GigaDevice Semiconductor Inc.

GD32F10x 系列移植到 GD32F30x 系列

应用笔记

AN011

目录

图索引

表索引

1. 前言

本应用笔记旨在帮助您快速将应用程序从 GD32F10x 系列微控制器移植到 GD32F30x 系列微控制器。

为了更好的利用本应用笔记中的信息，您需要从官网 www.GD32MCU.com 下载 GD32 各系列微控制器资料，如 Datasheet、用户手册、官方例程及各种开发工具等。

2. 引脚兼容性

GD32F10x 与 GD32F30x 在相同封装下是 Pin To Pin 兼容的。但由于 GD32F30x 较 GD32F10x 增加了内部 48MHz RC 振荡器给 USBD 模块提供固定频率，为满足精度要求，GD32F30x 含有一个时钟校准控制器(CTC)，所以两者引脚定义有细微差别，如*表 2-1. GD32F10x 系列和 GD32F30x 系列引脚区别*所示：

表 2-1. GD32F10x 系列和 GD32F30x 系列引脚区别

引脚号	GD32F10x 系列引脚定义	GD32F30x 系列引脚定义
PF0	Default: PF0 Alternate: EXMC_A0	Default: PF0 Alternate: EXMC_A0 **Remap: CTC_SYNC**
PD15	Default: PD15 Alternate: EXMC_D1 Remap: TIMER3_CH3	Default: PD15 Alternate: EXMC_D1 Remap: TIMER3_CH3 ,**CTC_SYNC**
PA8	Default: PA8 Alternate: USART0_CK, TIMER0_CH0, CK_OUT0	Default: PA8 Alternate: USART0_CK, TIMER0_CH0, CK_OUT0**, CTC_SYNC**

3. 内部资源兼容性

*表 3-1. GD32F10x 系列和 GD32F30x 系列内部资源对比总览*给出了 GD32F10x 与 GD32F30x 的资源对比总览(以 GD32F103xE 和 GD32F303xE 对比为例)：

表 3-1. GD32F10x 系列和 GD32F30x 系列内部资源对比总览

片内资源	GD32F103xE	GD32F303xE	兼容性说明
主频	108MHz	120MHz	兼容
内核	M3 内核	M4 内核	M4 内核向下兼容
Flash	512K	512K	完全兼容
RAM	64K	64K	完全兼容
GPTM	4	4	完全兼容
Advanced TM	2(RE/VE/ZE)	1(CE) 2(RE/VE/ZE)	完全兼容
Basic TM	2	2	完全兼容
Systick	1	1	完全兼容
Watch dog	2	2	完全兼容
RTC	1	1	完全兼容
USART	3	3	兼容 F303：最高支持 7.5MHz(异步)/60MHz(同步) F103：最高支持 4.5MHz(异步)/54MHz(同步)
UART	2(RE/VE/ZE)	0(CE) 2(RE/VE/ZE)	兼容 F303：可最高支持 7.5MHz 频率 F103：最高支持 4.5MHz
I2C	2	2	兼容 F303：可最高支持 1000KHz 频率 F103：最高支持 400KHz
SPI/IIS	3/2	3/2	兼容 F303：可最高支持 30MHz 频率 F103：最高支持 18MHz
SDIO	1(RE/VE/ZE)	0(CE) (RE/VE/ZE)	完全兼容
CAN	1	1	完全兼容
USBD	1	1	完全兼容
GPIO	51(RE)/80(VE)/112(ZE)	37(CE)/51(RE)/80(VE)/112(ZE)	完全兼容
EXMC	0(RE)/1(VE/ZE)	0(CE/RE)/1(VE/ZE)	完全兼容
ADC(CH)	3(16)(RE/VE) 3(21)(ZE)	3(16)(CE/RE/VE) 3(21)(ZE)	完全兼容

片内资源	GD32F103xE	GD32F303xE	兼容性说明
DAC	2	2	完全兼容
CTC	0	1	GD32F30x 可提供内部自动校准 48MHz RC 晶振
Package	LQFP64(RE)/ LQFP100(VE)/ LQFP144(ZE)	LQFP48(CE)/ LQFP64(RE)/ LQFP100(VE)/ LQFP144(ZE)	完全兼容

