Application Note for FT6x06 CTPM

Application Note for FT6x06 CTPM	
Project name	Touch panel
Version	0.1
Release date	Jul 26,2012
Owner	J.H. Kuo
Classification	Confidential
Approval	

R3-B4-A, South Area, Shenzhen Hi-Tech Industrial Park,
Shenzhen, Guangdong, P.R. China
ZIP: 518057
T +86 755 26588222
F +86 755 26712499
E support@focaltech-systems.com

www.focaltech-systems.com

Revision History

Date	Version	List of changes	Author + Signature
Jul 26,2012	1.0	Initial draft.	J.H. Kuo

Table of Contents

Terminology

CTP – Capacitive touch panel

CTPM – Capacitive touch panel module

1. CTPM interface to Host

Figure 1-1 shows how CTPM communicates with host device. I^2C interface supported by FT6x06 that is two-wire serial bus consisting of data line SDA and SCL clock line, used for serial data transferring between host and slave device.

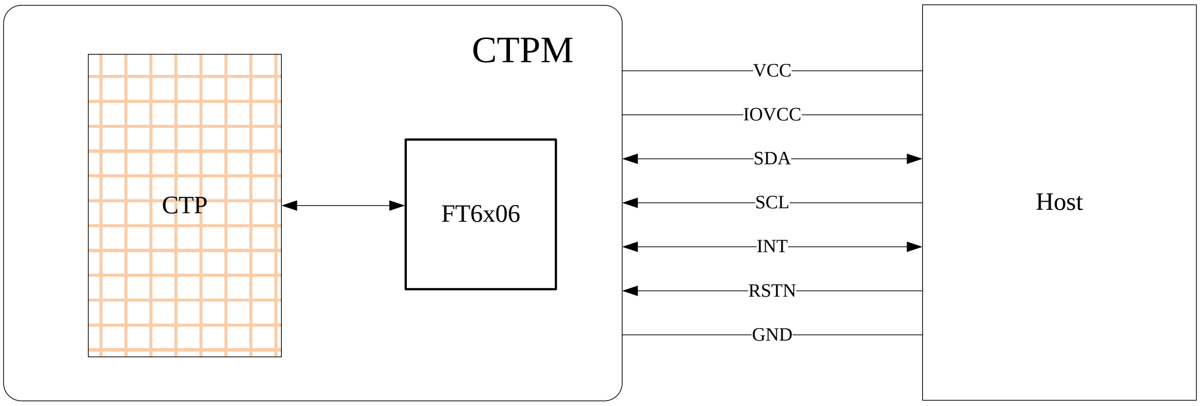

Figure 1-1 CTPM and Host connection

INT port and RSTN port form the control interface. The INT port controlled by FT6x06 will send out an interrupt request signal to the host when there is a valid touch on CTP. The INT port also has another input function that host can wake up FT6x06 from the Hibernate mode. Host can send the reset signal to CTPM via RSTN port to reset the FT6x06 if needed. The Power Supply voltage of CTPM ranges from 2.8V to 3.6V, and the interface supply voltage named IOVCC ranges from 1.8V to 3.6V. For details, please refer to Table 1-1.

Table 1-1 Description for CTPM and Host interface

Port Name	Description
VCC	CTPM power supply, ranges from 2.8V to 3.6V.
IOVCC	CTPM interface power supply for GPIO, ranges from 1.8V to 3.6V. If GPIO supply voltage is equal to VCC (2.8V~3.6V), IOVCC pin can be connected to VCC. If GPIO supply voltage is 1.8V, IOVCC pin can be connected to VDDD pin or external 1.8V power supply.
SDA	I^2C data input and output.
SCL	I^2C clock input.
INT	The interrupt request signal from CTPM to Host. The wake up signal from host to CTPM, active low and the low pulse width ranges from 0.5ms to 1ms.
RSTN	The reset signal from host to CTPM, active low, and the low pulse width should be more than or equal to 1ms.
GND	Power ground.

1.1 I^2C Read/Write Interface description

It is important to note that the SDA and SCL must connect with a pull-high resistor respectively before you read/write I^2C data.

