DSP 之时钟学习

先说软件设置 ：

时钟倍频 Initpll 在 DSP280X_SysCtrl.c 文件中的系统初始化函数 InitSysCtrl();中设置。

```
// - Enable the clocks to the peripherals

void InitSysCtrl(void)
{

   // Disable the watchdog
   DisableDog();

   // Initialize the PLL control: PLLCR and CLKINDIV
   // DSP28_PLLCR and DSP28_CLKINDIV are defined in DSP280x_Examples.h
   InitPll(DSP28_PLLCR,DSP28_CLKINDIV);

   // Initialize the peripheral clocks
   InitPeripheralClocks();
}
```

```
void InitSysCtrl(void)
{
   // Disable the watchdog
   DisableDog();  //首先应该关看门狗
   // Initialize the PLL control: PLLCR and CLKINDIV
   // DSP28_PLLCR and DSP28_CLKINDIV are defined in DSP280x_Examples.h
   InitPll(DSP28_PLLCR,DSP28_CLKINDIV);//PLL时钟设置
   InitPeripheralClocks();//外设时钟启动
}
```

//注：DSP28_PLLCR,DSP28_CLKINDIV这两个变量就是指定PLL倍频的数据，在DSP280x_Examples.h头文件中宏定义的，但是可以在此头文件中去修改这两个数值。如下就是宏定义的原型。

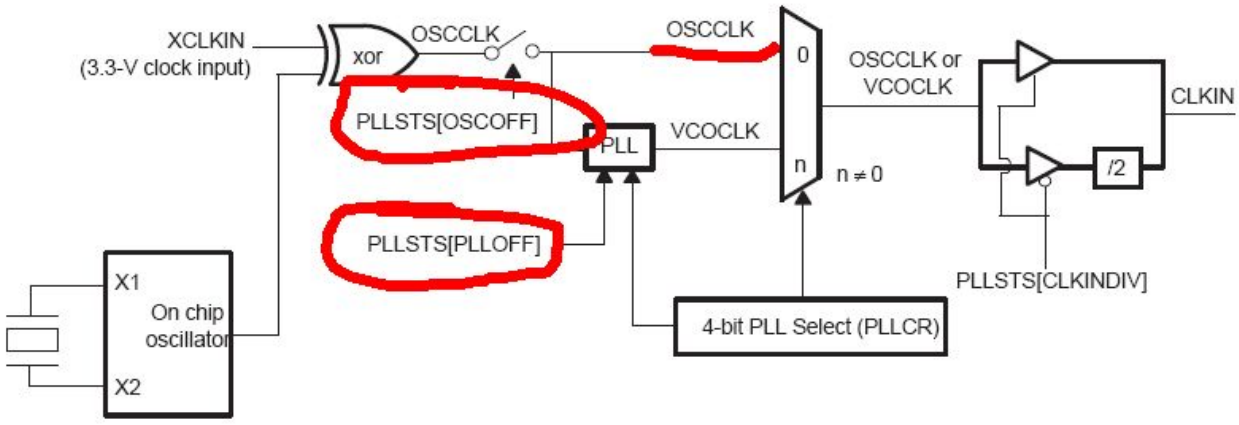
```
#define DSP28_CLKINDIV   0   // Enable /2 for SYSCLKOUT
//#define DSP28_CLKINDIV   1 // Disable /2 for SYSCKOUT

//#define DSP28_PLLCR   10
//#define DSP28_PLLCR    9
//#define DSP28_PLLCR    8
//#define DSP28_PLLCR    7
#define DSP28_PLLCR     6
//#define DSP28_PLLCR    5
//#define DSP28_PLLCR    4
//#define DSP28_PLLCR    3
//#define DSP28_PLLCR    2
//#define DSP28_PLLCR    1
//#define DSP28_PLLCR    0  // PLL is bypassed in this mode
   // Initialize the peripheral clocks
```

进出CPU时钟CLKIN可以是PLL关闭 可以是PLL旁路，还可以是PLL使能

3.2.1 PLL-Based Clock Module

The 280x devices have an on-chip, PLL-based clock module. The PLL has a 4-bit ratio control to select different CPU clock rates. Figure 3-8 shows the OSC and PLL block on the 280x.

Figure 3-8. OSC and PLL Block



The PLL-based clock module provides two modes of operation:

注：

- 晶振操作：

此模式允许使用外部为器件提供时基。晶体连接到X1/X2 引脚，XCLKIN 连接到低电平。

- 外部时钟源操作：

此模式允许绕过内部振荡器。器件时钟根据引脚上的外部时钟源输入生成。必须将X1 连接到低电平并让X2 悬空。在这种情况下，外部振荡器时钟连接到XCLKIN 引脚，允许使用3.3V 时钟源。

允许使用X1 和X2 引脚将晶体连接到280x 器件。如果未使用晶体，则可以将外部振荡器直接连接到XCLKIN 引脚，X2 引脚保留悬空，X1 连接到低电平。请参阅
TMS320F2808、TMS320F2806、TMS320F2801、UCD9501 数字信号处理器数据手册（文献编号SPRS230）

1：来自X1,X2或者XCLKIN的时钟可以通过控制寄存器PLL 状态寄存器(PLLSTS)来控制是否需要经过PLL倍频，

5	OSCOFF		振荡器时钟关闭位
		0	来自 X1/X2 或 XCLKIN 的 OSCCLK 信号馈送至 PLL 块。（默认值）
		1	来自 X1/X2 或 XCLKIN 的 OSCCLK 信号未馈送至 PLL 块。此模式对于测试主振荡器失败检测逻辑非常有用。此模式不关闭内部振荡器。

2	PLLOFF		PLL 关闭位 此位关闭 PLL. 有助于系统噪声测试。此模式应只在 PLLCR 寄存器设置为 0x0000 时使用。
		0	PLL 打开（默认值）
		1	PLL 关闭 设置了 PLLOFF 位时，PLL 模块应保持断电。器件必须处于 PLL 旁路模式 (PLLCR = 0x0000)，然后才能将 1 写入 PLLOFF。当 PLL 关闭 (PLLOFF = 1) 时，不要为 PLLCR 写入非零值。当 PLLOFF = 1 时，STANDBY 和 HALT 低功率模式将如预期正常工作。在从 HALT 或 STANDBY 唤醒之后，PLL 模块将继续断电。

当OSCOFF=0时来自X1,X2或者XCLKIN的时钟送至PLL，当OSCOFF=1时X1,X2或者XCLKIN的时钟不送PLL

2：a、简单的说就是若OSCOFF=1是关闭PLL，SYSCLKOUT直接就等于外部晶振或者晶体来的时钟OSCCLK或者是OSCCLK/2，具体是OSCCLK/2还是OSCCLK还要通过设置寄存器PLLSTS[CLKINDIV]位，如下图：

Table 3-7. Possible PLL Configuration Modes

PLL Mode	Remarks	PLLSTS[CLKINDIV](1)	SYSCLKOUT
PLL Off	Invoked by the user setting the PLLOFF bit in the PLLSTS register. The PLL block is disabled in this mode. This can be useful to reduce system noise and for low power operation. The PLLCR register must first be set to 0x0000 (PLL Bypass) before entering this mode. The CPU clock (CLKIN) is derived directly from the input clock on either X1/X2, X1 or XCLKIN.	0	OSCCLK/2
		1	OSCCLK
PLL Bypass	PLL Bypass is the default PLL configuration upon power-up or after an external reset (XRS). This mode is selected when the PLLCR register is set to 0x0000 or while the PLL locks to a new frequency after the PLLCR register has been modified. In this mode, the PLL itself is bypassed but the PLL is not turned off.	0	OSCCLK/2
		1	OSCCLK
PLL Enabled	Achieved by writing a non-zero value n into the PLLCR register. Upon writing to the PLLCR, the device will switch to PLL Bypass mode until the PLL locks.	0	OSCCLK*n/2
		1	OSCCLK*n

b、OSCOFF=0且PLLOFF=0时就是打开PLL，通过设置PLLCR寄存器来控制倍频的倍数，控制SYSCLKOUT的大小，见图PLLCR的值和CLKIN(SYSCLKOUT)对应关系：（还要设置PLLSTS[CLKINDIV]位）

