目 录

第1章 LPC2200 低层驱动（uCOSII）使用手册

1.1 GPIO

表 1.1 P0 口 GPIO 初始化

所属文件	GPIO.c
函数原型	void P0_GPIOInit(uint32 num,uint8 dir)
功能描述	将 P0 口中，num 为 1 的位初始化为 GPIO。
函数参数	num 需要初始化的管脚 dir 管脚的输入输出方向 1 输出 0 输入
函数返回值	无
说明	如果 dir 错误，则默认为输入方向。
范例	P0_GPIOInit((1<<4)|(1<<16),0); //将 P0.4 和 P0.16 初始化为 GPIO 输入模式

表 1.2 P1 口 GPIO 初始化

所属文件	GPIO.c
函数原型	void P1_GPIOInit(uint32 num,uint8 dir)
功能描述	将 P1 口中，num 为 1 的位初始化为 GPIO。
函数参数	num 需要初始化的管脚 dir 管脚的输入输出方向 1 输出 0 输入
函数返回值	无
说明	如果 dir 错误，则默认为输入模式； 对于 LPC2200，只有 P1.16～P1.31。

表 1.3 P2 口 GPIO 初始化

所属文件	GPIO.c
函数原型	void P2_GPIOInit(uint32 num,uint8 dir)
功能描述	将 P2 口中，num 为 1 的位初始化为 GPIO。
函数参数	num 需要初始化的管脚 dir 管脚的输入输出方向 1 输出 0 输入
函数返回值	无
说明	如果 dir 错误，则默认为输入方向。 如果禁止外部总线， 则 P2.0 ～P2.31 均可作为 GPIO 使用； 如果使用外部 8 位总线， 则 P2.8 ～P2.31 均可作为 GPIO 使用； 如果使用外部 16 位总线，则 P2.16～P2.31 均可作为 GPIO 使用； 如果使用外部 32 位总线，则 P2 口不可作为 GPIO 使用。

表 1.4 P3 口 GPIO 初始化

所属文件	GPIO.c
函数原型	void P3_GPIOInit(uint32 num,uint8 dir)
功能描述	将 P3 口中，num 为 1 的位初始化为 GPIO。
函数参数	num 需要初始化的管脚 dir 管脚的输入输出方向 1 输出 0 输入
函数返回值	无
说明	如果 dir 错误，则默认为输入方向。 如果将 P3.2～P3.23 中的任一引脚设置为 GPIO，那么该函数会将所有的这些引脚全部初始化为 GPIO。

表 1.5 P0 口 GPIO 方向设置

所属文件	GPIO.c
函数原型	void P0_GPIODir(uint32 num,uint8 dir)
功能描述	在 P0 口中，设置 num 为 1 的位输入、输出方式。
函数参数	num 需要设置的管脚 dir 管脚的输入输出方向 1 输出 0 输入
函数返回值	无
说明	如果 dir 错误，则默认为输入方向。

表 1.6 P1 口 GPIO 方向设置

所属文件	GPIO.c
函数原型	void P1_GPIODir(uint32 num,uint8 dir)
功能描述	在 P1 口中，设置 num 为 1 的位输入、输出方式。
函数参数	num 需要设置的管脚 dir 管脚的输入输出方向 1 输出 0 输入
函数返回值	无
说明	如果 dir 错误，则默认为输入方向。

表 1.7 P2 口 GPIO 方向设置

所属文件	GPIO.c
函数原型	void P2_GPIODir(uint32 num,uint8 dir)
功能描述	在 P2 口中，设置 num 为 1 的位输入、输出方式。
函数参数	num 需要设置的管脚 dir 管脚的输入输出方向 1 输出 0 输入
函数返回值	无
说明	如果 dir 错误，则默认为输入方向。

表 1.8 P3 口 GPIO 方向设置

所属文件	GPIO.c
函数原型	void P3_GPIODir(uint32 num,uint8 dir)
功能描述	在 P3 口中，设置 num 为 1 的位输入、输出方式。
函数参数	num 需要设置的管脚 dir 管脚的输入输出方向 1 输出 0 输入
函数返回值	无
说明	如果 dir 错误，则默认为输入方向。

表 1.9 P0 口 GPIO 置位

所属文件	GPIO.h
函数原型	P0_GPIOSet(uint32 num)
功能描述	在 P0 口中，置位 num 为 1 的位所对应的管脚。
函数参数	num 需要设置的管脚
函数返回值	无

