第 10 章 节能模式

目录

本章包括下列主题：

10.1 简介

本章介绍 PIC32MX 器件系列的节能工作模式。PIC32MX 器件具有 9 种低功耗模式（分为两类），允许用户在功耗和器件性能之间寻求平衡。在下面所述的所有模式中，器件可以通过软件选择所需的节能模式。

10.1.1 CPU 运行模式

在 CPU 运行模式下， CPU 保持运行，而外设可以选择开启或关闭。

- FRC RUN （运行）模式：CPU 时钟来自 FRC 时钟源 （带或不带后分频器）。
- LPRC RUN （运行）模式：CPU 时钟来自 LPRC 时钟源。
- SOSC RUN （运行）模式：CPU 时钟来自 SOSC 时钟源。
- 外设总线分频模式：
  外设所用时钟的频率为 CPU 时钟 （SYSCLK）的可编程分频。

10.1.2 CPU 暂停模式

在 CPU 暂停模式下， CPU 会暂停。根据所处模式，外设可以继续工作，也可以暂停。

- POSC IDLE （空闲）模式：系统时钟来自 POSC。系统时钟源继续工作。
  外设继续工作，但可以选择单独禁止。
- FRC IDLE （空闲）模式：系统时钟来自 FRC （带或不带后分频器）。
  外设继续工作，但可以选择单独禁止。
- SOSC IDLE （空闲）模式：系统时钟来自 SOSC。
  外设继续工作，但可以选择单独禁止。
- LPRC IDLE （空闲）模式：系统时钟来自 LPRC。
  外设继续工作，但可以选择单独禁止。这是时钟运行时器件的最低功耗模式。
- SLEEP （休眠）模式：暂停 CPU、系统时钟源以及工作在系统时钟源下的任何外设。
  使用特定时钟源的某些外设可在 Sleep （休眠）模式下继续工作。这是器件的最低功耗模式。

10.2 节能模式控制寄存器

节能模式控制包含以下特殊功能寄存器 （Special Function Register， SFR）：

- OSCCON：振荡器模块控制寄存器
  OSCCONCLR、 OSCCONSET 和 OSCCONINV：OSCCON 的原子级位操作只写寄存器
- WDTCON：看门狗定时器模块控制寄存器
  WDTCONCLR、 WDTCONSET 和 WDTCONINV：WDTCON 的原子级位操作只写寄存器
- RCON：复位模块控制寄存器
  RCONCLR、 RCONSET 和 RCONINV：RCON 的原子级位操作只写寄存器

下表汇总了所有与节能模式相关的寄存器。该汇总表之后列出了相应的寄存器，并且每个寄存器均附有详细的说明。

表 10-1： 节能模式 SFR 汇总

名称		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
OSCCON	31:24	—	—	PLLODIV<2:0>			FRCDIV<2:0>		
	23:16	—	SOSCRDY	—	PBDIV<1:0>		PLLMULT<2:0>		
	15:8	—	COSC<2:0>			—	NOSC<2:0>		
	7:0	CLKLOCK	ULOCK	LOCK	SLPEN	CF	UFRCEN	SOSCEN	OSWEN
OSCCONCLR	31:0	写入时会将 OSCCON 中的选定位清零，读取时获得的值未定义							
OSCCONSET	31:0	写入时会将 OSCCON 中的选定位置 1，读取时获得的值未定义							
OSCCONINV	31:0	写入时会将 OSCCON 中的选定位取反，读取时获得的值未定义							
WDTCON	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	ON	—	—	—	—	—	—	—
	7:0	—	SWDTPS<4:0>					—	WDTCLR
WDTCONCLR	31:0	写入时会将 WDTCON 中的选定位清零，读取时获得的值未定义							
WDTCONSET	31:0	写入时会将 WDTCON 中的选定位置 1，读取时获得的值未定义							
WDTCONINV	31:0	写入时会将 WDTCON 中的选定位取反，读取时获得的值未定义							
RCON	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	CM	VREGS
	7:0	EXTR	SWR	—	WDTO	SLEEP	IDLE	BOR	POR
RCONCLR	31:0	写入时会将 RCON 中的选定位清零，读取时获得的值未定义							
RCONSET	31:0	写入时会将 RCON 中的选定位置 1，读取时获得的值未定义							
RCONINV	31:0	写入时会将 RCON 中的选定位取反，读取时获得的值未定义							

寄存器 10-1： OSCCON：振荡器控制寄存器

r-x	r-x	R/W-x	R/W-x	R/W-x	R/W-0	R/W-0	R/W-1
—	—	PLLODIV<2:0>			FRCDIV<2:0>		
bit 31							bit 24

r-x	R-0	r-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	SOSCRDY	—	PBDIV<1:0>		PLLMULT<2:0>		
bit 23							bit 16

r-x	R-0	R-0	R-0	r-x	R/W-x	R/W-x	R/W-x
—	COSC<2:0>			—	NOSC<2:0>		
bit 15							bit 8

