
CW89F(E)系列iMCU™ 虚拟模块库VML™

用户手册

二〇〇八年十月

版本 1.00



目 录

1. VML™ 概述.....	1
2. VML™ 参数传递.....	1
3. VML™ 调用条件.....	2
4. VML™ 调用方法.....	2
5. VML™ 库配置.....	3
6. VML™ 库函数.....	4
6.1 System.....	4
6.1.1 系统初始化.....	4
6.1.2 中断优先级设置.....	5
6.1.3 中断控制.....	5
6.1.4 外部中断触发模式.....	6
6.1.5 定时器 0 中断服务子程序.....	6
6.1.6 定时器 1 中断服务子程序.....	6
6.2 Timer/Counter.....	6
6.2.1 模型框图.....	6
6.2.2 Timer 设置.....	7
6.2.3 Timer 装载.....	8
6.2.4 Timer 读取.....	8
6.2.5 Timer 开关.....	8
6.3 Watchdog.....	9
6.3.1 模型框图.....	9
6.3.2 Watchdog 初值.....	9
6.3.3 Watchdog 开关.....	9
6.3.4 Watchdog 刷新.....	9
6.4 PWM.....	10
6.4.1 模型框图.....	10
6.4.2 PWM 输出.....	10
6.4.3 PWM 停止.....	10
6.5 RTC.....	11
6.5.1 模型框图.....	11
6.5.2 RTC 设置与启动.....	11
6.5.3 RTC 读取.....	11
6.5.4 RTC 停止.....	12
6.5.5 RTC 报警设置与启动.....	12
6.5.6 RTC 报警查询.....	12
6.5.7 RTC 报警取消.....	12
6.6 UART.....	13
6.6.1 模型框图.....	13



CW89F(E)系列 iMCU™ 虚拟模块库 VML™ 用户手册

6.6.2	UART 初始化	13
6.6.3	UART 查询发送数据	14
6.6.4	UART 查询接收数据	14
6.7	I ² C Master.....	14
6.7.1	模型框图	14
6.7.2	I ² C 主机发送数据	15
6.7.3	I ² C 主机接收数据	15
6.8	A/D.....	15
6.8.1	模型框图	15
6.8.2	A/D 输入转换	16
6.9	Temp.....	16
6.9.1	模型框图	16
6.9.2	Temp 输入转换.....	16
6.10	EEPROM.....	17
6.10.1	EEPROM 写数据	17
6.10.2	EEPROM 读数据	17
6.11	ConfigFile	17
6.11.1	存储配置文件.....	17
6.11.2	读取配置文件.....	18
6.12	IAP.....	18
6.12.1	IAP 擦除扇区.....	18
6.12.2	IAP 写字节数据	18
6.12.3	IAP 读字节数据	18
6.13	Version.....	19



1. VML™概述

VML™是指虚拟模块库，即利用软件配以少数分立器件完成一些专用硬件模块或芯片的功能，可使用户节省底层软硬件的调试时间，减小应用开发的工作量，降低外围硬件的成本，同时又能够灵活动态地配置，实现多种用途。

VML™以内嵌固件提供给用户，用户程序可通过 API 函数实现调用。

2. VML™参数传递

用户使用 Keil C51 编译器时，通常采用固定寄存器方式传递函数参数。

注意传递参数占用的是工作寄存器区 0 (PSW.0 = 0, PSW.1 = 0)，即 R0~R7 对应的寄存器地址为：00H~07H。

具体如下表所示：

函数输入参数的寄存器

输入参数	第 1 个参数	第 2 个参数	第 3 个参数
char	R7	R5	R3
int	R6 = 高字节 R7 = 低字节	R4 = 高字节 R5 = 低字节	R2 = 高字节 R3 = 低字节
long	R4 = 高字节 R5 = 第 2 字节 R6 = 第 3 字节 R7 = 低字节	R4 = 高字节 R5 = 第 2 字节 R6 = 第 3 字节 R7 = 低字节	无效
一般指针	R3 = 0 (data 类型) R2 = 高字节 R1 = 低字节	R3 = 0 (data 类型) R2 = 高字节 R1 = 低字节	R3 = 0 (data 类型) R2 = 高字节 R1 = 低字节

