GigaDevice Semiconductor Inc.

GD32W51x TrustZone Development Guide

Application Note
AN103
# Table of Contents

Table of Contents .......................................................................................................................... 2  
List of Figures ............................................................................................................................... 3  
List of Tables ................................................................................................................................ 4  
1. Introduction ............................................................................................................................... 5  
2. TrustZone introduction ............................................................................................................ 6  
3. Software development ............................................................................................................... 8  
   3.1. Developing TrustZone in Keil ............................................................................................. 8  
      3.1.1. Secure Project .............................................................................................................. 8  
      3.1.2. Non-secure Project .................................................................................................... 11  
      3.1.3. Compile Project ......................................................................................................... 13  
      3.1.4. Download Project ..................................................................................................... 14  
   3.2. Developing TrustZone in IAR ............................................................................................ 14  
      3.2.1. Secure Project ............................................................................................................ 15  
      3.2.2. Non-secure Project .................................................................................................. 17  
      3.2.3. Compile Project ......................................................................................................... 20  
      3.2.4. Download Project ..................................................................................................... 20  
4. Code interpretation ..................................................................................................................... 22  
   4.1. Secure Project ..................................................................................................................... 22  
   4.2. Non-secure project ............................................................................................................. 24  
5. Revision history .......................................................................................................................... 26
List of Figures

Figure 2-1. Example of memory map security attribution vs SAU configuration regions ........ 6
Figure 2-2. Enable TrustZone using the GD-Link Programmer .......................................... 7
Figure 3-1. Keil project file structure ................................................................................. 8
Figure 3-2. Select secure project ......................................................................................... 8
Figure 3-3. Select Secure mode ............................................................................................ 9
Figure 3-4. Select scattered load file and NSC output library ............................................. 9
Figure 3-5. Debugger setup in secure project ....................................................................... 10
Figure 3-6. Download algorithm and function in secure project ......................................... 11
Figure 3-7. Select non-secure project .................................................................................. 11
Figure 3-8. Select non-secure mode ...................................................................................... 11
Figure 3-9. Select scattered load file and import NSC library ............................................. 12
Figure 3-10. Debugger setup in non-secure project .......................................................... 13
Figure 3-11. Download algorithm and function in non-secure project .............................. 13
Figure 3-12. Compile Project ............................................................................................ 14
Figure 3-13. Boot option ..................................................................................................... 14
Figure 3-14. Download project .......................................................................................... 14
Figure 3-15. IAR project file structure ................................................................................. 15
Figure 3-16. Select secure project ....................................................................................... 15
Figure 3-17. Select Secure mode ......................................................................................... 16
Figure 3-18. Select scattered load file and NSC output library ........................................... 16
Figure 3-19. Debugger setup in secure project .................................................................... 17
Figure 3-20. Select non-secure project ................................................................................. 18
Figure 3-21. Select non-secure mode ................................................................................... 18
Figure 3-22. Select scattered load file and import NSC library .......................................... 19
Figure 3-23. Debugger setup in non-secure project .......................................................... 20
Figure 3-24. Compile project ............................................................................................. 20
Figure 3-25. Boot option ..................................................................................................... 21
Figure 3-26. Download project ........................................................................................... 21
Figure 4-1. SAU configuration ............................................................................................ 22
List of Tables

Table 3-1. Project_S.sct code...........................................................................................................................................10
Table 3-2. Project_NS.sct code .......................................................................................................................................12
Table 3-3. gd32w51x_flash_s.icf code ..........................................................................................................................16
Table 3-4. gd32w51x_flash_ns.icf code..........................................................................................................................19
Table 4-1. main code in secure project..........................................................................................................................22
Table 4-2. FMC configuration.........................................................................................................................................23
Table 4-3. SRAM configuration.......................................................................................................................................23
Table 4-4. Call non-secure function in secure code .......................................................................................................24
Table 4-5. Call secure function in non-secure code ......................................................................................................25
Table 5-1. Revision history ..............................................................................................................................................26
1. **Introduction**

This article describes developing the TrustZone program on the GD32W51x series. The TrustZone security attribute is the main ARMv8-M extension, which provides hardware security separation and protection. The sample code is also divided into two parts, security and non-security. The development process and key points for Keil and IAR are described.