Write N bytes to I²C slave

	Slave Addr			Data Address[X]		Data [X]			Data [X+N-1]		
S	A6 A5 A4 A3 A2 A1 A0	RW	A	R7 R6 R5 R4 R3 R2 R1 R0	A	D7 D6 D5 D4 D3 D2 D1 D0	A	…	D7 D6 D5 D4 D3 D2 D1 D0	A	P
START		WRITE	ACK		ACK		ACK			ACK	STOP

Set Data Address

	Slave Addr			Data Address[X]		
S	A6 A5 A4 A3 A2 A1 A0	RW	A	R7 R6 R5 R4 R3 R2 R1 R0	A	P
START		WRITE	ACK		ACK	STOP

Read X bytes from I²C Slave

	Slave Addr			Data [N]			Data [X+N-1]		
S	A6 A5 A4 A3 A2 A1 A0	RW	A	D7 D6 D5 D4 D3 D2 D1 D0	A	…	D7 D6 D5 D4 D3 D2 D1 D0	A	P
START		READ	ACK		ACK			ACK	STOP

1.2 Interrupt/Wake-up signal from CTPM to Host

As for standard CTPM, host needs to use both interrupt signal and I²C interface to get the touch data. CTPM will output an interrupt request signal to the host when there is a valid touch. Then host can get the touch data via I²C interface. If there is no valid touch detected, the INT will output high level, and the host does not need to read the touch data. There are two kinds of method to use interrupt: interrupt trigger and interrupt polling.

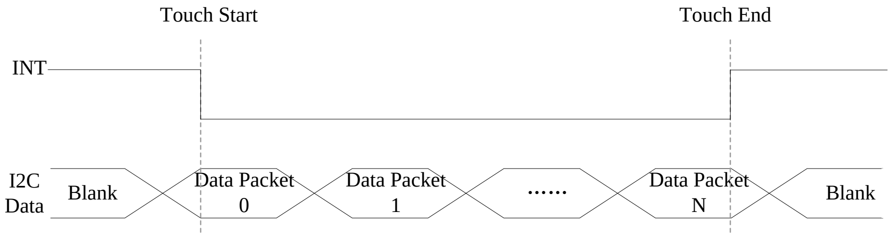

Figure 1-2 Interrupt polling mode

As for interrupt polling mode, INT will always be pulled to low level when there is a valid touch point, and be high level when a touch finished.

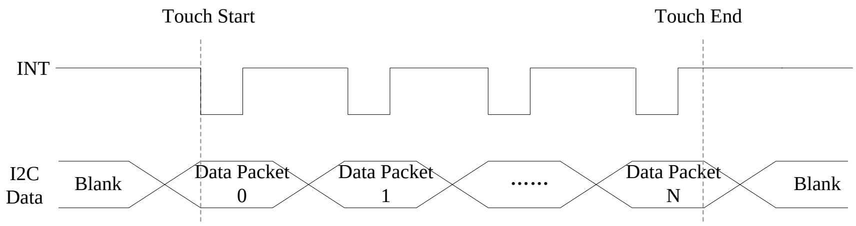

Figure 1-3 Interrupt trigger mode

While for interrupt trigger mode, INT signal will be set to low if there is a touch detected. But whenever an update of valid touch data, CTPM will produce a valid pulse on INT port for INT signal, and host can read the touch data periodically according to the frequency of this pulse. In this mode, the pulse frequency is the touch data updating rate.

When CTPM stays in hibernate mode, the INT port will act as a pull-high input port and wait for an external wake up signal. Host may send out a low pulse to wake up CTPM from the hibernate mode. The wake-up low pulse width ranges from 0.5 ms to 1 ms, the reason for this is that the INT port will act as an interrupt request signal output port after wake-up.

1.3 Reset signal from Host to CTPM

Host can send the reset signal via RSTN port to reset FT6x06. The reset signal should not be set to low while in normal running mode, but when programming flash, the RSTN port must be connected to GND. The RSTN port can also be used to active the CTPM in hibernate mode. Note that the reset pulse width should be more than 1ms.

2. Standard Application circuit of FT6x06

Table 2-1 is a brief summary of the FT6x06 application features. Figure2-1, Figure2-2, demonstrates the typical FT6x06 application schematic respectively. It consists of Capacitive Touch Panel (CTP), FT6x06 chip, and some peripheral components. According to the size of CTPM, you can choose the number of channels needed.

