GigaDevice Semiconductor Inc.

GD32 时钟切换配置使用指南

应用笔记

AN250

1.0 版本

（2024 年 12 月）

目录

表索引

1. 简介

本文主要介绍GD32系列需要时钟切换时如何对系统时钟进行重新配置。

当系统时钟频率从高主频直接切换到低主频或从低主频直接切换到高主频时，如在Boot+APP架构中，或进出deepsleep需要重新配置时钟等，在这类场景中需要加上阶段式地升频或降频操作，先将频率阶段式地降下来，然后将系统时钟源切换到其他时钟源，如内部高速晶振，再去修改PLL，最后阶段式升到目标频率。

2. 开发环境

2.1. 硬件环境

- 硬件开发板：GD32F307C-EVAL 开发板
- 烧录工具：板载 GD-Link

2.2. 软件环境

- 操作系统：Win10-64 位
- 开发环境：KEIL 5.29
- 固件库：GD32F30x_Firmware_Library

GD32F30x_Firmware_Library 的获取可以从网站链接为：www.gd32mcu.com 下载获得。

3. 时钟频率切换配置

系统级应用中，会有如 Boot+APP 的架构设计这种应用场景。在 Boot 程序中，系统时钟的变化为先由内部高速晶振切换到 PLL 最高频率的变化过程，需要应用层注意添加阶段式切频代码，或者在 APP 工程中，如对目标频率需求没有变化，无需重新配置时钟。

3.1. 阶段式降频

3.1.1. 降频代码

应用层需要避免由高频直接切换到低频（如 IRC8M）。*表 3-1. 三段式降频示例代码*给出了三段式降频的示例代码。假设当前时钟为 120MHz，调用下面三段式降频代码后，频率的变化为 120MHz→60MHz→30MHz→15MHz，接下来再将系统时钟切换到内部高速晶振。

表 3-1. 三段式降频示例代码

```
#define RCU_MODIFY_DE_3(__delay)   do{                                        \
                                     volatile uint32_t i,reg;                 \
                                     if(0 != __delay){                        \
                                         /* Insert a software delay */        \
                                         for(i=0; i<__delay; i++){            \
                                         }                                    \
                                         reg = RCU_CFG0;                      \
                                         reg &= ~(RCU_CFG0_AHBPSC);           \
                                         reg |= RCU_AHB_CKSYS_DIV2;           \
                                         /* AHB = SYSCLK/2 */                 \
                                         RCU_CFG0 = reg;                      \
                                         /* Insert a software delay */        \
                                         for(i=0; i<__delay; i++){            \
                                         }                                    \
                                         reg = RCU_CFG0;                      \
                                         reg &= ~(RCU_CFG0_AHBPSC);           \
                                         reg |= RCU_AHB_CKSYS_DIV4;           \
                                         /* AHB = SYSCLK/4 */                 \
                                         RCU_CFG0 = reg;                      \
                                         /* Insert a software delay */        \
                                         for(i=0; i<__delay; i++){            \
                                         }                                    \
                                         reg = RCU_CFG0;                      \
                                         reg &= ~(RCU_CFG0_AHBPSC);           \
                                         reg |= RCU_AHB_CKSYS_DIV8;           \
                                         /* AHB = SYSCLK/8 */                 \
                                         RCU_CFG0 = reg;                      \
                                         /* Insert a software delay */          \
                                         for(i=0; i<__delay; i++){              \
                                         }                                      \
                                     }                                          \
                                 }while(0)
```