Table 3-8. PLLCR Bit Descriptions

	SYSCLKOUT (CLKIN)(1)	
PLLCR[DIV] Value(2)	PLLSTS[CLKINDIV] = 0 (3)	PLLSTS[CLKINDIV] = 1 (3)
0000 (PLL bypass)	OSCCLK/2 (Default) (4)	OSCCLK
0001	(OSCCLK*1)/2	OSCCLK*1
0010	(OSCCLK*2)/2	OSCCLK*2
0011	(OSCCLK*3)/2	OSCCLK*3
0100	(OSCCLK*4)/2	OSCCLK*4
0101	(OSCCLK*5)/2	OSCCLK*5
0110	(OSCCLK*6)/2	OSCCLK*6
0111	(OSCCLK*7)/2	OSCCLK*7
1000	(OSCCLK*8)/2	OSCCLK*8
1001	(OSCCLK*9)/2	OSCCLK*9
1010	(OSCCLK*10)/2	OSCCLK*10
1011 - 1111	reserved	reserved

c、当OSCOFF=0且PLLOFF=1时就是PLL作为旁路使用，PLL 关闭设置了PLLOFF 位时，PLL 模块应保持断电。器件必须处于PLL 旁路模式(PLLCR = 0x0000)，然后才能将1 写入PLLOFF。当PLL 关闭(PLLOFF = 1) 时，不要为PLLCR 写入非零值。当PLLOFF = 1时，STANDBY 和HALT 低功率模式将如预期正常工作。在从HALT 或STANDBY 唤醒之后，PLL 模块将继续断电。

例：DSP280x_Examples.h头文件中

```
/*-----------------------------------------------------------------------------
      Specify the PLL control register (PLLCR) and clock in divide (CLKINDIV) value.

      if CLKINDIV = 0: SYSCLKOUT = (OSCCLK * PLLCR)/2
      if CLKINDIV = 1: SYSCLKOUT = (OSCCLK * PLLCR)
-----------------------------------------------------------------------------*/
#define DSP28_CLKINDIV   0   // Enable /2 for SYSCLKOUT
//#define DSP28_CLKINDIV   1 // Disable /2 for SYSCKOUT  设置输出的时钟二分频与否  0表示二分频  1表示不二分频。
//#define DSP28_PLLCR   10
//#define DSP28_PLLCR    9
//#define DSP28_PLLCR    8
//#define DSP28_PLLCR    7
```

```
#define DSP28_PLLCR    6
//#define DSP28_PLLCR    5
//#define DSP28_PLLCR    4
//#define DSP28_PLLCR    3
//#define DSP28_PLLCR    2
//#define DSP28_PLLCR    1
  //#define DSP28_PLLCR    0  // PLL is bypassed in this mode//旁路模式中使用
```

`void InitPll(Uint16 val, Uint16 clkindiv)` 函数也在 `DSP280X_SysCtrl.c` 中，

```
void InitPll(Uint16 val, Uint16 clkindiv)
{
   volatile Uint16 iVol;

   // Make sure the PLL is not running in limp mode
   if (SysCtrlRegs.PLLSTS.bit.MCLKSTS != 0)
   {
      // Missing external clock has been detected
      // Replace this line with a call to an appropriate
      // SystemShutdown(); function.
      asm("        ESTOP0");
   }

   // CLKINDIV MUST be 0 before PLLCR can be changed from
   // 0x0000. It is set to 0 by an external reset XRSn
   if (SysCtrlRegs.PLLSTS.bit.CLKINDIV != 0)
   {
        SysCtrlRegs.PLLSTS.bit.CLKINDIV = 0;
   }

   // Change the PLLCR
   if (SysCtrlRegs.PLLCR.bit.DIV != val)
   {

      EALLOW;
      // Before setting PLLCR turn off missing clock detect logic
      SysCtrlRegs.PLLSTS.bit.MCLKOFF = 1;
      SysCtrlRegs.PLLCR.bit.DIV = val;
      EDIS;
```

寄存器设置

Figure 3-3. Peripheral Clock Control 0 Register (PCLKCR0)

图 3-2./3-3. 外设时钟控制寄存器 0 (PCLKCR0/1)

Figure 3-3. Peripheral Clock Control 0 Register (PCLKCR0)

15	14	13	12	11	10	9	8
ECANBENCLK	ECANAENCLK	Reserved		SCIBENCLK	SCIAENCLK	SPIBENCLK	SPIAENCLK
R/W-0	R/W-0	R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
SPIDENCLK	SPICENCLK	Reserved	I2CAENCLK	ADCENCLK	TBCLKSYNC	Reserved	
R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 3-2. Peripheral Clock Control 0 Register (PCLKCR0) Field Descriptions

Bit	Field	Value	Description [1]
15	ECANBENCLK		ECAN-B Clock enable. This bit is reserved on devices without the eCAN-B module.[2]
		0	The eCAN-B module is not clocked. (default) [3]
		1	The eCAN-B module is clocked by the system clock (SYSCLKOUT).
14	ECANAENCLK		ECAN-A clock enable. This bit is reserved on devices without the eCAN-A module.
		0	The eCAN-A module is not clocked. (default) [3]
		1	The eCAN-A module is clocked by the system clock (SYSCLKOUT).
13-12	Reserved		Reserved
11	SCIBENCLK		SCI-B clock enable. This bit is reserved on devices without the SCI-B module.[2]
		0	SCI-B module is not clocked. (default) [3]
		1	The SCI-B module is clocked by the low-speed clock (LSPCLK).
10	SCIAENCLK		SCI-A clock enable. This bit is reserved on devices without the SCI-A module.
		0	The SCI-A module is not clocked. (default) [3]
		1	The SCI-A module is clocked by the low-speed clock (LSPCLK).
9	SPIBENCLK		SPI-B clock enable. This bit is reserved on devices without the SPI-B module.
		0	The SPI-B module is not clocked. (default) [3]
		1	The SPI-B module is clocked by the low-speed clock (LSPCLK).

Figure 3-4. Peripheral Clock Control 1 Register (PCLKCR1)

15	14	13	12	11	10	9	8
EQEP2ENCLK	EQEP1ENCLK	Reserved		ECAP4ENCLK	ECAP3ENCLK	ECAP2ENCLK	ECAP1ENCLK
R/W-0	R/W-0	R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
EPWM8ENCLK[1]	EPWM7ENCLK[1]	EPWM6ENCLK	EPWM5ENCLK	EPWM4ENCLK	EPWM3ENCLK	EPWM2ENCLK	EPWM1ENCLK
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

(1) Applicable for 28044 device only

Table 3-3. Peripheral Clock Control 1 Register (PCLKCR1) Field Descriptions

Bits	Field	Value	Description [1]
15	EQEP2ENCLK		eQEP2 clock enable. This bit is reserved on devices without the eQEP2 module.[2]
		0	The eQEP2 module is not clocked. (default) [3]
		1	The eQEP2 module is clocked by the system clock (SYSCLKOUT).
14	EQEP1ENCLK		eQEP1 clock enable. This bit is reserved on devices without the eQEP1 module.
		0	The eQEP1 module is not clocked. (default) [3]
		1	The eQEP1 module is clocked by the system clock (SYSCLKOUT).
13-12	Reserved		Reserved
11	ECAP4ENCLK		eCAP4 clock enable. This bit is reserved devices without the eCAP4 module.[2]
		0	The eCAP4 module is not clocked. (default) [3]

图 3-5. 高速外设时钟预分频器(HISPCP) 寄存器

HISPCP 位外围，并参阅图 3-6以了解 LOSPCP 外围。

图 3-5. 高速外设时钟预分频器 (HISPCP) 寄存器

15	3	2	0
保留		HSPCLK	
R-0		R/W-001	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-5. 高速外设时钟预分频器 (HISPCP) 字段说明

位	字段	值	说明(1)
15-3	保留		保留
2-0	HSPCLK		这些位配置高速外设时钟 (HSPCLK) 相对于 SYSCLKOUT 的比率： 如果 HISPCP(2)≠ 0，则 HSPCLK = SYSCLKOUT/(HISPCP X 2) 如果 HISPCP = 0，则 HSPCLK = SYSCLKOUT
		000	高速时钟 = SYSCLKOUT/1
		001	高速时钟 = SYSCLKOUT/2（复位默认值）
		010	高速时钟 = SYSCLKOUT/4
		011	高速时钟 = SYSCLKOUT/6
		100	高速时钟 = SYSCLKOUT/8
		101	高速时钟 = SYSCLKOUT/10
		110	高速时钟 = SYSCLKOUT/12
		111	高速时钟 = SYSCLKOUT/14

