



Example Description

SPC1168 Peripheral Examples

Revision 1 – April 2019

1 示例工程说明

SPC1168 示例工程提供了覆盖每个外设主要特性的代码。每个示例代码可以直接用 KEIL MDK 进行编译，然后将编译后的代码下载到目标芯片中即可运行。SPC1168 示例工程包含的示例代码如下：

模块名	示例	
	工程名	描述
ADC	ADC_Calibration	校准 ADC 的增益及偏置数值
	ADC_Continuous_Mode	设置 ADC 工作在连续采样模式（使用某一 SOC 结束信号触发另外一个 SOC 开始工作）
	ADC_Conversion_Priority	设置 ADC SOC 优先级，使高优先级 SOC 得以首先进行采样并转换
	ADC_Differential_Trim_Result	设置 ADC 为双端模式，并获取 trim 之后的结果
	ADC_Sequential_Mode	设置 ADC 为工作在顺序采样模式（SPC1168 有 3 个 ADC 采样器，分别为 A/B/C，默认情况下采样器 A 首先工作，然后是 B，最后是 C）
	ADC_SHA_Open_Detect	检测 A 采样器通路上被采样的 ADC GPIO 是否存在虚焊等导致开路的情况
	ADC_SHA_PPU	开启 ADC 的 PPU 功能，并使用 PPU 检测 ADC 采样结果是否超过设定的阈值范围
	ADC_SHA_Short_Detect	检测 A 采样器通路上被采样的 ADC GPIO 是否存在短路情况
	ADC_Simultaneous	设置 ADC 模块的 A/B 两个采样器同时进行采样
	ADC_Single_End_Result	设置 ADC 工作在单端模式（有一端被接到 GND）
	ADC_With_PGA	ADC 工作在双端模式，对经过 PGA 放大后的两个 ADC GPIO 进行采样
AES	AES	设置 AES 工作在 CBC 模式，并对随机数据进行加密，随后解密，对比原始数据和解密后的数据，验证是否相同
Comparator	Comparator	设置 Comparator 分别以 DAC2 和 DAC3 作为高低阈值的参考电压，以检测 Comparator 的输入是否超过此阈值
CRC	CRC_Calculate	设置 CRC 工作在 16CCITT 模式，并计算一段字符串，将结果与网上计算过的结果对比，以检查 CRC 是否工作正常
DAC	DAC	设置 DAC 输出某一数值的电压，并利用 ADC 进行采样，判断 ADC 的结果是否在合理范围区间，以此判断 DAC 是否工作正常
ECAP	ECAP_Continue_Absolute	设置 ECAP 工作在连续采样的 Absolute 模式，并连续对 PWM 波形进行事件采样，以此连续

		计算 PWM 的频率
	ECAP_Countinue_Delta	设置 ECAP 工作在连续采样的 Delta 模式,并连续对 PWM 波形进行事件采样,以此连续计算 PWM 的频率
	ECAP_Oneshot_Absolute	设置 ECAP 工作在 oneshot 的 Absolute 模式,并对 PWM 波形进行 4 次事件采样,以此计算 PWM 的频率
Flash	Flash_EEPROM_Emulation	演示如何将 flash 模拟成 EEPROM,以增加 flash 的使用寿命
	Flash_Frequency_Reduction	演示如何在系统运行过程中切换 flash 的工作频率
	Flash_M_Access_User	演示一个程序如何访问存储在 flash 上另一个存储区域的函数,与此示例相配合的另外一个示例名称为“Flash_User_FuncWrap”
	Flash_Operation	演示 flash 的读写擦操作
	Flash_Sector_Protect	设置 flash 的某一扇区 (sector) 为保护模式,对此扇区的写入及擦除操作将失败
	Flash_User_FuncWrap	配合“Flash_M_Access_User”的示例代码,此示例演示如何将一个单独的函数接口存储在 flash 中,并能够被其它程序访问
	Flash_With_INT	此示例代码演示,如何新建一个工程,将所有的代码及数据全部放在 RAM 中运行 (包括终端向量表)。此工程示例,在写/擦除 Flash 时,可能会有终端产生的应用情形中会用到。
GPIO	GPIO_Edge_Detect	检测 GPIO 是否发生电平反转 (采用边沿检测方式判断)
	GPIO_Level_Test	检测 GPIO 电平为高还是低
GTimer	GTimer_INT	设置 Timer 开始计时,结束后进入中断
I2C	I2C_Master_Bulk_Polling_TxRx	设置 I2C 作为 Master 工作在 Bulk 模式,并用 polling 模式向 Slave 发送数据,并用 polling 模式接收 Slave 返回的数据,对比发送和接收的数据是否一致;此示例为 Master 端代码,与“I2C_Slave_Bulk_Polling_TxRx”配套
	I2C_Master_INT_Rx	设置 I2C 作为 Master,采用 polling 方式向 Slave 发送数据,并采用中断方式接收 Slave 返回的数据
	I2C_Master_Polling_TxRx	设置 I2C 作为 Master,并用 polling 模式向 Slave 发送数据,并用 polling 模式接收 Slave 返回的数据,对比发送和接收的数据是否一致;此示例为 Master 端代码,与“I2C_Slave_Polling_TxRx”配套
	I2C_Slave_Bulk_Polling_TxRx	与“I2C_Master_Bulk_Polling_TxRx”配套,此示例代码为 Slave 端代码
	I2C_Slave_Polling_TxRx	与“I2C_Master_Polling_TxRx”配套,此示例代