函数返回值的寄存器

返回值	寄存器
bit	C (PSW.7)
char	R7
int	R6 = 高字节, R7 = 低字节
long	R4 = 高字节, R5 = 第 2 字节, R6 = 第 3 字节, R7 = 低字节

例如，若在 C51 代码中声明如下函数：

```
“unsigned int VMLFuntion1(unsigned int para1, unsigned char para2, unsigned char* para3)”
```

参数将通过以下寄存器传递：

R6 = 参数 para1 的高字节

R7 = 参数 para1 的低字节

R5 = 参数 para2

R3 = 0

R2 = 指针 para3 的高字节

R1 = 指针 para3 的低字节



返回值将通过以下寄存器传递：

R6 = 高字节

R7 = 低字节

3. VML™调用条件

在用户工程中需要包含以下文件，与用户程序一起编译。

VML 库函数头文件：VMLPro_xxxx.h

VML 函数库：CW89FxxxxVML.LIB 或 CW89FExxxxVML.LIB

（其中 xxxx 是与具体的器件名相关的编号）

4. VML™调用方法

用户可通过 C51 或汇编语言调用 VML™ 函数。

例如：调用函数 “void UARTInit (unsigned char mode, unsigned int baud, unsigned char parity)” 和 “void WatchdogRefresh(void)”。

当使用 C51 语言：

```
unsigned char bMode = 1;    // 工作方式 1, 8 位 UART
unsigned int iBaud = 9600;  // 波特率为 9600bps
unsigned char bParity = 0;  // 无校验码
UARTInit (bMode, iBaud, bParity); // UART 串口初始化
WatchdogRefresh();        // 看门狗刷新
```

当使用汇编语言：

```
MOV R7, #01H    ; 工作方式 1, 8 位 UART
MOV R5, #80H    ; 波特率为 9600bps (0x2580)
MOV R4, #25H
MOV R3, #00H    ; 无校验码
LCALL _UARTInit ; UART 串口初始化
LCALL WatchdogRefresh ; 看门狗刷新
```

注：如果函数需传递参数，则被调用的函数名前加下划线，例如 “_UARTInit”；否则不需要下划线，例如 “WatchdogRefresh”。



CW89F(E)系列 iMCU™ 虚拟模块库 VML™ 用户手册

5. VML™库配置

不同型号的 iMCU™ 配置的库函数会有所区别，具体如下表所示：

序号	库函数	资源	CW89FE5x	CW89FE305x	CW89FE605x
1	System		√	√	√
2	Timer/Counter		T0: Timer/Counter	T0: Timer	T0: Timer/Counter
			T1: Timer/Counter	T1: Timer	T1: Timer/Counter
3	Watchdog		√	√	√
4	PWM	T1	P15	×	P15
5	RTC	T0	√	×	×
6	UART		P30(R), P31(T)	P30(R), P31(T)	P30(R), P31(T)
7	I ² C Master		P12(C), P13(D)	×	P12(C), P13(D)
8	A/D		P04	×	P04
9	A/D2		P02	×	P02
10	PWM2	T0	P14	×	P14
11	Temp		P17, P37, P03	×	P17, P37, P03
12	EEPROM		√	√	√
13	ConfigFile		×	×	√
14	IAP		√	√	√
15	Version		√	√	√

注1：“√”表示具备的功能，“×”表示不具备的功能，其它表示具备且需要的引脚。

注2：因为都用定时器T0资源，所以RTC和PWM2不能同时使用。

注3：EEPROM的容量分别为3K字节（x51系列）、2K字节（x52系列）、1K字节（x54系列）。



6. VML™ 库函数

6.1 System

6.1.1 系统初始化

函数原型	void Sysnit (unsigned char sys_clock, unsigned char devider)
函数功能	设置主频参数
函数参数	<p>sys_clock—主频选择</p> <p>1: 3.57MHz 2: 4MHz 3: 6MHz 4: 11.0592MHz 5: 12MHz 6: 14.28MHz 7: 16MHz 8: 22.1184MHz 9: 24MHz 10: 保留 11: 3.6864MHz 12: 7.3278MHz 13: 14.7456MHz</p> <p>devider—分频系数</p> <p>0: 不分频 1: 1/4 分频 2: 1/16 分频 3: 1/256 分频 4: 1/1024 分频</p>
返回值	无
资源	
备注	设置系统的初始状态和变量