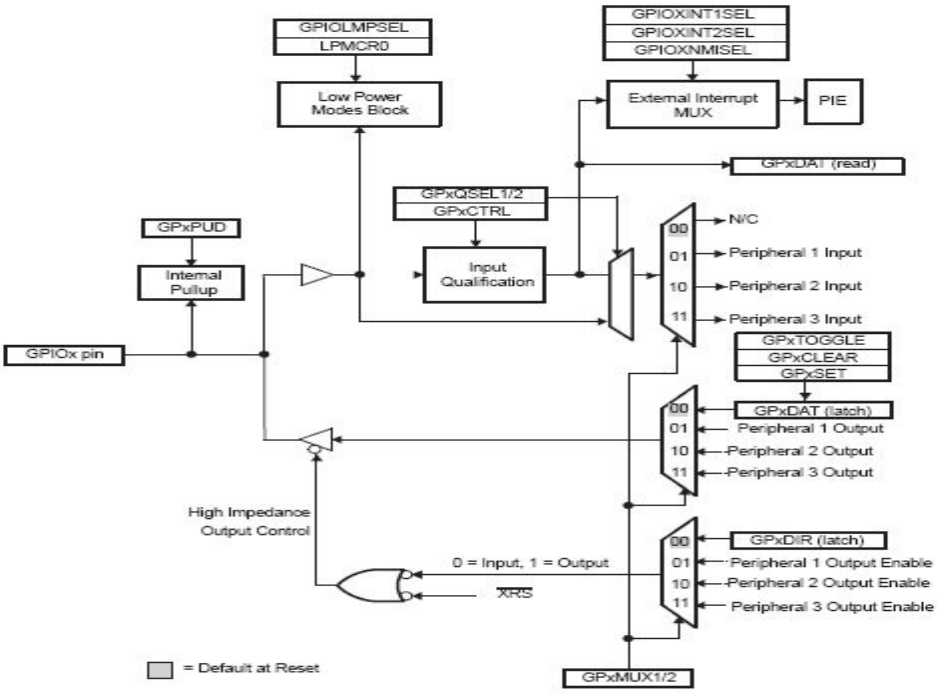
(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 此等式中的 HISPCP 代表 HISPCP 寄存器中位 2:0 的值。

```
// HISPCP/LOSPCP prescale register settings, normally it will be set to default values
   SysCtrlRegs.HISPCP.all = 0x001 ;//高速时钟HISPCP = SYSCLKOUT/2
   SysCtrlRegs.LOSPCP.all = 0x002;//低速时钟HISPCP = SYSCLKOUT/4

// XCLKOUT to SYSCLKOUT ratio.  By default XCLKOUT = 1/2 SYSCLKOUT
   SysCtrlRegs.XCLK.bit.XCLKOUTDIV=10 ;
```

DSP 之 GPIO

currently reserved for future expansion. Figure 4-1 shows the basic modes of operation for the GPIO module.

Figure 4-1. Modes of Operation



A x stands for the port, either A or B. For example, GPxDIR refers to either the GPADIR and GPBDIR register depending on the particular GPIO pin selected.

B GPxDAT latch/read are accessed at the same memory location.

GPIO0—35

寄存器：

1 GPIO Port A MUX 1 (GPAMUX1) Register

引脚功能控制寄存器：GPAMUX1　控制 0-15 号引脚

4.6 Register Bit Definitions

Figure 4-4. GPIO Port A MUX 1 (GPAMUX1) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND- R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-15. GPIO Port A MUX 1 (GPAMUX1) Register Field Descriptions

Bits	Field	Value	Description (1)
31-30	GPIO15		Configure the GPIO15 pin as:
		00	GPIO15 (I/O), General purpose I/O 15 (default)
		01	• For 28044 device: – If GPAMCFG[EPWMMODE] = 0,0 then $\overline{TZ4}$ (I), Trip zone 4 – If GPAMCFG[EPWMMODE] = 1,1 then EPWM16A (O), ePWM16 output A. • All other devices: $\overline{TZ4}$ (I), Trip zone 4
		10	SCIRXDB (I), SCI-B receive. This option is reserved on devices that do not have an SCI-B port. (2)
		11	SPISTEB (I/O), SPI-B transmit enable. This option is reserved on devices that do not have an SPI-B port. (2)

GPAMUX2　控制 16-31 号引脚

Figure 4-5. GPIO Port A MUX 2 (GPAMUX2) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO31	GPIO30	GPIO29	GPIO128	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-16. GPIO Port A MUX 2 (GPAMUX2) Register Field Descriptions

Bits	Field	Value	Description (1)
31-30	GPIO31		Configure the GPIO31 pin as:
		00	GPIO31 (I/O) General purpose I/O 31 (default)
		01	CANTXA (O), eCAN-A transmit This option is reserved on devices that do not have an eCAN-A port. (2)
		10	Reserved (2)
		11	On 28044 device: $\overline{TZ4}$ (I), Trip zone 4 On all other devices: Reserved (2)

(1) This register is EALLOW protected. See Section 5.2 for more information.
(2) If reserved configurations are selected, then the state of the pin will be undefined and the pin may be driven. These selections are reserved for future expansion and should not be used.

GPBMUX1　控制 32-34 号引脚：

Figure 4-6. GPIO Port B MUX 1 (GPBMUX1) Register

31 6	5 4	3 2	1 0
Reserved	GPIO34	GPIO33	GPIO32
R-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-17. GPIO Port B MUX 1 (GPBMUX1) Register Field Descriptions

Bits	Field	Value	Description (1)
31-6	Reserved		Reserved
5-4	GPIO34		Configure the GPIO34 pin as:
		00	GPIO34 (I/O), General purpose I/O 34 (default)
		01	Reserved (2)
		10	Reserved (2)
		11	Reserved (2)
3-2	GPIO33		Configure the GPIO33 pin as:
		00	GPIO33 (I/O), General purpose I/O 33 (default)
		01	SCLA (I/O), I2C clock
		10	EPWMSYNCO (O), ePWM synchronization output
		11	ADCSOCBO (O)
1-0	GPIO32		Configure the GPIO32 pin as:
		00	GPIO32 (I/O), General purpose I/O 32 (default)
		01	SDAA(I/O), I2C data
		10	EWPMSYNCI (I), ePWM ePWM synchronization input

注 ： GPIO 复用功能寄存器 MUX 可以根据设置相应的位来达到改变 GPIO 功能的作用
其中两位控制一个 GPIO

```
GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 0;  // GPIO6 = GPIO6
```

2 Figure 4-13. GPIO Port A Direction (GPADIR) Register

方向控制器 GPADIR 控制 31-0

functions.

Figure 4-13. GPIO Port A Direction (GPADIR) Register

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-25. GPIO Port A Direction (GPADIR) Register Field Descriptions

Bits	Field	Value	Description (1)
31-0	GPIO31-GPIO0		Controls direction of GPIO Port A pins when the specified pin is configured as a GPIO in the appropriate GPAMUX1 or GPAMUX2 register.
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output
			The value currently in the GPADAT output latch is driven on the pin. To initialize the GPADAT latch prior to changing the pin from an input to an output, use the GPASET, GPACLEAR, and GPATOGGLE registers.

(1) This register is EALLOW protected. See Section 5.2 for more information.

GPBDIR 控制 34-32

Figure 4-14. GPIO Port B Direction (GPBDIR) Register

31	3	2	1	0
Reserved		GPIO34	GPIO33	GPIO32
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-26. GPIO Port B Direction (GPBDIR) Register Field Descriptions

Bits	Field	Value	Description (1)
31-3	Reserved		
2-0	GPIO34-GPIO32		Controls the direction of GPIO Port B pins (input or output) when the specified pin is configured as a GPIO in the appropriate GPBMUX1 or GPBMUX2 register.
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output
			The value currently in the GPBDAT output latch is driven on the pin. To initialize the GPBDAT latch prior to changing the pin from an input to an output, use the GPBSET, GPBCLEAR, and GPBTOGGLE registers.

(1) This register is EALLOW protected. See Section 5.2 for more information.

注： 方向寄存器是控制 GPIO 的方向的
寄存器的每一位控制一个 GPIO
对其写 0 代表 GPIO 输入，写 1 代表 GPIO 输出

```
GpioCtrlRegs.GPADIR.bit.GPIO17 = 1;   // GPIO17 = output
```

3 Figure 4-15. GPIO Port A Pullup Disable (GPAPUD) Registers

内部上拉控制寄存器 **GPAPUD　控制 31-0 引脚**

Figure 4-15. GPIO Port A Pullup Disable (GPAPUD) Registers

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15(1)	GPIO14(1)	GPIO13(1)	GPIO12(1)	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

(1) On the F28044 device, GPIO12 - GPIO15 pull-ups are also disabled at reset.

Table 4-27. GPIO Port A Internal Pullup Disable (GPAPUD) Register Field Descriptions

Bits	Field	Value	Description (1)
31-0	GPIO31-GPIO0		Configure the internal pullup resister on the selected GPIO Port A pin. Each GPIO pin corresponds to one bit in this register as shown in Figure 4-15.
		0	Enable the internal pullup on the specified pin. (default for GPIO12-GPIO31)
		1	Disable the internal pullup on the specified pin. (default for GPIO0-GPIO11)

(1) This register is EALLOW protected. See Section 5.2 for more information.

