第 5 章

时钟和系统控制

身体各个器官的动力来自于心脏，正是心脏一刻不停有规律地进行跳动，人们才能有健康的身体去工作，去学习，去做自己想做的事情。可是有时候，身体过度疲劳了，或者受到了外来细菌和病毒的感染，就会开始生病，这时候就需要医生来对身体进行检查并进行医治。其实，DSP 也一样，当然不仅仅是 DSP，其他的 CPU 也一样，都需要一个类似于心脏的模块来提供其正常运行的动力和节奏。下面将详细介绍 F28335 的“心脏”——振荡器、锁相环和时钟机制，此外还将讲解给 DSP 做“身体检查”，以维持其正常工作的看门狗模块。

5.1 振荡器 OSC 和锁相环 PLL

视频讲解

为了能够让 F28335 按部就班地执行相应的代码，实现功能，就得让 DSP 芯片“活”起来，除了得给 DSP 提供电源以外，还需要向 CPU 不断地提供规律的时钟脉冲，这一功能就由 F28335 内部的振荡器 OSC 和锁相环模块 PLL 来实现。图 5-1 为 F28335 芯片内的 OSC 和 PLL 时钟模块。

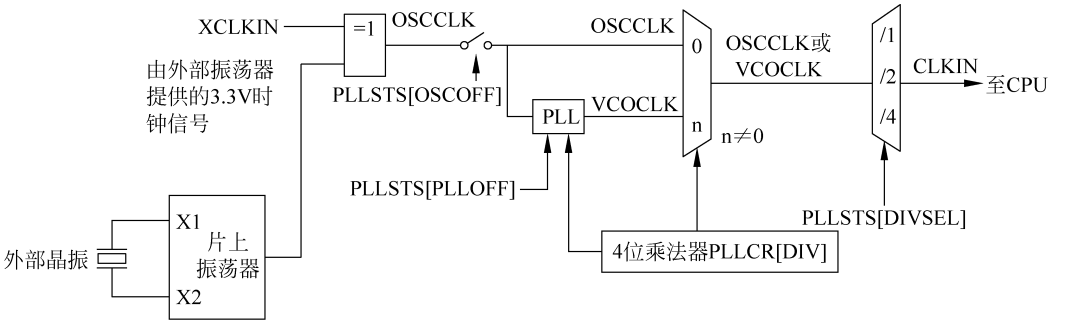

图 5-1　F28335 的 OSC 和 PLL 模块

先来简单介绍一下锁相环 PLL。锁相环是一种控制晶振使其相对于参考信号保持恒定的电路，在数字通信系统中使用比较广泛。目前 DSP 集成的片上锁相环 PLL 模块，主要作用是通过软件实时地配置片上外设时钟，提高系统的灵活性和可靠性。此外，由于采用软件可编程锁相环，所设计的处理器外部允许较低的工作频率，而片内经过锁相环模块提供较

高的系统时钟,这种设计可以有效地降低系统对外部时钟的依赖和电磁干扰,提高系统启动和运行时的可靠性,从而降低系统对硬件设计的要求。

从图 5-1 可以看到,外部晶振通过了片内振荡器 OSC 和 PLL 模块,产生了时钟信号 CLKIN,提供给 CPU。如果 PLL 状态寄存器 PLLSTS 的位 OSCOFF 为 1,则来自外部的振荡器时钟信号就不会送入 PLL;如果 OSCOFF 为 0,则来自外部的振荡器时钟信号送入 PLL。PLL 模块有 3 种工作模式,由 PLLSTS[PLLOFF]和 PLLCR[DIV]来决定。振荡器的时钟信号 OSCCLK 和送至 CPU 的时钟信号 SYSCLKOUT/CLKIN 之间的关系如表 5-1 所示。

