GigaDevice Semiconductor Inc.

GD32 TSI TouchKey Software Library User Guide

Application Note
AN089

Revision 1.0

(May. 2023)

Table of Contents

List of Figures

List of Tables

1. Introduction

Touch Sensing Interface (TSI) provides a convenient solution for touch keys, sliders and capacitive proximity sensing applications. The controller is based on charge transfer method of self-capacitance. Placing a finger near the electrode adds capacitance to the system and TSI is able to measure this capacitance change using charge transfer method.

This application note introduces the design principle and usage of TouchKey software library, which can help developers quickly use TouchKey software library for configuration and development.

2. Terminology

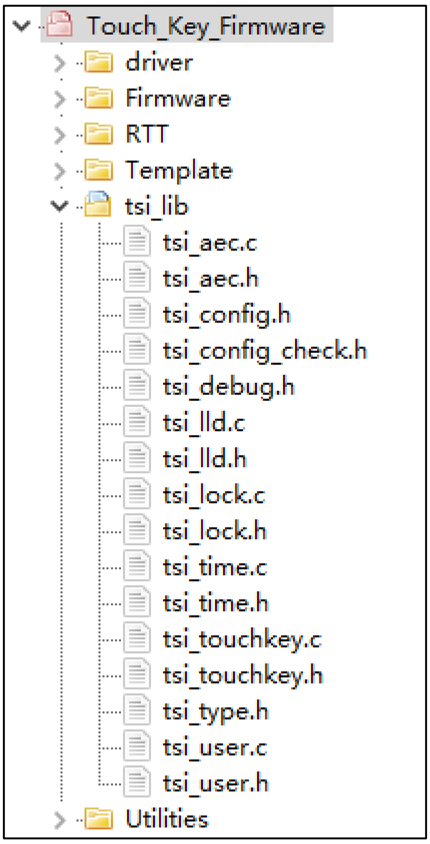
- Adaptive environment change detection (AEC).
- Detection timeout (DTO).
- Channel lock (LOCK).
- Noise filter (FILTER).
- State machine (SM).
- Timing management (TM).
- Log output (LOG).

3. TouchKey library

3.1. TouchKey library file structure

TouchKey library file structure refer to ***Figure 3-1. TouchKey software library file structure***. tsi_lib contains the main file of TouchKey library.

Figure 3-1. TouchKey software library file structure



```
Touch_Key_Firmware
    driver
    Firmware
    RTT
    Template
    tsi_lib
        tsi_aec.c
        tsi_aec.h
        tsi_config.h
        tsi_config_check.h
        tsi_debug.h
        tsi_lld.c
        tsi_lld.h
        tsi_lock.c
        tsi_lock.h
        tsi_time.c
        tsi_time.h
        tsi_touchkey.c
        tsi_touchkey.h
        tsi_type.h
        tsi_user.c
        tsi_user.h
    Utilities
```

tsi_aec.c / tsi_aec.h contains adaptive environment change detection code, which mainly modifys the channel reference value periodically.

tsi_lock.c / tsi_lock.h contains the channel priority configuration and lock / unlock function.

tsi_time.c / tsi_time.h contains the timing management function of TouchKey library.

tsi_touchkey.c / tsi_touchkey.h contains the TouchKey configuration and detection processing state machine function.

tsi_user.c / tsi_user.h contains the initialization / processing function of TouchKey.

tsi_lld.c / tsi_lld.h contains the TSI low level hardware initialization and bank configuration / processing function.

tsi_config.h contains the TSI parameter configuration and TouchKey parameter function macro definition.

tsi_type.h / tsi_debug.h / tsi_config_check.h contains the structure variables type declaration, debug interface macro definition and configuration parameter check.

3.2. TouchKey library architecture

TouchKey software library is divided into three layers:

- Acquisition layer
- Data layer
- Application layer

TouchKey library architecture refer to ***Figure 3-2. TouchKey softerware library architecture***.

Acquisition layer:

The acquisition layer implement the data acquisition of each sensor channel and transmits the collected original data to the data layer as input.

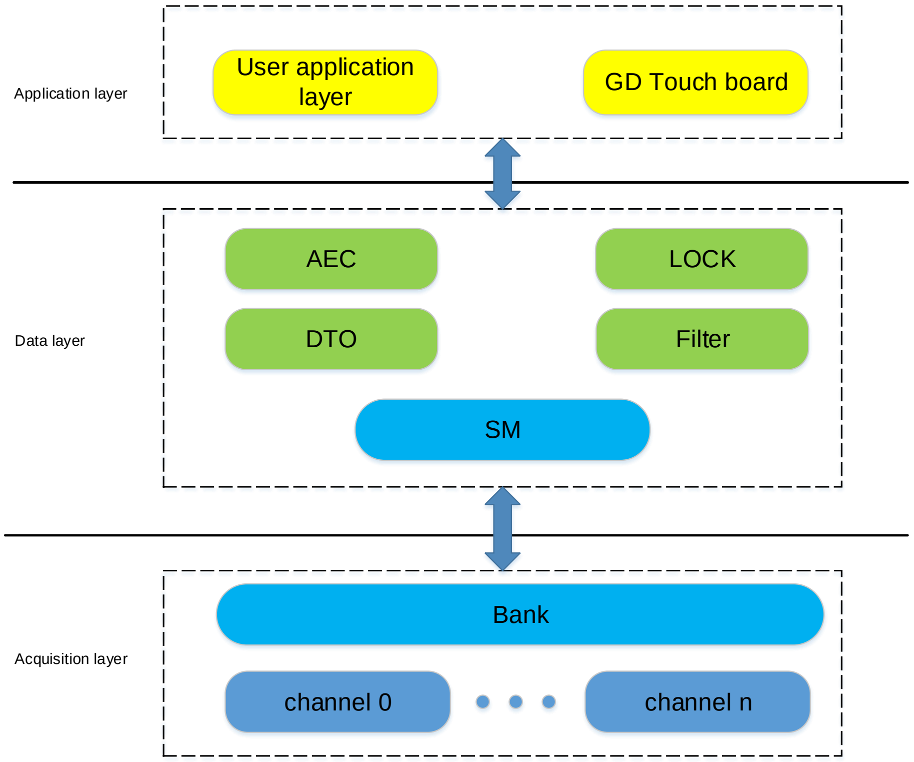
Data layer:

The data layer is to further process the original data of the acquisition layer (including AEC, LOCK, DTO, Filter) and obtain the touch state of each sensor channel as the input of the application layer.