R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-x	R/W-0
CLKLOCK	ULOCK	LOCK	SLPEN	CF	UFRCEN	SOSCEN	OSWEN
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 29-27 **PLLODIV<2:0>**：PLL 输出分频比位
111 = PLL 输出 256 分频
110 = PLL 输出 64 分频
101 = PLL 输出 32 分频
100 = PLL 输出 16 分频
011 = PLL 输出 8 分频
010 = PLL 输出 4 分频
001 = PLL 输出 2 分频
000 = PLL 输出 1 分频
注： 发生复位时，这些位被设置为 FPLLODIV 配置位 （DEVCFG2<18:16>）的值。

bit 26-24 **FRCDIV<2:0>**：快速内部 RC 时钟分频比位
111 = FRC 256 分频
110 = FRC 64 分频
101 = FRC 32 分频
100 = FRC 16 分频
011 = FRC 8 分频
010 = FRC 4 分频
001 = FRC 2 分频 （默认设置）
000 = FRC 1 分频

bit 23 **保留**：写入 0；忽略读操作

bit 22 **SOSCRDY**：辅助振荡器就绪指示位
1 = 指示辅助振荡器正在运行并且已稳定
0 = 辅助振荡器已关闭或仍在预热阶段

bit 21 **未实现**：读为 0

寄存器 10-1： OSCCON：振荡器控制寄存器 （续）

bit 20-19 **PBDIV<1:0>：**外设总线时钟分频比位
11 = PBCLK 是 SYSCLK 的 8 分频 （默认）
10 = PBCLK 是 SYSCLK 的 4 分频
01 = PBCLK 是 SYSCLK 的 2 分频
00 = PBCLK 是 SYSCLK 的 1 分频
注： 发生复位时，这些位被设置为配置位 （DEVCFG1<13:12>）的值。

bit 18-16 **PLLMULT<2:0>：**PLL 倍频比位
111 = 时钟进行 24 倍频
110 = 时钟进行 21 倍频
101 = 时钟进行 20 倍频
100 = 时钟进行 19 倍频
011 = 时钟进行 18 倍频
010 = 时钟进行 17 倍频
001 = 时钟进行 16 倍频
000 = 时钟进行 15 倍频
注： 发生复位时，这些位被设置为 PLLMULT 配置位 （DEVCFG2<6:4>）的值。

bit 15 **保留：**写入 0；忽略读操作

bit 14-12 **COSC<2:0>：**当前振荡器选择位
111 = 快速内部 RC 振荡器按照 OSCCON<FRCDIV> 位进行分频
110 = 快速内部 RC 振荡器 16 分频
101 = 低功耗内部 RC 振荡器 （LPRC）
100 = 辅助振荡器 （SOSC）
011 = 带 PLL 模块的主振荡器 （XTPLL、 HSPLL 或 ECPLL）
010 = 主振荡器 （XT、 HS 或 EC）
001 = 带 PLL 模块的快速 RC 振荡器 （通过后分频器）（FRCPLL）
000 = 快速 RC 振荡器 （FRC）
注： 发生复位时，这些位被设置为 FNOSC 配置位 （DEVCFG1<2:0>）的值。

bit 11 **保留：**写入 0；忽略读操作

bit 10-8 **NOSC<2:0>：**新振荡器选择位
111 = 快速内部 RC 振荡器按照 OSCCON<FRCDIV> 位进行分频
110 = 快速内部 RC 振荡器 16 分频
101 = 低功耗内部 RC 振荡器 （LPRC）
100 = 辅助振荡器 （SOSC）
011 = 带 PLL 模块的主振荡器 （XTPLL、 HSPLL 或 ECPLL）
010 = 主振荡器 （XT、 HS 或 EC）
001 = 带 PLL 模块的快速内部 RC 振荡器 （通过后分频器）（FRCPLL）
000 = 快速内部 RC 振荡器 （FRC）
发生复位时，这些位被设置为 FNOSC 配置位 （DEVCFG1<2:0>）的值。

bit 7 **CLKLOCK：**时钟选择锁定使能位
如果使能 FSCM （FCKSM1 = 1）：
1 = 时钟和 PLL 选择被锁定
0 = 时钟和 PLL 选择未被锁定，可以被修改
如果禁止 FSCM （FCKSM1 = 0）：
时钟和 PLL 选择永不锁定，可以被修改

bit 6 **ULOCK：**USB PLL 锁定状态位
1 = 指示 USB PLL 模块处于锁定状态或 USB PLL 模块起振定时器延时结束
0 = 指示 USB PLL 模块处于失锁状态、 USB PLL 模块起振定时器正在运行或 USB PLL 被禁止

bit 5 **LOCK：**PLL 锁定状态位
1 = PLL 模块处于锁定状态或 PLL 模块起振定时器延时结束
0 = PLL 模块处于失锁状态、 PLL 起振定时器正在运行或 PLL 被禁止

bit 4 **SLPEN：**SLEEP （休眠）模式使能位
1 = 执行 WAIT 指令后器件将进入 SLEEP （休眠）模式
0 = 执行 WAIT 指令后器件将进入 IDLE （空闲）模式