表 1.10 P1 口 GPIO 置位

所属文件	GPIO.h
函数原型	void P1_GPIOSet(uint32 num)
功能描述	在 P1 口中，置位 num 为 1 的位所对应的管脚。
函数参数	num 需要设置的管脚
函数返回值	无

表 1.11 P2 口 GPIO 置位

所属文件	GPIO.h
函数原型	void P2_GPIOSet(uint32 num)
功能描述	在 P2 口中，置位 num 为 1 的位所对应的管脚。
函数参数	num 需要设置的管脚
函数返回值	无

表 1.12 P3 口 GPIO 置位

所属文件	GPIO.h
函数原型	void P3_GPIOSet(uint32 num)
功能描述	在 P3 口中，置位 num 为 1 的位所对应的管脚。
函数参数	num 需要设置的管脚
函数返回值	无

表 1.13 P0 口 GPIO 清零

所属文件	GPIO.h
函数原型	void P0_GPIOClr(uint32 num)
功能描述	在 P0 口中，清零 num 为 1 的位所对应的管脚。
函数参数	num 需要设置的管脚
函数返回值	无

表 1.14 P1 口 GPIO 清零

所属文件	GPIO.h
函数原型	void P1_GPIOClr(uint32 num)
功能描述	在 P1 口中，清零 num 为 1 的位所对应的管脚。
函数参数	num 需要设置的管脚
函数返回值	无

表 1.15 P2 口 GPIO 清零

所属文件	GPIO.h
函数原型	void P2_GPIOClr(uint32 num)
功能描述	在 P2 口中，清零 num 为 1 的位所对应的管脚。
函数参数	num 需要设置的管脚
函数返回值	无

表 1.16 P3 口 GPIO 清零

所属文件	GPIO.h
函数原型	void P3_GPIOClr(uint32 num)
功能描述	在 P3 口中，清零 num 为 1 的位所对应的管脚。
函数参数	num 需要设置的管脚
函数返回值	无

表 1.17 读取 P0 口数据

所属文件	GPIO.h
函数原型	Read_P0( )
功能描述	读取 P0 口的数值。
函数参数	无
函数返回值	P0 口的数据

表 1.18 读取 P1 口数据

所属文件	GPIO.h
函数原型	Read_P1( )
功能描述	读取 P1 口的数值。
函数参数	无
函数返回值	P1 口的数据

表 1.19 读取 P2 口数据

所属文件	GPIO.h
函数原型	Read_P2( )
功能描述	读取 P2 口的数值。
函数参数	无
函数返回值	P2 口的数据

表 1.20 读取 P3 口数据

所属文件	GPIO.h
函数原型	Read_P3( )
功能描述	读取 P3 口的数值。
函数参数	无
函数返回值	P3 口的数据

表 1.21 向 P0 口写入数据

所属文件	GPIO.h
函数原型	Write_P0(value)
功能描述	向 P0 口写入数据。
函数参数	value 引脚输出的值
函数返回值	无

表 1.22 向 P1 口写入数据

所属文件	GPIO.h
函数原型	Write_P1(value)
功能描述	向 P1 口写入数据。
函数参数	value 引脚输出的值
函数返回值	无

表 1.23 向 P2 口写入数据

所属文件	GPIO.h
函数原型	Write_P2(value)
功能描述	向 P2 口写入数据。
函数参数	value 引脚输出的值
函数返回值	无

表 1.24 向 P3 口写入数据

所属文件	GPIO.h
函数原型	Write_P3(value)
功能描述	向 P3 口写入数据。
函数参数	value 引脚输出的值
函数返回值	无

1.2 UART0 接口

注意事项：

1、 需要在 config.h 文件中修改数据队列的配置信息和 UART0 发送数据队列的空间大小。
例：

```
/* 数据队列的配置 */
#include "queue.h"
#define     QUEUE_DATA_TYPE             uint8
#define     EN_QUEUE_WRITE              1       /* 禁止（0）或使能（1）FIFO 发送数据 */
#define     EN_QUEUE_WRITE_FRONT        0       /* 禁止（0）或使能（1）LIFO 发送数据 */
#define     EN_QUEUE_NDATA              1       /* 禁止（1）或使能（1）取得队列数据数目 */
#define     EN_QUEUE_SIZE               1       /* 禁止（1）或使能（1）取得队列数据总容量 */
#define     EN_QUEUE_FLUSH              0       /* 禁止（1）或使能（1）清空队列 */
/* UART0 的配置 */
#include "uart0.h"
/* 给 UART0 发送数据队列分配的空间大小(以字节为单位) */
#define     UART0_SEND_QUEUE_LENGTH         60
```