3.1.2. 降频具体示例

*表 3-2. 降频示例代码*给出了降频应用的示例代码，调用 switch_system_clock_to_72m_irc8m 函数前，系统工作在高频率下。

表 3-2. 降频示例代码

```
......
/* function declaration */
static void switch_system_clock_to_72m_irc8m(void);
......
int main(void)
{
    ......
    /* clock switching */
    switch_system_clock_to_72m_irc8m();
    ......
}
.......

static void switch_system_clock_to_72m_irc8m(void)
{
    uint32_t timeout = 0U;
    uint32_t stab_flag = 0U;

    /* enable IRC8M */
    RCU_CTL |= RCU_CTL_IRC8MEN;

    /* wait until IRC8M is stable or the startup time is longer than IRC8M_STARTUP_TIMEOUT */
    do{
        timeout++;
        stab_flag = (RCU_CTL & RCU_CTL_IRC8MSTB);
    }while((0U == stab_flag) && (IRC8M_STARTUP_TIMEOUT != timeout));

    /* if failed */
    if(0U == (RCU_CTL & RCU_CTL_IRC8MSTB)){
      while(1){
      }
    }
    /* add frequency reduction code to avoid switching directly from high frequency to the lowest
frequency */
    RCU_MODIFY_DE_3(0x50);
    /* select IRC8M as system clock source, deinitialize the RCU */
    rcu_system_clock_source_config(RCU_CKSYSSRC_IRC8M);
    rcu_deinit();
    /* AHB = SYSCLK */
    RCU_CFG0 |= RCU_AHB_CKSYS_DIV1;
    /* APB2 = AHB */
    RCU_CFG0 |= RCU_APB2_CKAHB_DIV1;
    /* APB1 = AHB */
    RCU_CFG0 |= RCU_APB1_CKAHB_DIV1;
    /* PLL = (IRC8M/2) * 18 = 72 MHz */
    RCU_CFG0 &= ~(RCU_CFG0_PLLSEL | RCU_CFG0_PLLMF);
    RCU_CFG0 |= (RCU_PLLSRC_IRC8M_DIV2 | RCU_PLL_MUL18);

    /* enable PLL */
    RCU_CTL |= RCU_CTL_PLLEN;

    /* wait until PLL is stable */
    while(0 == (RCU_CTL & RCU_CTL_PLLSTB));

    /* select PLL as system clock */
    RCU_CFG0 &= ~RCU_CFG0_SCS;
    RCU_CFG0 |= RCU_CKSYSSRC_PLL;

    /* wait until PLL is selected as system clock */
    while(0 == (RCU_CFG0 & RCU_SCSS_PLL));
}
```

3.2. 阶段式升频

3.2.1. 升频代码

应用层需要避免直接由低频（如 IRC8M）直接切到最高频率，建议加上*表 3-3. 三段式升频示例代码*所列出的代码。

表 3-3. 三段式升频示例代码

```
#define RCU_MODIFY_UP_3(__delay)   do{                              \
                                     volatile uint32_t i,reg;       \
                                     if(0 != __delay){              \
                                         /* Insert a software delay */  \
                for(i=0; i<__delay; i++){                \
                }                                        \
                reg = RCU_CFG0;                          \
                reg &= ~(RCU_CFG0_AHBPSC);               \
                reg |= RCU_AHB_CKSYS_DIV4;               \
                /* AHB = SYSCLK/4 */                     \
                RCU_CFG0 = reg;                          \
                /* Insert a software delay */            \
                for(i=0; i<__delay; i++){                \
                }                                        \
                reg = RCU_CFG0;                          \
                reg &= ~(RCU_CFG0_AHBPSC);               \
                reg |= RCU_AHB_CKSYS_DIV2;               \
                /* AHB = SYSCLK/2 */                     \
                RCU_CFG0 = reg;                          \
                /* Insert a software delay */            \
                for(i=0; i<__delay; i++){                \
                }                                        \
                reg = RCU_CFG0;                          \
                reg &= ~(RCU_CFG0_AHBPSC);               \
                reg |= RCU_AHB_CKSYS_DIV1;               \
                /* AHB = SYSCLK/1 */                     \
                RCU_CFG0 = reg;                          \
            }                                            \
        }while(0)
```