Table 2-1 Brief features of FT6X06

IC Type	**FT6206GMA**	**FT6306DMB**
Operating Voltage(V)	2.8 ~ 3.6	2.8 ~ 3.6
IOVCC(V)	1.8 ~ 3.6	1.8 ~ 3.6
Channel	28	36
Panel Size	2.8” ~ 4.3”	4.3” ~ 7.0”
Touch points	2	2
Interface	I^2C	I^2C
Report rate	>60Hz	>60Hz
Package (mm)	5*5 QFN40	6*6 QFN48

2.1 FT6206GMA typical application schematic for voltage of 2.8~3.6V

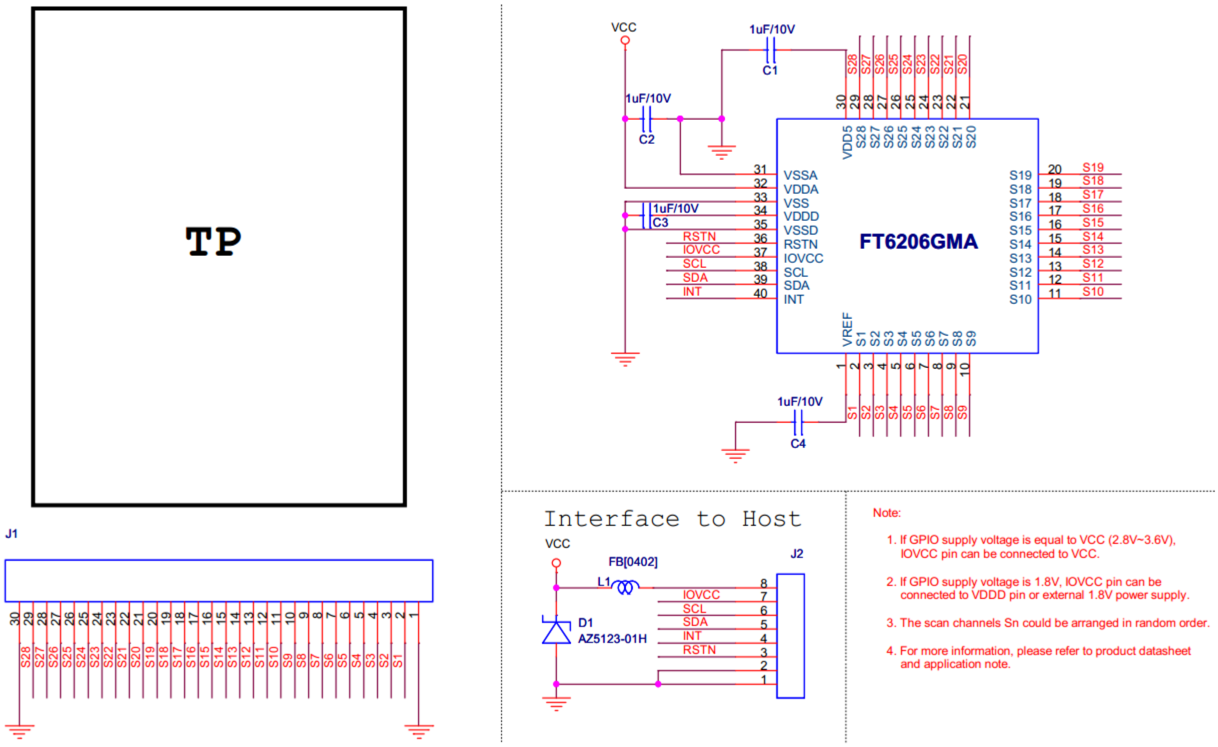

Figure 2-1 FT6206GMA typical application schematic for voltage of 2.8~3.6V

2.2 FT6306DMB typical application schematic for voltage of 2.8~3.6V

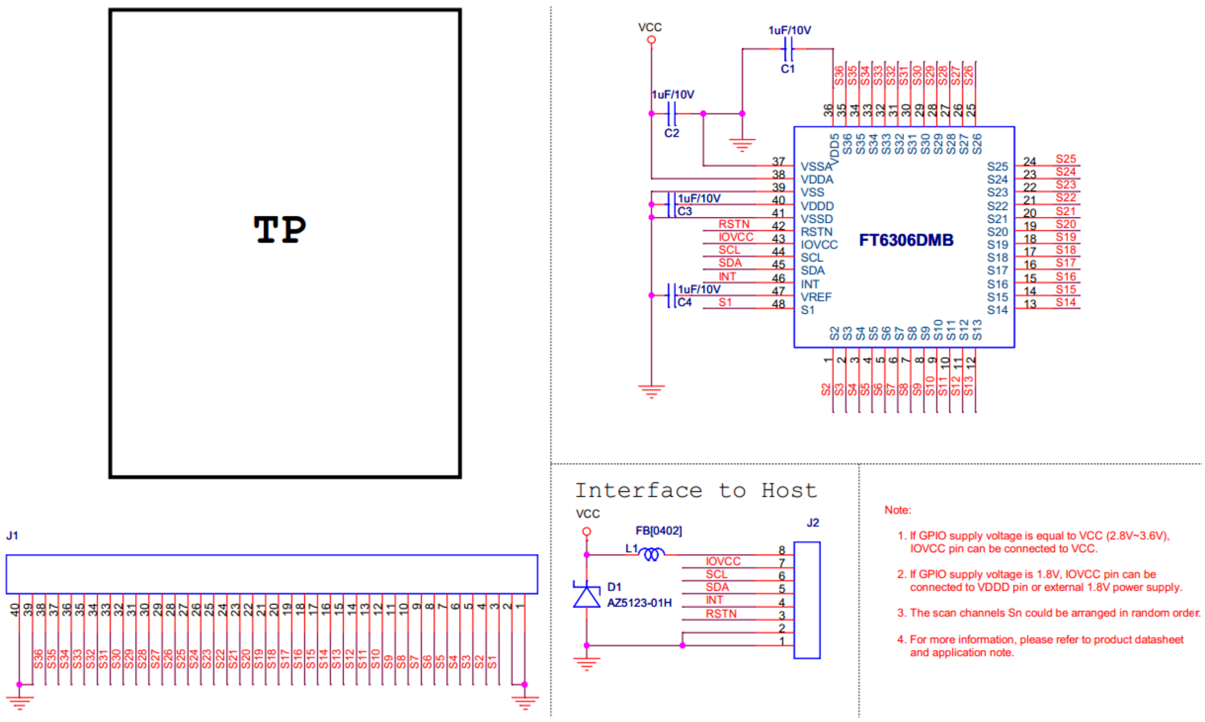

Figure 2-2 FT6306DMB typical application schematic for voltage of 2.8~3.6V

3. CTPM Register Mapping

This chapter describes the standard CTPM communication registers in address order for working mode. The most detailed descriptions of the standard products communication registers are in the register definitions section of each chapter.

3.1 Working Mode

The CTP is fully functional as a touch screen controller in working mode. The access address to read and write is just logical address which is not enforced by hardware or firmware. Here is the working mode register map.

Working Mode Register Map

Address	Name	Default Value	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Host Access
0x00	DEV_MODE	0x00		[2:0]Device Mode							R/W
0x01	GEST_ID	0x00	[7:0]Gesture ID								R
0x02	TD_STATUS	0x00					[3:0] Number of touch points				R
0x03	P1_XH	0xFF	[7:6]1st Event Flag				[3:0] 1st Touch X Position[11:8]				R