Application layer:

The application layer is the specific logic code made by the user according to the touch state of each sensor channel obtained by the data layer, so as to meet the specific touch application scenarios.

Figure 3-2. TouchKey softerware library architecture

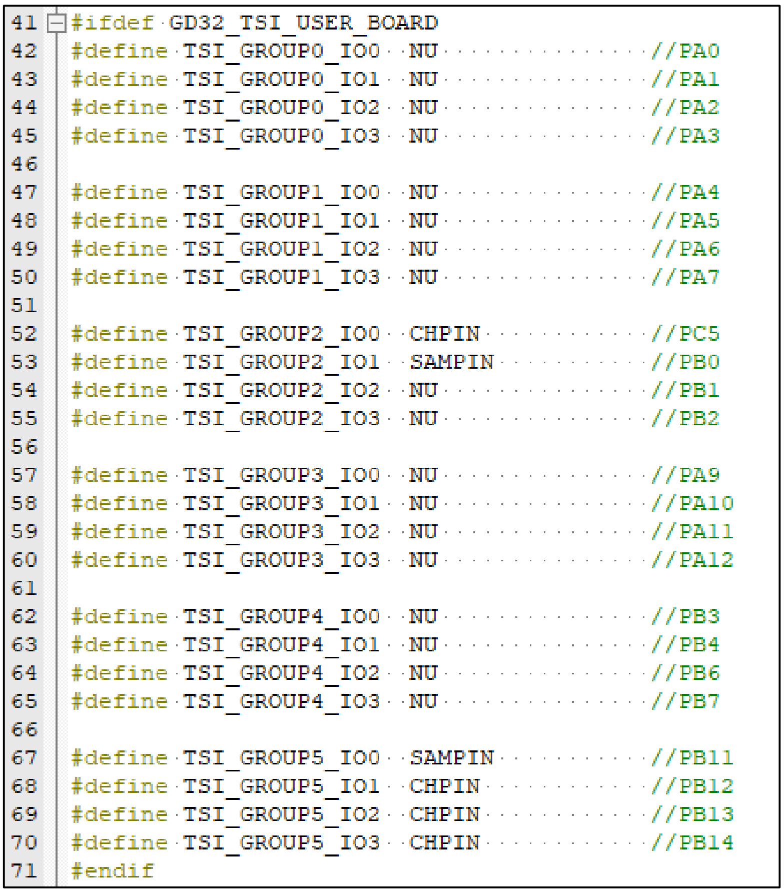


4. TouchKey library configuration

4.1. TSI pin configuration

User need to configure the TSI I/O port mode according to the TSI pins used in the application (can be configured in tsi_user.h). The user can configure it as sampling pin (SAMPIN), channel pin (CHPIN), shield pin (SHPIN) or not used pin(NU) according to the actual pin pattern, refer to ***Figure 4-1. TSI pin configuration***.

Figure 4-1. TSI pin configuration

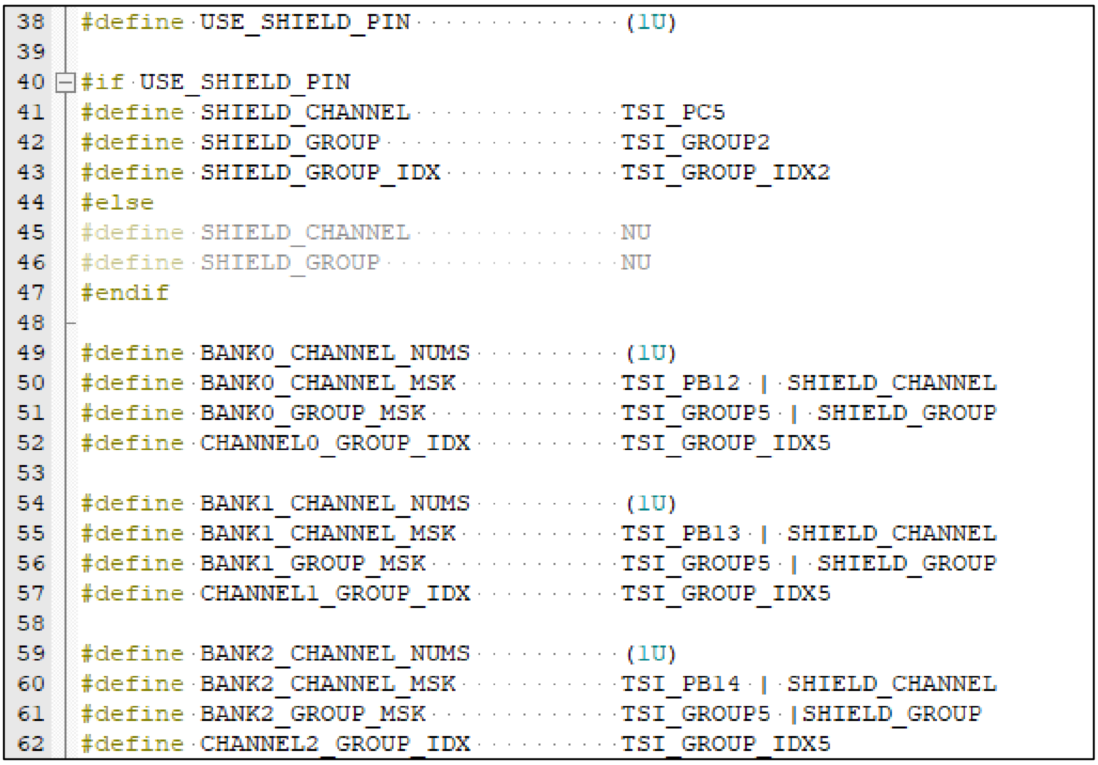


```
#ifdef GD32_TSI_USER_BOARD
#define TSI_GROUP0_IO0  NU                //PA0
#define TSI_GROUP0_IO1  NU                //PA1
#define TSI_GROUP0_IO2  NU                //PA2
#define TSI_GROUP0_IO3  NU                //PA3

#define TSI_GROUP1_IO0  NU                //PA4
#define TSI_GROUP1_IO1  NU                //PA5
#define TSI_GROUP1_IO2  NU                //PA6
#define TSI_GROUP1_IO3  NU                //PA7

#define TSI_GROUP2_IO0  CHPIN             //PC5
#define TSI_GROUP2_IO1  SAMPIN            //PB0
#define TSI_GROUP2_IO2  NU                //PB1
#define TSI_GROUP2_IO3  NU                //PB2

#define TSI_GROUP3_IO0  NU                //PA9
#define TSI_GROUP3_IO1  NU                //PA10
#define TSI_GROUP3_IO2  NU                //PA11
#define TSI_GROUP3_IO3  NU                //PA12

#define TSI_GROUP4_IO0  NU                //PB3
#define TSI_GROUP4_IO1  NU                //PB4
#define TSI_GROUP4_IO2  NU                //PB6
#define TSI_GROUP4_IO3  NU                //PB7

#define TSI_GROUP5_IO0  SAMPIN            //PB11
#define TSI_GROUP5_IO1  CHPIN             //PB12
#define TSI_GROUP5_IO2  CHPIN             //PB13
#define TSI_GROUP5_IO3  CHPIN             //PB14
#endif
```