In this article, the GD32W515P-EVAL development board was used, the chip model was GD32W515PIQ6, and the Cortex-M33 kernel with TrustZone was used, with max frequency up to 180MHz. The TrustZone security is activated by the TZEN option bit in the EFUSE_TZCTL register or the TZEN option bit in the option byte. The sample code is provided in 'GD32W515P_EVAL_Demo_Suites / 23_Trustzone’ and is provided by the software package in [GD32MCU.COM](http://GD32MCU.COM).

Applicable product: GD32W51x series
2. **TrustZone introduction**

The M33 kernel with TrustZone has both secure and non-secure states, which use a separate set of kernel registers and Systick. Memory is classified into secure (S), non-secure callable (NSC), and non-secure (NS) domain. A secure domain can access a non-secure domain, and a non-secure can only access a non-secure callable (in secure code), providing hardware protection for access. When TrustZone is enabled, SAU (security attribution unit) and IDAU (implementation defined attribution unit) are used together to set the security attribute of the memory address. The final security level is the attribute of higher security level defined in SAU and IDAU (S>NSC>NS). See Figure 2-1. Example of memory map security attribution vs SAU configuration regions.

IDAU provides a hardware partition of non-secure (NS) and non-secure Callable (NSC) security attributes, which cannot be changed by software, and memory mapped partitions refer to Section 1.4 Memory Mapping in the GD32W51x_User manual.

SAU can be software-configured with up to 8 security attribute zones, which can be made secure (S), non-secure (NS), or non-secure callable (NSC).

**Figure 2-1. Example of memory map security attribution vs SAU configuration regions**

![Diagram of memory map security attribution vs SAU configuration regions](image)

GD32W51x uses TrustZone protection controller union (TZPCU) to manage the security attributes of peripherals, SRAM, and flash memory. TZPCU provides unauthorized access control, working with the kernel to achieve full features of TrustZone.

Ensure that the TZEN bit in the EFUSE_TZCTL register or the TZEN bit in the option byte is enabled before using TrustZone or TZPCU. Using GD-Link Programmer to enable TZEN bit in option bytes, as shown in Figure 2-2, Enable TrustZone using the GD-Link Programmer. The GD-Link Programmer can be obtained from [GD32MCU.COM](http://GD32MCU.COM).
Figure 2-2. Enable TrustZone using the GD-Link Programmer
3. Software development

This article uses two development environments, MDK-ARM v5.28.0.0 and IAR v8.50.9. Please ensure that the corresponding software and device package are correctly installed. The software can be obtained from the official website of KEIL and IAR. The device package (GD32W51x_AddOn) can be obtained from GD32MCU.COM.

The sample project has secure and non-secure projects that use different Flash and SRAM. The secure project uses the first 256KB in the FMC (starting address 0x0C000000, size 0x40000), and the memory uses SRAM0 blocks (starting address 0x30000000, size 0x10000). The non-secure project uses the last 1792KB in the FMC (starting address 0x08040000, size 0x1C0000) and the memory uses SRAM1 block (starting address 0x20010000, size 0x10000).

3.1. Developing TrustZone in Keil

Multiple project can be used to develop two projects at the same time, as shown in Figure 3-1. Keil project file structure, project path:

GD32W51x_Demo_Suites\GD32W515P_EVAL_Demo_Suites\Projects\23_Trustzone\MDK-ARM\GD32W515P_EVAL_TRUSTZONE.uvmpw

Figure 3-1. Keil project file structure

3.1.1. Secure Project

1. Select secure project as current project, as shown in Figure 3-2. Select secure project.

Figure 3-2. Select secure project
2. In Project/Options for Target/Target, select Code Generation/Software Model as Secure Mode, as shown in Figure 3-3, Select Secure mode.

Figure 3-3. Select Secure mode

3. Secure code and non-secure code use different Flash and SRAM, and use a scattered load file to assign the Flash and SRAM for secure code. In Project/Options for Target/Linker, select not to assign addresses using the Target interface, select scatter load file, and set the output library for NSC functions. See Figure 3-4, Select scattered load file and NSC output library.