0x04	P1_XL	0xFF	[7:0] 1st Touch X Position								R
0x05	P1_YH	0xFF	[7:4] 1st Touch ID				[3:0] 1st Touch Y Position[11:8]				R
0x06	P1_YL	0xFF	[7:0] 1st Touch Y Position								R
0x07	P1_WEIGHT	0xFF	[7:0] 1st Touch Weight								R
0x08	P1_MISC	0xFF	[7:4] 1st Touch Area								R
0x09	P2_XH	0xFF	[7:6]2nd Event Flag				[3:0]2nd Touch X Position[11:8]				R
0x0A	P2_XL	0xFF	[7:0] 2nd Touch X Position								R
0x0B	P2_YH	0xFF	[7:4] 2nd Touch ID				[3:0] 2nd Touch Y Position[11:8]				R
0x0C	P2_YL	0xFF	[7:0] 2nd Touch Y Position								R
0x0D	P2_WEIGHT	0xFF	[7:0] 2nd Touch Weight								R
0x0E	P2_MISC	0xFF	[7:4] 2nd Touch Area								R
…											
0x80	TH_GROUP		[7:0] Threshold for touch detection								R/W
…											
0x85	TH_DIFF		Filter function coefficient[7:0]								R/W
0x86	CTRL	0x01	[7:0] 0: Will keep the Active mode when there is no touching. 1: Switching from Active mode to Monitor mode automatically when there is no touching.								R/W
0x87	TIMEENTERMONITOR	0x0A	[7:0] The time period of switching from Active mode to Monitor mode when there is no touching.								R/W
0x88	PERIODACTIVE		[7:0] Report rate in Active mode.								R/W
0x89	PERIODMONITOR	0x28	[7:0] Report rate in Monitor mode.								R/W
…											
0x91	RADIAN_VALUE	0x0A	[7:0] The value of the minimum allowed angle while Rotating gesture mode								R/W
0x92	OFFSET_LEFT_RIGHT	0x19	[7:0] Maximum offset while Moving Left and Moving Right gesture								R/W

