

文档编号: AN_095

上海东软载波微电子有限公司

应用笔记

ES8P508x

修订历史

版本	修订日期	修改概要
V1.0	2017-12-27	初版
V1.1	2018-05-08	1. 增加“对未使用 GPIO 的处理”说明 2. 增加“低功耗系统程序设计注意事项与推荐结构”

地 址：中国上海市龙漕路 299 号天华信息科技园 2A 楼 5 层

邮 编：200235

E-mail: support@essemi.com

电 话：+86-21-60910333

传 真：+86-21-60914991

网 址：http://www.essemi.com/

版权所有©

上海东软载波微电子有限公司

本资料内容为上海东软载波微电子有限公司在现有数据资料基础上慎重且力求准确无误编制而成，本资料中所记载的实例以正确的使用方法和标准操作为前提，使用方在应用该等实例时请充分考虑外部诸条件，上海东软载波微电子有限公司不担保或确认该等实例在使用方的适用性、适当性或完整性，上海东软载波微电子有限公司亦不对使用方因使用本资料所有内容而可能或已经带来的风险或后果承担任何法律责任。基于使本资料的内容更加完善等原因，上海东软载波微电子有限公司保留未经预告的修改权。使用方如需获得最新的产品信息，请随时用上述联系方式与上海东软载波微电子有限公司联系

目 录

内容目录

第 1 章	ES8P508x 应用注意	4
1.1	配置字 DEBUG 功能	4
1.2	开发环境	4
1.3	寄存器写保护	4
1.3.1	SCU 写保护	4
1.3.2	IAP 写保护	4
1.3.3	RTC 写保护	4
1.3.4	IWDT 写保护	4
1.3.5	WWDT 写保护	5
1.3.6	CRC 写保护	5
1.4	位操作	5
1.4.1	位带扩展原理	5
1.4.2	位带使用方法	5
1.5	写 1 清零寄存器	6
1.6	对未使用 GPIO 的处理	6
1.7	串行总线操作	6
1.8	I2C 高速从机编程操作	7
1.9	IAP 操作程序	7
1.10	多路 PWM 同步输出	7
1.11	深度睡眠模式下 WDT 操作	7
1.12	GPIO 端口输出电平位操作	7
1.13	GPIO 端口灌电流的限制要求	8
1.14	未使用的 GPIO 端口处理	8
1.15	ADC 模块应用注意事项	8
1.16	PLL 系统时钟选择注意事项	8
1.17	低功耗系统程序设计注意事项与推荐结构	8

第1章 ES8P508x应用注意

1.1 配置字DEBUG功能

调试时，CFG_DEBUG 位要使能。

加密时，CFG_DEBUG 位必须要禁止，否则加密无效（编程工具将提示错误）。

1.2 开发环境

推荐用户使用 Keil4 进行固件开发。由于 Keil5 限制用户对 Cortex-M0 进行 bitband 操作，并且我司提供的 Keil 插件不支持 Keil5 的部分版本，故不推荐用户使用 Keil5。

1.3 寄存器写保护

为避免程序的异常导致运行错误，芯片写保护寄存器用于阻止对被保护的寄存器误操作。

系统控制单元，RTC，WDT 等模块支持寄存器写保护，对被保护的寄存器进行写之前需要解除写保护状态（允许写），否则无法对写保护寄存器写入。操作完成后，再使能写保护（禁止写）。

1.3.1 SCU写保护

系统控制寄存器 SCU 的访问操作会影响整个芯片的运行状态，芯片提供系统设置保护寄存器 SCU_PROT。

对 SCU_PROT 寄存器以字方式写入 0x55AA6996 会解除写保护，对该寄存器写入其他任何值都会使能写保护。

SCU_PROT 保护的寄存器为 SCU_NMICON，SCU_PWRC，SCU_FAULTFLAG，SCU_WAKEUPTIME，SCU_MRSTN_SOFT，SCU_DBGHALT，SCU_FLASHWAIT，SCU_SOFTCFG，SCU_LVDCON，SCU_CCM，SCU_PLCLKCON，SCU_SCLKEN0，SCU_SCLKEN1，SCU_PCLKEN0，SCU_PCLKEN1，SCU_PRSTEN0，SCU_PRSTEN1，SCU_TIMEREN，SCU_TIMERDIS，SCU_TBLREMAPEN，SCU_TBLOFF。