Figure 3-4. Select scattered load file and NSC output library

The output command is --import_cmse_lib_out=\secure_nsclib.o. The compiler will compile the code at compile time with __attribute__((cmse_nonsecure_entry)) identifying code into .secure_nsclib.o. Non-secure code can access function of secure_nsclib.o.

4. The secure project uses Flash address 0x0C000000 with size 0x40000 and SRAM address 0x30000000 with size 0x10000. NSC function is assigned to address 0x0C03E000 and size is 0x00002000. Project_S.sct code is shown in Table 3-1, Project_S.sct code.
Table 3-1. Project_S.sct code

| LR_IROM1 0x0C000000 0x00040000 { | ; load region size_region |
| ER_IROM1 0x0C000000 0x0003E000 { | ; load address = execution address |
| *.o (RESET, +First) | |
| *(InRoot$S$Sections) | |
| .ANY (+RO) | |
| .ANY (+XO) | |
| } | |
| RW_IRAM2 0x30000000 0x00010000 { | ; RW data |
| .ANY (+RW +ZI) | |
| } | |
| LR_IROM2 0x0C03E000 0x00002000 { | |
| ER_IROM2 0x0C03E000 0x00002000 { | ; load address = execution address |
| *(Veneer$CMSE) | ; check with partition.h |
| } | |

5. Use GD-Link for debugging and downloading. Connect GD-Link interface of Eval-board directly to PC-USB. Select CMSIS-DAP ARMv8-M Debugger under Project/Options for Target/Debug. Identify the debugger ID in Settings, use the SW interface and set the maximum clock to 1MHz. See Figure 3-5. Debugger setup in secure project.

Figure 3-5. Debugger setup in secure project

6. Under Flash Download TAB, select Erase Sectors and uncheck Reset and Run. Add programming algorithm GD32W515_Secure_Flash. GD32W515_Secure_Flash is provided by the installed device package. See Figure 3-6. Download algorithm and function in secure project.
3.1.2. Non-secure Project

1. Select non-secure project as current project, as shown in Figure 3-7. Select non-secure project.

2. In Project/Options for Target/Target TAB, select Code Generation/Software Model as Non-secure Mode, as shown in Figure 3-8. Select non-secure mode.

3. Secure code and non-secure code use different Flash and SRAM, and use a distributed load file to assign the Flash and SRAM for secure code. In Project/Options for Target/Linker,
select not to assign addresses using the Target interface, select the scattered load file, and use the NSC function library exported by the Security Project. See **Figure 3-9. Select scattered load file and import NSC library**.

Library import command `./Secure_Project/secure_nsclib.o`, non-secure project can directly call the `__attribute__((cmse_nonsecure_entry))` identification code in the security project.

**Figure 3-9. Select scattered load file and import NSC library**

4. Non-secure code uses Flash address 0x08040000 with size 0x001C0000 and SRAM address 0x20010000 with size 0x10000. Project_NS.sct code is shown in **Table 3-2 Project_NS.sct code**.

| LR_IROM1 | 0x08040000 0x001C0000 | ; load region size_region |
| ER_IROM1 | 0x080400000 0x001C0000 | ; load address = execution address |
| * .o (RESET, +First) | |
| *(InRoot$$Sections) | |
| .ANY (+RO) | |
| .ANY (+XO) | |
| } | |
| RW_IRAM1 | 0x20010000 0x00010000 | ; RW data |
| .ANY (+RW +ZI) | |
| } | |

5. Use GD-Link for debugging and downloading. Connect GD-Link interface of Eval-board directly to PC-USB. Select CMSIS-DAP ARMv8-M Debugger under Project/Options for Target/Debug. Identify the debugger ID in Settings, use the SW interface and set the maximum clock to 1MHz. See **Figure 3-10. Debugger setup in non-secure project**.
6. Under Flash Download TAB, select Erase Sectors and uncheck Reset and Run. Add programming algorithm GD32W515_Nsecure_Flash. GD32W515_Nsecure_Flash is provided by the installed device package. See Figure 3-11. Download algorithm and function in non-secure project.

3.1.3. Compile Project

Firstly compile secure project and then non-secure project. Non-secure project need secure_nsclib.o library generated in secure project. Compile the secure and insecure projects in order.