4. 程序移植

由上节可看出，GD32F10x 和 GD32F30x 的主要差异性在于主频(RCU 系统时钟)、内核版本和 CTC 上，而 M4 内核是向下兼容 M3 的，所以无需修改，现就 RCU 方面阐述程序移植过程。

4.1. RCU 时钟配置

GD32F10x 系列和 GD32F30x 系列的时钟配置过程相同，但 GD32F30x 支持更高的系统时钟。若用户选择继续使用原有的时钟频率，则在应用程序中无需做任何改变；若用户选择使用更高的时钟频率，按以下步骤进行程序修改(以 GD32F103 移植到 GD32F303、使用外部 8MHz 高速晶振 HXTAL 为例，其他对应型号、使用内部晶振的移植过程类似)：

(1) 在 system_gd32f10x.c 文件中增加宏定义，如*图 4-1 在 system_gd32f10x.c 文件中增加宏定义*所示：

#define __SYSTEM_CLOCK_120M_PLL_HXTAL (uint32_t)(120000000)

图 4-1 在 system_gd32f10x.c 文件中增加宏定义

```
/* select a system clock by uncommenting the following line */
/* use IRC8M */
//#define __SYSTEM_CLOCK_48M_PLL_IRC8M            (uint32_t)(48000000)
//#define __SYSTEM_CLOCK_72M_PLL_IRC8M            (uint32_t)(72000000)
//#define __SYSTEM_CLOCK_108M_PLL_IRC8M           (uint32_t)(108000000)
//#define __SYSTEM_CLOCK_108M_PLL_IRC8M           (uint32_t)(108000000)

/* use HXTAL (XD series CK_HXTAL = 8M, CL series CK_HXTAL = 25M) */
//#define __SYSTEM_CLOCK_HXTAL                    (uint32_t)(__HXTAL)
//#define __SYSTEM_CLOCK_24M_PLL_HXTAL            (uint32_t)(24000000)
//#define __SYSTEM_CLOCK_36M_PLL_HXTAL            (uint32_t)(36000000)
//#define __SYSTEM_CLOCK_48M_PLL_HXTAL            (uint32_t)(48000000)
//#define __SYSTEM_CLOCK_56M_PLL_HXTAL            (uint32_t)(56000000)
//#define __SYSTEM_CLOCK_72M_PLL_HXTAL            (uint32_t)(72000000)
//#define __SYSTEM_CLOCK_96M_PLL_HXTAL            (uint32_t)(96000000)
//#define __SYSTEM_CLOCK_108M_PLL_HXTAL           (uint32_t)(108000000)
  #define __SYSTEM_CLOCK_120M_PLL_HXTAL           (uint32_t)(120000000)
```

(2) 在 system_gd32f10x.c 文件中增加使用 120MHz 频率函数的声明，如图 2 所示：

图 4-2 120MHz 函数声明

```
/* set the system clock frequency and declare the system clock configuration function */
#ifdef __SYSTEM_CLOCK_48M_PLL_IRC8M
uint32_t SystemCoreClock = __SYSTEM_CLOCK_48M_PLL_IRC8M;
static void system_clock_48m_irc8m(void);
#elif defined (__SYSTEM_CLOCK_72M_PLL_IRC8M)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_72M_PLL_IRC8M;
static void system_clock_72m_irc8m(void);
#elif defined (__SYSTEM_CLOCK_108M_PLL_IRC8M)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_108M_PLL_IRC8M;
static void system_clock_108m_irc8m(void);

#elif defined (__SYSTEM_CLOCK_HXTAL)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_HXTAL;
static void system_clock_hxtal(void);
#elif defined (__SYSTEM_CLOCK_24M_PLL_HXTAL)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_24M_PLL_HXTAL;
static void system_clock_24m_hxtal(void);
#elif defined (__SYSTEM_CLOCK_36M_PLL_HXTAL)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_36M_PLL_HXTAL;
static void system_clock_36m_hxtal(void);
#elif defined (__SYSTEM_CLOCK_48M_PLL_HXTAL)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_48M_PLL_HXTAL;
static void system_clock_48m_hxtal(void);
#elif defined (__SYSTEM_CLOCK_56M_PLL_HXTAL)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_56M_PLL_HXTAL;
static void system_clock_56m_hxtal(void);
#elif defined (__SYSTEM_CLOCK_72M_PLL_HXTAL)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_72M_PLL_HXTAL;
static void system_clock_72m_hxtal(void);
#elif defined (__SYSTEM_CLOCK_96M_PLL_HXTAL)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_96M_PLL_HXTAL;
static void system_clock_96m_hxtal(void);
#elif defined (__SYSTEM_CLOCK_108M_PLL_HXTAL)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_108M_PLL_HXTAL;
static void system_clock_108m_hxtal(void);
#elif defined (__SYSTEM_CLOCK_120M_PLL_HXTAL)
uint32_t SystemCoreClock = __SYSTEM_CLOCK_120M_PLL_HXTAL;
static void system_clock_120m_hxtal(void);
#endif /* __SYSTEM_CLOCK_48M_PLL_IRC8M */
```

