AT91SAM7X-EK softpack 1.5 代码解读之 PWM

——基于 SAM7X-EK 和 IAR EWARM

文档编号	MAN2013A_CH				
文档版本	Rev. A				
文档摘要	基于 AT91SAM7X-EK 开发板的代码解读，PWM 代码解读				
关键词	AT91SAM7X256 SAM7X-EK J-LINK IAR EWARM PWM				
创建日期	2010-06-25	创建人员	Cust126	审核人员	Robin
文档类型	公开发布/开发板配套文件				
版权信息	Mcuzone 原创文档，转载请注明出处				

更新历史

版本	时间	更新	作者
Rev. A	2010-06-25	初始创建	Cust126

微控电子 乐微电子
杭州市登云路 639 号 2B143
销售 TEL：86-571-89908193 13957118045
支持 TEL：18913989166 13957118045
FAX：86-571-89908193
www.mcuzone.com www.atarm.com

1.概述

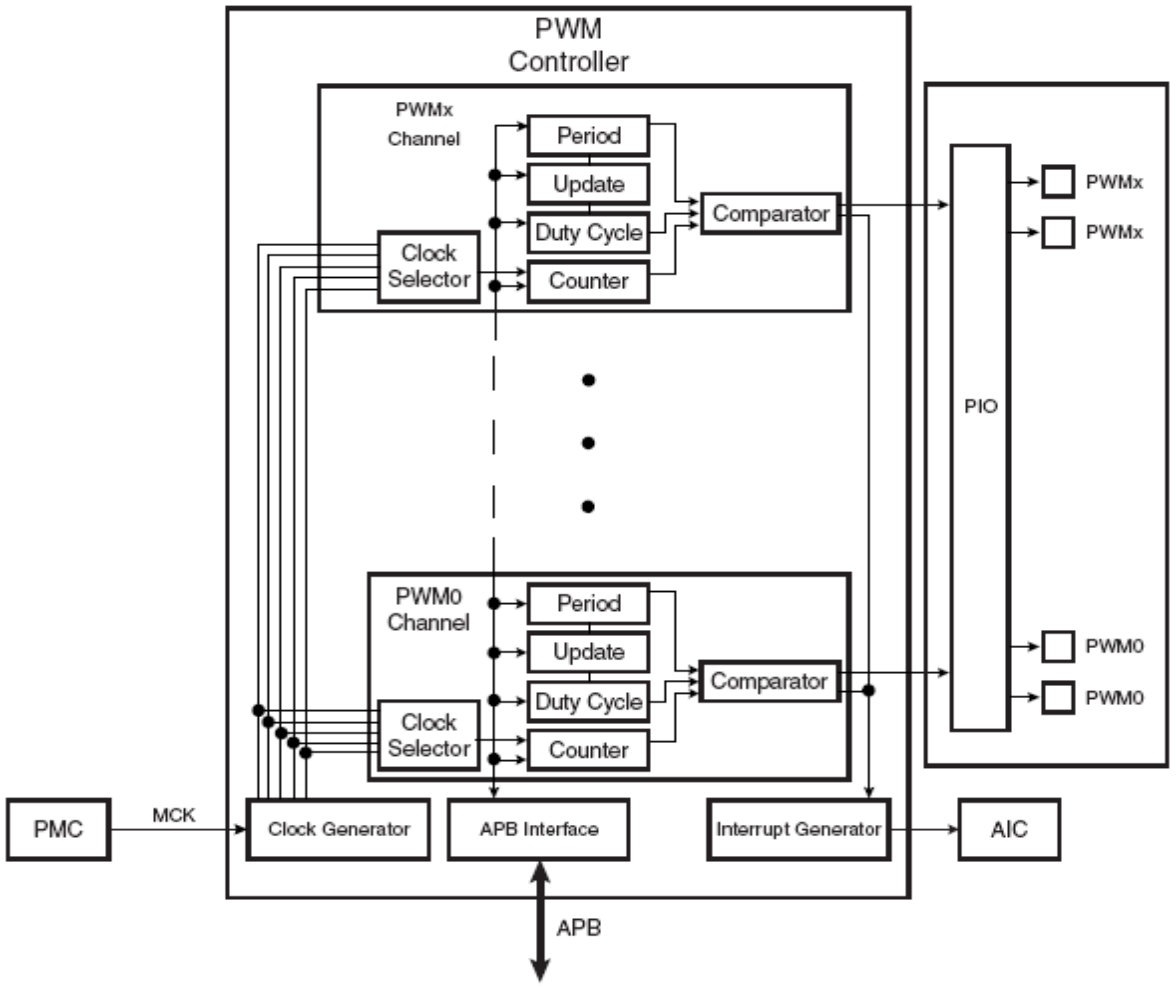
本文档以 SAM7X-EK 为硬件平台，IAR EWARM 为编译器平台，使用 J-Link 作为调试工具，演示并解读 AT91SAM7X256 的 PWM 操作流程。

2. PWM 操作

2.1 PWM 操作流程介绍

PWM 宏单元独立控制几个通道。每个通道控制一个输出方波。通过用户接口可配置输出波形的周期、占空比及极性。每个通道选择并使用一个由时钟产生器提供的时钟。时钟发生器提供几个由 PWM 宏单元分频主时钟后得到的时钟。