2、 需要在 IRQ.S 文件中增加 UART0 中断句柄，并在 target.c 文件的 VICInit 函数中添加 UART0 中断初始化。
例：添加 UART0 中断句柄

```
;/* 通用串口 0 中断 */
UART0_Handler  HANDLER      UART0_Exception         ; 添加 UART0 中断句柄
```

例：添加 UART0 中断初始化

```
extern void UART0_Handler(void);
……
VICVectAddr1 = (uint32)UART0_Handler;
VICVectCntl1 = (0x20 | 0x06);
VICIntEnable = 1 << 6;
```

表 1.25 UART0 初始化

所属文件	UART0.c
函数原型	uint8 UART0Init(uint32 bps)
功能描述	对 UART0 进行初始化。
函数参数	bps 通信波特率
函数返回值	TRUE 成功 FALSE 失败
说明	UART0 默认配置 8 位数据位、1 位停止位、无奇偶校验位

表 1.26 UART0Putch 函数

所属文件	UART0.c
函数原型	void UART0Putch(uint8 Data)
功能描述	发送一个字节数据。
函数参数	Data 发送的数据数据
函数返回值	无
说明	必须先初始化 UART0。

表 1.27 UART0Write 函数

所属文件	UART0.c
函数原型	void UART0Write(uint8 *Data, uint16 NByte)
功能描述	发送多个字节数据。
函数参数	Data 发送数据缓冲区首地址 NByte 发送字节数
函数返回值	无

表 1.28 UART0Getch 函数

所属文件	UART0.c
函数原型	uint8 UART0Getch(void)
功能描述	从 UART0 接收一个字节数据。
函数参数	无
函数返回值	接收到的数据

1.3 I^2C接口

注意事项：

1、 需要在IRQ.S文件中增加I^2C中断句柄，并在target.c文件的VICInit函数中添加I^2C中断初始化。

例：添加I^2C中断句柄

```
;/*I2C0 中断*/
 I2C_Handler            HANDLER      I2c_Exception              ;添加I2C中断句柄
```

例：添加I^2C中断初始化

```
 extern void I2C_Handler(void);
……
 VICVectAddr1 = (uint32)I2C_Handler;
 VICVectCntl1 = (0x20 | 0x09);
 VICIntEnable = 1 << 9;
```

表 1.29 I2cInit 函数

所属文件	I2C.c
函数原型	uint8 I2cInit(uint32 FI2c)
功能描述	初始化 I2c（主模式）
函数参数	FI2c I^2C总线频率
函数返回值	TRUE 成功 FALSE 失败

表 1.30 I2cWrite 函数

所属文件	I2C.c
函数原型	uint16 I2cWrite(uint8 Addr, uint8 *Data, int16 NByte)
功能描述	向 I2C 从器件写数据
函数参数	Addr 从机地址 Data 指向将要写的数据的指针 NByte 写的数据数目
函数返回值	发送的数据字节数
说明	必须先初始化I^2C接口。

表 1.31 I2cRead 函数

所属文件	I2C.c
函数原型	int16 I2cRead(uint8 Addr, uint8 *Ret, uint8 *Eaddr, int16 EaddrNByte, int16 ReadNbyte)
功能描述	从 I2c 从器件读数据
函数参数	Addr 从机地址 Ret 指向返回数据存储位置的指针 Eaddr 扩展地址存储位置 EaddrNByte 扩展地址字节数，0 为无 ReadNbyte 将要读取的字节数目
函数返回值	已读取的字节数
说明	必须先初始化I^2C接口。

1.4 SPI 接口

注意事项：

1、 需要在 config.h 文件中修改 SPI 接口的模式信息。

例：CPHA = 0、CPOL = 1、LSBF = 0。

```
#define   SPI_MOD  (SPI_CPHA_ONE | SPI_CPOL_LOW | SPI_LSBF_BIT7)
```

2、 需要在 IRQ.S 文件中增加 SPI 中断句柄，并在 target.c 文件的 VICInit 函数中添加 SPI 中断初始化。

例：添加 SPI 中断句柄

```
;/* SPI 中断*/
```

```
SPI_Handler        HANDLER     SPI_Exception          ;添加 SPI 中断句柄
```

