableInit = {
        PIE_RESERVED,   // 0    Reserved space
        PIE_RESERVED,   // 1    Reserved space
        PIE_RESERVED,   // 2    Reserved space
        PIE_RESERVED,   // 3    Reserved space
        PIE_RESERVED,   // 4    Reserved space
        PIE_RESERVED,   // 5    Reserved space
        PIE_RESERVED,   // 6    Reserved space
        PIE_RESERVED,   // 7    Reserved space
        PIE_RESERVED,   // 8    Reserved space
        PIE_RESERVED,   // 9    Reserved space
        PIE_RESERVED,   // 10 Reserved space
        PIE_RESERVED,   // 11 Reserved space
        PIE_RESERVED,   // 12 Reserved space
// Non-Peripheral Interrupts
        INT13_ISR,      // XINT13 or CPU-Timer 1
        INT14_ISR,      // CPU-Timer2
        DATALOG_ISR,    // Datalogging interrupt
        RTOSINT_ISR,    // RTOS interrupt
        EMUINT_ISR,     // Emulation interrupt
        NMI_ISR,        // Non-maskable interrupt
        ILLEGAL_ISR,    // Illegal operation TRAP
        USER1_ISR,      // User Defined trap 1
        USER2_ISR,      // User Defined trap 2
        USER3_ISR,      // User Defined trap 3
        USER4_ISR,      // User Defined trap 4
        USER5_ISR,      // User Defined trap 5
        USER6_ISR,      // User Defined trap 6
        USER7_ISR,      // User Defined trap 7
        USER8_ISR,      // User Defined trap 8
        USER9_ISR,      // User Defined trap 9
        USER10_ISR,     // User Defined trap 10
        USER11_ISR,     // User Defined trap 11
        USER12_ISR,     // User Defined trap 12
// Group 1 PIE Vectors
        SEQ1INT_ISR,    // 1.1 ADC
        SEQ2INT_ISR,    // 1.2 ADC
        rsvd_ISR,       // 1.3
        XINT1_ISR,      // 1.4
        XINT2_ISR,      // 1.5
        ADCINT_ISR,     // 1.6 ADC
```

```
        TINT0_ISR,          // 1.7 Timer 0
        WAKEINT_ISR,        // 1.8 WD, Low Power
// Group 2 PIE Vectors
        EPWM1_TZINT_ISR, // 2.1 EPWM-1 Trip Zone
        EPWM2_TZINT_ISR, // 2.2 EPWM-2 Trip Zone
        EPWM3_TZINT_ISR, // 2.3 EPWM-3 Trip Zone
        EPWM4_TZINT_ISR, // 2.4 EPWM-4 Trip Zone
        EPWM5_TZINT_ISR, // 2.5 EPWM-5 Trip Zone
        EPWM6_TZINT_ISR, // 2.6 EPWM-6 Trip Zone
        rsvd_ISR,           // 2.7
        rsvd_ISR,           // 2.8
        // Group 3 PIE Vectors
        EPWM1_INT_ISR,      // 3.1 EPWM-1 Interrupt
        EPWM2_INT_ISR,      // 3.2 EPWM-2 Interrupt
        EPWM3_INT_ISR,      // 3.3 EPWM-3 Interrupt
        EPWM4_INT_ISR,      // 3.4 EPWM-4 Interrupt
        EPWM5_INT_ISR,      // 3.5 EPWM-5 Interrupt
        EPWM6_INT_ISR,      // 3.6 EPWM-6 Interrupt
        rsvd_ISR,           // 3.7
        rsvd_ISR,           // 3.8
        // Group 4 PIE Vectors
        ECAP1_INT_ISR,      // 4.1 ECAP-1
        ECAP2_INT_ISR,      // 4.2 ECAP-2
        ECAP3_INT_ISR,      // 4.3 ECAP-3
        ECAP4_INT_ISR,      // 4.4 ECAP-4
        ECAP5_INT_ISR,      // 4.5 ECAP-5
        ECAP6_INT_ISR,      // 4.6 ECAP-6
        rsvd_ISR,           // 4.7
        rsvd_ISR,           // 4.8
        // Group 5 PIE Vectors
        EQEP1_INT_ISR,      // 5.1 EQEP-1
        EQEP2_INT_ISR,      // 5.2 EQEP-2
        rsvd_ISR,           // 5.3
        rsvd_ISR,           // 5.4
        rsvd_ISR,           // 5.5
        rsvd_ISR,           // 5.6
        rsvd_ISR,           // 5.7
        rsvd_ISR,           // 5.8
// Group 6 PIE Vectors
        SPIRXINTA_ISR,      // 6.1 SPI-A
        SPITXINTA_ISR,      // 6.2 SPI-A
        MRINTA_ISR,         // 6.3 McBSP-A
        MXINTA_ISR,         // 6.4 McBSP-A
```

