RZ/A1H Group

Touch Panel Utility

Introduction

This application note describes the operation of a FreeRTOS based, embedded firmware project which provides a development platform for a Touch Panel Utility using the RIIC driver.

Target Device / Target Board

This application note is covering the usage of the touch panel utility application, which is in of itself, not device or OS specific. However, the sample project containing this application is running FreeRTOS10 and contains RZ/A1H drivers.

Contents

List of Abbreviations and Acronyms

Abbreviation	Full Form
IIC (or I^2C)	Inter-Integrated Circuit
I/O	Input/Output
INTC	INTerrupt Controller
LCD	Liquid Crystal Display
MCU	MicroController Unit
OS	Operating System
RIIC	Renesas Inter-Integrated Circuit

Table 1-1 List of Abbreviations and Acronyms

1. Specifications

The Touch Panel utility controls a touch panel via RIIC device controller (ch0), which is implemented on RZ/A1H.

2. Operation Check Conditions

To ensure the touch screen application is enabled in software, please check that:

```
/* Enable control for src/application/app_touchscreen sample application */
#define R_SELF_INSERT_APP_TOUCH_SCREEN (R_OPTION_ENABLE)
```

is present inside of "application_cfg.h".

3. Application Functionality

The function of the touch panel sample application is to detect a touch event and draw a small green rectangle at the coordinates of the event, see figure 2. Additionally, the sample application will update the console to display the coordinates of the event and a categorisation of the event type. The sample application will place the event into one of three categories:

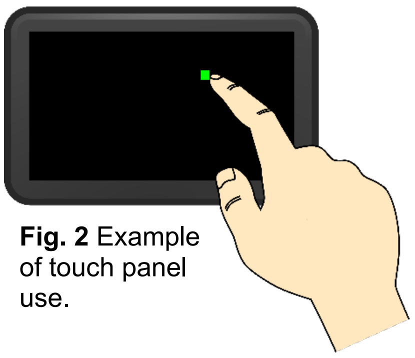


Fig. 2 Example of touch panel use.

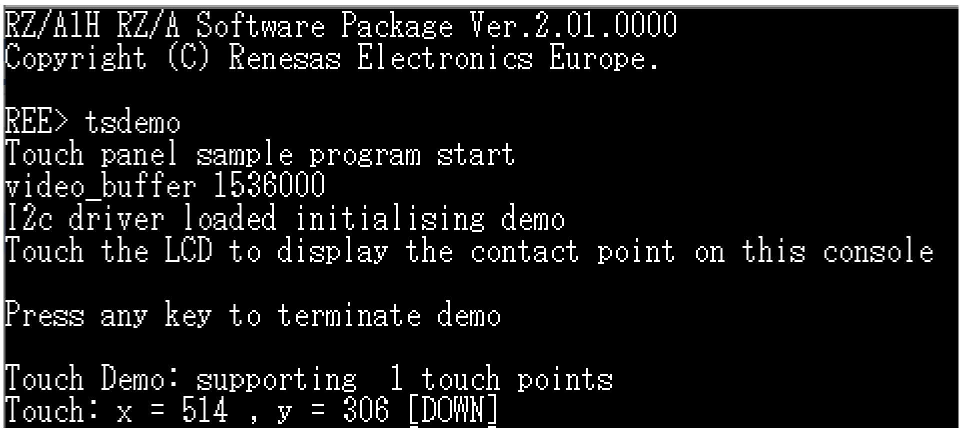
UP	Finger is no longer placed on the touch panel
DOWN	Finger is currently placed on touch panel, but is stationary
MOVE	Finger is currently placed on touch panel, but has moved

Fig. 3 Status Table

The image displayed in figure 4 shows the expected console output upon detection of an event.
This is displayed in the format:

```
Touch: x = $$ , y = ££ [category]
```

Where **$$** represents the X coordinate value, **££** represents the Y coordinate value and **[category]** holds the event categorisation.



```
RZ/A1H RZ/A Software Package Ver.2.01.0000
Copyright (C) Renesas Electronics Europe.

REE> tsdemo
Touch panel sample program start
video_buffer 1536000
I2c driver loaded initialising demo
Touch the LCD to display the contact point on this console

Press any key to terminate demo

Touch Demo: supporting  1 touch points
Touch: x = 514 , y = 306 [DOWN]
```

Fig. 4 Expected Console Output from Sample Application.

4. Software Description

This section of the application note describes the touch screen sample application.

4.1 Operation Outline

Figure 5 outlines the overall structure of the software modules used in this sample application and their interaction with the target hardware.

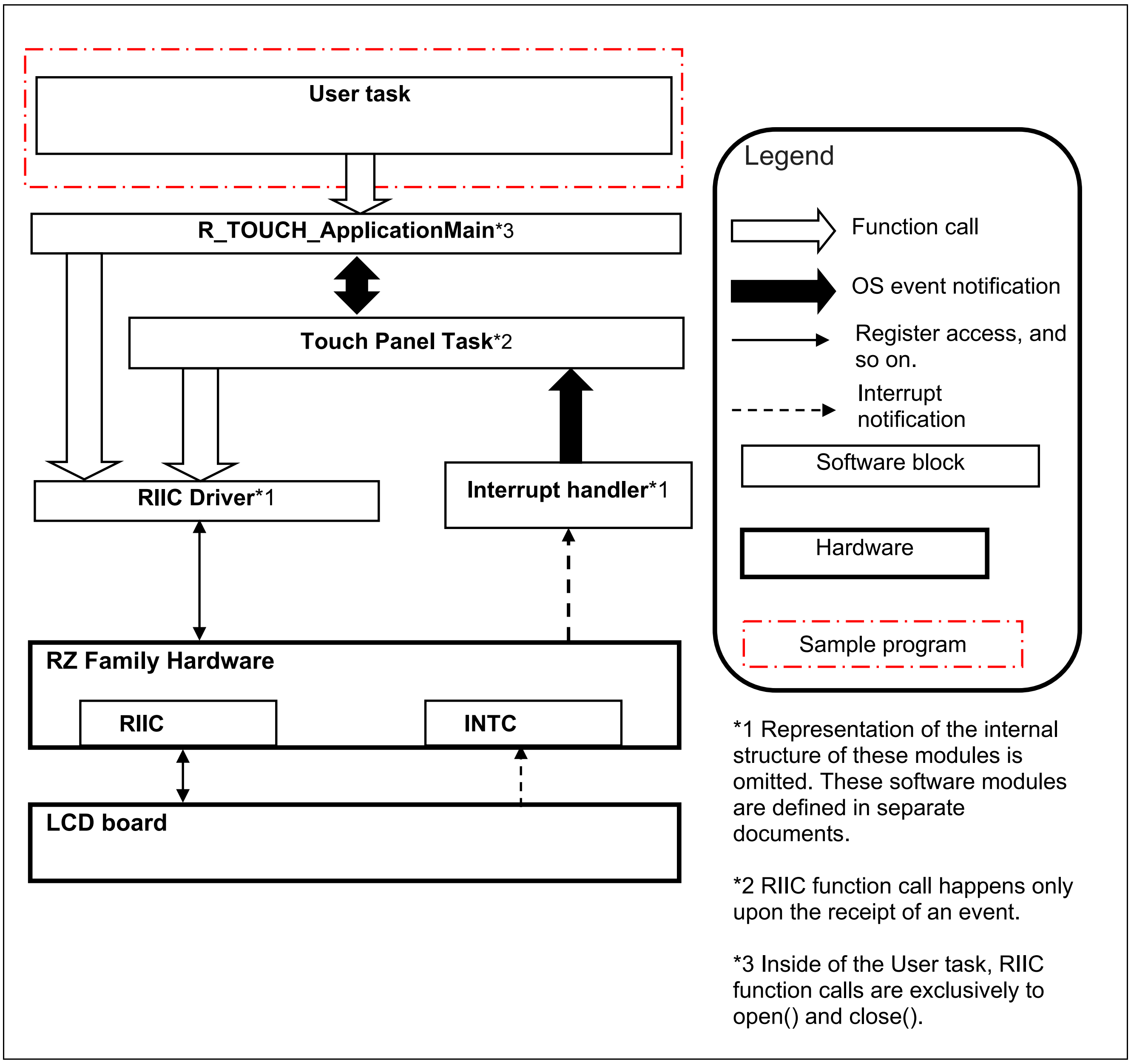

User task
R_TOUCH_ApplicationMain*3
Touch Panel Task*2
RIIC Driver*1
Interrupt handler*1
RZ Family Hardware
RIIC
INTC
LCD board
Legend
Function call
OS event notification
Register access, and so on.
Interrupt notification
Software block
Hardware
Sample program
*1 Representation of the internal structure of these modules is omitted. These software modules are defined in separate documents.
*2 RIIC function call happens only upon the receipt of an event.
*3 Inside of the User task, RIIC function calls are exclusively to open() and close().