寄存器 10-1： OSCCON：振荡器控制寄存器 （续）

bit 3 **CF**：时钟故障检测位
1 = 故障保护时钟监视器 （Fail Safe Clock Monitor， FSCM）检测到时钟故障
0 = 未检测到时钟故障

bit 2 **UFRCEN**：USB FRC 时钟使能位
1 = 将 FRC 使能为 USB 时钟源
0 = 将主振荡器或 USB PLL 用作 USB 时钟源

bit 1 **SOSCEN**：32.768 kHz 辅助振荡器 （SOSC）使能位
1 = 使能辅助振荡器
0 = 禁止辅助振荡器
注： 发生复位时，该位被设置为 FSOSCEN 配置位 （DEVCFG1<5>）的值。

bit 0 **OSWEN**：振荡器切换使能位
1 = 启动振荡器切换，切换为 NOSC2:NOSC0 位指定的振荡器
0 = 振荡器切换完成

寄存器 10-2： OSCCONCLR：振荡器控制清零寄存器

写入时会将 OSCCON 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OSCCON 中的选定位清零**

在一个或多个位中写入 1 会将 OSCCON 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例： OSCCONCLR = 0x00000001 时，会将 OSCCON 寄存器中的 bit 0 清零。

寄存器 10-3： OSCCONSET：振荡器控制置 1 寄存器

写入时会将 OSCCON 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OSCCON 中的选定位置 1**

在一个或多个位中写入 1 会将 OSCCON 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例： OSCCONSET = 0x00000001 时，会将 OSCCON 寄存器中的 bit 0 置 1。

寄存器 10-4： OSCCONINV：振荡器控制取反寄存器

写入时会将 OSCCON 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OSCCON 中的选定位取反**

在一个或多个位中写入 1 会将 OSCCON 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例： OSCCONINV = 0x00000001 时，会将 OSCCON 寄存器中的 bit 0 取反。

寄存器 10-5： WDTCON：看门狗定时器控制寄存器

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	r-x	r-x	r-x	r-x	r-x	r-x	r-x
ON	—	—	—	—	—	—	—
bit 15							bit 8

r-x	R-x	R-x	R-x	R-x	R-x	r-0	R/W-0
—	SWDTPS<4:0>					—	WDTCLR
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 15 **ON**：看门狗外设使能位

1 = 使能看门狗外设。寄存器中其他位的状态不会受该位置 1 影响。进入 Sleep（休眠）模式时，LPRC 振荡器不会被禁止。

0 = 禁止看门狗外设，不会消耗电流。允许进行 SFR 修改。该寄存器中其他位的状态不会受该位清零影响。

注： 使用 1:1 PBCLK 分频比时，在清零模块 ON 位的指令之后，用户的软件不应立即在 SYSCLK 周期中读 / 写外设的 SFR。

寄存器 10-6： WDTCONCLR：看门狗定时器控制清零寄存器

写入时会将 WDTCON 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 WDTCON 中的选定位清零**

在一个或多个位中写入 1 会将 WDTCON 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例： WDTCONCLR = 0x00008001 时，会将 WDTCON 寄存器中的 bit 15 和 bit 0 清零。

寄存器 10-7： WDTCONSET：看门狗定时器控制置 1 寄存器

写入时会将 WDTCON 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 WDTCON 中的选定位置 1**

在一个或多个位中写入 1 会将 WDTCON 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例： WDTCONSET = 0x00008001 时，会将 WDTCON 寄存器中的 bit 15 和 bit 0 置 1。

寄存器 10-8： WDTCONINV：看门狗定时器控制取反寄存器

写入时会将 WDTCON 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 WDTCON 中的选定位取反**

在一个或多个位中写入 1 会将 WDTCON 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例： WDTCONINV = 0x00008001 时，会将 WDTCON 寄存器中的 bit 15 和 bit 0 取反。

寄存器 10-9： RCON：复位控制寄存器

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-0	R/W-0	R/W-0
—	—	—	—	—	—	CM	VREGS
bit 15							bit 8

R/W-0	R/W-0	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EXTR	SWR	—	WDTO	SLEEP	IDLE	BOR	POR
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 3 **SLEEP**：从休眠模式唤醒位

1 = 器件是从 SLEEP （休眠）模式唤醒的
0 = 器件不是从 SLEEP （休眠）模式唤醒的
注： 只有清零该位时，才能检测下一次从 SLEEP （休眠）模式唤醒的情况。

bit 2 **IDLE**：从空闲模式唤醒位

1 = 器件是从 IDLE （空闲）模式唤醒的
0 = 器件不是从 IDLE （空闲）模式唤醒的
注： 只有清零该位时，才能检测下一次从 IDLE （空闲）模式唤醒的情况。

寄存器 10-10： RCONCLR：RCON 清零寄存器

写入时会将 RCON 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 RCON 中的选定位清零**

在一个或多个位中写入 1 会将 RCON 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例： RCONCLR = 0x0000000C 时，会将 RCON 寄存器中的 bit 3 和 bit 2 清零。