Batch setup can also compile two projects in order, as shown in Figure 3-12. Compile Project. Select two projects and click Build. The order is determined by project directory sequence, which can be modified by project/Manage/Multi-Project Workspace.
3.1.4. Download Project

Set up Eval-board correctly before downloading code. JP4 (BOOT0/BOOT1) connects to L. MCU will boot from the secure code. JP3 connects to SWD. See Figure 3-13. Boot option. JP21 connects to USART for printf. Connect Eval-board GD-Link and USART to PC, and ensure that software drivers are correctly installed.

![Figure 3-13. Boot option](image)

Firstly download non-secure code and then secure code. MCU always boots from secure code and jumps to non-secure code. See Figure 3-14. Download project.

![Figure 3-14. Download project](image)

Reset and Run are not checked in the Flash Download option. Press manually Reset button to restart MCU. Two messages will be printed and two LED lights flash respectively.

3.2. Developing TrustZone in IAR

Two projects can be developed at the same time using multiple project methods. The file structure is shown in Figure 3-15. IAR project file structure, project path:

GD32W51x_Demo_Suites\GD32W515P_EVAL_Demo_Suites\Projects\23_Trustzone\EWA RM\GD32W515P_EVAL_TRUSTZONE.eww
3.2.1. Secure Project

1. Select secure project as active project, as shown in Figure 3-16. Select secure project.

2. In Project / Options / General Options / Target, select TrustZone / Mode as Secure, as shown in Figure 3-17. Select Secure mode.
Figure 3-17. Select Secure mode

3. Secure code and non-secure code use different Flash and SRAM, and use a scattered load file to assign the Flash and SRAM for secure code. In Project / Options / Linker / Config, select scatter load file and set the output library for NSC functions. See Figure 3-18. Select scattered load file and NSC output library.

The output command is --import_cmse_lib_out=\secure_nsclib.o. The compiler will compile the code at compile time with __attribute__((cmse_nonsecure_entry)) identifying code into.\secure_nsclib.o. Non-secure code can access functions of secure_nsclib.o.

Figure 3-18. Select scattered load file and NSC output library

4. The secure project uses Flash address 0x0C000000 with size 0x40000 and SRAM address 0x30000000 with size 0x10000. NSC function is assigned to address 0x0C03E000 and size is 0x00002000. gd32w51x_flash_s.icf code is shown in Table 3-3. gd32w51x_flash_s.icf code.

Table 3-3. gd32w51x_flash_s.icf code

/* Specials */
```c
#define symbol __ICFEDIT_intvec_start__ = 0x0C000000;
/
-Memory Regions-/\n#define symbol __ICFEDIT_region_ROM_start__ = 0x0C000000;
#define symbol __ICFEDIT_region_ROM_end__ = 0x0C03FFF;
#define symbol __ICFEDIT_region_RAM_start__ = 0x30000000;
#define symbol __ICFEDIT_region_RAM_end__ = 0x3000FFF;
.....
#define symbol __region_ROM_NSC_start__ = 0x0C03E000;
#define symbol __region_ROM_NSC_end__ = 0x0C03FFFF;
.
.....
define region ROM_region = mem:[from __ICFEDIT_region_ROM_start__ to __ICFEDIT_region_ROM_end__];
define region ROM_NSC_region = mem:[from __region_ROM_NSC_start__ to __region_ROM_NSC_end__];
define region RAM_region = mem:[from __ICFEDIT_region_RAM_start__ to __ICFEDIT_region_RAM_end__];
.....
place in ROM_region { readonly };
place in ROM_NSC_region { section Veneer$$CMSE };
place in RAM_region { readwrite, block HEAP, block CSTACK };```

5. Use GD-Link for debugging and downloading. Select CMSIS-DAP under Project / Options / Debugger. See **Figure 3-19. Debugger setup in secure project**.

**Figure 3-19. Debugger setup in secure project**

![](image)

### 3.2.2. Non-secure Project

1. Select non-secure project as active project, as shown in **Figure 3-20. Select non-secure project**.
2. In Project / Options / General Options / Target, select TrustZone / Mode as Non-secure, as shown in **Figure 3-21. Select non-secure mode**.

3. Secure code and non-secure code use different Flash and SRAM, and use a scattered load file to assign the Flash and SRAM for secure code. In Project / Options / Linker / Config, select scatter load file and use the NSC function library exported by the Security Project. See **Figure 3-22. Select scattered load file and import NSC library**.

   Library import command `$PROJ_DIR$\..\Secure_Project\GD32W51x\Exe\secure_nsclib.o`, non-secure project can directly call the `__attribute__((cmse_nonsecure_entry))` identification code in the security project.
4. Non-secure code uses Flash address 0x08040000 with size 0x001C0000 and SRAM address 0x20010000 with size 0x10000. Project_NS.sct code is shown in Table 3-4. 

Table 3-4. gd32w51x_flash_ns.icf code

