Ameba-Z WAKEUP ON ADC

This document introduces usage of ADC WAKEUP.

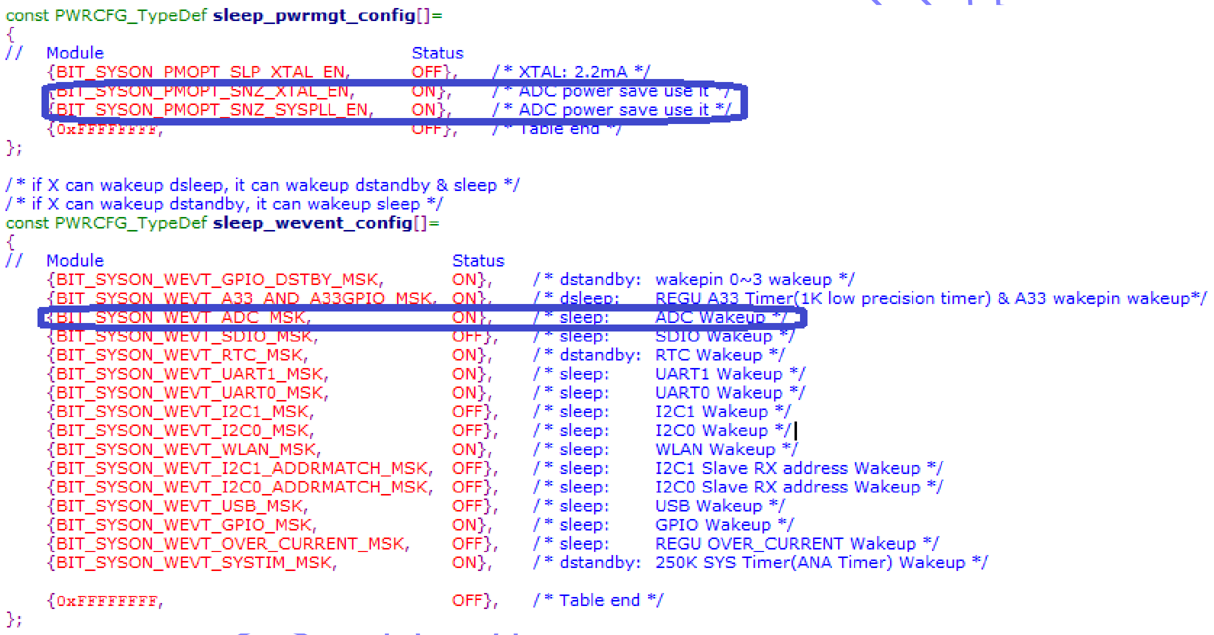
Table of Contents

1. Summary

ADC can receive data under sleep mode periodically, and wakeup CPU when the number of entries in the FIFO is more than or equal to the FIFO threshold value

2. ADC Wakeup

You should configure SDK like following:



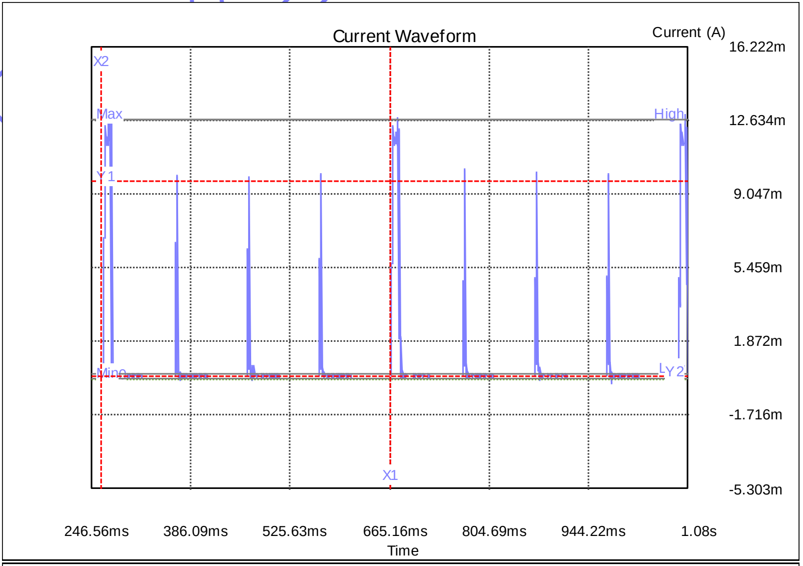
```
const PWRCFG_TypeDef sleep_pwrmgt_config[]=
{
//	Module						Status
	{BIT_SYSON_PMOPT_SLP_XTAL_EN,			OFF},		/* XTAL: 2.2mA */
	{BIT_SYSON_PMOPT_SNZ_XTAL_EN,			ON},		/* ADC power save use it */
	{BIT_SYSON_PMOPT_SNZ_SYSPLL_EN,			ON},		/* ADC power save use it */
	{0xFFFFFFFF,					OFF},		/* Table end */
};

/* if X can wakeup dsleep, it can wakeup dstandby & sleep */
/* if X can wakeup dstandby, it can wakeup sleep */
const PWRCFG_TypeDef sleep_wevent_config[]=
{
//	Module						Status
	{BIT_SYSON_WEVT_GPIO_DSTBY_MSK,			ON},	/* dstandby:	wakepin 0~3 wakeup */
	{BIT_SYSON_WEVT_A33_AND_A33GPIO_MSK,		ON},	/* dsleep:	REGU A33 Timer(1K low precision timer) & A33 wakepin wakeup*/
	{BIT_SYSON_WEVT_ADC_MSK,			ON},	/* sleep:	ADC Wakeup */
	{BIT_SYSON_WEVT_SDIO_MSK,			OFF},	/* sleep:	SDIO Wakeup */
	{BIT_SYSON_WEVT_RTC_MSK,			ON},	/* dstandby:	RTC Wakeup */
	{BIT_SYSON_WEVT_UART1_MSK,			ON},	/* sleep:	UART1 Wakeup */
	{BIT_SYSON_WEVT_UART0_MSK,			ON},	/* sleep:	UART0 Wakeup */
	{BIT_SYSON_WEVT_I2C1_MSK,			OFF},	/* sleep:	I2C1 Wakeup */
	{BIT_SYSON_WEVT_I2C0_MSK,			OFF},	/* sleep:	I2C0 Wakeup */
	{BIT_SYSON_WEVT_WLAN_MSK,			ON},	/* sleep:	WLAN Wakeup */
	{BIT_SYSON_WEVT_I2C1_ADDRMATCH_MSK,		OFF},	/* sleep:	I2C1 Slave RX address Wakeup */
	{BIT_SYSON_WEVT_I2C0_ADDRMATCH_MSK,		OFF},	/* sleep:	I2C0 Slave RX address Wakeup */
	{BIT_SYSON_WEVT_USB_MSK,			OFF},	/* sleep:	USB Wakeup */
	{BIT_SYSON_WEVT_GPIO_MSK,			ON},	/* sleep:	GPIO Wakeup */
	{BIT_SYSON_WEVT_OVER_CURRENT_MSK,		OFF},	/* sleep:	REGU OVER_CURRENT Wakeup */
	{BIT_SYSON_WEVT_SYSTIM_MSK,			ON},	/* dstandby:	250K SYS Timer(ANA Timer) Wakeup */

	{0xFFFFFFFF,					OFF},	/* Table end */
};
```

3. Example

```
void ADC_Handler(void *Data)
{
	u32 buf[30];
	u32 isr = 0;
	u32 i = 0;

	isr = ADC_GetISR();
	if (isr & BIT_ADC_FIFO_THRESHOLD) {
		for(i = 0; i < 30; i++) {
			buf[i] = (u32)ADC_Read();
		}
	}

	ADC_INTClear();
	DBG_8195A("ADC BUF: %08x \n", buf[0]);
}

VOID ADC_Init(VOID)
{
	ADC_InitTypeDef ADCInitStruct;

	/* ADC Interrupt Initialization */
	InterruptRegister((IRQ_FUN)&ADC_Handler, ADC_IRQ, (u32)NULL, 5);
	InterruptEn(ADC_IRQ, 5);

	/* To release ADC delta sigma clock gating */
	PLL2_Set(BIT_SYS_SYSPLL_CK_ADC_EN, ENABLE);

	/* Turn on ADC active clock */
	RCC_PeriphClockCmd(APBPeriph_ADC, APBPeriph_ADC_CLOCK, ENABLE);

	ADC_InitStruct(&ADCInitStruct);
	ADCInitStruct.ADC_BurstSz = 8;
	ADCInitStruct.ADC_OneShotTD = 8; /* means 4 times */
	ADC_Init(&ADCInitStruct);
	ADC_SetOneShot(ENABLE, 100, ADCInitStruct.ADC_OneShotTD); /* 100 will take 200ms */

	ADC_INTClear();
	ADC_Cmd(ENABLE);
} ? end ADC_Init ?
```

4. Power Consumption

ADC catches data every 100ms and wakeup CPU every 400ms.



Calculated Measurements (3374Hz sample rate)

Dc 738.3929uA	Rms 2.1044mA	X1 665.1562ms	Y1 9.6934m
Low 315.5780uA	Min 94.6646uA	X2 260.9517ms	Y2 143.2738u
High 12.6935mA	Max 12.6935mA	dX 404.2046ms	dY -9.5501m