表 5-1 OSCCLK 和送至 CPU 的时钟信号 SYSCLKOUT/CLKIN 之间的关系

PLL 工作模式	工作模式说明	PLLSTS[DIVSEL]	SYSCLKOUT/CLKIN
关闭 PLLSTS[PLLOFF]=1 PLLCR[DIV]=0	当 PLLSTS 的位 PLLOFF 为 1 时,PLL 模块关闭,从而可以减少系统噪声并减少功率损耗。在进入此模式前,需要将 PLLCR 寄存器设为 0x0000	0,1	OSCCLK/4
		2	OSCCLK/2
		3	OSCCLK/1
旁路 PLLSTS[PLLOFF]=0 PLLCR[DIV]=0	当 PLLSTS 的位 PLLOFF 为 0,且 PLLCR 的寄存器为 0x0000 时,PLL 被旁路,此时,时钟信号直接绕过 PLL 模块,但 PLL 模块却并没有被关闭	0,1	OSCCLK/4
		2	OSCCLK/2
		3	OSCCLK/1
使能 PLLSTS[PLLOFF]=0 PLLCR[DIV]=n(n>0)	通过给 PLLCR 寄存器写一个不为 0 的值来实现 PLL 的使能,时钟信号需要进入 PLL 模块进行 n 倍频,然后再被分频,最后送至 CPU	0,1	OSCCLK×n/4
		2	OSCCLK×n/2

需要注意的是,在写 PLLCR 寄存器前,PLLSTS[DIVSEL]必须是 0。在实际使用时,通常使用第三种方式,即 PLL 使能。从图 5-1 可以看到,通常使用 30MHz 晶振为 F28335 提供时基,因为当 PLLCR[DIV]设置为最大值 10,PLLSTS[DIVSEL]设置为 2 时,送至 CPU 的时钟可以达到 150MHz,这也是 F28335 所能支持的最高时钟频率。

视频讲解

5.2 各种时钟信号

5.2.1 外设时钟

F28335 芯片内各种外设时钟信号的产生情况如图 5-2 所示。CLKIN 是经过 PLL 模块后送往 CPU 的时钟信号,经过 CPU 分发,作为 SYSCLKOUT 送至各个外设。因此,SYSCLKOUT=CLKIN。

在使用 F28335 进行开发的时候,通常会用到一些外设,例如 SCI、SPI、I2C、CAN、PWM、CAP、QEP、McBSP、ADC、DMA 等。要使得这些外设正常工作,首要的就是向其提