例：添加 SPI 中断初始化

```
extern void SPI_Handler(void);
……
VICVectAddr1 = (uint32) SPI_Handler;
VICVectCntl1 = (0x20 | 10);
VICIntEnable = 1 <<10;
```

2、 操作 SPI 接口的方法：

a) SPIStart(); //初始化时调用，只能一次
……
b) 允许从机;
c) 多次 SPIRW(yy);
d) 禁止从机;
e) SPIEnd();

表 1.32 SPIInit 函数

所属文件	SPI.c
函数原型	uint8 SPIInit(uint8 Fdiv)
功能描述	初始化 SPI 总线为主模式。
函数参数	Fdiv 用于设定总线频率（总线频率=Fpclk/Fdiv）
函数返回值	TRUE 初始化成功 FALSE 初始化失败
说明	有关 SPI 中断向量的设置已在 TargetInit 函数中进行，占用通道 4。

表 1.33 GetSPIFlag 函数

所属文件	SPI.c
函数原型	uint8 GetSPIFlag(void)
功能描述	获取 SPI 状态。
函数参数	无
函数返回值	0 空闲 1 忙

表 1.34 SPIStart 函数

所属文件	SPI.c
函数原型	uint8 SPIStart(void)
功能描述	开始访问 SPI。
函数参数	无
函数返回值	TRUE 成功 FALSE 失败

表 1.35 SPIRW 函数

所属文件	SPI.c
函数原型	uint8 SPIRW(uint8 *Rt, uint8 Data)
功能描述	将数据通过 SPI 总线发送出去并从 SPI 总线接收一个数据。
函数参数	Rt 函数通过这个指针返回接收到的数据 Data 发送的数据
函数返回值	TRUE 成功 FALSE 失败

表 1.36 SPIEnd 函数

所属文件	SPI.c
函数原型	uint8 SPIEnd(void)
功能描述	访问 SPI 结束。
函数参数	无
函数返回值	TRUE 成功 FALSE 失败

1.5 定时/计数器

表 1.37 定时器 0 捕获模式初始化

所属文件	Time.c
函数原型	uint8 T0CAP_Init(uint8 CAP_MODE,uint32 PR_data,uint8 CAPn,uint8 Int_En)
功能描述	定时器 0 捕获模式初始化。
函数参数	CAP_MODE 捕获方式，按位操作方式。 bit0 1--CAP 上跳沿捕获 bit1 1--CAP 下降沿捕获 PR_data 预分频寄存器的值 CAPn 捕获通道选择，0～3 Int_En 中断使能 0 发生捕获事件时，不产生中断 1 发生捕获事件时，产生中断
函数返回值	0 初始化失败 1 初始化成功
说明	函数中缺少引脚初始化。

表 1.38 定时器 1 捕获模式初始化

所属文件	Time.c
函数原型	uint8 T1CAP_Init(uint8 CAP_MODE,uint32 PR_data,uint8 CAPn,uint8 Int_En)
功能描述	定时器 1 捕获模式初始化。
函数参数	CAP_MODE 捕获方式，按位操作方式。 bit0 1--CAP 上跳沿捕获 bit1 1--CAP 下降沿捕获 PR_data 预分频寄存器的值 CAPn 捕获通道选择，0～3 Int_En 中断使能 0 发生捕获事件时，不产生中断 1 发生捕获事件时，产生中断
函数返回值	0 初始化失败 1 初始化成功
说明	函数中缺少引脚初始化。

表 1.39 定时器 0 匹配模式初始化

所属文件	Time.c
函数原型	uint8 T0MAT_Init(uint32 time , uint32 PR_data , uint8 T_MODE, uint8 EXT_MODE , uint8 MATn , uint8 Int_En)
功能描述	定时器 0 匹配模式初始化。
函数参数	time 匹配时间，该值会直接写入到匹配寄存器中 PR_data 预分频寄存器的值 T_MODE 匹配控制模式 0：匹配时，定时器复位 1：匹配时，定时器停止 EXT_MODE 匹配时，外部输出控制 0：不执行任何动作 1：外部匹配输出 0 2：外部匹配输出 1 3：外部匹配输出翻转 MATn 匹配通道选择，0～3 Int_En 中断使能 0 发生匹配事件时，不产生中断 1 发生匹配事件时，产生中断
函数返回值	0 初始化失败 1 初始化成功
说明	函数中缺少引脚初始化。