库函数提供 SCU_RegUnLock 宏解除写保护，SCU_RegLock 宏使能写保护。

1.3.2 IAP写保护

对 IAP_UL 寄存器以字方式写入 0x4941_5055 会解除写保护，对该寄存器写入其他任何值都会使能写保护。

IAP_UL 保护的寄存器为 IAP_CON，IAP_ADDR，IAP_DATA，IAP_TRIG。

库函数提供 FlashIAP_RegUnLock 宏解除写保护，FlashIAP_RegLock 宏使能写保护。

1.3.3 RTC写保护

对 RTC_WP 寄存器以字方式写入 0x55AAAA55 会解除写保护，对该寄存器写入其他任何值都会使能写保护。

可通过读 RTC_WP 寄存器确认 RTC 模块是否处于写保护状态，读出值为 0x55AAAA55，表示当前处于解除写保护状态；读出值为 0x00000000 表示 RTC 模块处于使能写保护状态。

RTC_WP 保护的 RTC 寄存器为 RTC_CON，RTC_CAL，RTC_WA，RTC_DA，RTC_HMS，RTC_YMDW，RTC_IE，RTC_IF。

1.3.4 IWDG写保护

对 IWDG_LOCK 寄存器以字方式写入 0x1ACCE551 会解除写保护，对该寄存器写入其他任何值都会使能写保护。

IWDG_LOCK 保护的寄存器为 IWDG_LOAD, IWDG_CON, IWDG_INTCLR
库函数提供 IWDG_RegUnLock 宏解除写保护, IWDG_RegLock 宏使能写保护。

1.3.5 WWDG写保护

对 WWDG_LOCK 寄存器以字方式写入 0x1ACCE551 会解除写保护, 对该寄存器写入其他任何值都会使能写保护。

WWDG_LOCK 保护的寄存器为 WWDG_LOAD, WWDG_CON, WWDG_INTCLR
库函数提供 WWDG_RegUnLock 宏解除写保护, WWDG_RegLock 宏使能写保护。

1.3.6 CRC 写保护

对 CRC_UL 寄存器以字方式写入 0x4352_4355 会解除写保护, 对该寄存器写入其他任何值都会使能写保护。

CRC_UL 保护的寄存器为 CRC_CON, CRC_TRIG, CRC_ADDR, CRC_SIZE, CRC_DI, CRC_DO, CRC_STA

1.4 位操作

Cortex-M0 本身不支持位带操作(bitband), 本芯片为了方便用户操作, 为用户扩展了位带功能。

1.4.1 位带扩展原理

SRAM 位带扩展功能, 对 SRAM 的每个 bit, 都赋予了一个扩展地址, 通过该扩展地址, 可直接访问其对应的 SRAM 数据位, 从而极大的方便了对 SRAM 单元的位读写操作。对于 SRAM 的某个 bit, 记它所在字节地址为 A, 位序号为 N ($0 \leq N \leq 7$), SRAM 位带扩展映射区的基地址为 0x2200_0000, 则该 bit 的位带扩展地址为:

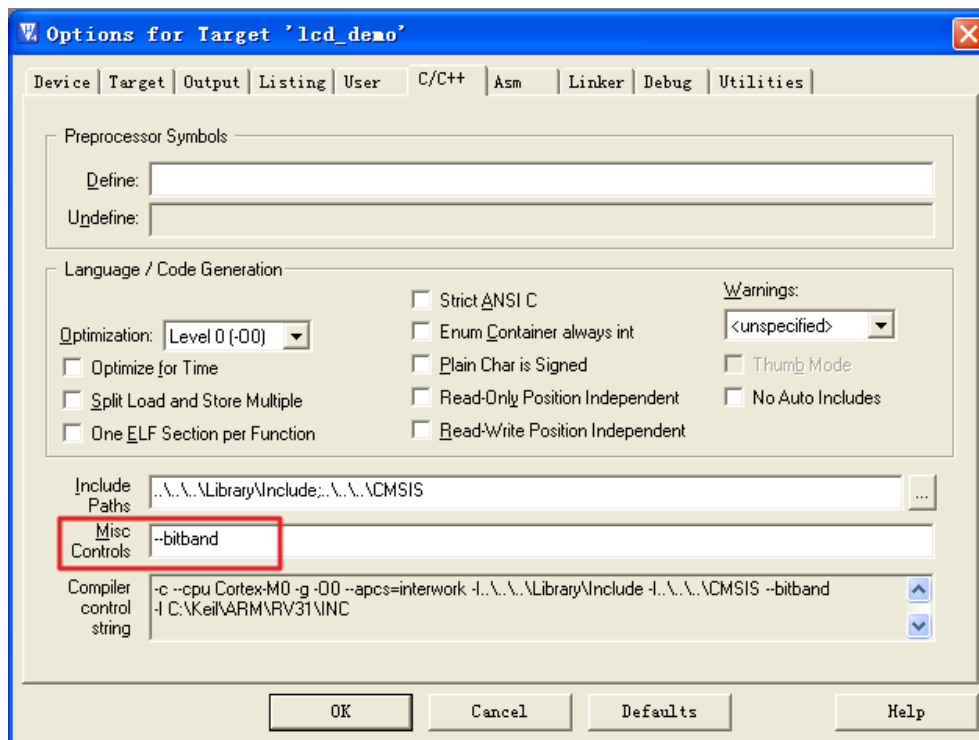
$$\text{AliasAddress_A_N} = 0x2200_0000 + (A - 0x2000_0000) \times 32 + N \times 4$$