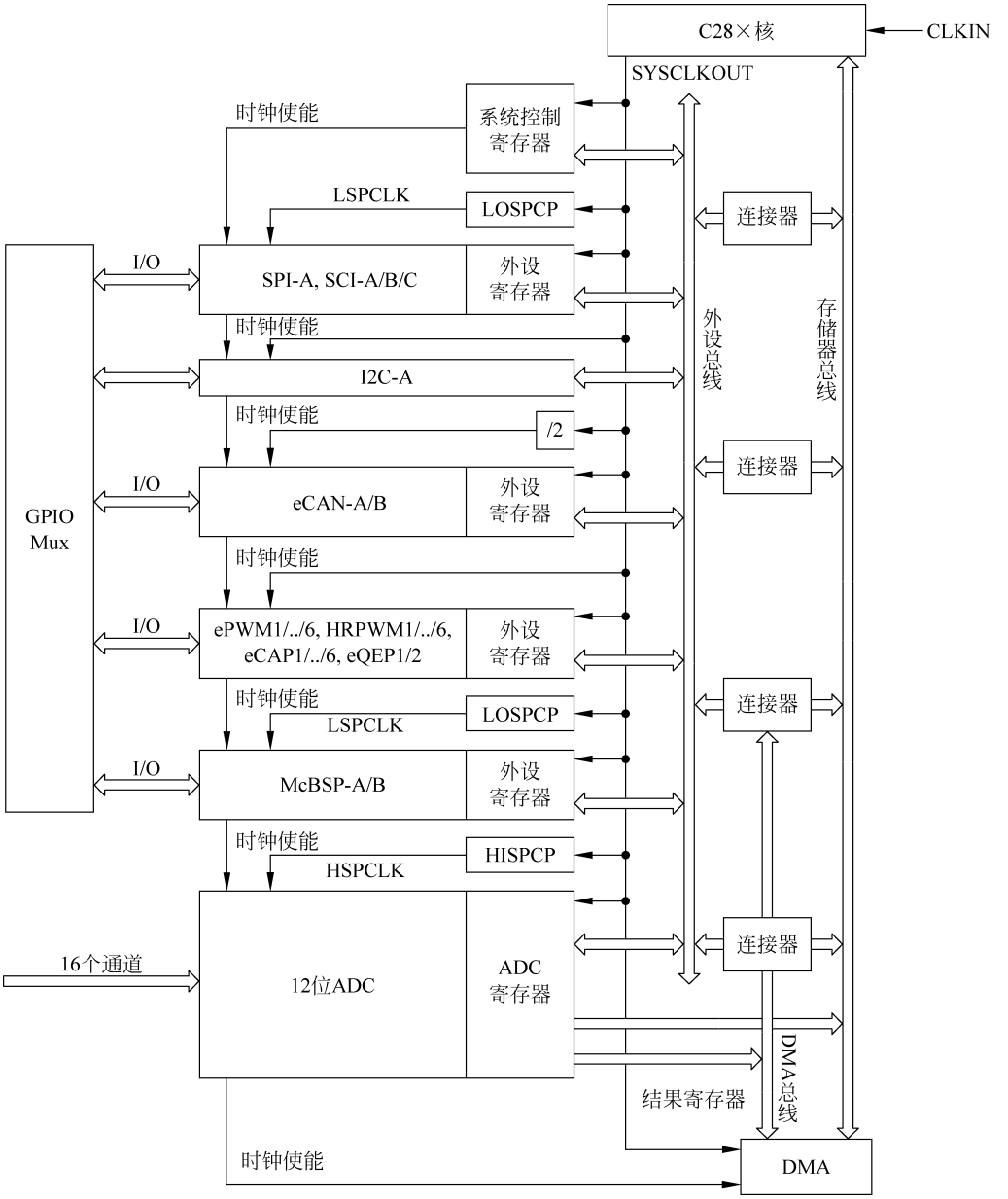

图 5-2　F28335 外设时钟信号产生情况

供时钟信号，因此，在系统初始化的时候，就需要对使用到的各个外设的时钟进行使能。假设现在某个项目中用到了 ADC、SCIA、eCAP1 和 GPIO 这 4 个外设，那么就需要按照下面的程序对这 4 个外设进行时钟的使能。和外设时钟使能相关的寄存器是外设时钟控制寄存器 PCLKCR0/1/3。本书相关的寄存器请打开 C2000 助手软件查阅。

```
SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1;                    //使能 ADC 的时钟
SysCtrlRegs.PCLKCR0.bit.SCIAENCLK = 1;                   //使能 SCIA 的时钟
SysCtrlRegs.PCLKCR1.bit.ECAP1ENCLK = 1;                  //使能 eCAP1 的时钟
SysCtrlRegs.PCLKCR3.bit.GPIOINENCLK = 1;                 //使能 GPIO 的时钟
```

由图 5-2 也能看到,SYSCLKOUT 信号经过低速外设时钟预定标寄存器 LOSPCP(取值范围为 0～7)变成了 LSPCLK,提供给低速外设 SCIA、SCIB、SCIC、SPI 和 McBSP; SYSCLKOUT 信号经过高速外设时钟预定标寄存器 HISPCP(取值范围为 0～7)变成了 HSPCLK,提供给外设 ADC; SYSCLKOUT 经过二分频后提供给外设 CAN; SYSCLKOUT 直接提供给了高速外设 PWM、CAP、QEP、DMA。当然在各个外设实际使用的时候,时钟源还需要经过外设各自的时钟预定标器,如果外设自己的时钟预定标位的值为 0,则外设实际使用的时钟就是各自的时钟源,即 LSPCLK、HSPCLK、SYSCLKOUT/2 或者 SYSCLKOUT。在实际使用时,为了降低系统功耗,不使用的外设最好将其时钟禁止。