0x93	OFFSET_UP_DOWN	0x19	[7:0] Maximum offset while Moving Up and Moving Down gesture	R/W
0x94	DISTANCE_LEFT_RIGHT	0x19	[7:0] Minimum distance while Moving Left and Moving Right gesture	R/W
0x95	DISTANCE_UP_DOWN	0x19	[7:0] Minimum distance while Moving Up and Moving Down gesture	R/W
0x96	DISTANCE_ZOOM	0x32	[7:0] Maximum distance while Zoom In and Zoom Out gesture	R/W
…				
0xA1	LIB_VER_H		[7:0] High 8-bit of LIB Version info	R
0xA2	LIB_VER_L		[7:0] Low 8-bit of LIB Version info	R
0xA3	CIPHER	0x06	[7:0] Chip Selecting	R
0xA4	G_MODE	0x01	[7:0] 0x00: Interrupt Polling mode 0x01: Interrupt Trigger mode	R/W
0xA5	PWR_MODE	0x00	[7:0] Current power mode which system is in	R/W
0xA6	FIRMID		[7:0] Firmware Version	R
0xA8	FOCALTECH_ID	0x11	[7:0] FocalTech's Panel ID	R
…				
0xAF	RELEASE_CODE_ID	0x01	[7:0] Release code version	R
…				
0xBC	STATE	0x01	[7:0] Current Operating mode	R/W

3.1.1 DEVICE_MODE

This is the device mode register, which is configured to determine the current mode of the chip.

Address	Bit Address	Register Name	Description
0x00	6:4	[2:0]Device Mode	000b WORKING Mode 100b FACTORY Mode

3.1.2 GEST_ID

This register describes the gesture of a valid touch.

Address	Bit Address	Register Name	Description
0x01	7:0	Gesture ID[7:0]	Gesture ID 0x10 Move Up 0x14 Move Right 0x18 Move Down 0x1C Move Left 0x48 Zoom In 0x49 Zoom Out 0x00 No Gesture

3.1.3 TD_STATUS

This register is the Touch Data status register.

Address	Bit Address	Register Name	Description
0x02	3:0	Number of touch points [3:0]	The detected point number, 1-2 is valid.
	7:4	Reserved	

3.1.4 Pn_XH (n:1-2)

This register describes MSB of the X coordinate of the nth touch point and the corresponding event flag.

Address	Bit Address	Register Name	Description
0x03 ~	7:6	Event Flag	00b: Press Down 01b: Lift Up

0x09			10b: Contact 11b: No event
	5:4		Reserved
	3:0	Touch X Position [11:8]	MSB of Touch X Position in pixels

3.1.5 Pn_XL (n:1-2)

This register describes LSB of the X coordinate of the nth touch point.

Address	Bit Address	Register Name	Description
0x04 ~ 0x0A	7:0	Touch X Position [7:0]	LSB of the Touch X Position in pixels

3.1.6 Pn_YH (n:1-2)

This register describes MSB of the Y coordinate of the nth touch point and corresponding touch ID.

Address	Bit Address	Register Name	Description
0x05 ~ 0x0B	7:4	Touch ID[3:0]	Touch ID of Touch Point, this value is 0x0F when the ID is invalid
	3:0	Touch Y Position [11:8]	MSB of Touch Y Position in pixels

3.1.7 Pn_YL (n:1-2)

This register describes LSB of the Y coordinate of the nth touch point.