Fig. 5 Figure Touch Panel Utility System Block Diagram

As can be seen in the figure 5 the expectation is for the user to create a task which calls the `R_TOUCH_ApplicationMain()` function.

The `R_TOUCH_ApplicationMain()` function is responsible for opening drivers and creating a “Touch Panel Task”, this task holds all subsequent responsibility for interaction with the touch panel.

4.2 Inserting the Application into a Project

It is assumed the specifications outlined in section 1 of this document have been met.

The touch panel sample application can be started by calling the `R_TOUCH_ApplicationMain()` function (found in file `r_touch_capacitive.c`), it is expected that this will be called from inside of a user created task.

Shown below is a control flowchart of the `R_TOUCH_ApplicationMain()` function.

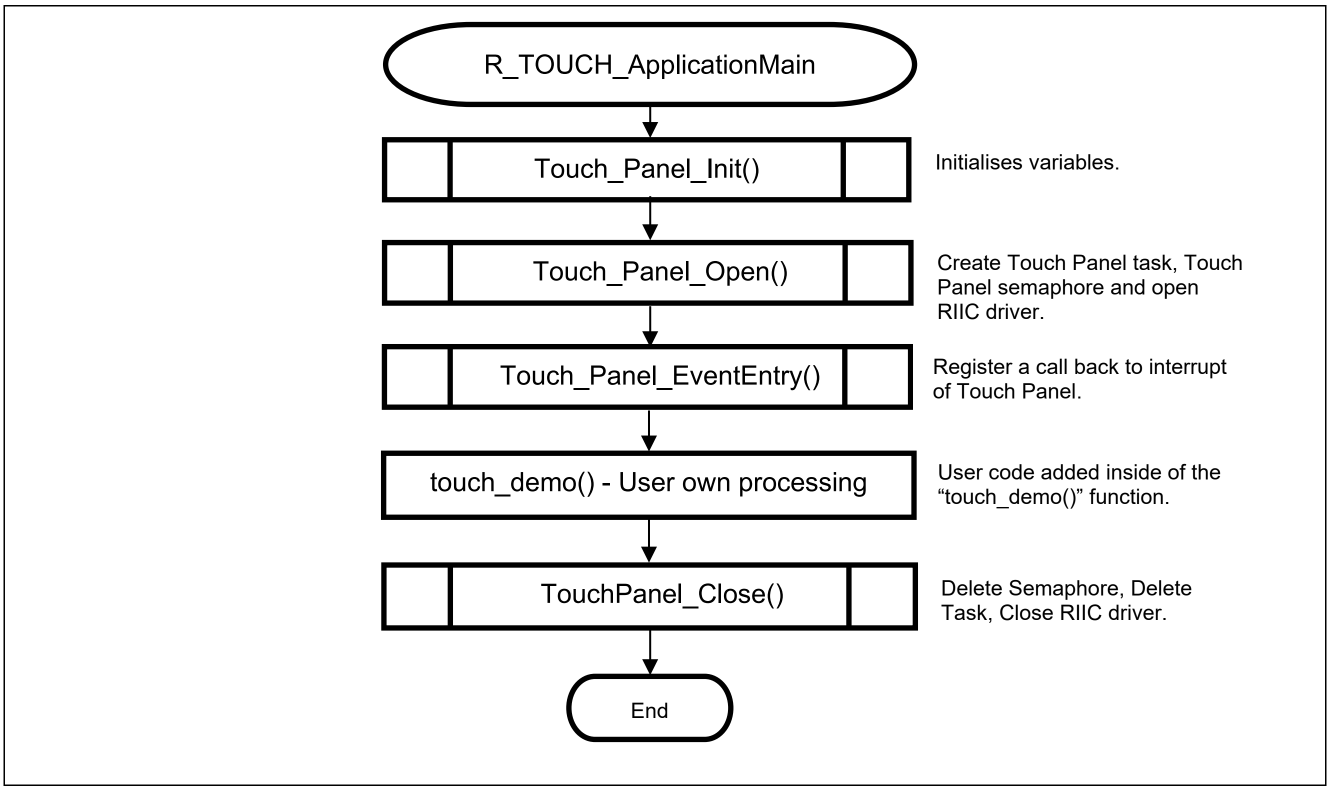

R_TOUCH_ApplicationMain
Touch_Panel_Init()
Initialises variables.
Touch_Panel_Open()
Create Touch Panel task, Touch Panel semaphore and open RIIC driver.
Touch_Panel_EventEntry()
Register a call back to interrupt of Touch Panel.
touch_demo() - User own processing
User code added inside of the "touch_demo()" function.
TouchPanel_Close()
Delete Semaphore, Delete Task, Close RIIC driver.
End

Fig. 6 Simplified Control Flow Scheme of the Touch Panel Utility

4.3 Modifying the Application

As a user, there are two primary sections of code suggested for modification:

The first section of code is the `touch_demo()` function seen in figure 6. The currently implemented `touch_demo()` function is responsible for initialising the touch screen and then blocking the "User Task" seen in figure 5 until receipt of a character through the serial console.

```
static void touch_demo (void *parameters)
{
    fprintf(s_dsp_console->p_out,"Touch Demo: supporting %2-d touch points\r\n", 1);

    /* initialize screen */
    R_TOUCH_init_screen();

    /* START - User Places Concurrent Code Here */
    while (control(R_DEVLINK_FilePtrDescriptor(s_dsp_console->p_in), CTL_GET_RX_BUFFER_COUNT,
NULL) == 0)
    {
        R_OS_TaskSleep(5);
    }
    /* END */

    /* un-initialize screen */
    R_TOUCH_uninit_screen();

    fgetc(s_dsp_console->p_in);
}
```

The expectation is for the user to place any operations desired to run concurrently with the "Touch Panel Task" inside of the `touch_demo()` function, between the `R_TOUCH_init_screen()` and `R_TOUCH_uninit_screen()` function calls.

The second section of code for user modification is the "Touch Panel Task", which is found inside of `tp_task.c`. This task is where the user should insert any code related to the processing of information to and from the touch screen.

Files

```
Software
  │
  ├────src
    ├────arm
    ├────FreeRTOS
    ├────supplierX
    └────renesas
       ├────configuration
       │  application_cfg.h                    - Application settings defined
       ├────application
       │   ├────app_ touchscreen
       │   │  r_drawrectangle.c                - Functions to draw on-screen rectangle
       │   │  r_touch_capacitive.c             - Handles communication with capacitive controllers
       │   ├────inc
       │   │   │  jcu_swap.h                   - Definitions for JCU feature
       │   │   │  r_draw_jpeg.h                - Draw cursor header
       │   │   │  r_touch_capacitive.h         - Touch_capacitive header
       │   │   └──Image
       │   │       Arrow.jpg                   - Arrow image to be drawn
       │   │       JCU_ExampleImage.h          - Arrow image information header
       │   │       JCU_ExampleImage.S          - Includes the arrow image
       │   ├────app_2
       │   ├────app_3...
       ├────compiler
       ├────configuration
       ├────drivers
       └────middleware
           ├────touch
           │   ├────inc
           │     lcd_controller_if.h           - LCD Control interface header
           │     tp_if.h                       - Touch Panel utility interface header
           └────src
           │   └────lcd_controller
           │       │  r_lcd_controller_if.c    - LCD Control interface
           │       ├────FT5x06
           │       │  lcd_ft5x06.c             - FT5x06 Control Operation
           │       │  lcd_ft5x06.h             - FT5x06 Control header
           │       │  lcd_ft5x06_int.c         - FT5x06 Control interrupt handler
           │       │  lcd_ft5x06_int.h         - FT5x06 Control interrupt header
           │       └────FT5216
           │          lcd_ft5216.c             - FT5216 Control Operation
           │          lcd_ft5216.h             - FT5216 Control header
           │          lcd_ft5216_int.c         - FT5216 Control interrupt handler
           │          lcd_ft5216_int.h         - FT5216 Control interrupt header
           └────touch
             tp.c                              - Touch Panel utility internal operation
             tp.h                              - Touch Panel utility internal header
             tp_if.c                           - Touch Panel utility interface
             tp_task.c                         - Touch Panel utility task operation
             tp_task.h                         - Touch Panel utility task header
```

5. Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

6. File Index

6.1 File List

Here is a list of all files with brief descriptions:

7. Data Structure Documentation

7.1 LCDEVT_ENTRY Struct Reference

```
#include <lcd_ft5216.h>
```

7.1.1 Data Fields

- **LcdEvt_EntryType mode**
- **LcdCBFunc function**
- **LcdEvt_LockState evtlock**

7.1.2 Detailed Description

Event entry struct

Definition at line 113 of file lcd_ft5216.h.