LSPCLK 是低速外设时钟,HSPCLK 是高速外设时钟,那么 LSPCLK 的值有没有可能会比 HSPCLK 的值来的大? 也就是说,提供给低速外设的时钟频率反而要比提供给高速外设的时钟频率来得快? 从 LSPCLK 和 HSPCLK 的计算公式可以看出,这两个时钟信号的频率是独立无关的,各自分别取决于 LOSPCP 或者 HISPCP 的值,和其他因素没有关系。当给 LOSPCP 寄存器所赋的值小于给 HISPCP 寄存器所赋的值时,LOSPCP 的值就会大于 HSPCLK 的值。虽然这完全取决于用户对于寄存器的初始化,但是一般情况下,是不会让这样的情况出现的,因为低速外设的所需要的时钟毕竟要比高速外设所需要的时钟来得慢些,否则就不叫低速外设或者高速外设了。

```
If(LOSPCP = 0),then LSPCLK = SYSCLKOUT;
If(LOSPCP≠0),then LSPCLK = SYSCLKOUT/(2 × LOSPCP);
If(HISPCP = 0),then HSPCLK = SYSCLKOUT;
If(HISPCP≠0),then HSPCLK = SYSCLKOUT/(2 × HISPCP);
```

5.2.2 XCLKOUT 信号

F28335 提供了一路可以输出到芯片外部的时钟信号,即 XCLKOUT,通过对 SYSCLKOUT 分频可以得到不同的时钟频率,XCLKOUT 信号通路如图 5-3 所示。

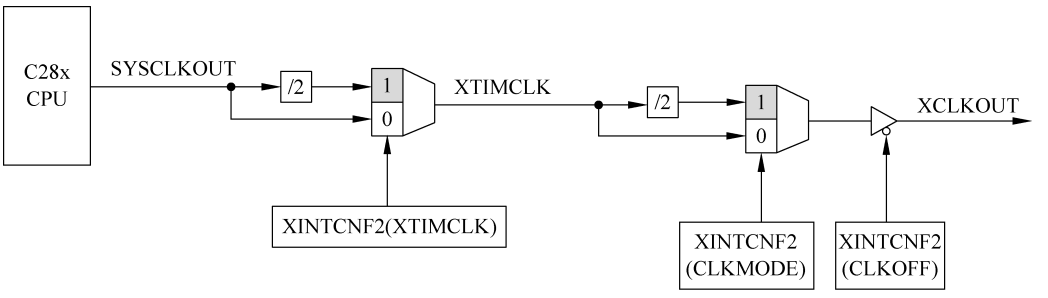

图 5-3 XCLKOUT 信号通路

经过配置，XCLKOUT 可以等于 SYSCLKOUT，也可以为其 1/2 或者 1/4。上电复位默认状态下，XCLKOUT 为 SYSCLKOUT 的 1/4。如果不使用 XCLKOUT 信号，可以通过 XINTCNF2 寄存器中的 CLKOFF 位将其关闭。

5.3　看门狗电路

在介绍 DSP 看门狗的内容之前，先来了解一下 MCU 中看门狗的原理，以便能够更好地理解 DSP 中的看门狗。在由 MCU 构成的微型计算机系统中，由于单片机的工作常常会受到来自外界电磁场的干扰，造成程序的跑飞，而陷入死循环，此时程序的正常运行被打断，由单片机控制的系统就无法继续工作，会造成整个系统陷入停滞状态，发生不可预料的后果，所以出于对单片机运行状态进行实时监测的考虑，便产生了一种专门用于监测单片机程序运行状态的电路，俗称“看门狗”(Watch Dog)。

看门狗电路的应用，使单片机可以在无人监控的状态下实现连续工作，其工作原理是：看门狗电路和单片机的一个 I/O 引脚相连，该 I/O 引脚通过程序控制定时地向看门狗电路送入高电平(或低电平)，这一程序语句是分散地放在单片机其他控制语句中间的，一旦单片机由于干扰造成程序跑飞后而陷入某一程序段，进入死循环状态时，写看门狗引脚的程序便不能被执行。这时，看门狗电路就会由于得不到单片机送来的信号而产生一个复位信号，使单片机发生复位，即程序从程序存储器的起始位置开始执行，这样便实现了单片机的自动复位。