寄存器 10-11： RCONSET：RCON 置 1 寄存器

写入时会将 RCON 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 RCON 中的选定位置 1**

在一个或多个位中写入 1 会将 RCON 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例： RCONSET = 0x0000000C 时，会将 RCON 寄存器中的 bit 3 和 bit 2 置 1。

寄存器 10-12： RCONINV：RCON 取反寄存器

写入时会将 RCON 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 RCON 中的选定位取反**

在一个或多个位中写入 1 会将 RCON 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例： RCONINV = 0x0000000C 时，会将 RCON 寄存器中的 bit 3 和 bit 2 取反。

10.3 节能模式下的操作

注： 在本手册中，对于特定模块中使用的功耗模式和器件使用的功耗模式进行了区分；例如，比较器的 Sleep（休眠）模式和 CPU 的 SLEEP（休眠）模式。为了指示所期望功耗模式的类型，模块功耗模式使用大写字母加小写字母 （Sleep， Idle，Debug）（休眠、空闲和调试）来表示，器件功耗模式使用全大写字母 （SLEEP，IDLE， DEBUG）（休眠、空闲和调试）来表示。

PIC32MX 器件系列具有 9 种低功耗模式。所有节能模式的目的都是通过降低器件时钟频率来降低功耗。要实现该目的，可以选择多种低频时钟源。此外，还可以暂停或禁止外设和 CPU，以进一步降低功耗。

10.3.1 SLEEP （休眠）模式

SLEEP（休眠）模式是器件节能工作模式中的最低功耗模式。 CPU 和大部分外设暂停。选定外设可以在 Sleep（休眠）模式下继续工作并可用于将器件从 Sleep（休眠）模式唤醒。关于 Sleep（休眠）模式下外设行为的描述，请参见各外设模块章节。

SLEEP （休眠）模式具有以下特性：

- CPU 暂停。
- 系统时钟源通常关闭。具体信息，请参见**第 10.3.1.1 节 “SLEEP （休眠）模式下的振荡器关闭”**。
- 针对所选的振荡器存在不同的唤醒延时 （见表 10-2）。
- Sleep （休眠）模式期间，故障保护时钟监视器 （FSCM）不工作。
- 如果使能了 BOR 电路，那么在 SLEEP （休眠）模式期间，该电路继续工作。
- 如果使能了 WDT，它在进入 SLEEP （休眠）模式之前不会自动清零。
- 某些外设可在 SLEEP （休眠）模式下继续工作。这些外设包括检测输入信号电平变化的 I/O 引脚、 WDT、 RTCC、 ADC、 UART 以及使用外部时钟输入或内部 LPRC 振荡器的外设。
- I/O 引脚将继续按照器件未处于 Sleep （休眠）模式下的方式拉或灌电流。
- USB 模块可改写 POSC 或 FRC 的禁止状态。具体信息请参见 USB 章节。
- 为了进一步降低功耗，可在进入 SLEEP （休眠）模式之前用软件单独禁止模块。

发生以下任一事件时，处理器将退出 SLEEP （休眠）模式或从 SLEEP （休眠）模式 “唤醒”：

- 在 Sleep （休眠）模式下继续工作的已允许中断源的任何中断。中断优先级必须大于当前 CPU 优先级。
- 任何形式的器件复位。
- WDT 超时。请参见**第 10.4.2 节 “在看门狗超时时从 SLEEP （休眠）或 IDLE （空闲）模式唤醒 （NMI）”**。

如果中断优先级小于或等于当前优先级，CPU 将保持暂停，但 PBCLK 将开始运行且器件将进入 IDLE （空闲）模式。

示例代码请参见例 10-1。

注： 该模块没有 FRZ 模式。

10.3.1.1 SLEEP （休眠）模式下的振荡器关闭

器件在 SLEEP（休眠）模式下禁止时钟源的条件为：振荡器类型、使用时钟源的外设，以及（选定时钟源的）时钟使能位。

- 如果 CPU 时钟源为 POSC，则在 SLEEP （休眠）模式下会被关闭。关于从 SLEEP （休眠）模式唤醒时适用的延时，请参见表 10-2。 USB 模块可改写 POSC 或 FRC 的禁止状态。具体信息请参见 USB 章节。
- 如果 CPU 时钟源为 FRC，则在 SLEEP （休眠）模式下会被关闭。关于从 SLEEP （休眠）模式唤醒时适用的延时，请参见表 10-2。 USB 模块可改写 POSC 或 FRC 的禁止状态。具体信息请参见 USB 章节。
- 如果 CPU 时钟源为 SOSC，则 SOSCEN 位未置 1 时会被关闭。关于从 SLEEP （休眠）模式唤醒时适用的延时，请参见表 10-2。
- 如果 CPU 时钟源为 LPRC，并且在 SLEEP （休眠）模式下工作的外设 （例如 WDT）不使用该时钟源，则会被关闭。关于从 SLEEP （休眠）模式唤醒时适用的延时，请参见表 10-2。