GPBPUD　控制 32-34 引脚

Figure 4-16. GPIO Port B Pullup Disable (GPBPUD) Register

31	6	5	4	3	2	1	0
Reserved		Reserved	Reserved	GPIO35	GPIO34	GPIO33	GPIO32
R-0		R/W-0	R/W-1	R/W-1	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-28. GPIO Port B Internal Pullup Disable (GPBPUD) Register Field Descriptions

Bits	Field	Value	Description (1)
31-3	Reserved		Reserved
2-0	GPIO35(2)-GPIO32		Configure the internal pullup resister on the selected GPIO Port B pin. Each GPIO pin corresponds to one bit in this register as shown in Figure 4-16.
		0	Enable the internal pullup on the specified pin. (default)
		1	Disable the internal pullup on the specified pin.

(1) This register is EALLOW protected. See Section 5.2 for more information.
(2) GPIO35 signal is internally available, but not pinned out. The internal pullup for this signal is disabled upon reset. To minimize the leakage currents in order to ensure that the low-power mode IDDIO current stays within the datasheet limits, this pullup has to be enabled by the user. i.e. bit 3 should be 0. Any write to GPBPUD should ensure bit 5 remains 0.

注：内部上拉控制寄存器是用来开启/关闭内部上拉

```
GpioCtrlRegs.GPAPUD.bit.GPIO8 = 0;   // Enable pullup on GPIO8
```

4　Figure 4-17. GPIO Port A Data (GPADAT) Register

状态寄存器 GPADAT 控制 GPIO31-0

Figure 4-17. GPIO Port A Data (GPADAT) Register

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset(1)

(1) x = The state of the GPADAT register is unknown after reset. It depends on the level of the pin after reset.

Table 4-29. GPIO Port A Data (GPADAT) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31-GPIO0		Each bit corresponds to one GPIO port A pin (GPIO0-GPIO31) as shown in Figure 4-17.
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode the pin is configured for.
			Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPAMUX1/2 and GPADIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode the pin is configured for.
			Writing a 1will force an output of 1if the pin is configured as a GPIO output in the appropriate GPAMUX1/2 and GPADIR registers; otherwise, the value is latched but not used to drive the pin.

状态寄存器 GPBDAT 控制 32-34

Figure 4-18. GPIO Port B Data (GPBDAT) Register

31	3	2	1	0
Reserved	GPIO35	GPIO34	GPIO33	GPIO32
R-0	R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset (1)

(1) x = The state of the GPBDAT register is unknown after reset. It depends on the level of the pin after reset.

Table 4-30. GPIO Port B Data (GPBDAT) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2-0	GPIO35(1)-GPIO32		Each bit corresponds to one GPIO port B pin (GPIO32-GPIO35) as shown in Figure 4-18
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode the pin is configured for.
			Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPBMUX1 and GPBDIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode the pin is configured for.
			Writing a 1 will force an output of 1 if the pin is configured as a GPIO output in the GPBMUX1 and GPBDIR registers; otherwise, the value is latched but not used to drive the pin.

(1) GPIO35 signal is internally available, but not pinned out.

注：DAT 状态寄存器可以用来读取 GPIO 的值　也可用来对 GPIO 进行赋值

寄存器的每一位控制一个 GPIO

当对位赋值 0 时并且但 GPIO 对应位是 OUTPUT 时才能驱动此端口为 0，如果 GPIO 设置的是

输入，则不能驱动端口输出 0。

Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode the pin is configured for.（读取0状态操作）

Reading a 1 indicates that the state of the pin is currently high irrespective of the mode the pin is configured for. （读取1状态操作）

```
GpioDataRegs.GPADDAT.all = 0xFFFDFFFF     //GPIO16输出低电平
```

5 Figure 4-19. GPIO Port A Set, Clear and Toggle (GPASET, GPACLEAR, GPATOGGLE) Registers

GPIO端口置位，清零，反转控制寄存器GPASET, GPACLEAR, GPATOGGLE

控制31-0引脚

Figure 4-19. GPIO Port A Set, Clear and Toggle (GPASET, GPACLEAR, GPATOGGLE) Registers

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-31. GPIO Port A Set (GPASET) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31-GPIO0		Each GPIO port A pin (GPIO0-GPIO31) corresponds to one bit in this register as shown in Figure 4-19.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to high. If the pin is configured as a GPIO output then it will be driven high. If the pin is not configured as a GPIO output then the latch is set high but the pin is not driven.

Table 4-32. GPIO Port A Clear (GPACLEAR) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31 - GPIO0		Each GPIO port A pin (GPIO0-GPIO31) corresponds to one bit in this register as shown in Figure 4-19.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to low. If the pin is configured as a GPIO output then it will be driven low. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Table 4-33. GPIO Port A Toggle (GPATOGGLE) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31-GPIO0		Each GPIO port A pin (GPIO0-GPIO31) corresponds to one bit in this register as shown in Figure 4-19.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to toggle from its current state. If the pin is configured as a GPIO output then it will be driven in the opposite direction of its current state. If the pin is not configured as a GPIO output then the latch is toggled but the pin is not driven.

注：对置位，清零，反转操作时对寄存器写0被忽略不计，只有当对其写1是才起作用，比如对某一位GPIO的GPACLEAR写1，则表示对此端口进行清零操作。

同理GPBSET, GPBCLEAR, GPBTOGGLE则是对32-34GPIO操作，

例：`GpioDataRegs.GPASET.bit.GPIO12 = 1;` `//GPIO置1`

6 Figure 4-21. GPIO XINT1, XINT2, XNMI Interrupt Select (GPIOXINT1SEL, GPIOXINT2SEL,GPIOXNMISEL) Registers

XINT1, XINT2, XNMI中断源控制寄存器

Figure 4-21. GPIO XINT1, XINT2, XNMI Interrupt Select (GPIOXINT1SEL, GPIOXINT2SEL, GPIOXNMISEL) Registers

15	5	4	0
Reserved		GPIOSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-37. GPIO XINT1 Interrupt Select (GPIOXINT1SEL) Register Field Descriptions

Bits	Field	Value	Description (1)
15-5	Reserved		Reserved
4-0	GPIOSEL		Select which port A GPIO signal (GPIO0 - GPIO31) will be used as the XINT1 interrupt source. In addition you can configure the interrupt in the XINT1CR register described in Section 6.6.
		00000	Select the GPIO0 pin as the XINT1 interrupt source (default)
		00001	Select the GPIO1 pin as the XINT1 interrupt source
		. . .	. . .
		11110	Select the GPIO30 pin as the XINT1 interrupt source
		11111	Select the GPIO31 pin as the XINT1 interrupt source

(1) This register is EALLOW protected. See Section 5.2 for more information.

Table 4-38. GPIO XINT2 Interrupt Select (GPIOXINT2SEL) Register Field Descriptions

Bits	Field	Value	Description (1)
15-5	Reserved		Reserved
4-0	GPIOSEL		Select which port A GPIO signal (GPIO0 - GPIO31) will be used as the XINT2 interrupt source. In addition you can configure the interrupt in the XINT2CR register described in Section 6.6. To use the signal as ADC start of conversion, enable it in the ADCTRL2 register. The ADCSOC is always rising edge sensitive.
		00000	Select the GPIO0 pin as the XINT2 interrupt source (default)
		00001	Select the GPIO1 pin as the XINT2 interrupt source
		. . .	. . .
		11110	Select the GPIO30 pin as the XINT2 interrupt source
		11111	Select the GPIO31 pin as the XINT2 interrupt source

(1) This register is EALLOW protected. See Section 5.2 for more information.

Table 4-39. GPIO XNMI Interrupt Select (GPIOXNMISEL) Register Field Descriptions

Bits	Field	Value	Description (1)
15-5	Reserved		Reserved
4-0	GPIOSEL		Select which port A GPIO signal (GPIO0 - GPIO31) will be used as the XNMI interrupt source. In addition you can configure the interrupt in the XNMICR register described in Section 6.6.
		00000	Select the GPIO0 pin as the XNMI interrupt source (default)
		00001	Select the GPIO1 pin as the XNMI interrupt source
		. . .	. . .
		11110	Select the GPIO30 pin as the XNMI interrupt source
		11111	Select the GPIO31 pin as the XNMI interrupt source

(1) This register is EALLOW protected. See Section 5.2 for more information.

注：此处只拿XINT1作为参考

想把哪个GPIO作为XINT1的中断源只需要设置对应位寄存器就OK

7 **Table 4-40. GPIO Low Power Mode Wakeup Select (GPIOLPMSEL)**

Register Field Descriptions 低功耗唤醒设置寄存器

Figure 4-22. GPIO Low Power Mode Wakeup Select (GPIOLPMSEL) Register

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-40. GPIO Low Power Mode Wakeup Select (GPIOLPMSEL) Register Field Descriptions

Bits	Field	Value	Description (1)
31-0	GPIO31 - GPIO0		Low Power Mode Wakeup Selection. Each bit in this register corresponds to one GPIO port A pin (GPIO0 - GPIO31) as shown in Figure 4-22.
		0	If the bit is cleared, the signal on the corresponding pin will have no effect on the HALT and STANDBY low power modes.
		1	If the respective bit is set to 1, the signal on the corresponding pin is able to wake the device from both HALT and STANDBY low power modes.