F28335 的看门狗原理和上面讲述的 MCU 的看门狗原理是类似的，其作用是为 DSP 的运行情况进行“把脉”，一旦发现程序跑飞或者状态不正常，便立即使 DSP 复位，从而提高了系统的可靠性。F28335 看门狗电路的功能框图如图 5-4 所示。

从图 5-4 可以看到，F28335 的看门狗电路有一个 8 位的看门狗加法计数器 WDCNTR，无论什么时候，当 WDCNTR 计数到最大值时，看门狗模块就会产生一个输出脉冲，脉冲宽度为 512 个振荡器时钟宽度。为了防止看门狗加法计数器 WDCNTR 溢出，通常可以采用两种方法：一种是禁止看门狗，即使得计数器 WDCNTR 无效；另一种就是定期地“喂狗”，通过软件向负责复位看门狗计数器的看门狗密钥寄存器(8 位的 WDKEY)周期性地写入 0x55＋0xAA，紧跟着 0x55 写入 0xAA 能够清除 WDCNTR。当向 WDKEY 写 0x55 时，WDCNTR 复位到使能的位置；只有在向 WDKEY 写 0xAA 后才会使 WDCNTR 真正地被清除。写任何其他的值都会使系统立即复位。只要向 WDKEY 写 0x55 和 0xAA，无论写的顺序如何都不会导致系统复位，而只有先写 0x55，再写 0xAA 才会清除 WDCNTR。

逻辑校验位(WDCHK)是看门狗的另一个安全机制，所有访问看门狗控制寄存器(WDCR)的写操作中，相应的校验位 WDCHK 必须是 101，否则将会拒绝访问并立即触发系统复位。