4.2. BANK configuration

According to the allocation of TSI pins, user can compose pins of different groups into a BANK. The pins in BANK can realize parallel sampling, so as to improve the running efficiency of Touch library. (can be configured in tsi_user.c). ***Figure 4-2. BANK configuration*** shows how PB12 (group 5) forms BANK0, PB13 (group 5) forms BANK1, and PB14 (group 5) forms BANK2. BANKx_CHANNEL_NUMS indicates the number of channel pins in the BANK, BANKx_CHANNEL_MSK indicates the TSI channel pin used in the BANK, BANKx_GROUP_MSK indicates the TSI group used in the BANK, CHANNELx_GROUP_IDX indicates the group index of each channel in the BANK. In addition, the USE_SHIELD_PIN macro indicates whether the active shield function is enabled.

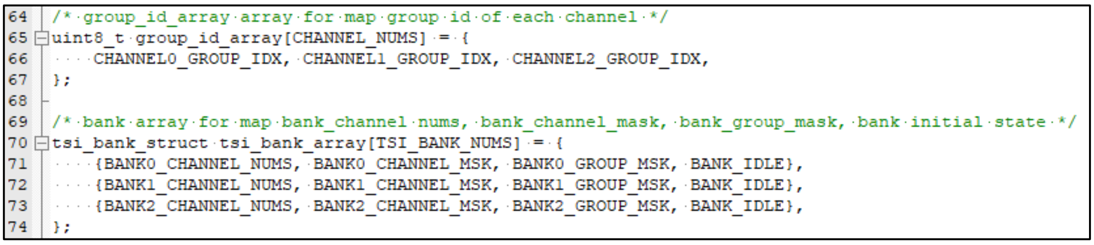
Figure 4-2. BANK configuration



```
#define USE_SHIELD_PIN              (1U)

#if USE_SHIELD_PIN
#define SHIELD_CHANNEL              TSI_PC5
#define SHIELD_GROUP                TSI_GROUP2
#define SHIELD_GROUP_IDX            TSI_GROUP_IDX2
#else
#define SHIELD_CHANNEL              NU
#define SHIELD_GROUP                NU
#endif

#define BANK0_CHANNEL_NUMS          (1U)
#define BANK0_CHANNEL_MSK           TSI_PB12 | SHIELD_CHANNEL
#define BANK0_GROUP_MSK             TSI_GROUP5 | SHIELD_GROUP
#define CHANNEL0_GROUP_IDX          TSI_GROUP_IDX5

#define BANK1_CHANNEL_NUMS          (1U)
#define BANK1_CHANNEL_MSK           TSI_PB13 | SHIELD_CHANNEL
#define BANK1_GROUP_MSK             TSI_GROUP5 | SHIELD_GROUP
#define CHANNEL1_GROUP_IDX          TSI_GROUP_IDX5

#define BANK2_CHANNEL_NUMS          (1U)
#define BANK2_CHANNEL_MSK           TSI_PB14 | SHIELD_CHANNEL
#define BANK2_GROUP_MSK             TSI_GROUP5 |SHIELD_GROUP
#define CHANNEL2_GROUP_IDX          TSI_GROUP_IDX5
```

After configuring the BANK, the user needs to modify the variables in ***Figure 4-3. BANK array configuration***. The group_id_array defines the group ID of each channel, and the sequence of the array elements also represents the location of the data of each channel in the key_data array, which is convenient for users to view the data of each channel during application / debugging. tsi_bank_array defines bank-related data, including the number of channels in each bank, channel pins, group of channels and initial state of bank.

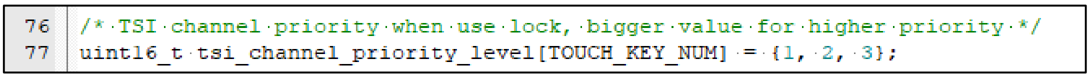
Figure 4-3. BANK array configuration



```
/* group_id_array array for map group id of each channel */
uint8_t group_id_array[CHANNEL_NUMS] = {
    CHANNEL0_GROUP_IDX, CHANNEL1_GROUP_IDX, CHANNEL2_GROUP_IDX,
};

/* bank array for map bank_channel nums, bank_channel_mask, bank_group_mask, bank initial state */
tsi_bank_struct tsi_bank_array[TSI_BANK_NUMS] = {
    {BANK0_CHANNEL_NUMS, BANK0_CHANNEL_MSK, BANK0_GROUP_MSK, BANK_IDLE},
    {BANK1_CHANNEL_NUMS, BANK1_CHANNEL_MSK, BANK1_GROUP_MSK, BANK_IDLE},
    {BANK2_CHANNEL_NUMS, BANK2_CHANNEL_MSK, BANK2_GROUP_MSK, BANK_IDLE},
};
```

When LOCK is used, the priority of each channel can be defined by the following array. The higher value is the higher priority, refer to ***Figure 4-4. Channel priority definition***.