10.3.1.2 从 SLEEP （休眠）模式唤醒时的时钟选择

处理器将恢复代码执行并使用与在进入 SLEEP （休眠）模式时有效的相同时钟源。如果在器件退出 SLEEP （休眠）模式时使用晶振和 / 或 PLL 作为时钟源，则器件会受启动延时影响。

10.3.1.3 从 SLEEP （休眠）模式唤醒的延时

表 10-2 中给出了与从 SLEEP（休眠）模式唤醒时关联的振荡器起振和故障保护时钟监视器延时（如果使能）。

表 10-2： 从休眠模式退出的延时

时钟源	振荡器延时	FSCM 延时
EC 和 EXTRC	—	—
EC + PLL	TLOCK	TFSCM
XT + PLL	TOST + TLOCK	TFSCM
XT、 HS 和 XTL	TOST	TFSCM
LP （休眠期间关闭）	TOST	TFSCM
LP （休眠期间开启）	—	—
FRC 和 LPRC	—	—

注： 关于 TPOR、 TFSCM 和 TLOCK 的规范值，请参见 PIC32MX 器件数据手册的 “电气规范”章节。

10.3.1.4 使用晶振或 PLL 从 SLEEP （休眠）模式唤醒

如果系统时钟源来自晶振和 / 或 PLL，则在系统时钟源可供器件使用之前将应用一段振荡器起振定时器（Oscillator Start-up Timer， OST）和 / 或 PLL 锁定延时。作为该规则的一个特例，如果系统时钟源是 POSC 振荡器且在 SLEEP （休眠）模式下运行，则不应用振荡器延时。

注： 虽然应用了各种延时，但晶振 （和 PLL）还是可能在 TOST 或 TLOCK 延时结束时并未启动和运行。为了正确工作，用户必须设计外部振荡器电路，以便可以在延时周期内产生可靠的振荡。

10.3.1.5 故障保护时钟监视器 （FSCM）延时和 SLEEP （休眠）模式

在器件处于 Sleep（休眠）模式时， FSCM 不工作。如果使能了 FSCM，则它会在器件从 Sleep（休眠）模式唤醒时继续开始工作。此时会应用一段 TFSCM 延时，让振荡器源可以在 FSCM 继续开始监视之前稳定下来。

如果以下条件为真，则会在从 SLEEP （休眠）模式唤醒时应用一段 TFSCM 延时：

- 振荡器在处于 SLEEP （休眠）模式时被关闭。
- 系统时钟来自晶振源和 / 或 PLL。

在大多数情况下，在器件恢复执行指令之前， TFSCM 延时为 OST 超时和 PLL 进入稳定状态提供足够时间。如果使能了 FSCM，它将在 TFSCM 延时结束后开始监视系统时钟源。

10.3.1.6 振荡器慢速起振

在振荡器慢速起振时， OST 和 PLL 锁定时间可能在 FSCM 发生超时之前还未结束。

如果使能了 FSCM，器件将检测到此条件并将其作为一个时钟故障，然后产生时钟故障陷阱。器件将切换到 FRC 振荡器，用户可以在时钟故障中断服务程序中重新使能晶振源。

如果未使能 FSCM，器件在时钟稳定之前不会开始执行代码。从用户角度来看，器件将处于 SLEEP （休眠）模式直到振荡器时钟起振。

10.3.1.7 SLEEP （休眠）模式下振荡器的 USB 外设控制

在 USB 模块工作时，在器件进入休眠模式时该模块会阻止器件禁止其时钟源。虽然振荡器保持工作，但 CPU 和外设将保持暂停。

例 10-1： 将器件置为 SLEEP （休眠）模式，然后通过 WDT 唤醒

```
// Code example to put the Device in sleep and then Wake the device
// with the WDT

OSCCONSET = 0x10;      // set Power-Saving mode to Sleep

WDTCONCLR = 0x0002;    // Disable WDT window mode
WDTCONSET = 0x8000;    // Enable WDT
                       // WDT timeout period is set in the device configuration

while (1)
{
    ... user code ...

    WDTCONSET = 0x01;  // service the WDT
    asm volatile( "wait" );// put device in selected Power-Saving mode

                       // code execution will resume here after wake

    ... user code ...
}

// The following code fragment is at the beginning of the 'C' start-up code

if ( RCON & 0x18 )
{
                       // The WDT caused a wake from Sleep
    asm volatile( "eret" );// return from interrupt
}
```


10.3.2 外设总线分频方法

器件上的大部分外设都使用 PBCLK 作为时钟。外设总线的时钟与 SYSCLK 成比例关系，以最大程度降低外设的动态功耗。 PBCLK 分频比由 PBDIV<1:0>（OSCCON<20:19>）控制，允许的 SYSCLK 与 PBCLK 的比值为 1:1、1:2、1:4 和 1:8。当分频比改变时，所有使用 PBCLK 的外设都会受影响。由于诸如 USB、中断控制器、 DMA、总线矩阵和预取高速缓存之类的外设都是直接从 SYSCLK 获得时钟，因此，它们不受 PBCLK 分频比变化的影响。