7.1.3 Field Documentation

(1) LcdEvt_LockState evtlock

Event lock state

Definition at line 116 of file lcd_ft5216.h.

(2) LcdCBFunc function

Definition at line 115 of file lcd_ft5216.h.

(3) LcdEvt_EntryType mode

The type of touch panel event entry

Definition at line 114 of file lcd_ft5216.h.

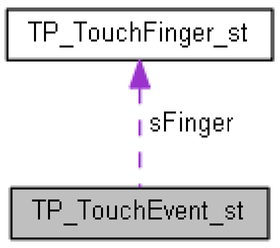
(4) The documentation for this struct was generated from the following file:

- **lcd_ft5216.h**

7.2 TP_TouchEvent_st Struct Reference

```
#include <tp_if.h>
```

Collaboration diagram for TP_TouchEvent_st:



7.2.1 Data Fields

- **TP_TouchFinger_st sFinger** [**TP_TOUCHNUM_MAX**]

7.2.2 Detailed Description

Definition at line 71 of file tp_if.h.

7.2.3 Field Documentation

(1) **TP_TouchFinger_st sFinger[TP_TOUCHNUM_MAX]**

Definition at line 72 of file tp_if.h.

(2) **The documentation for this struct was generated from the following file:**

- **tp_if.h**

7.3 TP_TouchFinger_st Struct Reference

```
#include <tp_if.h>
```

7.3.1 Data Fields

- **TpEvt_EntryType eState**
- uint16_t **unPosX**
- uint16_t **unPosY**

7.3.2 Detailed Description

Definition at line 65 of file tp_if.h.

7.3.3 Field Documentation

(1) **TpEvt_EntryType eState**

Definition at line 66 of file tp_if.h.

(2) **uint16_t unPosX**

Definition at line 67 of file tp_if.h.

(3) **uint16_t unPosY**

Definition at line 68 of file tp_if.h.

(4) **The documentation for this struct was generated from the following file:**

- **tp_if.h**

7.4 TPEVT_COORDINATES Struct Reference

```
#include <tp.h>
```

7.4.1 Data Fields

- int32_t **x**
- int32_t **y**

7.4.2 Detailed Description

Coordinate structure

Definition at line 115 of file tp.h.

7.4.3 Field Documentation

(1) **int32_t x**

x-coordinate [pixel]

Definition at line 116 of file tp.h.

(2) **int32_t y**

y-coordinate [pixel]

Definition at line 117 of file tp.h.

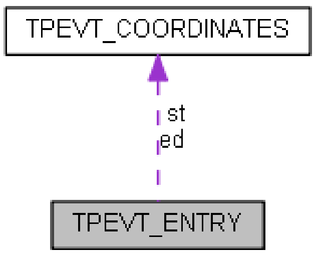
(3) **The documentation for this struct was generated from the following file:**

- **tp.h**

7.5 TPEVT_ENTRY Struct Reference

```
#include <tp.h>
```

Collaboration diagram for TPEVT_ENTRY:



7.5.1 Data Fields

- **TpEvt_EntryType mode**
- **TPEVT_COORDINATES st**
- **TPEVT_COORDINATES ed**
- void(* **function**)(int_t, **TP_TouchEvent_st** *)
- **TpEvt_LockState evtlock**

7.5.2 Detailed Description

Event entry struct

Definition at line 121 of file tp.h.

7.5.3 Field Documentation

(1) **TPEVT_COORDINATES ed**

The lower-right coordinates of the rectangular area in which touch event can be received. [pixel]

Definition at line 124 of file tp.h.

(2) **TpEvt_LockState evtlock**

Event lock state

Definition at line 126 of file tp.h.

(3) **void(* function) (int_t, TP_TouchEvent_st *)**

Event notification callback function pointer

Definition at line 125 of file tp.h.

(4) **TpEvt_EntryType mode**

The type of touch panel event entry

Definition at line 122 of file tp.h.

(5) **TPEVT_COORDINATES st**

The upper-left coordinates of the rectangular area in which touch event can be received. [pixel]

Definition at line 123 of file tp.h.

(6) **The documentation for this struct was generated from the following file:**

- **tp.h**

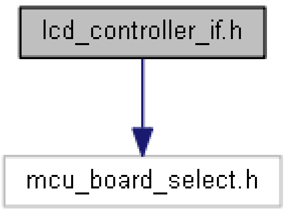
8. File Documentation

8.1 lcd_controller_if.h File Reference

LCD Driver API header.

```
#include "mcu_board_select.h"
```

Include dependency graph for lcd_controller_if.h:



8.1.1 Macros

- #define **LCD_SLAVE_ADDRESS** (0x38 << 1)

8.1.2 Typedefs

- typedef void(* **LcdCBFunc**) (void *)

8.1.3 Enumerations

- enum **LcdEvt_EntryType** { **LCDEVT_ENTRY_NONE** = 0x0000, **LCDEVT_ENTRY_TP** = 0x0001, **LCDEVT_ENTRY_ALL** = 0x0001 }

8.1.4 Functions

- void **R_LCD_Init** (void)
  Sets the LCD board initialization counter (nLcdInitCnt) to 0.
- int_t **R_LCD_Open** (const uint32_t unIrqLv, const int16_t nTskPri, const uint32_t unTskStk)
  Opens a communication environment with the LCD board.
  This function enables the user to perform multiple open operations.
- int_t **R_LCD_Close** (void)
  Closes a communication environment with the LCD board.
  When LCD_Open is used to perform multiple open operations, this function must be called the same number of times.
- uint8_t **R_LCD_WriteCmd** (const uint16_t unDevAddr, const uint8_t uCmd, const uint8_t uData, const uint32_t unSize)
- uint8_t **R_LCD_ReadCmd** (const uint16_t unDevAddr, const uint8_t uCmd, uint8_t *puData, const uint32_t unSize)
  Receives data from the LCD board via the RIIC.
- int_t **R_LCD_EventEntry** (const **LcdEvt_EntryType** eType, const **LcdCBFunc** function)
  Registers an LCD board event.
- int_t **R_LCD_EventErase** (const int_t nId)
  Removes an LCD board event.
- int_t **R_LCD_StartInt** (const **LcdEvt_EntryType** eType)
  Removes masking of specified interrupt type.
- int_t **R_LCD_Restart** (void)
  Reset LCD board.
- void **R_LCD_ReadVersion** (uint8_t *puData)
- void **R_LCD_SetBacklight** (const uint8_t uLevel)
  Set bright level of backlight.
- void **R_LCD_SetBuzzer** (const uint8_t uScale)

Set scale of buzzer.

8.1.5 Detailed Description

LCD Driver API header.

Rev: 30 Date:: 2016-12-21 12:02:44 +0900#

8.1.6 Macro Definition Documentation

(1) **#define LCD_SLAVE_ADDRESS (0x38 << 1)**

LCD slave address

Definition at line 48 of file lcd_controller_if.h.

8.1.7 Typedef Documentation

(1) **typedef void(* LcdCBFunc) (void *)**

Definition at line 41 of file lcd_controller_if.h.