```
        MRINTB_ISR,         // 6.5 McBSP-B
        MXINTB_ISR,         // 6.6 McBSP-B
        rsvd_ISR,           // 6.7
        rsvd_ISR,           // 6.8
// Group 7 PIE Vectors
        DINTCH1_ISR,        // 7.1  DMA channel 1
        DINTCH2_ISR,        // 7.2  DMA channel 2
        DINTCH3_ISR,        // 7.3  DMA channel 3
        DINTCH4_ISR,        // 7.4  DMA channel 4
        DINTCH5_ISR,        // 7.5  DMA channel 5
        DINTCH6_ISR,        // 7.6  DMA channel 6
        rsvd_ISR,           // 7.7
        rsvd_ISR,           // 7.8
// Group 8 PIE Vectors
        I2CINT1A_ISR,       // 8.1  I2C
        I2CINT2A_ISR,       // 8.2  I2C
        rsvd_ISR,           // 8.3
        rsvd_ISR,           // 8.4
        SCIRXINTC_ISR,      // 8.5  SCI-C
        SCITXINTC_ISR,      // 8.6  SCI-C
        rsvd_ISR,           // 8.7
        rsvd_ISR,           // 8.8
        // Group 9 PIE Vectors
        SCIRXINTA_ISR,      // 9.1 SCI-A
        SCITXINTA_ISR,      // 9.2 SCI-A
        SCIRXINTB_ISR,      // 9.3 SCI-B
        SCITXINTB_ISR,      // 9.4 SCI-B
        ECAN0INTA_ISR,      // 9.5 eCAN-A
        ECAN1INTA_ISR,      // 9.6 eCAN-A
        ECAN0INTB_ISR,      // 9.7 eCAN-B
        ECAN1INTB_ISR,      // 9.8 eCAN-B
        // Group 10 PIE Vectors
        rsvd_ISR,           // 10.1
        rsvd_ISR,           // 10.2
        rsvd_ISR,           // 10.3
        rsvd_ISR,           // 10.4
        rsvd_ISR,           // 10.5
        rsvd_ISR,           // 10.6
        rsvd_ISR,           // 10.7
        rsvd_ISR,           // 10.8
        // Group 11 PIE Vectors
        rsvd_ISR,           // 11.1
        rsvd_ISR,           // 11.2
```

```
        rsvd_ISR,           // 11.3
        rsvd_ISR,           // 11.4
        rsvd_ISR,           // 11.5
        rsvd_ISR,           // 11.6
        rsvd_ISR,           // 11.7
        rsvd_ISR,           // 11.8
        // Group 12 PIE Vectors
        XINT3_ISR,          // 12.1
        XINT4_ISR,          // 12.2
        XINT5_ISR,          // 12.3
        XINT6_ISR,          // 12.4
        XINT7_ISR,          // 12.5
        rsvd_ISR,           // 12.6
        LVF_ISR,            // 12.7
        LUF_ISR,            // 12.8
};
```

这是中断目的向量表

