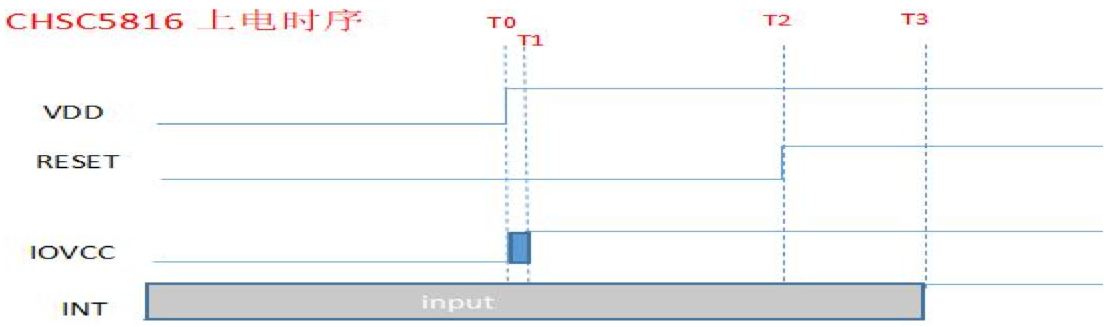
**This document applies to CHSC5816 and CHSC6417*

● Power-on timing



* CHSC5816 supports dual power supply (the motherboard provides VDD and IOVCC) and single power supply (the motherboard only provides VDD, and the IO voltage is provided by the internal 1.8V LDO of CHSC5816, or directly connected VDD)

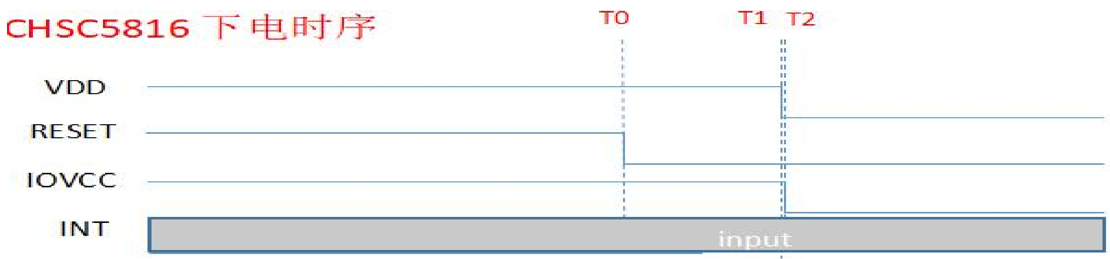
* VDD > = IOVCC

* If IOVCC of motherboard is used, it is required that T1-T0 > = 0, try to minimize the time interval as much as possible.

* T2-T0 > = 1ms, Pull RESET low before VDD is pulled up, T2-T0 is the reset time for chip hardware.

* Configure INT PIN to input mode (close pull-up) before VDD is pulled up, and set to interrupt input mode after CHSC5816 reset.

● Power-down timing

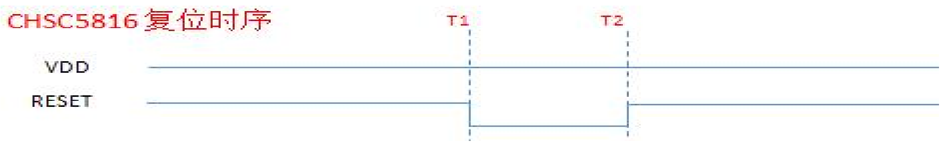


* If IOVCC of motherboard is used, it is required that T2-T1 > = 0, try to minimize the time interval as much as possible.