8.1.8 Enumeration Type Documentation

(1) **enum LcdEvt_EntryType**

The type of touch panel event entry

(a) **Enumerator:**

LCDEVT_ENTRY_NONE	None
LCDEVT_ENTRY_TP	None
LCDEVT_ENTRY_ALL	All

Definition at line 57 of file lcd_controller_if.h.

```
57               {
58      LCDEVT_ENTRY_NONE = 0x0000,
59      LCDEVT_ENTRY_TP   = 0x0001,
61      LCDEVT_ENTRY_ALL  = 0x0001
62 } LcdEvt_EntryType ;
```

8.1.9 Function Documentation

(1) **int_t R_LCD_Close (void)**

Closes a communication environment with the LCD board.

When LCD_Open is used to perform multiple open operations, this function must be called the same number of times.

(a) **Return values:**

NONE	

(2) int_t R_LCD_EventEntry (const LcdEvt_EntryType *eType*, const LcdCBFunc *function*)

Registers an LCD board event.

(a) **Parameters:**

in	*eType*	Specified Interrupt type
in	*function*	Call-back function

(b) **Return values:**

0-(LCDEVT_ENTRY_MAX-1)	registration value
-1	event registration failure

(3) int_t R_LCD_EventErase (const int_t *nId*)

Removes an LCD board event.

(a) **Parameters:**

in	*nId*	Event ID

(b) **Return values:**

NONE	

(4) void R_LCD_Init (void)

Sets the LCD board initialization counter (nLcdInitCnt) to 0.

R_LCD_Init

(a) **Return values:**

NONE	

(5) int_t R_LCD_Open (const uint32_t *unIrqLv*, const int16_t *nTskPri*, const uint32_t *unTskStk*)

Opens a communication environment with the LCD board.

This function enables the user to perform multiple open operations.

(a) **Parameters:**

in	*unIrqLv*	IRQ interrupt priority (0 to 255) Sets the GIC interrupt priority
in	*nTskPri*	Task Priority Sets the value of osPriority type.
in	*unTskStk*	Not Used.

(b) **Return values:**

0	Normal end
-1	Open error

(6) uint8_t R_LCD_ReadCmd (const uint16_t *unDevAddr*, const uint8_t *uCmd*, uint8_t * *puData*, const uint32_t *unSize*)

Receives data from the LCD board via the RIIC.

(a) **Parameters:**

in	*unDevAddr*	LCD Device Address
in	*uCmd*	Not Used
in	**puData*	Receive data buffer pointer
out	*unSize*	Receive Data Length

(b) **Return values:**

0	normal end
-1	data send processing error

(7) void R_LCD_ReadVersion (uint8_t * *puData*)

(a) **Parameters:**

out	**puData*	: pointer to receive buffer

(b) **Return values:**

0	

(8) int_t R_LCD_Restart (void)

Reset LCD board.

(a) **Return values:**

0	

(9) void R_LCD_SetBacklight (const uint8_t *uLevel*)

Set bright level of backlight.

(a) **Parameters:**

in	*uLevel*	bright level

(b) **Return values:**

None.	

(10) void R_LCD_SetBuzzer (const uint8_t *uScale*)

Set scale of buzzer.

(a) **Parameters:**

in	*uScale*	scale

(b) **Return values:**

None.	

(11) int_t R_LCD_StartInt (const LcdEvt_EntryType *eType*)

Removes masking of specified interrupt type.

(a) **Parameters:**

in	*eType*	Not Used

(b) **Return values:**

0	event successfully removed
-1	event removal failure

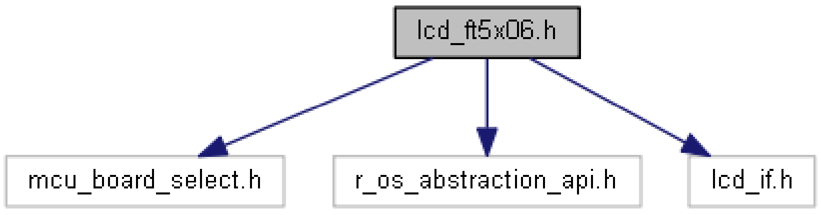
(12) uint8_t R_LCD_WriteCmd (const uint16_t *unDevAddr*, const uint8_t *uCmd*, const uint8_t *uData*, const uint32_t *unSize*)

8.2 lcd_ft5216.h File Reference

LCD Driver internal header.

```
#include "mcu_board_select.h"
#include "r_os_abstraction_api.h"
#include "lcd_if.h"
```

Include dependency graph for lcd_ft5216.h:



8.2.1 Data Structures

- struct **LCDEVT_ENTRY**

8.2.2 Macros

- #define **DBG_LEVEL_OT** (-1) /* onetime debug */
- #define **DBG_LEVEL_DEF** (0) /* default */
- #define **DBG_LEVEL_ERR** (1) /* error */
- #define **DBG_LEVEL_MSG** (2) /* message */
- #define **DBG_LEVEL_LOG** (3) /* log */
- #define **DBG_LEVEL_DBG** (4) /* debug */
- #define **DBG_LEVEL** (**DBG_LEVEL_ERR**)
- #define **DBG_printf_OT** printf
- #define **DBG_printf_DEF** printf
- #define **DBG_printf_ERR** printf
- #define **DBG_printf_MSG** 1 ? (int32_t) 0 : printf
- #define **DBG_printf_LOG** 1 ? (int32_t) 0 : printf
- #define **DBG_printf_DBG** 1 ? (int32_t) 0 : printf
- #define **SCOPE_STATIC** static
- #define **LCDEVT_ENTRY_MAX** (1)

8.2.3 Enumerations

- enum **LcdEvt_LockState** { **LCD_EVT_UNLOCK** = 0, **LCD_EVT_LOCK** }

8.2.4 Functions

- int_t **LCD_Ft5216_Open** (const uint32_t unIrqLv, int16_t nTskPri, uint32_t unTskStk)
  Opens the communication environment with the FT5216.
- int_t **LCD_Ft5216_Close** (void)
  Closes the communication environment with the FT5216.
- uint8_t **LCD_Ft5216_WriteCmd** (const uint16_t unDevAddr, const uint8_t uData, const uint32_t unSize)
  Sends data to the FT5216 via the RIIC DeviceController ch1.
- uint8_t **LCD_Ft5216_ReadCmd** (const uint16_t unDevAddr, uint8_t *puData, const uint32_t unSize)
  Reads data from the FT5216 via the RIIC DeviceController ch1.
- int_t **LCD_Ft5216_EventEntry** (const **LcdEvt_EntryType** eType, const **LcdCBFunc** function)
  Registers in the event management structure a call-back function linked to an interrupt from the FT5216.
  After registration finishes, the LCD interrupt is enabled and the event ID is sent as a return value.
- int_t **LCD_Ft5216_EventErase** (const int_t nId)
  Removes the registration information for the specified event ID from the event management structure.

- int_t **LCD_Ft5216_StartInt** (const **LcdEvt_EntryType** eType)
  Removes masking of specified interrupt type.
- **LCDEVT_ENTRY** * **LCD_Ft5216_GetEventTable** (const int_t nId)
  Get assigned callback event.
- int32_t **LCD_Ft5216_SendEvtMsg** (const uint32_t unEvtFlg)
  Send event message to synchronism.
- int32_t **LCD_Ft5216_WaitEvtMsg** (void)
  Wait event message to synchronism.
- void **LCD_Ft5216_ClearEvtMsg** (const uint32_t unEvtFlg)
  Clear assigned event flag.

8.2.5 Variables

- int32_t **sLcdSemIdAcc**

8.2.6 Detailed Description

LCD Driver internal header.

Rev: 30 Date:: 2016-12-21 12:02:44 +0900#

8.2.7 Macro Definition Documentation

(1) **#define DBG_LEVEL (DBG_LEVEL_ERR)**

Definition at line 56 of file lcd_ft5216.h.

(2) **#define DBG_LEVEL_DBG (4) /* debug */**

Definition at line 54 of file lcd_ft5216.h.

(3) **#define DBG_LEVEL_DEF (0) /* default */**

Definition at line 50 of file lcd_ft5216.h.

(4) **#define DBG_LEVEL_ERR (1) /* error */**

Definition at line 51 of file lcd_ft5216.h.

(5) **#define DBG_LEVEL_LOG (3) /* log */**

Definition at line 53 of file lcd_ft5216.h.

(6) **#define DBG_LEVEL_MSG (2) /* message */**

Definition at line 52 of file lcd_ft5216.h.