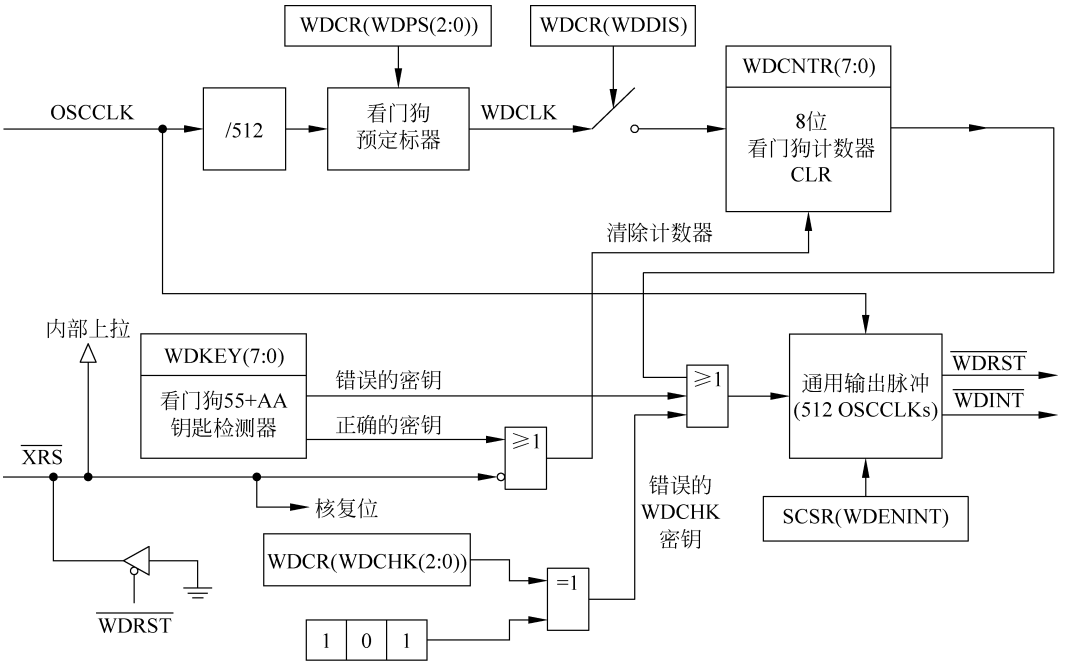

图 5-4 看门狗电路的功能框图

5.4 低功耗模式

F28335 的低功耗模式(Low Power Mode)如表 5-2 所示。F28335 的各种低功耗模式的操作如下：

(1) 空闲模式(IDLE)。只要把低功耗模块控制寄存器 LPMCR[1：0](LPMECR 的 D0、D1 位)都设置成 0，就可进入该模式。处理器可以通过任何使能的中断来退出空闲模式。

(2) 待命模式(STANDBY)。把低功耗模块控制寄存器 LPMCR[1：0]设置为 01b，可使器件进入 STANDBY 模式。在此模式下，CPU 的输入时钟信号 CLKIN 被禁用，从而关闭了所有从 SYSCLKOUT 分频得到的时钟信号，但内部振荡器、PLL 及看门狗依然工作。

当外部唤醒信号变为低电平后，必须保持足够的低电平时间才能将器件从 STANDBY 模式中唤醒，这个时间可以通过寄存器 LPMCR0 来设定。如果在量化周期内出现高电平，则需要重新开始采样。如果外部唤醒信号满足要求，那么在量化周期的最后时刻，PLL 使能 CPU 的输入时钟信号 CLKIN，并且 PIE 模块锁存 WAKEINT 中断，CPU 响应 WAKEINT 中断。

(3) 暂停模式(HALT)。把低功耗模块控制寄存器 LPMCR[1：0]设置为 1xb，可使器件进入 HALT 模式，这里 x 既可以是 0，也可以是 1。在此模式下，器件所有的时钟信号都

被禁止，内部振荡器、PLL 及看门狗电路停止工作。

复位 $\overline{\text{XRS}}$、GPIO PortA 和仿真器的信号能够从暂停模式唤醒 CPU。在 STANDBY 和 HALT 模式时，虽然 CPU 输入时钟 CLKIN 被关闭，仿真调试的 JTAG 端口仍可正常工作。

表 5-2　F28335 的低功耗模式

模式	LPMCR(1：0)	OSCCLK	CLKIN	SYSCLKOUT	退出方式
正常	X,X	开	开	开	
空闲 IDLE	0,0	开	开	开	$\overline{\text{XRS}}$ 看门狗中断 任何使能的中断
备用 STANDBY	0,1	开(看门狗仍在运行)	关	关	$\overline{\text{XRS}}$ 看门狗中断 GPIO PortA 信号 仿真器信号
暂停 HALT	1,X	关(振荡器和 PLL 关闭，看门狗不工作)	关	关	$\overline{\text{XRS}}$ GPIO PortA 信号 仿真器信号

5.5　实例：系统初始化函数

要使得 F28335 能够工作，在上电的时候就需要对 F28335 进行系统初始化，以提供其正常运行的基本条件，例如使能时钟信号，这是通过系统初始化函数来实现的。那么，系统初始化函数应该怎么写？需要在系统初始化函数中写哪些内容？需要注意些什么呢？接下来会通过详细的代码进行说明。系统初始化函数 InitSysCtrl 一般在工程的 DSP2833x_SysCtrl.c 文件中。