		码为 Slave 端代码
PGA	PGA_Calibration	校准 PGA 的增益及偏置数值
PWM	PWM_Complementary_Pair_Channel	设置某路 PWM，并使其两个通道（A/B）产生一对互补的波形
	PWM_Current_Protect_Trigger_TZ	采用 PGA 对电流监测点的电压进行放大，并送入 Comparator 进行过流或欠流检测，最后设置 PWM 接收 Comparator 的信号作为触发 TZ 的触发源，以此关闭发生过流或者欠流的 PWM 波形
	PWM_Force_SYNC	PWM0 发出同步信号同步使 PWM1/2 的波形与 PWM0 同步
	PWM_Global_GPIO_SYNC	采用 GPIO 同步 PWM0/1/2 的波形
	PWM_Global_Software_Force_SYNC	采用软件同步信号同步 PWM0/1/2 的波形
	PWM_Global_Timer_SYNC	采用计时器同步 PWM0/1/2 的波形
	PWM_GPIO_Trigger_Trip_Zone	设置 GPIO 为触发 Trip Zone 的触发源，示例中 GPIO16 作为 TZ0 触发源，并设置为 CBC 模式，当为低电平是触发 TZ 事件，PWM 关闭波形输出，直到清除 TZ 标志位。GPIO17/18 作为 TZ1/2 触发源，并设置为 one-shot 模式，当为低电平时，触发 TZ 事件，PWM 关闭波形输出，但在下一个 PWM 时钟又立马恢复波形输出
	PWM_Single_Output_With_Down_Counting_Mode	PWM 以向下计数模式输出单通道波形
	PWM_Single_Output_With_Up_Counting_Mode	PWM 以向上计数模式输出单通道波形
	PWM_Single_Output_With_Up_Down_Counting_Mode	PWM 以上下计数模式输出单通道波形
	PWM_TBCNT_0_SYNC	设置 PWM0 在 TBCNT=0 时发出同步信号，并发送给 PWM1/2，使 PWM1/2 波形与 PWM0 同步
	PWM_TBCNT_CMPD_SYNC	设置 PWM0 在 TBCNT=CMPD 时发出同步信号，并发送给 PWM1/2，使 PWM1/2 波形与 PWM0 同步
	PWM_Trigger_ADC_Sample	设置 PWM0 触发 ADC A/B/C 三个采样器同时工作
	PWM_TripPhase_Waveform	设置 PWM 输出电机三相波形
SSP	SSP_Master_B2B_TxRx_Polling	SSP 作为 Master 以 B2B 模式采用 polling 方式向 Slave 发送数据，并以 polling 方式接收 Slave 返回的数据，对比源数据与返回数据，检查数据是否一致；此示例为 Master 端代码，与 Slave 端的“SSP_Slave_TxRx_Polling”配套
	SSP_Master_TxRx_INT	SSP 作为 Master 采用中断方式向 Slave 发送数据，并以中断方式接收 Slave 返回的数据，对

		比源数据与返回数据，检查数据是否一致；此示例为 Master 端代码，与 Slave 端的“SSP_Slave_TxRx_INT”配套
	SSP_Master_TxRx_Polling	SSP 作为 Master 采用 polling 方式向 Slave 发送数据，并以 polling 方式接收 Slave 返回的数据，对比源数据与返回数据，检查数据是否一致；此示例为 Master 端代码，与 Slave 端的“SSP_Slave_B2B_TxRx_Polling”配套
	SSP_Slave_B2B_TxRx_Polling	与“SSP_Master_B2B_TxRx_Polling”配套，此示例为 Slave 端代码
	SSP_Slave_TxRx_INT	与“SSP_Master_TxRx_INT”配套，此示例为 Slave 端代码
	SSP_Slave_TxRx_Polling	与“SSP_Master_TxRx_Polling”配套，此示例代码为 Slave 端代码
T-Sensor	TSensor	设置 ECU 内部温度传感器，检测芯片温度
UART	UART_RX_and_SentBack	此示例为 UART 的数据接收端代码，采用 polling 方式接收发送端的数据，并将接收的数据发送回发送端
	UART_RX_INT	UART 以中断方式接收发送端的数据
	UART_TX_and_CheckRX	此示例为 UART 的数据发送端代码，采用 polling 方式发送数据给接收端，并将接收接收端返回的数据进行数据对比
WTD	WDT0_INT	WTD 计时 500ms，结束后进入中断
	WDT1_Feed_DOG	WTD 计时 500ms，但在结束之前重新喂狗，使其不进入中断
	WDT1_Rset_SYS	WTD 计时 500ms，进入中断后不清中断，使系统复位