CW89F(E)系列 iMCU™ 虚拟模块库 VML™ 用户手册

6.1.2 中断优先级设置

函数原型	void IntPriority (unsigned char int_id, unsigned char int_priority)
函数功能	设置各中断优先级
函数参数	<p>int_id—中断识别号</p> <ul style="list-style-type: none">1: EXT_INT0 外部中断 02: Watchdog 看门狗中断3: T0 T0 定时器/计数器 0 中断4: EXT_INT1 外部中断 15: T1 T1 定时器/计数器 1 中断6: UART UART 串行口中断 <p>int_priority—中断优先级</p> <ul style="list-style-type: none">0: 优先级 0 (最低)1: 优先级 12: 优先级 23: 优先级 3 (最高)
返回值	无
资源	外部中断和计数的引脚定义参见具体型号的芯片手册
备注	

6.1.3 中断控制

函数原型	void IntEnable (unsigned char int_id, unsigned char int_enable)
函数功能	控制各中断使能
函数参数	<p>int_id—中断识别号</p> <ul style="list-style-type: none">0: CPU 中断允许总标志 EA1: EXT_INT0 外部中断 0 允许标志 EX02: Watchdog 看门狗中断允许标志 EWD3: 定时器/计数器 0 中断允许标志 ET04: EXT_INT1 外部中断 0 允许标志 EX15: T1 定时器/计数器 1 中断允许标志 ET16: 串行口中断允许标志 ES <p>int_enable—中断使能</p> <ul style="list-style-type: none">0: 禁止中断1: 允许中断
返回值	
资源	外部中断和计数的引脚定义参见具体型号的芯片手册
备注	注意使用中断要避免和 VML 库函数功能发生冲突



6.1.4 外部中断触发模式

函数原型	void IntExtMode (unsigned char int_id, unsigned char int_trigger)
函数功能	外部中断触发模式选择
函数参数	int_id—中断识别号 0: EXT_INT0 外部中断 0 1: EXT_INT1 外部中断 1 int_trigger—触发模式 0: 电平触发 1: 下降沿触发
返回值	无
资源	外部中断的引脚定义参见具体型号的芯片手册
备注	

6.1.5 定时器 0 中断服务子程序

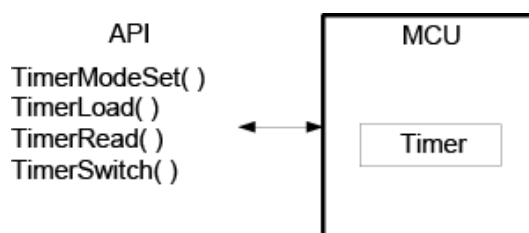
函数原型	void timer0_subpro (void)
函数功能	T0 中断调用，实现与 T0 有关的库函数功能
函数参数	无
返回值	无
资源	
备注	内部包含了有关 RTC 或 PWM2 的处理 需要用户在用户程序的 T0 中断函数中调用

6.1.6 定时器 1 中断服务子程序

函数原型	void timer1_subpro (void)
函数功能	T1 中断调用，实现与 T1 有关的库函数功能
函数参数	无
返回值	无
资源	
备注	内部包含了有关 PWM 的处理 需要用户在用户程序的 T1 中断函数中调用

