时钟和系统控制

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，人们才能有个健康的身体去工作，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PLL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。

振荡器OSC和锁相环PLL

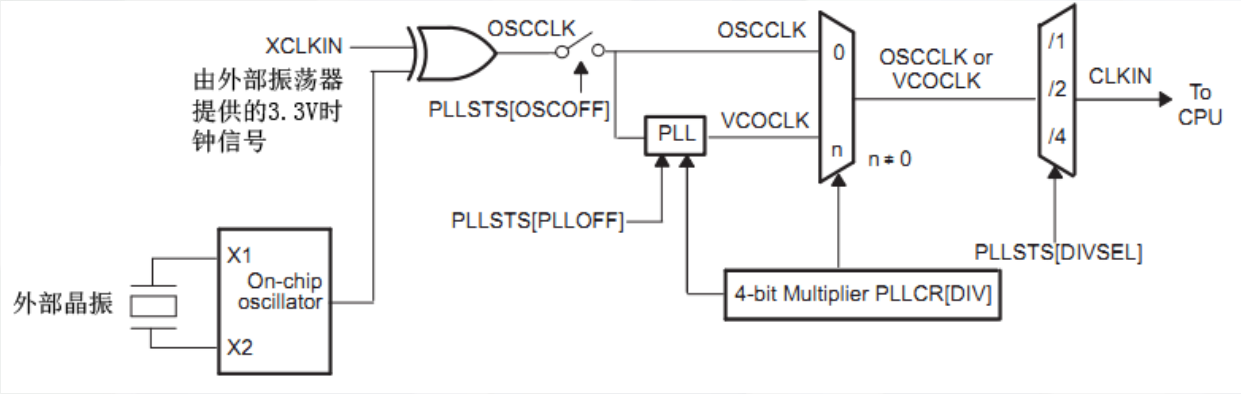


图5-1 F28335的OSC和PLL模块

为了能够让F28335按部就班的执行相应的代码，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不断的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来实现了。图5-1为F28335芯片内的OSC和PLL时钟模块。

振荡器OSC和锁相环PLL

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，这种设计可以有效的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，降低了系统对硬件设计的要求。

振荡器OSC和锁相环PLL

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模块有三种工作模式，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。

振荡器OSC和锁相环PLL

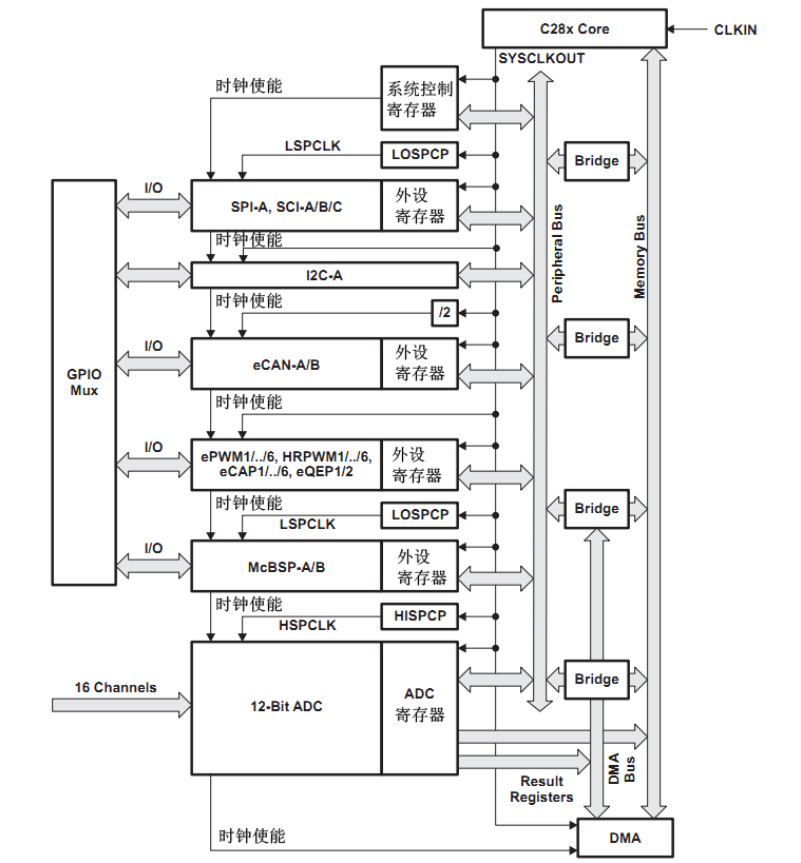
PLL工作模式	工作模式说明	PLLSTS[DIVSEL]	SYSCLKOUT/CLKIN
关闭 PLLSTS[PLLOFF]=1 PLLCR[DIV]=0	当PLLSTS的位PLLOFF为1时，PLL模块关闭，从而可以减少系统噪声并减少功率损耗。在进入此模式前，需要将PLLCR寄存器设为0x0000。	0,1	OSCCLK/4
		2	OSCCLK/2
		3	OSCCLK/1
旁路 PLLSTS[PLLOFF]=0 PLLCR[DIV]=0	当PLLSTS的位PLLOFF为0，且PLLCR的寄存器为0x0000时，PLL被旁路，此时，时钟信号直接绕过PLL模块，但PLL模块却并没有被关闭。	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