(3) 在 system_gd32f10x.c 文件中增加使用 120MHz 频率函数的定义：

Table 4-1. Ddfinition of system_clock_120m_hxtal function

```
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){
        }
    }
    /* HXTAL is stable */
    /* AHB = SYSCLK */
    RCU_CFG0 |= RCU_AHB_CKSYS_DIV1;
```

```
    /* APB2 = AHB/1 */
    RCU_CFG0 |= RCU_APB2_CKAHB_DIV1;
    /* APB1 = AHB/2 */
    RCU_CFG0 |= RCU_APB1_CKAHB_DIV2;
#if (defined(GD32F10X_MD) || defined(GD32F10X_HD) || defined(GD32F10X_XD))
    /* select HXTAL/2 as clock source */
    RCU_CFG0 &= ~(RCU_CFG0_PLLSEL | RCU_CFG0_PREDV0);
    RCU_CFG0 |= (RCU_PLLSRC_HXTAL | RCU_CFG0_PREDV0);
    /* CK_PLL = (CK_HXTAL/2) * 30 = 120 MHz */
    RCU_CFG0 &= ~(RCU_CFG0_PLLMF | RCU_CFG0_PLLMF_4);
    RCU_CFG0 |= RCU_PLL_MUL30;
#elif defined(GD32F10X_CL)
    /* CK_PLL = (CK_PREDIV0) * 30 = 120MHz */
    RCU_CFG0 &= ~(RCU_CFG0_PLLMF | RCU_CFG0_PLLMF_4);
    RCU_CFG0 |= (RCU_PLLSRC_HXTAL | RCU_PLL_MUL30);
    /* CK_PREDIV0 = (CK_HXTAL)/5 *8 /10 = 4 MHz */
    RCU_CFG1 &= ~(RCU_CFG1_PREDV0SEL | RCU_CFG1_PLL1MF | RCU_CFG1_PREDV1 |
RCU_CFG1_PREDV0);
    RCU_CFG1 |= (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(0U == (RCU_CTL & RCU_CTL_PLL1STB)){
    }
#endif /* GD32F10X_MD and GD32F10X_HD and GD32F10X_XD */
    /* enable PLL */
    RCU_CTL |= RCU_CTL_PLLEN;
    /* wait until PLL is stable */
    while(0U == (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(0U == (RCU_CFG0 & RCU_SCSS_PLL)){
    }
}
```

(4) 在 system_gd32f10x.c 文件中增加使用 120MHz 频率函数的调用，如*图 4-3 120MHz 函数调用*所示：

图 4-3 120MHz 函数调用

```
static void system_clock_config(void)
{
#ifdef __SYSTEM_CLOCK_HXTAL
    system_clock_hxtal();
#elif defined (__SYSTEM_CLOCK_24M_PLL_HXTAL)
    system_clock_24m_hxtal();
#elif defined (__SYSTEM_CLOCK_36M_PLL_HXTAL)
    system_clock_36m_hxtal();
#elif defined (__SYSTEM_CLOCK_48M_PLL_HXTAL)
    system_clock_48m_hxtal();
#elif defined (__SYSTEM_CLOCK_56M_PLL_HXTAL)
    system_clock_56m_hxtal();
#elif defined (__SYSTEM_CLOCK_72M_PLL_HXTAL)
    system_clock_72m_hxtal();
#elif defined (__SYSTEM_CLOCK_96M_PLL_HXTAL)
    system_clock_96m_hxtal();
#elif defined (__SYSTEM_CLOCK_108M_PLL_HXTAL)
    system_clock_108m_hxtal();
#elif defined (__SYSTEM_CLOCK_120M_PLL_HXTAL)
    system_clock_120m_hxtal();

#elif defined (__SYSTEM_CLOCK_48M_PLL_IRC8M)
    system_clock_48m_irc8m();
#elif defined (__SYSTEM_CLOCK_72M_PLL_IRC8M)
    system_clock_72m_irc8m();
#elif defined (__SYSTEM_CLOCK_108M_PLL_IRC8M)
    system_clock_108m_irc8m();
#endif /* __SYSTEM_CLOCK_HXTAL */
}
```