程序清单 5-1　系统初始化函数

```
//###########################################################################
//FILE: DSP2833x_SysCtrl.c
//TITLE: DSP2833x Device System Control Initialization & Support Functions.
//###########################################################################
#include "DSP2833x_Device.h"                    //包含头文件

#include "DSP2833x_Examples.h"                  //包含头文件

//系统初始化函数
void InitSysCtrl(void)
{
   DisableDog();                                //关闭看门狗电路
```

```
//初始化锁相环 PLL
   //DSP28_PLLCR 和 DSP28_DIVSEL 定义在 DSP2833x_Examples.h 中
   //DSP28_PLLCR = 10,DSP28_DIVSEL = 2,见 DSP2833x_Examples.h
   InitPll(DSP28_PLLCR,DSP28_DIVSEL);

   //初始化外设时钟
   InitPeripheralClocks();
}

//喂看门狗函数
void ServiceDog(void)
{
    EALLOW;
    SysCtrlRegs.WDKEY = 0x0055;
    SysCtrlRegs.WDKEY = 0x00AA;
    EDIS;
}

//关闭看门狗函数
void DisableDog(void)
{
    EALLOW;
    SysCtrlRegs.WDCR = 0x0068;
    EDIS;
}

//PLL 初始化函数
void InitPll(Uint16 val, Uint16 divsel)
{
   //在配置 PLLCR 前,DIVSEL 位必须是 0
   if (SysCtrlRegs.PLLSTS.bit.DIVSEL != 0)
   {
       EALLOW;
       SysCtrlRegs.PLLSTS.bit.DIVSEL = 0;
       EDIS;
   }

   //改变 PLLCR 的值
   if (SysCtrlRegs.PLLCR.bit.DIV != val)
   {
      EALLOW;
      //在设置 PLLCR 前,需要关闭是检测是否丢失时钟的逻辑
      SysCtrlRegs.PLLSTS.bit.MCLKOFF = 1;
      SysCtrlRegs.PLLCR.bit.DIV = val;
      EDIS;
```

```
     //关闭看门狗,调试时常关闭
       DisableDog();
       EALLOW;
       SysCtrlRegs.PLLSTS.bit.MCLKOFF = 0;          //继续检测是否丢失时钟信号
       EDIS;
     }
     if((divsel == 1)||(divsel == 2))
     {
         EALLOW;
         SysCtrlRegs.PLLSTS.bit.DIVSEL = divsel;
         EDIS;
     }
     if(divsel == 3)
     {
         EALLOW;
         SysCtrlRegs.PLLSTS.bit.DIVSEL = 2;
         DELAY_US(50L);
         SysCtrlRegs.PLLSTS.bit.DIVSEL = 3;
         EDIS;
     }
}
//外设时钟初始化函数,使能各个外设的时钟
void InitPeripheralClocks(void)
{
   EALLOW;

//设置 HISPCP/LOSPCP 预分频寄存器, 通常设置成默认值
SysCtrlRegs.HISPCP.all = 0x0001;
   SysCtrlRegs.LOSPCP.all = 0x0002;

//默认 XCLKOUT = 1/4 SYSCLKOUT
   //XTIMCLK = SYSCLKOUT/2
   XintfRegs.XINTCNF2.bit.XTIMCLK = 1;
   //XCLKOUT = XTIMCLK/2
   XintfRegs.XINTCNF2.bit.CLKMODE = 1;
   //使能 XCLKOUT
   XintfRegs.XINTCNF2.bit.CLKOFF = 0;
SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1;                //ADC
SysCtrlRegs.PCLKCR0.bit.I2CAENCLK = 1;               //I2C
SysCtrlRegs.PCLKCR0.bit.SCIAENCLK = 1;               //SCIA
SysCtrlRegs.PCLKCR0.bit.SCIBENCLK = 1;               //SCIB
SysCtrlRegs.PCLKCR0.bit.SCICENCLK = 1;               //SCIC
SysCtrlRegs.PCLKCR0.bit.SPIAENCLK = 1;               //SPIA
SysCtrlRegs.PCLKCR0.bit.MCBSPAENCLK = 1;             //McBSPA
SysCtrlRegs.PCLKCR0.bit.MCBSPBENCLK = 1;             //McBSPB
SysCtrlRegs.PCLKCR0.bit.ECANAENCLK = 1;              //eCANA
```

```
    SysCtrlRegs.PCLKCR0.bit.ECANBENCLK = 1;                 //eCANB

    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;                  //禁止 ePWM 的 TBCLK
    SysCtrlRegs.PCLKCR1.bit.EPWM1ENCLK = 1;                 //ePWM1
    SysCtrlRegs.PCLKCR1.bit.EPWM2ENCLK = 1;                 //ePWM2
    SysCtrlRegs.PCLKCR1.bit.EPWM3ENCLK = 1;                 //ePWM3
    SysCtrlRegs.PCLKCR1.bit.EPWM4ENCLK = 1;                 //ePWM4
    SysCtrlRegs.PCLKCR1.bit.EPWM5ENCLK = 1;                 //ePWM5
    SysCtrlRegs.PCLKCR1.bit.EPWM6ENCLK = 1;                 //ePWM6
    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;                  //使能 ePWM 的 TBCLK

    SysCtrlRegs.PCLKCR1.bit.ECAP3ENCLK = 1;                 //eCAP3
    SysCtrlRegs.PCLKCR1.bit.ECAP4ENCLK = 1;                 //eCAP4
    SysCtrlRegs.PCLKCR1.bit.ECAP5ENCLK = 1;                 //eCAP5
    SysCtrlRegs.PCLKCR1.bit.ECAP6ENCLK = 1;                 //eCAP6
    SysCtrlRegs.PCLKCR1.bit.ECAP1ENCLK = 1;                 //eCAP1
    SysCtrlRegs.PCLKCR1.bit.ECAP2ENCLK = 1;                 //eCAP2
    SysCtrlRegs.PCLKCR1.bit.EQEP1ENCLK = 1;                 //eQEP1
    SysCtrlRegs.PCLKCR1.bit.EQEP2ENCLK = 1;                 //eQEP2

    SysCtrlRegs.PCLKCR3.bit.CPUTIMER0ENCLK = 1;             //CPU 定时器 0
    SysCtrlRegs.PCLKCR3.bit.CPUTIMER1ENCLK = 1;             //CPU 定时器 1
    SysCtrlRegs.PCLKCR3.bit.CPUTIMER2ENCLK = 1;             //CPU 定时器 2

    SysCtrlRegs.PCLKCR3.bit.DMAENCLK = 1;                   //DMA 时钟
    SysCtrlRegs.PCLKCR3.bit.XINTFENCLK = 1;                 //XTIMCLK
    SysCtrlRegs.PCLKCR3.bit.GPIOINENCLK = 1;                //GPIO 时钟

    EDIS;
    }
```