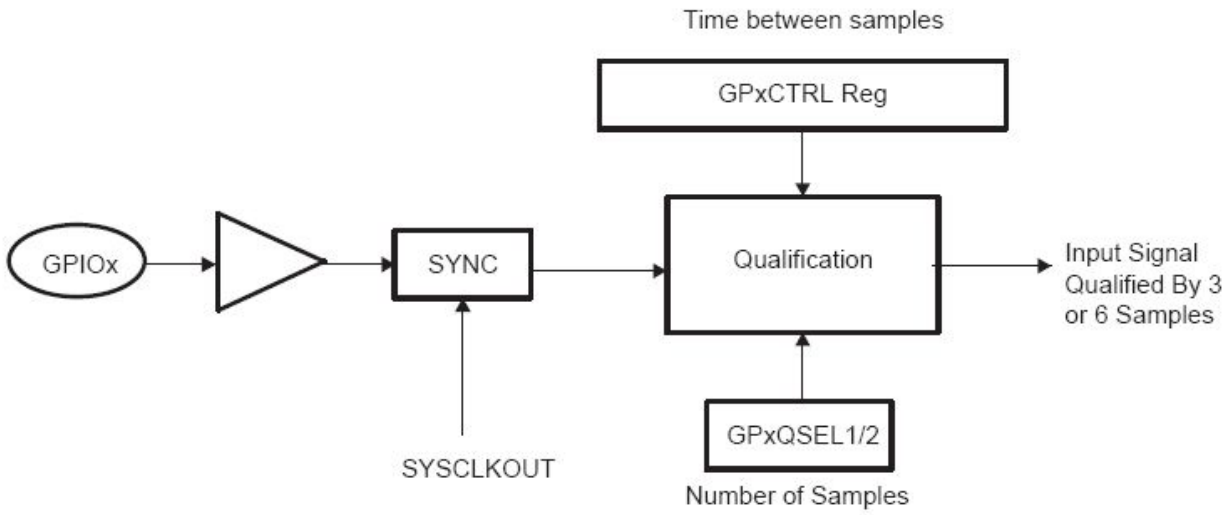
(1) This register is EALLOW protected. See Section 5.2 for more information.

DSP GPIO功能强大的硬件消抖

很多工程师在对按键去抖动都会采用软件延时的方法,今天介绍下关于 Piccolo DSP GPIO 所具有的硬件去抖功能.

信号首先同步于系统时钟，在输入到内核改变之前要经过一定周期个数的量化,这种方式可以滤除噪声。量化窗口输入的结构示意图如图所示。

Figure 4-2. Input Qualification Using a Sampling Window



由图可知，外部的管脚输入信号经过一个量化窗口到达 DSP 内核。量化窗口的时间宽度取决于采样间隔和采样个数。采样间隔由 GPxCTRL 寄存器决定，采样个数由寄存器 GPxQSELn 决定。

如果 QUxLPRDn = 0，采样间隔为一个系统周期整数倍。如果 QUxLPRDn = N（N≠0)，采样周期为系统周期的 2N 倍。如表所示

QUxLPRDn	Sampling period
0	$1 \times T_{SYSCLKOUT}$
N (N≠0)	$2N \times T_{SYSCLKOUT}$

采样个数可以配置位三个或六个。通过寄存器 GPAQSEL1、GPAQSEL2、GPBQSEL1 来配置，当外部管脚电平状态发生变化时，量化窗口检测到三个或六个周期内相同的状态时，才把此次状态改变传送到内核，否则会当成杂波滤除。

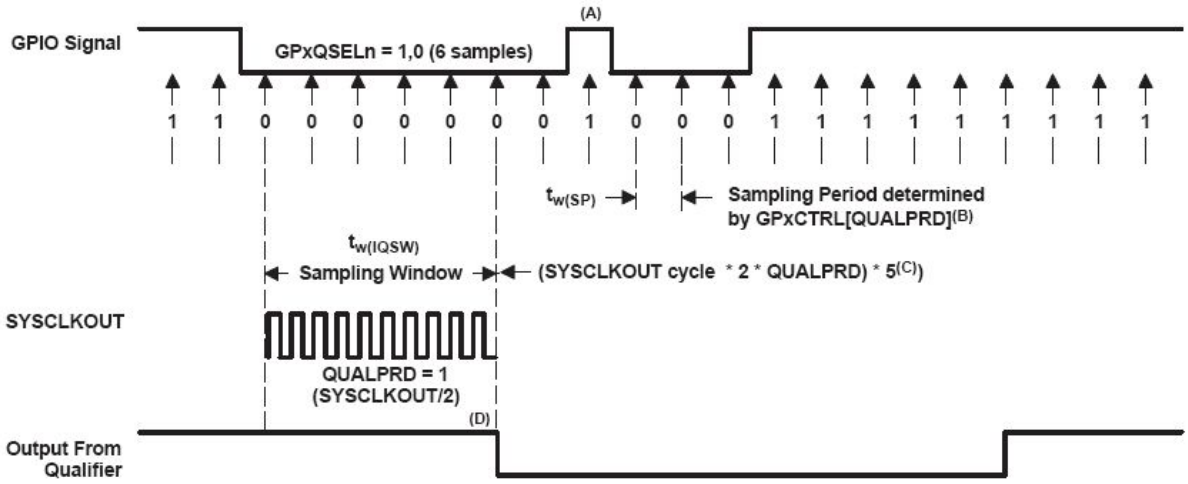
量化窗口的宽度与采样周期和采样个数有关，当采样周期和采样个数确定之后，量化窗口的宽度就确定了。量化窗口与采样周期及采样个数的关系如表所示。

采样个数	QUxLPRDn（采样周期）	窗口宽度
3	0	（3－1）×1×$T_{SYSCLKOUT}$
3	N（N≠0）	（3－1）×2N×$T_{SYSCLKOUT}$
6	0	（6－1）×1×$T_{SYSCLKOUT}$
6	N（N≠0）	（6－1）×2N×$T_{SYSCLKOUT}$

现举例说明量化窗口的用法及效果。假如某个 GPIO 引脚为数字输入，量化模式为 6 个采样周期量化，采样周期寄存器中 QUxLPRDn 值为一，则采样周期为两个系统时钟周期宽度，量化输入的结果如图所示。

qualifier.

Figure 4-3. Input Qualifier Clock Cycles



A. This glitch will be ignored by the input qualifier. The QUALPRD bit field specifies the qualification sampling period. It can vary from 00 to 0xFF. If QUALPRD = 00, then the sampling period is 1 SYSCLKOUT cycle. For any other value "n", the qualification sampling period in 2n SYSCLKOUT cycles (i.e., at every 2n SYSCLKOUT cycles, the GPIO pin will be sampled).
B. The qualification period selected via the GPxCTRL register applies to groups of 8 GPIO pins.
C. The qualification block can take either three or six samples. The GPxQSELn Register selects which sample mode is used.
D. In the example shown, for the qualifier to detect the change, the input should be stable for 10 SYSCLKOUT cycles or greater. In other words, the inputs should be stable for (5 x QUALPRD x 2) SYSCLKOUT cycles. That would ensure 5 sampling periods for detection to occur. Since external signals are driven asynchronously, an 13-SYSCLKOUT-wide pulse ensures reliable recognition.