Figure 4-4. Channel priority definition



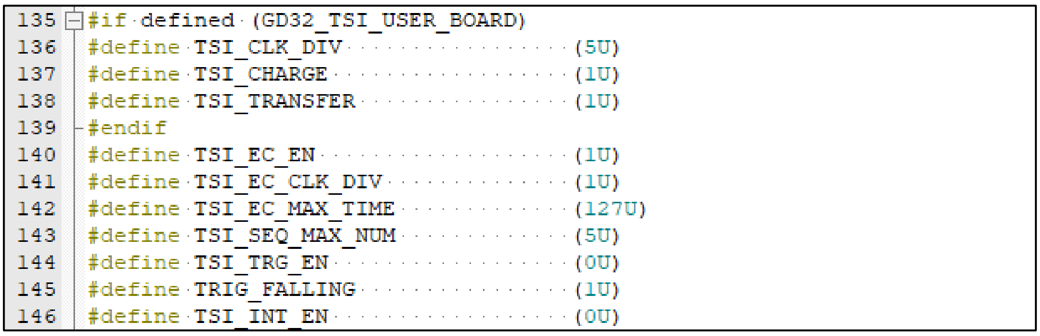
```
/* TSI channel priority when use lock, bigger value for higher priority */
uint16_t tsi_channel_priority_level[TOUCH_KEY_NUM] = {1, 2, 3};
```

4.3. TSI parameter configuration

Figure 4-5. TSI parameter configuration shows the TSI parameter configuration (can be configured in tsi_config.h), as follow:

1) Macro TSI_CLK_DIV defines the charge transfer clock (CTCLK) division factor.
2) Macro TSI_CHARGE defines the charge status duration time.
3) Macro TSI_TRANSFER defines the charge transfer state duration time.
4) Macro TSI_EC_EN defines the extend charge state enable.
5) Macro TSI_EC_CLK_DIV defines the extend charge clock (ECCLK) devision factor.
6) Macro TSI_EC_MAX_TIME defines the extend charge state maximum duration time.
7) Macro TSI_SEQ_MAX_NUM defines max cycle numbers of a sequence.
8) Macro TSI_TRG_EN defines trigger mode selection switch.
9) Macro TRIG_FALLING defines external edge trigger mode.
10) Macro TSI_INT_EN defines the TSI interrupt function switch.

Figure 4-5. TSI parameter configuration



```
#if defined (GD32_TSI_USER_BOARD)
#define TSI_CLK_DIV                  (5U)
#define TSI_CHARGE                   (1U)
#define TSI_TRANSFER                 (1U)
#endif
#define TSI_EC_EN                    (1U)
#define TSI_EC_CLK_DIV               (1U)
#define TSI_EC_MAX_TIME              (127U)
#define TSI_SEQ_MAX_NUM              (5U)
#define TSI_TRG_EN                   (0U)
#define TRIG_FALLING                 (1U)
#define TSI_INT_EN                   (0U)
```

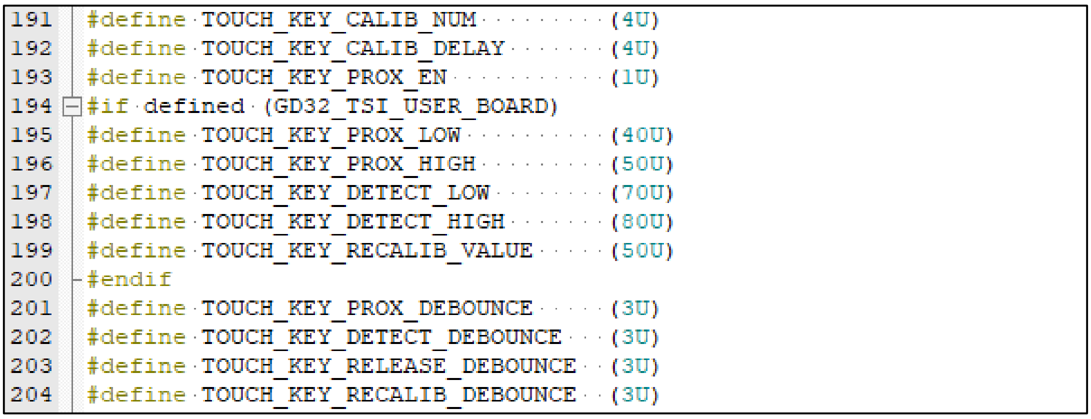
4.4. TouchKey parameter configuration

Figure 4-6. TouchKey parameter configuration shows the the TouchKey parameter configuration (can be configured in tsi_config.h), as follow:

1) Macro TOUCH_KEY_CALIB_NUM defines channel data calibration numbers.
2) Macro TOUCH_KEY_CALIB_DELAY defines the delay numbers before the calibration.
3) Macro TOUCH_KEY_PROX_EN defines whether to use proximity detection function.
4) Macro TOUCH_KEY_PROX_LOW defines the proximity detection threshold low.
5) Macro TOUCH_KEY_PROX_HIGH defines the proximity detection threshold high.
6) Macro TOUCH_KEY_DETECT_LOW defines the touch detection threshold low.
7) Macro TOUCH_KEY_DETECT_HIGH defines the touch detection threshold high.
8) Macro TOUCH_KEY_RECALIB_VALUE defines the touch detection re-calibration value.
9) Macro TOUCH_KEY_PROX_DEBOUNCE defines the proximity detection debounce counts.
10) Macro TOUCH_KEY_DETECT_DEBOUNCE defines the touch detection debounce counts.
11) Macro TOUCH_KEY_RELEASE_DEBOUNCE defines touch release debounce counts.

12) Macro TOUCH_KEY_RECALIB_DEBOUNCE defines touch re-calibration debounce counts.