6.2 Timer/Counter

6.2.1 模型框图



CW89F(E)系列 iMCU™ 虚拟模块库 VML™ 用户手册

6.2.2 Timer 设置

函数原型	void TimerModeSet (unsigned char timer_no, unsigned char mode, unsigned char gate)
函数功能	设置 Timer 工作方式
函数参数	<p>timer_no—定时器选择</p> <p>0: T0 1: T1 2: WDT (看门狗)</p> <p>mode—模式选择</p> <p><T0、T1></p> <p>0x00: 定时—13 位 0x01: 定时—16 位 0x02: 定时—8 位自动重载 0x03: 定时—2 个 8 位定时器 (仅对 T0 有效) 0x04: 计数—13 位 0x05: 计数—16 位 0x06: 计数—8 位自动重载 0x07: 计数—2 个 8 位计数器 (仅对 T0 有效)</p> <p><WDT></p> <p>0x00: 8 位自动重载</p> <p>gate—闸门选择 (仅对 T0、T1 有效)</p> <p>0: 无 1: 有, INT0 或 INT1 引脚</p>
返回值	无
资源	<p>所选定时器</p> <p>外部计数的引脚定义参见具体型号的芯片手册</p>
备注	<p>T0, T1 需避免冲突。</p> <p>WDT 直接使用外部晶振 fosc 计时, 需避免与看门狗复位重用。</p> <p>该设置函数一般结合定时器中断使用。</p> <p>设置 C/T = 0, 为定时器方式。</p> <p>设置 C/T = 1, 为计数器方式。</p> <p>若使用外部计数输入, 当有低跳变时, 则计数器加 1。</p>



6.2.3 Timer 装载

函数原型	void TimerLoad (unsigned char timer_no, unsigned char th_value, unsigned char tl_value)
函数功能	向 Timer 定时器装入初值
函数参数	timer_no—定时器选择 0: T0 1: T1 2: WDT (看门狗) th_value—装入初值的高8位 tl_value—装入初值的低8位 (WDT 取低8位)
返回值	无
资源	所选定时器
备注	对于 T0/T1, 高8位为 TH, 低8位为 TL 对于 WDT, 仅低8位有效

6.2.4 Timer 读取

函数原型	unsigned int TimerRead (unsigned char timer_no)
函数功能	读取 Timer 定时器当前值
函数参数	timer_no—定时器选择 0: T0 1: T1 2: WDT (看门狗)
返回值	定时器的当前值
资源	所选定时器
备注	对于 T0/T1, 高8位为 TH, 低8位为 TL 对于 WDT, 仅低8位有效

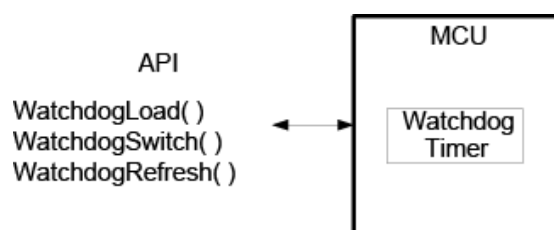
6.2.5 Timer 开关

函数原型	void TimerSwitch (unsigned char timer_no, unsigned char switch)
函数功能	启动或关闭 Timer 定时器的运行
函数参数	timer_no—定时器选择 0: T0 1: T1 2: WDT (看门狗) switch—开关选择 0: 停止 1: 启动
返回值	无
资源	所选定时器
备注	需先执行 Timer 设定工作方式



6.3 Watchdog

6.3.1 模型框图



6.3.2 Watchdog 初值

函数原型	void WatchdogLoad (unsigned char overflow)
函数功能	开启看门狗复位功能，并设置溢出时间参数
函数参数	overflow—溢出时间参数 (范围: 0~255, 8 位) 溢出时间周期为: $\text{Period} = (255 - \text{overflow}) * 344064 * 1/\text{fosc}$ 秒。 例如: fosc = 12MHz, overflow = 254, Period = 28.7ms。
返回值	无
资源	看门狗定时器
备注	如使用 WDT 作为看门狗复位，应禁止 WDT 作为定时器中断。

6.3.3 Watchdog 开关

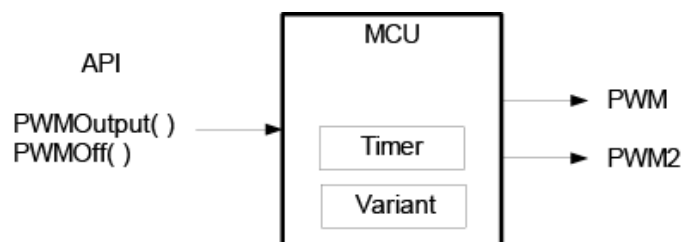
函数原型	void WatchdogSwitch(unsigned char switch)
函数功能	启动或关闭看门狗的运行
函数参数	switch—开关参数 0: 关闭 1: 启动
返回值	无
资源	看门狗定时器
备注	