外设寄存器位带扩展功能, 对外设寄存器的每个 bit, 都赋予了一个扩展地址, 通过该扩展地址, 可直接访问其对应的寄存器位, 从而极大的方便了对外设寄存器的位读写操作。对于外设寄存器的某个 bit, 记它所在字节地址为 A, 位序号为 N ($0 \leq N \leq 7$), 外设寄存器位带扩展映射区的基地址为 0x4200_0000, 则该 bit 的位带扩展地址为:

$$\text{AliasAddress_A_N} = 0x4200_0000 + (A - 0x4000_0000) \times 32 + N \times 4$$

1.4.2 位带使用方法

1. 直接对位带扩展地址进行读写操作: 按照上面的方法计算得到所要操作 bit 的位带扩展地址, 然后直接对其地址进行读写操作。
2. 将外设寄存器或者变量使用 C 语言定义成位域, 如果位域变量为 1bit 宽度, 并且对 C 编译器进行了如下设置, 那么编译出来的代码将自动对其进行位带操作。如果用户没有按照下面的方式对 C 编译器进行 “--bitband” 设置, 那么所有的位域写操作都将使用对原地址进行 “读-修改-写” 的方式实现。



1.5 写 1 清零寄存器

有很多中断标志寄存器都是用“写 1 清零”的方式来操作。对于“写 1 清零”的寄存器，不可使用“读-修改-写”的方式来进行“写 1 清零”，否则会引起标志位误清，进而产生漏中断的后果。

例如对 T16N0 的中断标志寄存器 MAT0IF 进行“写 1 清零”，应该按照如下操作：

```
T16N0->IF.Word = (uint32_t)0x01;
```

或者进行位域操作（必须对 C 编译器进行“--bitband”设置）：

```
T16N0->IF.MAT0IF = 1;
```

如果进行位域操作(T16N0->IF.MAT0IF = 1)而没有对 C 编译器进行“--bitband”设置，或者用按位“与、或、非”的方式(T16N0->IF.Word |= (uint32_t)0x01)，其最终都是按照“读-修改-写”的方式来操作的，这时如果 MAT1IF 也为 1，那么 MAT1IF 就会被误清，从而造成漏中断的后果。

1.6 对未使用GPIO的处理

因 GPIO 默认状态是输入悬空，在实际应用中，建议程序对不使用的 GPIO 设置为输出 0，并关闭内部上下拉电阻，可提高 EMC 抗干扰性能。

1.7 串行总线操作

串行总线 I2C 发送数据时，需等待 I2CxTBEF0~3 标志置 1，即发送缓冲器全空后才能发送停止位，否则会导致最后装载的数据不能正常发出。

SPI 总线发送数据时，需等待 SPIxIDLE 标志置 1，即发送缓冲器全空后才能关闭发送使能。

UART 总线发送数据时，需等待 UxTXIDLE 标志置 1，即发送缓冲器全空后才能关闭发送使能。

UART 的 RBIF 和 TBIF 两个标志位为只读，无法直接清除。其中 RBIF 在读取接收缓存后可自动清除；TBIF 在发送缓冲中有数据时可自动清除。因此在使能 RBIE 中断时，在中断服务函数中读取接收缓存 RBR 后可自动清除 RBIF。在使能 TBIE 中断时，在中断服务函数中向发送缓冲器写入

下一个想要发送的数据，可自动清除 TBIF；若要停止发送数据，则需在中断服务函数中关闭 TBIE 中断，以避免芯片不停的进入发送缓冲空中断。

1.8 I2C高速从机编程操作

I2C 支持 7 位从机地址匹配，由 I2C 主机控制发送或接收数据。当主机向从机发送数据时，从机通常判断 RBIF 标志，如果接收缓冲器不空，即接收到主机数据，则读接收缓冲器的数据；当主机读取从机数据时，从机可以判断 TIDLEIF 标志，如果发送空闲，则依次写入需要发送的数据。