Figure 4-6. TouchKey parameter configuration



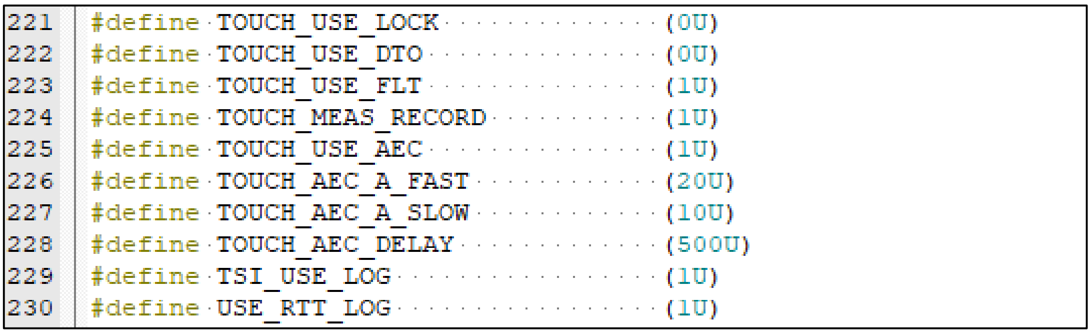
```
#define TOUCH_KEY_CALIB_NUM         (4U)
#define TOUCH_KEY_CALIB_DELAY       (4U)
#define TOUCH_KEY_PROX_EN           (1U)
#if defined (GD32_TSI_USER_BOARD)
#define TOUCH_KEY_PROX_LOW          (40U)
#define TOUCH_KEY_PROX_HIGH         (50U)
#define TOUCH_KEY_DETECT_LOW        (70U)
#define TOUCH_KEY_DETECT_HIGH       (80U)
#define TOUCH_KEY_RECALIB_VALUE     (50U)
#endif
#define TOUCH_KEY_PROX_DEBOUNCE     (3U)
#define TOUCH_KEY_DETECT_DEBOUNCE   (3U)
#define TOUCH_KEY_RELEASE_DEBOUNCE  (3U)
#define TOUCH_KEY_RECALIB_DEBOUNCE  (3U)
```

4.5. TouchKey function configuration

Figure 4-7. TouchKey function configuration shows the the TouchKey function configuration (can be configured in tsi_config.h), as follow:

1) Macro TOUCH_USE_LOCK defines whether to use TouchKey lock function, which prevents multiple touches from being detected at the same time.
2) Macro TOUCH_USE_DTO defines whether to use detection timeout function, which prevents error detection by external obstacle.
3) Macro TOUCH_USE_FLT defines whether to use filter function.
4) Macro TOUCH_MEAS_RECORD defines whether to use last measure as record.
5) Macro TOUCH_USE_AEC defines whether to use adaptive environment change detection.
6) Macro TOUCH_AEC_A_FAST and TOUCH_AEC_A_SLOW defines the factor of the first order low-pass filter used in adaptive environment detection.
7) Macro TOUCH_AEC_DELAY defines the adaptive environment detection period.
8) Macro TSI_USE_LOG defines whether to use log output function.
9) Macro USE_RTT_LOG defines whether to use RTT as log output.

Figure 4-7. TouchKey function configuration

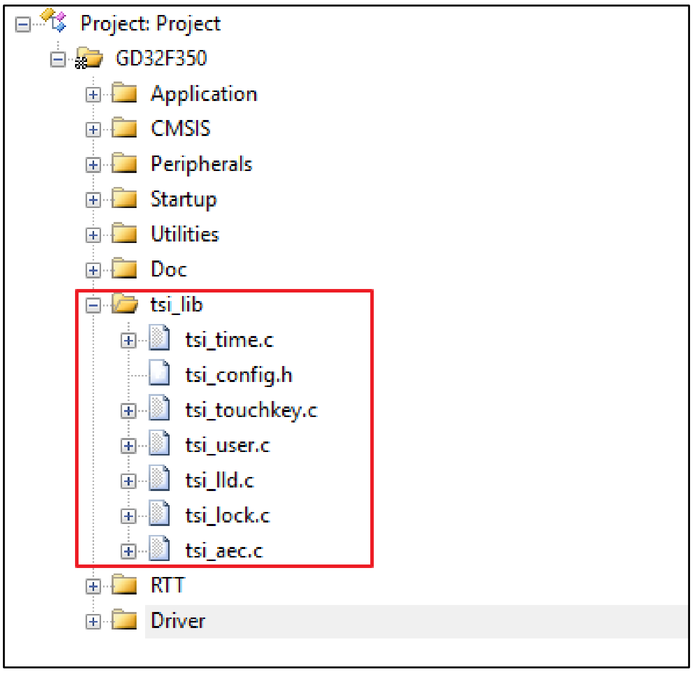


```
#define TOUCH_USE_LOCK              (0U)
#define TOUCH_USE_DTO               (0U)
#define TOUCH_USE_FLT               (1U)
#define TOUCH_MEAS_RECORD           (1U)
#define TOUCH_USE_AEC               (1U)
#define TOUCH_AEC_A_FAST            (20U)
#define TOUCH_AEC_A_SLOW            (10U)
#define TOUCH_AEC_DELAY             (500U)
#define TSI_USE_LOG                 (1U)
#define USE_RTT_LOG                 (1U)
```

5. TouchKey library use

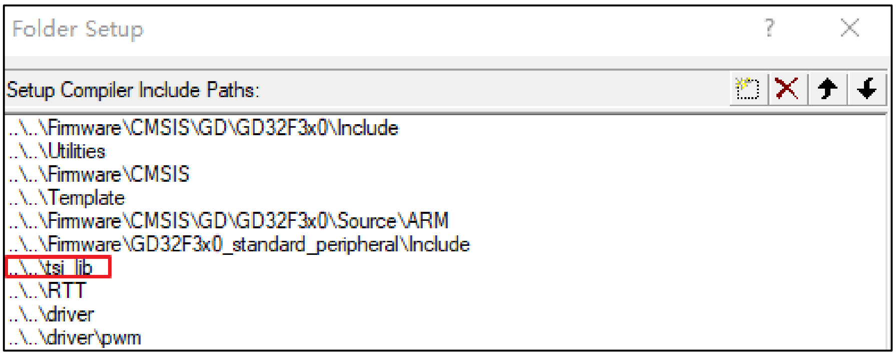
The following contents introduce how to add TouchKey library to KEIL V5.35 project by using GD32350R_EVAL as example.