6.3.4 Watchdog 刷新

函数原型	void WatchdogRefresh(void)
函数功能	刷新看门狗 (喂狗)
函数参数	无
返回值	无
资源	看门狗定时器
备注	为了防止复位，需周期性地在溢出时间之前刷新。

6.4 PWM

6.4.1 模型框图



6.4.2 PWM 输出

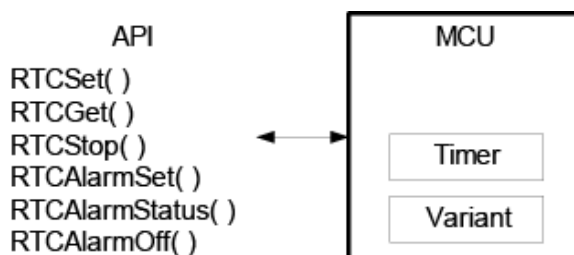
函数原型	void PWMOutput (unsigned char channel, unsigned int freq, unsigned char duty)
函数功能	根据频率和占空比输出 PWM 波
函数参数	channel—通道编号 0: 通道 1 1: 通道 2 freq—频率(Hz) duty—高电平占空比(%) (范围: 0~100)
返回值	无
资源	定时器 T1 (通道 1), 定时器 T0 (通道 2) 端口 PWM (通道 1), 端口 PWM2 (通道 2)
备注	应注意避免定时器 T1/T0 资源冲突。 采用定时器中断方式。 频率越高, 占空比误差越大。 当使用 PWM 时, 应在 T1 中断服务程序中被调用函数“time1_subpro()”, 并使能该中断。 当使用 PWM2 时, 应在 T0 中断服务程序中调用函数“time0_subpro()”, 并使能该中断。

6.4.3 PWM 停止

函数原型	void PWMOFF (unsigned char channel)
函数功能	停止 PWM 输出
函数参数	channel—通道编号 0: 通道 1 1: 通道 2
返回值	无
资源	
备注	释放定时器资源

6.5 RTC

6.5.1 模型框图



6.5.2 RTC 设置与启动

函数原型	void RTCSet (unsigned char *rtc_buf)
函数功能	设置并启动 RTC
函数参数	rtc_buf—需写入 RTC 的数据缓冲区指针（十进制数） rtc_buf[0]: 秒 sec rtc_buf[1]: 分 min rtc_buf[2]: 时 hr rtc_buf[3]: 日 date rtc_buf[4]: 月 mon rtc_buf[5]: 年 year（21 世纪）
返回值	无
资源	定时器 T0
备注	应注意避免定时器 T0 资源冲突。 采用定时器中断方式。 每秒更新一次，掉电则停止计时。 当使用 RTC 时，应在 T0 中断服务程序中调用函数“time0_subpro()”，并使能该中断。

6.5.3 RTC 读取

函数原型	void RTCGet (unsigned char *rtc_buf)
函数功能	读取 RTC
函数参数	rtc_buf—需读出 RTC 的数据缓冲区指针（十进制数） rtc_buf[0]: 秒 sec rtc_buf[1]: 分 min rtc_buf[2]: 时 hr rtc_buf[3]: 日 date rtc_buf[4]: 月 mon rtc_buf[5]: 年 year（21 世纪）
返回值	无
资源	定时器 T0
备注	

6.5.4 RTC 停止

函数原型	void RTCStop (void)
函数功能	停止 RTC 计时
函数参数	无
返回值	无
资源	
备注	释放定时器资源

6.5.5 RTC 报警设置与启动

函数原型	void RTCAlarmSet (unsigned char *rtc_buf)
函数功能	设置并启动 RTC
函数参数	rtc_buf—需写入 RTC 的数据缓冲区指针（十进制数） rtc_buf[0]: 秒 sec rtc_buf[1]: 分 min rtc_buf[2]: 时 hr rtc_buf[3]: 日 date rtc_buf[4]: 月 mon rtc_buf[5]: 年 year（21 世纪）
返回值	无
资源	
备注	打开报警软开关