3.2.2. 升频具体示例

*表 3-4. 升频示例代码*给出了升频应用的示例代码。deepsleep 唤醒之后频率由内部高速晶振倍频到特定频率，系统频率的变化是先由内部高速晶振进行 8 分频，然后配置 PLL 并将时钟源切到 PLL，接下来开始逐级升频。

表 3-4. 升频示例代码

```
......
/* function declaration */
static void switch_system_clock_reconfig(void);
static void __system_clock_120m_hxtal(void);
......
int main(void)
{
    ......
    /* enter deepsleep mode */
    pmu_to_deepsleepmode(PMU_LDO_LOWPOWER,            PMU_LOWDRIVER_DISABLE,
WFI_CMD);
```

```
        /* clock switching */
        switch_system_clock_reconfig();
        ......
}
.......

static void switch_system_clock_reconfig(void)
{
        /* FPU settings */
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
        SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));    /* set CP10 and CP11 Full Access */
#endif
        /* reset the RCU clock configuration to the default reset state */
        /* Set IRC8MEN bit */
        RCU_CTL |= RCU_CTL_IRC8MEN;
        while(0U == (RCU_CTL & RCU_CTL_IRC8MSTB)){
        }
        /* add frequency reduction code to avoid switching directly from high frequency to the lowest
frequency */
        if(((RCU_CFG0 & RCU_CFG0_SCSS) == RCU_SCSS_PLL)){
                RCU_MODIFY_DE_3(0x50);
         }

        RCU_CFG0 &= ~RCU_CFG0_SCS;

#if (defined(GD32F30X_HD) || defined(GD32F30X_XD))
        /* reset HXTALEN, CKMEN and PLLEN bits */
        RCU_CTL &= ~(RCU_CTL_PLLEN | RCU_CTL_CKMEN | RCU_CTL_HXTALEN);
        /* disable all interrupts */
        RCU_INT = 0x009f0000U;
#elif defined(GD32F30X_CL)
        /* reset HXTALEN, CKMEN, PLLEN, PLL1EN and PLL2EN bits */
        RCU_CTL    &=    ~(RCU_CTL_PLLEN    |RCU_CTL_PLL1EN    |    RCU_CTL_PLL2EN    |
RCU_CTL_CKMEN | RCU_CTL_HXTALEN);
        /* disable all interrupts */
        RCU_INT = 0x00ff0000U;
#endif

        /* reset HXTALBPS bit */
        RCU_CTL &= ~(RCU_CTL_HXTALBPS);

        /* reset CFG0 and CFG1 registers */
        RCU_CFG0 = 0x00000000U;
```

```
    RCU_CFG1 = 0x00000000U;
    /* configure the system clock source, PLL Multiplier, AHB/APBx prescalers and Flash settings */
    __system_clock_120m_hxtal();
}

static void __system_clock_120m_hxtal(void)
{
    uint32_t timeout = 0U;
    uint32_t stab_flag = 0U;

    /* enable HXTAL */
    RCU_CTL |= RCU_CTL_HXTALEN;

    /* wait until HXTAL is stable or the startup time is longer than HXTAL_STARTUP_TIMEOUT */
    do{
        timeout++;
        stab_flag = (RCU_CTL & RCU_CTL_HXTALSTB);
    }while((0U == stab_flag) && (HXTAL_STARTUP_TIMEOUT != timeout));

