GigaDevice Semiconductor Inc.

Migration from GD32F10x series to GD32F30x series

Application Note
AN011

Table of Contents

Table of Contents 2
List of Figures 3
List of Tables 4
1. Introduction 5
2. Pin compatibility 6
3. Internal Resource Compatibility 7
4. Program porting 9
4.1. System clock configuration 9
5. Peripheral differences 12
5.1. General-purpose and alternate-function I/Os (GPIO and AFIO) 12
5.2. Analog-to-digital converter (ADC) 12
5.3. Universal synchronous/asynchronous receiver/transmitter (USART) 13
5.4. Inter-integrated circuit interface (I2C) 13
5.5. Serial peripheral interface/Inter-IC sound (SPI/I2S) 13
5.6. Universal Serial Bus full-speed device interface (USBD) 13
5.7. Flash memory controller(FMC) 13
6. Revision history 15

List of Figures

Figure 4-1 Add macro definition in system_gd32f10x.c 9
Figure 4-2 The declaration of 120MHz function 10
Figure 4-3 120MHz function call 12

List of Tables

Table 2-1 Pin difference between GD32F10x series and GD32F30x series 6
Table 3-1 Overview of the resource comparison between GD32F10x series and GD32F30x series 7
Table 4-1. Ddfinition of system_clock_120m_hxtal function 10
Table 6-1. Revision history 15

1. Introduction

This application note is designed to help you quickly migrate your application from the GD32F10x series microcontrollers to the GD32F30x series microcontrollers.

In order to make better use of the information in this application note, you need to download the GD32 series microcontroller data from the official website www.GD32MCU.com, such as Datasheet, user manual, official routines and various development tools.

2. Pin compatibility

GD32F10x and GD32F30x are Pin To Pin compatible in the same package. However, compared with GD32F10x, GD32F30x adds an internal 48MHz RC oscillator to provide a fixed frequency for the USBD module. In order to meet the accuracy requirements, GD32F30x contains a clock trim controller (CTC), so the pin definitions of the two are slightly different, as shown in ***Table 2-1 Pin difference between GD32F10x series and GD32F30x series***:

Table 2-1 Pin difference between GD32F10x series and GD32F30x series

Pin name	Pin definition of GD32F10x series	Pin definition of GD32F30x series
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. Internal Resource Compatibility

The ***Table 3-1 Overview of the resource comparison between GD32F10x series and GD32F30x series*** gives an overview of the resource comparison between GD32F10x and GD32F30x (take the comparison of GD32F103xE and GD32F303xE as an example):

Table 3-1 Overview of the resource comparison between GD32F10x series and GD32F30x series

On-chip resources	GD32F103xE	GD32F303xE	Description of compatibility
Max frequency	108MHz	120MHz	Compatible
Core	Cortex®-M3 core	Cortex®-M4 core	M4 kernel is backward compatible
Flash	512K	512K	Fully compatible
RAM	64K	64K	Fully compatible
GPTM	4	4	Fully compatible
Advanced TM	2(RE/VE/ZE)	1(CE) 2(RE/VE/ZE)	Fully compatible
Basic TM	2	2	Fully compatible
Systick	1	1	Fully compatible
Watch dog	2	2	Fully compatible
RTC	1	1	Fully compatible
USART	3	3	Compatible F303: up to 7.5MHz(asynchronous) / 60MHz (synchronous) F103: up to 4.5MHz(asynchronous) / 54MHz (synchronous)
UART	2(RE/VE/ZE)	0(CE) 2(RE/VE/ZE)	Compatible F303: up to 7.5MHz F103: up to 4.5MHz
I2C	2	2	Compatible F303: up to 7. 1000KHz F103: up to 400KHz
SPI/IIS	3/2	3/2	Compatible F303: up to 30MHz F103: up to 18MHz
SDIO	1(RE/VE/ZE)	0(CE) 1(RE/VE/ZE)	Fully compatible
CAN	1	1	Fully compatible
USBD	1	1	Fully compatible
GPIO	51(RE)/80(VE)/112(ZE)	37(CE)/51(RE)/80(VE)/112(ZE)	Fully compatible
EXMC	0(RE)/1(VE/ZE)	0(CE/RE)/1(VE/ZE)	Fully compatible

ADC(CH)	3(16)(RE/VE) 3(21)(ZE)	3(16)(CE/RE/VE) 3(21)(ZE)	Fully compatible
DAC	2	2	Fully compatible
CTC	0	1	GD32F30x provide CTC to trim 48MHz RC oscillator
Package	LQFP64(RE) LQFP100(VE) LQFP144(ZE)	LQFP48(CE) LQFP64(RE) LQFP100(VE) LQFP144(ZE)	Fully compatible

4. Program porting

As can be seen from the previous section, the main differences between GD32F10x and GD32F30x are the main frequency (RCU system clock), kernel version and CTC, while the Cortex®-M4 is backward compatible with Cortex®-M4, so no modification is required. Then takes RCU as an example to illustrate the program porting process.