当 I2C 做从机需要高速传输时，用户需要注意以下几点：

1. 使能时钟线自动下拉功能。在通常情况下，从动器处于释放时钟线的状态，时钟线 SCL 完全由主控制器控制。但当从动器出现异常情况，短时间内无法继续进行数据传输时，从动器可以在时钟线 SCL 为低电平时输出 0（不可以高电平时输出 0，否则会破坏数据传输过程），强行使 SCL 保持低电平，使主控制器进入通讯等待状态，直到从动器释放时钟线；
2. 为实现 I2C 时钟线的下拉等待请求功能，还需 I2C_CON 寄存器中配置 SCL_OD，将通讯端口 SCL 选择为开漏输出模式，通过上拉电阻提供高电平（复用的 IO 口也需要设置为开漏输出，上拉模式），使从动器可对时钟线下拉控制，使主控制器等待；
3. 为避免从机自动下拉时间太长，超出主机的最大等待时间，程序需尽快将数据写入 TWB 寄存器；
4. 为了使代码的效率更高，可以使用直接操作寄存器的方法来控制 I2C 的传输。

1.9 IAP操作程序

ES8P508x 芯片内置 IAP 自编程固化模块，由硬件电路实现。IAP 操作既可以放在 SRAM 执行，也可以调用自编程固化模块，推荐用户调用自编程固化模块，以减少 SRAM 中的 IAP 操作代码量。

1.10 多路PWM同步输出

若要实现多路 PWM 的同步输出，可以通过 SCU_TIMEREN 和 SCU_TIMERDIS 控制寄存器，选择性同时启动或关停多个 T16N/T32N 定时器来实现。

1.11 深度睡眠模式下WDT操作

在深度睡眠状态下使用 WDT 功能，需要开启 WDT 中断，如果不开启 WDT 中断，需要使用以下方法：

1. 进入 deepsleep 模式之前先手动关停时钟滤波器，即：先旁路 CLKFLT，即设置 CLKFLT_BY=0x55，然后再设置 CLKFLT_EN=0 来关闭时钟滤波器。
2. 深度睡眠模式唤醒后，先手动使能系统时钟滤波器，即：先设置 SCU_WAKEUP_TIME 寄存器的 CLKFLT_EN 位为 1 以使能系统时钟滤波器，然后设置 SCU_SCLKEN0 寄存器的 CLKFLT_BY≠0x55 以选用经滤波的系统时钟。

1.12 GPIO端口输出电平位操作

GPIO 端口输出电平位操作寄存器 GPIO_PADATABSR，GPIO_PADATABCR，GPIO_PADATABRR，GPIO_PADIRBSR，GPIO_PADIRBCR，GPIO_PADIRBRR，GPIO_PBDATABSR，GPIO_PBDATABCR，GPIO_PBDATABRR，GPIO_PBDIRBSR，GPIO_PBDIRBCR，GPIO_PBDIRBRR 不能进行与或操作，只能按 word 写入。

GPIO 端口输出电平操作时建议用位操作寄存器而不是端口寄存器，以避免读-修改-写情况的

发生。

1.13 GPIO端口灌电流的限制要求

对于 PA6~PA13 端口，当使用其中某几个端口作为输出驱动时，总的灌电流不要超过 40mA，否则可能会导致这些 IO 端口中的其它端口输出的低电平被抬高到 0.4V 以上。

1.14 未使用的GPIO端口处理

系统中未使用的 GPIO 管脚建议设置为输出固定电平并悬空，若设置为输入，须加上拉或下拉电阻接到电源或地。建议在低功耗配置中，需要将未使用的 GPIO 端口固定设置为模拟端口，即 DIR 与 INEB 寄存器同时为 1。

1.15 ADC模块应用注意事项

屏蔽 ADC 的 VDD 检测，即 ADC 通道选择寄存器 (ADC_CHS) 的 bit8 (VDD5_FLAG_EN) 设为 0。

1.16 PLL系统时钟选择注意事项

1. CLKFLT 为系统时钟滤波器。当系统时钟为 PLL 输出 48MHz 时，需旁路 CLKFLT，否则可能会造成系统时钟有时失效；当系统时钟为其它时钟源时，则不建议旁路 CLKFLT，可进一步提升系统工作稳定性。
2. SCU_SCLKEN0.PLL_MUX 寄存器，置 1 后，软件复位无法还原为 0，需要硬件复位。

1.17 低功耗系统程序设计注意事项与推荐结构

在进行低功耗系统程序设计时需要注意以下几点：

1. 建议悬空的 GPIO 固定输出低电平，有上下拉的 GPIO 输出相应固定电平。输入功能的 IO 不可悬空。
2. 在进入休眠函数和退出休眠函数时使用芯片软复位进行切换；(NVIC_SystemReset());
3. 休眠函数初始化需要关闭总中断(__disable_irq());，禁止在休眠函数中响应任何中断服务程序，并使能相应唤醒源 IRQ(NVIC_EnableIRQ(XXX_IRQn));
4. 可靠的系统不应该在系统运行的过程中关闭看门狗，休眠时也不例外。建议休眠时将 WDT 做为唤醒源，唤醒后立即清狗，再让芯片进入深睡眠状态，这样的处理方式对系统平均功耗的增加可忽略不计。