6.5.6 RTC 报警查询

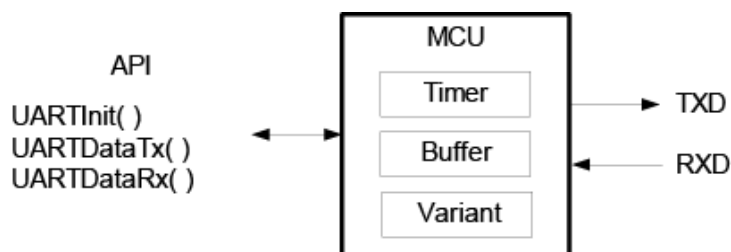
函数原型	unsigned char RTCAlarmStatus (void)
函数功能	查询 RTC 报警状态，查询后自动清除标志
函数参数	无
返回值	0: 无报警 1: 有报警
资源	
备注	

6.5.7 RTC 报警取消

函数原型	void RTCAlarmOff (void)
函数功能	取消 RTC 报警
函数参数	无
返回值	无
资源	
备注	关闭报警软开关

6.6 UART

6.6.1 模型框图



6.6.2 UART 初始化

函数原型	void UARTInit (unsigned char mode, unsigned int baud, unsigned char parity)
函数功能	设置 UART 初始参数
函数参数	<p>mode—工作方式</p> <p>0: 方式 0, 移位寄存器, 波特率 fosc/12 (12 时钟)</p> <p>1: 方式 1, 8 位 UART, 波特率可变</p> <p>2: 方式 2, 9 位 UART, 波特率 fosc/32 (12 时钟)</p> <p>3: 方式 3, 9 位 UART, 波特率可变</p> <p>baud—波特率 bps (当方式 2 或 3 才有效)</p> <p>300</p> <p>1200</p> <p>2400</p> <p>4800</p> <p>9600</p> <p>19200</p> <p>38400</p> <p>57600</p> <p>parity—第 9 位校验码 (当方式 2 或 3 才有效)</p> <p>0: 无校验—None</p> <p>1: 奇校验—Odd</p> <p>2: 偶校验—Even</p> <p>3: 1—Mark</p> <p>4: 0—Space</p>
返回值	无
资源	
备注	必须初始化后才能调用其它 UART 函数

CW89F(E)系列 iMCU™ 虚拟模块库 VML™ 用户手册

6.6.3 UART 查询发送数据

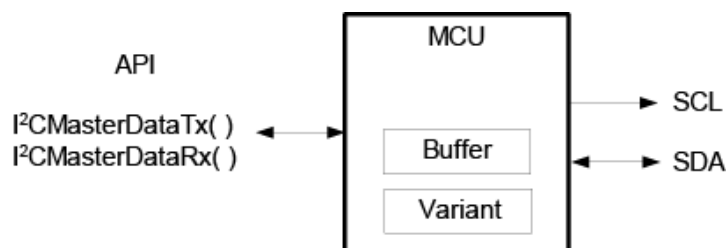
函数原型	void UARTDataTx (unsigned char n, unsigned char *buffer)
函数功能	发送缓冲区中的一组数据
函数参数	n—需发送的字节数, 1~255 buffer—需发送的数据缓冲区指针
返回值	无
资源	端口 TXD
备注	

6.6.4 UART 查询接收数据

函数原型	unsigned char UARTDataRx (unsigned char n, unsigned char overtime, unsigned char *buffer)
函数功能	接收一组数据到缓冲区
函数参数	n—需接收的字节数, 1~255 overtime—超时时间, 单位: 秒, 范围: 0~255s (为 0 时无超时) buffer—需接收的数据缓冲区指针
返回值	实际接收数据的个数
资源	端口 RXD
备注	