5. 外设差异性

GD32F10x 与 GD32F30x 在外设上都是兼容的，但 GD32F30x 作为更高级的 MCU，较 GD32F10x 在很多外设上增加了部分功能，用户可根据以下罗列出的外设差异性选择是否使用这些功能。

5.1. 通用和备用输入/输出接口（GPIO 和 AFIO）

I/O 端口在作为输出使用时，GD32F30x 可将 IO 的速度设置为 120MHz(GD32F10x 最大 50MHz),当 I/O 端口输出速度大于 50MHz 时，建议使用 I/O 补偿单元对 I/O 端口进行斜率控制，从而降低 I/O 端口噪声对电源的影响。具体功能以及寄存器设置，请用户参考 GD32F30x 用户手册。

5.2. 模数转换器（ADC）

为减轻 CPU 的负担，GD32F30x 较 GD32F10x 增加了片上硬件过采样单元。它能够处理多个转换，并将多个转换的结果取平均，得出一个 16 位宽的数据。 片上硬件过采样单元是以降低数据输出率为代价，换取较高的数据分辨率。 具体功能以及寄存器设置，请用户参考 GD32F30x 用户手册。

5.3. 通用同步异步收发器（USART）

GD32F30x 较 GD32F10x 在 USART 上增加了块模式(GD32F10x 只支持字节模式)、数据极性设置、数据位反转以及 TX、RX 引脚电平反转等功能，因此，GD32F30x 多了三个寄存器，分别为：USART_CTL3、USART_RT、USART_STAT1。具体功能以及寄存器设置，请用户参考 GD32F30x 用户手册。

5.4. 内部集成电路总线接口（I2C）

GD32F30x 和 GD32F10x 的 I2C 都支持标速(最高 100KHz)和快速(最高 400KHz)，同时 GD32F30x 可支持高速模式(最高 1MHz)，要使能高速模式，需将 I2C_FMPCFG 寄存器的 FMPEN 置 1。具体功能以及寄存器设置，请用户参考 GD32F30x 用户手册。

5.5. 串行外设接口/片上音频接口（SPI/I2S）

GD32F30x 和 GD32F10x 的 SPI/I2S 模块差异性主要表现在 GD32F30x 支持 SPI TI 模式、SPI NSS 脉冲模式和 SPI 四线功能(只有 SPI0)，其中 SPI 的四线模式是用于控制四线 SPI Flash 外设，此模式下，数据传输速率是普通模式下的 4 倍。具体功能以及寄存器设置，请用户参考 GD32F30x 用户手册。

5.6. 通用串行总线全速设备接口（USBD）

GD32F30x 较 GD32F10x 在 USBD 外设中增加了 USB2.0 链接电源管理(LPM)等级 L1，目的是为了优化在挂起/恢复状态下的电源消耗。LPM 包括从 L0 到 L3 共 4 种状态。LPM L1 状态（睡眠状态）是新的电源管理状态。具体功能以及寄存器设置，请用户参考 GD32F30x 用户手册。

5.7. 闪存控制器（FMC）

GD32F30x 较 GD32F10x 增加了位编程功能，为用户节省一定的 Flash 空间。其特点是，存储在闪存中的数据，其值为“1”的 bit 位可以改写为“0”，而不影响其它位。例如，地址 0x08000400 存储的数据为 0x5a5a5a5a，使用位编程功能，可直接将此地址的数据写为 0x0a0a0a0a，而不需要先把该地址的数据擦除，然后写 0x0a0a0a0a。

请注意，位编程功能不能将值为“0”的 bit 位写“1”，如上面的例子，将 0x08000400 地址写为 0xfafafafa，将不会成功。

具体功能以及寄存器设置，请用户参考 GD32F30x 用户手册。

6. 版本历史

表 6-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2022 年 3 月 15 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.