注：值得注意的是在DEBUG 的时候必须，当触发中断的时候不能用单步执行程序，因为由于DSP有端口硬件消抖的功能，输入的状态必须和系统时钟一致才能触发外中断的发生，应该用RUN直接执行程序 。

DSP之外部中断

PIE

CPU 有 1 各部可屏蔽中断 NMI。
14 个可屏蔽中断 INT1---INT14。
2812 内有很多外设每一个外设可以产生一个或多个中断、 CPU 没有能力同时处理所有中断请求专门设计一个 PIE 外设中断控制器。

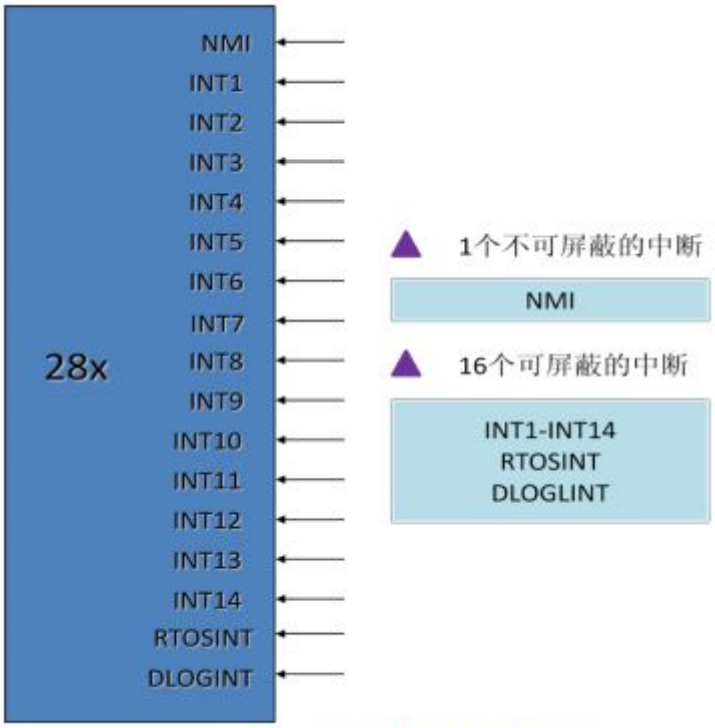
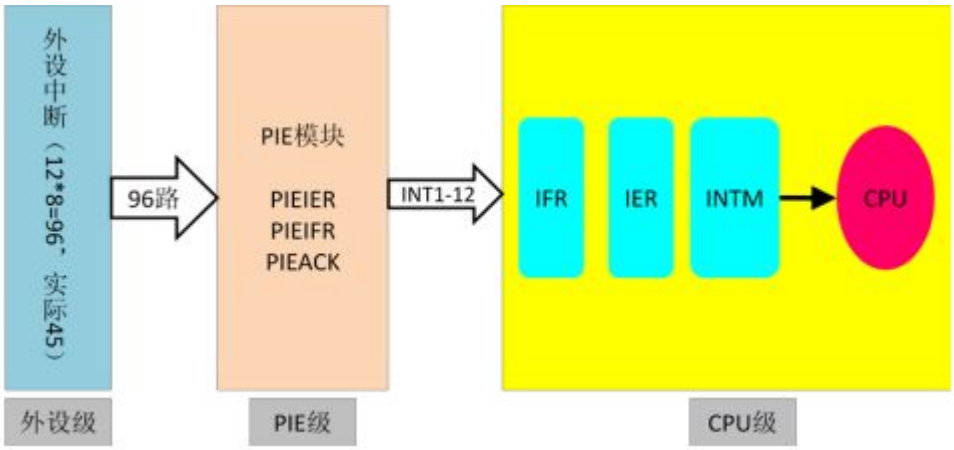


图 1 2812 的 CPU 所能支持的中断

PIE 支持 96 个中断 12 组每组 8 个中断。
而且每组都被反馈到 CPU 内核 12 条中短线的某一条上面。
其实我们作用了 96 中断的 45 个。

	INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
INT1	WAKEINT	TINT0	ADCINT	XINT2	XINT1		PDPINTB	PDPINTA
INT2		T1OFINT	T1UFINT	T1CINT	T1PINT	CMP3INT	CMP2INT	CMP1INT
INT3		CAPINT3	CAPINT2	CAPINT1	T2OFINT	T2UFINT	T2CINT	T2PINT
INT4		T3OFINT	T3UFINT	T3CINT	T3PINT	CMP6INT	CMP5INT	CMP4INT
INT5		CAPINT6	CAPINT5	CAPINT4	T4OFINT	T4UFINT	T4CINT	T4PINT
INT6			MXINT	MRINT			SPITXINTA	SPIRXINTA
INT7								
INT8								
INT9			ECAN1INT	ECAN0INT	SCITXINTB	SCIRXINTB	SCITXINTA	SCIRXINTA
INT10								
INT11								
INT12								

CPU 是 3 级中断机制　外设级　PIE 级　CPU 级。对于一个具体的外设中断请求，一个不许可 CPU 不会执行该中断。



当某个外设中断事件发生了，这个外设某个寄存器和该中断标志置 1
如果该中断的中断使能位被置 1　外设就会向 PIE 发送中断请求，否则不会往 PIE 发送中断请求。不管什么情况下外设中断标志寄存器中断中断标志必须手动清除
例如

```
EvaRegs.EVAIFRA.bit.T1PINT=1;
```

每一组 PIE 都有个 标志寄存器 PIEIFRx　使能寄存器 PIEIERx　还有个相应寄存器 ACK

一组同时一个中断被响应　因为 PIEACK 寄存器 12 位对应 INT1 --12
有中断 PIEACK 对应位置 1　等待手动清除　（标志寄存器自动清除）才能响应同组下一个中断（如果中断请求还在）。

```
DINT;     //禁止中断　在设置某些寄存器的时候
EINT;     // Enable Global interrupt INTM
ERTM;  // Enable Global realtime interrupt DBGM
EALLOW;   // This is needed to write to EALLOW protected registers    写自由
EDIS;     // This is needed to disable write to EALLOW protected registers  禁止写某些特殊寄存器
```

所以简单点说，就是有些寄存器受到写保护的，你要写它，就首先要用ＥＡＬＬＯＷ禁止写保护．写完了之后别忘了用ＥＤＩＳ再是能写保护．
CPU 级也有 IFR　IER　CPU 接到中断请求暂停现在执行的程序　清除 IER　IFR
EALLOW　、 INTM 置位就不响应其他中断

1、　PieVectTable.T1PINT = &eva_timer1_isr;
2、　// Enable PIE group 2 interrupt 4 for T1PINT
　　PieCtrl.PIEIER2.all = M_INT4;
3、IER |= (M_INT2 | M_INT3 | M_INT4 | M_INT5);
4、interrupt void eva_timer1_isr(void)

1、外围帧寄存器2812将外围帧寄存器分为3个空间，分别是：外围帧0：直接映射到CPU存储器总线外围帧1：映射到32位外围总线外围帧2：映射到16位外围总线，只允许16操作这里所说的外围帧寄存器就是外设寄存器，如ADC寄存器。映射就是分配地址，外围一个地址，CPU一个地址，外围地址映射到CPU地址上。有的寄存器受保护，对其进行操作的时候好，要屏蔽保护（EALLOW），操作完在开启保护(EDIS).2、外围中断扩展PIE外设寄存器中的中断标志位必须由软件清0，才能允许下一次中断进入，而且每次中断后要把PIEACKx清0，只有PIEACKx=0，才可以把中断送到CPU级中断分为3级：a、外设级b、PIE级c、CPU级下面分别介绍这个不同的中断级以及这3级的联系a、外设级一旦外设产生了中断，对应的外设中断标志寄存器中的中断标志位IF就会置位，如果此时对应的中断使能位设为1，那么外设中断信号可以送到PIE控制器，如果外设的中断被禁止输入进来，那么外设中断标志位保持为1，直到软件清0。外设级和PIE级的联系就在外设中断使能位那了，实际外设的中断使能位使能的就是允许外设中断进入PIE级，相当于PIE级的中断源，就像外设的中断源一样，有了中断事件，外设中断标志位就会置位，这里也是外设级就是PIE级得中断事件，有了外设级的中断标志位置位，中断使能位使能，那么PIE级的中断标志位才会置1.注意的是外设中断标志寄存器中的中断标志位必须由软件清0，才能允许下一次中断进入。b、PIE级PIE级有两中寄存器，一种是中断标志寄存器PIEIFRx，上面已经说过他跟外设级的联系，另一个是中断使能寄存器PIEIERx，这个跟外设级的中断使能寄存器功能差不多，实现的是和CPU级的联系.PIE级还有两个寄存器，一个是控制寄存器PIECTRL,这个是控制整个PIE级的，还有一个应答寄存器PIEACK,在PIEIFRx置位，PIEIERx使能还要PIEACKx清0才能把中断送到CPU级。这里注意的是PIEIFRx由硬件清0，但是PIEACKx要由软件清0.c、CPU级CPU级是最终控制中断响应的，也是有两种寄存器，一个是中断标志寄存器IFR，另一个是中断使能寄存器IER。IFR是这三级中断的最终的中断标志位，IER是这三级中断的最终中断使能位，只有这三级中断标志位同时置位，三级中断同时使能，这里还有一个CPU级的中断屏蔽位INTM，在以上条件满足的前提下，中断屏蔽位INTM=0，CPU才能响应中断，找到中断向量，跳转到中断函数，执行中断操作，CPU级的中断标志位由硬件清0，在中断函数中不用管。这就是它的中断过程。3、96个中断介绍2812分为12组中断，每组中断有8个中断源，以INTx.y表示，其中x是组（x=1~12），y（y=1~8）是组中的位.在配置中断的时候，外设级的中断使能位要使能；PIECTRL寄存器使能PIE；要知道INTx.y中的xy是多少，找到PIEIERx（x=1~12）的x是多少，使能PIEIERx；PIEACKx（x=1~12）的x是多少给其清0；IERx（x=1~12）的x是多少，使能IERx；中断屏蔽位INTM=0；中断函数中要处理的：外设中断标志位软件清0PIEACKx（x=1~12）软件清0；4、C代码分析以定时器0的C代码分析timer0的中断向量INT1.7（x=1，y=7），用到的PIE级的中断使能寄存器是PIEIER1，CPU级的IER1.在2812的库函数中有一个定义中断入口地址的函数，用户的中断函数内容可以在这里面写，就不容另外赋中断地址了。中断函数入口地址interrupt void
TINT0_ISR(void)
。在DSP复位后，进入中函数前，看门狗开着的，所以在进入中函数后第一件事情是关看门狗，在这个函数里面InitSysCtrl();中断是开着的需要关闭（DINT;），PIE寄存器（ InitPieCtrl();//初始化pie寄存器）（ IER = 0x0000;//禁止所有的中断 IFR = 0x0000;）、中断向量表是没有初始化的，所以要初始化（ InitPieVectTable();//初始化pie中断向量表）。InitSysCtrl();//初始化cpu

```
DINT;//关中断
InitPieCtrl();//初始化pie寄存器
IER = 0x0000;//禁止所有的中断
IFR = 0x0000;
InitPieVectTable();//初始化pie中断向量表
EALLOW;
// This is needed to write to EALLOW protected registers
PieVectTable.TINT0 = &cpu_timer0_isr;//指定中断服务子程序，这个地方要是用到了DSP的库函数DSP281X_DefaultIsr()函数就不需要了。
EDIS;
InitCpuTimers();//初始化定时器0
ConfigCpuTimer(&CpuTimer0,150,9.75);//配置你要定时的时间
StartCpuTimer0();//定时器开始计数
PieCtrlRegs.PIEIER1.bit.INTx7 = 1;//使能PIE级中断PIEIERx中的x=1，即使能的12组中的第1组第七个
```