1) Add .c files in tsi_lib



```
Project: Project
  GD32F350
    Application
    CMSIS
    Peripherals
    Startup
    Utilities
    Doc
    tsi_lib
      tsi_time.c
      tsi_config.h
      tsi_touchkey.c
      tsi_user.c
      tsi_lld.c
      tsi_lock.c
      tsi_aec.c
    RTT
    Driver
```

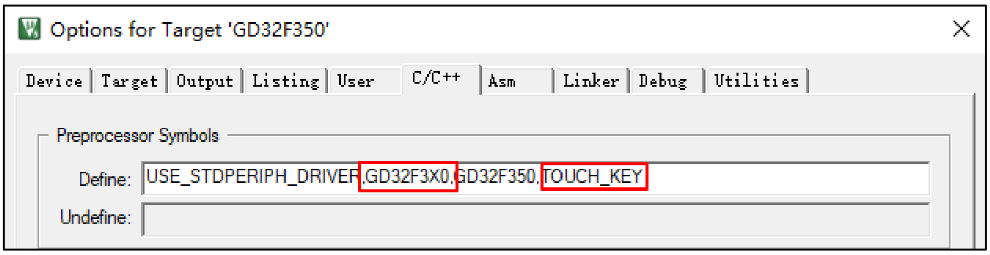
2) Include .h files in tsi_lib



Setup Compiler Include Paths:

```
..\..\Firmware\CMSIS\GD\GD32F3x0\Include
..\..\Utilities
..\..\Firmware\CMSIS
..\..\Template
..\..\Firmware\CMSIS\GD\GD32F3x0\Source\ARM
..\..\Firmware\GD32F3x0_standard_peripheral\Include
..\..\tsi_lib
..\..\RTT
..\..\driver
..\..\driver\pwm
```

3) Add predefined macro TOUCH_KEY and GD32F3X0 or GD32W515



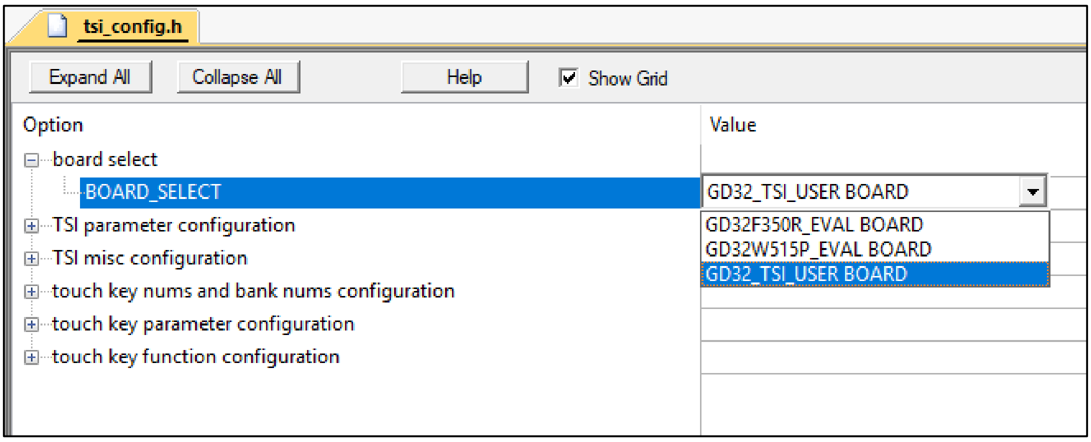
Preprocessor Symbols

Define: USE_STDPERIPH_DRIVER,GD32F3X0,GD32F350,TOUCH_KEY

Undefine:

4) Select needed board

If using a predefined development board, select the "GD32F350R_EVAL BOARD" or "GD32W515P_EVAL BOARD". If using the customer's development board, select "GD32_TSI_USER BOARD".