改变 PBCLK 分频比会影响：

- CPU 到外设的访问延时。 CPU 必须等待下一个 PBCLK 边沿才能完成读操作。在 1:8 模式下，这会产生 1 至 7 个 SYSCLK 延时。
- 外设的功耗。功耗与外设工作时钟的频率成正比。分频比越大，外设的功耗越低。

要使动态功耗最低，应选择适当的 PB 分频比，使外设在满足系统性能的前提下以最低频率运行。当选择 PBCLK 分频比时，应考虑外设时钟要求 （如波特率精度）。例如，根据 SYSCLK 的值，UART 外设可能在某个 PBCLK 分频比处无法达到所有波特率值。

10.3.2.1 动态外设总线分频方法

PBCLK 可以通过软件进行动态分频，以便在器件处于低活动量模式时节省额外的功耗。对 PBCLK 进行分频时，需要考虑以下问题：

- 通过 PBCLK 提供时钟的所有外设将同时以相同比率进行分频。对于即使在低功耗模式下也需要维持恒定波特率或脉冲周期的外设，需要考虑这一点。
- 如果在 PBCLK 改变时，有任何通信正通过外设总线上的某个外设，则可能会由于发送或接收期间频率改变而导致数据或协议错误。

如果用户希望动态调节 PBCLK 分频比，建议使用以下步骤：

- 禁止波特率会受影响的所有通信外设。在禁止外设之前，应小心确保当前没有任何通信正在进行，因为它可能导致协议错误。
- 根据需要更新外设的波特率发生器 （Baud Rate Generator， BRG）设置，以便在新的 PBCLK 频率下工作。
- 将外设总线比率更改为所需值。
- 使能波特率受影响的所有通信外设。

注：	修改外设波特率的方式是写入关联的外设 SFR。要最大程度减小响应延时，应在 PBCLK 以最高频率运行的模式下修改外设。

例 10-2： 改变 PB 时钟分频比

```
                                    // Code example to change the PBCLK divisor
                                    // This example is for a device running at 40 MHz
                                    // Make sure that there is no UART send/receive in
progress
... user code ...
U1BRG = 0x81;                       // set baud rate for UART1 for 9600
... user code ...
SYSKEY = 0x0;                       // write invalid key to force lock
SYSKEY = 0xAA996655;                // Write Key1 to SYSKEY
SYSKEY = 0x556699AA;                // Write Key2 to SYSKEY
OSCCONCLR = 0x3 << 19;              // set PB divisor to minimum (1:1)
SYSKEY = 0x0;                       // write invalid key to force lock

... user code ...
                                    // Change Peripheral Clock value
U1BRG = 0x0F;                       // set baud rate for UART1 for 9600 based on

                                    // new PB clock frequency
SYSKEY = 0x0;                       // write invalid key to force lock
SYSKEY = 0xAA996655;                // Write Key1 to SYSKEY
SYSKEY = 0x556699AA;                // Write Key2 to SYSKEY
OSCCONSET = 0x3 << 19;              // set PB divisor to maximum (1:8)
SYSKEY = 0x0;                       // write invalid key to force lock

                                    // Reset Peripheral Clock
SYSKEY = 0x0;                       // write invalid key to force lock
SYSKEY = 0xAA996655;                // Write Key1 to SYSKEY
SYSKEY = 0x556699AA;                // Write Key2 to SYSKEY
OSCCONCLR = 0x3 << 19;              // set PB divisor to minimum (1:1)
SYSKEY = 0x0;                       // write invalid key to force lock

U1BRG = 0x81;                       // restore baud rate for UART1 to 9600 based
                                    // on new PB clock frequency
```


10.3.3 IDLE （空闲）模式

在 IDLE（空闲）模式下，CPU 暂停，但系统时钟（SYSCLK）源仍然使能。这允许外设在 CPU 暂停时继续工作。外设可单独配置为在进入 IDLE （空闲）模式时暂停，方法是将其相应的 SIDL 位置 1。由于 CPU 振荡器源保持活动状态，所以退出 Idle （空闲）模式时的延时非常小。

有 4 种空闲工作模式：POSC IDLE、 FRC IDLE、 SOSC IDLE 和 LPRC IDLE。

- POSC IDLE （空闲）模式：SYSCLK 来自 POSC。 CPU 暂停，但 SYSCLK 源继续工作。外设继续工作，但可以选择单独禁止。如果使用了 PLL，则还可以降低倍频比值 PLLMULT<2:0> （OSCCON<18:16>），以降低外设的功耗。
- FRC IDLE （空闲）模式：SYSCLK 来自 FRC。 CPU 暂停。外设继续工作，但可以选择单独禁止。如果使用了 PLL，则还可以降低倍频比值 PLLMULT<2:0> （OSCCON<18:16>），以降低外设的功耗。 FRC 时钟可以通过后分频器通过 RCDIV<2:0> （OSCCON<26:24>）进一步进行分频。
- SOSC IDLE （空闲）模式：SYSCLK 来自 SOSC。 CPU 暂停。外设继续工作，但可以选择单独禁止。
- LPRC IDLE （空闲）模式：SYSCLK 来自 LPRC。 CPU 暂停。外设继续工作，但可以选择单独禁止。