* Pull RESET low before VDD is turned off

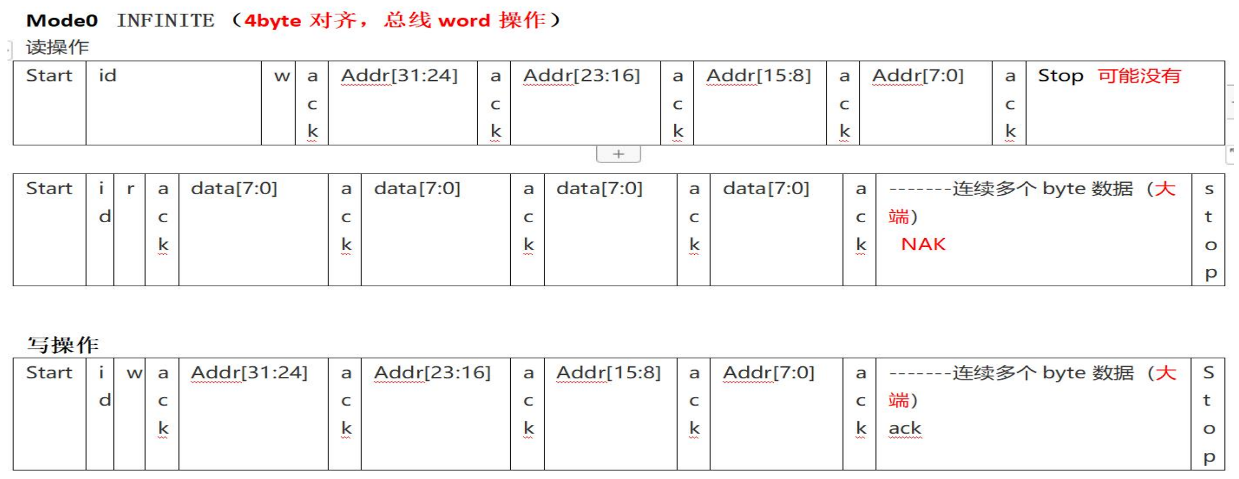
* Keep INT in input mode(turn off pull-up) and can be changed to output low after T2

● Reset timing



* T2-T1 > = 3ms, Interrupt of INT can be turned off before RESET, and re-enabled after RESET is completed.

● IIC read-write timing



Mode0 INFINITE （**4byte 对齐，总线 word 操作**）

读操作

Start	id	w	a c k	Addr[31:24]	a c k	Addr[23:16]	a c k	Addr[15:8]	a c k	Addr[7:0]	a c k	Stop 可能没有

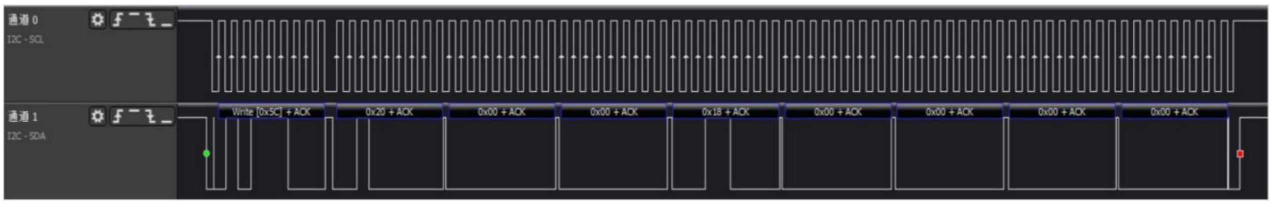
Start	i d	r	a c k	data[7:0]	a c k	data[7:0]	a c k	data[7:0]	a c k	data[7:0]	a c k	-------连续多个 byte 数据（大端） NAK	s t o p

写操作

Start	i d	w	a c k	Addr[31:24]	a c k	Addr[23:16]	a c k	Addr[15:8]	a c k	Addr[7:0]	a c k	-------连续多个 byte 数据（大端） ack	S t o p

IIC Writing:

START+0X5C+ACK +ADDR[31:24]+ACK +... +ADDR[7:0]+ACK +DATA+ACK + ... +DATA+ACK +STOP



(The above timing is about operation of writing 0x00000000 to 0x20000018)

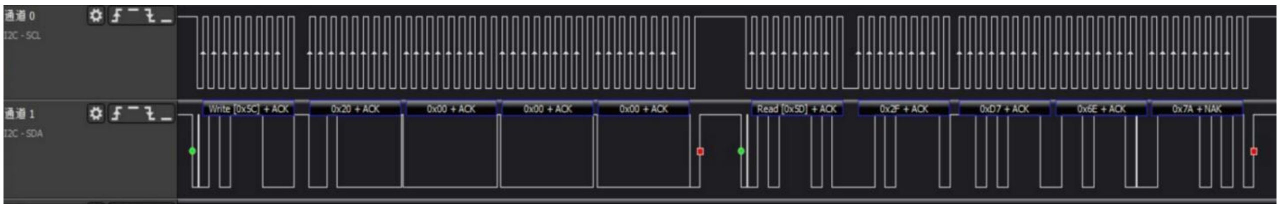
* ADDR [31: 0] is a 32-bit register address and must be 4 bytes aligned

* The length of the data written must be multiple of 4.