PWM 控制器的框图：



2.2 PWM 操作的目的与功能描述

AT91 softpack 1.5 的 SAM7X-EK 包内的 basic-pwm-project 工程的功能是向 DS3、DS4 两个 LED 逐渐亮和灭。

2.3 示例代码

以下代码截取自 AT91 softpack 1.5 的 SAM7X-EK 包内的 basic-pwm-project ， 基于 IAR EWARM 平台

2.3.1 main 函数代码注释解读

下面对 basic-pwm-project 的主要代码进行注释解读，首先是 main 函数内容：

```
//------------------------------------------------------------------------------
//               Global functions
//------------------------------------------------------------------------------
//输出 PWM 信号是 LED1 和 LED2 逐渐的亮和灭
//------------------------------------------------------------------------------
/// Outputs a PWM on LED1 & LED2 to makes it fade in and out.
//------------------------------------------------------------------------------
int main(void)
{
    PIO_Configure(pins, PIO_LISTSIZE(pins));
    TRACE_CONFIGURE(DBGU_STANDARD, 115200, BOARD_MCK);
    printf("-- Basic PWMC Project %s --\n\r", SOFTPACK_VERSION);
    printf("-- %s\n\r", BOARD_NAME);
    printf("-- Compiled: %s %s --\n\r", __DATE__, __TIME__);

    UTIL_WaitTimeInMs(BOARD_MCK, 1000);
    UTIL_WaitTimeInUs(BOARD_MCK, 1000);

    // Enable PWMC peripheral clock              //使能 PWMC 外设时钟
    AT91C_BASE_PMC->PMC_PCER = 1 << AT91C_ID_PWMC;

    // Settings:
    // - 100kHz PWM period (PWM_FREQUENCY)
    // - 1s rise/fall time for the LED intensity

    // Set clock A to run at 100kHz * MAX_DUTY_CYCLE (clock B is not used)
    PWMC_ConfigureClocks(PWM_FREQUENCY * MAX_DUTY_CYCLE, 0, BOARD_MCK);

    // Configure PWMC channel for LED0 (left-aligned)     //配置 PWMC 通道 1 为 LED0(左对齐)
    PWMC_ConfigureChannel(CHANNEL_PWM_LED0, AT91C_PWMC_CPRE_MCKA, 0, 0);
    PWMC_SetPeriod(CHANNEL_PWM_LED0, MAX_DUTY_CYCLE);
    PWMC_SetDutyCycle(CHANNEL_PWM_LED0, MIN_DUTY_CYCLE);

    // Configure PWMC channel for LED1 (center-aligned, inverted polarity) //配置 PWMC 通道 2 为 LED1(中心对
齐)

    PWMC_ConfigureChannel(CHANNEL_PWM_LED1, AT91C_PWMC_CPRE_MCKA, AT91C_PWMC_CALG,
AT91C_PWMC_CPOL);
    PWMC_SetPeriod(CHANNEL_PWM_LED1, MAX_DUTY_CYCLE);
    PWMC_SetDutyCycle(CHANNEL_PWM_LED1, MIN_DUTY_CYCLE);

    // Configure interrupt on channel #1    //配置通道 1 中断
    AIC_ConfigureIT(AT91C_ID_PWMC, 0, ISR_Pwmc);
    AIC_EnableIT(AT91C_ID_PWMC);
```

```
    PWMC_EnableChannelIt(CHANNEL_PWM_LED0);

    // Enable channel #1 and #2          //使能通道 1 和 2
    PWMC_EnableChannel(CHANNEL_PWM_LED0);
    PWMC_EnableChannel(CHANNEL_PWM_LED1);

    // Infinite loop
    while (1);
}
```

2.3.2 重要子函数代码注释解读

以下是对几个重要的子函数进行分析：

```
//------------------------------------------------------------------------------
/// Configures PWM clocks A & B to run at the given frequencies. This function
/// finds the best MCK divisor and prescaler values automatically.
/// \param clka    Desired clock A frequency (0 if not used).
/// \param clkb    Desired clock B frequency (0 if not used).
/// \param mck     Master clock frequency.
//配置 PWM 时钟 A 或 B 是 PWM 运行在给定的频率下，自动设定 MCK 的预分频的值。
//------------------------------------------------------------------------------
void PWMC_ConfigureClocks(unsigned int clka, unsigned int clkb, unsigned int mck)
{
    unsigned int mode = 0;
    unsigned int result;

    // Clock A
    if (clka != 0) {

        result = FindClockConfiguration(clka, mck);
        ASSERT(result != 0, "-F- Could not generate the desired PWM frequency (%uHz)\n\r", clka);
        mode |= result;
    }

    // Clock B
    if (clkb != 0) {

        result = FindClockConfiguration(clkb, mck);
        ASSERT(result != 0, "-F- Could not generate the desired PWM frequency (%uHz)\n\r", clkb);
        mode |= (result << 16);
    }

    // Configure clocks
    TRACE_DEBUG("Setting PWMC_MR = 0x%08X\n\r", mode);
```

```
    AT91C_BASE_PWMC->PWMC_MR = mode;
}
```

2.3.3 运行结果

LED0 和 LED1 灯逐渐的亮之后逐渐的灭，交替出现。