```
struct PIE_VECT_TABLE {
// Reset is never fetched from this table.
// It will always be fetched from 0x3FFFC0 in
// boot ROM
        PINT    PIE1_RESERVED;
        PINT    PIE2_RESERVED;
        PINT    PIE3_RESERVED;
        PINT    PIE4_RESERVED;
        PINT    PIE5_RESERVED;
        PINT    PIE6_RESERVED;
        PINT    PIE7_RESERVED;
        PINT    PIE8_RESERVED;
        PINT    PIE9_RESERVED;
        PINT    PIE10_RESERVED;
        PINT    PIE11_RESERVED;
        PINT    PIE12_RESERVED;
        PINT    PIE13_RESERVED;
// Non-Peripheral Interrupts:
        PINT    XINT13;     // XINT13 / CPU-Timer1
        PINT    TINT2;      // CPU-Timer2
        PINT    DATALOG;    // Datalogging interrupt
        PINT    RTOSINT;    // RTOS interrupt
        PINT    EMUINT;     // Emulation interrupt
        PINT    XNMI;       // Non-maskable interrupt
        PINT    ILLEGAL;    // Illegal operation TRAP
        PINT    USER1;      // User Defined trap 1
```

```
        PINT    USER2;      // User Defined trap 2
        PINT    USER3;      // User Defined trap 3
        PINT    USER4;      // User Defined trap 4
        PINT    USER5;      // User Defined trap 5
        PINT    USER6;      // User Defined trap 6
        PINT    USER7;      // User Defined trap 7
        PINT    USER8;      // User Defined trap 8
        PINT    USER9;      // User Defined trap 9
        PINT    USER10;     // User Defined trap 10
        PINT    USER11;     // User Defined trap 11
        PINT    USER12;     // User Defined trap 12
// Group 1 PIE Peripheral Vectors:
        PINT    SEQ1INT;
        PINT    SEQ2INT;
        PINT    rsvd1_3;
        PINT    XINT1;
        PINT    XINT2;
        PINT    ADCINT;     // ADC
        PINT    TINT0;      // Timer 0
        PINT    WAKEINT;    // WD
// Group 2 PIE Peripheral Vectors:
        PINT    EPWM1_TZINT; // EPWM-1
        PINT    EPWM2_TZINT; // EPWM-2
        PINT    EPWM3_TZINT; // EPWM-3
        PINT    EPWM4_TZINT; // EPWM-4
        PINT    EPWM5_TZINT; // EPWM-5
        PINT    EPWM6_TZINT; // EPWM-6
        PINT    rsvd2_7;
        PINT    rsvd2_8;
// Group 3 PIE Peripheral Vectors:
        PINT    EPWM1_INT;  // EPWM-1
        PINT    EPWM2_INT;  // EPWM-2
        PINT    EPWM3_INT;  // EPWM-3
        PINT    EPWM4_INT;  // EPWM-4
        PINT    EPWM5_INT;  // EPWM-5
        PINT    EPWM6_INT;  // EPWM-6
        PINT    rsvd3_7;
        PINT    rsvd3_8;
// Group 4 PIE Peripheral Vectors:
        PINT    ECAP1_INT; // ECAP-1
        PINT    ECAP2_INT; // ECAP-2
        PINT    ECAP3_INT; // ECAP-3
        PINT    ECAP4_INT; // ECAP-4
```

```
        PINT        ECAP5_INT; // ECAP-5
        PINT        ECAP6_INT; // ECAP-6
        PINT        rsvd4_7;
        PINT        rsvd4_8;
// Group 5 PIE Peripheral Vectors:
        PINT        EQEP1_INT; // EQEP-1
        PINT        EQEP2_INT; // EQEP-2
        PINT        rsvd5_3;
        PINT        rsvd5_4;
        PINT        rsvd5_5;
        PINT        rsvd5_6;
        PINT        rsvd5_7;
        PINT        rsvd5_8;
// Group 6 PIE Peripheral Vectors:
        PINT        SPIRXINTA; // SPI-A
        PINT        SPITXINTA; // SPI-A
        PINT        MRINTB;       // McBSP-B
        PINT        MXINTB;       // McBSP-B
         PINT        MRINTA;       // McBSP-A
        PINT        MXINTA;       // McBSP-A
        PINT        rsvd6_7;
        PINT        rsvd6_8;
// Group 7 PIE Peripheral Vectors:
        PINT        DINTCH1;     // DMA
        PINT        DINTCH2;     // DMA
        PINT        DINTCH3;     // DMA
        PINT        DINTCH4;     // DMA
        PINT        DINTCH5;     // DMA
        PINT        DINTCH6;     // DMA
        PINT        rsvd7_7;
        PINT        rsvd7_8;
// Group 8 PIE Peripheral Vectors:
        PINT        I2CINT1A;    // I2C-A
        PINT        I2CINT2A;    // I2C-A
        PINT        rsvd8_3;
        PINT        rsvd8_4;
        PINT        SCIRXINTC;    // SCI-C
        PINT        SCITXINTC;    // SCI-C
        PINT        rsvd8_7;
        PINT        rsvd8_8;
// Group 9 PIE Peripheral Vectors:
        PINT        SCIRXINTA;    // SCI-A
        PINT        SCITXINTA;    // SCI-A
```