这里需要对 EALLOW 和 EDIS 做一些说明。为了提高安全性能，TI 的 DSP 对很多关键寄存器作了保护处理。通过状态寄存器 1(ST1)的位 D6 设置与复位，来决定是否允许 DSP 指令对关键寄存器进行操作。

这些关键寄存器包括器件仿真寄存器、Flash 寄存器、CSM 寄存器、PIE 向量表、系统控制寄存器、GPIO MUX 寄存器、eCAN 寄存器的一部分。

DSP 由于在上电复位之后，状态寄存器基本上都是清零，而这样的状态下正是上述特殊寄存器禁止改写的状态。为了能够对这些特殊寄存器进行初始化，所以在对上述特殊寄存器进行改写之前，一定要执行汇编指令 asm("EALLOW")或者宏定义 EALLOW 来设置状态寄存器 1 的 D6 位。在设置完寄存器之后，一定要注意执行汇编指令 asm("EDIS")或者宏定义 EDIS 来清除状态寄存器 1 的 D6 位。

在工程的头文件 DSP2833x_Device.h 中可以找到“#define EALLOW asm("EALLOW")”语句。

本章详细介绍了 F28335 的系统时钟模块、看门狗电路及低功耗模式等内容，介绍了如何写系统初始化函数。第 6 章将详细介绍通用输入/输出端口 GPIO 的内容。

习题

5-1　当 PLLSTS[PLLOFF]＝0，PLLCR[DIV]＝n(n＞0)，PLLSTS[DIVSEL]＝2 时，SYSCLKOUT 和 OSCCLK 是什么关系？

5-2　请写语句使能外设 ECAP2 的时钟。

5-3　F28335 看门狗电路的作用是什么？请简述其工作原理。

5-4　F28335 有哪 3 种低功耗模式？