(7) **#define DBG_LEVEL_OT (-1) /* onetime debug */**

Definition at line 49 of file lcd_ft5216.h.

(8) **#define DBG_printf_DBG 1 ? (int32_t) 0 : printf**

Definition at line 85 of file lcd_ft5216.h.

(9) **#define DBG_printf_DEF printf**

Definition at line 63 of file lcd_ft5216.h.

(10) **#define DBG_printf_ERR printf**

Definition at line 68 of file lcd_ft5216.h.

(11) **#define DBG_printf_LOG 1 ? (int32_t) 0 : printf**

Definition at line 80 of file lcd_ft5216.h.

(12) **#define DBG_printf_MSG 1 ? (int32_t) 0 : printf**

Definition at line 75 of file lcd_ft5216.h.

(13) **#define DBG_printf_OT printf**

Definition at line 58 of file lcd_ft5216.h.

(14) **#define LCDEVT_ENTRY_MAX (1)**

The max number of event entry
Definition at line 96 of file lcd_ft5216.h.

(15) **#define SCOPE_STATIC static**

Definition at line 92 of file lcd_ft5216.h.

8.2.8 Enumeration Type Documentation

(1) **enum LcdEvt_LockState**

Touch panel event lock state

(a) **Enumerator:**

LCD_EVT_UNLOCK	Unlocked
LCD_EVT_LOCK	Locked

Definition at line 103 of file lcd_ft5216.h.

```
103             {
104     LCD_EVT_UNLOCK = 0,
105     LCD_EVT_LOCK
106 } LcdEvt_LockState ;
```

8.2.9 Function Documentation

(1) **void LCD_Ft5216_ClearEvtMsg (const uint32_t *unEvtFlg*)**

Clear assigned event flag.

(a) **Parameters:**

in	*unEvtFlg*	: event flag

(b) **Return values:**

None.	

(2) int_t LCD_Ft5216_Close (void)

Closes the communication environment with the FT5216.

(a) **Return values:**

NONE	

(3) int_t LCD_Ft5216_EventEntry (const LcdEvt_EntryType *eType*, const LcdCBFunc *function*)

Registers in the event management structure a call-back function linked to an interrupt from the FT5216. After registration finishes, the LCD interrupt is enabled and the event ID is sent as a return value.

(a) **Parameters:**

in	*eType*	Specified Interrupt type
in	*function*	Call-back function

(b) **Return values:**

0	to (LCDEVT_ENTRY_MAX - 1)
-1	event registration failure

(4) int_t LCD_Ft5216_EventErase (const int_t *nId*)

Removes the registration information for the specified event ID from the event management structure.

(a) **Parameters:**

in	*nId*	Event ID return value of LCD_EventEntry function.

(b) **Return values:**

NONE	

(5) LCDEVT_ENTRY* LCD_Ft5216_GetEventTable (const int_t *nId*)

Get assigned callback event.

(a) **Parameters:**

in	*nId*	event ID

(b) **Return values:**

LCDEVT_ENTRY	pointer to event.

(6) int_t LCD_Ft5216_Open (const uint32_t *unIrqLv*, int16_t *nTskPri*, uint32_t *unTskStk*)

Opens the communication environment with the FT5216.

(a) **Parameters:**

in	*unIrqLv*	IRQ interrupt priority (0 to 255) Sets the GIC interrupt priority
in	*nTskPri*	Task Priority Sets the value of osPriority type.
in	*unTskStk*	Not Used.

(b) **Return values:**

0	Normal end
-1	failure to open

(7) uint8_t LCD_Ft5216_ReadCmd (const uint16_t *unDevAddr*, uint8_t * *puData*, const uint32_t *unSize*)

Reads data from the FT5206 via the RIIC DeviceController ch1.

(a) **Parameters:**

in	*unDevAddr*	LCD Device Address
in	**puData*	Receive data buffer pointer
out	*unSize*	Receive Data Length

(b) **Return values:**

0	normal end
-1	data receive error

(8) int32_t LCD_Ft5216_SendEvtMsg (const uint32_t *unEvtFlg*)

Send event message to synchronism.

(a) **Parameters:**

in	*unEvtFlg*	event flag

(b) **Return values:**

0	Operation successful.
-1	Error occurred.

(9) **int_t LCD_Ft5216_StartInt (const LcdEvt_EntryType *eType*)**

Removes masking of specified interrupt type.

(a) **Parameters:**

in	*eType*	Specified interrupt type

(b) **Return values:**

0	Always, normal end

(10) **int32_t LCD_Ft5216_WaitEvtMsg (void)**

Wait event message to synchronism.

(a) **Return values:**

0	Event flag list.
-1	: Error occurred.

(11) **uint8_t LCD_Ft5216_WriteCmd (const uint16_t *unDevAddr*, const uint8_t *uData*, const uint32_t *unSize*)**

Sends data to the FT5216 via the RIIC DeviceController ch1.

(a) **Parameters:**

in	*unDevAddr*	LCD Device Address
in	*uData*	Send Data
in	*unSize*	Send Data Length

(b) **Return values:**

0	normal end
-1	data send processing error

8.2.10 Variable Documentation

(1) **int32_t sLcdSemIdAcc**

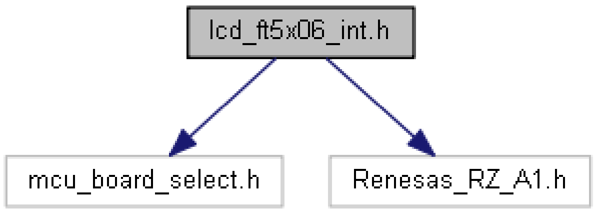
(2)

8.3 lcd_ft5216_int.h File Reference

LCD Driver internal header for interrupt.

```
#include "mcu_board_select.h"
#include "Renesas_RZ_A1.h"
```

Include dependency graph for lcd_ft5216_int.h:



8.3.1 Macros

- #define **LCD_FT5216_INT_NUM** (IRQ3_IRQn)

8.3.2 Functions

- int_t **LCD_Ft5216_Int_Open** (const uint32_t unIrqLv)
  Open LCD interrupt.
- int_t **LCD_Ft5216_Int_Close** (void)
  Close LCD interrupt.
- int_t **LCD_Ft5216_Int_Start** (void)
  Enable interrupt of assigned type.

8.3.3 Detailed Description

LCD Driver internal header for interrupt.

Rev: 30 Date:: 2016-12-21 12:02:44 +0900#

8.3.4 Macro Definition Documentation

(1) #define LCD_FT5216_INT_NUM (IRQ3_IRQn)

Definition at line 48 of file lcd_ft5216_int.h.

8.3.5 Function Documentation

(1) int_t LCD_Ft5216_Int_Close (void)

Close LCD interrupt.

(a) **Return values:**

0	Operation Successful
-1	Error occurred

(2) int_t LCD_Ft5216_Int_Open (const uint32_t *unIrqLv*)

Open LCD interrupt.

(a) **Parameters:**

unIrqLv	IRQ interrupt level

(b) **Return values:**

0	Operation Successful
-1	Error occurred

(3) int_t LCD_Ft5216_Int_Start (void)

Enable interrupt of assigned type.