注：	更改 PBCLK 分频比要求重新计算外设时序。例如，假设 UART 配置为 9600 波特，PB 时钟比为 1:1， POSC 为 8 MHz。当使用 1:2 的 PB 时钟分频比时，波特率时钟的输入频率进行二分频；因此，波特率降为原先值的 1/2。由于计算中的数字截断（例如波特率分频比），实际波特率可能会与期望的波特率有微小差别。因此，外设所需的任何时序计算都应使用新的 PB 时钟频率执行，而不是根据 PB 分频比按比例调节先前值。

注：	在切换为已被禁止并使用晶振和 / 或 PLL 的时钟源时，将会应用振荡器起振和 PLL 锁定延时。例如，假设时钟源在进入 Sleep（休眠）模式之前从 POSC 切换为 LPRC，以便节省功耗。退出 Idle （空闲）模式时，将不会应用振荡器起振延时。但是，在切换回 POSC 时，将会应用相应的 PLL 和 / 或振荡器起振 / 锁定延时。

在 SLPEN （OSCCON<4>）位清零，并且执行 WAIT 指令时，器件会进入 IDLE （空闲）模式。

发生以下事件时，处理器将从空闲模式唤醒或退出：

- 已允许中断源的任何中断事件。中断事件的优先级必须大于 CPU 的当前优先级。如果中断事件的优先级小于或等于 CPU 的当前优先级， CPU 将保持暂停，器件将继续处于 IDLE （空闲）模式。
- 任何器件复位源。
- WDT 超时中断。请参见**第 10.4.2 节 “在看门狗超时时从 SLEEP （休眠）或 IDLE （空闲）模式唤醒 （NMI）”**和**第 9 章 “看门狗定时器和上电延时定时器”**（DS61114）。

例 10-3： 将器件置为 IDLE （空闲）模式，并通过 ADC 事件唤醒

```
                            // Code example to put the Device in Idle and then Wake the device
                            // when the ADC completes a conversion
SYSKEY = 0x0;               // write invalid key to force lock
SYSKEY = 0xAA996655;        // Write Key1 to SYSKEY
SYSKEY = 0x556699AA;        // Write Key2 to SYSKEY
OSCCONCLR = 0x10;           // set Power-Saving mode to Idle
SYSKEY = 0x0;               // write invalid key to force lock

asm volatile ( "wait" );    // put device in selected Power-Saving mode
                            // code execution will resume here after wake and the ISR is
                               complete
... user code ...

                            // interrupt handler
void __ISR(27_ADC_VECTOR, ipl7) ADC_HANDLER(void)
{
                            // interrupt handler
unsigned long int result;

result = ADC1BUF0;          // read the result
IFS1CLR = 2;                // Clear ADC conversion interrupt flag
}
```


10.4 中断

有两个中断源可以将器件从节能模式唤醒：节能模式下的外设中断和通过 WDT 产生的不可屏蔽中断 （Non-Maskable Interrupt， NMI）。

10.4.1 在发生外设中断时从 SLEEP （休眠）或 IDLE （空闲）模式唤醒

任何可通过 IECx 寄存器中的相应 IE 控制位单独允许中断且在当前节能模式下工作的中断源都能将处理器从 SLEEP（休眠）或 IDLE（空闲）模式唤醒。在器件唤醒时，根据中断优先级，将产生两个事件之一：

- 如果分配给中断的优先级小于或等于当前 CPU 优先级， CPU 将保持暂停，器件将进入或继续处于 IDLE （空闲）模式。
- 如果分配给中断源的优先级大于当前 CPU 优先级，器件将被唤醒， CPU 将跳转到相应的中断向量处。在 ISR 完成时， CPU 将开始执行 WAIT 之后的下一条指令。

IDLE 状态位（RCON<2>）在从 IDLE（空闲）模式唤醒时被置 1。SLEEP 状态位（RCON<3>）在从 SLEEP （休眠）模式唤醒时被置 1。

注：	中断优先级设置为 0 的外设无法唤醒器件。

注：	在 CPU 继续执行代码之前，将会应用所有适用的振荡器起振延时。

10.4.2 在看门狗超时时从 SLEEP （休眠）或 IDLE （空闲）模式唤醒 （NMI）

当 WDT 在 SLEEP（休眠）或 IDLE（空闲）模式下发生超时时，会产生 NMI。NMI 会导致 CPU 代码执行跳转到器件复位向量处。虽然 CPU 会执行复位向量，但它不是器件复位——外设和大多数 CPU 寄存器都不会改变状态。

注：	在 CPU 继续执行代码之前，将会应用所有适用的振荡器起振延时。

要检测由于 WDT 计时结束导致的节能模式唤醒，则必须测试 WDTO （RCON<4>）、 SLEEP （RCON<3>）和 IDLE （RCON<2>）位。如果 WDTO 位为 1，则说明事件是由于 WDT 超时而发生的。然后，可以通过测试SLEEP和IDLE位，确定WDT事件是在休眠还是空闲模式下发生的。