表 1.40 定时器 1 匹配模式初始化

所属文件	Time.c
函数原型	uint8 T0MAT_Init(uint32 time , uint32 PR_data , uint8 T_MODE, uint8 EXT_MODE , uint8 MATn,uint8 Int_En)
功能描述	定时器 1 匹配模式初始化。
函数参数	time 匹配时间，该值会直接写入到匹配寄存器中 PR_data 预分频寄存器的值 T_MODE 匹配控制模式 0 匹配时，定时器复位 1 匹配时，定时器停止 EXT_MODE 匹配时，外部输出控制 0：不执行任何动作 1：外部匹配输出 0 2：外部匹配输出 1 3：外部匹配输出翻转 MATn 匹配通道选择，0～3 Int_En 中断使能 0 发生匹配事件时，不产生中断 1 发生匹配事件时，产生中断
函数返回值	0 初始化失败 1 初始化成功
说明	函数中缺少引脚初始化。

表 1.41 读取定时器 0 的当前计数值

所属文件	Time.h
函数原型	Read_T0( )
功能描述	读取定时器 0 的计数值。
函数参数	无
函数返回值	定时器 0 的当前计数值。

表 1.42 读取定时器 1 的当前计数值

所属文件	Time.h
函数原型	Read_T1( )
功能描述	读取定时器 1 的计数值。
函数参数	无
函数返回值	定时器 1 的当前计数值。

1.6 PWM

表 1.43 SingleEdgePWM_Init 函数

所属文件	PWM .C
函数原型	uint8 SingleEdgePWM_Init(uint32 F_PWM,uint32 PR_Data,uint32 PWMMRn, uint8 PWMn,uint8 INTEn)
功能描述	单边沿 PWM 初始化。
函数参数	F_PWM PWM 频率控制寄存器，该值直接写入到 PWMMR0 寄存器中 PR_Data 预分频器的值 PWMMRn PWM 匹配寄存器，控制 PWM 边沿的位置 PWMn PWM 通道选择，有效值为 1～6 INTEn 中断使能 >0 TC 值与 PWMMR0 匹配时产生中断 0 TC 值与 PWMMR0 匹配时不产生中断
函数返回值	1 操作成功 0 操作失败
说明	使用 PWMMR0 来控制 PWM 的频率。匹配时，复位 PWMTC； 包含有引脚初始化操作。

表 1.44 DoubleEdgePWM_Init 函数

所属文件	PWM .C
函数原型	uint8 DoubleEdgePWM_Init(uint32 F_PWM,uint32 PR_Data,uint32 SetPWMMRn, uint32 ClrPWMMRn,uint8 PWMn,uint8 INTEn);
功能描述	双边沿 PWM 初始化。
函数参数	F_PWM PWM 频率控制寄存器，该值直接写入到 PWMMR0 寄存器中 PR_Data 预分频器的值 SetPWMMRn PWM 置位匹配寄存器，控制 PWM 上升沿的位置 ClrPWMMRn PWM 复位匹配寄存器，控制 PWM 下降沿的位置 PWMn PWM 通道选择：2、3、4、5、6 为有效通道 INTEn 中断使能 >0 TC 值与 PWMMR0 匹配时产生中断 0 TC 值与 PWMMR0 匹配时不产生中断
函数返回值	1 操作成功 0 操作失败
说明	使用 PWMMR0 来控制 PWM 的频率。匹配时，复位 PWMTC； 包含有引脚初始化操作。

表 1.45 Read_PWM 函数

所属文件	PWM .h
函数原型	Read_PWM( )
功能描述	读取 PWM 定时器的值。
函数参数	无
函数返回值	PWM 定时器计数器的当前值

表 1.46 TimePWM_Init 函数

所属文件	PWM .C
函数原型	void TimePWM_Init(uint32 time,uint32 PR_Data,uint8 TMODE)
功能描述	PWM 初始化为 32 位定时器。
函数参数	PWM_time 定时时间,该值会直接写入到 PWMMR0 寄存器中 PR_Data 预分频器的值 TMODE 工作模式，按位操作 bit0--中断，为 1 时，产生中断 bit1--复位，为 1 时，PWMTC 复位 bit2--停止，为 1 时，PWMTC 停止
函数返回值	无
说明	PWM 使用 PWMMR0 完成定时器的初始化工作。