```
        PINT    SCIRXINTB;  // SCI-B
        PINT    SCITXINTB;  // SCI-B
        PINT    ECAN0INTA;  // eCAN-A
        PINT    ECAN1INTA;  // eCAN-A
        PINT    ECAN0INTB;  // eCAN-B
        PINT    ECAN1INTB;  // eCAN-B
// Group 10 PIE Peripheral Vectors:
        PINT    rsvd10_1;
        PINT    rsvd10_2;
        PINT    rsvd10_3;
        PINT    rsvd10_4;
        PINT    rsvd10_5;
        PINT    rsvd10_6;
        PINT    rsvd10_7;
        PINT    rsvd10_8;
// Group 11 PIE Peripheral Vectors:
        PINT    rsvd11_1;
        PINT    rsvd11_2;
        PINT    rsvd11_3;
        PINT    rsvd11_4;
        PINT    rsvd11_5;
        PINT    rsvd11_6;
        PINT    rsvd11_7;
        PINT    rsvd11_8;
// Group 12 PIE Peripheral Vectors:
        PINT    XINT3;      // External interrupt
        PINT    XINT4;
        PINT    XINT5;
        PINT    XINT6;
        PINT    XINT7;
        PINT    rsvd12_6;
        PINT    LVF;        // Latched overflow
        PINT    LUF;        // Latched underflow
};
//---------------------------------------------------------------------------
// PIE Interrupt Vector Table External References & Function Declarations:
//
extern struct PIE_VECT_TABLE PieVectTable;
实例说明
void main(void)
{
    Uint16 i;
    InitSysCtrl();  //初始化系统
```

```
    DINT;
    InitPieCtrl(); //初始化 PIE
    IER = 0x0000;
    IFR = 0x0000;
    InitPieVectTable();//实始化中断向量表
    EALLOW;
    PieVectTable.DINTCH1= &local_DINTCH1_ISR; //这个对就是 INT7.Y  的第一个位 DMA
    EDIS;
    IER = M_INT7 ;    //M_INT7=0x0040=0100,0000
                      //打开 INT7.Y 这个很重要
    EnableInterrupts();
}
void EnableInterrupts()
{
      // Enable the PIE
      PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
       // Enables PIE to drive a pulse into the CPU
       PieCtrlRegs.PIEACK.all = 0xFFFF;
       // Enable Interrupts at the CPU level
      EINT;
}