4.1. System clock configuration

The process for configuring clock for GD32F10x series and GD32F30x series is the same, but GD32F30x supports higher system clock. If the user continues to use the original clock frequency, the program does not need to be modified. If the user uses an higher clock frequency, follow the steps to modify the program (taking GD32F103 to GD32F303 and using external 8MHz high-speed crystal oscillator HXTAL as an example, the migration process of other corresponding models and using internal crystal oscillator is similar):

(1) Add macro definition in system_gd32f10x.c

#define __SYSTEM_CLOCK_120M_PLL_HXTAL (uint32_t)(120000000)

As shown in ***Figure 4-1 Add macro definition in system_gd32f10x.c***:

Figure 4-1 Add macro definition in system_gd32f10x.c

```
/* 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) Add the declaration of using 120MHz frequency function in system_gd32f10x.c, as shown in ***Figure 4-2 The declaration of 120MHz function***:

Figure 4-2 The declaration of 120MHz function

```
/* 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) Add the definition of using the 120MHz frequency function in the system_gd32f10x.c file:

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) Add a call to use the 120MHz frequency function in the system_gd32f10x.c file, as shown in ***Figure 4-3 120MHz function call***:

Figure 4-3 120MHz function call

```
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. Peripheral differences

The peripherals of GD32F10x and GD32F30x are compatible, but GD32F30x, as a more advanced MCU, has some functions added to many peripherals compared with GD32F10x. Users can choose whether to use these functions according to the differences of peripherals listed below.

5.1. General-purpose and alternate-function I/Os (GPIO and AFIO)

When the I/O port is used as output, the GD32F30x can set the IO speed to 120MHz (the maximum speed of the GD32F10x is 50MHz), When the output speed of the I/O port is greater than 50MHz, It is recommended to use the I/O compensation unit to control the slope of the I/O port to reduce the impact of I/O port noise on the power supply. For specific functions and register settings, please refer to the GD32F30x user manual.

5.2. Analog-to-digital converter (ADC)

The GD32F30x adds an on-chip hardware oversampling circuit compared with the GD32F10x to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16 bit. On-chip hardware oversampling circuit at the expense of lower data output rate in exchange for higher data resolution. For specific functions and register settings, please refer to the GD32F30x user manual.

5.3. Universal synchronous/asynchronous receiver/transmitter (USART)

Compared with GD32F10x, the GD32F30x adds block mode (GD32F10x supports character mode only), Data polarity setting and TX, RX pin level inversion, etc. Therefore, GD32F30x adds three registers: USART_CTL3, USART_RT and USART_STAT1. For specific functions and register settings, please refer to the GD32F30x user manual.

5.4. Inter-integrated circuit interface (I2C)

The I2C of GD32F30x and GD32F10x both support standard-mode (up to 100KHz) and fast-mode (up to 400KHz), while GD32F30x can support fast-mode-plus (up to 1MHz). Set FMPEN bit in the I2C_FMPCFG register to enable fast-mode-plus. For specific functions and register settings, please refer to the GD32F30x user manual.

5.5. Serial peripheral interface/Inter-IC sound (SPI/I2S)

The main difference between the SPI/I2S modules of GD32F30x and GD32F10x is that GD32F30x supports SPI TI mode, SPI NSS pulse mode and quad-SPI function (only SPI0), which the quad-SPI mode is used to control the four-wire SPI Flash peripheral. For specific functions and register settings, please refer to the GD32F30x user manual.

5.6. Universal Serial Bus full-speed device interface (USBD)

Compared with the GD32F10x, the GD32F30x USB 2.0 has achieved Link Power Management (LPM) level L1 in order to optimize power consumption in SUSPEND/RESUME state. LPM includes 4 states from L0 to L3. LPM L1 state (sleep state) is the new power management state. For specific functions and register settings, please refer to the GD32F30x user manual.

5.7. Flash memory controller(FMC)

Compared with GD32F10x, GD32F30x support bit programming, which saves some Flash space for users. Its characteristic is that the data stored in the flash memory, the bit whose value is "1" can be rewritten to "0" without affecting other bits. For example, the data stored at the address 0x0800 0400 is 0x5a5a5a5a. Using the bit programming function, the data at this address can be directly written as 0x0a0a0a0a without erasing the data at first and then writing 0x0a0a 0a0a.

Please note that the bit programming function cannot write the bit with value "0" as "1", as in

the above example, writing the address of 0x08000400 as 0xfafafafa will be failed.

For specific functions and register settings, please refer to the GD32F30x user manual.

6. Revision history

Table 6-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Mar.15, 2022

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.