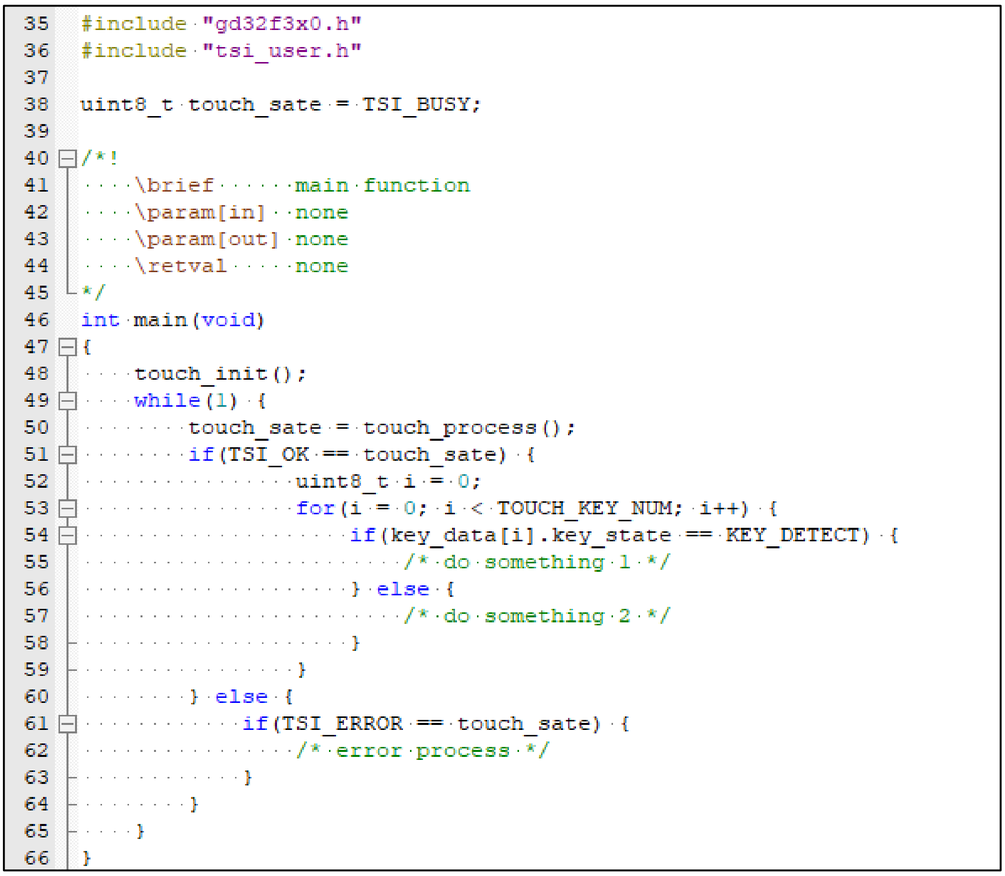


Option	Value
⊟ board select	
BOARD_SELECT	GD32_TSI_USER BOARD
⊞ TSI parameter configuration	
⊞ TSI misc configuration	
⊞ touch key nums and bank nums configuration	
⊞ touch key parameter configuration	
⊞ touch key function configuration	

GD32F350R_EVAL BOARD
GD32W515P_EVAL BOARD
GD32_TSI_USER BOARD

Note: For predefined development board "GD32F350R_EVAL BOARD" or "GD32W515P_EVAL BOARD", user needs not to configure the other ***TouchKey library configuration***.

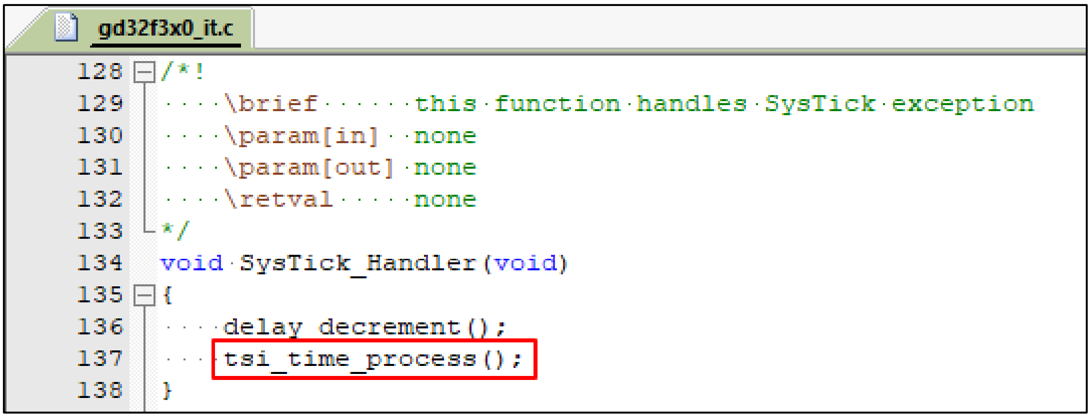
5) User code implement in main.c



```
#include "gd32f3x0.h"
#include "tsi_user.h"

uint8_t touch_sate = TSI_BUSY;

/*!
    \brief      main function
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
{
    touch_init();
    while(1) {
        touch_sate = touch_process();
        if(TSI_OK == touch_sate) {
                uint8_t i = 0;
                for(i = 0; i < TOUCH_KEY_NUM; i++) {
                    if(key_data[i].key_state == KEY_DETECT) {
                        /* do something 1 */
                    } else {
                        /* do something 2 */
                    }
                }
        } else {
            if(TSI_ERROR == touch_sate) {
                /* error process */
            }
        }
    }
}
```

6) Add timing management in tick handler

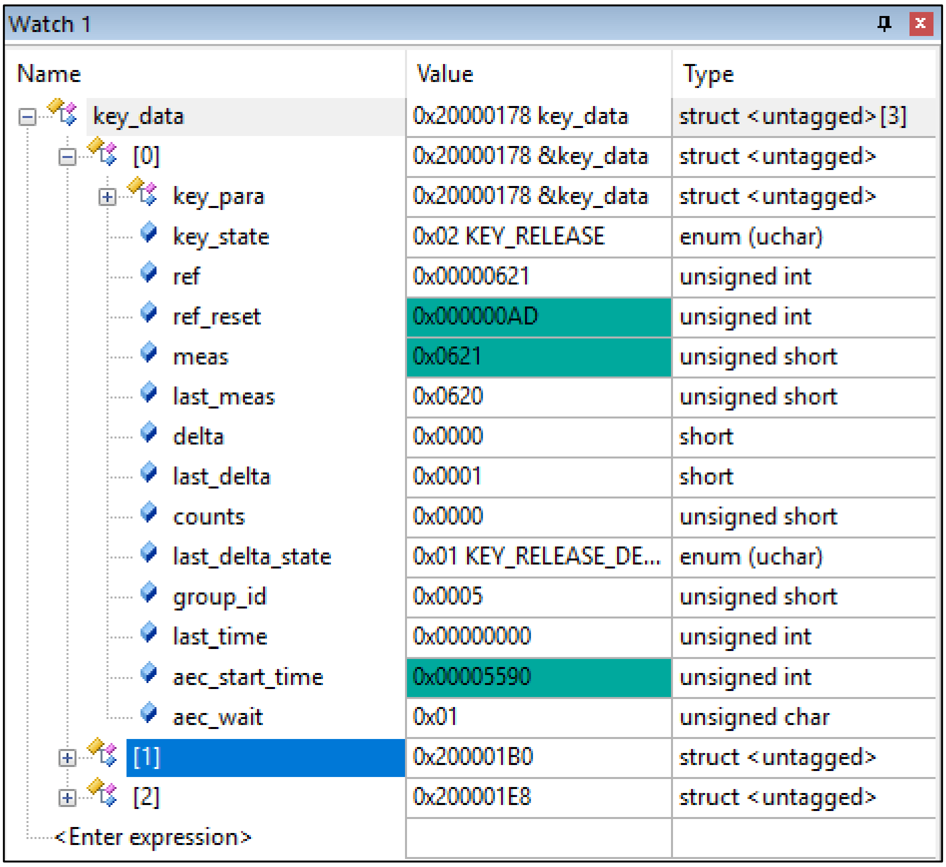
In project, using the systick as the tick, the code is as follow. otherwise, need to include "tsi_time.h" in this C file.



```c
/*!
    \brief      this function handles SysTick exception
    \param[in]  none
    \param[out] none
    \retval     none
*/
void SysTick_Handler(void)
{
    delay_decrement();
    tsi_time_process();
}
```