IER |= M_INT1；//使能CPU级的中断第1组

EINT;

// 使能INTM

ERTM;

// 使能仿真时 DBGM5、在中断函数中要做的interrupt void

TINT0_ISR(void)

// CPU-Timer 0{

PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //PIEACK清0

CpuTimer0Regs.TCR.bit.TIF = 1;//外设中断标志位清0

CpuTimer0Regs.TCR.bit.TRB = 1;//重新装载

中断矢量表

Table 6-4. 280x, 2801x 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)			-	(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	Reserved	Reserved	ECAP4_INT	ECAP3_INT	ECAP2_INT	ECAP1_INT
	-	-	-	-	(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	SPITXINTD	SPIRXINTD	SPITXINTC	SPIRXINTC	SPITXINTB	SPIRXINTB	SPITXINTA	SPIRXINTA
	(SPI-D)	(SPI-D)	(SPI-C)	(SPI-C)	(SPI-B)	(SPI-B)	(SPI-A)	(SPI-A)
	0xD9E	0xD9C	0xD9A	0xD98	0xD96	0xD94	0xD92	0xD90
INT7.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	0xDAE	0xDAC	0xDAA	0xDA8	0xDA6	0xDA4	0xDA2	0xDA0
INT8.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	I2CINT2A	I2CINT1A
	-	-	-	-	-	-	(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	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	0xDFE	0xDFC	0xDFA	0xDF8	0xDF6	0xDF4	0xDF2	0xDF0

```
PieCtrlRegs.PIEIER1.bit.INTx4 = 1;          // Enable PIE Gropu 1 INT4
PieCtrlRegs.PIEIER1.bit.INTx5 = 1;          // Enable PIE Gropu 1 INT5
```

注：指令集在程序的 `DSP280x_Device.h` 头文件中

```
    #define  EINT   asm(" clrc INTM")      //  //INTM置0，开中断
    #define  DINT   asm(" setc INTM")      //  //INTM置1，关中断
    #define  ERTM   asm(" clrc DBGM")        //使能调试事件
    #define  DRTM   asm(" setc DBGM")     //禁止调试事件
    #define  EALLOW asm(" EALLOW")
    #define  EDIS   asm(" EDIS")
    #define  ESTOP0 asm(" ESTOP0")
EALLOW;
  //GpioCtrlRegs.GPAMUX1.all = 0x0;    // GPIO pin
    //GpioCtrlRegs.GPADIR.all = 0xFF;     // Output pin
  //GpioDataRegs.GPADAT.all =0xFF;     // Close LEDs
 EDIS;
```

说明：EALLOW 一般和 EDIS 配套使用，在对受保护的寄存器操作之后，用 EDIS 恢复寄存器的被保护状态。

以下列寄存器受 EALLOW 保护：P

- 器件仿真寄存器
- 闪存寄存器
- CSM 寄存器
- PIE 矢量表
- 系统控制寄存器
- GPIO MUX 寄存器
- 某些 eCAN 寄存器

5.2 EALLOW-Protected Registers

Several control registers on the 280x devices are protected from spurious CPU writes by the EALLOW protection mechanism. The EALLOW bit in status register 1 (ST1) indicates if the state of protection as shown in Table 5-4.

Table 5-4. Access to EALLOW-Protected Registers

EALLOW Bit	CPU Writes	CPU Reads	JTAG Writes	JTAG Reads
0	Ignored	Allowed	Allowed(1)	Allowed
1	Allowed	Allowed	Allowed	Allowed

(1) The EALLOW bit is overridden via the JTAG port, allowing full access of protected registers during debug from the Code Composer Studio interface.

At reset the EALLOW bit is cleared enabling EALLOW protection. While protected, all writes to protected registers by the CPU are ignored and only CPU reads, JTAG reads, and JTAG writes are allowed. If this bit is set, by executing the EALLOW instruction, then the CPU is allowed to write freely to protected registers. After modifying registers, they can once again be protected by executing the EDI instruction to clear the EALLOW bit.

The following registers are EALLOW-protected:

- Device Emulation Registers
- Flash Registers
- CSM Registers
- PIE Vector Table
- System Control Registers
- GPIO MUX Registers
- Certain eCAN Registers

1 Figure 6-13. External Interrupt 1 Control Register (XINT1CR) (Address 7070h)

XINT1 中断 方式/使能 控制寄存器 XINT1CR

Figure 6-13. External Interrupt 1 Control Register (XINT1CR) (Address 7070h)

15	4	3	2	1	0
Reserved		Polarity		Reserved	Enable
R-0		R/W-0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-16. External Interrupt 1 Control Register (XINT1CR) Field Descriptions

Bits	Field	Value	Description
15-4	Reserved		Reads return zero; writes have no effect.
3-2	Polarity		This read/write bit determines whether interrupts are generated on the rising edge or the falling edge of a signal on the pin.
		00	Interrupt generated on a falling edge (high-to-low transition)
		01	Interrupt generated on a rising edge (low-to-high transition)
		10	Interrupt generated on a falling edge (high-to-low transition)
		11	Interrupt generated on both a falling edge and a rising edge (high-to-low transition and low-to-high transition)
1	Reserved		Reads return zero; writes have no effect
0	Enable		This read/write bit enables or disables external interrupt XINT1.
		0	Disable interrupt
		1	Enable interrupt

XINT2 中断 方式/使能 控制寄存器 XINT2CR

Figure 6-14. External Interrupt 2 Control Register (XINT2CR) (Address 7071h)

15	4	3	2	1	0
Reserved		Polarity		Reserved	Enable
R-0		R/W-0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-17. External Interrupt 2 Control Register (XINT2CR) Field Descriptions

Bits	Field	Value	Description
15-4	Reserved		Reads return zero; writes have no effect.
3-2	Polarity		This read/write bit determines whether interrupts are generated on the rising edge or the falling edge of a signal on the pin.
		00	Interrupt generated on a falling edge (high-to-low transition)
		01	Interrupt generated on a rising edge (low-to-high transition)
		10	Interrupt is generated on a falling edge (high-to-low transition)
		11	Interrupt generated on both a falling edge and a rising edge (high-to-low and low-to-high transition)
1	Reserved		Reads return zero; writes have no effect
0	Enable		This read/write bit enables or disables external interrupt XINT2.
		0	Disable interrupt
		1	Enable interrupt

注：此寄存器用来控制触发中断的方式以及使能和禁止中断的产生。

```
XIntruptRegs.XINT1CR.bit.POLARITY = 0;      // Falling edge interrupt
XIntruptRegs.XINT2CR.bit.POLARITY = 1;      // Rising edge interrupt
XIntruptRegs.XINT1CR.bit.ENABLE = 1;        // Enable XINT1
XIntruptRegs.XINT2CR.bit.ENABLE = 1;        // Enable XINT2
```