6.7 I²C Master

6.7.1 模型框图



CW89F(E)系列 iMCU™ 虚拟模块库 VML™ 用户手册

6.7.2 I²C 主机发送数据

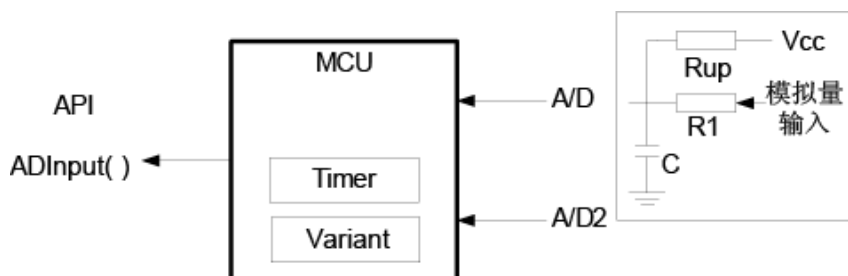
函数原型	unsigned char I2CMasterDataTx (unsigned char n, unsigned char *buffer)
函数功能	作为 I ² C 主机发送缓冲区中的一组数据
函数参数	n—需发送的数据字节数, 0~250 buffer—数据缓冲区指针 buffer[0]: 从机地址, bit7~bit1 有效, bit0 = X buffer[1]: 地址字节数: 1 为单字节, 2 为双字节 buffer[2]: 子地址高字节 (对于单字节无效) buffer[3]: 子地址低字节 buffer[4]~[n+3]: 发送的数据内容
返回值	0: 发送成功 1: 发送失败
资源	端口 SCL, SDA
备注	

6.7.3 I²C 主机接收数据

函数原型	unsigned char I2CMasterDataRx (unsigned char n, unsigned char *buffer)
函数功能	作为 I ² C 主机接收一组数据到缓冲区
函数参数	n—需接收的字节数, 1~250 buffer—数据缓冲区指针 buffer[0]: 从机地址, bit7~bit1 有效, bit0 = X buffer[1]: 地址字节数: 1 为单字节, 2 为双字节 buffer[2]: 子地址高字节 (对于单字节无效) buffer[3]: 子地址低字节 buffer[4]~[n+3]: 接收的数据内容
返回值	0: 接收成功 1: 接收失败
资源	端口 SCL, SDA
备注	buffer 中的地址为输入参数, 数据为输出参数。

6.8 A/D

6.8.1 模型框图

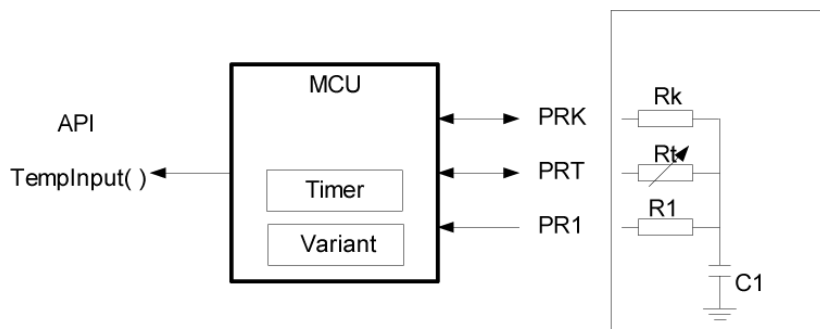


6.8.2 A/D 输入转换

函数原型	unsigned int ADInput (unsigned char channel)
函数功能	启动 A/D 输入转换，并送出采样结果
函数参数	channel—通道编号 0: 通道 1 1: 通道 2
返回值	转换结果，16 位
资源	端口 A/D
备注	电压测量范围：0~5V [器件参数] a) $R_{up} = 100K\Omega$ ，高精度电阻 b) $R_1 = 100K\Omega$ ，高精度电阻 c) $C = 0.1\mu F$ ，聚苯乙烯电容 由于输出的采样值与模拟电压成非线性递减的关系，因此建议用户根据采样结果，使用分段线性的算法来计算实际值。 模拟信号的输出阻抗越小，效果越好。

6.9 Temp

6.9.1 模型框图



6.9.2 Temp 输入转换

函数原型	unsigned char TempInput (void)
函数功能	启动温度输入转换，并送出热转换的结果
函数参数	无
返回值	转换结果，8 位（热敏电阻的阻值参数） （范围：50~150，单位：k Ω ）
资源	端口 PRK, PRT, PR1
备注	函数返回热敏电阻值，用户根据热敏电阻的特性曲线计算实际温度值。 [器件参数] a) 采用合适的电容电阻值 b) R_k ，精密电阻，100k Ω c) R_t ，热敏电阻，100k Ω d) R_1 ，100 Ω e) C_1 ，0.1 μF