```c
/*-Specials-* /
define symbol __ICFEDIT_intvec_start__ = 0x08040000;

/*-Memory Regions-* /
define symbol __ICFEDIT_region_ROM_start__ = 0x08040000;
define symbol __ICFEDIT_region_ROM_end__ = 0x081FFF;
define symbol __ICFEDIT_region_RAM_start__ = 0x20010000;
define symbol __ICFEDIT_region_RAM_end__ = 0x2001FFFF;

......
define region ROM_region = mem:[from __ICFEDIT_region_ROM_start__ to __ICFEDIT_region_ROM_end__];
define region RAM_region = mem:[from __ICFEDIT_region_RAM_start__ to __ICFEDIT_region_RAM_end__];

......

place in ROM_region (readonly);
place in RAM_region (readwrite, block HEAP, block CSTACK);
```

5. Use GD-Link for debugging and downloading. Select CMSIS-DAP under Project / Options / Debugger. See Figure 3-23. Debugger setup in non-secure project.
3.2.3. Compile Project

Firstly compile secure project and then non-secure project. Non-secure project need secure_nsclib.o library generated in secure project. Compile the secure and insecure projects in order.

Batch setup can also compile two projects in order, as shown in Figure 3-24. Compile project. The order is determined by the order of addition.

3.2.4. Download Project

Set up Eval-board correctly before downloading code. JP4 (BOOT0/BOOT1) connects to L. MCU will boot from the secure code. JP3 connects to SWD. See Figure 3-25. Boot option. JP21 connects to USART for printf. Connect Eval-board GD-Link and USART to PC, and ensure that software drivers are correctly installed.
Firstly download non-secure code and then secure code. MCU always boots from secure code and jumps to non-secure code. See Figure 3-26, Download project.

Figure 3-26. Download project

Reset and Run are not checked in the Flash Download option. Press manually Reset button to restart MCU. Two messages will be printed and two LED lights flash respectively.
4. Code interpretation

4.1. Secure Project

If Code Generation/Software Model is set to Secure Mode in the Project/Options for Target/Target TAB, __ARM_FEATURE_CMSE is defined as 3. After power-on and reset, MCU will execute system_gd32w51x.c/SystemInit()//sau_region_config() function to configure the SAU based on the partition_gd32w51x..h content. The partition_gd32w51x..h file can be configured using the Configuration Wizard interface and generate Text Editor code. As shown in Figure 4-1. SAU configuration. SAU is enabled and four regions are configured. Region0 configures NSC address, corresponding to Project_S.sct. Region1 configures non-secure code address. Region2 configures non-secure SRAM address, corresponding to Project_NS.sct. Region3 configures peripheral address to be non-secure. Peripheral can be accessed in both secure or non-secure code.

Figure 4-1. SAU configuration

In main function of secure project, secure interrupt is enabled, Systick configuration, FMC configuration, TZBMPA configuration, LED configuration, USART configuration, and finally jump to non-secure code. See Table 4-1. main code in secure project.

Table 4-1. main code in secure project

```c
int main(void)
{
    /* enable SecureFault handler */
    SCB->SHCSR |= SCB_SHCSR_SECUREFAULTENA_Msk;
}
```
/* configure systick */
systick_config();

/* configure mark secure pages */
if(SECM_SPAGE <= (FMC_SECMCFG0 & 0x3FF) || SECM_EPAGE <= ((FMC_SECMCFG0 >> 16) & 0x3FF)) {
    fmc_secmark_config();
}
if(0U == (FMC_OBR & FMC_OBR_TZEN)) {
    /* enable trustzone */
    fmc_trustzone_enable();
}

/* configure TZBMPC */
tzbmpc_config();
led_config();
com_config();

/* setup and jump to non-secure */
nonsecure_init();
while(1) {
}

Function fmc_secmark_config() configures pages 0-63 of FMC. First 256KB is used for secure code.

Table 4-2. FMC configuration