(a) **Return values:**

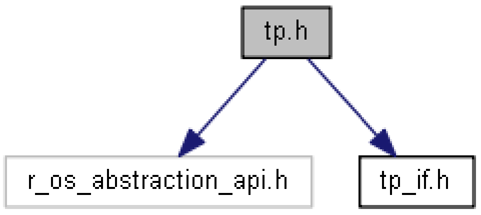
0	Operation Successful

8.4 tp.h File Reference

TouchPanel Driver internal header.

```
#include "r_os_abstraction_api.h"
#include "tp_if.h"
```

Include dependency graph for tp.h:



8.4.1 Data Structures

- struct **TPEVT_COORDINATES**
- struct **TPEVT_ENTRY**

8.4.2 Macros

- #define **DBG_LEVEL_OT** (-1) /* onetime debug */
- #define **DBG_LEVEL_DEF** (0) /* default */
- #define **DBG_LEVEL_ERR** (1) /* error */
- #define **DBG_LEVEL_MSG** (2) /* message */
- #define **DBG_LEVEL_LOG** (3) /* log */
- #define **DBG_LEVEL_DBG** (4) /* debug */
- #define **DBG_LEVEL** (**DBG_LEVEL_ERR**)
- #define **DBG_printf_OT** printf
- #define **DBG_printf_DEF** printf
- #define **DBG_printf_ERR** printf
- #define **DBG_printf_MSG** 1 ? (int32_t) 0 : printf
- #define **DBG_printf_LOG** 1 ? (int32_t) 0 : printf
- #define **DBG_printf_DBG** 1 ? (int32_t) 0 : printf
- #define **SCOPE_STATIC** static
- #define **TPEVT_ENTRY_MAX** (16)
- #define **TP_EVTFLG_NONE** (0x00000000)
- #define **TP_EVTFLG_PENIRQ** (0x00000001) /*! Touch Panel event flag, pen interrupt */
- #define **TP_EVTFLG_EXIT** (0x00000080) /*! Touch Panel event flag, exit and delete task */
- #define **TP_EVTFLG_ALL** (**TP_EVTFLG_PENIRQ** | **TP_EVTFLG_EXIT**)

8.4.3 Enumerations

- enum **TpEvt_LockState** { **TP_EVT_UNLOCK** = 0, **TP_EVT_LOCK** }

8.4.4 Functions

- void **TP_Init** (void)
  Initializes internal variables of the touch panel driver.
  • Securing of touch panel event entry area
  • Setting of internal variable nEvtEntryId to -1
  • Setting of internal variable TpEvtLockInf to TP_EVT_UNLOCK
  .
- int_t **TP_Open** (const int_t nWidth, const int_t nHeight, const uint32_t unIrqLv, const int16_t nTskPri, const uint32_t unTskStk)

Opens the touch panel driver.

• Setting the LCD size in the driver's variables ScreenWidth and ScreenHeight

• Generation of touch panel task synchronization semaphore

• Generation of touch panel task

• Setting of task priority of touch panel task

• Opening of communication environment with LCD board

• Registration of call-back event when touch panel interrupt occurs in LCD event.

- int_t **TP_Close** (void)

  Closes the touch panel driver.

  • Removal of call-back event when touch panel interrupt occurs in LCD event

  • Removal of all touch panel event registrations by the user

  • Removal of touch panel task

  • Removal of semaphore for synchronization with the touch panel task.

- int_t **TP_EventEntry** (const **TpEvt_EntryType** eMode, const int32_t nPosX, const int32_t nPosY, const int32_t nWidth, const int32_t nHeight, const **TpCBFunc** function)

  Registers in the event table a call-back function linked to a touch panel interrupt.

  After registration finishes, the event ID is sent as a return value.

  .

- int_t **TP_EventErase** (const int_t nId)

  Removes an event from the call-back event table of the touch panel driver.

  • Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

  • Disabling of event associated with event ID (TPEVT_ENTRY_NON)

- int_t **TP_ChangeEventEntry** (const int_t nId, const int32_t nPosX, const int32_t nPosY, const int32_t nWidth, const int32_t nHeight)

  The rectangular area to which the event ID specified by the 1st argument (nId)

  is registered is changed to the rectangular area specified by the 2nd to 5th arguments.

  • Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

  • Event ID checking (unregistered ID or removed ID)

  • Registration of event in area of specified ID in touch panel event table.

- int_t **TP_EventLockAll** (void)

  Locks all registered touch panel call-back events.

  Calls the function described in TP_EventLock, to set all events to the locked state (TP_EVT_LOCK).

- int_t **TP_EventUnlockAll** (void)

  Unlocks all registered touch panel call-back events.

  Calls the function described in TP_EventUnlock, to set all events to the unlocked state (TP_EVT_UNLOCK).

- int_t **TP_EventLock** (const int_t nId)

  Locks the touch panel call-back event specified by the 1st argument (nId).

  • Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

  • Setting the event specified by the event ID to the locked state (TP_EVT_LOCK) in the touch panel event table.

- int_t **TP_EventUnlock** (const int_t nId)

  Unlocks the touch panel call-back event specified by the 1st argument (nId).

  • Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

  • Setting the event specified by the event ID to the unlocked state (TP_EVT_UNLOCK) in the touch panel event table.

- **TPEVT_ENTRY * TP_GetEventTable** (const int_t nId)

  Acquires from the touch panel driver call-back event table the

  pointer address at which the event ID event information is registered.

- **TpEvt_LockState TP_GetEventLockInf** (void)

Acquires the lock state of the touch panel call-back event.

- void **TP_GetScreenSize** (int_t *pnWidth, int_t *pnHeight)
  Acquires the screen size of the LCD panel.
- int32_t **TP_SendEvtMsg** (const uint32_t unEvtFlg)
  Sends a synchronization event message.
- int32_t **TP_WaitEvtMsg** (void)
  Waits to receive a synchronization event message.
- void **TP_ClearEvtMsg** (const uint32_t unEvtFlg)
  Clears the specified event flag.

8.4.5 Variables

- os_task_t * **p_os_task**

8.4.6 Detailed Description

TouchPanel Driver internal header.

Rev: 30 Date:: 2016-12-21 12:02:44 +0900#

8.4.7 Macro Definition Documentation

(1) **#define DBG_LEVEL (DBG_LEVEL_ERR)**

Definition at line 53 of file tp.h.

(2) **#define DBG_LEVEL_DBG (4) /* debug */**

Definition at line 51 of file tp.h.

(3) **#define DBG_LEVEL_DEF (0) /* default */**

Definition at line 47 of file tp.h.

(4) **#define DBG_LEVEL_ERR (1) /* error */**

Definition at line 48 of file tp.h.

(5) **#define DBG_LEVEL_LOG (3) /* log */**

Definition at line 50 of file tp.h.

(6) **#define DBG_LEVEL_MSG (2) /* message */**

Definition at line 49 of file tp.h.

(7) **#define DBG_LEVEL_OT (-1) /* onetime debug */**

Definition at line 46 of file tp.h.

(8) **#define DBG_printf_DBG 1 ? (int32_t) 0 : printf**

Definition at line 82 of file tp.h.

(9) **#define DBG_printf_DEF printf**

Definition at line 60 of file tp.h.

(10) **#define DBG_printf_ERR printf**

Definition at line 65 of file tp.h.

(11) **#define DBG_printf_LOG 1 ? (int32_t) 0 : printf**

Definition at line 77 of file tp.h.

(12) **#define DBG_printf_MSG 1 ? (int32_t) 0 : printf**

Definition at line 72 of file tp.h.

(13) **#define DBG_printf_OT printf**

Definition at line 55 of file tp.h.

(14) **#define SCOPE_STATIC static**

Definition at line 89 of file tp.h.

(15) **#define TP_EVTFLG_ALL (TP_EVTFLG_PENIRQ | TP_EVTFLG_EXIT)**

Definition at line 98 of file tp.h.

(16) **#define TP_EVTFLG_EXIT (0x00000080) /*! Touch Panel event flag, exit and delete task */**

Definition at line 97 of file tp.h.

(17) **#define TP_EVTFLG_NONE (0x00000000)**

Definition at line 95 of file tp.h.

(18) **#define TP_EVTFLG_PENIRQ (0x00000001) /*! Touch Panel event flag, pen interrupt */**

Definition at line 96 of file tp.h.

(19) **#define TPEVT_ENTRY_MAX (16)**

The max number of event entry

Definition at line 93 of file tp.h.

8.4.8 Enumeration Type Documentation

(1) **enum TpEvt_LockState**

Touch panel event lock state

(a) **Enumerator:**

TP_EVT_UNLOCK	Unlocked

TP_EVT_LOCK	Locked

Definition at line 105 of file tp.h.

```
             {
    TP_EVT_UNLOCK = 0,
    TP_EVT_LOCK
} TpEvt_LockState ;
```

8.4.9 Function Documentation

(1) int_t TP_ChangeEventEntry (const int_t *nId*, const int32_t *nPosX*, const int32_t *nPosY*, const int32_t *nWidth*, const int32_t *nHeight*)

The rectangular area to which the event ID specified by the 1st argument (nId)
is registered is changed to the rectangular area specified by the 2nd to 5th arguments.
• Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)
• Event ID checking (unregistered ID or removed ID)
• Registration of event in area of specified ID in touch panel event table.