6.10 EEPROM

6.10.1 EEPROM 写数据

函数原型	void EEPROMDataWr (unsigned int address, unsigned char n, unsigned char *buffer)
函数功能	将缓冲区中的数据写入内部 EEPROM
函数参数	address—存储地址，起始地址为 0 n—字节数，1~128 buffer—需写入的数据缓冲区指针
返回值	无
资源	EEPROM 的容量参见具体型号的芯片手册
备注	采用内部 Flash 存储器模拟实现。 因考虑到写入速度和擦写寿命限制（1 万次），尽量避免频繁改写数据。 连续地址的数据尽可能一次性写入。

6.10.2 EEPROM 读数据

函数原型	void EEPROMDataRd (unsigned int address, unsigned char n, unsigned char *buffer)
函数功能	从内部 EEPROM 读数据到缓冲区中
函数参数	address—存储地址，起始地址为 0 n—字节数，1~128 buffer—需读出的数据缓冲区指针
返回值	无
资源	EEPROM 的容量参见具体型号的芯片手册
备注	

6.11 ConfigFile

6.11.1 存储配置文件

函数原型	unsigned char ConfigFileWr (unsigned char n, unsigned char *buffer)
函数功能	将缓冲区中的数据写入用户配置文件存储区
函数参数	n—字节数，1~64 buffer—需写入的数据缓冲区指针
返回值	0: 成功 1: 失败
资源	
备注	在 VML 的数据区开辟 256 字节空间，用于存放用户配置文件。 总共分成 4 个区，每个区 64 字节，其中 1 个数据区，3 个备份区。文件大小不超过 64 字节，采用覆盖写的方式。 写完后读出比对，若一致则返回成功。



6.11.2 读取配置文件

函数原型	unsigned char ConfigFileRd (unsigned char n, unsigned char *buffer)
函数功能	读出用户配置文件到缓冲区中
函数参数	n—字节数, 1~64 buffer—需读出的数据缓冲区指针
返回值	0: 成功 1: 失败
备注	

6.12 IAP

6.12.1 IAP 擦除扇区

函数原型	void IAPERaseSector (unsigned char bvWAddH, unsigned char bvWAddL)
函数功能	在用户 Flash 程序空间, 擦除一个扇区 (128 字节)
函数参数	bvWAddH—擦除扇区首地址的高字节 bvWAddL—擦除扇区首地址的低字节, 0x00 或 0x80
返回值	无
资源	
备注	[bvWAddH, bvWAddL]为该扇区的起始地址值

6.12.2 IAP 写字节数据

函数原型	void IAPWriteByte (unsigned char bvWAddH, unsigned char bvWAddL, unsigned char bvWData)
函数功能	向用户 Flash 程序空间, 写入一个字节数据
函数参数	bvWAddH—存储地址的高字节 bvWAddL—存储地址的低字节 bvWData—写入的数据
返回值	无
备注	

6.12.3 IAP 读字节数据

函数原型	unsigned char IAPReadByte(unsigned char bvRAddH, unsigned char bvRAddL)
函数功能	从用户 Flash 程序空间, 读取一个字节数据
函数参数	bvRAddH—存储地址的高字节 bvRAddL—存储地址的低字节
返回值	从指定存储地址读出的数据
备注	



CW89F(E)系列 iMCU™ 虚拟模块库 VML™ 用户手册

6.13 Version

函数原型	void GetVMLDate(unsigned char * bVMLDate)
函数功能	获取 VML™ 版本日期
函数参数	bVMLDate—VML 版本信息，定义缓存大小为 17 个字节（ASCII 码） bVMLDate [0]~[7]: 8 个字节的日期信息，格式为“月/日/年” bVMLDate [8]: 0x00 bVMLDate [9]~[16]: 8 个字节的时时间信息，格式为“时:分:秒”
返回值	无
备注	