要在 SLEEP（休眠）模式期间使用 WDT 超时作为唤醒中断，则必须在确定事件为 WDT 唤醒之后，在启动代码中使用从中断返回 （ERET）指令。这会导致代码从将器件置为节能模式的 WAIT 指令之后的指令处继续执行。

注：	如果外设中断和 WDT 事件同时发生，或者在接近的时间内发生，则可能会由于器件被外设中断唤醒而不会发生 NMI。为了避免在这种情况下发生意外的 WDT 复位，在器件唤醒时会自动清零 WDT。

关于 WDT 操作的详细信息，请参见**第 9 章 “看门狗定时器和上电延时定时器”**（DS61114）。

10.4.3 在节能指令执行期间的中断

在 WAIT 指令执行期间产生的任何外设中断都将延迟到进入 SLEEP（休眠）或 IDLE（空闲）模式后才产生。然后，器件将从 SLEEP （休眠）或 IDLE （空闲）模式唤醒。

10.5 与节能模式相关的 I/O 引脚

没有器件引脚与节能模式相关。

10.6 DEBUG （调试）模式下的操作

在调试器工作时，用户无法更改时钟模式。在调试器工作时，仍然会发生由于故障保护时钟监视器 （FSCM）而产生的时钟源更改。

10.7 复位

复位之后的器件行为由所发生复位的类型决定。对于与节能模式相关的行为，复位可以归类为两组：上电复位 （POR）和所有其他复位 （非 POR）。

10.7.1 在 SLEEP （休眠）或 IDLE （空闲）模式期间的非 POR 复位

CPU 将唤醒，代码将在器件复位向量处开始执行。并且会应用所有适用的振荡器延时。空闲状态位 （RCON<2>）或休眠状态位 （RCON<3>）将置 1，指示器件在复位之前处于节能模式。

10.7.2 在 SLEEP （休眠）或 IDLE （空闲）模式期间的 POR 复位

CPU 将唤醒，代码将在器件复位向量处开始执行。并且会应用所有适用的振荡器延时。空闲状态位 （RCON<2>）或休眠状态位 （RCON<3>）将强制清零。 POR 事件之前的节能状态会丢失。

10.8 设计技巧

问 1： ***软件在进入 SLEEP （休眠）或 IDLE （空闲）模式之前应该做什么？***

答 1： 确保将器件唤醒源的 IEC 位置 1。此外，确保特定中断源具有唤醒器件的能力。当器件处于 SLEEP （休眠）模式时，某些中断源不工作。

如果要使器件进入 Idle（空闲）模式，确保正确设置每个器件外设的“空闲模式停止”（stop-in-idle）控制位。这些控制位用于决定外设在 IDLE （空闲）模式下是否继续工作。更多详细信息，请参见本手册各外设的相应章节。

在进入 SLEEP （休眠）模式之前清零 WDT。如果处于窗口模式下，则只能在窗口周期内清零 WDT，以防止器件复位。

问 2： ***如何确定是哪个外设将器件从 SLEEP （休眠）或 IDLE （空闲）模式唤醒？***

答 2： 大部分外设都具有唯一的中断向量。如果需要，您可以通过查询每个已允许中断源的 IF 位来确定唤醒源。

10.9 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 PIC32MX 器件系列而编写的，但其概念是相近的，通过适当修改并受到一定限制即可使用。当前与节能模式相关的应用笔记包括：

标题	应用笔记编号
Low-Power Design using PIC® Microcontrollers	AN606

注： 如需获取更多 PIC32MX 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

10.10 版本历史

版本 A （2007 年 10 月）

这是本文档的初始版本。

版本 B （2007 年 10 月）

更新了文档 （删除了 “机密” 状态）。

版本 C （2008 年 4 月）

将状态修改为 “初稿”； 将 U-0 修改为 r-x。

版本 D （2008 年 7 月）

修改了例 10-1 和 10-3； 修改了表 10-1； 修改了寄存器 10-5 和 10-9； 修改了第 10.3.2 节 （第二段）； 将保留位从 “保持为” 更改为 “写入”； 为 ON 位 （WDTCON 寄存器）增加了注释。

版本 E （2008 年 7 月）

修改了例 10-2 和 10-3。

注：

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

ISBN: 978-1-60932-096-6

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2002 认证。公司在 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 216-447-0464

Fax: 216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200

Fax: 852-2401-3431

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 – 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 西安
Tel: 86-29-8833-7252

Fax: 86-29-8833-7256

中国 – 厦门
Tel: 86-592-238-8138

Fax: 86-592-238-8130

中国 – 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

台湾地区 – 高雄
Tel: 886-7-536-4818

Fax: 886-7-536-4803

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

亚太地区

台湾地区 – 新竹
Tel: 886-3-6578-300
Fax: 886-3-6578-370

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/05/10