// INT7.1
interrupt void local_DINTCH1_ISR(void)         // DMA Channel 1
{
   // To receive more interrupts from this PIE group, acknowledge this interrupt
    PieCtrlRegs.PIEACK.all |= PIEACK_GROUP7;
   // Next two lines for debug only to halt the processor here
   // Remove after inserting ISR Code
    asm ("          ESTOP0");
    for(;;);
}
```

Table 6-4. PIE MUXed Peripheral Interrupt Vector Table

	INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
INT1.y	*WAKEINT*	*TINT0*	*ADCINT*	*XINT2*	*XINT1*	Reserved	*SEQ2INT*	*SEQ1INT*
	(LPM/WD)	(TIMER 0)	(ADC)	Ext. int. 2	Ext. int. 1	-	(ADC)	(ADC)
	0xD4E	0xD4C	0xD4A	0xD48	0xD46	0xD44	0xD42	0xD40
INT2.y	Reserved	Reserved	*EPWM6_ TZINT*	*EPWM5_TZINT*	*EPWM4_TZINT*	*EPWM3_TZINT*	*EPWM2_TZINT*	*EPWM1_TZINT*
	-	-	(ePWM6)	(ePWM5)	(ePWM4)	(ePWM3)	(ePWM2)	(ePWM1)
	0xD5E	0xD5C	0xD5A	0xD58	0xD56	0xD54	0xD52	0xD50
INT3.y	Reserved	Reserved	*EPWM6_ INT*	*EPWM5_INT*	*EPWM4_INT*	*EPWM3_INT*	*EPWM2_INT*	*EPWM1_INT*
	-	-	(ePWM6)	(ePWM5)	(ePWM4)	(ePWM3)	(ePWM2)	(ePWM1)
	0xD6E	0xD6C	0xD6A	0xD68	0xD66	0xD64	0xD62	0xD60
INT4.y	Reserved	Reserved	*ECAP6_INT*	*ECAP5_INT*	*ECAP4_INT*	*ECAP3_INT*	*ECAP2_INT*	*ECAP1_INT*
	-	-	(eCAP6)	(eCAP5)	(eCAP4)	(eCAP3)	(eCAP2)	(eCAP1)
	0xD7E	0xD7C	0xD7A	0xD78	0xD76	0xD74	0xD72	0xD70
INT5.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	*EQEP2_INT*	*EQEP1_INT*
	-	-	-	-	-	-	(eQEP2)	(eQEP1)
	0xD8E	0xD8C	0xD8A	0xD88	0xD86	0xD84	0xD82	0xD80
INT6.y	Reserved	Reserved	*MXINTA*	*MRINTA*	*MXINTB*	*MRINTB*	*SPITXINTA*	*SPIRXINTA*
			(McBSP-A)	(McBSP-A)	(McBSP-B)	(McBSP-B)	(SPI-A)	(SPI-A)
	0xD9E	0xD9C	0xD9A	0xD98	0xD96	0xD94	0xD92	0xD90
INT7.y	Reserved	Reserved	*DINTCH6*	*DINTCH5*	*DINTCH4*	*DINTCH3*	*DINTCH2*	*DINTCH1*
			(DMA6)	(DMA5)	(DMA4)	(DMA3)	(DMA2)	(DMA1)
	0xDAE	0xDAC	0xDAA	0xDA8	0xDA6	0xDA4	0xDA2	0xDA0
INT8.y	Reserved	Reserved	*SCITXINTC*	*SCIRXINTC*	Reserved	Reserved	*I2CINT2A*	*I2CINT1A*
	-	-	(SCI-C)	(SCI-C)	-	-	(I2C-A)	(I2C-A)
	0xDBE	0xDBC	0xDBA	0xDB8	0xDB6	0xDB4	0xDB2	0xDB0
INT9.y	*ECAN1INTB*	*ECAN0INTB*	*ECAN1INTA*	*ECAN0INTA*	*SCITXINTB*	*SCIRXINTB*	*SCITXINTA*	*SCIRXINTA*
	(CAN-B)	(CAN-B)	(CAN-A)	(CAN-A)	(SCI-B)	(SCI-B)	(SCI-A)	(SCI-A)
	0xDCE	0xDCC	0xDCA	0xDC8	0xDC6	0xDC4	0xDC2	0xDC0
INT10.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	0xDDE	0xDDC	0xDDA	0xDD8	0xDD6	0xDD4	0xDD2	0xDD0
INT11.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	0xDEE	0xDEC	0xDEA	0xDE8	0xDE6	0xDE4	0xDE2	0xDE0
INT12.y	*LUF*	*LVF*	Reserved	*XINT7*	*XINT6*	*XINT5*	*XINT4*	*XINT3*
	(FPU)	(FPU)		Ext. Int. 7	Ext. Int. 6	Ext. Int. 5	Ext. Int. 4	Ext. Int. 3
	0xDFE	0xDFC	0xDFA	0xDF8	0xDF6	0xDF4	0xDF2	0xDF0

SCI 串口的一般用法

步骤 1 初始化 IO 口为串口