(a) **Parameters:**

in	*nId*	event ID
in	*nPosX*	X-coordinate of LCD area
in	*nPosY*	Y-coordinate of LCD area
in	*nWidth*	width of LCD area
in	*nHeight*	height of LCD area

(b) **Return values:**

0	Operation successful.
-1	Error occurred.

(2) void TP_ClearEvtMsg (const uint32_t *unEvtFlg*)

Clears the specified event flag.

(a) **Parameters:**

in	*unEvtFlg*	event flag

(b) **Return values:**

None.	

(3) int_t TP_Close (void)

Closes the touch panel driver.
• Removal of call-back event when touch panel interrupt occurs in LCD event
• Removal of all touch panel event registrations by the user
• Removal of touch panel task

• Removal of semaphore for synchronization with the touch panel task.

(a) **Return values:**

0	Operation successful.
-1	Error occurred.

(4) int_t TP_EventEntry (const TpEvt_EntryType *eMode*, const int32_t *nPosX*, const int32_t *nPosY*, const int32_t *nWidth*, const int32_t *nHeight*, const TpCBFunc *function*)

Registers in the event table a call-back function linked to a touch panel interrupt.
After registration finishes, the event ID is sent as a return value.
.
• Searching for a free area in the touch panel event table (Up to 16 touch panel events can be registered, and error processing occurs if no free area is available.)
• Making "specified touch action," "X coordinate of specified area," "Y coordinate of specified area," "width of specified area,"
"height of specified area," "specified call-back function" settings for the touch panel event table free area.
Note: When "X coordinate of specified area," "Y coordinate of specified area," "width of specified area," and "height of specified area" are registered in the touch panel event table, the following processing is performed to register the result as a rectangular area:
st.x (X coordinate of area start position) <- "X coordinate of specified area"
st.y (Y coordinate of area start position) <- "Y coordinate of specified area"
ed.x (X coordinate of area end position) <- ("X coordinate of specified area" - "width of specified area")
ed.y (Y coordinate of area end position) <- ("Y coordinate of specified area" - "height of specified area")

(a) **Parameters:**

in	*eMode*	event type
in	*nPosX*	X-coordinate of LCD area
in	*nPosY*	Y-coordinate of LCD area
in	*nWidth*	width of LCD area
in	*nHeight*	height of LCD area
in	*function*	callback function

(b) **Return values:**

0	to (TPEVT_ENTRY_MAX-1)
-1	Error occurred.

(5) int_t TP_EventErase (const int_t *nId*)

Removes an event from the call-back event table of the touch panel driver.
• Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)
• Disabling of event associated with event ID (TPEVT_ENTRY_NON)

(a) **Parameters:**

in	*nId*	event ID

(b) **Return values:**

0	Operation successful.
-1	Error occurred.

(6) int_t TP_EventLock (const int_t *nId*)

Locks the touch panel call-back event specified by the 1st argument (nId).

• Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

• Setting the event specified by the event ID to the locked state (TP_EVT_LOCK) in the touch panel event table.

(a) **Parameters:**

in	*nId*	event ID

(b) **Return values:**

0	Operation successful.
-1	Error occurred.

(7) int_t TP_EventLockAll (void)

Locks all registered touch panel call-back events.

Calls the function described in TP_EventLock, to set all events to the locked state (TP_EVT_LOCK).

(a) **Return values:**

0	Operation successful.
-1	Error occurred.

(8) int_t TP_EventUnlock (const int_t *nId*)

Unlocks the touch panel call-back event specified by the 1st argument (nId).

• Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

• Setting the event specified by the event ID to the unlocked state (TP_EVT_UNLOCK) in the touch panel event table.

(a) **Parameters:**

in	*nId*	event ID

(b) **Return values:**

0	Operation successful.
-1	Error occurred.

(9) int_t TP_EventUnlockAll (void)

Unlocks all registered touch panel call-back events.
Calls the function described in TP_EventUnlock, to set all events to the unlocked state (TP_EVT_UNLOCK).

(a) **Return values:**

0	Operation successful.
-1	Error occurred.

(10) TpEvt_LockState TP_GetEventLockInf (void)

Acquires the lock state of the touch panel call-back event.

(a) **Return values:**

TP_EVT_LOCK	In locked state
TP_EVT_UNLOCK	In unlocked state.

(11) TPEVT_ENTRY* TP_GetEventTable (const int_t *nId*)

Acquires from the touch panel driver call-back event table the
pointer address at which the event ID event information is registered.

(a) **Parameters:**

in	*nId*	event ID

(b) **Return values:**

TPEVT_ENTRY	pointer to event

(12) void TP_GetScreenSize (int_t * *pnWidth*, int_t * *pnHeight*)

Acquires the screen size of the LCD panel.

(a) **Parameters:**

out	**pnWidth*	pointer to width value
out	**pnHeight*	pointer to height value

(b) **Return values:**

None	

(13) void TP_Init (void)

Initializes internal variables of the touch panel driver.
• Securing of touch panel event entry area
• Setting of internal variable nEvtEntryId to -1
• Setting of internal variable TpEvtLockInf to TP_EVT_UNLOCK

(a) **Return values:**

None.	

(14) int_t TP_Open (const int_t *nWidth*, const int_t *nHeight*, const uint32_t *unIrqLv*, const int16_t *nTskPri*, const uint32_t *unTskStk*)

Opens the touch panel driver.
• Setting the LCD size in the driver's variables ScreenWidth and ScreenHeight
• Generation of touch panel task synchronization semaphore
• Generation of touch panel task
• Setting of task priority of touch panel task
• Opening of communication environment with LCD board
• Registration of call-back event when touch panel interrupt occurs in LCD event.

(a) **Parameters:**

in	*nWidth*	screen width
in	*nHeight*	screen height
in	*unIrqLv*	IRQ interrupt level
in	*nTskPri*	task priority
in	*unTskStk*	task stack size

(b) **Return values:**

0	Operation successful.
-1	Error occurred.

(15) int32_t TP_SendEvtMsg (const uint32_t *unEvtFlg*)

Sends a synchronization event message.

(a) **Parameters:**

in	*unEvtFlg*	event flag

(b) **Return values:**

0	Operation successful.
-1	Error occurred.

(16) int32_t TP_WaitEvtMsg (void)

Waits to receive a synchronization event message.

(a) **Return values:**

TP_EVTFLG_NONE	No event flags
TP_EVTFLG_PENIRQ	Interrupt pending
TP_EVTFLG_EXIT	End task

TP_EVTFLG_ALL	Both Interrupt pending and exit flag.
-1	Error occurred.

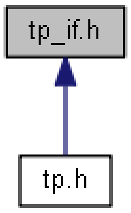
8.4.10 Variable Documentation

(1) os_task_t* p_os_task

8.5 tp_if.h File Reference

TouchPanel Driver API header.

This graph shows which files directly or indirectly include this file:



8.5.1 Data Structures

- struct **TP_TouchFinger_st**
- struct **TP_TouchEvent_st**

8.5.2 Macros

- #define **TP_TOUCHNUM_MAX** (2)

8.5.3 Typedefs

- typedef void(* **TpCBFunc**) (int_t, **TP_TouchEvent_st** *)

8.5.4 Enumerations

- enum **TpEvt_EntryType** { **TPEVT_ENTRY_NONE** = 0x0000, **TPEVT_ENTRY_UP** = 0x0001, **TPEVT_ENTRY_DOWN** = 0x0002, **TPEVT_ENTRY_MOVE** = 0x0004, **TPEVT_ENTRY_ALL** = 0x0007, **TPEVT_ENTRY_UNKNOWN** = 0x8000 }

8.5.5 Functions

- void **TouchPanel_Init** (void)
  Initializes the touch panel driver by calling the TP_Init.
- int_t **TouchPanel_Open** (const int_t nWidth, const int_t nHeight, const uint32_t unIrqLv, const int16_t nTskPri, const uint32_t unTskStk)
  Generates and initializes a touch panel task by calling the TP_Open.
  Do not call this function during touch panel utility has been opened.
- int_t **TouchPanel_Close** (void)
  Touch Panel utility close function.
- int_t **TouchPanel_EventEntry** (const **TpEvt_EntryType** eMode, const int32_t nPosX, const int32_t nPosY, const int32_t nWidth, const int32_t nHeight, const **TpCBFunc** function)
  Registers a call-back function linked to the LCD area where a touch panel event occurs in the touch panel event management structure.
  Calls the function described in TP_EventEntry, to perform the following processing:
  • Searching for a free area in the touch panel event table (Up to 16 touch panel events can be registered,

and error processing occurs if no free area is available.)