Address	Bit Address	Register Name	Description
0x06 ~ 0x0C	7:0	Touch Y Position [7:0]	LSB of the Touch Y Position in pixels

3.1.8 Pn_WEIGHT (n:1-2)

This register describes weight of the nth touch point.

Address	Bit Address	Register Name	Description
0x07 ~ 0x0D	7:0	Touch Weight[7:0]	Touch pressure value

3.1.9 Pn_MISC (n:1-2)

This register describes the miscellaneous information of the nth touch point.

Address	Bit Address	Register Name	Description
0x08 ~ 0x0E	7:4	Touch Area[3:0]	Touch area value

4. Communication between host and CTPM

4.1 Communication Contents

The data Host received from the CTPM through I^2C interface are different depend on the configuration in Device Mode Register of the CTPM. Please refer to Section 3---CTPM Register Mapping.

4.2 I^2C Example Code

The code is only for reference, if you want to learn more, please contact our FAE staff.

```
//////////////////////////////////////////////////////////////////
// I2C write bytes to device.
// Arguments: ucSlaveAdr - slave address
//                        ucSubAdr - sub address
```

```
//                   pBuf - pointer of buffer
//                   ucBufLen - length of buffer
/////////////////////////////////////////////////////////////////
void i2cBurstWriteBytes(BYTE ucSlaveAdr, BYTE ucSubAdr, BYTE *pBuf, BYTE ucBufLen)
{
    BYTE ucDummy; // loop dummy
    ucDummy = I2C_ACCESS_DUMMY_TIME;
    while(ucDummy--)
    {
        if (i2c_AccessStart(ucSlaveAdr, I2C_WRITE) == FALSE)
            continue;
        if (i2c_SendByte(ucSubAdr) == I2C_NON_ACKNOWLEDGE) // check non-acknowledge
            continue;
        while(ucBufLen--) // loop of writting data
        {
            i2c_SendByte(*pBuf); // send byte
            pBuf++; // next byte pointer
        } // while
        break;
    } // while
    i2c_Stop();
}

/////////////////////////////////////////////////////////////////
// I2C read bytes from device.
//
// Arguments: ucSlaveAdr - slave address
//                   ucSubAdr - sub address
//                   pBuf - pointer of buffer
//                   ucBufLen - length of buffer
/////////////////////////////////////////////////////////////////
void i2cBurstReadBytes(BYTE ucSlaveAdr, BYTE ucSubAdr, BYTE *pBuf, BYTE ucBufLen)
{
    BYTE ucDummy; // loop dummy

    ucDummy = I2C_ACCESS_DUMMY_TIME;
    while(ucDummy--)
    {
        if (i2c_AccessStart(ucSlaveAdr, I2C_WRITE) == FALSE)
            continue;
        if (i2c_SendByte(ucSubAdr) == I2C_NON_ACKNOWLEDGE) // check non-acknowledge
            continue;
        if (i2c_AccessStart(ucSlaveAdr, I2C_READ) == FALSE)
            continue;
```

```
        while(ucBufLen--) // loop to burst read
        {
            *pBuf = i2c_ReceiveByte(ucBufLen); // receive byte
            pBuf++; // next byte pointer
        } // while
        break;
    } // while
    i2c_Stop();
}

//////////////////////////////////////////////////////////////////////
// I2C read current bytes from device.
//
// Arguments: ucSlaveAdr - slave address
//            pBuf - pointer of buffer
//            ucBufLen - length of buffer
//////////////////////////////////////////////////////////////////////
void i2cBurstCurrentBytes(BYTE ucSlaveAdr, BYTE *pBuf, BYTE ucBufLen)
{
    BYTE ucDummy; // loop dummy

    ucDummy = I2C_ACCESS_DUMMY_TIME;
    while(ucDummy--)
    {
        if (i2c_AccessStart(ucSlaveAdr, I2C_READ) == FALSE)
            continue;
        while(ucBufLen--) // loop to burst read
        {
            *pBuf = i2c_ReceiveByte(ucBufLen); // receive byte
            pBuf++; // next byte pointer
        } // while
        break;
    } // while
    i2c_Stop();
}
```