```
void InitSciaGpio()
{
    EALLOW;
     GpioCtrlRegs.GPBPUD.bit.GPIO36 = 0;   //SCIRXDA
      GpioCtrlRegs.GPBPUD.bit.GPIO35 = 0;   //SCITXDA
     GpioCtrlRegs.GPBQSEL1.bit.GPIO36 = 3;   //SCIRXDA
      GpioCtrlRegs.GPBMUX1.bit.GPIO35 = 1;
     GpioCtrlRegs.GPBMUX1.bit.GPIO36 = 1;
     EDIS;
}
```

步骤 2 我在这里设置为中断方式及初始化程序

```
EALLOW;
    PieVectTable.SCIRXINTA = &sciaRxFifoIsra;
    PieVectTable.SCITXINTA = &sciaTxFifoIsra;
EDIS;
    scia_fifo_init();   // Init SCI-A 用 FIFO 模式
    PieCtrlRegs.PIECTRL.bit.ENPIE = 1;     // Enable the PIE block  使能总中断模块
    PieCtrlRegs.PIEIER9.bit.INTx1=1;       // PIE Group 9, int1    使能接收中断使能位
    //PieCtrlRegs.PIEIER9.bit.INTx2=1;       // PIE Group 9, INT2  使能发送中断使能位
    IER |= 0x100;                             // Enable CPU INT  打开 Group 9 所有可以使能中到 CPU
    EINT;
```

初始化 FIFO

```
void scia_fifo_init()
{
    SciaRegs.SCICCR.all =0x0007;     // 1 stop bit,    No loopback
                                        // No parity,8 char bits,
                                        // async mode 同步模式, idle-line protocol  空闲线协议
    SciaRegs.SCICTL1.all =0x0003;   // enable TX, RX, internal SCICLK,
                                        // Disable RX ERR, SLEEP, TXWAKE
    SciaRegs.SCICTL2.bit.TXINTENA =1; //使能发送中断
    SciaRegs.SCICTL2.bit.RXBKINTENA =1;//使能接收中断
    SciaRegs.SCIHBAUD = 0x0001;
    SciaRegs.SCILBAUD = 0x00E7;// 9600 baud @LSPCLK = 37.5MHz
    SciaRegs.SCICCR.bit.LOOPBKENA =0; // 1 Enable loop back 0 关闭自己测模式
    SciaRegs.SCIFFTX.all=0xC028;     //FIFO 小于或等于 8 个字节时发送中断产生
    SciaRegs.SCIFFRX.all=0x002F;     //FIFO 大于或等于 16 个字节时接收中断产生  (注意:  不大于时不会产
生中断)
   // SciaRegs.SCIFFCT.all=0x00;
```

```
    SciaRegs.SCICTL1.all =0x0023;        // Relinquish SCI from Reset  软件复位后使能中断
    SciaRegs.SCIFFTX.bit.TXFIFOXRESET=1; //发送 FIFO 复位
    SciaRegs.SCIFFRX.bit.RXFIFORESET=1;   //接收 FIFO 复位
}
```

步骤 3　编写中断处理程序

```
interrupt void sciaTxFifoIsra(void)
{     //中断处理内容
      SciaRegs.SCIFFTX.bit.TXFFINTCLR=1;      // Clear SCI Interrupt flag
      PieCtrlRegs.PIEACK.all|=0x100;          // Issue PIE ACK
}
//-------SCI 接收中断--------
interrupt void sciaRxFifoIsra(void)
{     Uint16 i;
      for(i=0;i<16;i++)
      {
          rdataB[i]=SciaRegs.SCIRXBUF.all;   // Read data
          //rdataB[i]=rdataA[i];
      }
      SciaRegs.SCIFFRX.bit.RXFFOVRCLR=1;     // Clear Overflow flag
      SciaRegs.SCIFFRX.bit.RXFFINTCLR=1;     // Clear Interrupt flag
      PieCtrlRegs.PIEACK.all|=0x100;          // Issue PIE ack
}
```

相信对你有帮助的：

dsp28335 之 GPIO

DSP28335SCI 总结

TMS320F28335 中文资料

ccs3.3(28335)如何新建并且配置工程