• Making "specified touch action," "X coordinate of specified area," "Y coordinate of specified area," "width of specified area," "height of specified area," "specified call-back function" settings for the touch panel event table free area.

Note: If events occur simultaneously in multiple registered areas that overlap, the associated call-back functions are executed in order, starting with the one with the lowest event ID.

- int_t **TouchPanel_EventErase** (const int_t nId)

  Removes registration information for the specified event ID from the touch panel event management structure.

  Calls the function described in TP_EventErase, to perform the following processing:

  • Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

  Disabling of event associated with event ID.

- int_t **TouchPanel_ChangeEventEntry** (const int_t nId, const int32_t nPosX, const int32_t nPosY, const int32_t nWidth, const int32_t nHeight)

  Changes the LCD area of the specified event ID.

  Calls the function described in TP_ChangeEventEntry, to perform the following processing:

  • Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

  • Event ID checking (unregistered ID or removed ID)

  Registration of event in area of specified ID in touch panel event table.

- int_t **TouchPanel_EventLockAll** (void)

  Locks processing of all touch panel events.

  Calls the function described in TP_EventLockAll, to perform the following processing:

  Setting all events in the touch panel event table to the locked state

  .

- int_t **TouchPanel_EventUnlockAll** (void)

  Unlocks processing of all touch panel events.

  Calls the function described in TP_EventUnlockAll, to perform the following processing:

  Setting all events in the touch panel event table to the unlocked state.

- int_t **TouchPanel_EventLock** (const int_t nId)

  Locks processing of the touch panel event specified by the event ID.

  Calls the function described in TP_EventLock, to perform the following processing:

  • Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

  Setting the event specified by the event ID to the locked state in the touch panel event table.

- int_t **TouchPanel_EventUnlock** (const int_t nId)

  Unlocks processing of the touch panel event specified by the event ID.

  Calls the function described in TP_EventUnlock, to perform the following processing:

  • Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

  Setting the event specified by the event ID to the unlocked state in the touch panel event table.

8.5.6 Detailed Description

TouchPanel Driver API header.

Rev: 30 Date:: 2016-12-21 12:02:44 +0900

8.5.7 Macro Definition Documentation

(1) **#define TP_TOUCHNUM_MAX (2)**

Definition at line 44 of file tp_if.h.

8.5.8 Typedef Documentation

(1) typedef void(* TpCBFunc) (int_t, TP_TouchEvent_st *)

Definition at line 75 of file tp_if.h.

8.5.9 Enumeration Type Documentation

(1) enum TpEvt_EntryType

The type of touch panel event entry

(a) **Enumerator:**

TPEVT_ENTRY_NONE	None
TPEVT_ENTRY_UP	Up
TPEVT_ENTRY_DOWN	Down
TPEVT_ENTRY_MOVE	Move
TPEVT_ENTRY_ALL	All
TPEVT_ENTRY_UNKNOWN	internal event state

Definition at line 50 of file tp_if.h.

```
              {
     TPEVT_ENTRY_NONE    = 0x0000,
     TPEVT_ENTRY_UP      = 0x0001,
     TPEVT_ENTRY_DOWN    = 0x0002,
     TPEVT_ENTRY_MOVE    = 0x0004,
     TPEVT_ENTRY_ALL     = 0x0007,
     TPEVT_ENTRY_UNKNOWN = 0x8000
 } TpEvt_EntryType ;
```

8.5.10 Function Documentation

(1) int_t TouchPanel_ChangeEventEntry (const int_t *nId*, const int32_t *nPosX*, const int32_t *nPosY*, const int32_t *nWidth*, const int32_t *nHeight*)

Changes the LCD area of the specified event ID.

Calls the function described in TP_ChangeEventEntry, to perform the following processing:

• Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

• Event ID checking (unregistered ID or removed ID)

Registration of event in area of specified ID in touch panel event table.

(a) **Parameters:**

in	*nId*	Event ID
in	*nPosX*	X coordinate of area after change
in	*nPosY*	Y coordinate of area after change
in	*nWidth*	Width of area after change
in	*nHeight*	Height of area after change

(b) **Return values:**

0	normal end
-1	LCD area change failure

(2) int_t TouchPanel_Close (void)

Touch Panel utility close function.

(a) **Return values:**

NONE	

(3) int_t TouchPanel_EventEntry (const TpEvt_EntryType *eMode*, const int32_t *nPosX*, const int32_t *nPosY*, const int32_t *nWidth*, const int32_t *nHeight*, const TpCBFunc *function*)

Registers a call-back function linked to the LCD area where a touch panel event occurs in the touch panel event management structure.
Calls the function described in TP_EventEntry, to perform the following processing:
• Searching for a free area in the touch panel event table (Up to 16 touch panel events can be registered, and error processing occurs if no free area is available.)
• Making “specified touch action,” “X coordinate of specified area,” “Y coordinate of specified area,” “width of specified area,” “height of specified area,” “specified call-back function” settings for the touch panel event table free area.
Note: If events occur simultaneously in multiple registered areas that overlap,
the associated call-back functions are executed in order, starting with the one with the lowest event ID.

(a) **Parameters:**

in	*eMode*	Specified touch action
in	*nPosX*	X coordinate of specified area
in	*nPosY*	Y coordinate of specified area
in	*nWidth*	width of specified area
in	*nHeight*	height of specified area
in	*function*	Specified call-back function

(b) **Return values:**

Success	event ID of 0 to (TPEVT_ENTRY_MAX -1) if successful

Fail	returns -1

(4) int_t TouchPanel_EventErase (const int_t *nId*)

Removes registration information for the specified event ID from the touch panel event management structure.

Calls the function described in TP_EventErase, to perform the following processing:

• Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

Disabling of event associated with event ID.

(a) Parameters:

in	*nId*	Event ID

(b) Return values:

0	normal end
-1	event removal failure

(5) int_t TouchPanel_EventLock (const int_t *nId*)

Locks processing of the touch panel event specified by the event ID.

Calls the function described in TP_EventLock, to perform the following processing:

• Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

Setting the event specified by the event ID to the locked state in the touch panel event table.

(a) Parameters:

in	*nId*	Event ID

(b) Return values:

0	normal end
-1	event removal failure

(6) int_t TouchPanel_EventLockAll (void)

Locks processing of all touch panel events.

Calls the function described in TP_EventLockAll, to perform the following processing:

Setting all events in the touch panel event table to the locked state

.

(a) Return values:

0	normal end
-1	touch panel event locking failure

(7) int_t TouchPanel_EventUnlock (const int_t *nId*)

Unlocks processing of the touch panel event specified by the event ID.

Calls the function described in TP_EventUnlock, to perform the following processing:

• Event ID checking (within range of 0 to TPEVT_ENTRY_MAX)

Setting the event specified by the event ID to the unlocked state in the touch panel event table.

(a) **Parameters:**

in	*nId*	Event ID

(b) **Return values:**

0	normal end
-1	event removal failure

(8) int_t TouchPanel_EventUnlockAll (void)

Unlocks processing of all touch panel events.

Calls the function described in TP_EventUnlockAll, to perform the following processing:

Setting all events in the touch panel event table to the unlocked state.

(a) **Return values:**

0	normal end
-1	touch panel event unlocking failure

(9) void TouchPanel_Init (void)

Initializes the touch panel driver by calling the TP_Init.

(a) **Return values:**

NONE	

(10) int_t TouchPanel_Open (const int_t *nWidth*, const int_t *nHeight*, const uint32_t *unIrqLv*, const int16_t *nTskPri*, const uint32_t *unTskStk*)

Generates and initializes a touch panel task by calling the TP_Open.

Do not call this function during touch panel utility has been opened.

(a) **Parameters:**

in	*nWidth*	LCD width
in	*nHeight*	LCD height
in	*unIrqLv*	IRQ interrupt priority (0 to 255), sets the GIC interrupt priority
in	*nTskPri*	Task priority, sets the values of the osPriority type
in	*unTskStk*	unTskStk, not used.

(b) **Return values:**

NONE	

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Nov. 29, 2019	-	First Edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.