

1、添加 TP 驱动程序

- a. TP 驱动添加目录: components/bk_peripheral/src/tp/tp_cst816d.c

2、I2C 引脚硬件适配

- a. 默认使用软件模拟 I2C, 对应代码 bk_idk/middleware/driver/i2c/sim_i2c_driver.c

3、TP 挂载

- a. 添加新增 TP 类型枚举

```
diff --git a/bk_idk/include/driver/tp_types.h b/bk_idk/include/driver/tp_types.h
index fd8f0479..a71f775b 100755
--- a/bk_idk/include/driver/tp_types.h
+++ b/bk_idk/include/driver/tp_types.h
@@ -34,6 +34,7 @@ typedef enum
     TP_ID_GT1151,
     TP_ID_FT6336,
     TP_ID_HY4633,
+    TP_ID_CST816D,
 } tp_sensor_id_t;
```

- b. 添加新增 TP 驱动索引

```
--- a/components/bk_peripheral/include/tp_sensor_devices.h
+++ b/components/bk_peripheral/include/tp_sensor_devices.h
@@ -34,6 +34,9 @@ extern "C" {
     #if CONFIG_TP_HY4633
         extern const tp_sensor_config_t tp_sensor_hy4633;
     #endif
+    #if CONFIG_TP_CST816D
+        extern const tp_sensor_config_t tp_sensor_cst816d;
+    #endif
 void tp_sensor_devices_init(void);
```

- c. 添加新增 TP 到 tp list

```
--- a/components/bk_peripheral/src/tp/tp_sensor_devices.c
+++ b/components/bk_peripheral/src/tp/tp_sensor_devices.c
@@ -22,6 +22,9 @@
     const tp_sensor_config_t *tp_sensor_configs[] =
     {
+    #if CONFIG_TP_CST816D
+        &tp_sensor_cst816d,
+    #endif
     #if CONFIG_TP_GT911
         &tp_sensor_gt911,
     #endif
@@ -41,6 +44,7 @@ const tp_sensor_config_t *tp_sensor_configs[] =
```

- d. 添加新增 TP Kconfig 项

```

--- a/components/bk_peripheral/src/tp/Kconfig
+++ b/components/bk_peripheral/src/tp/Kconfig
@@ -18,4 +18,9 @@ menu "TP"
    depends on TP
    bool "Enable TP_HY4633 API"
    default n

+
+
+   config TP_CST816D
+       depends on TP
+       bool "Enable TP_CST816D API"
+       default n
+
endmenu

```

- 4、在使用的 project 的 cpu1 的 config 文件中，需要将 3.d 中添加的 CONFIG_TP_CST816D 宏置 y
- 5、新增 TP 的调试过程：
 - a. 先确认好硬件引脚适配正确
 - b. 调试 I2C 通信，一般能正确读取到 TP 的 chip_id 即可，可参照 cst816d_detect 函数处理
 - c. TP 坐标的读取通过 tp_process_task 来定时读取的，可以打开下图的 log 检测 tp 数据读取情况

```

430: void tp_process_task(beken_thread_arg_t arg)
431: {
432:     int ret;
433:     tp_data_t tp_data[TP_SUPPORT_MAX_NUM];
434:
435:     while (1)
436:     {
437:         ret = rtos_get_semaphore(&tp_sema, BEKEN_NEVER_TIMEOUT);
438:         if(kNoErr != ret)
439:         {
440:             LOGE("%s, get semaphore fail!\n", __func__);
441:         }
442:
443:         os_memset(tp_data, 0x00, sizeof(tp_data));
444:         if (NULL != current_sensor->read_tp_info)
445:         {
446:             if (BK_OK != current_sensor->read_tp_info(&tp_i2c_cb, TP_SUPPORT_MAX_NUM, (void *)tp_data))
447:             {
448:                 LOGE("%s get tp info fail!\n", __func__);
449:             }
450:             else
451:             {
452:                 for (uint8_t i=0; i<TP_SUPPORT_MAX_NUM; i++)
453:                 {
454:                     if ((TP_EVENT_TYPE_DOWN == tp_data[i].event) || (TP_EVENT_TYPE_UP == tp_data[i].event) || (TP_EVENT_TYPE_MOVE == tp_data[i].event))
455:                     {
456:                         LOGD("event=%d, track_id=%d, x=%d, y=%d, s=%d, timestamp=%u.\n",
457:                             tp_data[i].event,
458:                             tp_data[i].track_id,
459:                             tp_data[i].x_coordinate,
460:                             tp_data[i].y_coordinate,
461:                             tp_data[i].width,
462:                             tp_data[i].timestamp);
463:                     }
464:                 }
465:             }
466:         }
467:     }
468: }

```

- d. TP 读取到的数据解析和转换处理参照 cst816d_read_point 函数，将 I2C 读取到的裸数据根据 tp 原厂提供的参考资料进行解析和转换，一般关注的 iu 是坐标值，动作类型等，最后将完整 tp 数据赋值到 tp_data_t *read_data 中即可