    /* if fail */
    if(0U == (RCU_CTL & RCU_CTL_HXTALSTB)){
        while(1){
        }
    }

    RCU_APB1EN |= RCU_APB1EN_PMUEN;
    PMU_CTL |= PMU_CTL_LDOVS;

    /* HXTAL is stable */
    /* AHB = SYSCLK/8 */
    RCU_CFG0 |= RCU_AHB_CKSYS_DIV8;
    /* APB2 = AHB/1 */
    RCU_CFG0 |= RCU_APB2_CKAHB_DIV1;
    /* APB1 = AHB/2 */
    RCU_CFG0 |= RCU_APB1_CKAHB_DIV2;

#if (defined(GD32F30X_HD) || defined(GD32F30X_XD))
    /* select HXTAL/2 as clock source */
    RCU_CFG0 &= ~(RCU_CFG0_PLLSEL | RCU_CFG0_PREDV0);
    RCU_CFG0 |= (RCU_PLLSRC_HXTAL_IRC48M | RCU_CFG0_PREDV0);

    /* CK_PLL = (CK_HXTAL/2) * 30 = 120 MHz */
    RCU_CFG0 &= ~(RCU_CFG0_PLLMF | RCU_CFG0_PLLMF_4 | RCU_CFG0_PLLMF_5);
    RCU_CFG0 |= RCU_PLL_MUL30;

#elif defined(GD32F30X_CL)
    /* CK_PLL = (CK_PREDIV0) * 30 = 120 MHz */
    RCU_CFG0 &= ~(RCU_CFG0_PLLMF | RCU_CFG0_PLLMF_4 | RCU_CFG0_PLLMF_5);
    RCU_CFG0 |= (RCU_PLLSRC_HXTAL_IRC48M | RCU_PLL_MUL30);

    /* CK_PREDIV0 = (CK_HXTAL)/5 *8 /10 = 4 MHz */
    RCU_CFG1     &=     ~(RCU_CFG1_PLLPRESEL     |     RCU_CFG1_PREDV0SEL     |
RCU_CFG1_PLL1MF | RCU_CFG1_PREDV1 | RCU_CFG1_PREDV0);
    RCU_CFG1     |=     (RCU_PLLPRESRC_HXTAL     |     RCU_PREDV0SRC_CKPLL1     |
RCU_PLL1_MUL8 | RCU_PREDV1_DIV5 | RCU_PREDV0_DIV10);

    /* enable PLL1 */
    RCU_CTL |= RCU_CTL_PLL1EN;
    /* wait till PLL1 is ready */
    while((RCU_CTL & RCU_CTL_PLL1STB) == 0U){
    }
#endif /* GD32F30X_HD and GD32F30X_XD */

    /* enable PLL */
    RCU_CTL |= RCU_CTL_PLLEN;

    /* wait until PLL is stable */
    while(0U == (RCU_CTL & RCU_CTL_PLLSTB)){
    }

    /* enable the high-drive to extend the clock frequency to 120 MHz */
    PMU_CTL |= PMU_CTL_HDEN;
    while(0U == (PMU_CS & PMU_CS_HDRF)){
    }

    /* select the high-drive mode */
    PMU_CTL |= PMU_CTL_HDS;
    while(0U == (PMU_CS & PMU_CS_HDSRF)){
    }

    /* select PLL as system clock */
    RCU_CFG0 &= ~RCU_CFG0_SCS;
    RCU_CFG0 |= RCU_CKSYSSRC_PLL;

    /* wait until PLL is selected as system clock */
    while(0U == (RCU_CFG0 & RCU_SCSS_PLL)){
    }
    /* add frequency escalation code to avoid switching directly from lowest frequency to the high frequency */
    RCU_MODIFY_UP_3(0x50);
}
```

3.3. 其他注意事项

除了上述提供的宏可以参考以外，还可以调用 gd32f30x_rcu.c 中提供的 rcu_ahb_clock_config 函数实现 AHB 时钟的变化，注意在每次 AHB 频率变化之后，需要加一点延时保证时钟稳定。

要想实现频率的变化，须要先将时钟由 PLL 切到其他时钟源，然后修改 PLL，再将系统时钟源切回到 PLL。

对于其他系列，均可以参考类似做法，避免在时钟频率切换时出现异常。

4. 版本历史

表 **4-1.** 版本历史

版本号.	说明	日期
1.0	首次发布	2024 年 12 月 17 日

Important Notice