2: Figure 6-10. Interrupt Flag Register (IFR) — CPU Register

终端标志寄存器IFR

Figure 6-10. Interrupt Flag Register (IFR) — CPU Register

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 6-13. Interrupt Flag Register (IFR) — CPU Register Field Descriptions

Bits	Field	Value	Description
15	RTOSINT		Real-time operating system flag. RTOSINT is the flag for RTOS interrupts.
		0	No RTOS interrupt is pending
		1	At least one RTOS interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
14	DLOGINT		Data logging interrupt fag. DLOGINT is the flag for data logging interrupts.
		0	No DLOGINT is pending
		1	At least one DLOGINT interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
13	INT14		Interrupt 14 flag. INT14 is the flag for interrupts connected to CPU interrupt level INT14.
		0	No INT14 interrupt is pending
		1	At least one INT14 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
12	INT13		Interrupt 13 flag. INT13 is the flag for interrupts connected to CPU interrupt level INT13I.
		0	No INT13 interrupt is pending

注：标志寄存器当对应位的终端来临时标志位值1，可以通过写0来清除它或者是中断请求，当写0表示没有中断请求

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

3：Figure 6-11. Interrupt Enable Register (IER) — CPU Register

中断使能/禁止寄存器

Figure 6-11. Interrupt Enable Register (IER) — CPU Register

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 6-14. Interrupt Enable Register (IER) — CPU Register Field Descriptions

Bits	Field	Value	Description
15	RTOSINT		Real-time operating system interrupt enable. RTOSINT enables or disables the CPU RTOS interrupt.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
14	DLOGINT		Data logging interrupt enable. DLOGINT enables or disables the CPU data logging interrupt.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
13	INT14		Interrupt 14 enable. INT14 enables or disables CPU interrupt level INT14.
		0	Level INT14 is disabled
		1	Level INT14 is enabled
12	INT13		Interrupt 13 enable. INT13 enables or disables CPU interrupt level INT13.
		0	Level INT13 is disabled

注：写0表示禁止中断，写1表示允许中断

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

中断优先级的设置在

DSP280x_SWPrioritizedIsrLevels.h中

```
// 0    = not used
// 1    = highest priority
// ...
// 16 = lowest priority
#define  INT1PL          2          // Group1 Interrupts (PIEIER1)
#define  INT2PL          1          // Group2 Interrupts (PIEIER2)
#define  INT3PL          4          // Group3 Interrupts (PIEIER3)
#define  INT4PL          2          // Group4 Interrupts (PIEIER4)
#define  INT5PL          2          // Group5 Interrupts (PIEIER5)
#define  INT6PL          3          // Group6 Interrupts (PIEIER6)
#define  INT7PL          0          // reserved
#define  INT8PL          0          // reserved
#define  INT9PL          3          // Group9 Interrupts (PIEIER9)
#define  INT10PL         0          // reserved
#define  INT11PL         0          // reserved
#define  INT12PL         0          // reserved
#define  INT13PL         4          // XINT13
#define  INT14PL         4          // INT14 (TINT2)
#define  INT15PL         4          // DATALOG
#define  INT16PL         4          // RTOSINT

// 0    = not used
// 1    = highest priority
// ...
// 8    = lowest priority
//
#define  G11PL           7          // SEQ1INT        (ADC)
#define  G12PL           6          // SEQ2INT        (ADC)
#define  G13PL           0          // reserved
#define  G14PL           1          // XINT1          (External)
#define  G15PL           3          // XINT2          (External)
#define  G16PL           2          // ADCINT         (ADC)
#define  G17PL           1          // TINT0          (CPU Timer 0)
#define  G18PL           5          // WAKEINT        (WD/LPM)
#define  G21PL           4          // EPWM1_TZINT (ePWM1 Trip)
#define  G22PL           3          // EPWM2_TZINT (ePWM2 Trip)
#define  G23PL           2          // EPWM3_TZINT (ePWM3 Trip)
#define  G24PL           1          // EPWM4_TZINT (ePWM4 Trip)
#define  G25PL           5          // EPWM5_TZINT (ePWM5 Trip)
```

```
#define  G26PL          6          // EPWM6_TZINT (ePWM6 Trip)
#define  G27PL          0          // reserved
#define  G28PL          0          // reserved
```

注：此优先级可以自己改动的

2.2 定时器 0 中断设置

2.2 定时器 0 中断设置

定时器 0 中断设置由以下几个步骤组成。

1）定时器 0 中断的基本条件

除了对周期寄存器（PRDH:PRD）及定时器分频器（TPRH:TPR）进行必要的设置之外，使能定时器 0 中断有两条必须的指令：

```
1.    CpuTimer0Regs.TCR.bit.TSS = 0; //启动定时器
2.    Timer->RegsAddr->TCR.bit.TIE = 1; //使能定时器中断
```

如果缺少上面两条指令中的一条，将不会产生中断。

2）确定中断向量的入口地址

主程序通过“DSP281x_PieVect.c”文件中的 InitPieVectTable()函数，已经为 PIE 向量表中的所有中断向量配置了对应向量的入口地址。如果针对某一个外设中断专门有一个中断服务程序，则这个中断服务程序的入口地址必须取代前面配置的入口地址，它由下面的指令完成：

```
1.    EALLOW; //允许访问受保护的寄存器
2.    PieVectTable.TINT0 = &cpu_timer0_isr; //取 cpu_timer0_isr 地址赋值给
3.    //TINT0 中断向量
4.    EDIS; //禁止访问受保护的寄存器
```

上面指令中，cpu_timer0_isr()函数是针对 TINT0 中断向量的一个中断服务程序，&cpu_timer0_isr 是该程序的入口地址。

3）使能 PIE 级及 CPU 级中断向量

这个步骤由下面 4 类指令完成：

先找出中断向量在 PIE 向量表中所在的组及在这组中所处的优先级。经查 TINT0 是第 1 组第 7 个中断，因此先通过 PIE 级指令“PieCtrlRegs.PIEIER1.bit.INTx7 =1;”，使能 PIE 第 1 组第 7 个 TINT0 中断（INTx7 由头文件定义为 PIE 中断使能寄存器（PIEIER）的第 7 位）。

通过 CPU 级的赋值指令“IER |= M_INT1;”，使能第 1 组（M_INT1 由头文件定义为 0x0001，指向第 1 组），即把 TINT0 中断汇集到 CPU 级的 INT1 中断线上。

通过“EINT;”指令使能全局中断，实际指令为：

```
1.    PieCtrlRegs.PIEIER1.bit.INTx7 = 1; //使能 PIE 中的 TINT0，1 组第 7 个中断
2.    IER |= M_INT1; //使能第 1 组中断
3.    EINT; //使能 INTM 全局中断
```

这里用了按位或复合运算符“|=”，其用意是不破坏 IER 原有结构。如果程序仅此一个中断，可以用 “IER = 0x0001;”指令，否则，会破坏 IER 原有结构。

使能 PIE 向量表，由下面一条指令完成。

```
1.    PieCtrlRegs.PIECRTL.bit.ENPIE = 1;
```

实际上这条指令包含在初始化 PIE 向量表 InitPieVectTable()函数中，主程序对这个函数已经调用。因此可省略。

4）中断服务程序

中断服务程序是以关键字 interrupt 开头的一个程序。通常在中断服务程序中有两条必须的指令：一条是中断应答，另一条是将中断标志位清 0。

定时器 0 中断应答指令为：

```
1.    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //PIEACK_GROUP1=0x0001
```

PIE 应答寄存器 PIEACK 是中断从 PIE 级进入 CPU 级的门禁。一个中断在进入 CPU 级之前，其对应的 PIEACK[x.1]必须通过软件清 0，打开后续 INTx 的 PIE 级到 CPU 的通道。而当这个中断进入 CPU 级 INTx 中断线时，硬件将 PIEACK[x.1]位置 1，关闭后续 INTx 的 PIE 级到 CPU 的通道。这条指令通过向 PIEACK[0]写 1，将 PIEACK[0]位清 0，从而打开后续 INT1 的 PIE 级到 CPU 级的中断。注意：PIEACK[x.1]对 INTx 中断线（1 x 12）。

源码的中断服务程序中没有将中断标志位清 0 的指令。当程序开始运行并执行到指令“CpuTimer0Regs.TCR.bit.TSS = 0;”时，定时器中断标志位 TIF(TCR[15])即从 0 变成 1。这意味着一旦启动定时器即触发定时器中断之后，即使通过软件将该位置 1，也不能将其清 0。

5）中断服务程序及中断初始化函数声明

如果中断服务程序及中断初始化函数放在主函数的下面，则在主函数头部要对中断服务程序及中断初始化函数进行声明：

```
1.    interrupt void cpu_timer0_isr();
```

如果中断服务函数放在主函数的上面，则不必进行声明。