7) Debug the project

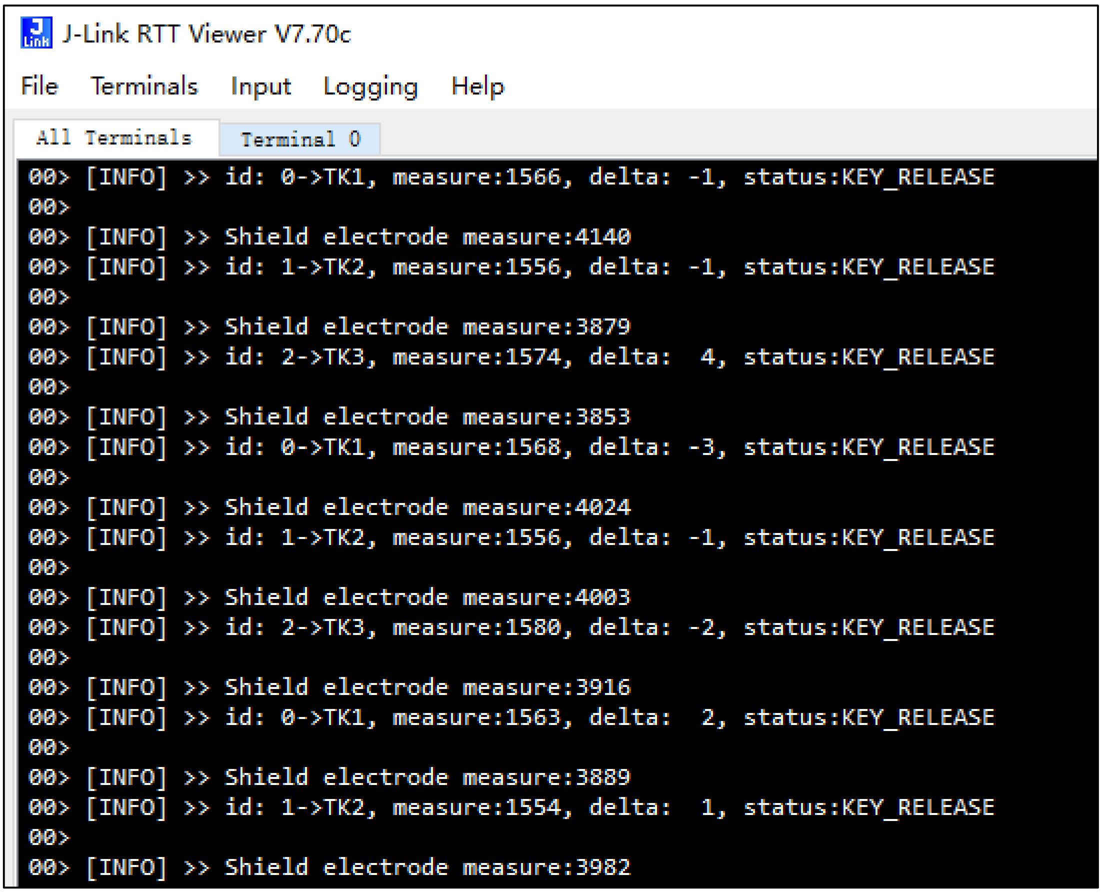
Firstly, Debug and run the project; then add variable "key_data" to watch window. The touch key channel data (include state, reference, delta…) can be showed in the following window.



Name	Value	Type
key_data	0x20000178 key_data	struct <untagged>[3]
[0]	0x20000178 &key_data	struct <untagged>
key_para	0x20000178 &key_data	struct <untagged>
key_state	0x02 KEY_RELEASE	enum (uchar)
ref	0x00000621	unsigned int
ref_reset	0x000000AD	unsigned int
meas	0x0621	unsigned short
last_meas	0x0620	unsigned short
delta	0x0000	short
last_delta	0x0001	short
counts	0x0000	unsigned short
last_delta_state	0x01 KEY_RELEASE_DE…	enum (uchar)
group_id	0x0005	unsigned short
last_time	0x00000000	unsigned int
aec_start_time	0x00005590	unsigned int
aec_wait	0x01	unsigned char
[1]	0x200001B0	struct <untagged>
[2]	0x200001E8	struct <untagged>
<Enter expression>		

8) Log output

If the log output function is enabled in the project, take J-Link RTT as an example, user needs need to initialize RTT and redirect printf in the code. Log output is as follows (including touch key channel ID, measurement, delta, status):



```
00> [INFO] >> id: 0->TK1, measure:1566, delta: -1, status:KEY_RELEASE
00>
00> [INFO] >> Shield electrode measure:4140
00> [INFO] >> id: 1->TK2, measure:1556, delta: -1, status:KEY_RELEASE
00>
00> [INFO] >> Shield electrode measure:3879
00> [INFO] >> id: 2->TK3, measure:1574, delta:  4, status:KEY_RELEASE
00>
00> [INFO] >> Shield electrode measure:3853
00> [INFO] >> id: 0->TK1, measure:1568, delta: -3, status:KEY_RELEASE
00>
00> [INFO] >> Shield electrode measure:4024
00> [INFO] >> id: 1->TK2, measure:1556, delta: -1, status:KEY_RELEASE
00>
00> [INFO] >> Shield electrode measure:4003
00> [INFO] >> id: 2->TK3, measure:1580, delta: -2, status:KEY_RELEASE
00>
00> [INFO] >> Shield electrode measure:3916
00> [INFO] >> id: 0->TK1, measure:1563, delta:  2, status:KEY_RELEASE
00>
00> [INFO] >> Shield electrode measure:3889
00> [INFO] >> id: 1->TK2, measure:1554, delta:  1, status:KEY_RELEASE
00>
00> [INFO] >> Shield electrode measure:3982
```

6. Revision history

Table 6-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	May.25, 2023

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.