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

振荡器OSC和锁相环PLL

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30M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的时候，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。

各种时钟信号·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。

图5-2 F28335外设时钟信号产生情况

（此处可右键选择“放大”功能查看图像）

各种时钟信号·外设时钟

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，假设现在某个项目里用到了ADC、SCIA、eCAP1和GPIO这4个外设，那么就需要按照下面的程序对这个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
```

各种时钟信号·外设时钟

从图5-2上也能看到，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0~7)变成了HSPCLK，提供给外设ADC；SYSCLKOUT经过2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的话，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。

各种时钟信号·外设时钟

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的快？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，也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的慢些，否则就不叫低速外设了或者高速外设了。

各种时钟信号·外设时钟

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);

各种时钟信号·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。

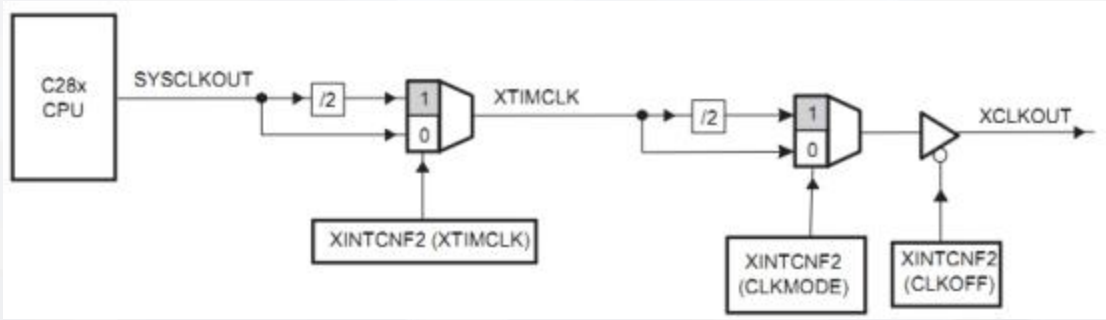


图5-3 XCLKOUT信号通路

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。

各种时钟信号·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中的开门狗。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"(WatchDog)。

各种时钟信号·看门狗电路

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往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。这个时候，看门狗电路就会由于得不到单片机送来的信号，便会产生一个复位信号，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。

各种时钟信号·看门狗电路

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。

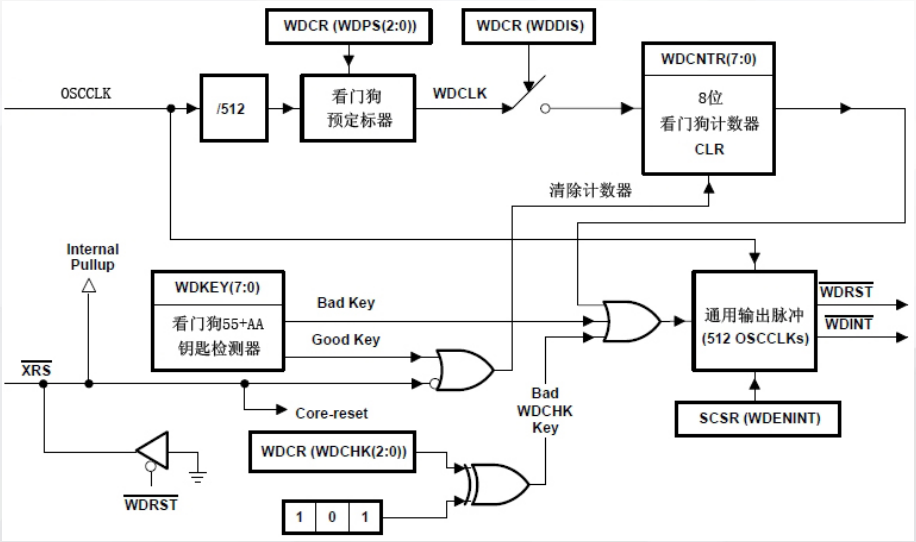


图5-4 看门狗电路的功能框图

各种时钟信号·看门狗电路

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，如果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；另一种就是定期的“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）周期性的写入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。

各种时钟信号·看门狗电路

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。

各种时钟信号·低功耗模式

F28335的低功耗模式(Low Power Modes)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.待命方式(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，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是指既可以是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。

各种时钟信号·低功耗模式

复位$\overline{XRS}$、GPIO PortA和仿真器的信号能够从HALT方式唤醒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。

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	$\overline{XRS}$ 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	$\overline{XRS}$ 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	$\overline{XRS}$ GPIO PortA信号 仿真器信号

表5-2 F28335的低功耗模式

各种时钟信号·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。

各种时钟信号·实例：系统初始化函数

在这里，需要对EALLOW和EDIS做一些说明。 TI的DSP为了提高安全性能，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。

各种时钟信号·实例：系统初始化函数

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。