1.7 A/D 转换器

表 1.47 AD 初始化

所属文件	ADC .C
函数原型	uint8 ADC_Init(uint8 ADn,uint32 Fadc)
功能描述	AD 初始化操作。
函数参数	ADn AD 通道，0～7 Fadc ADC 的转换时钟，最大的转换速率为 4.5M，即，4500000
函数返回值	1 操作成功 0 操作失败
说明	函数中已经包含了引脚设置； 转换时钟最大 4.5MHz； 使用软件方式启动。

表 1.48 读取 AD 转换值

所属文件	ADC .C
函数原型	uint32 Read_ADC(uint8 ADn)
功能描述	读取 AD 转换值。
函数参数	ADn AD 通道，0～7
函数返回值	读取出来的 AD 结果
说明	软件采用查询方式等待转换结束，同时已经将结果经过了处理，转换结果为：0～3ff。 如果参数有误，转换结果为 0。

1.8 看门狗

表 1.49 初始化看门狗

所属文件	WDT . c
函数原型	void WDT_Init(uint32 time)
功能描述	初始化看门狗。
函数参数	time 看门狗定时时间，该值直接写入到 WDTC 中。溢出时间＝Tpclk × 4 × time
函数返回值	无
说明	默认看门狗溢出复位。

表 1.50 喂狗函数

所属文件	WDT. c
函数原型	void FeedDog（void）
功能描述	执行喂狗序列。
函数参数	无
函数返回值	无

表 1.51 IsWDOverTimeFlg 函数

所属文件	WDT. h
函数原型	IsWDOverTimeFlg( )
功能描述	判断是否产生看门狗超时标志。
函数参数	无
函数返回值	>0：产生超时标志 0：未产生超时标志

表 1.52 IsWDIntFlg 函数

所属文件	WDT. h
函数原型	IsWDIntFlg( )
功能描述	判断是否产生看门狗中断标志。
函数参数	无
函数返回值	>0：产生中断标志 0：未产生中断标志

表 1.53 CleanWDOverTimeFlg 函数

所属文件	WDT. h
函数原型	CleanWDOverTimeFlg( )
功能描述	清零看门狗超时标志。
函数参数	无
函数返回值	无

表 1.54 GetWDTimeVal 函数

所属文件	WDT. h
函数原型	GetWDTimeVal( )
功能描述	读取看门狗定时器的当前值。
函数参数	无
函数返回值	看门狗定时器的当前值

1.9 外部中断输入

表 1.55 SetExtInt 函数

所属文件	ExtInterrupt . C
函数原型	uint8 SetExtInt(uint8 no, uint8 mode, uint8 bWakeUp)
功能描述	设置外部中断属性。
函数参数	no 外部中断，取值 0～3 mode 中断触发类型，取值如下： 0 低电平触发 1 高电平触发 2 下降沿触发 3 上升沿触发 bWakeUp 是否使能中断唤醒 CPU 0 不唤醒 CPU >0 使能唤醒 CPU
函数返回值	1 操作成功 0 操作失败
说明	需要在外部设置引脚模式。

表 1.56 CleanExtIntFlg 函数

所属文件	ExtInterrupt . C
函数原型	CleanExtIntFlg(no)
功能描述	清零指定的外部中断标志。
函数参数	no 外部中断，取值 0～3
函数返回值	无

表 1.57 IsExtInt 函数

所属文件	ExtInterrupt . C
函数原型	IsExtInt(no)
功能描述	判断是否产生指定的外部中断。
函数参数	no 外部中断，取值 0～3
函数返回值	>0 产生了外部中断 0 没产生外部中断

1.10 功率控制

表 1.58 Set_PCONP 函数

所属文件	Power . h
函数原型	Set_PCONP(value)
功能描述	设置外设功率控制寄存器。
函数参数	value PCONP 寄存器的值
函数返回值	无

表 1.59 Power_Down 函数

所属文件	Power . h
函数原型	Power_Down ( )
功能描述	进入掉电模式。
函数参数	无
函数返回值	无

表 1.60 IDLE 函数

所属文件	Power . h
函数原型	IDLE ( )
功能描述	进入空闲模式。
函数参数	无
函数返回值	无