IIC Reading:

Step1, START+0X5C+ACK +ADDR[31:24]+ACK +... +ADDR[7:0]+ACK +STOP

Step2, START+0X5D+ACK +DATA+ACK + ... +DATA+NACK +STOP



(The above timing is about operation of reading 4 bytes from 0x00000000)

* ADDR [31: 0] is a 32-bit register address and must be 4 bytes aligned

* The length of the read data must be multiple of 4

* Avoid accessing other device on the same IIC bus between Step1 and Step2， this will make wrong data reading from CHSC5816.

● Standard commands interface

CHSC5816 software has an interface of CMD(command) and RSP(response), and all CMD and RSP are 16 bytes. Commands include: pause CDSP scanning, restart CDSP scanning, update CFG, switch power mode, change reporting rate, etc.

The target address of CMD and RSP is 0x2000_0000. The CMD and RSP are defined as:

```
typedef struct _ctp_cmd_std_t {
    U16 chk; //Checksum
    U16 D0; //Parameter 1,
    U16 D1; //Parameter 2,
    U16 D2; //Parameter 3,
    U16 D3; //Parameter 4,
    U16 D4; //Parameter 5,
    U16 D5; //Parameter 6,
    U8 ID; //Command code
    U8 tag; //Command package identification tag, fixed to 0xE9
} ctp_cmd_std_t;

typedef struct _ctp_rsp_std_t {
    U16 chk; //Checksum
    U16 D0; //Parameter 1,
    U16 D1; //Parameter 2,
    U16 D2; //Parameter 3,
    U16 D3; //Parameter 4,
    U16 D4; //Parameter 5,
    U16 D5; //Parameter 6,
    U8 CC; //Result
    U8 id; //Command code
} ctp_rsp_std_t;
```

● Physical information of touchscreen

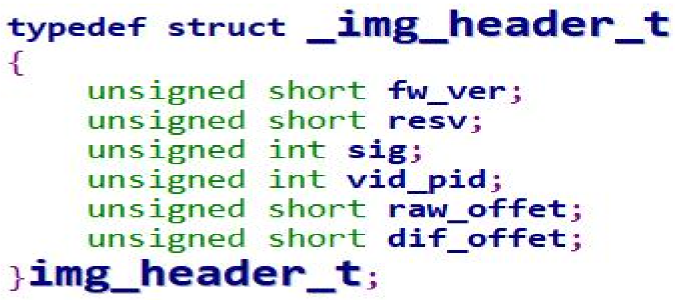
You can get the physical information of touchscreen by sending command(CMD=0x04), if this command can get right response, the response data contain some information of touchscreen, that is:

1. X resolution : ”D0” is X-axis resolution of touchscreen
2. Y resolution : ”D1” is X-axis resolution of touchscreen
3. Channel number: low byte of ”D2” is RX number, high byte of ”D2” is TX number

● Project information

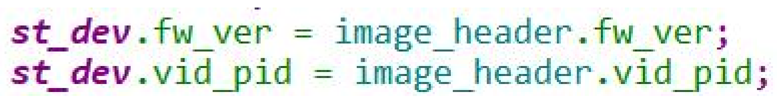
Little-Endian				
Addr.	byte 0	byte 1	byte 2	byte 3
0x20000014	fw-version		reserved	
0x20000018	firmware startup status			
0x2000001C	fw-version	projec ID		vendor ID
0x20000020	offset address of RAW data		offset address of DIF data	

We define a data structure named “img_header_t”to describe these information:



```
typedef struct _img_header_t
{
      unsigned short fw_ver;
      unsigned short resv;
      unsigned int sig;
      unsigned int vid_pid;
      unsigned short raw_offet;
      unsigned short dif_offet;
}img_header_t;
```