```c
void fmc_secmark_config(void) {
    fmc_unlock();
    ob_unlock();
    /* configure mark secure pages */
    ob_secmark_config(SECM_SPAGE, SECM_EPAGE, SECM_INDEX0);
    ob_start();
    while(0U != (FMC_SECSTAT & (FMC_SECSTAT_SECBUSY))) {
    }
    ob_reload();
    ob_lock();
    fmc_lock();
}
```

Function tzbmpc_config() configures SRAM1 as non-secure. This area is used for non-secure code.

Table 4-3. SRAM configuration

```c
void tzbmpc_config(void) {
```
uint16_t block_number = 0U;

/* enable TZPCU clock */
rcu_periph_clock_enable(RCU_TZPCU);

/* SRAM1 is used to nonsecure code, so all blocks of SRAM1 should set to nonsecure */
for(block_number = 0U; block_number <= TZBMPC1_BLOCK_NUMBER; block_number++){

tzpcu_tzbmpc_block_secure_access_mode_config(TZBMPC1, block_number,
BLOCK_SECURE_ACCESS_MODE_NSEC);
}
}

Function led_config() configures LED1 and LED2 ports as non-secure. LED1 and LED2 will be controlled by non-secure code. Function com_config() configures USART2 (serial port) as secure. USART2 will be controlled by secure code.

e
try _cb_func_register() and non_secure_print() are non-secure callable functions marked with __attribute__((cmse_nonsecure_entry)). Non-secure code can call these functions directly to pass non-secure function addresses and information to print.

If secure code needs to call non-secure function, use cmse_nsfptr_create(func_addr) to register the function address and __attribute__((cmse_nonsecure_call)) to convert the function to non-secure. Finally use the function. In this demo, non-secure code first calls the entry_cb_func_register() to get address of toggle_led1, and then calls nonsecure_func() to toggle LED1. The code is shown in Table 4.4. Call non-secure function in secure code.

<table>
<thead>
<tr>
<th>Table 4.4. Call non-secure function in secure code</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define CMSE_NS_ENTRY <strong>attribute</strong>((cmse_nonsecure_entry))</td>
</tr>
<tr>
<td>#define CMSE_NS_CALL <strong>attribute</strong>((cmse_nonsecure_call))</td>
</tr>
<tr>
<td>typedef void CMSE_NS_CALL (*ns_fptr)(void);</td>
</tr>
<tr>
<td>ns_fptr nonsecure_func = (ns_fptr)NULL;</td>
</tr>
<tr>
<td>........</td>
</tr>
<tr>
<td>CMSE_NS_ENTRY void entry_cb_func_register(void *callback)</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>if(callback != NULL)</td>
</tr>
<tr>
<td>nonsecure_func = (ns_fptr)cmse_nsfptr_create(callback);</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

4.2. Non-secure project

NSC functions can be used in non-secure code after imported secure_nsclib.o library. In main() function, entry_cb_func_register() passes address of toggle_led1 to security code. Then configure Systick of non-secure domain, initialize LED port. The while loop periodically flips LED2 and calls the NSC function non_secure_print to print the information.
Table 4-5. Call secure function in non-secure code

```c
extern void entry_cb_func_register(void *callback);
eextern void non_secure_print(const char * str);
int main(void)
{
    entry_cb_func_register((void *)toggle_led1);
    /* configure systick */
    systick_config();
    gd_eval_led_init(LED1);
    gd_eval_led_init(LED2);
    while(1){
        /* toggle LED2 */
        gd_eval_led_toggle(LED2);
        non_secure_print("non-secure code toggle LED2:\r\n");
        delay_1ms(1000);
    }
}
```
5. Revision history

Table 5-1. Revision history

<table>
<thead>
<tr>
<th>Revision No.</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Initial Release</td>
<td>Mar.3 2023</td>
</tr>
</tbody>
</table>
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 security 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 its 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 its 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.

© 2023 GigaDevice – All rights reserved