During TP driver initialization, we read these information from 0x20000014, and extract them to the private data of TP driver(named “st_dev”, you can refer to “semi_touch_start_up_check” function):

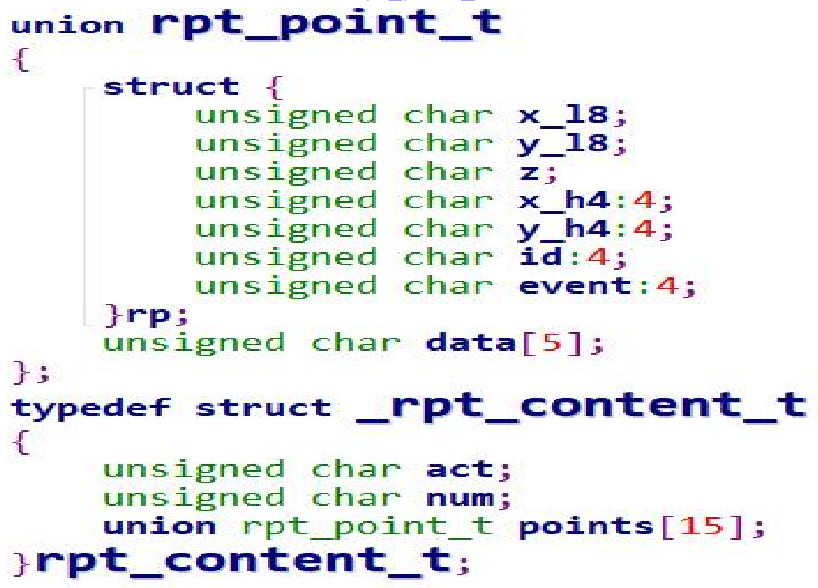


```
st_dev.fw_ver = image_header.fw_ver;
st_dev.vid_pid = image_header.vid_pid;
```

● Report data

Little-Endian								
Addr.	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
0x2000002C	EVENT type(0xFF:normal touch event, 0xFE:gesture event)							
0x2000002D	Finger number or gesture code							
0x2000002E	point 1: X coordinate - L8							
0x2000002F	point 1: Y coordinate - L8							
0x20000030	pressure value							
0x20000031	point 1: X coordinate - H4				point 1: Y coordinate - H4			
0x20000032	point 1: finger ID				point 1: touch event			
0x20000033	point 2: X coordinate - L8							
0x20000034	point 2: Y coordinate - L8							
0x20000035	pressure value							
0x20000036	point 2: X coordinate - H4				point 2: Y coordinate - H4			
0x20000037	point 2: finger ID				point 2: touch event			

We define a data structure named "*rpt_point_t*"to describe these information:



```
union rpt_point_t
{
    struct {
        unsigned char x_l8;
        unsigned char y_l8;
        unsigned char z;
        unsigned char x_h4:4;
        unsigned char y_h4:4;
        unsigned char id:4;
        unsigned char event:4;
    }rp;
    unsigned char data[5];
};
typedef struct _rpt_content_t
{
    unsigned char act;
    unsigned char num;
    union rpt_point_t points[15];
}rpt_content_t;
```

TP will notify HOST to read data by changing the INT pin level (the default setting is falling edge) when necessary. HOST reads 12 bytes from 0x2000_002c to get the reported data (CHSC5816 supports up to two-points).

❖ EVENT type

Touch EVENT:

act-> 0xFF

num-> Number of touch points

In addition, each touch is divided into three stages: pressing/touching/releasing, this information is contained in "*event*": 0 for pressing, 4 for releasing, and 8 for touching

Gesture EVENT:

act-> 0xFE,

num-> Gesture CODE, defined as:

Left slide = 0x20
Right slide = 0x21
Slide up = 0x22
Slide down= 0x23
Click = 0x24
Double-click = 0x25
Palm coverage = 0xde

ESD EVENT:
Act-> 0xFD
TP can intermittently send the EVENT, HOST received this EVENT to know that TP works properly.

● Power consumption mode

❖ Active mode

The bright screen state is the stage when the user is most likely to operate TP, which needs to respond quickly and accurately to the operation that the customer wants to perform. The working mode in this state is called active mode. Active mode can be roughly divided into two situations: touch and no touch.

Deep:
{0xf8, 0x16, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xe9}